

On the Modularization of Feature Models [★]

D. Benavides¹, S. Trujillo², and P. Trinidad¹

Dpto. de Lenguajes y Sistemas Informáticos
University of Seville
Av. de la Reina Mercedes S/N, 41012 Seville, Spain
{benavides, ptrinidad}@us.es

² ONEKIN Group. University of the Basque Country
PO Box: 649
20009 San Sebastián, Spain
struji@ehu.es

Abstract Feature Models (FM) are used to represent commonality and variability in Software Product Lines. Since the first proposal by Kang, the notation of FM has evolved in different ways: introducing cardinalities, using UML or XML notations, etcetera. In addition, the use of FMs is not only restricted to domain analysis because it is widely accepted that FMs can be used in different stages of SPL development in order to produce other assets such as requirements documents, architectures, reasoning frameworks or even pieces of code. Hence, FMs turn into an important focus of research in the field of model transformation. In this context, two characteristics are needed: *i*) an agreed base feature model and *ii*) a platform independent representation allowing extensions for specific needs. In this paper we propose an abstract feature model, providing an XML representation and mechanisms to extend this model for a particular approach. Therefore, the contribution of this work is the modularization of the FM in order to cope with distinct development stages.

1 Introduction

Feature Models (FM) are an important asset in Software Product Line (SPL) development. They have been quoted as one of the main contributions to domain analysis. Likewise, there are some works in the literature that propose the use of FMs to model SPL not only in domain analysis but in requirements elicitation [9], architecture development, and code generation [1,4]. Therefore, FMs are becoming a very important starting point in model transformation to derive other models. FMs have evolved during the last years and some extensions have been proposed to deal with particular needs in particular stages of development. Recently, cardinality-based feature models have been proposed [5] as an integration of previous notations and they have been formalized as context free grammars in order to avoid unnecessary misconceptions.

Owing to FMs are important assets in different stages of development there are at least two important characteristics that FM should fulfill: *i*) an agreed base FM and *ii*)

[★] This work was partially supported by the Spanish Science and Education Ministry (MEC) under contracts TIC2003-02737-C02-01 (AgilWeb), PRO-45-2003 (FAMILIES) and TIC2002-01442. Salvador Trujillo enjoys a doctoral grant for the MEC.

a platform independent representation allowing extensions for specific needs. The main idea behind this paper is to modularize FMs in order to fulfill these two characteristics. This idea is mostly inspired by two recent works on SPLs. On the one hand, Czarnecki's stage configuration idea stands for specialization of FMs [5]. Hence, FMs are specialized for each stage, and feature selection is modularized. On the other hand, Dashofy et Al. propose a modular software architecture, which consists of a base architecture module together with optional modules (e.g. evolution module, SPL module, etcetera) realizing additional architecture characteristics [6]. Hence, architecture model is modularized. The same idea applied to FMs is capitalized in this work, this is to say, having a base feature model and specific modules to extend them for specific purposes (e.g.: production module, reasoning module, etcetera).

2 Modular Feature Models

Feature models were introduced by Kang to domain analysis in a SPL [10] and were very appreciated because of its simplicity in domain analysis tasks. However, nowadays FMs are commonly used in other software product-line practices such as architecture management, code generation in generative approaches and so forth demanding the representation of additional information actually related to features. To attain this, we propose the idea of modular feature models. This permits to broaden the scope of FMs to other SPL practices than domain analysis.

In order to achieve the modularization of FMs, our approach provides: *i*) an XML abstract representation of base FM module derived from current FM approaches, *ii*) XML-based extension mechanisms for the definition of new FM modules, and *iii*) two possible cases to module definition using these extension mechanisms. Modular FMs enable slightly different FMs for distinct purposes. However, all FM share the very same base FM module, and are constructed from a common set of FM modules by using the same standardized mechanisms. Thus, the specific requirements for a given software practice are supported, but maintaining the advantages of a unique FM representation.

2.1 Base Feature Model: An XML Abstract Representation Model

Cardinality-based Feature Models (CFM) [5] have been introduced as an integration of several previous notations of FMs. They have been formally defined as context free grammars which avoid unnecessary misconceptions. Figure 1 presents the CFM of the JAMES system. JAMES is a SPL of collaborative web-based systems[8].

A CFM is composed by a **root** (*JAMES* in Figure 1) and an optional set of **constraints** (they refer to global constraints: requires and excludes; *R9* and *R10* in Figure 1). A root, as a **Feature** is composed by an optional set of **relations** (it would be controversial to discuss whether it makes sense to have a FM with only a root feature). Relations can be of two different types: **binary relations** which includes mandatory (e.g. *R1*), optional (e.g. *R2*) and cardinality-based like relations (e.g. *R4*) or **set relations** (e.g. *R7*). A **feature** can be of three different types and is composed by one or more relations. A binary relation is composed by one and only one **solitary feature** which is the child feature due that the parent feature is the one that has this relation (*Core* or

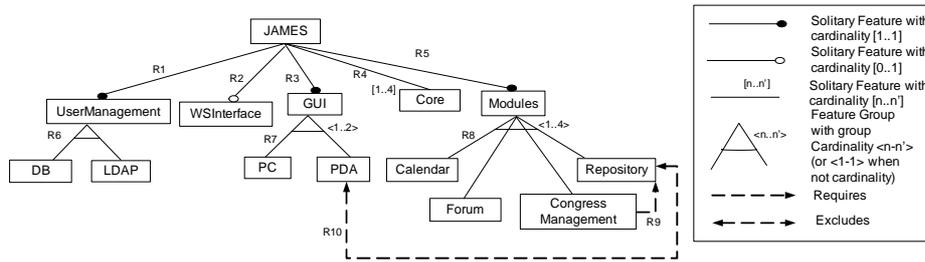


Figure 1. james CFM

UserManagement are examples of solitary features); A set relation is composed by at least two **grouped** features (*Calendar, DB* or *PDA* are examples of grouped features). In addition a solitary feature and set relations comprise one or more cardinalities. Note that in the graphical representation we can put no cardinality in set relations but in fact that means that the cardinality is $\langle 1-1 \rangle$. Likewise there are graphical representations for common used cardinalities of solitary feature like $[1..1]$ and $[0..1]$ (see Figure 1 notes).

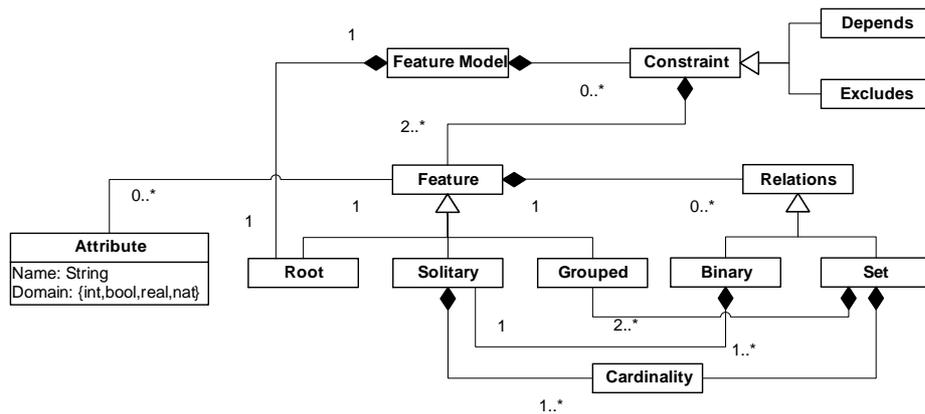


Figure 2. CFM meta model

Figure 2 represents our meta model for CFM. It has some improvements regarding the one presented in [3] and the later XML improvement in [11] (note that in this proposal there is no meta model but an XML Schema representation which could be considered as a meta model): *i*) we fully support CFM because we allow to have more than one cardinality to both solitary features and set relations which is not allowed in [11]; *ii*) we allow the possibility of having more than one attribute in any feature which is not allowed neither in [11] nor in [5], this could be seen as a syntax facility due that it is always possible to represent more than one attribute as sub features (see [5]) but we think is more natural to both customer and developers to have the possibility of having

more than a single attribute in a feature; *iii*) our proposal is not tool-oriented as XFeature is [3], so we allow to represent global constraint in the same FM and we do not add unnecessary information in our meta model with references as XFeature does, and *iv*) the models developed using XFeature can be easily transformed to our meta model due that they can be seen as a subset of our proposal but not always backwards.

Our XML representation can be fully accessed in <http://www.tdg-seville.info/topics/spl/>. The XML representation of UML composition relations of figure 2 are straightforward using the elements of XML Schemas

Some mechanisms are necessary in order to define new modules extending base FM. As proposed by [3] we agree that XML Schema is the appropriate technology to represent the meta-model of FM. In general, XML is an extensible and modular language. Particularly XML Schema provides **<redefine>** mechanism, which enables to redefine simple and complex types, groups, and attribute groups that are obtained from external schema files. Hence, a new module may be defined by creating a particular XML schema extending the base FM schema proposed previously.

2.2 Sample Cases

A case in point is AHEAD methodology, it could be of interest to integrate information related to layers along with the FM. By doing so, it bridges the gap between features and layers. Hence, layer equation is derived from feature selection making automatic product production possible. This is the case of web application-line production [7]. In this approach, layers are specified within the FM XML document. Thus, layer equation is obtained when features are selected, and production is automatically performed. To attain this, a new module is needed in order to extend FM with layer information. This extension to the FM enables production following AHEAD methodology.

Another example of FM XML document refinement is when FM wants to be used in order to extract knowledge about the SPL. To do so, it is necessary to transform the FM into a reasoning framework such as the one presented in [2] where a transformation from a FM into a CSP (Constraint Satisfaction Problem) was proposed. Doing so, it is possible to have information about the FM such as how many potential products are represented in the FM, whether a configuration is valid or not, how many variability is present in a given FM, how many commonality a given features has in a FM, which is the lowest price for a product, which is the best product in terms of quality, performance or security, and so forth.

In addition, new modules may be defined in order to cope with requirements, documentation, architectures, etcetera.

3 Conclusions and Future Work

This work is a first attempt to **modular feature models**. To this end, several mechanisms are provided in order to make modularization possible. Furthermore, modular feature models idea has been used tentatively in the examples presented in this paper. However, it lacks of specific tool support and needs more case studies in order to validate this proposal.

We have briefly presented a FM model with some minor additions (capability of having more than a single attribute for every feature). Then, we gave a representation of this model using XML technologies (i.e. XML and XML Schema). We argue that an independent representation of FM is desirable in order to do an automatic transformation into different assets of a Software Product Line. We improve some aspects of former XML representations and we are using this representation in the development of a tool to transform CFM into a CSP (Constraint Satisfaction Problem) which allows us to have an automatic management of FMs. In addition, this representation is used in application production. In the future we want to improve our tool support integrating our approach in XFeature using our reasoning framework [2]. Likewise, we are working in a prototype that allows the automatic generation of portal-based applications selecting features and transform them using the AHEAD tool [7].

References

1. D. Batory. Feature models, grammars, and propositional formulas. In *to be published at Software Product Line Conference (SPLC 2005)*, 2005.
2. D. Benavides, Ruiz A. Cortés, and P. Trinidad. Automated reasoning on feature models. *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 3520:491–503, 2005.
3. V. Cechticky, A. Pasetti, O. Rohlik, and W. Schaufelberger. Xml-based feature modelling. *LNCS, Software Reuse: Methods, Techniques and Tools: 8th ICSR 2004. Proceedings*, 3107:101–114, 2004.
4. K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Techniques, and Applications*. Addison–Wesley, may 2000.
5. K. Czarnecki, S. Helsen, and U. W. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.
6. E. M. Dashofy, A. van der Hoek, and R. N. Taylor. A comprehensive approach for the development of modular software architecture description languages. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 14(2):199–245, April 2005.
7. O. Díaz, S. Trujillo, and F. I. Anfurrutia. Supporting production strategies as refinements of the production process. In *to be published at Software Product Line Conference (SPLC 2005)*, 2005.
8. P. Fernandez and M. Resinas. James project. Available at <http://jamesproject.sourceforge.net/>, 2002-2005.
9. G. Halmans and K. Pohl. Communicating the variability of a software–product family to customers. *Journal on Software and Systems Modeling*, 2(1):15–36, 2003.
10. K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature–Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.
11. A. Pasetti and O. Rohlik. Technical note on a concept for the xfeature tool. Technical report, P and P Software GmbH / ETH-Zurich, 2005.