Towards More Relational Feature Models

Arnaud Gotlieb, Dusica Marijan and Sagar Sen
Certus Software V&V Center, Simula Research Laboratory, Lysaker, Norway

Keywords: Feature Modeling, Variability, Constraint Programming.

Abstract: Feature modeling is of paramount importance to capture variabilities and commonalities within a software product line. Nevertheless, current feature modeling notations are limited, representing only propositional formulae over attributed variables. This position paper advocates the extension of feature modeling formalisms with richer computational domains and relational operations. In particular, it proposes to extend feature modeling with finite and continuous domain variables, with first-order logic quantifiers ($\forall, \exists$), and with N-ary relations between features attributes, and with so-called global constraints. In order to extend the expressiveness while preserving automated analysis facilities, feature models could be semantically interpreted as first-order logic formulae (instead of propositional logic formulae), including global and continuous dependency between features. In simpler words, this paper emphasizes the importance of having more relational feature models and presents next-generation applications.

1 INTRODUCTION

This paper addresses the challenges imposed on Software Product Line (SPL) engineering (Birk et al., 2003)(Pohl et al., 2005) when engineers have to deal with heterogeneous concerns about modeling, expressiveness, documentation and automation. Roughly speaking, software engineers face a dual challenge: increase the expressiveness of feature modeling notations to deal with the complexity of product representations, while keeping a high degree of automation in the processing of their models. To illustrate this dilemma, consider the following scenario:

The Product-line Engineer. *Øystein works for a Norwegian company, called ELB, producing highly-configurable software control systems for automated subsea oil&gas exploitation. He works as a product-line engineer for ELB, meaning that he manages the variability and commonalities of several similar products for various customers. To facilitate the communication with customers and product managers, Øystein maintains a feature model (FM) representing all the configurations of ELB control systems.*

The Customer. *Following a productive meeting with Magne, a new customer of ELB working for a famous shipyard company, Øystein has tried to configure a new variant of the product line. However, during this process, he realized that Magne asked him to consider an intricate situation: if the control system embeds feature A, and if A is in mode 1, then feature B needs to be excluded from the final product, while if A is in mode 2, then feature C needs to be excluded! This is problematic for Øystein as features in FM can only be represented as binary-state variables; the notion of multiple modes is not available for features. Another request by Magne is to configure a product with no more than 25 features among a set of 57 features.*

The Product Managers. *Later on, Øystein talked to Kjetil and Hoyvind, both product managers at ELB. Kjetil said that including feature A in a product, whatever be its mode, will automatically exclude similar mode for feature B or feature C. Hoyvind, meanwhile, said that, in any case, a product having both features B and C in the same mode is impossible to develop. Øystein is now a bit confused on how to capture these important pieces of information in his FM. In addition, Hoyvind mentioned that feature E, representing the voltage at the bounds of a subcomponent of the control system, continuously depends on feature D, because this feature is linked to D through a physical formula. Øystein has absolutely no idea on how to capture this dependency.*

Issues in FM Modeling. *Now, Øystein has to return a call to Magne, the customer, to explain the impossibility of developing the requested variant, but he is totally confused. He does not know what are the remaining options for deriving a feasible variant*

Gotlieb A., Marijan D. and Sen S..
Towards More Relational Feature Models.
DOI: 10.5220/0005553403810386
In *Proceedings of the 10th International Conference on Software Engineering and Applications* (ICSOFT-EA-2015), pages 381-386
ISBN: 978-989-758-114-4

of the software product line? Are the two distinct modes of feature A responsible for this impossibility? Is the continuous dependency in between D and E the cause of the unrealisable variant? Of course, due to limited expressiveness of FM in general, the informations Øystein gathered during the discussions cannot be encoded in the FM and then it is useless to answer these questions. Magne, the customer, starts to be irritated not having clear answers to his requests. Moreover, Øystein is clueless about incorporating the second request of Magne that states that only 25 features be chosen for a subset of 57 features.

As illustrated by the above scenario, feature modeling is of paramount importance for software product configuration (or derivation), product-line knowledge documentation, negotiation among several professionals, product design optimization and so on. For managing these tasks altogether, one of the challenges in feature modeling is to come up with a formalism having the appropriate degree of expressiveness. As the current underlying semantics of FM is based on propositional logic, several restrictions prevent its usage in many realistic contexts. Another important point to notice is the tremendous importance of automated support analysis of FM. In our view, this is a key element for a widespread adoption of feature modeling.

In this position paper, we address these limitations by proposing a couple of extensions that are inspired from Constraint Programming (CP) (Rossi et al., 2006). The CP paradigm has brought constraint expressiveness to a higher degree by gathering notions from various domains, including Logic Programming, Operational Research and Artificial Intelligence. In CP, finite and continuous domains are recognized as powerful computational domains, overcoming limitations stemming from the usage of a boolean domain. CP is also a convenient tool to address complex mathematical and configuration problems, as a large body of efficient constraint solving techniques is available. Using CP in the context of product configuration (Klein, 1996) or feature modeling is not a new idea (Benavides et al., 2005). It has been successfully applied to solve difficult product configuration problems (Gelle and Weigel, 1996) and also to automatically analyse SPLs (Liang, 2012; Marijan et al., 2013). However, as already pointed out by others (Hubaux et al., 2012), CP is not yet fully exploited in the context of feature modeling. This paper advocates the extension of FM notations to facilitate design and automated analysis. In particular, the paper makes the following contributions:
- Several relational extensions of FMs are introduced. We propose 1) finite domain and continuous domain variable encoding of features, 2) usage of N-ary and global constraints, 3) universal quantification over a feature to increase the expressive power of a FM. These extensions confer a competitive advantage to a FM for variability modeling in SPLs;
- Using CP to support the relational FM extensions allows us to preserve the efficiency of automated FM analyses. For each proposed extension, we carefully discuss automated support and constraint solver usage. Note however that the paper does neither provide any concrete syntax for the relational extensions nor any specific recommendation for selecting constraint solvers;
- Two next-generation applications of these relational extensions are discussed, opening perspectives for a wider adoption of FM.

The paper is organized as follows: Sec.2 recalls the basic principle of a FM and its semantics. Sec.3 introduces and discusses several relational FM extensions. Finally, Sec.4 presents two next-generation applications.

2 FEATURE MODELING

Feature Model (FM) is a convenient graphical notation to represent variability in SPLs (Kang et al., 1990) and highly-configurable software systems (Czarnecki et al., 2005), (Eriksson M. and K., 2005). It is used to capture: (i) configurable software parameters (a.k.a. features) and (ii) dependencies between the features. The underlying interpretation (or semantics) of a FM is however based on propositional logic (Batory, 2005): A *Feature Model* is a propositional formula over a set of features; A *feature* is a boolean variable that represents a composition element or a configuration parameter; A *product* (or *configuration*) is a total assignment of features from a FM. A feature is present within a product or configuration iff its corresponding variable equals to 1 or *True*; A *valid product* is an assignment which satisfies the propositional formula of a FM; A *constraint* is a relation between two or more features. We distinguish between two types of constraints: *A hierarchical constraint* is a relation among a father feature and its children features based on the following operators AND, OR, XOR, OPT, CARD. *A cross-tree constraint* is a binary relation among a pair of features based on the operators REQUIRES, MUTEX. Fig.1 represents a FM for Video-Conference Software (VCS) products, extracted from a real-world case study (Marijan et al., 2013). The model specifies that VCS supports calls, which can be either

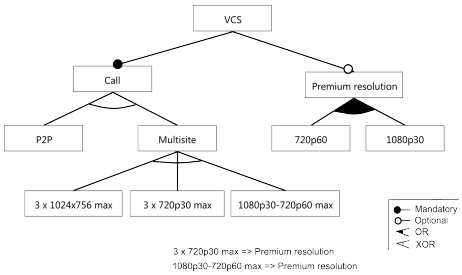

Figure 1: Excerpt of a feature model for a video conferencing software.

P2P or *Multisite* calls, i.e., either $3x1024x576$ *max*, $3x720p30$ *max* or $1080p30$–$720p60$ *max*. Optionally, VCS supports $720p60$ or $1080p30$ premium resolution, or both. Configurations with $3x720p30$ *max* or $1080p30$–$720p60$ *max* resolution must have *Premium resolution* feature.

For the FM shown in Fig.1, the set of features $[VCS, Call, Multisite, 3x1024x576\ max]$ is a valid product, but $[VCS, Call, Multisite, 3x720p30\ max]$ is invalid, because the constraint between $3x720p30$ *max* and *Premium resolution* features is not satisfied. Since its introduction in 1990 (Kang et al., 1990), feature modeling has been adopted in some industrial sectors, e.g., (Dordowsky and Hipp, 2009), and has been derived into several dialects (Schobbens et al., 2007).

3 RELATIONAL FEATURE MODELS

This section introduces several extensions to increase expressiveness of FMs. Each subsection discusses the Pros/Cons of a specific extension in terms of complexity and/or degree of automation.

3.1 From Boolean to Finite Domains

In traditional FMs, a feature is associated with a two-state variable: it can either be *present* or *absent* within a product. This is an over-simplistic view of the reality where features have various (but often discrete) states. We propose that each *feature* is associated with a finite set of possible values, called *modes*. These modes can advantageously be encoded using a Finite Domain (FD) variable A, where A is associated a domain of possible modes $modes(A) = \{d_1, \ldots, d_n\}$ where $d_i \geq 0$. Note that, in a product, a feature can only be assigned to a single mode. For example, mode d_1 of feature A is used in product P_1, while mode d_2 of the same feature is used in product P_2. Thus, cross-tree constraints can be defined over mode values, e.g., $A > 1$ Requires $B \leq 1$ meaning that any mode strictly grater than 1 for A implies either mode 1 for B or mode 0, interpreted as absence of B. This usage of FD encoding of features is not really new and has been reported in the literature, especially when automated analysis on FM is required (Benavides et al., 2010). As a concrete example, the *Multisite* feature of Fig.1 can be associated with a FD variable, having mode value 1 equal to $3x1024x576$ *max*, mode value 2 equal to $3x720p30$ *max*, and mode value 3 equal to $1080p30$–$720p60$ *max*. Using this representation, both cross-tree constraints can easily be encoded as *Multisite* > 1 Requires *Premium resolution* $= 1$. One could object that FD encoding is already possible in current purely boolean FM, using a *Xor* node. However, using a *Xor*-node for encoding subtrees with tens of leafs is not efficient and using a FD variable can considerably simplify the semantical interpretation of FMs. Indeed, as the *Xor* operator is binary in boolean logic, it is required for any pair of leafs. Thus, the number of *Xor* grows quadratically with the number of leafs in any boolean interpretation of FM. On the contrary, using FD variables, a single domain membership constraint is required in the semantical interpretation, e.g., $A \leftarrow Xor(A_1, A_2) \wedge Xor(A_1, A_3) \wedge Xor(A_2, A_3)$ is encoded as $A \in 1 \ldots 3$.

3.2 Featuring Global Constraints

Cross-tree constraints in FMs include binary constraints: (A Requires B) and (A Mutex B). However, several extensions have already been proposed such as ternary relations or N-ary relations (Uzuncaova et al., 2010; Martin Fagereng Johansen and Fleurey, 2011). For example, in SPLOT (Mendonça et al., 2009), cross-tree constraints come as ternary relations over booleans. There is no formal breakthrough to switch from ternary to N-ary relations in boolean algebra. It means that formal reasoning is independent from the arity of the initial relations. In fact, this extension comes for free when using an underlying SAT-solver or CP-solver to reason over FMs. Nevertheless, we propose to go further by using non-fixed arity global constraints among finite domain variables. Since two decades, researchers of the CP community have built a rich catalog of global constraints to express various and complex relations used for different applications. The most recent version of the catalog contains more than 350 global constraints (Beldiceanu et al., 2007).

Additionally, most CP solvers are incremental, meaning that adding a constraint provokes *constraint propagation*, which is a process that pushes configuration choices throughout the constraint system. If an

assignment is (partially) consistent, then other variable domains can be shrunk, while if it is inconsistent, a backtracking process removes it and undoes all the deductions. Constraint propagation is particularly useful during interactive product configuration as it enables early failure detection and backtracking. When configuring a product, one can easily evaluate the consequences of a configuration choice for other features. For example, including features F_1, F_2 in a product where **Sum_ctr**$([F_1, F_2, F_3], \leq, 2)$ is enforced, automatically assigns 0 for F_3.

3.3 Adding Logical Quantifiers

FMs are interpreted as propositional logic formulas: features cannot be quantified. Extending FMs to first-order logic formulas means that features can be existentially and universally quantified. Basically, this extension enables simple expression of complex relations among feature modes. Feature universal quantification is interesting to express invariants over SPLs and has already been drafted in the Clafer language (Antkiewicz et al., 2013). For example, expressing that for each mode value of feature A, there exists a mode value of feature B that satisfies a specific property P can easily be formulated as: $\forall d \in modes(A), \exists t_d \in modes(B)$, such that P holds. Support for automatic reasoning over first-order formulae is crucial when enriching the semantics of FMs. Recently, tools, called as QCSP-solvers, have been developed to support such reasoning in the context of finite domain constraint solving (Gent et al., 2008). It becomes possible to solve efficiently constraints such as $\exists x \forall y (x \neq y) \implies \exists z (z < x) \wedge (z < y)$. For example, QeCode is a QCSP-solver available online that shows reasonable performances, provided that constraints stay simple. However, extending FM to high-order logics, enabling quantification over operations among the features, would compromise the efficiency of automated analyses.

3.4 Continuous Domains and Dependencies

Using continuous variables in product configuration is well known (Xie et al., 2006). Even if discretization of continuous variables is often possible (e.g., considering that a voltage takes a value in only N possibilities $\{v_1, \ldots, v_N\}$), introducing continuous variable increases the expressive power of a FM. Indeed, very often, discrete values are not available at product configuration time because they depend on other components, not yet available. In addition, physical variables in systems are known with tolerance ranges, coming from the specifications of electronic components, that cannot be handled accurately with discretized values. Finally, configuration variables may only take values resulting from complex mathematical computations. As a consequence, these values cannot be represented using pre-selected discrete values. For example, consider the relation $y^2 - x \leq 0$ and the value 0.1 for x. The domain of y can only be represented as a range of possible real values, i.e., $[-0.01, 0.01]$. Any discretization of this range would inaccurately represent the relation. Most of the physical variables of a system are linked together with continuous relations (e.g., Ohm law for electronic components), but representing them with non-discrete variables in FMs for SPL is questionable. One can argue that physical entities may not correspond to configurable software parameters. We believe however that introducing continuous variables for configuring a product could open FMs to a wider range of applications. As a simple example, consider the design of a cylindrical vessel defined by two continuous variables: *height* and *radius*. The volume of the vessel could be maintained in a FM by the constraint $V = \pi \times radius^2 \times height$. Even if *height* and *radius* are sufficient to configure the vessel and could thus be discretized, a final user may want to configure the vessel through its volume instead of these two variables. Usually, physical variables take their values in predefined ranges, for which it is interesting to estimate the extreme values (e.g., what is the maximal vessel volume that can be computed). So, capturing the continuous relation between these variables, that are reflected in a FM by feature variables, is interesting for configuration purposes. This proposition would be meaningless without the support of operational constraint solvers over continuous domains. Fortunately, solvers such as Interlog (Lhomme et al., 1998), IBM Ilog-Labs CP-Optimizer or RealPaver (Granvilliers, 2003) are available for that purpose. Note also that recent developments in constraint solving over floating-point computations have led to the design of specific solvers that tackle correctly rounding modes and approximations (Bagnara et al., 2013). It could be crucial for some safety-critical software product lines to capture accurately relations over floating-point computations, especially if the variability models are used to select software components in order to integrate a complete product release.

4 NEXT-GENERATION APPLICATIONS

Extending feature modeling is interesting iff the evi-

dences of its benefits can be shown in practice. For instance, the scenario presented in the introduction is realistic and corresponds to a situation addressed by researchers of the Certus center. In this section, we go a step further and present the application of relational feature models to some next-generation application domains.

4.1 Context-aware Sensor Fusion

Raw data from sensors and user interaction come in variable frequencies, from finite/continuous domains, and have different domain types such as strings, reals, and integers. A fusion of the signals received at an instant needs to be modelled and analyzed to accurately define a *context*. For instance, in (Acher et al., 2009), FMs are used to represent environmental contexts with features such as luminosity. A context configuration is mapped to the configuration of a vision system. The choice of RGB camera or infra-red camera is based on detected luminosity, which is a variable with a continuous domain. However, a FM is severely limited by the boolean nature of feature variables, easily giving a rise to an exponentially growing context FM. We believe that extending FMs with continuous domain variables could simplify the modeling of sensor fusion applications.

4.2 Filtering in Big Data

The explosion of data from business processes, social networks, and scientific instruments presents both the boon of data and the bane of processing it. Often big data contains numerous redundancies or even faulty data due to measurement errors. Therefore, there is a necessity to select data records in a controlled manner. Relational FMs can be used to model a filter for a large number of multivariate data records in a database. Data records that are a valid configuration of a relational FM can be selected for further processing. A relational FM can be seen as filter for selecting correct and representative data records in Big Data. In (Sen and Gotlieb, 2013), FMs are used to model variation in data-intensive systems: 160 currencies and 900 different types of taxes are modeled as features. There are thousands of pairwise interactions between these two sets of features. Representing such a large variation of a data field as a set of boolean features in a graphical FM is highly unreadable. Therefore, a paradigm shift in the standard for feature modelling is needed.

5 CONCLUSIONS

Relational FMs are promising for opening new applications and research directions. For us, CP can drive such extensions through FD and continuous domains extensions, global constraints, and logical quantification, because they preserve automated analyses capabilities over FMs.

ACKNOWLEDGEMENTS

This work is partially supported by the Research Council of Norway (RCN) through the Research-based Innovation Center (SFI Programme) Certus.

REFERENCES

Acher, M., Collet, P., Fleurey, F., Lahire, P., Moisan, S., and Rigault, J.-P. (2009). Modeling Context and Dynamic Adaptations with Feature Models. In *Proceedings of the 4th International Workshop Models@run.time*, page 10, United States.

Antkiewicz, M., Bak, K., Murashkin, A., Olaechea, R., Liang, J., and Czarnecki, K. (2013). Clafer tools for product line engineering. In *Software Product Line Conference*, Tokyo, Japan.

Bagnara, R., Carlier, M., Gori, R., and Gotlieb, A. (2013). Symbolic path-oriented test data generation for floating-point programs. In *Proc. of the 6th IEEE Int. Conf. on Software Testing, Verification and Validation (ICST'13)*, Luxembourg.

Batory, D. (2005). Feature models, grammars, and propositional formula. *Software Product Lines, Lecture Notes in Computer Science*, 3714(3):7–20.

Beldiceanu, N., Carlsson, M., Demassey, S., and Petit, T. (2007). Global constraint catalogue: Past, present and future. *Constraints*, 12:21–62.

Benavides, D., Segura, S., and Ruiz-Cortés, A. (2010). Automated analysis of feature models: A detailed literature review. *Information Systems*, (35):615–636.

Benavides, D., Trinidad, P., and Ruiz-Cortés, A. (2005). Using constraint programming to reason on feature models. In *Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE'05), Jul. 14-16, 2005, Taipei, Taiwan, Republic of China*, pages 677–682.

Birk, A., Heller, G., John, I., Schmid, K., von der Massen, T., and Muller, K. (2003). Product line engineering, the state of the practice. *Software, IEEE*, 20(6):52–60.

Czarnecki, K., Helsen, S., and Eisenecker, U. (2005). Formalizing cardinality-based feature models and their specialization. *Software Process Improvement and Practice*, 10(1):7–29.

Dordowsky, F. and Hipp, W. (2009). Adopting software product line principles to manage software variants in a complex avionics system. In *Proc. of the 13th Int. Soft. Product Line Conf.*, SPLC '09, pages 265–274.

Eriksson M., B. R. J. and K., B. (2005). The pluss approach domain modeling with features, use cases and use case realizations. In *Software Product Line Conference (SPLC'05)*, pages 33–44.

Gelle, E. and Weigel, R. (1996). Interactive configuration based on incremental constraint satisfaction. In *Proc. of Practical Application of Constraint Technology (PACT'96)*.

Gent, I., Nightingale, P., Rowley, A., and Stergiou, K. (2008). Solving quantified constraint satisfaction problems. *Artificial Intelligence*, 172:738–771.

Granvilliers, L. (2003). *RealPaver User's Manual : Solving Nonlinear Constraints by Interval Computations*. University of Nantes, FR. Release 0.3.

Hubaux, A., Jannach, D., Drescher, C., Murta, L., Mnnist, T., Czarnecki, K., Heymans, P., Nguyen, T., and Zanker, M. (2012). Unifying software and product configuration: A research roadmap. In *Proceedings of Configuration Workshop at ECAI 2012, Montpellier, France*.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute.

Klein, R. (1996). A logic-based description of configuration: The constructive problem solving approach. pages 1–10. AAAI Press, Menlo Park, CA.

Lhomme, O., Gotlieb, A., and Rueher, M. (1998). Dynamic optimization of interval narrowing algorithms. *Journal of Logic Programming*, 37:164–182.

Liang, J. (2012). Solving clafer models with choco. (GSDLab-TR 2012-12-30).

Marijan, D., Gotlieb, A., Hervieu, A., and Sen, S. (2013). Practical pairwise testing for software product lines. In *Software Product Line Conference (SPLC'13), Industrial track*, Tokyo, Japan.

Martin Fagereng Johansen, O. H. and Fleurey, F. (2011). Properties of realistic feature models make combinatorial testing of product lines feasible. In *Conference on Model Driven Engineering Languages and Systems (MODELS'11)*, pages 638–652.

Mendonça, M., Branco, M., and Cowan, D. (2009). S.p.l.o.t.: software product lines online tools. In *OOPSLA Companion*, pages 761–762.

Pohl, K., Böckle, G., and Linden, F. J. v. d. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Rossi, F., Beek, P. v., and Walsh, T. (2006). *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA.

Schobbens, P., Heymans, P., Trigaux, J., and Bontemps, Y. (2007). Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479.

Sen, S. and Gotlieb, A. (2013). Testing a Data-intensive System with Generated Data Interactions: The Norwegian Customs and Excise Case Study. In *25th International Conference on Advanced Information Systems Engineering (CAISE'13)*, Valencia, Espagne.

Uzuncaova, E., Khurshid, S., and Batory, D. (2010). Incremental test generation for software product lines. *IEEE Trans. Softw. Eng.*, 36:309–322.

Xie, H., Henderson, P., and Kernahan, M. (2006). A constraint-based product configurator for mass customisation. *IJCAT Journal*, 26(1/2):91–98.