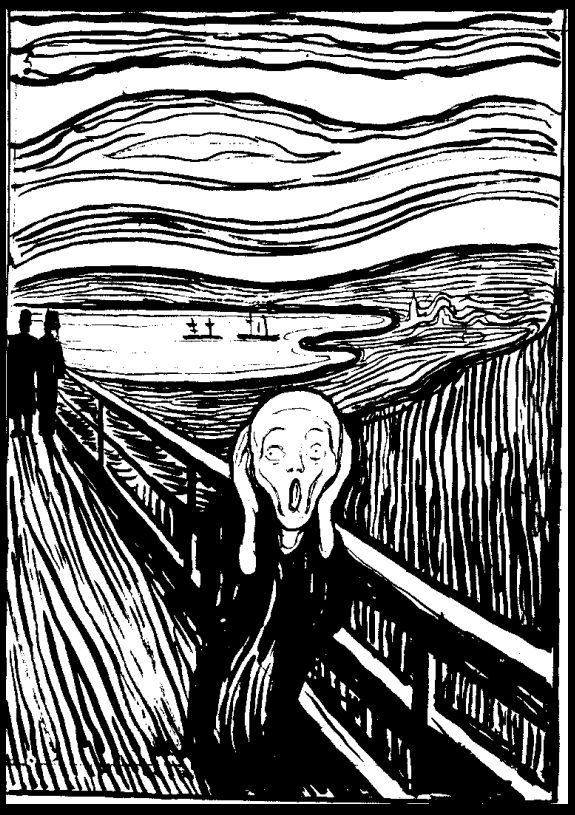
Enhancing Variability Modeling in Process-Aware Information Systems through Change Patterns

Clara Ayora Esteras



Advisors: Dr. Victoria Torres Bosch
Dr. Vicente Pelechano Ferragud

November 2015

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Clara Ayora Esteras

Enhancing Variability Modeling in Process-Aware Information Systems through Change Patterns

PhD Thesis. November, 2015

PROS
Centro de Investigación en Métodos de Producción de Software

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Enhancing Variability Modeling in Process-Aware Information Systems through Change Patterns:

This report was prepared by
Clara Ayora Esteras

Advisors
Dr. Victoria Torres Bosch
Dr. Vicente Pelechano Ferragud

Members of the Thesis Committee
Dr. Óscar Pastor López
Dr. Félix Óscar García Rubio
Dr. Antonio Ruiz Cortés

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camí de Vera s/n, Edif. 1F
46022 - València, Spain

Tel: (+34) 963 877 007 (Ext. 83530)
Fax: (+34) 963 877 359
Web: http://www.pros.upv.es

Release date: November, 2015
Comments: A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at the Universitat Politècnica de València.
Title page paintings of Edvard Munch. The Munch Museum. Oslo.

To Jose Luis

❝*Lo mejor está por venir*❞

Acknowledgements

This PhD thesis has been a difficult odyssey. Like Ulysses, it has been a hard journey full of contretempts before reaching the end. Through these lines I would like to express my gratitude to all these people who have contributed to finishing this journey.

First of all, I would like to thank my advisors Dr. Victoria Torres and Dr. Vicente Pelechano, for their advices and feedback during the development of this thesis. You both gave me the opportunity to develop this research and I have learnt many things during this process.

I am especially grateful to Ass. Prof. Dr. Barbara Weber and Prof. Dr. Manfred Reichert for their help, their interest in my work, their valuable contributions to this research, their good advices, and the opportunities they gave me through these years.

Another grattitude is to Visit. Prof. Dr. Jose Luis de la Vara for taking time out from his busy schedule and collaborating with us in the last part of the thesis.

I would also like to say thank you to María Ortega for her help in the implementation of the editor. Thanks to her efficient work, I could save time and effort when validating the proposed work. I extend this grattitude to Dr. Stefan Zugal and Dr. Jakob Pinggera for their much appreciated feedback and support when implementing the editor.

I would like to specially thank my friends Mario and María (Zape) for being there. Your understanding and daily support has been funda-

mental for succeeding in this thesis.

I would also like to thank Marcela and Mariajo for showing me a different perspective and sharing with me many laughs, bathroom confidences, and music experiences.

A sincere thanks to Salva, Pablo, and Isma for the good moments we shared both inside and outside the lab. I also like to express my gratitude to Prof. Óscar Pastor for his direction of our research center. Nor do I want to forget Ainoha, Sergio E., Paco, Vero, Ana M., Diego, Ignacio, los Perris, Paqui, Patricia, and Jesús. Being with you always means having a great time. I would also like to thank Ana Cidad for her help with the paperwork, Nacho for his technical support, Harvey for his collaboration, and the rest of current and former colleagues from the PROS research center for their assistance and collaboration. In addition, I would like to mention all the colleagues from the *Institute of Databases and Information Systems* of the University of Ulm (especially Carolina) and from the *Quality Engineering Research Group* of the University of Innsbruck for all their kindness and hospitality.

I would also like to include here my miserable colleagues from the DSIC department (Sonia, Sergio E., Víctor, Pablo, Alejandro, Lucía, Sergio L., Sergio P., Knut, and Raúl), with whom I have shared less working hours but many conversations and funny moments. Thanks for your friendship and support.

A special thanks is for 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 would like to thank as well to all those people (i.e., friends, musicians, and Cullera's people) that are part of my life, and, unwittingly, make it a little bit easier.

Finally, thanks to Jose Luis for believing in me and sharing his life with me every day. This odyssey would not have finished without you. I love you endlessly.

Clara Ayora

Abstract

The increasing adoption of process-aware information systems (PAIS) together with the high variability in business processes has resulted in collections of process families. These families correspond to a business process model and its variants, which can comprise hundreds or thousands of different ways of realizing this process. Managing process variability in this context can be very challenging, labor-intensive, and error-prone.

Motivated by this challenge, several approaches enabling process variability have been developed. However, with these approaches PAIS engineers usually are required to model and manage all the elements of a process family one by one and ensure its correctness by their own. This can be tedious and error-prone especially when a process family comprises hundreds or thousands of process variants. For example, PAIS engineers need to be aware of each variation and dependence of each process variant. Thus, there is a need of methods that allow PAIS engineers to model process variability more explicitly, especially at a level of abstraction higher than the one provided by the existing process variability approaches. However, how process variability is represented is critical for defining these methods (e.g., what language constructs are used to model process variability). In this context, using modeling patterns (reusable solutions to a commonly occurring problem) is a promising way to address these issues. For example, patterns have been proved as an efficient solution to model individual business processes.

The objective of this thesis is to enhance the modeling of variability in process families through change patterns. For such purpose, first, we conduct a systematic study to analyze existing process variability approaches regarding their expressiveness with respect to process variability modeling as well as their process support. Thus, we can identify the core set of variability-specific language constructs. In addition, based on the obtained empirical evidence, we derive the VIVACE framework, a complete characterization of process variability which comprises also a core set of features fostering process variability. VIVACE enables PAIS engineers to evaluate existing process variability approaches as well as to select that variability approach meeting their requirements best. In addition, it helps process engineers in dealing with PAISs supporting process variability.

Second, based on the identified language constructs, we present a set of 10 change patterns for process families and show how they can be implemented in a process variability approach. In particular, these patterns support process family modeling and evolution and ensure process family correctness by automatically introducing and deleting modeling elements. In order to prove their effectiveness and analyze their suitability, we applied these change patterns in a real scenario. More concretely, we conduct a case study with a safety standard with a high degree of variability. The case study results show that the application of the change patterns can reduce the effort for process family modeling by 34% and for evolution by 40%. In addition, we have analyzed how PAIS engineers apply the patterns and their perceptions of this application. Most of them expressed some benefit when applying the change patterns, did not perceived an increase of mental effort for applying the patterns, and agreed upon the usefulness and ease of use of the patterns.

Resumen

La creciente adopción de sistemas de información dirigidos por procesos de negocio (PAIS, según sus siglas en inglés) y la alta variabilidad en dichos procesos, han dado lugar a la aparición de colecciones de familias de procesos. Estas familias están constituidas por un modelo de proceso de negocio y sus variantes, las cuales pueden comprender entre cientos y miles de diferentes formas de llevar a cabo ese proceso. Modelar y gestionar la variabilidad de los procesos en este contexto puede resultar muy difícil , laborioso, y propenso a errores.

Por este desafío se han desarrollado distintas soluciones que permiten la gestión de esta variabilidad en los procesos de negocio. Sin embargo, los ingenieros que trabajan con PAIS al utilizar estas soluciones deben crear y gestionar uno por uno todos los elementos de las familias de procesos y asegurar ellos mismos su corrección. Esto puede ser tedioso y propenso a errores, especialmente cuando las familias están compuestas de múltiples variantes. Por ejemplo, los ingenieros deben ser conscientes de todas las variaciones y dependencias de todas las variantes. Por ello, son necesarios nuevos métodos que permitan a los ingenieros de PAIS modelar la variabilidad de los procesos de una manera más explícita, sobre todo a un nivel de abstracción más alto del proporcionado hasta ahora por dichas soluciones. Sin embargo, para definir estos nuevos métodos resulta clave cómo se representa la variabilidad (ej.: qué primitivas se utilizan para modelar la variabilidad en los procesos). En este contexto, el uso de patrones de modelado (soluciones

reutilizables a un problema recurrente) resultan una solución prometedora. Por ejemplo, los patrones resultan eficaces para modelar y gestionar procesos de negocio individuales.

El objetivo de esta tesis es mejorar el modelado de la variabilidad en las familias de procesos a través del uso de patrones de cambio. En primer lugar, hemos llevado a cabo un estudio sistemático con el fin de analizar las soluciones existentes que gestionan la variabilidad en los procesos, así como el soporte que estas proporcionan. De esta forma, hemos identificado y analizado cuál es el conjunto básico de primitivas específicas para representar la variabilidad. Además, basándonos en la evidencia empírica obtenida, hemos derivado el marco de evaluación VIVACE, el cual recoge las primitivas de variabilidad y un conjunto básico de características que favorecen la variabilidad en los procesos. Así, VIVACE conforma una completa caracterización de la variabilidad en los procesos de negocio. Asimismo, VIVACE permite a los ingenieros de PAIS evaluar las soluciones que permiten gestionar la variabilidad en los procesos, así como seleccionar la solución que se ajuste mejor a sus necesidades. Finalmente, VIVACE también puede ayudar a los ingenieros a gestionar PAISs que den soporte a esta variabilidad.

En segundo lugar, basándonos en las primitivas identificadas, hemos definido 10 patrones de cambio para familias de procesos y cómo pueden ser implementados. Estos patrones ayudan al modelado y a la evolución de familias de procesos y además son capaces de garantizar la corrección de la propia familia permitiendo la inserción y el borrado automático de elementos. Para probar su efectividad y analizar su idoneidad, hemos aplicado estos patrones de cambio en un escenario real. En concreto, hemos realizado un caso de estudio con un estándar de seguridad con un alto nivel de variabilidad. Los resultados de este caso demuestran que la aplicación de nuestros patrones de cambio puede reducir el esfuerzo para el modelado de familias de procesos en un 34% y en un 40% para su evolución. Además, hemos analizado cómo los ingenieros de PAIS aplican los patrones y sus percepciones de esta aplicación, obteniendo como resultado que la mayoría de ellos encontró beneficios al aplicarlos. Además, no percibieron un aumento en el esfuerzo mental necesario y estuvieron de acuerdo en su utilidad y facilidad de uso.

Resum

La creixent adopció de sistemes d'informació dirigits per processos de negoci (PAIS, segons les seues sigles en anglès) i l'alta variabilitat en eixos processos, han donat lloc a la aparició de col·leccions de famílies de processos. Estes famílies es formen d'un model de procés de negoci i les seues variants, les quals poden comprendre entre cents i milers de diferents formes de dur a terme eixe procés. Modelar la variabilitat dels processos en este context pot resultar molt difícil , laboriós, i propens a errors.

Per aquest desafiament s'han desenvolupat diverses solucions que permeten la gestió d'aquesta variabilitat en els processos de negoci. No obstant, els enginyers que treballen amb PAIS quen utilitzen aquestes solucions han de crear i gestionar un a un tots els elements de les famílies de processos i assegurar ells mateixos la seua correcció. Això pot ser tediós i propens a errors especialment quan les famílies es componen de múltiples variants. Per exemple, els enginyers han de ser conscients de totes i cadascuna una de les variacions i dependències de totes les variants. Per quest motiu, son necessaris nous mètodes que permeten als enginyers de PAIS modelar la variabilitat dels processos de manera més explícita, sobretot a un nivell d'abstracció més alt del fins ara proporcionat per les solucions actuals. No obstant, per a definir aquestos mètodes resulta clau com es representa la variabilitat (ex.: quines primitives s'utilitzen per a modelar la variabilitat en els processos). En aquest context, l'ús de patrons de modelatge (solucions reutilitzables

a un problema recurrent) resulten una solució prometedora. Per exemple, els patrons han sigut provats eficaçment per modelar i gestionar processos de negoci individuals.

L'objectiu d'aquesta tesi és millorar el modelatge de la variabilitat en les famílies de processos a través de l'ús de patrons de canvi. En primer lloc, hem dut a terme un estudi sistemàtic per a analitzar les solucions existents per a gestionar la variabilitat en els processos, així com el suport que aquestes proporcionen. D'aquesta manera, hem identificats i analitzat quin és el conjunt bàsic de primitives específiques per a representar la variabilitat. A més, basant-nos en l'evidència empírica obtinguda, hem derivat el marc d'evaluació VIVACE, el qual arreplega les primitives de variabilitat i un conjunt bàsic de característiques que afavoreixen la variabilitat en els processos. Així, VIVACE conforma una completa caracterització de la variabilitat en els processos de negoci. Així mateix, VIVACE permet als enginyers de PAIS avaluar les solucions per a gestionar la variabilitat en els processos, així com seleccionar la solució que s'ajusta millor a les seues necessitats. Finalment, VIVACE també pot ajudar als enginyers a gestionar PAISs que donen suport a aquesta variabilitat.

En segon lloc, basant-nos en les primitives identificades, hem definit 10 patrons de canvi per a famílies de processos i com poden ser implementats. Aquestos patrons ajuden al modelatge i a l'evolució de famílies de processos i garanteixen la correcció de la pròpia família permitint la inserció i eliminació automàtica d'elements. Per a provar la seua efectivitat i analitzar la seua idoneïtat, hem aplicat els patrons de canvi en un escenari real. En particular, hem realitzat un cas d'estudi amb un estàndard de seguretat amb un alt nivell de variabilitat. Els resultats de aquest cas demostren que l'aplicació dels nostres patrons de canvi poden reduir l'esforç per al modelatge de famílies de processos en un 34% i en un 40% per a la seua evolució. A més, hem analitzat com els enginyers de PAIS apliquen els patrons i les seues percepcions d'esta aplicació. Com a resultat, la majoria d'ells va trobar beneficis al aplicar-los. A més, no van percebre un augment en l'esforç mental necessari i van estar d'acord en la seua utilitat i facilitat.

Content of the thesis

List of Figures

List of Tables

1

Introduction

Information Systems (ISs) constitute software systems that deal with a large number of business requirements. These requirements can be referred either to functional features that describe the core functionalities of the system, and to non-functional features such as performance or scalability. This amount of requirements has driven organizations to describe and manage ISs in a more structured and systematic way [Sharp & McDermott, 2001], which has lead towards a new generation of ISs named *Process-Aware Information Systems* (PAISs) [Dumas et al., 2005].

Generally, a PAIS constitutes an IS that manages, analyzes, and executes operational processes which involve requirements, people, applications services, and business data [Dumas et al., 2005]. Examples of PAISs include workflow management systems (e.g., ADEPT2 [Reichert et al., 2005], YAWL [van der Aalst & ter Hofstede, 2003]), enterprise resource planning systems (e.g., SAP R/3 [SAP-Business-Suite, 1992]),

case management systems (e.g., FLOWer [Dumas et al., 2005], and PHILharmonicFlows [Künzle & Reichert, 2011]). As opposed to traditional ISs, a PAIS separates the process logic from the executed code by representing this logic in terms of a *process model* [Weske, 2007]. In particular, process models describe the business processes at a rather high level of abstraction providing the schema for the execution of the system [Weske, 2007]. In general, process models are embedded in a process lifecycle comprising analysis, design, configuration, enactment, diagnosis, and evolution [Weske, 2007]. In addition, they may serve as a basis for facilitating communication between stakeholders, process analysis, simulation, and visualization [Reichert & Weber, 2012].

However, in today's dynamic business world, the success of an organization increasingly depends on its ability to adapt to changes in its environment [Reichert & Weber, 2012]. Examples of these changes refer to new emerging regulations, market evolution, changes in customer behavior, and process improvement. This has led organizations to accumulate related process models in order to support these changes [Dijkman et al., 2012]. Related process models are typically referred as *process model variants* (*process variants* for short) [Reichert & Weber, 2012]. Process variants pursue the same or a similar *business objective* (e.g., product sale) and can have activities (and their ordering constraints) in common. Nevertheless, process variants differ in their *application context* (e.g., regulations to comply with in different countries or the type of product to deliver) [Reichert & Weber, 2012; Dijkman et al., 2012; Ayora et al., 2015]. Some activities may be relevant only for certain application contexts. All the context factors causing process variability are typically known at design time [Reichert & Weber, 2012].

A collection of related process variants is denoted as a *process family*. In practice, a process family may comprise hundreds or thousands of process variants [Reichert & Weber, 2012]. In the automotive industry, for example, we found a process family dealing with vehicle repair and maintenance in a garage, which comprises more than 900 process variants [Hallerbach et al., 2010a]. These process variants share *commonalities* (i.e., process fragments shared by all process variants), but also show country- and vehicle-specific *variations*. In turn, [Li, 2010] reports

on more than 90 process variants for handling medical examinations in a hospital. Finally, consider check-in procedures at airports [Ayora et al., 2015]. Even though this process is similar irrespective of the airport the passenger departs from and the airline flying with, numerous variations exist depending on distinguished factors such as the type of passenger (e.g., unaccompanied minors, handicapped people, or people carrying a pet), the type of check-in (e.g., online or at the counter), or the type of luggage (e.g., fragile or overweight). The complete description of the check-in process can be found in Appendix A. We will use this process as running example throughout the thesis.

All these processes illustrate the variability that a process family may have due to its heterogeneous application context. Trying to model and maintain each process variant of such process families from scratch would be too cumbersome and costly for organizations [Weber et al., 2011]. Modeling properly the variability involved in process families constitutes, therefore, a fundamental challenge for every PAIS. In this context, this thesis attempts to help in this problem.

The rest of the chapter is organized as follows. Section 1.1 explains the purpose of this thesis. Section 1.2 details the problems that this thesis addresses. Section 1.3 introduces the goals defined for this thesis. Section 1.4 introduces the research methodology that has been followed in the development of the thesis. Section 1.5 explains the context in which this thesis has been performed. Finally, Section 1.6 gives an overview of the structure of this document.

1.1 Motivation

In order to efficiently and effectively manage process families, organizations have been interested in modeling (capturing) common process knowledge only once and making it reusable in terms of a *reference process model* (*reference process* for short) [Reichert & Weber, 2012]. Along this trend, a multitude of reference processes have been developed in various domains. Examples include ITIL processes for IT service management [Hochstein et al., 2005], SAP reference pro-

cesses for organization resource management [Mendling et al., 2008], and medical guidelines for patient treatment [Lenz & Reichert, 2007]. Usually, these reference processes are described in a graphical way using a process modeling language like *Business Process Modeling Notation* (BPMN) [BPMN, 2011] or *Event-driven Process Chain* (EPC) [ARIS, 1990]. Such languages include collections of primitives (e.g., activities and gateways) to represent reference processes but they do not provide proper support for explicitly describing process variations [Reinhartz-Berger et al., 2010; de la Vara et al., 2010].

Motivated by this shortcoming, several approaches enabling process variability along the process lifecycle have been developed. That is, approaches allowing for the analysis, design, configuration, enactment, diagnosis, and evolution of process families [Puhlmann et al., 2006; Rosemann & van der Aalst, 2007; Hallerbach et al., 2010a]. In these approaches, process variants are defined in terms of a *configurable process model*, which represents a complete process family.[1] By treating variability as a first class citizen, these configurable process models contribute to avoiding model redundancies, fostering model reusability, and reducing modeling efforts [Hallerbach et al., 2010a]. For example, Figure 1.1 illustrates a configurable process model for the the check-in process (cf. Appendix A). This configurable model is represented in terms of the Configurable EPC (C-EPC) approach [Gottschalk et al., 2007].

However, modeling and evolving process families and ensuring their correctness can be very challenging due to their size and complexity. PAIS engineers need assistance for such purpose [Reichert & Weber, 2012; Ayora et al., 2015]. Once an approach is selected (e.g., C-EPC), PAIS engineers have to manually model and manage all the elements of a configurable process model one by one and ensure its correctness by their own [Hallerbach et al., 2010a]. This can be tedious and error-prone especially when a configurable process model represents a process family comprising a high number of process variants [Hallerbach et al., 2010a; Reichert & Weber, 2012]. For example, PAIS engineers need to be aware

[1] We use the terms *configurable process model* and *process family* synonymously throughout the thesis.

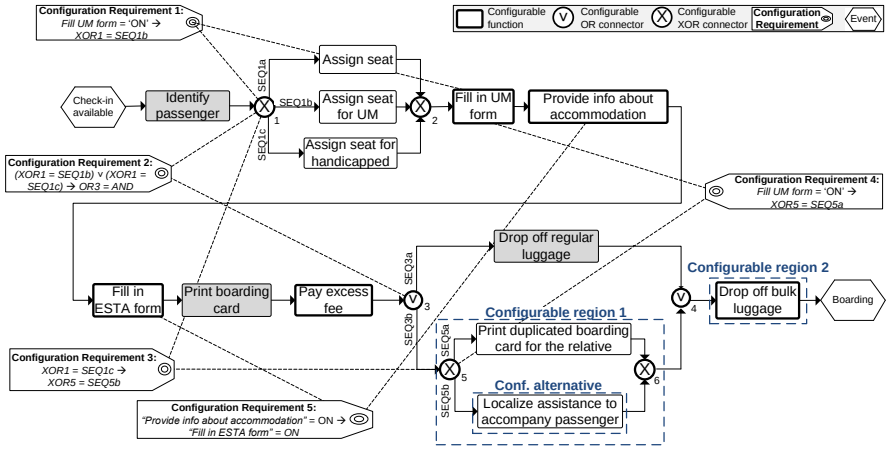


Figure 1.1: Configurable process model of the check-in process in C-EPC

of each variation and dependence of each process variant. However, there is a lack of efficient methods to deal with process variability in an explicit manner, especially at a level of abstraction higher than the one provided by the existing process variability approaches [Ayora et al., 2015].

Methods for explicitly dealing with process variability cannot be defined arbitrarily [Ayora et al., 2013]. How process variability is represented becomes critical. That means, for example, to identify what language constructs are commonly used to capture variability in a configurable process model [Ayora et al., 2013]. However, although several attempts to describe and characterize process variability modeling have been made (e.g., [Mechrez & Reinhartz-Berger, 2014; Aiello et al., 2010]), none of them identify these constructs. Further, the implementation and application of these methods are crucial factors as well [Reichert & Weber, 2012]. For example, if these methods are defined purely in a theoretical way, their realization and materialization cannot be proved. In addition, if their application is too time-consuming and difficult for modeling large configurable process models, they will not

be suitable. Thus, in order to prove their reality and effectiveness, such methods should be implemented and applied in real scenarios to analyze their suitability. In addition, the way PAIS engineers interact with these methods is also a relevant factor for determining their successful adoption. If PAIS engineers do not have a positive attitude towards these methods and are not willing to use them, their adoption will be hindered as well.

The use of modeling patterns (i.e., reusable solutions to a commonly occurring problem [Weber et al., 2008]) is a promising way to address these issues. For example, a language-independent and empirically grounded set of *adaptation patterns* has been proposed for the modeling and management of (individual) process models [Weber et al., 2008]. Adaptation patterns not only allow creating and modifying process models at a high level of abstraction, fostering model quality by ensuring *correctness-by-construction*, but also provide systematic means for realizing change operations in a process models [Döhring et al., 2011]. Further, adaptation patterns have served as basis for implementing changes in different stages of the process lifecycle; e.g., process model creation [Gschwind et al., 2008], process configuration [Hallerbach et al., 2010a], process instance change [Dadam & Reichert, 2009; Marrella et al., 2011], process model evolution [Dadam & Reichert, 2009; Küster et al., 2010; Zhao & Liu, 2013], model refactoring [Weber et al., 2011], change reuse [Aghakasiri & Mirian-Hosseinabadi, 2009], model comparison [Küster et al., 2008], and change analysis [Günther et al., 2006]. However, although adaptation patterns are well suited for creating and managing individual process models, they are not sufficient to cope with process variability in an explicit manner [Ayora et al., 2012a]. They are not accurate for dealing with the specific complexity that process variability introduces [Ayora et al., 2013].

1.2 Problem Statement

The modeling of process variability is not a closed research topic. The work presented in this thesis attempts to **enhance variability mod-**

eling in PAISs through change patterns. For such purpose, we state the following research questions:

RQ1. What language constructs are used to model variability in process families?

RQ2. What change patterns are needed to model variability in process families?

RQ3. How can the change patterns for process families be implemented?

RQ4. To what extent do the change patterns for process families improve the modeling of configurable process models?

These research questions are analyzed and answered in the following sections.

1.3 Thesis Goals

As stated above, the main goal of this thesis is to *enhance variability modeling in PAISs through change patterns.* This goal has been divided in different sub-goals in order to answer the presented research questions above. In the following, we summarize these sub-goals.

First of all, regarding **research question 1**, one of the sub-goals of this thesis is to study in-depth the process variability domain in order to identify how process variability is actually modeled. More precisely, we want to systematically analyze existing process variability approaches regarding their expressiveness with respect to process variability modeling as well as their support along the process lifecycle. In this context, we can identify the *language constructs that are used to represent process variability.* In addition, based on the empirical evidence provided by this study, we derive the *VIVACE framework.* Besides the identified variability-specific language constructs, this framework also comprises a core set of *features* fostering process variability. Thus, VIVACE

provides a characterization of process variability. In addition, VIVACE shall also allow for the systematic assessment and comparison of existing process variability approaches. Finally, VIVACE enables PAIS engineers to select for example that variability approach meeting their requirements best as well as help them in dealing with PAISs supporting process variability (e.g., to model or implement PAIS supporting process variability).

Regarding **research question 2**, another sub-goal of this thesis is to provide a set of generic patterns specifically tailored for modeling process variability. For such purpose, our *change patterns for process families* (CP4PF) are derived based on the variability-specific language constructs obtained from the previous sub-goal of the thesis. More concretely, CP4PF allow inserting, deleting, and modifying such constructs. In addition, our CP4PF are intended to ensure process family correctness, speed up the modeling process, and reduce the effort needed for such purpose by providing systematic means for introducing and deleting modeling elements.

In turn, regarding **research question 3** of this thesis, we want to illustrate how CP4PF can be realized. We do not only provide a theoretical and generic definition of the defined patterns, but also show how they can be implemented in a well-established approach for modeling configurable process models (i.e., *Configurable EPC* (C-EPC) [Gottschalk et al., 2007]). This implementation allows us to show that the proposed patterns support process variability management and can ensure process family correctness by inserting and deleting automatically modeling elements. For example, a pattern can facilitate the insertion of the function *Drop off bulk luggage* in the model of Figure 1.1. In particular, a PAIS engineer would indicate the position of the function in the model, and the implementation of the pattern would automatically take all the rest of necessary actions for correct insertion.

Finally, **research question 4** is aimed to show how CP4PF improve the modeling of process variability in configurable process models. First, in order to provide evidence that CP4PF are a feasible approach for modeling variability, we conduct a *case study* with a safety standard, which represents a process family with a high degree of variability. In

addition, in this case study we prove that CP4PF provide a considerable effort reduction needed for creating a configurable process model in comparison with three state-of-the-art approaches. Thus, configurable process models are modeled more efficiently when using CP4PF. Second, to complement the validation, we explore how *PAIS engineers experience* the application of CP4PF. More precisely, we implement the patterns in the *Cheetah Experimental Platform* [Pinggera et al., 2010] and study how PAIS engineers apply CP4PF, what is the impact of pattern application, and how PAIS engineers perceive pattern usefulness and ease of use. If PAIS engineers do not have a positive attitude toward CP4PF, their adoption will be hindered.

This thesis can be considered as a reference for implementing PAISs being able to effectively modeling process variability. First, we expect the VIVACE framework to be applied to various process variability approaches as well as related tools in order to assess their suitability with respect to process variability modeling. In this vein, the framework is expected to support organizations and PAIS engineers in deciding which process variability approach suits best to their needs. Second, the CP4PF are intended to be used for the modeling of process families. More precisely, since change patterns are based on the set of variability-specific language constructs, our CP4PF may be used, for instance, for creating new configurable process models or evolving existing ones. In addition, the results of the case study show that CP4PF are able to reduce the effort needed for modeling high-variable process families. Finally, PAIS engineers also find advantages in using CP4PF and have a positive attitude toward adopting them, which confirms the benefits of our patterns.

1.4 Research Methodology

In order to perform the work of this thesis, we have carried out a research project following the design methodology described in [Vaishnavi & Kuechler, 2004]. This design methodology was proposed for performing research in information systems. It involves the analysis of the use

and performance of designed artifacts to understand, explain and, very frequently, to improve on the behavior of aspects of information systems. Examples of such artifacts are system design methodologies and languages. In the case of this thesis, the designed artifacts are the characterization of process variability (i.e., the VIVACE framework) as well as the change patterns for process families (CP4PF).

The design methodology consists of a cycle of 5 process stages: (1) awareness of the problem, (2) suggestion, (3) development, (4) evaluation, and (5) conclusion (cf. Fig 1.2). In the following, we describe each stage in detail.

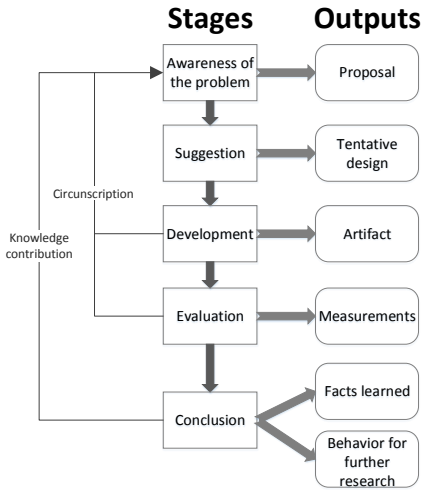


Figure 1.2: Design methodology cycle

Awareness of the problem: The awareness of a problem may come from multiple sources (e.g., new developments in industry or in a reference discipline, reading in an allied discipline). The output of this stage is a *proposal*, formal or informal, for a new research effort. In this thesis, the outputs of this stage refer to: (1) problem motivation, (2) research questions, and (3) review of the state of the art.

Suggestion: This stage follows immediately behind the proposal and is intimately connected with it. The output of this stage is the *tentative design*, which is an integral part of the proposal and must be targeted at it. This stage is an essentially creative step wherein a new artifact is envisioned based on a novel configuration of either existing or new and existing elements. In the context of this thesis, the outputs of this stage refer to: (1) thesis goals and (2) backgrounds of the proposed solution.
Development: The tentative design is further developed and implemented in this stage in order to produce an *artifact*. Implementation techniques will vary depending on the artifact to be constructed. The implementation itself may not involve novelty beyond the state-of-practice for the given artifact; the novelty is primarily in the design, not in the construction of the artifact. In the context of the thesis, the outputs of this stage refer to: (1) the VIVACE framework and (2) the definition and implementation of CP4PF.
Evaluation: Once the artifact has been constructed, it is evaluated according to criteria that are always implicit and frequently made explicit in the proposal. This stage contains an analytic activity in which *measurements* are usually taken about the behavior of the artifact. The results of the evaluation stage and additional information gained in the construction and running of the artifact are brought together and fed back to another round of suggestion (cf. circumscription arrow in Figure 1.2). In the context of this thesis, the outputs of this stage refer to: (1) the case study and (2) the validation with PAIS engineers.
Conclusion: This stage constitutes the end of a research. The results of this research are not only documented at this stage, but the knowledge gained is categorized as either *facts learned* or as *behavior that serves as basis of further research*. Awareness of the problem changes after conclusion thanks to the gained knowledge (cf. knowledge contribution arrow in Figure 1.2). In the context of this thesis, the outputs of this stage refer to: (1) contributions, (2) future work, (3) publications, and (4) the thesis itself.

1.5 Thesis Context

This thesis has been developed in the context of the research center *Centro de Investigación en Métodos de Producción de Software* (PROS)[2] of the *Universitat Politècnica de València*[3]. The work of this thesis has been developed in the context of the following research projects:

EVERYWARE: Construcción de Software Adaptativo para la Integración de Personas, Servicios y Cosas usando Modelos en Tiempo de Ejecución. CICYT project referenced as TIN-2010-18011.

The goal of EVERYWARE is to develop information systems that combine a set of available services (offered by an environment) to support the functionality required by end users. In many cases, this functionality is variable depending on the needs of the users. In this project, our CP4PF help to face the modeling of business processes reflecting this changing functionality.

SMART ADAPT: Desarrollo de Software Adaptativo en un Mundo Inteligente. Retos Tecnológicos en el ámbito de la Ingeniería Dirigida por Modelos. MINECO project referenced as TIN2013-42981-P.

One of the challenges of SMART ADAPT is to provide a framework to allow the self-evolution of models. In this context, the use of CP4PF contributes to ensuring the automatic evolution of these models. In addition, CP4PF guarantee the proper evolution of the models by ensuring model correctness.

[2] www.pros.upv.es
[3] www.upv.es

1.6 Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2: introduces the main research areas that are related to this work in order to provide a basic background for understanding the overall thesis.

Chapter 3: reviews the most relevant existing works related to the thesis. Their analysis is necessary to determine the current state of research and practice.

Chapter 4: presents the VIVACE framework. VIVACE is resulted from an systematic study of the process variability domain. Thus, it constitutes a complete process variability characterization. In addition, VIVACE enables for the systematic assessment and comparison of process variability approaches.

Chapter 5: presents the set of change patterns for modeling of process families (CP4PF). In particular, CP4PF are described, illustrated, and provided with implementation details.

Chapter 6: reports how we put CP4PF into practice in a real and industrial scenario. It describes a case study performed with a safety standard as a feasibility proof of CP4PF. In addition, we measure the effort needed to apply the patterns in a real scenario, which is compare with three state-of-the-art approaches in order to determine the advantages of using CP4PF.

Chapter 7: details the validation with PAIS engineers in order to analyze the impact of CP4PF as well as how PAIS engineers experience their application.

Chapter 8: summarizes the main conclusions that can be drawn as a result of the development of this thesis. It describes the contributions that have been made, discusses the impact of the thesis, and presents the future work that could be performed.

2

Background

This thesis deals with the enhancement of process variability modeling through the use of change patterns. In order to describe its specific domain, the thesis is placed in the intersection of three research areas: *Business Process Modeling, Software Variability Modeling*, and *Software Patterns* (cf. Figure 2.1). That is, this thesis relies on the different concepts and techniques from these areas. In this chapter, we introduce these concepts and techniques in order to clarify the foundations in which our work relies on and provide a basic background for understanding the overall thesis.

The rest of the chapter is organized as follows. Section 2.1 introduces the concepts that support business processes modeling. In turn, Section 2.2 describes the main concepts of software variability modeling. Section 2.3 outlines the pattern-based techniques used for the development of software. Finally, Section 2.4 concludes the chapter.

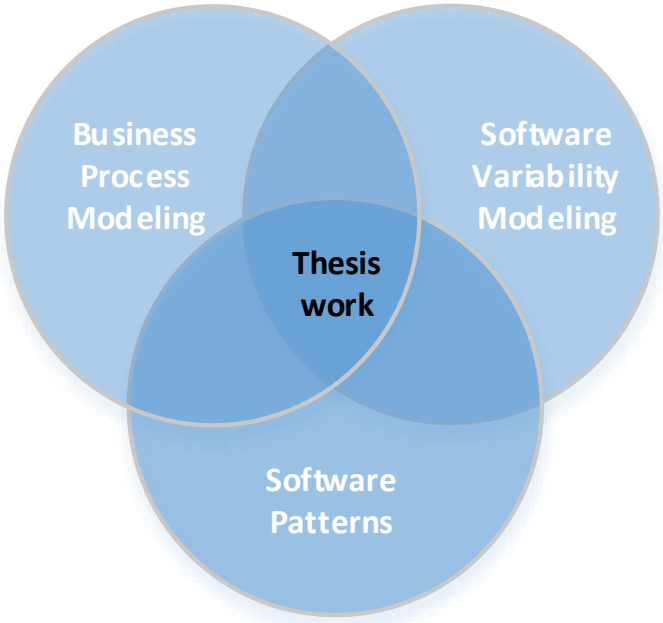


Figure 2.1: Research areas involved in this thesis

2.1 Business Process Modeling

Nowadays, the modeling of business processes is a very common practice in organizations. It plays a major role both in industry and academia, helping organizations to be competitive and to achieve their business goals [Indulska et al., 2009].

According to [Weske, 2007], a business process is defined as *"a set of activities performed in coordination in an organizational and technical environment"*. Analyzing this definition, a business process defines *what* (activities) shall be done, *how* it shall be done (coordination), and by *whom* (organizational and technical environment). In this context, *business process models* (also named process schemas) constitute the main artifacts for representing the respective business processes. A business process model (*process model* for short) is used within organizations for communication and learning purposes, for decision support about process development and design, for control and decision support during process execution, and for analysis of information technology support [Aguilar-Savén, 2004].

Basically, process models are defined from the basic primitives from the metamodel depicted in Figure 2.2.[1] These primitives allow for modeling the functional, behavioral, organizational, informational, temporal, and operational perspectives of a business process [Curtis et al., 1992; Melao & Pidd, 2000; Korherr, 2008; Reichert & Weber, 2012; Jablonski & Bussler, 1996; Lanz et al., 2010, 2012].

- The *functional perspective* specifies the decomposition of a business process into units of work, i.e., it represents the *activities* that may have to be performed to reach a particular *business objective* [Curtis et al., 1992]. An atomic activity is associated with a single action, whereas a complex activity refers to a sub-process or, more precisely, a sub-process model. In Figure 2.2, this perspective is represented by entities *activity, atomic activity,* and *complex activity.*

- The *behavioral perspective* captures the behavior of a process model and hence reflects the *control flow* between its activities. The latter defines the order of the activities as well as the constraints for their execution. This perspective is represented by entities *control connector* (i.e., gateway) and *control edge* (i.e., arrows) in Figure 2.2.

[1] This metamodel has been adopted from [BPDM, 2014].

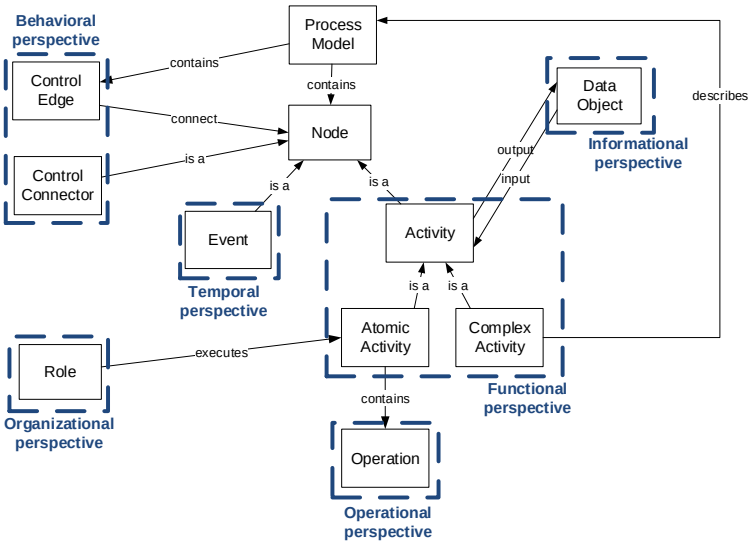


Figure 2.2: Process metamodel adopted from [BPDM, 2014]

- The *organizational perspective* represents the different *actors or roles* involved in a process model that are in charge of executing particular process activities (i.e., humans or systems). This perspective is represented by entity *role* in Figure 2.2.
- The *informational perspective* covers *data* and *data flow*, i.e., it represents the informational entities (e.g., data objects) consumed (i.e., used as input for activity execution) or produced (i.e., as output resulting from activity execution) during process execution. This perspective is represented by entity *data object* in Figure 2.2.
- The *temporal perspective* covers temporal constraints restricting the *execution and scheduling* of activities; e.g., the time an activity may be started or completed, a message arrived, a deadline expired, or an error occurred. This perspective is represented by entity *event* in Figure 2.2.

- The *operational perspective* refers to the implementation of atomic process activities, i.e., the application services (e.g., web services, electronic user forms) to be invoked when these *activities* are started. For a particular atomic activity, different implementations may exist. At enactment time, one of them is then dynamically selected and bound to the execution of this activity. This perspective is represented by entity *operation* in Figure 2.2.

In general, process models are embedded in a process lifecycle comprising different phases (cf. Figure 2.3) [Weske, 2007; Bridgeland & Zahavi, 2008; Aguilar-Saven, 2004]. These phases include: *Analysis & Design, Configuration, Enactment, Diagnosis,* and *Evolution.*

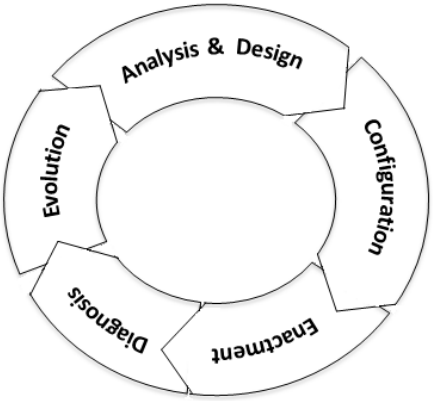


Figure 2.3: Business process lifecycle

During the *analysis and design* phase, based on domain requirements, relevant (emerging or existing) process information is gathered, analyzed, consolidated, and represented in terms of process models. These models are also validated and verified based on various techniques (e.g., simulation, correctness checks). In turn, during the *configuration* phase, the process model is enhanced with technical information that facilitates its execution (e.g., information about the enactment system). Once this configuration is completed, process instances can be *enacted.*

A process instance represents a concrete case in the operational business of an organization [Weske, 2007]. Thus, the process enactment phase encompasses the actual execution of the business process. Monitoring techniques are used in this phase in order to gather relevant information and data about the process instances' enactment. Later, in the *diagnosis* phase, this gathered information is analyzed in order to identify problems and to find process aspects that can be improved (e.g., bottlenecks). Finally, the identified improvements as well as emerging requirements (e.g., changes in the business context) can lead to the *evolution* of the process models. This may also entail the ability to change process instances accordingly (i.e., the ones being executed). Process evolution may be *incremental* (i.e., only requiring small changes of the implemented process) as for continuous process improvements, or be *revolutionary* (i.e., requiring radical changes) as in the context of process innovation or process re-engineering [Reichert & Weber, 2012].

There are two types of process models: (1) *business-oriented* process models and (2) *workflow* models [La Rosa, 2009]. Business-oriented process models are high-level models used for analyzing the domain requirements at the early stages of the lifecycle (i.e., the analysis and configuration stages). In general, these models provide a basis for communication among relevant stakeholders, and as such they must be unambiguous as well as intuitive. In turn, workflow models are designed for process automation. They are typically obtained by refining business-oriented process models with information that is relevant for implementation. Their execution is supported by a workflow management system.

Both types of process models are created by means of a graphical notation named *business process modeling language*. For example, for creating business-oriented process models, the most representative language are: Event-driven Process Chains (EPC) [SAP-Business-Suite, 1992], UML Activity Diagrams [UML, 2007], and Business Process Modeling Notation (BPMN) [BPMN, 2011]. In turn, for workflow models, the main languages are: the Web Services Business Process Execution Language (WS-BPEL or BPEL) [WS-BPEL, 2004] and Yet Another Workflow Language (YAWL) [van der Aalst & ter Hofstede, 2003]. Some of these languages have appeared in academia and adop-

ted later in industry (e.g., BPMN), whereas others have been initially defined for industrial purposes and improved by means of academic research (e.g., EPC). However, the common characteristics of all these languages is their capability for representing the sequence of activities of a business process, the involved stakeholders and the data or messages interchanged between them [La Rosa, 2009]. Since these languages have been developed with specific purposes, a "universal" language for business process modeling does not exist. Only BPMN has been standardized by the OMG consortium [OMG, 1989] since it contains notational information and execution semantics (e.g., BPMN 2.0).

2.2 Software Variability Modeling

During the last decade, variability modeling has become a hot topic [Schmid & John, 2004; Cetina et al., 2009; Alférez et al., 2014]. In the context of information systems, *variability modeling* is referred to representing the capability to change of a software system [Geyer & Becker, 2002]. One of the main interests on variability modeling comes from the field of *Software Product Lines* (SPLs) [Pohl et al., 2005].

SPLs encompass the creation and management of similar software products (products' families) for a particular domain. Most organizations build software within a few domains, repeatedly building system variants within those domains. This is achieved by defining only once the common product functionalities (i.e., shared by all family members) and combining them with a set of *variation points*. A variation point is an identifier of one or more locations in a product at which the variation will occur [Pohl et al., 2005]. By explicitly defining variation points, time, effort, cost and complexity of software creation and maintenance are reduced [Krueger, 2013].

In general, a SPL can be described in terms of four concepts (cf. Figure 2.4).

- **Software asset inputs**. A set of software assets (e.g., requirements, source code, test cases, and documentation) that can be

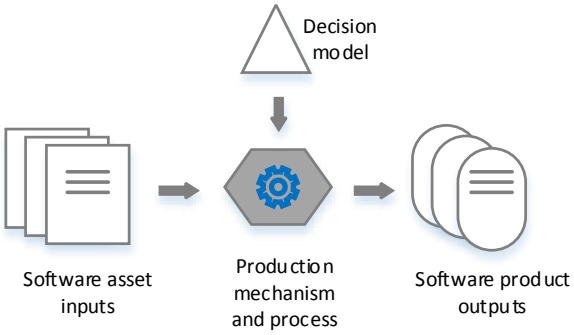


Figure 2.4: Main concepts of software product lines

configured and composed in different ways in order to create all of the products. To obtain the different products, assets may be optional and also they can be configured in different ways to provide different behavior.

- **Decision model**. Decisions describe optional and variable characteristics for the products (i.e., variation points). Each product is uniquely defined by its decisions.
- **Production mechanism and process**. A set of prescribed rules is used for composing and configuring each product from the software asset inputs, i.e., product decisions are used to determine which software asset inputs are used and how they are configured.
- **Software product outputs**. The set of all products that can be produced in a product line from the software assets inputs and the existing decision model.

Three main processes are involved in SPLs: *Domain Engineering*

(DE), *Application Engineering* (AE) and *Management* [Bosch et al., 2001]. During DE, the variability of a SPL is defined and common and variable domain artifacts are developed (i.e., the decision model is created). During AE, individual products are developed by selecting and configuring shared artifacts and, where necessary, adding product-specific extensions (i.e., the production mechanism and process is performed). Finally, during management, organizational issues are handled in order to obtain the products (e.g., by giving resources).

Recently, a number of methods and techniques for managing SPLs have been defined [Bayer et al., 2006]. In this thesis, we highlight two of them regarding their ability to model variability. The first one refers to *Software Configuration Management* (SCM) [Pressman, 2001], a methodology for controlling and managing a product family. Work on SCM has led to models and languages to capture how a set of available options impacts upon the way a software system is built from a set of assets. These options conform the variable parts of the system. Examples of SCM languages are the *Adele Configuration Manager* [Estublier & Casallas, 1994] and the *Proteus Configuration Language* [Tryggeseth et al., 1995]. The second method, and the most studied one, refers to *feature models* [Schobbens et al., 2006]. These models are tree-structures used to describe a set of products in terms of their *features*. A *feature* corresponds to a logical unit of behavior or functionality by which different products can be distinguished and defined (i.e., features represent variation points). It can be mandatory (the feature is always used in the product) or optional (the feature can be used in the product). In addition, a feature can be bound to other features via inclusion and exclusion constraints and it can be decomposed into a set of sub-features. The limit of sub-features that a feature can have is determined by logical relationships. The AND relationship indicates that all the sub-features must be selected. In turn, the XOR relationship indicates that only one sub-feature can be selected. Finally, the OR relationship indicates that one or more sub-features can be selected. This OR can be further specified with an [n..m] cardinality [Czarnecki & Antkiewicz, 2005], where n indicates the minimum and m indicates the maximum number of allowed sub-features. In addition, it is possible to define the

features that are included in the product by default. A *configuration* of a feature model specifies then a valid scenario in terms of features selected/deselected, i.e. a scenario that complies with the defined constraints. This configuration represents the configuration of the product in the software product line. Figure 2.5 shows an example of a generic feature model.

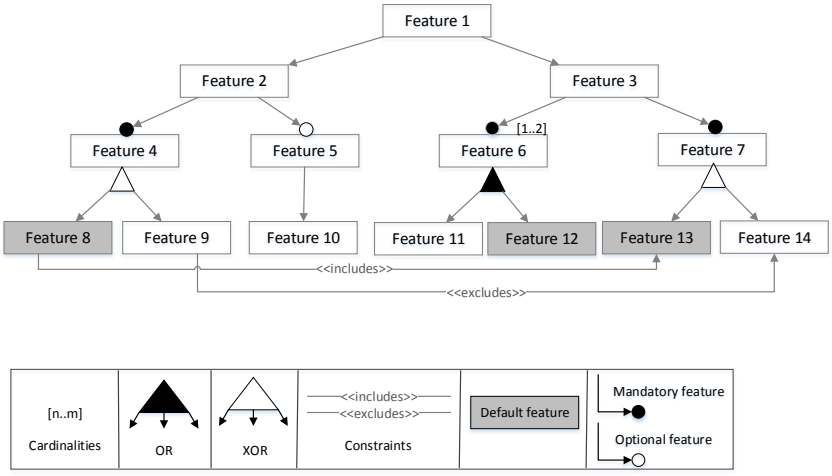


Figure 2.5: Generic feature model

2.3 Software Patterns

Another actual hot topic in information systems refers to the use of *patterns* for modeling software. Patterns were initially introduced in 1977 as an architectural (building) concept [Alexander et al., 1977]. But it was in 1989 when patterns were applied for the first time for software development [Fowler, 1997; Beck & Cunningham, 1987]. Since that year, there has arisen several definitions of the term 'pattern' [Martin, 2000;

Riehle & Züllighoven, 1996; Appleton, 1997; Gabriel, 1996]. However, in this thesis we consider the definition provided by *Martin* in [Martin, 2000] since it is the most commonly used. For *Martin*, a software pattern is defined as a *"general reusable solution to a commonly occurring problem within a given context"*. These software patterns constitute common practices in software development. In addition, software patterns need to be based on empirical observations (i.e., being demonstrable in practice).

Depending on the abstraction level they refer [Fernandez, 1998], patterns can be classified as:

- **Organizational patterns**, oriented to describe the structure of an organization.
- **Architectural patterns**, oriented to define the architectural structure of a system.
- **Idioms**, oriented to a given programming language.
- **Analysis patterns**, oriented to help in the conceptual modeling of the system.
- **Design patterns**, oriented to describe design constructs while designing the system.

2.3.1 Organizational patterns

Organizational patterns constitute solutions for describing structures of relationship which help an organization to achieve its goals [Coplien & Harrison, 2005]. These patterns are inspired by analyzing multiple professional organizations and finding common structures in their social networks [Coplien & Harrison, 2005]. Organizational patterns can be divided into four groups (cf. Figure 2.6):

- Patterns for *project management*. They point to the initial design of an organization.

Figure 2.6: Set of organizational patterns

- Patterns for the *picemeal growth*. They refer to the growth of an organization once it is up and running.

- Patterns for the *organizational style*. They shape the "style" of an organization and provide a good foundation for tailoring an organization to its business and market.
- Patterns for *people and code*. They help an organization in aligning the people (e.g., developers) and code structures properly.

The fundamental process for applying an organizational pattern in an organization is:

1. Find the weakest part of the organization.
2. Find a pattern that is likely to strengthen it.
3. Apply the pattern.
4. Measure the improvement or degradation.
5. If the pattern improved things, go to step 1 and find the next improvement; otherwise, undo the pattern and try an alternative.

Finally, organizational patterns capture the foundations of the agile software development movement. In particular, they have inspired the creation of parts of *Scrum* [Schwaber, 2004] and of *Extreme Programming* [Beck, 1999]. For a complete description of organizational patterns, we refer to [Coplien & Harrison, 2005].

2.3.2 Architectural patterns

Architectural patterns are well-established problem-solution pairs to architectural problems that occur in a given context and are affected by it [Coplien & Alexander, 1996]. The software architecture of a system comprise the set of structures needed for reasoning about the software components of a systems, their properties, and their relations [Avgeriou & Zdun, 2005]. An architectural pattern does not only document "how" to solve an architectural problem, but also "why" it needs to be solved (i.e. the rationale behind the solution) [Bass, 2007]. In addition,

architectural patterns help to document the architectural design decisions, facilitate communication between stakeholders through a common vocabulary, and describe the quality attributes of a software system as restrictions that must be fulfilled [Buschmann et al., 2007].

Architectural patterns can be classified according to the different architectural views of a system [Clements et al., 2002]. An architectural view is a representation of a system from the perspective of a related set of its components. Thus, an architectural pattern defines the types of components, properties, and relationships that work together to solve a particular problem from a certain view. The existing views are:

- The *layered* view. It deals with how a system, as a complex heterogeneous entity, can be decomposed into interacting components. Examples of architectural patterns in this view are: layers and indirection layer.
- The *data flow* view. It deals with the data streams that are successively processed or transform by the components of a system. Examples of patterns in this view are: batch sequencial and pipes and filters.
- The *data-centered* view. It deals with multiple components of a system that access a central data repository. Examples of patterns in this view are: shared repository, active repository, and blackboard.
- The *adaptation* view. It deals with how a system can adapt its own behavior/components at evolution time. Examples of patterns in this view are: microkernel, reflection, and interceptor.
- The *language extension* view. It deals with how a system provides an abstraction layer for the computing infrastructure. Examples of patterns in this view are: interpreter, virtual machine, and rule-based system.
- The *user interaction* view. It deals with the components of the user interface of a system that are shown at runtime. Examples of

patterns in this view are: model-view-controller and presentation-abstraction-control.

- The *component interaction* view. It deals with how individual components exchange messages, but keeping their autonomy. Examples of patterns in this view are: explicit invocation, implicit invocation, client-server, peer-to-peer, and publish-subscribe.

- The *distribution* view. It deals with how to distribute the components of a system in a network. Examples of patterns in this view are: broker, remote procedure calls, and message queuing.

2.3.3 Idioms

Idioms constitute the lowest-level patterns since they depend on a specific implementation technology such as a programming language (e.g., C++, Java) [Coplien, 1997]. In general, an idiom refers to a syntactical shortcut that does something not immediately obvious from the code itself but which is used often enough that other programmers recognize its meaning. For example, the structure `i += 1` in Java is obvious for a Java programmer, but could be a mystery to a non-expert in Java. Idioms are not restricted to any programming paradigm and can be defined for example for object-oriented programming (e.g., Java), functional programming (e.g., Haskell) [Gibbons, 2010], aspect-oriented programming (e.g., AspectJ) [Lesiecki, 2005], and special-purpose programming (e.g., SQL) [Tropashko & Burleson, 2007].

Finally, some programmers argue that data structures such as queues, linked lists, trees, graphs, or stacks constitute idioms as well. Although they may seem more primitive as compared to more high level patterns (e.g., architectural patterns), data structures define (non-obvious) solutions to store data and process them. The same could be stated to the entity `class` since it constitutes a template for managing objects (e.g., creation, implement behavior).

2.3.4 Analysis patterns

Analysis patterns capture an abstraction of a situation that can often be encountered in software modeling. They were defined by Fowler in 1997 based on empirical observations [Fowler, 1997]. An analysis pattern can be represented as a group of related, generic objects (metaclasses) with stereotypical attributes (data definitions), behaviors (method signatures), and expected interactions defined in a domain-neutral manner that represent a common construction in business modeling. In general, analysis patterns facilitate the transformation of the domain requirements of a system into a conceptual model. Since an analysis pattern may be relevant to many domains, they are very valuable for promoting reuse among the systems. In particular, there are six groups of analysis patterns whose names come from the domain patterns were observed the first time (cf. Figure 2.7). In this thesis, we outline analysis patterns very briefly. For a complete description of analysis patterns, we refer to [Fowler, 1997].

- Patterns for *accountability*. They describe relationships that define responsibilities between the parties of a system.
- Patterns for *observations and measurements*. They are used for recording facts.
- Patterns for *referring to objects*. They focus on indexing for referring objects in an exact way.
- Patterns for *inventory and accounting*. They focus on accounting, describing how a network of accounts can form an active accounting system.
- Patterns for *planning*. They depict the relationship between standard plans and one-off plans, and how to plan and record the use of resources.
- Patterns for *trading*. They focus on trading in situations where prices are fluid and we need to understand how these price changes affect the profits.

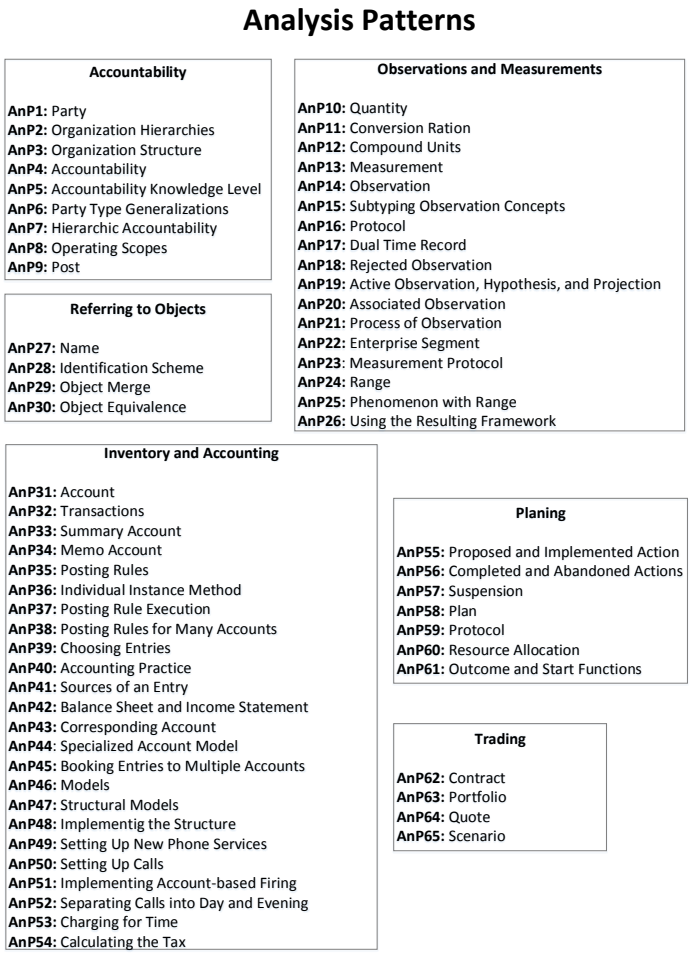


Figure 2.7: Set of analysis patterns

Figure 2.8 shows an example of an analysis pattern. More precisely, it shows the party pattern of the accountability category. This pattern is applied when people and units of an organization have similar features or responsibilities. The solution provided by the pattern lies on creating

a type party as a supertype of person and organization.

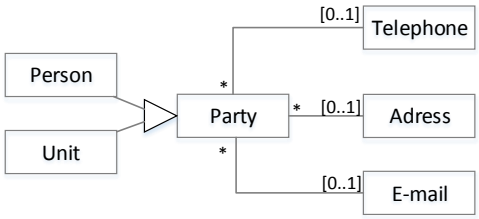


Figure 2.8: Analysis patterns: Accountability - Party pattern

2.3.5 Design patterns

Design patterns gained popularity after the book of the so-called *"Gang of Four"* [Gamma et al., 1995]. Unlike analysis patterns, design patterns are formalized best practices that can be used to solve common problems when designing a system in an object-oriented way. A design pattern has four essential parts: a statement of the *context* where the pattern is useful, the *problem* that the pattern addresses, the *forces* that play in forming a solution, and the *solution* that resolves those forces. There are 23 design patterns classified in three groups based on their functionality (cf. Figure 2.9). In the following, we summarizes these patterns. For a complete description of design patterns, we refer to [Gamma et al., 1995].

- *Creational* patterns. They deal with the initialization and configuration of classes and objects. More precisely, these patterns abstract the creation process of the instances of the classes of the model.

- *Structural* patterns. They deal with decoupling the interface and the implementation of classes and objects. More precisely, these

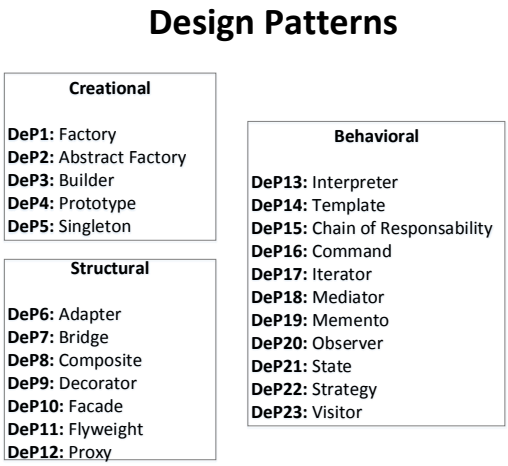


Figure 2.9: Set of design patterns

patterns focus on how classes and objects are used to compound bigger structures.

- *Behavioral* patterns. They deal with the dynamic interaction among societies of classes and objects. More precisely, these patterns describe the algorithms and assignation of responsibility among the existing objects.

Figure 2.10 shows an example of a design pattern. In this case, it shows the facade pattern. This pattern is aimed to provide a unified interface for a set of interfaces. In this way, the unified interface facilitates the interaction with the other interfaces reducing the complexity of the system and its dependencies.

Design patterns differ from analysis patterns in three ways [Fernandez, 1998]:

- Design patterns relate to system implementation and are focused

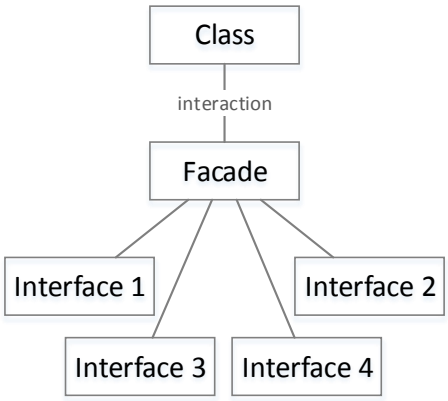


Figure 2.10: Design patterns: Structural - Facade pattern

on typical design aspects (e.g., user interfaces, objects' creation, and basic structural properties).

- Design patterns can be applied to any system (e.g., all systems have user interfaces or need to create objects).
- Analysis patterns depend on the specific system and their semantics describe aspects of this system or its application domain.

2.4 Conclusions

The purpose of this chapter is to provide an introduction to the foundations and the basic background of the research areas this work relies on. We have described the different concepts and techniques of three areas: *Business Process Modeling, Software Variability Modeling* and *Software Patterns*. This thesis is related with these areas since we aim to provide a set of change patterns (i.e., software patterns area) specifically tailored for modeling variability (i.e., software variability modeling

area) in process families (i.e., business process modeling area). Thus, much of the techniques introduced here are used as a basis in this thesis. In the following, we provide an overview of existing approaches closely related to the described research areas as well as to the goals of this thesis.

3

State of the Art

Once we have analyzed in Chapter 2 the main research areas in which this thesis relies on, in this chapter we analyze existing approaches closely related to these areas as well as to the goals of this thesis. Figure 3.1 illustrates the research areas and their subareas (intersections), where each of the analyzed approaches is placed. More precisely, we identify three subareas: *Business Process Variability Modeling*, *Software Variability Modeling Patterns*, and *Business Process Modeling Patterns*. Relevant approaches in these subareas are analyzed and discussed in this chapter.

The rest of the chapter is organized as follows. Section 3.1 describes approaches related to the modeling of business process variability. Section 3.2 outlines the main patterns that are used for modeling variability in software systems. In turn, Section 3.3 introduces patterns that support changes in business process modeling. In Section 3.4 we discuss the difference between the existing approaches and the work of this thesis.

Finally, Section 3.5 concludes the chapter.

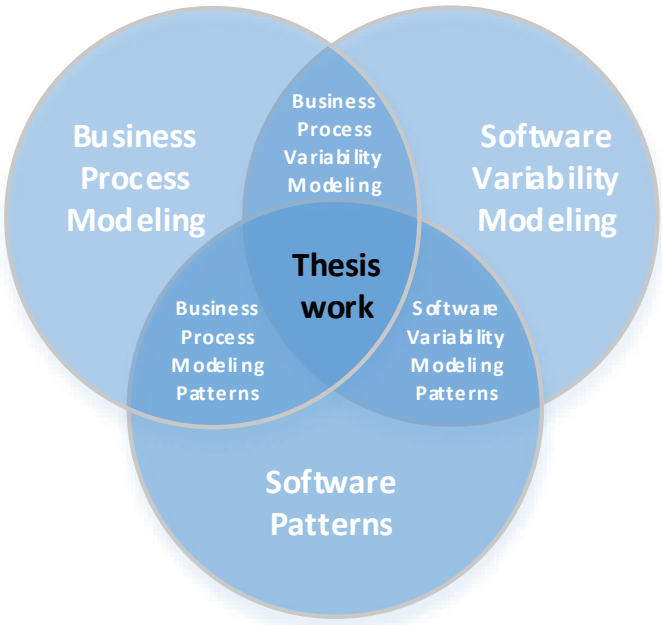


Figure 3.1: Research areas involved in this thesis and their intersecting sub-areas

3.1 Business Process Variability Modeling

The co-existence of multiple *variants* of the same business process model is a widespread phenomenon in contemporary organizations. These process variants pursue the same or similar business objective sharing certain commonalities (e.g., process fragments), while at the same time having differences due to their use in the different application context (e.g., certain fragments may be relevant for only some of the process variants depending on the application context) [Dijkman et al., 2012; Soffer, 2005; van der Aalst & Basten, 2002].

A collection of related process variants constitutes a *process family* [Reichert & Weber, 2012]. Applying conventional business process modeling approaches (cf. Section 2.1) to process families requires either (1) to model each variant separately or (2) to model multiple variants together [La Rosa et al., 2013]. In the first case, the result is a significant redundancy (i.e., duplication of process fragments) as the variants have much in common. In the second case, the complexity of the consolidated model grows rapidly and it becomes difficult to analyze and maintain individual variants [La Rosa et al., 2013].

Motivated by this observation, process variants are typically defined in terms of a *configurable process model*, which represents a complete process family [Reichert & Weber, 2012]. In particular, a configurable process model eliminates model redundancies by representing the commonalities of different process variants only once. Furthermore, it fosters model reuse since variant particularities can be shared among multiple variants [La Rosa et al., 2009a].

3.1.1 Process perspectives

Like business process models, in the configurable process models, the functional, behavioral, organizational, informational, temporal, and operational can be observed. In general, all these perspectives may be subject to variation. For example, in respect to the functional perspective, consider Variants 4-6 of the check-in process (cf. Appendix A). Depending on the type of passenger, the set of activities to be performed

may differ; e.g., *Assign seat for UM* in the context of unaccompanied minors, *Assign seat for handicapped* in the context of handicapped passengers, or *Assign seat* for regular passengers. In turn, regarding the behavioral perspective, the control flow of the check-in process differs for example in the model part preceding activity *Print boarding card*; e.g., activities *Provide information about accommodation* and *Fill in ESTA form* are only performed if the passenger is traveling to the US, but shall be omitted otherwise. Thus, there exist two options in the control flow of the process; i.e., either to perform the activities or to skip them. In turn, variations in the organizational perspective is found when the check-in can be performed by *passengers* using a web system (cf. Variants 1-2 in Appendix A), whereas check-in at the counter is performed by *airline staff* (cf. Variants 4-6). In addition, depending on the type of check-in, the resulting boarding card either is an electronic or a paper-based document, which refers to variability in respect to the informational perspective. In respect to the temporal perspective, the availability of the check-in service is delimited from 23 (cf. Variants 1-3) to 3 (cf. Variants 4-6) hours before departure, depending on the type of check-in. This is represented using different start events. Finally, regarding the operational perspective, the implementation of the *Print boarding card* activity differs depending on the type of check-in; i.e., online, counter, or with self-servicing machine.

3.1.2 Process lifecycle

Configurable process models also follow the phases of the process lifecycle (cf. Figure 2.3). In the context of process variability, at the analysis and design phase, process variants are *defined* in terms of a configurable process model, which must be verified and validated. In this case, verification means that it needs to be ensured that all process variants that may be derived from the configurable process model are syntactically correct and sound (e.g., no deadlocks or livelocks). In turn, validation shall ensure that the configurable process model properly reflects the semantics of all business processes. In order to derive a specific process variant from the configurable process model, at the

configuration phase, an *individualization* as well as a *selection procedure* are performed based on the respective application environment (i.e., application context) in which a process variant shall exist [La Rosa et al., 2009a]. Then, this individualized (and selected) process variant is deployed to the target process engine for its enactment. Figure 3.2 depicts the transitions from the analysis and design of a process family to the enactment of a process variant instance.

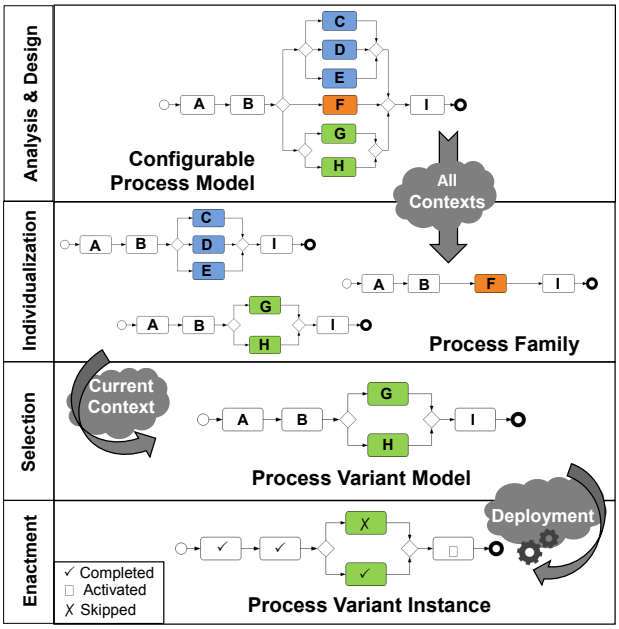


Figure 3.2: From process family definition to process variant enactment

At enactment time, it needs to be guaranteed that process variants are *executed* according to the configured process variant model [van der Aalst et al., 2010b; Hallerbach et al., 2009; Ayora et al., 2012a]. In addition, this phase covers configuration decisions that may only be made during enactment time (i.e., *dynamic configurations*) [Angles et al., 2013; Murguzur et al., 2014]. Even though configuration is partially

performed at enactment time, soundness should be ensured at design time. Another aspect to take into account during this phase refers to *dynamic re-configurations*, i.e., to switch from the current process variant to another [Soffer, 2005; van der Aalst & Basten, 2002]. In this case, sophisticated exception handling techniques are necessary (e.g., to abort instances that are no longer needed [Reichert & Weber, 2012]). Accordingly, monitoring techniques are required to provide accurate information about the current execution state of the process variant instance. The monitored data is then used in the *diagnosis* phase to identify possible model optimizations that, in turn, will guide the *evolution* of the family. This evolution may be referred to evolving (i.e., modifying) a single process variant or to evolve the schema of a process family, which results in a new process family (i.e., to evolve the configurable process model) [Ayora et al., 2012a]. In this context, co-existing schema versions of a configurable process model may have to be maintained. This means that conflicts between single process variants which have been individually evolved, and the evolution of the configurable process model need to be handled.

3.1.3 Process variability approaches

Recently, a number of approaches to create configurable process models have emerged. By treating variability as a first class citizen, *process variability approaches* avoid model redundancies and foster model reusability [Reichert & Weber, 2012]. In general, these approaches allow representing a configurable process model either in a single artifact or in a set of related artifacts [Ayora et al., 2015]. Using a single artifact, all process variants and related aspects (e.g., commonalities of the process variants, variant-specific parts, configuration constraints, and application context) are included in a single model. On the contrary, using a set of related artifacts these aspects are defined separately in different models. In the following, we provide an overview of specific approaches for both methods.

First, regarding the use of a single artifact, *hiding & blocking* operators have been defined to configure a process variant by making unob-

servable (i.e., hiding) or disabling (i.e., blocking) those execution paths that are not included in the variant [Schunselaar et al., 2012]. Another approach refers to the use of *configurable nodes*. These nodes represent variation points to which different alternatives can be assigned. In addition, configuration constraints (named configuration requirements) restrict the combination of allowed alternatives [Yao & Sun, 2012]. Configurable nodes have been used in combination with conventional process modeling languages such as EPC and YAWL [Gottschalk et al., 2007]. In addition, configurable nodes can be used to define variability at any process perspective (e.g., organizational, informational) [La Rosa et al., 2011]. Another approach to capture process variability in a single artifact refers to the use of *temporal logic* [Groefsema et al., 2011]. That is, processes are defined as directed graphs combined with logic formulae, which represent configuration constraints. Further, *annotated models* is also an approach used to represent process variability. Annotations (e.g., stereotypes, labels) are included in the configurable process model to accommodate variability. Annotations have been defined for BPMN [Frece & Juric, 2012; Döhring et al., 2011] and for EPC [Reijers et al., 2009; Becker et al., 2004]. In addition, *meta-model extensions* for UML Activity Diagrams [Moon et al., 2008; Saidani & Nurcan, 2014; Kolokolov et al., 2014; Marcolino et al., 2014] and BPEL [Lazovik & Ludwig, 2007] have been proposed in order to realize configurable process models. In the same vein, *multiplicity indicators* can be also attached to modeling elements (e.g., activities) to denote the possible lowest and upper-most numbers of variants these elements may have in a process family [Reinhartz-Berger et al., 2010]. Finally, a *hierarchical indexing structure* to capture variability at the business goal level has also been proposed [Derguech et al., 2010]. In this hierarchical representation, a variation point is a business goal that has more than one way (i.e., business variant) to be achieved.

Second, when a process family is represented in different artifacts, it is typically defined in terms of a *base model*, a set of *variable process fragments*, a set of *rules* to define when the variable process fragments are used in the base model, and the definition of an *application context* determining when these rules are applied. Thereby, the *base model*

is specified using a conventional business process modeling language (e.g., BPMN). However, different policies may be applied when defining this base process model, e.g., setting the latter to the most frequently used process variant or to the process model having minimum average edit distance to the rest of process variants of the family [Li et al., 2011]. Concerning the three other artifacts, different techniques for defining them exist. For representing variable process fragments, for example, *features models* (from software product lines) [Alférez et al., 2014; Montero et al., 2008; Schnieders & Puhlmann, 2007; Czarnecki & Antkiewicz, 2005] or *goal models* [Lapouchnian et al., 2007; Angles et al., 2013] can be used. In addition, variable process fragments may be defined based on a set of *process model components* [Sakr et al., 2011], a *variant list* [Meerkamm & Jablonski, 2011], or a set of pre-specified *change operations* [Hallerbach et al., 2010b; Lu et al., 2009]. In turn, the rules for adapting the base model may rely on techniques such as *business rules* [Kumar & Yao, 2012], *process model queries* [Sakr et al., 2011], and *non-functional constraints* [Lapouchnian et al., 2007]. Finally, *ontologies, semantic rules, questionnaire models,* and *context analysis methods* may be used for defining the application context of process variants [Alférez et al., 2014; Yao et al., 2012; La Rosa et al., 2009b; Santos et al., 2012; de la Vara et al., 2010].

Regarding lower-level representations of process families, it is also possible to define process variability by *coding* an algorithm [Tealeb et al., 2014]. In addition, *event logs* are also used to identify commonalities between processes of different organizations [Buijs & Reijers, 2014]. Further, *declarative specifications* can also be used with the same purpose [Jiménez-Ramírez et al., 2015]. However, these approaches are out of the scope of this thesis since we focus exclusively on process models.

Several empirical evaluations of process variability approaches have been conducted [Ayora et al., 2015]. Case studies are the most frequent method and have been conducted in different domains such as egovernment [Gottschalk et al., 2009], logistic [Lönn et al., 2012], risk management [Scherer & Sharmak, 2011], smart cities [Murguzur et al., 2013],and retail [Pascalau & Rath, 2010]. Regarding other types of evaluations, the *Goal/Question/Metric method* is used to evaluate how

good the design of a configurable process model is [Alférez et al., 2014]. In turn, [Reijers et al., 2009] reports on the benefits that practitioners found after they interacted with a configurable process model. Similarity metrics to measure the complexity (e.g., size) of a configurable process model are used in [Vogelaar et al., 2011]. Mapping patterns to compare different process variability approaches in terms of complexity (e.g., size of resulting models) are also used in [Baier et al., 2010].

3.2 Software Variability Modeling Patterns

Patterns have also been applied to model software product lines since they can reduce time, cost, and effort (cf. Section 2.3). Up to our knowledge, there are two main sets of patterns that allow modeling variability in product lines: (1) single, multiple, and option patterns [Keepence & Mannion, 1999], and (2) patterns for evolving event-based systems [Tragatschnig et al., 2013].

3.2.1 Single, Multiple, and Option patterns

In [Keepence & Mannion, 1999], authors have developed a method for building product family models using a set of predefined patterns to model family variations. The method starts by analyzing existing user requirements from systems within the product family and identifying the *discriminants*. Three types of discriminants are used: single discriminants (i.e., mutually exclusive features), multiple discriminants (i.e., optional features that are not mutually exclusive), and option discriminants (i.e., single optional features that might or might not be used). These discriminants and their three associated patterns (cf. Figure 3.3) are used and combined to produce a unified family model that includes all commonality and variation across the family.

1. **Single adapter pattern**. The single discriminant can be modeled as an inheritance hierarchy in which generic features are modeled in a base class and specific features are modeled as mutually exclusive subclasses (i.e., only one subclass can be instantiated in a

product). Virtual functions are used in the base class for accessing the methods in the subclasses so other model parts can refer to instances of this base class without knowing which subclass a given product will use. The single adapter pattern is implemented using the singleton design pattern (cf. Figure 3.3 part 1).

2. **Multiple adapter pattern**. The multiple discriminant is modeled in the same way as a single discriminant, i.e., as an inheritance hierarchy with a base class and specific subclasses. However, in this pattern, more than one subclass can be instantiated in any single system. To access methods in a particular subclass, each subclass instance is identified by a name, which is stored in a collection (cf. Figure 3.3 part 2).

3. **Option pattern**. The option discriminant is modeled by creating two associated peer classes: Class A and B. The associated classes must have a [0-1] relationship on at least one end. For example, in Figure 3.3 part 3, Class B is an optional class with Class A. Class A does not assume that Class B exists, and therefore can be reused whether or not Class B is reused. The option pattern can also be applied to "related-to" and "aggregation" associations [Keepence & Mannion, 1999].

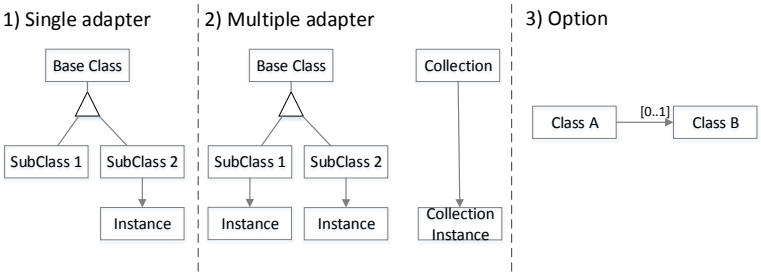


Figure 3.3: Single, multiple, and option patterns

The single, multiple, and option patterns have been put into practice for modeling product families in the spacecraft mission-planning domain. The resulting family encapsulates all the variability of the domain without restricting its family [Keepence & Mannion, 1999]. In addition, authors guaranteed that using the patterns simplifies the process of building complex models that support variability. However, they also claimed that quantifying the costs of applying the patterns is difficult since it depends on more than one technique.

3.2.2 Patterns for evolving event-based systems

In general, the implementation of a particular change in an event-based system involves executing relevant actions (e.g., adding or removing components, enabling or disabling components, or altering the components' inputs or outputs) while taking into account the consequences of these actions (e.g., other components might be affected by these actions) [Cleland-Huang et al., 2003]. The use of patterns to deal with changes in event-based systems is investigated in [Tragatschnig et al., 2013, 2014]. In an event-based system, an *actor* (i.e., component) is totally unaware of the others and is indirectly triggered by particular events emitted by other components, which leads to a high degree of flexibility. In the context of event-based systems, there are no prescribed execution descriptions and the constituent actors and their relationships can be arbitrarily changed at any time [Tragatschnig et al., 2014]. Related actors are encapsulated in logical groups named *execution domains*.

In order to deal with the complexity and the large degree of flexibility of event-based systems, system evolution is managed at different levels of abstraction. More precisely, instead of code statements, fundamental abstractions for describing primitive actions are used to modify the system at the low-level. On top of these actions, a set of high-level abstractions are described in terms of change patterns. Figure 3.4 presents these primitive actions and patterns.

A proof-of-concept implementation of these change patterns has been developed. More precisely, authors estimate the necessary effort (in number of statements) for manually implementing a change on an

Primitive action	Description
`add (a)`	It adds the actor *a* to the execution domain
`remove (a)`	It removes the actor *a* from the execution domain
`setTarget (a, d)`	It sets the target of the execution domain *d* for an actor *a*
`set (a,p)`	It sets a new port *p* for the actor *a*
`setDomain (a,d)`	It sets the execution domain *d* for actor *a*
`add (p,events)`	It adds a set of *events* to port *p*
`remove (p,events)`	It removes a set of *events* from port *p*
`replace (p,events)`	It replaces all events of port *p* with another set of *events*
`replace (p,e,e')`	It replaces event *e* of port *p* with event *e'*
Pattern	**Description**
Insert (x)	It adds an actor *x* in the current execution domain
Delete (x)	It removes the actor *x* from the current execution domain
Move (x,y,z)	It moves the actor *x* in a way that the actor *y* will become predecessor and the actor *z* will become successor of *x*, respectively
Replace (x,y)	It substitute the actor *x* by the actor *y*
Swap (x,y)	Given an actor *x* that precedes an actor *y*, this pattern will switch the execution order between *x* and *y*
Parallelize (x,y)	It enables the concurrent execution of two actors *x* and *y* that are performed sequentially before
Migrate (x, d, d')	It migrates an actor *x* from an execution domain *d* to another execution domain *d'*

Figure 3.4: Primitive actions and patterns for evolving event-based systems

event-based system and compare these results to the defined actions and change patterns. Results shows that describing changes using the defined change patterns is about 11% of the effort compared to the effort needed to implement each change manually. Using the change patterns, there are roughly 9 times less statements needed in comparison to perform each change individually [Tragatschnig et al., 2013]. These results demonstrate the benefits of using change patterns.

3.3 Business Process Modeling Patterns

The ability to efficiently deal with process change has been identified as one of the critical success factors for any PAIS [Lenz & Reichert, 2007]. The high complexity and cost of change management are considered as a major concern when modeling business processes [Reichert & Weber,

2012]. To overcome this problem and make PAIS better comparable, two types of patterns have been introduced: workflow patterns [van der Aalst et al., 2003] and patterns for business process change [Reichert & Weber, 2012].

3.3.1 Workflow patterns

Workflow patterns aim at delineating the fundamental requirements that arise during business process modeling on a recurring basis and describe them in an imperative way. In particular, workflow patterns provide a thorough examination of some perspectives that need to be supported by a business process modeling language (i.e., control flow, data, resource, time, and exception handling). In addition, workflow patterns are defined independently of specific workflow technologies and modeling languages. Thus, they are intended to be used for examining the suitability of a particular process language for a particular project and for assessing relative strengths and weaknesses of various approaches to process specification. In addition, workflow patterns help in implementing certain business requirements in a particular PAIS and serve as a basis for language and tool development [van der Aalst et al., 2003]. In the following, we describe existing workflow patterns for each process perspective.

Control flow patterns characterize the range of control flow constructs that might be encountered when modeling and analyzing workflows [van der Aalst et al., 2003]. There exist forty control flow patterns classified in eight groups (cf. Figure 3.5).

- Patterns for *basic control flow*. They describe elementary aspects to specify activities and their ordering.
- Patterns for *advanced branching and synchronization*. They characterize more complex branching and merging concepts which arise in business processes. Although relatively commonplace in practice, these patterns are often not directly supported or even able to be represented in many commercial offerings.

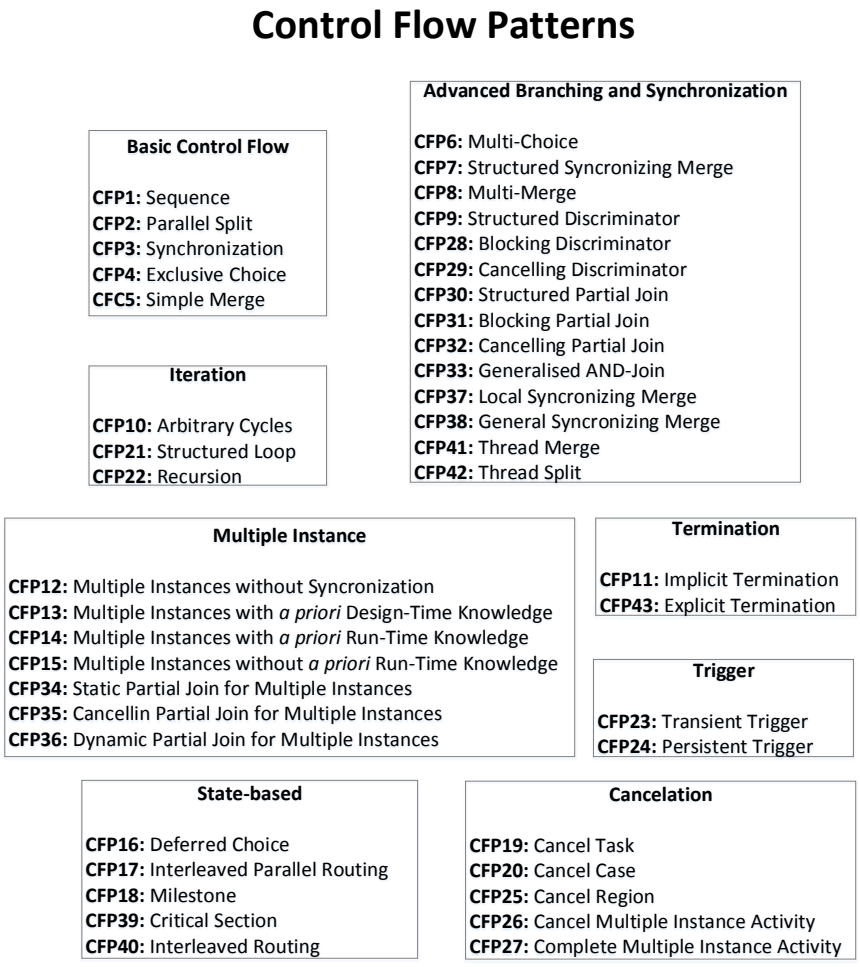


Figure 3.5: Overview of existing control flow patterns

- Patterns for *iteration*. They deal with capturing repetitive behavior in a workflow (e.g., loops, cycles). These patterns are divided

in *internal* and *external* interactions depending on the elements that interact in the workflow.

- Patterns for *multiple instances*. They describe situations where there are multiple threads of execution active in a process model which relate to the same activity (and hence share the same implementation definition). Multiple instances can arise in three situations: (1) an activity is able to initiate multiple instances of itself, (2) a given activity is initiated multiple times as a consequence of it receiving several independent triggers (e.g. as part of a loop), and (3) a set of activities share the same implementation definition (i.e., overlapping executions).

- Patterns for *termination*. They deal with the circumstances under which a workflow is considered to be completed.

- Patterns for *triggering*. They deal with the external signals that may be required to start certain tasks.

- Patterns *state-based*. They reflect situations for which solutions are most easily accomplished in process languages that support the notion of *state*. In this context, this state includes the broad collection of data associated with the execution of each process instance, including the status of various activities as well as process-relevant working data (e.g., activity and instance data elements).

- Patterns for *cancellation*. They deal with the concept of activity cancellation where enabled or active instances are withdrawn.

Workflow data patterns aim to capture the various ways in which data is represented and utilized in workflows [Russell et al., 2004a]. Workflow data patterns are classified in four groups referred to how data is characterized in workflows (cf. Figure 3.6).

- Patterns for *data visibility*. They relate to the extent and manner in which data elements can be viewed by various components of a workflow process.

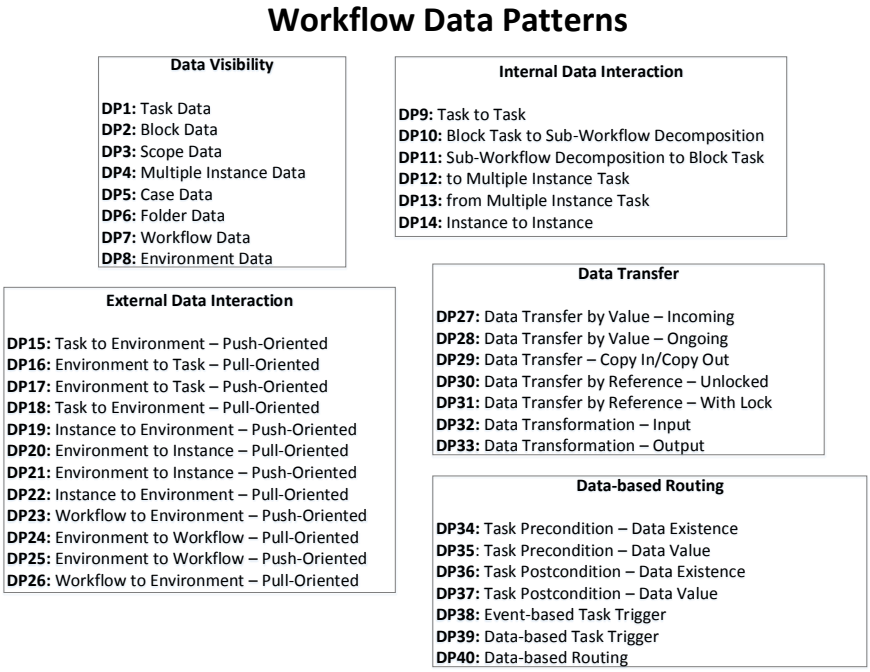


Figure 3.6: Overview of existing workflow data patterns

- Patterns for *data interaction*. They focus on the manner in which data is communicated between active elements within a workflow.

- Patterns for *transfer data*. They consider the means by which the actual transfer of data elements occurs between workflow components and describe the various mechanisms by which data elements can be passed across the interface of a workflow component.

- Patterns for *routing data*. They characterize the manner in which data elements can influence the operation of other aspects of the workflow, particularly the control flow perspective.

Workflow resource patterns describe how resources can be represented and utilized in workflows [Russell et al., 2004b]. A resource is considered an entity capable of doing work. This is usually assigned to the resource in the form of *work items*, each of which describe an integral unit of work that the resource should undertake. A resource is classified as either *human* or *non-human* (e.g., equipment). A human resource is typically a member of an *organization* that usually has a specific *position* in this organization. As a consequence, resources may have a number of associated *privileges*, which determine what they can actually do. Workflow resource patterns are classified in seven groups referred to how resources are presented in workflows (cf. Figure 3.7).

- Patterns for *creation*. They correspond to limitations on the manner in which a work item may be executed. They are specified at design time, usually in relation to a task, and serve to restrict the range of resources that can undertake work items that correspond to the task. They also influence the manner in which a work item can be matched with a resource that is capable of undertaking it. For all of these patterns it is assumed that there is an associated *organizational model* which allows resources to be uniquely identified and that there is a mechanism to distribute work items to specific resources identified in the organizational model. As creation patterns are specified at design time, they usually form part of the process model which describes a business process.

- Patterns for *pushing*. They characterize situations where newly created work items are proactively offered or allocated to resources by the system. These may occur indirectly by advertising work items to selected resources via a shared work list or directly with work items being allocated to specific resources. In both situations however, it is the system that takes the initiative and causes the distribution process to occur.

- Patterns for *pulling*. They correspond to the situation where individual resources are made aware of specific work items, which require execution, either via a direct offer from the system or indirectly through a shared work list. The commitment to undertake

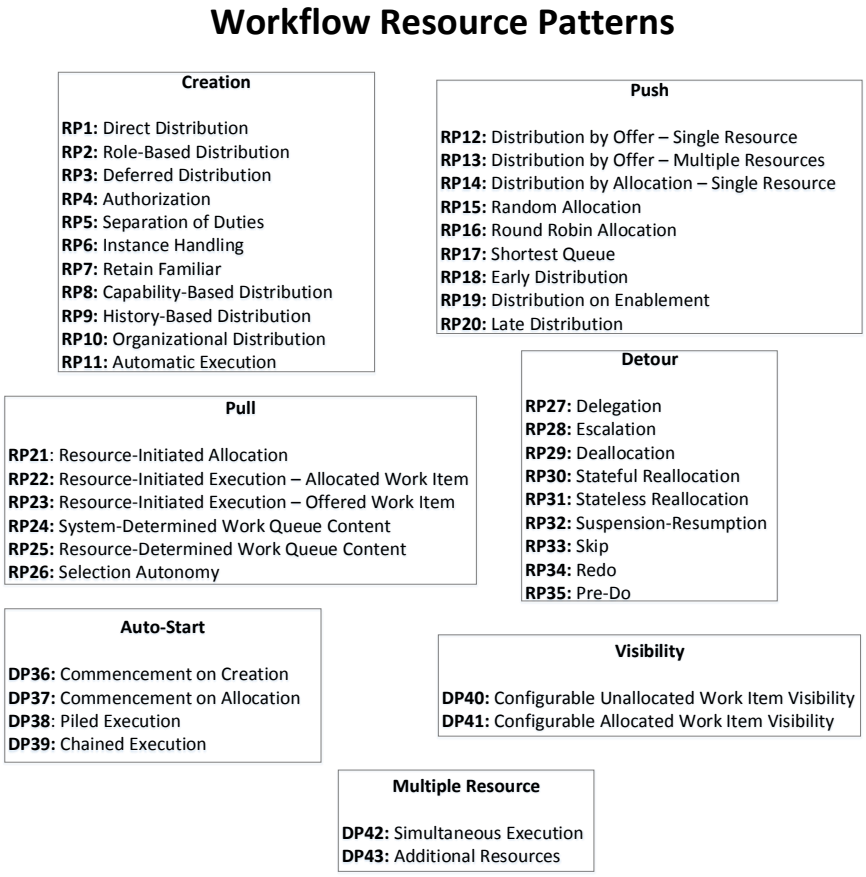


Figure 3.7: Overview of existing workflow resource patterns

a specific task is initiated by the resource itself rather than the system.[1] Generally this results in the work item being placed on the specific work list for the individual resource for later execu-

[1]Note that the distinction between push and pull patterns is identified by who allocates the work items.

tion although in some cases, the resource may elect to commence execution on the work item immediately.

- Patterns for *detouring*. They refer to situations where work item distributions that have been made for resources are interrupted either by the system or at the instigation of the resource. As a consequence of this event, the normal sequence of state transitions for a work item is varied. There is a number of possible impacts on a work item, depending on its current state of progression and whether the detour was initiated by the resource with which the work item was associated or by the system. Detouring patterns are defined for each one of these impacts.

- Patterns for *auto-start*. They relate to situations where execution of work items is triggered by specific events in the lifecycle of the work item or the related process definition. Such events may include the creation or allocation of the work item, completion of another instance of the same work item or a work item that immediately precedes the one in question.

- Patterns for *visualization*. They classify the various scopes in which work item availability and commitment are able to be viewed by resources.

- Patterns for *multiple resources*. They focus on many-to-many correspondences between the resources and work items in a given allocation execution (i.e., multiple resources working on the same work item). Unfortunately, contemporary management systems (e.g., ADEPT2 [Reichert et al., 2005]) do not properly support these patterns [Russell et al., 2004b]. This is a pity since for more complicated activities people tend to work in teams and collaborate to jointly execute work items. Moreover, there is also a lack of consideration for work items that require access to multiple non-human resources (e.g. plant and equipment, fuel, and consumables) in order to proceed.

Time patterns constitute solutions for representing commonly occurring temporal constraints. Time patterns are classified in four groups

based on their semantics (cf. Figure 3.8).

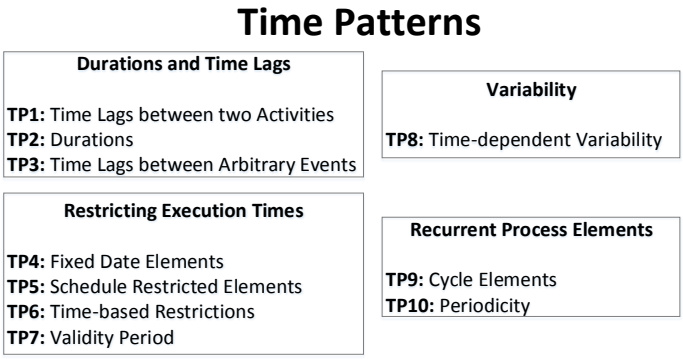


Figure 3.8: Overview of existing time patterns

- Patterns for *time duration and lags*. They provide support for expressing durations of different granularities (i.e., activities, activity sets, processes, or sets of process instances) as well as time lags between activities or–more generally– between process events (e.g., milestones).
- Patterns for *restricting execution times*. They allow specifying constraints regarding possible execution times of single activities or entire processes (e.g., activity deadlines).
- Patterns for *time variations*. They provide support for expressing time-based variability during process execution (e.g., varying control flow depending on temporal aspects).
- Patterns for *recurrent process elements*. They express temporal constraints in connection with recurrent activities or process fragments (e.g., cyclic flows and periodicity).

Exception handling patterns are used for changing the state of a process instance (i.e., its behavior) and its related elements (e.g., work

items) [Russell et al., 2006]. Exception handling patterns are classified in two groups depending on the level on which the exception is produced (cf. Figure 3.9).

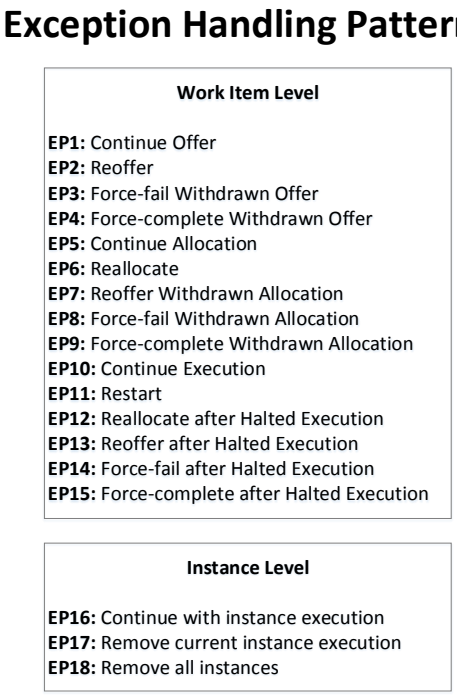


Figure 3.9: Overview of existing exception handling patterns

- Patterns for *handling exceptions at work item level*. They focus on dealing with exceptions in the assignment of resources to the work items. There is a multitude of ways in which these exceptions can be handled although the specific details will depend on the current state of execution of the work item (e.g., fail, complete).
- Patterns for *handling exceptions at instance level*. They allow dealing with exceptions that occur in the context of the execution

of a process instance. They define how the instance should be managed in an overall sense, particularly in regard to other work items that may currently be executing or will run at some future time.

3.3.2 Patterns for business process change

In order to create a process model, different approaches can be pursued. One involves the use of change primitives. Change primitives are edit operations that work on single process model elements (e.g., `add node`, `add edge`, `move node`, `remove node`, `remove edge`) [Reichert & Weber, 2012]. The disadvantage of designing process models with change primitives is, that soundness and data flow correctness cannot be guaranteed for models with three or more activities and has to be checked after model generation. Soundness is given if the model does not contain any dead activities and the option for a proper completion is guaranteed [Reichert & Weber, 2012].

Another approach for model creation comprises the application of *change patterns* [Weber et al., 2008]. Change patterns are high-level change operations, which impose pre- and post-conditions to guarantee the development of a sound and correct process model. These pre- and postconditions are particularly necessary upon conducting ad-hoc changes by end-users and become even more important if software agents execute these changes automatically during enactment time. Change patterns are derived from a set of change primitives, providing a higher level of abstraction [Reichert & Weber, 2012]. They combine a set of change primitives and offer correctness-by-construction [Weber et al., 2008], which ensures, that only certain high-level changes can be applied to a process model, upholding a sound and correct state. Change patterns for model creation are divided into two major categories: *adaptation patterns* and *patterns for changes in predefined regions*. Thereby, adaptation patterns support structural process adaptations, whereas patterns for changes in predefined regions allow for built-in flexibility. In the following, we describe each category of patterns in detail.

Adaptation patterns allow users to structurally modify a process schema at the type or instance level by using high-level change operations (e.g., to add an activity in parallel to another one) instead of low-level change primitives (e.g., to add a single node or to delete a single control flow edge). The use of a set of pre- and post-conditions with these high-level operations allows to guarantee soundness when applying the respective operations [Weber et al., 2008; Dadam & Reichert, 2009]. They can be applied along the entire process lifecycle (i.e., process analysis, design, configuration, enactment, diagnosis, and evolution) and do not have to be pre-planned, i.e., the region to which adaptation patterns may be applied can be chosen dynamically. Hence, adaptation patterns are well suited for realizing process changes at both design and enactment time. Like design patterns in software engineering, change patterns aim at reducing complexity by raising the level of abstraction for expressing changes [Gamma et al., 1995]. Generally, adaptation patterns can be applied to the whole process schema, i.e., the region to which the adaptation pattern is applied can be chosen dynamically. Therefore, adaptation patterns are well suited for dealing with exceptions or for coping with the evolving nature of business processes [Weber et al., 2008]. However, note that when working with adaptation patterns, process models need to be block-structured. A process model features a block-structure [Reichert & Weber, 2012], if it is composed of blocks (single-entry single-exit fragments), which can be nested, but must not overlap. A total of 14 adaptation patterns are classified in five groups (cf. Figure 3.10):

- Patterns for *adding/deleting fragments*. These allow for the insertion (AP1) and deletion (AP2) of process fragments at a specific point in a given process schema.
- Patterns for *moving/replacing fragments* is supported by adaptation patterns AP3 (Move Process Fragment), AP4 (Replace Process Fragment), AP5 (Swap Process Fragment), and AP14 (Copy Process Fragment).
- Patterns for *adding/deleting process levels*. These allow for adding or removing levels of hierarchy (i.e., subprocesses). Thereby, the

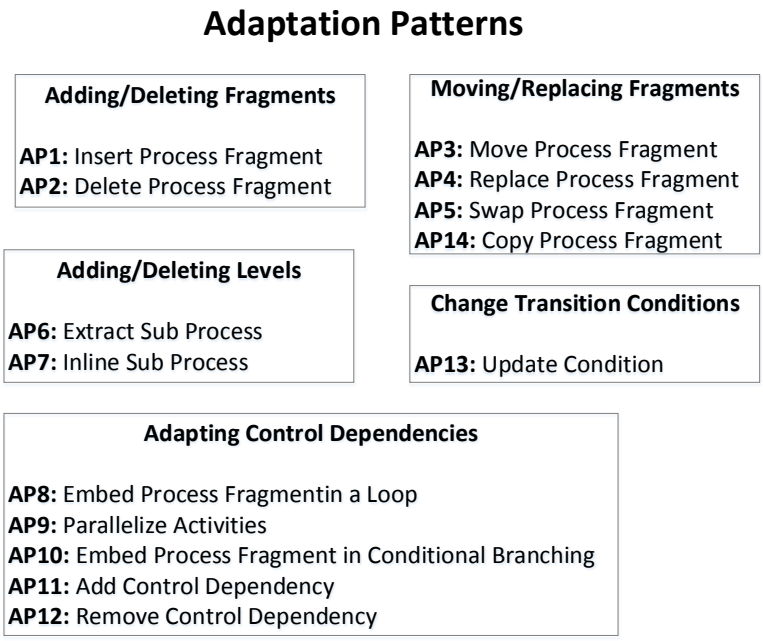


Figure 3.10: Overview of existing adaptation patterns

extraction of a sub process from a process schema is supported by AP6, whereas the inclusion of a sub process into a process schema is supported by AP7.

- Patterns for *adapting control dependencies*. These refer to 5 patterns: embed an existing process fragment in a loop (AP8), parallelize a process fragment (AP9), embed an existing process fragment in a conditional branch (AP10), and add/remove control dependencies (AP11, AP12).

- Patterns for *change transition conditions*. These refer to AP13 (Update condition) which allows changing transition conditions in logical decisions (e.g., XOR gateways).

Two general design choices are valid for the 14 adaptation patterns: (1) be applied at the process type and/or process instance level and (2) be applied on a process fragment (e.g., an atomic activity, an encapsulated sub process, or a hammock). If an adaptation pattern is supported at the process type level, the graphical editor of the PAIS should allow users to edit a process schema at design time using the respective pattern. If no pattern support is provided, process schema changes have to be conducted at a low level of abstraction using change primitives [Weber et al., 2008]. If a respective pattern is, in turn, supported at the process instance level, changes of single instances can be accomplished.

Patterns for Changes in Predefined Regions allow for better dealing with uncertainty by deferring decisions regarding the exact control flow to enactment time. Instead of requiring a process model to be fully specified prior to execution, parts of the model can remain unspecified. In contrast to adaptation patterns, whose application is not restricted a priori to a particular process part, patterns for changes in predefined regions define constraints concerning the parts of a process schema that can be changed or expanded. Thus, the application of these patterns has to be anticipated at build-time. This can be accomplished by defining regions in the process schema where potential changes may be performed during enactment time. As process schema changes or process schema expansions can only be applied to these predefined regions, respective patterns are less suited for dealing with arbitrary exceptions [Dadam & Reichert, 2009]. Instead they allow for dealing with situations where, due to uncertainty, decisions cannot be made at build-time, but have to be deferred to enactment time. Figure 3.11 summarizes existing patterns for changes in predefined regions.

There exist four patterns for changes in predefined regions. These four patterns differ regarding the parts that can remain unspecified resulting in a different degree of freedom during enactment time.

- *Late Selection* (PP1). It allows deferring the selection of the implementation of a particular activity to enactment time. Prior to execution only a placeholder activity has to be provided, the concrete implementation is selected during enactment time either

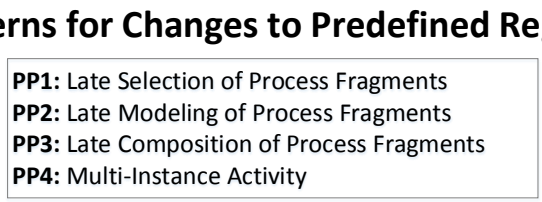


Figure 3.11: Overview of existing patterns for changes in predefined regions

based on predefined rules or on user decisions.

- *Late Modeling* (PP2). It offers more freedom and allows for modeling selected parts of the process schema at enactment time. Prior to execution only a placeholder activity has to be provided, its implementation is model during enactment time.

- *Late Composition of Process Fragments* (PP3). It enables the *on-the-fly composition* of process fragments from the process repository (e.g., by dynamically introducing control dependencies between a predefined set of fragments). There is no predefined plan, but the process instance is created in an ad-hoc way by selecting from the available activities in the repository. In addition, constraints may be defined, which have to be considered while composing a process fragment.

- *Multi-Instance Activity* (PP4). It allows for deferring the decision on how often a specific activity should be executed during enactment time. PP4 not only constitutes a change pattern, but a workflow pattern as well since it allows for the creation of multiple activity instances during enactment time [van der Aalst et al., 2003]. The decision of how many instances are created can be based either on knowledge available at design time or on some knowledge gained at enactment time (cf. Section 2.1).

Table 3.1 summarizes the main properties of the two major pattern categories for dealing with business processes.

	Adaptation patterns	**Patterns in changes to predefined regions**
Structural process change	yes	no
Anticipation of change	no	yes
Change restricted to predefined region	no	yes
Application area	Unanticipated exceptions, unforeseen situations	Address uncertainty by deferring decisions to enactment time

Table 3.1: Summary of the change patterns for business processes

3.4 Discussion

In this chapter we have described the most relevant approaches closely related to the work of this thesis. First, we have analyzed existing approaches that deal with process variability modeling. Using a heterogeneous set of techniques (e.g., configurable nodes), there exist plenty of process variability approaches that allow creating and managing configurable process models. However, these approaches are primarily focused on providing different formats for representing configurable processes (e.g., represent the concept of configurable node), instead of systematizing how to create and evolve configurable process models using the approaches (e.g., methodology). With the current process variability approaches, PAIS engineers usually are required to manually model and manage all the elements of a configurable process model one by

one and ensure its correctness by their own. This can be both a tedious and error-prone task especially when a configurable process model represents a process family comprising a high number of process variants. For example, PAIS engineers need to be aware of each variation and dependence of each process variant. In this thesis, on top of these approaches, we define a set of change patterns that enable the modeling and management of configurable process models at a higher level of abstraction than the one provided by existing process variability approaches. Used in combination with any process variability approach, our set of patterns are intended to speed up the modeling process and reduce the effort needed for such purpose. Even though the multiple definitions of the term 'pattern' (cf. Section 2.3), we consider this thesis as a pattern-based approach since we provide a reusable solution for modeling process families. In particular, our change patterns can be classified as design patterns (cf. Section 2.3.5) since they can be used to solve common problems when defining a process family. In addition, since they are based on the results obtained from a systematic literature review, our patterns are empirically grounded.

Regarding other use of patterns, in this chapter we have also analyzed how patterns have been applied for modeling software product families. In the analyzed approaches, patterns have been proved as a feasible approach to reduce modeling efforts. This results coincide with the benefits that we envision for our set of patterns. Further, we have examined workflow patterns that make PAISs better comparable. Respective patterns provide means for analyzing the expressiveness of process modeling tools and languages in respect to different workflow perspectives (e.g., resources or data). In the same vein, we have revised patterns for dealing with changes in business process models (i.e., adaptation patterns and patterns for better dealing with process uncertainty). Although all these patterns are well suited for product families and single process models, they are not sufficient to cope with the complexity that process variability introduces. Thus, our set of patterns complement existing work since we cover variability-specific needs for process families.

Furthermore, as well as existing patterns in the context of busi-

ness processes (i.e., workflow patterns and patterns dealing with process model changes) can be used for PAIS comparison, there is a lack of methods for comparing PAIS including process variability. This entails that PAIS engineers should select the proper process variability approach based on their previous experience and knowledge. In this thesis, we also fill this gap by providing a broad characterization of process variability. We perform an in-depth and systematic study of approaches enabling process variability in order to provide a clear picture of process variability and identify the main aspects of existing process variability approaches. Then, based on this empirical evidence we derive a framework named *VIVACE*, which shall allow for the systematic assessment and comparison of existing process variability approaches.

3.5 Conclusions

This chapter presents the most relevant approaches closely related to the work of this thesis. As discussed previously, the thesis complements these approaches. In the following, we describe in detail our in-depth and systematic study of the process variability domain. This study help us to provide a broad characterization of process variability.

4

VIVACE: Process Variability Characterization

The modeling of process variability is a way of capturing common process knowledge and reusing it in terms of configurable process models. For creating such models, several approaches have been defined (cf. Section 3.1). However, with these approaches, PAIS engineers are required to manually model and manage all the elements of a configurable process model one by one, which can be tedious and error-prone especially with large process families (e.g., thousands process variants). For example, PAIS engineers need to be aware of each variation and dependence of each process variant. Thus, more efficient methods that allow PAIS engineers to model process variability at a level of abstraction higher than the one provided by the existing process variability approaches are needed.

In this context, the use of modeling patterns [Weber et al., 2008]

is a promising solution. However, a set of patterns specifically tailored for modeling process variability cannot be defined arbitrarily. How process variability is represented in existing approaches becomes critical for such definition. That means, for example, to identify what language constructs are used to capture variability in a configurable process model. However, although several attempts to describe and characterize process variability modeling have been made (e.g., [Mechrez & Reinhartz-Berger, 2014; Aiello et al., 2010]), none of them identify these constructs.

In this chapter, we deal with this issue by studying in-depth the process variability domain. For such purpose, we conducted a systematic study to analyze existing process variability approaches regarding their expressiveness with respect to process variability modeling as well as their process support. This study was performed as a *systematic literature review* over the process variability domain. Thus, we could identify the specific language constructs used to represent process variability. In addition, in this study we decided to consider all the phases of the process lifecycle (cf. Section 2.1) since process variability modeling, as described in previous chapters, is involved in several phases such as analysis and design phase, enactment, and evolution (cf. Sections 2.1 and 3.1). As a result, apart from the language constructs, we are able to derive a complete characterization of process variability along the process lifecycle. This characterization is aggregated in the *VIVACE* framework. In particular, VIVACE can support PAIS engineers in (1) defining new process variability approaches, (2) improving their communication, (3) evaluating existing process management technologies enabling process variability, (4) selecting which of the approaches meets PAIS engineers' requirements best, and (5) dealing with (e.g., modeling, implementing) a PAIS that will effectively support variability along the process lifecycle.

More concretely, this chapter presents VIVACE and the essential aspects of the systematic study that allow understanding how it was obtained. For a complete description of the performed study, we refer to Appendix B. The rest of the chapter is organized as follows. Section 4.1 describes the research questions we defined in order to characterize the

process variability domain. Section 4.2 presents the VIVACE framework and describes each of its aspects in detail. Section 4.3 illustrates the way VIVACE can be applied in practice. Section 4.5 compares VIVACE with other process variability characterizations. Finally, Section 4.6 concludes the chapter.

4.1 Research Questions Formulation

A systematic study (in terms of a systematic literature review) is a means of identifying, evaluating, and interpreting relevant data in a specific area through a replicable, scientific, and transparent approach, which reduces the probability of any bias [Kitchenham & Charters, 2007]. To conduct such a study with respect to process variability, we designed a protocol following the guidelines, procedures, and policies proposed by *Kitchenham* in [Kitchenham & Charters, 2007]. According to the latter, this protocol described the formulation of the research questions, the search string, the data sources chosen for performing the search, the identification of inclusion and exclusion criteria, the quality assessment questions, the selection of studies[1], the method for extracting the data from the selected studies, and the way how the obtained data shall be analyzed (cf. Appendix B).

The overall goal of our systematic study was to analyze relevant papers regarding their expressiveness for modeling process variability and their support for handling process variability along the process lifecycle. To perform such analysis we investigated the following six issues, which allows us to better classify and characterize each relevant paper. First of all, since there exists no standard language for modeling process variability, we were interested in identifying what **process modeling languages** have been used for this purpose. Second, as literature refers to various **techniques for creating configurable process models** [Ayora et al., 2012a], we were interested in providing an overview of the way these techniques are used. Third, in order to allow assessing the expressiveness of existing approaches for modeling process variability

[1] In the given context, a *study* refers to a retrieved paper.

(and serve as a basis for our change patterns), we wanted to identify a core set of **variability-specific language constructs** frequently used by these approaches. Fourth, since variability may concern different **process perspectives**, we wanted to provide insights into the perspectives covered by existing process variability approaches. Fifth, in order to assess the practical applicability of existing process variability approaches, we were interested in identifying the available **tools** supporting these approaches. Sixth, we wanted to create an in-depth understanding of **variability support features** (e.g., to verify and validate process variants) that foster process variability along the different phases of the process lifecycle. Seventh, to assess the level of maturity of existing process variability approaches, we further investigated whether and–if so–how these approaches have been **empirically evaluated**. Finally, we analyzed the **domains** in which existing process variability approaches have been applied. In this context, we considered the following research questions:

- **RQ1.** What underlying *business process modeling languages* are used for modeling process variability?
- **RQ2.** Which *techniques* are used for representing process variability in a configurable process model[2]?
- **RQ3.** What *language constructs* are provided for representing process variability in a configurable process model?
- **RQ4.** Which *process perspectives* are covered by languages that enable the modeling of process variability?
- **RQ5.** What *tools* exist for enabling process variability?
- **RQ6.** What *variability support features* are provided for fostering process variability in all phases of the process lifecycle?
- **RQ7.** Have existing process variability approaches been *evaluated*? If so, how does this evaluation look like?

[2]Remember that related process variants are defined in terms of a configurable process model, which then represents a complete process family.

- **RQ8.** In which domains have existing process variability approaches been applied?

To answer these questions, we subjectively elaborated a search string using keywords we derived based on our in-depth knowledge of the topic and taking the defined research questions into account. This string was applied to relevant data sources to find studies related to the topic (i.e., process variability). These queries resulted in a total of 4947 studies, which were filtered based on a set of inclusion/exclusion criteria and a set of questions to assess their quality. Overall, this resulted in 63 *primary studies*, which are summarized in Table 4.1. Each of these studies is associated with a unique identifier (i.e., Study ID), which is used in the following to refer to the respective studies.

Study ID	Study ID
S1-Alférez et al. [Alférez et al., 2014]	S33-Czarnecki et al. [Czarnecki & Antkiewicz, 2005]
S2-Bucchiarone et al. [Bucchiarone et al., 2013]	S34-Becker et al. [Becker et al., 2004]
S3-Kumar et al. [Kumar & Yao, 2012]	S35-van der Aalst et al. [van der Aalst et al., 2012]
S4-Frece et al. [Frece & Juric, 2012]	S36-Li et al. [Li et al., 2011]
S5-Santos et al. [Santos et al., 2012]	S37-Weber et al. [Weber et al., 2011]
S6-W. Yao et al. [Yao et al., 2012]	S38-Derguech et al. [Derguech & Bhiri, 2011]
S7-Q. Yao et al. [Yao & Sun, 2012]	S39-Yahya et al. [Yahya & Bae, 2011]
S8-Ognjanovic et al. [Ognjanovic et al., 2012]	S40-Koetter et al. [Koetter et al., 2011]
S9-Gröner et al. [Gröner et al., 2012]	S41-Gröner et al. [Gröner et al., 2011]
S10-Boffoli et al. [Boffoli et al., 2012]	S42-van der Aalst et al. [van der Aalst et al., 2010a]
S11-Schunselaar et al. [Schunselaar et al., 2012]	S43-La Rosa et al. [La Rosa et al., 2010]
S12-Groefsema et al. [Groefsema et al., 2011]	S44-Mahmod et al. [Mahmod & Chiew, 2010]
S13-Döhring et al. [Döhring et al., 2011]	S45-Gottschalk et al. [Gottschalk et al., 2008]
S14-Park et al. [Park & Yeom, 2011]	S46-Thomas et al. [Thomas, 2008]
S15-Nguyen et al. [Nguyen et al., 2011]	S47-Koschmider et al. [Koschmider & Oberweis, 2007]
S16-Pascalau et al. [Sakr et al., 2011]	S48-Mendling et al. [Mendling et al., 2006]
S17-Meerkamm et al. [Meerkamm & Jablonski, 2011]	S49-Recker et al. [Recker et al., 2006]
S18-Derguech et al. [Derguech et al., 2010]	S50-Reinhartz-Berger et al. [Reinhartz-Berger et al., 2005]

S19-Hallerbach et al. [Hallerbach et al., 2010b]	S51-Döhring et al. [Döhring et al., 2014]
S20-de la Vara et al. [de la Vara et al., 2010]	S52-Derguech et al. [Derguech et al., 2012]
S21-Reinhartz-Berger et al. [Reinhartz-Berger et al., 2010]	S53-Lönn et al. [Lönn et al., 2012]
S22-Acher et al. [Acher et al., 2010]	S54-Bulanov et al. [Bulanov et al., 2011]
S23-Reijers et al. [Reijers et al., 2009]	S55-Vogelaar et al. [Vogelaar et al., 2011]
S24-La Rosa et al. [La Rosa et al., 2009b]	S56-Reinhartz-Berger et al. [Reinhartz-Berger & Sturm, 2012]
S25-La Rosa et al. [La Rosa et al., 2011]	S57-Scherer et al. [Scherer & Sharmak, 2011]
S26-Montero et al. [Montero et al., 2008]	S58-Pascalau et al. [Pascalau & Rath, 2010]
S27-Moon et al. [Moon et al., 2008]	S59-Baier et al. [Baier et al., 2010]
S28-Gottschalk et al. [Gottschalk et al., 2007]	S60-Gottschalk et al. [Gottschalk et al., 2009]
S29-Lapouchnian et al. [Lapouchnian et al., 2007]	S61-La Rosa et al. [La Rosa & Mendling, 2009]
S30-Schnieders et al. [Schnieders & Puhlmann, 2007]	S62-Schnieders et al. [Schnieders & Weske, 2007]
S31-Lazovik et al. [Lazovik & Ludwig, 2007]	S63-Giese et al. [Giese et al., 2007]
S32-Lu et al. [Lu et al., 2009]	

Table 4.1: Final list of primary studies

During the selection process, we organized these 63 primary studies in three groups:

1. Studies describing process variability approaches: S1 - S34.

2. Studies describing process variability support features: S35 - S50.

3. Studies describing solely empirical evaluations of process variability approaches: S51 - S63.

Each of the 63 primary studies was deeply analyzed with the goal to answer the defined research questions. The obtained results are aggregated in the *VIVACE* framework.

4.2 The VIVACE Framework

This section presents the *VIVACE* framework. The latter aggregates the results we gathered in the context of the defined research questions (cf. Section 4.1). Hence VIVACE draws a complete characterization of process variability support. It refers to (1) the process variability modeling language, (2) the techniques provided for building a configurable process model, (3) the process perspectives covered, (4) the variability-specific language constructs, (5) the features supporting process variability in the different phases of the lifecycle, (6) tools implementing process variability approaches, (7) empirical evaluations performed, and (8) their application domains (cf. Figure 4.1).

In the following, we present each of the aspects of VIVACE separately.

4.2.1 Languages for Modeling Business Process Variability

We first present the analysis related to RQ1 (*What underlying business process modeling languages are used for modeling process variability?*). In order to answer this research question, we analyzed the group of studies describing process variability approaches (i.e., S1-S34). In particular, these studies referred to the expressiveness of existing approaches with respect to the modeling of process variability. In this section, we focus on the languages they use as basis for modeling process variability.

Figure 4.2 shows the distribution of the 34 studies according to the modeling languages they use for representing both the *commonalities* (i.e., process fragments shared by all process variants) and *variations* of the members of a process family (i.e., the process variants).

As can be seen, 17 studies are conceived to be independent of a particular process modeling language; i.e., S1, S3, S5, S9, S10, S15, S18, S19, S20, S22, S24, and S28-S33. For example, these studies propose the use of feature models, ontologies, rules, or hierarchical indexing structures in order to capture and model process variability. In turn, respective approaches can be used in combination with any process

The *VIVACE* framework	
Modeling language used to represent process variability	
Technique used for building the configurable process model	
Method for modeling the process family	
Process perspectives covered	
Variability-specific language constructs	LC1 Configurable Region
	LC2 Configuration Alternative
	LC3 Configuration Context Condition
	LC4 Configuration Constraint
	LC5 Configurable Region Resolution Time
Variability support features	**Analysis & Design phase**
	F1.1 Modeling a configurable process model
	F1.2 Verifying a configurable process model and its related process family
	F1.3 Validating a configurable process model
	F1.4 Evaluating the similarity of different process variants
	F1.5 Merging process variants
	Configuration phase
	F2 Configuring specific regions of a process variant out of a configurable process model
	Enactment phase
	F3.1 Configuring specific regions of a process variant at enactment time
	F3.2 Dynamically re-configuring an instance of a process variant at enactment time
	Diagnosis
	F4 Analyzing a collection of process variants
	Evolution
	F5.1 Versioning of a configurable process model
	F5.2 Propagating changes of a configurable process model to already configured process variants
Tool implementation	
Empirical evaluation	
Application domain	

Figure 4.1: The VIVACE framework

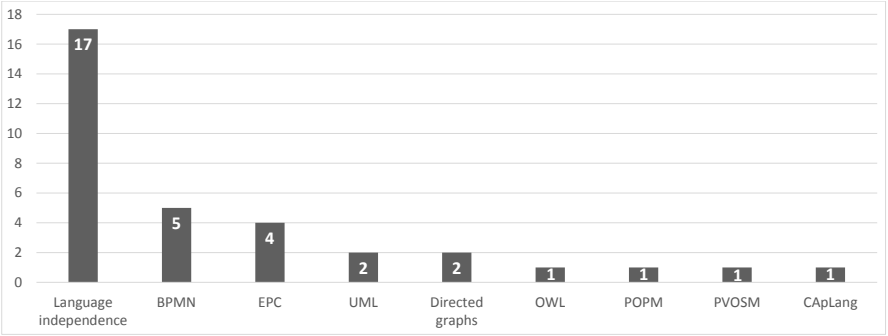


Figure 4.2: Distribution of studies S1-S34 according to the process modeling language used

modeling language (e.g., BPMN, EPC, or UML Activity Diagrams) for properly representing process variability.

On the contrary, the other 17 studies propose approaches that extend existing (process) modeling languages with specific constructs for modeling process variability, or that design proprietary languages for this purpose. In particular, 11 studies propose conceptual extensions of existing process modeling languages such as BPMN (i.e., S4, S8, S13, S16, S26), EPC (i.e., S21, S23, S25, and S34), and UML Activity Diagram (S14, S27) in order to enable the explicit modeling of process variability. In turn, 6 studies either make use of languages such as Directed Graphs (S11 and S12) or OWL [OWL, 2009] (S6), which are common in other fields, or they propose proprietary languages developed for the modeling of process variability; i.e., CApLang (S2), PVOSM (S7), and POPM (S17).

4.2.2 Techniques for Modeling Process Variability in a Configurable Process Model

We now consider RQ2 (*Which techniques are used for representing process variability in a configurable process model?*). We identified two

techniques that may be used to model process variability (cf. Figure 4.3). In particular, these techniques either allow capturing the entire process family (i.e., all process variants) in a single model artifact (i.e., *single artifact technique*) or in a set of related model artifacts (i.e., *multi-artifact technique*). The latter may represent different aspects of the process family, e.g., commonalities of the process variants, variant-specific parts, configuration constraints, and application context. In order to answer RQ2, again we analyze the 34 studies describing process variability approaches (i.e., S1-S34). In particular, these studies refer to the expressiveness of existing approaches regarding the modeling of process variability.

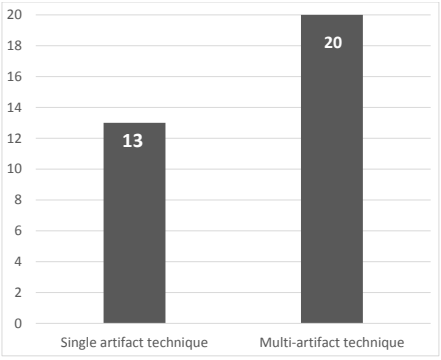


Figure 4.3: Distribution of studies S1-S34 according to the process variability modeling technique used

The *single artifact technique* has been realized by various studies based on different methods (cf. Figure 4.4). The latter include *hiding & blocking* (S11), *configurable nodes* (S7, S25, and S28), and *logic formulae* (S12). Furthermore, *annotations* for BPMN (S4 and S13), *labels* for EPC (S23 and S34), and *meta-model extensions* for UML Activity Diagrams (S27) and BPEL (S31) have been proposed in order to realize configurable process models. Finally, *multiplicity indicators* (S21) and a *hierarchical indexing structure* (S18) constitute two specific methods for representing a configurable process model in terms of a single arti-

fact. Note that all these methods enrich the configurable process model with additional information (e.g., *configuration constraints*) in order to guide users when deriving process variants.

In turn, the *multi-artifact technique* has been realized in the following studies: S1, S3, S5, S6, S8, S9, S10, S14-S17, S19, S20, S22, S24, S26, S29, S30, S32, and S33. Basically, approaches using this technique represent a process family in terms of four different modeling artifacts. The latter include a *base model*, a set of *variable process fragments*, *rules* for adapting the base model through adding/deleting the variable process fragments, and an *application context* determining when these rules apply. Thereby, the *base model* is specified using a particular business process modeling language (e.g., BPMN). However, different policies may be applied when defining a base process model, e.g., setting the latter to the most frequently used process variant or to the process model having minimum average edit distance to the process variants of the process family [Li et al., 2011].

Concerning the three other artifacts (i.e., variable process fragments, rules to adapt the base model, and application context), different methods for defining them exist (cf. Figure 4.4). In turn, these methods are based on specific techniques from various fields (e.g., software product lines, semantic web, and requirements engineering), or they are explicitly designed for the process variability approach at hand. For representing variable process fragments, for example, *features models*, as known from software product lines, can be used (cf. studies S1, S8, S9, S10, S14, S15, S22, S26, S30, and S33). In turn, in the requirements engineering field, S29 refers to *goal models* that may be applied to represent variability at a high level of abstraction. Finally, variable process fragments may be defined based on a set of *process model components* (S16), a *variant list* (S17), or a set of pre-specified *change operations* (S19 and S32).

In turn, the rules for adapting the base model may rely on methods such as *business rules* (S3) and *process model queries* (S16). The approach described by study S29, for example, uses *non-functional constraints* for deriving process variants.

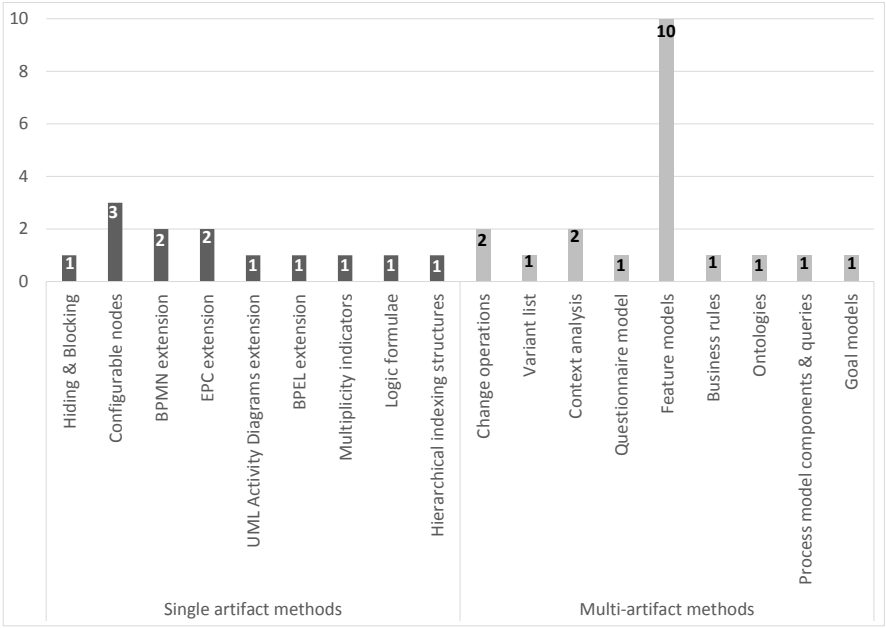


Figure 4.4: Distribution of studies S1-S34 according to the single and multi-artifact method used

Finally, for defining an application context, studies S1 and S6 use *ontologies* (described in the OWL language) and *semantic rules*. In turn, S24 uses a questionnaire model for defining the application context of each process variant. Finally, S5 and S20 redefine a *context analysis method* from a business process perspective in order to analyze context properties.

Note that study S2 has not been considered in the above classification since it describes process variability at the program code level, i.e., S2 does not use process models to represent process variability. Thus, it cannot be classified in any of the described techniques due to the absence of a configurable process model.

4.2.3 Language Constructs for Process Variability

Regarding RQ3 (*What language constructs are provided for representing process variability in a configurable process model?*), we identified five variability-specific *language constructs*: *configurable region, configuration alternative, configuration context condition, configuration constraint,* and *configurable region resolution time*. These constructs abstract from concrete process variability approaches since they are defined at a higher level of abstraction. In the following, we describe the identified variability-specific language constructs and illustrate them along the check-in process (cf. Appendix A). Note that for obtaining these constructs, again we only analyze studies describing process variability approaches (i.e., S1-S34).

- *Configurable Region Language Construct (LC1):* A configurable region corresponds to a region of a configurable process model for which different configuration choices exist, depending on the application context. Studies supporting language construct LC1 include S1-S5, S7-S23, and S25-S34.

  Example 1 (Configurable Region). Regarding the check-in process (cf. Appendix A), activity *Pay extra fee* is only performed if the luggage has overweight. Otherwise, it is skipped. Consequently, at the respective position of the *configurable process model*, there exist two choices depending on the weight of the luggage; i.e., either perform the activity or skip it. Accordingly, the respective position of the *configurable process model* constitutes a *configurable region*.

- *Configuration Alternative Language Construct (LC2):* A configuration alternative corresponds to a particular configuration choice that may be selected in the context of a specific configurable region (LC1). In general, respective alternatives may refer to any process perspective; i.e., the functional, behavioral, organizational, informational, temporal, and operational perspectives (cf. Section 2.1). The studies that support this construct include S1-S5,

S7-S23, and S25-S34. However, note that they do not support configuration alternatives with respect to all process perspectives.

Example 2 (Configuration Alternative). Several *configuration alternatives* exist for the check-in process. Regarding the behavioral perspective, for example, before performing activity *Print boarding card*, each activity that may be performed or skipped constitutes a *configuration alternative*, e.g., activity *Fill in ESTA form*. Concerning the organizational perspective, there exist different roles that may perform the *Print boarding card* activity, i.e., the passenger himself via the web system, the self-servicing machine, or the airline personnel at the check-in counters (i.e., each role constitutes a *configuration alternative*). The *configuration alternatives* related to the informational perspective refer to the different types of boarding cards resulting from the check-in process (e.g., electronic versus paper-based). Finally, *configuration alternatives* of the temporal perspective refer to the start events "23 hours before departure" or "3 hours before departure".

- *Configuration Context Condition Language Construct (LC3):* A configuration context condition defines the conditions under which a particular configuration alternative (LC2) of a configurable region (LC1) shall be selected. Studies that consider this construct include S1, S2, S5, S6, S8, S13-S15, S19, S20, S24, S28, S30, S33, and S34.

  Example 3 (Configuration Context Condition). Before activity *Print boarding card* will be performed in a check-in process (cf. Appendix A), different alternatives exist. For example, activity *Fill in ESTA form* is only performed if the passenger is traveling from EU to US. In this case, the *configuration context condition* "flight destination" determines whether or not this activity will be performed.

- *Configuration Constraint Language Construct (LC4):* A configuration constraint is defined as a restriction regarding the selection of configuration alternatives (LC2). Respective constraints are based on semantic restrictions to ensure the proper use of the defined configuration alternatives (e.g., exclusion or inclusion relationships). The studies supporting this language construct include S2, S5-S10, S12, S14-S17, S19, S21, S22, S24-S29, and S31-S33.

  Example 4 (Configuration Constraint). Regarding the check-in process, activity *Localize assistance to accompany passenger* shall be performed when handicapped passengers check-in. Accordingly, a *configuration constraint* is or needs to be introduced in the *configurable process model* to express that this activity shall be only performed if the passenger is a handicapped person. Otherwise, the activity shall be skipped.

- *Configurable Region Resolution Time Language Construct (LC5):* The configurable region resolution time allows modelers to distinguish between configurable regions (LC1) whose configuration either depends on the initial or the current context of a process instance (i.e., configuration or enactment time). Studies supporting this construct include S15, S28 and S32.

  Example 5 (Configurable Region Resolution Time). Regarding the check-in process, the process variant specifying the online check-in may be configured at configuration time by selecting the activities referring to the *web system* role. However, the activity related to the overweight luggage (i.e., *Pay excess fee*) is only performed if the passenger places the luggage at the desk scales. In this case, the decision whether or not this activity will be performed is postponed to enactment time.

Figure 4.5 summarizes which studies supported which variability-specific constructs. As shown, configurable regions (LC1) and configuration alternatives (LC2) were supported by 32 (out of 34) studies. In turn, configuration context conditions (LC3) were covered by 15 studies,

while 24 studies considered the definition of configuration constraints (LC4). Finally, only 3 studies allowed specifying the configurable region resolution time (LC5).

Interestingly, only 2 studies covered the entire set of the language constructs we identified (i.e., LC1-LC5): S15 and S28. In turn, 7 studies covered four language constructs (e.g., LC1, LC2, LC3, LC4); i.e., S2, S5, S8, S14, S19, S32, and S33. Altogether, Figure 4.5 confirms the high relevance of the five language constructs in respect to the explicit modeling of process families and the variability inherent to them.

		Studies supporting the language construct
Variability-specific language constructs	LC1 Configurable Region	S1-S5, S7-S23, S25-S34
	LC2 Configuration Alternative	S1-S5, S7-S23, S25-S34
	LC3 Configuration Context Condition	S1, S2, S5, S6, S8, S13-S15, S19, S20, S24, S28, S30, S33, S34
	LC4 Configuration Constraint	S2, S5-S10, S12, S14-S17, S19, S21, S22, S24-S29, S31-S33
	LC5 Configurable Region Resolution Time	S15, S28, S32

Figure 4.5: Variability-specific language constructs and studies supporting them

4.2.4 Covered Process Perspectives

This section describes the analysis in respect to research question RQ4, which refers to the process perspectives covered (*Which process perspectives are covered by languages that allow for the modeling of process variability?*). For answering RQ4, we re-analyzed the studies describing process variability approaches (i.e., S1-S34).

As illustrated by Figure 4.6, the most frequent process perspectives covered by existing process variability approaches are the *functional* and *behavioral* ones. As shown, 33 studies considered both process perspectives; i.e., the respective approaches define process variability in respect to the control flow perspective (i.e., S1-S24 and S26-S34). In turn, the *informational* perspective was only considered by 7 studies (i.e., S3, S4, S15, S17, S25, S32, and S34) and the *organizational* one

by 5 studies (i.e., S3, S17, S22, S25, and S34). Finally, none of the identified studies considered variability of the *temporal* or *operational* perspective.

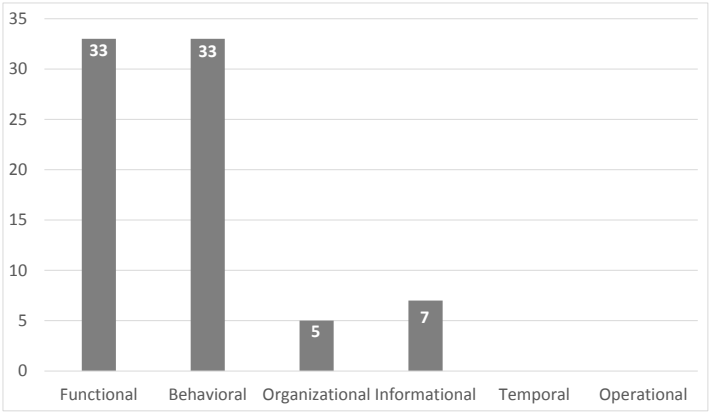


Figure 4.6: Distribution of studies S1-S34 according to the process perspectives covered

As can be seen in Figure 4.6, most studies solely cover variability with respect to control flow (i.e., the *functional* and *behavioral* perspectives). However, studies S3, S17 and S34 are more complete covering four perspectives (i.e., *functional, behavioral, organizational,* and *informational*). Finally, studies S4, S15, S22, and S32 at least cover 3 different perspectives (e.g., *functional, behavioral,* and *informational*).

4.2.5 Existing Tools for Managing Process Variability

This section deals with the analysis of RQ5, which refers to available tools providing support for process variability (i.e., *What tools exist for enabling process variability?*). For answering research question RQ5, we analyzed the studies describing process variability approaches (i.e., S1-S34) and variability support features (i.e., S35-S50). In particular, these studies might refer to available tool implementations.

We found 41 tools for managing process variability. Out of them,

however, only 10 were available online; i.e., these tools can be downloaded from websites (including manuals and tutorials); i.e., S2, S11, S13, S22, S24-S26, S28, S29, and S33. Figure. 4.7 lists these tools.

Furthermore, we observed that until 2013 most tool implementations constitute proof-of-concept prototypes not yet ready for industrial adoption; i.e., they were developed with the goal to validate the feasibility of the proposed approaches. Besides this, the kind of tool differs, depending on the objective of the respective study. In detail, existing implementations provide graphical editors for modeling process variability (S29), model transformations that allow generating the entire family of process variants from a feature model (S26), and extensions of existing process modeling languages for explicitly representing variability (S5, S13). In some studies, these implementations are realized as an Eclipse plug-in (S1, S8, S15, S25, S27, S28) or a proprietary Java tool (S3 and S24).

Other implementations integrate existing tools for realizing a particular process variability approach. Examples include S6, S9 and S33, which implement a set of plug-ins to integrate a feature model editor with the Protègè tool, a transformation from BPMN to Description Logic, and Rational Software Modeler, respectively. Finally, other process variability approaches are implemented by extending commercial BPM suites such as *ARIS Architect* (S19, S23), *IBM Rational Software Architect* (S30), and *WebSphere BPEL4WS* (S31).

4.2.6 Variability Support Features

This section summarizes the variability support features extracted in the context of RQ6 (*What variability support features are provided for fostering process variability in all phases of the process lifecycle?*). For answering this research question, we analyzed studies describing process variability approaches (i.e., S1-S34) and variability support features (i.e., S35-S50). We organize the features along the phases of the process lifecycle (cf. Section 2.1).

Study ID	Reference
S2	soa.fbk.eu/node/218
S11	www.promtools.org/prom6/
S13	www.markus-doehring.de/phd/index.php?option=com_content&view=article&id=59&Itemid=63
S22	modalis.polytech.unice.fr/softwares/manvarwor
S24	www.processconfiguration.com/download.html
S25	www.processconfiguration.com/download.html
S26	www.eclipse.org/m2m/atl/atlTransformations/#FM2BPMN
S28	www.yawlfoundation.org/
S29	https://se.cs.toronto.edu/trac/ome/wiki
S33	gp.uwaterloo.ca/fmp2rsm

Last accessed: October, 2015

Figure 4.7: Downloading links of available tools

Phase I: Analysis & Design

In the *analysis & design* phase, a process family is designed, modeled, validated, and verified using a particular process variability approach. In this context, language constructs such as the ones introduced in Section 4.2.3 are provided in order to specify and represent the common as well as the variable parts of the process variants of a process family in a configurable process model. Relevant features identified for this phase are as follows:

Feature F1.1 (Modeling a configurable process model). Tool support is needed for designing a configurable process model that represents an entire process family (i.e., the collection of all process variants). In this context, we must consider all language constructs introduced (cf. Section 4.2.3) as well as appropriate tool support for them. Since graphical process models are usually more comprehensible than non-graphical ones [Weske, 2007], in addition, graphical editors, navigation features, and visualization support are required to facilitate the creation of such models. Studies S1, S3, S4, S6, S7, S9, S10, S12-S23, S25-S28, and S30-S34 provide various techniques supporting this feature.

Feature F1.2 (Verifying a configurable process model and its related process family). Efficient techniques are needed in order to ensure that configurable process models are *syntactically* correct and

behaviorally sound.[3] This means, it must be guaranteed that solely syntactically correct and behaviorally sound process variants can be derived from a configurable process model. The verification may be accomplished during the creation of the configurable process model or latter. Feature F1.2 is considered by studies S23, S32, S35, S40, S41, S48, and S49.

Feature F1.3 (Validating a configurable process model). Techniques are needed for validating the *semantic* correctness of configurable process models. In particular, it must be ensured that a configurable process model properly covers all relevant variants of a business process. Again, such a validation may be accomplished during the creation of a configurable process model or afterwards. Studies S42 and S46 use logic formulas to address this issue.

Feature F1.4 (Evaluating the similarity of different process variants). In order to reduce modeling efforts, techniques for determining the similarity between related process variants are needed. Before adding a process variant to a process family, for example, it needs to be checked whether a similar process variant already exists. This is crucial in order to avoid redundancies and duplications. Studies S32, S44 and S47 provide methods and algorithms for this purpose.

Feature F1.5 (Merging process variants). In order to avoid mode redundancy and foster model reusability, techniques for integrating (i.e., merging) a collection of related process variants in a configurable process model are needed. Corresponding techniques are provided by studies S11, S23, S38, S39, and S43. Usually, a configurable process model resulting from their application covers the behavior of all process variants merged. In addition, reversibility techniques that allow deriving any of the input process variants from the configurable process model through individualization are useful as well. Studies S11, S38 and S43 describe methods for realizing such reversibility; i.e., they ensure the traceability of each variant after having performed the merging process.

[3] Regarding a sound process, all activities may be executed in the context of at least one process instance and no deadlocks or livelocks may occur.

Phase II: Configuration
The goal of the *configuration* phase is to derive an executable process variant (i.e., a member of the process family) through a configuration of the configurable process model. This is denoted as *individualization* process. Furthermore, the resulting process variant then needs to be deployed on the enactment system (e.g., workflow management system). We identified Feature F2 for this phase:

Feature F2 (Configuring a process variant out of a configurable process model). In general, tools should provide sophisticated user interfaces. Furthermore, proper techniques for retrieving the current application context and deriving an appropriate process variant for it are required. On one hand, algorithms for checking the syntactical correctness and soundness of configured process variants as well as their conformance with the specified configuration constraints (cf. Section 4.2.3) are required. For example, inclusion (exclusion) constraints may enforce (exclude) configuration alternatives with respect to a specific configurable region. In particular, users should be prohibited from deriving invalid process variants, e.g., by informing them about constraint violations. On the other hand, techniques enabling a high level of abstraction are required when specifying a particular application context; i.e., the configuration of a particular process variant should be accomplished at a high level of abstraction. Furthermore, the process variant resulting from a configuration (i.e., individualization) procedure should be graphically displayed to users. This feature is supported by studies S1, S8-S11, S14, S15, S17, S21-S26, S28-S30, S33, S34, S41, and S50 based on different techniques for configuring a process variant. For example, studies S24 and S25 provide a *questionnaire model* (i.e., form-based questionnaire) that allows individualizing a configurable process model by answering questions about the respective application context. Another well-known configuration technique is provided by *feature models* (e.g., S41 and S50). The latter map features to configuration alternatives; i.e., when a feature is selected, the configurable process model becomes configured automatically. Other techniques we discovered with respect to the support of the configuration phase include *configuration*

algorithms (e.g., S8 and S23), *goal models* (S29), and *decision tables* (S10). Figure 4.8 illustrates abstract examples of these techniques. To be more precise, it depicts a *questionnaire model* (part A), a *feature model* (part B), a *goal model* (part C), and a *decision table* (part D).

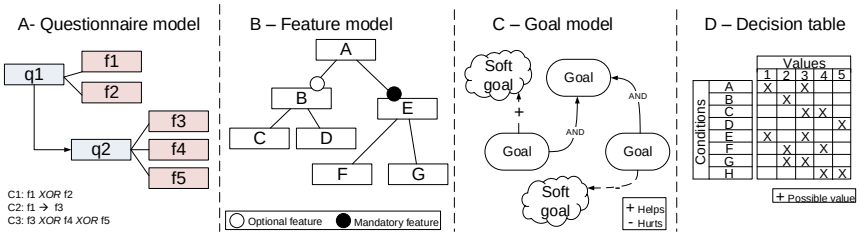


Figure 4.8: Example of configuration techniques

Phase III: Enactment

During the *enactment* phase, instances of a (configured) process variant may be created, started and executed. In this context, it should be possible to dynamically configure or re-configure a process variant if required [Ayora et al., 2012a]; i.e., to switch from one process variant to another during enactment time. Note that such dynamic (re-)configuration might become necessary due to contextual changes occurring during enactment time [Alférez et al., 2014]. However, a dynamic (re-)configuration must be accomplished in a controlled and robust manner. For example, automated re-configurations of a sound process variant instance should always result in a sound process variant instance again. Note that this differs from ad-hoc changes as supported in adaptive PAISs [Reichert & Weber, 2012]. Usually, ad-hoc changes correspond to unplanned dynamic changes, whereas a dynamic re-configuration switches the execution of a process instance from its current variant model to another pre-specified one. In detail, we observed the following features for the enactment phase:

Feature F3.1 (Configuring specific regions of a process variant at enactment time). Certain configuration decisions can solely be made at enactment time when required data becomes available. In order to address this issue, late modeling techniques [Reichert & Weber, 2012] for configuring process variants at enactment time are provided by S4, S5, S13, and S32. This feature is illustrated by Figure 4.9: A part of the process variant (indicated as activity X) is deemed to be of dynamic nature. In particular, X is defined based on a set of activities (i.e., C, D, E, F, G, and H) and a corresponding set of constraints restricting their use (presented as logic formulas in Figure 4.9). During enactment time, for a given process variant instance, X may be concretized based on available context data or user decisions. For the given scenario, Figure 4.9 shows a particular design of the process for which X is substituted by a process fragment composed out of activities G and H. Note that this substitution is valid in terms of the prescribed constraints (e.g., G and H *exclude* each other).

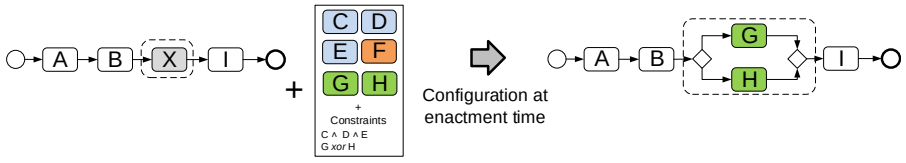


Figure 4.9: Configuring a specific region of a process variant at enactment time

Example 6 (Configuring specific regions of a process variant at enactment time). Whether or not a passenger carries overweight luggage is not known until she arrives at the counter. Hence, the *Pay extra fee* activity may only be selected when enacting instances of the respective process variant.

Feature F3.2 (Dynamically re-configuring an instance of a process variant at enactment time). For a particular instance of a

process variant, it might become necessary to dynamically switch its execution from the current process variant model to another pre-specified one. Such dynamic re-configurations might be required, for example, to react to contextual changes [Reichert & Weber, 2012]. Studies S1, S2, S12, and S19 provide techniques that support this advanced feature. Figure 4.10 illustrates it for a process variant instance that is dynamically re-configured to another variant model (i.e., the execution of activity F substitutes the one of activities C, D and E).

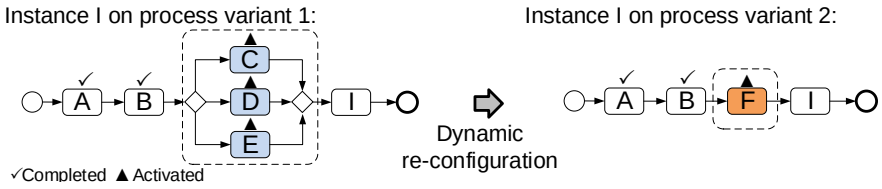


Figure 4.10: Dynamically re-configuring an instance of a process variant

Example 7 (Dynamically re-configuring an instance of a process variant at enactment time). Regarding the check-in process, changes of the passenger status might require dynamic variant switches. For example, consider a passenger not having entered her frequent flying number when buying the ticket and therefore being initially treated as a regular customer. When providing the frequent flying number later, a switch to another process variant needs to be performed.

Phase IV: Diagnosis

In the *diagnosis* phase, a collection of configured process variants is analyzed to learn from the configuration settings made at design and enactment time.

Feature F4 (Analyzing a collection of process variants). Techniques for learning from the configuration settings chosen when configuring the process variants at design or enactment time are needed; i.e., by analyzing the structure as well as the behavior of a given collec-

tion of process variants, an improved configurable process model might be obtained. Studies providing support for this advanced feature include S36 and S45.

Phase V: Evolution
This phase deals with the *evolution* of a process family and the configurable process model representing its members in order to cope with changing and evolving requirements. Examples of such evolutionary changes include the addition of new process variants (i.e., variants that cannot be configured out of the configurable process model so far), the removal of existing ones (i.e., process variants that must no longer be configured), and the modification of existing process variants to increase their quality (e.g., to improve model comprehensibility). In order to enable such an evolution, a configurable process model must be changed accordingly. In this context, we identified the following features:

Feature F5.1 (Versioning of a configurable process model). Techniques allowing for the co-existence of different versions of the same configurable process model are needed, particularly in the context of long-running processes. For example, study S46 presents a method using *version-graph models* to support this feature. Figure 4.11 shows four versions of a configurable process model and the associated *version graph* to manage them.

Feature F5.2 (Propagating changes of a configurable process model to already configured process variants). When changing a process fragment in a configurable process model, which is common to several process variants, the changes must be propagated across all (already configured) process variants in order to maintain overall consistency of the process family and to reduce maintenance efforts. Techniques for propagating changes of a configurable process model to already configured process variants are described in studies S3 and S37. Figure 4.12 illustrates this feature.

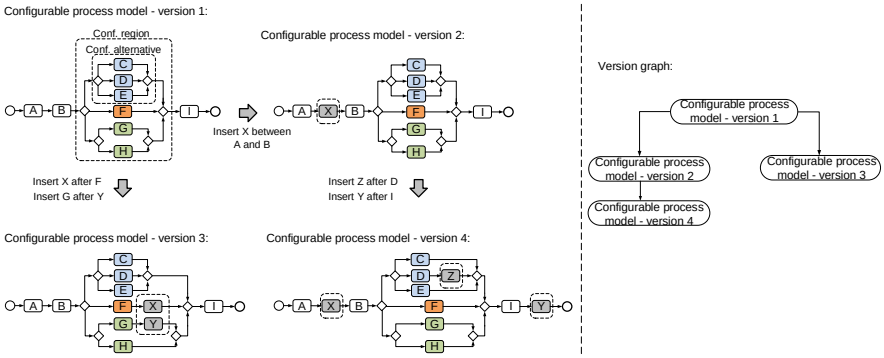


Figure 4.11: Versioning of a configurable process model

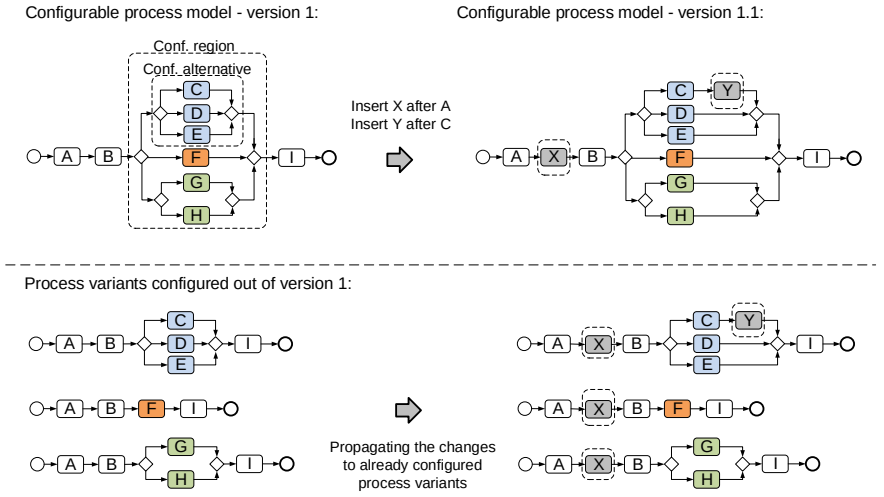


Figure 4.12: Propagating changes between configured process variants

Figure 4.13 summarizes the identified variability support features and presents the primary studies supporting them. Note that, in addition to these variability support features, well known features for managing single (i.e., individual) process models are applicable in the context of process families as well. As example consider algorithms measuring process model similarity [Dijkman, 2008; Dijkman et al., 2011a] or techniques enabling process model refactorings [Dijkman et al., 2011b]. Both might improve the management of process families as well [Reichert & Weber, 2012]. This work excludes such standard features since it focuses on variability-specific language constructs and support features.

		Studies supporting the feature
Variability support features	**Analysis & Design phase**	
	F1.1 Modeling a configurable process model	S1, S3, S4, S6, S7, S9, S10, S12-S23, S25-S28, S30-S34
	F1.2 Verifying a configurable process model and its related process family	S23, S32, S35, S40, S41, S48, S49
	F1.3 Validating a configurable process model	S42, S46
	F1.4 Evaluating the similarity of different process variants	S32, S44, S47
	F1.5 Merging process variants	S11, S23, S38, S39, S43
	Configuration phase	
	F2 Configuring specific regions of a process variant out of a configurable process model	S1, S8-S11, S14, S15, S17, S21-S26, S28-S30, S33, S34, S41, S50
	Enactment phase	
	F3.1 Configuring specific regions of a process variant at enactment time	S4, S5, S13, S32
	F3.2 Dynamically re-configuring an instance of a process variant at enactment time	S1, S2, S12, S19
	Diagnosis phase	
	F4 Analyzing a collection of process variants	S36, S45
	Evolution phase	
	F5.1 Versioning of a configurable process model	S46
	F5.2 Propagating changes of a configurable process model to already configured process variants	S3, S37

Figure 4.13: Variability support features and studies supporting them

4.2.7 Empirical Evaluation of Process Variability Approaches

This section describes the results for RQ7, which refers to empirical evaluations existing until 2013 in the context of process variability (i.e., *Have existing process variability approaches been evaluated? If so, how does this evaluation look like?*). For answering this research question, we

analyzed studies describing process variability approaches (i.e., S1-S34) and empirical evaluations of these approaches (i.e., S51-S63).

We found that two different methods have been applied to empirically evaluate process variability approaches: case studies and experiments (cf. Figure 4.14). Case studies constitute the most popular method applied in the context of 12 studies (i.e., S28, S51-54, S56, S57, S58, and S60-S63). Furthermore, 4 studies deal with experimental validations (i.e., S1, S23, S55, and S59). Study S1 uses the *Goal-Question-Metric method* to evaluate the design of a configurable process model. In turn, S23 reports on interviews with practitioners after they interacted with a configurable process model. In turn, study S55 uses similarity metrics to cope with the complexity (e.g., size) of configurable process models. Finally, S59 provides mapping patterns to compare two process variability approaches (i.e., S23 and S28) in terms of complexity (e.g., size of the resulting models).

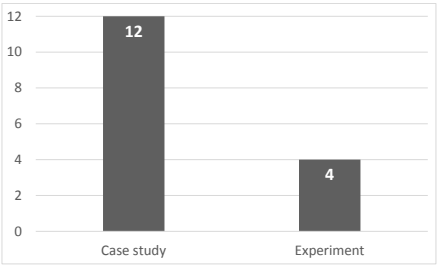


Figure 4.14: Methods applied to empirically evaluate process variability approaches

It is noteworthy that there only exist few concrete evaluations of process variability approaches. In turn, this indicates a lack of empirical evaluations of existing process variability approaches, which have not matured to the level of general industrial adoption yet (e.g., regarding scalability and usability).

4.2.8 Application Domains

This section gives insights into RQ8, which refers to the domains in which existing process variability approaches have been applied (i.e., *In which domains have existing process variability approaches been applied?*). For this purpose, we analyzed studies describing process variability approaches (i.e., S1-S34) and empirical evaluations of these approaches (i.e., S51-S63). We observed that process variability approaches have been applied to different domains. The latter include e-government (S12, S28, S53, S54, S55, and S60), retail (S10, S16, and S58), finance (S23), automotive industry (S2), healthcare (S17 and S61), and film production (S24 and S25). Note that this list only includes those domains for which there exists a clear evidence that the process variability approach is applied to real business processes; i.e., we do not considered domains for which fictitious processes are described.

4.2.9 Aspects Cutting Across VIVACE Aspects

The aspects of VIVACE described in the previous sections are not completely independent from each other. This section analyzes the relations among the results.

First, the analysis made in the context of research questions RQ1 and RQ2 reveals that many of the language-independent process variability approaches rely on a *multi-artifact technique* for building configurable process models. This applies to 14 out of 17 language-independent process variability approaches: S1, S3, S5, S9, S10, S15, S19, S20, S22, S24, S29, S30, S32, and S33.

Second, the *multi-artifact technique* (RQ2) enables a broader support of the variability-specific language constructs identified (RQ3). Most process variability approaches relying on a multi-artifact technique (e.g., S1, S14, and S15) support three or more of these language constructs. This may be explained by the fact that the use of additional artifacts allows defining broader aspects of a configurable process model; e.g., configuration context conditions or configuration constraints.

Third, it is noteworthy that the identified language constructs (RQ3)

are mainly supported with respect to the *functional* and *behavioral* process perspectives (i.e., control flow) (RQ4). This is plausible since the most supported variability-specific language constructs (i.e., configurable region and configuration alternative) are related to the control flow of the process.

Fourth, tool support available for a particular process variability approach (RQ5) does not entail an empirical evaluation of the respective approach (RQ7). Although tools may facilitate the evaluation of an approach, only 3 process variability approaches (i.e., S1, S23 and S28) have been both implemented and empirically evaluated.

4.3 VIVACE in Practice

Like other frameworks [Sinnema et al., 2006; Aiello et al., 2010], VIVACE is intended to systematically assess and compare process variability approaches with respect to their expressiveness and the features provided for the support of process variability in the different phases of the process lifecycle. In order to illustrate the way VIVACE can be applied in practice, we exemplarily assess selected process variability approaches. In detail, the latter include *C-EPC* (S28) (cf. Section 4.3.1), *Provop* (S19) (cf. Section 4.3.2), and *PESOA* (S30) (cf. Section 4.3.3). We select these approaches since they are (1) well established and highly cited, (2) there exists a mature tool support for them, and (3) they represent variability using different techniques (cf. Section 4.2.2). For each selected approach, based on VIVACE we provide a general description, discuss its expressiveness with respect to process variability modeling, and assess its lifecycle support for process variability. As evaluation criteria, we consider the results gathered in the context of the presented research questions (cf. Section B.1).

For both, variability-specific language constructs (cf. Section 4.2.3) and variability support features (cf. Section 4.2.6), we differentiate between *no support* [-], *partial support* [+/-], and *full support* [+]. In addition, regarding the process perspectives supported (RQ4), we use codes to indicate the perspectives covered by the approach: behavioral

(B), functional (F), organizational (O), informational (I), temporal (T), and operational (Op). In this vein, we use these codes for Language Construct LC2 (i.e., *configuration alternative*) in order to indicate the process perspectives it covers. Finally, we summarize evaluation results in Section 4.3.4.

4.3.1 Applying VIVACE to Configurable EPC

General description. A possible way of realizing a configurable process model is to enrich a process model with *configurable nodes*. A modeling language supporting this approach is C-EPC (i.e, Configurable EPC). C-EPC extends an existing process modeling language (i.e., EPC) by introducing configurable elements. In particular, this allows merging the behavior of all valid process variants in one and the same artifact, i.e., the configurable process model corresponds to one artifact (*single artifact technique*). Configurable nodes have been introduced for other process modeling languages as well (e.g., YAWL [van der Aalst & ter Hofstede, 2003]). Figure 4.15 illustrates the use of C-EPC in the context of the check-in process (cf. Appendix A). Configurable nodes are depicted with a thicker line. We do not add intermediate events between functions in order to keep the size of the configurable process model as small as possible. This helps us mitigate possible undesirable effects on understandability and likelihood of errors due to model size [Mendling et al., 2010]. In addition, practitioners recommend not to include events between functions in EPC for the sake of simplicity [ARIS-Community, 2010]. Thus, the configurable nodes correspond to process fragments with single entry and single exit (i.e., SESE fragment). They may have two different forms. On one hand, the SESE fragment may consist of a splitting configurable connector, immediately followed by a set of branches representing configuration alternatives, and a joining configurable connector; i.e., the configurable connectors delimit a configurable region (e.g., Configurable region 1 in Figure 4.15). Alternatively, the SESE fragment may consist of a configurable function (e.g., Configurable region 2 in Figure 4.15), which may be configured as ON (i.e., the function shall be kept in the configured process model),

OFF (i.e., the function shall not be included in the configured process model), or OPT (i.e., the function shall be conditionally included in the configured process model deferring the decision about its execution to enactment time). In turn, SESE fragments representing the different *configuration options* are included as branches between two configurable connectors (e.g., *Localize assistance to accompany passenger* in Configurable Region 1 in Figure 4.15). Further, note that the *application context* is represented separately from the configurable process model in a *questionnaire model* [La Rosa et al., 2009b]. Note that the latter is not depicted in Figure 4.15 due to lack of space. Finally, semantic constraints with respect to the configuration of configurable functions and connectors (e.g., mutual exclusion, inclusion) may be specified in terms of *configuration requirements* linked to the configurable nodes. For example, *Configuration Requirement 1* in Figure 4.15 states that the configurable function *Fill in unaccompanied form* is only included if SEQ1b is selected in XOR1; i.e., activity *Assign seat for UM* is selected.

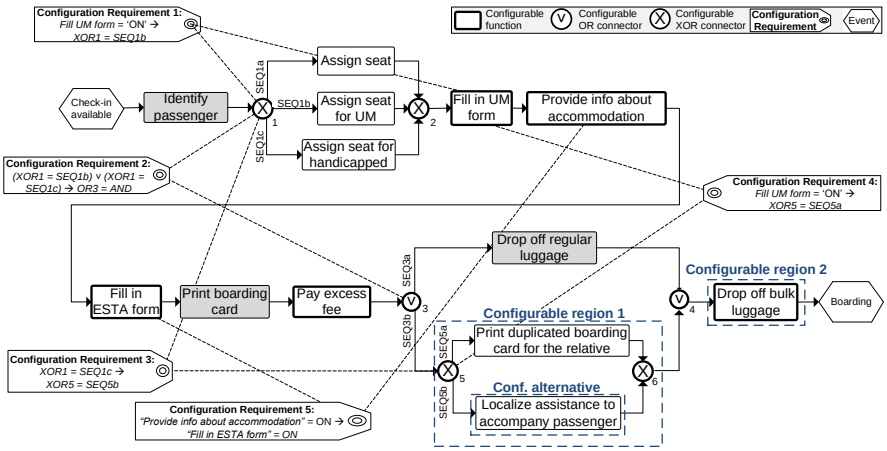


Figure 4.15: Configurable process model of the check-in process (in C-EPC notation)

Process variability expressiveness. Regarding the identified variability-specific language constructs presented in Section 4.2.3, in C-

EPC, a configurable region (LC1) is specified in terms of a configurable connector or function (LC1 [+]). In turn, a configuration alternative (LC2) corresponds to a SESE fragment that may either be included as a branch between two configurable connectors (e.g., *Localize assistance to accompany passenger* in Configurable Region 1 in Figure 4.15) or excluded. Basically, configuration alternatives consider the *functional* and *behavioral* perspectives. In addition, extended support with respect to the *organizational* and *informational* perspectives is provided [La Rosa et al., 2009b] (LC2 [F, B, O, I]). In turn, configuration context conditions (LC3 [+]) are represented separately in a *questionnaire model* (see [La Rosa et al., 2009b]). A configuration constraint is specified in terms of a configuration requirement. The latter may be linked to the configurable nodes that delimit the configurable region to which the respective configuration alternatives belong (LC4 [+]); e.g., *Configuration Requirement 1* in Figure 4.15. Finally, configurable region resolution time is supported since configurable functions can be configured to OPT, deferring their configuration to enactment time when the context information becomes available (LC5 [+]); e.g., configurable function *Pay excess fee* in Figure 4.15.

Process variability support. The C-EPC approach has been implemented in a toolset called *Synergia*, which provides a number of variability support features as well [Synergia, 2009]. In particular, Synergia supports the creation of a configurable process model using a graphical editor. Moreover, it allows defining the context of a configurable process model using configuration requirements (F1.1 [+]). Further, the Synergia toolset provides a mapper tool that can be used to verify a configurable process model and its related process family [Rosa, 2009] (F1.2 [+]). In addition, it is possible to validate configured process models using *C-EPC Validator* [Mendling, 2008] (F1.3 [+]). No support is provided for measuring the similarity between process variants (F1.4 [-]), whereas sophisticated merging techniques are presented in [La Rosa et al., 2010] (F1.5 [+]). The configuration of process variants is supported by a questionnaire-based approach, which has been implemented in the *Quaestio* tool [Rosa, 2009]. By answering a set of questions, domain experts are assisted and guided in configuring configurable process

models and hence in deriving a specific process variant (F2.1 [+]); i.e., domain facts (answers) are mapped to configuration choices. Configuration at enactment time and dynamic re-configuration of a process variant are not considered; once the configuration of a C-EPC model is finished, the resulting process model variant is deployed and cannot be changed anymore (F3.1 [-], F3.2 [-]). In turn, support for optimizing process variant models is provided in [Gottschalk et al., 2008] (F4.1 [+]). No explicit support exists for handling different versions of a configurable process model or propagating model changes to process variants (F5.1 [-], F5.2 [-]). In addition, the C-EPC approach has been empirically evaluated in a case study [Lönn et al., 2012]. Finally, C-EPC has been applied to business processes from different domains (e.g., e-government and film production).

4.3.2 Applying VIVACE to Provop

General description. In Provop, a pre-specified *base process model* (*base process* for short) is adjusted to the given application context through a sequence of model changes; i.e., context-specific structural adaptations are applied in order to derive a particular process variant from a configurable process model in Provop [Hallerbach et al., 2010a, 2009]. Furthermore, a base process may be specified with any process modeling language; i.e., Provop provides a language-independent approach.

Figure 4.16 illustrates how the check-in process family can be represented in Provop. In particular, the configurable process model can be represented through several artifacts (i.e., *multi-artifact technique*). The top of Figure 4.16 depicts the base process based model out of which the process variants can be configured. Figure 4.16 further shows the structural adaptations (i.e., *change options*) that may be applied in isolation or combination to this base process when configuring a particular process variant. In Provop, configurable regions of the base process are specified by a SESE fragment, delimited by two *adjustment points* (i.e., black diamonds). For example, in Figure 4.16 a configurable region comprises the process fragment delimited by *adjustment points* A

and B. In turn, a configuration alternative is specified in terms of a *change option*, which includes (1) a list of atomic *change operations* modifying a configurable region of the base process and (2) a *context rule* that defines the context conditions under which the change operations shall be applied (e.g., Option 1 in Figure 4.16). The application context is specified in terms of *context rules*, which include a set of context variables and their values specifying the conditions under which a configuration alternative (i.e., a change option) shall be chosen (e.g., Option 2 shall be applied if the passenger is a handicapped person). All *context variables* and their allowed values are gathered in the *context model* (cf. Figure 4.16). Finally, semantic constraints (e.g., mutual exclusion or inclusion) may be specified between two change options in the *option constraint model*; e.g., if Option 2 is applied, Option 3 shall be excluded (cf. Figure 4.16).

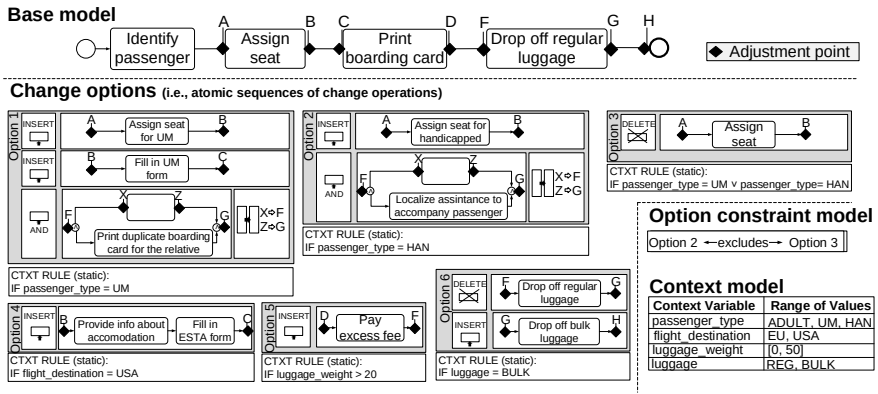


Figure 4.16: Provop model of the check-in process

Process variability expressiveness. In Provop, a configurable region is specified in terms of a SESE fragment of the *base process*, delimited by two adjustment points (LC1 [+]). In turn, a configuration alternative (LC2) is specified in terms of a change option. These alternatives may be defined with respect to the control flow perspective,

i.e., behavioral and functional perspective (LC2 [F, B]). In turn, configuration context conditions can be specified in terms of context rules (LC3 [+]), while configuration constraints are specified as constraints between change options in the *option constraint model* (LC4 [+]). Finally, Provop does not allow for the specification of the configurable region resolution time (LC5 [-]).

Process variability support. The Provop approach has been implemented in a *proof-of-concept* prototype based on the *ARIS Business Architect tool* [Reichert et al., 2009]. The creation of a configurable process model is supported by a graphical editor, which allows creating a base model and specifying the options that may be used to configure it (F1.1 [+]). The prototype further supports the definition of a context model through a set of context variables. Moreover, relationships between variants can be defined in the *option constraint model*. The configured process models can be verified to ensure their correctness (F1.2 [+]). However, no validation support is provided (F1.3 [-]). In addition, techniques for measuring the similarity of process variants [Li et al., 2008] as well as for merging process variants [Li et al., 2011] are provided (F1.4 [+], F1.5 [+]). The prototype further provides configuration support (F2.1 [+]). Depending on the actual context, a (sub-)set of the available change options is applied to the base process model resulting in a context-specific process variant. The Provop prototype checks whether the defined options violate any constraint defined on the total set of change options. If such a constraint violation is detected, the prototype notifies the process engineer accordingly. After selecting the change options relevant in the given application context, these are applied to the base process and the resulting process variant is displayed to the user. Dynamic configuration at enactment time is not supported by the Provop prototype (F3.1 [-]). Opposed to this, dynamic re-configuration of a process variant instance due to contextual changes at enactment time is supported by including variant branches in the process model and encapsulating the adjustments of single change options within these variant branches (F3.2 [+]). Support for analyzing a collection of process variants is provided through a separate tool [Li,

2010], which can import the process variants created by the Provop prototype (F4.1 [+]). In turn, the evolution of configurable process models is not considered; i.e., neither support for handling different versions of a configurable process model nor support for propagating the changes of the base process to already configured process variants has not been provided yet (F5.1 [-], F5.2 [-]). Finally, Provop has been illustrated using processes from the automotive and healthcare domains.

4.3.3 Applying VIVACE to PESOA

General description. The PESOA approach represents a process family through a configurable process model including a set of annotations. More precisely, annotations are attached to the process activities that may be subject to variation [Puhlmann et al., 2006]. For this purpose, language-independent techniques like encapsulation, inheritance, design patterns, and extension points are used. In principle, therefore, PESOA may be applied to any process modeling language. Further, the application context of the variable activities is specified in terms of features attached to them. Accordingly, process variants are configured by selecting features in the respective feature model. However, the relationships that may exist between the alternatives of different variation points are not considered.

Figure 4.17 illustrates the configurable process model corresponding to the check-in process as defined in PESOA. It composes the related feature model, which, in turn, contains the features associated with the configuration alternatives. For example, the configurable process model comprises five optional activities (e.g., *Fill in UM form*, *Provide information about accommodation*, and *Drop off bulk luggage*) specified in terms of extension points. Moreover, an inheritance technique is provided in order to model the alternatives for activity *Seat assignment*. Attached to the definition of each alternative, there are context conditions (i.e., features) defining when the alternative shall be selected (i.e., *multi-artifact technique*). For example, activity *Pay excess fee* will only become available if variable *luggage_overweight* has value TRUE.

Process variability expressiveness. Configurable regions are

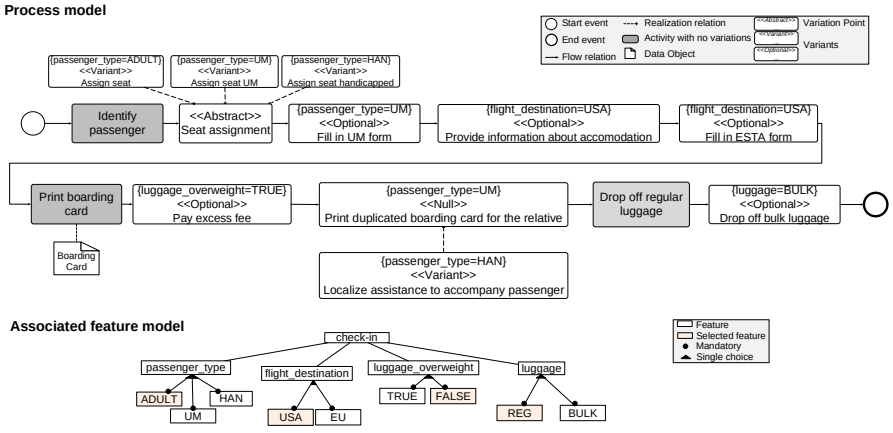


Figure 4.17: PESOA model of the check-in process

defined by attaching annotations to those activities that are subject to variation (LC1 [+]). In addition, configuration alternatives can be modeled using techniques like encapsulation, inheritance, design patterns, and extension points; e.g., *Provide information about accommodation* in Figure 4.17. However, these alternatives only refer to the behavioral and functional perspectives (LC2 [F, B]). Furthermore, the configuration context conditions of the alternatives are defined in terms of features attached to the activities instead of process variables (LC3 [+]). Finally, neither configuration constraints between configuration alternatives nor configurable region resolution time can be specified (LC4 [-], LC5 [-]).

Process variability support. PESOA has been realized as Eclipse plug-in that supports the creation of a configurable process model (F1.1 [+]). Its graphical editor allows representing configurable regions including configuration alternatives. In addition, it supports the definition of configuration context conditions. This is accomplished based on feature diagrams [Czarnecki & Antkiewicz, 2005], i.e., each process variant is tagged with features, determining the conditions under which this

variant is valid. However, neither verification nor validation support is provided (F1.2 [-], F1.3 [-]). In addition, neither similarity nor merging techniques exist (F1.4 [-], F1.5 [-]). The configuration of process models is supported by feature diagrams, i.e., for each disabled feature, the corresponding variable parts are removed from the process variant model (F2.1 [+]). Note that the usage of feature diagrams allows configuring process variants at a high level of abstraction. In addition, the resulting process variants are displayed to users. Since PESOA focuses on the modeling as well as configuration of process variants, features related to the deployment, execution, diagnosis, and evolution of process variants are not taken into account (F3.1 - F5.2 [-]). Finally, PESOA has been illustrated using processes from the retail domain.

4.3.4 Summary of the Evaluation

Figure 4.18 illustrates the VIVACE framework as well as its use for evaluating the selected process variability approaches. As shown, none of the selected process variability approaches supports the framework completely. To be more precise, the language constructs are fully supported only by one approach (i.e., C-EPC). In addition, the features introduced in Section 4.2.6 are only partially supported. The latter can be explained by the fact that all approaches lack support for the late phases of the lifecycle (i.e., enactment, diagnosis, and evolution).[4]

In absence of an established reference framework for process variability, VIVACE covers different aspects related to process variability. In particular, it covers modeling aspects of process variability (i.e., modeling languages and techniques, variability-specific language constructs, and process perspectives covered), existing support for process variability along the lifecycle (i.e., variability support features), existing tools, empirical evaluations of process variability, and application domains in which process variability approaches have been applied. In this context, based on the descriptions provided in [Gómez-Pérez, 2001], we discuss the *completeness, expandability,* and *consistency* of VIVACE.

[4]We will apply VIVACE to other approaches as well. Respective results will be made available on a website.

The *VIVACE* framework		Approach enabling process variability		
		C-EPC	**Provop**	**PESOA**
Modeling language used to represent process variability		Independent	Independent	Independent
Technique used for building the configurable process model		Single artifact	Multi-artifact	Multi-artifact
Method for modeling the process family		Configurable nodes	Operational	Annotations
Process perspectives covered		F, B, O, I	F, B	F, B
Variability-specific language constructs	LC1 Configurable Region	+	+	+
	LC2 Configuration Alternative	F, B, O, I	F, B	F, B
	LC3 Configuration Context Condition	+	+	+
	LC4 Configuration Constraint	+	+	-
	LC5 Configurable Region Resolution Time	+	-	-
Variability support features	**Analysis & Design phase**			
	F1.1 Modeling a configurable process model	+	+	+
	F1.2 Verifying a configurable process model and its related process family	+	+	-
	F1.3 Validating a configurable process model	+	-	-
	F1.4 Evaluating the similarity of different process variants	-	-	-
	F1.5 Merging process variants	+	-	-
	Configuration phase			
	F2 Configuring specific regions of a process variant out of a configurable process model	+	+	+
	Enactment phase			
	F3.1 Configuring specific regions of a process variant at enactment time	-	-	-
	F3.2 Dynamically re-configuring an instance of a process variant at enactment time	-	+	-
	Diagnosis			
	F4 Analyzing a collection of process variants	+	-	-
	Evolution			
	F5.1 Versioning of a configurable process model	-	-	-
	F5.2 Propagating changes of a configurable process model to already configured process variants	-	-	-
Tool implementation		+	+	+
Empirical evaluation		+	-	-
Application domain		e-government, film production	automotive, healthcare	retail

Figure 4.18: VIVACE framework applied to three selected approaches

- **Completeness**: The process variability aspects covered by VIVACE have been identified from an in-depth study, which provides a fundamental and profound understanding of business process variability. As a consequence of this methodological choice, the VIVACE framework only describes how process variability is currently supported (i.e., description and classification of existing literature), but not how support for process variability should look like (e.g., important variability support features that have not been addressed by existing research yet). In addition, VIVACE framework is influenced by the defined research questions (cf. Sec-

tion B.1). Although we are confident that these cover the most important process variability aspects (e.g., variability-specific language constructs and variability support features) and for characterizing existing literature, *completeness* cannot be guaranteed.

- **Expandability**: If other research proposes more aspects that cannot be fully mapped to the existing framework (e.g., need for adding other variability-specific language constructs), VIVACE may be expanded with the newly identified aspects. However, with every expansion of the framework it might potentially become necessary that aspects need to be merged and subsequently renamed (e.g., a newly added features is similar, but not identical, to an existing one and hence the two features might be merged and a new label covering both features might be assigned).

- **Consistency**: The orthogonality of the process variability aspects covered in VIVACE minimize the occurrence of inconsistencies in the results obtained after its application. For example, the assessment of a specific variability-specific language construct (e.g. configurable region) does not have an impact over the evaluation of variability support features.

4.4 Discussion

The data presented in the previous sections allowed us to answer the research questions that guided our study. This section provides a general discussion about VIVACE and its aspects.

First, we learned that we can distinguish between language-independent and language-specific process variability approaches (RQ1). Basically, language-independent approaches extend an existing process modeling language, but are intended to be applicable to other process modeling languages as well. However, we observed that the latter is far from being trivial due to existing language peculiarities. Despite the fact that language independence is useful for generalizing a process variability approach, it might be more suitable to focus on a well-established

process modeling language (e.g., standardized languages) as well as to develop variability-specific techniques optimized for this language. In particular, this would facilitate its industrial adoption and evaluation.

Second, we observed that 13 studies implement a *single artifact* technique to represent a process family. By contrast, 20 studies provide a *multi-artifact technique* making use of various modeling artifacts in order to represent the different aspects of a process family; e.g., common parts, variant-specific parts, constraints, and application context. Note that these figures do not provide evidence about which technique is more exposed. Hence, additional research is needed; e.g., experiments evaluating the techniques in different settings (e.g., varying model size or covered process perspectives).

Third, despite the high number of process variability approaches identified, a common set of variability-specific language constructs is essentially supported by most approaches (RQ3); i.e., *configurable regions* and *alternatives* (supported by 32 studies), *configuration context conditions* (15 studies), *configuration constraints* (24 studies), and *configurable region resolution time* (3 studies). Although existing approaches use different terminology (e.g., *configurable region* vs. *variation point*) and realize the language constructs in different ways (cf. Section 4.2.3), the five languages constructs we identified are the most prevalent ones in existing literature on process variability. In turn, this can be considered as a valuable insight. However, only two studies report on process variability approaches supporting all five variability-specific language constructs; i.e., most existing approaches do not cover the entire set of constructs. Hence, additional efforts are required to support all variability-specific language constructs in an integrated way in order to cover process variability in a more complete sense. In addition, the language constructs we identified as well as their descriptions can be used as a standard vocabulary when dealing with process variability. Thus, they will contribute to improving the communication among PAIS engineers.

Fourth, regarding the process perspectives covered by existing process variability approaches (RQ4), most efforts (i.e., 33 studies) have been spent on modeling the variability of the *functional* and *behavioral*

perspectives (i.e., activities and control flow). In turn, the *informational* and *organizational* perspectives are only covered by rather few studies (7 and 5 studies, respectively). On the contrary, none of the identified studies considers variability of the *temporal* or *operational* perspective even though variability support for these two perspectives is crucial in practice. As illustrated in the context of the check-in process (cf. Section 3.1), variability occurs with respect to all process perspectives. Accordingly, it must be properly handled for each of them in order to be able to represent processes families from the real world. Thus, an integrated approach for modeling process variability, which covers all language constructs as well as process perspectives, is needed. As a consequence, additional research is needed in order to extend existing process variability approaches such that they cover other process perspectives (i.e., the *temporal* and the *operational* ones) as well. Finally, it is noteworthy that the identified language constructs (RQ3) are mainly supported for the *functional* and *behavioral* perspectives (i.e., control flow). However, since variability is prevalent for all process perspectives, these language constructs need to be extended to cover all perspectives.

Fifth, process variability approaches should not only allow for the modeling of variability, but also provide tool support for managing process variants along the process lifecycle. In this line, we have identified 41 studies providing a tool implementation (RQ5). However, most of these implementations constitute proof-of-concept prototypes. In addition, only 10 of the 41 implementations are made available on respective websites. When making tools accessible to others, researchers promote a wider adoption of their solutions by practitioners. Thus, tools can be applied to real scenarios and other users might improve them.

Sixth, our findings reveal a core set of 11 variability support features to deal with process variability along the process lifecycle (RQ6). Five of these features are related to the modeling of a configurable process model: *modeling, verifying, validating, evaluating the similarity of variants,* and *merging process variants*. Regarding the *configuration* phase, there exist studies dealing with the *configuration* of process variants. In turn, for the *enactment* phase, techniques for *dynamically configuring* parts of a process variant instance as well as for *dynamically*

re-configuring process variant instances exist. Besides, several studies report on techniques for *analyzing* process variants. Finally, there are techniques for *maintaining different versions* of configurable process models as well as for *propagating changes* across process variants; i.e., techniques supporting the evolution of process families over time. Overall, this set of features will allow process engineers to evaluate the practical applicability of existing process variability approaches.

Basically, existing work on process variability has focused on modeling issues. However, more efforts are required with respect to the implementation, enactment, diagnosis, and evolution of process families and corresponding variants. In addition, similar to single (i.e., individual) business processes [Vergidis et al., 2008; Li et al., 2011; Li, 2010], more advanced techniques for optimizing process families need to be provided. For example, this includes support for identifying process variants that may be derived from a configurable process model, but never have been configured or deployed. In turn, respective optimizations might trigger the evolution of the configurable process model. In addition, optimization techniques might discover frequently applied configuration steps, which are then lifted up to the configurable process model to reduce future configuration efforts. Furthermore, refactoring support should be provided in order to improve the quality of a configurable process model; i.e., altering its schema, but without changing its behavior. These examples suggest emerging research areas related to process families (i.e., process variability).

Finally, there is no integrated process variability approach supporting the entire set of features along the process lifecycle. Basically, existing process variability approaches support the modeling, verification and configuration of configurable process models. However, none of the identified approaches supports a wider range of features; i.e., only very few process variability approaches support more than one feature; e.g., study S1 supports 3 features (i.e., F1.1., F2.1, and F3.2), while study S3 refers to 2 features (i.e., F1.1 and F5.1). As a consequence, more efforts are required in order to cover the entire set of features in an integrated way.

Used in combination with the identified language constructs, variability-specific features will help process engineers to analyze, compare, and assess the support provided by existing process variability approaches. While language constructs allow assessing the expressiveness required to explicitly model process variability in process families, the variability-specific features reflect the support required to adequately cope with such expressiveness along the different phases of the process lifecycle. Thus, another purpose of our work is to use these language constructs and variability-specific features as an *evaluation framework*, which we denote as *VIVACE* (cf. Figure 4.18). As opposed to other variability frameworks [Sinnema et al., 2006; Aiello et al., 2010], VIVACE has been the result of a systematic analysis of existing literature identifying which aspects are actually supported when dealing with process variability. Accordingly, we expect our framework to be applied to different process variability approaches as well as related tools in order to evaluate their suitability for process variability management. In the same vein, we expect VIVACE to support process engineers in implementing a PAIS supporting process variability along the process lifecycle. Finally, VIVACE may assist process engineers in selecting the variability approach meeting their requirements the best.

Seventh, only few process variability approaches have been extensively evaluated in practice so far (RQ7); i.e., 15 studies. However, the absence of a broader empirical evidence limits the acceptance of respective approaches and aggravates their practical use.

Eighth, we observed that process variability approaches have been applied in various domains, e.g., healthcare, retail, and e-government. This emphasizes the presence of variability in business processes from different domains.

4.5 Comparison with Other Characterizations

The overall goal of VIVACE is to provide a fundamental and profound characterization and understanding of process variability. In addition, we aim to comprehensively assess the support provided by existing pro-

cess variability approaches along the entire lifecycle of process families. However, to the best of our knowledge, there exist three other works that have to some degree characterized process variability as well.

First, *Lang* [Lang, 2002] reports on the results of a systematic mapping on flexible process modeling. A set of research questions is defined for identifying the types of approaches enabling flexible processes, the research method used (e.g., analysis, implementation), the contributions of the study (i.e., tool, method), the context in which these approaches were developed (i.e., industrial vs. academic), and their quality assessment. This work is broader in scope compared to our analysis since it focuses on process flexibility in general (see [Reichert & Weber, 2012] for a comprehensive description of process flexibility issues). On the contrary, we focus on a concrete aspect of process flexibility (i.e., the support of process variability through configurable process models) with the goal of providing profound insights into how existing approaches enable process variability support (i.e., modeling languages, techniques, language constructs, and features). Other aspects of process flexibility (i.e., looseness, adaptation, and evolution) were out of the scope of our work and hence we did not included them. In addition, we identified a set of language constructs and variability support features tailored towards the support of process variability; i.e., we establish the VIVACE framework for evaluating and comparing existing process variability approaches.

In [Valença et al., 2013], *Valença et al.* present a systematic mapping on process variability, which summarizes the theoretical background of this topic. Although the goal of this paper is related to the one of our work, the authors solely deal with design time issues. On the contrary, we considered all phases of the process lifecycle. This is relevant since execution and maintenance aspects of process variability will enrich its understanding. In addition, *Valença et al.* consider complementary keywords in the search string; e.g., “change”, “agility”, “reuse”, and “similarity”. As a result, they also retrieve studies related to features for managing single (i.e., individual) business processes (e.g., [Dijkman, 2008]). However, we restricted our search to variability-specific issues to set a clear focus on process variability support. Furthermore, we ex-

plore how process variability is actually supported and implemented by existing approaches, i.e., we provide a complete evaluation framework.

Furthermore, *Santos et al.* [dos Santos Rocha & Fantinato, 2013] conduct an systematic review on how software product lines (SPL) techniques have been applied to business process management. For this purpose, the authors analyze coarse-grained aspects of SPL techniques for business processes such as domain and application engineering, SPL architectures, variability management, and feature modeling. In our work, we are specifically investigating how variability can be managed in process families (e.g., languages, constructs, features). We attempt to provide detailed insights into the modeling of process variability as well as into the way process variants can be configured, enacted, evolved, and maintained, no matter what the used techniques are. On the contrary, [dos Santos Rocha & Fantinato, 2013] only considers process variability approaches related to SPL techniques (e.g., use of feature models to represent process variability), neglecting other methods for dealing with process variability (e.g., annotated configurable process models [Schnieders & Puhlmann, 2007]).

Finally, in the context of software processes, there exist reviews analyzing variability in software process tailoring. For example, [Martinez-Ruiz et al., 2012] identifies the requirements and mechanisms that consistently support process tailoring. Finally, [Pedreira et al., 2007] describes the tools, techniques, approaches, and experiences of variability in software engineering processes. However, these works were not deeply analyzed since their focus is not on business process variability as such.

4.6 Conclusions

This chapter aims to describe the study we conducted to analyze and characterized the domain of process variability. More precisely, we have studied systematically existing approaches dealing with process variability in order to analyze how process variability is actually modeled. As a result, we have identified a set of language constructs used to represent process variability. These constructs allows us to answer research

question 1 (cf. Section 1.2) of this thesis. In addition, they will be the base of our change patterns specifically tailored for modeling variability in process families.

The systematic study also allowed us to derive the *VIVACE framework*, a complete characterization of process variability. VIVACE is in a higher level of abstraction than the one provided by the existing process variability approaches. In addition, VIVACE supports PAIS engineers in (1) defining new process variability approaches, (2) improving their communication, (3) evaluating existing process management technologies enabling process variability, (4) selecting which of them meets their requirements best, and (5) dealing with (e.g., modeling, implementing) a PAIS that will effectively support variability along the process lifecycle.

In the following, and taking as a basis the identified variability-specific language constructs, we define the set of change patterns for modeling variability in process families.

5

Variability Management in Process Families through Change Patterns

Taking as a basis the variability-specific language constructs identified in VIVACE, we derive a set of change patterns for process families. Since existing adaptation patterns have been effectively applied in individual process models previously [Weber et al., 2008], we follow and adapt this perspective in order to provide patterns to deal with process variability in an explicit manner (i.e., based on the variability-specific constructs). We intend to provide a set of generic patterns that can be afterwards implemented in any of these approaches [Ayora et al., 2012c]. Thus, our patterns address process variability at a level of abstraction higher than the one provided by the existing process variability approaches [Ayora et al., 2015] (e.g., C-EPC).

The rest of the chapter is organized as follows. Section 5.1 explains how the patterns hav been derived. Sections 5.2-5.11 describe the set of change patterns in detail. Section 5.12 provides a discussion of the defined patterns. Finally, Section 5.13 concludes the chapter.

5.1 Change Patterns Derivation

For deriving the patterns, we applied the three basic operations over the identified variability-specific language constructs: *insertion, deletion,* and *modification* (cf. Table 5.1). As a result, we obtained four patterns to add variability-specific language constructs to a configurable process model (CP1, CP3, CP5, and CP8), four patterns to remove them (CP2, CP4, CP6, and CP9), and two patterns to modify them (CP7 and CP10). We do not consider patterns for modifying configurable regions, configuration alternatives, and configuration constraints. These modifications can be realized combining other change patterns and existing adaptation patterns. For example, modifying a configuration alternative may be implemented applying patterns CP3 and CP4. However, we defined CP7 and CP10 in order to modify in a more efficient way the variability-specific language constructs they affect (i.e., configuration context condition and configurable region resolution time).

In the following, we describe the set of change patterns in detail. In particular, for each pattern, we provide a name, a brief description, a description of the problem addressed, an illustrative example based on the check-in process (cf. Appendix A), and the design choices for its definition (i.e., indicating pattern variants). For example, CP1 presents three design choices: (1) insert a configurable region as a new region with a set of new configuration alternatives, (2) insert it by transforming a commonality into a configuration alternative (i.e., a common process fragment now is only applied in a specific process variant), or (3) insert it by transforming a set of commonalities into a set of configuration alternatives. To demonstrate that the patterns–despite their intended generic nature–can be realized, we show how they can be implemented in a well-established approach for modeling configurable process mod-

CP1: Insert Configurable Region
CP2: Delete Configurable Region
CP3: Insert Configuration Alternative in a Configurable Region
CP4: Delete Configuration Alternative from a Configurable Region
CP5: Insert Configuration Context Condition of a Configuration Alternative
CP6: Delete Configuration Context Condition of a Configuration Alternative
CP7: Modify Configuration Context Condition of a Configuration Alternative
CP8: Insert Configuration Constraint between Configuration Alternatives
CP9: Delete Configuration Constraint between Configuration Alternatives
CP10: Modify Configurable Region Resolution Time

Table 5.1: Change patterns for process families

els (i.e., *Configurable EPC* (C-EPC) [Gottschalk et al., 2007]). For example, for each design choice, we indicate how it can be implemented in C-EPC. This implementation in C-EPC guarantees *correctness-by-construction* in terms of structure and behavior [Dumas et al., 2013; Weske, 2007] by providing systematic means for introducing and deleting modeling elements correctly. For example, by only indicating the position of an element in the model, the implementation of the pattern would automatically take all the rest of necessary actions for correct insertion guaranteeing, for instance, that deadlocks are not introduced. Thus, configurable process models are also expected to be modeled more efficiently. In addition, for some cases pattern implementation is based on the use of adaptation patterns (cf. Section 3.3.2). This way we promote reuse between already existing patterns and the new ones. We further provide implementation details distinguishing between (i) configurable functions and (ii) configurable connectors since both allow representing configurable regions in C-EPC. In addition, we provide information about the parameters needed for each pattern. For example, realizing CP1 requires (1) the precise position in the configurable process model where the configurable region shall be inserted and (2) the configuration alternatives to be inserted in the configurable region (if needed). This information is highlighted in gray in the figures.

5.2 CP1: Insert Configurable Region

Description: A configurable region is added in a configurable process model.

Problem addressed: At a certain position in the configurable process model, different configuration alternatives that exist are not reflected in the configurable process model so far. Hence, a configurable region covering these configuration alternatives shall be added.

Example: The way how boarding cards are handled depends on the type of check-in (e.g., paper-based vs. electronic boarding cards). Assume that the configurable process model has not considered these configuration alternatives yet. Hence, a configurable region needs to be added to reflect this variability.

Design choices (DC):
(DC1) Insertion as a new configurable region with up to n configuration alternatives ($n \geq 0$)
(DC2) Insertion as a new configurable region by transforming a common process fragment into a configuration alternative
(DC3) Insertion as a new configurable region by transforming existing process fragments into a set of configuration alternatives

Implementation in C-EPC:
For DC1, CP1 is realized by
1. applying adaptation pattern AP1 (Insert Process Fragment) to insert the configurable region using either (i) a configurable function (cf. Figure 5.1) or (ii) two configurable connectors (i.e., split and join) at the precise position where the configurable region should be located (i.e., after activity B), and
2. applying repeatedly CP3 (Insert Configuration Alternative in a Configurable Region) to insert a process fragment representing the configuration alternative (only relevant for configurable connectors), i.e., the configuration alternative is added as a branch between the two config-

urable connectors delimiting the configurable region (i.e., activity X).

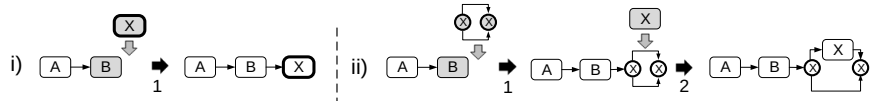


Figure 5.1: CP1: Design choice 1 implemented in C-EPC

For DC2, CP1 is realized by
1. applying adaptation pattern AP1 (Insert Process Fragment) to insert the configurable region using either (i) a configurable function (cf. Figure 5.2) or (ii) two configurable connectors (i.e., split and join) at the precise position where the configurable region should be located (i.e., after activity B),
2. applying adaptation pattern AP2 (Delete Process Fragment) to delete the common process fragment from its current position (i.e., activity B), and
3. applying CP3 (Insert Configuration Alternative in a Configurable Region) to re-insert the common process fragment as a configuration alternative of the configurable region (only relevant for configurable connectors), i.e., the alternative is added as a branch between the two configurable connectors delimiting the configurable region (i.e., activity B).

For DC3, CP1 is realized by
1. applying adaptation pattern AP1 (Insert Process Fragment) to insert the configurable region (only relevant for configurable connectors) at the precise position where the configurable region should be located (i.e., after the join XOR gateway),
2. applying adaptation pattern AP2 (Delete Process Fragment) to delete the existing process fragment from its current position, and
3. applying repeatedly CP3 (Insert Configuration Alternative in a Configurable Region) once per configuration alternative to re-insert the ex-

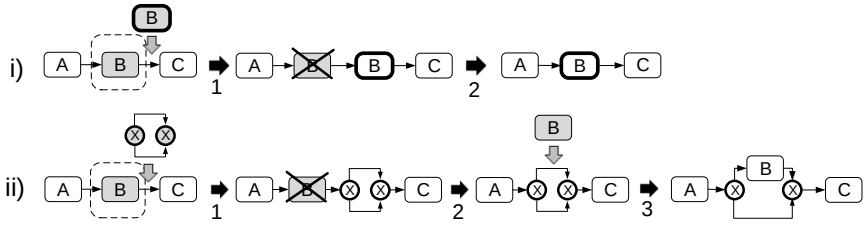


Figure 5.2: CP1: Design choice 2 implemented in C-EPC

isting process fragments as configuration alternatives of the configurable region, i.e., each process fragment is added as a branch between the two configurable connectors delimiting the configurable region (i.e., activity B is inserted as one alternative and activity C as another one in Figure 5.3).

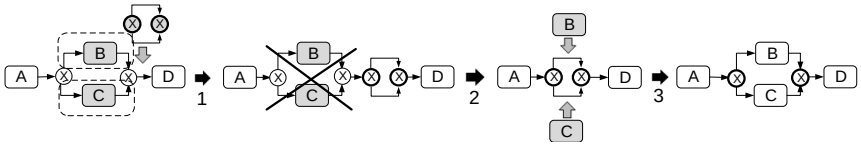


Figure 5.3: CP1: Design choice 3 implemented in C-EPC

Parameters:
– the position in the configurable process model where the configurable region shall be inserted
– the configuration alternative(s) to be added to the configurable region

5.3 CP2: Delete Configurable Region

Description: A configurable region of a configurable process model is deleted.

Problem addressed: A configurable region is no longer needed and thus it is deleted.

Example: Assume that a configurable region, capturing the variability for obtaining a boarding card, exists (i.e., paper vs. electronic document). However, in order to save money, the airline now only offers the electronic-based boarding card (i.e., other configuration alternatives are no longer offered) and hence the configurable region is no longer needed.

Design choices (DC):
(DC1) Deletion by removing all the configuration alternatives
(DC2) Deletion by keeping exactly one of the configuration alternatives (i.e., the configuration alternative remains as a common process fragment)
(DC3) Deletion by keeping the set of configuration alternatives

Implementation in C-EPC:
For DC1, CP2 is realized by
1. applying repeatedly change pattern CP4 (i.e., Delete Configuration Alternative in a Configurable Region) to delete each existing configuration alternative; i.e., once per configuration alternatives (only relevant for configurable connectors, i.e., activity X in Figure 5.4), and
2. applying adaptation pattern AP2 (Delete Process Fragment) to delete the configurable region in form of either (i) a configurable function or (ii) two configurable connectors (i.e., split and join).

For DC2, CP2 is realized by
1. applying repeatedly CP4 (Delete Configuration Alternative in a Configurable Region) to delete the existing configuration alternatives of the

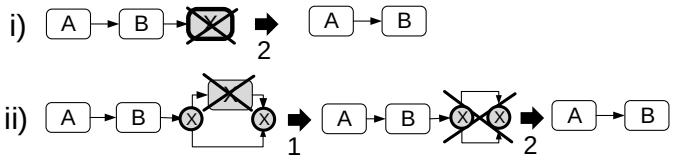


Figure 5.4: CP2: Design choice 1 implemented in C-EPC

configurable region (only relevant for configurable connectors, i.e., activity B in Figure 5.5),
2. applying adaptation pattern AP2 (Delete Process Fragment) to delete the configurable region in form of either (i) a configurable function or (ii) two configurable connectors (i.e., split and join), and
3. applying adaptation pattern AP1 (Insert Process Fragment) to reinsert the remaining configuration alternative as a (common) process fragment in the exact position where the configurable region was located (i.e., activity B).

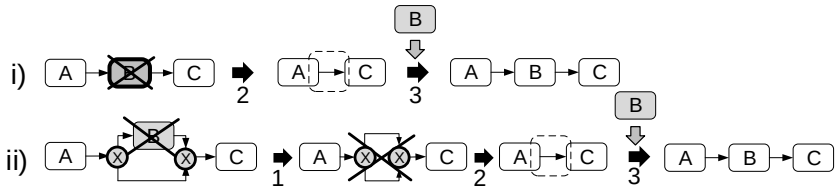


Figure 5.5: CP2: Design choice 2 implemented in C-EPC

For DC3, CP2 is realized by
1. applying adaptation pattern AP2 (Delete Process Fragment) to delete the existing process fragment (including the configurable region and its alternatives) from its current position,

2. applying adaptation pattern AP1 (Insert Process Fragment) to re-insert at the precise position where the configurable region was located a process fragment consisting of a two non-configurable connectors, and
3. applying repeatedly adaptation pattern AP1 (Insert Process Fragment) to re-insert the deleted configuration alternatives as branches between the two recently added non-configurable connectors (i.e., activity B is inserted as one branch and activity C as another one in Figure 5.6).

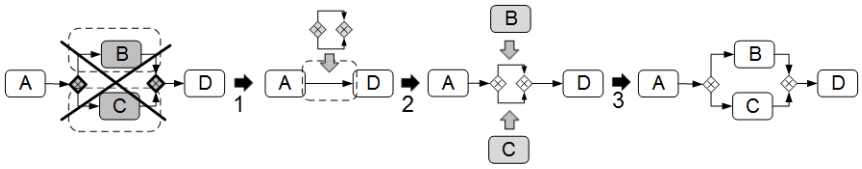


Figure 5.6: CP2: Design choice 3 implemented in C-EPC

Parameters:
– the configurable region to be deleted
– the configuration alternative(s) that should be kept

5.4 CP3: Insert Configuration Alternative in a Configurable Region

Description: A configuration alternative is added in a specific configurable region of a configurable process model.

Problem addressed: For a specific configurable region of the configurable process model, existing configuration alternatives do not cover all possible configuration choices so far.

Example: Assume that a configurable region, capturing the variability for obtaining a boarding card, exists (i.e., paper vs. electronic document). Assume further that the airline now wants to offer the possibility of obtaining the boarding card for smart phones as well. Thus, an alternative shall be added to this configurable region.

Implementation in C-EPC:
CP3 is realized by applying adaptation pattern AP1 (Insert Process Fragment) to insert the process fragment representing the configuration alternative, i.e., the configuration alternative is added as a branch between the two configurable connectors delimiting the configurable region (i.e., activity X in Figure 5.7).

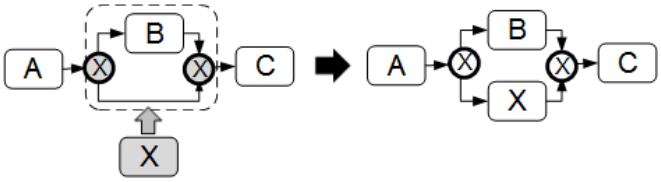


Figure 5.7: CP3 implemented in C-EPC

Parameters:
– the configurable region to which the configuration alternative belongs
– the configuration alternative to be inserted

5.5 CP4: Delete Configuration Alternative from a Configurable Region

Description: A configuration alternative is removed in a specific configurable region of a configurable process model.

Problem addressed: A configuration alternative is no longer needed and thus it is deleted.

Example: Assume that a configurable region capturing the variability for obtaining a boarding card exists (i.e., paper vs. electronic document). Assume further that for economic reasons, the airline does not offer paper-based boarding cards anymore allowing only electronic and mobile phone ones. Thus, the configuration alternative printing a paper boarding card is no longer needed.

Implementation in C-EPC:
CP4 is realized by applying adaptation pattern AP2 (Delete Process Fragment) to delete the process fragment representing the configuration alternative, i.e., the configuration alternative is deleted as a branch between the two configurable connectors delimiting the configurable region (i.e., activity X in Figure 5.8). If the configuration alternative is associated with configuration requirements, these may be deleted as well by applying CP9 (Delete Constraint between Configuration Alternatives), i.e., Configuration requirement 1.

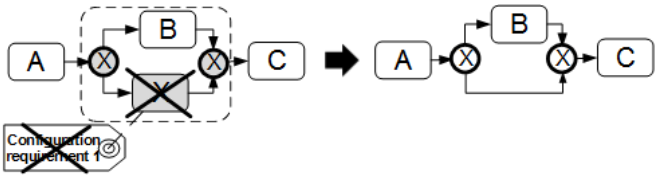


Figure 5.8: CP4 implemented in C-EPC

Parameters:
– the configurable region to which the configuration alternative belongs
– the configuration alternative to be deleted

5.6 CP5: Insert Configuration Context Condition of a Configuration Alternative

Description: A context condition related to a configuration alternative of a configurable region is added to define when the configuration alternative shall be selected.

Problem addressed: A context condition is added to a configurable process model to specify the condition under which a particular configuration alternative shall be selected.

Example: A passenger who carries luggage exceeding 20kg must pay an extra fee (where luggage exceeding 20kg refers to the new context condition).

Implementation in C-EPC:
Since the configuration context conditions are included in a separate questionnaire model, CP5 is realized by adding a new fact to the question referred to the application context of the condition (cf. Figure 5.9). If the questionnaire model does not include a question for the specific application context, a new question should be added. In addition, if the new fact implies new constraints, these should be included as well.

Parameters:
– the context condition to be inserted

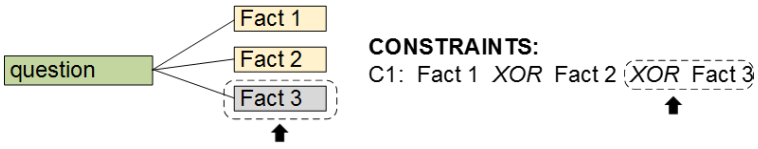


Figure 5.9: CP5 implemented in C-EPC

5.7 CP6: Delete Configuration Context Condition of a Configuration Alternative

Description: A context condition related to a configuration alternative of a configurable region of a configurable process model is deleted.

Problem addressed: A configuration context condition is no longer needed for selecting a configuration alternative in a configurable region and thus it is deleted.

Example: VIP passengers do not have to pay a fee for luggage overweight so far. However, the airline decides that from now on all passengers must pay such fee.

Implementation in C-EPC:
Since the configuration context conditions are included in a separate questionnaire model, CP6 is realized by removing an existing fact from the question referred to the application context of the condition (cf. Figure 5.10). If the question of the removed fact is not used by any other configuration alternatives, the question should be removed as well. In addition, the constraints referred to the removed fact should be removed as well.

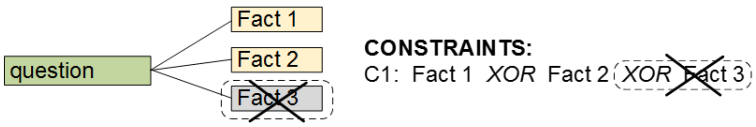


Figure 5.10: CP6 implemented in C-EPC

Parameters:
– the context condition to be deleted

5.8 CP7: Modify Configuration Context Condition of a Configuration Alternative

Description: A context condition related to a configuration alternative of a configurable region of a configurable process model is modified.

Problem addressed: A context condition is no longer adequate and shall be modified in the configurable process model.

Example: The payment of an extra fee is required when luggage weight exceeds over 20kg. Due to new business goals, this is changed and the extra fee is only required when the luggage weights more than 15kg.

Implementation in C-EPC:
Since the configuration context conditions are included in a separate questionnaire model, CP7 is realized by modifying the fact or the constraint referred to the application context of the alternative (cf. Figure 5.11).

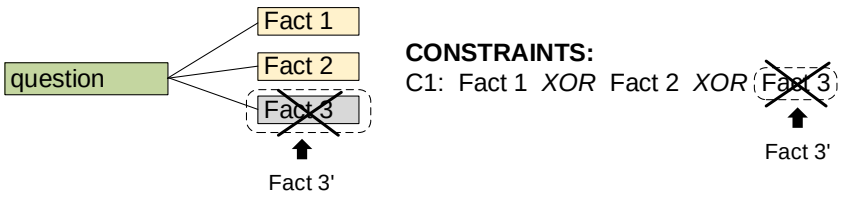


Figure 5.11: CP7 implemented in C-EPC

Parameters:
– the context condition to be modified

5.9 CP8: Insert Configuration Constraint Between Configuration Alternatives

Description: A constraint regarding the selection of configuration alternatives from one or more configurable regions is added to the configurable process model.

Problem addressed: The selection of configuration alternatives can only be done under certain conditions.

Example: When unaccompanied minors are travelling, extra staff is required to accompany them to the boarding gate, i.e., an inclusion constraint exists.

Implementation in C-EPC:
CP8 is realized by inserting a configuration requirement, which is then linked to the involved configurable nodes (either configurable functions or connectors) that delimit the configurable region of the configuration alternatives to be constrained (cf. Figure 5.12).

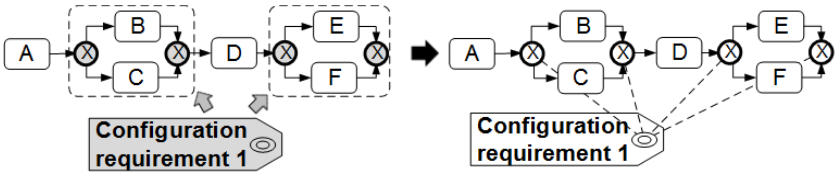


Figure 5.12: CP8 implemented in C-EPC

Parameters:
– the configuration region to which the alternatives whose selection will be constrained
– the configuration constraint to be inserted

5.10 CP9: Delete Configuration Constraint Between Configuration Alternatives

Description: A constraint between two or more configuration alternatives from one or more configurable regions is deleted.

Problem addressed: A constraint between two or more configuration alternatives is no longer needed and thus it is deleted.

Example: When unaccompanied minors are travelling, extra staff is required to accompany them to the boarding gate (i.e., inclusion constraint). Due to emerging legal regulations, from now on their relatives shall accompany them, i.e., the inclusion constraint is no longer needed.

Implementation in C-EPC:
CP9 is realized by deleting a configuration requirement, which is linked to the configurable nodes that delimit the configurable region of the configuration alternatives to be constrained (cf. Figure 5.13).

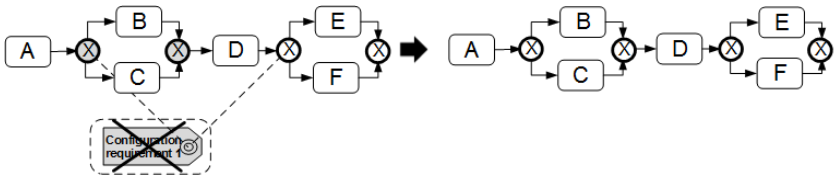


Figure 5.13: CP9 implemented in C-EPC

Parameters:
– the configuration constraint to be deleted

5.11 CP10: Modify Configurable Region Resolution Time

Description: In a configurable process model, the resolution time of a configurable region is modified.

Problem addressed: : The resolution time of a configurable region in a configurable process model is no longer adequate and is modified.

Example: Passengers travelling to US should fill in the ESTA form. However, due to new regulations, if the passenger already travelled to the US in smaller period than six months with the same airline, the latter may decide that the ESTA form is not needed again. This means that the activation of the activity "Fill in ESTA form" depends on the passenger and the airline (i.e., activity "Fill in ESTA form" becomes optional).

Implementation in C-EPC:
Since resolution time is only supported by the optionality of configurable functions (i.e., OPT configuration), CP10 is implemented by modifying to OPT the configuration requirements that restricts the configuration of the configurable function; e.g., from ON to OPT (cf. Figure 5.14). The constraints of the questionnaire model referred to the function which resolution time has been changed should be adapted accordingly.

Parameters:
– the configurable region whose resolution time is modified

5.12 Discussion

In absence of an established method to deal with process variability in an explicit manner, we provide a set of change patterns that enable the modeling and evolution of process families. In addition, these patterns

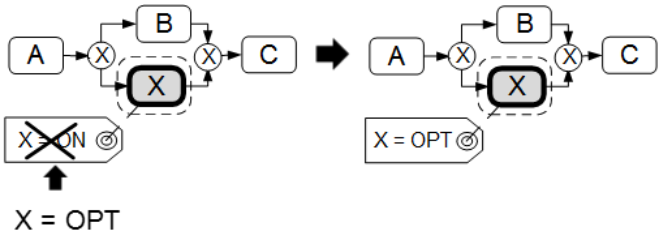


Figure 5.14: CP10 implemented in C-EPC

are intended to reduce the effort needed for such purposes and ensure process family correctness. In this context, in the following we discuss the completeness, generalizability and application order of our patterns.

Regarding the completeness of the proposed patterns, we ground the patterns on a set of variability-specific language constructs obtained from a large-scale systematic literature review. As a consequence of this methodological choice, our patterns describe how process variability is supported in the literature. Therefore, CP4PF is complete with regards to their support for such constructs. We have not specified patterns for modifying configurable regions, configuration alternatives, and configuration constraints because they can be realized by combining other change patterns and existing adaptation patterns. However, CP7 and CP10, which correspond to the combination of other patterns for some approaches (e.g., Provop, cf. Section 4.3.2), have been defined in order to modify in a more efficient way the variability-specific language constructs that they affect (i.e., configuration context condition and configurable region resolution time). As a consequence, our pattern set is not minimal.

Regarding the generalizability of CP4PF, our systematic review identified the variability-specific language constructs in 34 different approaches for process family management. Although these approaches use different terminology (e.g., *configurable region* vs. *variation point*)

and may realize the language constructs in different ways (cf. Section 4.2.3), the five languages constructs are widely used in the literature on process variability. Using this set of constructs as a basis, we can ensure that the proposed patterns are expressive enough to model and evolve process families in such approaches. That is, CP4PF can be implemented in any of these approaches for the constructs that they support.

When modeling with only CP4PF, the application order of the patterns is implicitly determined by the type of operation (insertion, modification, and deletion) and the constructs of each pattern. For example, configuration alternatives cannot be inserted if configurable regions have not been inserted previously. The same happens when inserting configuration context conditions and constraints, they need the previous insertion of configuration alternatives. Figure 5.15 shows the application dependence graph between the patterns.

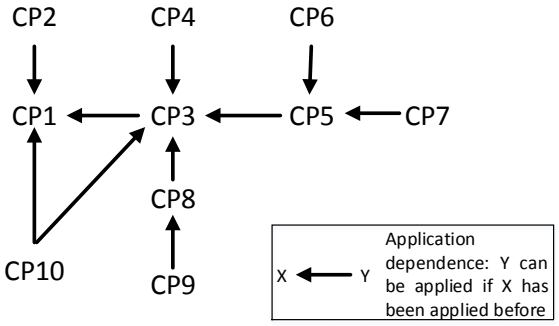


Figure 5.15: Application dependence graph between CP4PF

5.13 Conclusions

This chapter introduces 10 patterns for modeling (representing) and evolving process families. Our *change patterns for process families*

(CP4PF) have been derived from the five variability-specific constructs used for representing process variability. To show that our patterns–despite their intended generic nature–are specific enough to manage process variability, we have presented their implementation in C-EPC, a well-known process variability approach. This implementation allows us to show that the proposed patterns support process variability management and can ensure process family correctness by providing systematic means for introducing and deleting modeling elements. Finally, CP4PF are intended to reduce the effort needed for modeling and evolving configurable process models by automatically introducing modeling elements. CP4PF and its implementation allow us to answer research question 2 and 3 of this thesis (cf. Section 1.2). In the following, we validate our patterns in two ways: (1) by conducting a case study with a safety standard as a feasibility and effort reduction proof, and (2) by studying how users apply the patterns and their perceptions of this application.

6

Putting CP4PF into practice

In the previous chapter, we have defined a set of 10 change patterns for modeling process families (CP4PF). In this chapter, we put CP4PF into practice in a real and industrial scenario. In particular, we report on a case study conducted with a safety standard for validating CP4PF. We have selected this empirical method because it is a well-established and widely accepted approach for studying phenomena in their real life context [Robson, 2002], including for software and systems engineering research [Shull et al., 2008; Runeson & Höst, 2009]. Case studies are usually classified as flexible research and aim to provide new knowledge from and about actual situations. Their conclusions are based on evidence collected in a planned and consistent manner. We followed the guidelines and procedures proposed in [Runeson & Höst, 2009].

The case study allows us to validate the *feasibility* of our proposed CP4PF in a real scenario as well as analyzing the *effort* of applying them. Case studies with similar or the same purposes (i.e., feasibility

and effort analyses) can be easily found in the literature on e.g. business process modeling [Moreno-Montes de Oca et al., 2015; Ayora et al., 2015] and system assurance modeling [Panesar-Walawege et al., 2013; Nair et al., 2014].

The rest of the chapter is organized as follows. Section 6.1 describes the context of the case study. In turn, Section 6.2 presents the set of research questions, while Section 6.3 describes the case selection and the data collection procedure. Section 6.4 focuses on the case study results and Section 6.5 presents a discussion of these results. In addition, Section 6.6 discusses the validity of our case study. Finally, Section 6.7 concludes the chapter.

6.1 Context

A *safety–critical system* is one whose failure can lead to injury or death to people or damage to the environment [de la Vara & Panesar-Walawege, 2013]. These systems are subject to rigorous safety assurance and assessment processes as a way to ensure that they do not pose undue risks. These processes are usually performed in accordance with *safety standards*, whose compliance with must be proved. Although it is common that safety standards are sector-specific (e.g., for aerospace), some are generic and thus applicable to a wide range of systems. An example of this type of standards is IEC 61508 [IEC, 2010], which deals with functional safety of electrical, electronic, and programmable electronic systems. IEC 61508 has also been used as basis for sector-specific standards in the automotive, nuclear energy, process automation, and railway sectors.

Typically, safety standards indicate the requirements to fulfill, the artifacts to create, and the activities to execute in a safety-critical system's lifecycle. Standards also recommend *techniques* that might be used during the lifecycle. These techniques represent alternative ways to reach the standards' objectives. There are standards and parts of standards that focus on software safety aspects (e.g., IEC 61508-3, which is the third part of the IEC 61508). Figure 6.1 shows an example of the

techniques recommended in a software safety standard. The content of the figure is based on the software architecture design phase of the IEC 61508-3.

	Technique	SIL 1	SIL 2	SIL 3	SIL 4
1	Dynamic reconfiguration	--	--	NR	NR
2a	Structured methods	--	R	HR	HR
2b	Formal methods	--	R	HR	HR
3	Modular approach	HR	HR	HR	HR

Figure 6.1: Example of techniques extracted from a safety standard

Safety standards indicate when a technique should be used. For example, in IEC 61508 the techniques are assigned to *safety integrity levels* (SIL), which represent the relative level of risk reduction for a function or system. IEC 61508 defines four SILs, being SIL 1 the lowest risk-reduction and SIL 4 the highest one. In addition, the standard also provides a *recommendation* regarding the use of a technique for a specific SIL: *highly recommended* (HR), *recommended* (R), *no recommendation* for or against being used (- -), and *not recommended* (NR). For example, in Figure 6.1 the technique *Dynamic reconfiguration* has no recommendation for SIL 1 and SIL 2, while it is not recommended for SIL 3 and SIL 4. As a rule of thumb, the use of HR techniques is compulsory and NR techniques must not be used. Finally, IEC 61508 indicates the alternative use of some techniques to specify that only one of them might be used (e.g., techniques 2a and 2b in Figure 6.1 are alternative techniques since only one of them can be used).

In real practice, the variability associated to safety standard compliance (hereafter referred to as variability of a safety standard) is high and complex. For example, IEC 61508-3 recommends around 150 techniques, which in combination with the number of SILs, the given recommendations, and the existence of alternative techniques, result in thousands of possible ways of complying with the standard. For example, the technique *Structured methods* has three different recommendations in Figure 6.1. In addition, *Structured methods* is only used if the tech-

nique *Formal methods* is not used (i.e., alternative techniques). These variations even increase if, for example, it is considered that some systems might not use a HR technique because of the specific characteristics of the system (e.g., code automatically generated). Safety standards do not and cannot provide a unique algorithm for combining the techniques that will be correct for any application of the standard [IEC, 2010].

In order to facilitate their understanding, application, and compliance, safety standards can be represented with models [Nair et al., 2014]. Practitioners use process models [Nair et al., 2015] and some commercial tools for representing safety standards' processes with BPMN [ADONIS, 2008; Stages, 2014], including the modeling of techniques as BPMN activities [Atego, 2010]. In addition, specific models for safety assurance and certification have also been proposed during the last years. For example, *SafetyMet* is a metamodel for specifying how to comply with a safety standard [de la Vara & Panesar-Walawege, 2013]. Figure 6.2 shows an excerpt of this metamodel. In general terms, and using Figure 6.1 as a reference, the class *Reference Applicability* is used to represent a row of the table, and the class *Reference Technique* is used to specify a technique (e.g., *Structured methods*). The classes *Reference Criticality Level* and *Reference Applicability Level* are used to specify the SILs (SIL 1 - SIL 4) and the types of recommendation (e.g., HR), respectively, whereas the class *Reference Criticality Applicability* is used to link a technique with a recommendation for a specific SIL (e.g. technique *Structured methods* has a HR recommendation for SIL 4). The class *Reference Requirement* can be used to specify that some techniques are alternative to each other (e.g., techniques 2a and 2b in Figure 6.1).

Finally, safety standards are not static documents. New versions are usually released for including emerging techniques as well as making techniques adjustments; e.g., IEC 61508 has two versions: 1998 and 2010. This can lead, for instance, to the inclusion or removal of techniques or recommendation modifications. As a consequence, existing representations of the standards should be evolved in order to comply with these changes. For example, BPMN models and SafetyMet models representing IEC 61508:1998 should be evolved to comply with the 2010 version.

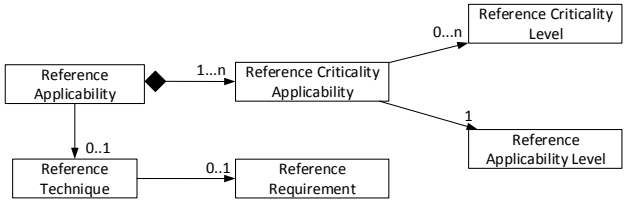


Figure 6.2: Excerpt of the SafetyMet metamodel (adapted from [de la Vara & Panesar-Walawege, 2013])

6.2 Research Questions

The goal of the case study was to analyze and validate the application of CP4PF in a real and industrial modeling scenario. For such a purpose, we focused on the variability of safety standards. In particular, we formulate the following research questions:

- RQ1. Is the application of CP4PF a *feasible approach* for modeling the variability of a safety standard?

This question refers to whether CP4PF can be used effectively for (1) creating a configurable process model representing the variability of a safety standard (e.g., when to use a technique) and (2) evolving this model in accordance to the changes introduced by a new version of the standard.

- RQ2. Does the application of CP4PF reduce the *effort* for modeling the variability of a safety standard?

This question is based on the level of effort spent throughout the creation and evolution of the configurable process model representing a safety standard when compared to state-of-the-art approaches. Effort is an important factor for determining if CP4PF can be successfully adopted in industry by practitioners. If the effort for modeling and

evolving a configurable process model using CP4PF is higher than the effort required with the approaches currently used (e.g., BPMN), then CP4PF adoption will be hindered.

6.3 Case Selection and Data Collection

The subject of the case study is the IEC 61508 standard. We chose IEC 61508 because it is a general standard that is applied in different sectors and for different systems. Among its parts, we chose part 3, which deals with software development. For answering **RQ1**, both the first (1998) and the second version (2010) were taken into account. More precisely, we used CP4PF for creating a configurable process model representing the variability of the IEC 61508-3:1998 and used them again for evolving the resulting model in order to comply with the 2010 version. We used C-EPC and applied therefore CP4PF using their implementation in this notation (cf. Chapter 5).

Regarding **RQ2**, we used the *number of operations* as the effort metric. The main advantages of operation measurement for effort analysis over, for instance, time measurement are that: (1) it allowed us to compare the modeling effort with different approaches without having to engage experts (in each approach) with similar levels of expertise and modeling skills; (2) finding experts that can spend the time necessary to create large models can be extremely difficult, especially if the experts must meet the above conditions; (3) it avoided threats to validity related to the modelers' fatigue in creating large models, and; (4) the results were more reliable since the operations could be measured twice and the outcome would be the same. In particular, we compared the number of operations needed with CP4PF with the number of operations needed to create and evolve a IEC 61508 model using two state-of-the-art approaches: BPMN and the SafetyMet metamodel. On the one hand, we chose BPMN because it is the *de-facto* standard for process modeling [BPMN, 2011] and it is used in industry for modeling safety standards [ADONIS, 2008; Stages, 2014]. On the other hand, we chose SafetyMet because it is a generic metamodel specifically targeted at representing

safety standards [de la Vara & Panesar-Walawege, 2013]. In addition, to determine the exact impact that CP4PF have in the representation of the standard in a configurable process model, we also compared the number of operations applying CP4PF with the number of operations needed to create and evolve the same configurable process model in C-EPC but without applying the patterns. In summary, we compared the number of operations using four approaches:

- Create and evolve a C-EPC model using CP4PF
- Create and evolve a C-EPC model adding/deleting/modifying language primitives (e.g., configurable connector)
- Create and evolve a BPMN model adding/deleting/modifying language primitives (e.g., activity)
- Create and evolve a SafetyMet model adding/deleting/modifying language primitives (e.g., Reference Technique)

Data collection involved two main activities. The first one consisted in creating the models representing the variability of the standard IEC 61508-3:1998 using the four approaches. In turn, the second activity consisted in evolving the four resulting models to comply with the 2010 version. For both activities, we measured the number of operations needed for creating/evolving the models. More precisely, we considered four types of actions:

1. *Insert a modeling element* (e.g., a link)
2. *Insert a named modeling element* (e.g., a BPMN activity) or *apply an insert pattern* (e.g., CP1)
3. *Delete a modeling element* (e.g., a link) or *apply a delete pattern* (e.g., CP2)
4. *Modify a modeling element* (e.g., rename a BPMN activity, rearrange a link) or *apply a modify pattern* (e.g., CP10)

We measured the first, third, and fourth type of action as one operation, and the second one as two operations (i.e., one operation for the insertion and another for writing a name). We made this difference in order to reflect that both inserting an element and naming it require a higher effort than only inserting, deleting, or modifying it. We considered that no distinction was necessary for the aspects common to all the types of actions (e.g., indication of element location, when either selecting an element for modification or deletion or specifying where to insert it).

Prior to data collection, we defined how the variability of the standard can be systematically represented with the selected approaches. Figure 6.3 shows the configurable process model of the techniques presented in Figure 6.1 with the C-EPC notation.[1]

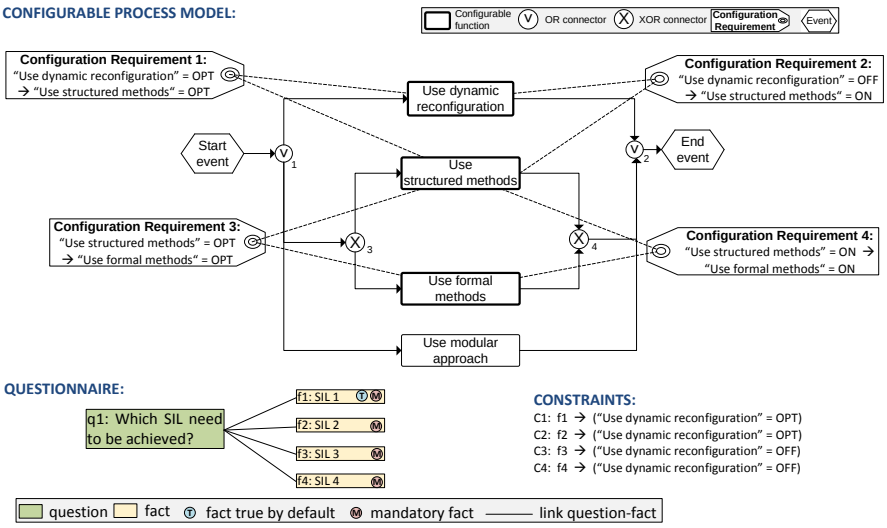


Figure 6.3: C-EPC model of the techniques of Figure 6.1 and the associated questionnaire and constraints

[1] The name of each technique has been adapted to the format "verb + object" in order to make them process-oriented [Mendling et al., 2010].

Techniques were represented depending on the provided recommendations. Techniques with two (or more) recommendations were defined as configurable functions, which can be configured to ON (i.e., the function is included), OFF (i.e., the function is not included), or OPT (i.e., the function is optionally included) depending on the SIL to achieve and the recommendation of use. Thus, the SILs corresponded to the application context of the configurable functions. For a given SIL, we considered that

- HR techniques must be used. Thus, the configurable function should be configured to ON.
- R techniques and techniques with no recommendation (- -) are optional. Thus, the configurable function should be configured to OPT.
- NR techniques must not be used. Thus, the configurable function is configured to OFF.

However, techniques with only one recommendation for all the SILs were represented as regular functions; i.e., no configurations were needed (e.g., function *Use modular approach* in Figure 6.3). The concrete configuration of each function for each SIL was defined in the configuration requirements. Since the application context in C-EPC is represented in a questionnaire model (cf. Figure 6.3), configuration requirements were defined based on previous configurations [La Rosa et al., 2009b]. For example, Configuration Requirement 3 states that the configurable function *Use formal methods* is configured based on the configuration of the function *Use structured methods*. Finally, we defined that alternative techniques (e.g., 2a and 2b of Figure 6.1) should be represented using XOR connectors (e.g., XOR 3 and 4 in Figure 6.3). We used OR connectors to represent that there is not predefined precedence order in the use of the techniques (e.g., OR 1 and 2 in Figure 6.3). In IEC 61508-3, the use of the techniques is provided in different tables, which are associated to specific lifecycle activities. For example, Figure 6.1 shows the techniques for the activity software architecture design.

Thus, for each table provided in the standard, we created a configurable process model for representing the variability of the corresponding lifecycle activity.

Figure 6.4 shows how the content of Figure 6.1 can be represented with BPMN. In line with the way of modeling presented in [Atego, 2010], we modeled each technique as an activity. In turn, we used XOR gateways to differentiate among the recommendations.

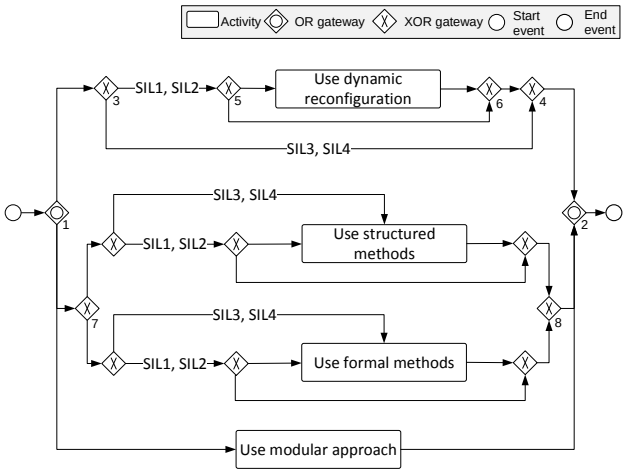


Figure 6.4: BPMN model of the techniques of Figure 6.1

For example, XOR 3 and 4 in Figure 6.4 are used to fork the sequence flow into two paths, one for each type of recommendation for the activity (i.e., technique) *Use dynamic reconfiguration*. SILs are then used as conditions of these gateways to decide which path should be taken. For example, the path of SIL 3 and 4 in XOR3 is taken when the technique is NR and thus must not be used. In the case of R techniques and techniques with no recommendation (- -), we modeled them using XOR gateways in order to represent that the techniques might be used or not (e.g., XOR gateways 5 and 6 in Figure 6.4). Like in C-EPC, alternative techniques are represented using XOR gateways (e.g., XOR 7 and 8 in Figure 6.4). In addition, we used OR gateways again to represent that

there is no precedence order in the use of the techniques (i.e., techniques can be applied in any order). We created a BPMN model for each table of the standard, as we did with C-EPC.

Finally, regarding the SafetyMet metamodel, we represented the variability of the IEC 61508 using the classes and relationships of the metamodel (cf. Figure 6.5).

During the data collection, the main author was the main responsible for systematically creating and evolving the models and measuring the operations. Nonetheless, she did not created the models completely alone. Dr. Jose Luis de la Vara iteratively validated the resulting models, as well as the measurement results. He has wide knowledge on safety assurance and certification (e.g., [Nair et al., 2014, 2015], and is co-creator of SafetyMet [de la Vara & Panesar-Walawege, 2013]. The advisors of this thesis also reviewed the collected data.

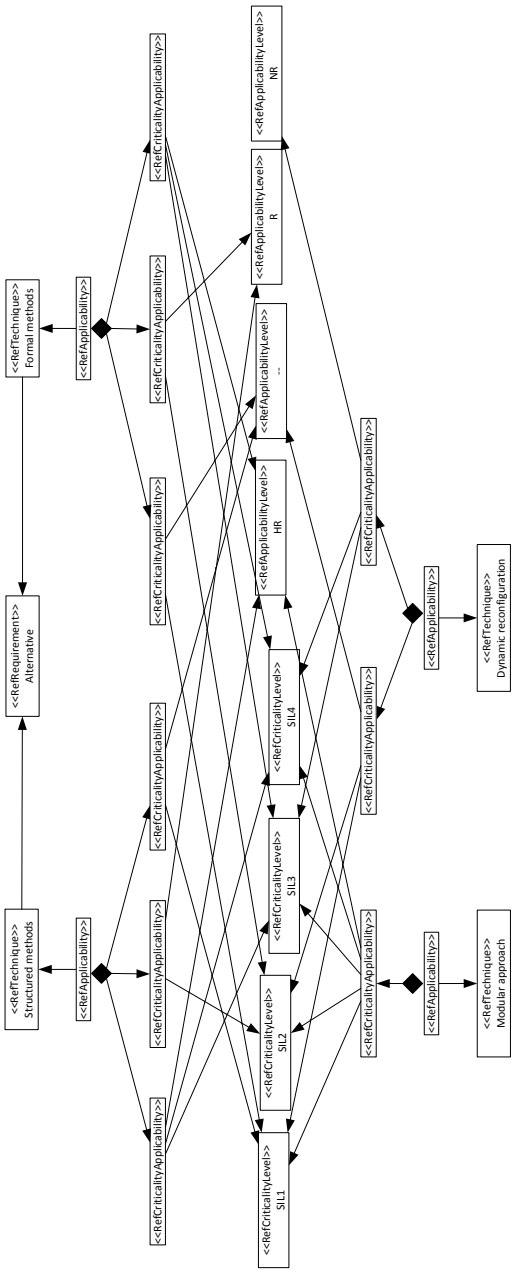


Figure 6.5: SafetyMet model of the techniques of Figure 6.1

6.4 Results

In this section, we describe the outcomes of the case study.

Results for Model Creation

Since the IEC 61508 documents are under copyright, we refrained from sharing the created models. We are not allowed to publish, for example, the content of their tables. However, for illustration purposes, we detail in Figure 6.6 the results obtained for representing the table presented in Figure 6.1 using the four approaches (i.e., the results for obtaining the models presented in Figures 6.3-6.5). Figure 6.6 shows the total number of modeling elements for the resulting models as a way to show their scale. In addition, it shows the number of actions needed for creating the models. Since we modeled from scratch, we mostly used insertions of modeling elements, insertions with name, and insert patterns. The total number of operations is the result of adding the number of inserted elements plus two times the number of insertions of named elements or the application of insert patterns. For example, for creating the configurable process model in C-EPC in combination with CP4PF, we needed 15 insertions of modeling elements (2+2+1+1+9), one insertion of a function, 11 applications of insert patterns (3+4+4), and four applications of modify patterns to change the constraints of the questionnaire model associated to the facts. This resulted in a total of 43 operations (15+1x2+11x2+4=43). Regarding the application of CP4PF, CP1 automatically introduces sequence flows and CP5 automatically introduces a question of the questionnaire and the respective constraints (cf. Chapter 5).

Figure 6.7 summarizes the results of the creation of the models representing IEC 61508-3:1998. A total of 119 techniques were represented. For creating the configurable process model with CP4PF, four patterns were used. We applied CP1 once per technique represented, CP5 once per existing SIL, CP8 for inserting the configuration requirements needed to specify the configuration of each configurable function, and CP7 for modifying the constraints of the questionnaire. In total,

Approach	Total of modeling elements	Type of actions				Total of operations
		1. Insert element	2. Insert named element / apply INSERT pattern	3. Delete modeling element / apply DELETE pattern	4. Modify modeling element / apply MODIFY pattern	
C-EPC + CP4PF *(model presented in Figure 6.3)*	47	- 2 XOR connectors - 2 OR connectors - 1 start event - 1 end event - 9 sequence flows	- 1 function - 3 app. of CP1 - 4 app. of CP5 - 4 app. of CP8	----------	- 4 app. of CP7	43
C-EPC *(model presented in Figure 6.3)*	47	- 2 XOR connectors - 2 OR connectors - 1 start event - 1 end event - 12 sequence flows - 8 requirement connectors - 4 links question/facts	- 1 function - 3 configurable functions - 4 configuration requirements - 1 question - 4 facts - 4 constraints	----------	- 4 modification constraints	68
BPMN *(model presented in Figure 6.4)*	48	- 12 XOR gateways - 2 OR gateways - 1 start event - 1 end event - 22 sequence flows	- 4 activities - 6 labeled sequence flows	----------	----------	58
SafetyMet *(model presented in Figure 6.5)*	57	- 4 Reference Applicability - 9 Reference Criticality Applicability - 31 links	- 4 Reference Technique - 1 Reference Requirement - 4 Reference Criticality Level - 4 Reference Applicability Level	----------	----------	70

Figure 6.6: Summary of the results for representing the table of Figure 6.1

we needed 295 pattern applications.

Results for Model Evolution

To evolve IEC 61508-3:198 into IEC 61508-3:2010, we first analyzed the difference between both versions. Figure 6.8 shows these differences. Then, we evolved the created models in order to represent these changes.

Figure 6.9 summarizes the results of this evolution. More precisely, it shows the number of the elements of the evolved models as well as the operations needed for the evolution. With CP4PF, we applied four patterns, resulting in a total number of 181 pattern applications. More precisely, we applied CP1 once per new technique with two different recommendations, CP2 once per technique with two different recommendations deleted, CP8 for adding the configuration constraints of the techniques introduced, and CP9 for modifying the constraints in the questionnaire model.

Approach	Total of modeling elements	Type of actions				Total of operations
		1. Insert element	2. Insert named element / apply INSERT pattern	3. Delete modeling element / apply DELETE pattern	4. Modify modeling element / apply MODIFY pattern	
C-EPC + CP4PF	841	- 62 XOR connectors - 38 OR connectors - 19 start event - 19 end event - 288 sequence flows	- 46 functions - 73 app. of CP1 - 4 app. of CP5 - 70 app. of CP8	----------	- 148 app. of CP7	960
C-EPC	841	- 62 XOR connectors - 38 OR connectors - 19 start event - 19 end event - 361 sequence flows - 140 requirement connectors - 4 links question/facts	- 46 functions - 73 configurable functions - 70 configuration requirements - 1 question - 4 facts - 4 constraints	----------	- 148 constraint modifications	1187
BPMN	1203	- 282 XOR gateways - 38 OR gateways - 19 start event - 19 end event - 582 unlabeled sequence flows	- 119 activities - 144 labeled sequence flows	----------	----------	1466
SafetyMet	1305	- 119 Reference Applicability - 216 Reference Criticality Applicability - 835 links	- 119 Reference Technique - 8 Reference Requirement - 4 Reference Criticality Level - 4 Reference Applicability Level	----------	----------	1440

Figure 6.7: Summary of the results for creating the models representing the IEC 61508-3:1198 standard

Techniques introduced	63
Techniques deleted	30
Techniques renamed	17
Techniques with different recommendations	5

Figure 6.8: Differences between the IEC 61508-3:1998 and IEC 61508-3:2010 versions

Synthesis of the Results

Figure 6.10 synthesizes the results. As shown, in the creation of the models, the use of CP4PF with C-EPC reduces the number of operations in a 19.1% with respect to using only C-EPC, a 34.5% with respect to BPMN, and a 33.3%, with respect to SafetyMet. For evolving the models, the use of CP4PF in combination with C-EPC reduces the number of operations in a 25.1% with respect to only C-EPC, a 40.4% with respect to BPMN, and a 33.1%, with respect to SafetyMet.

Approach	Total of modeling elements	Type of actions				Total of operations
		1. Insert element	2. Insert named element / apply INSERT pattern	3. Delete modeling element / apply DELETE pattern	4. Modify modeling element / apply MODIFY pattern	
C-EPC + CP4PF	1137	- 44 XOR connectors - 140 sequence flows	- 18 functions - 45 app. of CP1 - 72 app. of CP8	- 20 app. of CP2 - 10 functions - 18 XOR connectors	- 44 app. of CP7 - 17 function rename	563
C-EPC	1137	- 44 XOR connectors - 185 sequence flows - 144 requirement connectors	- 18 functions - 45 configurable functions - 72 configuration requirements	- 20 configurable functions - 10 functions - 18 XOR connectors	- 44 constraint modifications - 17 function rename	752
BPMN	1558	- 181 XOR gateways - 325 unlabeled sequence flows	- 63 activities - 92 labeled sequence flows	- 30 activities - 82 XOR gateways	- 17 activity rename	945
SafetyMet	1701	- 63 Reference Applicability - 117 Reference Criticality Applicability - 443 links	- 63 Reference Technique - 7 Reference Requirement	- 30 Reference Technique - 30 Reference Applicability - 1 Reference Requirement	- 17 technique rename	841

Figure 6.9: Summary of the results for evolving the created models

Approach	Model creation			Model evolution		
	Number of modeling elements	Number of operations	% of effort reduction	Number of modeling elements	Number of operations	% of effort reduction
C-EPC + CP4PF	841	960	na	1137	563	na
C-EPC	841	1187	19.1%	1137	752	25.1%
BPMN	1203	1466	34.5%	1558	945	40.4%
SafetyMet	1305	1440	33.3%	1701	841	33.1%

Figure 6.10: Synthesis of the results of the case study

6.5 Discussion

In this section, we discuss the results of the case study focusing on answering the research questions.

Regarding **RQ1** (*Is the application of CP4PF a feasible approach for modeling the variability of a safety standard?*), we could successfully model and evolve a configurable process model for IEC 61508-3 using CP4PF. We needed 295 pattern applications for creating the configurable process model and 181 for evolving it. The main challenge

was to define how the information of the standard had to be represented with C-EPC. More precisely, we had to decide how to capture all the variability in a C-EPC configurable process model and at the same time how to ensure model understandability and accuracy. For example, we discussed how to properly represent the techniques and the associated recommendations in the most suitable manner (e.g., represent techniques as configurable functions). For evolving the configurable process model, we also had to determine the impact that the differences between both versions of the standard had on the already created configurable process model. For example, we needed to decide how to reflect in the configurable process model that a recommendation had changed for a specific SIL (e.g., adding new configuration requirements).

For modeling and evolving the configurable process model, we used a total of six change patterns (CP1, CP2, CP5, CP8, CP9, and CP10) out of the 10 defined. More precisely, we applied patterns referred to four variability-specific language constructs: configurable region, configuration context condition, configuration constraint, and configurable region resolution time. We did not apply patterns related to configuration alternatives (i.e., CP3 and CP4) because we decided to model techniques as configurable functions, which implicitly define the configuration alternatives (i.e., ON, OFF, OPT). In addition, we did not use patterns for deleting or modifying the application context (i.e., CP6 and CP7) because the application context (i.e., SILs) did not change between the versions of the standard.

Regarding **RQ2** (*Does the application of CP4PF reduce the effort for modeling the variability of a safety standard?*), we consider that the results show that the application of CP4PF can significantly reduce the effort for modeling the variability of a safety standard. When compared to state-of-the-art approaches (i.e., BPMN and SafetyMet metamodel), CP4PF in combination with C-EPC can reduce up to 34.5% the number of operations for creating a configurable process model of the variability of a safety standard, and up to 40.4% for evolving it. We acknowledge that CP4PF were used in combination with a variability-specific approach that deals with process variability in an explicitly manner (i.e., C-EPC). This might be advantageous, for example, in respect to BPMN,

since it was not conceived to explicitly deal with process variability. In addition, we need to consider that the SafetyMet metamodel was conceived for being compliant with any safety standard, supporting any kind of information they may include (e.g., guidelines and techniques explanations). This adds a set of extra modeling actions (e.g., inserting a *Reference Applicability* element) that need to be done but are not needed for representing the selected information of IEC61508-3. However, even when compared to modeling with only C-EPC, the application of the change pattern is clearly advantageous to us (over 19% reduction in the number of operations). In this sense, we consider that most of the benefit comes from using CP4PF, not from the process variability approach.

Finally, the results for RQ2 coincide with the benefits we envisioned when defining CP4PF. The case study allowed us to determine the actual extent to which the application of CP4PF can reduce the effort for modeling a process family in a real scenario. This is mainly due to the fact that CP4PF can automatically insert or delete several modeling elements with a single modeling action (i.e., pattern application).

6.6 Validity

Like any other modeling situations, modeling IEC 61508-3 comprised decisions about how to create the models (e.g., modeling gateways in pairs). In addition, only one author was the main responsible for creating the models. These both factors affect *internal validity*. To mitigate possible threats, we decided, prior to data collection, how to systematically represent the variability of the standard with the selected approaches. The obtained models were also validated.

Regarding *conclusion validity*, CP4PF were implemented and applied with only one process variability approach (C-EPC). The results of the case study and thus the conclusions drawn could differ when implementing CP4PF with other approaches (e.g., Provop [Hallerbach et al., 2010a]).

Single case studies as the one conducted (with only one safety stand-

ard) pose threats to *external validity*. However, we expect similar results for standards with similar characteristics (e.g., other safety standards, and especially those based on IEC 61508). We also believe that variability management of process families from other domains can benefit from CP4PF. Since the implementation of CP4PF with C-EPC automatically introduces or deletes modeling elements, we may expect an effort reduction when modeling and evolving other process families.

6.7 Conclusions

This chapter presents a case study conducted with the process family of a safety standard (i.e., IEC 61508-3). The case study results have allowed us to show the feasibility of the CP4PF and their suitability in terms of effort reduction. When compared to other state-of-the-art approaches, the defined patterns are able to reduce the effort needed for modeling a process family by 34% and for evolving it by 40%. Thus, the application of the change patterns can help in effectively modeling and evolving large and highly-variable process families. Their application can also considerably reduce variability management effort.

To the best of our knowledge, it is the largest case study that has been conducted so far on process variability management, the first one that has dealt with process family evolution, and the first one that has compared the results of different approaches. This case study helps us to answer research question 4 of this thesis (cf. Section 1.2). In the following, we complement the validation of the patterns by validating them with PAIS engineers. This allows us to analyze how patterns are applied in practice and how they are perceived by PAIS engineers.

7

Validation of the proposal with PAIS engineers

In the previous chapter, we have validated our change patterns (CP4PF) by showing their feasibility for modeling and evolving process families. In addition, we have shown that CP4PF are able to reduce the effort needed for such purposes. To complement this validation, this chapter describes the validation of our CP4PF with PAIS engineers in order to explore how CP4PF are used. That is, to study how PAIS engineers apply CP4PF, what is the impact of patterns application, and how PAIS engineers perceive pattern usefulness and ease of use.

The rest of the chapter is organized as follows. Section 7.1 shows the research questions of this validation. While Section 7.2 presents the selected subjects, Section 7.3 describes the validation design. In turn, Section 7.4 focuses on the data collection procedure, and Section 7.5 on the results of the validation. Section 7.6 focuses on the results

of this validation and Section 7.7 in their validity. Finally, Section 7.8 concludes the chapter.

7.1 Research Questions

The main goal of this validation was to explore how PAIS engineers experience the use of CP4PF. To this end, we focused on the impact of change patterns in the evolution of configurable process models. We formulate the following research questions:

- RQ1. What is the *user experience* when using CP4PF for evolving configurable process models?

This question aims to analyze user's behaviors and attitudes about using CP4PF when evolving configurable process models.

- RQ2. What is the impact of CP4PF usage on the *perceived mental effort* (of PAIS engineers) for evolving configurable process models?

Evolving configurable process models requires a certain mental effort that can depend, for example, on the assignment, the own experience, and the level of difficulty. This question allows us to explore whether the use of CP4PF increments the mental effort for evolving configurable process models.

- RQ3. What is the impact of CP4PF usage on the *time* that PAIS engineers need to evolve configurable process models?

Since CP4PF are intended to speed up the evolution of configurable process models, this question focuses on determining if the use of CP4PF allow PAIS engineers to reduce the duration of this activity.

- RQ4. What is the PAIS engineers' *perception* about CP4PF usage for evolving configurable process models?

This question studies whether CP4PF are easy to learn, are clear and understandable, and can enhance productivity and effectiveness.

7.2 Subject Selection

We used *convenience sampling* [Wohlin et al., 2000] for subject selection, and contacted both researchers and practitioners. We selected four researchers from the *Centro de Investigación en Métodos de Producción de Software*[1] (S1-S4) and four practitioners from different software development and consulting companies (S5-S8). The researchers had knowledge about process modeling and the practitioners had experience in dealing with process models in real scenarios. Although the selected subjects were not experts in process families, they were convenient for the purpose of this validation because they had to deal with process modeling regularly. As described below, subjects received basic training on process families in order to ensure that they had a homogeneous level of knowledge in process families (cf. Appendix C).

7.3 Validation Design

We asked subjects to evolve two configurable process models with and without CP4PF. We compared two *factors*: (1) the configurable process models to be evolved and (2) the approach used (i.e., with and without CP4PF). For the first factor, two configurable process models were used: a check-in process at an airport and a medical examination process at a hospital. The complete description of the models can be found in Appendix C.2. We chose these models because they are simple, understandable, realistic, and have enough variability for the application of CP4PF (cf. Example 1). This way, subjects would more easily focus on the use of the patterns instead of being affected by the complexity of the selected models. For the second factor, we decided to extend BPMN with C-EPC constructs because all the subjects were familiar

[1] The researchers were independent and external to our work in process families.

with BPMN, but not with EPC or C-EPC. That is, we enable subjects to model configurable tasks and configurable gateways with BPMN in order to apply CP4PF. This resulted in two approaches: C-EPC-like BPMN in combination with CP4PF and C-EPC-like BPMN without CP4PF. By selecting two simple models and only two approaches, we mitigated the *boredom effect* preventing subjects of becoming tired and putting less effort and interest into the validation.

Subjects were distributed into two balanced groups: G1 and G2. For ensuring that both groups were homogeneous, we asked subjects to fill out a demographic survey in advance in order to determine their experience regarding the topics of the validation, i.e., process modeling, process families, and adaptation patterns. Appendix C.1 presents the questions of this survey. The results of the survey are shown in Figure 7.1. S1, S3, S5, and S8 were assigned to G1, and S2, S4, S6, and S9 to G2, to avoid great differences between the groups (cf. Figure 7.2). To mitigate the *learning and tiredness effects*, G1 first evolved the configurable process model referred to the airport check-in without CP4PF, and G2 first evolved the same configurable process model with CP4PF. Secondly, G1 evolved the configurable proces model referred to medical examinations with CP4PF, and G2 without them.

		1. Years modeling business processes	2. Years modeling with BPMN	3. BPMN models analyzed within last 12 months	4. BPMN models created within last 12 months	5. Average of activities of these models	6. Previous experience with adaptation patterns	7. Previous experience with process families
Acad.	S1	> 5	> 5	60 - 120	< 60	10 - 50	Yes	Yes
	S2	3 - 5	3 - 5	< 60	< 60	10 - 50	Yes	Yes
	S3	3 - 5	3 - 5	Less than 60	< 60	> 100	Yes	No
	S4	3 - 5	3 - 5	< 60	< 60	10 - 50	Yes	Yes
Pract.	S5	1 - 3	1 - 3	< 60	< 60	10 - 50	No	Yes
	S6	< 1	< 1	< 60	< 60	10 - 50	Yes	No
	S7	1 - 3	1 - 3	120 - 180	120 - 180	10 - 50	No	No
	S8	< 1	< 1	< 60	< 60	< 10	Yes	Yes

Figure 7.1: Subject's demographics

Distribution of subjects

G1: S1, S3, S5, S8

G2: S2, S4, S6, S7

	Group	Min.	Max.	Med.
1. Years modeling BPs	G1	< 1	> 5	1 - 3
	G2	< 1	3 - 5	1 - 3
2. Years modeling with BPMN	G1	< 1	> 5	1 - 3
	G2	< 1	3 - 5	1 - 3
4. BPMN models analyzed within last 12 months	G1	< 60	60 - 120	< 60
	G2	< 60	120 - 180	< 60
5. BPMN models created within last 12 months	G1	< 60	< 60	< 60
	G2	< 60	120 - 180	< 60
6. Average of activities of these models	G1	< 10	> 100	10 - 50
	G2	10 - 50	10 - 50	10 - 50

	Group	Yes	No
3. Certification in BPMN	G1	0	4
	G2	0	4
7. Previous experience with adaptation patterns	G1	3	1
	G2	3	1
8. Previous experience with process families	G1	3	1
	G2	2	2

Figure 7.2: Distribution of subjects and summary of the demographic results for G1 and G2

7.4 Data Collection Procedure

In order to apply CP4PF, we implemented a modeling editor based on the *Cheetah Experimental Platform* (CEP) [Pinggera et al., 2010]. This platform is used in experimental research on business process modeling. Our editor implemented CP4PF in C-EPC-like BPMN and allows applying the patterns to a configurable process model. Additionally, the implemented editor allowed us to automatically record subjects' modeling actions in order to replay step by step what subjects did when using the change patterns. The complete description of CEP can be found in Appendix D.

The procedure followed for data collection is presented in Figure 7.3. First, we introduced the context of the validation to the subjects. Then, subjects were guided through three tutorials in order to ensure

a homogeneous level of knowledge and that subjects were sufficiently literate in process modeling, process families, and CP4PF. Afterwards, subjects were asked to perform two guided familiarization tasks, one for the insert patterns and one for the delete patterns. These tasks aimed to ensure that the subjects obtained practical experience and familiarity using CP4PF and CEP.

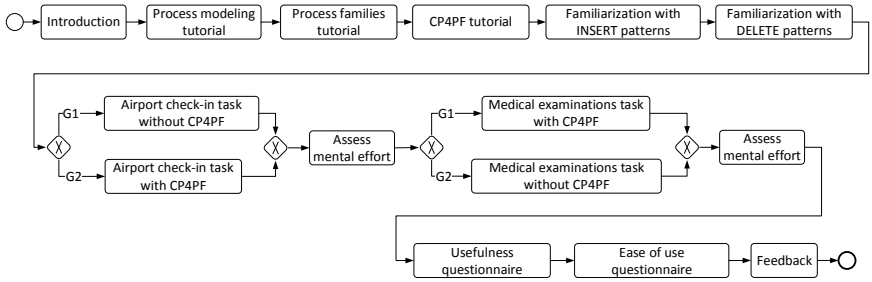


Figure 7.3: Data collection procedure

Afterwards, the subjects evolved two configurable process models (i.e., airport check-in and medical examinations) with and without CP4PF. For each model, the subjects received a source configurable process model (i.e., the one they needed to evolve), a target configurable process model (i.e., the one they needed to obtain). The subjects create the target configurable process model from the source. By giving both models, we tried to simulate the modeling situation in which an existing model needs to be adapted to a new version (e.g., adapting models of IEC 61508:1998 to IEC 61508:2010, or adapting CMMI level 2 models to level 3 [CMMI, 2010]). In addition, since subjects had the target solution available, the challenge lied in determining the set of patterns needed for evolving the source model and in combining them effectively. This allowed the subjects to analyze the impact of CP4PF when evolving a configurable process model.

The evolution tasks were executed individually (i.e., subjects perform them one by one). In addition, while performing this tasks, we

asked the subjects to think out loud in order to record their thoughts [Ericsson & Simon, 1980]. This allowed us to study the subjects' perspective and experience while applying CP4PF. Then, we transcribed the records and annotated each line using a coding scheme (cf. Figure 7.4). For such purpose, we first created an initial list of codes that were used to categorize the transcriptions. If new codes were identified while transcribing the records, they were added to the list or merged to the already existing ones. Five types of codes were identified: errors (e.g., wrong change pattern application), challenges (e.g., doubts), general comments (e.g., benefits and improvement suggestions), modeling strategies (e.g., control flow first), and knowledge (e.g., knowledge acquired "on the fly"). The detailed codes are presented in Figure 7.5. We also used the step-by-step replay (recorded by the editor) in order to check what the subjects were modeling when expressing some thought. The first author coded the transcriptions and the second author validated the coding. This data helped us in answering RQ1.

Content	Code
"I don't remember what needs to be selected for applying CP8"	Challenge
"I prefer to manage the control flow first"	Modeling strategy
"No, it doesn't work like this"	Error
"Applying CP3 is very similar to applying CP1 "	Knowledge
"The automatic layout is terrible"	General comment

Figure 7.4: Excerpt of the transcriptions

After finishing the evolution tasks, we asked the subjects to assess the mental effort they needed to accomplish each task. This mental effort was measured using a 7-point Likert scale ranging from *Very low* over *Neutral* to *Very high*. These answers helped us answer RQ2. In addition, we automatically stored the duration (in minutes) that the subjects needed to accomplish each task, which allowed us to answer RQ3.

Once configurable process models had been evolved, the subjects

answered two questionnaires about the *perceived usefulness* and the *perceived ease of use* [Davis, 1989] of the CP4PF used. The *perceived usefulness* questionnaire comprised six questions about whether CP4PF would facilitate a subject's job, if CP4PF are useful, and whether they would enhance a subject's job performance, productivity, and effectiveness. The *perceived ease of use* questionnaire included six questions concerning a subject's learning attitude towards CP4PF, as well as their flexibility and simplicity. Both questionnaires were based on the *Technology Acceptance Model* [Davis, 1989] and measured with 7-point *Likert scales* to evaluate the subject's own extent of agreement or disagreement (i.e., from *Strongly Agree* over *Neutral* to *Strongly Disagree*). These questionnaires allowed us to answer RQ4. Finally, we asked subjects to provide qualitative feedback.

7.5 Results

This section summarizes the results of the validation with PAIS engineers.[2]

Regarding subjects experience the use of CP4PF (RQ1), a total of 8 transcriptions were produced (one transcription per subject) with a 573 lines on average. Figure 7.5 summarizes the results of the transcription coding. For each code, we provide the total number of occurrences and the number of occurrences per subject. The most frequent codes are G1 and K1 referred to the perceived benefits of CP4PF and the knowledge that subjects acquire while applying them (e.g., *Ah! This pattern in applied like this. I didn't know before*), respectively. Moreover, all subjects referred to this code (K1) and most subjects to C1 (doubting about what elements should be selected for applying a pattern), G1, and S1 (focusing firstly on the control flow elements). On the contrary, only two subjects incur in E1 (incorrect pattern application), C2 (doubting what pattern should be applied), and K2 (stating the previous knowledge needed to apply a pattern).

[2]Further details on the results can be found in http://www.pros.upv.es/bpvar/evaluation/ValidationUsers.rar

	Codes	S1	S2	S3	S4	S5	S6	S7	S8	Total
Errors	*E1. Incorrect pattern application (e.g., "Ah! This pattern wasn't the correct one")*	0	0	0	0	0	3	1	0	4
Challenges	*C1. The subject doubts about what elements should be selected for applying a pattern (e.g., "I don't know what elements need to be selected")*	0	2	1	0	2	1	3	0	9
	C2. The subject doubts about what pattern should be applied (e.g., "What pattern do I need to apply for inserting a requirement?")	0	0	0	0	3	2	0	0	5
General comments	*G1. The subject expresses a benefit about the patterns (e.g., "This is much quicker with the patterns")*	5	0	5	0	5	1	7	1	24
	G2. The subject expresses an improvement for the patterns (e.g. "It would be great to apply two patterns in a row")	3	0	1	0	0	2	0	0	6
	G3. The subject complains about the automatic layouting (e.g., "I don't like the layout")	0	5	0	2	0	6	0	0	13
Modeling strategies	*S1. Control flow first (e.g., "I prefer to focus first on the control flow elements")*	9	1	2	1	0	0	1	1	15
Knowledge	*K1. Knowledge acquired while modeling (e.g., "Ah! This pattern is applied like this")*	1	2	6	2	3	2	4	2	22
	K2. Required knowledge to apply a pattern (e.g., "This can't be done if you don't know what a configurable node is")	0	2	0	0	0	2	0	0	4

Figure 7.5: Results of the transcription coding (RQ1)

Figure 7.6 shows the results concerning the perceived mental effort for evolving configurable process models with and without CP4PF (RQ2). As shown, there is a difference between the minimum value when not using CP4PF (i.e., *Low*) and the maximum (i.e., *High*). On the contrary, when using the patterns this difference is lower (i.e., the minimum value is *Very Low* and the maximum is *Medium*). The medians are *Low-Rather Low* for not using CP4PF and *Low* for using them. Only one subject reported higher mental effort when using the patterns, and four reported lower mental effort.

Figure 7.7 shows the results of the time (in minutes) that the subjects needed to accomplish the evolution of configurable process models with and without CP4PF (RQ3). The minimum value not using CP4PF is 3.78 minutes, the maximum value is 11.6, and the average is 7.15 minutes. When using CP4PF the minimum and maximum value are 4.32 and 10.05 minutes, respectively. The average time in this case

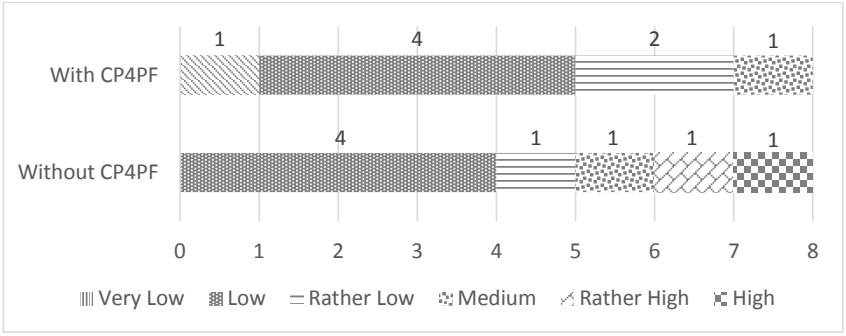


Figure 7.6: Results for the mental effort (RQ2)

is 7.24 minutes.

Figure 7.8 presents the results for the *perceived usefulness* questionnaire and Figure 7.9 presents the results for the *perceived ease of use* questionnaire (RQ4). Most subjects agreed or strongly agreed upon the usefulness and ease of use of CP4PF. The only statement for which some disagreement was indicated is *"I find CP4PF to be flexible to interact with"*, and all the subjects agreed or strongly agreed upon *"Learning to operate CP4PF is easy for me"*.

Finally, we obtained positive comments regarding CP4PF when asked for qualitative feedback. Most of the subject (six out of eight) indicated that CP4PF are very straightforward to understand and use. One subject mentioned that CP4PF could help her in avoiding modeling errors at her job and another stated that CP4PF could reduce modeling time and effort in her company. However, one subject complained about the troubles that she had had to evolve a configurable process model using CP4PF. She mentioned that although the patterns could be useful, she was more familiar with traditional modeling (i.e., without change patterns). Finally, two subjects explicitly complained about the automatic layout. They reported that this functionality, which rearranged

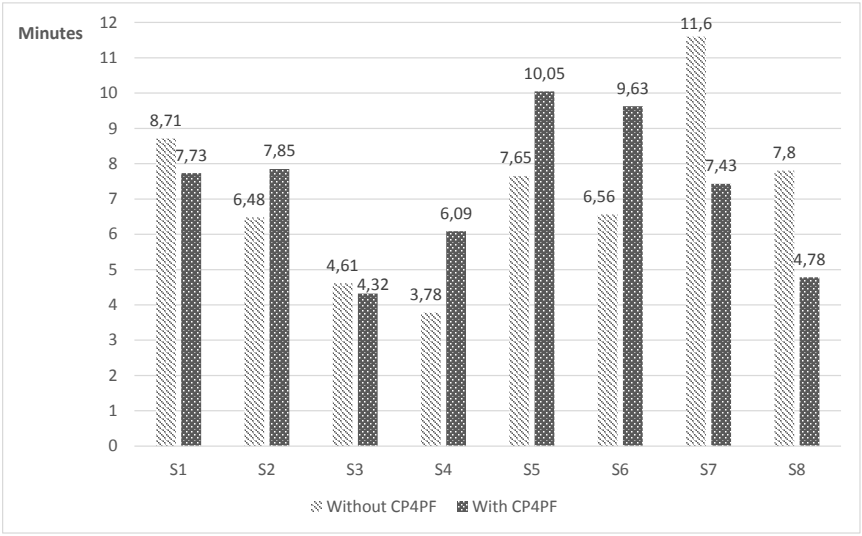


Figure 7.7: Results for the duration (RQ3)

all modeling elements after applying a change pattern, was not satisfactory at all.

7.6 Discussion

This section interprets the results of the validation and provides a general discussion.

First, regarding **RQ1** (*What is the user experience when using CP4PF for evolving configurable process models?*), the analysis of the transcriptions showed us that doubts arose while applying CP4PF. For example, five out of the eight subjects had troubles when deciding which elements should be selected for applying a change pattern. It was not clear to these subjects which elements they had to select to apply a specific pattern, especially the first time this pattern was used. In turn,

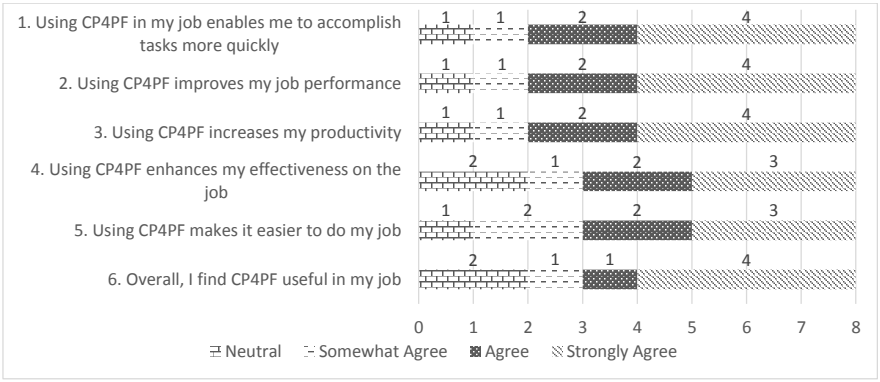


Figure 7.8: Results for the perceived usefulness (RQ4)

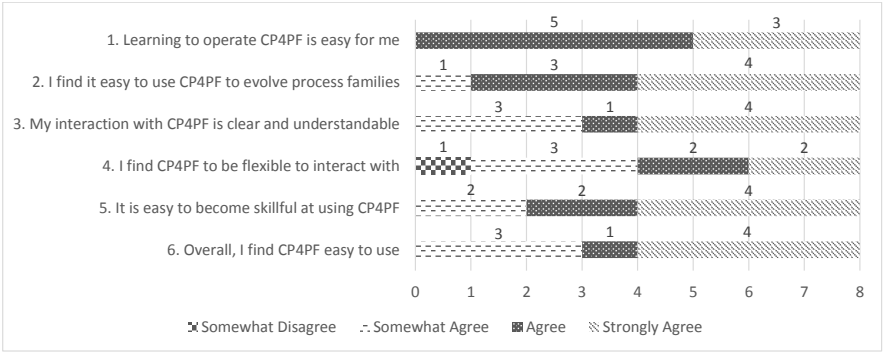


Figure 7.9: Results for the perceived ease of use (RQ4)

two subjects had doubts with respect to what pattern they needed to apply. We consider that these doubts are typical for the first use of a new approach. In addition, most of these doubts where solved "on the fly" while interacting with CP4PF, which suggest that the patterns are

intuitive for subjects.

The subjects also expressed benefits while using CP4PF, and in comparison with not using them (e.g., *"These patterns allow me to model quicker"*). We interpret this as a positive attitude of subjects for adopting CP4PF. In addition, the subjects provided improvement suggestions for the patterns. For example, one subject said that it would be much more useful to delete two configuration alternatives in a row when applying the CP4PF.[3] We consider that these comments indicate that subjects want to adapt CP4PF to the way they actually model. These type of comments are very valuable because they can help CP4PF for better fit subjects' expectations.

When analyzing the transcriptions, we observed that some subjects tended to focus first on the elements involved in the control flow of the configurable process model (e.g., configurable functions and connectors), and then on the attached elements (i.e., configuration requirements). We consider that this is related to the way subjects actually model process variability and may imply an order in the application of CP4PF. Although there is an implicit precedence in the application of the patterns (e.g., a configuration constraint–CP8–cannot be applied introduced configurable regions–C1–or configuration alternatives–CP3–have been introduced), and since most of the subjects preferred to deal with the control flow first, we think that the explicit definition of a methodology for applying the patterns would be beneficial. This methodology would provide guidelines for helping PAIS engineers to apply CP4PF correctly and consistently.

Second, regarding **RQ2** (*What is the impact of CP4PF usage on the perceived mental effort (of PAIS engineers) for evolving configurable process models*?), the results indicate that the use of CP4PF for evolving a configurable process model resulted in slightly lower *perceived mental effort*. This can be surprising because presumably the use of CP4PF may require a higher mental effort, especially if the patterns have not been used before [Weber et al., 2013]. These results may indicate that

[3]This pattern is defined for deleting only one alternative each time (cf. Chapter 5).

adopting CP4PF may be less costly than expected.

Third, regarding **RQ3** (*What is the impact of CP4PF usage on the time that PAIS engineers need to evolve configurable process models?*), the average time for evolving a configurable process model was slightly higher when using CP4PF. Our expectation was different, as CP4PF are intended to speed up the modeling and evolution of configurable process models. However, based on the transcriptions, this time difference could be a result of issues related to the automatic layout. Some subjects reorganized the modeling elements after each pattern was applied, thus performed additional actions when applying CP4PF. Indeed, if the three subjects that complained about the automatic layout (cf. Figure 7.5) while evolving the models are not taken into account, the application of the patterns resulted in a decrease of around a 17% in the average time, and only one subject needed more time for evolving a configurable process model with the patterns. Therefore, we are still confident that CP4PF can reduce the time needed for evolving a configurable process model. Further empirical studies will allow us to more strongly substantiate this claim, which is also already supported by the results of the case study.

Fourth, regarding **RQ4** (*What is the PAIS engineers' perception about CP4PF usage for evolving configurable process models?*), the subjects tended to perceive CP4PF as useful for evolving configurable process models, at least for the tasks performed. The results of the perceived usefulness questionnaire suggest that CP4PF can increase effectiveness when evolving a configurable process model. In addition, the subjects also perceived CP4PF easy to use. Based on the answers of the perceived ease of use questionnaire, we consider that, in general, subjects' attitude toward CP4PF is positive. Finally, the answers coincide with both the qualitative feedback that subjects provided and the analysis of the transcriptions we made. Except for one subject, no remarkable doubts or errors were detected.

7.7 Validity

A threat to *internal validity* is the size of the selected models. We selected two small and easily-understandable configurable process models from realistic scenarios (i.e., airport check-in and medical examinations) so that the subjects focused on applying CP4PF instead of on understanding the selected models. Using larger configurable process models may produce different results, but at the same time it may introduce other threats (e.g., increment of complexity).

The low number of subjects affects *conclusion validity* as well as the fact that the subjects worked only on model evolution. We performed this validation as an exploratory (pilot) study to better analyze how patterns are applied and what are the perceptions about them. Thus, this validation serve as a basis for future empirical studies. In addition, we have tried to make claims about CP4PF that are clearly based on the validation results.

Regarding *external validity*, the validation with PAIS engineers was exploratory. We did not aim to generalize its results but mainly to gain insights into how PAIS engineers apply CP4PF and their perceptions. Nonetheless, we selected subjects with enough background in process modeling, and with different experience levels to try to obtain an adequate sample. In addition, we divided subjects into two groups of similar average experience in the topics of the evaluation to reduce the impact of the experience level on the results. Although further experimentation is needed to better assess the extent to which the obtained results can be generalized, our validation with PAIS engineers serve as a basis for future empirical evaluations.

7.8 Conclusions

This chapter presents a validation with PAIS engineers in order to explore their experience and perception when applying the defined CP4PF. Although some drawbacks were found (e.g., automatic layout), the results show that PAIS enginners' attitude toward our patterns is positive.

This suggests that the defined patterns could be easily adopted and assist process engineers in managing process variability. In combination with the results of the case study, we can conclude that CP4PF can be used as an efficient solution for modeling and evolving configurable process models with a high degree of variability. This validation with PAIS engineers also helps us to answer research question 4 of this thesis (cf. Section 1.2).

8

Conclusions and Future Work

The present thesis deals with the enhancement of process variability modeling through the use of change patterns. On the one hand, we have identified the language constructs used to represent process variability by performing an systematic study on process variability. This has allowed us to characterize process variability and evaluate process variability approaches (cf. Chapter 4). On the other hand, we have defined an efficient solution (i.e., a set of change patterns) for modeling and evolving configurable process models (cf. Chapter 5-7). However, the research line in which the work of this thesis is by no means completed here. Further research can be performed in order to complement and extend this thesis.

This last chapter introduces the conclusions of the work developed and proposes new areas for future research. More precisely, Section 8.1 presents the main contributions of the thesis. Section 8.2 provides an overview of the publications that have been produced throughout the

development of the thesis. Further, Section 8.3 describes the research collaborations that have been done in the context of the thesis. Finally, Section 8.4 outlines the ongoing and future work that can extend the work developed in the thesis.

8.1 Contributions

The work of this thesis provides the following contributions:

- **Process variability characterization.** Through an in-depth study, we have systematically analyzed existing process variability approaches regarding their expressiveness with respect to process variability modeling as well as their support along the process lifecycle. In this context, we have identified, among others, a core set of *variability-specific language constructs* as well as a core set of *features* fostering process variability. Thus, we provide a complete characterization of process variability. In addition, based on the empirical evidence provided by this analysis, we have derived the **VIVACE framework**. VIVACE shall allow for the **systematic assessment and comparison** of existing process variability approaches and enables PAIS engineers to **select that variability approach** meeting their requirements best. Additionally, VIVACE helps them in dealing with PAISs supporting process variability.

- **A set of change patterns for modeling and evolving process families.** Based on the identified variability-specific language constructs, we have derived 10 change patterns for process families (CP4PF). These patterns enable the **modeling and evolution** of a configurable process model.

- **An implementation in C-EPC of the CP4PF**. Despite their intended generic nature, we have illustrated how the CP4PF can be realized and showed how they can be implemented in a well-established approach for modeling configurable process models (i.e., C-EPC). This implementation have allowed us to show that

the proposed patterns can **ensure process family correctness** by providing systematic means for introducing and deleting modeling elements.

- **Evidence of feasibility and effort reduction for variability management of real process families when applying the defined patterns.** Through a case study with a safety standard, we have shown that CP4PF are a **feasible** approach for modeling and evolving process families. In addition, we have determined that our CP4PF are able to **reduce the effort** needed for such purposes by 34% and by 40%, respectively.
- **Evidence of PAIS engineers' experience and perceptions when applying the defined patterns.** Complementing the case study, we have implemented the patterns in the *Cheetah Experimental Platform* and have studied how PAIS engineers apply CP4PF, what is the impact of pattern application, and how PAIS engineers perceive pattern usefulness and ease of use. The results show that most PAIS engineers expressed some **benefits when applying the CP4PF**, did not perceived an increase of mental effort for applying the patterns, and agreed or strongly agreed upon the **usefulness and ease of use** of the patterns.

8.2 Publications

The research activity presented in this thesis has produced innovative and different contributions that have been presented and discussed on different peer-review forums. In this section, we present the articles that have been produced in the context of this thesis.

8.2.1 Main publications

The research developed in this thesis has been published in different forums where relevant research about business processes is presented and discussed. In particular, three types of forums are included: international journals indexed in the *Journal Citation Reports* (JCR)

index [Thomson-Reuters, 2014], international conferences, and workshops. The position of the name of the author of this thesis is used as an indicator of the degree of contribution in each publication.

International Journals Indexed in the JCR:

- **Clara Ayora**, Victoria Torres, Barbara Weber, Manfred Reichert, and Vicente Pelechano. *VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems.* Information and Software Technology 57, pp. 248–276, Springer, (2015).

- **Clara Ayora**, Victoria Torres, Jose Luis de la Vara, and Vicente Pelechano. *Variability Management in Process Families through Change Patterns.* (Submitted to Information and Software Technology, 2015, Springer).

International Conferences:

- **Clara Ayora**, Victoria Torres, Barbara Weber, Manfred Reichert, and Vicente Pelechano. *Enhancing modeling and change support for process families through change patterns.* 14th International Working Conference on Business Process Modeling, Development, and Support, BPMDS 2013, Valencia (Spain), pp. 246–260. Springer, Lecture Notes in Business Information Processing, Volume 147.

International Workshops:

- **Clara Ayora**, Victoria Torres, Manfred Reichert, Barbara Weber, and Vicente Pelechano. *Towards run-time flexibility for process families: open issues and research challenges.* 2nd International Workshop on Process Model Collections, PMC 2012, Tallin (Estonia), pp. 477–488. Springer, Lecture Notes in Business Information Processing, Volume 132.

- Victoria Torres, Stefan Zugal, Barbara Weber, Manfred Reichert, **Clara Ayora** and Vicente Pelechano. *A qualitative comparison of approaches supporting business process variability.* 3rd International Workshop on Reuse in Business Process Management, rBPMN 2012, Tallin (Estonia), pp. 560–572. Springer, Lecture Notes in Business Information Processing, Volume 132.

All these publications present the results of this thesis. Table 8.1 shows the relation between the contributions presented in this thesis and the publications achieved during its execution. In this case, publications are presented in chronological order of publication.

Contribution	Publication
Process Variability Characterization	- PMC 2012 - rBPM 2012 - IST Journal 2015
CP4PF (derivation, implementation, and validation)	- BPMDS 2013 - IST Journal 2015 (submitted)

Table 8.1: Relation between the contributions and the publications achieved

Detail and relevance of the publications

This section provides some information about the relevance of the conferences where different aspects of the work of this thesis have been published.

Information and Software Technology. Information and Software Technology is a peer-reviewed scientific journal focused on research and experience that contributes to the improvement of software development practices. The journal's scope includes any methods and techniques that allow to better engineer software and manage its development. According to the JCR index of 2013, the journal has an impact factor of 1.328.

In addition, it is ranked as the 31th out of 105 journals in the category "Computer Science/Software Engineering" and the 51th out of 135 journals in the category "Computer Science/Information Systems". Both rankings show the relevance of the journal for the community.

International Working Conference on Business Process Modeling, Development, and Support. The BPMDS conference has an important role in the business process community agglutinating high quality papers. This conference provides a discussion forum where researchers and practitioners of this community can meet, disseminate and exchange ideas and problems, identify some of the key challenges related to business processes, and explore together possible solutions and future works. The conference papers are published by Springer and it is usually possible to include them in a special issue of a relevant journal (e.g., SoSyM). According to the *CORE conference ranking* of 2014 [CORE, 2014], BPMDS is a Core C conference.

International workshops. In addition to the above mentioned venues, the first part of the work of this thesis has been published in two workshops (i.e., PMC and rBPM) from the main conference in business process management field, i.e., *Conference on Business Process Management*. These workshops have provided us with very valuable feedback specially at the early stage of the thesis.

8.2.2 Other publications

In addition to the main publications, a set of publications was also produced during the development of this thesis. Although the content of these publications is not strictly related to the content of this thesis, they are highly related to it since all of them deal with its main topic, i.e., process variability. These publications are listed in the following in chronological order. Again, the position of the name of the author of this thesis is used as an indicator of the degree of contribution in each publication.

- **Clara Ayora**, Germán Harvey Alférez, Victoria Torres, and Vicente Pelechano. *Procesos de negocio auto-adaptables al contexto*. VII Jornadas de Ciencia e Ingeniería de Servicios, JCIS 2011, A Coruña (Spain), pp. 147–160.

- **Clara Ayora**, Victoria Torres, and Vicente Pelechano. *Feature modeling to deal with variability in business process perspectives*. VIII Jornadas de Ciencia e Ingeniería de Servicios, JCIS 2012, Almería (Spain), pp. 111–124. Best Paper Award.

- **Clara Ayora**, Victoria Torres, and Vicente Pelechano. *Modelos de características para la gestión de la variabilidad en las perspectivas de los procesos de negocio*. Novática - Revista de la Asociación de Técnicos de Informática, 2012, pp. 36–41.

- **Clara Ayora**, Germán Harvey Alférez, Victoria Torres, and Vicente Pelechano. *Applying CVL to business process variability management*. 2nd International Workshop VARiability for You, VARY 2012, Innsbruck (Austria), pp. 24–29.

JCIS is the main Spanish conference on Services Computing and it is organized by Sistedes.[1] This conference is a forum for discussing and exchanging both knowledge and experience regarding computing services and related topics (e.g., business processes that rule these services). In this context, first we published a methodology for adapting business processes at enactment time. Through the use of autonomic computing, business processed can be adapted to new needs emerged while process execution. Second, we published a feature-based method to deal with process variability in the organizational and informational perspective. This last paper obtained the best paper award of the conference. In addition, an extended version of it was published in the national journal Novática[2] edited by the ATI (*Asociación de Técnicos de Informática*).

[1] http://www.sistedes.es/
[2] http://www.ati.es/novatica/

In turn, VARY[3] is a workshop held in conjuction with the MODELS conference.[4] This workshop influences the ongoing standardization efforts within OMG to establish a common variability language. In this case, we contributed by presenting the usage of this common variability language to deal with business process variability.

8.3 Research Collaborations

Research collaborations with researchers from international institutions enriched the preparation of this thesis. More precisely, we collaborated with:

- Associate Professor Dr. Barbara Weber. University of Innsbruck (Austria).
- Professor Dr. Manfred Reichert. University of Ulm (Germany).
- Visiting Professor Dr. Jose Luis de la Vara. Universidad Carlos III, Madrid (Spain).

The aim of these collaborations was to be open, influenced and enriched by distinct research streams, works, visions and schools. Both, Dr. Barbara Weber and Dr. Manfred Reichert have a long tradition of teaching, industry collaboration, and research in the domains of information systems engineering, business processes, and business process flexibility. Thanks to this joint collaboration, we enriched both the VIVACE framework for the systematic evaluation of process variability as well as the set of change patterns dealing with process variability. As important results of this collaboration, we published a joint journal article [Ayora et al., 2015] and a conference paper [Ayora et al., 2013] in which the author of this thesis was the leader of the projects. Furthermore, along this collaboration, a one-month research visit to the

[3] http://vary2012.irisa.fr/
[4] http://www.modelsconference.org/

University of Ulm was accomplished. This visit fostered valuable discussion and eventually discovered new perspectives and improvements on the work of this thesis that otherwise would not be reached.

In turn, Dr. Jose Luis de la Vara is a researcher with wide knowledge on safety assurance and certification [Nair et al., 2014], and is co-creator of SafetyMet [de la Vara & Panesar-Walawege, 2013]. This collaboration allowed us to validate our set of change patterns (i.e., CP4PF) through a case study in a real scenario (i.e., a safety standard). As a result of this collaboration, a journal article has been submitted leaded by the author of this thesis.

8.4 Future work

The work proposed in this thesis can be extended in several ways. The following list summarizes the research directions that are planned to continue this work:

Extend VIVACE with other characteristics such as understandability, maintainability, tool support features, and formalization. By also considering these advanced requirements, we will be able to additionally assess the quality of an existing process variability approach from a quantitative perspective.

Extend CP4PF to other process perspectives. As described in Chapter 2.1., process models comprise different perspectives (i.e., functional, behavioral, organizational, informational, temporal, and operational). In this thesis, we have only focused on the functional and behavioral ones (i.e., control flow distribution). We believe that our patterns can be easily extended to support the representation of a range of variations in data objects or resources, e.g., insert resource, delete data object.

Consider process variability at enactment time. In this thesis, we have only considered variability that is known at design time. How-

ever, in the near future, we also aim to identify complementary patterns for covering runtime variability. These complementary patterns need to consider unforeseeable changes that occur during process execution and were not anticipated at design time.

Apply CP4PF to other domains. Although in this thesis we have considered a process family with a high degree of variability (i.e., safety standard), we would like to conduct further empirical studies in order to determine to what extent the outcomes of our case study can be transferred to other domains (e.g., healthcare).

Conduct more empirical validations with practitioners. We would also like to perform controlled experiments with a number of subjects higher than the current validation with users presented in this thesis.

Bibliography

Acher, M., Collet, P., Lahire, P., & France, R. (2010). Managing variability in workflow with feature model composition operators. In *Proc. SC'10*, (pp. 17–33).

ADONIS (2008). Community Edition 3.0 http://www.adonis-community.com/.

Aghakasiri, Z., & Mirian-Hosseinabadi, S.-H. (2009). Workflow change patterns: Opportunities for extension and reuse. In *Proc. SERA'09*, (pp. 265–275).

Aguilar-Savén, R. (2004). Business process modelling: Review and framework. *International Journal of Production Economics*, *90*(2), (pp. 129–149).

Aguilar-Saven, R. S. (2004). Business process modelling: Review and framework. *International Journal of Production Economics*, *90*(2), (pp. 129–149).

Aiello, M., Bulanov, P., & Groefsema, H. (2010). Requirements and tools for variability management. In *Proc. COMPSACW'10*, (pp. 245–250).

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press.

Alférez, G., Pelechano, V., Mazo, R., Salinesi, C., & Diaz, D. (2014). Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software*, *91*(0), (pp. 24–47).

Angles, R., Ramadour, P., Cauvet, C., & Rodier, S. (2013). V-BPMI: A variability-oriented framework for web-based business processes modeling and implementation. In *Prc. RCIS'13*, (pp. 1–11).

Appleton, B. (1997). Patterns and software: Essential concepts and terminology.

ARIS (1990). Event-driven Process Chain. http://www.ariscommunity.com/event-driven-process-chain.

ARIS-Community (2010). ARIS Community Basic rules on EPC modeling. http://www.ariscommunity.com/users/rbaureis/2010-03-22-basic-rules-epc-modelling.

Atego (2010). Atego Process Director http://www.atego.com/products/atego-process-director/.

Avgeriou, P., & Zdun, U. (2005). Architectural patterns revisited–a pattern.

Ayora, C., Torres, V., Reichert, M., Weber, B., & Pelechano, V. (2012a). Towards run-time flexibility for process families: Open issues and research challenges. In *Proc. BPM'12 Workshops*, (pp. 477–488).

Ayora, C., Torres, V., Weber, B., Reichert, M., & Pelechano, V. (2012b). Dealing with variability in process-aware information systems: language requirements, features, and existing proposals. Technical Report UIB-2012-07. Tech. rep., Faculty of Engineering and Computer Science, University of Ulm,

Ayora, C., Torres, V., Weber, B., Reichert, M., & Pelechano, V. (2012c). Variability management in business process models, technical report, pros-tr-2012-06. Tech. rep., PROS - UPV,

Ayora, C., Torres, V., Weber, B., Reichert, M., & Pelechano, V. (2013). Enhancing modeling and change support for process families through change patterns. In *Proc. BPMDS/EMSAD'13*, (pp. 246–260).

Ayora, C., Torres, V., Weber, B., Reichert, M., & Pelechano, V. (2015). VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems. *Information and Software Technology*, *57*, (pp. 248–276).

Baier, T., Pascalau, E., & Mendling, J. (2010). On the suitability of aggregated and configurable business process models. In *Proc. BMMDS/EMMSAD'10*, (pp. 108–119).

Bass, L. (2007). *Software architecture in practice*. Pearson Education India.

Bayer, J., Gerard, S., Haugen, Ø., Mansell, J., Møller-Pedersen, B., Oldevik, J., Tessier, P., Thibault, J., & Widen, T. (2006). Consolidated product line variability modeling. In *Proc. SPL'06*, (pp. 195–241).

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70–77.

Beck, K., & Cunningham, W. (1987). Using pattern languages for object oriented programs. In *Proc. OOPSLA'87*.

Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., & D.Kuropka (2004). Configurative process modeling – outlining an approach to increased business process model usability. In *Proc. IRMA'04*.

Boffoli, N., Caivano, D., Castelluccia, D., & Visaggio, G. (2012). Business process lines and decision tables driving flexibility by selection. In *Proc. Software Composition*, (pp. 178–193).

Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J., & Pohl, K. (2001). Variability issues in software product lines. In *Proc. PFE'01 Workshop*, (pp. 13–21).

BPDM (2014). Business Process Definition MetaModel, volume ii: Process Definitions. http://www.omg.org/spec/BPDM/1.0/volume2/PDF.

BPMN (2011). Business Process Model and Notation, OMG Standard, version 2.0. http://www.bpmn.org/.

Bridgeland, D. M., & Zahavi, R. (2008). *Business Modeling: A Practical Guide to Realizing Business Value*. Morgan Kaufmann Publishers Inc.

Bucchiarone, A., Mezzina, C. A., & Pistore, M. (2013). CAptLang: A language for context-aware and adaptable business processes. In *Proc. VaMoS'13*, (pp. 1–5).

Buijs, J., & Reijers, H. (2014). Comparing business process variants using models and event logs. In *Proc. BPMDS/EMMSAD*, (pp. 154–168).

Bulanov, P., Groefsema, H., & Aiello, M. (2011). Business process variability: A tool for declarative template design. In *Proc. ICSOC Workshops'11*, (pp. 241–242).

Buschmann, F., Henney, K., & Schimdt, D. (2007). *Pattern-oriented Software Architecture: On Patterns and Pattern Language*, vol. 5. John Wiley & Sons.

Cetina, C., Giner, P., Fons, J., & Pelechano, V. (2009). Autonomic computing through reuse of variability models at runtime: The case of smart homes. *IEEE Computer*, *42*(10), 37–43.

Cleland-Huang, J., Chang, C., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *Software Engineering, IEEE Transactions on*, *29*(9), (pp. 796–810).

Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., & Little, R. (2002). *Documenting software architectures: views and beyond*. Pearson Education.

CMMI (2010). Capability Maturity Model Integration www.cmmiinstitute.com.

Coplien, J. (1997). Advanced c++ programming styles and idioms. In *Proc. IEEE TOOLS'97*, (pp. 352–352).

Coplien, J. O., & Alexander, A. W. O. (1996). Software patterns.

Coplien, J. O., & Harrison, N. (2005). *Organizational patterns of agile software development*. Pearson Prentice Hall Upper Saddle River.

CORE (2014). Conference Ranking. http://www.core.edu.au/index.php/conference-rankings.

Curtis, B., Kellner, M. I., & Over, J. (1992). Process modeling. *Communication of the ACM*, *35*(9), (pp. 75–90).

Czarnecki, K., & Antkiewicz, M. (2005). Mapping features to models: A template approach based on superimposed variants. In *Proc. GPCE'05*, (pp. 422–437).

Dadam, P., & Reichert, M. (2009). The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. *Computer Science - Research and Development*, *23*(2), (pp. 81–97).

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, (pp. 319–340).

de la Vara, J., Ali, R., Dalpiaz, F., Sánchez, J., & Giorgini, P. (2010). COMPRO: A methodological approach for business process contextualisation. In *Proc. OTM'10*, (pp. 132–149).

de la Vara, J. L., & Panesar-Walawege, R. K. (2013). Safetymet: A metamodel for safety standards. In *Proc. MODELS'13*, (pp. 69–86).

Derguech, W., & Bhiri, S. (2011). An automation support for creating configurable process models. In *Proc. WISE'11*, (pp. 199–212).

Derguech, W., Gao, F., & Bhiri, S. (2012). Configurable process models for logistics case study for customs clearance processes. In *Proc. BPM Workshops'12*, (pp. 119–130).

Derguech, W., Vulcu, G., & Bhiri, S. (2010). An indexing structure for maintaining configurable process models. In *Proc. BPMDS/EMMSAD'10*, vol. 50, (pp. 157–168).

Dijkman, R. (2008). Diagnosing differences between business process models. In *Proc. BPM'08*, (pp. 261–277).

Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., & Mendling, J. (2011a). Similarity of business process models: Metrics and evaluation. *Information Systems*, *36*(2), (pp. 498–516).

Dijkman, R., Gfeller, B., Küster, J., & Völzer, H. (2011b). Identifying refactoring opportunities in process model repositories. *Information and Software Technology*, *53*(9), (pp. 937–948).

Dijkman, R., La Rosa, M., & Reijers, H. (2012). Managing large collections of business process models–current techniques and challenges. *Computers in Industry*, *63*(2), (pp. 91–97).

Döhring, M., Reijers, H., & Smirnov, S. (2014). Configuration vs. adaptation for business process variant maintenance: An empirical study. *Information Systems*, *39*, (pp. 108–133).

Döhring, M., Zimmermann, B., & Karg, L. (2011). Flexible workflows at design– and runtime using BPMN2 adaptation patterns. In *Business Information Systems*, vol. 87, (pp. 25–36).

dos Santos Rocha, R., & Fantinato, M. (2013). The use of software product lines for business process management: A systematic literature review. *Information and Software Technology*, *8*(55), (pp. 1355–1373).

Dumas, M., Rosa, M. L., Mendling, J., & Reijers, H. (2013). *Fundamentals of Business Process Management.*. Springer.

Dumas, M., van der Aalst, W., & ter Hofstede, A. (2005). *Process-aware Information Systems: Bridging people and software through process technology*. 0-47166-360-9. John Wiley & Sons, Inc.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological review*, *87*(3).

Estublier, J., & Casallas, R. (1994). *Configuration Management, Trends in software, chapter IV*. John Wiley & Sons,.

Fernandez, E. B. (1998). Building systems using analysis patterns. In *Proc. ISAW'98*, (pp. 37–40).

Fowler, M. (1997). *Analysis Patterns: Reusable Objects Models*. Addison-Wesley Longman Publishing Co., Inc.

Frece, A., & Juric, M. B. (2012). Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models. *Journal of Visual Languages & Computing*, *23*(4), (pp. 223–247).

Gabriel, R. (1996). *Patterns of software*, vol. 62. Oxford University Press New York.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc.

Geyer, L., & Becker, M. (2002). On the influence of variabilities on the application-engineering process of a product family. In *Proc. SPL'02*, (pp. 1–14).

Gibbons, J. (2010). https://patternsinfp.wordpress.com/welcome/.

Giese, C., Schnieders, A., & Weiland, J. (2007). A practical approach for process family engineering of embedded control software. In *Proc. ECBS'07*, (pp. 229–240).

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of Intelligent Systems*, *16*(3), (pp. 391–409).

Gottschalk, F., Aalst, W., & Jansen-Vullers, M. (2007). Configurable process models – a foundational approach. In *Reference Modeling*, (pp. 59–77). Physica-Verlag HD.

Gottschalk, F., Aalst, W., & Jansen-Vullers, M. (2008). Mining reference process models and their configurations. In *Proc. OTM'08*, (pp. 263–272).

Gottschalk, F., Wagemakers, T., Jansen-Vullers, M., van der Aalst, W., & La Rosa, M. (2009). Configurable process models: Experiences from a municipality case study. In *Proc. CAiSE'09*, (pp. 486–500).

Groefsema, H., Bulanov, P., & Aiello, M. (2011). Declarative enhancement framework for business processes. In *Proc. ICSOC'11*, (pp. 495–504).

Gröner, G., Boskovic, M., Parreiras, F. S., & Gasevic, D. (2012). Modeling and validation of business process families. *Information Systems*, *38*(5), (pp. 709–726).

Gröner, G., Wende, C., Boskovic, M., Silva Parreiras, F., Walter, T., Heidenreich, F., Gasevic, D., & Staab, S. (2011). Validation of families of business processes. In *Proc. CAiSE'11*, (pp. 551–565).

Gschwind, T., Koehler, J., & Wong, J. (2008). Applying patterns during business process modeling. In *Proc. BPM'08*, (pp. 4–19).

Günther, C., Rinderle, S., Reichert, M., & van der Aalst, W. (2006). Change mining in adaptive process management systems. In *Proc. OTM'06*, vol. 4275, (pp. 309–326).

Hallerbach, A., Bauer, T., & Reichert, M. (2009). Guaranteeing soundness of configurable process variants in Provop. In *Proc. CEC'09*, (pp. 98–105).

Hallerbach, A., Bauer, T., & Reichert, M. (2010a). Capturing variability in business process models: The Provop approach. *Software Maintenance and Evolution: Research and Practice*, *22*(6-7), (pp. 519–546).

Hallerbach, A., Bauer, T., & Reichert, M. (2010b). *Configuration and Management of Process Variants*, chap. International Handbook on Business Process Management. Springer-Verlag Berlin Heidelberg.

Hochstein, A., Zarnekow, R., & Brenner, W. (2005). ITIL as common practice reference model for it service management: Formal assessment and implications for practice. In *IEEE International Conference on e-Technology, e-Commerce, and e-Services*, (pp. 704–710).

Hsieh, H.-F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative health research*, *15*(9), 1277–1288.

IEC (2010). Functional safety of electrical/electronic/programmable electronic safety-related systems (iec 61508).

Indulska, M., Recker, J., Rosemann, M., & Green, P. (2009). Business process modeling: Current issues and future challenges. In *Proc. CAiSE'09*, (pp. 501–514).

Jablonski, S., & Bussler, C. (1996). *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thomson Computer Press.

Jalali, S., & Wohlin, C. (2012). Systematic literature studies: Database searches vs. backward snowballing. In *Proc. ESEM'12*, (pp. 29–38).

Jiménez-Ramírez, A., Weber, B., Barba, I., & Del Valle, C. (2015). Generating optimized configurable business process models in scenarios subject to uncertainty. *Information and Software Technology*, *57*, 571–594.

Keepence, B., & Mannion, M. (1999). Using patterns to model variability in product families. *IEEE software*, *16*(4), (pp. 102–108).

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3. Tech. rep., Keele University and Durham University Joint Report,

Koetter, F., Weidmann, M., & Schleicher, D. (2011). Guaranteeing soundness of adaptive business processes using ABIS. In *Proc. BIS'11*, (pp. 74–85).

Kolokolov, V., Ruehl, S., Baumann, P., Zhang, S., & Verclas, S. (2014). Modelling variability in activity diagrams for mobile business applications. In *Proc. COMPSAC'14*, (pp. 155–160).

Korherr, B. (2008). *Business process modelling – languages, goals and variabilities*. Ph.D. thesis, Vienna University of Technology.

Koschmider, A., & Oberweis, A. (2007). How to detect semantic business process model variants? In *Proc. SAC'07*, (pp. 1263–1264).

Krueger, C. (2013). Software Product Lines. Home Page. http://www.softwareproductlines.com.

Kumar, A., & Yao, W. (2012). Design and management of flexible process variants using templates and rules. *Computers in Industry*, *63*(2), (pp. 112–130).

Künzle, V., & Reichert, M. (2011). PHILharmonicFlows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, *23*(4), (pp. 205–244).

Küster, J., Gerth, C., & Engels, G. (2010). Dynamic computation of change operations in version management of business process models. In *Proc. ECMFA'10*, (pp. 201–206).

Küster, J., Gerth, C., Förster, A., & Engels, G. (2008). Detecting and resolving process model differences in the absence of a change log. In *Proc. BPM'08*, (pp. 244–260).

La Rosa, M. (2009). *Managing Variability in Process-Aware Information Systems*. Ph.D. thesis, Queensland University of Technology.

La Rosa, M., Dumas, M., & ter Hofstede, A. (2009a). Modelling business process variability for design-time configuration. In *Handbook of Research on Business Process Modeling*, (pp. 204–228).

La Rosa, M., Dumas, M., ter Hofstede, A., & Mendling, J. (2011). Configurable multi-perspective business process models. *Information Systems*, *36*(2), (pp. 313–340).

La Rosa, M., Dumas, M., Uba, R., & Dijkman, R. (2010). Merging business process models. In *Proc. OTM'10*, (pp. 96–113).

La Rosa, M., & Mendling, J. (2009). Domain-driven process adaptation in emergency scenarios. In *Proc. BPM Workshops'09*, (pp. 290–297).

La Rosa, M., van der Aalst, W., Dumas, M., & Milani, F. (2013). Business process variability modeling : A survey. Technical Report 61842.

La Rosa, M., van der Aalst, W., Dumas, M., & ter Hofstede, A. (2009b). Questionnaire-based variability modeling for system configuration. *Software and Systems Modeling*, (2), (pp. 251–274).

Lang, A. (2002). *Flexible business process modeling – A Systematic mapping study*. Master's thesis, Athabasca University.

Lanz, A., Weber, B., & Reichert, M. (2010). Workflow time patterns for process-aware information systems. In *Proc. BPMDS/EMMSAD'10*, (pp. 94–107).

Lanz, A., Weber, B., & Reichert, M. (2012). Time patterns for process-aware information systems. *Requirements Engineering Journal*, (pp. (pp. 1–29)).

Lapouchnian, A., Yu, Y., & Mylopoulos, J. (2007). Requirements-driven design and configuration management of business processes. In *Proc. BPM'07*, (pp. 246–261).

Lazovik, A., & Ludwig, H. (2007). Managing process customizability and customization: Model, language and process. In *Proc. WISE'07*, (pp. 373–384).

Lenz, R., & Reichert, M. (2007). IT support for healthcare processes – premises, challenges, perspectives. *Data & Knowledge Engineering*, *61*(1), (pp. 39–58).

Lesiecki, N. (2005). http://www.ibm.com/developerworks/java/library/j-aopwork6/index.html.

Li, C. (2010). *Mining Process Model Variants: Challenges, Techniques, Examples*. Ph.D. thesis, University of Twente, The Netherlands.

Li, C., Reichert, M., & Wombacher, A. (2008). Mining process variants: Goals and issues. In *Proc. SCC'08*, (pp. 573–576).

Li, C., Reichert, M., & Wombacher, A. (2011). Mining business process variants: Challenges, scenarios, algorithms. *Data & Knowledge Engineering*, *70*(5), (pp. 409–434).

Lönn, C.-M., Uppström, E., Wohed, P., & Juell-Skielse, G. (2012). Configurable process models for the Swedish public sector. In *Proc. CAiSE'12*, (pp. 190–205).

Lu, R., Sadiq, S. W., & Governatori, G. (2009). On managing business processes variants. *Data & Knowledge Engineering*, *68*(7), (pp. 642–664).

Mahmod, N., & Chiew, W. Y. (2010). Structural similarity of business process variants. In *Proc. ICOS'10*, (pp. 17–22).

Marcolino, A., Oliveira, E., Gimenes, I., & Barbosa, E. (2014). Empirically based evolution of a variability management approach at UML class level. In *Proc. COMPSAC'14*, (pp. 354–363).

Marrella, A., Mecella, M., & Russo, A. (2011). Featuring automatic adaptivity through workflow enactment and planning. In *Proc. CollaborateCom'11*.

Martin, R. C. (2000). Design principles and design patterns. Tech. rep., Object Mentor,

Martinez-Ruiz, T., Münch, J., García, F., & Piattini, M. (2012). Requirements and constructors for tailoring software processes: a systematic literature review. *Software Quality Journal*, *20*(1), (pp. 229–260).

Mechrez, I., & Reinhartz-Berger, I. (2014). Modeling design-time variability in business processes: Existing support and deficiencies. In *Proc. BPMDS/EMMSAD'14*, (pp. 378–392).

Meerkamm, S., & Jablonski, S. (2011). Configurable process models: experiences from a medical and an administrative case study. In *Proc. ECIS'11*.

Melao, N., & Pidd, M. (2000). A conceptual framework for understanding business processes and business process modelling. *Information Systems Journal*, *10*(2), (pp. 105–130).

Mendling, J. (2008). C-EPC Validator. http://www.mendling.com/EPML/C-EPC-Validator.xsl.

Mendling, J., Recker, J., Rosemann, M., & van der Aalst, W. (2006). Generating correct EPCs from configured C-EPCs. In *Proc. SAC'06*, (pp. 1505–1510).

Mendling, J., Reijers, H. A., & van der Aalst, W. (2010). Seven process modeling guidelines (7PMG). *Information and Software Technology*, *52*(2), 127–136.

Mendling, J., Verbeek, H., van Dongen, B., van der Aalst, W., & Neumann, G. (2008). Detection and prediction of errors in EPCs of the SAP reference model. *Data & Knowledge Engineering*, *64*(1), (pp. 312–329).

Montero, I., Peña, J., & Ruiz-Cortés, A. (2008). From feature models to business processes. In *Proc. IEEE SCC'08*, (pp. 605–608).

Moon, M., Hong, M., & Yeom, K. (2008). Two-level variability analysis for business process with reusability and extensibility. In *Proc. COMPSAC '08*, (pp. 263–270).

Moreno-Montes de Oca, I., Snoeck, M., Reijers, H. A., & Rodríguez-Morffi, A. (2015). A systematic literature review of studies on business process modeling quality. *Information and Software Technology*, *58*, 187–205.

Murguzur, A., De Carlos, X., Trujillo, S., & Sagardui, G. (2014). Context-aware staged configuration of process variantsruntime. In *Proc. CAiSE'14*, (pp. 241–255).

Murguzur, A., Truong, H. L., & Dustdar, S. (2013). Multi-perspective process variability: A case for smart green buildings (short paper). In *Proc. SOCA'13*, (pp. 25–29).

Nair, S., De La Vara, J. L., Sabetzadeh, M., & Briand, L. (2014). An extended systematic literature review on provision of evidence for safety certification. *Information and Software Technology*, *56*(7), 689–717.

Nair, S., de la Vara, J. L., Sabetzadeh, M., & Falessi, D. (2015). Evidence management for compliance of critical systems with safety standards: A survey on the state of practice. *Information and Software Technology*, *60*(0), 1–15.

Nguyen, T., Colman, A. W., & Han, J. (2011). Modeling and managing variability in process-based service compositions. In *Proc. ICSOC'12*, (pp. 404–420).

Ognjanovic, I., Mohabbati, B., Gaevic, D., Bagheri, E., & Bokovic, M. (2012). A metaheuristic approach for the configuration of business process families. In *Proc. SCC'12*, (pp. 25–32).

OMG (1989). Object Management Group. http://www.omg.org.

OWL (2009). Web Ontology Language, w3c. http://www.w3.org/TR/owl-features/.

Panesar-Walawege, R. K., Sabetzadeh, M., & Briand, L. (2013). Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation. *Information and Software Technology*, *55*(5), 836–864.

Park, J., & Yeom, K. (2011). A modeling approach for business processes based on variability. In *Proc. SERA'11*, (pp. 211–218).

Pascalau, E., & Rath, C. (2010). Managing business process variants at eBay. In *Proc. BPM'10*, (pp. 91–105).

Pedreira, O., Piattini, M., Luaces, M. R., & Brisaboa, N. R. (2007). A systematic review of software process tailoring. *ACM SIGSOFT Software Engineering Notes*, *3*(32), (pp. 1–6).

Perry, D. E., Porter, A. A., & Votta, L. G. (2000). Empirical studies of software engineering: a roadmap. In *Proceedings of the conference on The future of Software engineering*, (pp. 345–355). ACM.

Pinggera, J., Zugal, S., & Weber, B. (2010). Investigating the process of process modeling with cheetah experimental platform. In *Proc. ER-POIS'10*, (pp. 25–31).

Pohl, K., Böckle, G., & van der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York.

Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education.

Puhlmann, F., Schnieders, A., Weiland, J., & Weske, M. (2006). Variability mechanisms for process models. Technical Report 17/2005. Tech. rep., Hasso–Plattner–Institut, Postdam,

Recker, J., Rosemann, M., der Aalst, W., & Mendling, J. (2006). On the syntax of reference model configuration – transforming the C-EPC into lawful EPC models. In *Proc. BPM Workshops'06*, (pp. 497–511).

Reichert, M., Rechtenbach, S., Hallerbach, A., & Bauer, T. (2009). Extending a business process modeling tool with process configuration facilities: The Provop demonstrator. In *BPM Demonstration Track*.

Reichert, M., Rinderle, S., Kreher, U., & Dadam, P. (2005). Adaptive process management with ADEPT2. *In Proc. ICDE'05*, (pp. (pp. 1113–1114)).

Reichert, M., & Weber, B. (2012). *Enabling Flexibility in Process–Aware Information Systems: Challenges, Methods, Technologies*. Springer-Verlag Berlin Heidelberg.

Reijers, H., Mans, R., & van der Toorn, R. (2009). Improved model management with aggregated business process models. *Data & Knowledge Engineering*, *68*, (pp. 221–243).

Reinhartz-Berger, I., Soffer, P., & Sturm, A. (2005). A domain engineering approach to specifying and applying reference models. In *Proc. Workshop Enterprise Modelling and Information Systems Architectures 75*, (pp. 50–63).

Reinhartz-Berger, I., Soffer, P., & Sturm, A. (2010). Extending the adaptability of reference models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *40*(5), (pp. 1045–1056).

Reinhartz-Berger, I., & Sturm, A. (2012). Comprehensibility of UML-based software product line specifications: A controlled experiment. *Empirical Software Engineering*, (pp. (pp. 1–36)).

Riehle, D., & Züllighoven, H. (1996). Understanding and using patterns in software development. *TAPOS*, *2*(1), 3–13.

Robson, C. (2002). Real world research: A resource for social scientists and practitioners-researchers, vol. 2. *Oxford: Blackwell*.

Rosa, M. L. (2009). *Managing Variability in Process-Aware Information Systems*. Ph.D. thesis, Faculty of Science and Technology Queensland University of Technology Brisbane, Australia.

Rosemann, M., & van der Aalst, W. (2007). A configurable reference modelling language. *Information Systems*, *32*(1), (pp. 1–23).

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164.

Russell, N., ter Hofstede, A., Edmond, D., & van der Aalst, W. (2004a). Workflow data patterns. Technical Report FIT-TR-2004-01. Tech. rep., Eindhoven University of Technology,

Russell, N., ter Hofstede, A., Edmond, D., & van der Aalst, W. (2004b). Workflow resource patterns. Technical Report WP 127. Tech. rep., Queensland University of Technology,

Russell, N., van der Aalst, W., & ter Hofstede, A. (2006). Workflow exception patterns. In *Proc. CAiSE'06*, (pp. 288–302).

Saidani, O., & Nurcan, S. (2014). Business process modeling: A multi-perspective approach integrating variability. In *Proc. BPMDS/EMMSAD'14*, (pp. 169–183).

Sakr, S., Pascalau, E., Awad, A., & Weske, M. (2011). Partial process models to manage business process variants. *International Journal of Business Process Integration and Management*, *6*(2), (pp. 240–256).

Santos, E., Pimentel, J., Castro, J., & Finkelstein, A. (2012). On the dynamic configuration of business process models. In *Proc. BMMDS/EMMSAD'12*, vol. 113, (pp. 331–346).

SAP-Business-Suite (1992). http://www.sap.com/index.html.

Scherer, R., & Sharmak, W. (2011). Process risk management using configurable process models. In *Proc. IFIP AICT'11*, vol. 362, (pp. 341–348).

Schmid, K., & John, I. (2004). A customizable approach to full lifecycle variability management. *Science of Computer Programming*, *53*(3), 259–284.

Schnieders, A., & Puhlmann, F. (2007). Variability modeling and product derivation in e-business process families. In *Technologies for Business Information Systems*, (pp. 63–74).

Schnieders, A., & Weske, M. (2007). Activity diagram based process family architectures for enterprise application families. *Journal Enterprise Interoperability*, (pp. (pp. 67–76)).

Schobbens, P.-Y., Heymans, P., & Trigaux, J.-C. (2006). Feature diagrams: A survey and a formal semantics. In *Proc. RE'06*, (pp. 136–145).

Schunselaar, D., Verbeek, E., Aalst, W., & Raijers, A., Hajo (2012). Creating sound and reversible configurable process models using CoSeNets. In *Proc. BIS'12*, (pp. 24–35).

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft Press.

Sharp, A., & McDermott, P. (2001). *Workflow Modeling: Tools for process improvement and application development*. Artech House Inc.

Shull, F., Singer, J., & Sjøberg, D. I. (2008). *Guide to advanced empirical software engineering*, vol. 93. Springer.

Sinnema, M., Deelstra, S., & Hoekstra, P. (2006). The COVAMOF derivation process. In *Proc. ICSR'06*, (pp. 101–114).

Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., & Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *Software Engineering, IEEE Transactions on*, *31*(9), 733–753.

Soffer, P. (2005). Scope analysis: identifying the impact of changes in business process models. *Software Process: Improvement and Practice*, *10*(4), (pp. 393–402).

Stages (2014). The Stages Process Management System http://stages.methodpark.com/.

Synergia (2009). http://www.processconfiguration.com/download.html.

Tealeb, A., Awad, A., & Galal-Edeen, G. (2014). Context-based variant generation of business process models. In *Proc. BPMDS/EMMSAD'14*, (pp. 363–377).

Thomas, O. (2008). Design and implementation of a version management system for reference modeling. *Journal of Software*, *3*(1), (pp. 49–62).

Thomson-Reuters (2014). Journal Citation Reports (jcr). http://wokinfo.com/products_tools/analytical/jcr/.

Tragatschnig, S., Tran, H., & Zdun, U. (2013). Change patterns for supporting the evolution of event-based systems. In *Proc. CoopIS'13*, (pp. 1–8).

Tragatschnig, S., Tran, H., & Zdun, U. (2014). Impact analysis for event-based systems using change patterns. In *Proc. SAC'14*.

Tropashko, V., & Burleson, D. (2007). *SQL Design Patterns: Expert Guide to SQL Programming*. Rampant Techpress.

Tryggeseth, E., Gulla, B., & Conradi, R. (1995). Modelling systems with variability using the PROTEUS configuration language. In *Proc. ICSE'95*, (pp. 216–240).

UML (2007). Unified Modeling Language, OMG Standard, version 2.1.2. http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF.

Vaishnavi, V., & Kuechler, W. (2004). Design research in information systems. http://desrist.org/design-research-in-information-systems.

Valença, G., Alves, C., & Niu, N. (2013). A systematic mapping study on business process variability. *Journal of Computer & Science Information Technology*, *1*(5).

van der Aalst, W., & Basten, T. (2002). Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, *270*(1–2), (pp. 125–203).

van der Aalst, W., Dumas, M., Gottschalk, F., ter Hofstede, A., La Rosa, M., & Mendling, J. (2010a). Preserving correctness during business process model configuration. *Formal Aspects of Computing*, *22*(3–4), (pp. 459–482).

van der Aalst, W., Lohmann, N., & La Rosa, M. (2012). Ensuring correctness during process configuration via partner synthesis. *Information Systems*, *37*(6), (pp. 574–592).

van der Aalst, W., Lohmann, N., La Rosa, M., & Xu, J. (2010b). Ensuring correctness during process configuration: An approach based on partner synthesis. In *Proc. BPM'10*, (pp. 95–111).

van der Aalst, W., Reijers, H., Weijters, A., van Dongen, B., De Medeiros, A., Song, M., & Verbeek, H. (2007). Business process mining: An industrial application. *Information Systems*, *32*(5), 713–732.

van der Aalst, W., & ter Hofstede, A. (2003). YAWL: Yet another workflow language. *Information Systems*, *30*, (pp. 245–275).

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., & Barros, A. (2003). Workflow patterns. *Distributed and Parallel Databases*, *14*(1), (pp. 5–51).

Vergidis, K., Tiwari, A., & Majeed, B. (2008). Business process analysis and optimization: beyond reengineering. *IEEE Transactions on Systems, Man, and Cybernetics*, *1*(38), (pp. 69–82).

Vogelaar, J., Verbeek, H., Luka, B., & van der Aalst, W. (2011). Comparing business processes to determine the feasibility of configurable models: A case study. In *Proc. BPM Workshops'11*, (pp. 50–61).

Weber, B., Pinggera, J., Torres, V., & Reichert, M. (2013). Change patterns in use: A critical evaluation. In *Proc. BPMDS/EMSAD'13*, (pp. 261–276).

Weber, B., Reichert, M., Mendling, J., & Reijers, H. (2011). Refactoring large process model repositories. *Computers in Industry*, *62*(5), (pp. 467–486).

Weber, B., Reichert, M., & Rinderle-Ma, S. (2008). Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering*, *66*, (pp. 438–466).

Weske, M. (2007). *Business process management: concepts, languages, architectures*. Springer-Verlag Berlin Heidelberg.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.

WS-BPEL (2004). Web Services Business Process Execution Language (WS-BPEL), OASIS Standard, version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf.

Yahya, B., & Bae, H. (2011). Generating reference business process model using heuristic approach based on activity proximity. In *Proc. IDT'11*, (pp. 469–478).

Yao, Q., & Sun, Y. (2012). Design of the variable business process model based on message computing. In *Proc. CSO'12*, (pp. 169–172).

Yao, W., Basu, S., Li, J., & Stephenson, B. (2012). Modeling and configuration of process variants for on-boarding customers to it outsourcing. In *Proc. SCC'12*, (pp. 415–422).

Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. *Information & Software Technology*, *53*(6), (pp. 625–637).

Zhao, X., & Liu, C. (2013). Version management for business process schema evolution. *Information Systems*, *38*(8), (pp. 1046–1069).

Appendices

A

Check-in Process

For illustrating the size and complexity of a process family, we consider the check-in procedures at an airport, which are characterized by a high degree of variability. Even though this process is similar irrespective of the airport the passenger departs from and the airline flying with, numerous variations exist depending on distinguished factors. For example, variability is caused by the type of check-in (e.g., online, at the counter, or at the self-servicing machine), which, in turn, determines the type of boarding card (e.g., electronic versus paper-based). Other sources of variability include the flight destination (e.g., information about the accommodation is required when traveling to the US) and the type of passenger (e.g., unaccompanied minors and handicapped people might require extra assistance). Depending on the type of luggage (e.g., bulk or overweight luggage), moreover, the process slightly differs since an extra fee might have to be paid. Finally, temporal variations regarding the check-in procedure are typical as well (e.g., possibility to check-in several days before departure versus checking-in a few hours

before the flight).

Figures A.1 and A.2 show six simplified process variants of this check-in process represented in terms of the *Business Process Modeling Notation* (BPMN) [BPMN, 2011]. These variants have been modeled and validated in collaboration with subject matter experts. In particular, the showed process variants share commonalities while also showing differences. Activities common to all process variants are colored in grey. *Variants 1* and *2* (cf. Figure A.1) presume that the check-in is done online by the passenger. First, the passenger is identified and a seat is assigned. *Variant 1* describes the process in case the passenger is flying from Europe to the United States, which requires information about accommodation as well as filling in the electronic system for travel authorization (i.e., ESTA form). Finally, an electronic boarding card is printed and the passenger drops off the luggage at the business class counter. Regarding *Variant 2*, after printing the boarding card, the payment of an extra fee at the airline ticket office is required due to luggage overweight. In turn, for *Variant 3* the check-in is done at the self-servicing machine and the luggage is dropped off at the fast bag drop counter. Finally, for these three process variants, check-in becomes available 23 hours before departure.

In contrast, *Variants 4-6* (cf. Figure A.2) represent the check-in process accomplished at the respective counter at the airport. For example, *Variant 4* describes the check-in for an unaccompanied minor. In this variant, a special seat is assigned and an extra form is filled in. In addition, a copy of the boarding card is required for the relative accompanying the minor to the boarding gate. *Variant 5* refers to a handicapped passenger requiring extra assistance by a person accompanying him, whereas *Variant 6* corresponds to the check-in process of a passenger carrying bulk luggage. In these three process variants, a check-in may only be performed at maximum 3 hours before departure, once the counters will have opened. Finally, the boarding card is printed in paper format.

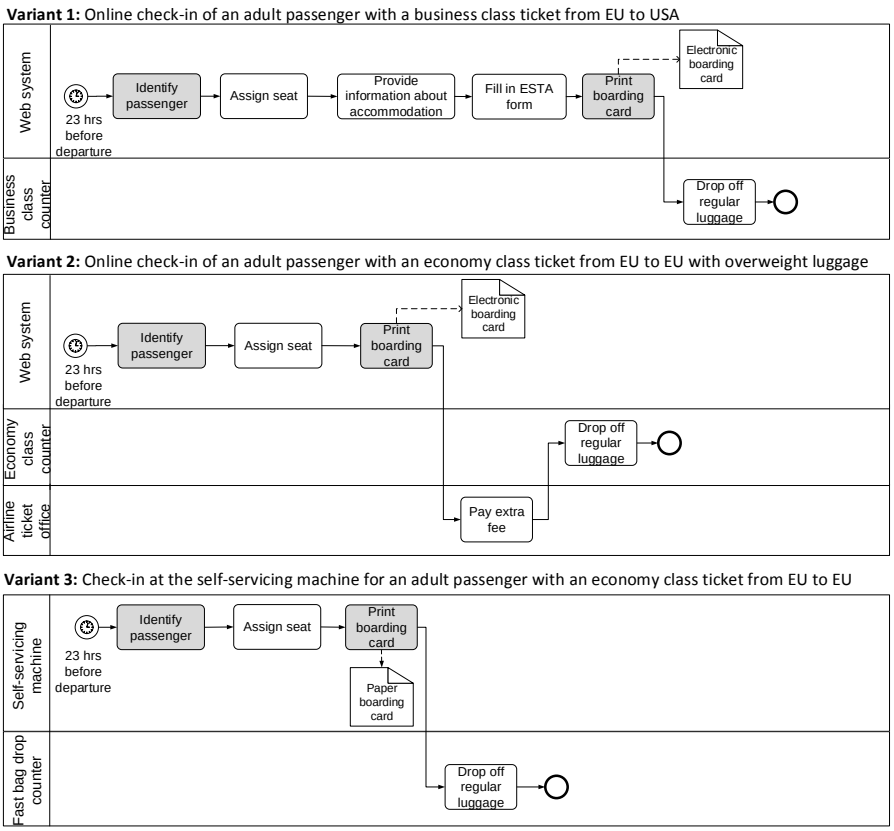
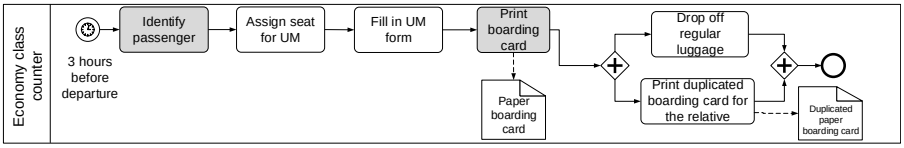
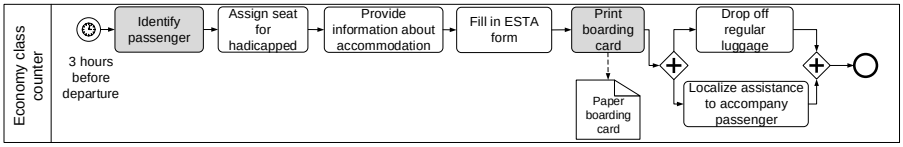


Figure A.1: Variants of the check-in process (1)

Variant 4: Check-in for an unaccompanied minor (UM) passenger with an economy class ticket from EU to EU with a relative accompanying him until the boarding gate



Variant 5: Check-in for a handicapped passenger with an economy class ticket from EU to USA



Variant 6: Check-in for an adult passenger with an economy class ticket from EU to EU with bulk luggage

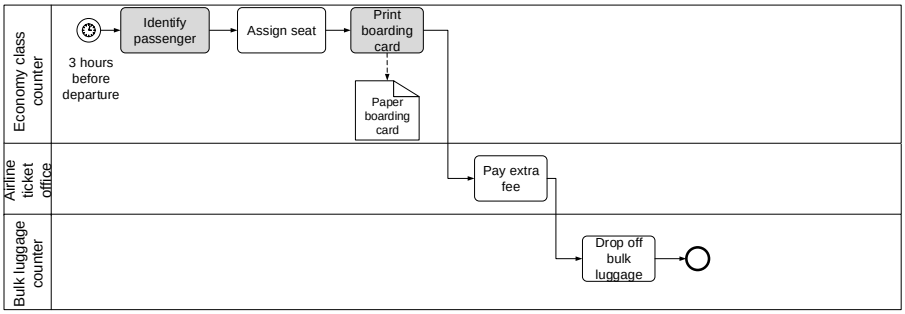


Figure A.2: Variants of the check-in process (2)

B

Procedure of the Systematic Study on Process Variability

This appendix contains the description of the study we performed about process variability approaches regarding their expressiveness with respect to process variability modeling as well as their support along the process lifecycle. More concretely, this study was performed as a *Systematic Literature Review* (SLR).

A SLR is a means of identifying, evaluating, and interpreting relevant data in a specific area through a replicable, scientific, and transparent approach, which reduces the probability of any bias [Kitchenham & Charters, 2007]. To conduct such an SLR with respect to process variability, we designed a protocol following the guidelines, procedures, and policies proposed by *Kitchenham* in [Kitchenham & Charters, 2007]. According to the latter, this protocol described the formulation of the research questions, the search string, the data sources chosen for per-

forming the search, the identification of inclusion and exclusion criteria, the quality assessment questions, the selection of studies[1], the method for extracting the data from the selected studies, and the way how the obtained data shall be analyzed. In the following, we describe these aspects in detail.

B.1 Research Questions Formulation

Our overall goal was to identify and analyze studies related to business process variability. Note that a detailed understanding of the way process variability is managed in the context of process families requires an in-depth analysis of various aspects; e.g., modeling languages, language constructs, tools, and features [Reichert & Weber, 2012]. Our SLR focused on the analysis of relevant papers regarding their expressiveness for modeling process variability, their support for handling process variability along the process lifecycle, and their empirical evaluations. For this purpose, we considered the following research questions, which will be discussed in the following.

- **RQ1.** What underlying *business process modeling languages* are used for modeling process variability?
- **RQ2.** Which *techniques* are used for representing process variability in a configurable process model[2]?
- **RQ3.** What *language constructs* are provided for representing process variability in a configurable process model?
- **RQ4.** Which *process perspectives* are covered by languages that enable the modeling of process variability?
- **RQ5.** What *tools* exist for enabling process variability?

[1] In the given context, a *study* refers to a paper retrieved in the SLR.

[2] Remember that related process variants are defined in terms of a configurable process model, which then represents a complete process family.

- **RQ6.** What *variability support features* are provided for fostering process variability in all phases of the process lifecycle?
- **RQ7.** Have existing process variability approaches been *evaluated*? If so, how does this evaluation look like?
- **RQ8.** In which domains have existing process variability approaches been applied?

Since there exists no standard language for modeling process variability, we were interested in identifying what process modeling languages have been used for this purpose (RQ1). As literature refers to various techniques for creating configurable process models [Ayora et al., 2012a], in addition, the SLR shall provide an overview of the way these techniques are used (RQ2). In order to allow assessing the expressiveness of existing approaches for modeling process variability, the SLR shall further identify a core set of variability-specific language constructs frequently used by these approaches (RQ3). Since variability may concern different process perspectives, the SLR shall provide insights into the perspectives covered by existing process variability approaches (RQ4). In order to assess the practical applicability of existing process variability approaches, the SLR shall further identify the available tools supporting these approaches (RQ5). Moreover, the SLR shall create an in-depth understanding of variability support features (e.g., to verify and validate process variants) that foster process variability along the different phases of the process lifecycle (RQ6). To assess the level of maturity of existing process variability approaches, we further investigate whether and–if so–how these approaches have been empirically evaluated (RQ7). Finally, we analyzed the domains in which existing process variability approaches have been applied (RQ8).

B.2 Search String

We subjectively elaborated a search string using keywords we derived based on our in-depth knowledge of the topic and taking the defined research questions into account; i.e., we applied subjective search string

definition [Zhang et al., 2011]. Since the keywords may be described with synonymous terms [Reichert & Weber, 2012], we attempted to use a wide range of terms in order to broadly cover the scope of the SLR. These terms were connected through the logical connector OR.

The search string was iteratively refined with the goal to maximize the number of different candidate studies to be retrieved for the SLR. More precisely, several pilot searches were performed in order to refine the keywords in the search string based on a trial and error approach. We excluded terms whose inclusion does not yield additional studies. These pilot searches were continuously inspected by experts on process variability in order to ensure that all relevant studies are found.

The search string of our SLR was as follows:

'process family' OR 'configurable process model' OR 'process model collection' OR 'reference process model' OR 'configurable workflow' OR 'process variant' OR 'business process variability' OR 'process configuration' OR 'process model configuration'

B.3 Data Source Selection

The defined search string was applied to relevant data sources to find studies related to the topic (i.e., process variability). More precisely, six electronic libraries were identified by topic experts as a basis for conducting the SLR:

1. SpringerLink
2. IEEE Xplore Digital Library
3. ACM Digital Library
4. Science Direct - Elsevier
5. Wiley InterScience
6. World Scientific

These libraries included the proceedings of the most relevant conferences, workshops and journals the business process management community publishes its research results in; e.g., *Data & Knowledge Engineering*, *Computers in Industry*, *Information Systems*, *Information and Software Technology*, *Conference on Business Process Management (BPM)*, *Conference on Advanced Information Systems Engineering (CAiSE)*, *Working Conference of Business Process Modeling, Development, and Support (BPMDS)*, *IEEE Enterprise Computer Conference (EDOC)*, *International Conference on Cooperative Information Systems (CoopIS)*, *Symposium on Applied Computing (SAC)*, and *International Conference on Service Computing (SCC)*.

With the above selection of libraries, we wanted to retrieve a maximum number of candidate studies from a minimum number of libraries, while reducing the overlap between them as much as possible. In addition, we checked whether papers about the topic, which we had already known, were included in the selected libraries as well. As an additional data source, we considered the literature cited by the retrieved studies themselves; i.e., we applied *backward reference searching* [Jalali & Wohlin, 2012]. In turn, this improved SLR results by covering a wide spectrum of directly relevant studies. Finally, Google Scholar Alerts (e.g., "process variability") were continuously analyzed in order to become aware of any publication on the topic emerging in 2013 during the writing process; i.e., after the search in the specified data sources was performed.

Due to the large amount of data sources chosen, the defined search string was suitably adapted where necessary; e.g., through the use of plural forms (e.g., 'process families' instead of 'process family'). In addition, the search string was applied to full text (i.e., title, abstract, and content of the study) in order to ensure that potentially relevant studies are not excluded.

B.4 Inclusion and Exclusion Criteria

We defined the following inclusion and exclusion criteria in order to identify relevant studies for our SLR:

Inclusion criterion:

1. The study is related to process variability and describes
   - a process variability approach or
   - process variability support features or
   - an empirical evaluation of a process variability approach.

Exclusion criteria:

1. The study is not related to process variability, or it merely mentions process variability terms in a generalized manner.
2. The study is not electronically available or requires the payment of access fees.[3]
3. The study refers to a non-peer reviewed publication (e.g., a preface, editorial, or technical report).
4. The study is not presented entirely in English language.
5. The study presents some type of review (e.g., survey, SLR), but does not deal with outcomes of a particular research work.
6. In case several studies refer to the same process variability approach, all studies except the latest and most complete version is excluded.

A study was eliminated if it met any of these exclusion criteria. Note that we did not apply any restriction with respect to the publication date.

[3] This only applies to fees that are not covered by the subscriptions to any of the selected data sources.

B.5 Quality Assessment

In addition to the inclusion and exclusion criteria, each selected study was assessed based on a set of quality assessment questions (QA). In particular, this was crucial for interpreting and synthesizing the data extracted from the selected studies [Kitchenham & Charters, 2007].

- QA1: Does the study include sufficient data to infer how process variability is explicitly modeled?
- QA2: Does the study include sufficient data about the support of process variability in one or several phases of the process lifecycle?
- QA3: Has the process variability approach described in the study been implemented, formalized or empirically evaluated?

These questions were scored as follows: 1 if the question is satisfied and 0 if it is not satisfied. The intention behind this quality assessment was to ensure a certain level of maturity for the studies included in the SLR. Further, we wanted to guarantee that studies of pure conceptual nature are not included.

B.6 Study Selection

The SLR was conducted by applying the defined search string to each of the six electronic libraries. These queries resulted in a total of 4947 studies (cf. Figure B.1, Stage 1). Meta-data related to them was then imported into an Excel file, which stored the source of each study together with its main information, i.e., title, authors, type of venue (e.g., conference, journal), and complete reference of the study. Following this, each of the studies was reviewed in order to determine its relevance for the SLR. Note that this step was accomplished based on the defined exclusion criteria.

First, the title of each retrieved study was analyzed in order to check whether it actually dealt with process variability (i.e., Exclusion Criteria 1-5). In cases the information from the title was not sufficient

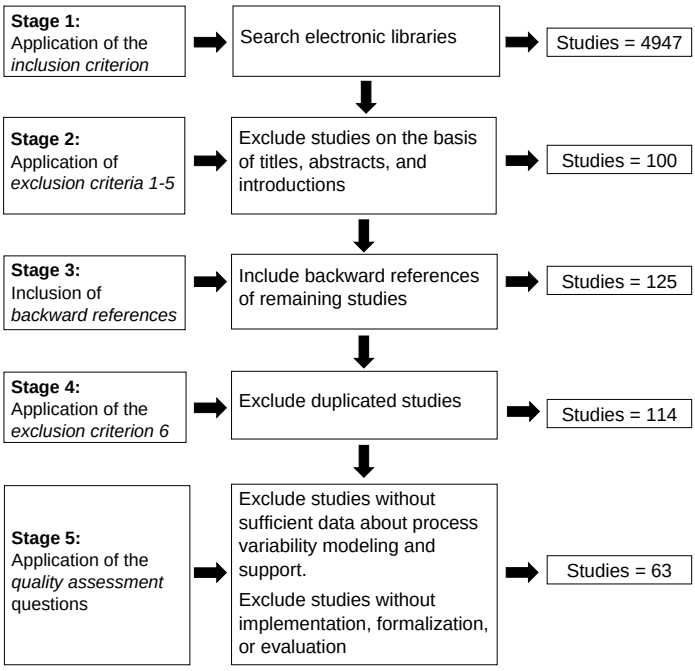


Figure B.1: Stages of the study selection process

to decide whether or not to include the study, the corresponding abstract and introduction sections were additionally scanned. After this filtering, we obtained 100 relevant studies (cf. Figure B.1, Stage 2). Following this, we analyzed the literature cited in the background or related work sections of these 100 studies (i.e., we apply *backward reference searching*). This results in 25 additional studies, which we included for further consideration (cf. Figure B.1, Stage 3). In the next stage, duplicated studies were removed (i.e., Exclusion Criterion 6) resulting in 114 relevant studies in total (cf. Figure B.1, Stage 4). Finally, studies related to process variability, but without sufficient data about how process variability is modeled or supported or with no tool implementation, formalization or empirical evaluation, were discarded in order to

ensure a sufficient level of maturity (i.e., quality assessment questions). Overall, this resulted in 63 *primary studies* (cf. Figure B.1, Stage 5), which are summarized in Table B.1. Each of these studies is associated with a unique identifier (i.e., Study ID), which is used in the following to refer to the respective studies.

Study ID	Study ID
S1-Alférez et al. [Alférez et al., 2014]	S33-Czarnecki et al. [Czarnecki & Antkiewicz, 2005]
S2-Bucchiarone et al. [Bucchiarone et al., 2013]	S34-Becker et al. [Becker et al., 2004]
S3-Kumar et al. [Kumar & Yao, 2012]	S35-van der Aalst et al. [van der Aalst et al., 2012]
S4-Frece et al. [Frece & Juric, 2012]	S36-Li et al. [Li et al., 2011]
S5-Santos et al. [Santos et al., 2012]	S37-Weber et al. [Weber et al., 2011]
S6-W. Yao et al. [Yao et al., 2012]	S38-Derguech et al. [Derguech & Bhiri, 2011]
S7-Q. Yao et al. [Yao & Sun, 2012]	S39-Yahya et al. [Yahya & Bae, 2011]
S8-Ognjanovic et al. [Ognjanovic et al., 2012]	S40-Koetter et al. [Koetter et al., 2011]
S9-Gröner et al. [Gröner et al., 2012]	S41-Gröner et al. [Gröner et al., 2011]
S10-Boffoli et al. [Boffoli et al., 2012]	S42-van der Aalst et al. [van der Aalst et al., 2010a]
S11-Schunselaar et al. [Schunselaar et al., 2012]	S43-La Rosa et al. [La Rosa et al., 2010]
S12-Groefsema et al. [Groefsema et al., 2011]	S44-Mahmod et al. [Mahmod & Chiew, 2010]
S13-Döhring et al. [Döhring et al., 2011]	S45-Gottschalk et al. [Gottschalk et al., 2008]
S14-Park et al. [Park & Yeom, 2011]	S46-Thomas et al. [Thomas, 2008]
S15-Nguyen et al. [Nguyen et al., 2011]	S47-Koschmider et al. [Koschmider & Oberweis, 2007]
S16-Pascalau et al. [Sakr et al., 2011]	S48-Mendling et al. [Mendling et al., 2006]
S17-Meerkamm et al. [Meerkamm & Jablonski, 2011]	S49-Recker et al. [Recker et al., 2006]
S18-Derguech et al. [Derguech et al., 2010]	S50-Reinhartz-Berger et al. [Reinhartz-Berger et al., 2005]
S19-Hallerbach et al. [Hallerbach et al., 2010b]	S51-Döhring et al. [Döhring et al., 2014]
S20-de la Vara et al. [de la Vara et al., 2010]	S52-Derguech et al. [Derguech et al., 2012]
S21-Reinhartz-Berger et al. [Reinhartz-Berger et al., 2010]	S53-Lönn et al. [Lönn et al., 2012]
S22-Acher et al. [Acher et al., 2010]	S54-Bulanov et al. [Bulanov et al., 2011]
S23-Reijers et al. [Reijers et al., 2009]	S55-Vogelaar et al. [Vogelaar et al., 2011]
S24-La Rosa et al. [La Rosa et al., 2009b]	S56-Reinhartz-Berger et al. [Reinhartz-Berger & Sturm, 2012]

S25-La Rosa et al. [La Rosa et al., 2011]	S57-Scherer et al. [Scherer & Sharmak, 2011]
S26-Montero et al. [Montero et al., 2008]	S58-Pascalau et al. [Pascalau & Rath, 2010]
S27-Moon et al. [Moon et al., 2008]	S59-Baier et al. [Baier et al., 2010]
S28-Gottschalk et al. [Gottschalk et al., 2007]	S60-Gottschalk et al. [Gottschalk et al., 2009]
S29-Lapouchnian et al. [Lapouchnian et al., 2007]	S61-La Rosa et al. [La Rosa & Mendling, 2009]
S30-Schnieders et al. [Schnieders & Puhlmann, 2007]	S62-Schnieders et al. [Schnieders & Weske, 2007]
S31-Lazovik et al. [Lazovik & Ludwig, 2007]	S63-Giese et al. [Giese et al., 2007]
S32-Lu et al. [Lu et al., 2009]	

Table B.1: Final list of primary studies

During the selection process, we organized these 63 primary studies in three groups:

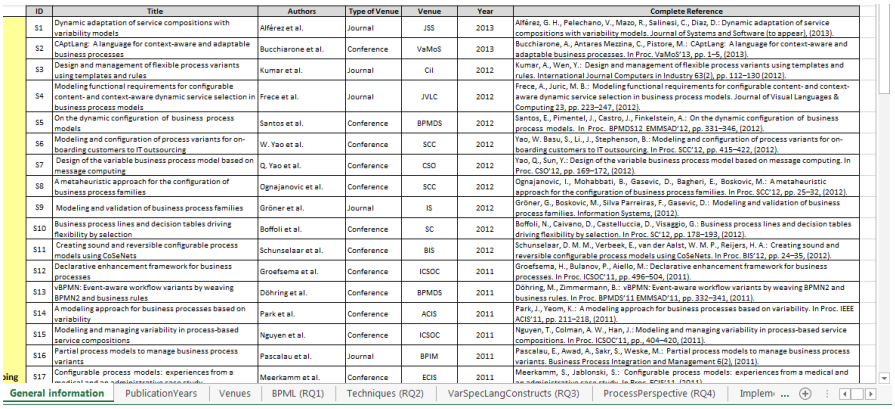
1. Studies describing process variability approaches: S1 - S34.
2. Studies describing process variability support features: S35 - S50.
3. Studies describing solely empirical evaluations of process variability approaches: S51 - S63.

The specified selection process was carried out by the author of this thesis and was continuously checked by her advisors and coauthors [Ayora et al., 2015]. More precisely, they randomly reviewed selected studies to ensure consistency of the process. Further, they ensured the correct application of the inclusion and exclusion criteria as well as the quality assessment questions. All disagreements were resolved through discussion.

B.7 Data Extraction Strategy

To each of the 63 primary studies, a *data extraction process* was applied with the goal to answer the research questions defined in Section B.1.

For this purpose, we used Excel sheets to capture and store the relevant information. Figure B.2 includes an excerpt of these sheets.[4]



ID	Title	Authors	Type of Venue	Venue	Year	Complete Reference
S1	Dynamic adaptation of service compositions with variability models	Alférez et al.	Journal	JSS	2013	Alférez, G. H., Pelechano, V., Mazo, R., Salinesi, C., Diaz, D.: Dynamic adaptation of service compositions with variability models. Journal of Systems and Software (to appear), (2013).
S2	CAptLang: A language for context-aware and adaptable business processes	Bucchiarone et al.	Conference	VaMoS	2013	Bucchiarone, A., Antares Mezzina, C., Pistore, M.: CAptLang: A language for context-aware and adaptable business processes. In Proc. VaMoS'13, pp. 1–5, (2013).
S3	Design and management of flexible process variants using templates and rules	Kumar et al.	Journal	CII	2012	Kumar, A., Wen, Y.: Design and management of flexible process variants using templates and rules. International Journal Computers in Industry 63(2), pp. 112–130 (2012).
S4	Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models	Frece et al.	Journal	JVLC	2012	Frece, A., Juric, M. B.: Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models. Journal of Visual Languages & Computing 23, pp. 223–247, (2012).
S5	On the dynamic configuration of business process models	Santos et al.	Conference	BPMDS	2012	Santos, E., Pimentel, J., Castro, J., Finkelstein, A.: On the dynamic configuration of business process models. In Proc. BPMDS12 EMMSAD'12, pp. 331–346, (2012).
S6	Modeling and configuration of process variants for on-boarding customers to IT outsourcing	W. Yao et al.	Conference	SCC	2012	Yao, W. Basu, S., Li, J., Stephenson, B.: Modeling and configuration of process variants for on-boarding customers to IT outsourcing. In Proc. SCC'12, pp. 415–422, (2012).
S7	Design of the variable business process model based on message computing	Q. Yao et al.	Conference	CSO	2012	Yao, Q., Sun, Y.: Design of the variable business process model based on message computing. In Proc. CSO'12, pp. 169–172, (2012).
S8	A metaheuristic approach for the configuration of business process families	Ognajanovic et al.	Conference	SCC	2012	Ognajanovic, I., Mohabbati, B., Gasevic, D., Bagheri, E., Boskovic, M.: A metaheuristic approach for the configuration of business process families. In Proc. SCC'12, pp. 25–32, (2012).
S9	Modeling and validation of business process families	Gröner et al.	Journal	IS	2012	Gröner, G., Boskovic, M., Silva Parreiras, F., Gasevic, D.: Modeling and validation of business process families. Information Systems, (2012).
S10	Business process lines and decision tables driving flexibility by selection	Boffoli et al.	Conference	SC	2012	Boffoli, N., Caivano, D., Castelluccia, D., Visaggio, G.: Business process lines and decision tables driving flexibility by selection. In Proc. SC'12, pp. 178–193, (2012).
S11	Creating sound and reversible configurable process models using CoSeNets	Schunselaar et al.	Conference	BIS	2012	Schunselaar, D. M. M., Verbeek, E., van der Aalst, W. M. P., Reijers, H. A.: Creating sound and reversible configurable process models using CoSeNets. In Proc. BIS'12, pp. 24–35, (2012).
S12	Declarative enhancement framework for business processes	Groefsema et al.	Conference	ICSOC	2011	Groefsema, H., Bulanov, P., Aiello, M.: Declarative enhancement framework for business processes. In Proc. ICSOC'11, pp. 496–504, (2011).
S13	vBPMN: Event-aware workflow variants by weaving BPMN2 and business rules	Döhring et al.	Conference	BPMDS	2011	Döhring, M., Zimmermann, B.: vBPMN: Event-aware workflow variants by weaving BPMN2 and business rules. In Proc. BPMDS'11 EMMSAD'11, pp. 332–341, (2011).
S14	A modeling approach for business processes based on variability	Park et al.	Conference	ACIS	2011	Park, J., Yeom, K.: A modeling approach for business processes based on variability. In Proc. IEEE ACIS'11, pp. 211–218, (2011).
S15	Modeling and managing variability in process-based service compositions	Nguyen et al.	Conference	ICSOC	2011	Nguyen, T., Colman, A. W., Han, J.: Modeling and managing variability in process-based service compositions. In Proc. ICSOC'11, pp., 404–420, (2011).
S16	Partial process models to manage business process variants	Pascalau et al.	Journal	BPIM	2011	Pascalau, E., Awad, A., Saky, S., Weske, M.: Partial process models to manage business process variants. Business Process Integration and Management 6(2), (2011).
S17	Configurable process models: experiences from a	Meerkamm et al.	Conference	ECIS	2011	Meerkamm, S., Jablonski, S.: Configurable process models: experiences from a medical and

General information | PublicationYears | Venues | BPML (RQ1) | Techniques (RQ2) | VarSpecLangConstructs (RQ3) | ProcessPerspective (RQ4) | Implem ...

Figure B.2: Overview of the Excel sheet for the general information table

In detail, we extracted the following information:

1. General information about the study; i.e., title, authors, type of venue (e.g., conference, journal), and complete reference of the study.

2. The underlying language used for modeling process variability; e.g., BPMN and EPC (RQ1).

3. The technique used to define a configurable process model (RQ2).

4. Variability-specific language constructs that may be used to represent process variability (RQ3).

5. The process perspectives covered; e.g., behavioral, organizational, and informational (RQ4).

[4]The complete filled Excel sheets can be downloaded from: http://www.pros.upv.es/bpvar/SLR/SLRDataExtraction.rar

6. Information about the implementation of the approach; i.e., availability of a tool implementing the approach, type of implementation, and link for downloading this tool (RQ5).

7. Features provided for the management of process variability (RQ6).

8. Available results from empirical evaluations of a process variability approach and type of evaluation performed (e.g., case study, survey) (RQ7).

9. Domain in which the process variability approach has been applied (RQ8).

For research questions RQ1, RQ2, RQ3, RQ5, and RQ6, data was extracted by first creating an initial list of categories based on our knowledge and experience about the topic (i.e., process variability) [Ayora et al., 2012a,b; Weber et al., 2008, 2011]. Once data extraction started, each study was then thoroughly analyzed and extracted data was assigned to a category based on content analysis techniques [Hsieh & Shannon, 2005]; if new categories were identified, they were added to the list. Throughout the analysis, process categories might be merged. In this case, already analyzed studies were re-assigned. However, regarding RQ4 (i.e., process perspectives covered), we started data extraction with a predefined list of the existing process perspectives (cf. Section 2.1). Then, we assigned each study to the process perspectives it covered; i.e., we used descriptive statistics to analyze the results (i.e., frequency counts). A similar procedure was applied in the context of RQ7. We first created a predefined list of existing types of empirical evaluations. Then, each study was assigned to the type of evaluation it described. Finally, for RQ8, we included each identified domain in which existing process variability approaches have been applied by analyzing the content of each study. Again, throughout this analysis similar domains might be merged. This may imply the reassignment of already analyzed studies. Figure B.3 summarizes the data extracted and the techniques used for data analysis.

RQ	Extracted item	Type of data	Analysis
General information	Title	Free text	--
	Author	Free text	--
	Venue	Free text	--
	Reference	Free text	--
RQ1	Underlying language for modeling process variability	Initial list based on previous knowledge	Content analysis techniques
RQ2	Technique used to create a configurable process model	Initial list based on previous knowledge	Content analysis techniques
RQ3	Variability-specific language constructs provided to represent process variability	Initial list based on previous knowledge	Content analysis techniques
RQ4	Process perspectives covered	Predefined list of existing process perspectives	Descriptive statistics (i.e., frequency counts)
RQ5	Existence of a tool implementing the approach	Yes/No	--
	Type of implementation	Initial list based on previous knowledge	Content analysis techniques
	Download link for the tool	Free text	--
RQ6	Features provided for the management of process variability	Initial list based on previous knowledge	Content analysis techniques
RQ7	Type of empirical evaluation performed	Predefined list of existing types of empirical evaluations	Descriptive statistics (i.e., frequency counts)
RQ8	Domain in which the process variability approach has been applied	Free text	Content analysis techniques

Figure B.3: Data extraction summary

B.8 Data Analysis

Data analysis shall provide suitable information to answer our research questions. This was achieved by synthesizing the data obtained from the data extraction process. More precisely, the respective research questions were answered by analyzing the identified categories based on the created Excel sheets as well as the results of the quality assessment process. In addition, for answering RQ1-RQ4, only studies of the first type (i.e., S1-S34) were considered since they describe the expressiveness of the respective approach regarding the modeling of process variability. In turn, for answering RQ5 and RQ6, studies of the first and second type were analyzed (i.e., S1-S50) since both types might deal with implementation support for process variability. Finally, for RQ7 and RQ8, studies of the first (i.e., S1-S34) and third (i.e., S51-S63) type were considered since they might provide empirical evaluations of

process variability approaches.

In order to simplify the synthesis of the extracted data, we used descriptive techniques to summarize them; e.g., graphics and tabular descriptions.

B.9 Statistics of the Primary Studies

This section presents some statistics of the primary studies we selected from the SLR. First, Figure B.4 shows the temporal distribution of the 63 primary studies by publication year (i.e., from 2004 to 2013). As can be seen, the yearly number of published studies on process variability has increased over time, with a peak in 2012. This indicates a growing interest in the topic (i.e., process variability).

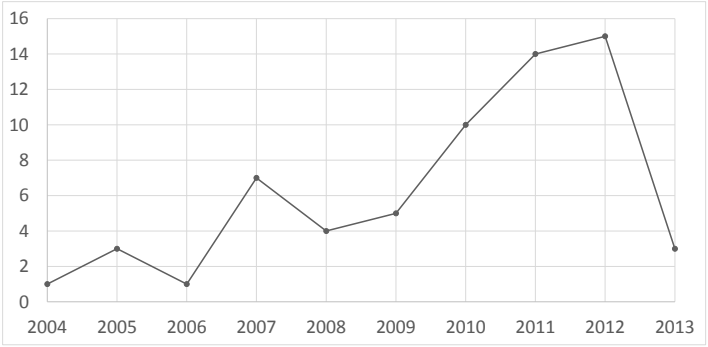


Figure B.4: Distribution of primary studies by publication year

Second, an additional analysis was performed regarding the publication venue in which the primary studies were published (primary studies published as book chapters were not taken into account in this analysis; therefore the latter was based on 60 out of the 63 studies). 42 of the primary studies were published in proceedings of conference and workshops (70%), while 18 studies (30%) appeared in journals (cf. Figure B.5).

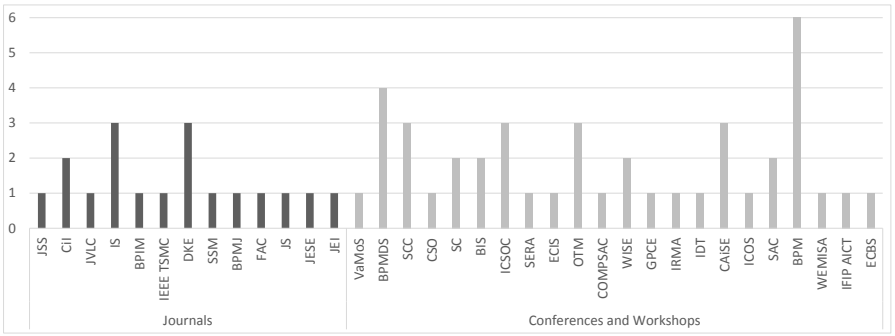


Figure B.5: Distribution of primary studies by publication venue

A total of 35 publication venues were identified. Interestingly, there are only two publications venues, namely *BPM* (*Conference on Business Process Management*) and *BPMDS* (*Working Conference of Business Process Modeling, Development, and Support*) with a relatively high number of primary studies (i.e., 4 and 6 studies respectively). All other venues published at most three studies on the topic. Finally, it is noteworthy that process variability was a topic addressed in various fields, i.e., publication venues from the business process management field (e.g., *BPM* conference), the web services field (e.g., *ICSOC* conference), the software engineering field (e.g., *JSS* journal), and the information systems field (e.g., *CAiSE* conference). Figure B.6 includes the list with the full names of the publication venues.

B.10 Threats of Validity

The main threats to the validity of our work are **selection bias**, **inaccuracy in data extraction & analysis**, and **reliability** [Kitchenham & Charters, 2007; Perry et al., 2000; Runeson & Höst, 2009; Sjøberg et al., 2005].

Journals	JSS	Journal of Systems and Software
	CII	Computers in Industry
	JVLC	Journal of Visual Languages & Computing
	IS	Information Systems
	BPIM	Business Process Integration and Management
	IEEE TSMC	IEEE Transactions on Systems, Man, and Cybernetics
	DKE	Data & Knowledge Engineering
	SSM	Software and System Modeling
	BPMJ	Business Process Management Journal
	FAC	Formal Aspects of Computing
	JS	Journal of Software
	JESE	Journal Empirical Software Engineering
	JEI	Journal Enterprise Interoperability
Conferences and Workshops	VaMoS	International Workshop on Variability Modelling of Software-intensive Systems
	BPMDS	Working Conference on Business Process Modeling, Development, and Support
	SCC	IEEE International Conference on Services Computing
	CSO	International Joint Conference on Computational Science and Optimization
	SC	International Conference on Software Composition
	BIS	International Conference on Business Information Systems
	ICSOC	International Conference on Service Oriented Computing
	SERA	International Conference on Software Engineering Research, Management, and Applications
	ECIS	European Conference on Information Systems
	OTM	On the Move Federated Conferences & Workshops
	COMPSAC	IEEE International Computer Software and Applications
	WISE	International Conference on Web Information Systems Engineering
	GPCE	International Conference on Generative Programming and Component Engineering
	IRMA	Information Resources Management Association Conference
	IDT	International Conference on Intelligent Decision Technologies
	CAiSE	International Conference on Advanced Information Systems
	ICOS	IEEE Conference on Open Systems
	SAC	ACM Symposium on Applied Computing
	BPM	International Conference on Business Process Management
	WEMISA	Workshop Enterprise Modelling and Information Systems Architecture
	IFIP AICT	IFIP Advances in Information and Communication Technologies
	ECBS	International Conference and Workshops on the Engineering of Computer-Based Systems

Figure B.6: List of publication venues

To ensure that the SLR is complete as far as possible and no important literature is missing, we used six well-known literature sources. These include the most important conference and journals on the topic (i.e., process variability). In addition, by scanning the references of the retrieved studies (i.e., backward reference searching), we ensured the completeness of the SLR. Further, we ensured that all relevant literature previously known to us was found by the SLR as well.

First, our systematic search was conducted in 2013. Accordingly,

studies published later were not included in our work.[5] In order to minimize the selection bias, the selection process performed by the main author was continuously reviewed by her co-authors. To be more precise, the co-authors randomly reviewed selected studies to ensure the consistency of the selection process, along with the correct application of inclusion and exclusion criteria. Disagreements emerging in this context were resolved through comprehensive discussions. Paper duplication constitutes another potential threat. Hence, the selected studies were checked twice in order to detect and remove duplicate papers, including only the most recent and complete version.

Second, data extraction and analysis were carried out by the main author. This might comprise subjective decisions since several primary studies did not provide a clear description of objectives and results. To mitigate this risk, a rigor extraction process was applied based on the guidelines of Kitchenham [Kitchenham & Charters, 2007]. In addition, the co-authors continuously checked the work of the first author resolving disagreements through discussion.

Finally, we ensure reliability as the search process can be replicated by other researchers. Since the data extraction process also considers subjective factors (e.g., in cases where studies did not provide clear descriptions), however, there is no guarantee that other researchers will obtain exactly the same results as presented in this work.

B.11 Comparison with other Reviews

Figure B.7 presents a comparative summary of the related revies (cf. Section 4.5) based on the retrieved studies. While column "Primary studies" presents the identified primary studies of each work, column "Overlapping studies" shows the studies identified in our SLR as well. In terms of primary studies, there is no significant difference between the SLRs. To be more precise, *Lang et al.* [Lang, 2002] retrieved 60, *Santos et al.* [dos Santos Rocha & Fantinato, 2013] 63, and our work

[5] We continuously scan newly emerging papers and approaches to evolve our framework as well as to apply it to the emerging approaches.

63 primary studies. Concerning *Valença et al.* [Valença et al., 2013], there is a higher number of primary studies (i.e., 80 studies). This can be explained with the complementary keywords the authors use in the search string (e.g., "change", "agility"). The overlap of the studies is relatively low between our SLR and *Lang et al.* and *Santos et al.*, respectively, since the goals of these works are different. On the contrary, the overlap increases in the case of *Valença et al.* since the focus is more related to our SLR (i.e., process variability). However, note that we expanded the analysis of process variability to other phases of the lifecycle and identify the set of features specifically tailored to process families as well.

Work	Primary studies	Overlapping studies
Lang et al.	60	S19, S25, S47
Valença et al.	80	S16, S19, S21, S22, S24, S25, S28, S32, S34, S35, S36, S38, S41, S42, S43, S45, S49
Santos et al.	63	S8, S9, S10, S14, S15, S26, S27, S41, S62

Figure B.7: Comparison of related studies

C

Material Used in the Validation with PAIS engineers

This appendix contains the material that was used in the validation of the CP4PF with PAIS engineers.

C.1 Demographic Survey

For ensuring that both groups of PAIS engineers were homogeneous, we asked subjects to fill out a demographic survey in advance in order to determine their experience regarding the topics of the validation, i.e., process modeling, process families, and adaptation patterns. This section presents the questions of this survey:

1. What description matches best your current status?
   - Academic
   - Professional
2. How many years ago did you start modeling business processes?
   - Less than 1 year
   - More than 1 year but less than 3 years
   - More than 3 years but less than 5 years
   - More than 5 years
3. How many years ago did you start modeling with BPMN?
   - Less than 1 year
   - More than 1 year but less than 3 years
   - More than 3 years but less than 5 years
   - More than 5 years
4. Have you got a certification in BPMN?
   - None
   - OCEB/OCEB2
   - BPMessentials
   - IBM Certified Solution
   - Other
5. If you have selected 'Other' please specify which certification:
6. How many BPMN models have you analyzed within the last 12 months? (A year has about 250 working days. In case you read one model per day, this would sum up to 250 models).
   - Less than 60
   - More than 60 but less than 120

- More than 120 but less than 180
- More than 180 but less than 250
- More than 250

7. How many BPMN models have you created within the last 12 months? (A year has about 250 working days. In case you read one model per day, this would sum up to 250 models).

   - Less than 60
   - More than 60 but less than 120
   - More than 120 but less than 180
   - More than 180 but less than 250
   - More than 250

8. How many activities did all these BPMN process models have on average?

   - Less than 10
   - More than 10 but less than 50
   - More than 50 but less than 100
   - More than 100

9. Have you used adaptation patterns before? i.e., any mechanism to automatically change the structure of a BPMN process model before? (e.g., to automatically insert an activity in parallel to another, to automatically embed an activity in a conditional branching)

   - Yes
   - No

10. Have you dealt with process families before? i.e., collection of related process models that share the same core but also show some differences depending on the application context (e.g., the same process executed in different countries may be handled in different ways due to differences in regulations)

- Yes
- No

C.2 Material Provided for the Tasks

In order to introduce PAIS engineers in the validation session, we distributed the material between the groups (G1 and G2). Note that to mitigate the learning and tiredness effects, G1 first evolve a configurable process model without CP4PF and G2 first evolve it using CP4PF.

C.2.1 Instructions for the validation

This session will be structured in the following way:

1. Guided part:
    - BPMN tutorial
    - C-EPC-like BPMN tutorial
    - Basic training
    - Change patterns tutorial
    - Familiarization task 1 (INSERT patterns)
    - Familiarization task 2 (DELETE patterns)
2. Not guided part:
    - Modeling task 1 without CP4PF
    - Assess mental effort
    - Modeling task 2 with CP4PF
    - Assess mental effort
    - Fill in the questionnaire Perceived Ease of Use
    - Fill in the questionnaire Perceived Usefulness
    - Feedback

C.2.2 Basic training

What are process model variants and process families?

The increasing adoption of business processes in enterprises has resulted in large process model repositories comprising collections of related ***process model variants***, which share the same core activities but also show differences depending on the application context (e.g., the same process executed in different countries may be handled in different ways due to differences in regulations). A collection of related process variants is denoted as ***process family***.

What is a configurable process model?

In this context, related process model variants are defined in terms of a ***configurable process model***, which represents the complete behavior of a process family (i.e., it contains the behavior of all the related process model variants). In particular, such a configurable process model eliminates model redundancies by modeling the shared core activities only ones. Furthermore, it fosters model reuse since variant differences can be shared among multiple variants. Configurable process models contain two groups of modeling elements:

1. Elements that represent the shared core activities by all process model variants (i.e., *commonalities*). Typically, these elements are defined using primitives of a business process language.
2. Elements that represent the *variability-specific language constructs* that define the difference between each process model variants.

Example of a configurable process model:

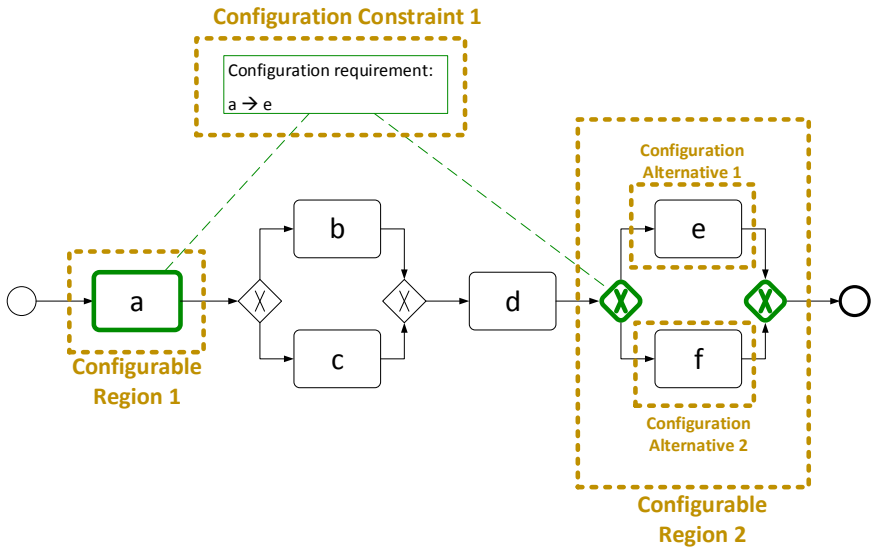


Figure C.1: Example of a configurable process model

Note that first two XOR gateways and activities b, c, d, e, and f correspond to commonalities shared by all process variants and are represented using BPMN primitives. Note as well that the variability-specific language constructs are highlighted with yellow squares and represented using C-EPC-like BPMN primitives marked in green.

What variability-specific language constructs are?

- **Configurable Region**: region of a configurable process model for which different choices (i.e., variations, differences) exist depending on the application context (cf. Configurable Region 1 and Configurable Region 2 in Figure C.1).
- **Configuration Alternative**: particular choice for a configur-

able region (cf. Configuration Alternative 1 and Configuration Alternative 2 in Figure C.1).

- **Configuration Constraint**: semantic restriction regarding the selection of configuration alternatives; e.g. exclusion, inclusion (cf. Configuration Constraint 1 in Figure C.1).

What is a change pattern?

A change pattern is a solution for a recurring problem when modeling process families. More precisely, they constitute automatic and structural adaptations for a configurable process model using high-level change operations (e.g., insert activity in parallel) instead of low level change primitives (e.g., add activity).

What are the change patterns for process families?

- Insert Configurable Region
- Delete Configurable Region
- Insert Configuration Alternative
- Delete Configuration Alternative
- Insert Configuration Constraint
- Delete Configuration Constraint

These patterns will be deeply analyzed and used in the following tutorial and familiarization tasks.

C.2.3 Familiarization task 1

The purpose of this task is to help you to get used to the tool, process families, and the INSERT change patterns. For such purpose, the tool

displays a source configurable process model (cf. Figure C.2). From the latter, you need to reproduce the target configurable process model (cf. Figure C.3) using the INSERT change patterns. When you are finished with modeling, please use the **Finish-Modeling** button on the top left to proceed.

Please keep in mind that the model will be automatically laid out after applying a change pattern.

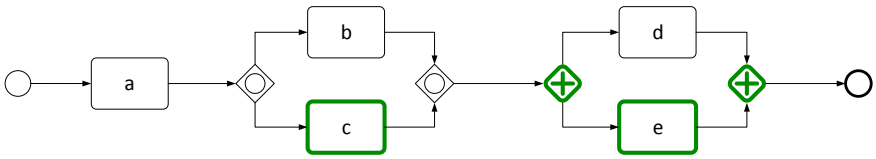


Figure C.2: Source configurable process model for the familiarization task 1

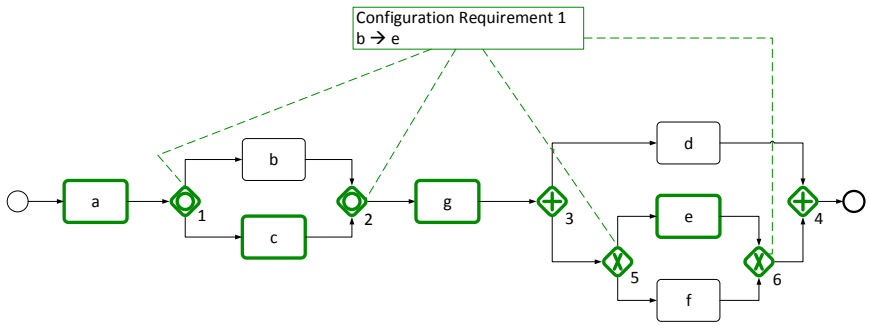


Figure C.3: Target configurable process model for the familiarization task 1

In order to help you in this task, we provide the solving path needed to obtain the target model from the source one. Each sentence of the solving path has the following format:
"Select (modeling elements to select) - CHANGE PATTERN TO APPLY"

1. Select (a) - INSERT Configurable Region Function
2. Select (OR1, b, c, OR2) - INSERT Configurable Region Conditional
3. Select (OR2, AND3) - INSERT Configurable Region Function (Name: g)
4. Select (e) - INSERT Configurable Region Exclusive
5. Select (XOR5, e, XOR6) - INSERT Configuration Alternative (Name: f)
6. Select (OR1, OR2, XOR5, XOR6) - INSERT Configuration Constraint (Constraint: b -> e)

Hint 1: When creating fragments between gateways, it does not matter whether activity "b" is the activity in the upper branch and "c" in the lower or the other way around.

Hint 2: When creating the model in the tool, it does not matter if the edges are not as sharp as in the model above.

Hint 3: To select modeling elements, you can drag the cursor over the elements or click each element plus the key CTRL of the keyboard. In both cases, selected elements will be highlighted in grey.

Hint 4: When pressing the *Layout* button in the panel dialog, the displayed model is reordered.

Hint 5: When pressing the *Undo* button in the panel dialog, the last performed operation is undone.

Hit 6: The sequence flows can be rearranged by dragging the startpoint (or endpoint) to another modeling element using the 'Select' tool from the Palette on the right hand side of the tool.

C.2.4 Familiarization task 2

The purpose of this task is to help you to get used to the tool, process families, and the DELETE change patterns. For such purpose, the tool displays a source configurable process model (cf. Figure C.4). From the latter, you need to reproduce a target configurable process model (cf. Figure C.5) using the DELETE change patterns. Once you are finished with modeling, please use the *Finish-Modeling-Button* on the top left to proceed.

Please be aware that the edges in the given model may look differently from the ones you are creating with the tool.

Also, keep in mind that the model will be automatically laid out after applying a change pattern.

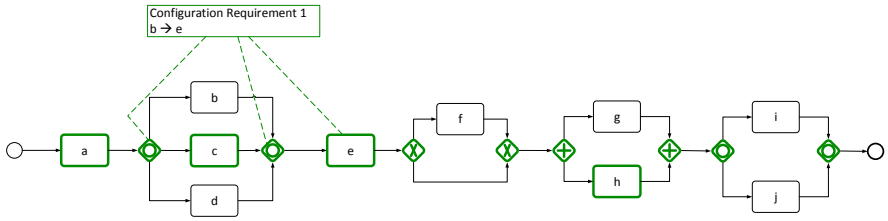


Figure C.4: Source configurable process model for the familiarization task 2

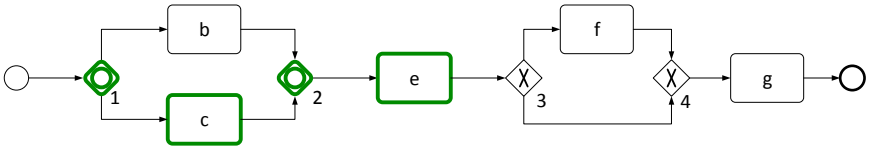


Figure C.5: Target configurable process model for the familiarization task 2

In order to help you in this task, we provide the solving path needed to obtain the target model from the source one. Each sentence of the solving path has the following format:
"Select (modeling elements to select) - CHANGE PATTERN TO APPLY"

1. Select (a) - DELETE Configurable Region
2. Select (Configuration Requirement 1) - DELETE Configuration Constraint
3. Select (d) - DELETE Configuration Alternative
4. Select (XOR3, f, XOR4) - DELETE Configurable Region All (Change to Non-configurable Gateways)

5. Select (AND, g, h, AND) - DELETE Configurable Region Keep [g]

6. Select (OR, I, j, OR) - DELETE Configurable Region None (Remove all the elements in the region)

Hint 1: When creating fragments between gateways, it does not matter whether activity "b" is the activity in the upper branch and "c" in the lower or the other way around.

Hint 2: When creating the model in the tool, it does not matter if the edges are not as sharp as in the model above.

Hint 3: To select modeling elements, you can drag the cursor over the elements or click each element plus the key CTRL of the keyboard. In both cases, selected elements will be highlighted in grey.

Hint 4: When pressing the *Layout* button in the panel dialog, the displayed model is reordered.

Hint 5: When pressing the *Undo* button in the panel dialog, the last performed operation is undone.

Hit 6: The sequence flows can be rearranged by dragging the startpoint (or endpoint) to another modeling element using the 'Select' tool from the Palette on the right hand side of the tool.

C.2.5 Modeling task 1 without CP4PF

The challenge in this task is to reproduce a configurable process model from the one depicted in the tool using the primitives provided in the palette. In particular, the tool displays a source configurable process model (cf. Figure C.6). From the latter, you need to reproduce a target configurable process model displayed in the auxiliar screen (cf. Figure C.7). Once you are finished with modeling, please use the *Finish-Modeling* button on the top left to proceed.

Please be aware that the edges in the given model may look differently from the ones you are creating with the tool.

Also, keep in mind that the model will be automatically laid out after applying a change pattern.

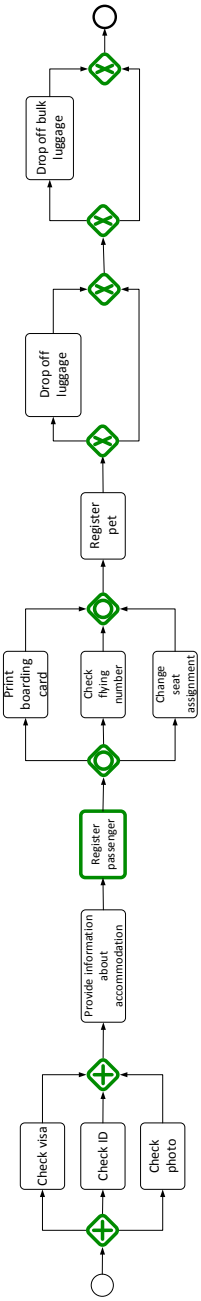


Figure C.6: Source configurable process model for modeling task 1

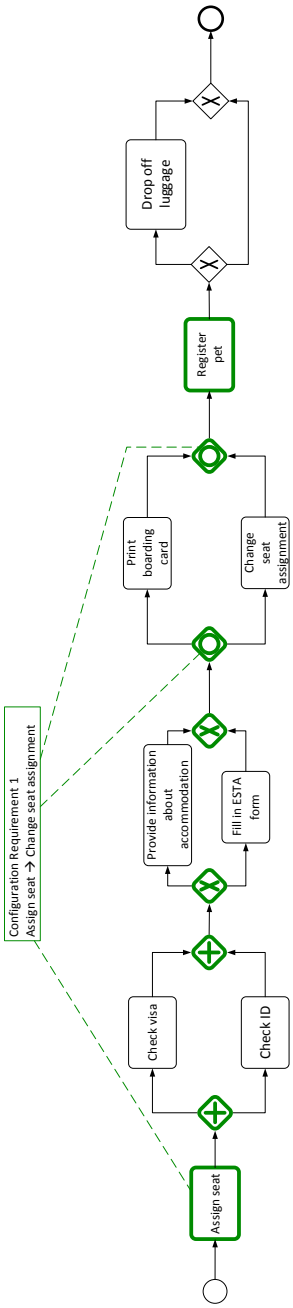


Figure C.7: Target configurable process model for modeling task 1

C.2.6 Modeling task 2 with CP4PF

The challenge is to reproduce a configurable process model from the one depicted in the tool using the provided set of change patterns. In particular, the tool displays a source configurable process model (cf. Figure C.8). From the latter, you need to reproduce a target configurable process model (cf. Figure C.9). Once you are finished with modeling, please use the *Finish-Modeling* button on the top left to proceed.

Please be aware that the edges in the given model may look differently from the ones you are creating with the tool.

Also, keep in mind that the model will be automatically laid out after applying a change pattern.

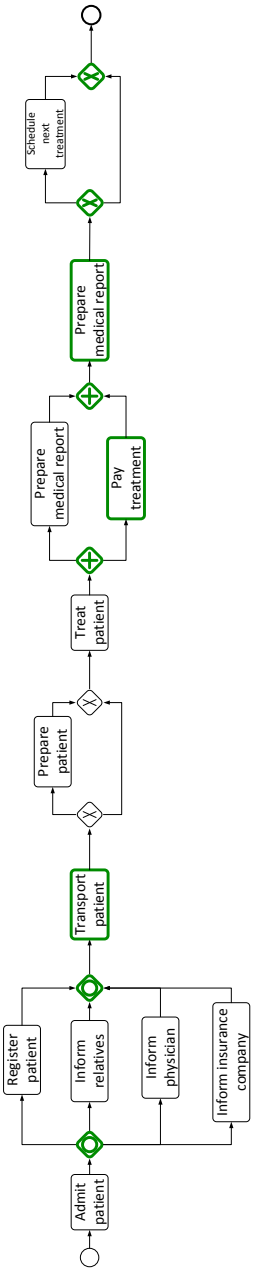


Figure C.8: Source configurable process model for modeling task 2

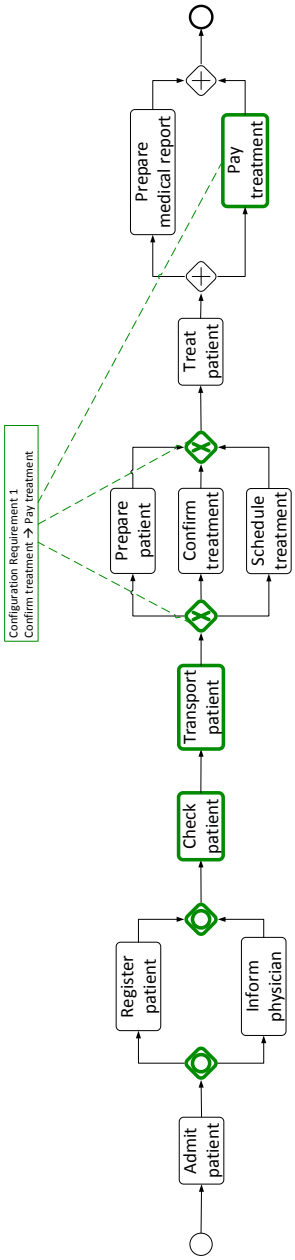


Figure C.9: Target configurable process model for modeling task 2

D

Cheetah Experimental Platform

This appendix contains the description of the *Cheetah Experimental Platform* (CEP) used in the validation of the CP4PF with PAIS engineers. CEP was originally defined by the *Quality Engineering Research Group* of the University of Innsbruck (Austria) [Pinggera et al., 2010]. We extended CEP by implementing the CP4PF in order to allow their application.

D.1 Design of CEP

Originally, CEP enables experimenters to quickly assemble experimental workflows from other components. In particular, CEP offers a set of frequently used components, including surveys, tutorials and editors for creating process models. In the context of the validation performed in this thesis, we followed the experimental process depicted in Figure D.1. When executing this process, CEP guides the user through the experimental process ensuring that the

setup is followed. Furthermore, data collected is stored on a central database server, giving researchers the possibility to check whether all activities were completed and to restore the experiment to a specific state (e.g., in case of a crashed system). If the database server cannot be accessed a local copy is created and the user is asked to send it to the experiment's supervisor via email. In addition, CEP ensures that all steps (or questions) marked as mandatory are followed (or answered) before the user continues with the next step in the process.

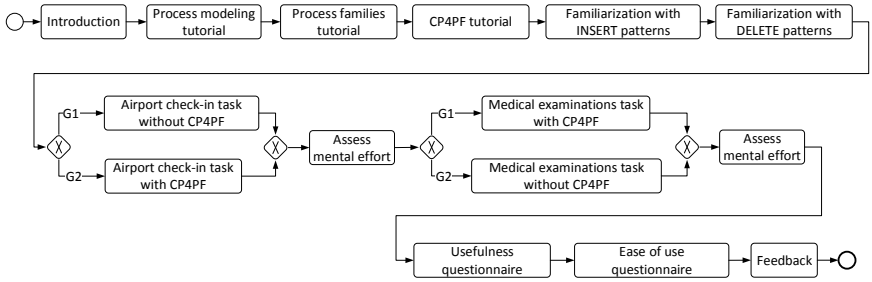


Figure D.1: Process followed in the validation with PAIS engineers

Besides monitoring the correct execution of the experimental process, CEP enables to gather the results of each questionnaire as well as to automatically record all the modeling actions. This is done to replay step by step what PAIS engineers did when applying the patterns. More precisely, every change to the configurable process model (e.g., add/delete/move configurable region, add/delete/move edge) and the corresponding timestamp are automatically recorded and stored separately, offering the possibility for detailed investigations concerning the application of the patterns.

D.2 Extension of CEP

In order to enable the investigation of how CP4PF are applied, we extended CEP by implementing an editor. This editor is a rather simple modeling component providing only basic modeling functionalities for simulating a "pen and paper" modeling session using a set of patterns. The focus was put on developing a tool facilitating the investigation of how CP4PF are applied,

rather than providing a full fledged modeling suite. In our editor, we decided to extend BPMN with C-EPC constructs because all the subjects were familiar with BPMN, but not with EPC or C-EPC. Thus, we enable the modeling of configurable tasks and configurable gateways with BPMN in order to apply CP4PF. Currently, C-EPC-like BPMN is the only process modeling language supported by CEP for creating configurable process models. Nevertheless, support for other notations was kept in mind when designing CEP and can easily be integrated. Figure D.2 includes a screenshots of the implemented editor.

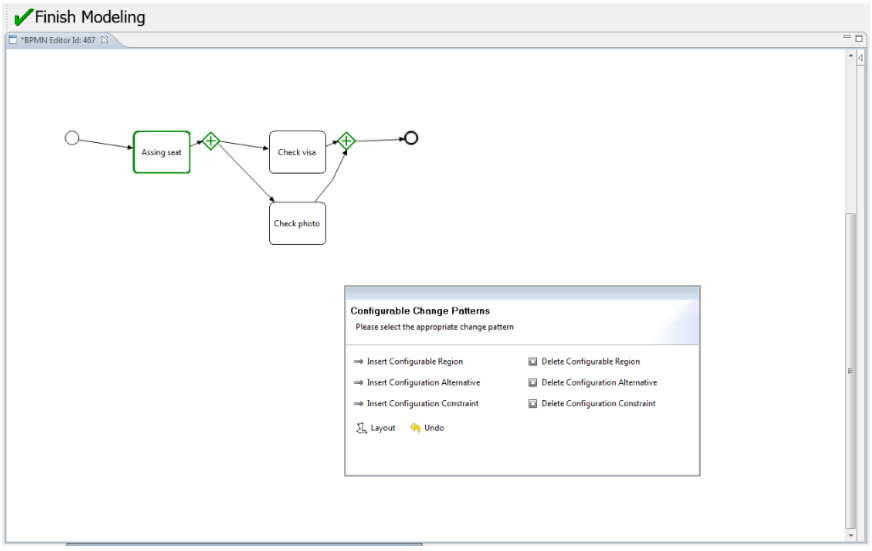


Figure D.2: Screeshot of the implemented editor

D.3 Analysis in CEP

In addition to efficiently executing and monitoring experiments, CEP allows for the analysis of the recorded data. This is done through the functionalities of the Cheetah Analyzer (e.g., data export features and means for replaying

process models).

To be able to analyze the data collected when executing the experimental process an export system is in place. By providing the option to export data as *Comma-Separated Values* (.csv) files, several tools for performing statistical analysis can be addressed (e.g., SPSS, Excel).

Another advantage of using CEP is the possibility of replaying process models created with the implemented editor. Recording all modeling steps enables researches to investigate how CP4PF are applied. For this purpose, the Cheetah Analyzer allows for a step-by-step execution of the steps performed during the experimental process. Additionally, researches can export the records using the *Mining XML* (.mxml) format, allowing them to apply process mining techniques using ProM [van der Aalst et al., 2007]. Figure D.3 shows an screeshot of the step-by-step records.

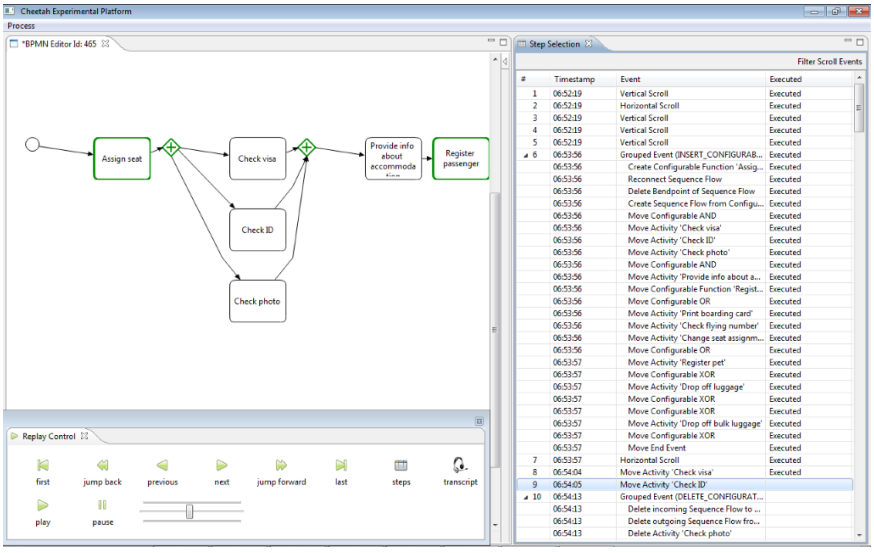


Figure D.3: Screeshot of the step-by-step records

www.pros.upv.es

Centro de Investigación en Métodos
de Producción de Software
Universitat Politècnica de València
Camí de Vera s/n, Edifici 1F, Dept. DSIC
46022 - València
Spain
Tel: (+34) 963 877 007 (Ext. 83530)
Fax: (+34) 963 877 359


P R O S
Centro de Investigación en Métodos
de Producción de Software


UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA