From Feature Models to Business Processes*

Ildefonso Montero, Joaquín Peña, Antonio Ruiz-Cortés

Depr Lenguajes y Sistemas Informáticos. University of Seville. Spain {monteroperez, joaquinp, aruiz}@us.es

Abstract

The variability level of average-size Business Information Systems (BIS) is highly enough for making the design of this kind of systems a complex task. There is an approach called Process Family Engineering (PFE) that tries to ease the design of BIS using ideas from the Software Product Lines (SPL) field. Roughly speaking, they propose to, first, study the variability of the system without entering into details by means of building a variability model (called feature model), that is used later for building the business process.

However, in PFE the process of deriving the business process from the feature model is performed manually. Authors use feature models with a different meaning that is commonly accepted in SPL. In this paper, we provide a rigorous description for the new meaning of feature models, and a mapping relationship that defines how to use the information in the FM for obtaining the basic structure of the business process. In addition, as a proof of concepts, we have implemented an MDD transformation that provides the expected results.

1 Introduction

The development of *Business Information Systems* (BIS) is focused on providing techniques and mechanisms for designing software systems based on the business processes of the companies, defined graphically by means of business process modeling notations, such as *Business Process Model Notation* (BPMN) [4]. The variability level of average-size BIS is usually highly enough for making the design of this kind of systems a complex task.

Software Product Lines (SPL) systematizes the reuse across the set of similar products that a software company provides. For that purpose, this approach requires to describe the products by means of variability models, such as *Feature Models* (FM), that contains only features and relationships between them. A feature is considered as a characteristic of the system that is observable by the end users. *Feature Models* describe which features are present in all the products, called core features, and which not, called variable features.

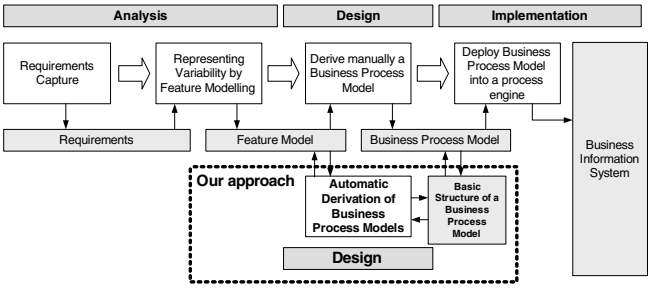

Figure 1. Overview of the PFE approach for modeling BIS and our approach

Schnieders *et al.* propose a methodology for designing highly variable business processes [9]. It is based on overcoming the complexity derived from variability, by means of applying software product lines for managing it. This methodology, called *Process Family Engineering* (PFE), presents three steps for modeling variant-rich BIS, as shown in Figure 1, namely: (i) *Analysis*, which is focused on performing a requirements capture that covers the user needs and describes the variability using feature models; (ii) *Design*, which is focused on deriving manually a business process model from a feature model that represents its variability; and finally (iii) *Implementation*, which is focused on deploying the business process model specification into a process engine that executes it and produces a BIS. Thus, PFE reduces the complexity derived from variability by means of studying features models that do not provide details on how each process is performed.In addition, PFE considers that sometimes a feature represents an activity, sometimes a business process, but without providing an equivalence definition. Thus, we can say that in PFE there not exist a

*This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472), Andalusian Government project ISABEL (TIC-2533), and under a scholarship from the Education and Universities Spanish Government Secretariat given to the author Ildefonso Montero.

mapping relationship between feature models and business process models (see Section 2).

However, although PFE may be the solution to manage the evolution of the business process of a company, the *Design* step of this approach, concretely the use of feature models and the derivation of business processes from it, presents three main drawbacks, which are the focus of this paper. First, *ambiguity*: PFE uses feature models to show the variability derived from enabling/disabling feature/process; however, given that feature models are devoted to represent design-time variability and not runtime variability [8][6], the approach redefine the semantics of feature models implicitly, but without providing a definition for it. Second, *maintenance*: PFE extends the notation of BPMN to add information about variability which is also present in the feature models, thus, information is duplicated with the obvious problems for maintenance. Third, *manual derivation*: the relationship between a feature model and its corresponding business process is not rigorously defined, and the development of the business process is performed manually using as input the feature model, what makes this activity error-prone and hinders the maintainability of both kind of models.

Thus, the main motivation of this paper is to improve the design step for modeling highly variant-rich business process models proposed by PFE. For that purpose, we provide a rigorous description for the new meaning of feature models, presented in Section 2, and mapping relationship that clearly defines how to use the information in the FM for obtaining the basic structure of the BP (that needs to be completed manually), detailed in Section 3. As shown in Figure 1, the derivation of the basic structure of a business process model from a feature model will be done automatically. This transformation is achieved by redefining the semantics of feature models (FM) using context-free grammars, a transformation of this grammar to a finite state machine model, and a representation of these state machines using business process models. Figure 2.a sketches the overview of this systematic mapping. In addition, as proof of concepts, we also provide an implementation, by means of a MDD transformation, of the mapping between feature models and business process models using *Atlas Transformation Language*(ATL)[1]. Figure 2.b presents the overview of this implementation.

As a result of our contributions, we improve the design of complex business process models. Concretely, we improve the *Design* step of the PFE approach by means of improving the maintainability of feature models and BPMN since we provide an automatic mapping that can reduce errors derived from manual transformations. In addition, we avoid the need of extending the standard notation of BPMN with information that is present in the feature model.

[1] http://www.eclipse.org/m2m/atl/

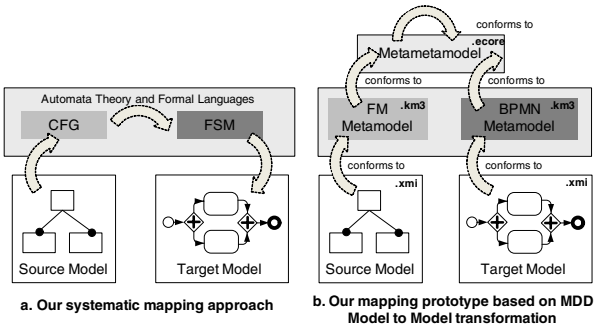

Figure 2. Overview of our approach and our first prototype for automated mapping

2 Definition of a New Semantic for Feature Models

In order to perform a *Process Family Engineering* (PFE) feature model grammar representation, we need to define a new *Context-Free Grammar* (CFG) taking into account that in SPL it is not needed to establish the order of appearance of the features into a family product, but in our context it is recognized as a core need. Process engineers need to perform business process definitions which establish the order that the processes must be performed and its dependencies with others (i.e: subprocess A and B must be done in parallel, and after that, subprocess C must be performed).

For defining this mapping between FM and business process, we have considered that:

- parent features in a feature model, namely *variation points*, are considered as complex processes.
- child features in a feature model, namely *variants*, are considered as subprocesses.

In order to rigorously define the new semantics, we reuse Batory's grammar representation of FM [3] as starting point for proposing a new grammar. The redefinition is shown in Figure 3. From this grammar, a regular expression of these languages can be obtained by means of operations of automata and formal languages theory defined in [7]. Figure 3 sketches also the equivalence of a feature and its relationships in terms of regular expressions (Notice that parallel execution of features are represented by means of • character). In addition, each possible composition between two or more different artifacts is resolved by means of parallel decompositions. Figure 4 presents an example of this composition which sketches how a feature model with three different relationships is defined by means of a composition of three simplified feature models with only one relationship. Thus, the CFG representation of composed feature model is

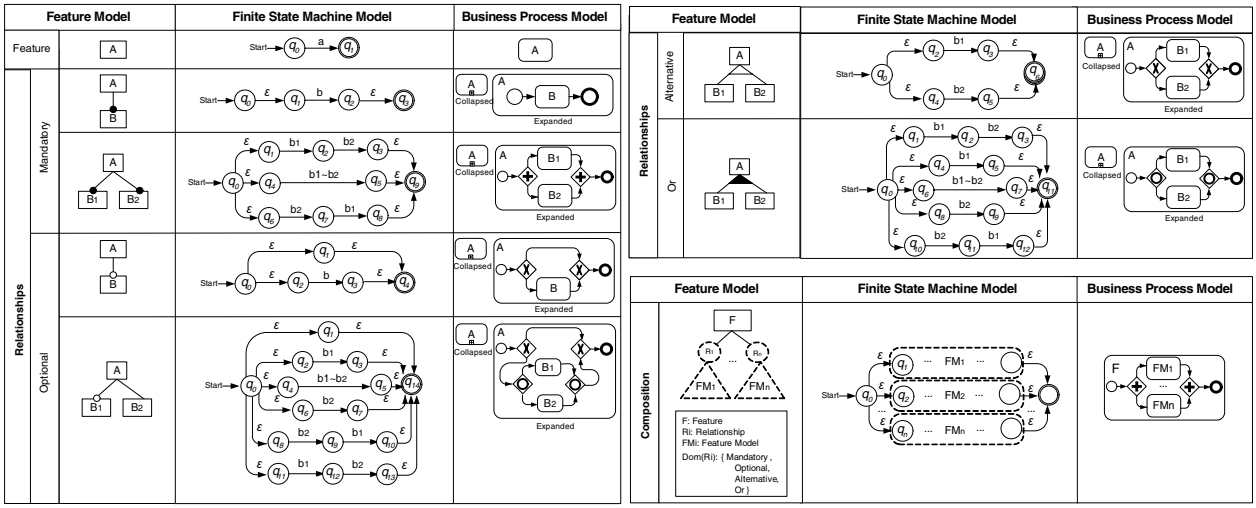

Figure 5. Feature Model to BPMN Mapping and Composition Catalog Proposed

obtained by means of applying • operator to three simplified feature models CFG representations defined previously.

3 Mapping a Feature Model to a BPMN Structure

Automata and formal languages theory sets the steps needed to obtain a *Finite State Machine* (FSM) model from a *Context Free Grammar* (CFG) and viceversa[7]. Applying this mapping we provide a FSM representation of the feature model grammars presented previously.

In addition, BPMN can be represented by means of FSMs[5]. In this approach, the equivalence based on which artifacts of BPMN can implement the behavior of a FSM has been explored, concluding that representing a FSM by means of BPMN is feasible.

The BPMN specification does not provide any constraint about the order of performing subprocesses in any situation. In addition, the BPMN gateways defines that the subprocesses contained in it can be done as a sequence or parallel under several constraints. Thus, the BPMN gateways are feasible to be used for implementing proposed finite state machines behavior, as shown in Figure 5 that presents the equivalence between each of FSMs and its representation using BPMN.

As stated previously, we have developed a proof of concepts of the mapping by means of MDD. For that purpose we have developed a prototype using the *FeAture Model Analyzer* (FAMA) metamodel as source and the *Eclipse SOA Tool Platform*[2] BPMN metamodel as target metamodel using an *Atlas Transformation Language* (ATL) transformation. It has been published on Eclipse ATL website.[3].

4 Related Work

According to [2], to perform a survey in the software engineering field, we have to define an analysis framework with the following components:(i) *research questions*: How is performed the mapping between feature diagrams and business process models?, and How is documented variability in a BIS context?; (ii) a *search strategy* to select sources: we have searched the bibliography at DBLP, Google Scholar, and ISI Web of Knowledge choosing papers with a higher number of cites; and finally (iii) a *catalogue*: we classify the approaches in those focused on the mapping between feature models and business process models, and those focused on variability representation.

After searching the selected sources, we have found only one proposal that cover our first research question: *How is performed the mapping between feature diagrams and business process models?*. Bae *et al.* [1] proposes a method for deriving a feature model from a business process model in order to provide a process family based on obtaining an intermediate use case representation of the business process. Each feature is considered as a group of use cases, which are associated to perform an specific business activity. The method proposed considers that a set of subprocesses is equivalent to an specific feature. In addition, transformation is performed manually.

On the other hand, regarding to our second research question: *How is documented variability in a BIS context?* only *Process Family Engineering* (PFE) [9] explores the

[2]http://www.eclipse.org/stp

[3]ATL code and specification is available in http://www.eclipse.org/m2m/atl/atlTransformations/#FM2BPMN.

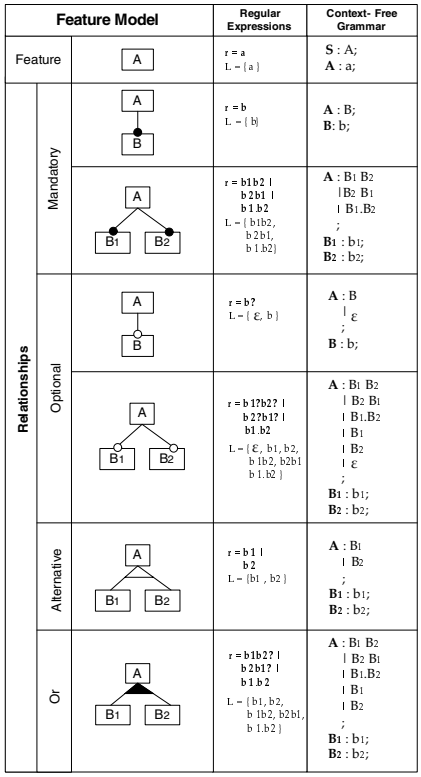

Figure 3. PFE Feature Model and its CFG representation

idea of using feature models for managing variability in a BIS context, but the relationship between these feature models and its products, defined by means of business process models, is not clearly defined as stated in Section 1.

5 Conclusions

We have explored the *Process Family Engineering* (PFE) approach for managing the complexity derived from modeling variant-rich business process models. Thus, we have detected some drawbacks in one of the steps of this modeling methodology, concretely on design phase, identifying ambiguities, maintenance problems and activities performed manually which can be performed automatically. The main motivation of this paper is to solve the identified problems. For that purpose, as shown in Figure 5 in Section 3, we provide a mapping from feature models for representing variability in BIS, whose semantic is significantly different than traditional, to basic structures of business process models, represented using BPMN. The main advantages of our approach are: (i) it is defined as a systematic process, (ii) it provides a maintenance improvement, and (iii) it defines an automatic mapping which maximizes quality level and minimizes error rate.

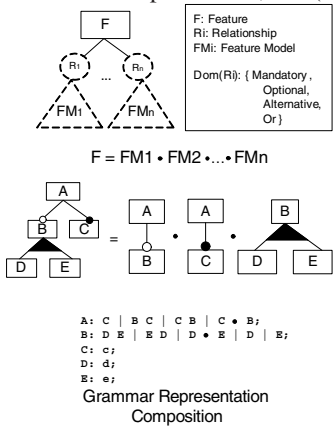

Figure 4. PFE FM Grammar Representation Composition

References

[1] J. Bae and S. Kang. A method to generate a feature model from a business process model for business applications. In *CIT '07: Proc. of 7th IEEE Int. Conf. on Computer and Information Technology (CIT 2007)*, pages 879–884, Washington, DC, USA, 2007. IEEE Computer Society.

[2] L. Barachisio, V. Cardoso, E. Santana, and S. Lemos. A systematic review on domain analysis tools. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement. ESEM07.*, 2007.

[3] D. Batory. Feature models, grammars, and propositional formulas. In J. H. Obbink and K. Pohl, editors, *SPLC*, volume 3714 of *Lecture Notes in Computer Science*, pages 7–20. Springer, 2005.

[4] BPMI. Business process modeling notation BPMN version 1.0 - may 3, 2004. *OMG*.

[5] E. Börger and B. Thalheim. A Semantical Framework for Business Process Modeling. With an Application to BPMN. not published. 2007.

[6] H. Gomaa. Feature dependent coordination and adaptation of component-based software architectures. In *WCAT '07: 4th Workshop on Coordination and Adaptation Techniques for Software Entities*, 2007.

[7] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.

[8] I. Montero, J. Peña, and A. Ruiz-Cortés. Representing Runtime Variability in Business-Driven Development systems. In *Proc. of the 7th Int. Conf. on Composition-Based Software Systems (ICCBSS08)*, 2008.

[9] A. Schnieders and F. Puhlmann. Variability mechanisms in e-business process families. In *Proceedings of BIS '06: Business Information Systems*, 2006.