

Modelling and Managing Variability in Business Process Models

Clara Ayora Esteras

Supervisors: Victoria Torres Bosch

Vicente Pelechano Ferragud



**UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA**

Clara Ayora Esteras

Modelling and Managing Variability in Business Processes

Masters Thesis. September, 2011

Clara Ayora Esteras

Modelling and Managing Variability in Business Processes

Masters Thesis. September, 2011

Modelling and Managing Variability in Business Processes:

This report was prepared by

Clara Ayora Esteras

Supervisor

Dra. Victoria Torres Bosch

Dr. Vicente Pelechano Ferragud

Members of the Thesis Committee

Dr. Juan Carlos Casamayor Ródenas, Universidad Politècnica de València

Dr. Vicente Pelechano Ferragud, Universitat Politècnica de València

Dr. César Ferri Ramírez, Universitat Politècnica de València

Centro de Investigación en Métodos de Producción de Software

Universitat Politècnica de València

Camí de Vera s/n, 46022 València

Spain

Tel: (+34) 963 877 007 (Ext. 83530)

Fax: (+34) 963 877 359

Web: <http://www.pros.upv.es>

Date: September-2011

Comments: This document is submitted in partial fulfillment of the requirements for the degree of Masters on Software Engineering, Formal Methods and Information Systems by Universitat Politècnica de València.

To my parents

*“Nunca aceptes la derrota,
la debilidad y el desánimo
como parte tuya.
Tú eres el éxito mismo.”*

Acknowledgements

First and foremost, I would like to especially thank Victoria Torres for her effort and dedication. Without her, this work would not have been possible. I am very happy and pleased to work with her. A special gratitude is also due to Vicente Pelechano for his great advices and his guidance but, above all, for giving me the opportunity to develop this work.

I would like also to thank my family, especially to my parents, for their constant encouragement, love and unlimited patience. They care about me giving me always the unconditional support I need.

I also want to give special thanks to María, Nacho, Miriam, Pablo, Isma, Mario, Salva, José Luis, Ainoha, Raúl, Arthur, Sergio L. Harvey and Carla for all the good moments we have spent together in and out of the lab. Being with them always means having a great time.

Nor do I want to forget Manoli, Joan, Fani, Sergio E, Marcela, Ana M. and the rest of colleagues from the PROS research centre for their collaboration and support.

Finally, I would like to thank all those people (school/institute friends, musicians, Cullera's people, swimmers, etc.) that are part of my life, and, unwittingly, make it a little bit easier.

To all of them, thank you very much.

Clara Ayora

Abstract

Business Process (BP) models capture the coordination of a set of activities whose execution realizes specific business goals within a company. However, the construction of such models entails a big challenge for modellers and strongly depends on the nature of the domain being modelled. Moreover, when this nature involves handling many process alternatives, the use of variability modelling mechanisms becomes essential to succeed in the BP modelling task.

Even though BP modelling variability has already been addressed by researches from the BPM community, it still remains as a challenge that is requested as hot topic in the most relevant conferences related to the BP area (i.e. the international conference on Business Process Modeling (BPM), CoopIS or BPDSM). This demand appears since the solutions provided in the literature do not deal with variability in a broad sense (considering all types of variability that we can find in a BP model), and in a scalable manner.

In this context, this work provides a modelling approach that brings variability concepts as first order aspects of the modelling process. Concretely, the approach isolates the variability factors that affect a BP and allows managing independently their impact over the whole model.

For such purpose, we rely on the techniques proposed in the field of the Software Product Lines to deal with variability issues. These techniques allows enhancing variability expressiveness as well as promoting model maintenance, legibility, understanding and reuse regarding variability.

Finally, a running example is described and developed to illustrate the proposal and its applicability.

Resumen

Los modelos de Procesos de Negocio capturan la coordinación de un conjunto de actividades cuya ejecución lleva a cabo objetivos de negocio específicos de las empresas. Sin embargo, construir estos modelos supone un gran reto ya que esta construcción está fuertemente ligada a la naturaleza del dominio que se está modelando. Además, cuando esta naturaleza conlleva la aparición y manejo de muchas alternativas del propio proceso, es esencial el uso de mecanismos de modelado de la variabilidad para llevar a cabo con éxito el modelado de estos procesos.

En la actualidad ya existen propuestas que permiten el modelado de estos procesos de negocio variables. Sin embargo, ninguna de ellas proporciona soporte completo al conjunto de conceptos de variabilidad que deben ser abordados.

En este contexto, este trabajo proporciona una propuesta de modelado que trae los conceptos de la variabilidad a aspectos de primer orden del proceso de modelado. Concretamente, la propuesta aísla los factores de variabilidad que afectan al proceso de negocio y permite gestionar, de forma independiente, su impacto sobre el resto del modelo.

Con este fin, se hace uso de las técnicas de variabilidad presentadas en el ámbito de las Líneas de Producto Software. Estas técnicas permiten mejorar la expresividad de la variabilidad, así como la facilitar el mantenimiento, legibilidad, comprensión y reutilización de los modelos respecto a la propia variabilidad.

Además, en este trabajo se describe un caso de estudio con el fin de ilustrar la aplicabilidad de la propuesta presentada.

Resum

Els models de Processos de Negoci capturen la coordinació d'un conjunt d'activitats que en executar porta a terme objectius de negoci específics de les empreses. No obstant això, construir aquests models suposa un gran repte ja que aquesta construcció està fortament lligada a la naturalesa del domini que s'està modelant. A més, quan aquesta naturalesa comporta l'aparició i maneig de moltes alternatives del mateix procés, és essencial l'ús de mecanismes de modelat de la variabilitat per dur a terme amb èxit el modelat d'aquests processos.

En l'actualitat ja existeixen propostes que permeten el modelat d'aquests processos de negoci variables. No obstant això, cap d'elles proporciona suport complet al conjunt de conceptes de variabilitat que han de ser abordats.

En aquest context, aquest treball proporciona una proposta de modelat que porta els conceptes de la variabilitat a aspectes de primer ordre del procés de modelat. Concretament, la proposta aïlla els factors de variabilitat que afecten el procés de negoci i permet gestionar, de forma independent, el seu impacte sobre la resta del model.

Amb aquesta finalitat, es fa ús de les tècniques de variabilitat presentades en l'àmbit de les Línies de Producte Software. Aquestes tècniques permeten millorar l'expressivitat de la variabilitat, així com la facilitar el manteniment, llegibilitat, comprensió i reutilització dels models respecte a la pròpia variabilitat.

A més, en aquest treball es descriu un cas d'estudi per tal d'il·lustrar la aplicabilitat de la proposta presentada.

Index

List of Figures	xvi
List of Tables	xviii
1 Introduction	1
1.1 Research motivation	2
1.2 Problem statement	3
1.3 The proposed solution	4
1.4 Reasearch methodology of the thesis	4
1.5 Context of the thesis	5
1.6 Outline	6
2 Evaluation framework	9
2.1 Variability within the BPM analysis/design phase	9
2.2 Evaluation Framework	12
2.3 Conclusions	16
3 State of the Art	17
3.1 C-EPC (Configurable EPC)	17
3.2 Variant-Rich Process Models (within the PESOA project)	21
3.3 Provop (PROcess Variants by OPTions)	23
3.4 Analysis of the evaluation results	26
3.4.1 Concept analysis	26

3.4.2	Quality factors analysis	28
3.5	Conclusions	29
4	A modelling approach to support business process variability	31
4.1	Overview of the approach	32
4.2	Base model	34
4.3	Variation model	35
4.3.1	Control flow & tasks variation model	36
4.3.2	Task related elements variation model	38
4.4	Resolution model	40
4.5	Evaluation of the approach	42
4.6	Conclusions	45
5	Methodology. Putting the approach into practice	47
5.1	Introduction	48
5.1.1	Using BPMN to specify the methodology	49
5.2	Outlining the methodology	49
5.3	Process commonalities specification stage	50
5.4	Process individualities specification stage	52
5.5	Process configuration stage	53
5.6	Conclusions	54
6	A running example	55
6.1	Overview	55
6.2	Description of the running example	56
6.3	Development of the running example	58
6.3.1	Base model	58
6.3.2	Variation model	59
6.3.3	Resolution model	60
6.4	Conclusions	62

- 7 Conclusions** **63**
- 7.1 Contributions 63
- 7.2 Validation of the proposal 64
- 7.3 Future work 65
- 7.4 Publications 66

- Bibliography** **68**

List of Figures

1.1	Design research cycle (<i>Takeda et al</i> , 1990)	6
2.1	Business process lifecycle (<i>Weske</i> , 2007)	10
4.1	Overview of the approach	33
4.2	Example of the <i>Base model</i>	35
4.3	Example of the <i>Control flow & tasks variation model</i>	36
4.4	Example of the <i>Constraint model</i>	38
4.5	Example of the <i>Task related elements variation model</i>	39
4.6	Example of the <i>Resolution model</i>	42
5.1	Proposed stages to apply the approach	51
6.1	University access process model with all the model variants included	57
6.2	Process model variant for Galicia and Standard students	58
6.3	Base model for the university access process	59
6.4	The control flow & tasks variation model for the university access process	60
6.5	The task related elements variation model for the <i>Evaluate exam</i> task	61
6.6	Galician region resolution model	62

List of Tables

3.1	Summary of the approaches evaluation regarding concepts	27
3.2	Summary of the approaches evaluation regarding quality factors	28
4.1	Notation of the feature model	39

1. Introduction

Nowadays, Business Process Management (BPM) is a “hot topic” in Computer Science since it offers many challenges for software developers and scientists.

The concept of Business Process (BP) comes from that each product that a company/organization produces is the result of a set of activities performed [47]. Depending on the application context¹ where the BP is being performed, this set of activities may differ making the process to be variable. Dealing properly with this set of variable activities is an important issue for coping with market conditions in an effective manner. Organizations would not only reduce efforts (temporary and economics), but also would gain competitive advantages over other organizations.

In this context, BP models become one of the main artefacts to properly deal with BPM. They not only capture the activities being performed, but also their relationships, context and constraints [48]. Nevertheless, the construction of such models is not an easy task, it entails a big challenge, specially when dealing with variability.

For such purpose, this work presents a modelling approach that faces BP variability. The cornerstone of the approach is to bring variability

¹e.g. when models are built for the international market and require their adaptation for different legal or cultural environments

concepts as first order aspects of the modelling process. In particular, the approach decouples variability from those common parts of the BP, reducing its impact over the rest of model. Thus, variability is considered as a domain by its own, receiving the importance it deserves.

The remainder of the chapter is organized as follows: Section 1.1 presents the motivation of the work presented in this document. Then, section 1.2 states the problem that this work tackles and section 1.3 presents a brief overview of the solution proposed to face the stated challenges. Section 1.4 explains the research method followed to develop this masters' thesis. Section 1.5 explains the context of this work and, finally, section 1.6 outlines the structure of this document.

1.1 Research motivation

Business process models represent, by means of tasks and resources, how organizational goals are achieved. These models are commonly built either for organization design purposes and/or information system design purposes. In both cases, to take advantage as much as possible of these models, they should represent in an accurate way the organizational reality that is interesting for a specific goal [47].

However, BP modelling is not a trivial task. Many factors can affect the way a model is built [30], it not only requires mastering the problem domain being represented, but also the language or notation that is being used to perform this representation. In addition, when this domain involves variability (for instance, due to different legal regulations), BP models turn into complex artifacts that are error-prone and difficult to build, manage, and understand. In these cases, other considerations such as process granularity, context dependency, or scalability need to be considered turning these issues into first order requirements of the modelling process [33].

Coping with such variability in BPs constitutes one of the current challenges faced by the BPM community [8]. In fact, it is possible to find different approaches ([38], [23] or [34]) that provide diverse solutions to deal with BP variability at the modelling level. In these works, through

the use of different techniques (such as feature models) and approaches (such as operational or model projection), the authors deal with the problems that arise with variability during process modelling and/or execution (which involves also considering process evolution techniques such as the ones presented in [46]). Nevertheless, these proposals present some limitations referred to: (1) the degree in which these proposals provide support to the entire set of elements that are subject to variation (e.g. control flow, roles or documents), (2) BP variants management, and (3) the methodological support to assist during the construction and evolution of BP models.

Thus, from the modelling perspective, there is a need for a solid approach which covers these drawbacks. This new approach should give a solution to such limitations in order to provide support for the complete set of concepts that need to be addressed when dealing with BP variability at the modelling level. The research of the development of this approach has constituted the central focus of this work.

1.2 Problem statement

The challenge of modelling BP variability has already been addressed by existing approaches. However, the proposed solutions do not provide variability support in a wide sense as would be suggested from the SPL community. Therefore, in this context, we provide a modelling approach to deal with BP variability which covers all the concepts that need to be addressed when dealing with variability. Concretely, this work tackles the following challenges:

- **Challenge 1:** Identify the concepts that need to be addressed when dealing properly with BP variability at the modelling level.
- **Challenge 2:** Design of a modelling approach that provides support for such concepts.
- **Challenge 3:** Establish a methodological framework to guide and assist BP modellers to put into practice the approach properly.

These challenges are analyzed and answered in the rest of the chapters of this document.

1.3 The proposed solution

This section briefly summarizes the solutions proposed in this masters' thesis to face the challenges stated above. These challenges represent a key factor for the successful of the goal stated in this work.

First of all, regarding challenge 1, one of the goals of this work is to study the BP variability domain at the modelling level. For such purpose, it will be necessary to consider both the functional and non-functional requirements regarding variability of a software system. Concepts such as variation point, variant context, flexibility, or scalability are key concepts to deal properly with variability. Gathering and stabilising a clear definition of these concepts will allow us to define properly the new approach.

Regarding challenge 2, the main goal of this work is to define a modelling approach that provides complete support for the set of concepts extracted from the challenge 1. Specifically, this approach faces BP variability by decoupling it from its impact over the rest of the BP model. Thus, variability is expressed independently gaining in model reusability, legibility, understanding, scalability and management.

Finally, regarding challenge 3, having a good methodology to put into practice the approach will help modellers to use it correctly in real scenarios.

1.4 Research methodology of the thesis

The research methodology followed along this research is the *Design research cycle* developed by *Takeda et al.* in 1990 [42]. It is a cognitive model of design and describes how to solve an existing problem through five subprocesses, i.e., the *awareness-of-problem* subprocess, the *suggestion* subprocess, the *development* subprocess, the *evaluation*

subprocess, and the *conclusion* subprocess. The next list describes the activities that should be done within each subprocess:

1. **Awareness of a problem:** Study and identify the problems that appear in the area object of this masters' thesis. In addition, the existing proposals and approaches that try to solve these problems are also described.
2. **Suggestion:** State and describe clearly the key concepts that need to be addressed to properly solve the problem identified in the previous subprocess
3. **Development:** Design a solution to overcome the problems identified in the previous subprocess.
4. **Evaluation:** Evaluate the developed solutions in various ways, such as structural computation, economic performance, simulation of behaviour, etc.
5. **Conclusion:** to decide if the developed solutions are acceptable and valid.

Figure 1.1 presents the *Design research cycle*. A single design cycle solves a specific problem. Nevertheless, it is possible that solving this problem causes new problems, which are left to be solved by other design cycles. Therefore, a whole design process is represented as a set of single design cycles, which are connected with each other.

1.5 Context of the thesis

This masters' thesis has been developed at the research center Centro de Investigación en Métodos de Producción de Software (PROS)² of the Universitat Politècnica de València³.

This work has been developed within the following research projects:

²<http://www.pros.upv.es>

³<http://www.upv.es>

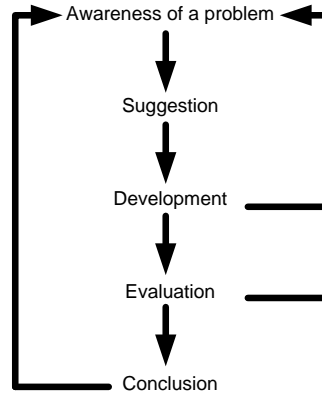


Figure 1.1: Design research cycle (*Takeda et al, 1990*)

OPEES: Open Platform for Engineering of Embedded Systems. Proyecto ITEA 2 con referencia TSI-020400-2010-36.

EVERYWARE: Construcción de Software Adaptativo para la Integración de Personas, Servicios y Cosas usando Modelos en Tiempo de Ejecución. Proyecto del Ministerio de Ciencia e Innovación con referencia TIN-2010-18011, co-financiado con fondos FEDER.

1.6 Outline

The remainder of this document is organized as follows:

Chapter 2: presents the evaluation framework stated to evaluate the approaches developed in the BPM field to cope with BP variability at the modelling level, specifically during the analysis/design phases of the development process.

Chapter 3: provides an overview of the most well-know approaches developed in the BPM field to cope with BP variability at the

analysis/design phase. Furthermore, these approaches are evaluated against the evaluation framework defined in the previous chapter.

Chapter 4: provides a detailed explanation of the proposed approach to deal with variability in business processes. More concretely, it presents the different proposed models used in the proposal.

Chapter 5: presents the methodology proposed to face the task of modelling BP variability using the presented approach.

Chapter 6: introduces the running example used to exemplify and illustrate the proposal.

Chapter 7: presents the conclusions of this masters' thesis and outlines further work that can be carried out in order to extend the presented approach. Furthermore, this chapter lists the research publications that have been produced during the development of this work.

List of the acronyms used in this document

BP:	Business Process
BPM:	Business Process Management
SPL:	Software Product Lines
BPML:	Business Process Modelling Language
BPMN:	Business Process Modelling Notation
C-EPC:	Configurable Event-driven Process Chain
Provop:	PROcess Variants by OPTions
PESOA:	Process family Engineering in Service-Oriented
EC:	Evaluation Criteria
BVR:	Base-Variation-Resolution model
CFT-VM:	Control flow & tasks variation model
TRE-VM:	Task related elements variation model
CIT:	Consellería de Infraestructuras, Territorio y Medio Ambiente

2. Evaluation framework

This chapter states the evaluation framework that gathers the set of concepts that should be addressed when modelling BP variability. This framework can be used against the existing approaches (that deal with BP variability modelling) and the presented one in this work in order to determine their suitability for BP variability modelling.

The remainder of the chapter is structured as follows: Section 2.1 describes the impact that variability has within the analysis/design phase of the BP lifecycle. Then, section 2.2 presents the evaluation criteria that make up the framework and that it is used to evaluate the different approaches. Finally 2.3 explains the conclusions of the chapter.

2.1 Variability within the BPM analysis/design phase

In order to state the relevant concepts when dealing with BP variability modelling, in this section, we provide an overall understanding of what is the impact that variability has over the modelling stage of the BP lifecycle.

The BP lifecycle consists of four different phases that are related to each other. These phases are organized in a cyclical structure, show-

ing their logical dependencies. Nevertheless, these dependencies do not imply a strict temporal order in which the phases need to be executed. Many design and development activities are conducted during each of these phases, and incremental or evolutionary approaches involving concurrent activities in multiple phases are common nowadays.

Figure 2.1 presents the BP lifecycle described by *Weske* in 2007 [47].

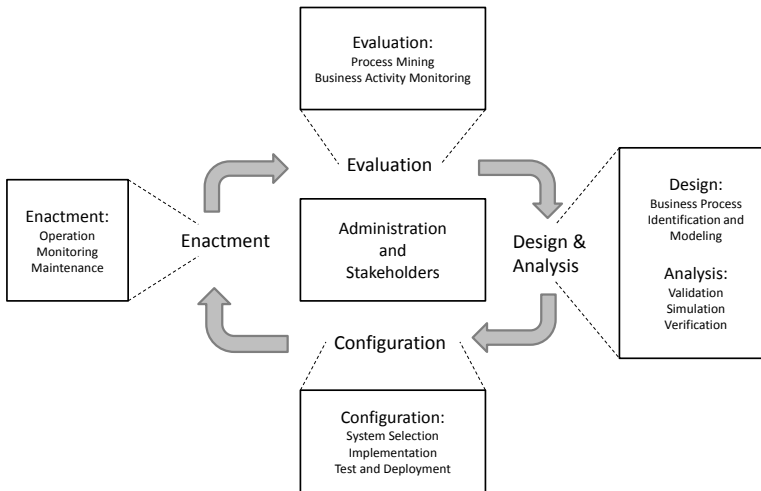


Figure 2.1: Business process lifecycle (*Weske*, 2007)

Within these four phases that make up the BPM lifecycle, in this work, we focus on the first phase which deals with BP analysis/design¹. Specifically, we reflect on the impact of dealing with BP variability in this phase. Concretely, during the analysis/design phase, and based on domain requirements, BPs are identified, reviewed, validated, and represented by means of BP models. These models are built by means

¹Business proces modelling is the main subphase during process analysis and design

of BPMLs (Business Process Modelling Languages) such as Petri nets, EPC, YAWL, BPMN or UML Activity Diagrams. However, the original constructs provided by these languages do not provide enough expressiveness to properly deal with BP variability (e.g. the Boolean conditions included in the BP models do not allow the nature of the different choices to be described [43]). According to [33], “Being aware of variability and dealing with it consciously is an important prerequisite of variability modelling”, therefore, BPMLs should allow model variability as such to be represented, bringing the concepts related to variability as first order concepts of the language.

Within the Software Product Lines (SPL) field, where variability management constitutes a key aspect, the questions that may be set out to characterize variability are: (1) what does vary?, (2) why does it vary?, and (3) how does it vary? [33]. The first question refers to the precise identification of the items or properties of the real world that are variable. In terms of a BP representation these items refer to the different language elements that make up a BP model, that is language elements such as tasks, control flow, roles, documents or even compositions of these basic elements. The second question refers to the different reasons that make an item or property to vary. In a BP model these reasons are usually represented by the conditions which are determined by the value of the associated variables. Finally, the third question refers to the different shapes that a variability subject can adopt. In terms of a BP model these shapes refer to the different alternatives that can be applied in a specific part of a model.

However, in addition to answering these questions, it is also important to follow a strategy that facilitates the management of these variability issues. This is better performed following a separate approach where model individualities are represented separately from model commonalities. This separation allows modellers focusing on model variants and also reducing the impact that variant evolution can have over the BP model.

Based on this set of considerations, next section presents, in the shape of an evaluation framework, the set of concepts and issues that need to be considered when dealing with variability in BP at the mod-

elling level.

2.2 Evaluation Framework

Modelling BP variability is a topic that has already been studied in the BP field. Nowadays, there already exist different works that define “metrics” for process models (see [28] for a detailed review). The metrics provided in these works allow measuring different aspects of BP models such as their size (e.g. based on the number of nodes or the hierarchy levels) or length (e.g. based on the longest path from an start node to an end node). However, the aim of these metrics is the evaluation of the resulting BP models and not the language used for that purpose. In our case, since we are not focused on the resulting models, but on the suitability of the approaches for BP variability modelling, these metrics cannot be used.

Concretely, based on the discussion of the previous section, we need to identify the set of concepts that BPMLs need to address in order to properly deal with variability during the BPM analysis/design phase. On the one hand, *Hallerbach et al.* stated in [21] a set of requirements for the definition, adaptation, and management of process variants². These requirements were identified based on the conduction of several case studies and gather the properties needed to reduce both efforts and costs of process variant management. However, these requirements are too wide and general to perform a more concrete and objective evaluation about variability in BPs. Therefore, a more detailed and specific evaluation criteria needs to be defined. On the other hand, *Pohl et al.* defined in [33] the set of key concepts that need to be supported to model variability properly. Nevertheless, since this work was focused on variability in Software Product Lines, these requirements need to be mapped and extrapolated to the business process field.

Based on these two previous works, we have proposed a more detailed and complete set of concepts that defines the set of properties that should be satisfied by any approach to deal properly with BP vari-

²variations of a general business process

ability during the analysis/design phase. These properties have been defined as an evaluation criteria within an evaluation framework. This framework allows us not only to perform an objective evaluation of the approaches, but also to identify their strengths and weaknesses. The criteria have been organized in two building blocks which refer to: (1) concepts and (2) quality factors. While the first block gathers a set of concepts that need to be addressed by BPMLs to properly deal the BP modelling task, the second block gathers desirable properties that contribute positively to improving the BPML with regard to its adoption.

- **Concepts.** In order to bring variability concepts as first-class constructs, BPMLs should provide the mechanisms that allow the specification of the following concepts.
 - **EC1. Variation point**
A variation point is defined as a precise position within a process model that admits different possibilities according to the current context or situation.
 - **EC2. Process fragment**
A process fragment is defined as an alternative that can be applied in a specific *variation point* for a particular context.
 - **EC3. Process fragment context**
A process fragment context is defined by the set of domain variables that make a particular process fragment to be instantiated.
 - **EC4. Process fragment relationships**
A process fragment relationship is defined as a constraint between two or more *process fragments* that establishes the proper use of the *process fragments* involved within a specific context (e.g. mutual exclusion or implication).
 - **EC5. Language support regarding variability**
The support provided by the language to specify variability should include any of the elements that make up a process model (control flow, physical objects, data, roles, events, etc.).

- **EC6. Process context regarding variability**

The process context regarding variability is defined as the set of domain variables that have an impact on process variability design and/or execution. The value of these variables determines the current state of a BP according to BP variability.

- **EC7. Variation point resolution time**

The variation point resolution time is defined as the moment when the related variation point is going to be solved, which can be either at design time or at run time. The language should allow modellers to distinguish between *variation points* whose resolution depends on the initial context (design time) or on the current context of an instance process (run time).

- **Quality factors.** This block gathers a set of quality factors that are desirable in the different BPMLs to ensure their successful adoption.

- **EC8. Flexibility**

In order to deal with *process fragment* evolution, the language should provide mechanisms to easily make changes over the model, in particular those parts related to *process fragments*.

- **EC9. Scalability**

The language should provide techniques that allow BP models to be evolved without losing the ability to handle a great number of model variants³.

- **EC10. Legibility**

The language should provide graphical elements to easily distinguish between model commonalities and model individualities.

- **EC11. Understanding**

The language should provide abstractions close to the con-

³a model variant represents an individualization of the process model

cepts used to represent BP variability. A correct abstraction of these concepts will result in more easily understandable models.

– **EC12. Low learning curve**

The language should be easy to learn. It is easier to learn a language when it is based on a well-known and commonly used language. Moreover, if this well-known language is a standard language, we can take advantage of its benefits such as existing tools, guidelines, etc.

– **EC13. Separation of Concerns**

The language should allow modellers to manage *process fragments* separately from the base model (the model that gathers commonalities). This separation contributes to reduce the impact that changes have on the model, reuse the existing *process fragments*, and improving the understanding of the model ([32],[41]).

– **EC14. Tool support**

The language should be supported by a tool since this support is usually critical for its adoption. In addition, it facilitates the management of the variability, specifically during the analysis/design phase.

– **EC15. Guideline support**

The existence of guidelines that assist users during variability management in BP models facilitates the learning process and the use of the language. Specifically, these guidelines are desirable during the analysis/design phase while defining model commonalities, process fragments, and configuration. As a consequence, these guidelines turn the modelling process into a more objective task, making it less dependent on the modellers's skills.

2.3 Conclusions

In this chapter, we have introduced the concepts that need to be addressed when modelling BP variability. We have presented them as evaluation criteria within an evaluation framework. This framework is the result of an analysis made on the requirements that need to be addressed when dealing with variability at the analysis/design phase. For the definition of this framework, we have taken into account not only variability concepts, but also quality factors that contribute to the successful adoption of an approach.

In addition, the framework has been designed in such a way that it can be applied to any BPML that deals with variability modelling and, even more, to other variability modelling languages since it is not focused on BPMLs aspects, but on how languages deal with variability. Another important aspect of the framework is that it can also help organizations in the decision of which approach better suits to their needs (the BP modelling process of each organization may vary according to its characteristics, e.g. starting the modelling process from scratch, level of expertise of its business modellers, etc.). Moreover, the framework can be applied to any approach that deals with variability at the modelling level since it is not focused on specific aspects of BPML but focused on the variability concepts themselves.

More concretely, in this work we have used the defined framework to evaluate the different existing approaches that deal with BP variability at the modelling level. A complete fulfillment of the presented evaluation criteria would mean that the approach is able to provide BP variability modelling coverage in a wide sense. This coverage implies considering variability modelling issues related for instance to model structure and behaviour, language elements (not limiting just to control flow elements) or model resolution time.

3. State of the Art

This chapter introduces the different existing approaches that deal with variability at the modelling level and have been developed within the field of the BPMN. We have considered those approaches that have been designed specifically to deal with BP variability modelling, independently of the mechanisms used for this purpose. The objective is to provide a clear picture of the different possibilities while variability modelling and also understanding and evaluating their suitability according to the context of use. For this purpose, we have analyzed these approaches against the evaluation framework presented in the previous chapter.

The remainder of the chapter is structured as follows: Sections 3.1, 3.2, and 3.3 study and evaluate the C-EPC, PESOA, and Provop approaches respectively. Then, section 3.4 provides an analysis of the evaluated approaches. Finally, section 3.5 concludes the chapter.

3.1 C-EPC (Configurable EPC)

C-EPC is an extension of EPC (Event-driven Process Chain) that includes new constructs to represent variability in reference process models [38]. These new constructs allow multiple model variants to be rep-

resented within a single model which needs to be configured¹ prior to its deployment. This single representation of multiple model variants is achieved by combining the use of *configurable nodes* with *configuration requirements* and *guidelines*. While *configurable nodes* represent functions and connectors, *configuration requirements* and *guidelines* state, by means of logical predicates, the valid configurations of the model.

Even though other approaches such as C-YAWL [43] and Feature-EPC [45] differ from C-EPC in the techniques used to obtain an individualization of the model (*Hiding and Blocking* for C-YAWL and *Feature Models* for Feature-EPC), all share the idea of having a single model to represent all model variants. However, since C-EPC is widely known and extended, we have decided to evaluate it as the representative of this type of approaches.

The evaluation of the C-EPC approach against the evaluation criteria defined in the previous chapter is presented below:

- EC1. *Variation point*: It is possible to clearly identify them by means of *configurable functions* and *connectors*.
- EC2. *Process fragment*: It is not possible to clearly identify which process fragments are being considered in the process from the model. This information is scattered among model functions, events, and connectors.
- EC3. *Process fragment context*: It is not possible to specify this context in terms of domain variables. Specifically, the process fragment context is defined in terms of the configuration of other model elements by means of *configuration requirements* and *guidelines*. This hinders the identification of the specific underlying condition(s) that make a specific process fragment to be instantiated.
- EC4. *Process fragment relationships*: It is not possible to explicitly define relationships between process fragments since in

¹La Rosa defined a questionnaire-driven approach in [36] to reduce the complexity that the C-EPC model configuration task entails.

C-EPC these are not defined as such. Moreover, although *configuration requirements* define relationships between functions and connectors, this information is scattered throughout the entire model which implies that these relationships are neither clear nor transparent to the modellers.

- EC5. *Language support regarding variability*: It is possible to define variability related to different language elements such as control flow, functions, and connectors. In addition, the language variability scope was extended in [37] to also include roles and physical objects.
- EC6. *Process context regarding variability*: It is not possible to define the process context regarding variability in terms of domain variables. C-EPC does not provide enough expressiveness to explicitly define the set of variables that determine the process context. Again, it is scattered throughout *configuration requirements* and defined in terms of the configuration of other elements of the model.
- EC7. *Variation point resolution time*: C-EPC is designed for model configuration prior to model deployment, and the expressiveness provided to deal with variability is not used to represent the variability that may appear during process execution. Therefore, this evaluation criteria does not apply to the C-EPC language.
- EC8. *Flexibility*: It is not possible to evolve or change C-EPC models easily since they are built as integrated representations where model variants cannot clearly be identified.
- EC9. *Scalability*: It is not possible to evolve the model when handling a great number of model variants. Even though *configurable nodes* allow a great number of model variants to be represented jointly, this joint representation makes variant evolution difficult since one change in a specific part of the model may affect related configurable model elements.

- EC10. *Legibility*: It is partially possible to distinguish model commonalities and individualities since model variants are not easily extracted from the diagram. However, model elements that are related to model variability (i.e. *configurable nodes* and *configuration requirements* and *configuration guidelines*) can be identified graphically in the model.
- EC11. *Understanding*: It is possible to understand C-EPC since it provides the appropriate abstractions to represent model variants. These abstractions are *configurable nodes* (which can be valued as included, excluded, or conditionally skipped), *configuration requirements* and *configuration guidelines*.
- EC12. *Low learning curve*: It is partially possible to learn the language quickly. The learning process of the new language elements is fast due to the closeness to the original EPC language elements (*configurable nodes* constitute small graphical variations to the original nodes). However, this speed does not apply to the definition of *configuration requirements* or *configuration guidelines* which requires some knowledge about logical predicates.
- EC13. *Separation of concerns*: It is not possible to manage process fragments separately from the base model. The C-EPC approach forces model commonalities and individualities to be handled jointly.
- EC14. *Tool support*: There is no available tool to perform the design of C-EPC models. Nevertheless, [29] presents a tool (C-EPC Validator) to check the compliance of a C-EPC model configuration against *configuration requirements* and *configuration guidelines*.
- EC15. *Guideline support*: Despite the fact that there exist documentation about the language [38], this documentation does not provide clear guidelines that assists modellers during the model design and configuration tasks.

3.2 Variant-Rich Process Models (within the PESOA project)

PESOA (www.pesoa.org) is a cooperative project that was carried out by a group of companies and academics whose main objective was the investigation of an approach for the development and customization of families of process-oriented software. As a result, by means of focusing on the design level and taking into account the relevance of the reusability aspect, a set of *basic* and *composite* variability mechanisms were identified [34]. The *basic* set includes (1) encapsulation of varying subprocesses, (2) addition, replacement, omission of encapsulated subprocesses, (3) parameterization, and (4) variability in data types. The *composite* set includes (5) inheritance, (6) design patterns, and (7) extensions/extension points. This set was transferred to different languages such as UML Activity Diagrams, UML State Machines, BPMN, and Matlab/Simulink.

The evaluation of the PESOA approach against the evaluation criteria defined in the previous chapter is presented below:

- EC1. *Variation point*: It is possible to identify them in the model by means of the use of the «VarPoint» label (stereotype), which is attached to those places (i.e., tasks) where variability may occur. According to the semantics associated to process fragment substitution, these labels may vary into one of the following stereotypes: «Abstract» and «Null».
- EC2. *Process fragment*: It is possible to identify process fragments in the model by the introduction of the «Variant» stereotype, which is attached to those tasks that can fit in a *variation point*. This stereotype may change into the stereotypes «Default», «Alternative» and «Optional» depending on the process fragment behaviour.
- EC3. *Process fragment context*: It is partially possible to express it since the context is not explicitly represented by domain variables, but it can be represented by means of features which are

associated to the different process fragments defined in the model.

- EC4. *Process fragment relationships*: It is partially possible to specify relationships between process fragments due to the limited support provided. Only two different types of relationships are defined: *implementation*, to associate possible resolutions to the variation points; and *inheritance*, to express alternative behaviour which adds or removes elements regarding specific rules.
- EC5. *Language support regarding variability*: It is not possible to support variability issues beyond control flow and tasks. It does not take into account the variability associated to other language elements such as roles, objects or events.
- EC6. *Process context regarding variability*: It is partially possible to determine the current state of a BP regarding variability by evaluating the values of the features associated to the different process fragments included in the model.
- EC7. *Variation point resolution time*: It is not possible to distinguish whether a variation point needs to be solved at design time (depending on the initial context) or at run time (depending on the current instance context).
- EC8. *Flexibility*: It is not possible to evolve or change process fragments easily. Since process model commonalities and individualities are defined together, changes in process fragments may imply reconsidering other parts of the model, which makes process fragment evolution difficult.
- EC9. *Scalability*: It is not possible to properly handle the evolution of a great number of model variants since process model commonalities are modelled jointly with model individualities. The higher the level of the variants that a model has, the more difficult they become to evolve.
- EC10. *Legibility*: It is possible to graphically differentiate model commonalities from individualities. Even though process frag-

ments are integrated in the model, the enrichment of task/subprocess elements by means of stereotypes allows this differentiation.

- EC11. *Understanding*: It is not possible to easily comprehend the language. The stereotypes introduced to represent concepts related to BP variability do not seem suitable. Their associated semantics is broader than the semantics of the word that represents the stereotype. For instance, the «Null» stereotype indicates not only task/subprocess optional behaviour (behaviour also associated to the «Optional» stereotype) but also restricts the number of possible process fragments associated to the variation point to one process fragment.
- EC12. *Low learning curve*: It is not possible to learn the language easily. The broad semantics associated to the introduced stereotypes takes time to learn and use properly, which results in a steep learning curve.
- EC13. *Separation of Concerns*: It is partially possible to separate model commonalities from individualities. The PESOA approach forces them to be handled jointly. However, variation points and their different alternatives are labelled through stereotypes that allow individualities to be distinguished and managed separately from commonalities.
- EC14. *Tool support*: The approach has been implemented as an Eclipse plugin [6]. This tool provides support for configuring a feature diagram as well as for applying the selected configuration to the process model.
- EC15. *Guideline support*: [6] provides a methodology and guidelines for the application of the PESOA approach.

3.3 Provop (PROcess Variants by OPTions)

Provop is an operational approach for managing large collections of process variants during the process life cycle. It was motivated by the fact

that a “process variant can be created by adjusting (configuring) a given process model to a given context” [20]. These variant-specific adjustments are expressed by means of a set of high-level *change operations* (INSERT, DELETE, MOVE and MODIFY) [23]. Furthermore, Provop allows more complex process adjustments by grouping multiple *change operations* into so-called *options* [24]. Thus, a specific process variant is determined (configured) by applying one or more of these *options* to the process model. The *options* required to configure a process variant are chosen when the process context is evaluated. Provop provides a model for capturing this process context by means of *context variables*, which represent different domain dimensions of the context.

The evaluation of the Provop approach against the evaluation criteria defined in the previous chapter is presented below:

- EC1. *Variation point*: It is partially possible to identify them within the model. When it refers to INSERT and DELETE model changes, variation points can be clearly identified by means of *adjustment points*, which are represented graphically in the diagram by black diamonds. However, this does not occur when the operation refers to MOVE or MODIFY changes.
- EC2. *Process fragment*: It is possible to identify them clearly by means of the specification of the proposed *change operations*.
- EC3. *Process fragment context*: It is possible to represent it by means of the context variables that are defined by a name, a description, a value range, and a mode (*static* or *dynamic* depending on whether or not its value changes during the execution of the process).
- EC4. *Process fragment relationships*: It is possible to define relationships such as implication, mutual exclusion, application order, hierarchy, and at-most-n-out-of-m-options [22] between the *options* that make up a model variant.
- EC5. *Language support regarding variability*: It is not possible to manage variability aspects beyond language elements such as

control flow, functions, and connectors.

- EC6. *Process context regarding variability*: It is possible to express it by means of context variables, which capture the different domain dimensions of the context.
- EC7. *Variation point resolution time*: It is not possible to distinguish when variation points are solved. Even though Provop provides context rules for determining the process fragments that can be applied for a specific variation point, it cannot be explicitly defined whether variation points depend on the initial process context or on the current process instance context.
- EC8. *Flexibility*: It is possible to evolve or change models easily by means of the definition of new *options*. For instance, to obtain a new process variant, it is only necessary to define a new set of *options*.
- EC9. *Scalability*: It is possible to handle and evolve a great number of model variants thanks to the separate representation used in Provop. Moreover, this evolution is also facilitated by the *change operations* and *options*.
- EC10. *Legibility*: It is possible to clearly identify model commonalities and individualities since they are specified separately. Model commonalities are represented in a base model while model individualities are represented separately by means of *options*.
- EC11. *Understanding*: It is possible to understand the language since Provop provides an adequate set of abstractions to represent model variability. These abstractions are *adjustment points* (to identify variation points within the model), *change operations* (to define the difference between the basic process model and its individualization), and *options* (to group *change operations*).
- EC12. *Low learning curve*: It is possible to learn Provop easily due to the simplicity of the *change operations* (INSERT, DELETE, MODIFY and MOVE) and how the model variants are built.

- EC13. *Separation of Concerns*: It is possible to manage process fragments separately from the base model. The Provp approach provides a proper separation of concerns since process fragments are clearly defined and identified by means of *change operations*.
- EC14. *Tool support*: A proof-of-concept prototype has been implemented based on the ARIS Business Architect modelling tool. The prototype introduces the facilities for process variant configuration and management [23].
- EC15. *Guideline support*: There are clear guidelines that explain step by step how to model and configure BP models [24]. Also, there exists documentation ([20], [23]) that gives a complete and formal description of the approach.

3.4 Analysis of the evaluation results

In the previous section, we have evaluated the existing approaches that deal with variability at the analysis/design phase. The results of this evaluation are summarized in Table 3.1 and 3.2, where a set of symbols has been used to specify the fulfillment of each evaluation criterion: a “+” indicates that the approach completely fulfills the evaluation criterion; a “-” indicates that it is not fulfilled; a “+/-” indicates that it is partially fulfilled and “na” indicates that the criterion is not applicable to the approach.

3.4.1 Concept analysis

The results of the evaluation regarding the concepts defined before are presented in Table 3.1.

According to the results shown in this table, none of the evaluated approaches provides complete support to the entire set of evaluation criteria considered in the concepts block.

In C-EPC, the reason for not satisfying EC2, EC3, EC4, and EC6 is mainly due to the fact that variability concepts are not introduced into

	Concepts						
	EC1	EC2	EC3	EC4	EC5	EC6	EC7
C-EPC	+	-	-	-	+	-	na
PESOA	+	+	+/-	+/-	-	+/-	-
Provop	+/-	+	+	+	-	+	-

Table 3.1: Summary of the approaches evaluation regarding concepts

the language as first-class elements. For instance, process fragments (represented by EC2) are not explicitly defined in the model and have to be deduced by analyzing the different configurations defined for each configurable node (function or connector). As a consequence, we cannot explicitly define relationships between process fragments (EC4), either. The same occurs with process fragment context (EC3) and process context (EC6). Even though variant conditions are specified by means of *configuration requirements* and *configuration guidelines* associated to *configurable nodes*, these conditions are expressed in terms of the configuration of other model elements, not making explicit the underlying condition for a specific variant to be instantiated.

In contrast to C-EPC, in PESOA and Provop we find mechanisms to specify variability as first-class elements of the language. However, both approaches delimit their scope in order to address variability regarding control flow and tasks (EC5), leaving apart the variability associated to other language elements that can occur such as roles, objects or events. In addition, PESOA and Provop are conceived to deal with variability that can be solved either at design time (variability that depends on the initial context of the process) and at run time (variability that depends on the current context of a process instance). However, neither of these approaches provides mechanisms to differentiate in the model between these two types (EC7). This differentiation would delimit the set of decisions that need to be taken prior to model deployment.

3.4.2 Quality factors analysis

The results of the evaluation regarding the quality factors are presented in Table 3.2.

As this table shows, only the Provop approach provides a complete fulfillment of the evaluation criteria included in the quality factor block (EC8-EC15). A key factor for this fulfillment is the application of a separate approach where model commonalities are specified apart from model individualities. This approach contributes positively to the flexibility (EC8) and scalability (EC9) of the model. On the contrary, the joint approach followed by C-EPC and PESOA contributes negatively to the flexibility (EC8) and scalability (EC9) factors of the models. The application of the joint or separate approach is also directly related to the fulfillment of the separation of concerns criterion (EC13).

	Quality factors							
	EC8	EC9	EC10	EC11	EC12	EC13	EC14	EC15
C-EPC	-	-	+/-	+	+/-	-	-	-
PESOA	-	-	+	-	-	+/-	+	+
Provop	+	+	+	+	+	+	+	+

Table 3.2: Summary of the approaches evaluation regarding quality factors

With regard to the understanding criterion (EC11), the overload semantics associated to the new concepts introduced in PESOA hinders the understanding of the resulting model. In addition, this overload also has a negative impact on the learning curve of the introduced concepts (EC12).

Finally, the lack of tool and guideline support (EC14 and EC15) in C-EPC is also worth mentioning. The existence of tools and guidelines contributes to improve the modelling process by turning it into a more objective task that is independent of the modeller's skills or experience. Not providing such support favours the construction of error-prone models that do not ensure the quality of the resulting model.

3.5 Conclusions

In this chapter, we have applied the framework defined in section 2 to three BP modelling approaches that deal with variability in a different manner.

However, after analyzing these approaches, from a modelling point of view, we can determine that none of them provides support for the entire set of variability concepts identified. This hinders putting into practice any of these approaches when the required variability is broader than the one provided by them. As a result, this forces their combination to support variability in a broad sense. However, this combination is not easy to achieve since the mechanisms used by each of the approaches are not easily and directly transferred to other BPMLs. Therefore, it has been proved that an approach that provides such complete support is necessary.

After this evaluation, we can extract that this new approach should cover the following issues:

- the required expressiveness to clearly define what is variable in a BP model, which alternatives fit in such variation, why each alternative is selected and when a variation point should be solved.
- the ability to extend such expressiveness to the set of BPML elements in which we can find variations. This set refers to tasks, control flow, roles, objects and any other language element introduced by the BPML used (e.g. events in BPMN).
- the ability to specify separately variability aspects from model commonalities. This separation allows (1) improving the management and reuse of both, model commonalities and model individualities, and (2) reducing the impact of changes may have on other parts of the model.

4. A modelling approach to support business process variability

As we have seen in the previous chapter, there already exist different approaches that deal with BP variability at the modelling level. Nevertheless, the main problem of these approaches is that they do not provide complete support for the set of concepts identified in chapter 2. This chapter presents in detail a modelling approach that covers completely this set by bringing variability as first order aspect of the modelling process.

The remainder of the chapter is structured as follows: section 4.1 introduces the approach outlining the set of models (base model, variation model, and resolution model) that make up the approach and that are deeper explained in the following sections. Section 4.2 describes the *Base model*, where BP commonalities are defined. Then, section 4.3 presents the *Variation model*, where BP individualities are gathered. Section 4.4 describes the *Resolution model* where the BP model configuration is described. Section 4.5 presents the evaluation of the approach against the evaluation framework defined in chapter 2. Finally, section 4.6 concludes the chapter.

4.1 Overview of the approach

As it is discussed in [24], using conditional branching to represent and define together all model variants into one single model results in huge models. As a result, these models are less scalable and difficult to understand and maintain. Moreover, models are cluttered with variability specification where variant particularities are mixed with normal branching. This does not allow modellers to distinguish if the branching conditions are variants-specific or part of the common logic. For this reason, we propose to decouple model variant particularities from BP model commonalities. Thus, we bring variability concepts as first order aspect of the modelling process. For such purpose, we make use of the ideas and techniques from the Software Product Line community.

SPL community is basically focused on the development of techniques for the creation of a portfolio of similar software systems. The objectives this community is looking for are: *commonality capitalization* and *variation management* [12]. Addressing properly these two aspects allows reducing time, effort, cost and complexity of creating and maintaining a product line of similar software systems. On the one hand, *commonality capitalization* avoids duplication and divergence of shared assets. On the other hand, *variation management* allows making explicit location, rationale and dependences for variations. Thus, by transferring SPL ideas we are gaining in model reusability and scalability.

In this context, we made use of the Base-Variation-Resolution (BVR) modelling approach defined by *Bayer et al.* [6]. Specifically, the authors assert that the possible process variants and their different configurations should be parts of different models. Therefore, variation can be documented explicitly and the process model is not overloaded with variant-specific information.

Specifically, based on this approach and transferring its ideas to the BP field, we overcome BP model construction in three different parts. These parts refer to: (1) model commonalities (represented in the *Base model*), (2) model individualities (represented in the *Variation model*), and (3) model configuration (represented in the *Resolution model*). The

techniques used to implement such approach are *feature models* [27] and *Common Variability Language* (CVL, the new standard being developed for variability modelling) [25] [14]. The combination of both techniques provides enough expressiveness to represent and manage variability in BP modelling. The set of models proposed in the approach as well as their relationships are depicted in Fig. 4.1.

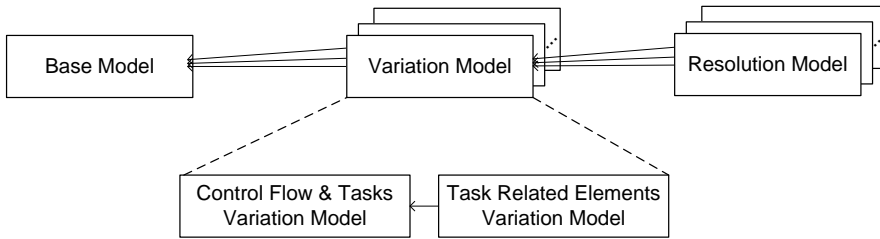


Figure 4.1: Overview of the approach

The main advantage of this approach is that there may be more than one variability model for each base model, as indicated in figure 4.1. Thus, modellers can define associated to one *Base model* (e.g. merchandise delivery BP), as many *Variation models* as they require. Ideally, modellers would build a different *Variation model* for each domain/industrial sector where the *Base model* is applied (e.g. health-care, education or banking) and then, within each *Variation model* they would define all the different choices that are possible in a specific domain. Therefore, this set of models would represent different variability scenarios at different levels regarding a common shared *Base model*.

Nevertheless, BP models can be seen from a number of perspectives, including the control-flow, the data and the resource perspectives [26]. When dealing with variability in a wide sense, it is necessary to take into account that this variability may appear not only regarding tasks and control flow modelling elements, but also in other different modelling elements (for instance when variability refers to roles and documents).

Again, trying to gather all the possibilities within the same model, results in a scalability problem [22]. Thus, to overcome this problem, we propose a refinement of the BVR *Variation model* based on the elements provided by BPMLs. This refinement results in two sub-models which refer to: (1) control flow and tasks (represented in the *Control flow & tasks Variation model*) and (2) task related elements such as roles and objects (represented in the *Task related elements variation model*). To represent such refinement we make use of two different techniques: (1) CVL to represent the *Control flow & tasks variation model* and (2) feature models for the *Task related elements variation model*. Another important factor is that, with this refinement, the approach is language independent, which allows modellers to apply it to any BP modelling language. However, for explanatory purposes, we have taken the *Business Process Modelling Notation* (BPMN) since (1) it is the standard language for BP definition and (2) collaborative works with local public administration (Conselleria de Infraestructuras, Territorio y Medio Ambiente, CIT) demand support for variability issues in such notation.

Finally, the last model is the *Resolution model*. It is the place where modellers specify which alternatives, from those defined in a specific *Variation model*, constitute a valid model variant of the process for a specific context. Therefore, there are as many *Resolution models* as valid model variants exist.

A deeper explanation of all of these models is presented below.

4.2 Base model

The *Base model* is the model where modellers specify all the commonalities shared by all model variants. Following the BVR approach, this model is built by using modelling elements of a domain specific language [6] (a specific BPML, BPMN in our case). In addition, in this model modellers also define those parts of the model subject to vary (i.e. *variation points*, named *placements* in terms of CVL) where different alternatives can be applied depending on a specific context. These *placements* represent black boxes delimited by *boundary elements*, whose in-

stantiation can be solved either at design time (when the selection of an alternative depends on the initial context of the process - e.g. when the process is going to be configured for a specific domain such as health-care) or at run time (when the selection of an alternative depends on the current context of a process instance - e.g. in a hospital, the emergency procedure depends on the characteristics of the patient being assisted). In this work this differentiation has been made explicit by specializing the *placement* concept into *design-time* and *run-time placements*. This specialization has been done using two stereotypes: «design-time» and «run-time», respectively.

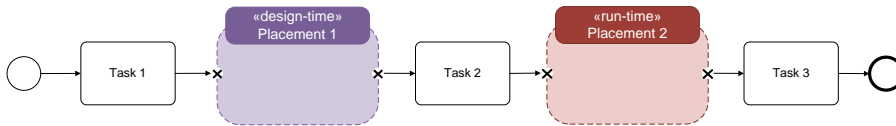


Figure 4.2: Example of the *Base model*

Figure 4.2 shows a generic example of the *Base model* described above. In it, two *placements* are depicted, one *design-time placement* (the one marked with the «design-time» stereotype), which will be solved at analysis/design time, and one *run-time placement* (marked with the «run-time» stereotype), whose resolution is performed during process execution.

4.3 Variation model

In contrast to the *Base model*, the *Variation model* gathers all the particularities introduced by each model variant, which can refer to any element of the model (e.g. tasks, control flow, roles, objects or events in BPMN). As we have already explained in the overview section, attempting to represent variability of each model element within a single model results in a scalability problem. For this reason we propose to deal with model element variability separately in two separate models

which are (1) the *Control flow & tasks variation model* and (2) the *Task related elements variation model*.

4.3.1 Control flow & tasks variation model

The *control flow & tasks variation model* gathers, for each *placement* found in the *Base model*, all the different choices (named *replacements* in terms of CVL) regarding process model control flow and tasks. These *replacements* are delimited by *boundary elements*. Thus, when we want to instantiate a *replacement*, we only have to bind these *boundary elements* with the ones presented in the *placement* of the *Base model* that is going to be solved.

In turn, within a *replacement* it is also possible to find variability (according to different alternatives that can appear) which is represented by a new nested *placement*. By nesting *placements* we are promoting the reusability of process fragments along the model.

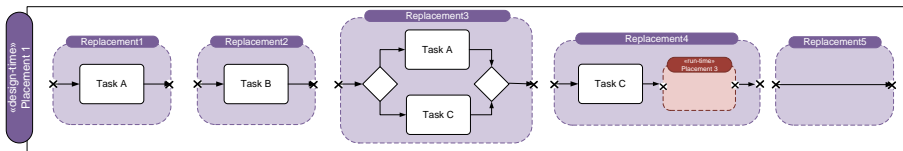


Figure 4.3: Example of the *Control flow & tasks variation model*

Figure 4.3 shows a generic example of the *Control flow & tasks variation model* described above. In it, we can see the five different *replacements* that can be instantiated in the *Placement 1*. In addition, *Replacement4* presents a nested *placement* to allow modellers to represent that different alternatives can be presented in this branch of the control flow.

Constraint model

There may exist semantic and structural dependencies (relationships) between *replacements* of different *placements*, e.g. the instantiation of a *replacement* excludes the instantiation of another *replacement* in another *placement*. For such purpose, associated to the *Control flow & tasks variation model* modellers can also define the *Constraint model*. This model gathers all the relevant constraints existing between all the *replacements* defined in the same *Control flow & tasks variation model*.

Two types of relations are considered in this model: *Requires* and *Mutual Exclusion*. On the one hand, the *Requires* relationship is a relation defined between two or more *replacements*¹ where the instantiation of one of them in a specific *placement* forces the instantiation of the related *replacements* in others *placements*. On the other hand, the *Mutual Exclusion* relationships is defined as a relation between two or more *replacements*¹ where the instantiation of one of them in a specific *placement* forces the no instantiation of the related *replacements* in other *placements*.

These relationships can be defined between:

1. *Replacements* whose instantiation depends on the initial context (*design-time replacements*).
2. *Replacements* whose instantiation depends on the current instance context (*run-time replacements*).
3. *Design-time replacements* and *run-time replacements*.

Furthermore, gathering these constraints between *replacements* and *placements* allows modellers to verify (ensure) the consistency and correctness of the *Resolution Models*, e.g. a resolution including two *replacements* that are mutually excluded according to the *Constraint model* should be forbidden and notified to the modeller.

Figure 4.4 shows a generic example that illustrates the *Constraint model* described above. In it, we can see four different *replacements* and

¹of the same *Control flow & tasks variation model*

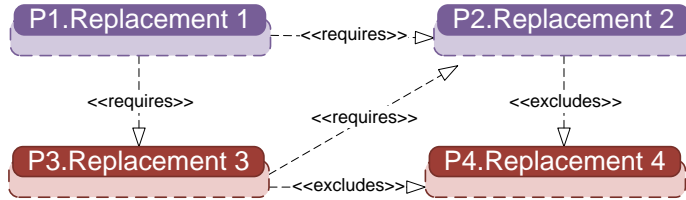


Figure 4.4: Example of the *Constraint model*

their different *requires* and *excludes* relationships. For instance, the instantiation of *Replacement1* requires the instantiation of *Replacement3*, which, at the same time, excludes the instantiation of *Replacement4*.

4.3.2 Task related elements variation model

The *task related elements variation model* gathers for each task of the process whose related modelling elements (e.g. roles or objects) are variable, the different alternatives that these related elements may have. To be able to express such variability without losing model understandability or scalability, we propose to build this model as a feature model. More concretely, we make use of a combination of the extensions provided by Czarnecki [15] and Riebisch [35] due to such combination results in a feature model with richer expressiveness. On the one hand, from the Czarnecki's extension we use the OR and XOR representation to define the different alternatives for a specific element and also to express choice constraints. On the other hand, from the Riebisch's extension we use the feature cardinalities and dependency relationships (*excludes* and *requires*) between alternatives defined by different features. Both notations are summarized in table 4.1.

Figure 4.5 shows a generic example of the *Task related elements variation model* described above. As it is shown in the figure, we propose to structure the feature model in four different levels as follows:

- On the top level modellers indicate the task whose related ele-







ICON	NAME	FORMAL REPRESENTATION
	Feature	f is the feature
	Default feature	f is the default feature
[n..m]	Cardinality	number of elements of the set
	Mandatory	$f_1 \Leftrightarrow f$
	Optional	$f_1 \Rightarrow f$
	Alternative	$(f_1 \vee f_2 \vee \dots \vee f_n \Leftrightarrow f) \wedge \forall_{i < j} \bar{A}(f_i \wedge f_j)$
	Or	$f_1 \vee f_2 \vee \dots \vee f_n \Leftrightarrow f$
---<<requires>>---	Requires	$f_i \Rightarrow f_j$
---<<excludes>>---	Excludes	$\bar{A}(f_i \wedge f_j)$

Table 4.1: Notation of the feature model

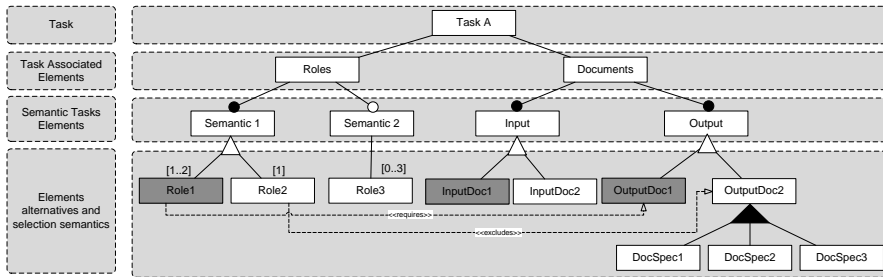


Figure 4.5: Example of the *Task related elements variation model*

ments are variable.

- The second level is used to categorize the set of task related elements that are going to be considered as variable; in particular in the example, we have included two categories which refer to *roles* and *documents*. However, depending on the expressiveness of the used BPML, this level could be extended with many categories as

we require by simply adding new features in the tree structure.

- The third level is used to specify the semantics that relates the type of element being considered with the task. For instance, modellers could define the participation of two or more *roles* in the same task but behaving differently (e.g. differentiating between a task performer and a task supervisor).
- The fourth level is used to specify all the different alternatives according to all the categories defined in the previous level. In addition, constraints between different task related elements can also be defined using the “requires” or “excludes” relationship constraint between features.

4.4 Resolution model

The set of models presented previously gather, in conjunction, many BP model variants. Specifically, the *Variation model* allows modellers to define the different alternatives that can be applied to a *Base model*. However, we still need to specify which alternatives’ combinations are valid for a particular context. This information is presented in the *Resolution model*.

We propose to structure this model in three blocks. While the first two blocks (*Applicable fragments* and *Selected features*) include the list of choices needed to derive a new model variant, the last one (conditions) includes the conditions that determine the selection of these choices. Specifically, the first block specifies, for each *design time placement* defined in the *base model*, the selected *replacements* from the *CFT-VM* that constitute a specific model variant. Note that not all the placements of the *base model* are mandatorily considered in a resolution (run time *placements* cannot be resolved before process deployment). However, if a *design time placement* is considered, just one *replacement* can be chosen for it. Regarding the second block, it gathers for each task whose related elements are variable, a configuration of its associated *TRE-VM*. This configuration is performed by (1) selecting

the appropriate features and (2) specifying their cardinality (if necessary). Finally, the third block defines the conditions that determine the selection of these *replacements* and features based on the current context. Concretely, conditions are specified by determining which context variables have to be evaluated and their values that define a specific process variant. In formal terms, a *resolution* R for a specific model variant *Configuration* is a tuple

$$R_{Configuration} = (\{(P, R)\}, \{(T, S)\}, \{(CV, V)\})$$

where

- $P \in BM$,
- $R \in CFT-VM$,
- $T \in (BM \vee CFT-VM)$,
- Σ is the set of T associated elements defined in the second level of the *TRE-VM* (e.g. roles and objects),
- $I_i = \{(x_0)_s, \dots, (x_j)_s\}$ where $(x)_s$ denotes that x is type of s such that s is in Σ ,
- $S = \bigcup \{\phi_i\} : \phi_i \subseteq I_i$ such that $0 \leq i \leq |I|$,
- CV is a *process context variable*,
- V is a *CV value* that defines the context of a specific process variant

Figure 4.6 shows a generic example of the *Resolution model* described above.

As it is shown in the figure, first of all the *applicable fragments* block is defined. There modellers specify the *replacement* (*Replacement1*) that is chosen for the *Placement1* (the only one of the *Base model* which is a *design time placement*). Second, the *selected features* block specifies

Resolution V_1	<p>Applicable fragments: Placement1: Replacement1</p> <p>Selected features: Task A.Roles.Semantic 1.Role1 Task A.Roles.Semantic 2.Role3 Task A.Documents.Input.InputDoc1 Task A.Documents.Output.OutputDoc1</p> <p>Conditions: processContext_variable1 = "X" processContext_variable2 = "Y"</p>
------------------------------------	--

Figure 4.6: Example of the *Resolution model*

the selected features that configure the roles and documents associated to the *Task A*. Finally, the *conditions* block includes all the context variables and their corresponding values that configure the specific process variant. Concretely, to apply the *Resolution V_1* , the context variables *processContext_variable1* and *processContext_variable2* have to be valued to *X* and *Y* respectively.

4.5 Evaluation of the approach

In the previous sections, we have described the proposed approach to deal with variability in BP at the modelling level. In order to make an objective evaluation of it, in this section we evaluate it against the evaluation criteria defined in chapter 2. This criteria helps us to determine if the approach provides proper support to deal with BP variability. This evaluation is presented below.

- EC1. *Variation point*: It is possible to clearly identify them by

means of the *placements* defined in the *Base model*. In fact, each *placement* constitutes a *variation point* since they determine the places within the model that admit different possibilities.

- EC2. *Variant*: It is possible to clearly identify which variants are being considered in the process. This information is gathered in the *Control flow & tasks variation model*. Concretely, each *replacements* defined in this model constitutes a *variant* by its own.
- EC3. *Variant context*: It is possible to specify it in terms of context variables in the *Resolution model*. Concretely, they are defined at the third block of this model to determine which context conditions have to be evaluated to instantiate specific *replacements*.
- EC4. *Variant relationships*: It is possible to explicitly define relationships between variants (*replacements*) in the *Constraint model*. These relations are *Requires* and *Mutual Exclusion*.
- EC5. *Language variant scope*: It is possible to define variability beyond control flow and tasks. This variability is gathered in the *Task related elements variation model* where it is possible to define the different alternatives for other modelling elements, such as roles or documents.
- EC6. *Process context regarding variability*: It is possible to define the process context regarding variability in terms of context variables. Again, this information is gathered in the *Resolution model*.
- EC7. *Variation point resolution time*: It is possible to distinguish in the model between variation points that need to be solved at *design time* from variation points that need to be solved at *run time*. This distinction is done by means of *placements* specialization with the *design-time* and *run-time* stereotypes.
- EC8. *Flexibility*: It is possible to make changes over the process model easily by only defining new *replacements*. For instance, to

obtain a new process variant, it is only necessary to define a new set of *replacements* that configure this new variant.

- EC9. *Scalability*: It is possible to handle and evolve a great number of variants thanks to the separate representation used in this approach. This separation allow managing independently variability from the rest of the process model.
- EC10. *Legibility*: It is possible to clearly identify model commonalities and individualities since they are specified in different models. Model commonalities are represented in the *Base model* while model individualities are represented in the *Variation model*.
- EC11. *Understanding*: It is possible to understand the languages used (CVL and feature models) since both provide an adequate set of abstractions to represent model variability concepts. These abstractions are the *placements* (to identify variation points within the model), the *replacements* (to define the alternatives that fit with in the *placements*), and the *features* (to define the alternatives for other modelling elements such as roles or documents).
- EC12. *Low learning curve*: It is possible to learn easily the approach due to the simplicity of the introduced concepts. Concepts such as 'replace' or 'selection' are intuitive and easy to learn.
- EC13. *Separation of concerns*: It is possible to manage model variants separately from the commonalities since they are defined and identified in different models.
- EC14. *Tool support*: Nowadays, there is no available tool to perform the design of this approach since we are now focused in defining the approach. Nevertheless, in the foreseeable future we plan to develop a prototype that supports the presented approach.
- EC15. *Guideline support*: The chapter 5 of this document provides a clear guideline to assist modellers during the model task using the presented approach.

As it is shown in this evaluation, the approach fulfils all the criteria defined (except EC14 referred to tool support). This means that the presented approach is able to provide BP variability modelling coverage in a wide sense. It considers variability modelling issues related to model structure and behaviour, language elements (not limiting just to control flow elements), model resolution time and quality factors that ensure its successful adoption.

4.6 Conclusions

In this chapter we have presented a modelling approach to deal with variability in BP. It is based on the BVR approach. This approach proposes to deal with variability modelling by defining separately model commonalities (*Base Model*), model individualities (*Variation Model*), and model configurations (*Variation Model*).

To cope with the variability that arises according to different modelling elements, we have refined the model that refer to model individualities. This refinement solves the scalability problem that appears when the model been represented involves variability in different modelling elements.

In addition, the approach addresses all the key issues that are required to cope properly with BP variability at the modelling level by considering the entire set of concepts outlined in section 2. The presented approach allows clearly specifying variation points, the different alternatives that fit in all these points, the relationships between these alternatives, the set of variables that define a specific variant and the resolution time in which a variation point needs to be solved. Moreover, the approach considers not only the variability that can occur regarding control flow but also the variability that appears regarding task related elements.

Another important factor of the proposed approach is that it is language independent, which allows modellers to apply it to any BP modelling language.

5. Methodology. Putting the approach into practice

This chapter presents a coarse-grained methodology that has been designed to put into practice the presented approach. This methodology is formed by a development process (in BPMN notation) in order to identify the different steps that should be performed when dealing with BP variability using the presented approach. Along the chapter, each step of the process is described in detail specifying the set of models that are produced.

The remainder of the chapter is structured as follows: Section 5.1 introduces the need of having a methodology within this context and it explains how to put into practice the approach. Section 5.2 outlines the development process of the defined methodology. Then, section 5.3 and 5.4 explain, respectively, in detail the specification of process commonalities and individualities. Section 5.5 describes the configuration stage of the development process. Finally, section 5.6 closes the chapter outlining its conclusions.

5.1 Introduction

The construction of BP models constitute a very complicated task where, sometimes, the results are not the expected ones [7]. In addition, when the model being constructed involves handling variability, as it is the case, the task becomes even more complicated. Therefore, to ensure the correctness of the resulting models, the construction of such models requires mechanisms that assist and guide modellers during this task [11].

Generally speaking, a methodology is a framework that is used to structure, plan, and control the process of developing something [17]. Methodologies in software engineering have been used since 1960 when the Software Development Methodology emerged. It gathers the steps to pursue the development of software in a very deliberated, structured and methodical way. In the same sense, a methodology for BP modelling contains a set of sequential steps that are followed and completed in order to obtain a BP model [39]. This methodology must ensure that, if followed properly, the resulting models are correct. This creates an environment under which modellers follow the required steps and consistently achieve the right result. Thus, modellers are guided and helped to ensure the correctness of the modelling process.

Nevertheless, good standards for BP modelling regarding variability are still missing [44]. Despite the efforts that the BP community is making in this context ([30], [7]), the resulting works are simply guides to advise modellers what are the best practices when modelling BPs. On the one hand, they do not consider BP variability itself and, on the other hand, they do not provide a methodological framework that establishes the steps to follow when modelling BPs. For such reason, we define a new and clear methodology that guides modellers to put into practice the presented approach and to obtain correct BP models that present variability.

Our methodology is mainly formed by three stages. The first one consists of identifying and modelling these parts of the process that are shared by all the process variants. Then, during the second stage, the particularities/individualities of each one of these process variants are

defined. Finally, at the third stage, it is necessary to specify the process configurations that define these process variants.

Before presenting the methodology itself, in the following subsection we advocate for the use of the BPMN notation to represent graphically the development process.

5.1.1 Using BPMN to specify the methodology

There are different proposals that can be used to specify software development processes. Some of these proposals are the PIE methodology [13], the OPEN Process Framework [18] or SPEM [19]. However, since in this work we use the BPMN notation to describe BPs, we find more appropriate to use this notation to specify the development process. In fact, in [40], authors proved that BPMN can be used to represent graphically methods defined according the SPEM specification. This proof is based on a comparison made between the elements defined by SPEM and the ones defined by the BPMN notation. This comparison shows that it is possible to find BPMN elements to represent each of the elements defined by SPEM. In addition, the expressiveness provided by the BPMN notation is enough to describe the whole process, including “Flow Objects”, “Activities” and “Gateways”.

However, since the BPMN notation defines different types of sub-models to create BPs. The different types of sub-models provided by the BPMN notation are “private (internal) business processes”, “abstract processes” and “collaboration processes”. From these types, we have used the “private (internal) business process” type of sub-model. In this case, the internal business processes refers to the development process itself.

5.2 Outlining the methodology

To reduce the existing complexity when modelling BP variability, in this section we outline the development process that represents the way to proceed when modelling BPs using the approach designed and presented

in this work.

Independently of the software development model used in the definition of a software development process (sequential or cyclical), a complete process involves four different steps which cover analysis, design, implementation and testing phase. However, in this chapter, the development process is focused just on two of these steps which relate to the Analysis and Design of a BP that presents variability. This simplification is due to the fact that the approach presented in this work has only impact in these two steps. Figure 5.1 presents this development process which consists in three main stages.

As it is shown in the figure, the process consist in three different stages. During the first stage, the commonalities of the BP variants are identify. This identification is specified by the construction of the *Base model*. At the second stage, modellers identify and specify process individualities. This identification implies the construction of the *CFT-VM* and the *TRE-VM*. Finally, at the third stage, the BP model is configured to determine when and how the process variants will be conformed. This configuration implies the construction of the *Resolution model*.

Following the stages of this development process, modellers are able to put into practice the presented approach properly. Thus, it is ensured the correctness of the resulting BP models. In addition, this process is completely independent if modellers start the modelling process from the scratch or from existing models that already include BP variations. This is due to the fact that modellers always need to identify process commonalities and individualities directly from the domain being modelled.

5.3 Process commonalities specification stage

This section presents the first stage in the development process. This stage involves identifying process commonalities and specifying them in the *Base model*.

More concretely, during this first stage (depicted in blue in figure

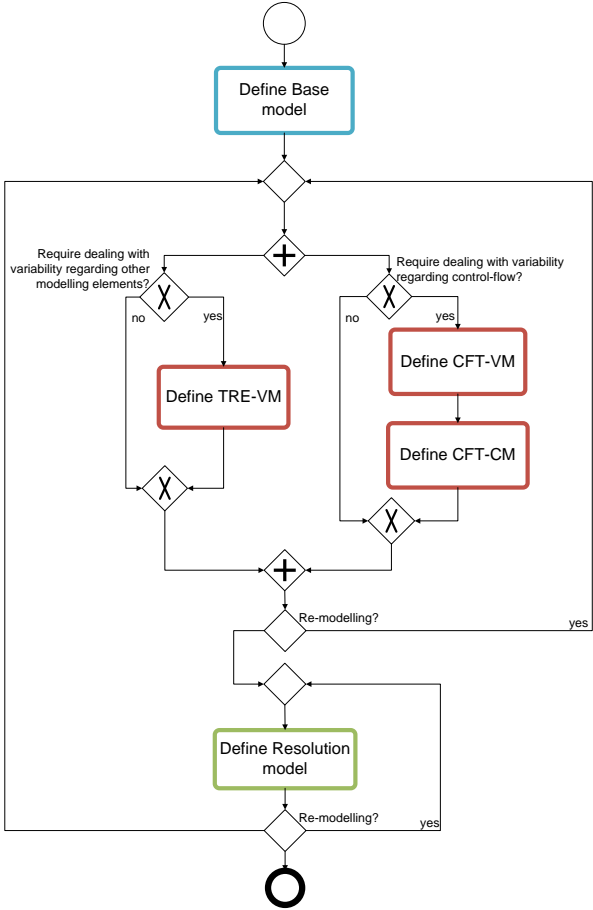


Figure 5.1: Proposed stages to apply the approach

5.1), modellers should recognize and identify all these parts¹ of the whole process that are shared by all the process variants (this is, process commonalities). Furthermore, modellers should identify at this stage which parts of the whole process are subject to variations. These parts

¹isolated activities or complete process fragments

are the variation points of the process and become the *placements* in terms of CVL.

Once the process commonalities and the *placements* are being identified, modellers should gather all this information building the *Base model* of the proposed approach. Process commonalities are represented normally, using the current notation (BPMN in our case) and the *placements* should be represented as black boxes with their respective stereotypes («design-time» and «run-time»), depending on the time when the variation point is solved.

It is also very recommendable at this early stage to identify which context variables will determine the resolution of the *placements*. This will allow modellers to visualize from the very beginning which conditions make the process vary. Nevertheless, this decision can be postponed until modellers decide.

5.4 Process individualities specification stage

This section presents the second stage in the development process. This stage (depicted in red in figure 5.1) involves identifying process individualities and specifying them in the *Variation model*.

Once the *Base model* has been specified, modellers can proceed with the identification of the different alternatives that make up a specific model variant. These alternatives may refer to (1) control flow structures and tasks, and (2) other task associated elements (such as roles or objects).

If the process alternatives refer to control flow structures and tasks, then the modeller should gather them by building the *CFT-VM*. In it, for each *placement* identified in the model, the modeller should define all the different alternatives that fit in it. In turn, if the alternatives refer to other task associated elements (such as roles, objects or events), then modellers should build the *TRE-VM* for each task that present such alternatives.

At this stage, if there are semantic constraint between the different alternatives, modellers should also specify and define them. In case

these constraints refer to control flow structures and tasks, modellers should define them apart in the *CFT-CM*. In turn, if the semantics constraints refer to other task associated elements, they should be defined by explicit arrows in the same *TRE-VM* feature diagram.

All these models only are defined if the BP being modelled presents variability in their modelling elements. That means that, for instance, if the BP which is being modelled does not present variability in roles or objects, the definition of the *TRE-VM* should be omitted.

5.5 Process configuration stage

This section presents the third and last stage in the development process. This stage involves the configuration of the BP to determine when and how the process variants will be conformed. This configuration implies the construction of the *Resolution model*.

At this point modellers have already defined model variants commonalities and individualities. However, they still have to specify: (1) which alternatives (from those specified in the *CFT-VM*) make up each variant and (2) what configurations of the *TRE-VM* are valid. This is specified by means of the definition of the *Resolution model* (depicted in green in figure 5.1). It contains the specific *replacements* for each *design placement* of the model, the selected features that configure the *TRE-VM* for the variant as well as the value of the context variables that make these *replacements* be instantiated.

This model contains only the resolutions that are applied at design time since their associated conditions depend on the initial context of the process, when the configuration is done. Those resolutions that are applied at run time (the associated conditions depend on the current context of a process instance) need to be handled during process execution, and a strategy to cope with them has to be defined. However, this is out of the scope of this work since it is focused only on BP variability at analysis/design phase.

5.6 Conclusions

In this chapter we have presented the development process that explains in detail how to proceed when modelling BP variability with the presented approach. Thanks to this process, modellers can be helped to put into practice the approach properly reducing mistakes.

The development process consists mainly in three stages according to the different models that need to be built. First of all, it is necessary to identify the commonalities shared by all the process variants as well as those points of the process subject to variations. This information is gathered into the *Base model*. Next, the individualities of the process variants need to be identified (construction of the *Variation model*) and, finally, the valid variant configurations should be defined into the *Resolution model*.

This development process ensures that, if modellers follow it properly, the resulting models are correct regarding the presented approach. The corresponding models are developed under control since modellers are guided and helped to build them.

Next chapter shows, through a running example, how to put into practice the approach following the presented methodology.

6. A running example

This chapter presents a running example that has been developed to illustrate the applicability of the modelling approach presented in this work.

The remainder of the chapter is structured as follows: Section 6.1 provides an overview of the running example. Then, section 6.3 describes in detail how it has been developed using the different models presented in the previous chapter. Finally, section 6.4 outlines some conclusions.

6.1 Overview

To illustrate the applicability of the proposed approach we make use of a running example. This example is a simplified version of the university access process followed in Spain (a detailed description of this example can be found in [5]).

The process is made up mainly of five different phases which refer to exam (1) definition (national and local), (2) preparation, (3) execution, and (4) evaluation. The process starts by the national government who defines the exams for the common subjects of the whole country (Mathematics, History, Philosophy and Foreign Language). Once these

exams are defined, they are sent to the different regional governments. At the same time, each regional government prepares its own exam for its local language (Galician, Basque or Catalan). Since Catalan is spoken in three different regions of the country (Catalonia, Balearic Islands and Valencian Community), the Catalan exam is made up of (1) a common writing for all Catalan regions and (2) a specific textual analysis based on the local writers of each region. Once the whole exam is prepared and sent, the preparation phase starts. This phase consists of preparing all the physical material that has to be handed out to the students during the exam execution. Then, the day of the exam, different exam formats will be provided to students depending on their needs (paper-format exam instructions are provided to deaf students while a Braille edition of the exam is provided to blind students). Finally, when the exam is concluded, the evaluation phase starts. This phase depends on the region the exam is performed. Common subjects of the exam can only be evaluated by national markers while region-specific exams can only be evaluated by local markers. Optionally, the exam evaluation can be supervised by an extra marker to ensure that the evaluation process is performed correctly. The result of the evaluation phase is an exam certificate which is delivered to the student. Figure 6.1 shows the university access process model together with all the different process alternatives.

As it is shown in this figure, many variants exist for the same process depending on (1) the region where the exam is taking place and on (2) the needs of the student who will take the exam. For instance, figure 6.2 shows one process model variant regarding the Galician region, where a specific exam for its language is required, and the standard students, who will be delivered with the standard material for the exam.

6.2 Description of the running example

This section presents a detailed explanation of each one of the models that have been briefly outline in section 4. To facilitate the understanding and the potential of the proposed approach we use the example

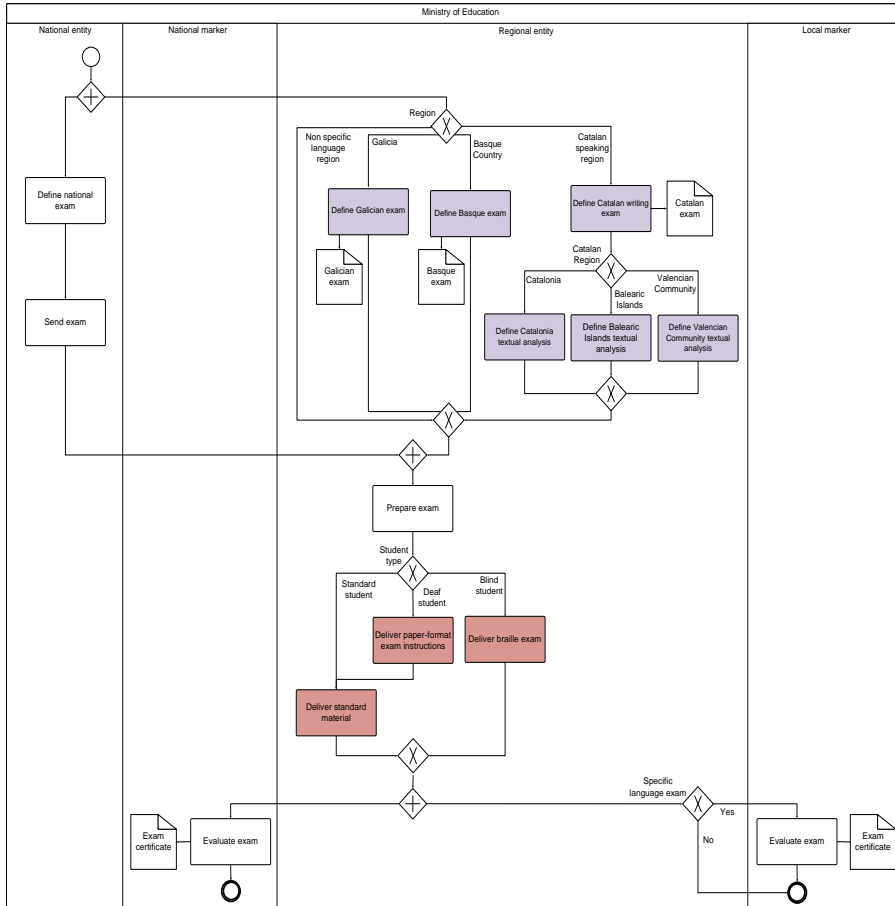


Figure 6.1: University access process model with all the model variants included

presented above.

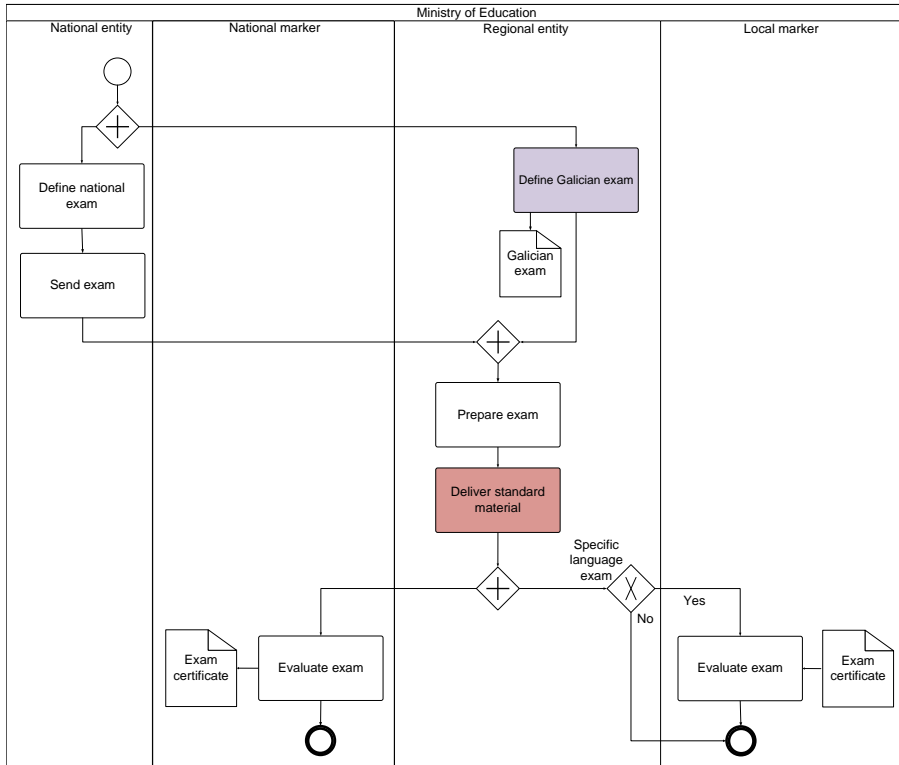


Figure 6.2: Process model variant for Galicia and Standard students

6.3 Development of the running example

6.3.1 Base model

Figure 6.3 depicts the *base model* of the running example. In it, they are defined: (1) the set of modelling elements shared by all the regions (*start* and *finish* events and *Define national exam*, *Send exam*, *Prepare exam*, and *Evaluate exam* tasks), and (2) the set of *variation points* (*Define local exam* and *Execute exam* placement) whose instantiation depends

on the region where the exam is taken (*Define local exam* placement) and on the needs of the students who will take the exam (*Execute exam* placement). The different alternatives that fit in each *placement* are defined separately in the *variation model*.

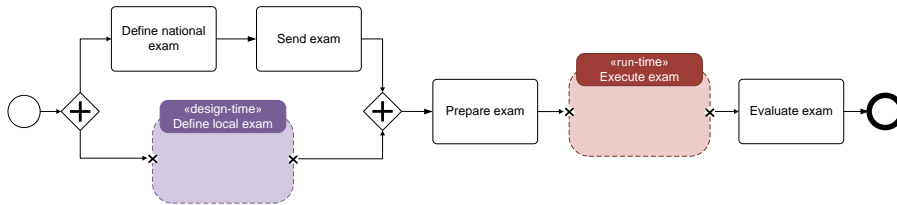


Figure 6.3: Base model for the university access process

6.3.2 Variation model

Control flow & tasks variation model

Figure 6.4 shows the graphical representation of the different *replacements* for the *Define local exam* placement depending on the region of the country where the process will take place. In turn, as the *Catalan replacement* depicts, within a *replacement* it is also possible to find variability (according to the three regions where Catalan is spoken) which is represented by a new *placement* (*Define textual analysis*). By nesting *placements* we are promoting the reusability of process fragments along the model.

Task related elements variation model

Figure 6.5 exemplifies the use of this feature model in our approach. In particular, it shows the variability that arises during the *Evaluate exam* task with roles and documents.

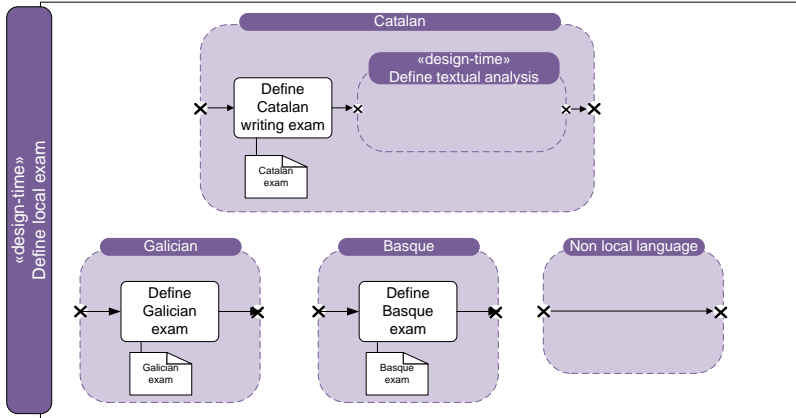


Figure 6.4: The control flow & tasks variation model for the university access process

Constraint model

In this running example the *Constraint model* is not applicable. Since it is a small running example, there is only one *design-time placement*. Thus, its *replacements* are exclusive with each other by definition. Therefore, there are not any semantic or structural dependencies that need to be represented in the *Constraint model*.

Nevertheless, if it was the case, the *Constraint model* should contain the different *Required* and *Mutual Exclusion* constraints that the *replacements* of different *placements* may present.

6.3.3 Resolution model

Figure 6.6 shows how the *resolution model* is defined for the specific process variant of the Galician region. First, the *applicable fragments* block is defined. There modellers specify the *replacement* that is chosen for the *Define local exam placement* of the *base model*. Second, the *selected*

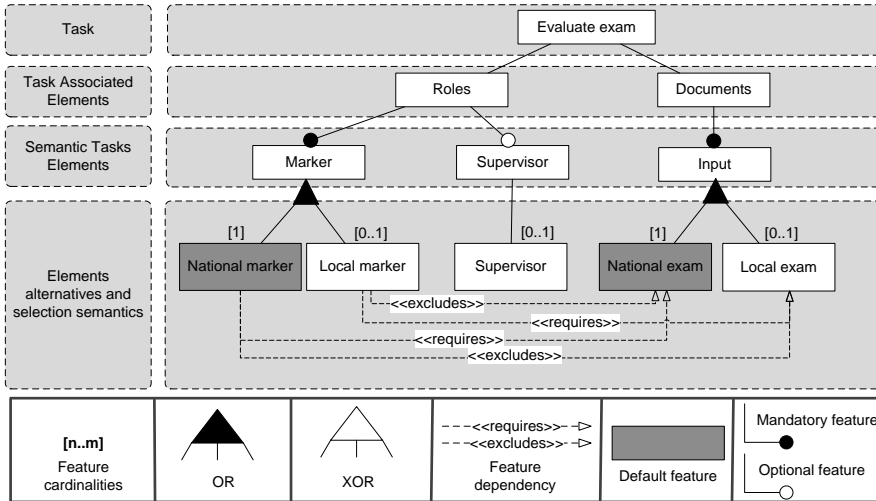


Figure 6.5: The task related elements variation model for the *Evaluate exam* task

features block specifies the selected features that configure the roles¹ and documents associated to the *Evaluate exam* task for the Galician region. Finally, the *conditions* block includes all the context variables and their corresponding values that configure the Galician process variant. In particular, to obtain the process variant for the Galician region, the context variable *region* has to be valued to “Galicia”.

In formal terms the resolution shown in figure 6.6 would be:

$$R_{Galicia} = (\{(Define\ local\ exam, Galician)\}, \\ \{(Evaluate\ exam, \{\{Marker.National\ marker, Marker.Local\ marker, \\ Supervisor.Supervisor\}, \{Input.National\ exam, Input.Local\ exam\}\})\}, \\ \{(region, Galicia)\})$$

¹we assume that a Supervisor is needed

G a l i c i a	<p>Applicable fragments: Define local exam: Galician</p> <p>Selected features: Evaluate exam.Roles.Marker.National marker Evaluate exam.Roles.Marker.Local marker Evaluate exam.Roles.Supervisor.Supervisor Evaluate exam.Documents.Input.National exam Evaluate exam.Documents.Input.Local exam</p> <p>Conditions: region = "Galicia"</p>
--	--

Figure 6.6: Galician region resolution model

6.4 Conclusions

This chapter has illustrated how the different models that compose the approach have been built. These models have been built following the methodology presented in the previous chapter and they represent the existing variability in the university access process example.

Even though the used example is a small running example, it is representative enough of the enterprise problems and it illustrates good enough the problematic being faced in this work. In addition, it has allowed us to present how all the concepts identified in chapter 2 are supported by the presented proposal.

7. Conclusions

In this work we have presented an approach to deal with BP variability at the analysis/design phase. To achieve this goal, we have identified the set of variability concepts that need to be addressed when modelling BP variability. Then, based on the ideas of the Software Product Line community, we have adapted the BVR approach to provide a modelling solution to deal with BP variability. In addition, we have defined a methodology to put into practice the developed approach.

This last chapter presents the conclusions of the work developed. First of all we revise the main contributions made to the BP community. Then, we present the ongoing and further work related to the work presented in this document. Finally, we present the publications that have been produced along the development of this thesis.

7.1 Contributions

The main contributions of this work are:

- We have identified and defined the set of concepts that need to be addressed by any approach that faces BP variability at the modelling level. These concepts are presented as an evaluation

framework in order to perform an objective evaluation of the existing approaches and the one presented in this work.

- We have developed a modelling approach to deal with BP variability at the modelling level. The cornerstone of the approach is the separation of BP model commonalities from individualities. This separation, introduced by the SPL community, allows modellers not only avoiding duplication and divergence of model commonalities but also making explicit location, rationale and dependences between variations.
- We have defined a methodology to put into practice the approach in order to help and guide modellers with the presented approach. Thus, following this development process, the approach can be applied to real scenarios.

7.2 Validation of the proposal

The proposal presented in this thesis has been put into practice in order to validate it. Specifically, a running example (i.e. university access process) has been developed.

Currently, we are studying the implementation of a prototype that supports the presented approach. This prototype is defined and implemented within the context of the MOSKitt project [31]. MOSKitt (Modeling Software KIT) is a free Eclipse-based CASE tool which is being developed jointly by the Local Regional Ministry CIT and the PROS research center. Its goal is to support the *gvMétrica* methodology, the adaptation of *Métrica III* for the specific needs of the CIT. Within this context, there are processes that present variability not only regarding control flow, but also regarding other modelling elements such as roles or objects. For instance, the CIT has processes where different roles can be in charge of the same activity or activities that produce different documents, depending on the section of the Local Ministry where the process is taking place. Thereby, this context constitutes an adequate environment to test and validate the presented approach. In

addition, within the CIT there is a big community of people involved in the MOSKitt project, ranging from analysts (software and business analysts) to end users, which are in charge of validating each new release of the tool. Their experience will provide us valuable feedback that will contribute to the improvement of the approach presented in this work.

It is also remarkable that we are in contact with other entities (other universities and some hospitals) which can provide us different and real BPs that present variability. These real cases can be used to develop them using the presented approach. They constitute another adequate environment to validate the proposal since these BPs are currently being used in real projects within different domains (e.g. healthcare).

7.3 Future work

The presented work is been developed as part of a bigger project aimed at supporting BP variability during the entire BPM lifecycle. There are several interesting directions that can be taken to provide the proposal with a wider spectrum of application. The following list summarizes the research activities that are planned to continue this work.

- **Prototype implementation:** At this moment, we are implementing a proof of concept prototype of the approach presented in this document. This prototype is been developed in the context of Eclipse [16], specifically, it is based on the MOSKitt tool [31]. In particular, we are implementing the graphical editors to allow modellers to define the models proposed in this work.
- **Assisting the user during BP definition:** It is very important at the modelling level a mechanism that allows assisting the user during the BP specification phase. This mechanism can be based on a recommender system that can guide the modeller during the definition of the proposed models.
- **Validator of the definition:** As well as the assistance, another important point is to implement a validator that helps modellers

during the definition stage. Thus, it will be able automatically of validating the semantics of the BP defined (e.g. correctness, constraints, etc.).

- **Business process enactment phase:** A very important step is to define model transformations that allow us bringing BP models defined in our approach into a process engine deployable representation. This implies also defining a strategy to cope with the variability that needs to be solved at run time. For such purpose, we will have to take into account other approaches (such as [1]) that already consider variability at run time. In particular, we are considering to explore the possibility of adopting the experience that we already have ([9], [10]) in the reconfiguration of systems at run time based on the use of models.
- **Construction of a repository of process fragments:** In order to provide reusability between models, it is also necessary a repository that contains the process fragments definitions. Thus, they are able to use them on-demand.

7.4 Publications

During the development of the work presented in this thesis, the following publication have been produced:

1. **Clara Ayora**, Germán H. Alférez, Victoria Torres, Vicente Pelechano. *Procesos de Negocio Auto-Adaptables al contexto*. VII Jornadas de Ciencia e Ingeniería de los Servicios (JCIS 2011) (JCIS'11). A Coruña, España, 2011.
2. **Clara Ayora**, Victoria Torres, Vicente Pelechano. *BP Variability Case Studies Development using different Modeling Approaches*. Technical Report n°3 of the PROS center (2011) [2].
3. **Clara Ayora**, Victoria Torres, Vicente Pelechano. *Dealing with Variability in Business Process Models: An Evaluation Framework*. Technical Report n°5 of the PROS center (2011) [3].

4. **Clara Ayora**, Victoria Torres, Vicente Pelechano. *The university Access process: A business process variability example*. Technical Report n°6 of the PROS center (2011) [4].

The first paper presents a proposal for handling and automatically adapt BPs according to context changes. Specifically, the described proposal allows to 1) model the process by preventing variations in the execution context and 2) adapt the process according to these variations. In addition, it is also proposed the software infrastructure that allows this adaptation at run time.

Regarding the technical reports, the first one (n°3) presents the development of three different case studies with the already existing approaches that deal with BP variability at the modelling level (C-EPC, PESOA y Provop). The second technical report (n°5) describes these approaches and presents the evaluation framework described in chapter 2. Finally, the last technical report (n°6) describes in detail the case study (university access process) used to show the applicability of the presetedn approach.

Bibliography

- [1] Adams, M. J. (2007). *Facilitating Dynamic Flexibility and Exception Handling for Workflows*. Ph.D. thesis, Faculty of Information Technology. Queensland University of Technology.
- [2] Ayora, C., Torres, V., & Pelechano, V. (2011). Technical report n°3: <http://www.pros.upv.es/index.php/es/informes-tecnicos/70-informes-tecnicos/18-technical-report-pros-tr-2011-03>.
- [3] Ayora, C., Torres, V., & Pelechano, V. (2011). Technical report n°5: <http://www.pros.upv.es/index.php/es/informes-tecnicos/70-informes-tecnicos/20-technical-report-pros-tr-2011-05>.
- [4] Ayora, C., Torres, V., & Pelechano, V. (2011). Technical report n°6: <http://www.pros.upv.es/index.php/es/informes-tecnicos/70-informes-tecnicos/21-technical-report-pros-tr-2011-06>.
- [5] Ayora, C., Torres, V., & Pelechano, V. (2011). The university access process: A business process variability example. Tech. rep., ProS-TR-2011-06, ProS-UPV,
- [6] Bayer, J., Gerard, S., Haugen, Ø., Mansell, J. X., Møller-Pedersen, B., Oldevik, J., Tessier, P., Thibault, J.-P., & Widen, T. (2006). Consolidated product line variability modeling. In *Software Product Lines*, (pp. 195–241).
- [7] Becker, J., Rosemann, M., & Uthmann, C. v. (2000). Guidelines of business process modeling. In *Business Process Management*,

- Models, Techniques, and Empirical Studies*, (pp. 30–49). London, UK: Springer-Verlag.
- [8] Cardoso, J., Mendling, J., Neumann, G., & Reijers, H. A. (2006). A discourse on complexity of process models. In *Business Process Management Workshops*, (pp. 117–128).
- [9] Cetina, C., Giner, P., Fons, J., & Pelechano, V. (2009). Autonomic computing through reuse of variability models at runtime: The case of smart homes. *Computer*, 42(10), 37–43.
- [10] Cetina, C., Haugen, O., Zhang, X., Fleurey, F., & Pelechano, V. (2009). Strategies for variability transformation at run-time. In *Proceedings of the 13th International Software Product Line Conference, SPLC '09*, (pp. 61–70). Pittsburgh, PA, USA: Carnegie Mellon University.
- [11] Chinosi, M., & Trombetta, A. (2009). A design methodology for bpmn.
- [12] Clements, P. (2001). *Software product lines: practices and patterns*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
- [13] Cunin, P.-Y., Greenwood, R., Francou, L., Robertson, I., & Warboys, B. (2001). The pie methodology — concept and application. In V. Ambriola (Ed.) *Software Process Technology*, vol. 2077 of *Lecture Notes in Computer Science*, (pp. 3–26). Springer Berlin / Heidelberg.
- [14] CVL-RFP (2009). <http://www.omg.org/cgi-bin/doc?ad/2009-12-3>.
- [15] Czarnecki, K. (1998). *Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. Ph.D. thesis, Technical University of Ilmenau.
- [16] Eclipse (2011). <http://www.eclipse.org>.

-
- [17] Elliott, G. (2004). *Global business information technology: an integrated systems approach*. Pearson/Addison Wesley.
- [18] Firesmith, D., & Sellers, H. B. (2001). *The OPEN Process Framework*. Harlow: Addison-Wesley Longman.
- [19] Group, O. M. (2005). Software process engineering metamodel specification (spem), v1.1. Tech. rep., OMG,
- [20] Hallerbach, A., Bauer, T., & Reichert, M. (2008). Context-based configuration of process variants. In *3rd International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008)*, (pp. 31–40).
- [21] Hallerbach, A., Bauer, T., & Reichert, M. (2008). Managing process variants in the process lifecycle. In *10th Int'l Conf. on Enterprise Information Systems (ICEIS'08)*, (pp. 154–161).
- [22] Hallerbach, A., Bauer, T., & Reichert, M. (2008). Managing process variants in the process lifecycle. In *10th Int'l Conf. on Enterprise Information Systems (ICEIS'08)*, (pp. 154–161).
- [23] Hallerbach, A., Bauer, T., & Reichert, M. (2010). Capturing variability in business process models: The provop approach. *Software Process: Improvement and Practice*.
- [24] Hallerbach, A., Bauer, T., & Reichert, M. (2010). *Configuration and Management of Process Variants*, chap. International Handbook on Business Process Management. Springer.
- [25] Haugen, O., Moller-Pedersen, B., Oldevik, J., Olsen, G., & Svendsen, A. (2008). Adding standardized variability to domain specific languages. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, (pp. 139–148).
- [26] Jablonski, S., & Bussler, C. (1996). *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thomson Computer Press.

- [27] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Tech. rep., Carnegie-Mellon University Software Engineering Institute,
- [28] Mendling, J. (2008). *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction and Guidelines for Correctness*, vol. 6 of *Lecture Notes in Business Information Processing*. Springer.
- [29] Mendling, J. (2008). <http://www.mendling.com/EPML/C-EPC-Validator.xsl>.
- [30] Mendling, J., Reijers, H. A., & van der Aalst, W. M. P. (2010). Seven process modeling guidelines (7pmg). *Information & Software Technology*, 52(2), 127–136.
- [31] MOSKitt (2011). <http://www.moskitt.org/>.
- [32] Myllymäki, T. (2001). Variability management in software product lines.
- [33] Pohl, K., Böckle, G., & van der Linden, F. J. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 1 ed.
- [34] Puhlmann, F., Schnieders, A., Weiland, J., & Weske, M. (2006). Variability mechanisms for process models. Tech. rep., BMBF-Project,
- [35] Riebisch, M., Böllert, K., Streitferdt, D., & Philippow, I. (2002). Extending feature diagrams with UML multiplicities. In *6th World Conference on Integrated Design & Process Technology (IDPT2002)*. Pasadena, California.
- [36] Rosa, M. L. (2009). *Managing Variability in Process-Aware Information Systems*. Ph.D. thesis, Queensland University of Technology.

- [37] Rosa, M. L., Dumas, M., ter Hofstede, A. H. M., Mendling, J., & Gottschalk, F. (2008). Beyond control-flow: Extending business process configuration to roles and objects. In *ER*, (pp. 199–215).
- [38] Rosemann, M., & van der Aalst, W. M. P. (2007). A configurable reference modelling language. *Inf. Syst.*, *32*(1), 1–23.
- [39] Smith, P. R., & Sarfaty, R. (1993). Creating a strategic plan for configuration management using computer aided software engineering (case) tools. Paper For 1993 National DOE/Contractors and Facilities CAD/CAE User's Group.
- [40] Sousa, K., Mendonça, H., & V, J. (2007). Towards method engineering of model-driven user interface development.
- [41] Stan Bühne, K. P., Kim Lauenroth (2005). Modelling requi variability across product lines.
- [42] Takeda, H., Hamada, S., Tomiyama, T., & Yoshikawa, H. (1990). A cognitive approach of the analysis of design processes. In The Second ASME Design Theory and Methodology Conference, pp. 153–160. The American Society of Mechanical Engineers (ASME).
- [43] van der Aalst, W. M. P., Dreiling, A., Gottschalk, F., Rosemann, M., & Jansen-Vullers, M. H. (2005). Configurable process models as a basis for reference modeling. In *Business Process Management Workshops*, (pp. 512–518).
- [44] van der Aalst, W. M. P., ter Hofstede, A. H. M., & Weske, M. (2003). Business process management: A survey. In *Business Process Management*, (pp. 1–12).
- [45] Vervuurt, M. (2007). Modeling business process variability : a search for innovative solutions to business process variability modeling problems. Student Theses of University of Twente. October 2007.
- [46] Weber, B., Sadiq, S. W., & Reichert, M. (2009). Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, *23*(2), 47–65.

- [47] Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures*. 978-3-540-73521-2. Springer-Verlag Berlin Heidelberg 2007.
- [48] White, S. A., & Miers, D. (2008). *BPMN Modeling and Reference Guide*. Feature Strategies Inc. Lighthouse Point, FL. USA.

www.pros.upv.es

Centro de Investigación en Métodos
de Producción de Software
Universitat Politècnica de València
Camí de Vera s/n
Edificio 1F
46022 València
Spain

Tel: (+34) 963 877 007 (Ext. 83530)

Fax: (+34) 963 877 359



Centro de Investigación en Métodos
de Producción de Software



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA