A Methodological Framework for Obtaining the Core Architecture of Business Information Systems Families.*

Ildefonso Montero, Joaquin Peña, and Antonio Ruiz-Cortés

Dept. of Languages and Computer Systems, University of Seville (Spain)
{monteroperez, joaquinp, aruiz}@us.es

Abstract. Nowadays large organizations are the result of the interrelation of many business units which tend to be managed based on its business processes. The development of the *Information Technology* (IT) infrastructure that supports these organizations is a complex task, due to the existence of different versions of the same process, namely *core process*, tailored in terms of each unit that executes it. Thus, organizations lost the matching between each version and the original. It implies problems in the maintenance of process specifications, that drives to an inaccurate execution of the business strategies of the organization. The introduction of *Software Product Lines* (SPL) techniques into the development of *Business Information Systems* (BIS) is expected to become a new development paradigm, of what we call *Business Families*, maximizing reuse and dealing with variability on process definitions. Current SPL-based proposals do not solve the identified problems, because the proposed design of the reusable core specification is not performed in a systematic way that maintains its traceability with derived processes. The main contribution of this paper is to provide a methodological framework that taking into account the identified drawbacks makes feasible to obtain, systematically, the core architecture of a *Business Information System Family*, composed by the *core process* and the extra information needed to derive, systematically too, each version. We exemplify our approach with reference to a real-life *E-Government* case study.

Keywords: *Business Information Systems (BIS), Software Product Lines (SPL), E-Government.*

1 Introduction

Bider states in [4] that any organization can be considered from a business process perspective. In addition, is a fact that the larger the size of the organization and the number of business units with which it interacts, the more accurate is this perspective focused on its business process and on how the organization is

* 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).

managed based on its specifications. In this context, 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 organizations, defined graphically by means of business process modeling notations, mainly *Business Process Model Notation* (BPMN)[2]. This software systems are commonly integrated into the *Information Technology* (IT) infrastructure of the organization with the purpose of providing support for performing its business process definitions.

In this context, we have identified the following situations: On the one hand, is a fact that the process definitions are very changeable, and the existence of several versions of the same business processes, in function of the characteristics of the unit that performs it, is very common. On the other hand, we must also take into account other less likely but possible situations where a company has to put in the disposal of other its business processes definitions so that these are shared by both (i.e: company mergers and acquisitions, creation of delegations, or situations regulated under Government Acts). Both situations support the claim that nowadays organizations lost the matching between different versions of the process established in each of its business units and the original, which implies ambiguity and problems in the maintenance of its specifications. In addition, is a fact that the variability level of average-size BIS is usually highly enough for making the design of this kind of systems a complex task. Once the problem is identified, to maximize the level of reuse of these definitions is considered a core need because it can minimize: (i) the investment that company owners must take to redefine their business processes, under a business management perspective; and (ii), the development times and costs of this redefinition by a process engineer, under an operational perspective. The importance of this need is emphasized by the frequent failures in the execution of business strategies, 70-90%, identified in [9], defining strategy as the direction, scope and goals of an organization and the set of business processes to execute it.

To contextualize our approach, we must enter into detail of current solutions. Mainly, there is an approach, called *Process Family Engineering* (PFE) [5] (See Section 3.2 for more details), that tries to increase the reusability on the development of BIS using ideas from the *Software Product Lines* (SPL) field [20]. Roughly speaking, the SPL field 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 *Feature Models* (FM), that contains only features observable by the end user, and relationships between them, showing which features are present in all the products, called core features, and which not, called variable features. (See Section 3.1 for a more detailed description). In addition, the PFE approach identifies some variability aspects on business process models, and proposes to extend the standard BPMN for representing it [18]. However, although the PFE approach is quite valuable, it presents some drawbacks [12] related with ambiguity and maintenance that are presented at Section 3.2.

Given the problems of the current proposal, the main motivation of this paper is to define a methodological framework focused on obtaining the reusable core

architecture of a family of BIS. For that purpose, as shown in Section 4, we provide a methodology named *Business Family Engineering* (BFE), and we define the fragment of this methodology, namely *Business Family Domain Engineering*, that is the focus of this paper. Roughly speaking, the proposed framework is composed by two different activities focused (i) on capturing the information of the problem domain and the needs of each member of the family, see Section 4.1; and (ii) on the design of the core architecture, as shown in Section 4.2. This core architecture is defined by means of the reusable process definition, namely *core process* (Section 4.2.c), and the extra information needed to derive each specific version from it, namely *transformation rules* (Section 4.2.d). In addition, we explore how to ensure the systematization in the discovery, resolution and application of the *transformation rules*, proposing a set of technological components to be implanted in the infrastructure of the specific members of the family. Thus, we motivate our approach by means of a real-life *E-Government* case study, defined in Section 2. Finally, as proof of concepts, we have developed an Eclipse-based tool that provides support for obtaining the *core process* definition. See Section 4.2 for more details about tool support and on the proposed components in the infrastructure of the organization.

As a result of our contributions, under a business management perspective, we improve the development of business information systems reducing its complexity level in situations where is needed to perform a business family, using our proposed methodological framework. It implies to companies owners that the traceability between the core, and reusable, specification of its business processes and its variations is better guaranteed, improving alignment between the deployed process definitions for each unit, and its business strategies. In addition, under a research perspective, we provide to process engineers an improvement based on the systematization of their activities. In addition, our proposal provides a basic structure of the business process model that supports the variability aspects identified by the PFE approach, without the need of extending the standard notation of BPMN.

2 Motivating BFE with an E-Government case study

The *E-Government* context provides a comprehensive scenario where there exists multiple public agencies that share similar processes (even identical) defined on the basis of governmental decrees and laws. Thus, our case study illustrates how to model an scenario that reflects the real life business process of managing interior communications between administration units without using papers. Actually, several public entities, such as *Andalusian Government*, *University of Seville*, etc. performs this business process, and a reusable definition is needed.

The scenario is the following. We assume in the context of an administration unit that there exists three different participants involved into the interior telematic communication business process, concretely an Editor (a Public Employee), a Head of Service (other Public Employee), and an Interior Telematic Communication System. Each participant collaborate in the overall process called *Interior Telematic Communication*. Thus, an Editor may perform: (i) the redaction of

the documents associated with an interior telematic communication, related with an specific administrative record selected by him in the Interior Telematic Communication System, which evaluates these documents, and verify their validity in function of the administrative record selected; (ii) cancel a redacted communication; (iii) change the documents of the communication; and (iv) send the interior telematic communication, activity decomposed by four subprocesses: validate the communication, performed by means of a collaboration between the Head of Service and the Editor; identify signers, previous selection of the available signers associated with the administrative record; send to electronic signature; and finally, send communication. In addition, several specific processes from any possible public entity could be introduced, such as: (v) obtain information about interior communications (evolution of the administrative record, applicable regulations, etc.) obtained from an Administrative Process Engine (we have introduced a new participant in addition); and (ii) extra services, such as the monitorization subprocess of the signature facade and the management of the administrative record registry and timestamp respectively.

We use this case study as starting point for defining how to develop families of business information systems that provides support for this *E-Government* business process, motivating that the BFE approach is feasible addressing into the following issues, that are covered in the paper: (i) increasing the business process definitions reusability by means of *Software Product Line* (SPL) techniques; (ii) improving the design of highly variable business process models proposed by the PFE approach; and (iii) visualizing the variability of the business information system and its process specifications improving their traceability.

3 Background Information

3.1 Software Product Lines and Feature Models

Pohl *et al.* define in [11,20] that *Software Product Line* (SPL) development aims at and achieves pro-active, constructive reuse, based on the idea to develop software products which share a significant amount of characteristics, namely *features*. Thus, a *feature* is a characteristic of the system that is observable by the end user. Roughly speaking, SPL systematizes the reuse across the set of similar products, defined by means of their features, that a software company provides.

The SPL approach is devoted to overcome complexity providing all the techniques needed for enabling the mass production of software in a certain application domain. The variability concept appears in SPL to represent the differences and commonalities inside an application domain. Variability is one of the critical aspects of SPL and it must be managed at all the stages of SPL development. Thus, the software process of SPL is divided into two main stages: *Domain Engineering*, which is in charge of providing the reusable core assets that are exploited during the derivation of products, done at a second stage named *Application Engineering* [20].

One of the most accepted techniques to represent the set of products of an SPL are *Feature Models* (FM) [8]. The main goal of feature modeling is to

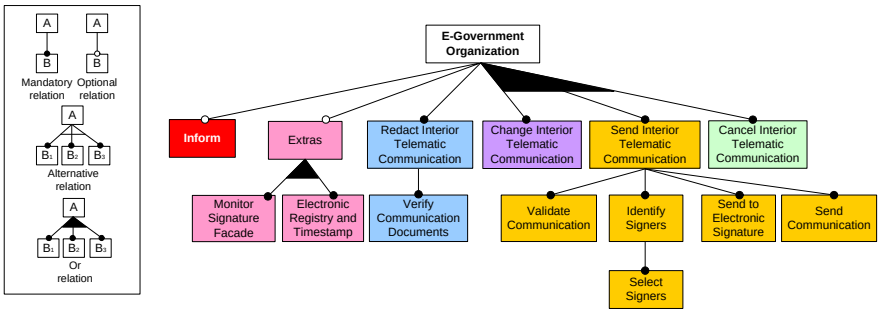

Fig. 1. Feature Model of our case study

identify commonalities and differences among all products of an SPL. A feature model is a compact representation of all potential products of an SPL showing a set of features in an hierarchical structure where it is shown which features are present in a product of the product line. Figure 1 shows the feature model of our case study. A FM establishes a parental relationship between each feature, as shown in Figure 1, that can be: (A)*Mandatory*: if a child feature node is defined as mandatory, it must be included in every product that contains the parent; (B)*Optional*: if a child feature node is defined as optional, it can be included or not when its father feature appears in a product; (C)*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 (D)*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.

3.2 Process Family Engineering

The *Process Family Engineering* (PFE) [5] approach explores the idea of applying SPL philosophy for managing the variability of business information systems. In PFE, feature models are used for representing the set of processes contained into a business. In addition, a business process diagram notation, such as BPMN, is used for representing an specific process. However, the syntax of this notation is redefined for providing support for representing highly variable business process models, namely variant-rich business process models [18].

The PFE approach defines a variant-rich business process model as a process model that represents how to deal with some identified variability aspects: (A)*Alternative behaviors*: it defines when there exists several different ways for performing an activity into a business process definition. Figure 2.a shows an example about a refinement of the *Inform* business process from our case study, represented using BPMN. When the Administrative Process Engine submit some information to the Interior Communication Editor or to the Head of Service it could be done by e-mail, publishing the information as a new into the Intranet, etc. As shown, the PFE approach introduces some stereotypes: (i) ≪*Abstract*≫,

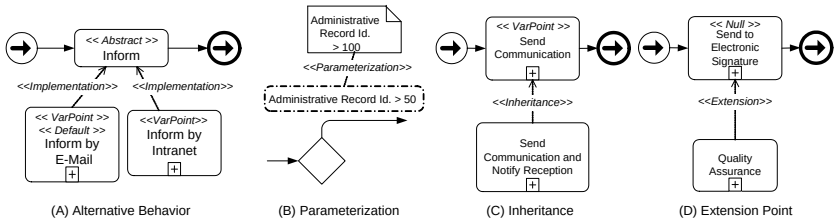

Fig. 2. Variability aspects defined by the PFE approach

for defining the activity that has an alternative behavior; (ii) ≪*VarPoint*≫, for identifying the activities that implements each possible behavior, this stereotype sometimes is represented as a puzzle piece into the activity; and (iii) ≪*Implementation*≫, for describing a new kind of flow not defined in the standard BPMN notation, that represents that an activity marked as ≪*VarPoint*≫ implements the behavior of other activity marked as ≪*Abstract*≫. In addition, the default implementation can be provided introducing the stereotype ≪*Default*≫; (B) *Parameterization*: it defines that each BPMN attribute can be parameterized to support a range of values. Figure 2.b shows how to represent possible ranges of the value *Administrative Record Identifier*, into the subprocess *Verify Communication Documents* from the case study of this paper. This attribute is marked using a grouping box and associated to other possible values by means of a new association flow defined with a new stereotype named ≪*Parameterization*≫; (C) *Inheritance*: it defines a modification of an existing subprocess by adding or removing elements regarding to specific rules. This allows for realizing alternative variation points. As shown in Figure 2.c, the business process *Send Communication* has been modified to *Send Communication and Notify Reception* because the Interior Telematic Communication System uses a Notification System when it sends the communications. This situation is marked with a new association defined using the stereotype ≪*Inheritance*≫; and (D)*Extension Points*: it defines an optional behavior. Figure 2.d depicts how an optional behavior from *Send to Electronic Signature* subprocess could be included for quality assurance of this activity. This situation is marked using the stereotype ≪*Extension*≫. The main difference between SPL and PFE is that the SPL field provides a set of different products that shares common features, and the PFE approach provides only one product, which represents a company, that evolves at runtime, and each possible configuration of this company is managed as a product that contains a subset of processes enabled at a certain moment of the execution. On the one hand, the SPL products are implemented by software artifacts. For each of them there exists a feature selection phase that generates the final products (a set of core and variable features). On the other hand, the PFE products are implemented by processes. For each product, the system can evolve to another product increasing or decreasing the variable set of processes thus, each product is a software system based on processes. However, although the PFE approach is valuable, it presents some drawbacks. For example, the use of feature models and the derivation of business processes from it presents some problems, such as am-

biguity, that has been explored by us in [12]. Another consideration is the need of redefining the BPMN syntax introducing some information about variability which is also present in the feature models, thus, information is duplicated with the obvious problems for maintenance. In addition, there not exists support for this new syntax of BPMN, because it is not an standard notation.

4 Obtaining the Core Architecture of a Business Family

In this section we present our approach: *Business Family Engineering* (BFE). In the BFE software process there are two main activities: (i) *Business Family Domain Engineering*, where we build the BFE core architecture; and (ii) *Business Family Application Engineering*, where we obtain specific business information systems, that are described by means of execution languages, such as BPEL[1].

The *Business Family Domain Engineering* software process is composed by two different activities: (i) *Domain Requirements Engineering*, that is focused on capturing the requirements of the problem domain; and (ii) *Domain Design*, that is focused on exploring the variability of the system and providing the core architecture. As stated previously, *Business Family Application Engineering* stage is out of the scope of this paper.

4.1 Domain Requirements Engineering

The *Domain Requirements Engineering* activity is focused on identifying the set of companies (including its business units) and its business processes that would be members of the business family. This step takes into account the traditional requirements elicitation activities of software engineering, and provides as resulting artifact the documents that reflects this elicitation. Figure 3.a shows the *Domain Requirements Engineering* overview using the SPEM[2] notation. The main activities are:

a) Identify the Companies and its Business Processes (BPs): it is focused on obtaining a first conceptual description about the companies and its business processes. It could be done using a free textual specification, see Section 2 for an example. This activity generates: (i) the *Companies and Business Processes Definition*, that contains the description of the problem domain; and (ii) the *Glossary of Terms*, that contains a terminology summary.

b) Refine Identified BPs into Elementary Business Processes (EBPs): it is focused on identifying the business processes contained into the *Companies and Business Processes Definition* artifact, that are considered an EBP. Larman defines in [1] that *"one task performed by a person in a place, in an instant, in response to an event business, which adds a quantifiable value to the business and leaves the data in a consistent state is an EBP"*. The objective of this activity is to filter the processes that define tasks at a very low level, namely *atomic tasks*, as are those who do not concern us. In addition, this activity will filter those

[1] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[2] http://www.omg.org/technology/documents/formal/spem.htm

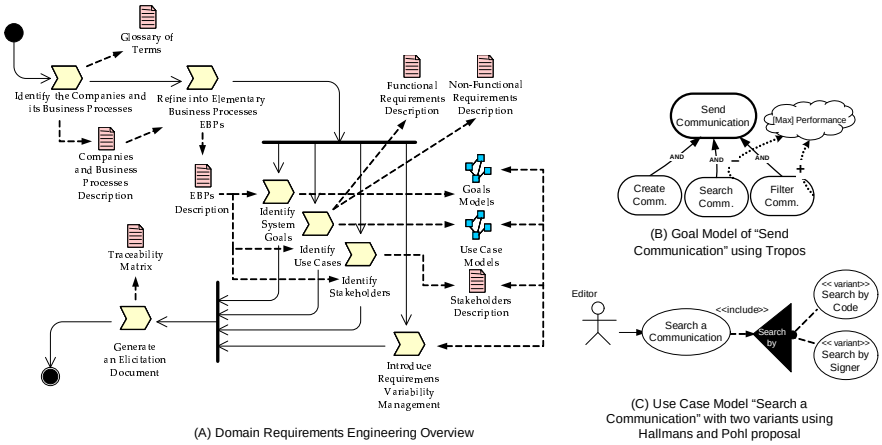

Fig. 3. Domain Requirements Engineering overview and models obtained

activities that require direct interaction of the user with the system. Thus, this activity generates the *EBPs Definition*, that contains their description.

c) Identify System Goals, Use Cases and Stakeholders: these activities are focused on defining the system goals, stakeholders and requirements (functional and non-functional). A goal is an objective the system under consideration should achieve and a feature is an end-user visible characteristic of a system. There is an overlap between the goal and feature definitions that motivates that some authors uses feature models, shown in Section 3.1, for describing goals [20]. These goals are obtained from the EBPs identified in the previous step, due to the realization of a business process covers a set of goals [19]. However, there exists other goal-oriented representations, such as Tropos [14] or i* models [16], that can be used too, depending on several factors, such as tool support or non-functional requirements graphical representation. Figure 3.b sketches the goal model of *Send Communication* represented using a Tropos model. In addition, the stakeholders and requirements are defined by means of use cases. Thus, these activities generates the following artifacts: (i) *Goal Models*; (ii) *Use Case Models*; (iii) *Functional and Non-Functional Requirements Descriptions*, in a textual representation by means of templates, as shown in [3], and in a graphical representation using use cases (the Non-Functional Requirements can be represented using Tropos, as shown in Figure 3.b with the *Performance* requirement); and (iv) *Stakeholder Descriptions*, in a textual representation. In addition, once these artifacts are obtained, these activities generate a *Traceability Matrix* between them. This matrix represents a table that correlates requirements with system goals and with business processes. Roughly speaking, this artifact provides a response for the following questions: *What requirements fulfill a system goal?* and *What system goals are contained into a business process?*.

d) Introduce Requirement Variability Management: it is focused on increasing the reuse of the artifacts obtained at the previous stage. For that purpose, some requirement variability techniques has been identified. The use of

feature models and/or Tropos for defining system goals provides the enough support for representing variability in system goals [14]. For representing variability in use cases models, there exists several proposals that extends the standard notation of use case diagrams, such as proposed by Gomaa [23] or by Hallmans and Pohl [11], or proposes other notations and languages, such as COVAMOF [24]. Figure 3.c shows an use case diagram using the extensions proposed by Hallmans and Pohl. In addition, there exists approaches for representing variability in the textual representation of use cases too, such as proposed by Bertolino [26] or John and Muthig [25]. We recommend a combination of feature models for representing variability in system goals and a modification of the traditional use case representation, in order to specify variabilities [21]. This combination allows to capture variability and essential activities of a domain with use cases and to detail common and variable characteristics with feature diagrams. In addition, both techniques has tool support.
e) Generate an Elicitation Document: this activity is oriented on generating an elicitation document that contains all the artifacts generated in the previous steps as output of the *Domain Requirements Engineering* activity.

4.2 Domain Design

The *Domain Design* activity is focused on performing a variability summary of the set of businesses identified at the previous step and on providing the core architecture of the product line. Figure 4 shows the *Domain Design* overview. In this stage, the *Requirement Engineer* performs the following activities:
a) Define a Variability Summary and Obtain Commonalities: these activities are focused on obtaining a variability summary from the *Goal Model* artifact, obtained at the *Domain Requirements Engineering* stage. For that purpose, there exists several ways to perform a variability summary, but in our approach we propose the derivation of feature models from system goals [20,19], due to we can analyze them [6] for obtaining the commonalities summary needed to perform the following phase. In addition, with this analysis we can evaluate the percentage or probability of occurrence of a feature in a product. If this ratio is high, we can consider that the feature is part of the core. Thus, we introduce an optional commonality threshold for introducing some features/processes into the core of the product line, as shown in our previous work [13]. Figure 1 presents the variability summary of the business family of the case study by means of feature modeling. In addition, the commonalities summary is provided on Figure 5.a using feature models too, and introducing *Change Interior Telematic Communication*, *Send Interior Telematic Communication* and *Cancel Interior Telematic Communication* into the core of the family by means of a commonality threshold, as shown in Figure 5.b. Notice that if we did not introduce this commonality threshold this processes would not be into the core architecture.
b) Obtain Core and Variable Reusable Assets Definition: it is focused on obtaining the reusable assets that are the basis of the core architecture. For that purpose, we analyse the *Variability Model* and *Commonality Summary* artifacts, using the operations defined in [6]. The result of this analysis provides the set of reusable assets identified as *Core* and *Variable* features. Once the reusable

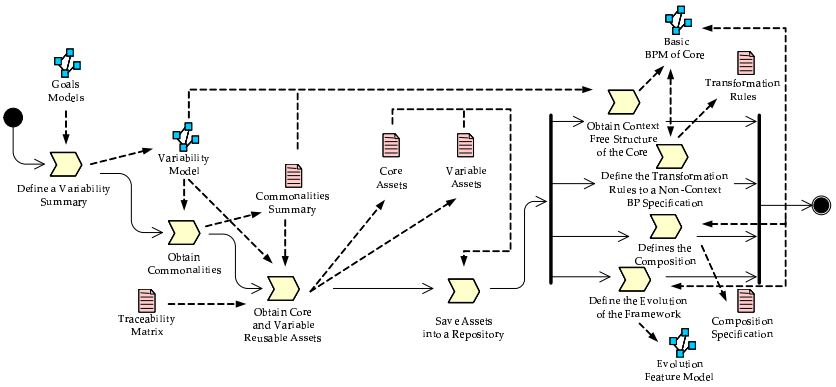

Fig. 4. Domain Design overview

assets are obtained, we use the *Traceability Matrix* defined at the *Requirements* artifact, from the *Domain Requirements Engineering* stage, for obtaining the respective equivalence between features and business process. Thus, the reusable assets are defined by means of business process models. These assets are contained into the generated artifacts *Core* and *Variable Assets*, which are saved into a repository.

c) Obtain the Context-Free Structure of the Core: it is focused on obtaining a basic structure of a business process model that represents the core architecture. It is represented by the *Basic BPM of Core* artifact. For that purpose, we propose in [12] a derivation from the feature model that represents the *Variabily Model* to a business process model. This derivation is based on *Automata Theory and Formal Languages* by means of *Context-Free Grammar Representations*. The derivation process provides a mapping between feature models and business process models that improves the design step of the PFE approach by means of improving the maintainability of feature models and BPMN, 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. Our transformation is currently appplied in [15]. Thus, the variability aspects identified by the PFE approach, described in Section 3.2, can be represented by our approach as follows: (A) *Alternative behavior* is represented by an alternative relation on the feature model, defined at Section 3.1, where the parent feature is considered the variation point, and the children are considered variants [20]; and a gateway *Xor* of BPMN, which defines that only one subprocess controlled by this gateway, *Inform by e-mail* or *Inform by Intranet*, must be completed for performing the subprocess *Inform*; (B) *Parameterization* is resolved by rewriting the condition for accepting several value ranges; (C) *Inheritance* is defined by means of an *Or* relation on the feature model, defined at Section 3.1, and a gateway *Or* of BPMN, which defines that almost one subprocess controlled by this gateway, *Send Communication* and *Notify Reception*, must be completed for performing the process *Send Communication*;

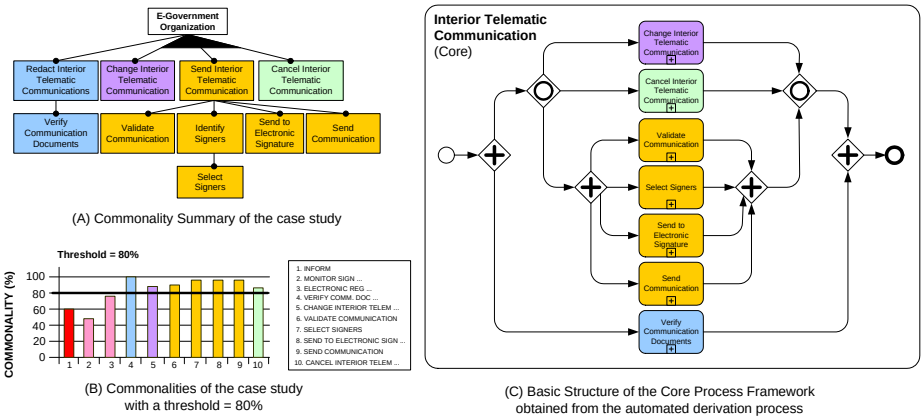

Fig. 5. Commonality summary of the case study obtained in Variability Analysis represented using feature modeling and basic structure of the Core Process Framework obtained from the derivation process

and finally, (D) *Extension Points* is defined by means of an optional relation at the feature model, defined at Section 3.1, and an *Xor* gateway of BPMN with an empty option. We have developed an automated tool support for obtaining the basic structure of a BPMN derived from the commonalities summary into this phase. This basic structure is the *Core Process Framework*. This mapping has been implemented by means of MDD transformations. For that purpose, we have performed a transformation between the *FeAture Model Analyzer* (FAMA)[3] metamodel as source and the *Eclipse SOA Tool Platform*[4] BPMN metamodel as target metamodel using the *Atlas Transformation Language* (ATL). It has been published on Eclipse ATL website[5]. Figure 5.c presents the basic structure of the *Core Process Framework* of our case study in BPMN, obtained from the automated derivation process.

d) Define the Transformation Rules to a Non-Context Free BP Specification: it is focused on providing the rules needed to build a non-context free business process specification. These rules need to be performed manually by the process engineer. The identified rules are the following: (i) introduce the guards into the gateways that defines the conditions; (ii) identify loops and multiple instances of a subprocess; (iii) introduce start and finalization specific events (message, timer, etc.); (iv) introduce exception handling; and (v) introduce the stakeholders whose are identified at the *Requirements Capture* stage. This activity generates the artifact that contains these rules, namely *Transformation Rules*. Figure 6 depicts an overview about how to obtain a derivation for a concrete public entity, such as *Andalusian Government*, applying the specific *Transformation Rules* of this context, defined, for this example, in the *w@ndA E-Government*

[3] http://www.isa.us.es/fama
[4] http://www.eclipse.org/stp
[5] http://www.eclipse.org/m2m/atl/atlTransformations/#FM2BPMN

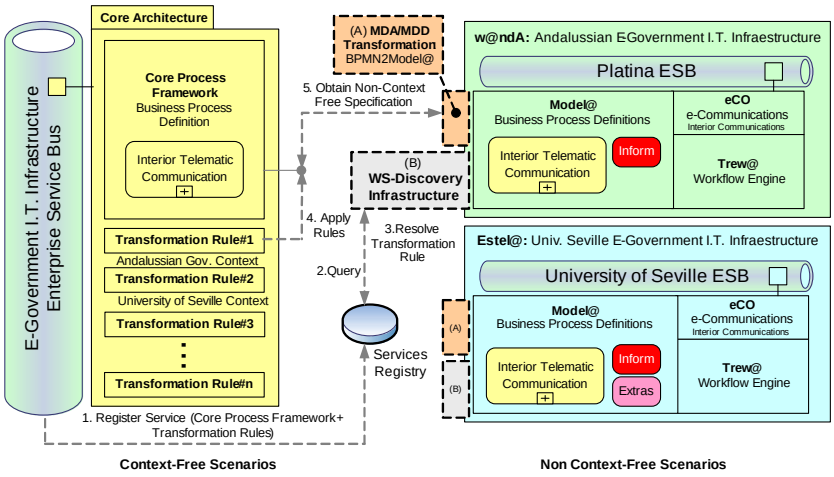

Fig. 6. Derivation to a Non-Contex Free Specification applying Transformation Rules

infrastructure initiative[6]. For that purpose, this infrastructure must contain two additional components: (i) a *MDA/MDD Transformation Engine* for obtaining process definitions modeled with the concrete specification used by the workflow engine of the infrastructure, that could be BPMN or other, such as *Model@*[7] running on *Trew@*[8] execution engine, as shown in Figure 6; and (ii) a *WS-Discovery Infrastructure*, for discovering and resolving the concrete transformation rules applied in an specific context. For that purpose: (1) the *Core Process Framework* and its *Transformation Rules* must be registered in a services registry. This service registry guarantees the matching between the reusable core architecture and the different versions of this for each specific context, defined by means of its transformation rules; (2) The *WS-Discovery Infrastructure* would query about the services that provides an implementation of the *Interior Telematic Communication* business process in the context of the *Andalusian Government*; Once this query is resolved (3), the desired *Non-Context Free BP Specification* is obtained as result of applying its corresponding *Transformation Rules* (4); Finally, this specification will be translated to the modeling language of the specific scenario (5), and executed in its specific workflow engine, which supports the *E-Government* services and applications for performing *Interior Telematic Communications*, such as *eCO*[9] in the context of the *Andalusian Government*, and the *University of Seville*. Figure 6 depicts this behavior, and denotes that the *Interior Telematic Communication* business process can be composed with other business process definitions such as *Inform*, in the context of *w@ndA*, or *Inform* and *Extras* in the context of *Estel@*, the *University of Seville E-Government*

[6] https://ws024.juntadeandalucia.es/pluton/adminelec/ArTec/wanda.jsp
[7] https://ws024.juntadeandalucia.es/pluton/adminelec/ArTec/modela.jsp
[8] https://ws024.juntadeandalucia.es/pluton/adminelec/ArTec/trewa.jsp
[9] http://www.juntadeandalucia.es/repositorio/usuario/listado/fichacompleta.jsf?idProyecto=667

initiative[10]. This compositions are identified in the following stage.
e) Define the Composition: it is focused on providing the mechanisms needed to perform a composition of reusable assets into the *Core Process Framework*. A composition between two or more business process models define the order of the execution of each business process to be composed. Notice that we must usually preserve the order of execution of each business process model to be composed. We have determined that there exists two different types of business process models composition: *Non-Overlap Composition* and *Overlap Composition*. On the one hand, a *Non-Overlap Composition* is defined as a composition between two or more business process models that do not consume or generate any needed artifact for other. Roughly speaking, these business processes could be considered as independent processes that do not collaborate with anyone, but all these processes need to be performed for performing a concrete activity. On the other hand, an *Overlap Composition* is defined as a composition between two or more business process models that collaborates between them, in terms of generating and consuming some artifacts between them that rules the dependency from other to perform its activities. This type of composition can only be done manually. Nowadays, one of the topics of our research is focused on providing algorithms for assisting these kinds of composition and for checking the consistency of the composed business process models. For that purpose, we take the algorithms presented in [22] as starting point for this future work.
f) Define the Evolution of the Framework: it is focused on defining the evolving behavior of the *Core Process Framework*. Roughly speaking, we can consider the *Core Process Framework* as an evolving system, defining these evolutions as each addition or reduction of some business process models into the core architecture by means of the compositions defined at the previous step. Thus, the main difference between the process framework provided by the PFE approach and our *Core Process Framework* is that PFE defines only one product (core architecture) not variable, and our approach provides a variable core architecture managed as a set of products using SPL techniques [17].

5 Related Work

In addition of the *Process Family Engineering* (PFE) [5] approach, introduced in Section 3.2, other proposals have focused on increasing the degree of reuse of the business process definitions: (i) Van der Aalst *et al.* [10] proposes an extension of YAWL (*Yet Another Workflow Language*), named *extended workflow-nets* (EWF-nets), for performing configurable and reusable workflow definitions of business processes. YAWL is a workflow modelling language inspired by Petri nets for defining business process models. However, although this approach provides a formalization of EWF-nets for defining possible variations into a workflow, it does not provide support for other notations, such as BPMN. In addition, the application of SPL techniques is not explored; (ii) Ciuksys *et al.* [7] provides a process ontology for introducing semantic information into a process model.

[10] http://www.us.es/admonelec

Thus, this approach propose to analyse this semantic information for obtaining a commonality summary, but without defining how to represent these reusable assets; and finally, (iii) Asadi *et al.* [15] proposes a methodology for development of business process families exploting SPL techniques, taking into account our previous work in [12], annotating feature models for introducing declaration of semantic relations of business process models, such as sequences or choices. Our approach states that these relations could be defined as transformation rules, also known as business rules, and without the need of extending current feature model notation with annotations.

6 Conclusions and Future Work

The development of the *Information Technology* (IT) infrastructure that supports business processes of large organizations is a very complex task, due to the existence of different versions of the same process defined in terms of the units that execute it. This variability claims that maximizing the reuse of these definitions is a core need. In this paper we have proposed a methodological framework named *Business Family Engineering*, and we have defined the fragment focused on obtaining, systematically, the reusable core architecture of a business family, namely *Business Family Domain Engineering*, exemplifying it in an *E-Government* case study. The main contributions of our approach can be decomposed into two perspectives: (i) *contributions for business management*, focusing on minimizing the deviation in the definition of the *core process* regarding its variants and to improve the matching between them. It implies an improvement in the alignment between the different versions deployed in all the units and its business strategies [9]; and (ii) *research contributions*, focusing on minimizing the development times and costs, and improving current proposals, such as *Process Family Engineering* (PFE), by means of the proposed systematization and eliminating its ambiguity problems. The main research lines of the future work derived from this paper are centered on to refine the proposed methodology by means of: (i) defining the composition algorithms into the *Core Process Framework* and the syntax of the *Transformation Rules*; (ii) defining the activities of *Business Family Application Engineering*; and (iii) tooling the *Business Family Engineering* approach.

References

1. C. Larman. UML and Patterns. An Introduction to Object-Oriented Analysis and Design and Unified Process. Second Edition. Prentice-Hall.
2. BPMI. Business Process Modeling Notation BPMN Version 1.0, 2004. OMG.
3. A. Cockburn. Structuring Use Cases with Goals. J. of Object-Oriented Prog., 1997.
4. I. Bider. Striving for Better Administrative Quality as a Stimulus for Business Process Change. In Proc. 10th Workshop on Business Process Modeling, Development, and Support (BPMDS), co-located with CAISE, 2009.
5. 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. PESOA-Report No. 18/2005.

6. D. Benavides, A. Ruiz-Cortes, and P. Trinidad. Automated Reasoning on Feature Models. In Proc. 17th Int. Conference, CAiSE 2005, 3520:491-503, 2005.
7. D. Ciuksys and A. Caplinskas. Ontology-based Approach to Reuse of Business Process Knowledge. Informacijos Mosklai ISSN:1392-0561, (42-43):168-174, 2007
8. K. Czarnecki and M. Antkiewicz. Mapping Features to Models: A Template Approach Based on Superimposed Variants. Generative Programming and Component Engineering, (422-437). 2005.
9. M. Morgan, R. E. Levitt, W. Malek. Executing Your Strategy. How to Break It Down and Get It Done. Harvard Business School Press. ISBN 9781591399568 2007.
10. F. Gottschalk, W. van der Aalst, M. Jansen-Vullers, and M. L. Rosa. Configurable workflow models. BETA Working paper 222, Eindhoven Univ. Technology, 2007.
11. G. Halmans and K. Pohl. Communicating the Variability of a Software-Product Family to Customers. Inform., Forsch. Entwickl., 18(3-4):113-131, 2004.
12. I. Montero, J. Peña, and A. Ruiz-Cortes. From Feature Models to Business Processes. In Proc. of IEEE Int. Conference on Services Computing, 2008.
13. J. Peña, M. Hinchey, A. Ruiz-Cortes, and P. Trinidad. Building the Core Architecture of a NASA Multiagent System Product Line. In Proc. 7th Int. Workshop on Agent Oriented Software Engineering, 2006.
14. Y. Yu, A. Lapouchnian, S. Liaskos J. Mylopoulos and J.C.S.P. Leite. From Goals to High-Variability Software Design. In Proc. 17th Int. Symposium on Methodologies for Intelligent Systems 2008.
15. M. Asadi, B. Mohabbati, N. Kaviani, D. Gasevic, M. Boskovic and M. Hatala. Model-Driven Development of Families of Service-Oriented Architectures. In Proc. 1st Int. Workshop on Feature-Oriented Software Development MODELS 2009.
16. AK. Ghose, G. Koliadis, A. Vranesevic, M. Bhuiyan, and A. Krishna. Combining i* and BPMN for Business Process Model Lifecycle Management. In Proc. of the BPM 2006 Workshop on Grid and Peer-to-Peer based Workflows, LNCS, 2007.
17. I. Montero, J. Peña, and A.Ruiz-Cortes. Representing Runtime Variability in Business-Driven Development Systems. In Proc. 7th Int. Conference on Composition-based Software Systems. 228-231. ICCBSS 2008.
18. A. Schnieders and F. Puhlmann. Variability Mechanisms in E-business Process Families. In Proc. 9th Int. Conference on Business Information Systems BIS 2006.
19. J. Bae and S. Kang. A Method to Generate a Feature Model from a Business Process Model for Business Applications. In Proc. 7th IEEE Int. Conference on Computer and Information Technology, 2007.
20. K. Pohl, G. Böckle, and F. van der Linden. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, September 2005.
21. I. Montero. A Case Tool for Specifying Domain Requirements Document with Variability. Technical Report, 2008.
22. J. Peña, R. Corchuelo and J.L. Arjona. Towards Interaction Protocol Operations for Large Multi-agent Systems. In Proc. of FAABS, volume 2699 of LNAI, 2002.
23. H. Gomaa. Modeling Software Product Lines with UML. In Proc. 2nd. Int. Workshop on SPL: Economics, Architectures and Implications, 2001
24. M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. COVAMOF: A Framework for Modeling Variability in Software Product Families, volume 3154. LNCS, 2004.
25. I. John and D. Muthig. Tailoring Use Cases for Product Line Modeling. In Proc. of the Int. Workshop on Requirements Engineering for Product Lines, 2002.
26. A. Bertolino, A. Fantechi, S. Gnesi, G. Lami, and A. Maccari. Use Case Description of Requirements for Product Lines. In Proc. of the Int. Workshop on Requirements Engineering for Product Lines, 2002.