Analysis Operations On The Run: Feature Model Analysis in Constraint-based Recommender Systems

Sebastian Lubos
Graz University of Technology
Graz, Austria
slubos@ist.tugraz.at

Alexander Felfernig
Graz University of Technology
Graz, Austria
alexander.felfernig@ist.tugraz.at

Viet-Man Le
Graz University of Technology
Graz, Austria
vietman.le@ist.tugraz.at

Thi Ngoc Trang Tran
Graz University of Technology
Graz, Austria
ttrang@ist.tugraz.at

David Benavides
University of Seville
Seville, Spain
benavides@us.es

José A. Zamudio
University of Seville
Seville, Spain
jzamudio@us.es

Damian Garber
Graz University of Technology
Graz, Austria
dgarber@ist.tugraz.at

ABSTRACT

The development and maintenance of feature models is often an error-prone activity requiring different types of analysis operations that help developers to restore required feature model properties. Fulfilling such properties helps to assure compliance between feature model and corresponding domain variability properties and – at the same time – helps to increase feature model maintainability. In this paper, we propose a set of additional analysis operations that provide insights regarding potential impacts of applying feature models in constraint-based recommendation scenarios where feature models are used to define user preference spaces. Our proposed analysis operations provide a.o. insights into aspects such as feature restrictiveness and product accessibility when applying a constraint-based recommender system. We analyze usage scenarios of the operations on the basis of an example implementation with a digital camera feature model and discuss open research issues.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; • **Social and professional topics** → **Quality assurance**.

KEYWORDS

Variability models, feature models, analysis operations, constraint solving, constraint-based recommender systems

ACM Reference Format:
Sebastian Lubos, Alexander Felfernig, Viet-Man Le, Thi Ngoc Trang Tran, David Benavides, José A. Zamudio, and Damian Garber. 2023. Analysis Operations On The Run: Feature Model Analysis in Constraint-based Recommender Systems. In *27th ACM International Systems and Software Product Line Conference - Volume A (SPLC '23), August 28-September 1, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3579027.3608982

SPLC '23, August 28-September 1, 2023, Tokyo, Japan

ACM ISBN 979-8-4007-0091-0/23/08.
https://doi.org/10.1145/3579027.3608982

1 INTRODUCTION

Feature models (FMs) are used for the representation of variability aspects of software systems and other types of configurable systems such as cars and financial services [1, 4, 13, 18]. Developing feature models (FMs) is often an error-prone process due to cognitive overloads of model engineers (i.e., feature model developers), missing domain knowledge, and outdated feature model elements. In order to deal with such situations, intelligent techniques are needed that help to identify feature model anomalies [6].

Feature model analysis approaches [5, 6, 14, 24] can be differentiated with regard to their need of a solver support (e.g., a SAT solver or constraint solver). Examples of *analysis operations without solver support* are counting operations such as #features in a feature model and statistics about type-specific numbers of constraints (e.g., how many *requires* cross-tree constraints are part of the feature model?). Examples of *analysis operations with solver support* are the checking if a configuration is valid, if a feature is a dead one (i.e., cannot be included in any configuration), if an FM constraint is redundant, and if two feature models are equivalent or one is a generalization of another one. Beyond basic properties of feature models defined in terms of analysis operations, feature model *understandability* and *maintainability* can also be analyzed on the basis of different structural metrics such as branching factors of parent features and cycles created by cross-tree constraints [3].

Existing analysis operations analyze specific structural and logical properties of FMs but often do not take into account the impacts on the application level. In this paper, we focus on constraint-based recommender applications [7–9, 32] where feature models are used to define user preference (requirement) spaces [2]. Following the idea of distinguishing between internal and external variability [23], we define allowed combinations of user preferences as feature models representing the customer (feature) view on a given product

assortment (i.e., the external variability). For example, digital cameras can be described by features such as *goal* (i.e., primary usage, e.g., *landscape* photography) and *lightweight* (i.e., ease of transportation). A corresponding product catalog defines the complete set of available digital cameras at a specific point of time (which can be regarded as a specific type of internal variability). Finally, a set of *constraints* defines relationships between features requested (preferred) by the user and corresponding cameras.

Such scenarios show a need for analysis operations which are of specific importance, for example, for marketing and sales departments responsible for strategic decisions regarding the feature space of the offered product assortment. For a digital camera, we are interested in its *accessibility*, i.e., how "easy" it is for a camera to get recommended. If the accessibility of such a camera is low, this could have an impact on sales since the camera is rarely or never presented to a user (customer). Furthermore, we could be interested in the *restrictiveness* of a feature. For example, if many customers are interested in digital cameras that support a *waterproof* feature, but only a minority of the offered products supports this feature, there could be a need to adapt the offered assortment or at least include new upgrade products that support this feature. On a global level, i.e., not feature-specific level, we need to assure that the preferences of customer communities are satisfied by the provided product assortment. Thus, our scenario is related to a specific type of *scoping* [16, 20, 29] where analysis operations are needed that help to decide which features and also corresponding products should be provided to cover the preferences of customers.

The contributions of our paper are the following. (1) We extend the scope of feature model analysis operations to the impact of variability model use specifically in recommendation scenarios. (2) We propose a set of basic analysis operations (metrics) intended to support the development of constraint-based (feature model based) recommender systems. (3) We demonstrate the application of these analysis operations on the basis of a working example from the domain of digital camera recommendation.

The remainder of this paper is organized as follows. In Section 2, we introduce the concept of feature model based recommendation and a corresponding working example (recommendation of digital cameras). In Section 3, we introduce analysis operations specifically relevant in the mentioned recommendation scenarios. Thereafter, in Section 4, we discuss threats to validity. Finally, we conclude the paper with Section 5.

2 FEATURE MODEL BASED RECOMMENDATION

In the context of this paper, we regard feature model based recommendation as a typical constraint-based recommendation scenario [8] where features are used to represent user preference spaces and constraints are used to describe feature model relationships (and cross-tree constraints) and relationships between user preferences and corresponding products (see Figure 1).

In this scenario, feature models take over the role of representing an external variability [23] representing features that can be selected by users (customers). In contrast, internal variability is represented by a predefined product assortment (product table) in our example setting represented by a set of predefined digital cameras. Such product assortments are not fully configurable but rather specified in an explicit fashion in terms of a corresponding product table specifying the available item types at a specific point of time. If we assume that our example recommender system supports users with limited technical background in photography, the product properties will not be visible or at least selectable in the corresponding recommender application.

A *feature model based recommendation task* (FMR-task) can be defined as a specific type of Constraint Satisfaction Problem [11, 25, 28] (CSP) (see Definition 1).

Definition 1 (FMR-task). A feature model based recommendation task is a tuple (F, C, P, PF, PC, R) where $F = \{f_1..f_n\}$ is a set of Boolean-domain type features describing user preferences and $C = FC \cup FILT$ where $FC = \{c_1..c_l\}$ is a set of relationships and cross-tree constraints in the feature model and $FILT = \{c_{l+1}..c_m\}$ defines constraints between features and corresponding product properties.[1] Furthermore, P is the set of available products (the product assortment), $PF = \{pf_1..pf_r\}$ the set of finite domain type product properties, and PC a constraint in disjunctive normal form describing the product assortment. Finally, $R = \{req_1..req_q\}$ is a set of customer requirements defining intended feature inclusions.

For demonstration purposes, we introduce a feature model (see Figure 2) representing the customer preference space with regard to an assortment of digital cameras (see Table 1).

Each user (customer) has to specify his/her primary *goal* regarding camera usage – the goal has to be one (or more) out of *landscape*, *portrait*, or *sports* photography. Furthermore, the model includes optional features representing *lightweight* cameras, cameras with an *exchangeable lens*, and *waterproof* cameras.

Example 1 (Digital Camera FMR-task). On the basis of this example feature model, we are now able to define a corresponding FMR-task (F, C, P, PF, PC, R). The customer requirements $R = \{req_1 : sports = true\}$ in our working example indicate that the user is interested in a camera that supports the primary goal of sports photography. Note that we assume that each $pf_i \in PF$ describes a product property, for example, *pid* represents the id of a product and *resolution* describes the resolution of a digital camera. Finally, the complete product assortment is described by a constraint in disjunctive normal form (the specifications of products p_1 and p_8 are exemplified – see Table 1 for a complete specification of the products in our working example).

- $F = \{f_1 : camera, f_2 : goal, f_3 : landscape, f_4 : sports, f_5 : portrait, f_6 : lightweight, f_7 : exchangelens, f_8 : waterproof\}$.
- $FC = \{c_1 : camera = true, c_2 : goal = true \leftrightarrow camera = true, c_3 : goal = true \leftrightarrow landscape = true \vee sports = true \vee portrait = true, c_4 : lightweight = true \rightarrow camera = true, c_5 : exchangelens = true \rightarrow camera = true, c_6 : waterproof = true \rightarrow camera = true, c_7 : sports = true \rightarrow waterproof = true\}$.
- $FILT = \{c_8 : landscape = true \rightarrow accumulator = h, c_9 : portrait = true \rightarrow resolution = h, c_{10} : sports = true \rightarrow fps = h, c_{11} : lightweight = true \rightarrow weight = l, c_{12} :$

[1] These constraints are also denoted as filter constraints [8].

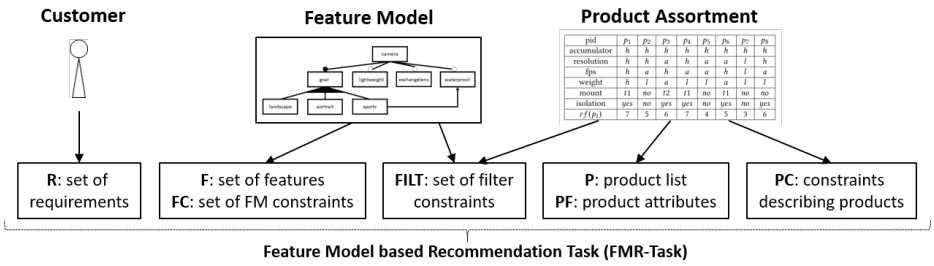

Figure 1: Sketch of feature model based recommendation tasks: the feature model is used to represent the customer preference space (external features [23] selectable by customers) and a product assortment defines the set of available products (internal features [23] not selectable by customers). A specific set of constraints (filter constraints) describes relationships between customer preferences and products [8].

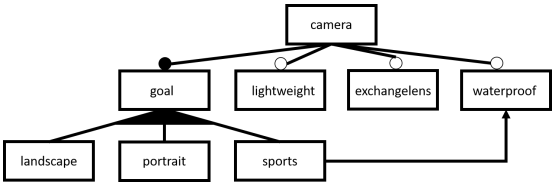

Figure 2: Example feature model representing user preference spaces in a digital camera recommendation scenario.

$exchangelens = true \rightarrow mount \neq no, c_{13} : waterproof = true \rightarrow isolation = h\}$.

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$.
- $PF = \{pid, accumulator, resolution, fps, weight, mount, isolation\}$.
- $PC = \{(pid = p_1 \wedge accumulator = h \wedge resolution = h \wedge fps = h \wedge weight = h \wedge mount = t1 \wedge isolation = h) \vee .. \vee (pid = p_8 \wedge accumulator = h \wedge resolution = h \wedge fps = a \wedge weight = l \wedge mount = no \wedge isolation = h\}$.
- $R = \{req_1 : sports = true\}$.

Table 1: Example product table P. Products $p_i \in P$ are described in terms of their technical properties (pid (*product id*), $accumulator$, $resolution$, fps (*frames per second*), $weight$, $mount$, $isolation$) using the abbreviations $\{\underline{h}igh, \underline{a}verage, \underline{l}ow\}$, $\{t1, t2\}$ as specific mount types, and $\{no\}$ indicates that lenses are fixed. Furthermore, $rf(p_i)$ denotes the number of features supported by product p_i (see Formula 1).

pid	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
accumulator	h	h	h	h	h	h	h	h
resolution	h	h	a	h	a	a	l	h
fps	h	a	h	a	a	h	l	a
weight	h	l	a	l	l	a	l	l
mount	$t1$	no	$t2$	$t1$	no	$t1$	no	no
isolation	h	l	h	h	l	h	l	h
$rf(p_i)$	7	5	6	7	4	6	4	6

Definition 2 (Feature Model based Recommendation). A feature model based recommendation (REC) for a FMR-task is a set of tuples $REC = \cup(rank_i, p_i)$ where $rank_i$ represents the rank assigned to product $p_i \in P$ by a corresponding recommendation function rf, and $\forall(rank_i, p_i) \in REC : consistent(C \cup PC \cup R \cup \{pid = p_i\})$ meaning that each recommended product p_i must be consistent with (support) the constraints in C, PC, and R.

A feature model based recommendation task can be solved, for example, by a constraint solver, a SAT solver, or on the basis of a conjunctive query.[2] Typically, there are different alternative products that satisfy the given set of requested features. For these products, we need the mentioned recommendation function rf to determine a corresponding ranking. For the purposes of our example, we chose a simple ranking function which is shown in Formula 1 (the number of features supported by product p_i).[3] For example, with the exception of not being a lightweight camera (feature f_6) product p_1 supports all remaining features $f_1, f_2, f_3, f_4, f_5, f_7, f_8$, i.e., $rf(p_1) = 7$.

$$rf(p_i) = |\{supported\ features(p_i)\}| \tag{1}$$

Example 2 (Digital Camera Recommendation). In the context of our example, we are able to identify three digital cameras supporting the given customer requirements in $R = \{req_1 : sports = true\}$ which are p_1, p_3, and p_6 (only these products support $c_{10} : sports = true \rightarrow fps = h$). By applying our simplified recommendation function, we are able to derive the following product recommendation (product ranking): $REC = \{(1, p_1), (2, p_3), (3, p_6)\}$ due to the fact that $rf(p_1) = 7$, $rf(p_3) = 6$, and $rf(p_6) = 6$ (see also Table 1).

After having introduced this simplified example of a feature model based recommendation task (and a corresponding recommendation *REC*), we now want to analyze potential impacts on recommender applications. In the following, we discuss the corresponding analysis operations.

3 ANALYSIS OPERATIONS "ON THE RUN"

Following the idea of analyzing potential impacts of using a feature model as a descriptor for user preference spaces, we now introduce the following analysis operations (metrics). These metrics are also

[2] Further implementation details can be found here: https://github.com/AIG-ist-tugraz/AO4FMA and https://github.com/diverso-lab/splc23-analysis-operations.

[3] For alternative recommendation functions we refer to [8]. In case of an equal score, we assume tie breaking with a related randomized ranking.

exemplified on the basis of the feature model in Figure 2 and the corresponding product table (Table 1).[4]

Restrictiveness of Features. For each feature $f_i \in F$ (see also Definition 1), we are interested in the potential impact of selecting this feature, specifically, in terms of its restrictiveness, i.e., to which extent the feature contributes to a reduction of the number of recommendation candidates $p_i \in P : consistent(\{pid = p_i\} \cup C \cup PC)$. For example, when including the feature *sports*, this results in the recommendation candidates p_1, p_3, and p_6. Following this idea, we can define the *restrictiveness* r of a feature f (see Formula 2).

$$r(f) = 1 - \frac{|\{p_i \in P : consistent(\{pid = p_i\} \cup \{f = true\} \cup C \cup PC)\}|}{|\{p_i \in P\}|} \quad (2)$$

Measuring feature restrictiveness is important to understand to which extent the available product assortment (table) P is capable of covering the preferences of a customer community. For example, the share of *waterproof* sports cameras could be increased if the cameras are offered by a winter sports equipment company. In the extreme case, there is no product p_i supporting feature f – in such case, the feature can be regarded as a kind of *dead* feature [6].

Following Formula 2, $r(sports) = 1 - \frac{3}{8} = 0.625$. The same concept can be applied to feature sets, for example, $r(\{sports, exchangelens\}) = 1 - \frac{3}{8} = 0.625$. The complement *excluding* both features, i.e., $r(\{\neg sports \wedge \neg exchangelens\})$ is $1 - \frac{8}{8} = 0$.

Accessibility of Products. We propose to measure product accessibility in terms of the number of times a product is part of a recommendation list, i.e., a product is consistent with $C \cup PC \cup R$. In this context, accessibility can be measured in terms of the share of recommendation sets including p compared to the overall number of different recommendation sets that can be generated from user requirements in R which is $|RECS|$ (see Formula 3).

$$a(p) = \frac{|\{REC \in RECS : (X, p) \subseteq REC\}|}{|RECS|} \quad (3)$$

Given the definition of a feature model based recommendation task, we can calculate all possible recommendations $REC \in RECS$. The total number of recommendations in our working example, i.e., $|RECS|$, is 47 since there are 47 distinct sets of customer requirements. Note that in this context we assume that *RECS* is a bag-type set, since different customer requirements could result in exactly the same recommendation *REC*. Table 2 provides an overview of the number of times a product p_i is included in a set *REC*.

Table 2: Product occurrencies in recommendations.

pid	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	Σ
#occurrencies	31	7	15	31	3	15	3	15	120

Following Formula 3, $a(p_6) = \frac{15}{47}$ since p_6 is part of a recommendation 15 times and the total number of different consistent requirements is 47. The accessibility $a(p_7)$ is $\frac{3}{47}$, i.e., 0.064 which is quite low. This could indicate different issues, for example, p_7 could be outdated or there could also be a need for including additional features making the product more accessible, for example, p_7 could have an excellent usability specifically for beginners, however, this aspect is not covered (taken into account) by the currently offered set of features. Furthermore, the accessibility of p_7 could also be improved by including an upgrade of p_7 supporting additional features, for example, a waterproof version of p_7.

If we would include a new type of low-price sports camera p_9 not supporting the feature *waterproof*, then $a(p_9) = 0$ since the feature model would be too restrictive in this case. Such issues can be resolved, for example, by testing individual products with regard to their support by the corresponding feature model. Different types of analysis operations can be applied to recover accessibility [5].

Product Catalog Coverage. We are also interested in the share of $p_i \in P$ that can be recommended at least once, i.e., $a(p_i) > 0$. This is important due to the fact that we want to avoid situations where some products $p_i \in P$ do not have the chance of being included in at least one recommendation set – such a situation would be indicated by $cv < 1$ (cv is the product catalog coverage – see Formula 4).

$$cv = \frac{|\{p_i \in P : a(p_i) > 0\}|}{|\{p_i \in P\}|} \quad (4)$$

Visibility of Products. Since we are dealing with basic recommendation scenarios, we also have to take into account the item (product) ranking in recommendation lists. The more often a product is shown in a prominent position of a recommendation list[5], the higher the corresponding visibility for customers, since due to primacy effects, products at the beginning of a recommendation list are analyzed more often [21]. Taking into account a products' ranking position X in different recommendation lists results in the following proposed measure of visibility v (see Formula 5).

$$v(p) = 1 - \frac{\Sigma_{REC \in RECS:(X,p) \subseteq REC} X}{\Sigma_{REC \in RECS:(Y,p) \subseteq REC} worstrank(REC)} \quad (5)$$

For example, product p_5 is part of 3 recommendations – within these recommendations, p_5 is ranked 4th in two out of 3 recommendations and 7th in one out of three recommendations (the worst rank in those recommendations is 5, 5, and 8). When applying Formula 5, the overall visibility of p_5 (recommendation lists where p_5 occurs), is quite low ($(1 - \frac{4+4+7}{5+5+8}) = 0.167$), since p_5 is ranked in the almost worst positions. At the same time, p_5 has a low accessibility (Formula 3) since it is only part of three recommendation lists (see Table 2). Having such products could be intended (a product only relevant for specific customer segments – in our case, customers interested in lightweight cameras). At the same time, depending on the popularity of a product, this can also be regarded as a replacement candidate in future product assortment plannings.

Controversy of Features. In the mentioned interactive recommendation scenarios, it can often be the case that a user specifies a set of preferences which do not allow the identification of a solution (recommendation) [10, 12, 26]. For example, it is impossible to find a lightweight sports camera in a recommendation set of our working example. In this context, we are able to measure the controversy

[4]We have calculated the metrics on the basis of Choco (choco-solver.org).

[5]Best *rank*=1 and *worstrank* represents the last position in a recommendation *REC*.

of a feature in terms of the number of times a feature is part of a conflicting set of user requirements R.

To measure the controversy c of a feature f, we need to figure out all possible combinations of requirements R which include f and induce an inconsistency with $C \cup PC$. Then, $c(f)$ represents the share of inconsistency-inducing requirements including f (see Formula 6). For example, the feature *waterproof* is part of 8 different sets of requirements which induce an inconsistency. In total, we have 16 different sets R inducing an inconsistency resulting in $c(\{waterproof\}) = \frac{8}{16} = 0.5$.

$$c(f) = \frac{|\{R \in REQS : \{f = true\} \subseteq R \wedge inconsistent(R \cup C \cup PC)\}|}{|\{R \in REQS : inconsistent(R \cup C \cup PC\}\}|} \tag{6}$$

In this setting, we evaluate (on the level of individual features) how often requirements (R) including feature f derived from the feature model are inconsistent with the constraints in $C \cup PC$, i.e., how often a set of requirements including feature f does not have a corresponding recommendation. In this context, the controversy $c(f)$ can also be regarded as a measure of constrainedness of a feature model with regard to feature f. Omitting the context of a specific feature f, we are also able to measure the global controversy cg of the feature model (i.e., all features).

$$cg = \frac{|\{R \in REQS : inconsistent(R \cup C \cup PC)\}|}{|\{R \in REQS\}|} \tag{7}$$

Note that in such scenarios, if available, we can also apply conflict detection and diagnosis algorithms that help to determine minimal sets of feature selections (so-called conflict sets [17, 30]) that need to be adapted in order to be able to find a recommendation.

Efficiency of Products. Up to now, we have just analyzed different properties of features and products. Now, we go one step further and try analyze the way, an item is perceived by users by taking into account previous user interactions. We try to measure this aspect on the basis of efficiency, i.e., the ratio between the number of product inclusions in recommendation lists compared to the number of times the product has been selected (or even purchased) by a user. We regard this measure of efficiency as a specific type of conversion rate, i.e., how often a displayed product has been converted into a purchased product. Formula 8 provides a basic measure for evaluating the efficiency e of a product. For example, if product p_6 has been selected (or purchased) 222 times after been having shown (part of a recommendation list) 771 times, the corresponding efficiency would be $\frac{222}{771} = 0.288$.

$$e(p) = \frac{|\{selections(p)\}|}{|\{displaycounts(p)\}|} \tag{8}$$

Prominence of Features. In interactive recommendation scenarios, users tend to specify features they are interested in and they know about to some extent. On the other hand, users avoid to choose features they do not know about – in such situations, they often rely on a system recommendation (e.g., in terms of a default [19]). In this context, we specify the prominence pr of a feature in terms of the number of times the feature has been explicitly specified by the user compared to the number of times the feature was included in a final recommendation (see Formula 9). In addition to irrelevancy, a low prominence can be explained by missing prior feature-related knowledge but also by low-quality explanations that help to make a feature transparent for the user.

$$pr(f) = \frac{|\{explicitly\ selected(f)\}|}{|\{included(f)\}|} \tag{9}$$

Popularity of Features and Products. We are also interested in the global popularity of features as well as products. With this we mean how often a feature or a product has been selected by users compared to the total number of feature (product) selections. This aspect can, for example, be measured with Formulae 10 and 11.

$$p(f) = \frac{|\{selections(f)\}|}{|\{feature\ selections\}|} \tag{10}$$

$$p(p) = \frac{|\{selections(p)\}|}{|\{product\ selections\}|} \tag{11}$$

For example, if product p_5 has been selected (purchased) 26 times and the total number of product selections (purchases) is 2.500, the popularity of p_5 is $\frac{26}{2.500} = 0.01$.

4 THREATS TO VALIDITY

For explaining analysis operations, we have introduced a simplified working example from the domain of digital cameras. For reasons of understandability, we have used simplified technical product properties, for example, instead of explicitly stating a frame per second (fps) rate, we have used the attribute domain ($\underline{high}, \underline{low}$). In real-world scenarios, such properties are specified on a detailed technical level. Furthermore, we have used a simplified recommendation function for demonstration purposes, however, this can be replaced with alternative recommendation functions [7, 22, 27].

Some of the proposed analysis operations require the determination of large result sets, for example, the set of different combinations of customer requirements or the set of conflicts than can be induced by customer requirements. Such information can be directly determined in typical constraint-based recommendation scenarios with a limited number of offered items and corresponding feature combinations [8]. In the general case, where products are not defined in the form of a product table and the configuration space is huge and sometimes computationally hard, there is a potential need of approximations. This can be achieved, for example, by applying model counting [15] and local search based conflict detection [31]. We regard related implementations including a detailed performance analysis in different recommendation and configuration settings as a major issue of future work.

We are also aware that the set of proposed analysis operations is limited, i.e., could be extend with further metrics, for example, in the context of recommending financial services we could be interested in the customer communities' preparedness to take risks. Another example are sustainability aspects, for example, we could be interested in the degree of sustainability of selected features.

5 CONCLUSIONS

In addition to existing FM analysis operations, we have proposed operations that take into account the pragmatics of applying a feature model. We have introduced such operations in the context

of constraint-based recommendation where FMs specify customer preference spaces, i.e., which combinations of features can be selected by customers. Selected features (requirements) are the basis for recommending relevant products. The discussed analysis operations can support the process of defining a feature model and also the related task of product space scoping, i.e., deciding about different variability aspects of the product assortment (when setting up a new product assortment but also in the context of adapting already existing assortments and related feature spaces). We have introduced a set of such analysis operations and regard these as a basis for future work focusing on extending this set but also proposing solutions that assure applicability when dealing with complex configuration spaces not restricted by a predefined product table.

Major topics for future work are the following. First, the proposed analysis operations can be further extended to be aware of trends, e.g., shifts in the preferences of customer communities or specific community segments. Depending on such shifts, analysis operations could recommend corresponding adaptations in the feature model and the corresponding product assortment. Second, the proposed measures could also help to analyze impacts on the underlying production processes, for example, what does it mean in terms of needed investments and production capacities to include a new feature. As mentioned, further research is needed to find ways to apply the proposed metrics also in the context of FM configuration scenarios without a predefined product assortment that reduces the size of the solution space.

ACKNOWLEDGMENTS

The work presented in this paper has been partially conducted within the scope of the OpenSpace project funded by the Austrian research promotion agency (FO999891127) and the Data-pl project supported by FEDER/Ministry of Science and Innovation/State Research Agency (PID2022-138486OB-I00).

REFERENCES

[1] S. Apel, D. Batory, C. Kästner, and G. Saake. 2013. *Feature-Oriented Software Product Lines: Concepts and Implementation.* Springer.

[2] M. Atas, A. Felfernig, S. Polat-Erdeniz, A. Popescu, T.N.T. Tran, and M. Uta. 2021. Towards Psychology-Aware Preference Construction in Recommender Systems: Overview and Research Issues. *J. Intell. Inf. Syst.* 57, 3 (dec 2021), 467–489. https://doi.org/10.1007/s10844-021-00674-5

[3] E. Bagheri and D. Gasevic. 2011. Assessing the Maintainability of Software Product Line Feature Models Using Structural Metrics. *Software Quality Journal* 19 (2011), 579–612.

[4] D. Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Software Product Lines Conference*. LNCS, Vol. 3714. Springer, 7–20.

[5] D. Benavides, A. Felfernig, J. Galindo, and F. Reinfrank. 2013. Automated Analysis in Feature Modelling and Product Configuration. In *ICSR'13 (LNCS, 7925)*. Springer, Pisa, Italy, 160–175.

[6] D. Benavides, S. Segura, and A. Ruiz-Cortes. 2010. Automated analysis of feature models 20 years later: A literature review. *Inf. Sys.* 35 (2010), 615–636. Issue 6.

[7] A. Falkner, A. Felfernig, and A. Haag. 2011. Recommendation Technologies for Configurable Products. *AI Magazine* 32, 3 (2011), 99–108. https://doi.org/10.1609/aimag.v32i3.2369

[8] A. Felfernig and R. Burke. 2008. Constraint-Based Recommender Systems: Technologies and Research Issues. In *10th Intl. Conf. on Electronic Commerce (ICEC '08)*. ACM, Article 3, 10 pages. https://doi.org/10.1145/1409540.1409544

[9] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. 2006. An Integrated Environment for the Development of Knowledge-Based Recommender Applications. *Int. J. Electron. Commerce* 11, 2 (2006), 11–34. https://doi.org/10.2753/JEC1086-4415110201

[10] A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan. 2009. Plausible Repairs for Inconsistent Requirements. In *IJCAI*. 791–796.

[11] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. 2014. *Knowledge-based Configuration: From Research to Business Cases.* Morgan Kaufmann Publishers.

[12] A. Felfernig, M. Schubert, and C. Zehentner. 2012. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *AI for Engineering Design, Analysis, and Manufacturing (AIEDAM)* 26, 1 (2012), 53–62.

[13] C. Fritsch, R. Abtand, and B. Renz. 2020. The Benefits of a Feature Model in Banking. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A* (Montreal, Quebec, Canada) *(SPLC '20)*. ACM, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/3382025.3414946

[14] J. Galindo, D. Benavides, P. Trinidad, A. Gutiérrez-Fernández, and A. Ruiz-Cortés. 2019. Automated Analysis of Feature Models: Quo Vadis?. In *23rd International Systems and Software Product Line Conference - Volume A* (Paris, France) *(SPLC '19)*. ACM, 302. https://doi.org/10.1145/3336294.3342373

[15] Carla P. Gomes, Ashish Sabharwal, and Bart Selman. 2009. Model Counting. In *Handbook of Satisfiability*, A. Biere, M. Heule, H. vanMaaren, and T. Walsh (Eds.). Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press, 633–654. https://doi.org/10.3233/978-1-58603-929-5-633

[16] I. John, J. Knodel, T. Lehner, and D Muthig. 2006. A Practical Guide to Product Line Scoping. In *10th International on Software Product Line Conference (SPLC '06)*. IEEE Computer Society, USA, 3–12.

[17] U. Junker. 2004. QuickXPlain: preferred explanations and relaxations for over-constrained problems. In *AAAI 2004*. 167–172.

[18] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. 1990. Feature-oriented Domain Analysis (FODA) – Feasibility Study. *Tech.Rep. – SEI-90-TR-21* (1990).

[19] M. Mandl, A. Felfernig, J. Tiihonen, and K. Isak. 2011. Status Quo Bias in Configuration Systems. In *IEA/AIE 2011*. Syracuse, New York, 105–114.

[20] L. Marchezan, E. Rodrigues, W. Klewerton Guez Assunção, M. Bernardino, F. Basso, and J. Carbonell. 2022. Software product line scoping: A systematic literature review. *Journal of Systems and Software* 186 (2022), 111189. https://doi.org/10.1016/j.jss.2021.111189

[21] J. Murphy, C. Hofacker, and R. Mizerski. 2012. Primacy and Recency Effects on Clicking Behavior. *Computer-Mediated Communication* 11 (2012), 522–535.

[22] J. Pereira, P. Matuszyk, S. Krieter, M. Spiliopoulou, and G. Saake. 2016. A feature-based personalized recommender system for product-line configuration. In *ACM Intl. Conf. on Generative Programming: Concepts and Experiences*. 120–131.

[23] K. Pohl, G. Böckle, and F. Van Der Linden. 2005. *Software Product Line Engineering*. Vol. 10. Springer.

[24] M. Pol'la, A. Buccella, and A. Cechich. 2021. Analysis of Variability Models: A Systematic Literature Review. *Software and Systems Modeling* 20, 4 (2021), 1043–1077. https://doi.org/10.1007/s10270-020-00839-w

[25] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V. Le, K. Pilsl, M. Enzelsberger, and T. Tran. 2022. An Overview of Machine Learning Techniques in Constraint Solving. *JIIS* 58, 1 (2022), 91–118.

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

[27] J. Rodas-Silva, J. Galindo, J. García-Gutiérrez, and D. Benavides. 2019. Selection of software product line implementation components using recommender systems: An application to wordpress. *IEEE Access* 7 (2019), 69226–69245.

[28] F. Rossi, P. van Beek, and T. Walsh. 2006. *Handbook of Constraint Programming*. Elsevier.

[29] K. Schmid. 2000. Scoping Software Product Lines. In *Software Product Lines: Experience and Research Directions*, P. Donohoe (Ed.). Springer US, Boston, MA, 513–532. https://doi.org/10.1007/978-1-4615-4339-8_27

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

[31] C. Uran and A. Felfernig. 2018. Lazy Conflict Detection with Genetic Algorithms. In *Recent Trends and Future Technology in Applied Intelligence*, M. Mouhoub, S. Sadaoui, O. Ait Mohamed, and M. Ali (Eds.). Springer, Cham, 175–186.

[32] M. Uta, A. Felfernig, V. Le, A. Popescu, T. Tran, and D. Helic. 2021. Evaluating Recommender Systems in Feature Model Configuration. In *25th ACM International Systems and Software Product Line Conference – Volume A* (Leicester, United Kingdom) *(SPLC '21)*. ACM, New York, NY, USA, 58–63. https://doi.org/10.1145/3461001.3471144