Reviewing Diagnosis Solutions for Valid Product Configurations in the Automated Analysis of Feature Models

Cristian L. Vidal-Silva
Ingeniería Civil Informática
Escuela de Ingeniería, Universidad Viña del Mar,
Viña del Mar, Chile

***Abstract*—A Feature Model (FM) is an information model to represent commonalities and variabilities for all the products of a Software Product Line (SPL). The complexity and big size of real feature models makes their manual analysis for determining the product configurations validity a tedious or even infeasible task. Efficient solutions for the diagnosis of errors in the Automated Analysis of Feature Models (AAFM) already exist such as FMDiag and FlexDiag. Thus, this work describes the fundamental basis for both diagnosis algorithms to apply the first of them on the validity of FM product configurations. The results highlight the applicability and efficiency of FMDiag and invite us to look for additional applications of that algorithm in the AAFM scenarios.**

***Keywords*—AAFM; Feature Model; valid product; valid configuration; FMDiag; FlexDiag**

I. Introduction

The main goal of Software Product Line Engineering (SPLE) is the reuse of assets (features) for the products definition looking for improving the quality, accelerating the production and time-to-market, and reducing the production costs of all the process [1]. Such as Bastos et al. [2] highlight, organizations emphasize the proactive reuse, interchangeable components, and multi-product planning cycles for the faster and cheaper construction of high-quality products.

According to [3], the processes of domain and application engineering constitute the SPLE of which, the first one looks for a development for reuse, that is, defining software products in terms of commonalities and variabilities, and also their constraints, whereas the second process refers to development with reuse by the products derivation from Feature Models (FMs).

Different proposals of variability modeling techniques exist for the common and varying assets representation within a SPL, but FMs are one of the most widely used techniques for the SPL variability modeling [4]. Feature Model (FM) is a visual language for accomplishing the domain engineering in Software Product Line (SPL) [5], that is, FM is adequate for a SPL representation [6]. The application engineering process results in the set of valid features selection (products). Figure 1 [7] shows a feature model for the variability of a simple car. As we can appreciate, different types of model elements exist which describe constraints. For example, such as Apel et al. [7] argue, a car always has a body, transmission, and engine (filled circle), a car does not necessarily have a trailer (empty circle), the engine can be powered with gasoline or with electricity or both (filled arc).

The configuration of a FM refers to the process of choosing and unchoosing features in a feature model to reach a full configuration [8]. Paz [9] remarks that making decisions in a FM for deriving valid products usually is not a straightforward task. Example of valid products for the FM of Figure 1 are P_1 = $\{Car, Body, Transmission, Engine, Automatic, Gasoline\}$ and P_2 = $\{Car, Body, Transmission, Engine, Manual, Electric\}$.

The Automated Analysis of Feature Models (AAFM) is about computer-assisted operations for obtaining information from FMs. The "valid product" or "valid configuration" is an AAFM operation that receives a FM and a product to return a value to know if the product is or not valid, that is, if the product belongs to the set of products that the FM represents [10].

For the dynamism and growing size of SPLs, defining and applying efficient solutions for an AAFM operation to determine "valid product" instances constitute a relevant work.

FMDiag [11] and FlexDiag [12] are two efficient diagnosis solution on FMs, that is, to determine minimal sets of constraints in the configuration knowledge base (FM) to delete or adapt for making the remaining constraints a consistent set. This article look to answer: How can we apply existing diagnosis solutions for the "valid product" AAFM operation? This paper describes the theory of FMDiag an FlexDiag for applying them into the "valid product" AAFM operation to potentially get efficient solutions. For the FMDiag and FlexDiag algorithmic simplicity and their previously known efficiency on diagnosis, their application represent a significant advance for the SPL community.

This work uses the FAMA tool [13] [14] that includes implementations of the FMDiag and FlexDiag for different reasoner tools. Specifically, I applied the FAMA working on the Choco reasoner to adapt and apply FMDiag and FlexDiag for the "valid product" analysis to evaluate and validate their performance and results.

In the next, this paper structures as follows: Section II describes related works for the "valid product" AAFM operation; Section III describes main FMs ideas and discusses the main structure and functioning of the diagnosis algorithms FMDiag and FlexDiag; Section IV presents main concepts of the Valid

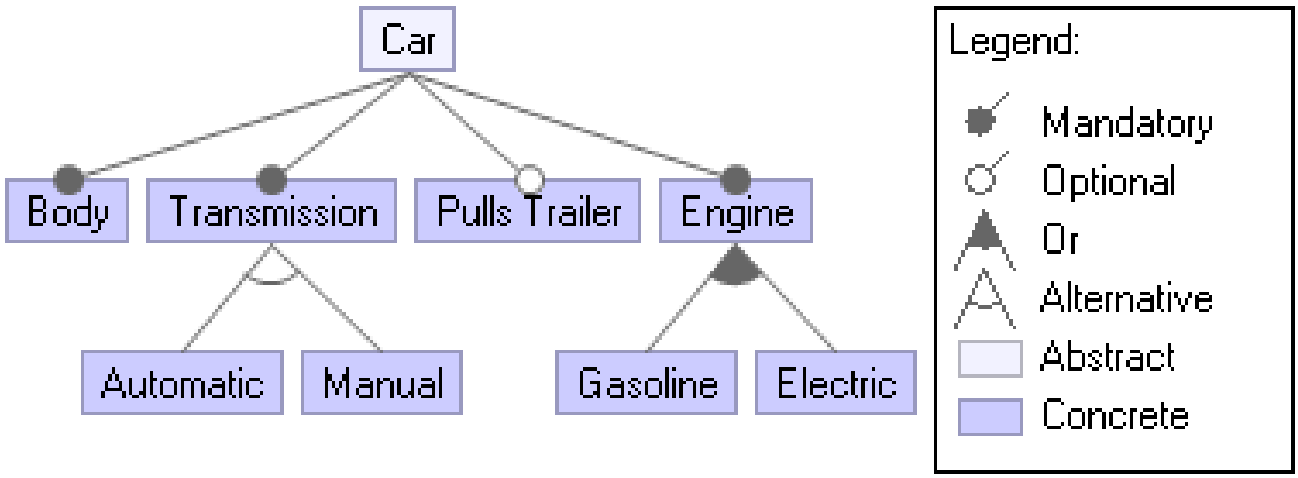

Fig. 1. Feature model example for a simple car variability model.

product AAFM operation; Section V shows and highlights the results of applying FMDiag for the "valid product" AAFM operation, and Section VI concludes and gives feature work ideas.

II. Related Work

AAFM constitutes an active research and application area. Next, we mentioned only a few of related work:

- White et al. [15] uses a CSP solver for debugging basic feature model configurations and automating the evolving product configurations on basic feature models.
- The work of Ross-Frantz et al. [16] uses the Orthogonal Variability Model (OVM) instances for their mapping on a CSP for the developing of FaMa-OVM to identify void models, dead and false optional features.
- Hu et al. [17] proposes an approach for evolving basic feature model and analyze their products evolution. They present refactoring strategies and semi-automated support for the commonality extraction and feature refactoring.
- Mauro et al. [18] present a framework for the modeling and evolving reconfiguration of context-aware SPL instances. They consider the environmental impact on updated features and new contextual information on the SPL evolution. That research work defines a meta-model for Hybrid Feature Model (HyFM) for the attributes in features and represent contextual information. According to their findings, being possible of proposing configuration on FMs of up 10.000 features in less than a minute is more than enough for the majority of the daily use cases.

III. Feature Models & AAFM Diagnosis

A Feature Model (FM) is a information modeling tool useful for the representation all the products of a SPL along with their features and relations [10]. Different kind of FMs already exist such as cardinality-based FMs, extended-FMs which support the attributes definition, and basic FMs. For their simplicity and usability, in this work we use basic FMs. The following relationships are distinguishable in a FM:

- Unary relations. These are between a father and a child feature, and we can distinguish between mandatory (black circle in top of the child feature) and optional (white circle in top of the child feature) child features. In Figure 1, (Car, Body) and (Car, Pulls Trailer) are examples of mandatory and optional relations.
- Set relations. These relations define a father feature of a set of children features, and we can distinguish between optional and alternative sets, that is, for the optional set we can select more than one feature if their father is selected, and for the alternative set only one child feature from the options set of features. In both cases of set features, we need to select at least one feature. In Figure 1, (Engine, {Gasoline, Electric} and (Transmission, {Automatic, Manual} are examples of optional and alternative set relations of features, respectively.

Such as Benavides et al. [10] highlight, defining and optimizing AAFM operations represent a current and active research area such as the diagnosis on a FM to discover and inform possible mistakes. FMDiag [11] and FastDiag [19] are diagnosis solution applicable on FMs and both solutions determines a minimal diagnosis for a given ranking of preferences.

Structurally, FMDiag defines a base function (algorithm 1) and a recursive function (algorithm 2). The base function receives a set of customer requirements *S* to analyze its consistency, and a configuration knowledge base *AC* that includes *S*, that is, *AC* without *S* should be consistent. FMDiag returns an empty diagnosis if *S* is not empty and $(AC - S)$ is non-consistent. Otherwise, the base function calls the recursive function Diag for $D = \emptyset$, *S* and *AC*.

The function Diag receives *D* that represents a subset of the main set to analyze previously removed from the base of

knowledge *AC* (initially *D* is empty), *S* (the current set in analysis), and *AC* (the current base of knowledge that contains S). Diag presents two base cases:

- If *D* is not $\emptyset$ and the current *AC* is consistent then Diag returns an empty diagnosis, that means the diagnosis was not in *S*, that is, *D* contains the diagnosis. This situation can occur after the second recursive call (in the first call *D* is empty and *AC* is not-consistent).

- If the size of *S*, that is, the number of constraints in *S* were either $<= m$ for FlexDiag or $= 1$ for FMDiag then Diag returns the current set *S* as a diagnosis. That base case can occur if the previous described one is not true, that is, *S* contains a diagnosis.

If no base-case were valid, we know that *S* contains the diagnosis and the size of *S* is not minimal. Hence, the function Diag partitions *S* in S_1 (from the 1st until the constraint in the middle) and S_2 (from the middle + 1 to the last constraint) to go to the 1st recursive call for the arguments $D = S_2$, $S = S_1$, and $AC = AC - S_2$, that is, to take off S_2 from AC to evaluate the consistency of *AC* (*AC* the second half of the current *S*). If *AC* were consistent, because current *D* is not empty, Diag returns $\emptyset$ (the diagnosis is in *D*). If *AC* were non-consistent, the current set *S* contains inconsistencies and repeat the division process on the current *S*, and new recursive calls proceed again. The Diag function receives $D = S_2$, $S = S_1$, and $AC = AC - S_2$ from the 1st recursive call. The first recursive call returns Δ_1. The second recursive call receives $D = \Delta_1$, $S = S_2$, and $AC = AC - \Delta_1$ (the second recursive call needs the result of the 1st one to proceed). The 2nd recursive call returns Δ_2. Thus, the function Diag returns $\Delta_1 \cup \Delta_2$ as a diagnosis.

Applying FMDiag for diagnosis on product configurations seems direct, that is, *S* consists of the constraints for the features selection of the product in analysis, and *AC* contains the set of constraints for the consistent FM plus *S*.

Algorithm 1 FMDiag(S, AC): Δ

```
if isEmpty(S) or inconsistent(AC-S) then
    return ∅;
else
    return Diag(∅, S, AC);
end if
```

Algorithm 2 Diag(D, $S = \{s_1, ..., s_r\}$, AC): Δ

```
if D ≠ ∅ and consistent(AC) then
    return ∅;
end if

if FlexActive then
    if size(S) <= m then
        return S;
    end if
else
    if size(S) = 1 then
        return S;
    end if
end if
k = ⌈r/2⌉;
S1 = {s1, ..., sk}; S2 = {sk+1, ..., sr};
Δ1 = Diag(S2, S1, AC − S2);
Δ2 = Diag(Δ1, S2, AC − Δ1);
return(Δ1 ∪ Δ2);
```

Syntactically, the main difference between FMDiag and FlexDiag are the size of the minimal set to return, but semantically their differences are highly relevant. Such as the works of Felfernig et al. [12] [20] remark, FlexDiag is more adequate for diagnosis within time limits regardless the trade-offs between diagnosis quality and performance of the diagnostic search.

IV. Valid Product AAFM Operation

Products of an SPL are the combination and configuration of features through the assembly of corresponding and reusable artifacts [21]. Such as Benavides et al. [10] describe, "valid product" is an AAFM operation example that receives as input a FM and a product (a set of features), and returns a boolean result that determines either the product belongs to the set of products that the feature model represents or not. The products P_1 and P_2 are examples of valid products of the FM of Figure 1, whereas P_3 = {Car, $Body$, $Transmission$, $PullsTrailer$, $Engine$, $Automatic$, $Manual$, $Gasoline$, $Electric$} is a non-valid product: P_3 selects all features and does not respect a defined alternative set cross-tree constraint in the model. Figures 2 and 3 show the features selection (green features) for valid products of P_1 and P_2, respectively whereas Figure 4 shows the features selection (green features) for the non-valid products P_3 .

Applying FMDiag and FlexDiag for the validity of FM product would be direct: we need to define the constraints for the product to analyze, and the constraints for the FM definition to review in. We can repeat that process for analyzing multiple products.

V. Application Results & Discussion

For space reasons we only present the FMDiag testing results on the diagnosis of "valid product". FlexDiag follows the same functioning idea and by applying it we can get more efficient and lesser precise solutions. Tables 1 and 2 present the steps of applying FMDiag for the diagnosis of product P_3 and P_4 for the SPL of Figure 1 to appreciate the algorithmic functioning for the diagnosis on the "valid

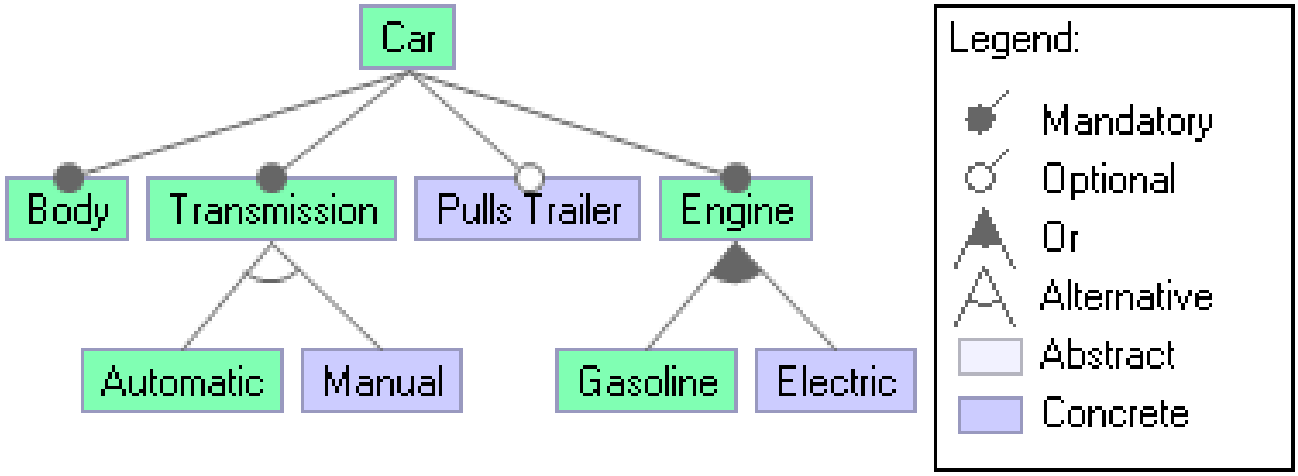

Fig. 2. P_1: A valid product example for the simple car variability model.

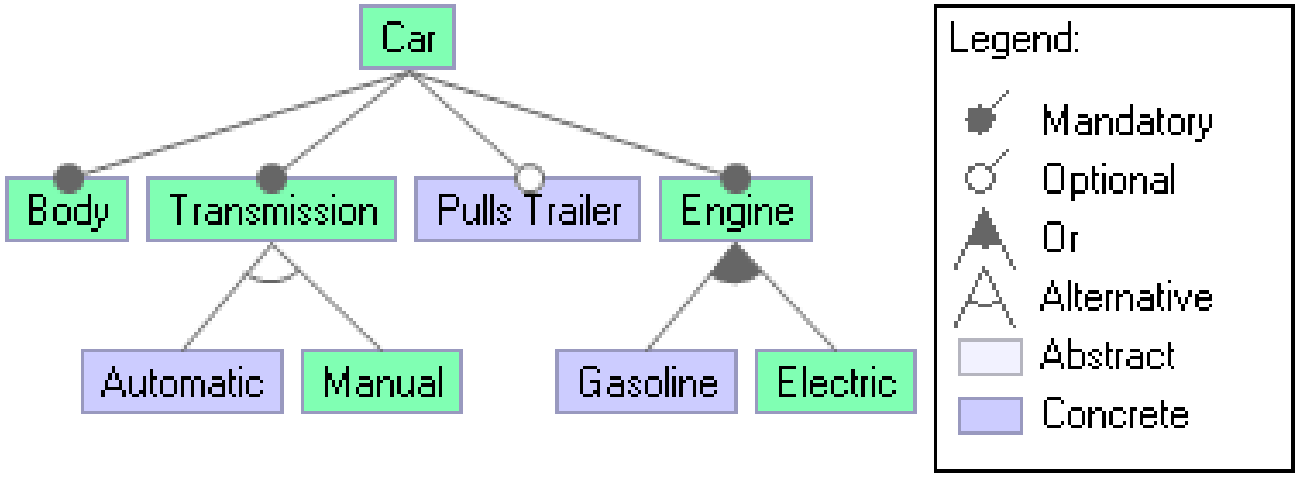

Fig. 3. P_2: A valid product example for the simple car variability model.

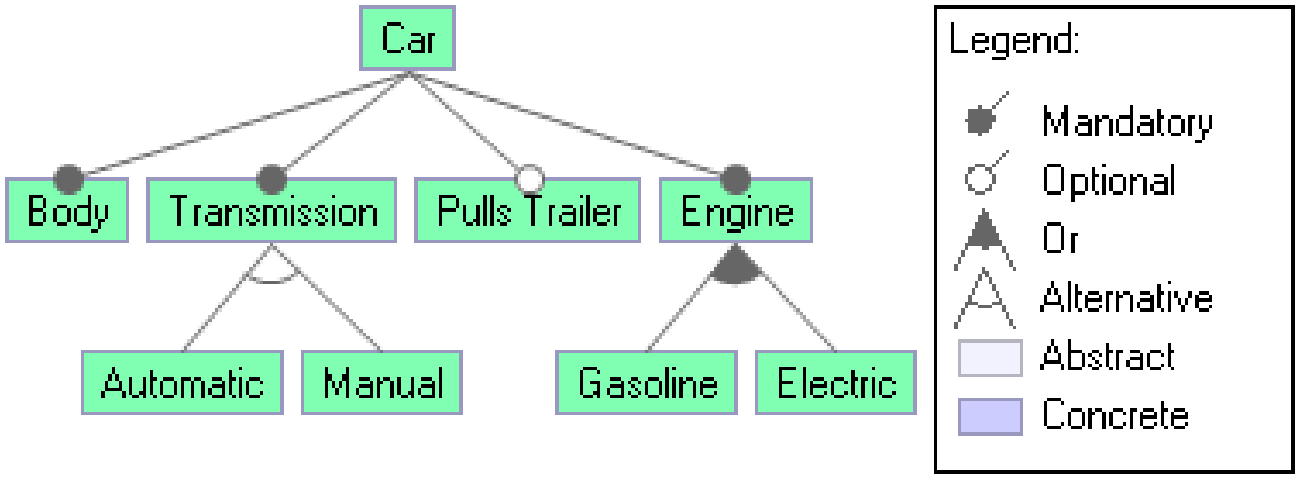

Fig. 4. P_3: A non-valid product example for the simple car variability model.

TABLE I. FMDIAG APPLICATION EXAMPLE FOR THE DIAGNOSIS OF INCONSISTENCIES OF P_3.

Step	D	S	AC	S_1	S_2	return	prev
1	$\emptyset$	{ *Car*, *Body*, *Transmission*, *PullsTrailer*, *Engine*, *Automatic*, *Manual*, *Gasoline*, *Electric* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Gasoline*, *Electric*, *Engine* }	{ *Car*, *Body*, *Transmission*, *PullsTrailer* }	{ *Engine*, *Automatic*, *Manual*, *Gasoline*, *Electric* }	{ *Electric* }	0
2	{ *Engine*, *Automatic*, *Manual*, *Gasoline*, *Electric* }	{ *Car*, *Body*, *Transmission*, *PullsTrailer* }	{ *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *PullsTrailer* }	$\emptyset$	$\emptyset$	$\emptyset$	1
3	$\emptyset$	{ *Engine*, *Automatic*, *Manual*, *Gasoline*, *Electric* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Gasoline*, *Electric*, *Engine* }	{ *Engine*, *Automatic* }	{ *Manual*, *Gasoline*, *Electric* }	{ *Electric* }	1
4	{ *Manual*, *Gasoline*, *Electric* }	{ *Engine*, *Automatic* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *PullsTrailer*, *Engine* }	$\emptyset$	$\emptyset$	$\emptyset$	3
5	$\emptyset$	{ *Manual*, *Gasoline*, *Electric* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Gasoline*, *Electric*, *Engine* }	{ *Manual* }	{ *Gasoline*, *Electric* }	{ *Electric* }	3
6	{ *Gasoline*, *Electric* }	{ *Manual* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Engine* }	$\emptyset$	$\emptyset$	$\emptyset$	5
7	$\emptyset$	{ *Gasoline*, *Electric* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Gasoline*, *Electric*, *Engine* }	{ *Gasoline* }	{ *Electric* }	{ *Electric* }	5
8	{ *Electric* }	{ *Gasoline* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Gasoline*, *Engine* }	$\emptyset$	$\emptyset$	$\emptyset$	7
9	$\emptyset$	{ *Electric* }	{ *Automatic*, *Transmission*, *Body*, *c1*, *c2*, *c3*, *Car*, *c4*, *c5*, *c6*, *Manual*, *PullsTrailer*, *Gasoline*, *Electric*, *Engine* }	$\emptyset$	$\emptyset$	{ *Electric* }	7

product" AAFM operation. FMDiag can directly determine the validity of a product, and return involved constraints for a non-valid configuration.

Tables 3 and 4 show the diagnosis results for the "valid product" operations on small size FMs of the SPLOT [22] and big size FMs generated by the Betty tool [23], respectively. FMDiag reaches a fast performance for the feature models diagnosis even though FMDiag results contains only one inconsistency cause. Felfernig et al. [11] detail how to obtain all the diagnosis of a FM applying FMDiag. FMDiag and FlexDiag are appropriate for interactive scenarios which demand for a fast-time for diagnosis.

The works of Felfernig et al. [11] and [12] compared the performance of FMDiag and FlexDiag to Constraint Satisfaction Problem (CSP) solutions to highlight the efficiency advantages of both solutions for preferred diagnosis.

Such as Felfernig et al. [24] remark, variability management is essential for the product configurations of SPL looking for extensive customization to attend different clients' needs. The AAFM solutions such as FMDiag [11] and FlexDiag [12] [20] efficiently achieve that goal. Undoubtedly, FMDiag and FlexDiag represent and advance concerning the HSDAG solution [25].

Main limitations of this study are the dependency of discussed solutions concerning CSP tools.

VI. CONCLUSION

Defining a "valid product" configuration is a current and demanding activity in product-lines areas such as in-component-based software and SPL applications. We show that existing diagnosis solution are applicable for efficiently determining the validity of a product configuration which can be non-valid for non-respecting the model constraints. Thus, our defined research question is effectively answered. Even though we show FMDiag application results only, FlexDiag is usally more efficient, but lesser precise.

For the efficient obtained results of applying FMDiag and FlexDiag solutions and their algorithmic simplicity, we can look for new applications on the AAFM area since diagnosis algorithms are able for error detection in general.

REFERENCES

[1] W. Heider, R. Rabiser, and P. Grünbacher, "Facilitating the evolution of products in product line engineering by capturing and replaying configuration decisions," *Int. J. Softw. Tools Technol. Transf.*, vol. 14, no. 5, pp. 613–630, Oct. 2012. [Online]. Available: https://doi.org/10.1007/s10009-012-0229-y

TABLE II. FMDIAG APPLICATION EXAMPLE FOR "VALID PRODUCT" OPERATIONS OF SPLOT FMS.

Model Name	Model Features #	Model Dependencies #	Product Size	Diagnosis Size	Time (ms)
$aircraft\_fm.xml$	13	0	13	1	452
			6	1	33
$car\_fm.xml$	14	1	14	1	35
			7	1	19
$connector\_fm.xml$	20	0	20	1	85
			10	1	35
DELL-LAPTOP-NOTEBOOK-FM.xml	47	105	35	1	171
			32	1	78
$fame\_dbms\_fm.xml$	21	0	21	1	59
			10	1	26

TABLE III. FMDIAG APPLICATION EXAMPLE FOR "VALID PRODUCT" OPERATIONS OF BIG SIZE FMS.

Model Name	Model Features #	Model Dependencies #	Product Size	Diagnosis Size	Time (ms)
model-50-10-1.xml	50	5	49	1	17
			24	1	16
model-50-100-1.xml	50	50	45	1	41
			24	1	15
model-100-10-1.xml	100	10	98	1	805
			50	1	186
model-100-100-1.xml	100	100	95	1	176
			48	1	162
model-1000-10-1.xml	1000	100	997	1	1149
			498	1	727
model-1000-100-1.xml	1000	*1000	994	1	993
			497	1	685
model-1000-30-1.xml	1000	300	996	1	464
			497	1	442
model-2000-10-1.xml	2000	200	1996	1	896
			998	1	602
model-2000-100-1.xml	2000	2000	1994	1	1403
			996	1	963
model-2000-30-1.xml	2000	600	1996	1	865
			999	1	669

[2] J. F. Bastos, P. A. da Mota Silveira Neto, P. OLeary, E. S. de Almeida, and S. R. de Lemos Meira, "Software product lines adoption in small organizations," *J. Syst. Softw.*, vol. 131, no. C, pp. 112–128, Sep. 2017. [Online]. Available: https://doi.org/10.1016/j.jss.2017.05.052

[3] S. Chen and M. Erwig, "Optimizing the product derivation process," *2011 15th International Software Product Line Conference*, pp. 35–44, 2011.

[4] J. A. Galindo, H. Turner, D. Benavides, and J. White, "Testing variability-intensive systems using automated analysis: An application to android," *Software Quality Journal*, vol. 24, no. 2, pp. 365–405, Jun. 2016. [Online]. Available: http://dx.doi.org/10.1007/s11219-014-9258-y

[5] C. Vidal, D. Benavides, J. Galindo, and P. Leger, "Exploring the synergies between joing point interfaces and feature-oriented programming." Cantabria, Spain: Proceeding of the Doctoral Symposium at 21th International Systems and Software Product Line Conference, 2015.

[6] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (foda) feasibility study," in *Proceedings of the 34th International Conference on Software Engineering*, no. CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, 1990. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231

[7] S. Apel and C. Kästner, "An overview of feature-oriented software development," *Journal of Object Technology*, vol. 8, no. 5, pp. 49–84, 2009. [Online]. Available: https://doi.org/10.5381/jot.2009.8.5.c5

[8] D. Benavides, A. Felfernig, J. A. Galindo, and F. Reinfrank, "Automated analysis in feature modelling and product configuration," in *Safe and Secure Software Reuse*, J. Favaro and M. Morisio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–175.

[9] A. Paz, "Fama framework: Fama extensions development guide," in *Research Report*. Seville, Spain: University of Seville, August, 2014.

[10] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review," *Journal Information Systems*, vol. 35, no. 6, pp. 615–636, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.is.2010.01.001

[11] A. Felfernig, D. Benavides, J. Galindo, and F. Reinfrank, "Towards anomaly explanation in feature models," in *Proceedings of the 15th International Configuration Workshop*, Vienna, Austria, August 2013.

[12] A. Felfernig, R. Walter, and S. Reiterer, "Flexdiag: Anytime diagnosis for reconfiguration," in *Proceedings of the 17th International Configuration Workshop*, Vienna, Austria, September 2015.

[13] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz–Cortés, "FAMA: Tooling a framework for the automated analysis of feature models," in *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, Limerick, Ireland, 2007, pp. 129–134. [Online]. Available: http://www.lsi.us.es/ trinidad/docs/benavides07-vamos.pdf

[14] P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura, and A. Jimenez, "Fama framework," in *Proceedings of the 2008 12th International Software Product Line Conference*, ser. SPLC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 359–. [Online]. Available: https://doi.org/10.1109/SPLC.2008.50

[15] J. White, D. Benavides, D. C. Schmidt, P. Trinidad, B. Dougherty, and A. R. Cortés, "Automated diagnosis of feature model configurations," *Journal of Systems and Software*, vol. 83, no. 7, pp. 1094–1107, 2010. [Online]. Available: https://doi.org/10.1016/j.jss.2010.02.017

[16] F. Roos-Frantz, D. Benavides, A. Ruiz-Cortés, A. Heuer, and K. Lauenroth, "Quality-aware analysis in product line engineering with the orthogonal variability model," *Software Quality Journal*, vol. 20, no. 3-4, pp. 519–565, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1007/s11219-011-9156-5

[17] J. Hu and Q. Wang, "Extensions and evolution analysis method for software feature models," *JS*, vol. 27, no. 5, pp. 1212–1229, may 2016.

[18] J. Mauro, M. Nieke, C. Seidl, and I. C. Yu, "Context-aware reconfiguration in evolving software product lines," *Science of Computer Programming*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642318301692

[19] A. Felfernig, M. Schubert, and C. Zehentner, "An efficient diagnosis algorithm for inconsistent constraint sets," *Artif. Intell. Eng. Des. Anal. Manuf.*, vol. 26, no. 1, pp. 53–62, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1017/S0890060411000011

[20] A. Felfernig, R. Walter, J. A. Galindo, D. Benavides, S. P. Erdeniz, M. Atas, and S. Reiterer, "Anytime diagnosis for reconfiguration," *Journal of Intelligent Information Systems*, Jan 2018. [Online]. Available: https://doi.org/10.1007/s10844-017-0492-1

[21] J. A. Galindo, M. Acher, J. M. Tirado, C. Vidal, B. Baudry, and D. Benavides, "Exploiting the enumeration of all feature model configurations: A new perspective with distributed computing," in *Proceedings of the 20th International Systems and Software Product Line Conference*, ser. SPLC '16. Beijing, China: ACM, 2016, pp. 74–78. [Online]. Available: http://doi.acm.org/10.1145/2934466.2934478

[22] M. Mendonca, M. Branco, and 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*, ser. OOPSLA '09. New York, NY, USA: ACM, 2009, pp. 761–762. [Online]. Available: http://doi.acm.org/10.1145/1639950.1640002

[23] S. Segura, J. A. Galindo, D. Benavides, J. A. Parejo, and A. Ruiz-Cortés, "Betty: Benchmarking and testing on the automated analysis of feature models," in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, ser. VaMoS '12. New York, NY, USA: ACM, 2012, pp. 63–71. [Online]. Available: http://doi.acm.org/10.1145/2110147.2110155

[24] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2014.

[25] R. Reiter, "A theory of diagnosis from first principles," *AI Journal*, vol. 23, no. 1, pp. 57—95, 1987.