Semantics-Preserving Merging of Feature Models

Mathias Uta[1], Viet-Man Le[2], Alexander Felfernig[2], Damian Garber[2], Gottfried Schenner[3] and Thi Ngoc Trang Tran[2]

[1]*Siemens Energy AG, Erlangen, Germany*
[2]*Graz University of Technology, Graz, Austria*
[3]*Siemens AG, Vienna, Austria*

Abstract
Large and globally operating enterprises can be confronted with situations where local variability models representing the constraints of individual countries and markets have to be integrated to support a centralized variability management. For example, a car producer operating in the U.S. as well as the European market, could be interested in having a centralized variability (feature) model representing the variability spaces of all supported markets. To achieve this goal, existing local feature models and the corresponding knowledge bases have to be integrated in such a way that the configuration spaces remain the same, for example, for the European market, we would request to support exactly the same set of car configurations that are supported by the corresponding local feature model. In this paper, we introduce an algorithmic approach that supports the merging of feature models in such a way that the semantics of the original feature models is preserved. We present our algorithm and the results of a solver performance analysis which has been conducted on the basis of real-world feature models.

Keywords
Variability Modeling, Feature Models, Model Merging, Redundancy Elimination, Configuration

1. Introduction

Feature models (FMs) are an intuitive way of representing commonality and variability properties of complex systems [1, 2, 3]. Specifically, in scenarios where companies are operating on a global basis, integration scenarios can arise where country or region-specific feature models have to be integrated to support a more globalized variability management. Think about a scenario where a car producer operating in the European and the US market decides to centralize variability management activities. On the technical (feature model) level, formerly region- or country-specific models have to be integrated in a systematic fashion in one centralized variability model. In this paper, we present an algorithmic approach to integrate (merge) two different ("old") feature models (e.g., the feature model $FMUS_{old}$ could denote a local feature model of a US car provider) in a semantics-preserving way where the solution (configuration) spaces of the local feature models are "transferred" to an integrated feature model which reflects exactly the same set of solutions: $solutions(FMUS_{old}) \cup solutions(FMEU_{old})$ equals $solutions(FM_{new})$. In this context, we assume that $FMUS_{old}$ and $FMEU_{old}$ represent the local feature models of a globally operating car manufacturer and FM_{new} is the result of merging the local feature models (and related knowledge bases).

Knowledge base merging has been approached in various ways. For example, the *alignment* of knowledge bases is based on the idea of knowledge base integration by identifying concepts in different knowledge bases that represent the same underlying concept but are represented by different names. Knowledge base alignment is specifically performed in situations where numerous knowledge bases have to be integrated [4]. Knowledge base *merging* is based on a set of predefined merging operations [5, 6], for example, consistency-based merging follows the goal of deriving a maximally consistent set of logical formulae that represent the union of the formulae of the original knowledge bases. Such integrations basically follow the idea of generating maximally satisfiable subsets (of rules) [7], i.e., sets that cannot be further extended (with original rules) without making the resulting knowledge base inconsistent.

Feature model merging [8, 9, 10, 11, 12, 13] is also in the line of the ideas of the previously mentioned approaches. Feature models can become quite large and complex [14], which makes the development and maintenance of single models a challenging task. Following the idea of *separation of concerns* [15], Aydin et al. [16] propose an approach to construct stakeholder-individual feature models which are then merged for the purpose of providing a unified view on the feature space. In the context

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
mathias.uta@siemens-energy.com (M. Uta); vietman.le@ist.tugraz.at (V. Le); alexander.felfernig@ist.tugraz.at (A. Felfernig); dgarber@ist.tugraz.at (D. Garber); gottfried.schenner@siemens.com (G. Schenner); ttrang@ist.tugraz.at (T. N. T. Tran)
0000-0002-1670-7508 (M. Uta); 0000-0001-5778-975X (V. Le); 0000-0003-0108-3146 (A. Felfernig); 0000-0002-3550-8352 (T. N. T. Tran)

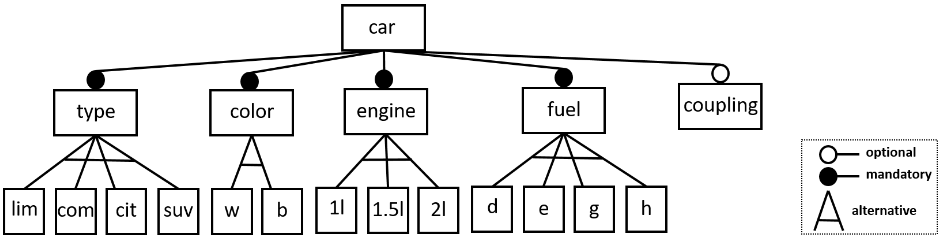

Figure 1: Example basic feature model from the automotive domain where *type* refers to the car type which can be (lim)ousine, (com)bi, (cit)y, and suv. Furthermore, the car *color* can be (b)lack or (w)hite, the *engine* can be 1l, 1.5l, and 2l. *Fuel* can be (d)iesel, (e)lectric, (g)asoline, and (h)ybrid, representing the supported types of fuel. Finally, a *coupling* unit is regarded as an optional feature.

of such a merging process, different "issues" have to be resolved, for example, some stakeholders regard a feature as optional while others think it should be mandatory. Furthermore, depending on the given scenario, feature naming can also become an issue if no "maximum feature set" has been specified ahead of the merging process. For such scenarios, Aydin et al. [16] propose a standard merging procedure that is able to generate a reference feature model, which then serves as a basis for further discussions and decision-making.

With a similar motivation, i.e., making large feature model development easier, Acher et al. [8], propose a set of integration operations for "local" feature models which basically support the goal of integrating local models into a global one. In this context, the authors also specify feature model relationships on a logical basis, for example, one feature model $FM1$ is the specialization of a feature model $FM2$ if the configuration space of $FM1$ is a subset of the configuration space of $FM2$ – see also Thüm et al. [17, 3]. The authors also introduce a *merge* operation where the introduced semantics does not support semantics preservation but requires that the result of the merging operation is equivalent or a superset of the solution (configuration) spaces of the two original feature models, i.e., $solutions(FM1) \cup solutions(FM2) \subseteq solutions(merge(FM1, FM2))$. Such a semantics of a merge operation is also considered in the contributions of Broek et al. [10], Carbonell et al. [11], and She et al. [18].

Following the *union* merge semantics introduced in Schobbens et al. [12], the feature model merging approach presented in this paper focuses on the preservation of the semantics of the source feature models used as an input for the merging procedure. In other words, it supports a semantics-preserving merging where the configuration space of the feature model resulting from a merging operation is exactly the union of the configuration spaces of the original feature models: $solutions(FM1) \cup solutions(FM2) = solutions(merge(FM1, FM2))$ which is more restrictive compared to the union semantics introduced by Acher et al. [8].

Compared to related work on feature model semantics preservation [10, 13], our approach provides a generalization in terms of (1) supporting arbitrary constraint types (in contrast to specific feature model related constraints such as *requires* and *incompatible*) and (2) taking into account redundancy-freeness in terms of assuring that redundant constraints as a result of a merging procedure can be detected and eliminated from the feature model. In our approach, the original feature models and the resulting feature model (result of the merging operation) are represented as constraint satisfaction problems (CSPs) [19]. To demonstrate the applicability of our approach, we present the results of a corresponding performance analysis.

The remainder of this paper is structured as follows. In Section 2, we introduce a working example consisting of simplified feature models from the automotive domain. Using this example, we discuss our algorithmic approach to semantics-preserving feature model merging in Section 3. To show the performance of our approach, we report the results of a corresponding performance evaluation (see Section 4). Finally, we conclude the paper with a discussion of existing threats to validity (Section 5) and a corresponding summary of the contributions of this paper (Section 6).

2. Example Scenario

We now introduce a simplified example of a feature model merging scenario. Our basic underlying assumption is that the original feature models are consistent, i.e., it is possible that at least one solution can be identified and also that the feature set of the original models are the same, i.e., the differences are primarily observable in terms of the constraints defined in the individual models. In our example from the automotive domain, the original feature models are denoted as $FMUS_{old}$ (the original US feature model) and $FMEU_{old}$ which denotes the original European Union feature model. In this context, we assume that these feature models are consistent, i.e., non-void [20], meaning that at least one configuration can be identified for each of those models. Finally, we denote the resulting model (the merging result) as FM_{new}.

Figure 1 represents the basic feature model (i.e., configuration model [21]) that in the following will be used as a working example. This feature model represents all relevant features that can be used to define variability knowledge, i.e., we assume that the same set of features is used to represent variability knowledge in $FMEU_{old}$ and $FMUS_{old}$. Differences in the two variability models can exist in terms of constraints representing individual configuration spaces. In the following, we specify constraints that define the properties of the two original feature models $FMEU_{old}$ and $FMUS_{old}$ represented in terms of individual constraint satisfaction problems (CSPs) representing the European and the US feature model [19].[1] These CSPs are defined in terms of variables with corresponding domain definitions (e.g., *type(lim,com,sit,suv)* denotes the variable *type* with the allowed values) and a corresponding set of constraints [22].

Note that *region* is an additional variable representing a contextual information, i.e., to which region a generated configuration belongs to. Contexts follow the idea of *separation of concerns* [15] which supports a kind of decentralized modeling [23]. For example, using the context variable *region*, the constraint c_{1us} : $fuel \neq h$ would be expressed as c_{1us} : $region = US \rightarrow fuel \neq h$ explicitly indicating that this constraint has to hold for configurations generated on the basis of the $FMUS_{old}$ CSP.

- $FMUS_{old}$: {*region(US), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, c_{1us} : $fuel \neq h$, c_{2us} : $fuel = e \rightarrow coupling = no$, c_{3us} : $fuel = d \rightarrow color = b$}
- $FMEU_{old}$: {*region(EU), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, c_{1eu} : $fuel \neq g$, c_{2eu} : $fuel = e \rightarrow coupling = no$, c_{3eu} : $fuel = d \rightarrow type \neq cit$}

To show the differences between the feature models $FMEU_{old}$ and $FMUS_{old}$, Table 1 provides an overview of the number of solutions supported by the original (region-specific) feature models.

3. Merging Feature Models

In order to be able to merge the two original feature models ($FMEU_{old}$ and $FMUS_{old}$) in a semantics-preserving fashion, each constraint of the two original feature models (represented as CSPs) has to be contextualized using the context variable *region*.[2] Assuming the two regions European Union and US, our context variable could be defined as *region(EU,US)* denoting the variable *region* with the allowed values $\{EU, US\}$. More precisely, each constraint $c_{[i]eu}$ ($c_{[i]us}$) of the "EU" ("US") CSP (derived from the $FMEU_{old}$ ($FMUS_{old}$) feature model) has to be translated into a contextualized representation – see the following example: c_{1eu} : $fuel \neq g$ would be translated into a corresponding contextualized form $c_{1eu'}$: $region = EU \rightarrow (fuel \neq g)$. The resulting contextualized variants of the original knowledge bases $FMEU_{old}$ and $FMUS_{old}$ are denoted as $FMEU'_{old}$ and $FMUS'_{old}$.

- $FMUS'_{old}$: {*region(US), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, c'_{1us} : $region = US \rightarrow (fuel \neq h)$, c'_{2us} : $region = US \rightarrow (fuel = e \rightarrow coupling = no)$, c'_{3us} : $region = US \rightarrow (fuel = d \rightarrow color = b)$}
- $FMEU'_{old}$: {*region(EU), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, c'_{1eu} : $region = EU \rightarrow (fuel \neq g)$, c'_{2eu} : $region = EU \rightarrow (fuel = e \rightarrow coupling = no)$, c'_{3eu} : $region = EU \rightarrow (fuel = d \rightarrow type \neq cit)$}

Note that the solution (configuration) spaces of the contextualized feature models $FMEU'_{old}$ and $FMUS'_{old}$ are the same as those of the original ones (assuming a corresponding context setting, e.g., $region = EU$). Following this argumentation, $solutions(FMEU_{old}) \cup solutions(FMUS_{old}) = solutions(FMEU'_{old} \cup FMUS'_{old})$ which supports our goal of achieving a semantics-preserving merging of the original knowledge bases (see Table 1).

The algorithmic approach to support such a semantics-preserving merging is shown in Algorithm 1 (MergeFM) which itself is a Flama [24] prototype implementation. In a first step (starting with line 6 of MergeFM), those constraints in the contextualized original knowledge bases

Table 1
Number of consistent solutions (configurations) related to the original and contextualized feature models.

Feature model	#configurations
$FMEU_{old}$	108
$FMUS_{old}$	96
$FM' = FMEU'_{old} \cup FMUS'_{old}$	204
$FMEU'_{old} \cap FMUS'_{old}$	84

[1] The feature name abbreviations of $FMEU_{old}$ and $FMUS_{old}$ are defined in Figure 1.

[2] In general, contexts can be represented by a set of variables (i.e., not necessarily one).

(in Algorithm 1 denoted as FM'_1 and FM'_2) can be decontextualized where such a contextualization is not needed (c is a decontextualized version of c'): if $\neg c$ is consistent with $FM'_1 \cup FM'_2$, there (obviously) exist solutions supporting $\neg c$. In such a case, the constraint c must be added in a contextualized fashion to the resulting knowledge base FM, since some feature model configuration (in the other knowledge base) supports $\neg c$. If $\neg c$ is inconsistent with $FM'_1 \cup FM'_2$, c can be added in decontextualized fashion to the resulting knowledge base FM. In a second step (starting with line 14 of Algorithm 1), each constraint of the resulting knowledge base has to be checked for redundancy: in a logical sense, a constraint c can be regarded as redundant if $FM - \{c\}$ is inconsistent with $\neg c$ which means that the constraint does not reduce the solution space of FM and thus logically follows from the constraints in FM (and can be deleted from the constraints in FM).

Algorithm 1 MergeFM(FM'_1, FM'_2) : FM

```
1:  {FM'_1, FM'_2: two contextualized and consistent feature models}
2:  {c': constraint c in contextualized form}
3:  {FM: feature model as a result of MergeFM}
4:  FM ← {};
5:  FM' ← FM'_1 ∪ FM'_2;
6:  for all c' ∈ FM' do
7:      if inconsistent({¬c} ∪ FM' ∪ FM) then
8:          FM ← FM ∪ {c};
9:      else
10:         FM ← FM ∪ {c'};
11:     end if
12:     FM' ← FM' − {c'};
13: end for
14: for all c ∈ FM do
15:     if inconsistent((FM − {c}) ∪ {¬c}) then
16:         FM ← FM − {c};
17:     end if
18: end for
19: return FM;
```

When applying Algorithm 1 to $FMUS_{old}$ and $FMEU_{old}$, the resulting knowledge base FM_{new} looks like as follows. In the resulting knowledge base, the constraint c'_{2us} has been decontextualized. Also, as a result of applying Algorithm 1, constraint c'_{2eu} can be regarded as redundant and thus can be deleted from FM_{new}.[3]

- FM_{new}: {*region(US,EU), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, c'_{1us} : $region = US \rightarrow (fuel \neq h)$, c'_{2us} : $fuel = e \rightarrow coupling = no$, c'_{3us} : $region = US \rightarrow (fuel = d \rightarrow color = b)$, c'_{1eu} : $region = EU \rightarrow (fuel \neq g)$, c'_{3eu} : $region = EU \rightarrow (fuel = d \rightarrow type \neq cit)$}

[3]Alternatively, c'_{2us} could be deleted as a redundant constraint (instead of c'_{2eu}).

On the algorithmic level, the resulting knowledge base FM_{new} is represented in terms of a constraint satisfaction problem. One possibility of representing the integrated knowledge base as the resulting integrated feature model is depicted in Figure 2.

4. Performance Evaluation

In this section, we discuss the results of an initial performance analysis we have conducted to evaluate MergeFM (Algorithm 1) [4]. For this analysis, we applied 8 real-world variability models with varying sizes collected from the S.P.L.O.T. feature model repository [25] and the Diverso Lab's benchmark[5] [26]. Table 4 shows the characteristics of these models (denoted as ϕ). In order to generate "to-be-merged" feature models (FM'_1 and FM'_2) with different shares of contextualized constraints from individual ϕs, we determined the needed number of relationships or cross-tree constraints. We then modified these selected relationships/cross-tree constraints by changing their type, for example, changing mandatory to optional, changing alternative to or, or changing requires to excludes. The resulting models ($FM'_1 \cup FM'_2 = FM'$) are represented as constraint satisfaction problems [19] that differ individually in terms of the number of constraints (#constraints) and the degree of contextualization (expressed as percentages in Tables 2 and 3). In order to take into account deviations in time measurements, we repeated each experimental setting 10 times where in each repetition cycle the constraints in the individual (contextualized) knowledge bases FM' were ordered randomly. All analyses have been conducted with an Apple M1 Pro (8 cores) computer with 16-GB RAM. For evaluation purposes, we used the Choco solver[6] to perform the needed consistency checks.

The number of consistency checks needed for decontextualization is linear in terms of the number of constraints in FM'. A performance evaluation of MergeFM with different knowledge base sizes and degrees of contextualized constraints in FM is depicted in Table 2. In MergeFM, the runtime (measured in terms of milliseconds needed by the constraint solver[7] to find a solution) increases with the number of constraints in FM' and decreases with the number of contextualized constraints in

[4]The dataset, the implementation of algorithms, and evaluation programs can be found at *https://github.com/AIG-ist-tugraz/FMMerging*.
[5]https://github.com/flamapy/benchmarking
[6]choco-solver.org
[7]For the purposes of our evaluation we generated variability models represented as constraint satisfaction problems formulated using the Choco constraint solver – www.choco-solver.org.

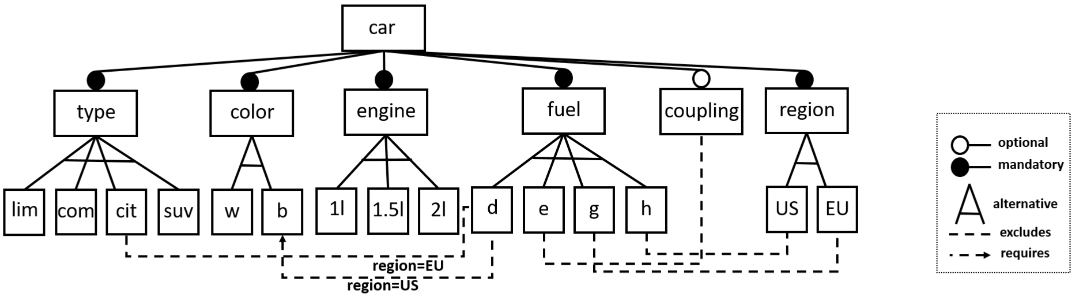

Figure 2: Example integrated feature model derived from FM_{new}. This model includes contextual information (the region) represented as feature(s). Simple contextualized constraints such as c'_{1us} : $region = US \rightarrow (fuel \neq h)$ are translated directly into a corresponding feature model constraint (as excludes relationship), for the representation of more complex constraints such as c'_{3eu} : $region = EU \rightarrow (fuel = d \rightarrow type \neq c)$, the corresponding feature model constraint is textually annotated with the context information (e.g., region=EU). This graphical representation of contexts in feature models follows the idea of contextual diagrams as introduced by Felfernig et. al [23].

Table 2
Avg. runtime (*in seconds*) of MergeFM measured with different *configuration knowledge base sizes* (of FM'_1 and FM'_2) and shares of related contextualized constraints (10-50% contextualization).

feature model (ϕ)	#constraints(ϕ)	10%	20%	30%	40%	50%
IDE	13	0.008	0.007	0.007	0.006	0.006
Arcade	66	0.060	0.056	0.054	0.052	0.054
FQA	101	2.560	2.341	2.794	2.812	3.684
Invest	233	3.018	3.860	4.879	5.781	5.915
Win8	405	154.825	171.516	165.988	158.998	149.323
EMB	1,029	1,621	1,361	1,138	1,043	972
EA	2,670	3,810	3,870	3,899	4,023	4,032
Linux	13,972	45,641	52,711	47,516	56,536	57,034

Table 3
Avg. runtime (*msec*) of the merged configuration knowledge bases (FM) to calculate a configuration measured with different *knowledge base sizes* (of FM) and shares of contextualized constraints in FM (10-50% contextualization).

feature model (ϕ)	#constraints(ϕ)	10%	20%	30%	40%	50%
IDE	13	0.050	0.042	0.039	0.037	0.037
Arcade	66	0.069	0.057	0.060	0.053	0.055
FQA	101	0.072	0.069	0.071	0.072	0.079
Invest	233	4.755	2.992	2.742	2.346	2.293
Win8	405	3.832	4.058	5.385	4.695	4.413
EMB	1,029	22.034	24.190	25.029	25.603	26.980
EA	2,670	40.501	41.227	43.741	45.311	51.483
Linux	13,972	143.698	199.822	143.756	159.515	112.986

Table 4
Feature models used for MergeFM evaluation (IDE=IDE product line, Arcade=Arcade Game PL, FQA=Feature model for Functional Quality Attributes, Invest=Feature model for Decision-making for Investments on Enterprise Information Systems, Win8=Accessibility options provided by Windows 8 OS, EMB=EMB Toolkit, EA=EA 2468, Linux=Linux kernel version 2.6.33.3).

feature model (ϕ)	IDE	Arcade	FQA	Invest	Win8	EMB	EA	Linux
#features	14	61	178	366	451	1,179	1,408	6,467
#hierarchical constraints	11	32	92	41	267	862	1,389	6,322
#cross-tree constraints	2	34	9	192	138	167	1,281	7,650
#CSP constraints	13	66	101	233	405	1,029	2,670	13,972

FM. The increase in efficiency can be explained by the fact that a higher degree of contextualization includes more situations where the inconsistency check in Line 7 (Algorithm 1) terminates earlier (a solution has been found) compared to situations where no solution could be found. In addition, Table 3 indicates that the performance of solution search does not differ depending on the degree of contextualization in the resulting knowledge base *FM*.

Consequently, integrating individual variability models can trigger the following improvements. (1) De-contextualization in *FM* can lead to less cognitive efforts when adapting / extending knowledge bases (due to a potentially lower number of constraints [27] and a lower degree of contextualization). (2) Reducing the overall number of constraints in *FM* can also improve runtime performance of the resulting integrated knowledge base.

5. Threats to Validity

We are aware that our evaluation is in fact based on real-world feature models, however, synthesized variants thereof have been used for MergeFM evaluation purposes. Furthermore, our approach is based on the assumption that the "to-be-merged" feature models have the same set of features, i.e., we assume feature equivalence. In this context, we assume that in real-world scenarios further streamlining tasks (e.g., feature name alignment) have to be completed before MergeFM can be activated. Our basic assumption behind redundancy elimination and de-contextualization in MergeFM is that the understandability and maintainability of feature models can be improved – although already confirmed by related work [27], further research is needed to better understand major impact factors that make feature models (and underlying knowledge bases) easier to understand and maintainable.

6. Conclusions and Future Work

In this paper, we have introduced an approach to the consistency-based merging of variability models represented as constraint satisfaction problems. The approach helps to build semantics-preserving feature models in the sense that the solution spaces of the resulting knowledge bases (result of the merging process) correspond to the union of the solution spaces of the original knowledge bases. Such an approach can be useful in the mentioned integration scenario but as well in situations where different parts (representing different contexts) of a knowledge are developed in a de-centralized fashion and integrated thereafter. Besides the preservation of the original semantics, our approach also helps to make the resulting knowledge base compact in the sense of deleting redundant constraints and not needed contextual information. The runtime performance of our approach is shown on the basis of a first performance analysis with real-world feature models. Future work will include the evaluation of our concepts with further knowledge bases and the development of alternative merging algorithms with the goal to further improve runtime performance. Furthermore, we will evaluate different alternative feature model representations that help to represent contextualized constraints – one such representation has been discussed in this paper.

References

[1] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented domain analysis feasibility study (foda), Technical Report, CMU/SEI-90-TR-021 (1990).

[2] K. Czarnecki, S. Helsen, U. Eisenecker, Formalizing cardinality-based feature models and their specialization, SoftwareProcess: Improvement and Practice 10 (2005) 7–29.

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

[4] L. Galarraga, N. Preda, F. Suchanek, Mining rules to align knowledge bases, in: Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, San Francisco, CA, 2013, pp. 43–48.

[5] J. Delgrande, T. Schaub, A consistency-based framework for merging knowledge bases, Journal of Applied Logic 5 (2007) 459–477.

[6] P. Liberatore, M. Schaerf, Arbitraton (or how to merge knowledge bases), IEEE Transactions on Knowledge and Data Engineering 10 (1998) 76–90.

[7] R. Reiter, A theory of diagnosis from first principles, AI Journal 23 (1987) 57–95.

[8] M. Acher, P. Collet, P. Lahire, R. France, Composing feature models, in: M. van den Brand, D. Gašević, J. Gray (Eds.), Software Language Engineering, Springer, Berlin, Heidelberg, 2010, pp. 62–81.

[9] V. Bischoff, K. Farias, L. Gonçales, J. Victória Barbosa, Integration of feature models: A systematic mapping study, Information and Software Technology 105 (2019) 209–225. URL: https://www.sciencedirect.com/science/article/pii/S0950584916302178. doi:https://doi.org/10.1016/j.infsof.2018.08.016.

[10] P. van den Broek, I. Galvao, J. Noppen, Merging feature models, in: 15th International Software Product Line Conference, Jeju Island, South Korea, 2010, pp. 83–90.

[11] J. Carbonnel, M. Huchard, A. Miralles, C. Nebut, Feature model composition assisted by formal concept analysis, in: 12th International Conference on Evaluation of Novel Approaches to Software Engineering, 2017, pp. 27–37. doi:10.5220/0006276600270037.

[12] P. Y. Schobbens, P. Heymans, J. C. Trigaux, Feature Diagrams: A Survey and a Formal Semantics, in: 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis/St. Paul, MN, USA, 2006, pp. 139–148. doi:10.1109/RE.2006.23.

[13] S. Segura, D. Benavides, A. Ruiz-Cortés, P. Trinidad, Automated Merging of Feature Models Using Graph Transformations, in: Generative and Transformational Techniques in Software Engineering II, volume 5235 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 139–148. doi:10.1109/10.1007/978-3-540-88643-3_15.

[14] M. Acher, H. Martin, L. Lesoil, A. Blouin, J. Jézéquel, D. Khelladi, E. Djamel, O. Barais, J. Pereira, Feature Subset Selection for Learning Huge Configuration Spaces: The Case of Linux Kernel Size, in: 26th ACM International Systems and Software Product Line Conference - Volume A, ACM, 2022, pp. 85–96. URL: https://doi.org/10.1145/3546932.3546997. doi:10.1145/3546932.3546997.

[15] B. Nuseibeh, J. Kramer, A. Finkelstein, Viewpoints: meaningful relationships are difficult!, in: 25th International Conference on Software Engineering, 2003, pp. 676–681. doi:10.1109/ICSE.2003.1201254.

[16] E. A. Aydin, H. Oguztuzun, A. H. Dogru, A. S. Karatas, Merging Multi-view Feature Models by Local Rules, in: 9th International Conference on Software Engineering Research, Management and Applications, Baltimore, MD, USA, 2011, pp. 140–147. doi:10.1109/SERA.2011.34.

[17] T. Thüm, D. Batory, C. Kästner, Reasoning about edits to feature models, in: 31st International Conference on Software Engineering, ICSE '09, IEEE Computer Society, USA, 2009, pp. 254–264. URL: https://doi.org/10.1109/ICSE.2009.5070526. doi:10.1109/ICSE.2009.5070526.

[18] S. She, U. Ryssel, N. Andersen, A. Wąsowski, K. Czarnecki, Efficient synthesis of feature models, Information and Software Technology 56 (2014) 1122–1143. URL: https://www.sciencedirect.com/science/article/pii/S0950584914000238. doi:https://doi.org/10.1016/j.infsof.2014.01.012.

[19] E. Tsang, Foundations of Constraint Satisfaction, Academic Press, London, 1993.

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

[21] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1st ed., Morgan Kaufmann Publishers, 2014.

[22] D. Benavides, P. Trinidad, A. Ruiz-Cortes, Using constraint programming to reason on feature models, in: 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'2005), Taipei, Taiwan, 2005, pp. 677–682.

[23] A. Felfernig, D. Jannach, M. Zanker, Contextual diagrams as structuring mechanisms for designing configuration knowledge bases in uml, in: 3rd International Conference on the Unified Modeling Language (UML2000), volume 1939 of *Lecture Notes in Computer Science*, Springer, York, UK, 2000, pp. 240–254.

[24] 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: 27th ACM International Systems and Software Product Line Conference - Volume B, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 16–19. URL: https://doi.org/10.1145/3579028.3609008. doi:10.1145/3579028.3609008.

[25] M. Mendonca, M. Branco, D. Cowan, S.P.L.O.T.: Software Product Lines Online Tools, in: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA '09, ACM, New York, NY, USA, 2009, pp. 761–762. doi:10.1145/1639950.1640002.

[26] R. Heradio, D. Fernandez-Amoros, J. A. Galindo, D. Benavides, D. Batory, Uniform and scalable sampling of highly configurable systems, Empirical Software Engineering 27 (2022) 44.

[27] A. Felfernig, M. Mandl, A. Pum, M. Schubert, Empirical knowledge engineering: cognitive aspects in the development of constraint-based recommenders, in: Proceedings of the 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 631–640.