Comprehensibility of Variability in Model Fragments for Product Configuration

Jorge Echeverría[1], Francisca Pérez[1], Carlos Cetina[1], and Óscar Pastor[2]

[1] SVIT Research Group
Universidad San Jorge
Autovía A-23 Zaragoza-Huesca Km.299, 50830, Zaragoza, Spain
{jecheverria, mfperez, ccetina}@usj.es
[2] Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera, s/n, 46022 Valencia, Spain
opastor@pros.upv.es

Abstract. The ability to manage variability in software has become crucial to overcome the complexity and variety of systems. To this end, a comprehensible representation of variability is important. Nevertheless, in previous works, difficulties have been detected to understand variability in an industrial environment. Specifically, domain experts had difficulty understanding variability in model fragments to produce the software for their products. Hence, the aim of this paper is to further investigate these difficulties by conducting an experiment in which participants deal with variability in order to achieve their desired product configurations. Our results show new insights into product configuration which suggest next steps to improve general variability modeling approaches, and therefore promoting the adoption of these approaches in industry.

Keywords: Variability modeling, Software Product Line Engineering, Model Comprehension, Product configuration

1 Introduction

Since software artifacts have become essential and more complex, a fundamental challenge in almost any business is how to manage their software variability to optimize the development process (i.e., improve code reuse). Variability is extensively studied in the field of Software Product Line Engineering [3,16] to support the development and maintenance of families of software products.

General variability modeling approaches (that are independent of the language in which the software products are specified) include Feature-Oriented Domain Analysis (FODA) [9], Orthogonal Variability Model (OVM) [16] and Common Variability Language (CVL) [5]. FODA is widely used for variability management by describing characteristics (features). OVM is a language and a methodology for superimposing variability over any software development artifact without interfering into its contents. CVL is recommended for adoption

as a standard by the Architectural Board of the Object Management Group to manage variability. These approaches can manage variability of product models by describing features in terms of model fragments.

In previous work [7], we detected that the domain experts of our industrial partner (BSH group) had difficulty understanding variability in model fragments to produce the software for the myriad of induction hob models that they produce (under the Bosch and Siemens brands among others). Since the ability to manage variability in models using a comprehensible representation is crucial [19], we further seek to investigate difficulties in understanding the variability of model fragments. Hence, the aim of this paper is to examine difficulties in comprehending variability in model fragments for product configuration.

To do this, we conducted an experiment in which variations points were expressed in models of a Domain-Specific Language (DSL). Using these models, the participants had to perform model fragment substitutions in order to reach a target product configuration.

Our results show four main findings that are relevant for general variability modeling approaches (FODA, OVM and CVL). First, our previous research [7] considered that using the concrete syntax of model fragments as configuration criterion was a source of incorrect solutions, but this experiment reveals that it is possible to harness this criterion to significantly reduce the time required to configure products. Second, the participants intuitively combine model fragments before performing product configuration. This operation turns out to improve the efficiency of product configurations. However, general variability modeling approaches do not support these fragment combinations. Third, the participants obtained the worst results in those fragments substitutions that involve recursive model fragments. Nevertheless, research on product configuration have neglected recursive model fragments. Finally, participants perform redundant configuration steps because they believe that they must perform model fragment substitutions for each variation point even though it already holds the elements of target configuration. Available training materials of general variability approaches lack instructions to avoid these redundant configuration steps. These findings provide insights into product configuration which suggest next steps to improve general variability modeling approaches.

The remainder of the paper is structured as follows: Section 2 provides the required background on product model configuration. Section 3 presents the experiment design and procedure. Section 4 presents the analysis procedure and results. Section 5 presents a discussion of the results. Section 6 describes the threats to validity. Section 7 reviews the related work, and Section 8 concludes the paper.

2 Background on Product Model Configuration

This section illustrates the relationship between the variability specification and model fragments. In addition, this section shows examples of model fragment substitutions to reach a desired product model configuration.

The upper half of Fig. 1 shows a hierarchy of features that represent the variability specification of a induction hob product using the three different general modeling approaches (FODA, OVM and CVL). These approaches follow the idea of superimposed variants [6] in which each feature is related to a model fragment that is expressed in the same terms of the product instances. The lower half of Fig. 1 shows a model fragment for each feature, which is expressed using the Induction Hob DSL. The superimposition of a feature in a model fragment could represent a variation point (see the *Hotplate* feature of Fig. 1), or an available option to set the configuration of a variation point (see the *Triple Inductor* and *Quad Inductor* features of Fig. 1 in which one of them can be selected to configure the *Hotplate* feature).

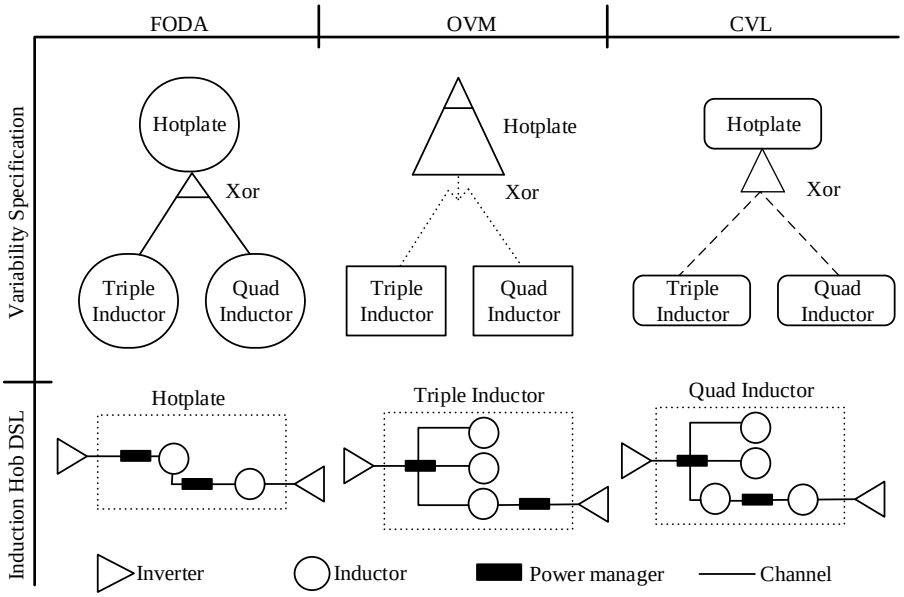

Fig. 1. Relationship between the variability specification and model fragments

After the superimposition of features in model fragments, these model fragments may not share any element (**isolated fragments**) as the upper-left part of Fig. 2 shows. By contrast, it is also possible that after the superimposition some elements are shared (**crossing fragments**) as the upper-middle part of Fig. 2 shows. Furthermore, all the elements of a model fragment can be shared with another model fragment (**recursive fragments**) as the upper-right part of Fig. 2 depicts.

In order to reach a target product configuration (e.g., the middle part of Fig. 2 shows a target induction hob configuration), model fragment substitutions are performed. A model fragment substitution is an operation that may substitute

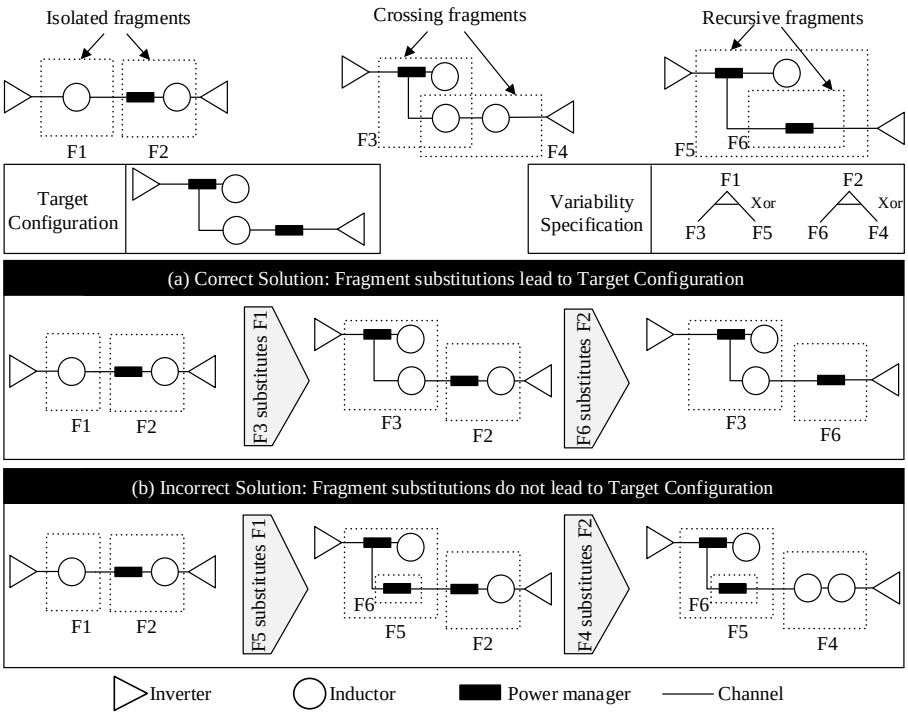

Fig. 2. Examples of correct and incorrect product configurations

any model fragment (a set of arbitrary model elements and the references among them) with any other model fragment described in the same DSL according to the variability specification. For example, the variability specification that is shown in the middle part of Fig. 2 indicates that the model fragment F1 can be substituted with the content of either model fragment F3 or F5.

The lower part of Fig. 2 shows two examples of model fragment substitutions. On the one hand, Fig. 2 (a) shows an example that reaches a **correct solution** since the product model obtained after the fragment substitutions corresponds with the target product configuration. This correct solution encompasses two model fragment substitutions as follows: first the F3 model fragment substitutes the content of the F1 model fragment, and second the F6 model fragment substitutes the content of the F2 model fragment. On the other hand, the example that is shown in Fig. 2 (b) illustrates an **incorrect solution** because the target product configuration is not reached after the fragments substitutions.

A video showing product configuration of induction hobs by means of model fragments substitutions in an industrial environment can be found at: `http://svit.usj.es/variabilitytool.htm`

3 Experiment Design and Procedure

3.1 Research Goal and Questions

The main goal of this research is to determine whether there are difficulties in comprehension of variability in model fragments for product configuration, and whether there are differences in comprehension of isolated, crossing, and recursive model fragments for product configuration. In relation to the above goal, we seek to answer the following research questions:

RQ1: Are there difficulties in comprehending variability in model fragments for product configuration?
RQ2: Are there differences in comprehension of isolated model fragments, crossing model fragments, and recursive model fragments for product configuration?

To our knowledge, there are no cognitive theories about variability in model fragments for product configuration. For this reason, we have not created a hypothesis related to this research question [13].

To answer the above research questions, we used a experimental design. The main dependent variables in our research design were comprehension score (as the percentage of correct product configurations), time spent, and the self-rated difficulty. The independent variable was the type of model fragments.

3.2 Procedure

The participants were asked to fill out a prequestionnaire and to sign a consent form to process the given data. The prequestionnaire obtained general information about the participants and their background, including age, gender, degree and subject of studies, background as software developers, background in software engineering, knowledge about software modeling, knowledge about variability specifications, and knowledge about DSLs.

After filling out the prequestionnaire, an instructor (using slides) explained the Induction Hob DSL used in the exercises. The participants got hard copies of the slides, which the participants could consult while answering the questions.

During the experiment, the participants were provided with exercises to perform product configurations. Each exercise has a target model configuration, a DSL model, and a set of model fragments. To do each exercise, the participants were asked to configure the DSL model by means of fragment substitutions. After the product configuration the resulting model must match the target model of the exercise in order to achieve the correct solution as the example shown in Fig. 2.

The exercises were divided into three groups based on the type of fragments produced by the superimposition of the features on the DSL model. First, isolated exercises denote exercises where the superimposition of the features produced isolated models fragments. Second, crossing exercises denote exercises where the superimposition of the features produced crossing models fragments. Finally, recursive exercises denote exercises where the superimposition of the features produced recursive models fragments. For each target configuration, there

was an isolated exercise and a crossing exercise and a recursive exercise to reach that target configuration. Specifically, the experiment included three target configurations and three exercises for each configuration, which makes a total of nine exercises.

No rigid time constraints were imposed on the participants and the time spent on each question was saved. Once the participants had performed an exercise, they could not modify their response. Then, participants had to answer a subjective question about the perceived difficulty for every exercise, before beginning the next one. The answers ranged from 1=very easy to 7=very difficult [11]. Finally, the participants answered an open question in order to obtain their opinion about the exercise. After the exercises, a focus group interview [12] was performed. In this interview, the participants expressed their opinions about the solutions to the exercises.

The materials used in this experiment (the consent to process the data, the prequestionnarie, the training material, the exercises, the open questions and the notes of the focus group interview) are available at `http://svit.usj.es/productconfigurationexperiment`.

3.3 Participants

The participants were undergraduate students from Computer Engineering at San Jorge University of Zaragoza (Spain) and practitioners from Software Development Companies of Zaragoza. There were 20 students and 12 practitioners. Of the total, 27 were male (84%) and 5 were female (16%). The mean age was 24.2 years.

In the pre-questionnaire, all students stated that they had never worked as software developers or as software engineers. On the other hand, all practitioners stated that they had knowledge about software modeling, knowledge about feature diagrams, and familiarity with domain-specific languages. Furthermore, all the practitioners had developed software (with a mean of 6.5 years).

4 Analysis procedure and results

To analyze the results, we used the dependent variables: the comprehension score (as the percentage of exercises performed correctly), the time needed to perform the exercises and the participants' perceived difficulty. According to a Shapiro-Wilk test [22], the comprehension score, and time to perform the exercise follow a normal distribution.

Prior to our analysis, we checked whether the control variable 'type of participant', i.e. student or practitioner, had an influence on comprehension score and on time to perform the tasks. Since 'type of participant' did not have an influence on the comprehension score and the on time results, we decided to drop it from the final statistical tests we report.

(a) Comprehension score. An ANOVA test [4] was conducted to compare the comprehension score for the different model fragments. There was a signifi-

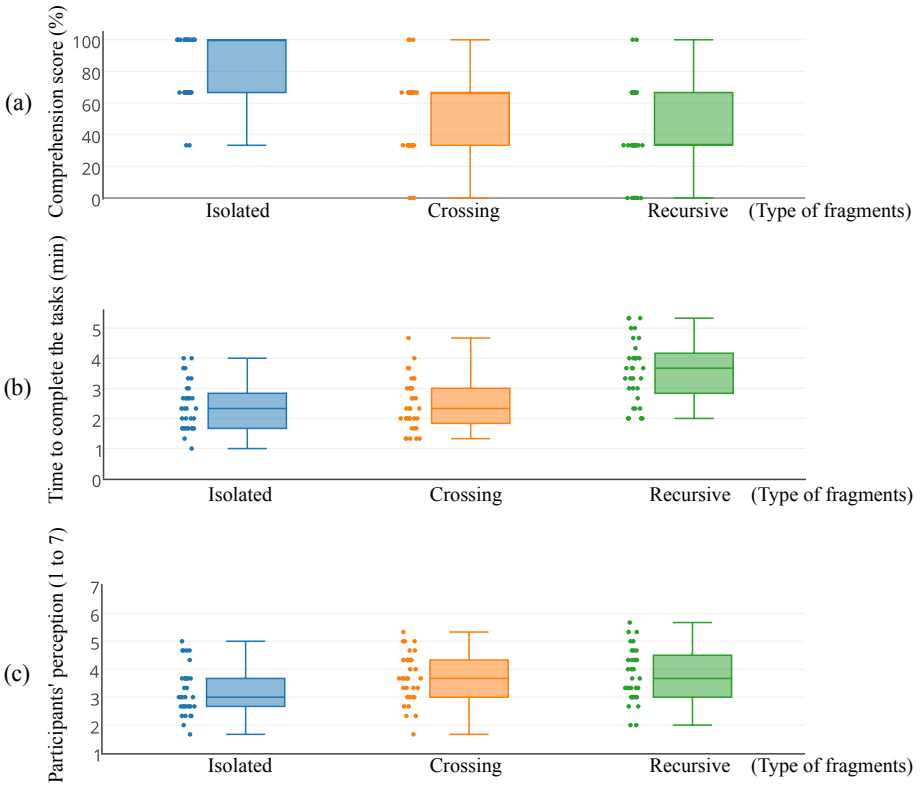

Fig. 3. (a) Comprehension score. (b) Time spent to complete the tasks. (c) Participants' perceived difficulty.

cant effect of the model fragments on the comprehension score at $p<0.05$ for isolated fragments, crossing fragments, and recursive fragments [$F(2,93) = 22.729$, $p = 0.000$]. Fig. 3 (a) shows that the exercises related to isolated fragments were the easiest to comprehend and correctly answer (82.29%), followed by exercises related to crossing fragments (51.04%) and recursive fragments (40.62%). We performed t-tests to find out differences between model fragment types in terms of comprehensibility. The differences between the comprehension score for isolated fragments and the scores for crossing and recursive fragments were significant. However, there were no significant differences between the comprehension score for crossing fragments and recursive fragments.

(b) Time spent to complete the tasks. We analyzed the time needed to complete the exercises with an ANOVA test. The results showed that there was a significant effect of the different model fragments on the time needed to perform the exercises [$F(2,93) = 17.512$, $p = 0.000$]. Fig. 3 (b) shows that the participants needed more time to perform the exercises related to recursive fragments (a mean of 3.55 minutes); the exercises related to isolated fragments

and crossing fragments were performed by the participants in the same amount of time (a mean of 2.41 minutes). We performed t-tests to find out which model fragment types significantly differed from each other in terms of time. The t-test showed that there were no significant differences between the isolated fragments and the crossing fragments. In contrast, the differences between the recursive fragment exercises and the others exercises were significant.

(c) Participants' perceived difficulty. An ANOVA test was conducted to analyze the participants' perceived difficulty for different model fragments. There was a significant effect of the model fragments on the perceived difficulty at $p<0.05$ for isolated fragments, crossing fragments and recursive fragments [$F(2,93) = 3.561$, $p = 0.032$]. Fig. 3 (c) shows that the exercises with recursive fragments (3.79) and crossing fragments (3.70) were more difficult to understand than the exercises with isolated fragments (3.23). We performed t-tests to find out the users' perception of which model fragments significantly differed from each other. The differences between the users' perception for isolated fragments and both crossing and recursive fragments were significant.

The analysis of the data enables to answer the Research Questions as follows. *RQ1: Are there difficulties in comprehending variability in model fragments for product configuration?* As shown by Fig 3 (a), not all product configuration exercises were correctly solved. Specifically, 42.01% of the exercises failed to perform a product configuration that matches the target configuration. *RQ2: Are there differences in comprehension of isolated model fragments, crossing model fragments, and recursive model fragments for product configuration?* Only exercises that include isolated model fragments for product configuration were comprehended by the majority (82.29%). By contrast, exercises that include crossing model fragments or recursive model fragments obtained the worst results (51.04% and 40.62% respectively). Next section discusses these results and highlights our findings.

5 Discussion

Some participants (6.25%) reported that the geometric shape of the model fragments seemed to be significantly important for product configuration. Specifically, these participants considered a model fragment as valid option whether its geometric shape fits with the geometric shape of a variation point. This is especially important because these participants gave to the geometric shape more priority than to the variability specification, which became shape criterion in a source of errors. Shape criterion produced incorrect configurations in the 27.5% of the exercises in which it has been used. The concrete syntax of model fragments misleading product configurations matches with our previous research [7].

Although this geometric shape criterion was a source of errors, we also measured in this experiment the time spent in the exercises. We detected that the exercises, which both applied the geometric shape criterion and obtained a correct solution (72.5%), reduced the time spent in the product configuration a 38%. It turns out that we previously considered the geometric shape criterion as

a source of incorrect solutions only, but this experiment reveals that this criterion could also significantly reduce the time required to configure products. These results suggest general variability modeling approaches should devise a mechanism to align geometric shape and variability specification in order to take advantage of the time improvement.

The participants also mentioned that they intuitively combined model fragments (creating a new model fragment) in order to inject the combination in a variation point. The participants said that these combinations of model fragments helped them to solve easier the exercises. The results reveal that the participants, who combined model fragments (34.4% of participants), reduced both the number of fragment substitutions (an average of 33.18%) to reach the target product configuration, and the number of errors (an average of 14%) with regard to the participants who did not combined model fragments. Nevertheless, this operation is not supported in general variability modeling approaches (FODA, OVM and CVL). As our results reveal that this operation for combining model fragments is a positive complement for product configuration, new versions of general variability modeling approaches should consider the inclusion of this operation.

The results also show that participants obtained the highest percentage of correct solutions in the isolated model fragments (the mean of correct solutions is 82.29%). This result matches with previous research works [18], which used isolated model fragments to perform product configurations. Crossing model fragments obtained a mean of 51.04% of the correct solutions. The scientific community is already aware that crossing model fragments are difficult to understand [24,15,23], which is also confirmed in this experiment. Recursive model fragments obtained the worst results (a mean of 40.62% correct solutions). Overall, the participants got a incorrect solution when they deepened the levels of recursion to reach the target configuration. To the best of our knowledge, recursive model fragments are neglected by the scientific community. The results of this experiment suggest that recursive model fragments are the major source of incorrect solutions to reach a target product configuration. Therefore, new experiments need to be performed in order to further investigate recursive model fragments in product configurations.

We found that participants made redundant fragment substitutions because they thought that variation points have to be always substituted with one of its available options. They considered this substitution mandatory even though the variation point already holds the model elements of the target product configuration. According to the participants, the tree layout of the variability specification (e.g., the variability specification that the upper half of Fig. 1 shows) reinforces the participants' idea that they should choose an option for each variation point. The results show that the 5.9% of exercises included redundant fragment substitutions, which increase in a 6.93% the necessary fragment substitutions to reach the target product configuration. It is important to highlight that we did not find instructions to avoid this redundancy in the materials [16,5,9] of the general variability modeling approaches, which were used in this experiment to train

the participants. Hence, the results of this experiment suggest that the training materials of the variability modeling approaches should be extended to explicitly avoid this redundancy.

Table 1 summarizes the main findings of this work that are relevant for general variability modeling approaches. Each finding is tagged as type Confirms (the finding confirms results of previous research works) or type New (new finding revealed by this work). Finally, the table also summarizes the next steps for variability modeling approaches that we suggest taking into account the findings.

Table 1. Summary of findings

#	Finding	Type	Next Step
1	Concrete syntax of model fragments misleads product configuration.	Confirms [7]	New versions of concrete syntax of model fragments should align geometric shape and variability specification in order to take advantage of the time improvement.
2	Geometric shape criterion reduces the time required (38% less time) to perform product configurations.	New	
3	Model fragment combination improves the efficiency (33.18% less steps) of product configurations and reduce the incorrect solutions (14%).	New	General variability modeling approaches should consider the inclusion of the combination operation as a complement for product configuration.
4	Highest percentage of correct solutions in the isolated model fragments (82.29% of correct solutions).	Confirms [18]	Specific experiments need to be conducted to further investigate recursive model fragments in product configurations.
5	Crossing model fragments are difficult to understand (51.04% of correct solutions).	Confirms [24,15,23]	
6	Recursive model fragments are the major source of incorrect solutions (40.62% of correct solutions).	New	
7	Redundant configuration steps are produced by misunderstanding variation points as mandatory substitutions.	New	Training materials of the variability modeling approaches should be extended to explicitly avoid this redundancy.

6 Threats to Validity

We use the classification of threats to validity of [21,25], which distinguishes four aspects of validity to acknowledge the limitations of our experiment.

Construct validity: This aspect of validity reflects the extent to which the operational measures that are studied represent what the researchers have in mind and what is investigated based on the research questions. In this work, the proposed exercises do not have a true/false answer. Therefore, it is very difficult for users to answers correctly if they do not understand the question.

Furthermore, to minimize this threat exercises and the responses were designed by two variability modeling experts. These experts have developed industrial variability modeling tools (in the induction hob domain and train control

software domain). Their participation was limited to the design of the exercises and they were not involved in this paper.

Finally, the measures used in our research are the percentage of correct solutions, the time spent, and a self-rated difficulty. These measures are widely accepted in the software engineering research community [18,19].

Internal validity: This aspect of validity is of concern when causal relations are examined. There is a risk that the factor being investigated may be affected by other neglected factors. In this work, we explained the DSL to the participants. The slides used in that explanation were given to the participants, so that lack of comprehension of DSL would not be a problem in performing the exercises.

External validity: This aspect of validity is concerned with to what extent it is possible to generalize the finding, and to what extent the findings are of relevance for other cases. The experiment was performed by students and practitioners, and this students' participation can be a source of weakness. However, using students as subjects instead of software engineers is not a major issue [20,10] as long as the research questions are not specifically focused on experts. We considered that the variability modeling concepts under study are also relevant to students.

In the analysis procedure we have used a confidence interval $p<0.05$ where conclusions are 95% representative. This means that if they followed a normal distribution, the results would be true 95% of the times.

Since the DSL used in this study is a very simple language in a specific domain, we think that the generalizability of findings should be undertaken with caution. The selected DSL is appropriate for easy comprehension by the participants. However, other experiments with other DSLs should be performed to validate our findings.

Reliability: This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. To reduce this threat, two variability modeling experts who were not involved in the research designed both the exercises and the correct answers. We also performed this research using methods that are widely accepted by the software engineering community.

7 Related Work

There are research studies in the literature that analyze difficulties in model fragments for product configuration. In [24], Vasilevskiy and Haugen identify the problems generated after the superimposition of features in crossing model fragments. In [15], Oldevik et al. analyze confluence and conflict properties of multiple variation points. In [23], Svendsen et al. identify difficulties in crossing model fragments when the DSL models are modified. In our work, we have not only found that crossing model fragments are difficult to understand (which confirms the finding of the previous works) but also we analyzed isolated and recursive model fragments, the necessary time to reach a product configuration and the participants' perceived difficulty. This enables us to obtain new findings

about the geometric shape of model fragments, combinations of model fragments, the major source of incorrect solutions, and redundant configuration steps.

There are research efforts in the literature regarding the comprehension of variability modeling. In [19], Reinhartz-Berger et al. present a study for comprehending feature models by performing an experiment with participants who are familiar with feature modeling and participants who are not familiar with it. Furthermore, Reinhartz-Berger and Figl [18] present an approach that investigates the comprehensibility of orthogonal variability modeling languages. Specifically, they conducted an experiment to examine potential comprehension problems in two orthogonal variability modeling languages: CVL and OVM. In [20], Reinhartz-Berger and Tsoury present a comparative analysis for managing variability using Feature-oriented and UML-based modeling methods. Nevertheless, these works do not analyze the differences in comprehensibility using isolated, crossing and recursive model fragments to reach a desired product configuration as our work does.

Medeiros et al. [14] perform interviews with developers about the use of the C preprocessor to handle variability since developers use conditional directives to provide optional features or to select between alternative implementations. Berger et al. [1] present an exploratory case study of three companies that apply variability modeling to research about practices, characteristics, benefits and challenges of variability modeling. Even though these works provide empirical data on variability management in industrial application, they do not investigate difficulties in understanding product configuration when features are superimposed to a realization model.

There are also works that facilitate the comprehension of variability modeling by end users. For instance, Grünbacher et al. present a Configurable Product Line tool that enable customization by end users [8]. The authors abstract the technical issues of this customization to help end users understand the implications of the decisions that they make. Furthermore, Rabiser et al. [17] present an end-user oriented tool that can support diverse end-users such as project managers, salespeople, or engineers. Botterweck et al. present a metamodel and a tool that employs visualization techniques to support users in the process of product configuration [2]. These works focus on augmenting the capabilities of variability modeling tools to improve the feedback that the tools provide to their users. By contrast, our work does not address variability tool support an we focus on general variability modeling languages.

In [7], we performed a usability evaluation of a variability modeling tool in which we detected that the domain experts of our industrial partner (BSH group) had difficulty understanding variability due to the concrete syntax of model fragments. The work that this paper presents confirms that the concrete syntax of model fragments misleads participants during product configurations but this paper reveals that it is possible to harness concrete syntax to significantly reduce the time required to configure products. This paper also obtains new findings about combinations of model fragments, the major source of incorrect solutions, and redundant configuration steps.

8 Conclusions

In previous work [7], we detected that the domain experts of BSH group had difficulty understanding variability to configure the firmware of their products (induction hobs under the Bosch and Siemens brands). Specifically, domain experts had difficulty understanding variability in the model fragments that results of superimposing features [6], of the variability specification, on a DSL model.

In this work, we conducted an experiment in which participants deal with variability in model fragments to achieve their desired product configurations. The exercises were divided into three groups based on the type of fragments produced by the superimposition of the features on the DSL model: isolated model fragments, crossing model fragments and recursive model fragments. We measured comprehension score, time spent, and the self-rated difficulty. In addition, we obtained participants' opinion about the exercises by means of open questions and a focus group interview.

Our results show findings that are relevant for general variability modeling approaches (FODA, OVM and CVL). Specifically, results show four new findings revealed by this work:

- Geometric shape criterion reduces the time required to perform product configurations.
- Model fragment combination improves the efficiency of product configurations and reduce the incorrect solutions.
- Recursive model fragments are the major source of incorrect solutions.
- Redundant configuration steps are produced by misunderstanding variation points as mandatory substitutions.

And three findings that confirm results of previous research works:

- Concrete syntax of model fragments misleads product configuration.
- Highest percentage of correct solutions in the isolated model fragments.
- Crossing model fragments are difficult to understand.

Taking into account the above findings, we suggested next steps for variability modeling approaches that cover (1) new concrete syntax of model fragments, (2) the inclusion of the model fragment combination operation, (3) further investigation of recursive model fragments in product configurations and (4) clarification of training materials of the variability modeling approaches. These next steps would contribute to promote the adoption of variability management approaches in industry.

Acknowledgments

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO), through the Spanish National R+D+i Plan and ERDF funds under The project Model-Driven Variability Extraction for Software Product Lines Adoption (TIN2015-64397-R).

References

1. T. Berger, D. Nair, R. Rublack, J. M. Atlee, K. Czarnecki, and A. Wąsowski. Three cases of feature-based variability modeling in industry. In *ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2014.
2. G. Botterweck, S. Thiel, D. Nestor, S. bin Abid, and C. Cawley. Visual tool support for configuring and understanding software product lines. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 77–86, Sept 2008.
3. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
4. N. Condori-Fernández, J. I. Panach, A. I. Baars, T. E. J. Vos, and O. Pastor. An empirical approach for evaluating the usability of model-driven tools. *Sci. Comput. Program.*, 78(11):2245–2258, 2013.
5. CVL Submission Team. Common variability language (CVL), OMG revised submission. `http://www.omgwiki.org/variability/lib/exe/fetch.php?id=start&cache=cache&media=cvl-revised-submission.pdf`, 2012.
6. K. Czarnecki and M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In *Generative Programming and Component Engineering*, volume 3676 of *Lecture Notes in Computer Science*, pages 422–437. Springer Berlin Heidelberg, 2005.
7. J. Echeverría, J. Font, C. Cetina, and O. Pastor. Usability evaluation of variability modeling by means of common variability language. In *Proceedings of the CAiSE 2015 Forum at the 27th International Conference on Advanced Information Systems Engineering co-located with 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015), Stockholm, Sweden, June 10th, 2015.*, pages 105–112, 2015.
8. P. Grünbacher, R. Rabiser, and D. Dhungana. Product line tools are product lines too: Lessons learned from developing a tool suite. In *Automated Software Engineering, 2008. 23rd IEEE/ACM International Conference on*, pages 351–354, Sep 2008.
9. K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute, November 1990.
10. B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Softw. Eng.*, 28(8):721–734, Aug. 2002.
11. J. A. Krosnick and S. Presser. Question and questionnaire design. *Handbook of survey research*, 2:263–314, 2010.
12. R. A. Krueger and M. A. Casey. Designing and conducting focus group interviews. *Social Analysis, Selected Tools and Techniques, Krueger, RA, MA Casey, J. Donner, S. Kirsch and JN Maack*, pages 4–23, 2002.
13. S. Kumar and V. Karoli. *Handbook Of Business Research Methods*. Thakur Publishers, 2011.
14. F. Medeiros, C. Kästner, M. Ribeiro, S. Nadi, and R. Gheyi. The love/hate relationship with the C preprocessor: An interview study (artifact). *DARTS*, 1(1):07:1–07:32, 2015.
15. J. Oldevik, Ø. Haugen, and B. Møller-Pedersen. Confluence in domain-independent product line transformations. In M. Chechik and M. Wirsing, editors, *FASE*, volume 5503 of *Lecture Notes in Computer Science*, pages 34–48. Springer, 2009.

16. K. Pohl, G. Böckle, and F. J. v. d. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
17. R. Rabiser, D. Dhungana, W. Heider, and P. Grünbacher. Flexibility and end-user support in model-based product line tools. In *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference*, pages 508–511, 2009.
18. I. Reinhartz-Berger and K. Figl. Comprehensibility of orthogonal variability modeling languages: The cases of CVL and OVM. In *Proceedings of the 18th International Software Product Line Conference - Volume 1*, SPLC '14, pages 42–51, New York, NY, USA, 2014. ACM.
19. I. Reinhartz-Berger, K. Figl, and Ø. Haugen. Comprehending feature models expressed in CVL. In *Model-Driven Engineering Languages and Systems*, volume 8767 of *Lecture Notes in Computer Science*, pages 501–517. Springer International Publishing, 2014.
20. I. Reinhartz-Berger and A. Tsoury. Experimenting with the comprehension of feature-oriented and UML-based core assets. In *Enterprise, Business-Process and Information Systems Modeling*, volume 81 of *Lecture Notes in Business Information Processing*, pages 468–482. Springer Berlin Heidelberg, 2011.
21. P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.
22. S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.
23. A. Svendsen, X. Zhang, Ø. Haugen, and B. Møller-Pedersen. Towards evolution of generic variability models. In J. Kienzle, editor, *Models in Software Engineering*, volume 7167 of *Lecture Notes in Computer Science*, pages 53–67. Springer Berlin Heidelberg, 2012.
24. A. Vasilevskiy and Ø. Haugen. Resolution of interfering product fragments in software product line engineering. In *Model-Driven Engineering Languages and Systems*, volume 8767 of *Lecture Notes in Computer Science*, pages 467–483. Springer International Publishing, 2014.
25. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.