ATL Transformation: Feature Models for representing runtime variability in BIS to Business Process Model Notation*

Ildefonso Montero, Joaquín Peña, Antonio Ruiz-Cortés
Departamento de Lenguajes y Sistemas Informáticos
Av. Reina Mercedes s/n, 41012 Seville (Spain)
University of Seville
{monteroperez, joaquinp, aruiz}@us.es

- **Eclipse URL**: http://www.eclipse.org/m2m/atl/atlTransformations/#FM2BPMN
- **ISA URL**: http://www.isa.us.es/index.php?module=52&action=singlenews&idN=8
- **Screencast**: http://www.isa.us.es/uploads/screencasts/demo.htm
- **Paper**: *Improving the Design of Highly Variant-Rich Business Process Models* ,sent to 2008 IEEE International Conference on Services Computing SCC 2008 (under review)

*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.

Improving the Design of Highly Variant-Rich Business Process Models*

Ildefonso Montero, Joaquín Peña, Antonio Ruiz-Cortés
Departamento de Lenguajes y Sistemas Informáticos
Av. Reina Mercedes s/n, 41012 Seville (Spain)
University of Seville
{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. In addition, they 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 mapping relationship that clearly defines how to use the information in the FM for obtaining the basic structure of the business process (that needs to be completed manually). 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) [5]. The variability level of average-size BIS is usually highly enough for making the design of this kind of systems a complex task.

*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.

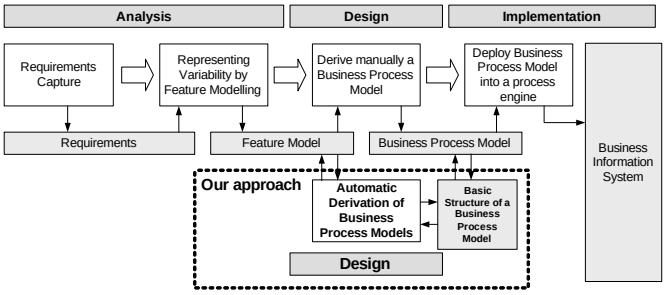

Figure 1. Overview of the PFE approach for modeling BIS and our approach

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. (See Section 2.1 for a more detailed description).

Schnieders *et al.* propose a methodology for designing highly variable business processes [13]. 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 derive 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 pro-

cess 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, just its name.In addition, PFE considers that sometimes a feature represents an activity, sometimes a business process, but without providing an equivalence definition for both artifacts. Thus, we can say that in PFE there not exist a mapping relationship between feature models and business process models. (See Section 3 for more details).

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 [11][7], 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 3, 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 4. As shown in Figure 1, the derivation of the basic structure of a business process model from a feature model will be done automatically. For that purpose we propose in this paper a systematic mapping approach for obtaining a business process model from a feature model. This transformation is achieved by: (i) a feature model redefinition of its semantic, presented in Section 4, which is based on using context-free grammars for describing feature models, detailed in Section 2.1; and (ii) a transformation of this grammar to a finite state machine model, which can be represented by means of a business process model, detailed in Section 4. Figure 2.a sketches the overview of this systematic mapping. In addition, as proof of concepts, we also provide in Section 5 a case study and an implementation, by means of a MDD transformation, of

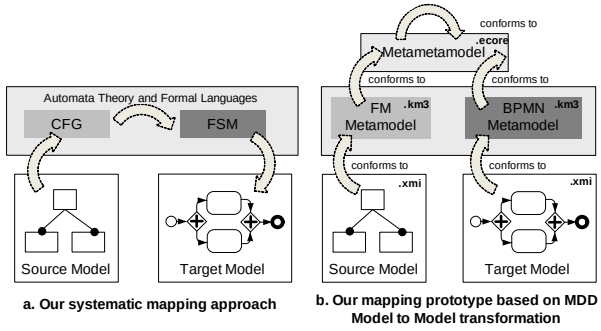

Figure 2. Overview of our approach and our first prototype for automated mapping

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 providing an automatic mapping into the derivation process and eliminating 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.

2 Preliminaries

In order to clarify the context of this paper, in this section we provide a set of definitions and considerations about possible *Feature Models* semantics and *Business Process Model Notation* (BPMN).

2.1 Traditional Feature Models

A *feature* is considered as a characteristic of the system that is observable to the user. *Feature Models* (FM) represents all possible products in an SPL in terms of features. This representation is based on describing which features are present in all the products of the product line, namely core features, and which are considered variable features, that is to say, features that do not appear in all the products. Thus, feature models are considered key artifacts for modeling variability in a SPL.

There exists several notations of FM, such as FODA [10], or J. Bosch [8]. A FM establishes a parental relationship between each feature, as shown in Figure 3, that can be: (i) *Mandatory*: if a child feature node is defined as mandatory, it must be included in every product that contains the

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

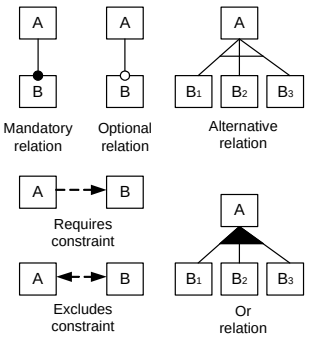

Figure 3. Feature Model Notations

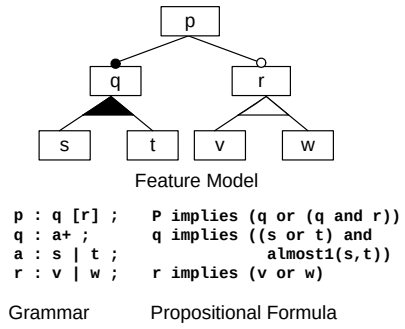

Figure 4. A feature model, its grammar and its propositional formula representation by Batory [3]

parent; (ii) *Optional*: if a child feature node is defined as optional, it can be included or not when its father feature appears in a product; (iii) *Alternative*: if the relationship between a set of children nodes and their father is defined as alternative, only one of the children features could be included in every product that contains the parent; and (iv) *Or*: if the relationship between a set of children nodes and their father is defined as or, one or more of them could be included in every product that contains the parent.

In addition to the parental relations between features, a FM can also contain cross-tree constraints between couples of features. These are: (i) *Requires*: if a feature A requires a feature B, the inclusion of A in a product implies the inclusion of B in such product; and (ii) *Excludes*: if a feature A excludes a feature B, both features can not be part of the same product.

In addition, FM can be described by propositional formulas and grammars. This representation is proposed by Batory *et al.* in [3]. Figure 4 shows the correspondence of a traditional feature model, a context-free grammar and a propositional formula. It defines a product-line where each application contains a feature `q` and optionally `r`, where `q` is an *or* feature: almost one of `s` and `t` can be present in an application; and `r` is an *alternative* feature: only one of `v` and `w` can be present in a member of the family. In this paper, we focus on Batory's work as start point for redefining feature model semantic on BIS context.

2.2 Business Process Model Notation

Business Process Model Notation (BPMN) is defined by OMG in [5] as a flow chart based notation for defining business processes. BPMN provides (i) a graphical notation based on *Business Process Diagram* (BPD), which is a diagram used to design and manage business processes; and (ii) a formal mapping to an execution language: *Business Process Execution Language* (BPEL). Figure 5 depicts the subset of BPMN elements needed for our approach. These elements can be grouped by the following categories:

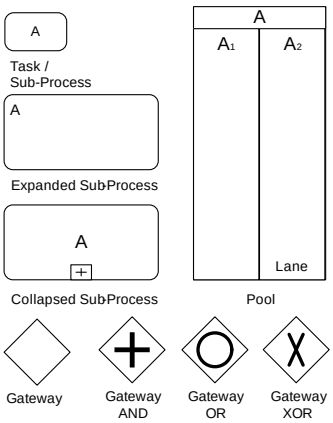

Figure 5. Business Process Model Notation subset

- **Swimlanes**: a set of symbols that allows us to organize the model. This set contains the elements named *Pool* and *Lane*. A pool represents a participant in a process and acts as a container of elements. A lane represents a sub-partition of a pool and are used to organize and categorize activities.

- **Flow objects**: a set of symbols that represents the core elements of a business process model. Usually this elements are contained in a swimlane. This set contains the elements named *Task*, *Event* and *Gateway*. A task, also called activity or sub-process, is the basic element of a work in an organization, it can be atomic or non-atomic. Event is something that happens in our process that fires the execution of one or more activities. There exists a lot of events grouped by: *start*, *intermediate* or *end* event, as for example *timer* or *message* events. A gateway is used to control the divergence or convergence of flows as logic doors. In this paper we

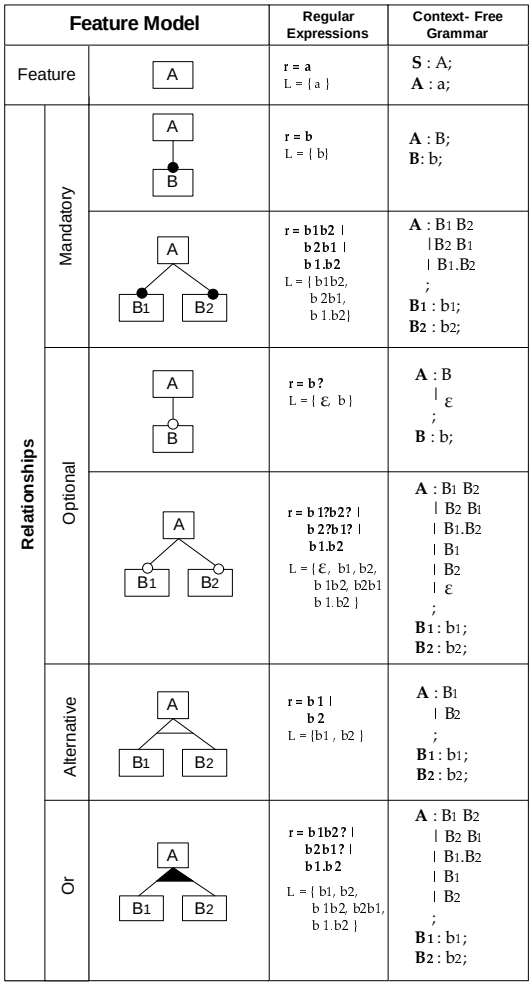

Figure 6. PFE Feature Model and its CFG representation

focus on three different gateways: (i) *And*: which defines that all the subprocesses controlled by this gateway must be completed for performing a task, (ii) *Xor*: which defines that only one subprocess controlled by this gateway must be completed for performing a task, and (iii) *Or*: which defines that almost one subprocess controlled by this gateway must be completed for performing a task. The specification of BPMN does not provide any constraint about the order of performing this subprocesses in this situations, it can be done as a sequence or parallel.

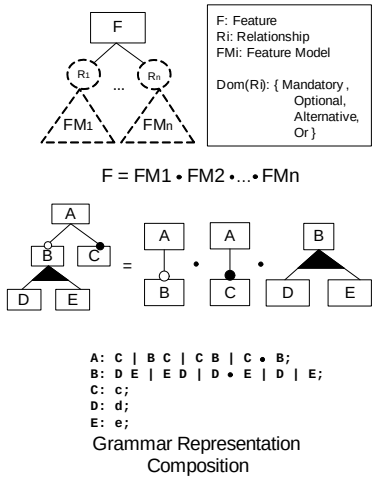

Figure 7. PFE FM Grammar Representation Composition

3 Definition of a New Semantic for Feature Models

3.1 Redefining 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 establishes 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). This kind of information it is not present into traditional proposals for SPL modeling.

For defining a CFG for PFE feature diagrams, we consider all the products of each feature model, hereafter named *artifacts*, as a language which can be defined by means of a regular expression [9]. Let A be a complex process defined by means of a feature model which establishes a mandatory relationship with two subprocesses B_1 and B_2. Figure 6 shows this FM. There exists three possible alternatives for performing A:

- B_1, B_2: In order to perform an activity named A, it is necessary to perform the sequence of subprocesses B_1 and B_2.
- B_2, B_1: In order to perform an activity named A, it is necessary to perform the sequence of subprocesses B_2 and B_1.

- $B_1 \bullet B_2$: In order to perform an activity named A, it is necessary to perform subprocesses B_1 and B_2 in parallel.

In addition, PFE considers that sometimes a feature represents an activity, sometimes a business process, but without providing an equivalence definition for both artifacts. Thus, we can say that in PFE there not exist a mapping relationship between feature models and business process models. In our approach, we are going to take into account the following considerations:

- 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.

3.2 Feature Model Grammars

Once feature models are redefined for BIS context, we are going to reuse Batory's grammar representation [3] for proposing a new grammar. Thus, as shown on Figure 6, the language which defines this artifact is:

$$L_{a-mand} = \{b_1b_2, b_2b_1, b_1 \bullet b_2\}$$

Now, consider an artifact with an optional relationship instead of a mandatory, as shown on Figure 6. Let ε an empty set, the language which defines it is:

$$L_{a-opt} = \{\varepsilon, b_1, b_2, b_1b_2, b_2b_1, b_1 \bullet b_2\}$$

In addition, if we consider an artifact with an alternative relationship instead of an optional, as shown on Figure 6, the language which defines it is:

$$L_{a-alt} = \{b_1, b_2\}$$

And finally, if we consider an artifact but with an or relationship instead of an alternative, as shown on Figure 6, the language which defines it is:

$$L_{a-or} = \{b_1, b_2, b_1b_2, b_2b_1, b_1 \bullet b_2\}$$

Thus, a regular expression of these languages can be obtained by means of operations of automata and formal languages theory defined in [9]. Let r_{mand} be the regular expression which defines L_{a-mand}, let r_{opt} be which defines L_{a-opt}, let r_{alt} be which defines L_{a-alt}, and let r_{or} be which defines L_{a-or}, they can be defined as follows:

$$r_{mand} = b_1b_2|b_2b_1|b_1 \bullet b_2$$

$$r_{opt} = b_1?b_2?|b_2?b_1?|b_1 \bullet b_2$$

$$r_{alt} = b_1|b_2$$

$$r_{or} = b_1b_2?|b_2b_1?|b_1 \bullet b_2$$

where ? represents the operator *one-or-zero* token occurrences defined in [9].

Once regular expressions are obtained, a context-free grammar definition of these regular expressions can be described. Figure 6 sketches the equivalence of a feature and its relationships in terms of regular expressions and context-free grammars. Parallel definitions are described by means of $\bullet$ character.

In addition, each possible composition between two or more different artifacts is resolved by means of parallel decompositions. Figure 7 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 obtained by means of applying $\bullet$ operator to three simplified feature models CFG representations defined previously on Figure 6. Obtained CFG for composed feature model is shown on Figure 7.

4 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[9]. Applying this mapping we provide a FSM representation of the feature model grammars presented previously. Figure 8 presents each feature model grammar with its FSM correspondence.

In addition, BPMN can be represented by means of FSMs[6]. 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.

Thus, as stated previously in Section 2.2, the specification of BPMN 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, presented in Section 2.2 too. Thus, the BPMN gateways are feasible to be used for implementing proposed finite state machines behavior, as shown in Figure 9 that presents the equivalence between each of FSMs and its representation using BPMN. As stated previously, we have developed, as a proof of concepts, an automated support for our mapping approach by means of MDD transformations. For that purpose we have developed a first step towards tooling this mapping using the *FeAture Model Analyzer* (FAMA) metamodel as source and the *Eclipse SOA*

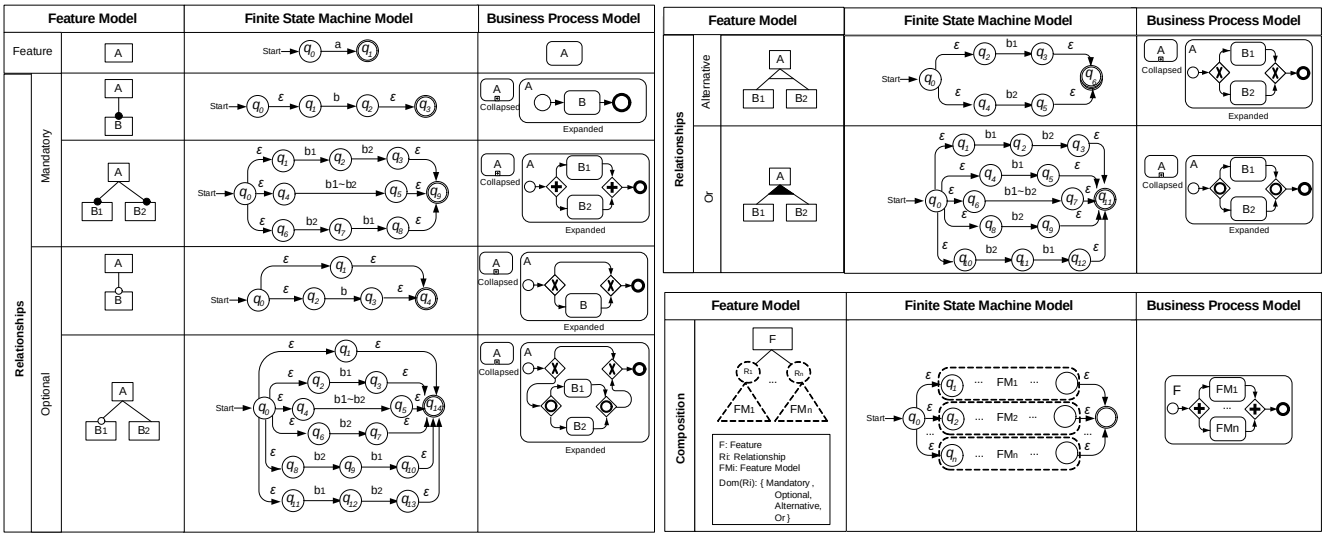

Figure 9. Feature Model to BPMN Mapping and Composition Catalog Proposed

Tool Platform[2] BPMN metamodel as target metamodel using an *Atlas Transformation Language* (ATL) transformation. It has been published on Eclipse ATL website.[3].

5 Case Study: Airline Travel Agency

We present an example of a business process of an airline travel agency reservation system. It has been taken from the WSCI specification [15] and a derivation of this example, for defining an hotel booking process has been used for illustrating the PFE approach in [14].

We use this example to illustrate how a BPMN diagram can be obtained from a PFE feature model using our approach. Figure 10.a sketches the feature model of our example which is partially similar than presented in [14]. It shows a *Booking Process* which is composed by two alternatives subprocesses, *Cancel Reservation* or *Booking* based on *Performing Booking* and *Send Tickets*. Figure 10.b presents the airline reservation process defined using BPMN obtained automatically by means of our proposal.

With this approach we can provide the needed infrastructure for building business information systems product lines. Figure 11 sketches how to build two different airline agencies business process models. Figure 11.a presents a PFE feature model of a simplified *Airline Travel Agency*, which is composed by three different subprocesses: *Booking Process* which is the same that is presented previously, *Inform* which is composed by two different subprocesses: *Flying Schedules*: it contains all the subprocesses needed to inform the flying schedules to the airports, and *Delayed*: it express how to inform a delayed fly; and *Extras* which is composed by *Restaurant*: it defines the restaurant on fly subprocess, and *Travel Card*: it contains all the subprocesses related to manage travel cards.

For describing how to perform a product line of airline travel agencies, first we calculate the commonality of this product line, using the operation defined in [4]. We introduce a commonality threshold for introducing the complete *Booking Process* into the core of our product line. This commonality threshold must be calculated empirically for each product, as shown in [12]. Once commonality is calculated, we derive two different products selecting different features/subprocesses in an stage configuration phase. We derive two different products: *Iberia* and *Ryanair*, as shown in Figure 11.a. After that, we apply our mapping proposal and obtain a core process framework adaptable to each product, as shown in Figure 11.b. This core is present in each product member of our family: *Iberia* and *Ryanair*, and the selected features/subprocesses corresponding with each product are present too and properly connected with the core.

Finally, once this abstract initial structure of a business process model is provided, process engineers need to refine it manually for deriving a concrete business process model. This refinement is focused on providing an specific business process model for each child feature: *Restaurant*, *Travel Card*, *Flying Schedules*, *Delayed*, *Performing Booking* and *Send Tickets* in this case study. For example, for *Delayed*, it is needed to specify the complete process when it is performed, specifying lanes, pools, data objects, etc. In addition, it is needed to specify another relevant details such as guards in the gateways or start/end events (context-

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

[3] ATL code and specification is available in http://www.eclipse.org/m2m/atl/atlTransformations/#FM2BPMN. We also provide an screencast of our case study transformation.

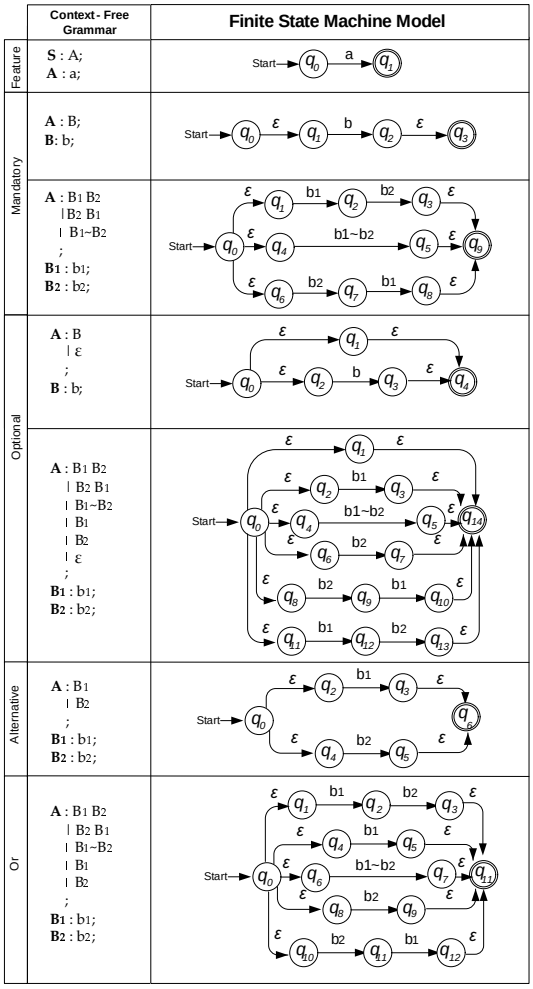

Figure 8. Mapping Grammars to Finite State Machines

dependent information).

6 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 appearing at DBLP, Google Scholar and ISI Web of Knowledge choosing those 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.

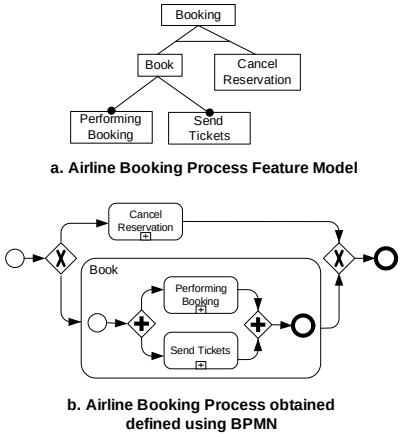

Figure 10. Case Study: Airline Travel Agency Booking Process

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?*, as shown previously in Section 2, only *Process Family Engineering* (PFE) [13] explores the 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.

7 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 9 in Section 4, we provide a mapping from feature models for representing variability in BIS, whose semantic is significantly different than

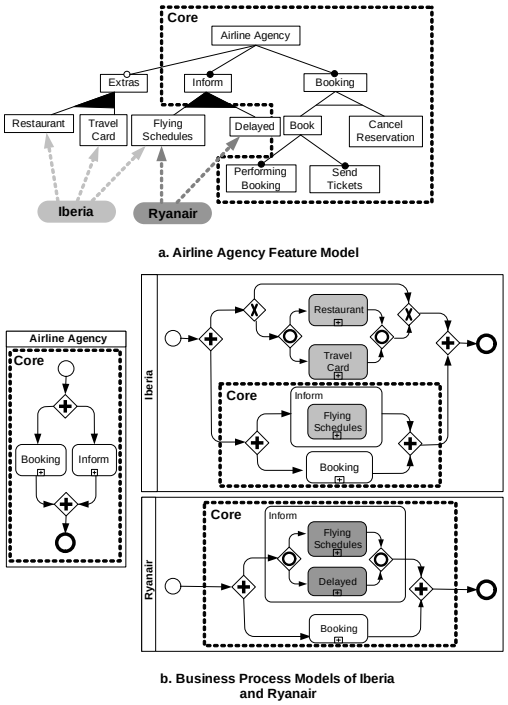

Figure 11. Case Study: Building an Airline Travel Agency Business Family

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.

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: Proceedings of the 7th IEEE International Conference 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] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 3520:491–503, 2005.

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

[6] E. Börger and B. Thalheim. A Semantical Framework for Business Process Modeling. With an Application to BPMN. Available at http://www.di.unipi.it/ boerger/CourseMaterial/Bpmn.pdf. 2007.

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

[8] J. V. Gurp, J. Bosch, and M. Svahnberg. On the notion of variability in software product lines. In *WICSA '01: Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01)*, 2001.

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

[10] K. Kang, S. Cohen, J. hess, W. Novak, and S. Peterson. Feature-oriented domain analysis FODA feasibility study. CMU/SEI-90-TR-21. Technical report, Carnegie Mellon University. SEI, 1990.

[11] I. Montero, J. Peña, and A. Ruiz-Cortés. Representing Runtime Variability in Business-Driven Development systems. In *Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS08)*, 2008.

[12] J. Peña, M. G. Hinchey, A. R. Cortés, and P. Trinidad. Building the core architecture of a nasa multiagent system product line. In *AOSE*, volume 4405 of *Lecture Notes in Computer Science*, pages 208–224. Springer, 2006.

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

[14] J. Schulz-Hofen and S. Golega. Generating web applications from process models. In *ICWE '06: Workshop proceedings of the sixth international conference on Web engineering*, page 6, New York, NY, USA, 2006. ACM.

[15] W3C. Web service choreography interface (WSCI) 1.0, nov. 2002. www. w3.org/tr/wsci.