Automated Analysis in Feature Modelling and Product Configuration

David Benavides[1], Alexander Felfernig[2], José A. Galindo[1], and Florian Reinfrank[2]

[1] University of Seville
Av. de la Reina Mercedes S/N, 41012 Seville, Spain
{benavides,jagalindo}@us.es
[2] Institute for Software Technology
Graz University of Technology
Inffeldgasse 16b/II
Graz, Austria
{afelfern,freinfra}@tugraz.at

Abstract. The automated analysis of feature models is one of the thriving topics of research in the software product line and variability management communities that has attracted more attention in the last years. A recent literature review reported that more than 30 analysis operations have been identified and different analysis mechanisms have been proposed. Product configuration is a well established research field with more than 30 years of successful applications in different industrial domains. Our hypothesis, that is not really new, is that these two independent areas of research have interesting synergies that have not been fully explored. To try to explore the potential synergies systematically, in this paper we provide a rapid review to bring together these previously disparate streams of work. We define a set of research questions and give a preliminary answer to some of them. We conclude that there are many research opportunities in the synergy of these independent areas.

Keywords: Software Product Lines, Feature Models, Product Configuration, Rapid Review, Knowledge-based Systems

1 Introduction

Variability modelling and management is a key issue in software product line engineering. Feature models are one of the most used mechanisms to model the variability within a software product line. A feature model consists of a set of features and a set of relationships that connect features. It is arranged in a tree–like structure with additional cross-tree constraints. There are different feature model dialects identified in the literature [53] which include basic feature models, cardinality based feature models and extended feature models using feature attributes.

Figure 1 shows an example feature model using the most well known modelling elements[3]. The model illustrates how features are connected to specify a software product line in the *mobile phone* domain. We assume that the software loaded in the phone is determined by the features that it supports. According to the model, all phones must include features supporting *calls*, and displaying information in either *basic*, *colour* or *high resolution* screens. Furthermore, it is possible to optionally include support for *GPS* and multimedia elements such as *camera*, *MP3* player or both of them.

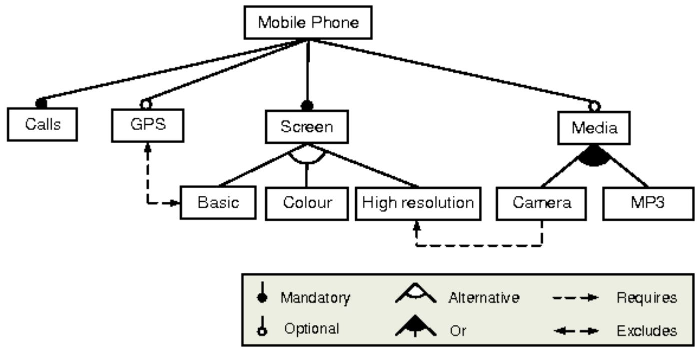

Fig. 1. A sample feature model

The automated analysis of feature models is one of the areas of research that have attracted more attention in the last two decades [8]. It can be defined as the computer–aided extraction of information from feature models. The analysis is performed by means of *analysis operations* which take several inputs and provide an output. As input we have a feature model with optionally some additional information such as a set of features to be selected or deselected. As output it is possible to find numbers, set of features and others depending on the kind of analysis operation. An example of a feature model analysis operation would be counting the number of possible products represented by the feature model. In the example of Figure 1 the number of products is 14. 30 different analysis operations have been surveyed [8] including operations for model consistency, error detection, explanations, and feature model configuration capabilities. The general analysis process is shown in Figure 2 where a feature model is translated to a logical representation and using some technique (e.g. logical solvers or specific algorithms) the analysis operations are performed.

The configuration of feature models can be defined as the process of selecting and deselecting features in a feature model until reaching a full configuration, i.e. a configuration where no additional decision on the feature model needs to be made to have all the information to configure a given software product of the software product line. The configuration of feature models is no more than

[3] This Figure has been taken from [8].

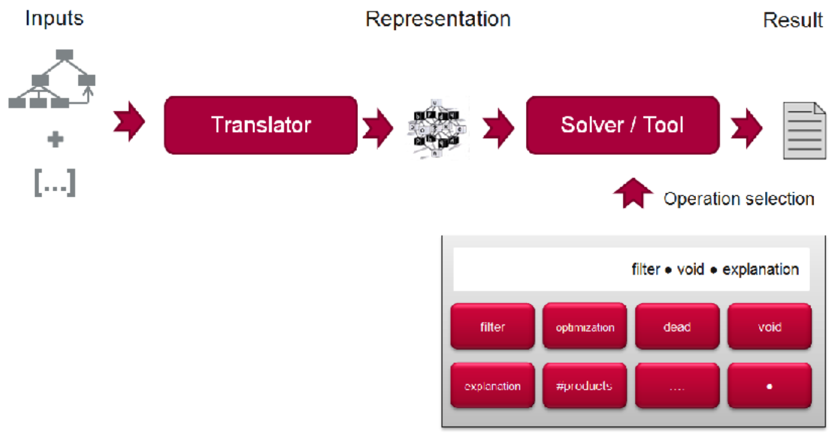

Fig. 2. Process for the automated analysis of feature models taken from [8]

an analysis operation where the input is a feature model with a set of decisions on the state of a given set of features (a feature can be selected, deselected or undecided) and the output is the feature model together with the new states of the features.

Product configuration is an independent area of research from software product line engineering that has a long history as an application of Artificial Intelligence technologies [43,23,52]. The first paper on product configuration was published back in 1978 [36]. Similar to feature model based configuration, product configuration can be interpreted as the process of partially or completely instantiating component types and related attributes with concrete components and attribute values [52] in a way that preserves the consistency with a predefined set of constraints (restrictions). Configuration technologies are typically applied in complex product domains such as telecommunication [23], automotive [34], and digital equipment [7,19].

Although product configuration is a well established area of research with numerous industrial applications, the synergies between feature model configuration and product configuration have been rarely explored. Our hypothesis, that can be easily formulated from the previous descriptions, is that feature model and product configuration have a lot of potential synergies that can be explored and exploited. In this paper, we show first steps to explore those synergies towards a more systematic literature review to fully gather the spread knowledge from the different areas to start a cross fertilization process to benefit both communities from the independent results.

This hypothesis of existing synergy potentials is not really new. In the past, there have been already some attempts to connect these two areas [6,39] and the importance of such a connection has been explored in the last years within the software product line community. As an example, there have been 2 invited keynotes in recent workshops of the software product line conference by well known researchers from the configuration area (see [16,47]). Also, a recent contribution to a workshop in the product configuration area proposes a research roadmap to try to connect these two areas and revealed the importance of surveying the literature to find synergies [31]. In this paper, in difference with respect to previous work, we give a first step forward to complete a systematic literature review to bring together these previously disparate streams of work and we provide first answers to some research questions. Thus, we define a set of research questions and give a preliminary answer to some of them which start opening research opportunities.

Although in a systematic literature review a well established method is required [10], we do not present such a systematic method in this paper. We rather use a rapid review approach which is also a very common method in evidence-based research in areas such as medicine [27]. A rapid review is a method to provide an assessment of what is already known in a given research field. In contrast to literature reviews, it does not need a tedious and time–consuming method trying to be "quick but not dirty" [27]. In this sense, it can serve as a first step towards a more systematic literature review. It is fair to recognize that this rapid literature review has an important bias due to the fact that there is a good amount of the surveyed references that are works done by the authors. However, we still think it is valuable to show these results since the authors have been working independently in the surveyed areas namely, automated analysis of feature models and product configuration. In any case, this bias is also addressed adding a good amount of references to other existing work.

In the following section we discuss research questions related to the further development of both research fields. Thereafter – in Section 3 – we try to provide first answers to the posed questions.

2 Research questions

The goal of this review is to provide first answers to the following research questions (RQ 1–4). Some of them have been already answered in a recent literature review about the automated analysis of feature models [8]. The main goal here is to investigate how these questions have been addressed in the product configuration field and see the similarities, differences and discover research opportunities. We will try to answer these questions always comparing how different activities are performed in the feature modelling field and how they are addressed in the product configuration field. Although there are also other potential research questions to be addressed we selected these 4 mainly because they cover important parts of the engineering process such as modelling (RQ1), implementation and design (RQ2–3), quality assurance (RQ4).

RQ1: How are the different modelling approaches related? There are different dialects of feature models as described in [53]. In contrast, how are configuration problems modelled? Can a feature model configuration problem be represented as a configuration problem? Are there modelling elements in product configuration that are not used in feature models? And the opposite? Are there approaches to standardize configuration knowledge representations and how can these representations be exploited in the context of feature model development?

RQ2: Which are the automated mechanisms proposed? There are mainly three basic reasoning approaches used when automatically analysing feature models [8]: propositional logic based analysis, constraint programming based analysis and description logic based analysis. Are those paradigms also used in product configuration? Are there any special techniques developed in that field that could be used in feature model configuration?

RQ3: Are there similar operations? In feature models, 30 different analysis operations have been recently reported [8]. How similar are the operations in configuration problems? In product configuration it is well known that one of the main important tasks is the user support which includes providing explanations when a erroneous configuration step is reached. Are there special mechanisms developed in the product configuration community that could be used in feature model configurations?

RQ4: Which are the functional and performance mechanisms used? In feature models, there are some proposals to perform functional and performance testing of analysis tools [56,54,55,57]. The challenge is how to assess the quality of feature model analysis tools in terms of functionality (is the analysis tool doing what is supposed to do?) and performance (is the analysis tool performing well?). Are there also functional and performance testing mechanisms described in the product configuration literature? How are the different mechanisms related?

3 Preliminary results

To provide a preliminary answer to the research questions of Section 2 we searched papers in academic databases guided from our previous experience in the field. In this section we give a first answer to some of the research questions by quoting and explaining some of the papers studied to show the potential synergy between the two areas.

3.1 RQ1: How are the different modelling approaches related?

From our rapid review we have detected that the existing research on product configuration does not have a well established or standard language to define configuration problems. There have been attempts to use domain-specific languages for product configuration, for example, on the basis of the Unified Modeling Language (UML) [15]. Furthermore, ontology based configuration knowledge representations [60] and description logics based representations have been developed [18,40]. These representations are either not supporting the needed expressivity (for an in-depth analysis of description logic based knowledge representations see [18]) or are not based on a formal semantics (UML is based on a semi-formal definition, the same holds for the ontology specified in [60]). In other cases, configuration problems are formally defined on the basis of logic-based approaches which are often not accessible for domain experts and even developers. In this sense, it is easy to find configuration problems described in description logic, constraints or propositional logic (see Section 3.2) but there is still a need for a standardized representation with a clear underlying formal semantics. There have been some general standardization efforts in constraint representations [45] but not specifically in the product configuration domain. Also, there have been some efforts to clearly define configuration tasks [43]

In contrast, configuration problems in software product lines are mainly modelled using any of the following families of notations: decision-based modelling notations or feature model–based notations [12]. There are also other notations such as OVM [49] or COVAMOF [59] but these are less common in the literature. There is even a current effort to define a common variability modeling language (CVL) [24] which could also serve as a basis for the definition of configuration problems. There exist different dialects of feature models as described in [53] and also some textual syntax of feature models like TVL [11]. In addition, formal semantics of feature model dialects have been reported [13,53].

We will now try to answer the following sub question: *can a feature model configuration problem be represented as a configuration problem?* To do so, we provide the following definitions adapted from the discussions in Section 1 and from any general definition of a configuration problem that can be found in the literature. Note that this definition can be exploited for the representation of basic configuration problems which do not include complex connection structures and component hierarchies [23,38]. However, it is a good basis for having a common representation for both, basic configuration problems and feature model configuration problems.

Definition 1 (Feature Model Configuration Problem). A feature model configuration problem is defined by the tuple (F,D,C) where F = $\{f_1, f_2, ..., f_n\}$ is a set of features f_i. Furthermore, D= {dom(f_1), dom(f_2), ..., dom(f_n)} is the set of corresponding feature domains where dom(f_i) = {true, false}. Finally, C = CR $\cup$ CF is a set of constraints restricting the possible configurations which can be derived from the feature model. In this context, CR = $\{c_1, c_2, ..., c_k\}$ represents a set of user requirements (e.g. selection or deselection of features) and CF = $\{c_{k+1}, c_{k+2}, ..., c_m\}$ represents a set of feature model constraints.

The hypotheses here is that any relationship defined in a feature model dialect can be translated to a constraint in a Constraint Satisfaction Problem (CSP)(see [5] for an introduction on CSP).

Definition 2 (Feature Model Configuration). A feature model configuration for a given feature model configuration problem is a complete assignment of the variables $f_i \in F$. Such a configuration is consistent if the constraints $c_i \in C$ are consistent with the given variable assignment. Furthermore, a feature model configuration is valid, if it is consistent and complete, i.e. it does not violate any constraint defined in the feature configuration problem and all the variables have an assigned value.

Results. We conjecture that a feature model configuration problem in particular and any software product line configuration problem in general could be seen as a special case of a product configuration problem.

In configuration problems not only boolean constraints are used as in most of the cases of feature model configuration problems. In product configuration problems there is no standard language to describe configuration problems while in software product lines there is de facto standard which are feature models and an effort to a standardized notation such as CVL. These more established notations in software product line engineering could inspire product configuration researchers to identify ways to share, disseminate, and model configuration problems. On the other hand there are also challenges with respect to product domains (e.g., telecommunication switches [23]) where complex connection structures and related (aggregation) constraints have to be specified (see, e.g., [18]). We want to emphasize that a detailed analysis of needed extensions of existing feature model representations is a major challenge for future research if feature models want to be adopted as a sort of standard language in the product configuration field.

3.2 RQ2: Which are the automated mechanisms proposed?

There are mainly three categories of mechanisms used for the automated analysis of feature models [8]: propositional logic based analysis, constraint programming based analysis, and description logic based analysis. In a survey on product configuration in 1998[4], Sabin et al. [52] divided the existing paradigms on product configuration as the following: *rule-based reasoning* and *model-based* approaches. In the former, rules of the form *if condition then action* are used to represent configuration knowledge. According to [52], this kind of configuration systems have maintenance problems. In model-based configuration systems, the assumption is that the configuration knowledge is expressed in an explicit language in terms of a *model*. Among the approaches in model–based configuration problems, description logics and constraint–based approaches are presented [52].

Results. Among constraint based approaches there are some based on so–called *conditional constraint satisfaction problems* (CCSP) [26], *dynamic constraint satisfaction problems*(DCSP) [42], and *generative constraint satisfaction problems* (GCSP) [23,38]. There are also some proposals to combine description logics and constraint satisfaction problems [35]. Furthermore, we have found papers in the product configuration literature that use *binary decisions diagrams* (BDD) to represent and solve configuration problems [9,4,28] and also proposals which combine CSPs with BDD techniques to obtain better product configurators [61].

A product configuration problem is *interactive* if the configuration process is performed interactively, i.e. the user makes selections and the system has to provide feedback to the user as soon as possible. In such scenarios the response time of the systems is crucial. It is desirable to guarantee a given response time. In this sense, there is a branch of research on product configuration that deals with the *off-line compilation* of configuration problems for a later *on-line*

[4] It is interesting to note that we have not found any more recent review on product configuration

configuration process. An off–line compilation of a configuration problem is a process where the configuration problem is translated to a given representation that ensures a good response time. In the best case, the compilation will deliver a backtrack-free configurator. Once the compilation is performed, the system can be used for an on-line configuration process. There are several proposals in the literature of product configuration using compilation techniques, some are based on translation the configuration process into a BDD representation [33,29] and others are based on transforming a configuration problem into an automata [14,50].

Although there have been some efforts to use efficient techniques for feature model analysis [41], in general, these techniques have been rarely studied in the feature model analysis literature and there is significant work to be done to include those techniques in the feature model analysis field.

3.3 RQ3: Are there similar analysis operations?

An analysis operations over a feature model, as stated in Section 1, is an operation that takes a feature model as input and returns a result as output. An analysis operation over a configuration model would be the same but taking as input a configuration model. In feature models, 30 different analysis operations have been recently reported [8] and it is common to find more and more papers describing some new operations or a sort of redefinition of them (e.g. [44]).

Results. In the product configuration field, it is not common to find new operations besides the basic ones like propagate a configuration decision, provide feedback to the user in terms of explanations or maintain the consistency of the configuration knowledge base. This could be the case because in software product line models such as feature models, a very important aspect is the connection of the variability model with other software artefacts like code, software components, test cases or UML diagrams among others. It would be necessary to have a catalogue of operations in product configuration similar to the one found in feature model analysis [8] to explore if there are operations in one side or the other that could be used as well as existing techniques to automate them.

A special case of analysis operations are the so-called *explanations*. In the feature model analysis field an explanation is defined as an operation of analysis that not only provides a result but also an explanation of *why* or *why not* a given result is provided [64]. There are some proposals to explain why a feature model is inconsistent, why a feature is *dead* or why a feature is *false optional* [63]. Also, there are some proposals to explain why a given configuration is erroneous with respect to a given feature model [66,65]. Most of these approaches are based on Reiter's theory of diagnosis [51] which means that the method is complete and provides an explanation which is minimal. A minimal explanation is the one that explain a given analysis result with the minimum number of elements. For instance, given a flawed configuration with a set of selected features and a set of deselected features, an explanation could be used to determine the changes to be made in the configuration to repair it [65]. It would be possible to say that all the selections or deselections have to be changed, but usually the interesting information is to know the minimum changes required to repair the flawed configuration.

In product configuration it is well known that the methods based on Reiter's theory of diagnosis are computationally hard to solve. To face this problem, there are several proposals in the product configuration literature to provide faster explanations that either preserve minimality (in terms of the number of needed repair actions) [17,32,48] or focus on the determination of personalized repairs which are also minimal but are not necessarily minimal cardinality repairs [46,20,22].

From this discussion it seems clear that explanation mechanisms for feature model analysis have to be synchronized with respect to existing product configuration mechanisms.

3.4 RQ4: Which are the functional an performance mechanisms used?

Developing and maintaining feature model analysis tools is difficult and costly as any other software system due to its complexity and changeability [56]. As any other software tool, a feature model analysis tool has to use functional testing mechanisms to detect bugs in the development and evolution process. In product configuration, tools for providing configuration capabilities are known as *configurators*. Configurators suffer from the same problems that feature model analysis tools, i.e. they are difficult to develop and maintain and most of the configuration operations are computationally hard to tract.

Results. In the feature model analysis literature we have found specific functional testing mechanisms to detect bugs in analysis tools [56,54]. The basic idea is to have an automated test data generator that can generate a feature model together with its represented set of products by means of so-called *metamorphic relations*. Once we have a feature model and its set of products, this test input can serve as an *oracle* to see if the expected output of an analysis tool is correct with respect to the test data. The conceptual underpinning of this idea is that most of the analysis operations could be calculated once the set of products represented by the feature model is available. This is a black box testing technique that has been shown to be useful in detecting bugs in some feature model analysis tool like FaMa[5] and SPLOT [6] [56].

Besides functional testing mechanisms for feature model analysis tools, performance is also an additional problem to be taken into account when assuring the quality of this kind of tools. Most of the feature model analysis operations are known to be computationally complex to perform [53] and this is especially important when analysis operations are used for feature model interactive configuration. Most of the times, feature model analysis tools have been tested for performance evaluation using random inputs [8], i.e. a set of random feature models are generated to stress the analysis tools to see how they perform when

[5] `www.isa.us.es/fama`

[6] `www.splot-research.org/`

increasing the size of the models or the percentage of modelling elements like cross-tree constraints. Although this mechanisms are useful to be used to get averages (e.g. in terms of time or memory consumption) there are some proposals to provide mechanisms to build hard feature models in pessimistic situations [55,57]. The idea is to define the problem of looking for hard feature models as an optimization problem. A tool is build to generate hard feature models using metaheuristic algorithms like evolutionary search to explore the space and try to guide the tool to find hard feature model instances for a given tool for a given operation. For instance, the tool can be used to find feature models with between 100 and 200 features with 10% of cross-tree constraints that take more than 5 minutes in detecting the set of dead features which could be considered as an non affordable time constraint.

There are some similar approaches in the product configuration literature to what has been done in feature model analysis. For instance, a configurator can be performance tested by using real configuration models (a.k.a configuration benchmarks [1]) or by using randomly generated configuration models [62]. On the other hand, we found a work providing a technique for white-box testing of configuration systems [21]. However, we have not found any approach to systematically perform functional testing of configurators using metamorphic relations as we found for feature model analysis tools. Similarly, we have not found techniques to systematically search for difficult configuration problems as proposed for hard feature models.

3.5 Summary of findings

Figure 3 provides a first overview [7] of existing related research in the fields of variability models (putting special attention to feature model related results) and product configuration and how they can be used in the other area. Next, we explain the results from our literature review detecting opportunities for cross–fertilization either from feature model analysis to product configuration or backwards.

Feature model analysis can contribute to product configuration in:

- Defining a standard configuration language similar to some of the existing variability languages in the software product line community like any variant of feature models [53], CVL [24] or TVL [11] and providing formal syntax and semantics to the standard configuration language as it has been done in feature models [13,53].
- Providing a historical catalogue of configuration operations similar to what has been reported in the feature model analysis literature [8]. In the feature model literature more than 30 analysis operations exists. Finding a similar catalogue in the product configuration field remains as a challenge. Having such a catalogue can help to summarize the results in the product

[7] Due to the fact that we are reporting first results of our ongoing research, we do not claim for completeness with regard to this overview.

configuration field and ease the adoption of the results by the feature model community.
- Providing more elaborated mechanisms for functional and performance testing of configurators like the ones reported for feature model analysis tools either for functional testing using an automated test data generator [56] or using metaheuristic techniques for finding difficult configuration instances for a given configurator [55,57].

Product configuration can contribute to feature model analysis in:
- Exploring similar automated mechanisms to perform analysis operations using existing approaches like DCSP [26], CCSP [42] or GCSP [23,38]. Also combinations of different paradigms depending on the kind of the operation like description logics and CSP [35].
- Adapting off-line compilation techniques for interactive configuration remains as a challenge in feature model configuration tools in order to provide back–track free feature model configurators [33,29,14,50].
- Reusing explanations mechanisms [25,17,32,46,48,20,22] since the known feature model explanation mechanisms mostly rely on Reiter's theory of diagnosis.

4 Conclusions and future work

Feature model analysis and product configuration has a lot more in common than what has been reported until now. We think that the cross–fertilization of these two independent areas is a mandatory step for the next years at least in the software product line and variability management communities. In this paper, we have reported a rapid literature review that put this fact in evidence and give concrete research opportunities.

To better explore the results of one and other communities a more exhaustive literature review specially in the field of product configuration seems to be desirable and this paper is a first step forward.

Other research questions remained can be related to other engineering task such as maintenance or requirement analysis. In this sense, we have found in the recent variability management related venues papers about reverse engineering of variability models [2,3,30,37,58,67]. Exploring if similar problems haven been addressed in the configuration literature remains as part of our future work.

Acknowledgements

We would like to thank Sergio Segura for giving some comments on a previous version of this paper. This work was supported, in part, by the European Commission (FEDER), the Spanish Government under project SETI (TIN2009-07366) – by the Andalusian Government under project THEOS (TIC-5906), and the Austrian Research Promotion Agency under the project ICONE (827587).

Variability models

- Modelling
  - • standard common variability language (CVL) [24]
  - • different feature modelling dialects [53]
  - • textual variability languages like TVL [11]
  - • formal syntax and semantics [13,53]
- Operations
  - • catalogue of analysis operations [8]
- Quality assurance
  - • functional testing by means of metamorphic relations [56]
  - • performance testing by means of metaheuristic search [57]

Product configuration

- Modelling
  - • UML/OCL-based representations [15]
  - • semantic web and description logics based representations [18,40]
  - • standardization efforts in constraint representations [45]
  - • definition of configuration tasks [43]
- Automated support
  - • Different forms of constraint satisfaction problems DCSP[26], CCSP [42] or GCSP[23,38]
  - • combination of descripton logics and CSP [35]
  - • off-line compilation techniques for interactive configuration [33,29,14,50]
  - • explanation mechanisms [25,17,32,46,48,20,22]
- Quality assurance
  - • configuration benchmark [1]
  - • white–box testing techniques [21]

Fig. 3. Summary of potential synergies

References

1. Configuration Benchmarks Library. available online at `www.itu.dk/research/cla/externals/clib`.
2. M. Acher, B. Baudry, P. Heymans, A. Cleve, and J. Hainaut. Support for reverse engineering and maintaining feature models. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, page 20. ACM, 2013.
3. M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire. On extracting feature models from product descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pages 45–54, New York, NY, USA, 2012. ACM.
4. H. R. Andersen, T. Hadzic, and D. Pisinger. Interactive cost configuration over decision diagrams. *J. Artif. Intell. Res. (JAIR)*, 37:99–139, 2010.
5. K. R. Apt. *Principles of Constraint Programming.* Cambridge University Press, Cambridge, United Kingdom, 2003.
6. T. Asikainen, T. Männistö, and T. Soininen. Kumbang: A domain ontology for modelling variability in software product families. *Advanced Engineering Informatics*, 21(1):23 – 40, 2007.
7. V. Barker, D. OConnor, J. Bachant, and E. Soloway. Expert systems for configuration at digital: Xcon and beyond. *Communications of the ACM*, 32(3):298–318, 1989.
8. D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: a literature review. *Information Systems*, 35(6):615–636, Sept 2010.
9. F. Bouquet and P. Jegou. Using obdds to handle dynamic constraints. *Information Processing Letters*, 62(3):111–120, 1997.
10. P. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, 2007.
11. A. Classen, Q. Boucher, and P. Heymans. A text-based approach to feature modelling: Syntax and semantics of tvl. *Sci. Comput. Program.*, 76(12):1130–1143, 2011.
12. K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wasowski. Cool features and tough decisions: a comparison of variability modeling approaches. In *VaMoS*, pages 173–182, 2012.
13. A. Durán, D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. Flame: Fama formal framework (v 1.0). Technical Report ISA–12–TR–02, Seville, Spain, March 2012.
14. H. Fargier and M.-C. Vilarem. Compiling csps into tree-driven automata for interactive solving. *Constraints*, 9:263–287, 2004.
15. A. Felfernig. Standardized configuration knowledge representations as technological foundation for mass customization. *IEEE Transactions on Engineering Management*, 54(1):41–56, 2007.
16. A. Felfernig. Intelligent techniques for software product line engineering. In *Proccedings of the 2nd International Workshop on Formal Methods and Analysis in Software Product Line Engineering, FMSPLE at SPLC, `www.iese.fraunhofer.de/en/events/fmsple2012.html`*, 2011.
17. A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner. Consistency-based Diagnosis of configuration knowledge bases. *Artificial Intelligence*, 152(2):213–234, 2004.

18. A. Felfernig, G. Friedrich, D. Jannach, M. Stumptner, and M. Zanker. Configuration Knowledge Representations for Semantic Web Applications. *Artificial Intelligence in Engineering, Design, Analysis and Manufacturing (AIEDAM)*, 17(2):31–50, 2003.
19. A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. Web-based configuration of virtual private networks with multiple suppliers. In *Proceedings of 7th International Conference on Artificial Intelligence in Design (AID02)*, pages 41–62, Cambridge, UK, 2002.
20. A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan. Plausible repairs for inconsistent requirements. In *IJCAI*, pages 791–796, 2009.
21. A. Felfernig, K. Isak, and T. Kruggel. Testing knowledge-based recommender systems. *OEGAI Journal*, 4:12–18, 2005.
22. A. Felfernig, M. Schubert, and C. Zehentner. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, 25(2):175–184, 2011.
23. G. Fleischanderl, G. Friedrich, A. Haselboeck, H. Schreiner, and M. Stumptner. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems*, 13(4):59–68, 1998.
24. F. Fleurey, Ø. Haugen, B. Møller-Pedersen, G. K. Olsen, A. Svendsen, and X. Zhang. A generic language and tool for variability modeling. Technical Report A13505, SINTEF, Oslo, Norway, 2009.
25. F. Gedikli, M. Ge, and D. Jannach. Explaining online recommendations using personalized tag clouds. *i-com*, 10(1):3–10, 2011.
26. E. Gelle and B. Faltings. Solving mixed and conditional constraint satisfaction problems. *Constraints*, 8:107–141, 2003.
27. M. J. Grant and A. Booth. A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Information and Libraries Journal*, 26(2):91–108, 2009.
28. T. Hadzic and H. R. Andersen. A bdd-based polytime algorithm for cost-bounded interactive configuration. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI'06, pages 62–67. AAAI Press, 2006.
29. T. Hadzic, S. Subbarayan, R. M. Jensen, H. R. Andersen, J. Møller, and H. Hulgaard. Fast backtrack-free product configuration using a precompiled solution space representation. In *Proceedings of the International Conference on Economic, Technical and Organisational Aspects of Product Configuration Systems*, pages 131–138, 2004.
30. E. Haslinger, R. Lopez-Herrejon, and A. Egyed. Reverse engineering feature models from programs' feature sets. In *18th Working Conference on Reverse Engineering, WCRE 2011, Limerick, Ireland, October 17-20, 2011*, pages 308–312, 2011.
31. A. Hubaux, D. Jannach, C. Drescher, F. Murta, T. Männistö, K. Czarnecki, P. Heymans, N. Nguyen, and M. Zanker. Unifying software and product configuration: A research roadmap. In *Proceedings of the configuration workshop at ECAI*, 2012.
32. D. Jannach and J. Liegl. Conflict-directed relaxation of constraints in content-based recommender systems. In *IEA/AIE*, pages 819–829, 2006.
33. R. Jensen. Clab: A c++ library for fast backtrack-free interactive product configuration. In M. Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 816–816. Springer Berlin Heidelberg, 2004.
34. E. Juengst and M. Heinrich. Using resource balancing to configure modular systems. *IEEE Intelligent Systems*, 13(4):50–58, 1998.

35. U. Junker and D. Mailharro. The logic of ilog (j)configurator: Combining constraint programming with a description logic. In *Proceedings of the IJCAI-03 configuration workshop*, pages 13–20, 2003.
36. F. Liguori and F. Schreiber. The software configurator : an aid to the industrial production of software. In *Proceedings of the IEEE Second International Computer Software and Applications Conference (COMPSAC)*, pages 487–492, 1978.
37. R. Lopez-Herrejon, J. Galindo, D. Benavides, S. Segura, and A. Egyed. Reverse engineering feature models with evolutionary algorithms: An exploratory study. In *Search Based Software Engineering - 4th International Symposium, SSBSE 2012, Riva del Garda, Italy, September 28-30, 2012. Proceedings*, pages 168–182, 2012.
38. D. Mailharro. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing (AIEDAM)*, 12(4):383–397, 1998.
39. T. Männistö, T. Soininen, and R. Sulonen. Product configuration view to software product families. In *In Software Configuration Management Workshop (SCM-10)*, 2001.
40. D. McGuiness and J. Wright. An industrial strength description logics-based configurator platform. *IEEE Intelligent Systems*, 13(4):69–77, 1998.
41. M. Mendonça, A. Wasowski, K. Czarnecki, and D. Cowan. Efficient compilation techniques for large scale feature models. In *Generative Programming and Component Engineering, 7th International Conference, GPCE , Proceedings*, pages 13–22, 2008.
42. S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *AAAI*, pages 25–32, 1990.
43. S. Mittal and F. Frayman. Towards a generic model of configuration tasks. In *Proceedings of 11th International Joint Conference on Artificial Intelligence (IJCAI 1989)*, pages 1395–1401, Detroit, MI,USA, 1989.
44. S. Mohalik, S. Ramesh, J.-V. Millo, S. N. Krishna, and G. K. Narwane. Tracing spls precisely and efficiently. In *Proceedings of the Software Product Line Conference, SPLC(1)*, pages 186–195. ACM, 2012.
45. N. Nethercote, P. Stuckey, R. Becket, S. Brand, G. Duck, and G. Tack. Minizinc: Towards a standard cp modelling language. In *Proceedings of 13th International Conference on Principles and Practice of Constraint Programming (CP'2007)*, pages 529–543. Springer, 2007.
46. B. O'Callaghan, B. O'Sullivan, and E. C. Freuder. Generating corrective explanations for interactive constraint satisfaction. In *CP*, pages 445–459, 2005.
47. B. O'Sullivan. Tutorial on product configuration. In *ASPL 2008, First Workshop on Analyses of Software Product Lines at SPLC, www. isa. us. es/ aspl08/*, 2008.
48. B. O'Sullivan, A. Papadopoulos, B. Faltings, and P. Pu. Representative explanations for over-constrained problems. In *AAAI*, pages 323–328, 2007.
49. K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer, Berlin Heidelberg New York, 2005.
50. V. N. Rao. Solving constraint satisfaction problems using finite state automata. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pages 453–458. AAAI Press, 1992.
51. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
52. D. Sabin and R. Weigel. Product configuration frameworks - a survey. *IEEE Intelligent Systems*, 13(4):42–49, 1998.

53. P. Schobbens, J. T. P. Heymans, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, Feb 2007.
54. S. Segura, D. Benavides, and A. Ruiz-Cortés. Functional testing of feature model analysis tools: a test suite. *IET Software*, 5(1):70–82, 2011.
55. S. Segura, J. Galindo, D. Benavides, J. A. Parejo, and A. Ruiz-Cortés. Betty: benchmarking and testing on the automated analysis of feature models. In *VaMoS*, pages 63–71, 2012.
56. S. Segura, R. M. Hierons, D. Benavides, and A. Ruiz-Cortés. Automated metamorphic testing on the analyses of feature models. *Information & Software Technology*, 53(3):245–258, 2011.
57. S. Segura, J. A. Parejo, R. M. Hierons, D. Benavides, and A. Ruiz-Cortés. Ethom: An evolutionary algorithm for optimized feature models generation (v. 1.1). Technical Report ISA-2012-TR-01, ETSII. Avda. de la Reina Mercedes s/n, 2 2012.
58. S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki. Reverse engineering feature models. In *ICSE*, pages 461–470, 2011.
59. M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. COVAMOF: A framework for modeling variability in software product families. In *Third Software Product Line Conference*, volume 3154 of *Lecture Notes in Computer Science*, pages 197–213. Springer, September 2004.
60. T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen. Towards a general ontology of configuration. *Artificial Intelligence in Engineering Design Analysis and Manufacturing (AIEDAM)*, 12(4):357–372, 1998.
61. S. Subbarayan. Integrating csp decomposition techniques and bdds for compiling configuration problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3524 of *Lecture Notes in Computer Science*, pages 351–365. Springer Berlin Heidelberg, 2005.
62. J. Tiihonen, T. Soininen, I. Niemelä, and R. Sulonen. Empirical testing of a weight constraint rule based configurator. In *Proceedings of the ECAI Configuration Workshop*, 2002.
63. P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.
64. P. Trinidad and A. Ruiz-Cortés. Abductive reasoning and automated analysis of feature models: How are they connected? In *Third International Workshop on Variability Modelling of Software-Intensive Systems. Proceedings*, pages 145–153, 2009.
65. J. White, D. Benavides, D. C. Schmidt, P. Trinidad, B. Dougherty, and A. Ruiz-Cortés. Automated diagnosis of feature model configurations. *Journal of Systems and Software*, 83(7):1094–1107, 2010.
66. J. White, D. Schmidt, D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated diagnosis of product-line configuration errors in feature models. In *Proceedings of the Sofware Product Line Conference*, 2008.
67. L. Yi, W. Zhang, H. Zhao, Z. Jin, and H. Mei. Mining binary constraints in the construction of feature models. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 141 –150, sept. 2012.