Suggestions from the software engineering practice for applying consistency-based diagnosis to configuration knowledge bases

Gerhard Fleischanderl

Siemens AG Österreich, Program and System Engineering, CES Design Services
Erdberger Laende 26, A-1030 Vienna, Austria
gerhard.fleischanderl@siemens.com

Abstract

A configuration knowledge base is software that needs debugging during maintenance and can benefit from consistency-based diagnosis. The paper describes suggestions and practical experience from the introduction of this diagnosis technique in the work flow for maintaining configuration knowledge bases. Consistency-based diagnosis is suitable for detecting bugs in knowledge bases, but needs tailoring to fit in the work flow of the knowledge engineers.

1 Introduction

Configurators have already been applied to different industry domains. For instance, telecommunication systems are among the products successfully handled with configurators. The crucial information is in the knowledge bases of the configurators.

Configurators using declarative constraints [Mittal and Frayman, 1989] are in everyday use and can generate and modify configurations with more than 50,000 objects [Fleischanderl *et al.*, 1998]. Declarative constraints offer easier maintenance compared to procedural specifications, but also benefit from effective debugging methods. Consistency-based diagnosis [Reiter, 1987] [Greiner *et al.*, 1988] is applicable to fault detection in configuration knowledge bases [Felfernig *et al.*, 2000], which is the topic of this paper. The extensions towards hierarchical models [Felfernig *et al.*, 2001] are not discussed here because the author did not apply this yet.

This paper discusses suggestions and practical experience from applying diagnosis techniques to the debugging of declarative knowledge bases for configurators. The experience ranges from the planning of an engineering process including diagnosis to the early adoption of diagnosis for the debugging of knowledge bases. The requirements of the development process for knowledge bases are compared with the specification of the diagnosis method.

2 Maintaining knowledge bases

Creating and maintaining knowledge bases is essentially a software engineering process.

After collecting and analyzing new requirements, the knowledge base is modified and tested. Regression tests are essential for long-term maintenance. So the results from replaying regression tests should be fed into a diagnosis tool if the new output differs from the expected output of a regression test.

In an ideal world the discrepancies from regression tests would be analyzed with a diagnosis tool and suggestions be made which constraints in the knowledge base are responsible for the discrepancies. Unfortunately this is not that easy.

3 Preconditions for consistency-based diagnosis

Consistency-based diagnosis needs a consistency checker, i.e. a solver that yields conflict sets when a knowledge base is in contradiction to a positive example. The configurator kernel COCOS [Stumptner *et al.*, 1998] applied by the author is a solver that uses declarative constraints for statically checking or expanding a partial configuration. The kernel was extended to also yield conflict sets. So a sufficiently powerful consistency checker is available.

The elements that can be faulty have to be identifiable parts of a knowledge base. In our case the constraints can be faulty with respect to positive examples and are the "components" for model-based diagnosis.

4 Requirements and consequences of consistency-based diagnosis

4.1 Definition of a CKB-diagnosis

A CKB-diagnosis (i.e. diagnosis of configuration knowledge bases) uses the model-based diagnosis paradigm and is defined as follows [Felfernig *et al.*, 2000].

Definition (CKB-Diagnosis Problem): A CKB-Diagnosis Problem is a triple (DD,E+,E-) where DD is a configuration knowledge base, E+ is a set of positive and E- of negative configuration examples. The examples are given as sets of logical sentences. It is assumed that each example on its own does not contain inconsistencies.

Definition: A CKB-diagnosis for a CKB-Diagnosis Problem (DD,E+,E-) is a set $S \subseteq DD$ of sentences such that there exists an extension EX, where EX is a set of logical sentences, such that

$DD - S \cup EX \cup e+$ consistent $\forall e+ \in E+$

$DD - S \cup EX \cup e-$ inconsistent $\forall e- \in E-$

Let NE be the conjunction of all negated negative examples. This is the most easily found EX.

Proposition: Given a CKB-Diagnosis Problem (DD,E+,E-), a diagnosis S for (DD,E+,E-) exists iff

$\forall e+ \in E+ : e+ \cup NE$ is consistent.

Corollary: S is a diagnosis iff

$\forall e+ \in E+ : DD - S \cup e+ \cup NE$ is consistent.

4.2 Representation of examples

The definition of a CKB-Diagnosis Problem says that the examples are given as sets of logical sentences. This is usually not the case in configurator implementations. Yet, databases or other data representations can easily be transformed into facts, i.e. logical sentences. This transformation need not be done for the implementation of diagnosis for configurator knowledge bases, but is a precondition for the applicability of CKB-diagnosis.

With logical sentences one can define a configuration as a set of fragments. In configurator applications, configurations are based on an object model, which is usually defined with UML. All objects usually are reachable from one entry object. So the positive or negative examples cannot just be isolated sub-configurations, but must be connected objects. This is a slight restriction that does not limit the diagnosis.

This property of configurations ensures that trivial inconsistencies are avoided, e.g. there cannot be two modules in the same slot. Therefore each example (i.e. its structure of objects and connections) does not contain inconsistencies among its elements.

4.3 Conjunction of negated negative examples

The definition of a CKB-diagnosis requires an extension EX. The question is: Where does EX come from?

The simplest EX would be the negation of all negative examples, i.e. NE as defined above. This is not a useful solution for maintaining configurator knowledge bases in real life. This would reduce the advantages of declarative constraints, namely that knowledge bases contain little redundant information and can be understood easily by domain experts. Furthermore, the constraints should be sufficiently general to be applicable to similar situations in the future. The negation of configurations (i.e. negative examples) would clutter the knowledge base with facts that may overlap and would not prevent examples that are slightly different.

4.4 Diagnosis is part of the existing knowledge base

According to the definition of CKB-diagnosis, a diagnosis is a subset of the knowledge base. That means faults are found among the constraints in the existing knowledge base. This is useful in real-life projects and makes the consistency-based diagnosis worthwhile. Yet, defining new constraints (thus extending the knowledge base) has to be accomplished with other approaches.

5 Integrating consistency-based diagnosis in the software engineering process

The definitions for consistency-based diagnosis of configuration knowledge bases do not tell a lot about how to proceed (step by step) to reach a correct knowledge base. However, the conditions for the correctness check for knowledge bases are specified.

This section describes how to use diagnosis in the software engineering process for knowledge bases.

5.1 Use the examples one by one

Examples, i.e. stored configurations, may be partial or complete. Due to restrictions coming from the usual object models in software development, each example is a network of objects that can be reached from an entry object. Therefore, only one example can be loaded at one time. This holds for positive and negative examples.

5.2 Negative examples are outsiders

In the diagnosis process discussed here, negative examples do not yield hints for mistakes in a knowledge base.

We expect that negative examples lead to inconsistencies. If a negative example is consistent with the knowledge base, the consistency-based diagnosis has no discrepancy to start from. The practical suggestion then is to analyze the consistent negative examples "by hand" and modify the knowledge base to rule out those examples. This corresponds to finding the mysterious EX in the definition of CKB-diagnosis.

The good news, however, is that negative examples usually are modifications of positive examples or previously positive examples that became negative after a modification to the knowledge base. Our experience from maintenance over many years shows that these negative examples will mostly remain negative examples after more modifications to the knowledge base.

Help also comes from good practice in software engineering. When knowledge bases are stored in a version control (configuration management) system, we can find the latest previous version where some negative example was still rejected by the knowledge base. Comparing that older version with the current knowledge base shows the constraints that were modified or removed in the meantime. This is of course an excellent starting point for modifying the current knowledge base such that it again rejects the negative example.

When all negative examples are rejected by the knowledge base, start looking at the positive examples. So the negative examples are treated outside the diagnosis step.

5.3 Use the results from regression tests

Like any software, knowledge bases can be maintained more efficiently by using regression tests and checking them after a modification.

When a regression test produces an output different from its reference, find out whether the new output is expected (after a modification to the knowledge base). Only if the new output is different from what is expected, feed this output into diagnosis.

5.4 Do diagnosis and repeat the cycle

Finally, we use consistency-based diagnosis to detect faults in the knowledge base. This follows the definition of CKB-diagnosis as described above. The well-defined preconditions and semantics of the method make it particularly valuable.

After the knowledge base was modified, we must repeat the cycle of testing and diagnosis until all negative examples are inconsistent and all positive ones are consistent.

The cycle described here starts with the negative examples (by modifying or extending the knowledge base) and continues with the positive examples (by modifying or reducing the knowledge base). This could be done the other way round. The "optimal" sequence, however, depends on the structure of the knowledge base and the expert's experience and point of view. The objective is to modify the knowledge base such that it remains easy to maintain and easy to understand. We are confident that the steps described above help us get close to this objective.

6 Beyond diagnosis

Beyond the scope of CKB-diagnosis, other methods can be useful for maintaining knowledge bases.

Automatic generation of test cases would be helpful for producing a large set of regression test cases. This would assure the quality of knowledge bases that are maintained over several years.

If a negative example is consistent, automatic generalization of the negated negative example could yield a non-redundant modification to the knowledge base. Here the optimum between introducing too many new constraints and over-generalization has to be found. For this purpose the methods for automatic learning of concepts have to be analyzed with respect to the semantics of the configuration knowledge base.

7 Summary and conclusion

Consistency-based diagnosis is applicable to the debugging of configuration knowledge bases. The method is particularly valuable because of its well-defined preconditions and semantics.

Integrating CKB-diagnosis in the software engineering process for knowledge bases can be done efficiently and effectively. There are minor limitations where CKB-diagnosis cannot be fully applied, i.e. with respect to automatic suggestions from negative examples. Altogether the experience from the planning of a debugging process with diagnosis and from the early adoption is encouraging. Results from wide usage will follow.

Acknowledgement

I want to thank Gerhard Friedrich and Dietmar Jannach for their valuable contributions to our discussions.

References

[Felfernig *et al.*, 2000] Alexander Felfernig, Gerhard E. Friedrich, Dietmar Jannach, and Markus Stumptner. Consistency-based Diagnosis of Configuration Knowledge Bases. *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000),* pp. 146-150, Berlin, Aug. 2000, IOS Press.

[Felfernig *et al.*, 2001] Alexander Felfernig, Gerhard E. Friedrich, Dietmar Jannach, and Markus Stumptner. Hierarchical diagnosis of large configurator knowledge bases. *Working Notes of the 12th Intl. Workshop on Principles of Diagnosis (DX-2001),* Via Lattea, Italy, March 2001.

[Fleischanderl *et al.*, 1998] Gerhard Fleischanderl, Gerhard E. Friedrich, Alois Haselböck, Herwig Schreiner, and Markus Stumptner. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems & their applications,* 13(4):59-68, July/Aug. 1998.

[Greiner *et al.*, 1988] 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, Nov. 1989.

[Mittal and Frayman, 1989] Sanjay Mittal and Felix Frayman. Towards a generic model of configuration tasks. *Proceedings of the 11th Intl. Joint Conference on Artificial Intelligence (IJCAI-1989),* pp. 1395-1401, Detroit, Aug. 1989, Morgan Kaufman Publishers.

[Reiter, 1987] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence,* 32(1):57-95, Apr. 1987.

[Stumptner *et al.*, 1998] Markus Stumptner, Gerhard E. Friedrich, and Alois Haselböck. Generative Constraint-Based Configuration of Large Technical Systems. *AI-EDAM (Artificial Intelligence for Engineering, Design, Analysis and Manufacturing)*, 12(4):307-320, Special Issue on Configuration, Sep. 1998.