QuickXPlain Explanations for Feature Model Configuration

Alexander Felfernig[†], Damian Garber[†], Viet-Man Le[†] and Sebastian Lubos[†]

Institute of Software Engineering and AI, Graz University of Technology, Graz, Austria

Abstract

Explanations play an important role in the context of feature model (FM) configuration. First, they can assure the interpretability of the calculated solutions (configurations) as a result of a feature model configuration process. Beyond this, explanations can support engineers (developers) of feature models in the identification of issues in the model, i.e., to figure out as to why on a semantic level the feature model does not fully represent the existing product (service) domain knowledge. In this paper, we discuss different basic explanation scenarios in the context of feature model development and feature model configuration. We show how these explanations can be supported on the basis of the concepts of conflict detection and model-based diagnosis.

Keywords

QuickXPlain, Explanation, Feature Model Configuration, Conflict Detection, Model-based Diagnosis

1. Introduction

Feature models (FMs) can be used for the representation of commonality and variability properties of highly-variant artifacts such as physical products and software [1, 2, 3, 4, 5, 6]. Formal representations of feature models such as SAT problems [7] and constraint satisfaction problems (CSPs) [8, 9] are often used to find a solution for a given feature model configuration task and – beyond that – for supporting different types of analysis operations used to assure the well-formedness and semantic correctness of feature models [3, 10]. Compared to the representation as SAT problem, CSPs allow for a more flexible knowledge representation, for example, in terms of a direct representation of logical equivalence properties and implications [4].

Independent of the used FM knowledge representation, it is important to provide users with explanations [4, 11, 12, 13, 14]. Such explanations can serve different purposes ranging from assuring interpretability for the user, increasing the trust level of a user, enhancing a user's domain knowledge, to persuading users to include/exclude specific features into/from an FM configuration [12, 15, 16, 17, 18, 19, 20, 21]. In this paper, we focus on the aspect of *interpretability*. We discuss different explanation scenarios in the context of feature model development and feature model configuration. For example, in the context of feature model development and maintenance, a modeler needs to know the set of features responsible for the violation of a specific property (e.g., void feature models or dead features in feature models). Furthermore, users of an FM configurator are interested as to why specific features have been included but also why other features have been excluded unexpectedly (the counterfactual case). In the following, we discuss different explanation scenarios and show how standard QuickXPlain-style conflict detection [22, 23] and diagnosis [24] algorithms can be applied to generate explanations and corresponding repairs in the case of inconsistencies.

The major contributions of our paper are the following: (1) we formalize basic explanation and repair tasks in FM development and configuration, (2) we show how corresponding explanations and repairs can be determined with existing conflict detection and diagnosis algorithms, and (3) to increase

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

[†]These authors contributed equally.

alexander.felfernig@tugraz.at (A. Felfernig); damian.garber@tugraz.at (D. Garber); v.m.le@tugraz.at (V. Le); sebastian.lubos@tugraz.at (S. Lubos)

0000-0003-0108-3146 (A. Felfernig); 0009-0005-0993-0911 (D. Garber); 0000-0001-5778-975X (V. Le); 0000-0002-5024-3786 (S. Lubos)

understandability, we provide examples of how to generate explanations and repairs. Specifically, we show how to create the mentioned explanations and repairs with the algorithms QuickXPlain [22] and FastDiag [24, 25]. With these contributions, we aim to show different ways of integrating conflict detection and diagnosis as a basis of explanation generation in FM modeling and configuration.

The remainder of this paper is organized as follows. In Section 2, we introduce an example feature model that serves as a working example throughout the paper. Thereafter, in Section 3, we introduce two basic algorithms supporting the tasks of conflict detection and diagnosis. In Section 4, we show how these algorithms can be applied to support different FM-related explanation scenarios. A discussion of threats to validity is provided in Section 5. In Section 6, the paper is concluded with an overview of research issues.

2. Example Feature Model

In the following, we present an example feature model from the domain of *survey software configuration*, which will serve as a running example throughout this paper (see Figure 1).

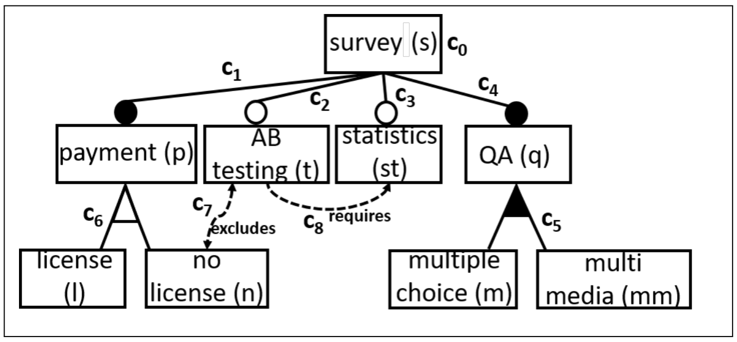

Figure 1: An example feature model (*survey software*).

The features in this model are arranged hierarchically including the following relationships: (1) *mandatory relationships* specify that certain features must be included in every configuration (e.g., the payment feature is required to be included in every configuration), (2) *optional relationships* indicate that certain features may be included, but their inclusion is optional (e.g., the statistics feature can optionally be added to a configuration), (3) *alternative relationships* specify that, within a set of sub-features, exactly one sub-feature must be chosen if the parent feature is included (e.g., one license type must be selected), and (4) *or relationships* require that *at least one* feature from a set of sub-features must be chosen if the parent feature is included (e.g., question answering (QA) can be handled with multiple-choice questions, multimedia-based representations, or both). In addition, *cross-tree constraints* can be used to impose further restrictions: (1) *excludes constraints* between two features prevent both from being included in the same configuration (e.g., if no license is selected as the payment model, ABtesting cannot be included in the same configuration), and (2) *requires constraints* between two features f_a and f_b specify that if f_a is included, f_b must also be included in the final configuration (e.g., the inclusion of the ABtesting feature necessitates the inclusion of statistics).

To enable FM configuration, feature models must be translated into a formal representation. Common approaches for this translation include SAT problems [26, 27], answer set programs (ASPs) [28], and constraint satisfaction problems (CSPs) [8, 29, 30]. In this paper, we adopt CSPs as the formal representation for feature models. For a detailed discussion of the rules governing the translation of feature models into logic-based representations, we refer to [4, 8]. The constraint-based representations we generate in this paper follow these established translation rules.

3. Conflict Detection and Model-based Diagnosis

In Table 1, we introduce a CSP-based formalization of the FM shown in Figure 1. In this formalization, c_0 is the *root constraint* (part of every feature model) that ensures that in each FM configuration at least one feature is included, i.e., no empty feature model configurations are allowed.

Table 1
CSP derived from the feature model in Figure 1. Abbreviated variable names are used, e.g., survey (s).

ID	Description
c_0	$s = true$
c_1	$p \leftrightarrow s$
c_2	$t \rightarrow s$
c_3	$st \rightarrow s$
c_4	$q \leftrightarrow s$
c_5	$q \leftrightarrow (m \vee mm)$
c_6	$p \leftrightarrow (l \wedge \neg n \vee \neg l \wedge n)$
c_7	$\neg(n \wedge t)$
c_8	$t \rightarrow st$

Based on this example CSP (Table 1), we now introduce the concept of a *feature model configuration task* (see Definition 1) and a corresponding *feature model configuration* (see Definition 2) [4].

Definition 1. An *FM configuration task* is defined as a constraint satisfaction problem (F, C), where F is a set of Boolean variables (features) f_i (with domain (f_i) = {true, false}), and $C = REQ \cup KB$ is a set of constraints. Here, $KB = \{c_0 \ldots c_n\}$ represents a set of domain constraints, and $REQ = \{c_{n+1} \cdots c_k\}$ represents a set of user requirements.

In our example, $KB = \{c_0 \ldots c_8\}$ (see Table 1) and $REQ = \{c_9 : t = true\}$, meaning that the user has specified ABtesting (t) to be included in the final configuration.

Definition 2. An *FM configuration* $CONF = \{f_1 = a(f_1), f_2 = a(f_2), \ldots, f_k = a(f_k)\}$ is a set of variable assignments, where $a(f_i)$ is the value assigned to the variable (feature) f_i. A configuration $CONF$ is considered consistent if $(\bigcup \{f_i = a(f_i)\} \in CONF) \cup REQ \cup KB$ is consistent (i.e., a solution exists). A configuration is complete if every variable in F has a corresponding assignment in $CONF$. Finally, $CONF$ is valid if it is both, consistent and complete.

Example 1. An example configuration could be $CONF = \{s = true, p = true, l = true, n = false, t = true, st = true, q = true, m = true, mm = false\}$.

However, there are often situations where a configuration is inconsistent with the constraints in the feature model or the feature model constraints are inconsistent resulting in a void feature model. Also, customer requirements can induce an inconsistency with the FM constraints resulting in a situation where no solution can be identified. To deal with such situations, the concepts of *conflict sets* (see Definition 3) and *diagnoses* (see Definition 4) are fundamental [4]. In the following, these two concepts are formulated with regard to a constraint set C which is inconsistent, i.e., no solution could be found for the constraints in C.

Definition 3. A conflict set is a set $\Gamma \subseteq C$ with inconsistent(Γ). A conflict set is minimal if $\neg\exists\Gamma' : \Gamma' \subset \Gamma$ and Γ' is a conflict set.

QuickXPlain For a minimal conflict set to be resolved, only one constraint needs to be deleted from the conflict set. The term *minimal conflict set* is often used synonymously to the term *minimal unsatisfiable subset* (MUS) [31]. Minimal conflict sets can be determined on the basis of the QuickXPlain algorithm [22] which determines one minimal conflict set (Γ) at a time. Given a set of constraints $C = \{c_1, .., c_k, c_{k+1}, .., c_n\}$ (k is assumed to be $\frac{n}{2}$), if $\{c_1, .., c_k\}$ is inconsistent, the QuickXPlain conflict set search will focus on $\{c_1, .., c_k\}$ and immediately omit $\{c_{k+1}, .., c_n\}$. In many scenarios, all minimal conflict sets need to be determined. In such a situation, QuickXPlain needs to be combined with a corresponding hitting set directed acyclic graph based approach which helps to determine all minimal conflict sets in a systematic fashion [32].

Definition 4. A diagnosis $\Delta \subseteq C$ fulfills: consistent($C - \Delta$). Δ is minimal if $\neg\exists\Delta' : \Delta' \subset \Delta$ and Δ' is a diagnosis.

FastDiag In each case, a minimal diagnosis consists of a set of constraint which – if deleted from C – assure that $C - \Delta$ is consistent. The term *minimal diagnosis* of often used synonymously to the term *minimal correction subset* (MCS). Furthermore, the complement of a MCS, i.e., $C - MCS$, is denoted as *maximal satisfiable subset* (MSS) [31]. Minimal diagnoses (Δ) can be determined with the FastDiag algorithm [24] which is similar to QuickXPlain in terms of the used divide-and-conquer based search strategy. Given a set of constraints $C = \{c_1, .., c_k, c_{k+1}, .., c_n\}$ (k is assumed to be $\frac{n}{2}$), if $\{c_1, .., c_k\}$ is *consistent*, the FastDiag diagnosis search will focus on $\{c_{k+1}, .., c_n\}$ and immediately omit $\{c_1, .., c_k\}$.

In the following, we show how QuickXPlain can be applied to generate *explanations* in FM development, maintenance, and configuration contexts. Furthermore, we show how FastDiag can be applied to generate *repairs* to recover from unintended (often inconsistent) situations. Note that both algorithms are standard algorithms in the context of conflict detection and diagnosis. For related algorithmic/implementation details we refer to [22, 24].

4. Explanations in FM Development and Configuration

We now introduce a schema of how to apply QuickXPlain [22] and FastDiag [24] for creating explanations and to propose corresponding repair actions where needed. In this context, QuickXPlain(α, β) is assumed to return a minimal conflict set Γ where α represents a set of constraints that can be used for explanation purposes and β represents the background knowledge, i.e., a set of constraints assumed to be correct. Furthermore, FastDiag(ϵ, β) is assumed to return a minimal diagnosis Δ where ϵ represents a set of constraints to be used for diagnosis purposes and β again represents the background knowledge. In our discussion, we distinguish between the phases of (1) *feature model development* (where analysis operations and corresponding explanations play a major role) and (2) *feature model configuration* where users are building their own configurations on the basis of the configuration model (knowledge base) defined by the feature model.

Feature Model Development The major focus of feature model development is to identify the set of features relevant for describing the variability properties of the underlying domain and to integrate constraints that specify relevant commonality and variability properties. In the context of FM development and maintenance, analysis operations play a major role in terms of assuring well-formedness properties of the created models [3]. Indicating the violation of a well-formedness rule defined by an analysis operation is of enormous help [33]. In this context, explanations can help to further advance the state of practice by supporting more in-depth insights into the reasons of the violation of a well-formedness rule.

While being aware of further analysis operations, we provide a selected set of operations specifically in the need of a constraint solver (configurator) support (see Table 2). Analysis operations in the need of such a reasoning support are also in the need of explanations that help to better understand the (negative) outcome of an analysis operation and to trigger repair actions [4]. Each entry in Table 2 consists of the description of the analysis operation (formulated as a corresponding question), a corresponding explanation (*why not*?), and a repair in the case that the analysis operation provides a negative answer.

Table 2
Explanations for FM analysis operations – *cons/incons* are abbreviations of *consistent/inconsistent*.

id	question (analysis operation)	explanation (why not?)	repair
1	is satisfiable(KB)?	$\Gamma \subseteq KB : incons(\Gamma)$	$\Delta \subseteq KB : cons(KB - \Delta)$
2	is life($f_i \in F, KB$)?	$\Gamma \subseteq KB : incons(\Gamma \cup \{f_i = true\})$	$\Delta \subseteq KB : cons(KB - \Delta \cup \{f_i = true\})$
3	is optional($f_i \in F, KB$)?	$\Gamma \subseteq KB : incons(\Gamma \cup \{f_i = false\})$	$\Delta \subseteq KB : cons(KB - \Delta \cup \{f_i = false\})$
4	is irredundant($c_i \in KB, KB$)?	$c_i \notin \Gamma \subset KB : incons(\Gamma \cup \overline{KB})$	$\Gamma \subset KB : incons(\Gamma, \overline{KB})$
5	is generalization(KB_g, KB_s)?	$\Gamma \subseteq KB_g : incons(KB_g \cup \overline{KB_s})$	$\Delta \subset KB_g : cons((KB_g - \Delta) \cup \overline{KB_s})$
6	is satisfiable($CONF$, KB)?	$\Gamma \subseteq CONF : incons(\Gamma \cup KB)$	$\Delta \subseteq CONF : cons(CONF - \Delta \cup KB)$

The analysis operations included in Table 2 are the following:

(1) $issatisfiable(KB)$? helps to figure out if at least one solution can be identified for the given feature model (represented by the knowledge base KB). If no solution can be identified (see also Table 3), i.e., the feature model is *void*, explanations (in terms of minimal conflict sets) Γ can be provided which represent minimal subsets of constraints which are inconsistent [4, 34]. A corresponding repair can be proposed by a diagnosis Δ that represents a set of constraints in KB that – if deleted from KB – help to assure the consistency of the remaining constraints in KB. For determining Γ and Δ, QuickXPlain(KB) and FastDiag(KB) need to be activated. In the example of Table 3, there exists only one minimal explanation (Γ_1) and four related diagnosis Δ_i, i.e., alternative repair options for restoring consistency.

Table 3
Explanation of a void FM (constraint c_π is assumed to be new) with corresponding repair options $\Delta_1..\Delta_4$.

constraint	c_0	c_1	c_4	$c_\pi : \neg(p \wedge q)$
Γ_1	×	×	×	×
Δ_1	×	–	–	–
Δ_2	–	×	–	–
Δ_3	–	–	×	–
Δ_4	–	–	–	×

(2) $is\ life(f_i \in F, KB)$? helps to figure out if at least one FM configuration can be created where the variable (feature) f_i is included [3, 4]. If no such configuration is possible (see Table 4), explanations (in terms of minimal conflict sets) Γ can be provided which represent minimal sets of constraints in KB which are inconsistent with $\{f_i = true\}$. If we want f_i to be true, we need to calculate a diagnosis Δ which represents a minimal constraint set to be deleted from KB to assure that the remaining constraints in KB allow the inclusion of f_i. In our example FM, if the feature $ABtesting(t)$ would have been defined as mandatory, the feature $nolicense(n)$ is not life. An explanation is $\{c_0, c_1, c_2, c_\pi\}$.

Table 4
Explanation of dead feature n (constraint c_π is assumed to be new) with corresponding repair options $\Delta_1..\Delta_4$.

constraint	c_0	c_1	c_2	$c_\pi : \neg(p \wedge t)$
Γ_1	×	×	×	×
Δ_1	×	–	–	–
Δ_2	–	×	–	–
Δ_3	–	–	×	–
Δ_4	–	–	–	×

(3) $is\ optional(f_i \in F, KB)$? helps to analyze if a feature can really be regarded as an optional

feature, i.e., there should be configurations where the feature is excluded [3, 4]. If the feature model represented by KB does not allow the exclusion of f_i (see Table 5), explanations (in terms of minimal conflict sets Γ) can be provided which represent minimal subsets of constraints inconsistent with the exclusion of f_i. A corresponding diagnosis Δ (which represents a resolution of all minimal conflict sets), is a minimal set of constraints that need to be deleted from KB to assure the possibility of having configurations with f_i excluded. In our example FM, if the feature $ABtesting(t)$ would have been defined as mandatory, the feature $statistics(st)$ is not optional anymore. An explanation is $\{c_0, c_2', c_8\}$.

Table 5
Explanation of false optional feature st (c_2' is an adaptation of c_2) with corresponding repair options $\Delta_1..\Delta_3$.

constraint	c_0	$c_2' : t \leftrightarrow s$	c_8
Γ_1	$\times$	$\times$	$\times$
Δ_1	$\times$	$-$	$-$
Δ_2	$-$	$\times$	$-$
Δ_3	$-$	$-$	$\times$

(4) the redundancy of a constraint c_i can be explained by the fact that c_i is not part of an irredundant constraint set Γ determined for KB. Making KB irredundant means to delete those elements from KB which are not in Γ [35]. In our example, we would define the statistics feature (st) as mandatory, i.e., $st \leftrightarrow s$, the requires constraint c_8 can be regarded as redundant.

(5) if a generalization between the feature models KB_g (the more general one allowing more solutions [4]) and KB_s (the more specific one representing a solution-wise subset of KB_g) does not exist, an explanation can indicate those elements (constraints) responsible for a situation where KB_g does not represent a superset of KB_s. A related diagnosis Δ indicates those elements that need to be deleted from KB_g such that the mentioned superset property is fulfilled.

(6) if a configuration $CONF$ is inconsistent with the constraints in KB, an explanation indicates constraints (i.e., variable value assignments) in $CONF$ that induce an inconsistency with KB. Furthermore, a diagnosis Δ indicates minimal sets of constraints (assignments) to be deleted from $CONF$ such that consistency with the constraints in KB can be restored [36].

Table 6 provides an overview of how the discussed analysis operations (see Table 2) can be supported by the conflict detection algorithm QuickXPlain [22] and the diagnosis algorithm FastDiag [24]. Note that these algorithms could also be replaced by other alternatives if conflict set and diagnosis minimality is assured by those algorithms. For example, if $issatisfiable(KB)$ is not fulfilled (i.e., the feature model does not allow the identification of a solution), Γ=QuickXPlain$(KB, \emptyset)$ returns a minimal set of elements of KB as an explanation indicating that those elements are in conflict. Note that more than one such explanation (conflict) could exist in KB. If this is the case, a hitting set directed acyclic graph (HSDAG) based approach [32] can be applied (in combination with QuickXPlain) to identify all related explanations (minimal conflict sets). If a repair proposal is requested, FastDiag can determine a corresponding diagnosis Δ which expresses a minimal set of elements to be deleted from KB in order to be able to restore consistency in KB. In a similar fashion, QuickXPlain and FastDiag can be activated in the other mentioned explanation scenarios.

Table 6
Explanations for FM analysis operations using QuickXPlain [22] and FastDiag [24].

id	question (analysis operation)	explanation (why not?)	repair
1	is satisfiable(KB)?	QuickXPlain$(KB, \emptyset)$	FastDiag$(KB, \emptyset)$
2	is life($f_i \in F, KB$)?	QuickXPlain$(KB, \{f_i = true\})$	FastDiag$(KB, \{f_i = true\})$
3	is optional($f_i \in F, KB$)?	QuickXPlain$(KB, \{f_i = false\})$	FastDiag$(KB, \{f_i = false\})$
4	is irredundant($c_i \in KB, KB$)?	$c_i \notin$ QuickXPlain$(KB, \overline{KB})$	QuickXPlain$(KB, \overline{KB})$
5	is generalization(KB_g, KB_s)?	QuickXPlain$(KB_g, \overline{KB_s})$	FastDiag$(KB_g, \overline{KB_s})$
6	is satisfiable($CONF$, KB)?	QuickXPlain$(CONF, KB)$	FastDiag$(CONF, KB)$

Feature Model Configuration Feature model configuration supports the identification of a decision regarding the inclusion or exclusion of specific features in a configuration. In the context of such (often interactive) configuration sessions, explanations can help the user of a feature model configurator to better understand the reason of a specific feature inclusion or exclusion but also as to why specific features have not been included (i.e., an explanation of the counterfactual case) [37, 12, 17]. Example questions that need to be answered (i.e., explained to users) are shown in Table 7.

Table 7
Explanations for FM configuration settings – *cons/incons* are abbreviations of *consistent/inconsistent*.

id	question	explanation	repair
1	why($\{f_i = true\} \in CONF, KB \cup REQ$)?	$\Gamma \subseteq REQ \cup KB : incons(\Gamma \cup \{f_i = false\})$	$\Delta \subseteq CONF : cons(CONF - \Delta \cup \{f_i = false\})$
2	why not($\{f_i = true\} \in CONF, KB \cup REQ$)?	$\Gamma \subseteq REQ \cup KB : incons(\Gamma \cup \{f_i = true\})$	$\Delta \subseteq CONF : cons(CONF - \Delta \cup \{f_i = true\})$
3	why not(REQ, KB)?	$\Gamma \subseteq REQ : incons(REQ \cup KB)$	$\Delta \subseteq REQ : cons(REQ - \Delta \cup KB)$

Table 8 provides a complete listing of the QuickXPlain and FastDiag activations used in the explanation/repair scenarios shown in Table 7.

Table 8
Explanations for FM configuration settings using QuickXPlain [22] and FastDiag [24].

id	question	explanation	repair
1	why($\{f_i = true\} \in CONF, KB \cup REQ$)?	QuickXPlain($KB \cup REQ, \{f_i = false\}$)	FastDiag($KB \cup REQ, \{f_i = false\}$)
2	why not($\{f_i = true\} \in CONF, KB \cup REQ$)?	QuickXPlain($KB \cup REQ, \{f_i = true\}$)	FastDiag($KB \cup REQ, \{f_i = true\}$)
3	why not(REQ, KB)?	QuickXPlain(REQ, KB)	FastDiag(REQ, KB)

(1) given a configuration $CONF$ which includes a variable value assignment $f_i = true$, a corresponding (counterfactual) explanation identifies those elements (the minimal conflict set Γ) in $REQ \cup KB$ which induce an inconsistency with the negation of $f_i = true$ and are therefore responsible for the inclusion of feature f_i [12, 38]. The corresponding activation of QuickXPlain to determine the minimal conflict set Γ is shown in Table 8. In this setting, QuickXPlain identifies a minimal set of constraints in $REQ \cup KB$ that induce an inconsistency with the constraint $f_i = true$ [12]. In this scenario, FastDiag can be applied to identify a diagnosis Δ consisting of elements from $CONF$ which have to be deleted/adapted in such a way that $f_i = true$ becomes part of $CONF$.

(2) given a configuration including a variable value assignment $f_i = false$, a counterfactual explanation identifies those elements (the minimal conflict set Γ) in $REQ \cup KB$ which induce an inconsistency with the negation of $f_i = false$ and are therefore responsible for the exclusion of feature f_i. A corresponding diagnosis Δ determined by FastDiag would indicate those elements to be adapted[1] in $CONF$ such that the inclusion of f_i becomes possible.

(3) if the user requirements in REQ do not allow the determination of a solution, an explanation will indicate a subset of requirements that induce an inconsistency with KB [22]. A diagnosis Δ includes a set of requirements that need to be deleted/adapted to restore consistency. If we assume the existence of the user requirements $REQ = \{c_9, c_{10}, c_{11}\}$, the corresponding explanations are Γ_1 and Γ_2 with the diagnoses Δ_1 and Δ_2 indicating a way to resolve the conflicts in Γ_1 and Γ_2. In this context, Δ_1 is a singleton diagnosis, i.e., a diagnosis consisting of the minimal number of elements needed to resolve all existing conflicts.

[1]Due to the Boolean nature of feature model variables, an adaptation/deletion of a feature always means either to switch from feature inclusion to exclusion or vice-versa.

Table 9
Explanation of inconsistent user requirements $\{c_9, c_{10}, c_{11}\}$ with corresponding repair options $\Delta_1..\Delta_2$.

requirements	$c_9 : t = true$	$c_{10} : n = true$	$c_{11} : st = false$
Γ_1	$\times$	$\times$	$-$
Γ_2	$\times$	$-$	$\times$
Δ_1	$\times$	$-$	$-$
Δ_2	$-$	$\times$	$\times$

5. Threats to Validity

In this paper, we have shown how to apply consistency-based conflict detection and diagnosis to determine corresponding explanations and repair actions. We are aware of the fact that there are related open issues, for example, instead of deleting constraints from an envisioned superset KB_g to assure an entailment relationship with the specialized feature model KB_s, we could also think about adding additional constraints to KB_s. We regard such open repairs as being beyond the scope of this paper, however, this appears to be a challenging topic for future work. In this paper, we did not conduct performance evaluations of the different explanation and repair settings, however, the used algorithms are well-established and have shown to be applicable in various configuration scenarios which we took as a reason for not including another performance evaluation of QuickXPlain and FastDiag.

6. Conclusions

In this paper, we have shown how to apply conflict detection and diagnosis for the creation of consistency-based explanations and corresponding repair actions. We have identified two major application scenarios which are (1) the generation of explanations and repair actions in the context of supporting analysis operations in FM development and maintenance and (2) the support of users in interactive FM configuration sessions. We have shown how to apply/integrate QuickXPlain as standard algorithm for the detection of minimal conflict sets and FastDiag as an algorithm for the identification of minimal diagnoses. In FM development and maintenance, *why not*? explanations help to understand why specific well-formedness rules defined by analysis operations, fail. In FM configuration, *why not*? and *why*? explanations can be used to explain the inclusion or exclusion of specific features but also the reason as to why no solution exists. Open issues for future work include user studies to better understand in which context to provide which explanation or repair, an analysis of further explanation types to be included in feature model design and configuration, and an analysis of the improvement potentials of existing conflict detection and diagnosis algorithms using machine learning techniques.

Declaration on Generative AI

The authors used ChatGPT for language refinement and improving readability. All AI-generated suggestions were carefully reviewed and edited by the authors, who take full responsibility for the content of this publication.

References

[1] M. Acher, P. Temple, J.-M. Jézéquel, J. A. Galindo, J. Martinez, T. Ziadi, VaryLATEX: Learning Paper Variants That Meet Constraints, in: 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 83–88. doi:10.1145/3168365.3168372.

[2] S. Apel, C. Kästner, An overview of feature-oriented software development, Journal of Object Technology 8 (2009) 49–84.

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

[4] A. Felfernig, A. Falkner, D. Benavides, Feature Models: AI-Driven Design, Analysis and Applications, Springer, 2024. doi:10.1007/978-3-031-61874-1.

[5] J. Galindo, J. Horcas, A. Felfernig, D. Fernandez-Amoros, D. Benavides, Flama: A collaborative effort to build a new framework for the automated analysis of feature models, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B, SPLC '23, ACM, New York, NY, USA, 2023, pp. 16–19. doi:10.1145/3579028.3609008.

[6] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented Domain Analysis (FODA) – Feasibility Study, TechnicalReport CMU – SEI-90-TR-21 (1990).

[7] J. Gu, P. Purdom, J. Franco, B. Wah, Algorithms for the satisfiability (sat) problem: A survey, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1996, pp. 19–152.

[8] D. Benavides, P. Trinidad, A. Cortés, Using constraint programming to reason on feature models, in: W. Chu, N. Juzgado, W. Wong (Eds.), Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'2005), Taipei, Taiwan, Republic of China, July 14-16, 2005, 2005, pp. 677–682.

[9] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V.-M. Le, K. Pilsl, M. Enzelsberger, T. N. T. Tran, An overview of machine learning techniques in constraint solving, J Intell Inf Syst 58 (2022) 91–118. doi:10.1007/s10844-021-00666-5.

[10] A. Felfernig, D. Benavides, J. Galindo, F. Reinfrank, Towards Anomaly Explanation in Feature Models, in: ConfWS-2013: 15th International Configuration Workshop (2013), volume 1128, 2013, pp. 117–124.

[11] A. Felfernig, M. Schubert, S. Reiterer, Personalized diagnosis for over-constrained problems, in: 23rd International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 1990–1996.

[12] A. Felfernig, D. Garber, V.-M. Le, S. Lubos, Causality-based explanations for feature model configuration, in: 19th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS '25, ACM, New York, NY, USA, 2025, pp. 86–90. doi:10.1145/3715340.3715438.

[13] S. Lubos, T. N. T. Tran, A. Felfernig, S. Polat Erdeniz, V.-M. Le, LLM-generated Explanations for Recommender Systems, in: 32nd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct '24, ACM, New York, NY, USA, 2024, pp. 276–285. doi:10.1145/3631700.3665185.

[14] S. Lubos, M. Gartner, A. Felfernig, R. Willfort, Leveraging LLMs to Explain the Consequences of Recommendations, in: 33rd ACM Conference on User Modeling, Adaptation and Personalization, ACM, 2025, pp. 318–322. doi:10.1145/3699682.3728328.

[15] A. Felfernig, M. Wundara, T. Tran, S. Polat-Erdeniz, S. Lubos, M. E. Mansi, D. Garber, V. Le, Recommender systems for sustainability: overview and research issues, Frontiers in Big Data 6 (2023). doi:10.3389/fdata.2023.1284511.

[16] M. Hentze and T. Pett and T. Thüm and I. Schaefer, Hyper Explanations for Feature-Model Defect Analysis, in: 15th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS'21, Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3442391.3442406.

[17] D. Kramer, C. Sauer, T. Roth-Berghofer, Towards explanation generation using feature models in software product lines, in: 9th Workshop on Knowledge Engineering and Software Engineering, CEUR, 2013, pp. 13–23. URL: https://ceur-ws.org/Vol-1070/.

[18] L. Ochoa, O. González-Rojas, T. Thüm, Using decision rules for solving conflicts in extended feature models, in: ACM SIGPLAN International Conference on Software Language Engineering, Association for Computing Machinery, New York, NY, USA, 2015, pp. 149–160. doi:10.1145/2814251.2814263.

[19] N. Tintarev, J. Masthoff, A survey of explanations in recommender systems, ICDEW '07, IEEE

Computer Society, USA, 2007, pp. 801–810. doi:10.1109/ICDEW.2007.4401070.

[20] T. Tran, S. Polat Erdeniz, A. Felfernig, S. Lubos, M. El Mansi, V. Le, Less is more: Towards sustainability-aware persuasive explanations in recommender systems, in: Proceedings of the 18th ACM Conference on Recommender Systems, RecSys ’24, Association for Computing Machinery, New York, NY, USA, 2024, p. 1108–1112. doi:10.1145/3640457.3691708.

[21] T. Tran, A. Felfernig, V. Le, An overview of consensus models for group decision-making and group recommender systems, User Modeling and User-Adapted Interaction 34 (2023) 489–547. doi:10.1007/s11257-023-09380-z.

[22] U. Junker, QuickXPlain: preferred explanations and relaxations for over-constrained problems, in: 19th National Conference on Artifical Intelligence, AAAI’04, AAAI Press, 2004, pp. 167–172.

[23] O. A. Tazl, C. Tafeit, F. Wotawa, A. Felfernig, DDMin versus QuickXplain - An Experimental Comparison of two Algorithms for Minimizing Collections, in: R. Peng, C. E. Pantoja, P. Kamthan (Eds.), 34th International Conference on Software Engineering and Knowledge Engineering, KSI Research Inc., 2022, pp. 481–486. doi:10.18293/SEKE2022-172.

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

[25] V. Le, C. Silva, A. Felfernig, D. Benavides, J. Galindo, T. Tran, FastDiagP: an algorithm for parallelized direct diagnosis, AAAI’23/IAAI’23/EAAI’23, AAAI Press, 2023. doi:10.1609/aaai.v37i5.25792.

[26] C. Gomes, H. Kautz, A. Sabharwal, B. Selman, Satisfiability Solvers, Handbook of Knowledge Representation (2008) 89–134.

[27] M. Mendonça, A. Wasowski, K. Czarnecki, SAT-based analysis of feature models is easy, in: D. Muthig, J. McGregor (Eds.), Software Product Lines, 13th International Conference, SPLC 2009, San Francisco, California, USA, August 24-28, 2009, Proceedings, volume 446 of *ACM International Conference Proceeding Series*, ACM, 2009, pp. 231–240.

[28] V. Myllärniemi, J. Tiihonen, M. Raatikainen, A. Felfernig, Using Answer Set Programming for Feature Model Representation and Configuration, in: 16th International Workshop on Configuration, CEUR, Novi Sad, Serbia, 2014, pp. 1–8.

[29] A. Falkner, G. Friedrich, A. Haselböck, G. Schenner, H. Schreiner, Twenty-five years of successful application of constraint technologies at siemens, AI Mag. 37 (2016) 67–80. doi:10.1609/aimag.v37i4.2688.

[30] T. W. F. Rossi, P. van Beek, Handbook of Constraint Programming, Elsevier, 2006.

[31] S. Gupta, B. Genc, B. O’Sullivan, Explanation in constraint satisfaction: A survey, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4400–4407. doi:10.24963/ijcai.2021/601, survey Track.

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

[33] M. Kowal, S. Ananieva, T. Thüm, Explaining anomalies in feature models, SIGPLAN Not. 52 (2016) 132–143. doi:10.1145/3093335.2993248.

[34] R. Bakker, F. Dikker, F. Tempelman, P. Wogmim, Diagnosing and solving over-determined constraint satisfaction problems, in: Proceedings of IJCAI-93, Morgan Kaufmann, 1993, pp. 276–281.

[35] V. Le, A. Felfernig, M. Uta, T. Tran, C. Silva, WipeOutR: automated redundancy detection for feature models, in: 26th ACM International Systems and Software Product Line Conference - Volume A, SPLC ’22, ACM, New York, NY, USA, 2022, pp. 164–169. doi:10.1145/3546932.3546992.

[36] J. White, D. Benavides, D. Schmidt, P. Trinidad, B. Dougherty, A. Ruiz-Cortes, Automated diagnosis of feature model configurations, Journal of Systems and Software 83 (2010) 1094–1107.

[37] C. Dubslaff, K. Weis, C. Baier, S. Apel, Causality in configurable software systems, in: 44th International Conference on Software Engineering, ICSE ’22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 325–337. doi:10.1145/3510003.3510200.

[38] G. Friedrich, Elimination of spurious explanations, in: 16th European Conference on Artificial Intelligence, ECAI’04, IOS Press, NLD, 2004, pp. 813–817.