Business Family Engineering. Managing the Evolution of Business Driven Systems.*

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

Nowadays most companies in whichever field have a software system that helps managing all the aspects of the company, from the strategic management to daily activities.

Companies are in continuous evolution to adapt to market changes, and consequently, the Information Technology (IT) infrastructure that supports it must also evolve. Thus, software companies are currently supporting this evolution with ad hoc *techniques.*

We think that, as it is being done for traditional software systems (non-oriented to business process) in the software product line (SPL) field, institutionalized techniques for performing a systematic reuse of business processes across different businesses can be introduced.

In this paper, we propose to adapt SPL techniques, oriented to reuse software, to Business-Driven Development (BDD), oriented to reuse processes, across different businesses; we call this proposal Business Family Engineering (BFE). We present a first approach to build a SPL of BDD systems that evolves at runtime.

1 Introduction

Nowadays, software systems supports most of the activities of a company due to it improves their daily work and their strategic management. Thus, Information Technology (IT) infrastructure must evolve to adapt companies to the continuous evolution of markets. Currently this evolution is supported by *ad hoc* techniques.

Business-Driven Development (BDD) is a research field, which is the focus of this paper, that tries to solve this problem designing software systems starting from the business processes of the companies.

In BDD, business processes are designed to be executed over a process engine. Of course, current process engineers redesign the processes every time that is needed using *ad hoc* techniques to maximize the level or reuse from one version to another. In addition, when dealing with several businesses in a certain domain, many common features can be found, and reuse across businesses is also exploited.

Software product lines (SPL) systematizes the reuse across the set of similar products that a software company produces. We think that such systematization can be also applied in BDD improving the results achieved by current *ad hoc* techniques.

The main goal of SPL is to obtain a reduction of the overall development costs and times for the products derived from the product line [7]. Basically, in SPL we obtain a set of software systems, called products. Each product contains common functionalities, and a set of specific features that differentiates one product from another.

The idea of applying SPL to BDD, has been explored by Schnieders *et al.* in an approach called *Process Family Engineering* (PFE) [12]. Basically, PFE follows the SPL philosophy for managing the evolution of the business process of an unique business, thus, managing only one software system. That is to say, each product in PFE represents an evolution of the process (at runtime).

Thus, PFE may be the solution to manage the evolution of the business process of a company, but no techniques has been described to systematize the reuse across several businesses. In addition, the models proposed by PFE approach are not expressive enough for documenting dynamic evolutions on runtime.

The contribution of this paper is twofold. On the one hand, we propose a new research field called *Business Family Engineering* (BFE) that is oriented to provide the techniques and methods needed to build a product line of BDD systems. Roughly speaking, BFE is a product line of prod-

*This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472).

uct lines, where the products of the first SPL are abstract processes that acts as assets for the second product line where evolution of this processes are managed by means of PFE. We focus our efforts on minimizing the implicit risk of combinatorial explosion, and on the other hand, on solving some deficiencies we have detected in current approaches for PFE regarding the expressiveness of models for documenting the evolution.

This paper is structured as follows: Section 2 presents the related work and motivation of our work; Section 3 presents a description of BFE and its relationships with PFE; Section 4 and 5 presents an overview of the software process and how to obtain the core process framework of the business family; Section 6 presents how to obtain the products of our approach and how to manage the evolution of the products of our business family; and finally, in the last section, we draw the main conclusions and the future research lines needed to enable a business family infrastructure.

2 Related work and motivation

In this section, we present the related work needed to understand and to motivate this paper. Regarding BFE, to the best of our knowledge, there not exists any approach that provides all the techniques needed for enabling the mass production of BDD systems that evolves at runtime. In [9] we present a first step to the definition of a SPL of BDD systems that evolves at runtime, oriented to reuse processes across different businesses. Roughly speaking, BFE is a product line of product lines, where the products of the first SPL are abstract processes that acts as assets for the second product line where the evolution of each business is managed. In this paper, we extend the ideas presented in [9].

In SPL [8, 10] a product is composed of a set of common features and a set of variable features. Common features appears in all products and variable features appears under demand of products's consumers. Observing a certain product of a SPL, although it is described as a set of fixed features, some features can be in use in a certain moment and some not. Thus, in SPL the evolution of the system at runtime, also called runtime variability, is not taken into account in the feature model.

PFE approach is given by PESOA research group from Hasso Plattner Institute for IT Systems Engineering in [1]. In the same way that SPL approach provides all the techniques needed for enabling the mass production of software in a certain application domain, PFE provides all the techniques needed for enabling the mass production of processes in a certain business. Each product represents a set of processes enabled at a certain moment of the execution. In PFE we obtain only one software system that evolves at runtime, where the features are processes. Every process evolution represents a product that contains a subset of features, but the PFE system contains all the features. PESOA uses feature model for representing the set of processes contained into a business, and *Business Process Model Notation* (BPMN) to represent an specific process. BPMN is defined by OMG in [6] 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).

PESOA introduces an extension of BPMN to represent variability in a process [11]. Figure 1 shows an example of a feature model of an E-shop business and an extended BPMN to represent a checkout process. Feature model represents all the processes contained into the E-shop business. Roughly speaking if a process is denoted as *mandatory* it must be present in all the possible configurations of the business, for example: *Checkout*. Each process is represented using BPMN with the extensions for variation points and the extensions needed to represents the relationship between features and processes. As shown in Figure 1 the variation point is represented as a puzzle-piece icon and for feature and processes relationship we see that *Calculate Sum* can be implemented as a sequence of *Calculate Sum* and *Calculate Discount* sub-processes that is applied when the feature *Personalized Shopping Cart* is selected.

Thus, the main motivation of our work is that, although PFE may be the solution to manage the evolution of the business process of a company, no techniques has been described to systematize the reuse across several businesses and proposed models are not expressive enough for documenting dynamic evolutions on runtime due to feature models are not designed to support runtime variability.

As can be observed in Figure 2, where we depict how SPL and PFE products are generated, SPL products are implemented by software artifacts that for each of them there exists a feature selection phase that generates the final products (a set of core and variable features). PFE products are implemented by processes. For each product, the system can evolve to another product increasing or decreasing the variable set of features thus, each product is a software system based on processes.

3 Preliminaries

In this section, we present a rigorous description of business families describing also its relationship with process family engineering.

3.1 BFE Definition

Business Family Engineering (BFE) can be defined as: *a set of software systems driven by business processes (here-*

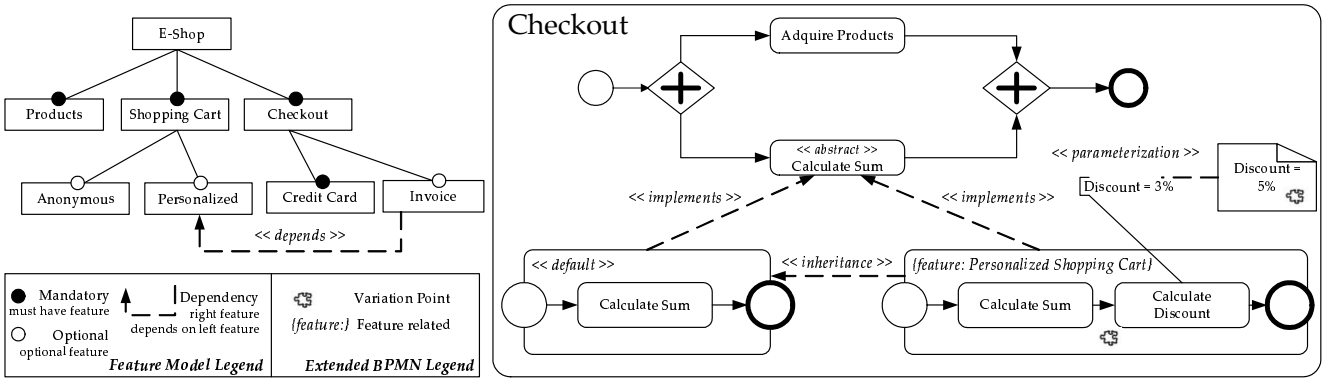


Figure 1. Example of feature model and extended BPMN by PFE approach

after business) where each product of the family has a set of common processes and a set of variable processes. The formal definition of BFE can be represented as follows: Let BF be a Business Family that contains a set of *n > 1* businesses

$$BF = \{B_1, B_2, ..., B_n\}$$

where each B represents a business. Each business B can be defined as a set of processes (denoted with P). Thus, each B_i in BF can be defined as follows:

$$B_i = \{P_1, P_2, ..., P_k\}; k > 0; 1 \leq i \leq n$$

Given this it holds there exists a set of common processes between whichever set of businesses. Let B_i and B_j be two businesses contained in BF where $1 \leq i, j \leq n$:

$$B_i \cap B_j \neq \phi$$

Thus, we can say that a business family can be also defined as a set of core and variable processes/features. We assume that there exists a binary relationship between features and processes, thus a feature must not represent a set of processes. Let *CF* be the set of common processes or features and let *VF* be the set of variable features,thus *BF* can be defined as follows:

$$BF = (CF, VF)$$

In that way, a business B_i is defined formally as a tuple containing all the CF and a subset of VF denoted as SVF:

$$B_i = (CF, SVF \in VF)$$

3.2 Description of the integration of BFE with PFE

Figure 3 depicts the integration between BFE and PFE. As defined before, each business contains a set of core processes, CF, and a set of variable processes, VF. However, in PFE the processes/features appears and disappears at runtime. As shown before, each configuration of the set of processes enabled at certain moment represents a product. Thus, we can say that the CF of a BF are always enabled at runtime, but the set of processes in VF is not fixed at runtime.

Thus, as PFE defines, we can set up a product line that takes into account this runtime variability. For formalizing these concepts we should redefine each business B of a BF as:

$$B = (CF, SVF \in VF, F_\Delta :$$
$$: t, \{Feature \times ... \times Feature\} \mapsto$$
$$\mapsto \{Feature \times ... \times Feature\})$$

where F_Δ is a function that given an instant t transform the set of SVF_t into the new set of variable features of the following time instant $t+1$, that is to say SVF_{t+1}, formally:

$$F_\Delta(t, SVF_t) = SVF_{t+1} \in VF$$
$$\bullet SVF_t \neq SVF_{t+1}$$

Figure 4 sketches a graphical representation of F_Δ, where it is represented the transformation of SVF_t into SVF_{t+1}. In an instant t there exists an specific set of SVF_t for business B_j that evolves in instant $t+1$ to another different set SVF_{t+1}. The evolution is defined by the F_Δ function in t.

4 Overview of the software process of our approach

In this section, we describe the software process and modeling elements needed to develop a business family.

Figure 5 shows the software process of our approach in SPEM[1]. In this software process there exists two main activ-

[1]http://www.omg.org/technology/documents/formal/spem.htm

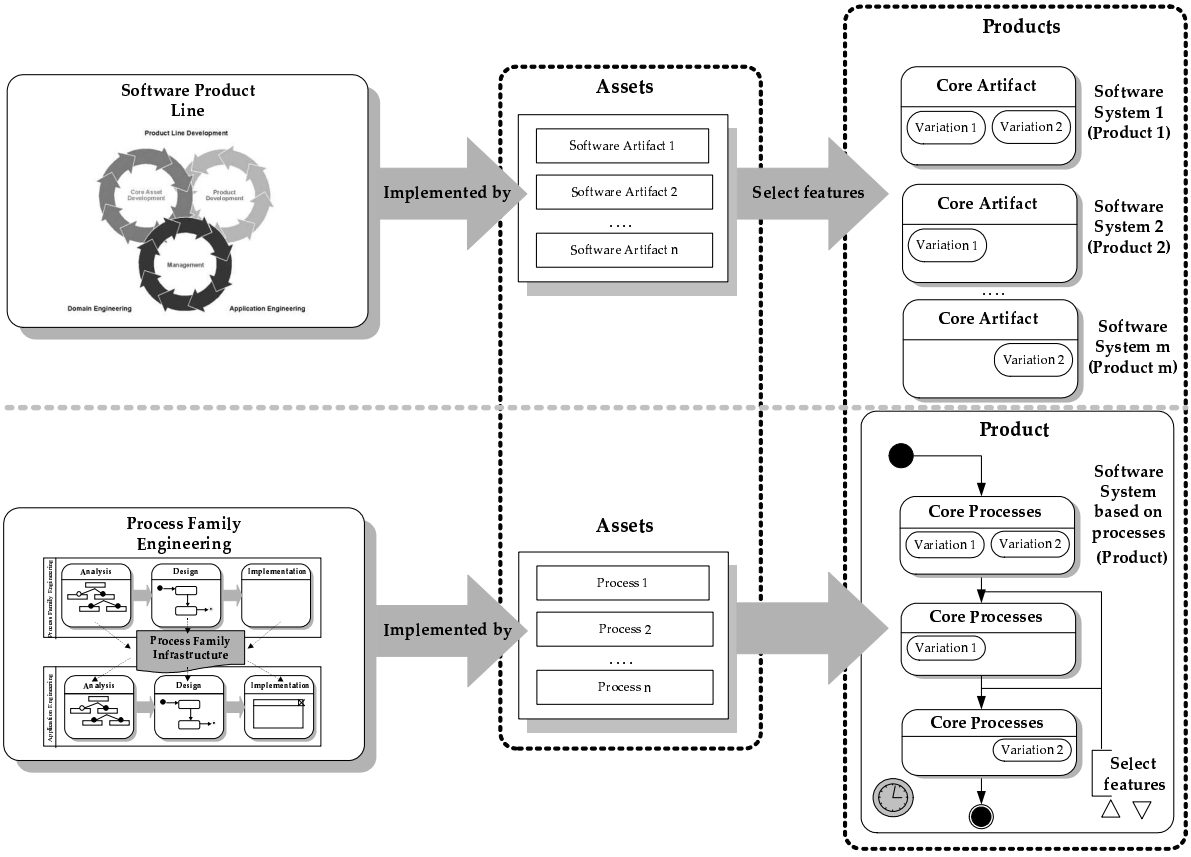


Figure 2. SPL and PFE approaches

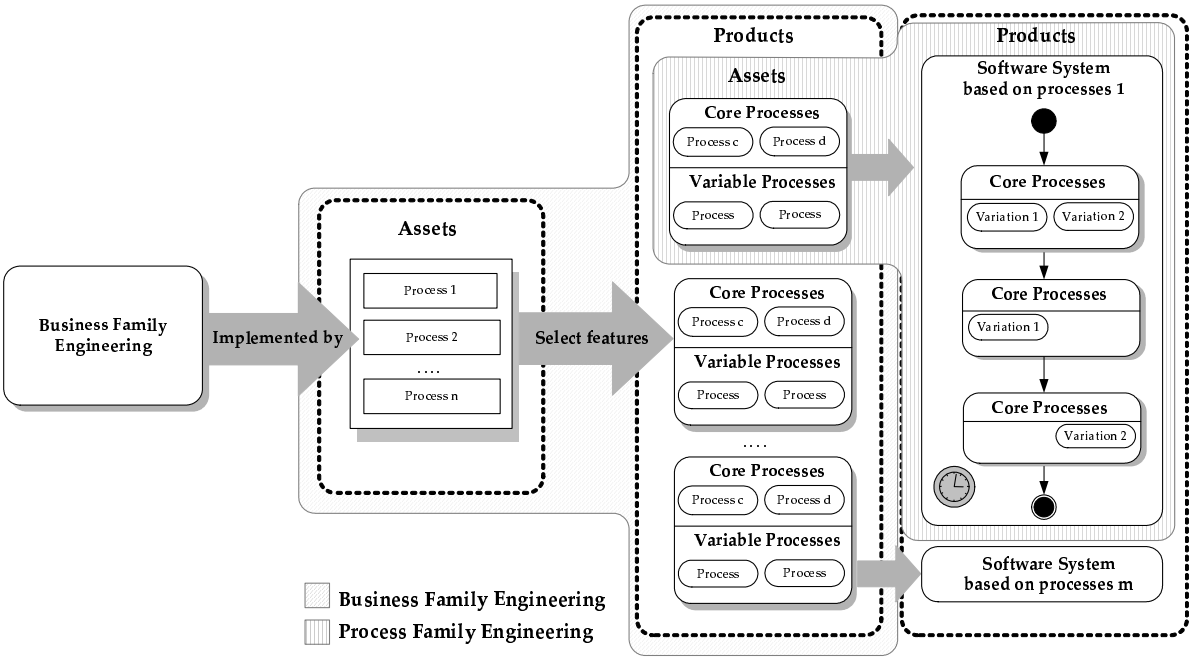


Figure 3. BFE approach

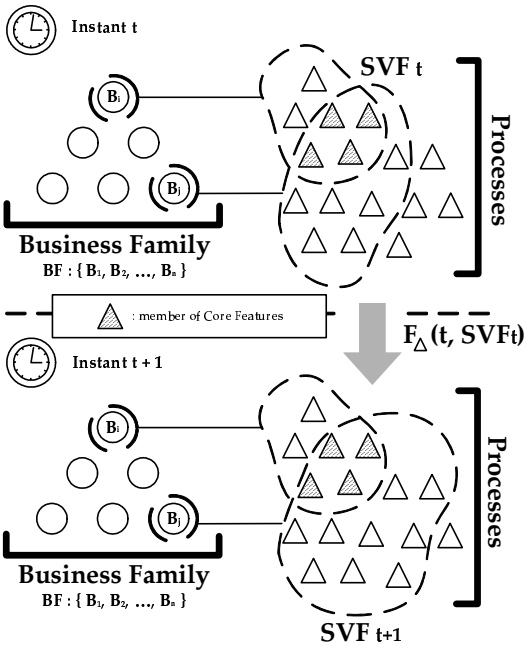


Figure 4. Evolution of a business into BFE

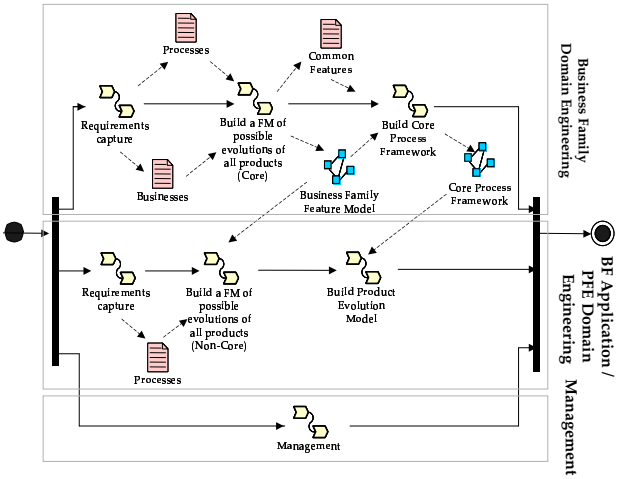


Figure 5. Overview of the software process of BFE in SPEM notation

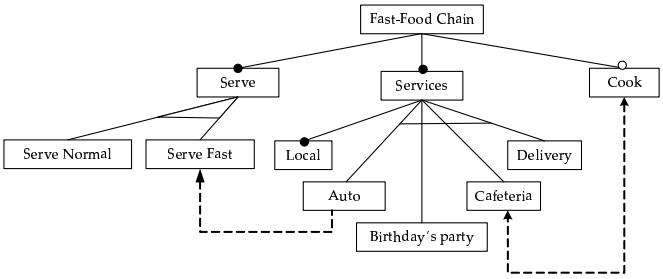


Figure 6. Case Study: Fast Food Restaurant Chain

ities: *Business Family Domain Engineering* (BFDE), where we build the BFE core architecture and the assets and *Business Family Application Engineering* (BFAE), where we obtain specific products.

Roughly speaking, in BFDE we build the feature model of the business family and we build a core process framework.

In BFAE, we build a feature model for each business that represents the evolution without using the core features of the business family (thus, we minimize the risk of combinatorial explosion) and we build the product evolution model for the business.

It is important to say that our scenario is by now limited to binary relationships between features and processes, in other words, a feature can not represent a set of processes.

We have identified two different ways to build a business family: *top-down* and *bottom-up*. In *top-down* approach we define the set of businesses and processes from zero and apply the normal sequence of BFE software process. In *bottom-up* approach, we can not apply the normal sequence of the software process defined due to we have a set of businesses or processes defined in PFE previously to apply BFE software process. In this paper we only focus in *top-down*.

5 Obtaining the core process framework of the business family

To show our approach we use a fast food restaurant chain business family case study. In this business family there exists several businesses that share common processes as cook burgers and other specific for a concrete business,such cafeteria service. To show our approach we build a BFE system of this example and derive two businesses from the family: *McDonald's* and *Burger King*.

Figure 6 shows an example of a business family of a fast-food restaurant chain. This feature model depicts a simplified set of processes contained into a fast-food restaurant: *Cook*, *Serve* that depending on the moment *Serve* can be *Serve Fast* or *Serve Normal*, and *Services*, that represents all the possible services given by restaurants: *Local*, that represents the basic service, *Auto*, *Birthday´s party*, *Cafeteria* and *Delivery*, that are special services that depends on different situations or moments at runtime.

To obtain the core processes of this business family we apply the activities of *Business Family Domain Engineering* defined in Section 4. We have used FAMA [5, 3, 4] software tool to build the feature model of the business

family. FAMA is an academic tool that provides a framework for the automated analysis of feature models integrating some of the most commonly used logic representations and solvers proposed in the literature. Figure 7(a) shows the feature model of our case study in FAMA. After that, we calculate the core processes of all the businesses applying analysis operation to feature model defined in [2]. The result is the set of features selected in 7(a). Then, we build a feature model of possible evolutions of all products taking into account only core features. Figure 7(b) shows this feature model modeled with FAMA. Finally we build core process framework model in BPMN as shown in Figure 8. It is done by combination of each process contained in core feature model. We are using BPMN extended notation of PESOA approach presented in Section 2. In this example there exists an abstract process, *Serve in Local*, that can be implemented by two different ways, *Serve Normal in Local* and *Serve Fast in Local*, and these ways are processes that contains subprocesses defined in BPMN too. Thus, we are able to develop a business family applying *Business Family Application Engineering* activities. In this example we build two fast-food restaurants: *McDonald´s* and *Burger King*. Figure 9 sketches both feature models, that are needed to identify the relationships of variable features and the core process framework built.

6 Building BFE products and managing the evolution

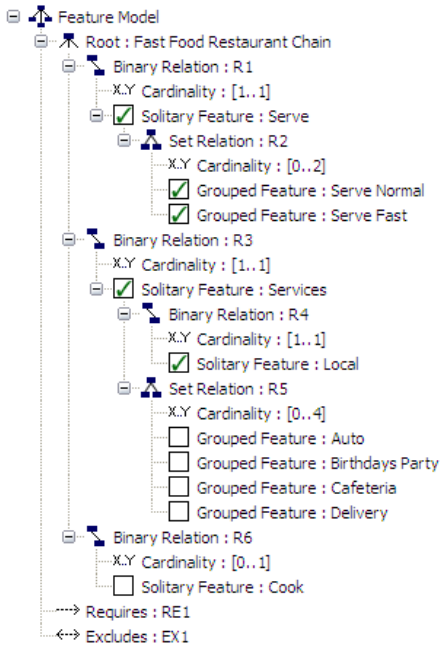
BFE approach introduces two different point of views of a business: *static* and *dynamic*. In *static* point of view, a business is an entity that contains a set of processes; and in *dynamic*, a business is an entity that evolves on runtime. BFE must provide models to represent these point of views.

6.1 Motivation

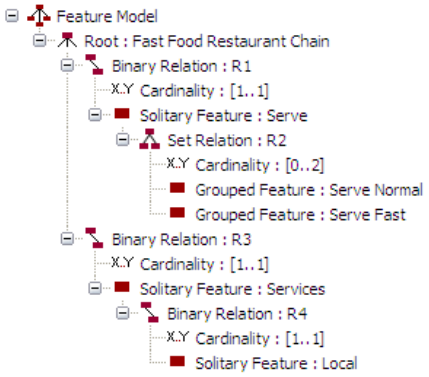
As shown previously, a certain business in a business family is described as:

$$B = (CF, SVF \in VF, F_{\Delta}(...)$$

Feature modeling is a good choice to represent the commonalities and differences of a set of specific business processes. Thus, in PFE feature models are used to represent which features are variable and which do not. From this, using automated analysis of feature models, a the set of common features (CF) and (VF) can be obtained easily [2]. However, the feature model cannot establish the order of apparition of them represented as F_{Δ}, because this models are not expressive enough for documenting dynamic evolutions on runtime. As a result, we can conclude that for representing a certain business B_i in a business family BF graphically, we



(a) Fast food restaurant feature model



(b) Core processes feature model

Figure 7. Fast food restaurant feature models modeled in FAMA

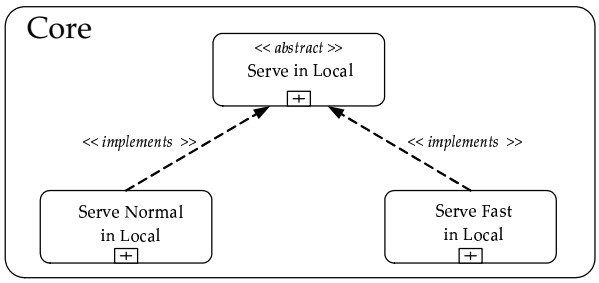


Figure 8. Core processes in BPMN

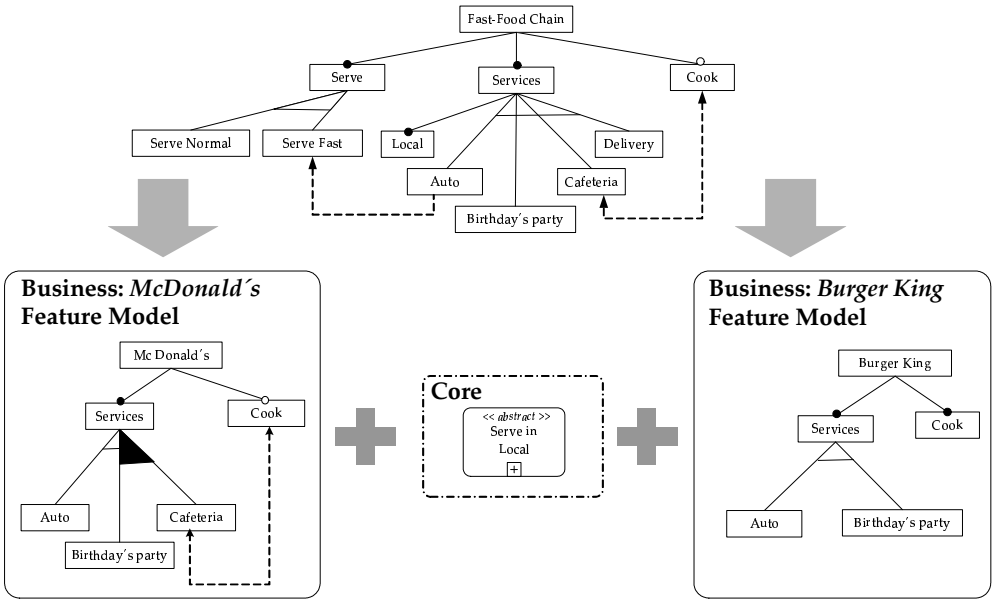


Figure 9. McDonald's and Burger King Feature Models

have to decorate feature models with a graphical notation that represents F_{Δ}. We call this model *Product Evolution Model*.

6.2 Product Evolution Model

Product Evolution Model is represented with a BPMN state machine where each state represents a product and each evolution between two or more states is an application of F_{Δ} function. Each process contains a BPMN state chart that represents how is done all the processes that represents the configuration of the business at this moment. This BPMN model that can be deployed and simulated into a business model software engine.

Figure 10 presents McDonald's *Product Evolution Model*. It sketches that in every runtime instant t there exists a different SVF selected that represents an evolution. In this example, on a time instant t *McDonald's* open his cafeteria service, thus, there exists in parallel two different processes: *Serve in Cafeteria* and Core (*Serve in Local*). When *McDonald's* close his cafeteria service on time instant $t+1$, F_{Δ} function is applied and an evolution is done to another state composed by only Core processes. After that *McDonald's* open his Auto-Service and a new evolution in runtime is applied for $t+2$ time instant.

Thus, the main benefit of this model is that we can see a set of business processes that evolves on runtime, and we can compare this with other of our family and take decisions about our business based on evolution of all the businesses of our family.

7 Conclusions

The main conclusion of this paper is that BFE is feasible. Its main benefit is that software companies that provide BDD solutions, can reuse process building a product line where a set of common processes is extended with the processes needed for each customer in a systematic way, thus reducing costs (in time and money) and improving the quality of their products, since they are tested for several clients.

Another important conclusion is that PFE cannot be used directly for BFE. PFE provides techniques to manage the evolution of the business process of a company based on SPL ideas, however all the variable features of the process are added to the final software system, enabling or disabling them at runtime. While BFE needs techniques that allow adding only those features that the customer requires. In addition, techniques used in PFE presents drawbacks. Mainly, feature models are used to represent runtime variability, while these models are devoted to static variability.

As PFE is quite valuable for runtime variability, we conclude that BFE must be integrated with PFE, but a number of problems arise. Mainly, as a result of having a product line (BFE) of product lines (PFE) it occurs an state explosion that hinders the feasibility of this approach. Possible solutions to the identified problems are: (i) using one feature model to BFE and one feature model for each product, obtaining an hypercube structure, that represents all the possible variations of products, (ii) using one feature model to BFE and one feature model for all the possible products introducing a sugar syntax to feature model for enabling it for represent changes at runtime and (iii) using one feature

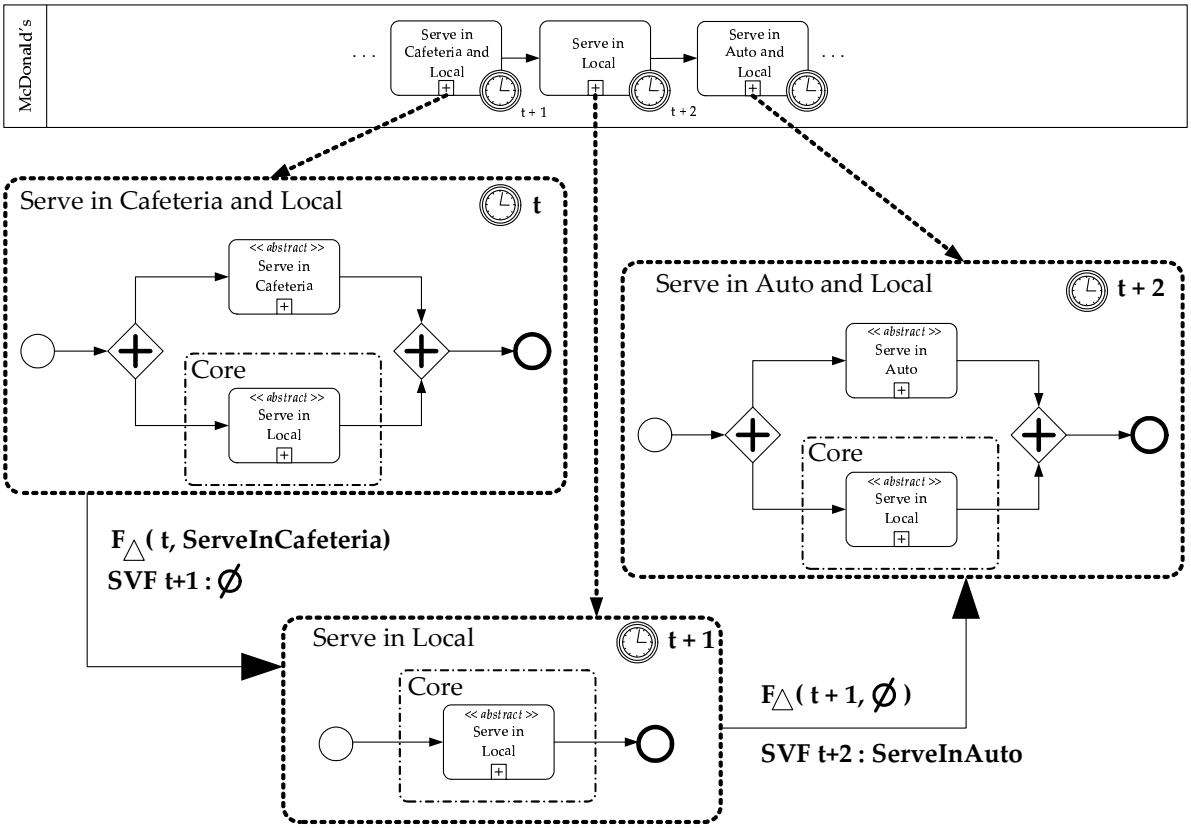


Figure 10. McDonald's Product Evolution Model

model to BFE with annotations that represents changes at execution time, that is the alternative used in our approach where the annotation is the product evolution model.

References

[1] J. Bayer, W. Buhl, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, E. Richter, A. Schnieders, J. Weiland, and M. Weske. Process family engineering. modeling variant-rich processes. Technical report.

[2] D. Benavides. *On The Automated Analysis of Software Product Lines using Feature Models. A Framework for Developing Automated Tool Support*. PhD thesis.

[3] D. Benavides, A. Ruiz-Cortés, P. Trinidad, and S. Segura. A survey on the automated analyses of feature models. *XV Jornadas de Ingeniería del Software y Bases de Datos,JISBD 2006*, 2006.

[4] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. A first step towards a framework for the automated analysis of feature models. In *Managing Variability for Software Product Lines: Working With Variability Mechanisms*, 2006.

[5] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, 2007.

[6] BPMI. Business process modeling notation (bpmn) version 1.0 - may 3, 2004. *OMG*.

[7] P. Clements, L. Northrop, and L. M. Northrop. *Software Product Lines : Practices and Patterns*. Addison-Wesley Professional, August 2001.

[8] G. Halmans and K. Pohl. Communicating the variability of a software-product family to customers. *Inform., Forsch. Entwickl.*, 18(3-4):113–131, 2004.

[9] I. Montero, J. Peña, and A. Ruiz-Cortés. Business Family Engineering. Does it make sense? In *Proceedings of the First Workshop: Procesos de Negocio e Ingeniería del Software (PNIS07)*, pages 34–40, 2007.

[10] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, September 2005.

[11] F. Puhlmann, A. Schnieders, J. Weiland, and M. Weske. Variability mechanisms for process models. Technical report.

[12] A. Schnieders and F. Puhlmann. Variability mechanisms in e-business process families. In *BIS*, pages 583–601, 2006.