Feature-Oriented Composition of Declarative Artifact-Centric Process Models

Rik Eshuis(✉)

Eindhoven University of Technology,
P.O. Box 516, 5600 MB Eindhoven, The Netherlands
h.eshuis@tue.nl

Abstract. Declarative business process models that are centered around artifacts, which represent key business entities, have proven useful to specify knowledge-intensive processes. Currently, declarative artifact-centric process models need to be designed from scratch, even though existing model fragments could be reused to gain efficiency in designing and maintaining such models. To address this problem, this paper proposes an approach for composing model fragments, abstracted into features, into fully specified declarative artifact-centric process models. We use Guard-Stage-Milestone (GSM) schemas as modeling language and let each feature denote a GSM schema fragment. The approach supports feature composition at different levels of granularity. Correctness criteria are defined that guarantee that valid GSM schemas are derived. The approach is evaluated by applying it to an industrial process. Using the approach, declarative artifact-centric process models can be composed from existing model fragments in an efficient and correct way.

1 Introduction

In many business processes, data items are being processed in a non-routine way by knowledge workers. Such data-driven processes are knowledge-intensive [10], since expert knowledge is required to make progress for individual cases. Application domains are case management [1,28,29] and emergency management [10]. *Artifact-centric process models* are a natural fit to model data-driven processes [8,21]. An artifact represents a real-world business entity, such as Product or Order, about which data is processed and for which activities are performed.

Data-driven business processes are much more unpredictable than traditional activity-centric processes. The structure of a data-driven process model depends on the case data that is being processed [10]. Knowledge workers play a key role in determining this structure while performing the process. Given their expertise, they have a lot of freedom in deciding how to process individual cases. This suits a declarative specification of processes [15], in which workers have more freedom in performing processes than with a procedural specification. Several declarative process modeling languages have been proposed in the past years [15], including the Guard-Stage-Milestone (GSM) language for artifacts [9,17].

M. Weske et al. (Eds.): BPM 2018, LNCS 11080, pp. 66–82, 2018.
https://doi.org/10.1007/978-3-319-98648-7_5

Designing declarative artifact-centric process models is currently done in an inefficient way. Such models need to be developed from scratch, even though they could be defined by reusing fragments in existing models. Since similar fragments need to be specified again and again for new artifact-centric process models, the design process is inefficient. Also, maintenance of the different process models is inefficient, since each occurrence of a fragment across different process models needs to be maintained separately, so updates need to be replicated.

To address this problem, this paper proposes a *feature-oriented approach for composing declarative artifact-centric process models*. A feature denotes a fragment of a declarative artifact-centric process model. The approach formally defines how to compose features into declarative artifact-centric process models. From the same set of features, different artifact-centric process models can be composed that share common fragments. The approach supports reuse of those common fragments across different declarative artifact-centric process models.

We use Guard-Stage-Milestone (GSM) [17] as declarative modeling language for business artifacts. GSM is well-defined [9], thus providing a sound basis for the approach. Next, its core modeling constructs have been adopted in the OMG Case Management Model and Notation (CMMN) [7,21]. Therefore, the results can provide a stepping stone towards managing variability for CMMN models.

The approach is inspired by feature-oriented design in software product lines [2,23]. Features are used to distinguish common and variable parts in software artifacts [2] and this way support reuse of software artifacts [5]. Similar to feature-oriented composition for software product lines, the approach supports composition at different levels of granularity [3]. Key difference is that the approach considers the syntactic and semantic level of business artifacts, whereas software product lines only consider the syntactic level of software artifacts [2,23]. Section 7 discusses related work, also from the area of BPM, in more detail.

The paper is organized as follows. Section 2 presents a running example and also introduces GSM schemas [9], which Sect. 3 formally defines. Section 4 defines the feature composition approach while Sect. 5 presents correctness criteria that guarantee that the compositions are valid GSM schemas. Section 6 evaluates the feasibility and potential gain of the approach by applying it to an industrial process that has several variants. Section 7 discusses related work. Section 8 ends the paper with conclusions.

2 Motivating Example

To motivate the use of the feature-oriented composition approach and to informally introduce GSM schemas, we first present an example. It is based on a real-world process from an international high tech company. The example is revisited in Sects. 3 and 4 to illustrate key definitions.

In the process, business criteria for a partner contract are assessed: first data about the partner is gathered and pre-checked, and next a detailed check is performed to decide whether the criteria should be changed or not. If new

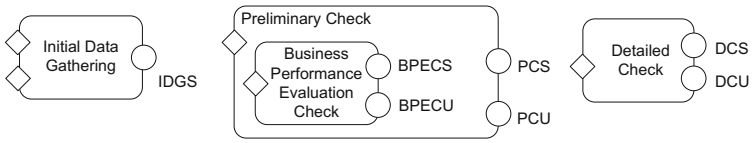

Fig. 1. Base GSM schema Business Criteria Assessment (BCA_{base})

Table 1. Stages and sentries for BCA_{base} in Fig. 1. ';' separates different sentries

Stage	Plus sentries (guards)	Minus sentries (closing)
Initial Data Gathering	E:StartAssessment ; E:AdditionalInfo	IDGS
Preliminary Check	IDGS	PCS ; PCU ; E:AdditionalInfo
Business Performance Evaluation Check	+Preliminary Check	BPECS ; BPECU ; -Preliminary Check
Detailed Check	PCS	DCS ; DCU

information arrives before the business criteria have been assessed, the data is gathered anew and the business criteria check is restarted, if applicable. Figure 1 shows a graphical representation of a GSM schema that models this process. Each rounded rectangle represents a stage, in which work is performed. Each circle represent a milestone, which is a business objective. A milestone is typically achieved by completing the work in a stage, represented by putting the milestone on the right border of the stage. The status of each stage and milestone is represented by a Boolean attribute, which is true (false) if the stage is open (closed) or the milestone is achieved (invalid). Stages and milestones change status if certain conditions, called *sentries*, are met (Tables 1 and 2). Each stage has plus sentries, called guards, that specify when it is opened and minus sentries that specify when it is closed, which could be never (false). Each guard of a stage is visualized as a diamond. Each milestone has plus (minus) sentries that specify when it is achieved (invalidated). Sentries can refer to external or internal events. External events are named events (prefix E:) or completion events of atomic stages (prefix C:). Internal events denote status changes of stages and milestones; a status can become true (prefix +) or false (prefix -). Besides status changes, sentries can refer to the statuses of stages and milestones; for instance, the minus sentry of stage Initial Data Gathering closes the stage if the status of milestone IDGS is true (achieved). Section 3 gives more details on GSM schemas.

The company has offices in different geographic regions, each of which has its own flavor of the process. This results, among others, in three basic variations for the preliminary check in this process (see Table 3). These variations can be combined in a concrete variant, yielding in total eight variants, one of which is the base process in Fig. 1, in which none of the variations has been chosen.

Table 2. Milestones and sentries for BCA_{base} in Fig. 1

Milestone	Full name	Plus sentries (achieving)	Minus sentries (invalidating)
IDGS	Initial Data Gathering Successful	C:Initial Data Gathering	E:AdditionalInfo
BPECS	Business Performance Evaluation Check Successful	C:Business Performance Evaluation Check $\wedge$ BP_good	E:AdditionalInfo
BPECU	Business Performance Evaluation Check Unsuccessful	C:Business Performance Evaluation Check $\wedge\neg$BP_good	E:AdditionalInfo
PCS	Pre-checks Successful	BPECS	false
PCU	Pre-checks Unsuccessful	BPECU	false
DCS	Detailed Check Successful	C:Detailed Check $\wedge\ldots$	false
DCU	Detailed Check Unsuccessful	C:Detailed Check $\wedge\ldots$	false

Table 3. Variations of BCA_{base}

1	Business performance is checked if the company has more than 300 employees
2	The credit is checked as part of the preliminary check
3	The addressable market is checked as part of the preliminary check

Specifying these eight variants in separate process models leads to redundancy, both regarding the models and the modeling process. In that case modifying one common element like milestone PCS in those eight variants has to be done eight times rather than once. Moreover, drawing each of the eight models has to be done from scratch, since there is no reuse.

To address these problems, we develop an approach based on the technique of feature-oriented composition, well known from Software Product Lines [2]. The same feature can be reused in different compositions. In this paper, a feature denotes a GSM schema, which can be either fully or partially defined (a schema fragment). The GSM schema fragments for variations 1 and 2 in Table 3 are presented and discussed in Sect. 4.

A key challenge is how to compose features that denote GSM schema fragments. In Sect. 4, we define a binary feature composition operator '$\bullet$' that supports composition at different levels of granularity [2]. A coarse-grained feature composition adds new stages and milestones to a GSM schema. A fine-grained feature composition modifies sentries of existing stages or milestones. Feature composition for GSM schemas requires both granularity levels to be of practical value. For instance, for the variations in Table 3, the coarse-grained feature for variation 2 (defined in Sect. 4) specifies extra work, so it needs to introduce additional stage and milestones to the base schema, while the fine-grained feature for

variation 1 (also defined in Sect. 4) needs to modify the plus sentries of Business Performance Evaluation Check and PCS in the base schema.

Designing a new GSM schema variant based on this approach comes down to selecting the relevant features and defining their composition order. The GSM schema variant is then automatically derived by composing the GSM schema fragments corresponding to the selected features in the defined order. This way, the approach shields designers from the complexity of designing GSM schemas and allows them to quickly generate different variants from a set of fragments. Maintaining the variants is efficient, since an update to a shared fragment can be propagated automatically to all the affected variants by recomposing them.

3 GSM Schemas

Syntax. A GSM schema [9] consists of data attributes and status attributes. A status attribute is a Boolean variable that denotes the status of a stage or milestone. For a status attribute of a stage, value *true* denotes that the stage is open, value *false* that the stage is closed. For a status attribute of a milestone, value *true* denotes that the milestone is achieved, value *false* that the milestone is invalid.

Event-Condition-Action rules define for which event under which condition a status attribute changes value (action). The event-condition part of a rule is called a *sentry*. The event of a sentry is optional. We distinguish between external and internal events. An external event signifies a change in the environment. It is either a stage completion event C:S, where S is an atomic stage, as defined below, or a named external event E:n, where n is an event name. An internal event signifies a change in value of a status attribute a: internal event $+a$ denotes that a becomes true, $-a$ that a becomes false. For instance, +Preliminary Check in Table 1 is an internal event that signifies that stage Preliminary Check gets opened. The condition of a sentry is a Boolean expression that can refer to data attributes or status attributes. The action of each rule is that a status attribute becomes true or false. Given these two distinct actions, we distinguish between two types of sentries. A *plus sentry* defines when a stage becomes open or a milestone gets achieved. A *minus sentry* defines when a stage is closed or a milestone gets invalid.

Stages and milestones can be nested inside other stages. A milestone cannot contain any other milestone or stage. We require that the nesting relation induces a forest, i.e., the nesting relation is acyclic and if a stage or milestone is nested in two other stages S_1, S_2, then either S_1 is nested in S_2 or S_2 in S_1. Stage completion events only exist for the most nested stages, which are called *atomic*.

We next formally define GSM schemas. We assume a global universe $\mathcal{U}$ of named external events and attributes, partitioned into sets of named external events $\mathcal{U}_E$, data attributes $\mathcal{U}_d$, stage attributes $\mathcal{U}_S$, and milestone attributes $\mathcal{U}_m$.

Definition 1 (GSM schema). *A* GSM schema *is a tuple* $\Gamma = (\mathcal{A} = \mathcal{A}_d \cup \mathcal{A}_S \cup \mathcal{A}_m, \mathcal{E} = \mathcal{E}_{ext} \cup \mathcal{E}_{cmp}, \preceq, \mathcal{R} = \mathcal{R}_+ \cup \mathcal{R}_-)$*, where*

- $\mathcal{A}_d \subseteq \mathcal{U}_d$ *is a finite set of data attributes;*
- $\mathcal{A}_S \subseteq \mathcal{U}_S$ *is a finite set of stage attributes;*
- $\mathcal{A}_m \subseteq \mathcal{U}_m$ *is a finite set of milestone attributes;*
- $\mathcal{E}_{ext} = \{\ \mathsf{E}{:}n \mid n \in Ev\ \}$ *is a finite set of named external events, where* $Ev \subseteq \mathcal{U}_E$*;*
- $\mathcal{E}_{cmp} = \{\ \mathsf{C}{:}S \mid S \in \mathcal{A}_S \wedge S$ is atomic $\}$ *is the set of stage completion events;*
- $\preceq \subseteq (\mathcal{A}_S \cup \mathcal{A}_m) \times \mathcal{A}_S$ *is a partial order on stages and milestones that induces a forest, i.e., if* $a_1 \preceq a_2$ *and* $a_1 \preceq a_3$*, then* $a_2 \preceq a_3$ *or* $a_3 \preceq a_2$*;*
- $\mathcal{R}_+$, $\mathcal{R}_-$ *are functions assigning to each status attribute* $\mathcal{A}_S \cup \mathcal{A}_m$ *non-empty sets of* sentries *(see Definition 2). For* $a \in \mathcal{A}_S \cup \mathcal{A}_m$, $\mathcal{R}_+(a)$ *is the set of* plus sentries *that define the conditions when to open stage* $a \in \mathcal{A}_S$ *or achieve milestone* $a \in \mathcal{A}_m$*, while* $\mathcal{R}_-(a)$ *is the set of* minus sentries *that define the conditions when to close stage* $a \in \mathcal{A}_S$ *or invalidate milestone* $a \in \mathcal{A}_m$*.*

Stage S is atomic if there is no other stage $S' \in \mathcal{A}_S$ such that $S' \preceq S$. The definition of $\preceq$ ensures that milestones are atomic by default. Relation $\preceq$ is visually depicted using nesting. For instance, Business Performance Evaluation Check $\preceq$ Preliminary Check and BPECS $\preceq$ Preliminary Check in Fig. 1.

Each sentry φ in set $\mathcal{R}_+(a)$, where $a \in \mathcal{A}_S \cup \mathcal{A}_m$, maps into an Event-Condition-Action rule "φ **then**$+a$", where sentry φ is the Event-Condition part and action $+a$ denotes for $a \in \mathcal{A}_S$ that stage a gets opened and for $a \in \mathcal{A}_m$ that milestone a gets achieved. Each sentry φ in set $\mathcal{R}_-(a)$ maps into a rule "φ **then**$-a$", where action $-a$ denotes for $a \in \mathcal{A}_S$ that stage a gets closed and for $a \in \mathcal{A}_m$ that milestone a gets invalid. For example, the plus sentry for milestone BPECS (cf. Table 2) is C:Business Performance Evaluation Check Successful $\wedge$ BP_good; the corresponding Event-Condition-Action rule is C:Business Performance Evaluation Check Successful $\wedge$ BP_good **then** $+$BPECS. Each sentry in set $\mathcal{R}_+(a)$ or $\mathcal{R}_-(a)$ is sufficient for triggering a status change in the stage or milestone a.

For the definition of sentries, we assume a condition language $\mathcal{C}$ that includes predicates over integers and Boolean connectives. The condition formulas may refer to stage, milestone and data attributes from the universe of attributes $\mathcal{U}$.

Definition 2 (Sentry). *A* sentry *has the form* $\tau \wedge \gamma$*, where* τ *is the event-part and* γ *the condition-part. The event-part* τ *is either empty (trivially true), a named external event* $E \in \mathcal{U}_E$*, a stage completion event* $\mathsf{C}{:}S$*, where* $S \in \mathcal{U}_S$ *is an atomic stage, or is an internal event* $+a$ *or* $-a$*, where* $a \in \mathcal{U}_S \cup \mathcal{U}_m$ *is a stage or milestone attribute. The condition* γ *is a Boolean formula in the condition language* $\mathcal{C}$ *that refers to data attributes in* $\mathcal{U}_d$ *and status attributes in* $\mathcal{U}_S \cup \mathcal{U}_m$*. The condition-part can be omitted if it is equivalent to true.*

Note that a sentry in a GSM schema Γ may or may not refer to attributes that are defined in $\mathcal{A}$. This distinction leads to two disjoint classes of GSM schemas.

Definition 3 (Base schema; schema fragment). *Let* $\Gamma = (\mathcal{A}, \mathcal{E}, \preceq, \mathcal{R})$ *be a GSM schema. Then* Γ *is a* base schema *if each sentry in* $\mathcal{R}$ *only refers to data and status attributes in* $\mathcal{A}$ *and events in* $\mathcal{E}$*. Otherwise,* Γ *is a* schema fragment.

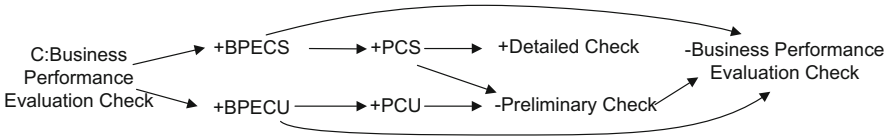

Fig. 2. Event-relativized dependency graph for C:Business Performance Evaluation Check (Fig. 1)

Each sentry in a base schema is "grounded", i.e., it can be evaluated in the context of the schema. A sentry φ in a schema fragment can be non-grounded, i.e., referring to a data or status attribute not defined in the schema fragment. By composing the schema fragment with a base schema or other schema fragments, defined in the next section, sentry φ becomes grounded. The GSM schema presented in Sect. 2 is a base schema; Sect. 4 shows example schema fragments.

Semantics and Well-formedness. In a state of the GSM schema Γ, each attribute in $\mathcal{A}$ has been assigned a value. Initially, all status attributes are false. An incoming external event $E \in \mathcal{E}$, i.e., a stage completion event or a named external event, is processed as follows [9]. Event E can carry in its payload new values for some data attributes; first, these new values are assigned to the relevant data attributes in $\mathcal{A}_d$. Next, one or more Event-Condition-Action rules are fired. Each fired rule changes the value of a status attribute a. If a becomes false, internal event $-a$ is generated; if a becomes true, internal event $+a$ is generated. Generated internal events may trigger additional rules to be fired. Eventually a new state is reached, in which no rules can be applied. In this new state, the next incoming external event can be processed. The full effect of processing an incoming external event is called a Business step, or B-step for short [9,17]. For instance, suppose stage Business Performance Evaluation Check is open and milestones BPECS and BPECU are invalid. If completion event C:Business Performance Evaluation Check with payload (BP_good,*true*) is processed, in the B-step data attribute BP_good is assigned *true*, milestone BPECS is achieved, so stage Business Performance Evaluation Check is closed and milestone PCS is achieved, which means stage Detailed Check gets opened; no further rules can be applied.

The rules need to be processed for each B-step in an order that ensures that each rule for a stage or milestone attribute a is only evaluated if the event- and condition-parts of the rule are stable, so the stage and milestone attributes referenced in those parts do not change value in the B-step after the rule for a has been processed. Therefore, the rules of these referenced stages and milestones are processed before the rule for a. The processing order of rules must be acyclic. To check this, for each external event $E \in \mathcal{E}$ an *event-relativized dependency graph*, denoted $erDG(E)$, can be constructed [9]. The graph contains E plus all internal events (in)directly caused by E. An edge from E or $\odot a_1$, for $\odot \in \{+, -\}$, to $+a_2$ (to $-a_2$) is inserted if a plus (minus) rule for a_2 either (i) contains a_1 in the condition-part, or (ii) contains either E, or $+a_1$, if $\odot{=}+$, or $-a_1$, if $\odot{=}-$, in

the event-part. Figure 2 shows the event-relativized dependency graph for event C:Business Performance Evaluation Check of BCA_{base}, defined in Sect. 2.

Definition 4. *Let Γ be a GSM schema. Γ is well-formed if for each event $E \in \mathcal{E}$, the event-relativized dependency graph erDG(E) is acyclic [9].*

4 Feature Composition

In this section, we introduce and define feature composition. A feature is a specific functionality of a software artifact that is discernible for an end user [2]. In this paper, each feature denotes a GSM schema. If a feature corresponds to a base GSM schema, we call it *complete*, since it is executable by itself. Otherwise, the feature corresponds to a GSM schema fragment and we call it *partial*, since composition with other features is required to derive a base GSM schema, which can be executed.

To define feature composition, we use a function composition operator '$\bullet$' [5], also known as superimposition [3]. Let Γ^{comp} be a (partial) feature composition and Γ^{add} a new feature. Both Γ^{comp} and Γ^{add} denote GSM schemas. Then GSM schema $\Gamma^{add} \bullet \Gamma^{comp}$ is the result of adding, also called applying, Γ^{add} to Γ^{comp}. In $\Gamma^{add} \bullet \Gamma^{comp}$, the entire schema of Γ^{add} is embedded into Γ^{comp}. Sentries of stages and milestones that are defined in Γ^{comp} are redefined (overridden by Γ^{add}) in $\Gamma^{add} \bullet \Gamma^{comp}$, if these stages and milestones also are in Γ^{add}.

In some cases, the sentries in the schemas of Γ^{comp} and Γ^{add} for a common stage or milestone should be merged rather than overridden. To specify merging, we use an additional keyword **orig** in conditions of sentries of GSM schemas. Given a feature composition $\Gamma^{add} \bullet \Gamma^{comp}$, if a sentry of Γ^{add} contains keyword **orig**, this refers to the original definition of the sentry according to Γ^{comp}. For instance, if a milestone m has a plus sentry **orig** $\wedge\, x = 10$ in Γ^{add} and m has a plus sentry E:$n \wedge y < 5$ in Γ^{comp}, then **orig** refers to E:$n \wedge y < 5$. Consequently, **orig** can only be used in sentries for stages and milestones that belong to both Γ^{comp} and Γ^{add}.

Example. The example in Sect. 2 has one complete feature, base GSM schema BCA_{base}, and three partial features, one for each variant in Table 3. The sentries for the GSM schema fragments F_1 and F_2 of variants 1 and 2, respectively, are shown in Tables 4 and 5. Feature F_1 only refers to stage, milestone and data attributes that have been defined already in BCA_{base}. In F_1 there are two plus sentries for PCS. The semantically equivalent sentry **orig** $\vee$ (IDGS $\wedge$ employee_count$<$300) is not allowed, since it is not sentry composable (defined in Sect. 5). The minus sentries in F_1 are not redefined, indicated with **orig**.

The GSM schema for feature F_2 does introduce new attributes (see Fig. 3): stage attribute Check Credit, milestone attributes CCS and CCU, and data attribute rating. Since the base schema does not contain these new stage and milestones, their plus and minus sentries cannot use **orig**. However, these sentries may refer to stages or milestones not present in the fragment. For instance, the plus sentry of new stage Check Credit refers to milestone IDGS, which is not part of F_2.

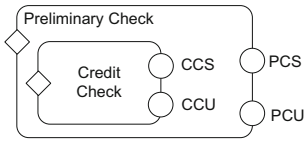

Fig. 3. GSM schema for partial feature Credit Check (F_2)

Definition. In the formal definition of '•', sentries in Γ^{add} may contain the keyword **orig** in the condition-part. Given sentries φ and ψ, sentry $\varphi[\mathbf{orig}/\psi]$ is the result of replacing each occurrence of **orig** in φ by (ψ).

Definition 5 (Feature composition). *Let Γ^{comp} be a base GSM schema and Γ^{add} a GSM schema fragment that is added to Γ^{comp}. Sentries of Γ^{add} can use the keyword **orig** in their condition-parts. Then $\Gamma^{add} \bullet \Gamma^{comp}$ is the GSM schema $\Gamma = (\mathcal{A} = \mathcal{A}_d \cup \mathcal{A}_S \cup \mathcal{A}_m, \mathcal{E} = \mathcal{E}_{ext} \cup \mathcal{E}_{cmp}, \preceq, \mathcal{R} = \mathcal{R}_+ \cup \mathcal{R}_-)$ where*

Table 4. Sentries for partial feature F_1. S = Stage; M = Milestone

Type	Name	Plus sentries	Minus sentries
S	Business Performance Evaluation Check	**orig** $\wedge$ employee_count ≥ 300	**orig**
S	Preliminary Check	**orig**	**orig**
M	PCS	IDGS $\wedge$ employee_count < 300 ; **orig**	**orig**
M	PCU	**orig**	**orig**

Table 5. Sentries for partial feature F_2. S = Stage; M = Milestone; CCS = Credit Check Successful; CCU = Credit Check Unsuccessful

Type	Name	Plus sentries	Minus sentries
S	Check Credit	IDGS	CCS ; CCU;-Preliminary Check
S	Preliminary Check	**orig**	**orig**
M	CCS	C:Credit Check $\wedge$ rating ≥ 8	+Check Credit
M	CCU	C:Credit Check $\wedge$ rating < 8	+Check Credit
M	PCS	**orig** $\wedge$ CCS	**orig**
M	PCU	**orig** ; CCU	**orig**

- $\mathcal{A}_d = \mathcal{A}_d^{comp} \cup \mathcal{A}_d^{add}$;
- $\mathcal{A}_S = \mathcal{A}_S^{comp} \cup \mathcal{A}_S^{add}$;
- $\mathcal{A}_m = \mathcal{A}_m^{comp} \cup \mathcal{A}_m^{add}$;
- $\mathcal{E}_{ext} = \mathcal{E}_{ext}^{comp} \cup \mathcal{E}_{ext}^{add}$;
- $\mathcal{E}_{cmp} = \mathcal{E}_{cmp}^{comp} \cup \mathcal{E}_{cmp}^{add}$;
- $\preceq = \preceq^{comp} \cup \preceq^{add}$;
- *for each* $a \in \mathcal{A}$,

$$\mathcal{R}_+(a) = \begin{cases} \{\varphi[\mathbf{orig}/\psi] \mid \varphi \in \mathcal{R}_+^{add}(a), \psi \in \mathcal{R}_+^{comp}(a)\} & \text{, if } a \in \mathcal{A}^{comp} \cap \mathcal{A}^{add} \\ \mathcal{R}_+^{comp}(a) & \text{, if } a \in \mathcal{A}^{comp} \setminus \mathcal{A}^{add} \\ \mathcal{R}_+^{add}(a) & \text{, if } a \in \mathcal{A}^{add} \setminus \mathcal{A}^{comp} \end{cases}$$

$$\mathcal{R}_-(a) = \begin{cases} \{\varphi[\mathbf{orig}/\psi] \mid \varphi \in \mathcal{R}_-^{add}(a), \psi \in \mathcal{R}_-^{comp}(a)\} & \text{, if } a \in \mathcal{A}^{comp} \cap \mathcal{A}^{add} \\ \mathcal{R}_-^{comp}(a) & \text{, if } a \in \mathcal{A}^{comp} \setminus \mathcal{A}^{add} \\ \mathcal{R}_-^{add}(a) & \text{, if } a \in \mathcal{A}^{add} \setminus \mathcal{A}^{comp} \end{cases}$$

Most lines in the definition above use simple set union. The definition of $\mathcal{R}$ is most involved. The basic principle is that if a stage or milestone a is defined in only one of the two input GSM schemas, the sentries for a in the composition Γ are those of the input schema. If a occurs in both input schemas, the sentries for a in Γ^{add} override the sentries for a in Γ^{comp}. Using **orig**, the original sentries in Γ^{comp} can be reused in the definition of the overridden sentries in Γ.

The GSM schema resulting from the composition $\mathsf{F}_1 \bullet \mathsf{BCA}_{base}$ (Table 6) illustrates that using the '•' operator, sentries can be merged with the original sentries (the plus sentry of Business Performance Evaluation Check in F_1 has been merged with the one in BCA_{base}) and added to the original sentries (the plus sentry for PCS in F_1 has been added to the plus sentry for PCS in BCA_{base}).

Table 6. Sentries of $\mathsf{F}_1 \bullet \mathsf{BCA}_{base}$ for the stages and milestones that are both in F_1 and BCA_{base}. S = Stage; M = Milestone

Type	Name	Plus sentries	Minus sentries
S	Business Performance Evaluation Check	+Preliminary Check ∧ employee_count ≥ 300	BPECS ; BPECU ; -Preliminary Check
S	Preliminary Check	IDGS	PCS ; PCU ; E:AdditionalInfo
M	PCS	IDGS ∧ employee_count < 300 ; BPECS	false
M	PCU	BPECU	false

Discussion. If more than two features are composed, they are ordered in a composition chain. For instance, a possible chain is $\mathsf{F}_1 \bullet \mathsf{F}_2 \bullet \mathsf{BCA}_{base}$. Operator '•' (Definition 5) requires that the righthand feature is complete. Therefore '•' is right associative.

The ordering of features in a composition chain influences the outcome. In other words, the feature composition operator '•' is not commutative. A simple

example to illustrate this: consider $\Gamma_{12} = \mathsf{F}_1 \bullet \mathsf{F}_2 \bullet \mathsf{BCA}_{base}$ versus $\Gamma_{21} = \mathsf{F}_2 \bullet \mathsf{F}_1 \bullet \mathsf{BCA}_{base}$. In Γ_{12}, one of the plus sentries for PCS is $\mathsf{IDGS} \wedge \mathsf{employee\_count} < 300$. while in Γ_{21} the corresponding plus sentry is $\mathsf{IDGS} \wedge \mathsf{employee\_count} < 300 \wedge \mathsf{CCS}$. The first sentry allows PCS to become true while CCS is false, which is obviously not desirable. So in this example, feature F_1 should be applied before F_2, so the valid composition order is $\mathsf{F}_2 \bullet \mathsf{F}_1 \bullet \mathsf{BCA}_{base}$.

In general, there are two options to handle this issue: either define correctness conditions that guarantee that '$\bullet$' is commutative or help designers to live with this lack of commutativity. Defining additional correctness conditions that guarantee that '$\bullet$' is commutative would obviously have to rule out features F_1 and F_2, which are perfectly valid. We therefore favor the other option. In particular, it is useful to define additional dependency constraints between features that help designers to manage the correct sequencing of features, if multiple need to be applied to derive an artifact-centric process model. The feature composition chain then has to respect these dependencies.

5 Correctness

The outcome of feature-oriented composition of two GSM schemas may be a structure that is not a GSM schema or not a well-formed GSM schema. We define constraints in this section that ensure that feature-oriented composition produces (well-formed) GSM schemas.

From a syntax point of view, two constraints need to be satisfied. First, the hierarchies of Γ^{add} and Γ^{comp} should be composable to ensure that the resulting relation $\preceq$, defined in Definition 5, is a hierarchy.

Definition 6 (Hierarchy composable). *Let Γ^{comp} be a base GSM schema and Γ^{add} be a GSM schema fragment that is added to Γ^{comp}. The hierarchies of Γ^{comp} and Γ^{add} are* composable *if the following conditions are met:*

- *For each pair $a_1, a_2 \in (\mathcal{A}_S^{comp} \cup \mathcal{A}_m^{comp}) \cap (\mathcal{A}_S^{add} \cup \mathcal{A}_m^{add})$, where $a_1 \neq a_2$, $a_1 \preceq^{comp} a_2$ iff $a_1 \preceq^{add} a_2$.*
- *For each pair $a_1, a_2 \in \mathcal{A}_S^{add} \cup \mathcal{A}_m^{add}$, if $a_1 \preceq^{add} a_2$ and $a_2 \notin \mathcal{A}_S^{comp} \cup \mathcal{A}_m^{comp}$ then $a_1 \notin \mathcal{A}_S^{comp} \cup \mathcal{A}_m^{comp}$.*
- *For each pair $a_1, a_2 \in \mathcal{A}_S^{add} \cup \mathcal{A}_m^{add}$, if $a_1 \preceq^{add} a_2$ and $a_2 \in \mathcal{A}_S^{comp} \cup \mathcal{A}_m^{comp}$ then a_2 is not atomic in Γ^{comp}.*

The first condition states that for stages and milestones that are shared between Γ^{comp} and Γ^{add} the hierarchy relations $\preceq^{comp}$ and $\preceq^{add}$ are consistent. The second condition rules out that a new stage is inserted by Γ^{add} in the middle of the hierarchy of Γ^{comp}. For instance, the condition is violated for Fig. 4; if the hierarchies are composed, stage S gets two parents. The third

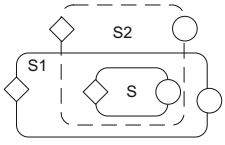

Fig. 4. Base schema $(\mathsf{S}_1, \mathsf{S})$ and fragment schema $(\mathsf{S}_2, \mathsf{S})$ that are not hierarchy composable

condition is that inserting a new stage inside an existing stage S is allowed, provided S is not atomic in Γ^{comp}. If S is atomic in Γ^{comp}, then S generates a completion event $\mathsf{C}{:}S$, which is not generated if S is not atomic. Hence, the condition ensures that rules triggered by $\mathsf{C}{:}S$ in Γ^{comp} are also triggered in $\Gamma^{add} \bullet \Gamma^{comp}$. Under these conditions, the hierarchy relation in the composition $\Gamma^{add} \bullet \Gamma^{comp}$ is consistent with the hierarchy relations in both Γ^{add} and Γ^{comp}.

The second constraint is that the sentries must be composable if **orig** is used, i.e., merged sentries for $\mathcal{R}$ as defined in Definition 5 satisfy the sentry syntax of Definition 2.

Definition 7 (Sentry composable). *Let Γ^{comp} be a base GSM schema and Γ^{add} be a GSM schema fragment that is added to Γ^{comp}. A sentry $\varphi \in \mathcal{R}^{add}(a)$, where $a \in \mathcal{A}_S^{add} \cup \mathcal{A}_m^{add}$, is* sentry composable *if the following conditions are met:*

- *If φ contains* **orig***, then $a \in \mathcal{A}_S^{comp} \cup \mathcal{A}_m^{comp}$.*
- *If φ contains* **orig** *and has a non-empty event-part, then*
  - *if $\varphi \in \mathcal{R}_+^{add}(a)$, then each rule in $\mathcal{R}_+^{comp}(a)$ has an empty event-part;*
  - *if $\varphi \in \mathcal{R}_-^{add}(a)$, then each rule in $\mathcal{R}_-^{comp}(a)$ has an empty event-part.*
- *If φ contains* **orig** *and has an empty event-part, then φ is in conjunctive normal form and* **orig** *only occurs as conjunct in φ.*
- *If φ references a stage or milestone a that is not in $\mathcal{A}^{add}$, then $a \in \mathcal{A}^{comp}$.*

The first condition in the definition ensures that if a sentry φ for a uses **orig**, then the attribute a exists in the base GSM schema. This way, **orig** can always be substituted by another sentry. The second and third condition ensure that a sentry for a in the base GSM schema and a sentry for a in the schema fragment can be composed properly into a new sentry of the form $\tau \wedge \gamma$. For instance, if a sentry φ in Γ^{add} uses **orig** and is triggered by a completion event $\mathsf{C}{:}S$, then the sentry of Γ^{comp} referred to by **orig** should have no trigger event. Note that a sentry of the form **orig**$\vee\varphi$, which is ruled out by the third condition, is equivalent to pair of sentries **orig** and φ. Thus, this condition does not diminish expressive power. The fourth condition makes sure that each sentry in the schema fragment becomes grounded.

Under these two constraints, the result of feature composition is guaranteed to be a GSM schema.

Lemma 1. *Let Γ^{comp} be a base GSM schema and Γ^{add} be a GSM schema fragment such that $\Gamma = \Gamma^{add} \bullet \Gamma^{comp}$. If Γ^{comp} and Γ^{add} have composable hierarchies and each rule in Γ^{add} is sentry composable, then Γ is a GSM schema.*

Proof. (Sketch.) We focus on showing that each sentry in Γ satisfies the syntax defined in Definition 2. Suppose a sentry $\varphi = \tau_1 \wedge \gamma_1$ from Γ^{add} contains **orig**. By Definition 5, **orig** is contained in γ_1. By the first condition of Definition 7, $a \in \mathcal{A}_S^{comp} \cup \mathcal{A}_m^{comp}$. We prove that the new sentry is of the form $\tau \wedge \gamma$ (cf. Definition 2). There are two cases.

(a) Sentry φ has a non-empty event-part τ_1. By the second condition of Definition 7, any sentry $\psi = \tau_2 \wedge \gamma_2$ from Γ^{comp} that **orig** is substituted with, has an empty event-part, so the new sentry is $\tau_1 \wedge \gamma_1'$, where $\gamma_1' = \gamma_1[\textbf{orig}/\gamma_2]$.

(b) Sentry φ has an empty event-part τ_1. Let $\psi = \tau_2 \wedge \gamma_2$ be the sentry from Γ^{comp} that **orig** is substituted with. Either the sentry ψ has an empty event-part, in which case the new sentry is $\gamma_1[\textbf{orig}/\gamma_2]$. Or the sentry ψ has a non-empty event-part. By the third condition of Definition 7, γ_1 is in conjunctive normal form with **orig** as a conjunct. Let γ_{min} be γ_1 minus the conjunct **orig**. The new sentry is $\tau_2 \wedge \gamma_2 \wedge \gamma_{min}$. □

Though Lemma 1 shows under which conditions Γ is a GSM schema, it might be that Γ is not well-formed due to a cycle in an event-relativized dependency graph for some event E. Such a cycle is caused by a different processing order of the rules (in)directly triggered by E in Γ^{comp} and Γ^{add}. To rule out such cycles, we introduce a third constraint: the ordering of status attributes in both Γ^{add} and Γ^{comp} is consistent for each event E, so it is not the case that a_1 before a_2 in Γ^{comp} yet a_2 before a_1 in Γ^{add} while processing E. This can be checked by inspecting all the event-relativized dependency graphs of both Γ^{add} and Γ^{comp}.

Definition 8. (Consistent rule orderings). *Let Γ^{comp} be a base GSM schema and Γ^{add} a GSM schema fragment that is added to Γ^{comp}. Then Γ^{comp} and Γ^{add} have* consistent rule orderings *if for each event E the orderings of status attributes in $erDG_{\Gamma}^{comp}(E)$ and $erDG_{\Gamma}^{add}(E)$ are compatible, so for $a_1, a_2 \in \mathcal{A}^{comp} \cap \mathcal{A}^{add}$, a_1 before a_2 in $erDG_{\Gamma}^{comp}(E)$ implies a_2 not before a_1 in $erDG_{\Gamma}^{add}(E)$.*

If the third constraint is satisfied too, then composition Γ is guaranteed to be a well-formed GSM schema.

Lemma 2. *Let Γ^{comp} be a well-formed base GSM schema and Γ^{add} be a well-formed GSM schema fragment such that $\Gamma = \Gamma^{add} \bullet \Gamma^{comp}$. If Γ^{comp} and Γ^{add} have composable hierarchies, each sentry in Γ^{add} is sentry composable, and Γ^{comp} and Γ^{add} have consistent rule orderings, then Γ is well-formed.*

Proof. By Lemma 1, Γ is a GSM schema. Suppose Γ is not well-formed. By Definition 4 there is an event E such that $erDG(E)$ contains a cycle between nodes a_1 and a_2. Since Γ^{comp} and Γ^{add} are well-formed, the event-relativized dependency graphs for E in Γ^{comp} and Γ^{add} are acyclic. Hence, in one graph the ordering is a_1 before a_2, in the other a_2 before a_1. Therefore, Γ^{comp} and Γ^{add} have inconsistent rule orderings. □

6 Evaluation

To evaluate the feasibility and potential gain of the approach, we applied the approach to model variants of a real-world process of an international high tech company with offices in different regions of the worlds. In the process the expired due diligence qualification of a business partner of the company is renewed. The company has defined a standard due diligence process, but offices in certain regions can use their own variant of the process. The standard process and the variants had been modeled before in separate GSM schemas [30].

Based on the existing GSM schemas for this process, we defined a base schema DDP_{base} for the standard process and four features that refine the base schema: $\mathsf{F}_{1a}^{\mathsf{DDP}}$, $\mathsf{F}_{1b}^{\mathsf{DDP}}$, $\mathsf{F}_{2}^{\mathsf{DDP}}$, and $\mathsf{F}_{3}^{\mathsf{DDP}}$; all are available in an online appendix [11]. The first two features are alternatives. Similar to F_1 (Table 4) and F_2 (Table 5), each fragment schema of a feature uses **orig** as sentry or as conjunct of a sentry to specify the connection between the base schema and the fragment schema.

Table 7. Descriptive statistics of due diligence process and its variants

GSM schema	Feature composition	# Stages	# Milestones	# Sentries	Overlap with DDP_{base} in:			
					% Stages	% Milestones	# Sentries	% Sentries
Base schema	DDP_{base}	9	15	60				
Variant 1	$\mathsf{F}_{1a}^{\mathsf{DDP}} \bullet \mathsf{DDP}_{base}$	10	16	66	90	93	59	89
Variant 2	$\mathsf{F}_{1b}^{\mathsf{DDP}} \bullet \mathsf{DDP}_{base}$	11	17	71	82	88	59	83
Variant 3	$\mathsf{F}_{2}^{\mathsf{DDP}} \bullet \mathsf{DDP}_{base}$	10	16	65	90	93	60	92
Variant 4	$\mathsf{F}_{3}^{\mathsf{DDP}} \bullet \mathsf{DDP}_{base}$	10	16	66	90	93	60	91

Table 7 shows descriptive statistics of the base schema and four different variants that are derived by applying each of the four features to the base schema. Each variant is equivalent to an existing variant [30]. Note that the base schema is embedded in each of the variants. Hence, there is an overlap between each variant and the base schema. For instance, the first variant shares 9 out of 10 stages and 15 out of 16 milestones with the base schema. Variant 1 and 2 each have a sentry that is derived from a sentry of the base schema; the corresponding sentry in the feature uses conjunct **orig**. Hence, for variants 1 and 2, 59 sentries of the base schema appear in the variant rather than 60.

Table 7 shows that the overlap between the variants is huge. If the variants are modeled separately, this causes maintenance problems. For instance, changing the name of a milestone m shared by all four variants needs to be done four times. By using features, this overlap can be managed efficiently. The milestone m needs to be updated only once, for the base schema. The change is then propagated to the variants derived from the base schema by recomposing them. Another benefit of the approach is that many more variants can be derived than just the ones in Table 7. In total $3 * 2 * 2 = 12$ variants (incl. the base schema) can be derived, without modifying any of the fragment schemas or the base schema.

To conclude, the preliminary evaluation on a real-world process shows that by using the approach, variants in a family of GSM schemas can be expressed as feature compositions. Using features avoids duplicates and thus eases the design and maintenance of declarative artifact-centric process variants.

7 Related Work

For artifact-centric process models, no directly related work on composition of model fragments exists. The general problem of designing artifact-centric process

models, either by defining a methodology for specifying declarative business artifacts [6] or by defining an automated synthesis of artifact-centric process models [13,14,20,24] has been addressed. The feature-oriented composition approach facilitates reuse of model fragments and the generation of different but related variants, rather than designing a single artifact-centric process model.

Alternatives to artifact-centric process models are object-aware process models [18,25] and case management models [1,22,28] (though artifact-centric process models can be used for case management too [12,21]). For one of these alternatives, an approach has been defined for composing production case management models out of procedural process models [22]. Composition at a fine-grained level (overriding) is not supported and features are not used.

Variability has been well studied for activity-centric business processes. Recent surveys [4,19] describe the state of the art in variability modeling from the angles of procedural modeling languages [19] and of different phases of the business process life cycle [4], which includes procedural modeling languages. The surveyed mechanisms for modeling variability [4,19] are specific to procedural, flowchart-like languages (e.g. specializing activities [19]). Variability support for declarative, activity-centric business processes has been developed [27], but all variants are encoded in a single declarative process model from which a process variant is generated by hiding activities and omitting constraints, rather than composing a process variant from fragments in a modular fashion using features. Another survey [26] lists papers that have applied variability techniques from software product lines, such as feature models, to activity-centric, procedural business processes. To the best of our knowledge, there is no related work that applies feature composition to declarative or procedural process models.

In earlier work [12], we defined the change operations insertion and deletion for monotonic GSM schemas (each attribute can be written only once during an execution) without hierarchy. An alternative approach for deriving GSM schema variants can be defined by using the GSM change operations in combination with Provop [16], an existing approach for managing variability in procedural process models in terms of change operations. Though that alternative approach allows reducing a base schema, not possible with feature-oriented composition, it is restricted to monotonic GSM schemas without hierarchy, while the feature-oriented composition approach supports non-monotonic, hierarchical GSM schemas.

In software engineering, features have been applied both at the level of modeling languages [23] and programming languages [2]. The feature-oriented design approach defined in Sect. 4 resembles most closely the feature composition approach for software artifacts developed by Batory et al. [5] and extended by Apel et al. [3]. That approach also uses a composition operator (called superimposition [3]) and supports merging of method bodies using an **orig**-like construct. Since merging of GSM schemas can result in incorrect schemas, we consider correctness issues, which are ignored for software artifacts [3,5].

8 Conclusion

The main contribution of this paper is a formally defined, feature-oriented approach for composing declarative artifact-centric process models. The approach defines how to compose features that denote GSM schemas, some of which are partially specified, into completely specified GSM schemas. Correctness criteria are defined that guarantee that valid GSM schemas are derived. The approach supports reuse of model fragments. The approach has been evaluated by applying it to a GSM schema of an industrial process. Using the approach, declarative artifact-centric process models can be designed in an efficient and correct way.

One direction for further work is evaluating the approach in more case studies. Another direction is developing tool support for the approach based on an existing feature composition tool like FeatureHouse [3].

References

1. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005)
2. Apel, S., Batory, D.S., Kästner, C., Saake, G.: Feature-Oriented Software Product Lines: Concepts and Implementation. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37521-7
3. Apel, S., Kästner, C., Lengauer, C.: Language-independent and automated software composition: the featurehouse experience. IEEE Trans. Softw. Eng. **39**(1), 63–79 (2013)
4. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: VIVACE: a framework for the systematic evaluation of variability support in process-aware information systems. Inf. Softw. Technol. **57**, 248–276 (2015)
5. Batory, D.S., Sarvela, J.N., Rauschmayer, A.: Scaling step-wise refinement. IEEE Trans. Softw. Eng. **30**(6), 355–371 (2004)
6. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Modeling, pp. 503–531 (2009). Chapter 23
7. BizAgi and others: Case Management Model and Notation (CMMN), v1.1. OMG Document Number formal/16-12-01, Object Management Group, December 2016
8. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
9. Damaggio, E., Hull, R., Vaculín, R.: On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. Inf. Syst. **38**, 561–584 (2013)
10. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. J. Data Semant. **4**(1), 29–57 (2015)
11. Eshuis, R.: Appendix to: Feature-Oriented Composition of Declarative Artifact-Centric Process Models (2018). http://is.ieis.tue.nl/staff/heshuis/foc-appendix.pdf
12. Eshuis, R., Hull, R., Yi, M.: Property preservation in adaptive case management. In: Barros, A., Grigori, D., Narendra, N.C., Dam, H.K. (eds.) ICSOC 2015. LNCS, vol. 9435, pp. 285–302. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48616-0_18

13. Eshuis, R., Van Gorp, P.: Synthesizing data-centric models from business process models. Computing **98**(4), 345–373 (2016)
14. Fritz, C., Hull, R., Su, J.: Automatic construction of simple artifact-based business processes. In: Proceedings of (ICDT), pp. 225–238 (2009)
15. Goedertier, S., Vanthienen, J., Caron, F.: Declarative business process modelling: principles and modelling languages. Enterp. IS **9**(2), 161–185 (2015)
16. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. J. Softw. Maint. **22**(6–7), 519–546 (2010)
17. Hull, R., et al.: Introducing the Guard-Stage-Milestone approach for specifying business entity lifecycles. In: Bravetti, M., Bultan, T. (eds.) WS-FM 2010. LNCS, vol. 6551, pp. 1–24. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19589-1_1
18. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. J. Softw. Maint. **23**(4), 205–244 (2011)
19. La Rosa, M., van der Aalst, W.M.P., Dumas, M., Milani, F.: Business process variability modeling: a survey. ACM Comput. Surv. **50**(1), 2:1–2:45 (2017)
20. Lohmann, N.: Compliance by design for artifact-centric business processes. Inf. Syst. **38**(4), 606–618 (2013)
21. Marin, M., Hull, R., Vaculín, R.: Data centric BPM and the emerging case management standard: a short survey. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 24–30. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_4
22. Meyer, A., Herzberg, N., Puhlmann, F., Weske, M.: Implementation framework for production case management: modeling and execution. In: Proceedings of EDOC 2014, pp. 190–199. IEEE Computer Society (2014)
23. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering - Foundations, Principles, and Techniques. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-28901-1
24. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. Int. J. Coop. Inf. Syst. **24**(1), 1–44 (2015)
25. Redding, G., Dumas, M., ter Hofstede, A.H.M., Iordachescu, A.: A flexible, object-centric approach for business process modelling. Serv. Oriented Comput. Appl. **4**(3), 191–201 (2010)
26. dos Santos Rocha, R., Fantinato, M.: The use of software product lines for business process management: a systematic literature review. Inf. Softw. Technol. **55**(8), 1355–1373 (2013)
27. Schunselaar, D.M.M., Maggi, F.M., Sidorova, N., van der Aalst, W.M.P.: Configurable declare: designing customisable flexible process models. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7565, pp. 20–37. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33606-5_3
28. Slaats, T., Mukkamala, R.R., Hildebrandt, T., Marquard, M.: Exformatics declarative case management workflows as DCR graphs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 339–354. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_28
29. Swenson, K.D.: Mastering the Unpredictable: How Adaptive Case Management will Revolutionize the Way that Knowledge Workers Get Things Done. Meghan-Kiffer, Tampa (2010)
30. Yi, M.: Managing business process variability in artifact-centric BPM. Master's thesis, Eindhoven University of Technology (2015)