Business process variability modeling

La Rosa, M.; van der Aalst, W.M.P.; Dumas, M.; Milani, F.P.

Published in:
ACM Computing Surveys

DOI:
10.1145/3041957

Published: 01/03/2017

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Business Process Variability Modeling: A Survey

MARCELLO LA ROSA, Queensland University of Technology, Australia
WIL M. P. VAN DER AALST, Eindhoven University of Technology, The Netherlands
MARLON DUMAS, University of Tartu, Estonia and Queensland University of Technology, Australia
FREDRIK P. MILANI, University of Tartu, Estonia

It is common for organizations to maintain multiple variants of a given business process, such as multiple sales processes for different products or multiple bookkeeping processes for different countries. Conventional business process modeling languages do not explicitly support the representation of such families of process variants. This gap triggered significant research efforts over the past decade, leading to an array of approaches to business process variability modeling. In general, each of these approaches extends a conventional process modeling language with constructs to capture customizable process models. A customizable process model represents a family of process variants in a way that a model of each variant can be derived by adding or deleting fragments according to customization options or according to a domain model. This survey draws up a systematic inventory of approaches to customizable process modeling and provides a comparative evaluation with the aim of identifying common and differentiating modeling features, providing criteria for selecting among multiple approaches, and identifying gaps in the state of the art. The survey puts into evidence an abundance of customizable process-modeling languages, which contrasts with a relative scarcity of available tool support and empirical comparative evaluations.

Categories and Subject Descriptors: H.4.1 [**Office Automation**]: Workflow Management; A.1 [**Introductory and Survey**]

General Terms: Design, Management, Standardization

Additional Key Words and Phrases: Variability modeling, process model, customizable process model

ACM Reference Format:
Marcello La Rosa, Wil M. P. van der Aalst, Marlon Dumas, and Fredrik P. Milani. 2017. Business process variability modeling: A survey. ACM Comput. Surv. 50, 1, Article 2 (March 2017), 45 pages.
DOI: http://dx.doi.org/10.1145/3041957

1. INTRODUCTION

The coexistence of multiple variants of the same business process is a widespread phenomenon in contemporary organizations. As a concrete example, The Netherlands has around 430 municipalities, which in principle execute the same or a very similar set of processes. All municipalities have processes related to building permits, such as the process for handling applications for permits and the process for handling objections against such permits. Due to demographics and political choices, though, each municipality executes its processes differently. Variations are justified by different priorities

This research is partly funded by the Australian Research Council (grant DP150103356) and the Estonian Research Council (grant IUT20-55).
Authors' addresses: M. La Rosa, Queensland University of Technology, GPO Box 2434, Brisbane, QLD 4001, Australia; email: m.larosa@qut.edu.au; W. M. P. van der Aalst, Eindhoven University of Technology, PO Box 513 NL-5600 MB Eindhoven, The Netherlands; email: w.m.p.v.d.aalst@tue.nl; M. Dumas and F. P. Milani, University of Tartu, J. Liivi 2, Tartu 50409, Estonia; emails: {marlon.dumas, milani}@ut.ee.

DOI: http://dx.doi.org/10.1145/3041957

and customs, often referred to as the "Couleur Locale." At present, these differences have come to be accepted and there is no willingness to flatten them out. Still, capturing multiple municipality processes in a consolidated manner is necessary in order to develop information systems that can support multiple or all municipalities at once.

Similarly, *Suncorp Group* – the largest insurance group in Australia – offers a range of insurance products, including home, motor, commercial, and liability insurance. Each product exists for different brands of the group (e.g., Suncorp, AAMI, APIA, GIO, and Vero). As a result, there are more than 30 variants of the process for handling an insurance claim at Suncorp Group. There is a case for modeling and maintaining these variants in a consolidated manner not only to avoid redundancy but also so that improvements and automation efforts made on one variant can benefit other variants.

The application of conventional business process modeling approaches [Mili et al. 2010] to families of process variants requires one of two paths to be chosen. Either each variant is modeled separately, resulting in duplication as the variants have much in common, or multiple variants are modeled together, leading to highly complex consolidated models, which hampers the analysis and maintenance of individual variants.

Motivated by this observation, a number of approaches to model families of business-process variants have emerged. A common trait of these approaches is that they support the representation of a family of business-process variants via a single model, from which each variant can be derived via certain model transformations. We use the term *customizable process model* to refer to such a consolidated model of process variants and the term *variation point* to indicate an element of the customizable process model that can be customized via transformations.

A wide array of approaches to customizable process modeling have been proposed in recent years without it being generally clear what trade-offs they strike relative to each other and how potential users should select an approach for a given purpose. In this setting, this survey draws up a systematic inventory of approaches to customizable process modeling, identifies and classifies major approaches in the field and provides a comparative evaluation aimed at answering the following questions:

—**RQ1.** What are the commonalities and distinctive features of approaches to customizable process modeling?
—**RQ2.** What criteria can be used to select between different approaches?
—**RQ3.** What general limitations or research gaps exist in the literature on customizable process modeling that may require further work?

The rest of the article is organized as follows. Section 2 delimits the scope of the survey. Section 3 defines and justifies the criteria used to analyze approaches in the field. Section 4 presents a working example. Sections 5 to 9 illustrate and analyze major approaches in the field identified from a systematic literature review. Section 10 provides a synthesis of the commonalities and differences between the surveyed approaches and answers the research questions just framed. Section 11 positions this survey with respect to related work. Finally, Section 12 exposes common limitations, leading to an outline of future research directions.

Six appendices complement the survey. Appendix A describes the literature search procedure and summarizes the results. Appendix B surveys secondary approaches identified during the search process. Appendices C and D discuss techniques employed by the surveyed approaches to provide decision support during process customization and to ensure correctness of the customized process models. Finally, Appendix E provides a list of all relevant terms and their definitions used in this article, while Appendix F shows a mapping between the different process-modeling languages used to exemplify the approaches surveyed.

2. SCOPE

Customizable process models capture a family of process-model variants in a way that the individual variants can be derived via transformations, for example, adding or deleting fragments. Accordingly, a customizable process model encapsulates customization decisions between process variants that need to be made either at design time or runtime. *Design-time customization* decisions lead to a customized process model that is intended to be executed in a particular organizational setting. Thus, these decisions affect all instances of the customized process executed in this setting. The time frame associated with these decisions may be long (e.g., months or years). In contrast, *runtime customization* decisions are punctual and affect only one or a few process instances. Such decisions may be visualized on top of a process model, but they are not intended to modify the executed process model itself beyond its effects on the process instance(s) in which the decision is applied.

Processes in which customization decisions are made at runtime are called *flexible processes* [Reichert and Weber 2012]. The challenges associated with managing such processes have been widely studied in the literature [Rinderle et al. 2004; Weber et al. 2008]. The present survey focuses on *design-time process variability management* as opposed to runtime flexible process management. In other words, the focus is on capturing a family of processes via a single-process model that is customized at design-time. Approaches to runtime flexible process management are generally not concerned with maintaining multiple process models that together form a family of processes. Instead, these approaches rely on a unitary process model. In some approaches, this unitary process model is seen as an indicative roadmap with respect to which individual process instances may deviate [Reichert and Dadam 1998], while in other approaches – for example, Declare [Pesic et al. 2007] or Pockets of Flexibility [Sadiq et al. 2001, 2005] – the process model is left underspecified and individual process instances refine this underspecified model rather than deviating from it. In both cases, there is still a single-process model that serves as a reference during process execution.

Customization decisions may result in the removal or addition of behavior to a customizable process model. In this respect, we distinguish two approaches to variability management: *by restriction* and *by extension*.

Variability by restriction starts with a customizable process model that contains all behavior of all process variants. Customization is achieved by restricting the behavior of the customizable process model. For example, activities may be skipped or blocked during customization. In this setting, one can think of the customizable process model as the union or Least Common Multiple (LCM) of all process variants. Customizable process models of this type are sometimes called *configurable process models*.

Variability by extension takes the opposite starting point. The customizable process model does not contain all possible behavior; instead, it represents the most common behavior or the behavior that is shared by most process variants. At customization time, the model's behavior needs to be extended to serve a particular situation. For example, one may need to insert new activities in order to create a dedicated variant. In this setting, one can think of a customizable process model as the intersection or Greatest Common Denominator (GCD) of all process variants under consideration.

This survey covers *both* variability by restriction and by extension. In fact, as discussed later, it is possible for one same approach to combine both types of variability.

Customizable process models ought to be distinguished from so-called *reference process models* [Fettke and Loos 2003; Rosemann 2003]. Some vendors and consultancy firms provide reference process models that are intended to capture common knowledge or best practices in a given field (e.g., in supply chain management or IT service delivery). While a reference process model can be very useful in its own way, it should

be understood that it is in essence a concrete process model intended to be used as an example. Reference process models do not support customization in a structured manner. In this survey, we focus on approaches that provide support for customization rather than serving only as reference.

Having discussed the scope of the survey, we next define a set of criteria to characterize the approaches in the field.

3. EVALUATION CRITERIA

To derive criteria for assessing approaches to business-process variability modeling, we analyzed the solution space using the six "W questions" (Who, What, Where, When, Why, and How). We determined that the "who" and "why" questions do not allow us to distinguish between approaches in the field since all the approaches identified in the search (see Appendix A) have the same aim, that is, to support process modelers ("who") in the definition of customizable process models and in the customization thereof ("why"). Similarly, the "where" question is not relevant as there is no spatial dimension that distinguishes approaches in the field. The "when" question ("when does customization occur?") has been discussed in the previous section (design time vs. runtime) and a choice was made to focus on design time, given that runtime customization has been studied as a separate topic in the literature (see process flexibility). This leaves us with the "what" and "how" questions, which we refine into: "What is captured in the customizable model?" and "How are customized models derived from customizable ones?"

To answer the "what" question, we reuse a classification of elements of a process model spelled out in previous surveys, in which the elements of a process model are divided into those concerned with the *control-flow perspective* (the flow of control between activities), the *resource perspective* (organizational aspects), and the *object perspective* (physical and data objects manipulated in the process) [Georgakopoulos et al. 1995; Mili et al. 2010]. Accordingly, we characterize process variability modeling approaches depending on their support for each of these three perspectives.

Another classification of process models identified in previous work is based on their purpose [Georgakopoulos et al. 1995]. Along this direction, we distinguish between *conceptual* process models, which are intended for communication and analysis, and *executable* ones, which are intended for deployment in an execution engine.

Moving to the "how" question, we note that customized process models are derived from customizable models by applying transformations based on decisions made by a user. Thus, customization involves *decisions* and *transformations*.

The transformations applied during customization can be classified into those that restrict the process behavior captured by the customizable process model, for example, by removing an element (customization by *restriction*), and those that extend the process behavior, for example, by adding an element (customization by *extension*), as discussed in the previous section. Meanwhile, customization decisions can be expressed in terms of concepts that refer to the domain of discourse (abstract level) or concepts related to the process model itself (concrete level). Thus, approaches can be assessed depending on whether their customization decisions support *abstraction* to the domain level or not. Putting aside the concepts used to express decisions, some approaches guide the user step by step when making these decisions (i.e., by presenting the decisions in a certain order) and prevent inconsistent or irrelevant decisions to be made, while other approaches leave it up to the user to decide what decisions to perform and in what order. Accordingly, we can assess approaches depending on the *guidance* that they provide during customization.

Transformations applied to derive a customized process model may in some cases lead to syntactically incorrect models, whether structurally or behaviorally incorrect.

Some approaches guarantee that the customized process models are correct, but others do not. Accordingly, we can characterize an approach depending on whether or not it guarantees *structural* and/or *behavioral correctness* of the customized process model.

With respect to the "how" question, we considered alternative criteria. Since customizable process models generally extend a host modeling language, the latter could be used as a classification criterion. We did not retain this criterion because we observed that it is not a fundamental characteristic of an approach. An approach that has been designed for EPCs can be adapted to BPMN and vice-versa. We also considered the specific abstraction mechanism as a classification criterion. In this respect, approaches may differ in terms of the mechanism employed to link the elements of the process model to elements of the domain of discourse. Some approaches rely on simple "annotations" attached to model elements referring to implicitly defined elements of the domain of discourse, while others may opt for a more explicit linkage, in which elements of the process model are linked to concepts in an explicit domain model (or vice-versa). The latter could be further subdivided depending on the approach employed to represent the domain model (e.g., feature model vs. questionnaire model). We opted, however, to simply classify approaches depending on whether they support abstraction or not, because we found that the choice of the domain modeling approach and the choice of the mechanism for linking the domain model to the process model are very approach-specific. In Section 5, we discuss these design choices for each approach separately.

Having identified assessment criteria based on the "what" and "how" questions, we moved to the "meta" level, by considering the design of the approach itself. Research papers that propose customizable process modeling approaches rely, implicitly or explicitly, on a design science method [Hevner et al. 2004]. According to design science principles, the conceived artifacts should be specified, implemented where applicable, and validated to determine if they fulfill the intended requirements. Artifacts in the field under study can be specified informally or formally. They may or may not be implemented as a prototype and they may or may not be validated in order to assess their applicability and qualities. Accordingly, we identify three extra-functional requirements: *formalization*, *implementation*, and *validation*. The criteria resulting from this analysis are explained here.

1 **Scope.** This category refers to the "what" question discussed earlier. It is broken down into two subcategories: *Process Perspective* and *Process Type*.
  1.1 **Process Perspective.** This category refers to the supported process modeling perspectives.
    1.1.1 **Control flow.** Ability of a customizable model to capture variability along the control-flow perspective, that is, activities and routing elements such as gateways can become variation points (e.g., capturing that a credit history check is not required in some of the variants of a loan origination process). A language is considered to only partially fulfill this criterion if routing elements or activities are not customizable or if such elements are customizable but the corresponding customization options are not graphically represented.
    1.1.2 **Resources.** Ability of a customizable model to capture variability in the involved human and nonhuman resources, that is, resources can become variation points (e.g., capturing that a risk assessor is not involved in some of the variants of a claims handling process). A language partially fulfills this criterion if resources are customizable but the options are not graphically represented.

1.1.3 **Objects.** Ability of a customizable model to capture variability in the physical and data objects produced and consumed by a process, that is, objects can be become variation points (e.g., capturing that an invoice is not required in some of the variants of an order-to-cash process). A language partially fulfills this criterion if objects are customizable but their customization options are not graphically represented.

1.2 **Process Type.** This category refers to the purpose of the process models.

1.2.1 **Conceptual.** An approach meets this criterion if it is designed to support conceptual process models only, that is, process models that are not meant to be executed on top of a Business Process Management System (BPMS).

1.2.2 **Executable.** An approach is considered to fulfill this criterion if the customization prevents or resolves inconsistencies in the associations between activities and data objects, thus making the customized models executable on top of a concrete BPMS. If the customized models can be executed on a BPMS but these inconsistencies are not addressed, the approach is considered to only partially fulfill the criterion. Similarly, the criterion is partially fulfilled if there is no BPMS that can support the execution of the customized models even if inconsistencies are prevented or resolved by the approach.

2 **Customization Type.** Do the supported transformations restrict/extend the process behavior?

2.1 **Restriction.** An approach matches this criterion if a process model is customized by restricting its behavior.

2.2 **Extension.** An approach matches this criterion if a process model is customized by extending its behavior.

An approach could support both criteria in principle, that is, there could be transformations to restrict some parts and extend others.

3 **Supporting Techniques.** This category refers to techniques to support the customization of process models. The two subcategories are based on common functionality frequently reported in the literature: decision support for the selection of suitable customization options and ensuring the correctness of the customized model.

3.1 **Decision Support.** How are users supported in their customization decisions?

3.1.1 **Abstraction.** An approach supports process model abstraction if users can customize a model without directly referring to its model elements but instead to properties of the application domain (e.g., customizing an order-to-cash process model based on the available sales channels rather than based on the activities and gateways that are customizable in the model).

3.1.2 **Guidance.** This criterion is met if there is support to (i) guide users when making customization decisions, for example, in the form of recommendations for selecting one option or another; and (ii) prevent users from making inconsistent or irrelevant customization decisions from a domain viewpoint. Approaches that only provide support for one of these two aspects partially fulfill this criterion.

3.2 **Correctness Support.** Is the syntactical correctness of the customized models guaranteed? Syntactical correctness is divided into correctness of the model structure and correctness of the model behavior.

3.2.1 **Structural correctness.** Ability to guarantee the correct structure of the customized models, for example, by avoiding disconnected nodes.

3.2.2 **Behavioral correctness.** Ability to guarantee the correct behavior of the customized models, for example, by avoiding behavioral anomalies such

as deadlocks and livelocks when the model is instantiated. In other words, the model must be *sound* [van der Aalst et al. 2011], that is, it should always be possible to complete any process instance properly.

4 **Extra-Functional.** Criteria related to the design of the approach itself.

4.1 **Formalization.** Some approaches present only ideas and do not provide concrete algorithms or definitions. Therefore, we include a criterion indicating whether the approach has been described rigorously in terms of mathematical notations. In order to fulfill this criterion, the approach has to be formally defined, including algorithms used during customization. If such algorithms are missing, the approach partially fulfills the criterion.

4.2 **Implementation.** Approaches may only exist on paper. However, the usability and maturity of an approach heavily depends on tool support to design and customize customizable process models. If the approach is fully implemented (including algorithms used during customization), then this criterion is fulfilled. Approaches with partial implementations, for example, offering only design or customization support, partially fulfill this criterion.

4.3 **Validation.** The applicability of some approaches has been validated using real-life process variants and through discussions with domain experts, but this does not necessarily apply to all approaches. An approach fulfills this criterion if it has been tested on models not created by the authors and the results verified by domain experts. If one of these two aspects is lacking (e.g., an approach that has been validated without the involvement of domain experts), then this criterion is only partially fulfilled.

The next section introduces an example of a family of process variants that is used later to illustrate the approaches retrieved by the search described in Appendix A.

4. ILLUSTRATIVE SCENARIO

The example process family described in this section is the result of a case study in picture postproduction that we conducted with domain experts from the Australian Film, Television and Radio School (AFTRS) in Sydney.[1]

In the film industry, picture postproduction (postproduction hereafter) is the process that starts after the shooting has been completed, and deals with the creative editing of the motion picture. Figure 1 shows several variants of the postproduction process. A process model is a directed graph consisting of nodes of type event, activity, and gateway and arcs (called sequence flows) linking these elements. Events are triggers to and signal the results of activities or of the entire process (e.g., a start event triggers the entire process, while an end event signals its completion). Activities capture work done in the process. Gateways are used to model alternative and parallel branching and merging and are divided into splits (with multiple outgoing flows and one incoming flow) and joins (with multiple incoming flows and one outgoing flow). Splits and joins have a logical type. They can be of type OR or XOR (for inclusive, resp., exclusive decision and merging) and AND (for parallelism and synchronization).

The example in Figure 1 is represented in the *Event-driven Process Chains (EPCs)* language [Davis and Brabander 2007]. There is a variety of languages in addition to EPCs to represent process models, for example, BPMN, UML Activity Diagrams, YAWL, and BPEL. While in this article we will illustrate process model examples using different languages, depending on the approach being reviewed, for uniformity we will always use the terminology described earlier, which is borrowed from the BPMN

[1]See www.aftrs.edu.au.

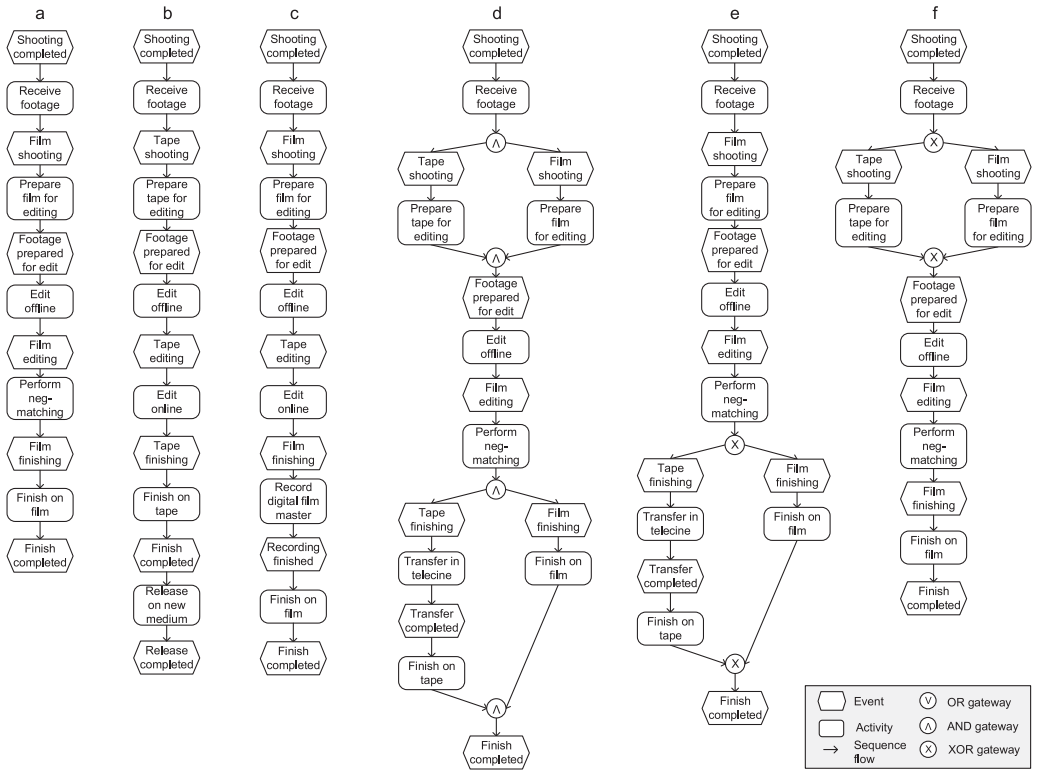

Fig. 1. Different variants of the picture postproduction process in the EPC language.

standard, and abstract from language-specific terms. Appendix F provides a mapping between the languages used to exemplify the approaches surveyed.

As depicted in Figure 1, postproduction starts with the receipt, from the shooting that needs to be prepared for editing. The footage can either be prepared on film (see for example, variant *a* of Figure 1, in which activity "Prepare film for editing" in performed), on tape (e.g., variant *b*, in which activity "Prepare film for editing" is performed) or on both media (variant *d*) depending on whether the motion picture was shot on a film roll and/or on a tape. Next, the medium is edited offline to achieve the first rough cut (thus, activity "Edit offline" exists in all variants). However, after this, online editing is carried out if the footage was shot on tape (variants *b* and *c*), while a negmatching is performed if the footage was shot on film (e.g., variant *a*). Online editing is a cheap editing procedure suited for low-budget movies typically shot on tape. Negmatching offers better-quality results but entails higher costs; thus, it is more suitable for high-budget productions typically shot on film. The choice between online editing and negmatching is an important postproduction decision: depending on drivers such as budget, creativity, and type of project, one option, the other, or both need to be taken. Thus, each variant in Figure 1 reflects a common practice in postproduction. For example, variant *a* is a typical low-budget practice (shooting and releasing on tape), whereas variant *d* illustrates a more expensive procedure (shooting and releasing on both tape and film).

The final step of postproduction is the finishing of the edited picture. This can be be done on film (see variant *a*), on tape (variant *b*) or on both media (variant *d*). The finishing may involve further activities based on the combination of editing type and

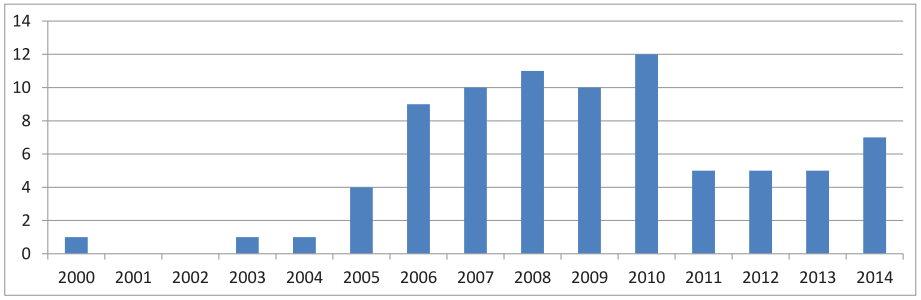

Fig. 2. Number of publications on process model variability management via customizable process models.

final medium. For example, if the editing was done online and the final version is on film, a digital film master is to be recorded from the edited tape (see variant *c*). Alternatively, if a negmatching was performed and the final version is on tape, the edited film is to be transferred onto a tape via a telecine machine (variants *d* and *e*).

The process may conclude with an optional release on a new medium (e.g., DVD or digital stream), which follows the finishing on tape or film (e.g., in variant *b*, the release on the new medium follows a tape finish).

5. OVERVIEW OF PROCESS MODEL CUSTOMIZATION APPROACHES

We conducted a literature search using the protocol described in Appendix A. This search resulted in 66 relevant publications. In many cases, multiple publications pertain to the same approach. Also, some approaches are subsumed by other approaches, that is, the concepts in one approach are contained in another. By grouping the publications accordingly, we found that the 66 publications cover 23 approaches, out of which 11 main approaches subsume the other 12 approaches.

The 66 publications covered by this survey are listed in a supplemental spreadsheet available at https://goo.gl/mmxZf3. For each approach, the table identifies a primary (earliest) publication describing the approach and, when available, additional publications describing further aspects of the same approach.

A histogram of papers per year of publication is shown in Figure 2. This histogram is based on the list of publications satisfying the inclusion and exclusion criteria defined in the search protocol (see Appendix A).[2] The histogram shows an increasing trend of publications on the topic starting in 2005 and reaching a peak in 2010.

We classified the 23 identified approaches by asking the following question for each approach: "How does the approach capture the relation between an element or set of elements of a customizable process model and a corresponding element or set of elements of each of the possible customized process models thereof?"

Answering this question for each approach led us to observe that, in some approaches, a node of the customizable model can be retained, removed, or its behavior can be restricted by selecting one of multiple possible customization options. This class of approaches is hereby called *node configuration*. In other approaches, an element (i.e., a node or a sequence flow) of the customizable process model is linked to a predicate over a domain model via an *annotation*. Customization then takes place by evaluating these predicates with respect to an instantiation of the domain model. We call this class of approaches *element annotation*. In a third class of approaches, a given activity in

[2]The histogram includes papers with less than 10 citations, even if this was an exclusion criterion, since its intent is to show the volume of research publications in the field over time. Thus, the number of references covered by the histogram is larger than the 66 publications mentioned earlier.

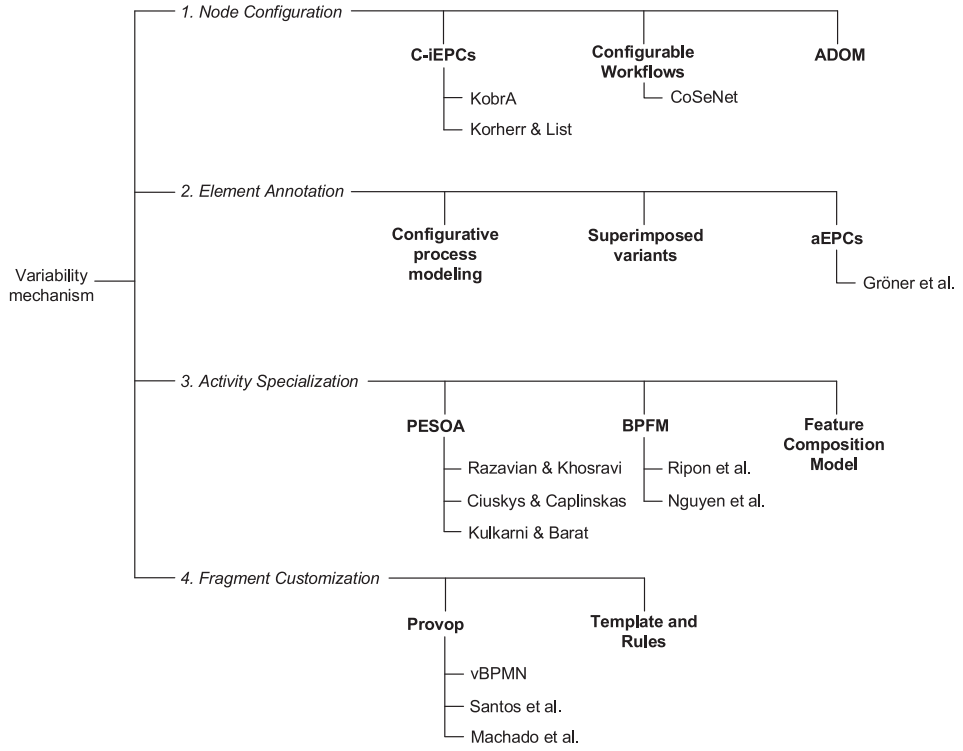

Fig. 3. Taxonomy of approaches for process model customization.

the customizable process model can be replaced by one of multiple specialized versions thereof. These approaches allow specialization only of activities and their attributes and not of other types of elements. Thus, we call this class *activity specialization*. Finally, in a fourth type of approaches, the relation between the customizable process model and its customized models is specified by means of *change operations* that can add, delete, or modify fragments of the customized model. Since the latter approaches can manipulate entire fragments, we call this class *fragment customization*.

The taxonomy induced by these observations is given in Figure 3, in which the main approaches are shown in bold and the subsumed ones are listed under their respective main approach. In line with previous work on variability modeling [Svahnberg et al. 2005; Bachmann and Clements 2005; Becker et al. 2007], we use the term *variability mechanism* to refer to a set of modeling constructs and their corresponding semantics, used to specify the relations between a customizable process model and its possible customized models (a list of all terms and their definitions is provided in Appendix E). The taxonomy presented in Figure 3 effectively classifies the variability mechanisms underpinning the identified approaches.

In the next four sections, we briefly introduce and evaluate the main approaches under each of the groups identified earlier using the 14 criteria described in Section 3. We discuss the subsumed approaches in Appendix B. The results of the assessment are then summarized in Table XII (Section 10). The evaluation is complemented by an overview of techniques for customization decision support (Appendix C) and for correctness support (Appendix D).

The assessment of each approach was performed independently by two authors of this article. The results were compared in order to resolve inconsistencies with the

mediation of a third author. Finally, we sought confirmation of our assessment from the authors of each primary publication.

6. GROUP 1: NODE CONFIGURATION

In the approaches in this group, a variation point is a node (called *configurable node*) of the customizable process model that is assigned different customization options. Activities, events, and gateways, as well as resources and objects associated with activities, may be marked as configurable nodes. Customization is achieved by selecting one customization option per configurable node. Each configurable node has an option to keep the node as is in the customized model and one or more options to restrict its behavior.

Configurable activities, events, resources, and objects can be customized by being kept *on* (they remain in the customized model) or switched *off* (they do not appear in the customized model). The semantics of switching an activity or event off is approach-specific, that is, the activity or event may be hidden without breaking the path to which it belonged or be removed altogether. Configurable gateways can be customized to an equal or more restrictive gateway in such a way that the customized process model produces the same or fewer execution traces than the customizable process model.

Three main approaches fall into this group: Configurable integrated EPCs (C-iEPCs), Configurable Workflows, and Application-based Domain Modeling (ADOM). They support different subsets of the configurable node types and customization options, for example, C-iEPCs do not support configurable events. In addition to customization by restriction, ADOM offers a weak form of extension in that extension points are not identified in the model.

6.1. Configurable Integrated Event-driven Process Chains (C-iEPCs)

Configurable integrated EPCs (C-iEPCs) [Rosemann and van der Aalst 2003; Dreiling et al. 2005, 2006; La Rosa et al. 2011] are an extension of the EPC language. Essentially, an iEPC is an EPC with resources and objects assigned to activities. A C-iEPC model is intended to capture the least common multiple of a family of iEPC variants. Differences among the process variants are indicated by configurable nodes. Each configurable node can be assigned a set of customization options, each referring to one or more process variants. Customization is achieved by restricting the behavior of the C-iEPC by assigning one customization option to each configurable node. Then, the C-iEPC is transformed into an iEPC by removing all those options that are no longer relevant. By doing so, one can derive one of the original variants of the given process family.

Activities and gateways can be marked as configurable with a thicker border. Events cannot be customized. Figure 4 shows the C-iEPC model for the postproduction example, which captures all variants of Figure 1.

Configurable gateways can be customized to an equal or more restrictive gateway. A configurable OR can be left as a regular OR (no restriction), or restricted to an XOR or to an AND gateway. Moreover, the number of its outgoing flows (if the gateway is a split) or the number of its incoming flows (if a join) can be restricted to any combination (e.g., two flows out of three), including being restricted to a single flow, in which case the gateway disappears.

For example, we can capture the choice of shooting medium by customizing the first OR-split in Figure 4. We can restrict this gateway to the outgoing flow leading to the event “Tape shooting” if the choice is tape. As a result, the branch starting with the event “Film shooting” is removed, and vice versa. Restricting the gateway to an AND-split ensures that both media are prepared for editing. In the three cases described, we anticipate the decision of the medium at configuration time. Alternatively,

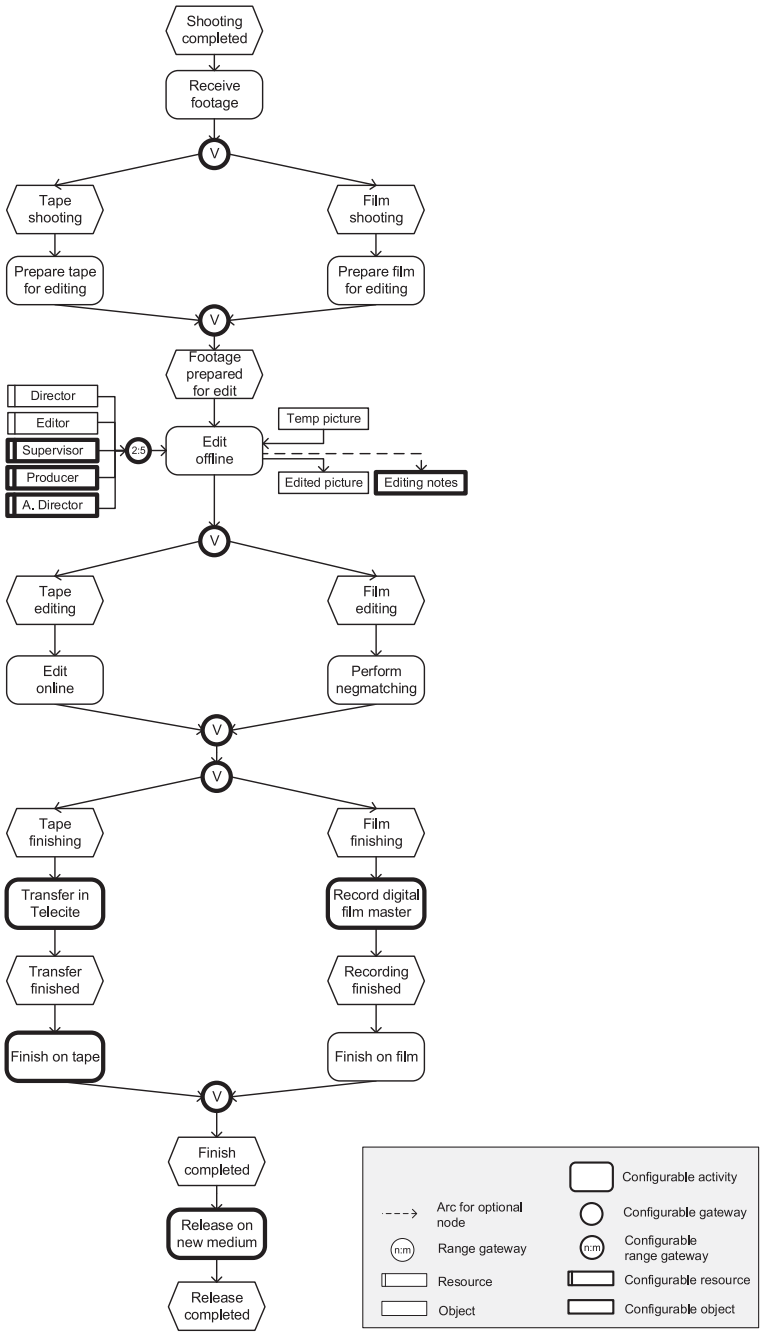

Fig. 4. The C-iEPC model representing all postproduction variants.

by configuring this gateway to an (X)OR-split, we postpone the decision till runtime, when the postproduction process is actually enacted (see, e.g., variant *f* in Figure 1).

Configurable activities can be kept *on* or switched *off*. In the latter case, the activity is simply hidden in the customized model. In addition, they can be customized to *optional*. This allows the deferral of the choice of whether to keep the activity or not until runtime. For example, the function "Release on new medium" is configurable in Figure 4; thus, we can switch it off for those postproduction projects for which this is not required.

Resources (called *roles* in C-iEPCs) and objects can also be made configurable. In the C-iEPC semantics, when an object is used as input to an activity, it can be marked as consumed to indicate that it will be destroyed upon use by the activity. Moreover, resources and objects can be mandatory or optional and can be connected to activities via logical gateways, called *range gateways*. Range gateways subsume the three logical types of OR, XOR, and AND but also allow any combination of the associated resources (objects), for example, at least 2 and at most 5 resources. Range gateways can also be optional, in which case they indicate that all connected resources (objects) are optional.

For simplicity, Figure 4 depicts only the resources and objects associated with activity "Edit offline." This activity is performed by at least two resources, requires a Temp picture as input, and produces an Edited picture as output. Editing notes are optional, since they might not be produced during the offline editing. In our example, three resources, one object, and one range gateway have been marked as configurable with a thicker border. This fine-grained mechanism to allocate resources and objects to activities leads to different customization options. If a resource, object, or range gateway is optional, it can be customized to *mandatory* so that is kept in the customized model or switched *off*. If it is mandatory, it can only be switched off. Further, resources and objects can be specialized to a subtype (e.g., a resource Producer can be specialized to an Executive Producer) according to a hierarchy model that complements the C-iEPC model (not shown in Figure 4). Configurable input objects that are consumed can be restricted to *used* so that they are not destroyed by the activity after use.

C-iEPCs are a conceptual process modeling language – they do not provide any execution support. C-iEPCs are formally defined in La Rosa et al. [2011], which also defines an algorithm to derive an iEPC from a C-iEPC. If the C-iEPC is structurally correct, this algorithm preserves correctness when creating a customized model by removing all nodes that are no longer connected to the initial and final events via a path and by reconnecting the remaining nodes. Behavioral correctness is ensured via constraints inference (see Appendix D.1).

Abstraction and guidance during customization are achieved by means of a questionnaire that captures domain properties and their values (see Appendix C.2) and is linked to the configurable nodes of a C-iEPC. C-iEPCs and associated questionnaire models are supported by the *Synergia*[3] and *Apromore*[4] toolsets. Using these toolsets, one can design C-iEPCs and questionnaire models, link these models, customize C-iEPCs via questionnaires, and obtain the resulting customized models. The use of C-iEPCs has been validated via a case study in the film industry [La Rosa et al. 2011].

Table I summarizes the evaluation results for C-iEPCs. Each column indicates to what extent the approach in question covers each evaluation criterion defined in Section 3. We used a "+" on a green background to indicate a criterion that is fulfilled, a "−" on a red background to indicate a criterion that is not fulfilled, and a "±" on an orange background to indicate partial fulfilment.

[3]See www.processconfiguration.com.
[4]See www.apromore.org.

Table I. Evaluation of C-iEPCs

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
+	+	+	+	-	+	-	+	+	+	+	+	+	+

6.2. Configurable Workflows

The Configurable Workflows approach [van der Aalst et al. 2006; Gottschalk et al. 2007, 2008] was first designed for conceptual models and later applied to executable languages with the aim to guarantee that the customized models can be executed. This led to the extension of several executable languages, such as SAP WebFlow, YAWL, and BPEL. In this survey, we focus on the extension to the YAWL language [Gottschalk et al. 2008], Configurable YAWL (C-YAWL), since this is the most significant one. The other extensions work in a similar way.

In YAWL, split and join gateways are graphically attached to activities: a join precedes an activity and models the activity's joining behavior; a split follows an activity and models its splitting behavior. C-YAWL extends YAWL with *ports* to identify configurable gateways. Configurable gateways are represented graphically with a thicker border, similar to C-iEPCs [van der Aalst et al. 2012]. A configurable split has an *outflow port* for each combination of subsequent flows that can be triggered after activity completion, while a configurable join has an *inflow port* for each combination of sequence flows through which the activity can be triggered.

Figure 5(a) depicts the postproduction example in C-YAWL in which, for illustration purposes, we modeled the preparation and editing of tape and film as mutually exclusive activities. To illustrate the concept of port, let us consider the case of the first XOR-split, that of activity τ_1. This XOR-split is used to route the process flow according to the shooting media. This split can either give control to the top or to the bottom of its outgoing flows. Thus, given that the combination of outgoing flows is equal to the number of such flows for an XOR-split, this split has only two outflow ports, one to trigger the flow to event 0*a* (leading to the preparation of the film) and the other to trigger the flow to event 0*b* (leading to the preparation of the tape).

Similarly, an XOR-join can be activated by each of its incoming flows; thus, it has one inflow port for each incoming flow. This is the case of the XOR-join of activity "Edit offline" in our example. In contrast, if the join (or split) is of type AND, it has only one inflow (or outflow) port. This is because an AND-join can only be activated by all its incoming flows (due to its synchronizing behavior); similarly, an AND-split gives control to all its outgoing flows simultaneously (due to its parallel behavior).

Let us now consider the OR-split of activity τ_2 in Figure 5(a). An OR-split has one outflow port for each combination of its outgoing flows, as it can give control to any combination of these flows. In our example, the OR-split is used to route control to activity "Record digital film master," or "Transfer in telecine," or both. Thus, this OR-split has three outflow ports: one to trigger the flow to event 4*b*, another to trigger the flow to event 4*a*, and the last to trigger both flows (4*a*, 4*b*).

The OR-join, on the other hand, only has one inflow port: this type of join is considered as an AND-join from a customization perspective due to its synchronizing merge behavior. This is the case of the OR-join of activity "Release on new medium," which has a single inflow port to receive control from both its incoming flows.

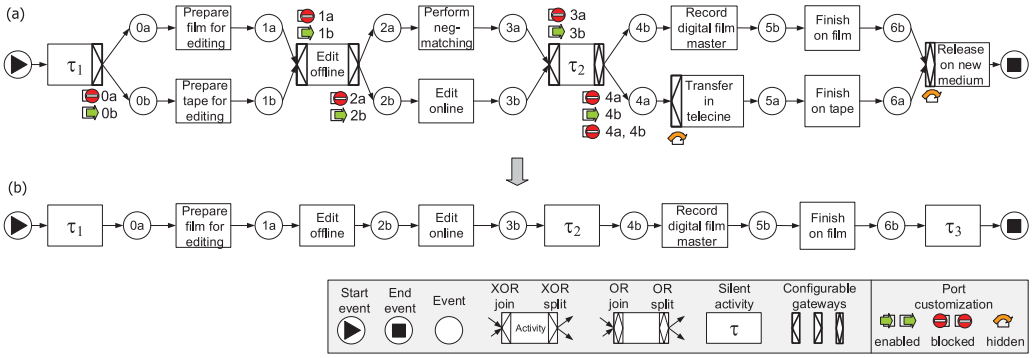

Fig. 5. (a) The postproduction example in C-YAWL with a sample customization. (b) The customized model.

Inflow ports have three customization options: *allowed*, *hidden*, and *blocked*. An inflow port can be blocked to prevent the triggering of its activity or hidden to skip the activity execution without blocking subsequent activities. Thus, in C-YAWL, both the semantics of hiding and removing an activity are supported. An inflow port that is neither blocked nor hidden is allowed, that is, it is kept as is in the customized model. An outflow port can only be blocked to prevent the triggering of the outgoing flows or left allowed. For convenience, all the ports can be allowed or blocked by default. Activities can be customized via their inflow ports. Resources and objects cannot be customized.

Figure 5(a) also shows a sample port customization for a project shot on tape, edited online, and finished on film, overlaid on the C-YAWL model. Let us consider the first XOR-split. The only outflow port allowed by the example customization is the one that leads to activity "Prepare tape for editing." The inflow port from event 1*a* to the XOR-join of activity "Edit offline" is customized as blocked while the other inflow port for this join is allowed in order to match the customization of the preceding XOR-split. Since the project is edited online, the outflow port of activity "Edit offline" triggering condition 2*b* is the only one to be allowed. In YAWL, an activity with a single incoming or outgoing flow has an implicit XOR behavior. This behavior is shown graphically if the gateway must be made configurable, as in the case of the join of activity "Transfer in telecine," since we needed to hide this activity's inflow port.

The hiding and blocking operations can also be applied to other YAWL elements, such as cancellation regions, composite activities, and multi-instance activities.

Figure 5(b) shows the YAWL model resulting from the example customization after applying the transformation algorithm defined in Gottschalk et al. [2008]. This algorithm removes all nodes that after customization are no longer on a path from the input to the output condition. In this way, the structural correctness of the model is guaranteed. Moreover, potential conflicts in the data conditions of the outgoing arcs of (X)OR-splits are taken care of in order for the resulting models to be fully executable. Two alternative techniques are available for ensuring the behavioral correctness of the customized models, one based on constraints inference and the other on partner synthesis, both described in Appendix D. Decision support is offered via the use of questionnaire models (see Appendix C.2).

This approach has been formalized [Gottschalk et al. 2008] and implemented in the *YAWL Editor*.[5] This tool allows one to create, customize, and transform C-YAWL

[5]See www.yawlfoundation.org.

Table II. Evaluation of Configurable Workflows

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
+	-	-	+	+	+	-	+	+	+	+	+	+	+

models into YAWL models; support for customization via questionnaire models is offered by the *Synergia* toolset. The use of C-YAWL models with questionnaire models has been validated in the municipality domain [Gottschalk et al. 2009; Lönn et al. 2012] involving domain experts as well as in software development processes for very small entities [Boucher et al. 2012].

Table II summarizes the evaluation results for Configurable Workflows.

6.3. ADOM: Application-Based Domain Modeling

In ADOM [Reinhartz-Berger and Sturm 2007; Reinhartz-Berger et al. 2009, 2010], configurable nodes (activities, events, and gateways) have a cardinality attribute of the form <min,max>. The cardinality specifies how many times a given node can be instantiated in the customized model. For example, an activity tagged with <0,1> is *optional* and as such it can be dropped in the customized model; an activity tagged with <1,n> is *mandatory* and can be instantiated up to n times in the customized model; an activity tagged with <1,1> must be instantiated exactly once, that is, it is kept as is in the customized model. The default cardinality <0,n> implies no constraints.

The cardinality assigned to gateways of type (X)OR indicates when the decision captured by the gateway should be made. A cardinality of <0,0> indicates that the gateway must not appear in the customized model. Thus, at customization time, one has to decide which outgoing branch(es) to keep in the case of a split or which incoming branch(es) to keep in the case of a join. If the cardinality is <0,1>, this decision can be deferred till runtime, that is, the gateway is optional. An OR gateway can be restricted to become an AND or XOR in the same way as in C-iEPCs and Configurable Workflows.

In ADOM, sequence flows can also be assigned a cardinality, unlike C-iEPCs and Configurable Workflows, for which sequence flows are not configurable. However, the customization of these flows is constrained by design by the customization of the configurable nodes in order to avoid disconnections in the customized model. For example, a flow with cardinality <0,1> between two nodes with cardinality <0,1> cannot be dropped if the two nodes are kept; otherwise, it would lead to a disconnection.

Figure 6(a) shows the postproduction example in EPCs with ADOM cardinality constraints. For example, event "Shooting completed," activity "Receive footage," and the flow in-between are mandatorythus, they can neither be removed nor instantiated more than once during customization. The OR-split and its matching OR-join are optional, as are the nodes in-between. This is done to allow a choice between either of the two branches or both. All elements after activity "Edit offline" are mandatory but have a maximum cardinality greater than 1. By doing so, each of these elements can be instantiated multiple times to model the various options that exist for editing and finishing in postproduction, though these options are not represented in the model.

In ADOM, commonalities between variants are thus captured by mandatory elements while variability is captured by optional elements and by those that can be instantiated multiple times. Since an ADOM model is meant to be used as a template,

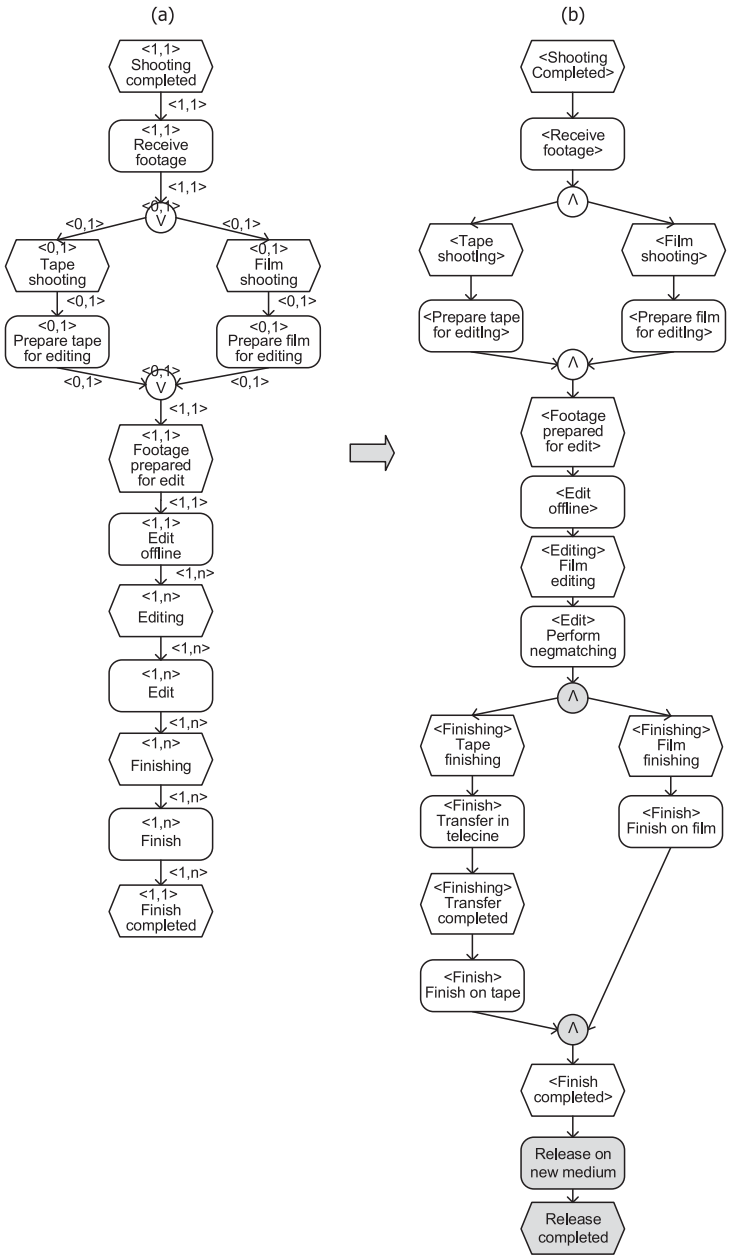

Fig. 6. (a) Postproduction example in ADOM-EPC. (b) A customized model.

some parts can be left underspecified. During customization, each configurable element can be instantiated according to its cardinality constraint. Moreover, *application-specific* elements can be added anywhere. These elements only appear in the customized model without any counterpart in the customizable model. Thus, ADOM supports customization by restriction (removing optional elements) and by extension (instantiating an element multiple times and adding application-specific elements).

Table III. Evaluation of ADOM

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
+	-	-	+	-	+	+	-	-	±	-	+	-	±

In a customized model, each node that has been derived from a configurable node bears a *model classifier* (indicated between "<" and ">"), that is, a reference to the originating node in the customizable model. If the label of the node needs to be changed, e.g. a more specific one is required, this can be added below the model classifier. Figure 6(b) shows a possible customization of the postproduction model, in which application-specific elements are highlighted in gray. For example, the first two gateways have been obtained by restricting the type of the first two OR gateways to an AND. Event "Film editing" and activity "Perform negmatching" derive from event "Editing," (resp., activity "Edit") and each have been given a new name. The second pair of AND gateways and the flow between activity "Transfer in telecine" and event "Transfer completed" are application-specific elements added to allow multiple instantiations of event "Finishing" and function "Finish."

ADOM has been applied to the control flow of EPCs, UML Activity Diagrams (ADs) and BPMNs at the conceptual level. For EPCs, specific rules have been defined to bind the customization of an event to that of an activity in order to maintain the alternation between events and activities required by EPCs, though disconnected nodes cannot be avoided. Behavioral correctness of the customized models is not guaranteed.

Customization is performed directly on the model level. There is no means to specify which combinations of instantiations are unfeasible from a domain viewpoint, and the addition of application-specific elements cannot be constrained. Reinhartz-Berger et al. [2009] describe a validation technique for ADOM-BPMN. This technique checks *a posteriori* that a customized model is compliant with its customizable model but does not prevent the user from generating inconsistent or irrelevant customizations in the first place. As such, decision support is not offered.

A formalization of ADOM is provided in Reinhartz-Berger et al. [2009] for BPMN and in Reinhartz-Berger et al. [2010] for EPCs. The approach has not been implemented in a tool. A subset of ADOM-BPMN has been validated in the development of a process-driven, service-oriented system but without involving domain experts [Reinhartz-Berger et al. 2009].

Table III summarizes the evaluation results for ADOM.

6.4. Recap

At the core, the approaches in this group allow different types of nodes in a customizable process model to be tagged as "configurable." A configurable node can be restricted at customization time. Activities can be removed, gateways can be restricted (an OR gateway can be turned into an AND or XOR gateway), and their incident arcs can be blocked (dropped altogether) or made mandatory. The approaches differ in terms of the types of nodes that can be made configurable, the configuration options offered, and supporting techniques and extra-functional criteria.

C-iEPC is the only approach in this group that supports customization of data objects and resources; C-YAWL is the only one that provides execution support; and ADOM is

the only one that supports customization by extension in addition to customization by restriction. C-iEPC and C-YAWL offer decision and correctness support, and fulfill the extra-functional criteria, whereas ADOM only partially does so.

7. GROUP 2: ELEMENT ANNOTATION

The three main approaches that fall in this group, Configurative Process Modeling, Superimposed Variants, and aEPCs, rely on the graphical *annotation* of model elements with properties of the application domain. Model elements that can be annotated include control-flow nodes (activities, events, and gateways), sequence flows, resources, and objects. Those model elements that are annotated become variation points. Different approaches support different subsets of model elements. Domain properties are assigned to model elements via *domain conditions*, which are Boolean expressions over domain properties (e.g., "low budget = true").

Customization is achieved by selecting domain properties. The selection may be done directly or may be aided by a domain model, such as a feature model or a product hierarchy. Based on this selection, the domain conditions are evaluated; those that are false lead to the corresponding model elements to be removed from the model. The required model transformation after the removal of the model elements is approach-specific.

7.1. Configurative Process Modeling

In configurative process modeling [Becker et al. 2004, 2007; Delfmann et al. 2006, 2007; Becker et al. 2006, 2007a, 2007b], customization is achieved by fading out model elements that are not relevant to a given business scenario. A set of domain properties, called *business characteristics*, are used to determine the available scenarios and later drive the customization.

In the case of the postproduction example, we can define a business characteristic "Shooting type" (ST) with values "Tape" (T) or "Film" (F) and a characteristic "Budget Level" (BL) with values "Low" (L), "Medium" (M), or "High" (H). The latter is a high-level characteristic since a choice on the budget typically affects multiple decisions in postproduction. These characteristics are linked to the elements of a process model by means of domain conditions, which are logical expressions over the characteristics. The link is rendered graphically by encapsulating the model elements into a shaded box to which the domain condition is attached (see Figure 7 for an example).

Process models are captured in extended EPCs (eEPCs), an extension of EPCs that incorporates resources and objects, similar to iEPCs. Business characteristics can be assigned to the following model elements: activities, events, resources and objects. Gateways cannot be directly configured. Rather, the approach expects the modeler to include all possible gateway variants in the customizable model.

Figure 7 shows a process model for postproduction in eEPCs (the control flow is the same as that of the C-iEPC model of Figure 4). Here, some elements have been associated with a logical expression referring to the project's budget. For instance, event "Film editing" and activity "Perform negmatching" are linked to the expression BL(H), which means that these elements are only suitable for a high-budget project due to the high costs involved in editing on film. On the other hand, activity "Edit online" is not associated with any condition, since it is suitable for any type of budget.

The customization of a process model to a specific scenario is done by marking those elements whose domain conditions evaluate to false as hidden. Then, a transformation is performed to remove the hidden elements, including those gateways that become irrelevant, and reconnect the remaining nodes. For example, by customizing the example in Figure 7 for a low-budget project, we obtain variant *b* in Figure 1. The transformation can fix simple structural issues, for example, the removal of gateways that have

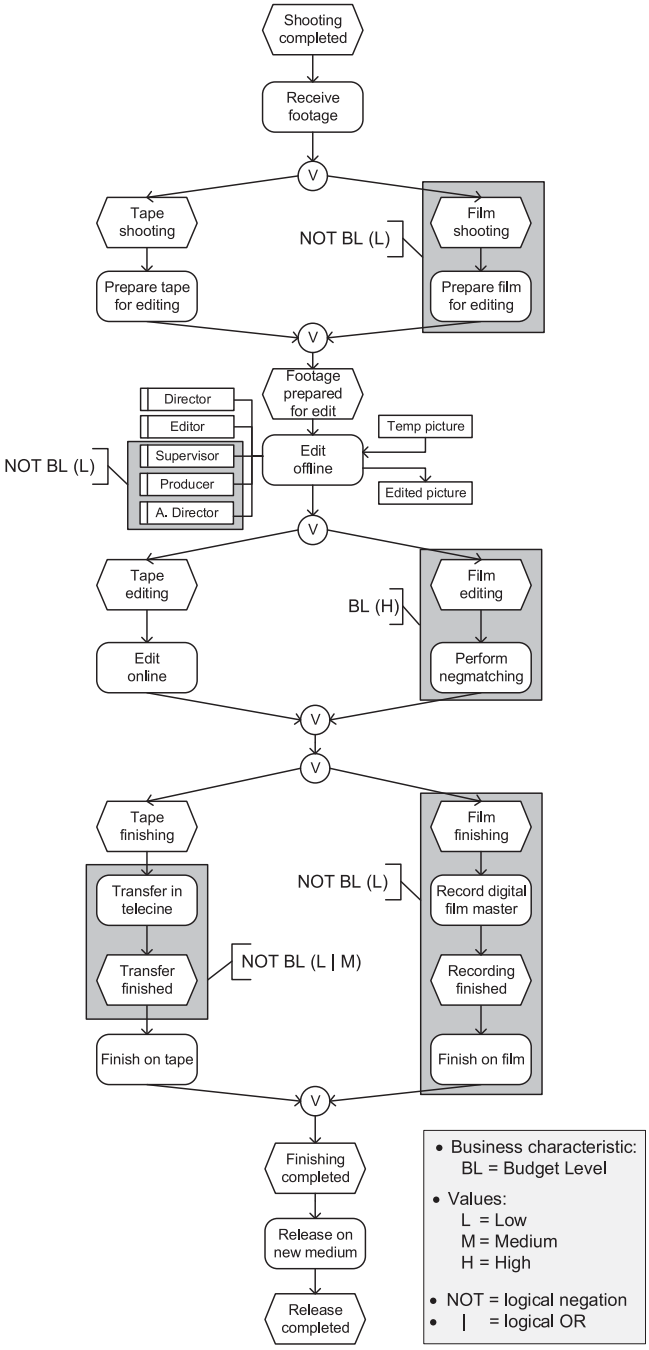

Fig. 7. A process model for postproduction in eEPCs with logical terms for the budget.

Table IV. Evaluation of Configurative Process Modeling

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
±	+	+	+	-	+	+	+	-	±	-	-	+	±

one input and one output flow, as in the example, but cannot ensure the structural and behavioral correctness of the resulting models. For instance, since both events and activities can be removed, the algorithm does not work in the presence of cycles or when an activity between two events is removed. Structural issues that cannot be fixed are prompted to the modeler, who has to manually correct them.

Business characteristics can also be applied to the meta-model layer (i.e., to the eEPC meta-model) to remove process modeling perspectives that are not relevant to a specific scenario. For example, one can hide all resources at once.

Customization is not carried out at the process-model level but rather via the evaluation of a set of business characteristics. However, the approach does not offer guidance to users when assigning values to the characteristics.

The approach also supports a set of *generic adaptation mechanisms* that can be used to refine and extend a process model after customization, for example, by adding new model fragments. The possible combinations of components can be restricted by interface definitions. However, the application of these mechanisms is left to the user without specific support, that is, extension points are not specified in the customized process model. Thus, this approach provides only a weak form of customization by extension.

The approach builds on eEPCs, which are a conceptual language. The approach has not been formalized. The model projection mechanism has been implemented as a prototype tool [Delfmann et al. 2006] that interacts with the ARIS platform. Business characteristics can be defined and linked to elements of an eEPC designed in ARIS. Users select the desired characteristics and the initial eEPC is customized to remove irrelevant elements. Similar features are also available in the *[em]* tool.[6] This approach has been applied in the fields of method engineering [Becker et al. 2007b] and change management [Becker et al. 2007a] and validated in the German public administration sector [Becker et al. 2006], though without domain expert involvement.

Table IV summarizes the evaluation results for Configurative Process Modeling.

7.2. Superimposed Variants

The idea of annotating model elements to capture variability is also investigated in Czarnecki and Antkiewicz [2005], Czarnecki et al. [2005], and Czarnecki and Pietroszek [2006]. In this approach, any control-flow element of UML ADs can be annotated using *presence conditions* and *meta-expressions*. A precedence condition determines if a model element is retained or removed. A meta-expression allows one to select the value of an attribute of a model element (e.g., an activity's label) from among a range of options. Customization is thus achieved by restriction only.

Both presence conditions and meta-expressions are captured by Boolean formulae over the features and feature attributes of a feature model (see Appendix C.1) and are

[6]http://em.uni-muenster.de.

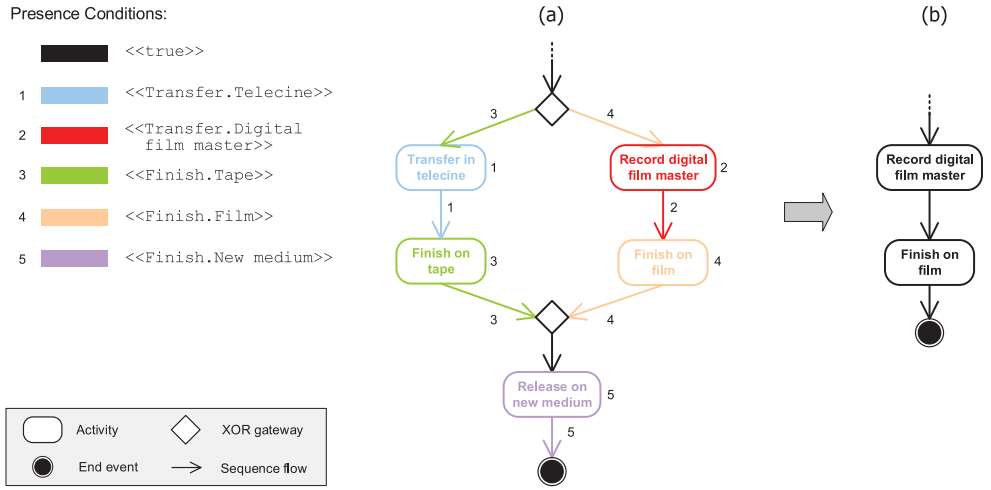

Fig. 8. (a) The postproduction example in annotated UML ADs. (b) A customized model.

evaluated against a feature configuration. These formulae are represented in disjunctive normal form, in which the basic terms are features designated by means of UML stereotypes. For example, the stereotype <<Tape ∨ Film>> indicates the disjunction between features "Tape" and "Film." The assignment of stereotypes to modeling elements is done through rendering mechanisms, such as labels, color schemes, or icons.

Figure 8(a) shows the finishing phase of the postproduction process as an annotated UML AD. For simplicity, in this example, we have specified only presence conditions. These annotations, rendered with a color and a number in the example, have been defined over the features of the feature model of Figure 15 (see Appendix C.1). This feature model captures the features (i.e., properties) of the postproduction domain, such as type of finish and type of transfer. For example, activity "Transfer in telecine" is associated with the subfeature "Telecine" of feature "Transfer" (annotated in blue with label "1"), while the two outgoing flows of the decision point are associated with the two subfeatures of "Finish": "Tape", respectively, "Film." All nonlabeled elements (in black) are associated with the *always-true* formula. These represent the commonalities of the model and cannot be removed, for example, the gateways and the end event in our example.

Customization is achieved by evaluating presence conditions and meta-expressions against a feature configuration. Model elements whose conditions evaluate to false are removed, while model attributes that are affected by meta-expressions are modified accordingly. No guidance is provided for the selection of the features to be kept.

Figure 8(b) shows a possible customized model for the postproduction example in which only the activities "Record digital film master" and "Finish on film" have been kept. This model can be obtained via a transformation algorithm that applies *patches* to reconnect model elements that have been disconnected during customization and *simplifications* to remove splits and joins that have been left with one incoming and one outgoing flow. Patches can be applied only to those nodes that have exactly one incoming and one outgoing flow; otherwise, an annotation error is raised. However, an automated verification procedure [Czarnecki and Pietroszek 2006] can be used to provide an *a priori* guarantee that no structurally incorrect customized model can be generated from a customization. Behavioral correctness is not dealt with, and no execution support is provided. The approach supports customization of only control-flow elements. Resources and objects cannot be customized.

Table V. Evaluation of Superimposed Variants

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
+	-	-	+	-	+	-	+	-	+	-	+	+	-

The approach has been formalized and implemented in an Eclipse plug-in[7] allowing users to configure UML ADs via so-called *cardinality-based feature models* [Czarnecki et al. 2005] and to check that the feature model and UML AD do not lead to structurally incorrect customized models. The approach has not been validated in practice.

Table V summarizes the evaluation results for Superimposed Variants.

7.3. aEPCs: Aggregated EPCs

Aggregated EPCs (aEPCs) [Reijers et al. 2009] are an extension of EPCs to capture a family of process variants. Similar to Configurative Process Modeling and Superimposed Variants, the idea is to annotate certain model elements (in this case, EPC activities and events) with domain properties, which are called *products* in aEPCs.

Figure 9(a) shows an example aEPC in which products associated with activities and events refer to the budget levels in postproduction. For example, activity "Transfer in telecine" occurs only in high-budget projects, while activity "Record digital film master" can also occur in medium-budget projects. Accordingly, "High budget" and "Medium budget" are subproducts of a composite product "Budget" in postproduction, that is, they capture the values of a given domain property. Other possible products include the shooting formats, picture cut methods, and finishing formats.

An activity or event may be associated with more than one product. In our example, activity "Record digital film master" is associated with two products ("High budget" and "Medium budget"). In order to avoid cluttering the model with many product associations, an aEPC can be accompanied by one or more *product hierarchies* in which the various products are organized hierarchically. A product hierarchy is a rooted tree in which the leaves are products and all other nodes are composite products representing product generalizations. In this way, a process model element can be associated with a composite product in place of a set of products. For example, Figure 9(b) shows the product hierarchy for the budget. The composite product "Budget" in this hierarchy can be used when an element is present in all budget levels, for example, activity "Receive footage."

Instead of capturing implications among model elements or domain properties (e.g., "Edit online" can be present only if "Prepare tape for editing" is present) as in other approaches, in aEPCs, all possible variants have to be resolved beforehand and mapped to a set of products (i.e., a composite product). Thus, while the use of composite products can reduce the number of products associated with a given element in principle, there may be a large number of composite products. This, in turn, may lead to cluttered aEPCs [Baier et al. 2010]. The choice of not modeling implications explicitly is motivated by the observation that, in practice, these logical expressions are difficult to conceive and interpret by domain experts. This was the result of testing C-EPCs (an

[7]See http://gp.uwaterloo.ca/fmp2rsm.

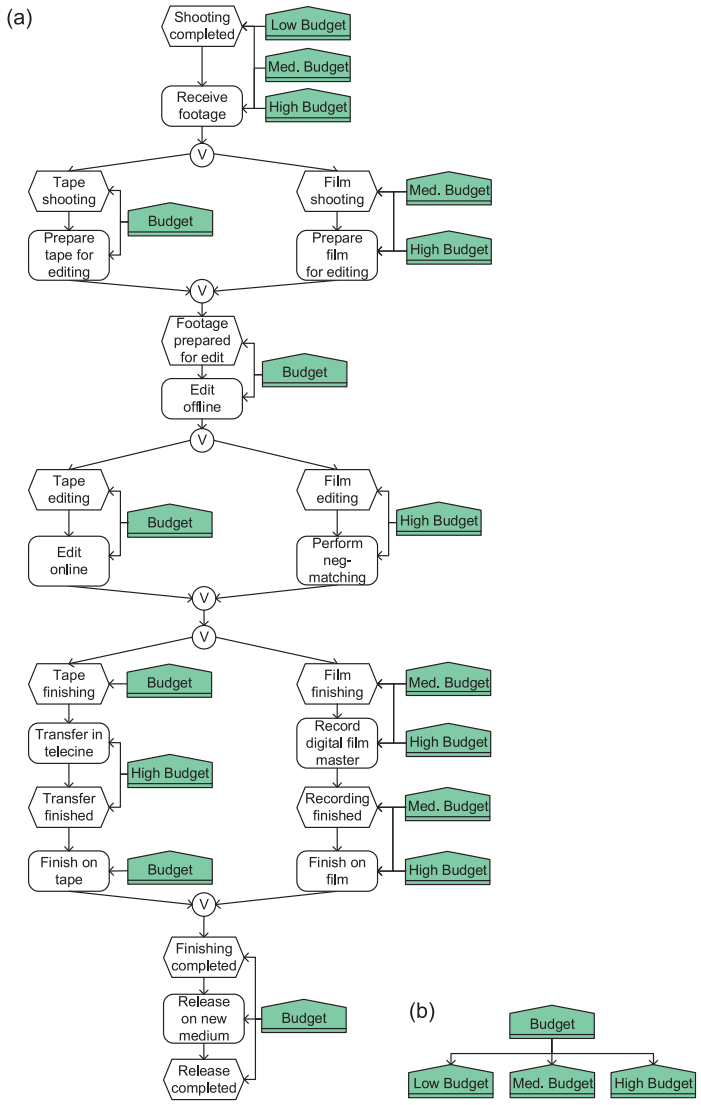

Fig. 9. (a) Postproduction example in aEPC. (b) Associated product hierarchy for budget.

ancestor of C-iEPCs) with domain experts of ING Investment Europe, with whom the aEPC approach was later validated [Reijers et al. 2009].

An aEPC is customized by choosing one or more products and removing all elements that are not associated with the products chosen. Customization is restricted to activities and events; gateways, objects, and resources are not customizable.

This approach works at the conceptual level only since aEPCs are conceptual models. The approach is fully formalized, including a transformation algorithm that removes the unneeded elements and cleans up the customized model in order to keep it structurally correct. In fact, in addition to the requirements of an EPC, there are requirements on how products can be associated with elements appearing before or after a sequence of gateways. Behavioral correctness is not dealt with. The transformation

Table VI. Evaluation of aEPCs

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
±	-	-	+	-	+	-	+	-	+	-	+	+	+

algorithm has been implemented in a tool that can import EPCs from ARIS and extend them into aEPCs. An advantage of organizing products into hierarchies is that an aEPC can be customized by removing products from the associated product hierarchy. Thus, this approach achieves process abstraction, though guidance is not offered.

Table VI summarizes the evaluation results for aEPCs.

7.4. Recap

Approaches in this group capture variability via annotations attached to elements of the customizable process model. These elements link an element in the customizable process model to an element in a domain model. Customization is performed by instantiating the domain model to capture a given set of requirements. Given an instance of the domain model and the annotations in the customizable process model, a transformation algorithm is applied to derive a customized model. In Configurative Process Modeling, the domain model consists of business characteristics; in Superimposed Variants, it takes the form of a feature model; and in aEPCs, it consists of products.

The approaches differ in terms of the model elements that can be customized. All approaches support the customization of control-flow model elements, although Configurative Process Modeling is limited in its support for customization of gateways. On the other hand, Configurative Process Modeling is the only one that supports resources and objects. In all three approaches, customization is achieved by restriction, though Configurative Process Modeling also supports a weak form of extension. All three approaches provide abstraction support, since the customization is driven by domain concepts. However, none provides customization guidance. All three approaches ensure structural correctness (at least to some extent) but not behavioral correctness. They all target conceptual process models rather than executable ones.

8. GROUP 3: ACTIVITY SPECIALIZATION

The main approaches in this group – Process Family Engineering in Service-Oriented Applications (PESOA), Business Process Family Model (BPFM), and Feature Model Composition – rely on activity specialization to achieve process model customization. An abstract activity can be defined as a variation point by assigning one or more *variants* to it. A variant is a *specialization* of an abstract activity, that is, one of its possible concrete refinements. For example, activities “Prepare tape for editing” and “Prepare film for editing” are two specializations of “Prepare medium for editing.” A special type of variation point is the *optional* activity: an abstract activity that can be specialized to an empty activity.

Variants can also be assigned to activity attributes, such as objects and resources, which become variation points. Events and gateways cannot be customized. Accordingly, variability is graphically rendered by marking activities and their attributes as variation points and connecting variants to variation points via a specialization arc.

Customization is achieved by selecting one or more variants per variation point, while optional activities can be switched off. Customization can be done directly on the process model or via the use of a domain model, such as a feature model. The routing behavior to be used when selecting more than one variant for a variation point, as well as the transformation needed to clean up the model from all unused variants and to remove optional activities that have been switched off, are approach-specific.

8.1. PESOA: Process Family Engineering in Service-Oriented Applications

The idea of capturing variability in process models has been explored in the PESOA project [Puhlmann et al. 2005; Schnieders and Puhlmann 2006; Schnieders 2006]. The aim of this project was not to provide a language for representing and customizing process models but rather to improve the customization of process-oriented software systems, that is, systems that are developed from the specification of process models. If the variability of a software system can be directly represented in a process model that describes the system's behavior, it is then possible to generate code stubs for the system from the customization of the process model itself. Since code generation is outside the scope of this article, we focus only on the way the authors represent process variability.

According to PESOA, a customizable process model is a conceptual process model in which certain activities have been marked with stereotypes to accommodate variability. Although stereotypes are an extensibility mechanism of UML, in this approach, they are applied to both UML ADs and BPMN models. The activities of a process model in which variability can occur are marked as variation points with the stereotype <<VarPoint>>. A variation point represents an abstract activity, such as "Prepare medium for editing," that needs to be specialized with a concrete variant (<<Variant>>) among a set of possible ones. For example, "Prepare medium for editing" can be specialized into "Prepare tape for editing" or "Prepare film for editing," or both. One can also mark the default variant for a variation point with the stereotype <<Default>>. Figure 10(a) shows the process model for postproduction in BPMN, for which some activities have been marked as variation points with their variants shown below the activity. For example, "Prepare tape for editing" is marked as the default variant of "Prepare medium for editing," as this is the most common choice in postproduction.

If the variants are exclusive, that is, if only one variant can be assigned to a given variation point, the stereotype <<Abstract>> is used instead of <<VarPoint>>. In Figure 10(a), we assume that the variants "Edit online" and "Perform negmatching" are exclusive; thus, the associated variation point "Cut picture" is marked with the tag <<Abstract>>. As a shortcut, when the variants are exclusive, the default specialization can be depicted directly on the variation point with the stereotype <<Alternative>>.

A variation point marked with the stereotype <<Null>> indicates an optional activity. It can only be associated with one variant and its resolution into the variant is not compulsory, in which case the activity is switched off. This is the case of the variation point "Transfer tape to film" that may be specialized into the variant "Record digital film master" or completely dropped from the process model. A shortcut for a <<Null>> variation point and its variant is to depict the variant directly on the variation point using the stereotype <<Optional>>, such as task "Transfer in telecine," which subsumes the variation point "Transfer film to tape."

Stereotypes can be assigned to activities and to activity attributes related to objects (e.g., input and output data). Gateways, events, and resources cannot be customized. During customization, each variation point is specialized into one or more variants depending on its type. Figure 10(b) shows a fragment of the BPMN process model for postproduction configured for a project shot on tape and edited online. The variants that are not required have been removed from the model. Extension mechanisms are

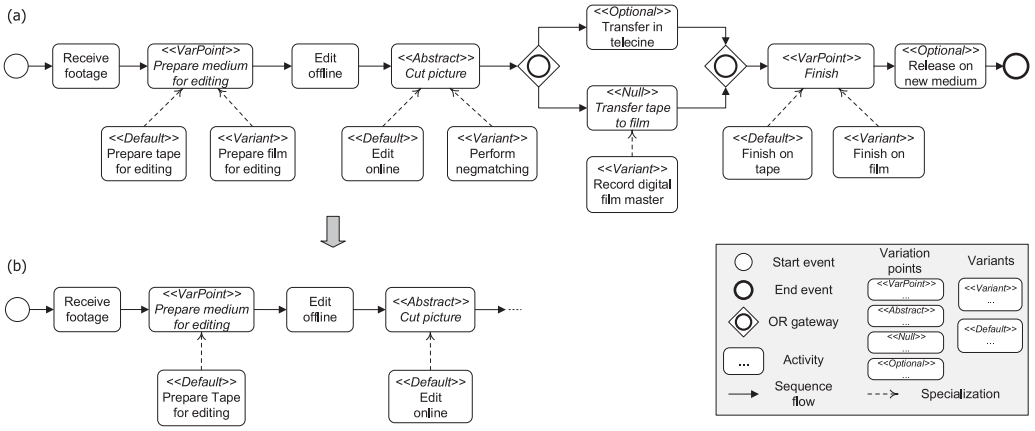

Fig. 10. (a) Postproduction example in PESOA-BPMN. (b) A customized model.

Table VII. Evaluation of PESOA

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
±	-	+	+	-	+	-	+	-	-	-	±	±	+

not provided. Abstraction from the process modeling language is achieved by linking process variants with domain properties, captured as features in a feature model (see Appendix C.1). Each process variant is tagged with a feature such that when a feature is disabled in a feature model configuration, the corresponding variant is removed from the process model. Domain constraints can be defined over feature values, thus restricting the possible combinations of variants in the process model. However, there is no guidance for the selection of a suitable set of features.

PESOA does not provide a transformation algorithm to derive customized models. The removal of certain variation points, such as <<Null>> or <<Optional>>, as well as the customization of a variation point when multiple variants are selected may lead to correctness issues that have to be fixed manually. A formalization is provided for selected concepts only [Puhlmann et al. 2005].

PESOA has been implemented as an Eclipse plug-in. In this implementation, the customization of a process model is limited to removal of undesired variants. The approach has been validated in the hotel booking domain in collaboration with ehotel and Delta Software Technology [Schnieders and Puhlmann 2006]. In this study, a set of BPMN process models were configured to drive the generation of Web applications in collaboration with domain experts.

Table VII summarizes the evaluation results for PESOA.

8.2. BPFM: Business Process Family Model

The BPFM [Moon et al. 2008] is a two-level approach to capture customizable process models using an extended version of UML ADs. The first level deals with basic

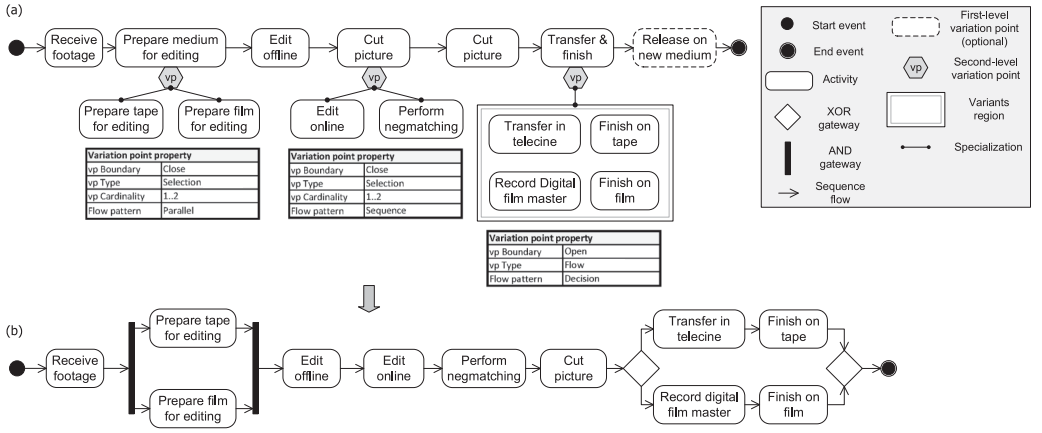

Fig. 11. (a) Postproduction example in BPFM. (b) A customized model.

customization. At this level, an activity can be defined as *common* if it cannot be customized or *optional* if it can be omitted during customization. The second level enables finer-grained customization by setting an activity as a *variation point* and assigning to it one or more specialized variants. Events, gateways, resources, and objects cannot be customized.

A variant is a concrete activity; a variation point is an abstract activity of one of the following types: (i) *Boolean*, (ii) *selection*, or (iii) *flow*. A Boolean variation point can be specialized into exactly one variant. A selection requires at least one variant to be selected. In this case, the exact number of variants to be selected can be set with a cardinality (e.g., `1..2`). When selecting more than one variant, in the BPFM, one needs to specify the control-flow relation between the selected variants (called *flow pattern*). This can be a *sequence* (the selected variants are ordered sequentially), *parallel* (the selected variants are executed in parallel using an AND-split and an AND-join), or *decision* (they are made mutually exclusive using an XOR-split and an XOR-join). A flow variation point is assigned a *variants region*, that is, a set of activities whose flow relations may be underspecified. At customization time, one needs to restrict the behavior by adding the required flows. A *flow pattern* can be specified for the flow variation point, in which case the activities in the variants region are organized according to the pattern, though the precise order needs to be decided by the user at customization time.

Further, the boundary of a variation point can be classified as either *closed* or *open*. A closed boundary restricts the choice of variants to those already identified; an open boundary allows the introduction of new variants during customization. Thus, in principle, this approach supports both customization by restriction and extension. However, there is no support for plugging in new variants into a variation point.

Figure 11(a) depicts the postproduction example in BPFM. Here, there are three activities marked with a variation point and one optional activity. Activities “Prepare medium for editing” and “Cut picture” are of type selection. They have been assigned two variants each. The first activity prescribes a parallel flow pattern, while the second prescribes a sequence flow pattern, each with the option of selecting at least one and at most two variants. Accordingly, Figure 11(b) shows a customized model in which the first variation point has been customized to the parallel execution of both its variants, while the second has been customized to the sequence of its variants. Activity “Transfer & finish” is an open variation point of type flow with a decision pattern between the

Table VIII. Evaluation of BPFM

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
±	-	-	+	-	+	+	-	-	-	-	-	±	-

activities in the associated variants region. Accordingly, in Figure 11(b), this variation point has been customized to a decision between two branches, each hosting two of the four activities present in the variants region. In the case of an open-flow variation point, the arrangement of activities inside the variants region within a given control-flow structure is entirely left to the user. Finally, in our example, the optional activity "Release on new medium" has been dropped in the customized model.

In BPFM, it is also possible to define dependencies (called *dependency constraints*) between variation points, between variants, or between variation points and variants. If, for example, a variant is chosen for a given variation point, this can restrict the choice of the variants for another variation point.

A tool implementing this approach is available as an Eclipse plug-in. The tool can prune a customized process model by removing the unused variants but does not offer a complete transformation algorithm for embedding the selected variants into the process model. The approach has not been formalized or validated in practice. It does not provide any correctness, abstraction, or decision support.

Table VIII summarizes the evaluation results for BPFM.

8.3. Feature Model Composition

In Feature Model Composition [Acher et al. 2010a], a process model (called *workflow*) is defined as a collection of activities (called *services*). Activities are implicitly related via data dependencies. Specifically, each activity has a collection of attributes called *dataports*. A dataport captures an input or an output data object of the activity. If an input dataport of an activity refers to the same object as an output dataport of another activity, there exists an implicit data dependency between these two activities.

In order to capture variability, an activity is allowed to have any number of variation points (called *concerns*). A concern refers to any activity attribute. Examples of attributes are dataports, functional interfaces, activity behavior, and other low-level implementation aspects. Each concern is modeled as a separate feature model that captures the variants that exist for a concern and their relations. Customization of concerns is achieved by deselecting features from the respective feature models.

Figure 12 shows a customizable process model in the Feature Model Composition approach using our running example. A feature model has been defined for the concern "Shooting medium" and mapped to the output dataport of "Prepare medium for editing" in order to capture the fact that this activity can have a film, a tape, or both media as output. Similarly, the same feature model has been associated with the input dataport of the subsequent activity "Edit offline." Furthermore, the concern "Cut" with variants "Online" and "Negmatching" has been associated with the functional interface of activity "Perform cut" to indicate that this type of activity can also be configured.

A concern of one activity may be incompatible with that of a subsequent activity; thus, a consistency check is needed when customizing a model. This check is performed by analyzing input and output dataports based on dependency rules. Specifically, the

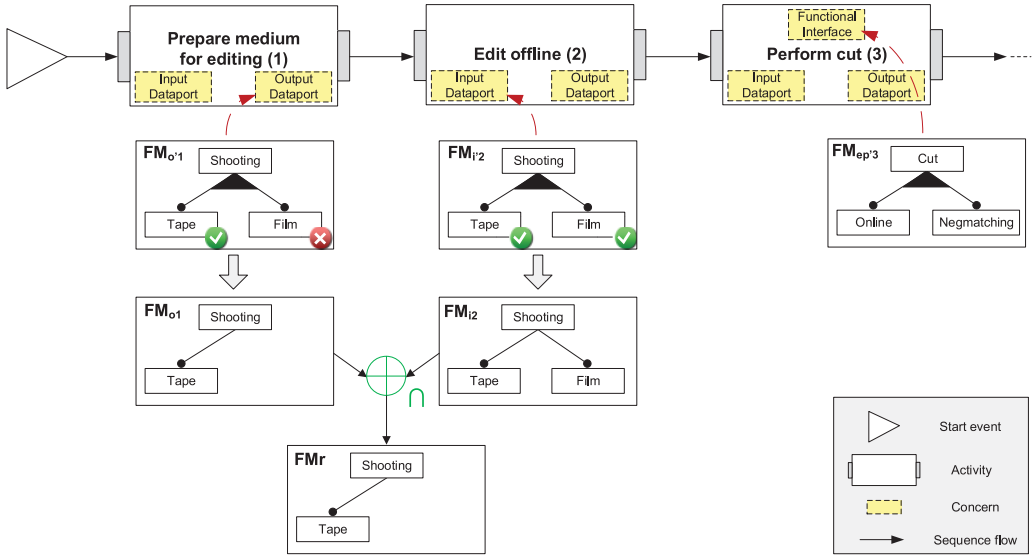

Fig. 12. Postproduction example using Feature Model Composition and a possible customization.

feature models of the relevant concerns are checked for mutual consistency; then, a merged diagram is created by intersecting the various feature models. In this way, the consistency of two connected activities is ensured. The merge operator is used to compose feature models that refer to the same activity dimension. Its syntax and semantics are defined in Acher et al. [2010b], while the syntax of a customizable process model is defined in Acher et al. [2010a]. While inconsistencies in data dependencies that may arise during customization are addressed by this approach, the process modeling language adopted is abstract and not actually executable.

When producing a customized model, it is necessary to add the control-flow dependencies based on the implicit dependencies of the various activity attributes, such as data dependencies. Three types of control-flow dependencies are possible: sequential, concurrent (AND behavior), and conditional (XOR behavior). The dependency rules for consistency checks between two activities (see Figure 12) are not sufficient when there is a sequential, concurrent, or conditional ordering of more than two activities. This is addressed via an extended set of dependency rules that ensures the consistency of the activities in a process model.

As shown in the example, this approach can be used to customize business objects and other activity attributes, such as the associated resources. However, concerns are internal to each activity. As such, the control flow cannot be configured. This is the only evaluated approach that suffers from this limitation.

In this approach, feature models do not provide abstraction for the customization of concerns since they refer to low-level aspects, such as different dataports related to a software service. Moreover, one has to configure one feature model per concern. There is no overarching feature model to customize the process model using properties of the application domain, such as, for example, in PESOA. Similarly, no guidance support is offered.

Since the control flow cannot be configured and data dependencies are preserved during customization, the approach guarantees that the customized models are both structurally and behaviorally correct.

Table IX. Evaluation of Feature Model Composition

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
-	+	+	+	±	+	-	-	-	+	+	+	+	-

An implementation is described online,[8] though the tool cannot be downloaded and the authors have not replied to our request for assistance with their tool. The Feature Model Composition approach is motivated by the need for customizing medical imaging grid services, though it has not been validated in practice.

Table IX summarizes the evaluation results for Feature Model Composition.

8.4. Recap

A distinctive characteristic of approaches in this group is that they allow customization of only individual activities and not of other control-flow elements (events and gateways). Feature Model Composition does not even support the customization of an activity itself but rather focuses on the customization of an activity's inputs and outputs. In other words, every activity in the customizable process model will appear in every customized model thereof. The customized models differ only in terms of the involved resources and data objects. Control-flow relations between activities have to be specified over the customized process model based on the data dependencies between activities.

Approaches in this group focus on conceptual process models. Since specialization is a form of behavior restriction, the approaches support customization by restriction. The BPFM also supports a weak form of extension. PESOA provides abstraction support, but none of the approaches provides guidance. The approaches in this group do not ensure structural or behavioral correctness except for Feature Model Composition, which trivially achieves correctness support since it does not capture control-flow dependencies between activities.

9. GROUP 4: FRAGMENT CUSTOMIZATION

Approaches in this group are based on the application of *change operations* to restrict or extend the customizable process model. Two atomic change operations can be used to customize the control flow: *delete*, to remove a fragment from the model, and *insert*, to add a fragment into the model. More complex operations, such as *move* or *replace*, can be provided by combining delete and insert. The fragment to be deleted or inserted must be single-entry, single-exit. Accordingly, each operation requires two sequence flows of the process model to delimit the portion of the base model to be deleted or inserted (the two flows may coincide). These variation points (called *adjustment points*) may be explicitly represented by marking selected flows of the model; otherwise, each flow is assumed to be an adjustment point. The required model transformation after the application of the change operations is approach-specific. A third change operation, *modify*, is used to modify the resources or objects associated with an activity, for example, replacing a resource with another or assigning a new resource to an activity.

[8]See http://modalis.polytech.unice.fr/softwares/manvarwor.

Operations can be organized in an *operation sequence* so that multiple operations can be performed in a given order on the customizable process model. Moreover, these sequences can be associated with *domain conditions*, that is, predicates over domain properties, to determine when the sequence of operations in question should be applied.

This group counts two main approaches: Provop and Template and Rules.

9.1. Provop: Process Variants by Options

In Provop [Hallerbach et al. 2008, 2009a, 2009b, 2010], customization is achieved by applying change operations to a *base model* marked with adjustment points. The base model can be a standard process (e.g., a reference model for a particular domain), the most frequently used process variant, a generic model, the superset of all variants, or their intersection. For example, in Figure 13, we identified variant *a* from the set of postproduction variants in Figure 1 as the base model, since this is one of the simplest variants for postproduction, and defined eight adjustment points on this model.

In addition to the two atomic change operations for the control flow (DELETE and INSERT) and the MODIFY operation to customize objects and resources, Provop supports a fourth operation, MOVE, to relocate a fragment delimited by two adjustment points in the base model to another part of the model delimited by two different adjustment points. Operation sequences are called *options* in Provop.

For example, the DELETE operation in Option 1 of our example will delete the content between adjustment points "w" and "z." As a shorthand notation, it is possible to delete a single node simply by providing its identifier. A fragment is inserted in parallel to the portion of the base model delimited by two given adjustment points if this portion contains some node. For example, in the case of the first INSERT of Option 4, an AND-split and an AND-join are used to link the fragment to the adjustment points. If the portion contains a sequence flow only or is empty (e.g., as a result of a previous DELETE), the fragment is inserted in place of the flow or between the two adjustment points, respectively. An example of this is the second INSERT of Option 2, in which the sequence "Record digital film master"–"Recording completed" is inserted in place of the flow between "y" and "n."

Since adjustment points can be defined only on the control flow, in Provop it is not possible to represent variability in the resource and object perspectives.

We organized the change operations in our example in four options. The application of Options 1 and 2 on the base model yields variant *b* of postproduction, Option 3 yields variant *c*, while Option 4 yields variant *d* (see Figure 1). The use of certain combinations of options can be restricted by defining *option constraints*, such as mutual exclusion, implication, and n-out-of-m choices. For example, Options 1 and 3 of our example are set as mutually exclusive, since Option 1 removes the adjustment point "x" required by Option 3. The rationale behind the use of these constraints is to avoid creating situations that may prevent the application of an option or that may introduce errors in the resulting variants.

A five-step method can be used to drive the customization of process models via properties of the application domain [Hallerbach et al. 2009a, 2010]. In Step 1, the user determines all the possible *contexts* in the application domain. A context is a domain property represented as a variable, such as "budget," with all its possible values, for example, "high," "medium," and "low." One can also specify domain constraints (called *context constraints*) in the form of Boolean expressions to limit the interplay among contexts, for example, "budget = low $\Rightarrow$ finish = tape." Each option is then assigned a domain condition (*context rule*) in the form of a Boolean expression over the values of context variables to limit the applicability of that option to a particular business scenario. For instance, Option 1 can only be applied if shooting and finish are done on tape and editing is done online. In Step 2, for each context, the set of relevant options is

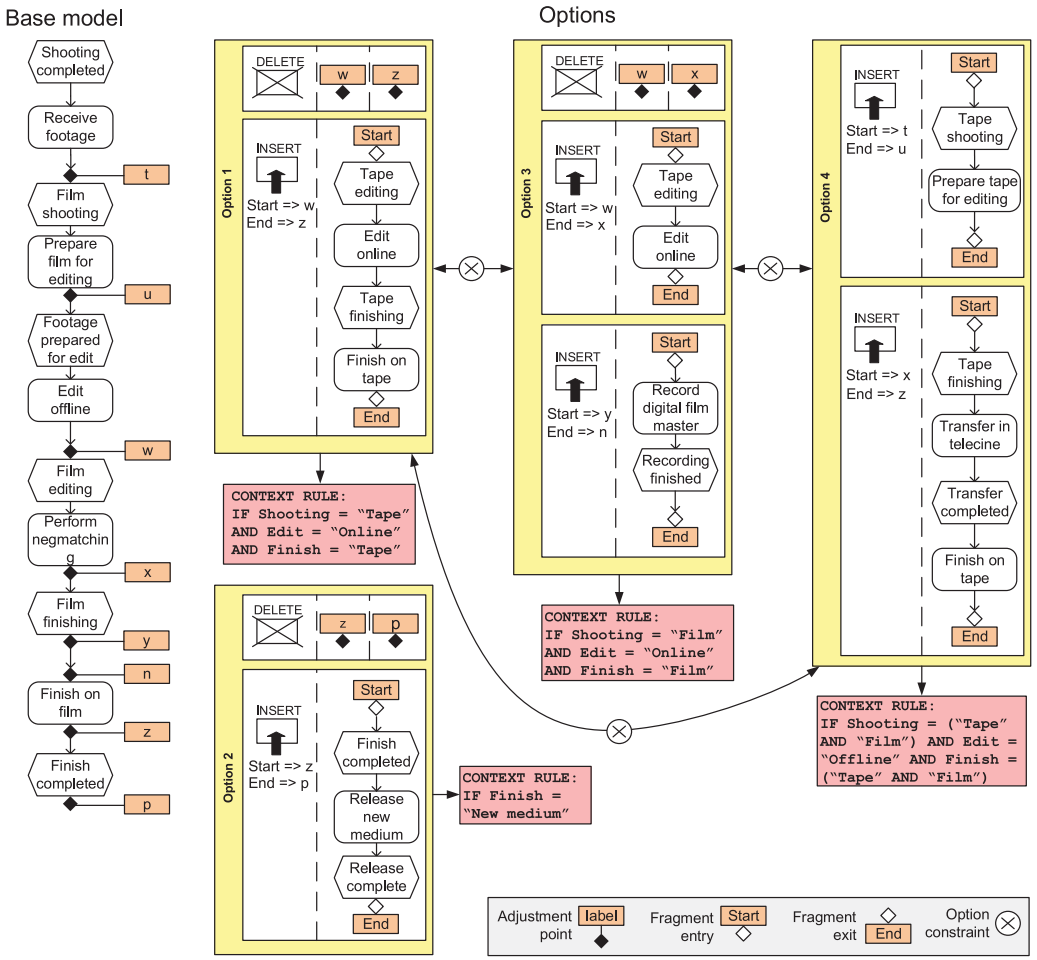

Fig. 13. Postproduction example in Provop.

automatically determined. In Step 3, the consistency of the retrieved options for each context is checked against the option constraints. If inconsistencies are found, these are prompted to the user, for example, an option constraint may contradict a context constraint. In Step 4, all valid sets of options are applied to the base model for each context, and the resulting variant is checked for correctness in Step 5. Those models that are incorrect are discarded. Contexts and context rules offer abstraction for the customization of the base model, though guidance in the selection of the various options is not provided.

Provop can be applied to any modeling language with the only structural restriction that the fragments to be customized must be single-entry, single-exit. The base model is not required to be correct. This, however, cannot guarantee the correctness of the customized model *a priori*. For example, the model may have disconnections or splits and joins of different types in a given single-entry, single exit fragment, leading to behavioral anomalies. Instead, correctness is checked *a posteriori* (in Step 5) using existing correctness-checking techniques. In fact, the number of valid combinations

Table X. Evaluation of Provop

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
+	±	±	+	-	+	+	+	-	-	-	±	+	±

of context variables into contexts may be very large, making an *a priori* check of all derivable customized models unfeasible in such cases.

The approach addresses customization of only conceptual process models. The basic concepts are formalized, though the semantics of the change operations is not specified. Provop has been implemented on top of ARIS [Hallerbach et al. 2010]. This tool allows users to define change operations and organize them in options and to apply them to BPMN models enhanced with adjustment points in order to derive customized models. The tool is not publicly available. Provop's design requirements have been derived from various case studies in the automotive and health care industries [Hallerbach et al. 2010], and Provop models have been created by the authors in these domains. However, these models have not been validated with domain experts.

Table X summarizes the evaluation results for Provop.

9.2. Template and Rules

Template and Rules [Kumar and Yao 2009, 2012] captures the variability of a process family by processing a set of *business rules* associated with a *process template*. The process template is the customizable process model: a simple, block-structured process model that should be chosen in order to have the shortest structural distance from all process variants of the family. The rules are sequences of change operations used to customize the template by restricting or extending its behavior. Change operations affect the control flow (by deleting, inserting, moving, or replacing a fragment), the resources (by assigning a resource to an activity or changing the value of a resource property), and the data objects (by assigning a value to a data attribute or changing the value of an activity's input/output data). The operations on resources and objects are "modify" operations, while those on the control flow are (a combination of) delete and insert. Unlike Provop, gateways cannot be directly customized and adjustment points are not explicitly represented, meaning that change operations can be applied to virtually any process model fragment. As a result, though, in Template and Rules, variability is not graphically represented in any process model perspective and can only be inferred from the rules accompanying the template.

Rules are assigned domain conditions (e.g., "process.budget = high"), which, if satisfied, allow the corresponding change operations to be applied to the process template. The use of such conditions provides abstraction from the customizable process model, though there is no guidance support.

Figure 14 shows the application of this approach to our running example, using the BPMN language. Here, the template describes a simple variant for editing and finishing on tape and releasing on new medium. This template is accompanied by three rules (*R1*, *R2*, and *R3*) embracing control-flow and resource aspects. For example, *R1* is used to customize the template for a high-budget production process. Accordingly, we need to insert the activities required for editing and finishing also on film, such as "Prepare film for editing," to be inserted in parallel to "Prepare tape for editing," and "Transfer in

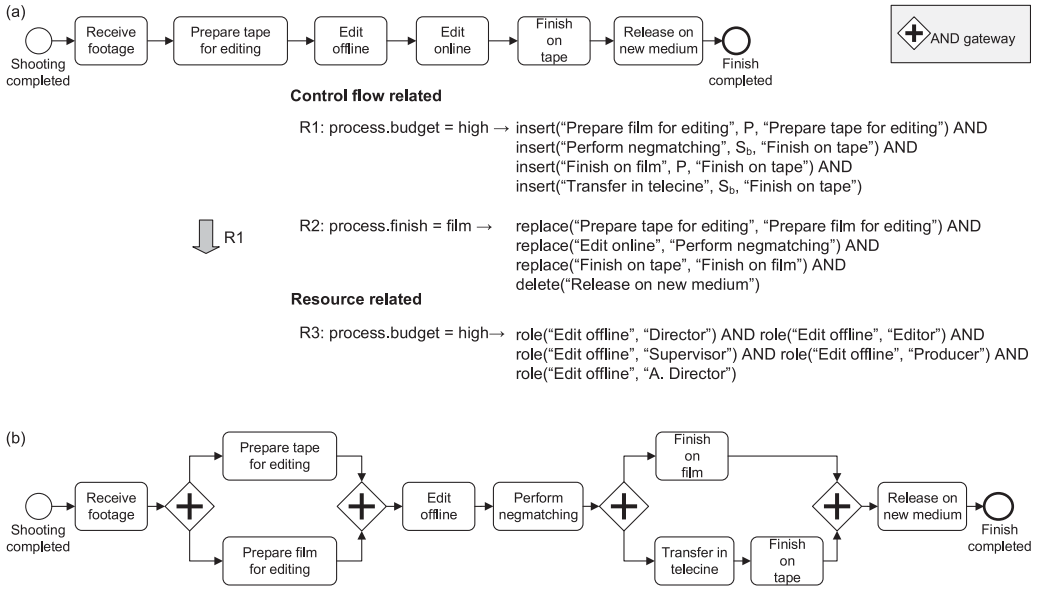

Fig. 14. (a) Postproduction example in Template & Rules. (b) Customized model.

telecine," which goes before "Finish on tape," and so on (where $insert(t_1, P, t_2)$ in a rule indicates to insert activity t_1 in parallel to t_2 while $insert(t_1, S_b, t_2)$ indicates to insert t_1 before t_2). *R3* is an example of a rule to configure resource aspects: if the budget is high, multiple resources (e.g., "Director," "Editor," "Supervisor") will perform activity "Edit offline." Predicate $role(t, r)$ indicates that resource r is assigned to activity t.

Change operations are applied to a tree representation of the template and, similar to Provop, affect only single-entry, single-exit fragments of the template. Moreover, the application of each change operation triggers some cleaning operations to avoid disconnected model elements and remove trivial gateways and sequence flows. For example, after deleting activity "Release on new medium" from the template in Figure 14 via the application of rule *R2*, activity "Finish on tape" and event "Finish completed" will be reconnected. Similarly, if there remains one branch only between two AND gateways, the two gateways will be removed altogether. Thus, change operations cannot cause any structural or behavioral issues in the process template.

Change operations are described in detail in terms of changes to the tree representation of the template and an algorithm is provided to customize the tree. However, a formalization of all notions is missing. Also, an algorithm to transform a process model into a tree representation and vice versa is missing, while rule-conflict resolution is only exemplified by a matrix that disallows certain combinations of rules.

The approach has been implemented using BPEL as the base language, though the tool is not publicly available. Given a template and a set of rules, the tool uses the Drools-expert rule engine to check for conflicts between the available rules. If conflicts exist (e.g., one rule deletes an activity that another rule is trying to insert), the user is notified to either resolve them or assign a priority to each rule. The applicability of each rule is checked (e.g., it is not possible to delete a nonexistent node) and errors are triggered for those rules that are not applicable. Finally, a customized process model is obtained from the template by using only those rules that are nonconflicting and applicable. This model is checked for dataflow inconsistencies, for example, a task whose data input is no longer available, in order to guarantee the executability of the

Table XI. Evaluation of Template and Rules

Scope					Customization Type		Supporting techniques				Extra-Functional		
Process Perspective			Process Type				Decision Support		Correctness Support				
Control-flow	Resources	Objects	Conceptual	Executable	Restriction	Extension	Abstraction	Guidance	Structural	Behavioral	Formalization	Implementation	Validation
±	±	±	+	+	+	+	+	-	+	+	±	+	-

customized model. This check is done *a posteriori*; as a result, a customized model may be unfeasible altogether. The approach has not been validated in practice.

Table XI summarizes the evaluation results for Template and Rules.

9.3. Recap

The approaches in this group capture variability by means of (sequences of) change operations applied to the customizable process model. These change operations can add, delete, or modify. Hence, the approaches in this group naturally support both customization by restriction and by extension. Unlike Provop, Templates and Rules supports both conceptual and executable models and supports abstraction as well as structural and behavioral correctness (at the expense of structural restrictions on the types of fragments that can be deleted or added). The combination of these characteristics makes Templates and Rules stand out in terms of its comprehensive coverage of the evaluation criteria.

10. DISCUSSION

This section compares the surveyed approaches (including the subsumed ones) in terms of the criteria introduced in Section 3. This comparison is followed by a discussion on the research questions introduced in Section 1.

10.1. Comparative Analysis

The comparative analysis of approaches is summarized in Table XII. The first column lists the eleven main approaches and the twelve subsumed approaches. The next three columns indicate the year of the primary publication, the total number of citations (including all papers related to a given approach), and the modeling language(s) employed by the approach. The remaining columns indicate the coverage of each criterion.

Regarding the modeling scope, all approaches (except Feature Model Composition) provide customization mechanisms along the control-flow perspective, but only a handful support the customization of resources and objects. Approaches based on BPMN and UML ADs do not support the customization of resources, except for Santos et al. [2010]. This is probably because these two languages provide limited support for capturing resources beyond the ability to associate a lane or a pool with each activity in the process. Accordingly, it is mainly in the context of EPCs or other languages that the question of customization of resources is posed. Customization of objects, on the other hand, is available in different languages, but only one (Templates and Rules) addresses customization of data objects in the context of executable process models.

In a similar vein, most approaches are based on conceptual modeling languages (UML ADs, EPCs, BPMN) and are thus focused on the customization of conceptual rather than executable process models. BPMN version 2.0 supports the specification of executable processes, but no customization approach so far covers the executable features of BPMN (e.g., customization of data variables). Configurable Workflows and Template and Rules

Table XII. Evaluation Results at a Glance, Ordered by Year of Primary Publication

	Approach	Year of primary publication	Total citations	Process modeling language	Scope: Process Perspective: Control-flow	Scope: Process Perspective: Resources	Scope: Process Perspective: Objects	Scope: Process Type: Conceptual	Scope: Process Type: Executable	Customization Type: Restriction	Customization Type: Extension	Supporting techniques: Decision Support: Abstraction	Supporting techniques: Decision Support: Guidance	Supporting techniques: Correctness Support: Structural	Supporting techniques: Correctness Support: Behavioral	Extra-Functional: Formalization	Extra-Functional: Implementation	Extra-Functional: Validation
Main	C-iEPCs	2003	1,313	C-iEPCs	+	+	+	+	-	+	-	+	+	+	+	+	+	+
	Configurative Proc. Modeling	2004	278	eEPCs	±	+	+	+	-	+	+	+	-	±	-	-	+	±
	PESOA	2005	226	BPMN, UML ADs	±	-	+	+	-	+	-	+	-	-	-	±	±	+
	Superimposed Variants	2005	1,287	UML ADs	+	-	-	+	-	+	-	+	-	+	-	+	+	-
	Configurable Workflows	2006	772	C-YAWL, C-SAP, C-BPEL	+	-	-	+	+	+	-	+	+	+	+	+	+	+
	ADOM	2007	125	UML ADs, EPCs, BPMN	+	-	-	+	-	+	+	-	-	±	-	+	-	±
	BPFM	2008	22	UML ADs	±	-	-	+	-	+	+	-	-	-	-	-	±	-
	Provop	2008	577	Any	+	±	±	+	-	+	+	+	-	-	-	±	+	±
	aEPCs	2009	90	aEPCs	±	-	-	+	-	+	-	+	-	+	-	+	+	+
	Template and Rules	2009	52	Block-structured BPEL	±	±	±	+	+	+	+	+	-	+	+	±	+	-
	Feature Model Composition	2010	29	Any	-	+	+	+	±	+	-	-	-	+	+	+	+	-
Subsumed	KobrA	2000	297	UML ADs	+	-	-	+	-	+	-	+	-	-	-	-	+	±
	Ciuksys & Caplinskas	2006	15	UML ADs	±	-	-	+	-	+	-	+	-	-	-	-	-	-
	Korherr & List	2007	33	UML ADs	+	-	-	+	-	+	-	+	-	-	-	-	-	-
	Razavian & Khosravi	2008	55	UML ADs	±	-	+	+	-	+	-	-	-	-	-	-	-	-
	Kulkarni & Barat	2010	15	BPMN	±	-	-	+	-	+	-	-	-	-	-	±	-	-
	Ripon et al.	2010	10	UML ADs	±	-	-	+	-	+	-	+	-	-	-	-	-	-
	Santos et al.	2010	22	BPMN	+	+	-	+	-	+	+	+	-	-	-	-	-	-
	CoSeNet	2011	34	CoSeNets	+	-	-	+	±	+	-	-	-	+	+	±	+	±
	Machado et al.	2011	16	BPMN	+	-	-	+	±	+	+	+	-	-	-	+	±	-
	Nguyen et al.	2011	27	BPMN	±	-	+	+	-	+	-	+	-	-	-	-	+	-
	vBPMN	2011	36	Block-structured BPMN	+	-	-	+	±	-	+	-	-	+	+	+	+	±
	Gröner et al.	2013	22	Block-structured BPMN	±	-	-	+	-	+	-	+	±	-	-	±	+	±

are the only approaches that fully support customization of executable models (in YAWL, BPEL, and SAP WebFlow) down to the level of producing models that can be deployed in a BPMS. One can hypothesize that the observed emphasis on conceptual process modeling stems from the fact that variability in executable process models is usually tackled via runtime customization [Reichert and Weber 2012] rather than design-time customization (see Section 2).

All but one approach (vBPMN) support customization by restriction, while only a minority of approaches support customization by extension (eight out of 23). There appears to be a trade-off between supporting customization by extension and preserving correctness. Indeed, approaches that support customization by extension do not support correctness, except for Template and Rules and vBPMN, which support correctness at the expense of imposing constraints on the structure of the customizable model and allowed extensions, namely, that they both must be block-structured.[9] This observation highlights the fact that, in order to reconcile customization by extension and correctness support, it is necessary to constrain the allowed extensions and the places in the customizable process model where these extensions can be inserted.

CoSeNet also achieves correctness support at the expense of structural constraints on the customizable process models (block-structured). On the other hand, C-iEPCs and Configurable Workflows achieve both structural and behavioral correctness without imposing structural constraints. This is achieved via incremental checks that detect combinations of customization options leading to incorrect models. However, these approaches allow customization only by restriction in line with the earlier observation.

The majority of approaches support customization based on domain models (i.e., abstraction), which may take the form of predicates over domain properties (as in

[9]Configurative Process Modeling partially supports structural correctness only when customizing the model by restriction. Customization by extension in this approach does not guarantee correctness.

Configurative Process Modeling and Provop), feature models, questionnaire models, or decision tables, as discussed in Appendix C. On the other hand, only two approaches (C-iEPCs and Configurable Workflows) provide step-by-step guidance to make customization decisions while avoiding inconsistent or irrelevant decisions. The approach by Gröner et al. [2013] does not provide step-by-step guidance, but prevents inconsistencies between decisions made during customization.

It can be observed that the majority of approaches have tool implementations, at least partially, and about half, of the approaches are fully or partly formalized. Also, about half of the approaches have been validated at least partially using real-life scenarios, although, in many cases, the validation has not involved domain experts. Overall, these observations highlight the relative maturity of the field.

C-iEPCs and Templates and Rules come close to supporting all the criteria. C-iEPCs focus on customization by restriction in conceptual process models. Templates and Rules covers both conceptual and executable models but leaves aside the issue of providing customization guidance. These approaches demonstrate that the identified criteria are rather orthogonal, meaning that it is possible to support all of them. The only partial trade-off is the one between supporting customization by extension and supporting correctness preservation. This trade-off, however, is not necessarily unsurmountable. One can conceive of approaches that achieve correctness preservation while supporting customization by extension by setting boundaries on the way that the customizable process model can be extended; for example, only certain predefined templates can be employed and these templates are defined in a way such that behavioral correctness is preserved.

10.2. Discussion on Research Questions

This comparative analysis provides a basis to answer the research questions formulated in Section 1. With respect to RQ1, the previous discussion puts into evidence a number of core elements shared across all approaches. All approaches take as the starting point a host process modeling language—usually a conceptual one rather than an executable one—on top of which a notion of *variation point* is added. Variation points are associated with specific model elements, which usually are control-flow elements (activities or gateways) but in some approaches can also be resources and objects.

On top of this common core, shared by all customizable process modeling approaches, we observe three key differentiating features. First, some approaches allow variation points in a customizable model to be linked to elements in a domain model in order to assist the user during the customization of the model. Second, some approaches ensure that the customized models are structurally and behaviorally correct, disallowing combinations of customization options that would lead to an incorrect model. In three of the surveyed approaches, correctness is ensured at the expense of constraints on the structure of the models (block-structuredness), but in other cases, it is ensured for models with arbitrary topology. A third differentiating feature is given by the dichotomy between customization by restriction versus extension. While support for the former is widespread, the latter is only supported by a handful of approaches.

These distinguishing features constitute possible criteria for selecting an approach for a given purpose (see RQ2). If the set of variants of a given process is expected to grow incrementally after initial creation of a customizable process model, customization by extension is more convenient from a maintenance perspective. In this case, one starts with a customizable process model capturing the known variants. When a new variant is identified, its additional behavior with respect to the existing customizable model can be added as an extension point if it is well confined. In approaches that only support customization by restriction, each new variant requires one to update the customizable

process model because the customizable model captures the union of all variants. In contrast, when configuration by extension is used, the customizable process model may capture only a core subset of the behavior of the variants. Variant-specific behavior can be captured in the extension points.

Meanwhile, if the decisions required for customization are complex and interdependent, approaches that link the customizable process model to a domain model and that provide customization guidance are preferable. If, in addition, the customizable process model is large and complex, approaches that support correctness checking during configuration may prove most useful. In this respect, it is not surprising that approaches based on customization by restriction tend to put emphasis on correctness-checking and guidance. Indeed, as the customizable model becomes larger, updates to it become more error-prone, and approaches based on customization by restriction lead to larger models since these models need to capture the union of all variants.

With reference to RQ3, we note that only a handful of approaches offer guidance and iterative feedback to the user in the selection of customization options. The few approaches that offer such guidance focus on ensuring that each selected customization option satisfies the domain constraints or that the customized process model is correct. However, they do not address the question of which option (among those that are feasible) can lead to a customized process model with better performance with respect to relevant process performance measures. In other words, the relation between customization and business process performance has been neglected so far.

Second, while it can be observed that about half of the approaches have been implemented and at least partially validated in one way or another, there is a relative scarcity of comparative empirical evaluations. Barring two comparative studies [Torres et al. 2013; Döhring et al. 2014] focusing on a couple of approaches, there is a lack of evidence to back any statement that one customizable process modeling approach is more usable than others in a particular setting. This lack of comparative evaluations is arguably a major gap in the state of the art.

Third, we observe a lack of discussion on the question of how to construct a customizable process model in the first place and how to maintain this artifact over time. It is generally assumed that a modeler will manually design the customizable process model using techniques similar to those employed to design classical (noncustomizable) process models. Yet, given that a customizable process model represents an entire family of processes, the amount of information required to design such a model is usually an order of magnitude larger than that required to design a model of one singular process. This observation calls for the development of methods to assist process modelers during the design and update of customizable models.

Initial research on the design of customizable process models has led to algorithms for constructing a customizable process model from a collection of separate models of process variants [La Rosa et al. 2013; Assy et al. 2014] as well as algorithms for constructing a customizable process model from event logs extracted from enterprise systems [Buijs et al. 2013; Ekanayake et al. 2013]. Another approach is to extract a "common" (reference) process model out of a collection of models of process variants [Li et al. 2009]. This reference model may be particularly suited as a starting point for "customization by extension" approaches, since the reference model captures the "greatest common denominator" between existing variants and the variant-specific behavior can then be added via extension points.

11. RELATED WORK

Ayora et al. [2015] conducted a systematic literature review to evaluate existing variability support across all stages of the business process life cycle. The authors considered 63 primary studies based on eight research questions (such as underlying

business process modeling language used, tools available for enabling process variability, and validation of methods proposed). Based on their findings, they developed the VIVACE framework to enable process engineers to evaluate existing process variability approaches. They then evaluated three approaches in depth using their framework: C-EPCs, Provop, and PESOA. Our survey differs, as we focus on the customization (configuration) of process models rather than approaches dealing with one or several phases of the life cycle. Also, our comparison covers a superset of these approaches.

Valenca et al. [2013] presented a literature mapping study in the field of business process variability covering 80 publications. Their objectives were to identify characteristics of business process variability (including design-time and runtime variability), to identify available approaches for business process variability management, and to identify challenges in this field. In line with the nature of literature mapping studies, their study lists and categorizes a wide range of approaches but without analyzing and comparing them in detail. In contrast, the present survey describes each main approach in detail, applies it to an example, and includes a comparative analysis. Further, our search returned a superset of the publications in Valenca et al. [2013]. The latter remark also applies to a shorter and less systematic survey by Mechrez and Reinhartz-Berger [2014].

A number of systematic reviews within the related domain of Software Product Line Engineering (SPLE) have been conducted. For instance, Chen et al. [2009] conducted a systematic review of variability management in SPLE that included 33 papers. Their purpose was to provide an overview of different aspects of variability management approaches, such as scalability and product derivation. Another is by dos Santos Rocha and Fantinato [2013]. They conducted a systematic literature review to assess Software Product Line (SPL) approaches for BPM. Having reviewed 63 papers, they conclude that SPL approaches for BPM, while gaining maturity, are still at an inadequate level. Benavides et al. [2010] conducted a comprehensive literature review covering 53 papers to investigate existing proposals of automated analysis of feature models (within the context of SPLE). Finally, Chen and Babar [2011] performed a systematic literature review that resulted in 97 papers being closely examined for assessing the status of evaluation of variability management approaches within SPLE. These reviews had a different objective; as such, they cover other aspects of variability management as compared to our survey, namely, understandability and maturity of evaluation. While they all contribute valuable insights, they share the commonality of being focused on the domain of SPLE. Furthermore, Chen et al. [2009] and Chen and Babar [2011] considered variability management within SPLE but dos Santos Rocha and Fantinato [2013] and Benavides et al. [2010] did not focus on variability management in particular. Our survey distinguishes itself from these surveys in that it focuses on variability management and it focuses on business processes as the artifact for which variability is captured and exploited. The distinctness of our survey with respect to these surveys is confirmed by the fact that the overlap of primary study papers is limited to a maximum of five papers – the overlap between dos Santos Rocha and Fantinato [2013] and ours is 5, the overlap between Chen et al. [2009] and ours is 2, and for Benavides et al. [2010], the overlap is 1.

Finally, Torres et al. [2013] and Döhring et al. [2014] compared a subset of the approaches reviewed in the present survey using different evaluation lenses. Torres et al. [2013] compared C-EPCs and Provop in terms of understandability based on a cognitive psychology framework. Döhring et al. [2014] conducted an empirical evaluation to assess the maintainability of process model variants in C-YAWL versus vBPMN in terms of modularization support and customization type (i.e., restriction vs. restriction + extension). These papers examine nonfunctional aspects not considered in our survey and, as such, they are complementary.

Mili et al. [2010] survey, categorize, and summarize different modeling languages used to describe business processes, covering in particular the languages figuring in the "Process modeling language" column of Table XII (e.g., BPMN, UML ADs). Their survey, however, does not touch on the question of how to capture process variability in general and design-time variability in particular. As such, the scope of the present survey is disjoint and complementary to the one by Mili et al. [2010].

12. CONCLUSION

This survey has put into evidence a wide heterogeneity of features and levels of sophistication across existing approaches to customizable process models. Still, the survey has highlighted a common core shared by all of them and key differentiating features.

All approaches take as a starting point a host process modeling language and add to it a notion of variation point. A variation point may be a modeling element that appears, does not appear, or appears in one of multiple possible forms (customization by restriction) or a point in the process when additional behavior is allowed (configuration by extension). While virtually all approaches support customization by restriction, only a handful support customization by extension. Support for the latter constitutes one of the key differentiating features across the surveyed approaches and gives rise to a fundamental trade-off that potential users need to consider when selecting an approach for a given scenario.

On the one hand, customization by extension is more suitable from a maintenance perspective. It allows one to start with a model capturing a core set of variants of the process. The behavior of additional variants can then be added via extension points, especially if the additional behavior of these variants can be confined to specific points in the customizable process model. The price to pay, however, is that the customizable process model itself captures only a subset of the behavior of the variants – additional behavior remains somehow hidden behind the extension points.

On the other hand, customization by restriction is more suitable when the set of variants is stable since every new variant or every change to an existing variant requires an update to the customizable process model. Additionally, this approach leads to larger models since the customizable model has to capture the union of all variants. The latter hinders maintainability. Still, the models used in customization by restriction give a full picture of all variants and their dependencies. Also, because the behavior of all variants is captured in the customizable model, customization by restriction lends itself better to correctness checking, something that can only be achieved in customization by extension at the price of constraining the set of allowed extensions and the places in the customizable process model where these extensions can be inserted.

Despite the breadth of literature in the field of customizable process modeling, we have noted two areas that remain underdeveloped. First, there is a lack of effective methods and tool support to assist users in the creation, use (in particular, customization), and maintenance of these models. Surprisingly, there has been little research on these questions. A handful of automated approaches to constructing a customizable process model from a collection of variants have been proposed [Li et al. 2009; La Rosa et al. 2013; Assy et al. 2014]. However, the user is then left with the task of fine-tuning these models, linking them with domain models and subsequently maintaining them. In a similar vein, little attention has been given to the question of how to guide users during the customization of customizable process models. The lack of methods and tools to support the full life cycle of customizable process models (creation, use, and maintenance) might explain the very limited adoption of customizable process modeling languages in practice.[10]

[10]To the best of our knowledge, only the aEPCs approach is currently used in practice by NN Investment Partners to manage their process model collection, which counts some 500 models.

Second, while about half of reviewed approaches have been validated (typically, via case studies), there is a lack of comparative empirical evaluations with end users that would provide evidence to back any statement that one customizable process modeling approach is more usable than others in a particular setting. There is a case for shifting the focus in this field from the design of modeling approaches to the evaluation of existing ones [Torres et al. 2013; Döhring et al. 2014].

Looking forward, the widespread adoption of multitenant enterprise systems has opened the possibility of using customizable process models to drive the configuration of such systems. At present, the configuration of multitenant systems is manual and resource-intensive due to the large number of configuration points offered by such systems. Initial visions for multitenant system configuration based on customizable process models have been put forward [Fehling et al. 2011; van der Aalst 2011]. However, the realization and validation of these visions remain avenues for future research.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

We thank Arthur ter Hofstede for his valuable comments on early versions of this manuscript.

REFERENCES

Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. 2010b. Composing Feature Models. In *Proceedings of SLE 2009*, M. van den Brand, D. Gašević, and J. Gray (Eds.), Lecture Notes in Computer Science, Vol. 5969. Springer, 62–81.

Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. 2010a. Managing Variability in Workflow: Managing Variability in Workflow with Feature Model Composition Operators. In *9th International Conference on Software Composition (SC'10), Malaga, Spain*. Springer, 17–33.

Nour Assy, Walid Gaaloul, and Bruno Defude. 2014. Mining Configurable Process Fragments for Business Process Design. In *Proceedings of DESRIST*, Lecture Notes in Computer Science, Vol. 8463. Springer, 209–224.

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

Felix Bachmann and Paul C. Clements. 2005. *Variability in Software Product Lines*. Technical report CMU/SEI-2005-TR-012. Carnegie Mellon University, U.S.A.

T. Baier, E. Pascalau, and J. Mendling. 2010. On the Suitability of Aggregated and Configurable Business Process Models. In *Enterprise, Business-Process and Information Systems Modeling—11th International Workshop, (BPMDS'10), and 15th International Conference, EMMSAD 2010, held at CAiSE 2010, Hammamet, Tunisia, Proceedings*, Lecture Notes in Business Information Processing, Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, and Roland Ukor (Eds.), Vol. 50. 108–119.

J. Becker, L. Algermissen, P. Delfmann, and B. Niehaves. 2006. Configurable Reference Process Models for Public Administrations. In *Encyclopedia of Digital Government*. 220–223.

J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuropka. 2004. Configurative Process Modeling–Outlining an Approach to Increased Business Process Model Usability. In *Proceedings of the 14th Information Resources Management Association International Conference*, M. Khosrow-Pour (Ed.). IRM Press.

J. Becker, P. Delfmann, and R. Knackstedt. 2007. Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In *Proceedings of the Reference Modeling Conference (RM'06)*, J. Becker and P. Delfmann (Eds.). Springer, 27–58.

J. Becker, C. Janiesch, R. Knackstedt, and T. Rieke. 2007a. Facilitating Change Management with Configurative Reference Modelling. *International Journal for Information Systems and Change Management* 2, 1 (2007), 81–99.

J. Becker, R. Knackstedt, D. Pfeiffer, and C. Janiesch. 2007b. Configurative Method Engineering—On the Applicability of Reference Modeling Mechanisms in Method Engineering. In *Proceedings of the 13th Americas Conference on Information Systems (AMCIS'07)*. 1–12.

D. Benavides, S. Segura, and A. Ruiz-Corts. 2010. Automated analysis of feature models 20 years later: A literature review. *Information Systems* 35, 6, 616–636.

Quentin Boucher, Gilles Perrouin, Jean-Christophe Deprez, and Patrick Heymans. 2012. Towards Configurable ISO/IEC 29110-Compliant Software Development Processes for Very Small Entities. In *Proceedings of the 19th European Conference "Systems, Software and Services Process Improvement" (EuroSPI'12)*, Communications in Computer and Information Science, Dietmar Winkler, Rory V. O'Connor, and Richard Messnarz (Eds.), Vol. 301. Springer, 169–180.

Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. 2013. Mining Configurable Process Models from Collections of Event Logs. In *BPM*, Lecture Notes in Computer Science, Vol. 8094. Springer-Verlag, Berlin, 33–48.

L. Chen and M. A. Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology* 53, 4, 344–362.

L. Chen, M. A. Babar, and N. Alio. 2009. Variability management in software product lines: a systematic review. 7328, 190–205.

K. Czarnecki and M. Antkiewicz. 2005. Mapping Features to Models: A Template Approach Based on Superimposed Variants. In *Proceedings of the 4th International Conference on Generative Programming and Component Engineering*, Robert Glück and Michael R. Lowry (Eds.). Springer, 422–437.

K. Czarnecki, S. Helsen, and U. W. Eisenecker. 2005. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice* 10, 1, 7–29.

K. Czarnecki and K. Pietroszek. 2006. Verifying feature-based model templates against well-formedness OCL constraints. In *Proceedings of Generative Programming and Component Engineering*. ACM, 211–220. DOI:http://dx.doi.org/10.1145/1173706.1173738

R. B. Davis and E. Brabander. 2007. *ARIS Design Platform: Getting Started with BPM*. Springer, New York, NY.

P. Delfmann, C. Janiesch, R. Knackstedt, T. Rieke, and S. Seidel. 2006. Towards Tool Support for Configurative Reference Modeling – Experiences from a Meta Modeling Teaching Case. In *Proceedings of the 2nd International Workshop on Meta-Modelling (WoMM'06) (LNI)*, S. Brockmans, J. Jung, and Y. Sure (Eds.), Vol. 96. GI, 61–83.

P. Delfmann, T. Rieke, and C. Seel. 2007. Supporting Enterprise Systems Introduction by Controlling Enabled Configurative Reference Modelling. In *Proceedings of the Reference Modeling Conference (RM'06)*, J. Becker and P. Delfmann (Eds.). Springer, 79–102.

M. Döhring, H. A. Reijers, and S. Smirnov. 2014. Configuration vs. adaptation for business process variant maintenance: an empirical study. *Information Systems* 39, 108–133.

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

A. Dreiling, M. Rosemann, W. M. P. van der Aalst, L. Heuser, and K. Schulz. 2006. Model-Based Software Configuration: Patterns and Languages. *European Journal of Information Systems* 15, 6, 583–600.

A. Dreiling, M. Rosemann, W. M. P. van der Aalst, W. Sadiq, and S. Khan. 2005. Model-Driven Process Configuration of Enterprise Systems. In *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*, O. K. Ferstl, E. J. Sinz, S. Eckert, and T. Isselhorst (Eds.). Physica–Verlag, 687–706.

C. C. Ekanayake, M. Dumas, L. García-Bañuelos, and M. La Rosa. 2013. Slice, Mine and Dice: Complexity-Aware Automated Discovery of Business Process Models. In *Business Process Management*, *Lecture Notes in Computer Science*, Lecture Notes in Computer Science, Vol. 8094. Springer, 49–64.

Christoph Fehling, Frank Leymann, David Schumm, Ralf Konrad, Ralph Mietzner, and Michael Pauly. 2011. Flexible Process-Based Applications in Hybrid Clouds. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD'11)*. IEEE, 81–88.

P. Fettke and P. Loos. 2003. Classification of Reference Models—A Methodology and its Application. *Information Systems and e-Business Management* 1, 1, 35–53.

D. Georgakopoulos, M. Hornick, and A. Sheth. 1995. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases* 3, 119–153.

F. Gottschalk, W. M. P. van der Aalst, and M. H. Jansen-Vullers. 2007. SAP WebFlow Made Configurable: Unifying Workflow Templates into a Configurable Model. In *International Conference on Business Process Management (BPM'07)*, Lecture Notes in Computer Science, G. Alonso, P. Dadam, and M. Rosemann (Eds.), Vol. 4714. Springer, Berlin, 262–270.

F. Gottschalk, W. M. P. van der Aalst, M. H. Jansen-Vullers, and M. La Rosa. 2008. Configurable Workflow Models. *International Journal of Cooperative Information Systems* 17, 2, 177–221.

F. Gottschalk, T. Wagemakers, M. H. Jansen-Vullers, W. M. P. van der Aalst, and M. La Rosa. 2009. Configurable Process Models: Experiences From a Municipality Case Study. In *Proceedings of CAiSE*, Lecture

Notes in Computer Science, P. van Eck, J. Gordijn, and R. Wieringa (Eds.), Vol. 5565. Springer, Berlin, 486–500.

A. Hallerbach, T. Bauer, and M. Reichert. 2008. Managing Process Variants in the Process Life Cycle. In *10th International Conference on Enterprise Information Systems (ICEIS'08)*, Vol. 22. 154–161.

A. Hallerbach, T. Bauer, and M. Reichert. 2009a. Guaranteeing Soundness of Configurable Process Variants in Provop. In *CEC*. IEEE, 98–105.

A. Hallerbach, T. Bauer, and M. Reichert. 2009b. Issues in Modeling Process Variants with Provop. In *Business Process Management 2008 Workshops*, Lecture Notes in Business Information Processing, D. Ardagna, M. Mecella, and J. Yang (Eds.), Vol. 17. Springer, Berlin.

A. Hallerbach, T. Bauer, and M. Reichert. 2010. Capturing Variability in Business Process Models: The Provop Approach. *Journal of Software Maintenance and Evolution: Research and Practice* 22, 6–7, 519–546.

A. R. Hevner, S. T. March, J. Park, and S. Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28, 1, 75–105.

A. Kumar and W. Yao. 2009. Process Materialization Using Templates and Rules to Design Flexible Process Models. In *RuleML*, Lecture Notes in Computer Science, G. Governatori, J. Hall, and A. Paschke (Eds.), Vol. 5858. Springer, Berlin, 122–136.

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

M. La Rosa, M. Dumas, A. H. M. ter Hofstede, and J. Mendling. 2011. Configurable Multi-Perspective Business Process Models. *Information Systems* 36, 2, 313–340.

Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. 2013. Business Process Model Merging: An Approach to Business Process Consolidation. *ACM Transactions on Software Engineering Methodology* 22, 2, 11.

C. Li, M. Reichert, and A. Wombacher. 2009. Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. In *Business Process Management (BPM'09)*, Lecture Notes in Computer Science, U. Dayal, J. Eder, J. Koehler, and H. Reijers (Eds.), Vol. 5701. Springer, Berlin, 344–362.

Carl-Mikael Lönn, Elin Uppström, Petia Wohed, and Gustaf Juell-Skielse. 2012. Configurable Process Models for the Swedish Public Sector. In *Proceedings of the 24h International Conference on Advanced Information Systems Engineering (CAiSE'12)*, Lecture Notes in Computer Science, Jolita Ralyté, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza (Eds.), Vol. 7328. Springer, 190–205.

I. Mechrez and I. Reinhartz-Berger. 2014. Modeling Design-Time Variability in Business Processes: Existing Support and Deficiencies. In *Proceedings of BPMDS'14 and EMMSAD'14*. Springer, 378–392.

Hafedh Mili, Guy Tremblay, Guitta Bou Jaoude, Eric Lefebvre, Lamia Elabed, and Ghizlane El-Boussaidi. 2010. Business process modeling languages: Sorting through the alphabet soup. *Computing Surveys* 43, 1, Article 4.

Mikyeong Moon, Minwoo Hong, and Keunhyuk Yeom. 2008. Two-Level Variability Analysis for Business Process with Reusability and Extensibility. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC'08), Turku, Finland*. IEEE Computer Society, 263–270.

M. Pesic, H. Schonenberg, and W. M. P. van der Aalst. 2007. DECLARE: Full Support for Loosely-Structured Processes. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC'07)*, M. Spies and M. B. Blake (Eds.). IEEE Computer Society, 287–298.

F. Puhlmann, A. Schnieders, J. Weiland, and M. Weske. 2005. *Variability Mechanisms for Process Models*. PESOA-Report TR 17/2005. Process Family Engineering in Service-Oriented Applications (PESOA). BMBF-Project.

M. Reichert and P. Dadam. 1998. ADEPT$_{flex}$: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems* 10, 2, 93–129.

M. Reichert and B. Weber. 2012. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer-Verlag, Berlin.

H. A. Reijers, R. S. Mans, and R. A. van der Toorn. 2009. Improved Model Management with Aggregated Business Process Models. *Data and Knowledge Engineering* 68, 2, 221–243.

I. Reinhartz-Berger, P. Soffer, and A. Sturm. 2009. Organizational Reference Models: Supporting an Adequate Design of Local Business Processes. *International Journal of Business Process Integration and Management* 4, 2 (2009), 134–149.

I. Reinhartz-Berger, P. Soffer, and A. Sturm. 2010. Extending the Adaptability of Reference Models. *IEEE Transactions on Systems, Man and Cybernetics Part A* 40, 5, 1045–1056.

I. Reinhartz-Berger and A. Sturm. 2007. Enhancing UML Models: A Domain Analysis Approach. *Journal on Database Management (special issue on UML Topics)* 19, 1, 74–94.

S. Rinderle, M. Reichert, and P. Dadam. 2004. Correctness Criteria For Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering* 50, 1, 9–34.

M. Rosemann. 2003. Application Reference Models and Building Blocks for Management and Control (ERP Systems). In *Handbook on Enterprise Architecture*, P. Bernus, L. Nemes, and G. Schmidt (Eds.). Springer, 596–616.

M. Rosemann and W. M. P. van der Aalst. 2003. A Configurable Reference Modelling Language. BPM Center Report BPM-03-08, BPMcenter.org. (Later published as M. Rosemann and W. Aalst. 2007. A Configurable Reference Modelling Language. Information Systems, Vol. 32(1), 1-23).

S. Sadiq, M. E. Orlowska, and W. Sadiq. 2005. Specification and validation of process constraints for flexible workflows. *Information Systems* 30, 5, 349–378.

S. Sadiq, W. Sadiq, and M. Orlowska. 2001. Pockets of Flexibility in Workflow Specification. In *Proceedings of the 20th International Conference on Conceptual Modeling (ER'01)*, Lecture Notes in Computer Science, Vol. 2224. Springer, Berlin, 513–526.

A. Schnieders. 2006. Variability Mechanism Centric Process Family Architectures. In *Proceedings of the 13th IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06)*, M. Riebisch, P. Tabeling, and W. Zorn (Eds.). IEEE Computer Society, 289–298.

A. Schnieders and F. Puhlmann. 2006. Variability Mechanisms in E-Business Process Families. In *Proceedings of the 9th International Conference on Business Information Systems (BIS'06) (LNI)*, W. Abramowicz and H.C. Mayr (Eds.), Vol. 85. GI, 583–601.

Mikael Svahnberg, Jilles van Gurp, and Jan Bosch. 2005. A taxonomy of variability realization techniques. *Software Practice and Experience* 35, 8, 705–754. DOI: http://dx.doi.org/10.1002/spe.652

V. Torres, S. Zugal, B. Weber, M. Reichert, C. Ayora, and V. Pelechano. 2013. A Qualitative Comparison of Approaches Supporting Business Process Variability. In *Business Process Management Workshops*, *Lecture Notes in Business Information Processing*, Vol. 132. Springer, Berlin, 560–572.

G. Valenca, C. Alves, V. Alves, and N. Niu. 2013. A Systematic Mapping Study on Business Process Variability. *International Journal of Computer Science & Information Technology* 5, 1, 1–21.

W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M. H. Jansen-Vullers. 2006. Configurable Process Models as a Basis for Reference Modeling. In *Proceedings of the Business Process Management 2005 Workshops*, E. Kindler and M. Nüttgens (Eds.). Springer, 76–82.

W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve, and M. T. Wynn. 2011. Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing* 23, 3, 333–363.

W. M. P. van der Aalst, N. Lohmann, and M. La Rosa. 2012. Correctness Ensuring Process Configuration: An Approach Based on Partner Synthesis. *Information Systems* 37, 6, 574–592.

Wil M. P. van der Aalst. 2011. Business Process Configuration in the Cloud: How to Support and Analyze Multi-tenant Processes? In *Proceedings of the 9th IEEE European Conference on Web Services (ECOWS'11)*. IEEE, 3–10.

B. Weber, M. Reichert, and S. Rinderle-Ma. 2008. Change Patterns and Change Support Features: Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering* 66, 3, 438–466.

Received October 2015; revised November 2016; accepted December 2016