Software and Systems Modeling (2022) 21:2517–2553
https://doi.org/10.1007/s10270-022-00985-3

REGULAR PAPER

Model-driven management of BPMN-based business process families

Andrea Delgado[1] · **Daniel Calegari**[1] · **Félix García**[2] · **Barbara Weber**[3]

Received: 22 December 2020 / Revised: 18 December 2021 / Accepted: 25 January 2022 / Published online: 19 March 2022

Abstract

Business processes can have variants depending on specific business requirements, which lead to the definition of a so-called *business process family*. Since conventional business process modeling languages, e.g., the Business Process Model and Notation (BPMN), do not explicitly support variants' specification, several proposals have emerged to deal with it. However, they mainly focus on languages' definition, while less emphasis is made on providing complete variability management. This article presents a Model-Driven Engineering approach for managing BPMN-based business process families composed of a metamodel for conceptualizing process families, a high-level process for managing them (involving model transformations for the configuration of variants), and tool support for the complete approach. We validated the proposal using a real-world example from a university and an empirical study with real users. Users rated the support tool's principal functional suitability and usability features as very good. Many improvement opportunities were detected, e.g., version control, collaborative work, and error reporting. We also provide a literature review and thorough evaluation of BPMN-based business process families' proposals using the VIVACE framework.

Keywords Business process families · Variability · BPMN 2.0 · Model-Driven Engineering · VIVACE

1 Introduction

New technological advances provide organizations with renewed support and opportunities to enact their business processes and gather execution data to improve their business. Business Process Management (BPM) [23,69,71] provides support for the business process life cycle from modeling, configuration, implementation, enactment, and operation, to evaluation and improvement. However, many organizations have their processes embedded in traditional systems with no explicit definition or management of the business process control flow, roles, or associated organizational data.

On the contrary, a Process-Aware Information System (PAIS) [22] provides support for managing and executing operational processes based on process models. Business Process Management Systems (BPMS) [10] take process models as the main input for the complete life cycle support. A BPMS supports different languages like the Business Process Model and Notation (BPMN 2.0) [47] standard, which organizations widely adopt today based on the large number and versatility of the existing offer of this type of platform. Within this type of system, processes are very structured and predictable, i.e., users know a priori possible paths and actions to be taken. Structured processes can be viewed as an advantage since they are easier to support than unstructured ones [60].

Modeling and executing processes can be challenging as organizations and their systems gain complexity, e.g., in a worldwide context or with complex collaborative inter-organizational processes involving several participants. Flexibility is needed to be able to manage different types of processes within a PAIS. In [57] four flexibility needs for PAISs to support flexible processes are introduced: i) vari-

Communicated by Martin Glinz.

✉ Félix García
felix.garcia@uclm.es

Andrea Delgado
adelgado@fing.edu.uy

Daniel Calegari
dcalegar@fing.edu.uy

Barbara Weber
barbara.weber@unisg.ch

[1] Universidad de la República, Montevideo, Uruguay

[2] University of Castilla La Mancha, Ciudad Real, Spain

[3] University of St. Gallen, St. Gallen, Switzerland

ability, different variants of the same process where the course of actions is known a priori, i.e., prespecified processes; ii) looseness, when the specific course of action is mostly not known a priori; iii) adaptation, the ability to adapt the process and its structure to emerging events; and iv) evolution, the ability to change when the process evolves.

In this article, we focus on process variability. It appears when several processes share many common elements while others are different depending on some **context**, leading to specific **process variants** which are members of a so-called **business process family** (BP family). Although there is a common **base process**, the specific course of action differs depending on which specific **variants** are selected for the many **variation points** defined for the BP family. Most modeling languages and tools do not support the modeling of BP families natively. Process variants have to be modeled as separate models or within one model but based on using conditional branchings for selecting the specific path of execution for each one [57].

Dealing with each process variant as a separate process leads to duplication of models and increases the maintenance effort due to replicating changes in common parts and failing to see and conceptualize the whole picture regarding the BP family. Using the conditional branching approach leads to complex and less understandable models. On the contrary, modeling the complete BP family, i.e., to model variability explicitly, prevents those issues. Moreover, this kind of modeling allows the introduction of new process variants quickly by adding specific elements associated with the base process's variation points. Obtaining a particular process variant reduces selecting a specific variant for each variation point, known as configuration [57].

Several approaches to deal with variability emerged in recent years [60,68], mainly focused on languages' definition, with less emphasis on providing complete variability management, i.e., on the full description of activities, actors, and artifacts for the specification of a BP family, its configuration, and the automatic generation of process variants. Moreover, although organizations have detected the need to apply BP families' general vision to manage their processes' variability [6,44], they are not yet ready to introduce these concepts, partly due to the lack of support tools.

In this context, our work aims in the long term to improve support for managing BP families, reducing technical complexity to end-users, increasing usability, reuse, and productivity, and decreasing errors within variants' configurations. We considered the Model-Driven Engineering (MDE) [34] paradigm for two reasons. First, we are dealing with BP families based on BPMN 2.0, formally defined using the standard Meta-Object Facility (MOF, [49]). Thus, there is a formal definition of modeling languages used within the proposals. Second, model transformations, e.g., using high-level languages like ATLAS Transformation Language (ATL) [33], can perform the automatic configuration of variants and thus simplify the configuration process.

Based on this, we identified the following research question for our work:

How can MDE support BPMN-based business process families management?

We followed the Design Science research method [28,53,72] which defines activities for design and build artifacts to solve a given problem evaluating their usefulness to do so. Design Science validation of artifacts can be carried out following different suitable approaches regarding the artifact under evaluation. For example, to assess artifacts' usefulness, e.g., of a methodology or a tool within an organization, empirical methods can be used [72,74,75]. Assessment can have human support through surveys, interviews, or experiments, or they can be technology-oriented using benchmarks or theoretical validations over algorithms or tools. We carried out a problem-centered approach, identified and defined the problem (c.f. Sect. 3), specified objectives for the solution, designed and developed corresponding artifacts (c.f. Sect. 4), and assessed their usefulness to solve the problem (c.f. Sect. 5).

In previous work, we have presented two BPMN-based approaches that use MDE for modeling and configuring BP families: BPMNext [18], an extension of BPMN 2.0 inspired by [41], and the adaptation [8] of the Common Variability Language (CVL) [48], and its successor the Base Variability Resolution approach (BVR) [27], to BPMN 2.0. In both cases, we provide ATL model transformations for the automatic generation of process variants. We also reviewed many BPMN-based approaches to put our work in context, and, for the CVL/BVR approach, we evaluate it using the VIVACE framework [5]. VIVACE is a framework for evaluating and comparing process variability approaches and their support of the business process life cycle.

This article builds upon our previous work and extends it in several ways, aiming to answer the research question introduced above. Its contributions are twofold:

1. A literature review and evaluation of BPMN-based BP family approaches. The review is an extended and revised version of the one in [9]. The evaluation uses a refined version of the VIVACE framework and a BP family from a real university case [19].
2. A generic approach for managing BP families composed of a metamodel conceptualizing key elements for BP families, a high-level process for its management involving model transformations for the configuration of variants, and tool support for the complete approach.

We validated the complete proposal through a proof-of-concept with the BP family from a real university case and an empirical assessment over the same BP family with real

users, considering the approach and tool support's characteristics.

The rest of this paper is organized as follows. In Sect. 2, we present an illustrative scenario that is used throughout the paper. In Sect. 3, we summarize the literature review and evaluation of BPMN-based variability approaches. In Sect. 4, we present our proposal for model-driven management of BPMN-based BP families, and in Sect. 5 we provide validation of such proposal. In Sect. 6 we discuss many aspects related to the proposal. Finally, in Sect. 7, we present some conclusions and an outline of future work.

2 Illustrative scenario

This section presents a real-world BP family, which is used as an illustrative scenario throughout the article. We conducted a research project within Universidad de la República, aiming at experiencing a paradigm shift from traditional information systems to PAIS [19]. We worked with domain experts from the General Directorate of Cooperation and International Relations (DGRC[1]) on selected processes regarding student academic exchange programs between universities. We identified a BP family with different variants regarding the type of exchange program. We modeled the BP family with BPMN 2.0, using conditional branching as the language does not provide specific elements for modeling variability. Figure 1 depicts the model.

The DGRC announces positions for each exchange program in which the university participates (e.g., Erasmus or Santander), and students apply for them. Candidates must present several documents and certificates required by each program (`Receive application`) and the DGRC controls it (`Control documentation`). Then, the DGRC and other authorities analyze the applications. Although the open positions are initially defined when the call is issued, depending on the budget and other context elements for each call, they can be enlarged or reduced (`Analyze budget availability`). Depending on the exchange program, an evaluation committee evaluates candidates (`Evaluate candidates`). It prioritizes them in an ordered list (`Prioritize candidates`) to assign the available positions for each exchange within universities. Some programs need several board meetings to agree on these available positions (`Negotiate open positions`). With the final available positions and the list of ordered candidates, the DGRC assigns the positions for each university (`Assign positions`), notifies the corresponding candidates (`Notify assignments`), and processes the related documents to start the mobility (`Process additional documentation`).

[1] DGRC. https://cooperacion.udelar.edu.uy/.

For improving the understanding of business people, we provided a colored path within the complete BPMN model that corresponds to each process variant. Figure 2 depicts the Erasmus Mundus exchange program. Process execution ignores not colored activities. In this variant, an external organization defines both the budget and the positions. Thus, conditions "need budget analysis?" and "need negotiation?" are set on "NO" since there is no space for budget analysis and negotiating open positions. The other gateway conditions are set on YES, except for condition "there was assignment?" which depends on runtime information.

The scenario is a process family since it represents a collection of processes that pursue the same business objective (i.e., assigning positions) but have differences depending on their application context. It is worth noting that the type of exchange program is the application context, a design-time decision that needs to be defined to configure the process instance that is going to execute. The project considered seven different student academic exchange programs that can be extended to fifteen programs, including postgraduate, teachers, and research projects programs. The variants differ on the activities that need to be done and the participants, e.g., no foreign counterpart if there is no need to negotiate open positions. Also, there are dependencies between activities that depend on design decisions, e.g., there is no need to notify assignments if there are no assignments in the first place (some programs just prioritize candidates). Although BP families can be represented and enacted as a simple BP, as we did in this project, the context of a single BP is very different since participants are fixed, and activities are performed or not depending on runtime information.

Although using explicit conditional branching is the straightforward option, it has drawbacks compared to a BP family modeling language. For example, there are two kinds of XOR gateways: those that depend on the exchange program type and those that rely on runtime information, creating confusion about whether XOR gateways are part of a design or execution-time decision. Moreover, its complexity increases with more extensive processes, variants, and non-trivial behaviors, and it affects the comprehension of a variant since it avoids its explicit representation before execution.

3 Review and evaluation of BPMN-based BP families

Although there are literature reviews [42,60,68], and evaluation frameworks [5,60] about BP families' approaches, they do not focus on BPMN-based approaches.

The goal of our literature review is to consolidate existing knowledge in the context of BPMN, since BPMN is the de-facto standard for modeling processes and is widely sup-

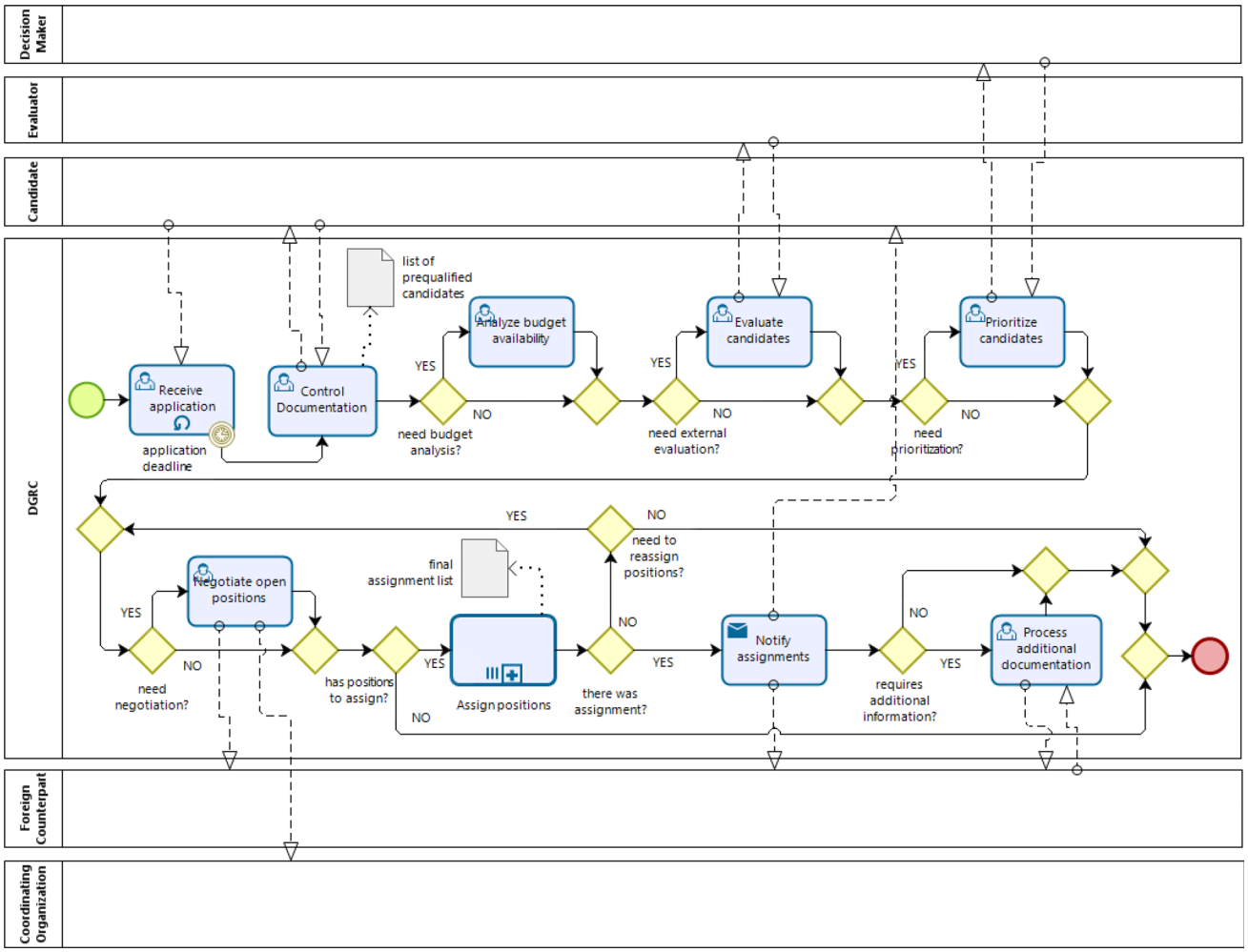


Fig. 1 Assignment of positions in academic exchange program BP family from [19]

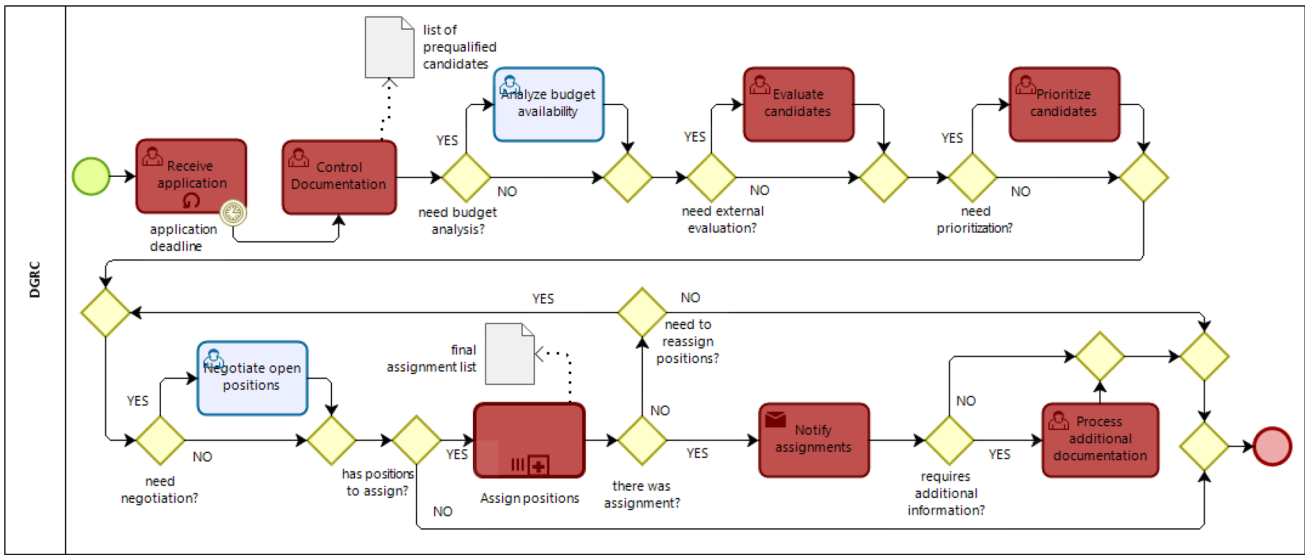


Fig. 2 Erasmus exchange program process variant from [19]

ported by existing BPMS. In our literature review we focused on answering the following question:

Which BPMN-based approaches exist to deal with business process families?

In what follows, we present a literature review that aims to answer such a question. We also evaluate approaches using a refined version of the VIVACE framework to deepen existing knowledge about them.

3.1 Review process

We conducted a literature review loosely following the systematic review method in [35]. The review process has four steps. First, we considered the existence of previous literature reviews. Thus, our work consisted of deepening the search for complementary BPMN-based approaches. The works considered on each step are listed in a supplemental spreadsheet available at [15].

Step 1. Analyze Existing Reviews. We first analyzed those works referenced in previous reviews [42,60,68] that are focused on BPMN. We also included some language-independent approaches, such as CVL /BVR, PROVOP, and Pesoa. Although they are not BPMN-based, they are known to be quite powerful approaches for variability adapted to BPMN and used within our general approach. Some works adapted and improved these approaches from the BPMN perspective. Thus, they are interesting to consider for a better comparison of approaches. After this step we got **12 papers** presenting **9 different approaches**: ABIS [70], ADOM [58], BPFM_N [45], BPFM_P [51], DECO [62], PESOA [25,37,64], PPM [52], Provop [40,63], and vBPMN [21].

Step 2. Perform Digital Library Search. We identified candidate primary studies by searching within electronic databases, using the following search string:

BPMN AND ("business process family" OR "business process line" OR "configurable process model")

We built a query within titles, abstracts, and keywords for the following electronic databases: ACM Digital Library, IEEE Digital Library, ScienceDirect, SCOPUS, Springer, and Web of Science (WOS). We conducted the searches in early 2018 and reran them in May 2021. The searches returned 276 papers: ACM (41 papers), IEEE (21 papers), Science Direct (31 papers), SCOPUS (60 papers), Springer (114 papers), and WOS (9 papers). After eliminating duplicates, we got **211 candidate papers**.

Step 3. Apply Incl. and Exclusion Criteria. We analyzed candidate primary studies for inclusion or exclusion based on predefined criteria. As inclusion criteria, we considered:

- IC-1 The paper addresses an approach for BP families support, from modeling to execution and evaluation rather than BP family terms merely mentioned in a generalized manner.
- IC-2 The paper addresses BPMN as the base modeling language and not only mentions BPMN as an example language.
- IC-3 Both aspects: BP families and BPMN are considered together to present an approach for BP families support involving a modeling language with their corresponding variability mechanism.

As exclusion criteria, we considered:

- EC-1 The paper is not electronically available on the web.
- EC-2 The paper is not presented entirely in the English language.
- EC-3 The data register identified after applying the search string does not refer to a scientific paper but some non-peer-reviewed publication, such as technical reports, books, book chapters; proceedings' prefaces; and journal's editorials.

We filtered the papers by reading their title, abstract, and keywords and also considered the introduction and conclusion sections if it was not completely clear if the paper should be included or excluded based on title, abstract, and keywords only. We selected **21 primary studies** out of the 211 candidate works, written between 2007 and 2020.

Step 4. Reinforce Results. We unified the lists of papers found in Step 1 and Step 3. We reinforced the results by searching for authors' and conferences' web pages and the DBLP database. It allowed finding minor conference material and followed some work's evolution. We also select works that refer to the same approach from the same authors and consider other author's works that provide a complementary perspective of the former approaches. We finally identified **27 papers** presenting **16 different approaches**, which are summarized in Appendix A: ABIS [70], ADOM [58], BPFM_C [7], BPFM_N [45], BPFM_P [51], BPMN* [67], BPMNext [18], BPMNt [55], C-BPMN [3,65,78], ConfBPMF [46], CVL/BVR [4,9], DECO [62], PESOA [24,25,37,38,64,76], PPM [14,52,56], Provop [40,63], and vBPMN [21].

We evaluated each approach using the selected papers identified for each approach by applying a refined version of the VIVACE framework, which is described in the next section. A summary of the evaluation is shown in Tables 2, 3, and 4.

Table 1 Refined VIVACE framework (adapted from [5]). Refined features of VIVACE are typeset in italics

VIVACE	
(a) Modeling language used to represent process variability: *extension of BPMN, BPMN-inspired, or language-independent*	
(b) Technique used for building the process family	
(c) Method for modeling the process family: *Node Configuration, Element Annotation, Activity Specialization, or Fragment Customization*	
(d) 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 configurable process model
	F5.2 Propagating changes of a configurable process model to already configured process variants
(e) Tool implementation: *support not available (N/A); support for modeling (M), configuring (C), deriving variants (D)*	
(f) Empirical evaluation	
(g) Application domain	

3.2 Refinement of the VIVACE framework

The VIVACE framework [5] is devoted to the systematic assessment and comparison of process variability approaches. It defines three categories with several characteristics for evaluation, depicted in Table 1.

The first category is composed of general features of interest: (a) modeling language, (b) the technique and (c) method (variability mechanism) for expressing the BP family, (d) the process perspectives covered, (e) its tool support, (f) the existent empirical evaluation, and (g) the application domain used. In the case of techniques (b), there are two options: capturing the entire BP family in a single model (single artifact) or a set of related models (multi-artifact). About process perspectives (d), a business process can vary from any of them: *functional (F)* involves the activities that are performed; *behavioral (B)* represents the control flow between activities; *organizational (O)* represents the actors or roles performing the activities; *informational (I)* represents the data objects; *temporal (T)* covers temporal constraints restricting process execution; and *operational (Op)* refers to the implementation of atomic process activities, i.e., web services. In the case of (a), (c), and (e), we refine the VIVACE proposal providing concrete evaluation categories that were not present in the former. We integrated these refinements into Table 1 and showed them in italic. This refinement considers that we are dealing with BPMN and the more structured alternatives defined in [60]. These refinements are:

(a) Modeling language. We can define a language for BP families as an *extension* of BPMN, i.e., extending its metamodel or defining a profile; by the definition of a *BPMN-inspired* language without real connection with the former definition; and by the application of a *language-independent* approach without modifying BPMN.

(c) Method (variability mechanism). The method express the relations between a BP family and its process variants [60]: *Node Configuration* in which node elements, e.g., activities and gateways, can be retained, removed, or changed for any of their multiple possible variants; *Element Annotation* in which any element is linked, via an annotation, to a predicate over a domain model; *Activity Specialization* in which activities in the base model, and not of other types of elements, can be replaced by one of their multiple specialized versions; *Fragment Customization* in which process fragments, i.e., specific parts instead of an entire model, can be added to the base model, and fragments of it can be deleted or modified.

(e) Tool implementation. Although approaches need support on many aspects through the BP family life cycle [5], three basic requirements exist: *modeling (M)* of the BP family, *configuring (C)* an application context of a process variant, and *deriving (D)* such variant from the configuration, automatically or based on suggestions. It is also important to express if tool support is currently available (N/A if not available).

The second category considers the variability-specific language constructs provided: (LC1) configurable region, i.e., a variation point; (LC2) configuration alternative, i.e., a variant; (LC3) configuration context condition, i.e., defines the conditions for selecting a variant; (LC4) configuration constraint, i.e., a restriction regarding the selection of a variant; and (LC5) configurable region resolution time, i.e., a region providing flexibility by adaptation during process execution.

The third category considers features for variability support throughout the process life cycle (F1 to F5), from Analysis and Design to Evolution of a BP family.

For both language constructs and support features, VIVACE defines a scale for the support provided: no support [−], partial support [+/−], and full support [+].

Table 2 VIVACE-based evaluation—Group 1: Node configuration

VIVACE		ADOM	BPMN*	BPMNt	C-BPMN	DECO
Language		extension	extension	extension	inspired	extension
Technique		single-artifact	multi-artifact	multi-artifact	single-artifact	single-artifact
Mechanism		Node Conf.	Node Conf.	Node Conf.	Node Conf.	Node Conf.
Perspectives		F, B	F	F, B, O, I, T, Op	F, B	F, I, O
Variability-specific language constructs	LC1	+	+	+	+	+
	LC2	-	+	+	+	+
	LC3	-	+	-	-	-
	LC4	+	-	-	-	+
	LC5	-	-	-	-	-
Variability support features	**Analysis & Design phase**					
	F1.1	-	+	+	-	-
	F1.2	-	-	+	-	-
	F1.3	-	-	-	-	-
	F1.4	-	-	-	-	-
	F1.5	-	-	-	-	-
	Configuration phase					
	F2	-	-	+	-	-
	Enactment phase					
	F3.1	-	-	-	-	-
	F3.2	-	-	-	-	-
	Diagnosis					
	F4	-	-	-	-	-
	Evolution					
	F5.1	-	-	-	-	-
	F5.2	-	-	-	-	-
Tool		-	+ (M)	+ (N/A)	-	-
Evaluation		-	+	+	-	-
Domain		education	car rental	software development	-	banking

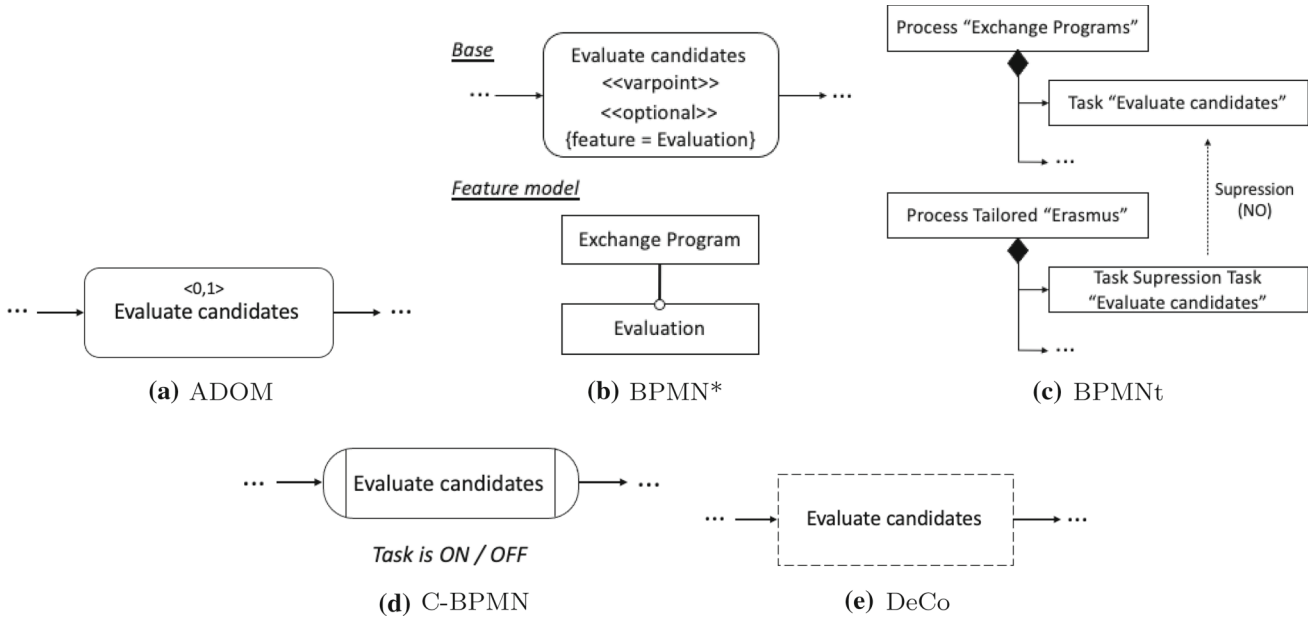


Fig. 3 Representation of the variable "Evaluate candidates" task using Node Configuration approaches

3.3 Evaluation of BPMN approaches with VIVACE

We evaluate all the approaches we found considering the whole VIVACE framework and also their refinement. In what follows, for more clarity, we demonstrate how each approach represents the very simple "Evaluate candidates" variation point introduced as part of the illustrative example in Sect. 2. The corresponding works contain more samples using the approaches. We group the approaches by their variability mechanisms, as done in [60], which expresses the relations between a BP family and its process variants. The groups contain almost the same number of approaches. The only variability mechanism not found was *Element Annotation*.

3.3.1 Group 1: Node configuration

Node elements can be configured to be retained, removed, or changed for any of their multiple possible variants on these approaches. A summary of the evaluation for this group is shown in Table 2. The representation of the variable "Evaluate candidates" task using these approaches is depicted in Fig. 3.

3.3.1.1 *Applying VIVACE to ADOM*

General description. The Application-Based Domain Modeling (ADOM) approach [58] proposes the creation of reference models for supporting the construction of other models. It aims at providing a way of expressing a BP family and a formal validation mechanism of the process variants against the corresponding reference models, rather than on the configuration of a BP family. The approach is general enough to be applied to different modeling languages, but it has a concrete BPMN extension.

It introduces a cardinality attribute to model elements (activities, events, gateways, and sequence flows), specifying how many times a given element can be instantiated in the process variant (a node configuration mechanism). Therefore, the mechanism partially covers F and B perspectives. It uses a single-artifact technique since there is no definition of variants. Configuration is done by describing a process variant satisfying the cardinality constraints on each configurable element. New elements can be added since they do not contradict these constraints. The approach describes a validation technique that checks a posteriori that a process variant complies with its reference model. There is no supporting tool, and a process from the education domain illustrated the approach.

Variability-specific language constructs. The extension defines multiplicity indicators attached to reference model elements defining a variation point (LC1 [+]). These indicators denote the lowest and upper-most boundaries of times variants of these elements may appear in a process variant, i.e., a constraint (LC4 [+]). Variants are not explicitly represented in the reference model (LC2 [−]), and the approach neither supports context conditions (LC3 [−]), not configurable region resolution time (LC5 [−]). The extension also defines reference model classifiers associated with elements in a process variant to specify that such element corresponds to a selected variant of an element in the reference model. It allows applying the validation technique between the process variant and its corresponding reference model.

Figure 3a shows an example of how "Evaluate candidates" can vary, just by specifying its multiplicity.

Variability support features. Although the approach has a graphical representation (F1.1) with formal semantics and defines how to verify a process instance (F1.2 and F1.3), it has no supporting tool.

3.3.1.2 *Applying VIVACE to BPMN**

General description. BPMN* [67] is a BPMN extension. It is an approach closely associated with PESOA's BPMN application (see Sect. 3.3.2.6). It uses a multi-artifact technique composed of a BPMN* model and a feature model. The approach provides a node configuration mechanism that can express variation points to any BPMN process element. However, a few additional information and the examples only show simple tasks as variation points (covering F perspective). Tasks within the BPMN* model can be stereotypes as variation points and related to their variants and tagged with a corresponding feature in the feature model. An Eclipse-based editor called BPL-Framework supports the approach providing modeling capabilities. A process from the car rental service domain illustrated the approach and was used for an empirical evaluation.

Variability-specific language constructs. The base model adds stereotypes to tasks for expressing variation points (LC1 [+]) that can be optional or mandatory. Variants are stereotyped model elements (LC2 [+]) connected through arrows (variability associations) with the variation points. Stereotypes added to these arrows identify the variation point's behavior (exclusive or inclusive choices). A feature model defines context conditions (LC3 [+]) relating stereotyped elements and their variants. The approach does not support other constructs.

Figure 3b shows an example of the "Evaluate candidates" variation point marked as an optional task and associated with the feature "Evaluation." Since it is optional within the feature model, there is no need to express other options.

Variability support features. The BPL-Framework only supports the BP family's graphical modeling (F1.1 [+]).

3.3.1.3 *Applying VIVACE to BPMNt*

General description. BPMNt [55] is an extension of BPMN inspired by SPEM 2.0. It uses a multi-artifact technique composed of a base BPMN process model and a BPMNt tailored

process model providing a node configuration mechanism for any process element, partially covering every process perspective (F, B, O, I, T, Op). It uses the standard extension mechanism of BPMN for introducing new elements expressing deletion, replacement, and addition of process elements within a base process. The standard XML representation of BPMN expresses the BPMN tailored process model; it does not provide any graphical modeling extension for BPMN. An Eclipse-based tool that no longer exists supported the approach, providing modeling and configuration capabilities. A process from the software domain was used for assessing and illustrating the approach.

Variability-specific language constructs. A BPMN tailored process model contains an incomplete BPMN process composed of BPMNt extension elements. The extension elements represent variants (LC2 [+]) and define the type of operation carried out (deletion, replacement, and addition). The model also contains tailoring relationships that express the relation between the BPMNt extension elements and the base process elements considered variation points (LC1 [+]). There is no representation of a context since tailoring relationships represents a concrete configuration; i.e., just one variant applies for each variation point.

Figure 3c shows an example of how "Evaluate candidates" can be deleted. A tree-based representation of the base process contains the configurable task. A tailored process model defines that the task must be deleted (Task Suppression operation), and a tailoring relationship connects both tasks. There is no need to express the case of keeping the task since every element is kept if there is no tailoring operation.

Variability support features. The tool supported the definition of a BPMNt tailored process model represented as an element tree using the EMF editor (F1.1 [+]). It applies Java rules to interpret tailoring relationships and generate a variant (F2 [+]). The tool checks consistency rules on the resulting BPMN variant during this process, e.g., the connection of flows between elements when an intermediate element is deleted (F1.2 [+]).

3.3.1.4 *Applying VIVACE to C-BPMN*

General description. Configurable-BPMN (C-BPMN, [77]) is a BPMN-inspired approach based on the Configurable Event-driven Process Chain (C-EPC, [61]) approach. In [5] there is an evaluation of C-EPC using VIVACE, but C-BPMN focuses on the control-flow perspective and does not cover every aspect of the C-EPC proposal. It follows a single-artifact technique composed of a BPMN model consisting of core elements, i.e., activities, events, and gateways. Tasks and gateways can be marked as configurable, partially covering F and B perspectives using a node configuration mechanism. The authors neither extend the BPMN metamodel nor provide tool support or perform an empirical evaluation of the approach. In contrast, they give a formal definition of the syntax and semantics of the language, define a proper algorithm for deriving a concrete variant, and analyze syntax and execution semantics.

Related work is the one in [3]. The authors present an algorithm for extracting, clustering and merging process fragments around a particular activity to construct a configurable fragment based on C-BPMN. In [65], the authors propose a different C-EPC-inspired approach, also named C-BPMN. However, this approach is subsumed by the C-BPMN approach we have presented.

Variability-specific language constructs. A base C-BPMN model is a core BPMN model and configurable elements such as activities, events, and gateways (LC1 [+]). Some configurable attributes allow defining possible variants (LC2 [+]) but with very limited possibilities. It is possible to set configurable activities as ON, OFF, and OPT for inclusion, exclusion, or inclusion as an option from the derived process variant. A configurable gateway has a generic behavior, and during configuration, it is possible to select its type (AND, OR, XOR) or even select a concrete branch (deleting the others).

Figure 3d shows an example of how "Evaluate candidates" is represented as a configurable task. The task can be set to ON and OFF to keep or delete it.

Variability support features. Although there is a graphical representation and an algorithm for performing a variant configuration, there is no tool supporting it.

3.3.1.5 *Applying VIVACE to DECO*

General description. Declarative Configurable specifications (DeCo) [62] is a BPMN-extension providing a mechanism for descriptive process modeling, formal analysis, and stepwise refinement from a configurable design to concrete process execution. It uses a single-artifact technique in which activities, roles, and data objects can be configured to be optional or having alternatives, covering F, I, and O perspectives. These elements can be substituted by the same kind of elements, i.e., a node configuration mechanism. The proposal only presents the language and a simple application example from the banking domain. The proposal was discontinued. Thus, there is neither information on how BPMN needs to be extended nor on how the configuration process is done. There is no supporting tool.

Variability-specific language constructs.

The extension defines new elements to the BPMN graphical notation, e.g., optional tasks/data/conditions, which specifies variation points depicted by dashed lines within the base process, and configurable tasks/data/actors represented by bold lines (LC1 [+]). Variants (LC2 [+]) are connected to the variable parts in the same model. The notation also rep-

resents configuration rules (LC4 [+]), but it is unclear how they are modeled.

Figure 3e shows an example of how "Evaluate candidates" can vary, just by specifying that it is optional within the sequence flow of activities (dotted square).

Variability support features. Although there is a graphical representation, there is no supporting tool.

3.3.2 Group 2: Activity specialization

On these approaches, activities in the base model can be replaced by one of their multiple specialized versions. A summary of the evaluation for this group is shown in Table 3. The representation of the variable "Evaluate candidates" task using these approaches is depicted in Fig. 4.

3.3.2.1 *Applying VIVACE to ABIS*

General description. The Adaptive Business process modeling in the Internet of Services (ABIS) [70] is a BPMN 2.0 extension focused on process orchestrations, i.e., multiple interrelated processes. It extends BPMN by introducing variable regions that are like configurable subprocesses. It uses a single-artifact technique composed of a process template, i.e., a BPMN model containing variable regions (variation points), and complimentary process fragments, i.e., BPMN models defining the variants. It supports multi-level variable regions (F), thus being an activity specialization mechanism. However, variable regions can be substituted by complete process fragments. Therefore, the mechanism partially covers the behavioral (B) and information (I) perspectives, close to fragment customization. The proposal only presents the language and a simple application example about an insurance company. There is neither a definition of how specifically BPMN is extended nor how the configuration must be done. There is no supporting tool.

Variability-specific language constructs. The approach defines variable regions within a process template. These regions are activity-like elements with a special symbol representing a variation point (LC1 [+]). They have exactly one incoming and one outgoing sequence flow. A variable region is a placeholder for one or more process fragments that define variants (LC2 [+]). Within a process fragment, it can be other variable regions and also variable links, i.e., an element similar to a throwing link event that allows connecting the source and target of a message or sequence flows. The language does not support context conditions (LC3 [−]), constraints (LC4 [+]) or configurable region resolution time (LC5 [−]).

Figure 4a shows an example of how "Evaluate candidates" can vary. It involves the definition of the variable region and a concrete task as a variant. It is supposed that the region can also be deleted during configuration, so there is no need to express the removal as a variant.

Variability support features. The approach has a graphical representation, but there is no tool supporting it.

3.3.2.2 *Applying VIVACE to BPFM_C*

General description. The Business Process Family Model (BPFM) [12] can be considered a BPMN-inspired approach with the potential of being language independent since it is an extension of feature models whose features are BPMN elements. We refer to it as BPFM_C (C for Cognini, its first author) to avoid conflicts with others.

It uses a single-artifact technique defining a tree-based feature model such that the root identifies the BP family, and each level represents a different level of detail in the BP specification. In particular, internal activities denote subprocesses, and leaves represent atomic activities (possibly with associated data objects), i.e., a node configuration mechanism which considers the F, B, I, and Op perspectives. The configuration consists of selecting the single activities (leaves in the tree) to include in the process variant. The configuration can be verified according to well-formed rules before deriving a process variant. The derivation mechanism does not derive a whole process but a set of fragments. This declarative specification can be later enriched with control flow information to express a prescriptive process variant. In this sense, the approach has a declarative perspective. The approach is supported by a tool that allows modeling the BP family, defining a concrete configuration, and automatically producing a process variant (M, C, D). A process from the public administration illustrated the approach and was used for an empirical evaluation.

Variability-specific language constructs. The authors define a metamodel for representing a feature model with activities and constraints. The former connects with the BPMN specification (it has the same meaning and symbolic representation) and represents variation points (LC1 [+]) and variants (LC2 [+]). Constraints connect activities at different tree levels and are traditional feature model constraints, e.g., optional, mandatory, inclusive choice, exclusive choice, etc. (LC4 [+]). Constraints also specify a partial execution order of the activities. The language neither supports the definition of context conditions (LC3 [−]) nor configurable region resolution time (LC5 [−]).

Figure 4b shows an example of how "Evaluate candidates" can vary. The feature model must define an optional activity (blank circle) that can be deleted during configuration. The feature model must be complete in terms of the variable and mandatory parts and does not provide a way of expressing the sequence flows from and to the different branches of the tree (process fragments that will be derived).

Variability support features. The tool allows graphically defining a BP family (F1.1[+]), configuring a process vari-

Table 3 VIVACE-based evaluation—Group 2: activity specialization

VIVACE		ABIS	BPFM_C	BPFM_P	BPMNext	ConfBPMF	PESOA
Language		extension	inspired	extension	extension	inspired	lang-indep
Technique		single-artifact	single-artifact	single-artifact	single-artifact	multi-artifact	multi-artifact
Mechanism		Act. Special.	Act. Special.	Act. Special.	Act. Special.	Act. Special.	Act. Special.
Perspectives		F, B	F, B, I, Op	F, B, I	F, B, O, I	F, B, Op	F, B
Variability-specific language constructs	LC1	+	+	+	+	+	+
	LC2	+	+	+	+	+	+
	LC3	-	-	-	-	+	+
	LC4	-	+	+	+	-	-
	LC5	-	-	-	-	-	-
Variability support features	**Analysis & Design phase**						
	F1.1	-	+	+	+	+	+
	F1.2	-	+	-	-	-	+
	F1.3	-	-	-	-	-	+
	F1.4	-	-	-	-	-	-
	F1.5	-	-	-	-	-	-
	Configuration phase						
	F2	-	+	+	+	+	+
	Enactment phase						
	F3.1	-	-	-	-	-	-
	F3.2	-	-	-	-	-	-
	Diagnosis						
	F4	-	-	-	-	-	-
	Evolution						
	F5.1	-	-	-	-	-	-
	F5.2	-	-	-	-	-	-
Tool		-	+ (M, C, D)	+ (N/A)	+ (M, C, D)	+ (N/A)	+ (N/A)
Evaluation		-	+	-	-	-	-
Domain		insurance -	government	e-commerce	education	e-commerce	retail, commerce

ant, verifying a configuration (F1.2 [+]), and automatically derive a process variant (F2 [+]).

3.3.2.3 *Applying VIVACE to BPFM_P*

General description. The Business Process Family Model (BPFM) [51] is a BPMN extension inspired by the vSPEM approach for software processes [16]. We refer to it as BPFM_P (P for Park, its first author) to avoid conflicts with others. It extends BPMN by introducing configurable elements for simple tasks, i.e., an activity specialization mechanism. It uses a single-artifact technique in which both variants and variation points are represented. The mechanism considers the functional perspective (F), and since process fragments can substitute single tasks, the mechanism partially covers the behavioral (B) and information (I) perspectives. The approach has a supporting tool that is not available. E-commerce processes were used for illustration.

Variability-specific language constructs. BPMN tasks are extended to express variability information, depicted with special symbols within the base process, and have the same properties as any other regular task. A task can be marked as optional, and with different multiplicity combinations (LC1 [+]), e.g., one or more variants can be selected. Variants are represented in the same model through a variant binding that connects variation points with their variants (LC2 [+]). The extension also defines a variant region that contains a process fragment. There is no definition of context condition (LC3 [−]), but the multiplicities impose some constraints (LC4 [+]) for the configuration. The language does not support configurable region resolution time (LC5 [−]).

Figure 4c shows an example of how "Evaluate candidates" can vary, just identifying the activity as optional (using the OP symbol in the upper-right corner).

Variability support features. The tool allows graphically defining a BP family (F1.1[+]) and does not support any other Analysis and Design feature (F1.2 [−] to F1.5 [−]). The configuration allows selecting the available variants for variation point (F2 [+]), but there is no automatic derivation of process variants.

3.3.2.4 *Applying VIVACE to BPMNext*

General description. BPMNext [18] is a BPMN 2.0 extension of ours, inspired by the vSPEM approach for software processes [16]. It extends BPMN by introducing configurable elements for activities (tasks and subprocesses). It uses a single-artifact technique composed of a base process defining the variation points and complimentary BPMN models describing the variants. It supports multi-level variation

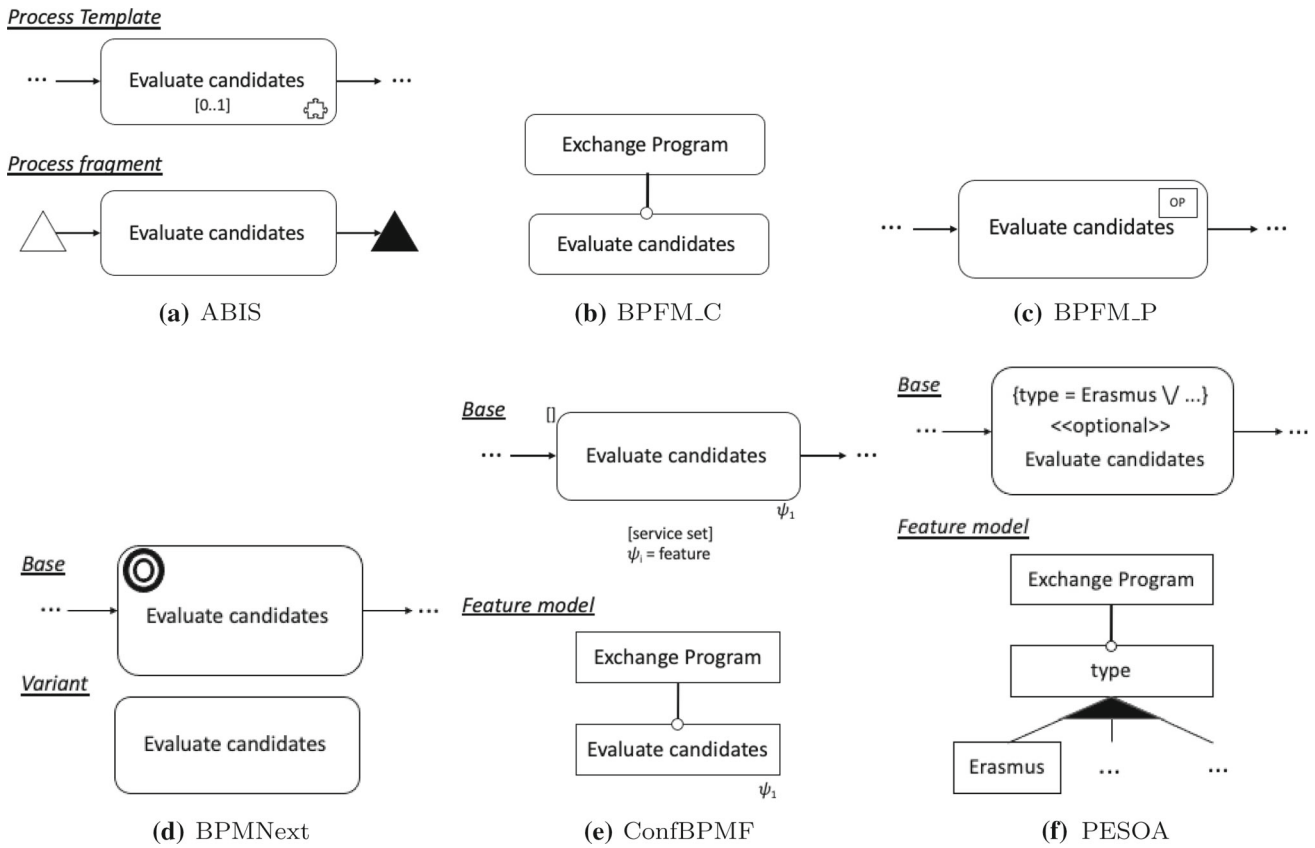


Fig. 4 Representation of the variable "Evaluate candidates" task using Activity Specialization approaches

points for activities (F), considering the roles that perform them (O), thus being an activity specialization mechanism. However, activities can be substituted by any other kind of activity, e.g., a subprocess. Therefore, the mechanism partially covers the behavioral (B) and information (I) perspectives, close to fragment customization. Activities with a single entry and a single exit flow can also be deleted. We provide an ATL model transformation for the derivation of a process variant. We also provided BPMN2 Modeler-based tool support, which allows modeling the BP family, defining a concrete configuration, and automatically producing a process variant (M, C, D).

Variability-specific language constructs. The extension defines a VPActivity (VPTask or VPSubProcess), which specifies a variation point (LC1 [+]), depicted with special symbols within the base process, and has the same properties as any other regular activity. These activities can be substituted by any other activity or deleted, represented by a VActivity (LC2 [+]). Variant activities can define new roles in charge of them. The language does not describe any context condition (LC3 [−]), and the only constraint (LC4 [+]) that imposes is that a variant can be selected only once for a process. The "delete" variant can be chosen anytime. Finally, the language does not support configurable region resolution time (LC5 [−]).

Figure 4d shows an example of how "Evaluate candidates" can vary. It involves the definition of the VPActivity, and a concrete task as a variant. The task can also be deleted during configuration, so there is no need to express the removal as a variant.

Variability support features. The tool allows graphically defining a BP family (F1.1[+]) and does not support any other Analysis and Design feature (F1.2 [−] to F1.5 [−]). The configuration uses a configuration metamodel, allowing selecting the available variants for variation point (F2 [+]). An ATL model transformation performs the derivation of a process variant. It inserts, substitutes, and suppresses activities and re-connects flows between them, and checks well-formed constraints.

3.3.2.5 *Applying VIVACE to ConfBPMF*

General description. Business Process Modeling Framework (ConfBPMF, [46]) is a BPMN-inspired approach focused on deriving process instances based on stakeholders' non-functional requirements. It uses a multi-artifact technique composed of a feature model to describe the variability and a Business Process Model Template (BPMT) describing the

base process. It defines an activity specialization variability mechanism allowing selecting and skipping activities and the services executing them, according to their selection in the feature model (partially covering F, B, and Op perspectives). At first, it looks similar to the conditional branching mechanism described in Sect. 2. However, the authors neither give details about the language definition nor the concrete configuration mechanism. In contrast, they focus on analyzing how different techniques, e.g., genetic algorithms, can derail the process variant that best fits considering QoS characteristics. An Eclipse-based tool supported the approach but is not anymore available. Although there are no details, the authors express that the tool provided modeling and configuration capabilities. A process from the e-commerce domain illustrated the approach. Finally, there is an evaluation not of the approach but the algorithms used for configuration.

Variability-specific language constructs. A BPMT describes a BPMN-based business process and a set of available services performing the activities in the BPMT, including their QoS characteristics. Services implement activities, possibly with the same functionality but with different QoS characteristics. Activities have presence conditions that are annotations for each activity expressing the connection of such activities with the elements in a feature model, i.e., variation points (LC1 [+]). A BPMT contains every possible activity in the BP family. Thus it defines both variation points and their variants (LC2 [+]). With a quality model defined by stakeholders, the feature model defines the context conditions for the configuration (LC3 [+]).

Figure 4e shows an example of the variable "Evaluate candidates" task. The task defines a set of services that can perform the task (non in this case) and a presence condition Ψ_1 that relates the task with the optional feature within the feature model.

Variability support features. The tool supported the definition of a BPMT and its corresponding feature model (F1.1 [+]). It also allows expressing QoS of those services enacting activities within the BPMT. A genetic algorithm, replaced by other optimization techniques in later works, derives an optimized process variant by selecting the services with the most suitable quality attributes for each activity (F2 [+]), i.e., selecting and skipping activities.

3.3.2.6 *Applying VIVACE to PESOA*

Process Family Engineering in Service-Oriented Applications (PESOA [64]) provides a language-independent approach. In [5], the authors evaluated PESOA using VIVACE but not considering its adaptation to BPMN. In what follows, we resume and complement such evaluation for the sake of completeness without delving into details of the former assessment.

General description and variability-specific language constructs. A so-called variant-rich process model contains every possible variant of the BP family. Annotations attached to activities define variants, i.e., an activity specialization mechanism. It partially covers F and B perspectives, and neither considers a different kind of activities nor roles or data objects. Activities can be optional or replaced by another activity. It uses a multi-artifact approach with a feature model determining the context conditions to which each variant applies. Each condition is local to a given variation point. Although it is language-independent, the whole approach was adapted to BPMN. Some authors name this adaptation vrBPMN.

Figure 4f shows an example of the variable "Evaluate candidates" task. The task is optional, and the annotation `type` connects the task with a feature model that determines the different options, e.g., Erasmus.

Other authors also improve PESOA in the light of its application to BPMN. In [37], the authors extend the BPMN metamodel for defining templates capable of expressing process variants and a set of adaptation operators. In [76], the authors identify some issues with the variability mechanism of PESOA and propose a pattern-based mechanism to tackle the problems. In [25], the authors proposed a validation approach for process families. Also, in [24] the authors propose using an ontology for expressing both domain knowledge and variability knowledge to check syntactic (verification) and semantic (validation) quality of the configurable process model. These two works, i.e., [24,25] provided new capabilities to the former one (F1.2 [+] and F1.3 [+]). Finally, in [38] the authors present an extension to the PESOA notation, which together with a feature model tackles some modeling limitations, e.g., the representation of an inclusive OR between variants.

Variability support features. An Eclipse plug-in that no longer exists supported PESOA's general approach. The tool provided modeling and configuration capabilities. Processes from the retail and commerce domains illustrated the approach. However, neither the examples nor the tools of the related works are available.

3.3.3 Group 3: Fragment customization

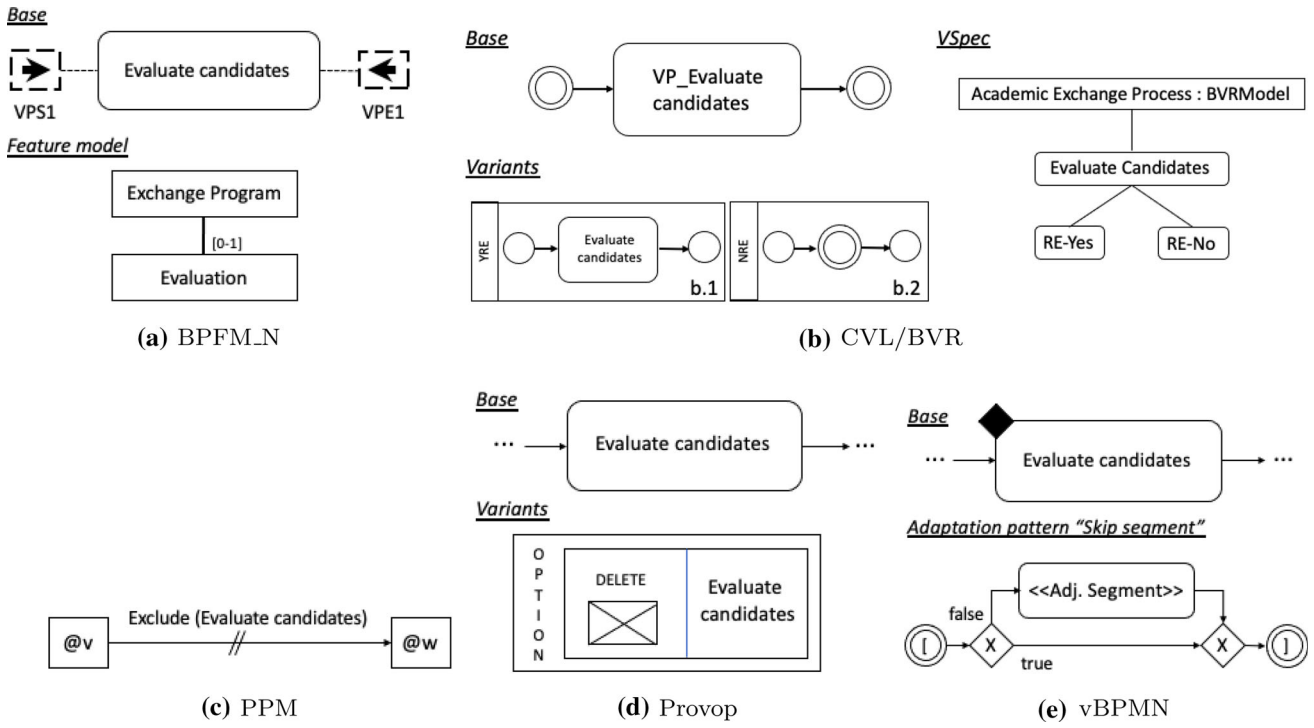
On these approaches, process fragments can be added, deleted, or modified. A summary of the evaluation for this group is shown in Table 4. The representation of the variable "Evaluate candidates" task using these approaches is depicted in Fig. 5.

3.3.3.1 *Applying VIVACE to BPFM_N*

The Business Process Family Model (BPFM) is an extension of UML Activity Diagrams for capturing variability. The work in [45] is considered in [60] to be subsumed by

Table 4 VIVACE-based evaluation—Group 3: fragment customization

VIVACE		BPFM_N	CVL/BVR	PPM	Provop	vBPMN
Language		extension	lang-indep	lang-indep	lang-indep	extension
Technique		multi-artifact	multi-artifact	multi-artifact	multi-artifact	multi-artifact
Mechanism		Frag. Custom.	Frag. Custom.	Frag. Custom.	Frag. Custom.	Frag. Custom.
Perspectives		F, B, I, Op	F, B, O, I, T, Op	F, B	F, B	F, B, T
Variability-specific language constructs	LC1	+	+	+	+	+
	LC2	+	+	+	+	+
	LC3	+	+	-	+	+
	LC4	+	+	+	+	+
	LC5	-	-	-	-	+
Variability support features	**Analysis & Design phase**					
	F1.1	+	+	+	+	+
	F1.2	-	-	+	+	-
	F1.3	-	-	-	-	-
	F1.4	-	-	-	-	-
	F1.5	-	+	-	-	-
	Configuration phase					
	F2	+	+	+	+	-
	Enactment phase					
	F3.1	-	-	-	-	+
	F3.2	-	-	-	+	+
	Diagnosis					
	F4	-	-	-	-	-
	Evolution					
	F5.1	-	-	+	-	-
	F5.2	-	-	+	-	-
Tool		+ (N/A)	+ (M,C,D)	+ (N/A)	+ (N/A)	+ (N/A)
Evaluation		-	+	-	-	-
Domain		e-commerce	education	commerce	automotive, healthcare	ship maintenance



(a) BPFM_N **(b)** CVL/BVR **(c)** PPM **(d)** Provop **(e)** vBPMN

Fig. 5 Representation of the variable "Evaluate candidates" task using Fragment Customization approaches

BPFM. However, since it presents a BPMN-based approach, we describe it here. We refer to it as BPFM_N (N for Nguyen, its main author) to avoid conflicts with others.

General description. BPFM_N [45] is a BPMN 2.0 extension focused on process-based service compositions. It uses a multi-artifact technique composed of a base BPFM_N model, i.e., a BPMN model containing both variation points and variants in control, data, message flows, and a feature model capturing variability concerning service compositions. In general terms, it provides a fragment customization mechanism since it allows replacing fragments between specific points with other fragments. However, examples are focused on simple service tasks. It is not clear that they provide variability mechanisms involving different kinds of tasks and other BPMN elements, e.g., gateways and events (F and B perspectives partially covered). Moreover, it allows changing the information items and the interactions between processes and services (I, Op). The proposal uses an MDE approach to automate process variant generation. An Eclipse-based tool that no longer exists supported the approach. The tool provided modeling and configuration capabilities. Process-based service compositions from the e-commerce domain illustrated the approach.

Variability-specific language constructs. A base BPFM_N model defines three ways of expressing variation points (LC1 [+]). First, it represents adjustment points which are special nodes within the control flow where a variation point starts and ends, i.e., delimiting the fragment. It is possible to define variation points within variation points. Second, it represents data object variation points with special data associations to tasks. Third, it defines conversation variation points and abstract partners defining variable data flows between partners. Variants are represented in the same model (LC2 [+]). Dashed lines connect adjustment points to associated variants, i.e., possibly many complex flows that start and end with the adjustment points. Also, dashed arrows connect data and message variation points with their variants. The feature model defines the context conditions for the configuration (LC3 [+]). A mapping model relates and constraints variation points and variants in the process model with features in the feature model (LC4 [+]).

Figure 5a shows an example of the variable "Evaluate candidates" task. The base model contains the task within two adjustment points (VPS1 and VPE1). Since the task is optional, the feature model presents the optional feature, "Evaluation," connected with the task using the mapping model.

Variability support features. The tool supports modeling BPFM_N process models and the feature model (F1.1 [+]). It also supports the derivation of executable process variants using ATL model transformations (F2 [+]). These variants are specified using BPEL, and the transformation also generates the service interface descriptions assuming that tasks are all service tasks.

3.3.3.2 *Applying VIVACE to CVL/BVR*

In [9] we evaluated CVL/BVR using VIVACE. In what follows, we provide an excerpt of such evaluation for the sake of completeness.

General description. The Common Variability Language (CVL, [48]) is a language-independent approach taking any MOF-based domain-specific language (DSL) and adding up constructions to manage variability concepts. The Base Variability Resolution (BVR, [27]) approach is built on CVL, simplified, and enhanced. It uses a multi-artifact technique. The host language, e.g., BPMN 2.0, describes the Base Model and the concrete variants. The base model contains the BP family's common elements (e.g., a base process) and the variation points (e.g., process fragments within the base process). It also expresses variants as independent models (e.g., process fragments). A Variability Model defines the correspondence between the base model elements with the variation points. A VSpec model is a feature model representing abstract variation points and logical decisions related to them (called VSpecs). It needs to be resolved for each BP family's configuration. A Realization Model contains the mapping between the abstract variation points within the VSpec and the base model elements. A Resolution Model provides a particular configuration of the VSpec model to produce a process variant.

In [4], the authors adapted the approach for BPMN. They briefly described how CVL and BPMN could be used together. Although the language allows varying any process element, this work only focuses on activities. They also focus on how to adapt process variants at runtime automatically. Its prototype system is not available. In [9], we took a step further by applying a fragment customization mechanism. We implement an Eclipse-based editor [54], which allows modeling the BP family, defining a concrete configuration, and automatically producing a process variant using the execution of a model transformation (M, C, D). CVL/BVR covers every process perspective (F, B, O, I, T, Op). However, they do not provide any graphical model for expressing the variability model.

Variability-specific language constructs. We use a realization primitive called Fragment Substitution. It involves the definition of a placement fragment in the base model (LC1 [+]), determining which elements (regions) in the base model can be substituted. It also involves defining one or more replacement fragments that can replace the placement fragments (LC2 [+]). The context for selecting a variant is defined by the correspondence between fragment substitutions and nodes of the VSpec tree (LC3 [+]). It is possible to express additional constraints related to VSpec elements (LC4 [+]).

Finally, the language does not support configurable region resolution time (LC5 [−]).

Figure 5b shows an example of how "Evaluate candidates" can vary. It involves defining a placement fragment in the base model as a `VP_EvluateCandidates` and the two possible replacement fragments corresponding to a concrete task or an intermediate event. A post-processing model transformation removes this auxiliary event. The fragments are related through the VSpec that determines the two options.

Variability support features. Our integrated tool graphically supports the definition of a BP family (F1.1 [+]). It also indirectly supports the reuse of models (F1.5 [+]) since both the Placements and the Replacements of a BVR model refer to the process models' objects defined in separate files. The tool also provides ATL model transformations for carrying the configuration process (F2 [+]).

3.3.3.3 *Applying VIVACE to PPM*

General description. Partial Process Models (PPM, [52]) is a language-independent approach based on BPMN-Q, a visual query language for business process models. Although it is a BPMN-based language, it can query core concepts of business processes. Thus, it is adaptable to any other business process language. The approach uses a multi-artifact technique composed of partial process models describing base processes and BPMN-Q queries. The approach uses a fragment customization mechanism. BPMN-Q can query a process fragment and extend or limit the behavior inherited from parent processes (F, B), potentially several simultaneously. It does not consider advanced BPMN elements such as different activities, roles, or data objects. The approach focuses on maintaining consistency of process model variants and provides benefits such as allowing multiple level inheritance and multiple inheritances between variants. An Eclipse-based tool that no longer exists supported the approach. The tool provided modeling and queries execution capabilities. Processes from the commerce domain illustrated the approach.

A related (subsumed) work is the one in [14] in which the authors define an extension of BPMN for defining tags within a process to express variant operations on each BPMN element (adding, deleting, and modifying an element). The authors also define an algorithm to extract a process variant from configurable templates. There is neither a specific notation for the BPMN extension nor an available implementation. Another related work is the one in [56] in which the authors axiomatize process refinements from BPMN models using ATL model transformations.

Variability-specific language constructs. A base process can be any BPMN process or partial process model, i.e., derived from the application of a BPMN-Q query. A BPMN-Q query considers any control-flow element, i.e., no pre-specified variation points (LC1 [+]). BPMN-Q queries to determine the possible adaptation variants (LC2 [+]). They describe behavioral inheritance relations from parent processes. These variants and concrete parts of the process need to be composed into a concrete process variant. The approach does not define context conditions (LC3 [−]). However, the approach supports configuration constraints via the inheritance chain of queries (LC4 [+]).

As an example of configuring the "Evaluate candidates" task, take the BPMN-Q query in Fig. 5c. It defines two variables V and W that are connected through a sequence flow that excludes the task. Executing this query over the whole base process in Fig. 1 gives pairs of connected tasks assuming the deletion of "Evaluate candidates." Thus, when composing the partial models, it provides the whole process but such a task. There is no need to express the case of keeping the task since every element is held if there is no query.

Variability support features. The tool supported the modeling of BPMN business processes and the definition of BPMN-Q queries (F1.1 [+]). It also allows verification of a model derived from a query against a parent process model (F1.2 [+]). The tool can run BPMN-Q queries and compose queries with the parent process's concrete parts (F2 [+]). Finally, the tool stores the parent process and the set of partial process models obtained from the execution of queries, together with the queries related to parent and child processes. In this way, the tool provides versioning of processes (F5.1 [+]) and an easy way of propagating changes by re-running BPMN-Q queries and composing their results (F5.2 [+]).

3.3.3.4 *Applying VIVACE to Provop*

PROcess Variants by OPtions (Provop, [26]) is another language-independent approach which was also evaluated in [5] using VIVACE, but without considering its adaptation to BPMN. As with PESOA, we resume and complement such assessment for the sake of completeness without delving into details.

General description and variability-specific language constructs. Provop uses a multi-artifact technique composed of a base model, change options, a context model, and a constraint model. The language allows the definition of configurable regions within a base process delimited by so-called adjustment points. A sequence of model changes determines variants, allowing insertion, deletion, and modification of fragments concerning the control-flow perspective. These variants are defined together with context rules within a context model, defining constraints between change options. It partially covers F and B perspectives, and neither considers a different kind of activities nor roles or data objects.

Figure 5d shows an example of the variable "Evaluate candidates" task. The base model defines the task, and a change option defines it as possible to delete it. The deletion of this task depends on the selection of such an option during configuration.

As stated in [60], Provop subsumes some works in the BPMN context but not graphically modifying the language. In [63], the authors propose an approach based on non-functional requirements for guiding the analysis and configuration of process families. In [40], the authors define a Haskell-based representation of BPMN and extend it for representing the variant part of a process using aspect-oriented constructs. They also propose a set of transformations (Haskell functions) for resolving variability in BPMN models.

Variability support features. A tool within the ARIS Business Architect supported Provop's general approach. The tool provided capabilities for modeling, verifying, and configuring a BP family. The authors define new elements within BPMN representing operation types and adjustment points. The tool used some existent BPMN elements changing their semantics, e.g., BPMN pools represent change option and lanes define parameters of a change operation (e.g., a fragment substituting another). This tool and the ones of the related works are not available. Processes from the automotive and healthcare domains illustrated the approach.

3.3.3.5 *Applying VIVACE to vBPMN*

Although vBPMN is considered in [60] to be subsumed by Provop since it builds upon the general underlying idea, we describe it here since it complements the approach with a concrete BPMN-based application.

General description. vBPMN [21] is an extension of the Provop adaptation to BPMN. It uses a multi-artifact technique composed of a base vBPMN model, i.e., a BPMN model with Provop's adjustment points, event-aware adaptation patterns defining the available variants, and R2ML rules connecting the context data with the adaptation patterns. The authors define a pattern catalog for more than thirty different adaptation purposes to change the functional, behavioral, and temporal perspectives (F, B, T). The evaluation of R2ML rules at design time derives concrete process variants. Moreover, the approach allows adaptation at runtime, but it depends on having the infrastructure to change the model according to rules evaluation dynamically.

A jBoss Drools-based tool that no longer exists supported the approach. The tool provided modeling and configuration at runtime capabilities. Processes from the ship maintenance domain illustrated the approach.

Variability-specific language constructs. The base model defines adjustment points for representing adaptative segments, i.e., variation points (LC1 [+]). A segment is a structured fragment of the model having only one entry and one exit point (fragment customization for the functional and behavioral perspectives). Enclosing intermediate throw event nodes with opening or closing square brackets mark those segments. Also, a black diamond marks a single adaptive task in its upper left corner. Event-aware adaptation patterns take place when entering a segment and define the available variants, ranging from simple behaviors (e.g., skipping of tasks) to interrelated behaviors between variants (LC2 [+]). The rules-language R2ML defines adaptation rules as context conditions (LC3 [+]). Adaptation patterns restrict the set of adaptations, which are partial configuration constraints (LC4 [+]), without a way of expressing restrictions between patterns. The use of R2ML determines configurable region resolution time (LC5 [+]).

Figure 5e shows an example of the variable "Evaluate candidates" task. The base model contains the task marked as an adaptative segment (the diamond on the task's upper side). There is also an adaptation pattern name "Skip Segment" that, when applied, wraps the segment (the task in this case) into an exclusive choice, which always evaluates to true on the bypassing path.

Variability support features. The tool supported the modeling of adaptive segments and the specification of adaptation rules (F1.1 [+]) and did not support any other Analysis and Design feature. Iy does not support the configuration of a process variant at design time (F2 [−]) but at runtime (F3.1 [+]). It uses process variables, and when process execution enters an adaptation segment, the rule engine determines whether an adaptation pattern applies, determining the path to follow. In this sense, it is a reconfiguration of the process instance (F3.2 [+]) in some way similar to the use of conditional branching, as exemplified in Sect. 2.

3.4 Summary of the evaluation

Tables 2, 3, and 4 provide a summary of the assessment of each of the features proposed in the framework for each of the approaches presented. Within the title, there is a reference for its first publication and its corresponding year. To the best of our knowledge, there is no literature review using VIVACE for evaluating the approaches, except for the case of CVL/BVR for BPMN (in [9]), and PESOA and Provop (in [5]).

VIVACE provides a practical approach for comparing variability approaches and their support for the phases of the BP life cycle. However, perspectives require a more in-depth evaluation. For example, the functional perspective ranges from deleting or substituting an activity with another to modifying its type or replacing it with a complex fragment. In the context of VIVACE, BPMN* and CVL/BVR seems to be similar, but they are not. It could be helpful to analyze other levels of refinement of features addressed in VIVACE.

Most approaches provide a way for varying the control flow, involving the functional and behavioral perspectives. Most of them only focus on simple tasks, gateways, and start and end events. Very few also involve other kind of activ-

ities, events, roles, and data objects. The most expressive approaches, e.g., BPMNt and CVL/BVR, provide mechanisms for varying every possible element or segment but sacrificing usability since they do not provide any graphical BPMN extension but an XML-based configuration mechanism.

Multi-artifacts is the most common technique, having at least a base BPMN model and a feature model. Having variation points and variants in different models requires the definition of more components. However, since there is a limit on the number of elements that a user can handle on a single model [43], multiple models benefit understandability when the family increases.

Most approaches do not provide any currently available tool. Most of them are exclusively focused on proposing the variability approach, some of them also giving configuration support, but not focused on other aspects of a BP family. This fact negatively affects the potential adoption of some approaches within organizations interested in managing their BP families. Being based on BPMN, most approaches have a standard XML file representation of the BP family. In this context, it is possible to use a software configuration management tool to provide versioning (feature F5.1 in VIVACE).

Finally, there is a lack of empirical studies that determine the utility and limitations of the approaches. The vast majority of them are academic proposals that have not been used in a broader context.

4 Model-driven BP families management (BPFM)

The terminology used to describe variability is diverse [57]. However, the review we carried out for BPMN-based variability approaches showed that although each one presents specific elements representing the BP family, they have mostly the same meaning and purpose with different names and models. It not only happens with BPMN-based approaches but with variability approaches in general, as it is reported in the literature we analyzed [42,60,68]. Heterogeneous terminology and concepts with the same meaning and purpose can be a problem when dealing with different approaches that co-exist within an organization or project, adding a barrier to understanding elements involved and the variability approach in general.

We addressed this challenge by developing a generic BP family metamodel that conceptualizes the BP family's principal elements and relationships to provide a common language to manage BP families with different variability approaches. We are not defining a new language or variability approach but a metamodel that models a process family at a high level of abstraction and independently of any modeling language.

We focus on organizations that manage BP families and deal with existing heterogeneity within approaches regarding languages and supporting technologies. In Sect. 4.1 we present the BP family metamodel we propose, based on which we defined the BP family management approach described in Sect. 4.2 and its tool support in Sect. 4.3.

4.1 Metamodel-based conceptualization

Based on the analysis we carried out of existing approaches BPMN-based and others from the literature, including our own, we defined a metamodel to conceptualize the principal elements and their relationships to manage BP families. The metamodel in Fig. 6 presents an approach-independent conceptualization of a BP family.

A business process family (`BPFamily`, also known as Configurable Process Model [5], Business Process Line [68], Reference Process Model [57], or Customizable Process Model [60]) represents a collection of variants of a given business process (`ProcessVariant`, also known as Customized Model [5]), capturing both the commonalities and the differences of the process variants. The process variants pursue the same or similar business objective and have a common behavior (`BaseProcess`).

The base process can define multiple variation points (`VariationPoint`, also known as Configurable Region [5], Configurable Nodes or Adjustment Points [68]). Variation points allows fragments of the base model to accept variants (`Variant`, also known as Configuration Alternative [5]), depending on the application context of the process variant (`Context`). This context defines multiple requirements `ContextRequirement` that define conditions under which a particular variant of a variation point shall be selected (`Context Condition`, also known as Configuration Context Condition [5] or Configuration Decisions [68]), restrictions regarding the selection of variants (`Constraint`, also known as Configuration Constraint [5] or Configuration Requirement [68]), and guidelines (`Configuration Guidelines` [68]) which are just recommendations.

From the business process family specification, it is possible to derive a process by defining a configuration (`Configuration`). It provides a specific context and fulfills the context requirements, typically choosing a particular variant for each variation point in the base process. In most approaches, this derivation process is a manual task; in our proposal, process variants are generated in an automated manner based on Model to Model (M2M) transformations, using the configuration defined by the user as input. Moreover, this

In Table 5 we present a mapping of concepts from the BPMN-based approaches we evaluated in Sect. 3 to the metamodel concepts. This table aims to show how the corre-

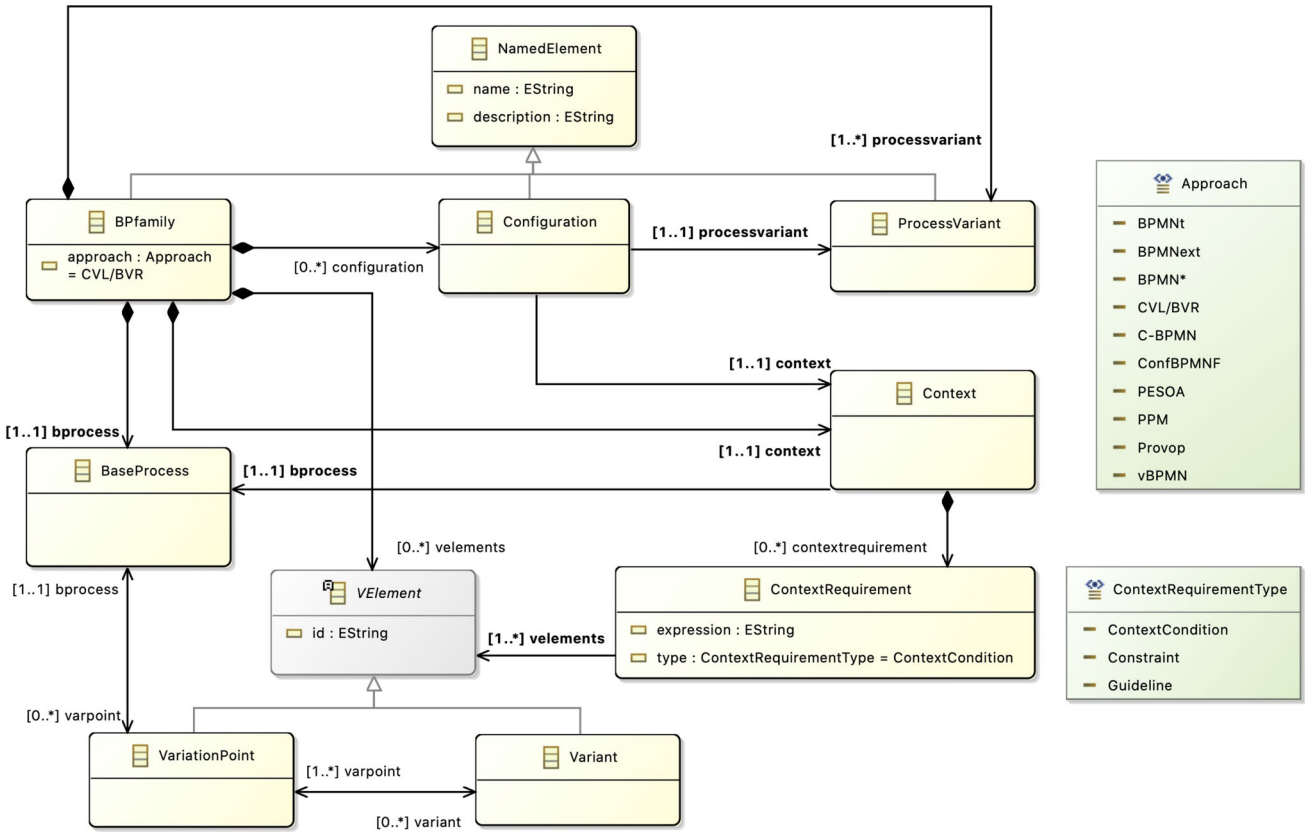


Fig. 6 BP families metamodel

Table 5 Mapping of approaches to metamodel concepts

Approach	Base process	Variation point	Variant	Context	Configuration
ABIS	ABIS proc. template	Variable region	Process fragment	–	–
ADOM	ADOM-BPMN model	Any element	–	–	–
BPFM_C	BPFM_C model	Tree node (non-leaf)	Tree node (leaf)	BPFM_C model	Conf. file
BPFM_N	BPFM_N model	Adjustment points	Fragments	Feature model	Conf. file
BPFM_P	BPFM_P model	Stereot. element	Fragments	–	Conf. file
BPMN*	BPMN* model	Stereot. element	Stereot. element	Feature model	Conf. file
BPMNext	BPMNext model	VPActivity	VActivity	–	Conf. file
BPMNt	BPMN model	Tailoring rel.	Tailored model	–	Tailored model
C-BPMN	C-BPMN model	Config. element	Config. attrib.	–	–
ConfBPMF	BPMT	Presence condition	Activities	Feature model	–
CVL/BVR	BPMN model	Placement Frag.	Replacement Frag.	VSpec	Resolution model
DECO	DECO model	Config. element	Activities	–	–
PESOA	Any process model	Annotations	Activities	Feature Model	Conf. file
PPM	BPMN model/PPM	Any element	–	BPMN-Q query	–
Provop	Any process model	Adjustment points	Change options	Context model	Conf. file
vBPMN	vBPMN model	Adapt. segments	Adapt. pattern	Adapt. rules	Conf. file

sponding concept would be managed within our metamodel, describing a mapping to a specific concept from it. However, as the metamodel is generic, other mappings should be defined to other state-of-the-art variability approaches that are not BPMN-based to complete/refine its capabilities to support heterogeneous variability approaches.

Regarding the mappings presented, for example, for two different approaches such as CVL/BVR, and BPMNext, the base process would be a BPMNext model in the first case, and simply a BPMN model in the second one. It means that when creating a BP family (which is the same concept for all approaches) and selecting the corresponding approach (attribute approach of type Approach), in the first case, a BPMNext model would be expected. In the second one, a BPMN model would be. From a more practical point of view, regarding implementing the approach in the BPFM application, once the language is selected for a BP family, we would check the base process model against the corresponding metamodel for the language (e.g., .ecore) to be compliant with the metamodel.

Nevertheless, this metamodel is an initial conceptualization that showed works for the approaches we have implemented. However, there are some elements that need further analysis. For example, when the approach is multi-artifact such as CVL/BVR, in the creation of the BP family also the feature model (VSpec) defining the variability over the base process should be part of the definition of the family. This could be easily solved by adding an attribute specifying the technique to the BPfamily concept, so when selecting the approach, if the technique is multi-artifact other models could be required to define it apart from the base process. As for now, in the BPFM application we have implemented support only for single-artifact techniques.

Variation points and variants take many forms ranging from annotations or special symbols defining adjustment points (e.g., BPMN*) to complete process fragments (e.g., CVL/BVR) through stereotyped BPMN elements (e.g., BPMNext). Context requirements also take many forms. In approaches without any specific model to represent them, e.g., BPMNext, the context only expresses the configurable relations between variation points and variants. The metamodel is generic enough to represent all these cases abstractly.

Based on these mappings, each variability approach could be managed similarly, i.e., by using the concepts and relationships defined in the metamodel. The metamodel unifies concepts to understand better and compare the approaches. It also allows thinking on managing different approaches within the same organization settings, providing support for other needs in a homogenized way. Nevertheless, it is still in a preliminary version, and further evaluation is needed to improve its definition and connect it with concrete approaches.

4.2 Business process families management

A typical organization needs to deal with an ecosystem of different applications, technologies, and processes, including process families. In [19], we presented an experience report of the case study in Sect. 2. We detected that providing support and guides for managing BP families was a significant challenge for the organization to shift the focus of their daily operation to BPs, including automated support with a BPMS.

Although our case study was restricted to the DGRC organizational unit, several other units were involved in the BPs they carried out, with decentralized management and heterogeneous technology support. We detected that to help all participating units provide a homogeneous way of managing BPs and BP families, a centralized approach to managing the elements involved could help. We identify several activities to support the three basic steps for managing a BP family: modeling, configuration, and variant derivation. Other activities, e.g., verification and evolution, can be considered but are still out of the scope of this work.

In [16] we experimented with an existing variability approach specific for software processes, the vSPEM approach, based on the Software Process Engineering Model (SPEM) [50] standard. Two roles are involved in managing BP families, i.e., Process Engineer and Final User, where the first one is in charge of modeling the family using the selected variability approach. The second one configures the process variant based on the organization's needs. Both roles can participate in the variants' generation with or without automated support, taking the base process and the defined configuration as input. A third role, i.e., Domain Expert, assists both of the roles mentioned above in their activities.

We extended our initial view to support managing process families of any kind, including different variability languages and approaches. However, we focus on the definition of BPMN-based families.

Modeling. A process engineer, assisted by a domain expert, models a process family using a selected approach integrated into a centralized management repository. Although we focus on BPMN-based proposals, any approach can be used, and also different approaches can be used for different process families. The main activities that support the modeling of process families are *Create family* that defines the approach to follow, and *Model family*, that creates the BP family models, i.e., as a minimum: the base process, variation points, and corresponding variants, and other models depending on the approach, such as a feature model. Other supporting activities are *Import family*, *Export family*, *Save family*, and *Delete family*.

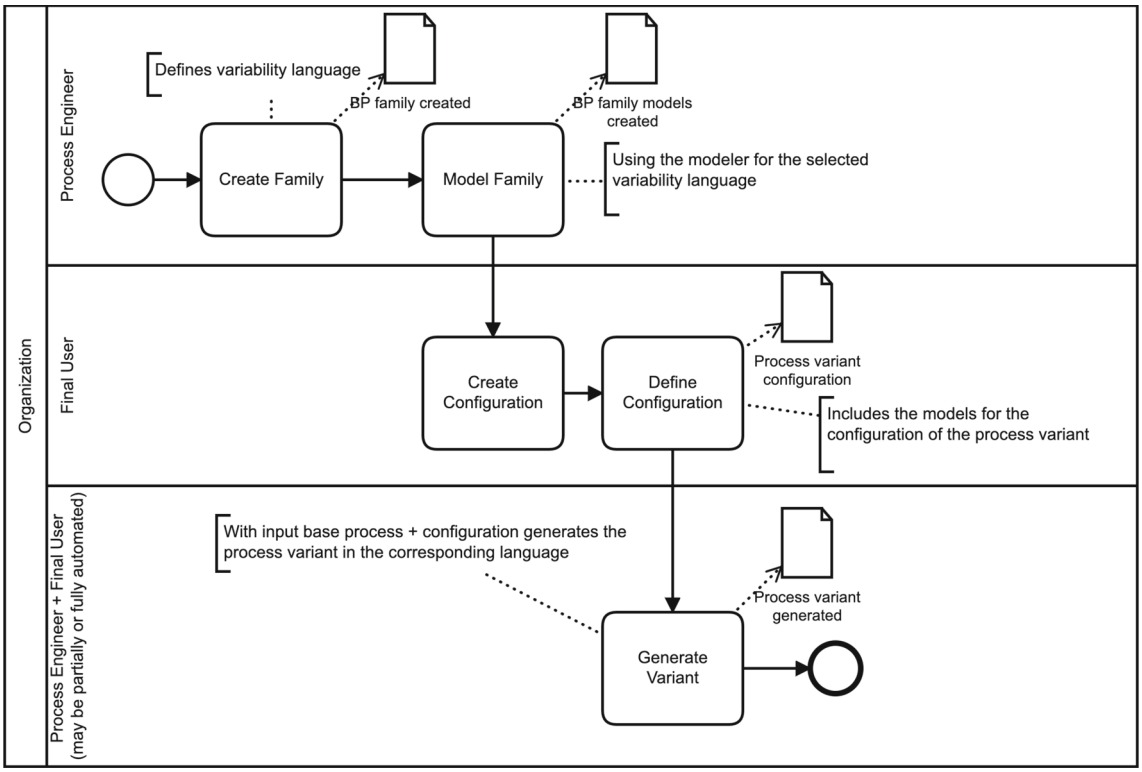


Fig. 7 Successful path for the main activities of the BP families management process specified in BPMN 2.0

Configuring. A final user, assisted by a domain expert, participates in the configuration of a process variant. Depending on the chosen approach, the configuration of a process variant can involve several models to specify the specific variant's context. The main activities that support the configuration are *Create configuration* that creates a new configuration definition, and *Define configuration*, which defines which variants are selected for each variation point. Depending on the BP family approach, other models must also be defined. Other supporting activities are *Import configuration*, *Export configuration*, *Save configuration*, and *Delete configuration*.

Variant Generation. Based on the defined configuration and the process family model (i.e., base process, variation points, selected variants, and other required models), the specific process variant is obtained by performing the *Generate variant* activity. The result of the generation is a process model that corresponds to the chosen process variant, ready to be used as input for implementation and enactment within the organization. Variants' generation can be performed entirely or partially automated, with or without human assistance (e.g., based on input by the Final User/Process Engineer) or even performed directly by any of them within the task. Thus, the activity can be delivered as a user or automatic task (e.g., a service task). The implementation of the process will depend on each variability approach's tool support level. Other supporting activities are *Import variant*, *Export variant*, *Save variant*, and *Delete variant*.

The flow for the activities defined for managing process families can be modeled as a business process itself. The main activities focus on creating and modeling a family, configuring it for a specific process variant using defining configuration models, and generating the corresponding variant. Figure 7 presents the successful path for the defined activities. Other activities such as save, delete, import, and export for each stage, can be performed at any moment, so they are not included in the control flow for simplicity.

4.3 Tool support

To support our proposal, we have developed a web application named Business Process Family Manager (BPFM) [11] that can be used by final users and process engineers in the organization to manage different variability approaches in a homogenized way. The web application is generic since it is based on the concepts and relationships of the metamodel we have presented in Sect. 4.1. It allows the organization to use and manage different variability approaches that can co-exist within the application, each with its specific configuration. In Sect. 5 we illustrate its use with the BPMNext approach, but so far, we have also implemented the vSPEM approach to show its capabilities.

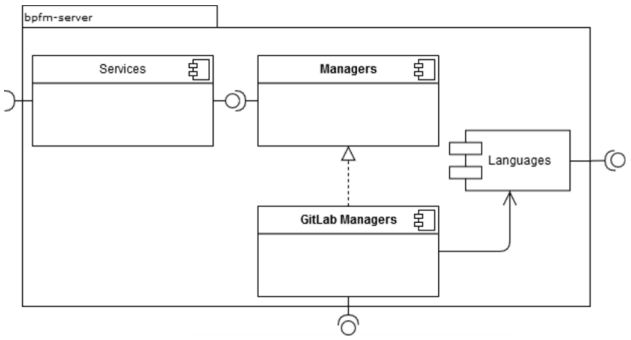


Fig. 8 Architecture of the BPFM server

The web application works with a central repository that is transparent to final users since the application carries out the check-in and check-out actions and commits. It follows a client-server architecture with a thin client that interacts with the server using a REST API exposed by it. The client provided the user interfaces to navigate the web application and was implemented using the framework HTML/Javascript ReactJS[2] and a CSS template. The server provides services for CRUD of process families, configurations, and variants, managing the integration with external tools (e.g., Gitlab repository, ATL engine, BPMN 2.0 modeler). It is also responsible for processing and storing the data received through the API using JSON files, managing the Gitlab structure (repository, directories, files), and users' sessions and permits. It is implemented using Java[3].

The *Services* package includes the REST API services that the server exposes, which invokes the logic from the *Managers*. The *Managers* package defines a set of implementation-independent business logic interfaces: IFamilyManager, IConfigurationManager, IVariantManager, ILanguageManager, which define the methods to support the CRUD operations regarding managing families, their configurations and variants, and invoking the execution of external tools.

The integration with the Gitlab repository is performed via the *Gitlab Managers* which invokes the Gitlab REST API. To support the interaction with the Gitlab repository in a transparent way to the user, we defined a structure for families within an organization that allows us to use the built-in hierarchical view and memberships as part of the defined activities.

The *Languages* package includes several interfaces that must be implemented to add a new language to the BPFM and their corresponding tool support. The interfaces manage the integration with external tools for configuration (ConfigureVariantTool), generation (GenerateVariantTool) and visualization (ViewVariantTool). Each languages' specific configuration is defined in a .properties file where for each one, the specific information is included. All elements must be included in the server configuration directory to be accessed by the web application at runtime to manage the defined tools.

To support the automated generation of process variants homogeneously, the inputs for the transformations would be the same for every language that is integrated, i.e., (i) the input base process model defining the control flow of the BP family, the variation points, and possible variants for each one (multi-artifact approaches would also need extra models such as the feature model), and (ii) a configuration model defining the instantiation of variation points for the specific variant. For the implementation of the prototype, we used a simplified version of the configuration relations defined in the metamodel in Fig. 6.

5 Validation of the proposal

To validate the BPFM approach for managing BP families and the tool support we provide, we first carried out a proof-of-concept using the BPFM application to show the support our managing approach provides for different variability approaches (c.f. Sect. 5.1). Then we conducted an empirical study with real users to test functionality and usability characteristics of the BPFM (c.f. Sect. 5.2).

5.1 Proof-of-concept

In this section, we present the proof-of-concept we carried out over the BPFM prototype we have developed, using the real BP family from our university 'Assignment of positions in academic exchange program" introduced in Sect. 2.

To show the support for different variability approaches, we created a process family and generated specific process variants for the vSPEM software process language (extension of SPEM) and the BPMNext business process language (our extension for BPMN 2.0). However, in this application, we focus on the BPMNext approach for which we provide a step-by-step execution of the complete successful flow of activities for managing BP families, as defined in Fig. 7.

5.1.1 Modeling

Figure 9 depicts the full BPMNext model of the illustrative example specified in the BPMNext language we will use in this section.

The first activity for managing a BP family within our approach is *Create family*, which defines as primary elements, the name, and the variability language used. As mentioned, when creating a BP family, the corresponding Gitlab structure is created within the family repository. The

[2] ReactJS. https://reactjs.org/.

[3] Java. https://www.oracle.com/java/.

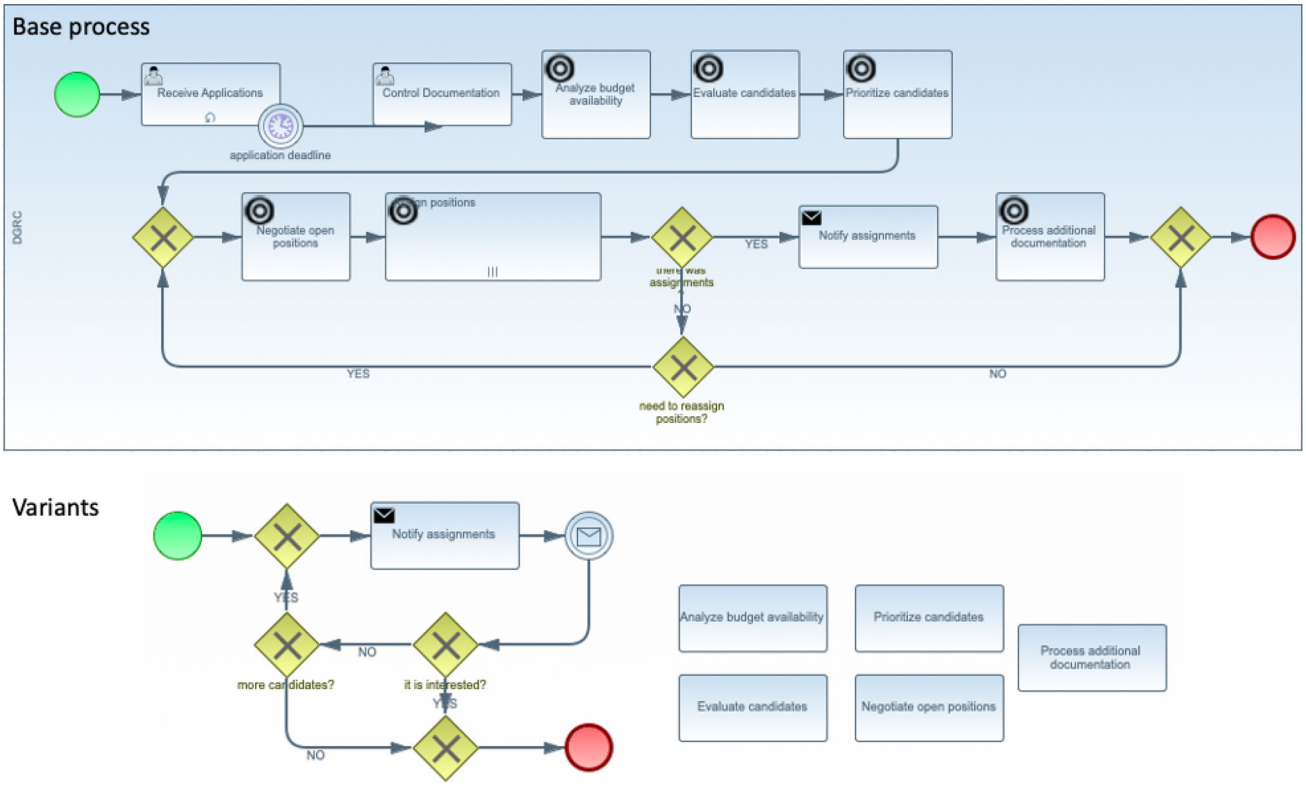


Fig. 9 BPMNext model of the exchange program process

second activity is *Model family* where the editor registered for the selected language would be invoked and open for the user to specify the base process for the process family, including at least the defined variation points and corresponding variants. Depending on the selected approach, other models would also be included. The application can also import an XMI file of the base process compliant with the chosen language's metamodel. So far, we have implemented the second option, so the base process models are imported.

5.1.2 Configuring

The BPFM web application allows defining a configuration for specific variants after the family's definition. In the first place, the *Create Configuration* activity allows defining as primary elements the name for the configuration and the selected BP family. After that, the activity *Define Configuration* allows selecting the specific variants for each variation point by invoking the registered tool supporting the chosen language. Depending on the chosen approach, other models would also be included. The application can also import an XMI configuration file compliant with the variant configuration metamodel as with the base process. So far, we have implemented the second option, so the user must import the variants' configuration models as XMI files.

The content of the XMI file of the "Erasmus Mundus" configuration is presented in Listing 1, which is compliant with the metamodel's variant configuration (simplified). For each variation point "varPoint" defined in the base process, a variant is selected e.g. variation point "VP1" has as selected variant "DEL1," "VP2" variant "VT2," etc.

5.1.3 Variant Generation

Once the configuration for a specific variant is defined and the corresponding XMI file is available, the *Generate variant* activity can be done to generate the corresponding variant automatically. There is a "Generate" button in the configuration screen performs the generation by invoking the registered tool. The models are defined as input for it and provide the process variant model in the defined language. As mentioned before, the default generation language is ATL, which we integrated as an ATL runner module invoked with the data described in the properties file for each variability language implemented.

The ATL transformation goes from the BPMNext metamodel (BPMN 2.0 extension with variation points) to the BPMN 2.0 metamodel to generate the process variant without variability. It includes several rules that apply to elements in the BPMNext extension, which can be grouped into two categories: i) rules for copying "as is" BPMN 2.0 elements that

present no variability, and ii) rules for "occupying" (including deleting) variation points elements with the selected variant that is defined in the configuration file. In Listing 2 we present an example of the second type of rule for occupying a UserTask with the corresponding variant. In the transformation, we copy all the information regarding the UserTask to be generated from the base process's variant definition (as defined in the "using" tag of the rule). Figure 10 shows the generated variant.

Listing 1 .XMI file of the configuration "Erasmus Mundus" compliant with the variant configuration metamodel

```
<?xml version="1.0" encoding="UTF-8"?>
<org.csic:configuracion
   xmi:version="2.0"
   xmlns:xmi="http://www.omg.org/XMI"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance"
   xmlns:org.csic="http://www.ejemplo.org/
     confvar"
   xsi:schemaLocation="http://www.ejemplo.org/
     confvar/BPMNext2BPMN/metamodels/
     ConfiguracionVariantes.ecore">
   <varPoints id="VP1" selectedVariants="DEL1"/>
   <varPoints id="VP2" selectedVariants="VT2"/>
   <varPoints id="VP3" selectedVariants="VT3"/>
   <varPoints id="VP4" selectedVariants="DEL4"/>
   <varPoints id="VP5" selectedVariants="VSP5"/>
   <varPoints id="VP6" selectedVariants="VT6"/>
   <variants id="DEL1"/>
   <variants id="VT2"/>
   <variants id="VT3"/>
   <variants id="DEL4"/>
   <variants id="VSP5"/>
   <variants id="VT6"/>
</org.csic:configuracion>
```

5.1.4 Summary of the proof-of-concept

As we have shown in the proof-of-concept, the BPFM application provides support for our model-driven approach for managing BP families throughout the complete process presented in Fig. 7. The tool is generic based on the BP family metamodel, allowing different variability approaches to coexist within the application and the entire organization by providing a user transparent integration with a centralized Gitlab repository to store and manage each BP family. It supports the definition of roles and permits within the organization to work with the families, also registering its use.

The application also allows import/export of files for every integrated variability approach (the base process in the selected language, variant configuration, process variant generated) to store them and make them available directly in the Gitlab repository. Also, new variability approaches and external tools can be easily integrated into the existing implementation, as presented in Sect. 4.3.

Another key element is the support we provide for the automated generation of process variants. In most approaches, this derivation process is a manual task; in our proposal, process variants are generated automatically based on Model to Model (M2M) transformations, using as key input the user's configuration.

5.2 Empirical study

We have carried out an initial empirical assessment of the BPFM application supporting our approach for managing BP families with potential users. We recruited participants via generic mailing lists to reach a broad audience interested in the subject. Finally, 12 of them participated. The study was defined and carried out with the required scientific rigor following the definitions and guides in [72–75] to guarantee the quality of its results.

5.2.1 Study design

The object of the study (OoS) is the BPFM application prototype implementing our BP family management approach interacting with real users in a similar context to the one of its intended use [73]. The objective (purpose) of the study is the assessment of the functional suitability and usability characteristics of the BPFM application implementing our BP family management approach. The research questions for the study are defined as:

> **RQ1.** Does the BPFM application provide adequate functional support to manage BP families as proposed in a centralized context of use?
> **RQ2.** Does the BPFM application provide usability support for users to operate on it with satisfaction in a centralized context of use?

The evaluation of quality characteristics over software products is not new. Still, it has been pursued over the last three decades to gain insight into desired properties of a software product. Quality characteristics to be evaluated over a software product are defined in the quality model, and corresponding quality characteristics standard ISO/IEC SQUARE 25010 [29] (which superseded the previous series of standards ISO/IEC 9126 [30,31]). It defines eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability. Our main interest was to evaluate the functional support that our approach provides for managing BP families and how satisfied users are when operating the software. We choose to evaluate functional suitability and usability characteristics. The definition of these characteristics [29] is as follows:

Listing 2 Rule (excerpt) for occupying a variation point in BPMNext with the variant defined in the configuration

```
-- @path BPMN=/BPMNext2BPMN/metamodels/BPMN20.ecore
-- @path BPMNext=/BPMNext2BPMN/metamodels/bPMNext_2018.ecore
-- @path confvar=/BPMNext2BPMN/metamodels/ConfiguracionVariantes.ecore
module configurarVariante;
create OUT: BPMN from IN: BPMNext, IN1: confvar;
......

-- rule for occupying a VPTask variation point
rule VPTask2UserTask{
    from input: BPMNext!VPTask
            (if thisModule -> findVTaskInBaseProcess(input.id) -> oclIsUndefined() then
                false
            else
                (thisModule -> findVariantInBaseProcess(input.id) -> oclIsKindOf(BPMNext!VTask))
                and (thisModule -> findVTaskInBaseProcess(input.id).task ->
                    oclIsKindOf(BPMNext!UserTask))
            endif
            )
    using { variant: BPMNext!VTask = thisModule -> findVTaskInBaseProcess(input.id); }
    to output: BPMN!UserTask(
            id <- variant.task.id,
            incoming <- input.incoming,
            lanes <- variant.task.lanes,
            name <- variant.task.name,
            outgoing <- input.outgoing,
            ...
        )
}
```

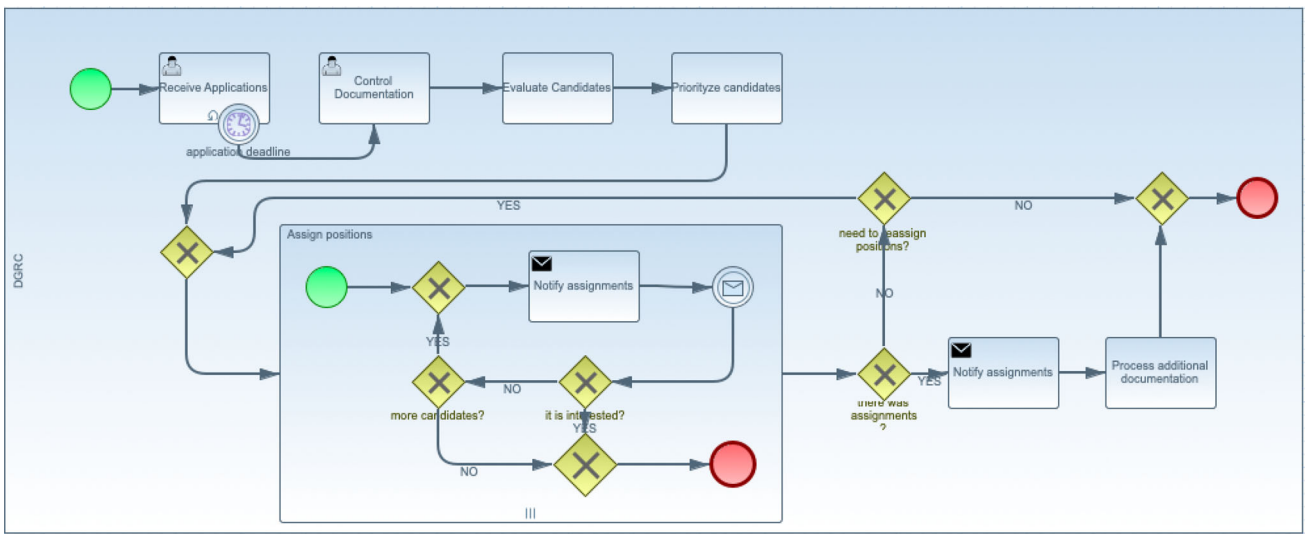


Fig. 10 BPMN 2.0 model of the generated variant for the "Erasmus Mundus" process variant

Functional suitability. "Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions."
Usability. "Degree to which specified users can use a product or system to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."

The context corresponds to the deployment of the BPFM application (c.f. Sect. 5.2.2) in a central web server connected with a central repository, which is accessible for users in a similar way that it is intended when deployed in a specific organization for users to operate it.

Selection of participants: We reached out to researchers in our network from several Latin Ameri-

can countries and Spanish research groups, as well as professionals from Uruguay. We did not require any specific knowledge apart from basic modeling knowledge. Participation was voluntary, and the questionnaire could be answered anonymously to guarantee honest answers without the fear of revealing the participant's identity to us. Finally, 12 people participated in the study from universities and the software industry.

Measures and variables: To be able to measure the degree of Functional suitability and Usability as perceived by real users operating the system, we defined specific criteria for each characteristic based on its subcharacteristics. We evaluate each subcharacteristic using one to three questions regarding specific criteria associated with the subcharacteristic, answered on an ordinal 1 to 5 scale (c.f. Tasks and questionnaire below).

For Functional suitability (FunSuit) we consider the three subcharacteristics defined for it: *Functional completeness* (FunComp), *Functional correctness* (FunCorr), and *Functional appropriateness* (FunAppr). For Usability (Usab) we consider four of the six characteristics defined for it: *Operability* (Oper), *Learnability* (Lear), *Appropriateness recognizability* (ApprRec), *User error protection* (UserErr). We excluded in this evaluation characteristics that focus on *User interface aesthetics* and *Accessibility*, for future work.

We directly measure each criterion from the answers to each value in the ordinal scale as provided by the users. We calculate the mean, median, and mode values for each criterion, and then we aggregate them to provide values for the subcharacteristic. Finally, we aggregate these values for each sub characteristic to provide values for the complete characteristic. In Table 6 the characteristics, subcharacteristics, criteria, and corresponding questions are presented.

Tasks and questionnaire: We defined four tasks for users to carry out within the BPFM application and a questionnaire for them to assess the functional support that it provides for performing the tasks and their satisfaction with the operation of the software.

The materials provided to participants included first an introduction with concepts, definitions, and the presentation of the BP family of the proof-of-concept "Assignment of positions in academic exchange program" modeled with the BPMNext variability language showing variation points and corresponding variants. We presented the four tasks to be carried out within the BPFM application in a logical flow script (following the BP family management process shown in Fig. 7), which are defined as follows:

> *Task 1 (T1.)* Create Family: Create a new process family in the application using the BPMNext language. The family name to create must be of the form FamilyUUID where UUID is an identifier for which we provide a generator. We also provide the family model to be used.
>
> *Task 2 (T2.)* Define Configuration. Define a configuration for the Erasmus Mundus variant of the process family. The name of the configuration must be ErasmusMundus and be within the family defined in Task 1. We also provide the configuration model.
>
> *Task 3 (T3.)* Generate Variant. Automatically generate the process variant corresponding to the configuration created. To do this, you must use the generation option provided in the application for such configuration.
>
> *Task 4 (T4.)* New Configuration and Variant. Create a new ErasmusMundus configuration by following all the steps described in Task 2 with the same configuration file. This time the name of the configuration should be ErasmusMundusNew. Generate a variant again by following all the steps in Task 3.

Each task involved interacting with the BPFM application to complete the assignment, including manipulation of models, navigating over different parts of the interface, and executing the functions it provides. The tasks were defined to provide a complete view of the capabilities of the approach and the BPFM prototype and be agile and take the less possible time to do them for participants to finish the study.

For the definition of the criteria and related questions, we identified key functionalities in the management of BP families and usability items, mainly based on previous works we have carried out on the evaluation of functional and non-functional characteristics of BPMS platforms [17,20], the references we took into account for those evaluations and others from other domains such as [1,13]. These questions are directly related to the execution of tasks we propose and can be answered after executing the BPM application tasks.

At the end of the questionnaire, we asked for the identifier generated for the family name they worked with (to relate what they did in the BPFM application and the form). We included a demographic section where we asked questions regarding their origin (country and work area), education level, and knowledge level on BP families and variability. We also included an optional space to enter a contact email.

For each question, we use an ordinal 5-point Likert scale [39] including in the answers the specific values in addition to the scale numbers [36], to minimize risks regarding the scale values and the possibility of not understanding their meaning. We defined the scale for each question as follows: 1 - not adequate 2 - somewhat adequate 3 - neither adequate nor inadequate 4 - adequate 5 - very adequate.

Table 6 Quality characteristics and subcharacteristics and defined criteria and questions

Characteristic	Sub-Characteristic	Criteria	Questions
Functional suitability	Functional completeness	Variability variety	Does the software provide an adequate variety of variability approaches and languages?
		Configuration of variants	Does the software allows to define a configuration for a specific process variant ?
		Generation of variants	Does the software allows to generate automatically the process variants with the configuration?
	Functional correctness	Model Validation	Does the tool support model validation?
		Model Importing/Exporting	Does the software provide a variety of ways to import/export models for ongoing use?
		Version control	Does the software provides version control over models, configurations and process variants?
	Functional appropriateness	Prescribed Methodology	Does the software aid the user by presenting a user-friendly methodology?
		Collaboration	Does the software allows to work in a collaborative way over a process family ?
Usability	Operability	User Interface	Is the user interface easy to navigate presenting results in a meaningful way?
	Learnability	Learning Curve	Is the tool easy to learn? Is the tool easy to use correctly (beginning/advanced users)?
	Appropriateness recognizability	Data Visualization	How well does the tool present the data and the modeling results?
	User error protection	Error Reporting	How meaningful is the error reporting?

5.2.2 Study execution and data collection

The execution of the study consisted of five steps:

Step 1 we prepared the infrastructure to carry out the validation, i.e., deploying the BPFM application to a central server , with a virtual machine running Ubuntu, an Apache Tomcat 8[4] web server and connected with a central Gitlab Omnibus[5] repository. We had to install all software and configure all elements to have the BPFM application up and running for the study.
Step 2 we made available the materials [15] in Google drive for carrying out the study, i.e., the tasks document with the script of tasks execution, which were tested and adjusted to provide a logical flow in the use of the application, and the models to be used.
Step 3 we created and published the questionnaire [15] in Google forms referencing the materials and the BPFM application from (i) and (ii), and tested and adjusted it for a better user experience, including the addition of a free text box at the bottom of each question for participants to write open comments.
Step 4 we sent the invitations with the link to the questionnaire to the potential participants, and the ones who accepted carried out the tasks over the BPFM application and answered the questionnaire provided.
Step 5 we recovered the excel sheet from the Google form with the answers to the questionnaire, which makes the data set we analyzed for answering the research questions and conclude the study.

The first three steps took us almost two months to have all elements in place, tested, improved, and functioning. We planned on having the questionnaire open online from the first week of June 2021 to the end of the month, but finally, we left it open until middle July 2021. We got 12 participants whose answers we analyzed in the following. The raw data downloaded from Google forms are in a .csv file available at [10] (within the documents/validation).

Table 7 shows the complete dataset we collected with the answers from participants, grouped by characteristic, subcharacteristic, and criteria. We present values for each criterion since each one had a corresponding question (c.f., Table 6) that was answered in the 1 to 5 scale. For each one, we present how many participants selected it (columns 1–5), the minimum value for the criteria (min) and the maximum (max), the mean value, the median and mode, and standard deviation (std).

5.2.3 Results analysis

In Fig. 11 we present the summary of the result analysis for the functional suitability and usability characteristics. For functional suitability we present a box plot for each criteria grouped by subcharacteristics in Fig. 11a functional

[4] https://tomcat.apache.org/download-80.cgi.

[5] https://about.gitlab.com/install/#ubuntu.

Table 7 Dataset collected from the empirical study

Characteristic	Sub-Characteristic	Criteria	1	2	3	4	5	min	max	mean	med	mod	std
Functional suitability	Functional completeness	Variability variety	0	2	4	5	1	1	5	3,42	3,5	4	0,90
		Configuration of variants	1	0	1	5	5	1	5	4,08	4	4/5	1,16
		Generation of variants	0	2	1	4	5	2	5	4,00	4	5	1,13
	FunComp									3,83	4	4	
	Functional correctness	Model Validation	2	0	6	3	1	1	5	3,08	3	3	1,16
		Model Importing/Exporting	0	1	0	5	6	2	5	4,33	4,5	5	0,88
		Version control	1	0	5	6	0	1	4	3,33	3,5	4	0,88
	FunCorr									3,58	4	4	
	Functional appropriateness	Prescribed Methodology	0	2	4	4	2	2	5	3,50	3,50	3/4	1,00
		Collaborative work	1	0	4	6	1	1	5	3,5	4	4	1,00
	FunAppr									3,50	4	4	
FunSuit										3,64	4	4	
Usability	Operability	User Interface	0	0	2	7	3	3	5	4,08	4	4	0,66
	Learnability	Learning Curve	0	0	1	7	4	3	5	4,25	4	4	0,62
	Appropriateness recognizability	Data Visualization	4	3	4	1	0	1	4	2,17	2	1/3	1,02
	User error protection	Error Reporting	4	0	4	3	5	1	5	2,75	3	1/3	1,42
Usab										3,31	4	4	

completeness, (b) functional correctness, and (c) functional appropriateness. For usability we present a box plot for each criteria in Fig. 11d since they correspond one to one to each subcharacteristic (c.f., Table 6). Although we used the Likert scale for answers (c.f. Sect. 5.2.1 Tasks and questionnaire) that goes from 1 to 5 (value 1 means inadequate and value 5 means very adequate), in the box plots, the y-axis goes from 0 to 6, just for a better presentation (if using 1 and 5 as extremes of the axis, points and lines corresponding to that values were not so clear).

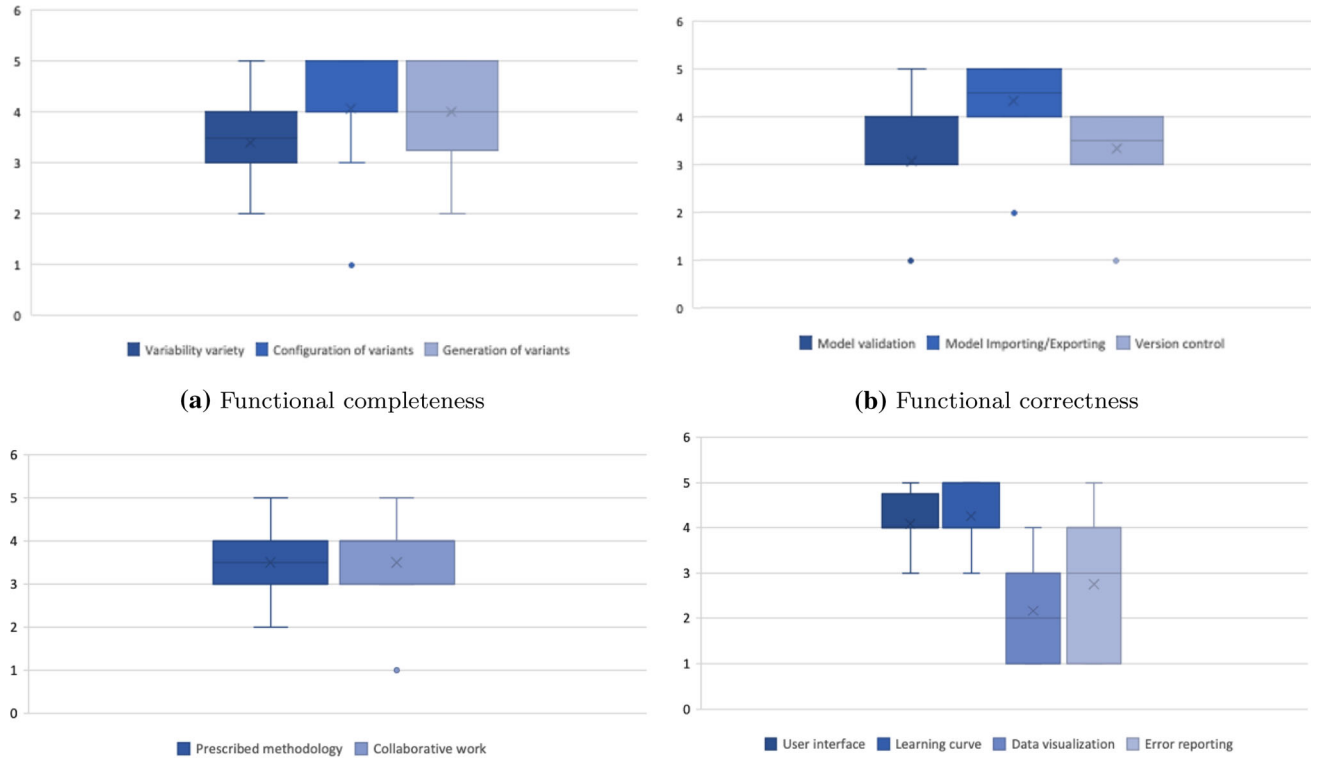
Regarding Functional completeness, which evaluated the most important functionalities to manage BP families within our proposal, i.e., variability variety, the configuration of variants, and generation of variants, it can be seen in Fig. 11a that for variability variety, most answers correspond to values within 3 and 4 with a median of 3,5 for configuration of variants, most answers correspond to values 4 and 5 with a median of 4 , and for generation of variants also correspond to values within 4 and 5 and a median of 4, but data are more dispersed than in the previous ones. Finally, the aggregated value for the variable FunComp (corresponding to the complete subcharacteristic) is of value 4 for the median and 3,83 for the mean. We can conclude that functional completeness is very well perceived by users, meaning that the support for the main functionalities for managing BP families is present in a comprehensive manner.

Functional correctness deals with model manipulation, i.e., model validation, model importing/exporting, and version control. It can be seen in Fig. 11b that the answers follow a similar distribution to the ones presented before, with most values within 3 and 4 for model validation with a median of 3 and version control criteria with a median of 3,5. Differently for model importing/exporting, most values are within 4 and 5 with a median of 4,5, which suggests that this function is very well supported. Finally, the aggregated value for the variable FunCorr of the complete subcharacteristic is 4 for the median and 3,58 for the mean; we can conclude that there is still room for improvements to be made for these criteria.

Regarding functional appropriateness, the criteria we evaluated correspond to support for carrying out the tasks, i.e., prescribed methodology and collaborative work. In Fig. 11c, it can be seen that most answers are also within values 3 and 4 for both criteria with a median of 3,5, and 4, respectively. The values for FunAppr are 4 for the median and 3,50 for the mean, meaning that there are still elements that should be improved within these criteria.

Finally, the aggregation of the three subcharacteristics values presented above (FunComp, FunCorr, FunAppr) makes the variable FunSuit value of the characteristic functional suitability of value 4 for the median and 3,64 for the mean. It suggests that although some issues are to be reviewed and improved to provide better support for the BP family management approach we propose, functionality was mostly well evaluated. The RQ1 can be answered positively since the results obtained showed that key functionalities to manage BP families are well supported within the approach implemented by the BPM application.

On the usability side, it can be seen in Fig. 11d that the first two criteria of User interface and Learning curve were very well evaluated with most values of 4 and 5 and median values of 4 for each one. The other two criteria regarding Data visualization and Error reporting mainly were evaluated with



(a) Functional completeness

(b) Functional correctness

(c) Functional appropriateness

(d) Usability

Fig. 11 Summary of the functional suitability and Usability result analysis for the empirical study

3 or less for the first case with a median of 2 and with 4 or less for the second case with a median of 3 In both cases with more dispersed data than in the previous ones.

It implies that we need to provide better support for data visualization and error reporting, to help users in a more satisfying experience when using the BPM application. This fact does not surprise us since we already knew that these two criteria were the weakest ones within the BPM application. The aggregated values of the variable Usab (the complete characteristic) are 4 for the median and 3,31 for the mean. RQ2 can be partially answered positively since important criteria such as Data visualization need to be improved.

Comments from the participants also showed us how they appreciated certain aspects of the approach, also providing us with good insight for improving others:

> *Configuration of variants:* P6—"It allows uploading a configuration already made, not defining it there, which would be helpful to not depend on another tool." P8—"Yes, I can see how an organization can maintain in an ordered and the structured way its BP families configurations." P9—"It is very easy to define variants configuration." P10 - "It is really good to be able to define several configurations and only defined once the family." P11—"If you have the option to define a configuration with a file but not to create it on the web itself."
>
> *Generation of variants:* P7—"The application provides automatic generation of variants that are fast and simultaneously downloads the output file automatically, which I consider good." P11—"Easily auto-generated with the base process and one-button setup."
>
> *Collaborative work:* P8—"Yes, because it provides a common repository for BP that offers variability." P10—"Collaborative work is easy by sharing the same workspace."
>
> *Data visualization:* P6—"The information is present, but it does not allow displaying models that have a graphical representation graphically." P10—"It could be improved by connecting with a tool for viewing models." P11 - "The files of the models used are presented, the models themselves are not displayed."

From the 12 participants that carried out the study, most were from Universities, 7 people (58%), and 5 were computing professionals. Regarding their countries, 5 participants (41.7%) came from Uruguay and 5 from Spain, 1 from Chile, and 1 from Ecuador. Half of the participants had an education level of PhD, 3 participants (25%) of MsC, 2 were professionals, and 1 technical. Regarding their level of knowledge on BP families and variability, 5 participants (41.7%) have an intermediate level of knowledge, 3 (25%) good knowl-

edge, and 2 (16.7%) very good knowledge. In comparison, the other two participants had minimal and no knowledge.

5.2.4 Threats to validity

Regarding the threats to validity [74] for the empirical validation, we discuss here the construct, internal, external, and conclusion validity.

Construct validity: the variables we considered correspond to subcharacteristics of quality characteristics that have been long used for assessing product quality, as defined in the standard [29]. The questions defined to assess each criterion were based on previous experiences on assessing approaches and tools used in other works. However, we have not assessed metrics that are usually considered, such as efficiency and efficacy, due to this initial validation. Further work is needed in this regard.
Internal validity: for the selection of participants, we reached out to known researchers and professionals, which could introduce a bias on their answers. To mitigate this and promote honest answers, the questionnaire was anonymous. The study can be reproduced as we made available all the material used and the protocol to carry out the study.
External validity: the setting of the empirical study has several characteristics of organizations that could be interested in using our approach. It is available in a centralized manner which users can access and work as they would do in an organization applying our proposal. The number of participants (12) in the study is also a threat to generalization. Although generalization can be made by analogy ([72,73]), we need further work in this regard.
Conclusion validity: we collected data automatically from the questionnaire so we can rely on its quality for the analysis. For the definition of tasks and questionnaires, we followed existing guides to minimize the threat of dependency on the specific researchers.

5.2.5 Summary of the empirical validation

As we have presented in the analysis of results of the study, the generic approach to manage BP families and the BPMF application supporting it were mainly well evaluated by participants. Tasks were carried out by participants without reporting any setbacks and obtaining the expected results for each one.

Functional aspects of the approach, such as configuration and generation of variants and model importing/exporting, were mainly very well evaluated. Other functional elements such as variability variety, model validation, version control, methodology, and collaborative work have room for improvement. Regarding usability, user interface and learnability were very well evaluated, while data visualization and error reporting need to be reviewed and improved.

With regards to elements well perceived and improvements to be made, valuable comments from participants also provided us with insight for future work, mainly to improve the BPFM prototype, for example, the need for support to directly define configurations in the application (e.g., graphically or questionnaire aided), the need to connect the application with other tools to visualize models, and some clarification regarding the versioning of the families.

This empirical validation was carried out as a first step on getting real users' feedback within a context similar to the intended context of using the approach and application. We can conclude that the validation of the artifacts we propose, i.e., the BPMF generic management approach and the BPFM application, was successfully carried out with good results that provide us with future work guides. We need to carry out more studies ideally in a real organizational setting to confirm the trends perceived within this initial validation.

6 Discussion

We focus on identifying general aspects of BPMN-based approaches and providing a generic solution for its management. All this is grounded in using well-defined modeling languages, such as BPMN, to define a BP family and model transformations supporting the variant generation process. This perspective provides a high level of abstraction with many benefits. First, the metamodel unifies concepts that can be the center of the management activities. Also, many variability approaches could be managed homogeneously and at the same time within an organization. Finally, model transformations improve the generation of process instances. They could support other interesting aspects, e.g., the transformation of BP families from one variability approach to another.

As far as we know, no other proposal aims to support the whole BP family life cycle. The initial validation in Sect. 5 provided fruitful feedback about the BPFM tool and the general approach itself. First, the proof-of-concept showed that it is feasible to implement the general approach for a given variability approach, not absent of limitations. Second, the empirical study allows detecting new strengths and limitations from the perspective of potential users. Although many aspects were well perceived, e.g., functional completeness, others still require further improvements, e.g., prescribed methodology, model validation, collaborative work, data visualization, and error reporting.

Concerning the methodology and error reporting, we need to provide further assistance to users, e.g., showing the process flow. We focus on managing the artifacts defined during the definition process for any approach. There are complimentary works that focus on how to model a process family from a methodological point of view, e.g., steps to follow for

the definition of the base process, identification of variation drivers, and construction of a variation map [44]. They also focus on defining guidelines for deciding on the best method to follow in a given context [6]. This aspect can be integrated into our approach. For example, the Decomposition Driven Method [44] could refine the Model Family activity into more specific tasks. These tasks could provide support to identifying business drivers, assessing their relative strength, and constructing a variation map related to the generation of variants providing traceability from business decisions to variants.

For now, we focused on the three main steps for managing a BP family: modeling, configuration, and variant derivation. Model validation is part of other aspects, such as evolution, that need to be considered but are still out of the scope of this work. As shown in Sect. 3, few proposals provide the formal settings for verifying a BP family. However, in the cases that provide it, e.g., BPMN, and PESOA, we could add tools for verifying syntactic and semantic correctness of a configuration. The metamodel could also be extended to include these constraints and specific context requirements expressed as string expressions for now.

Although the metamodel seems general enough to be the basis for representing different approaches, it is still a preliminary conceptualization. It requires further evaluation and improvement to ensure that the approaches can be connected to it. With this necessary improvement, it is possible to think about a more model-driven repository of concrete aspects. For example, the configuration of a BP family depends on each variability approach, e.g., in some cases, it is just a mapping between variation points and variants (e.g., BPMNext), and in others is a more complex resolution model (e.g., CVL/BVR). BPM considers the configuration file that a given approach provides, but not their details. When defining BPMNext, we considered a configuration metamodel for showing how we can achieve the automatic configuration of variants through model transformations. Nevertheless, it could be considered as a central aspect of the approach. For example, we have applied the same metamodel for the configuration of vSPEM. Thus, by analyzing the configuration needs of each approach, we can define a more general metamodel connected with the one in Fig. 6.

We are focusing on variability at design time, i.e., process variants must be configured before execution. Few approaches also consider defining variants at runtime. Other flexibility issues, such as adaptation, involve different kinds of solutions. The existing approaches focus mainly on imperative processes rather than declarative ones since they present an additional complexity in managing their life cycle. For this reason, we need to analyze how a declarative BP family can be managed. We also need to study the link between the two family types to define hybrid processes [2,66], that is, for example, a family of imperative processes where parts of the process are declaratively defined.

Given the universe of variants generated, approaches should guide the configuration process, ensuring that the chosen configuration options result in process variant models with the expected behavior. The formalization of the whole configuration concept that we provided by defining a metamodel provides means for formally expressing such configuration with tool support simplifying its definition by final users, while in most of the existing approaches, this formalism does not exist. There are proposals to assist the user, both in the configuration of process families [59,78], and in the execution of declarative processes [32]. An open question is whether it is possible to integrate similar strategies to assist in generating variants formally.

We claim that this approach improves support for managing BP families in the long term, reduces technical complexity to end-users, increases usability, reuse, and productivity, and decreases errors within variants' configurations. Technical complexity reduces using the BPFM tool that abstracts and structures the BP family management process and introduces model transformations that automatize some activities. The extensive use of models and transformations, and the incorporation of questionnaires and methodological guidelines that abstract the technical details, increase usability concerning the existent approaches that focus on defining languages and do not provide tool support. Transformations also increase productivity and decrease errors within variants' configurations, avoiding a manual configuration process. The reuse can also be increased since the metamodel provides an abstract representation of concepts for every approach, allowing to share knowledge between different BP families and approaches.

7 Conclusions

Managing business process families within an organization is not an easy task. In the first place, several challenges arise regarding identifying the family, i.e., being aware that process variants present common behavior. After such identification, a second challenge implies the selection of the variability approach to model the family. Our literature review and evaluation of the BPMN-based BP family's approaches using VIVACE showed multiple alternatives. These alternatives range from very basic approaches that allow varying single tasks to complex approaches considering the variability of process fragments involving any process element, e.g., activities, gateways, events, and data objects. Moreover, most approaches focus on modeling the process family. Some of them also give configuration support, but they do not consider other aspects of the process life cycle, which requires

features for supporting enactment, diagnosis, and evolution as defined in VIVACE.

Even focusing on the process family's modeling and configuration, the variant generation process from the base process usually requires an error-prone manual configuration, imposing a great effort on the user that defines the configuration. Although we found a few proposals that try to automate this process, there is no complete approach for the business process specified with the BPMN 2.0 language. Most approaches do not provide any currently available supporting tool. In this sense, we believe the contribution of a model-driven approach, such as the one we presented in this article, can help improving business process management in organizations. We provided a metamodel defining BP family concepts and their relationships, corresponding tool support for family management, and automated generation of process variants from the base process regarding the configuration defined by the user. The BPFM web application is generic, allowing different variability approaches to coexist within an organization. It is extensible by design, allowing to add variability approaches and corresponding languages by implementing the defined interfaces for the new language and configuration files. The validation provides interesting feedback about the BPFM tool and the general approach itself.

Future work focuses on extending the support to other languages into the BPFM application to ease its adoption in organizations. Also, integrating automated support for the configuration process of variants, such as user-friendly forms, helps identify the most suitable variants for each variation point to generate the process variant' configuration. We can also enhance the proposal with less effort and better maintainability, such as adding support for variability rules. Using a standard way to define the rules, we can extend process configuration and generation capabilities. We will also strengthen the approach's validation, carrying out new case studies on the general approach and tool support.

Acknowledgements This work was partially funded by the Comisión Sectorial de Investigación Científica (CSIC), Universidad de la República (UdelaR), Uruguay. We want to thank the students and collaborators who also worked on this project: Emiliano Alonzo, Virginia Bacigalupe, Ignacio Betancurt, Alejandro Brusco, Nicolás Dinetti, Darwin Fernández, Santiago Góngora, Gonzalo López, Leonel Peña, Martín Prino, Fabiana Roldán, Marcela Viera, and Hernán Winter. Also by BIZDEVOPS-GLOBAL project (Ministerio de Ciencia, Innovación y Universidades, y Fondo Europeo de Desarrollo Regional FEDER, RTI2018-098309-B-C31); and G3SOFT project (Consejería de Educación, Cultura y Deportes de la Junta de Comunidades de Castilla La Mancha, y Fondo Europeo de Desarrollo Regional FEDER, SBPLY/17/180501/000150).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Appendix A: Primary studies

Approach	Ref	Year	Title	Author
ABIS	[70]	2011	Adaptive Business Process Modeling in the Internet of Services	Weidmann, M., Koetter, F., Kintz, M., Schleicher, D., Mietzner, R.
ADOM	[58]	2009	Organizational reference models:Supporting an adequate design of local business processes	Reinhartz-Berger, I., Soffer, P., Sturm, A.
BPFM_C	[7]	2016	Business Process Feature Model: An Approach to Deal with Variability of Business Processes	Cognini, R., Corradini, F., Polini, A., Re, B.
BPFM_N	[45]	2011	Modeling and Managing Variability in Process-Based Service Compositions	Nguyen, T., Colman, A., Han, J.
BPFM_P	[51]	2011	A Modeling Approach for Business Processes Based on Variability	Park, J.
BPMN*	[67]	2015	BPMN* - A notation for representation of variability in business process towards supporting business process line modeling	Terenciani, M., Paiva, D., Landre, G., Cagnin, M.
BPMNext	[18]	2017	BPMN 2.0 based modeling and customization of variants in business process families	Delgado, A., Calegari, D.
BPMNt	[55]	2015	BPMNt: A BPMN extension for specifying software process tailoring	Pillat, R., Oliveira, T., Alencar, P., Cowan, D.
C-BPMN	[3]	2014	Deriving configurable fragments for process design	Assy N., Chan N., Gaaloul W., Defude B.
	[78]	2014	Extending BPMN for Configurable Process Modeling	Zhang, H., Han, W., Ouyang, C.
	[65]	2015	Individualization of Process Model from Configurable Process Model constructed in C-BPMN	Sharma, D., Hitesh, Rao, V.
ConfBPMF	[46]	2012	A metaheuristic approach for the configuration of business process families	Ognjanovic, I., Mohabbati, B., Gasevic, D., Bagheri, E.,Boskovic, M.
CVL/BVR	[4]	2012	Applying CVL to business process variability management	Ayora, C., Torres, V., Pelechano, V., Alferez, G.
	[9]	2020	Model-driven support for business process families with the Common Variability Language (CVL)	Calegari, D., Delgado, A., Peña, L.
DECO	[62]	2011	Towards Adaptability and Control for Knowledge-Intensive Business Processes: Declarative Configurable Process Specifications	Rychkova, I., Nurcan, S.
PESOA	[64]	2007	Variability Modeling and Product Derivation in E-Business Process Families	Schnieders, A., Puhlmann, F.
	[37]	2011	Business Process Families Using Model-Driven Techniques	Kulkarni, V., Barat, S.
	[25]	2013	Modeling and validation of business process families	Groner, G., Boskovic, M., Gasevic, D.
	[38]	2014	vrBPMN* and FM: An Approach to Model Business Process Line	Landre, G., Palma, E., Paiva, D., Nakagawa, E., Cagnin, M.
	[76]	2016	Variability patterns for business processes in BPMN	Yousfi, A., Saidi, R., Dey, A.
	[24]	2017	Ontology-Based Framework for Quality in Configurable Process Models	Faquih, E., Fredj, M.
PPM	[52]	2011	Partial process models to manage business process variants	Pascalau, E., Awad, A., Sakr, S., Weske, M.
	[56]	2015	Automated refinement of business processes through model transformations specifying business rules	Popp, R., Kaindl, H.
	[14]	2017	An approach implementing template-based process development on BPMN	Cui, X.
Provop	[63]	2010	Configuring the variability of business process models using non-functional requirements	Santos, E., Pimentel, J., Castro, J., Sanchez, J., Pastor, O.
	[40]	2011	Managing Variability in Business Processes: An Aspect-Oriented Approach	Machado, I., Bonifacio, R., Alves, V., Turnes, L., Machado, G.
vBPMN	[21]	2011	vBPMN: Event-Aware Workflow Variants by Weaving BPMN2 and Business Rules	Dohring, M., Zimmermann, B.

References

1. Altalhi, A.H., Luna, J.M., Vallejo, M.A., Ventura, S.: Evaluation and comparison of open source software suites for data mining and knowledge discovery. Wiley **7**(3), e1204 (2017). https://doi.org/10.1002/widm.1204
2. Andaloussi, A.A., Burattin, A., Slaats, T., Petersen, A.C.M., Hildebrandt, T.T., Weber, B.: Exploring the understandability of a hybrid process design artifact based on DCR graphs. In: Enterprise, Business-Process and Information Systems Modeling—20th International Conference, BPMDS 2019, 24th International Conference, EMMSAD 2019, Held at CAiSE 2019, Rome, Italy, June 3–4, 2019, Proceedings, LNBIP, vol. 352, pp. 69–84. Springer (2019). https://doi.org/10.1007/978-3-030-20618-5_5
3. Assy, N., Chan, N.N., Gaaloul, W., Defude, B.: Deriving configurable fragments for process design. Int. J. Bus. Process. Integr. Manag. **7**(1), 2–21 (2014). https://doi.org/10.1504/IJBPIM.2014.060602
4. Ayora, C., Torres, V., Pelechano, V., Alférez, G.H.: Applying CVL to business process variability management. In: Proceedings of the VARiability for You Workshop, VARY, pp. 26–31. ACM (2012). https://doi.org/10.1145/2425415
5. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems. Inf. Softw. Technol. **57**, 248–276 (2015). https://doi.org/10.1016/j.infsof.2014.05.009
6. Aysolmaz, B., Yaldiz, A., Reijers, H.A.: A process variant modeling method comparison: experience report. In: Enterprise, Business-Process and Information Systems Modeling—17th International Conference, BPMDS 2016, 21st International Conference, EMMSAD 2016, Proceedings, LNBIP, vol. 248, pp. 285–300. Springer (2016). https://doi.org/10.1007/978-3-319-39429-9_18
7. Azouzi, S., Ghannouchi, S.A., Brahmi, Z.: Software product line to express variability in e-learning process. In: Information Systems—14th European, Mediterranean, and Middle Eastern Conference, EMCIS, Proceedings, LNBIP, vol. 299, pp. 173–185. Springer (2017). https://doi.org/10.1007/978-3-319-65930-5_14
8. Calegari, D., Delgado, A., Peña, L.: Automated generation of variants in business process families based on the common variability language (CVL). In: XLV Latin American Computing Conference, CLEI, pp. 1–10. IEEE (2019). https://doi.org/10.1109/CLEI47609.2019.235116
9. Calegari, D., Delgado, A., Peña, L.: Model-driven support for business process families with the common variability language (CVL). CLEI Electron. J. **23**(1), 1–24 (2020). https://doi.org/10.19153/cleiej.23.1.3
10. Chang, J.: Business Process Management Systems: Strategy and Implementation. CRC Press, Boca Raton (2016)
11. COAL Group: Business Process Family Manager (BPFM). https://gitlab.fing.edu.uy/open-coal/bpfm
12. Cognini, R., Corradini, F., Polini, A., Re, B.: Business process feature model: an approach to deal with variability of business processes. In: Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, pp. 171–194. Springer (2016). https://doi.org/10.1007/978-3-319-39417-6_8
13. Collier, K., Carey, B., Sautter, D., Marjaniemi, C.: A methodology for evaluating and selecting data mining software. In: 2014 47th Hawaii International Conference on System Sciences, vol. 6, p. 6009. IEEE Computer Society (1999). https://doi.org/10.1109/HICSS.1999.772607
14. Cui, X.: An approach implementing template-based process development on BPMN. In: 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS, pp. 239–244. IEEE Computer Society (2017). https://doi.org/10.1109/ICIS.2017.7960000
15. Delgado, A., Calegari, D., García, F., Weber, B.: Model-driven management of BPMN-based business process families (supplementary material) (2021). https://doi.org/10.5281/zenodo.5768180
16. Delgado, A., Calegari, D., García, F.: Modeling of software process families with automated generation of variants (S). In: The 30th International Conference on Software Engineering and Knowledge Engineering, SEKE,, pp. 330–329. KSI Research Inc. and Knowledge Systems Institute Graduate School (2018). https://doi.org/10.18293/SEKE2018-019
17. Delgado, A., Calegari, D., Milanese, P., Falcon, R., Garcia, E.: A systematic approach for evaluating BPM systems: Case studies on open source and proprietary tools. In: Open Source Systems: Adoption and Impact—11th IFIP WG 2.13 International Conference, OSS 2015, Proceedings, IFIP, vol. 451, pp. 81–90. Springer (2015). https://doi.org/10.1007/978-3-319-17837-0_8
18. Delgado, A., Calegari, D.: BPMN 2.0 based modeling and customization of variants in business process families. In: XLIII Latin American Computer Conference, CLEI, pp. 1–9. IEEE (2017). https://doi.org/10.1109/CLEI.2017.8226450
19. Delgado, A., Calegari, D.: Changing the focus of an organization: from information systems to process aware information systems. In: Enterprise, Business-Process and Information Systems Modeling—16th International Conference, BPMDS, Proceedings, LNBIP, vol. 214, pp. 53–67. Springer (2015). https://doi.org/10.1007/978-3-319-19237-6
20. Delgado, A., Calegari, D.: Systematic evaluation of business process management systems: a comprehensive approach. CLEI Electron. J. **21**, 1–19 (2018). https://doi.org/10.19153/cleiej.21.2.7
21. Döhring, M., Zimmermann, B.: vbpmn: Event-aware workflow variants by weaving BPMN2 and business rules. In: Enterprise, Business-Process and Information Systems Modeling—12th International Conference, BPMDS 2011, Proceedings, LNBIP, vol. 81, pp. 332–341. Springer (2011). https://doi.org/10.1007/978-3-642-21759-3_24
22. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M. (eds.): Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005). https://doi.org/10.1002/0471741442
23. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of BPM, 2nd edn. Springer, Berlin (2018)
24. Faquih, L.E., Fredj, M.: Ontology-based framework for quality in configurable process models. J. Electron. Commer. Organ. **15**(2), 48–60 (2017). https://doi.org/10.4018/JECO.2017040104
25. Gröner, G., Boskovic, M., Parreiras, F.S., Gasevic, D.: Modeling and validation of business process families. Inf. Syst. **38**(5), 709–726 (2013). https://doi.org/10.1016/j.is.2012.11.010
26. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process life cycle. In: Proceedings of the Tenth International Conference on Enterprise Information Systems, ICEIS, pp. 154–161 (2008). https://doi.org/10.1007/978-3-642-45100-3_11
27. Haugen, Ø., Øgård, O.: BVR—better variability results. In: System Analysis and Modeling: Models and Reusability—8th International Conference, SAM, Proceedings, LNCS, vol. 8769, pp. 1–15. Springer (2014). https://doi.org/10.1007/978-3-319-11743-0

28. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
29. ISO/IEC: Iso/iec 25010:2011 systems and software engineering—systems and software quality requirements and evaluation (square)—system and software quality models. Technical reports, ISO/IEC International Organization for Standardization/International Electrotechnical Comission (2011). https://www.iso.org/standard/35733.html
30. ISO/IEC: Iso/iec 9126:1991 software engineering—product quality. Technical report, ISO/IEC International Organization for Standardization/International Electrotechnical Comission (1991). https://www.iso.org/standard/16722.html
31. ISO/IEC: Iso/iec 9126:1991 software engineering—product quality—part 1: model quality. Technical report, ISO/IEC International Organization for Standardization/International Electrotechnical Comission (2001). https://www.iso.org/standard/22749.html
32. Jiménez-Ramírez, A., Barba, I., Weber, B., Valle, C.D.: Automatic generation of questionnaires for supporting users during the execution of declarative business process models. In: Business Information Systems—17th International Conference, BIS. Proceedings, LNBIP, vol. 176, pp. 146–158. Springer (2014). https://doi.org/10.1007/978-3-319-06695-0_13
33. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1–2), 31–39 (2008). https://doi.org/10.1016/j.scico.2007.08.002
34. Kent, S.: Model-Driven Engineering, LNCS, vol. 2335. Springer, Berlin (2002)
35. Kitchenham, B.: Procedures for performing systematic reviews. Technical Report tr/se-0401, Keele University (2004)
36. Krosnick, J.A.: Survey research. Annu. Rev. Psychol. **50**, 537–567 (1999)
37. Kulkarni, V., Barat, S.: Business process families using model-driven techniques. Int. J. Bus. Process. Integr. Manag. **5**(3), 204–217 (2011). https://doi.org/10.1504/IJBPIM.2011.042525
38. Landre, G., Palma, E., Paiva, D.M.B., Nakagawa, E.Y., Cagnin, M.I.: vrbpmn* and FM: an approach to model business process line. In: Business Process Management Workshops—BPM 2014 International Workshops, Revised Papers, LNBIP, vol. 202, pp. 130–141. Springer (2014). https://doi.org/10.1007/978-3-319-15895-2_12
39. Likert, R.: A technique for the measurement of attitudes. Arch. Psychol. **22**, 5–55 (1932)
40. Machado, I., Bonifácio, R., Alves, V., Turnes, L., Machado, G.: Managing variability in business processes: an aspect-oriented approach. In: Proceedings of the 2011 International Workshop on Early Aspects, EA '11, pp. 25–30. ACM (2011). https://doi.org/10.1145/1960502.1960508
41. Martínez-Ruiz, T., García, F., Piattini, M.: Towards a SPEM v2.0 extension to define process lines variability mechanisms, pp. 115–130. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-70561-1_9
42. Mechrez, I., Reinhartz-Berger, I.: Modeling design-time variability in business processes: existing support and deficiencies. In: Enterprise, Business-Process and Information Systems Modeling—15th International Conference, BPMDS, 19th International Conference, EMMSAD. Proceedings, LNBIP, vol. 175, pp. 378–392. Springer (2014). https://doi.org/10.1007/978-3-662-43745-2_26
43. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Inf. Softw. Technol. **52**(2), 127–136 (2010). https://doi.org/10.1016/j.infsof.2009.08.004
44. Milani, F., Dumas, M., Ahmed, N., Matulevicius, R.: Modelling families of business process variants: a decomposition driven method. Inf. Syst. **56**, 55–72 (2016). https://doi.org/10.1016/j.is.2015.09.003
45. Nguyen, T., Colman, A.W., Han, J.: Modeling and managing variability in process-based service compositions. In: Service-Oriented Computing: 9th International Conference, ICSOC, Proceedings, LNCS, vol. 7084, pp. 404–420. Springer (2011). https://doi.org/10.1007/978-3-642-25535-9_27
46. Ognjanovic, I., Mohabbati, B., Gasevic, D., Bagheri, E., Boskovic, M.: A metaheuristic approach for the configuration of business process families. In: 2012 IEEE Ninth International Conference on Services Computing, pp. 25–32. IEEE (2012). https://doi.org/10.1109/SCC.2012.6
47. OMG: Business Process Model and notation (BPMN) v2.0. Technical Reports, Object Management Group (OMG) (2013)
48. OMG: Common Variability Language (CVL). Revised Submission. Technical Reports, Object Management Group (OMG) (2012)
49. OMG: Meta Object Facility (MOF) v2.5.1. Technical Reports, Object Management Group (OMG) (2016)
50. OMG: Software and Systems Process Engineering Metamodel (SPEM) v2.0. Technical Reports, Object Management Group (OMG) (2008)
51. Park, J., Yeom, K.: A modeling approach for business processes based on variability. In: 2011 Ninth International Conference on Software Engineering Research, Management and Applications, pp. 211–218 (2011). https://doi.org/10.1109/SERA.2011.19
52. Pascalau, E., Awad, A., Sakr, S., Weske, M.: Partial process models to manage business process variants. IJBPIM **5**(3), 240–256 (2011)
53. Peffers, K., Tuunanen, T., Gengler, C.E., Rossi, M., Hui, W., Virtanen, V., Bragge, J.: The design science research process: a model for producing and presenting information systems research. In: Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006), pp. 83–106 (2006)
54. Peña, L., Fernandez, D., López, G.: BVR+BPMN 2.0 Editor. https://gitlab.fing.edu.uy/open-coal/bpmncvl (2018)
55. Pillat, R.M., Oliveira, T.C., Alencar, P.S.C., Cowan, D.D.: Bpmnt: a BPMN extension for specifying software process tailoring. Inf. Softw. Technol. **57**, 95–115 (2015). https://doi.org/10.1016/j.infsof.2014.09.004
56. Popp, R., Kaindl, H.: Automated refinement of business processes through model transformations specifying business rules. In: 9th IEEE International Conference on Research Challenges in Information Science, RCIS, pp. 327–333. IEEE (2015). https://doi.org/10.1109/RCIS.2015.7128893
57. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer (2012). https://doi.org/10.1007/978-3-642-30409-5
58. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Organisational reference models: supporting an adequate design of local business processes. Int. J. Bus. Process. Integr. Manag. **4**(2), 134–149 (2009). https://doi.org/10.1504/IJBPIM.2009.027781
59. Rosa, M.L., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based variability modeling for system configuration. Softw. Syst. Model. **8**(2), 251–274 (2009). https://doi.org/10.1007/s10270-008-0090-3
60. Rosa, M.L., van der Aalst, W.M.P., Dumas, M., Milani, F.: Business process variability modeling: a survey. ACM Comput. Surv. **50**(1), 2:1-2:45 (2017). https://doi.org/10.1145/3041957
61. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1), 1–23 (2007)
62. Rychkova, I., Nurcan, S.: Towards adaptability and control for knowledge-intensive business processes: Declarative configurable process specifications. In: 44th Hawaii International Conference on Systems Science (HICSS-44), Proceedings, pp. 1–10. IEEE Computer Society (2011). https://doi.org/10.1109/HICSS.2011.452
63. Santos, E., Pimentel, J., Castro, J., Sánchez, J., Pastor, O.: Configuring the variability of business process models using non-functional requirements. In: Enterprise, Business-Process and Information Systems Modeling - 11th International Workshop, BPMDS 2010,

Proceedings, LNBIP, vol. 50, pp. 274–286. Springer (2010). https://doi.org/10.1007/978-3-642-13051-9_23
64. Schnieders, A., Puhlmann, F.: Variability Modeling and Product Derivation in E-Business Process Families, pp. 63–74. Springer (2007). https://doi.org/10.1007/1-4020-5634-6_6
65. Sharma, D.K., Hitesh, Rao, V.: Individualization of process model from configurable process model constructed in c-bpmn. In: International Conference on Computing, Communication Automation, pp. 750–754 (2015). https://doi.org/10.1109/CCAA.2015.7148510
66. ter Beek, M.H., Lluch-Lafuente, A., Petrocchi, M.: Combining declarative and procedural views in the specification and analysis of product families. In: 17th International Software Product Line Conference Co-located Workshops, pp. 10–17. ACM (2013). https://doi.org/10.1145/2499777.2500722
67. Terenciani, M., Paiva, D.M.B., Landre, G., Cagnin, M.I.: Bpmn*—A notation for representation of variability in business process towards supporting business process line modeling. In: The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE, pp. 227–230. KSI Research Inc. and Knowledge Systems Institute Graduate School (2015). https://doi.org/10.18293/SEKE2015-55
68. Valenca, G., Alves, C., Alves, V., Niu, N.: A systematic mapping study on business process variability. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) **5**(1), 1–21 (2013). https://doi.org/10.5121/ijcsit.2013.5101
69. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business process management: a survey. In: Business Process Management, International Conference, BPM, Proceedings, LNCS, vol. 2678, pp. 1–12. Springer (2003). https://doi.org/10.1007/3-540-44895-0
70. Weidmann, M., Koetter, F., Kintz, M., Schleicher, D., Mietzner, R.: Adaptive business process modeling in the internet of services (abis). In: ICIW 2011 (2011)
71. Weske, M.: Business Process Management: Concepts, Languages, Architectures, 3rd edn. Springer, Berlin (2019). https://doi.org/10.1007/978-3-662-59432-2
72. Wieringa, R.J.: Design Science Methodology for Inf. Springer, Systems and Software Engineering (2014)
73. Wieringa, R.: Empirical research methods for technology validation: scaling up to practice. J. Syst. Softw. **95**, 19–31 (2014). https://doi.org/10.1016/j.jss.2013.11.1097
74. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B.: Experimentation in Software Engineering. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-29044-2
75. Yin, R.K.: Case study research. In: Design and Methods, 6th edn. Sage Publications (2017)
76. Yousfi, A., Saidi, R., Dey, A.K.: Variability patterns for business processes in BPMN. Inf. Syst. E Bus. Manag. **14**(3), 443–467 (2016). https://doi.org/10.1007/s10257-015-0290-7
77. Zhang, H., Han, W., Ouyang, C.: Extending BPMN for configurable process modeling. In: Proceedings of the 21st ISPE Inc. International Conference on Concurrent Engineering, Advances in Transdisciplinary Engineering, vol. 1, pp. 317–330. IOS Press (2014)
78. Zhang, H., Han, W., Ouyang, C.: Extending BPMN for configurable process modeling. In: Proceedings of the 21st ISPE Inc. International Conference on Concurrent Engineering, Advances in Transdisciplinary Engineering, vol. 1, pp. 317–330. IOS Press (2014). https://doi.org/10.3233/978-1-61499-440-4-317

Andrea Delgado is an associate professor at the Instituto de Computación, Facultad de Ingeniería, Universidad de la República (UdelaR), Uruguay. She received her Ph.D. in Computer Science from the University of Castilla-La Mancha (UCLM), Spain and from Universidad de la República (PEDECIBA), Uruguay (2012). She is a member of the COAL research group and her main research interests are business process management and technologies, process mining, service-oriented computing and model driven development.

Daniel Calegari is an associate professor at the Instituto de Computación, Facultad de Ingeniería, Universidad de la República (UdelaR), Uruguay. He received his Ph.D. in Computer Science from Universidad de la República (PEDECIBA), Uruguay (2014). He is a member of the COAL research group and his current research interests include model-driven engineering and its application in the context of business process management.

Félix García is full Professor in the Department of Information Technologies and Systems at the University of Castilla-La Mancha (UCLM). He received his M.Sc. (2001) and Ph.D. (2004) degrees in Computer Science from the UCLM. He is a member of the Alarcos Research Group and his research interests include business and software process management, software measurement, agile methods and software sustainability. He holds the following professional certifications: PMP, CISA and Scrum Master.

Barbara Weber is Professor for Software Systems Programming and Development at the University of St. Gallen, Switzerland. She is Chair for Software Systems Programming and Development and Dean of the School of Computer Science. Barbara's research interests include human and cognitive aspects in software and process engineering. Barbara currently has an h-index of 48 and has published more than 160 refereed papers, for example, in Nature Scientific Reports, Information and Software Technology, Journal of Systems and Software, Information Systems, Data and Knowledge Engineering, and Journal of Management Information Systems.