UNIVERSIDAD POLITECNICA DE VALENCIA

PROS
Centro de Investigación en Métodos de Producción de Software

Informe Técnico / Technical Report




Dealing with Variability in Business Process Models: An Evaluation Framework

Clara Ayora, Victoria Torres, Vicente Pelechano





Ref. #:	ProS-TR-2011-11				
Title:	Dealing with Variability in Business Process Models: An Evaluation Framework				
Author (s):	Clara Ayora, Victoria Torres, Vicente Pelechano				
Corresponding author (s):	cayora@pros.upv.es vtorres@pros.upv.es pele@pros.upv.es				
Document version number:	1.0	Final version:	Yes	Pages:	14
Release date:	March 2011				
Key words:	Business Process Modelling, Variability Modelling				

Dealing with Variability in Business Process Models: An Evaluation Framework*

Clara Ayora, Victoria Torres, and Vicente Pelechano

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain
{cayora,vtorres,pele}@pros.upv.es

Abstract. Business Process (BP) models are representations of real-world BP that gather relevant aspects for a specific purpose, usually organization or information system design. Apart from very specialized and concrete BPs, there is a large number of BPs, e.g. buying and selling, that are shared at a high level of abstraction by different domains but that differ in the way BPs are carried out, e.g. due to legal regulations. Related to the commonalities and differences that can appear during the BP modelling process, different approaches have been developed in the BP Modelling (BPM) research community. This paper gathers these existing approaches and presents an evaluation framework to compare and evaluate them according to their suitability for variability modelling. The framework is based on variability key concepts and quality factors and can be applied to any approach since it is not focused on specific aspects of the language, but focused on the variability concepts themselves.

Keywords: Business Process Modelling, Variability Modelling, Evaluation Framework

1 Introduction

Business Process models represent how organizational goals are achieved by means of tasks and resources. These models are commonly built either for organization design purposes and/or Information System design purposes. To take advantage as much as possible of these models, they should represent the organizational reality that is interesting for a specific goal in an accurate way. However, BP modelling is not a trivial task [1]. It not only requires mastering the problem domain being represented, but also the language or notation that is being used to perform this representation. In addition to this complexity, the different variations of a specific model based on the application context[1] turn models

* This work has been developed with the support of MICINN under the project EVERYWARE TIN2010-18011.

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

into artifacts that are more complex to build, to manage and to understand. In these cases, considerations such as process granularity, context dependency, or scalability turn into first-order requirements during the modelling process.

In order to deal with this complexity and to facilitate the construction and management of these models, many efforts from the BPM research community have been conducted ([2], [3], [4], [5], [6]). In these works, through the use of different techniques (such as feature models) and approaches (such as operational or model projection), the authors deal with the problems that arise with variability during the process modelling and/or execution (which involves also considering process evolution techniques such as the ones presented in [7]). In this work, we have considered those approaches that have been designed to deal with business process variability modelling, independently of the mechanisms used for this purpose. The objective is to provide a clear picture of the different possibilities while variability modelling and also understanding the suitability of each approach according to the context of use. For this purpose, we have analyzed these approaches based on a set of properties that reflect the issues that need to be addressed in order to properly handle BP variability at the analysis/design phase. These properties have been defined based on a refinement of the requirements already stated in [8] to support model variability. In addition, the properties are presented as an evaluation framework which allows us to identify the strengths and the weaknesses of the selected approaches. This framework can help organizations in the decision of which approach better suits their needs (the BP modelling process of each organization may vary according to its characteristics, e.g. starting the modelling process from scratch, level of expertise of its business modellers, etc.). The framework can be applied to any approach since it is not focused on specific aspects of BPML but focused on the variability concepts themselves.

The remainder of the paper is structured as follows. Section 2 describes the impact that variability has within the analysis/design phase of the BP lifecycle. Then, section 3 presents the evaluation criteria that make up the framework and that it is used to evaluate the different approaches. Section 4 proceeds with such evaluation criteria and for each of the evaluated approaches provides a brief description about its main characteristics. Section 5 provides an analysis of the evaluation and presents the conclusions that can be extracted from it. Finally, section 6 closes the paper with the conclusions and outlines further research directions.

2 Variability within the BPM analysis/design phase

Within the four main phases that make up the BPM lifecycle [9], in this work, we focus on the first phase which deals with BP analysis/design. Specifically, we reflect on the impact of dealing with BP variability in this phase. During the analysis/design phase, and based on domain requirements, BPs are identified, reviewed, validated, and represented by means of BP models. These models are built by means of Business Process Modelling Languages (BPML) such as Petri

nets, EPC, YAWL, BPMN or UML Activity Diagrams. However, the original constructs provided by these languages do not provide enough expressivity to properly deal with BP variability (e.g. the Boolean conditions do not allow the nature of the different choices to be described [10]). According to [8], "Being aware of variability and dealing with it consciously is an important prerequisite of variability modelling", therefore, BPMLs should allow model variability as such to be represented, bringing the concepts related to variability as first-order concepts of the language.

Within the Software Product Lines (SPL) field, where variability management constitutes a key aspect, the questions that may be set out to characterize variability are: (1) what does vary?, (2) why does it vary?, and (3) how does it vary? [8]. The first question refers to the precise identification of the items or properties of the real world that are variable. In terms of a BP representation these items refer to the different language elements that make up a BP model, that is language elements such as tasks, control flow, roles, documents or even compositions of these basic elements. The second question refers to the different reasons that make an item or property to vary. In a BP model these reasons are usually represented by the conditions which are determined by the value of the associated variables. Finally, the third question refers to the different shapes that a variability subject can adopt. In terms of a BP model these shapes refer to the different alternatives that can be applied in a specific part of a model.

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

Based on this set of considerations, next section presents an evaluation framework to analyze the support provided by the existing approaches to deal with BP variability modelling issues.

3 Evaluation framework

To evaluate the existing approaches that deal with BP variability modelling, we first need an evaluation framework that establishes the concepts that BPMLs need to address in order to properly deal with variability during the BPM analysis/design phase. In [11] a set of general requirements to deal with variability issues through the complete BPM lifecycle were introduced. However, these requirements are too wide and general to perform a more concrete and objective evaluation about these issues. Therefore, a more detailed and specific evaluation criteria needs to be defined. It has motivated us to propose an evaluation framework that specifies the set of properties that should be satisfied by any approach to properly deal with BP variability. These properties have been defined based on the central concepts introduced in [8] for modelling variability, but focusing on the BP artefacts. The evaluation criteria that we propose have been orga-

nized in two building blocks which refer to: (1) concepts and (2) quality factors. While the first block gathers a set of concepts that need to be addressed by BPMLs to properly deal the BP modelling task, the second block gathers desirable properties that contribute positively to improving the BPML with regard to its adoption.

- **Concepts**. In order to bring variability concepts as first-class constructs, BPMLs should provide the mechanisms that allow the specification of these concepts.
  - **EC1. Variation point**
    A variation point is defined as a precise position within a process model that admits different possibilities according to the current context or situation.
  - **EC2. Process fragment**
    A process fragment is defined as an alternative that can be applied in a specific *variation point* for a particular context.
  - **EC3. Process fragment context**
    A process fragment context is defined by the set of domain variables that make a particular process fragment to be instantiated.
  - **EC4.Process fragment relationships**
    A process fragment relationship is defined as a constraint between two or more *process fragments* that establishes the proper use of the *process fragments* involved within a specific context (e.g. mutual exclusion or implication).
  - **EC5. Language support regarding variability**
    The support provided by the language to specify variability should include any of the elements that make up a process model (control flow, physical objects, data, roles, events, etc.).
  - **EC6. Process context regarding variability**
    The process context regarding variability is defined as the set of domain variables that have an impact on process variability design and/or execution. The value of these variables determines the current state of a BP according to BP variability.
  - **EC7. Variation point resolution time**
    The language should allow modellers to distinguish between *variation points* whose resolution depends on the initial context (design time) or on the current context of an instance process (run time).
- **Quality factors**. This block gathers a set of quality factors that are desirable in the different BPMLs to ensure their successful adoption.
  - **EC8. Flexibility**
    In order to deal with *process fragment* evolution, the language should provide mechanisms to easily make changes over the model, in particular those parts related to *process fragments*.
  - **EC9. Scalability**
    The language should provide techniques that allow BP models to be

evolved without losing the ability to handle a great number of model variants[2].

- **EC10. Legibility**
  The language should provide graphical elements to easily distinguish between model commonalities and model individualities.
- **EC11. Understanding**
  The language should provide abstractions close to the concepts used to represent BP variability. A correct abstraction of these concepts will result in more easily understandable models.
- **EC12. Low learning curve**
  The language should be easy to learn. It is easier to learn a language when it is based on a well-known and commonly used language. Moreover, if this well-known language is a standard language, we can take advantage of its benefits such as existing tools, guidelines, etc.
- **EC13. Separation of Concerns**
  The language should allow modellers to manage *process fragments* separately from the base model (the model that gathers commonalities). This separation contributes by reducing the change impact on the process model, reusing the existing *process fragments*, and improving the understanding of the process model ([12],[13]).
- **EC14. Tool support**
  The language should be supported by a tool since this support is usually critical for language adoption. In addition, it facilitates the management of the variability, specifically during the analysis/design phase.
- **EC15. Guideline support**
  The existence of guidelines that assist users during variability management in BP models facilitates the learning process and the use of the language. Specifically, these guidelines are desirable during the analysis/design phase while defining model commonalities, process fragments, and configuration. As a consequence, these guidelines turn the modelling process into a more objective task, independent of the modeller's skills or experience.

4 Evaluation of the approaches

The goal of this section is to provide a brief explanation of the existing approaches that deal with BP variability and to evaluate them against the framework presented above.

The approaches that have been evaluated in this paper are C-EPC [2], PESOA [14], and Provop [15]. All these approaches have been put into practice (see *Case Studies Development*[3]) by means of three different case studies from different domains, all extracted from the reviewed literature. These case studies include (1) a service process that handles vehicle repair in a garage [16], (2)

[2] a model variant represents an individualization of the process model

[3] `http://arxiv.org/vc/arxiv/papers/1102/1102.4518v1.pdf`

a simplified version of a healthcare process that represents a cruciate rupture treatment [7], and (3) an e-business shop [14]. This exercise has allowed us to better understand BP variability requirements at the modelling level and also to detect some deficiencies of the analyzed approaches while applying the different case studies.

C-EPC (Configurable EPC) C-EPC is an extension of EPC (Event-driven Process Chain) that includes new constructs to represent variability in reference process models [2]. These new constructs allow multiple model variants to be represented within a single model which needs to be configured[4] prior to its deployment. This single representation of multiple model variants is achieved by combining the use of (1) *configurable nodes* (functions and connectors), which allow different behaviours to be specified depending on the context of use with (2) *configuration requirements* and *guidelines*, which state, by means of logical predicates, the valid configurations of the model.

Even though other approaches such as C-YAWL [10] and Feature-EPC [3] differ from C-EPC in the techniques used to obtain an individualization of the model (*Hiding and Blocking* for C-YAWL and *Feature Models* for Feature-EPC), all share the idea of having a single model to represent all model variants. However, since C-EPC is widely known and extended, we have decided to evaluate it as a representation of this group of approaches.

The evaluation of the C-EPC approach against the evaluation criteria defined in section 3 is presented below:

- EC1. *Variation point:* It is possible to clearly identify them by means of *configurable functions* and *connectors*.
- EC2. *Process fragment:* It is not possible to clearly identify which process fragments are being considered in the process from the model. This information is scattered among model functions, events, and connectors.
- EC3. *Process fragment context:* It is not possible to specify this context in terms of domain variables. Specifically, the process fragment context is defined in terms of the configuration of other model elements by means of *configuration requirements* and *guidelines*. This hinders the identification of the specific underlying condition(s) that make a specific process fragment to be instantiated.
- EC4. *Process fragment relationships:* It is not possible to explicitly define relationships between process fragments since C-EPC does not allow process fragments as such to be defined. Moreover, although *configuration requirements* define relationships between functions and connectors, this information is scattered throughout the entire model which implies that these relationships are neither clear nor transparent to the modellers.

[4] La Rosa defined a questionnaire-driven approach in [17] to reduce the complexity that the C-EPC model configuration task entails.

- EC5. *Language support regarding variability:* It is possible to define variability related to different language elements such as control flow, functions, and connectors. In addition, the language variability scope was extended in [18] to also include roles and physical objects.
- EC6. *Process context regarding variability:* It is not possible to define the process context regarding variability in terms of domain variables. C-EPC does not provide enough expressiveness to explicitly define the set of variables that determine the process context. Again, it is scattered throughout *configuration requirements* and defined in terms of the configuration of other elements of the model.
- EC7. *Variation point resolution time:* C-EPC is designed for model configuration prior to model deployment, and the expressivity provided to deal with variability is not used to represent the variability that may appear during process execution. Therefore, this evaluation criteria does not apply to the C-EPC language.
- EC8. *Flexibility:* It is not possible to evolve or change C-EPC models easily since they are integrated representations where model variants cannot clearly be identified.
- EC9. *Scalability:* It is not possible to evolve the model when handling a great number of model variants. Even though *configurable nodes* allows a great number of model variants to be represented jointly, this joint representation makes variant evolution difficult since one change in a specific part of the model may affect related configurable model elements.
- EC10. *Legibility:* It is partially possible to distinguish model commonalities and individualities since model variants are not easily extracted from the diagram. However, model elements that are related to model variability (i.e. *configurable nodes* and *configuration requirements* and *configuration guidelines*) can be identified graphically in the model.
- EC11. *Understanding:* It is possible to understand C-EPC since it provides the appropriate abstractions to represent model variants. These abstractions are *configurable nodes* (which can be valued as included, excluded, or conditionally skipped), *configuration requirements* and *configuration guidelines*.
- EC12. *Low learning curve:* It is partially possible to learn the language quickly. The learning process of the new language elements is fast due to the closeness to the original EPC language elements (*configurable nodes* constitute small graphical variations to the original nodes). However, this speed does not apply to the definition of *configuation requirements* or *configuration guidelines* which requires some knowledge about logical predicates.
- EC13. *Separation of concerns:* It is not possible to manage process fragments separately from the base model. The C-EPC approach forces model commonalities and individualities to be handled jointly.
- EC14. *Tool support:* There is no available tool to perform the design of C-EPC models. Nevertheless, [19] presents a tool (C-EPC Validator) to check the compliance of a C-EPC model configuration against *configuration requirements* and *configuration guidelines*.

– EC15. *Guideline support:* Despite the fact that there exist documentation about the language [2], this documentation does not provide clear guidelines that assists modellers during the model design and configuration tasks.

Variant-Rich Process Models (within the PESOA project) PESOA (www.pesoa.org) is a cooperative project that was carried out by a group of companies and academics whose main objective was the investigation of an approach for the development and customization of families of process-oriented software. As a result, by means of focusing on the design level and taking into account the relevance of the reusability aspect, a set of *basic* and *composite* variability mechanisms were identified [14]. The *basic* set includes (1) encapsulation of varying subprocesses, (2) addition, replacement, omission of encapsulated subprocesses, (3) parameterization, and (4) variability in data types. The *composite* set includes (5) inheritance, (6) design patterns, and (7) extensions/extension points. This set was transferred to different languages such as UML Activity Diagrams, UML State Machines, BPMN, and Matlab/Simulink.

The evaluation of the PESOA approach against the evaluation criteria defined in section 3 is presented below:

– EC1. *Variation point:* It is possible to identify them in the model by means of the use of the «VarPoint» label (stereotype), which is attached to those places (i.e., tasks) where variability may occur. According to the semantics associated to process fragment substitution, these labels may vary into one of the following stereotypes: «Abstract» and «Null».
– EC2. *Process fragment:* It is possible to identify process fragments in the model by the introduction of the «Variant» stereotype, which is attached to those tasks that can fit in a *variation point*. This stereotype may change into the stereotypes «Default», «Alternative» and «Optional» depending on the process fragment behaviour.
– EC3. *Process fragment context:* It is partially possible to express it since the context is not explicitly represented by domain variables, but it can be represented by means of features which are associated to the different process fragments defined in the model.
– EC4. *Process fragment relationships:* It is partially possible to specify relationships between process fragments due to the limited support provided. Only two different types of relationships are defined: *implementation*, to associate possible resolutions to the variation points; and *inheritance*, to express alternative behaviour which adds or removes elements regarding specific rules.
– EC5. *Language support regarding variability:* It is not possible to support variability issues beyond control flow and tasks. It does not take into account the variability associated to other language elements such as roles, objects or events that can arise.

- EC6. *Process context regarding variability:* It is partially possible to determine the current state of a BP regarding variability by evaluating the feature values associated to the different process fragments included in the model.
- EC7. *Variation point resolution time:* It is not possible to distinguish whether a variation point is solved at design time (depending on the initial context) or at run time (depending on the current instance context).
- EC8. *Flexibility:* It is not possible to evolve or change process fragments easily. Since process model commonalities and individualities are defined together, changes in process fragments may imply reconsidering other parts of the model, which makes process fragment evolution difficult.
- EC9. *Scalability:* It is not possible to properly handle the evolution of a great number of model variants since process model commonalities are modelled jointly with model individualities. The higher the level of the variants that a model has, the more difficult they become to evolve.
- EC10. *Legibility:* It is possible to graphically differenciate model commonalities from individualities. Even though process fragments are integrated in the model, the enrichment of task/subprocess elements by means of stereotypes allows this differentiation.
- EC11. *Understanding:* It is not possible to easily comprehend the language. The stereotypes introduced to represent concepts related to BP variability are not suitable. Their associated semantics is broader than the semantics of the word that represents the stereotype. For instance, the «Null» stereotype indicates not only task/subprocess optional behaviour (behaviour also associated to the «Optional» stereotype) but also restricts the number of possible process fragments associated to the variation point to one process fragment.
- EC12. *Low learning curve:* It is not possible to learn the language easily. The broad semantics associated to the introduced stereotypes takes time to learn and use properly, which results in a steep learning curve.
- EC13. *Separation of Concerns:* It is partially possible to separate model commonalities from individualities. The PESOA approach forces them to be handled jointly, but, nevertheless, variation points and their different alternatives are labelled through stereotypes that allow individualities to be distinguished and managed separately from commonalities.
- EC14. *Tool support:* The approach has been implemented as an Eclipse plugin [20]. This tool provides support for configuring a feature diagram as well as for applying the selected configuration to the process model.
- EC15. *Guideline support:* [20] provides a methodology and guidelines for the application of the PESOA approach.

Provop (PROcess Variants by OPtions) Provop is an operational approach for managing large collections of process variants during the process life cycle. It was motivated by the fact that a "process variant can be created by adjusting (configuring) a given process model to a given context" [21]. These variant-specific adjustments are expressed by means of a set of high-level *change operations* (INSERT, DELETE, MOVE and MODIFY) [15]. Furthermore, Provop

allows more complex process adjustments by grouping multiple *change operations* into so-called *options* [16]. Thus, a specific process variant is determined (configured) by applying one or more of these *options* to the process model. The *options* required to configure a process variant are chosen when the process context is evaluated. Provop provides a model for capturing this process context by means of *context variables*, which represent different domain dimensions of the context.

The evaluation of the Provop approach against the evaluation criteria defined in section 3 is presented below:

- EC1. *Variation point:* It is partially possible to identify them within the model. When it refers to INSERT and DELETE model changes, variation points can be clearly identified by means of *adjustment points*, which are represented graphically in the diagram by black diamonds. However, this does not occur when the operation refers to MOVE or MODIFY changes.
- EC2. *Process fragment:* It is possible to identify them clearly by means of the specification of the proposed *change operations*.
- EC3. *Process fragment context:* It is possible to represent it by means of the context variables that are defined by a name, a description, a value range, and a mode (*static* or *dynamic* depending on whether or not its value changes during the execution of the process).
- EC4. *Process fragment relationships:* It is possible to define relationships such as implication, mutual exclusion, application order, hierarchy, and at-most-n-out-of-m-options [22] between the *options* that make up a model variant.
- EC5. *Language support regarding variability:* It is not possible to manage variability aspects beyond language elements such as control flow, functions, and connectors.
- EC6. *Process context regarding variability:* It is possible to express it by means of the context variables, which capture the different domain dimensions of the context.
- EC7. *Variation point resolution time:* It is possible to distinguish when variation points are solved through the context rules.
- EC8. *Flexibility:* It is possible to evolve or change models easily by means of the definition of new *options*. For instance, to obtain a new process variant, it is only necessary to define a new set of *options*.
- EC9. *Scalability:* It is possible to handle and evolve a great number of model variants thanks to the separate representation used in Provop. Moreover, this evolution is also facilitated by the *change operations* and *options*.
- EC10. *Legibility:* It is possible to clearly identify model commonalities and individualities since they are specified separately. Model commonalities are represented in a base model while model individualities are represented separately by means of *options*.
- EC11. *Understanding:* It is possible to understand the language since Provop provides an adequate set of abstractions to represent model variability. These

abstractions are *adjustment points* (to identify variation points within the model), *change operations* (to define the difference between the basic process model and its individualization), and *options* (to group *change operations*).
- EC12. *Low learning curve:* It is possible to learn Provop easily due to the simplicity of the *change operations* (INSERT, DELETE, MODIFY and MOVE) and how the model variants are built.
- EC13. *Separation of Concerns:* It is possible to manage process fragments separately from the base model. The Provop approach provides a proper separation of concerns since process fragments are clearly defined and identified by means of *change operations*.
- EC14. *Tool support:* A proof-of-concept prototype has been implemented based on the ARIS Business Architect modelling tool. The prototype introduces the facilities for process variant configuration and management [15].
- EC15. *Guideline support:* There are clear guidelines that explain step by step how to model and configure BP models [16]. Also, there exists documentation ([21], [15]) that gives a complete and formal description of the approach.

5 Analysis of the evaluation results

In the previous section, we have evaluated the existing approaches that deal with variability at the analysis/design phase. The results of this evaluation are summarized in Table 1, where a set of symbols has been used to specify the fulfillment of each evaluation criterion: a "+" indicates that the approach completely fulfills the evaluation criterion; a "-" indicates that it is not fulfilled; a "+/-" indicates that it is partially fulfilled and "na" indicates that the criterion is not applicable to the approach.

	Concepts							Quality factors							
	EC1	EC2	EC3	EC4	EC5	EC6	EC7	EC8	EC9	EC10	EC11	EC12	EC13	EC14	EC15
C-EPC	+	-	-	-	+	-	na	-	-	+/-	+	+/-	-	-	-
PESOA	+	+	+/-	+/-	-	+/-	-	-	-	+	-	-	+/-	+	+
Provop	+/-	+	+	+	-	+	+	+	+	+	+	+	+	+	+

Table 1. Summary of the approaches evaluation

5.1 Concept analysis

According to the results shown in Table 1, none of the evaluated approaches provides complete support to the entire set of evaluation criteria considered in the concepts block.

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

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

In contrast to C-EPC, in PESOA and Provop we find mechanisms to specify variability as first-class elements of the language. However, both these approaches delimit their scope in order to address variability regarding control flow and tasks (EC5), not to consider the variability associated to other language elements that can occur such as roles, objects or events. In addition, PESOA and Provop are conceived to deal with variability that can be solved either at design time (variability that depends on the initial context of the process) and at run time (variability that depends on the current context of a process instance). However, neither of these approaches provides mechanisms to differentiate in the model between these two types (EC7). This differentiation would delimit the set of decisions that need to be taken prior to model deployment.

5.2 Quality factors analysis

As Table 1 shows, only the Provop approach provides a complete fulfillment of the evaluation criteria included in the quality factor block (EC8-EC15). A key factor for this fulfillment is the application of a separate approach where model commonalities are specified apart from model individualites. This approach contributes positively to the flexibility (EC8) and scalability (EC9) of the model. On the contrary, the joint approach followed by C-EPC and PESOA contributes negatively to the flexibility (EC8) and scalability (EC9) factors of the models. The application of the joint or separate approach is also directly related to the fulfillment of the separation of concerns criterion (EC13).

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

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

6 Conclusions and further work

In this work, we have presented the first evaluation framework to evaluate and compare different approaches for BP variability modelling. It is the result of an analysis made on the requirements that need to be addressed when dealing with variability at the analysis/design phase. For the definition of this framework, we have taken into account not only variability concepts, but also quality factors that contribute to the successful adoption of an approach. In addition, the framework can be applied to any BPML that deals with variability modelling and, even more, to other variability modelling languages since it is not focused on BPMLs aspects, but on how languages deal with variability. Specifically, we have applied the framework to three BPM approaches that deal with variability differently. A complete fulfillment of the evaluation criteria included in this framework would mean that the approach is able to provide BP variability modelling coverage in a broad sense. This coverage implies considering variability modelling issues related to model structure and behaviour, language elements, and model resolution time. However, after analyzing the selected approaches, from a modelling point of view, we can determine that none of them provides support for the entire set of variability concepts. This hinders putting into practice any of these approaches when the required variability is broader than the one provided by them. This forces their combination to support variability in a broad sense. However, this combination is not easy to achieve since the mechanisms used by each of the approaches are not easily and directly transferred to other BPMLs. Therefore, an approach that provides this complete support becomes necessary. According to the requirements identified in section 3, this approach should provide support for the following issues:

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

In future work, we plan to extend the presented framework for the remaining phases of the BPM lifecycle. This involves also considering variability issues during the enactment and monitoring/evaluation phases. During the enactment phase, considerations such a strategy for process fragment selection and execution should be addressed. Also, during the monitoring/evaluation phase, support to capture variability information produced during the enactment phase must be provided. Then, this information can be processed to correct design errors

within the model. In addition, based on these requirements identified for all BPM phases, we plan to design a complete approach to deal with variability in BP through the entire BPM lifecycle.

References

1. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Information & Software Technology **52**(2) (2010) 127–136
2. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1) (2007) 1–23
3. Vervuurt, M.: Modeling business process variability : a search for innovative solutions to business process variability modeling problems. Student Theses of University of Twente. October 2007
4. Schnieders, A., Puhlmann, F.: Variability mechanisms in e-business process families. In: BIS. (2006) 583–601
5. Hallerbach, A.: Management von Prozessvarianten. PhD thesis (2009)
6. Adams, M.J.: Facilitating Dynamic Flexibility and Exception Handling for Workflows. PhD thesis, Faculty of Information Technology. Queensland University of Technology (2007)
7. Weber, B., Sadiq, S.W., Reichert, M.: Beyond rigidity - dynamic process lifecycle support. Computer Science - R&D **23**(2) (2009) 47–65
8. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. 1 edn. Springer (September 2005)
9. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Number 978-3-540-73521-2. Springer-Verlag Berlin Heidelberg 2007 (2007)
10. van der Aalst, W.M.P., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M.H.: Configurable process models as a basis for reference modeling. In: Business Process Management Workshops. (2005) 512–518
11. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process lifecycle. In: 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08). (June 2008) 154–161
12. Myllymäki, T.: Variability management in software product lines (12 2001)
13. Stan Bühne, Kim Lauenroth, K.P.: Modelling requi variability across product lines (November 2005)
14. Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: Variability mechanisms for process models. Technical report, BMBF-Project (2006)
15. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: The provop approach. Software Process: Improvement and Practice (2010)
16. Alena Hallerbach, Thomas Bauer, M.R.: International Handbook on Business Process Management. In: Configuration and Management of Process Variants. Springer (2010)
17. Rosa, M.L.: Managing Variability in Process-Aware Information Systems. PhD thesis, Queensland University of Technology (2009)
18. Rosa, M.L., Dumas, M., ter Hofstede, A.H.M., Mendling, J., Gottschalk, F.: Beyond control-flow: Extending business process configuration to roles and objects. In: ER. (2008) 199–215
19. Validator, C.E.: http://www.mendling.com/epml/c-epc-validator.xsl
20. Bayer, J., Gerard, S., Haugen, Ø., Mansell, J.X., Møller-Pedersen, B., Oldevik, J., Tessier, P., Thibault, J.P., Widen, T.: Consolidated product line variability modeling. In: Software Product Lines. (2006) 195–241

21. Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants. In: 3rd International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008). (June 2008) 31–40
22. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process lifecycle. In: 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08). (June 2008) 154–161