

Formalizing Concepts for Efficacy-aware Business Process Modeling^{*}

Matthias Lohrmann and Manfred Reichert

Ulm University,
Databases and Information Systems Institute,
{matthias.lohrmann,manfred.reichert}@uni-ulm.de

Abstract. In business process design, business objective models can fulfill the role of formal requirement definitions. Matching process models against objective models would, for instance, enable sound comparison of implementation alternatives. For that purpose, objective models should be available independently of their concrete implementation in a business process. This issue is not addressed by common business process management concepts yet. Moreover, process models are currently not sufficiently expressive to determine business process efficacy in the sense of fulfilling a business objective. Therefore, this paper develops and integrates constructs required for efficacy-aware process modeling and apt to extend common modeling approaches. The concept is illustrated with a sample scenario. Overall, it serves as an enabler for progressive applications like automated process optimization.

Key words: Business Process Modeling and Analysis, Business Process Design, Business Objectives and Goals

1 Introduction

The notion of business goals, objectives, or similar concepts has been widely used to define the term *business process* (e.g., [1]). At the same time, semantic quality of process models has been recognized as an important prerequisite for successful adoption [2]. Nevertheless, objectives are still a notable exception to the progress towards formal business process semantics, and are only rudimentarily considered in common modeling approaches [3, 4]. The effectiveness of processes in regard to achieving business objectives can be subsumed as *business process efficacy*. The case for assessing and controlling business process efficacy with formal business objective models can be illustrated by considering exemplary application scenarios:

Scenario 1 (Automated Business Process Optimization). Process-aware information systems (PAISs) collect data on process execution that could be leveraged for

^{*} This technical report constitutes an amended version of our CoopIS 2012 conference paper titled “Efficacy-aware Business Process Modeling”. Beyond the content presented there, it comprises an appendix formalizing the concepts developed.

automated business process optimization [5]. Consider, for instance, process abortions: if a process instance cannot be completed, it should abort as early as possible to avoid unnecessary consumption of resources. Next-generation PAISs might re-arrange control flow to foster this behavior based on the execution logs of past instances. However, this must be done in a way to maintain the overall efficacy of the business process. Thus, a semantic link between business objectives and business process models is required.

Scenario 2 (Identification of Business Process Variants). The management of business process variants has emerged as an important business process management (BPM) issue [6–8]. However, criteria to determine whether two process models are variants of the same reference process remain a “missing link”. In this respect, modeling business processes in a way that enables tracing to common business objectives can provide an effective characteristic to assess the “equivalence” of process variants.

Scenario 3 (Benchmarking). Qualitative benchmarking deals with good practices to identify opportunities for process improvement [9]. This often meets the resistance of practitioners as the equivalence of process alternatives regarding their outcome is doubted. Formalizing efficacy can help to alleviate this issue. Similar considerations apply to more recent approaches like process performance management [10].

As depicted in Figure 1, our approach contributes capabilities which are required to address the scenarios lined out, but not provided by the state of the art. This includes a clear distinction between business objectives (as a formal requirements definition) and business processes (as an implementation), and the ability to formally determine whether and under which circumstances a business objective is achieved, i.e. whether a business process is *efficacious*. Proper formalization will, in the end, be key to efficient automation. Seamless integration of new concepts with existing BPM approaches fosters applicability.

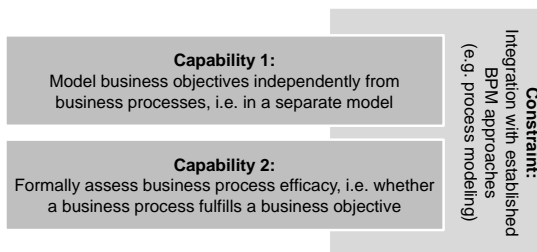


Fig. 1. Contribution Overview

2 Methodology and Outline

In general, business objectives exist independently of business processes. A certain process constitutes just one of many potential alternatives to achieve its

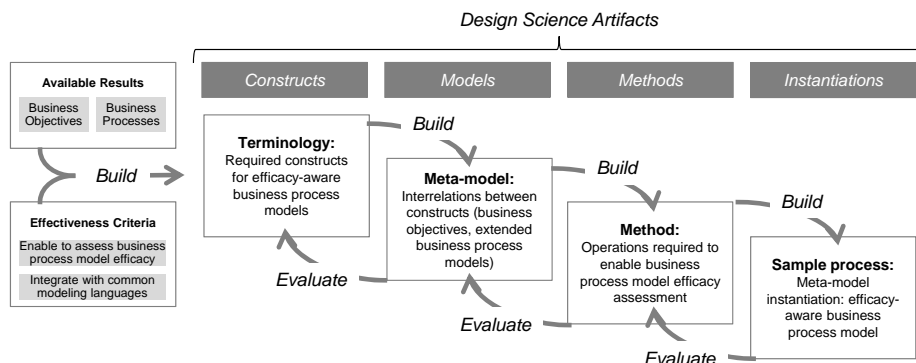


Fig. 2. Methodology to Contrive an Efficacy-aware Business Process Meta-model

objective, e.g. by using another IT system or re-arranging the order of activities. In other words, a business objective is achieved by inducing a state that satisfies particular criteria – no matter how this is done. Therefore, assessment and control of business process efficacy generally require two modeling facilities:

1. A *business objective meta-model* that is sufficiently expressive to model business objectives independently of corresponding business processes in the sense of a formal requirements definition.
2. An *efficacy-aware business process meta-model* that is sufficiently expressive to determine whether a corresponding business process model fulfills a business objective, i.e. whether it is efficacious.

As we presented results towards business objective modeling in [11], the focus of this paper lies on the second modeling facility, the *efficacy-aware business process meta-model*. Since this is a *goal-bound artificial construct*, we orient our methodology at the design science paradigm [12, 13]. Figure 2 presents an overview reflecting the design procedures *build* and *evaluate*, and the design artifacts *constructs*, *models*, *methods*, and *instantiations* [13]. As a first step, we *build* a set of required *constructs* based on available results. This provides us with a terminology for further considerations. We then describe the relations between the constructs identified, thus *building* a meta-model.¹ The meta-model is amended with operations required for efficacy assessment in the sense of a *method*. On that basis, efficacy-aware business process models can be *built*. *Evaluation* of results then occurs in reversed order: a sample process is used to evaluate the effectiveness of the method, the meta-model and, in turn, the set of constructs.

In terms of functionality, the resulting artifacts can be considered as effective if efficacy assessment has been enabled. As an additional effectiveness criterion to ensure applicability, we also demand that results should integrate well with

¹ Note the shift in terminology: in terms of design science artifacts, our meta-model constitutes a *model*.

existing business process modeling languages. This means that new constructs and meta-model elements should only be used where required due to limitations in well-established languages, and all new constructs should be well-connected to existing terminology. Note that these effectiveness criteria correspond to the second capability and the constraint stipulated in Figure 1. The first capability, separating objective from process models, has been addressed in [11] which provides the basis for this paper.

Accordingly, Section 3 reflects existing results and related work as a starting point to build our approach. Section 4 derives terminology required for efficacy assessment. Section 5 builds a meta-model for efficacy-aware business processes. Section 6 presents operations required for efficacy assessment as well as an exemplary instantiation of the meta-model to discuss the validity of our results.

3 Background

This section discusses related approaches and summarizes our previous work.

3.1 Related Work

Since the notion of goals or objectives is part of most definitions of the term “business process”, there have been a number of approaches towards integrating objectives into process modeling. Table 1 summarizes related approaches along a set of semantic requirements relevant in the context of business objectives. A more detailed analysis is presented in [11].

Semantic Requirements	Reflection in Related Work	Conclusions
<i>Consideration of the affecting environment:</i> Objectives must consider the state of both target artifacts to be created and altered <i>and</i> environmental conditions (e.g. in decision processes).	Mostly no formal, state-based concept of objectives achievement (e.g., [14–16]); objectives may be viewed not as state to be achieved, but as set of tasks to be executed (e.g., [17, 18]).	The requirement to delineate objectives from activities and to sufficiently consider environmental conditions are still open issues.
<i>Varying target environment:</i> Depending on environmental conditions, the set of artifacts to be created or altered and the set of operations to be carried out may vary.	Goals are mostly discussed on an abstract level without referring to single target artifacts (e.g., [14, 15]) or without an environmental conditions concept (e.g., [18, 16]).	Flexible adaptation to environmental conditions in terms of actually required process results is still an open issue.
<i>Order constraints:</i> Constraints to the order of activities to be carried out must be representable).	Constraints are partially considered as an abstract construct [18] or via consistency of paths [17]. Other approaches omit order constraints (e.g. [14–16]).	The notion of constraints is partly available, but needs to be refined.

Table 1. Semantic Requirements and Related Work

The gap of existing approaches regarding the coverage of semantic requirements is not surprising considering that goal structures are mostly used as an auxiliary tool in process design, but not as a means to formally manage efficacy. Typically, this leads to the absence of separate business *objective* models. Instead, business objectives are integrated into business *process* models.

This view is also reflected in a number of general process modeling formalisms (e.g., EPCs [4]) as well as, in a broader context, enterprise architecture approaches (e.g., [19]). It corresponds to the *methods* category of design science artifacts, and the merits of related concepts should be evaluated in this context. In contrast, this paper is mainly focused on *constructs* and *models*.

Beyond the related work discussed in Table 1, it is instructive to consider i^* [20] and KAOS [21] from the requirements engineering field. The i^* framework aims at documenting actors' goals and dependencies in early-phase requirements engineering, but not on formalizing objectives. Accordingly, i^* addresses a different lifecycle stage and cannot be matched against the semantic requirements for our approach. In turn, KAOS provides a framework for capturing aspects relevant to information systems requirements engineering via an “acquisition strategy”. The *constraints* construct in KAOS corresponds to the concept of business objectives in [11] and could be extended by the aspects relevant to BPM as discussed there (cf. Section 3.2). The focus of this paper, however, is on integrating with common BPM approaches instead of requirements engineering. To avoid unnecessary complexity, we thus settle for our approach which is more specific in this respect.

Approaches towards the compliance of process models to given rules (e.g., [22]) are aimed at ensuring the compliance of process execution with constraints imposed (e.g., legal requirements), but do not address issues such as deriving required resources from a process model. They are thus not sufficient to enable efficacy assessment.

3.2 Available Results from Previous Work

Addressing the first capability stipulated in Section 1, we suggested and evaluated a meta-model for formal business *objectives* modeling in [11]. This paper focuses on its integration with common business *process* modeling concepts to enable efficacy assessment. Consequently, this section provides an overview on relevant business objective concepts needed for the understanding of our results.

Business processes are enacted to induce a change to their environment. The intended change constitutes a *business objective*. A naïve approach might be to simply model a set of actions required to fulfill the business objective. However, this is not sufficient, as business processes need to *interact* with their environment. This means that the intended final state must be derived from the initial state of the environment. As an example, consider the approval of loans. The loan decision must consider the customer's credit history. Thus “approve loan” is not sufficient to describe the business objective. This challenge lies behind most of the constructs shortly presented in the following.

For business objective modeling, we discern *target elements* as characteristics of the business process’s environment to be altered, and *conditional elements* as characteristics that need to be considered to determine the intended final state. Both make up the *environmental elements* of a business process. In our loan approval example, the loan decision constitutes a target element, and its aspired value depends on the state of the customer’s credit history as a conditional element. To describe the state of both target and conditional elements, we use the concept of *binary state determinants (BSDs)*.

We discern between *target BSDs* and *conditional BSDs*, which relate the state of target and conditional elements, respectively, to a value range, either absolutely or in terms of other conditional elements. If the respective relation holds, the BSD is *fulfilled*. Regarding our loan example, “loan decision = approved” might be a target BSD, and “overdues < 5% of credit volume” might be a conditional BSD. Target BSDs are classified according to types reflecting their semantic interrelation with environmental conditions which can, in turn, be expressed through sets of conditional BSDs. Figure 3 provides an overview.

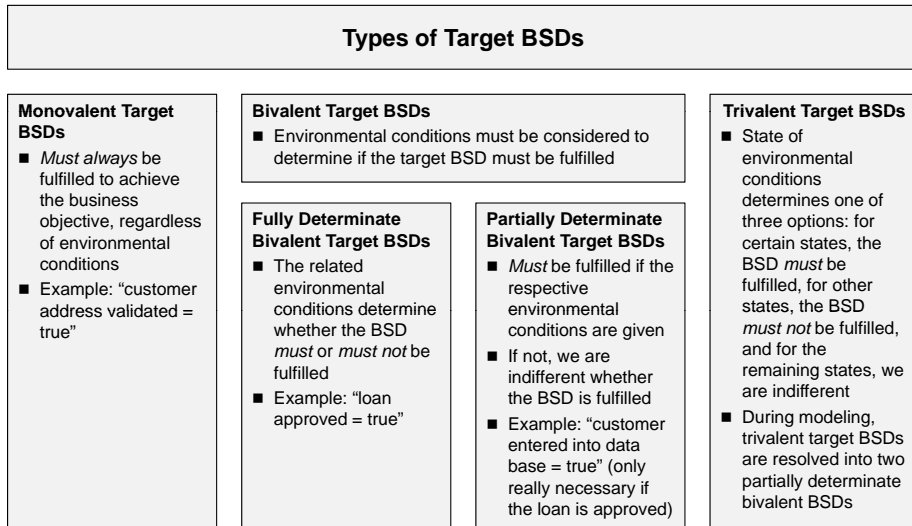


Fig. 3. Types of Target BSDs

To model the environmental conditions that determine whether a bivalent target BSD must be fulfilled, each bivalent target BSD is linked to a *conditional proposition*. Conditional propositions consist of conditional BSDs that are arranged into sets of necessary and sufficient sub-conditions (cf. Example 1). The sub-conditions allow minimizing the number of required checking actions by following a strategy to quickly approve or disapprove the proposition.

Thus, a *business objective* bundles a set of target BSDs. It is achieved if and only if each target BSD comprised has assumed a state reflecting its conditional propositions. An *efficacious business process* has to approve or reject each conditional proposition, and manipulate target elements accordingly. Figure 4 shows an exemplary business objective model. The corresponding semantics are described in Example 1.

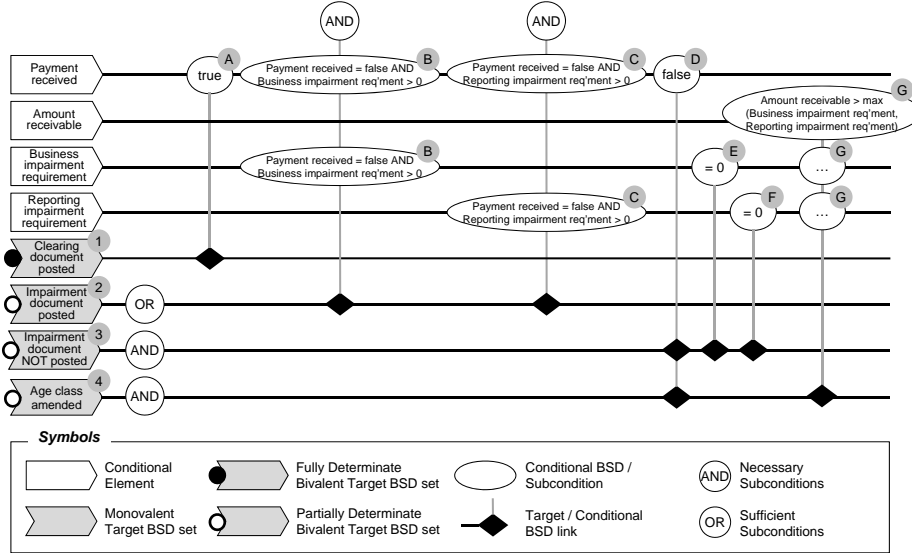


Fig. 4. Exemplary Business Objective: Year-end Receivables Processing

Example 1 (Business Objective: Year-end Receivables Processing). Properly processing receivables, e.g. open customer invoices, constitutes a business objective during year-end closing in accounting. Figure 4 informally presents the respective objective model described in the following. The top four horizontal lines in the model correspond to the relevant conditional elements, and the bottom four lines correspond to target BSDs. Vertical lines and nodes are used to link conditional elements and target BSDs by way of conditional propositions. For reference, target BSDs and sub-conditions have been amended with numbers and literals, respectively. The individual target BSDs are modeled as follows:

- Target BSD *Clearing document posted* (1): If payment has been received for the receivable, it must be cleared, i.e. removed from the list of open items. If not, a clearing document must not be posted. Accordingly, *Clearing document posted* constitutes a fully determinate bivalent target BSD linked to *Payment received* (A) as a conditional BSD. Since there are no other

conditional BSDs to be considered, there is no need to discern sufficient and necessary sub-conditions.

- Target BSDs *Impairment document posted* / *Impairment document NOT posted*: An open receivable must be impaired (i.e., its book value as an asset must be reduced) in certain cases, but it must not be impaired in others. Moreover, there may be circumstances where we are indifferent. Accordingly, *Impairment document posted* constitutes a trivalent target BSD which we resolve into two partially determinate bivalent ones: *Impairment document posted* and *Impairment document NOT posted*.
 - Target BSD *Impairment document posted* (2): Business and reporting impairment requirements are needed if the receivable is not being fully recoverable according to management’s judgment or having to be impaired for (legal) reporting guidelines, respectively. If there is no payment but a business impairment requirement (B), or if there is no payment but a reporting impairment requirement (C), the impairment must be posted. Accordingly, the OR label associated with the target BSD indicates that there are two sufficient sub-conditions, each consisting of two conditional BSDs.
 - Target BSD *Impairment document NOT posted* (3): On the other hand, if no payment has been received (D), and there is neither a business nor a reporting impairment requirement (E and F), an impairment must not be posted. Thus, there are three necessary sub-conditions as indicated by the AND label going with the target BSD.
- Target BSD *Age class amended* (4): If an open receivable has not been cleared (D) and its amount is greater than the amount to be impaired (G), it must be amended with an age class for correct balance sheet reporting (e.g., “> 12 months”). If the receivable has been reduced to zero through clearing or impairment, we are indifferent whether an age class is amended. Therefore, *Age class amended* constitutes a partially determinate target BSD with two necessary sub-conditions.

Note that for target BSDs with more than one conditional BSD, the notation allows showing either necessary or sufficient sub-conditions, depending on the modeler’s choice, and that monovalent Target BSDs do not occur in our example.

The business objectives modeling approach has been derived from semantic requirements (cf. Table 1) and tested against usability criteria by applying it to an exemplary real-world case with a modeling *method* [11]. In contrast to related work (cf. Section 3.1), it provides the ability to formally describe business objectives as intended states of the environment. It also provides a convention to model order constraints. However, this topic is not covered by our example, as it is more of a challenge when modeling business objectives *without* referring to a concrete process model.

4 Business Process Model Efficacy

In Section 3.2, we discussed how business objectives can be modeled independently from business processes, thus addressing the first capability in Figure 1. This section focuses on the second capability, enabling efficacy assessment. We first deepen our understanding of what constitutes an efficacious business process. Then, we discuss what information is required to assess efficacy.

Considering the notion of business objectives in [11], we can define:

Definition 1 (Business Process Model Efficacy). *A business process model is formally efficacious iff no target BSD in its business objective can be fulfilled unless the respective conditions defined by the business objective are fulfilled.*

A business process model is fully efficacious iff it is formally efficacious and all conditions which the model poses to target BSDs, but which are not defined by the business objective, are considered as reasonable by subject matter experts.

A business process model is ideally efficacious iff it is formally efficacious and there are no additional conditions posed to target BSDs beyond those defined by the business objective.

Note that *ideally efficacious business processes* do not occur in practice as each business process requires resources which are not part of the business objective (e.g., expenditure of labor).

According to Def. 1, assessing efficacy requires to analyze process models regarding the conditions they pose towards the fulfillment of target BSDs. To assess formal efficiency, the conditions obtained are then compared to the conditions posed by the objective model. Moreover, to assess full efficacy, they are in matched against subject matter experts' expectations.

Example 2 (Efficacy Assessment). As an example, reconsider the loan approval process. Comparing it with the business objective, in this case, will clarify whether decision criteria for loan approval such as the credit history are observed, i.e. whether the process is formally efficacious. For full efficacy, however, we also need to consider whether an unreasonable amount of working time is required for the process. This is not documented in the business objective, but requires the judgment of subject matter experts.

Efficacy assessment can be supported by consolidating and properly structuring the conditions a business process poses to individual target BSDs. Similar to the modeling of business objectives in [11], this can be achieved by amending target BSDs with conditional propositions consisting of conditional BSDs. In contrast to business objectives, business processes pose a conditional proposition even to monovalent target BSDs, since each business process requires resources to be available (i.e., there are no ideally efficacious processes, cf. Def. 1). As discussed in Section 3.2, assessment can be simplified by structuring conditional propositions into necessary and sufficient sub-conditions.

Example 3 (Necessary and Sufficient Sub-conditions). Again, consider the approval of loans. The availability of customer master data and the responsible manager both constitute necessary sub-conditions. Assuming that the customer’s credit history is usually available in the data base, but may also have to be obtained manually, we have two sufficient sub-conditions: the described necessary sub-conditions plus the data base entry, and the described necessary sub-conditions plus the availability of a clerk for manual evaluation.²

Moreover, to achieve formal efficacy (cf. Def. 1), the environmental conditions resulting from a process model with respect to a target BSD must “encompass” the environmental conditions specified in the objective model. More precisely, each necessary sub-condition of the target BSD in the business objective should be a necessary sub-condition in the process model as well. Therefore, we discern between the *outer conditional environment* and the *inner conditional environment* defined by process and objective model, respectively. Figure 5 summarizes the constituents of the outer conditional environment. We use the *Referent Modeling Language* (RML) described in [23], since it provides a concise means of describing set relations. The concept of *target presumptions* will be illustrated in Section 6.

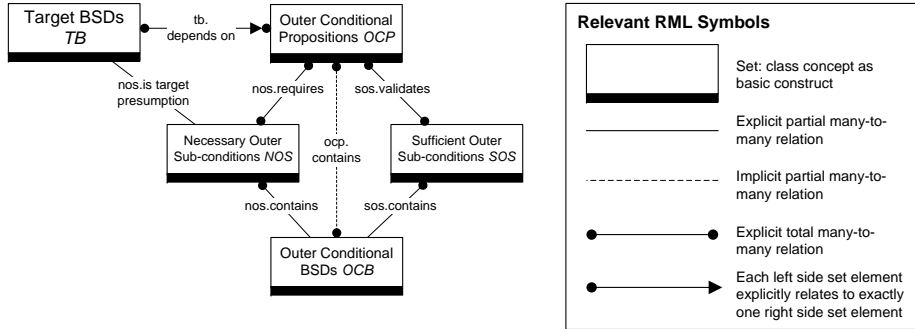


Fig. 5. Relating Target BSDs and the Outer Conditional Environment

5 Efficacy-aware Business Process Models

Business process modeling languages are mostly oriented at execution semantics of business processes, for instance because this is required for computerized workflow implementation. In terms of semantics, this requires modeling possible task sequences, but not the formalization of the impact on target BSDs or enactment preconditions (e.g., the availability of labor). We thus need to extend existing approaches towards *efficacy-aware process models*.

The Business Process Model and Notation (BPMN) constitutes a broadly applied language covering common process modeling concepts [3]. Since we aim at seamless integration with well-established concepts (cf. Figure 1), we use BPMN as a basis for extension instead of defining an entirely new formalism. Table 2 summarizes the additional terminology required. The meta-model presented in Figure 6 shows how the necessary terms are interrelated, and how they integrate with BPMN concepts.


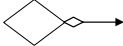
BPMN Terms	Efficacy-aware Meta-model Terms	Semantic Adaptations
Data objects 	Environmental elements: affecting and affected elements, target and conditional elements	Environmental elements replace data objects. From the perspective of the business process, they comprise the overlapping sub-sets of affecting elements (e.g., data fields altered) and affected elements (e.g., resources spent). From the perspective of the business objective, they comprise target elements and conditional elements (cf. Section 3.2). Both perspectives overlap. For example, a target element can be an affected element and an affecting element.
Conditions attached to split gateways 	Branches and branch-conditional BSDs	A branch is a sequence flow succeeding a conditional split gateway. Branch-conditional BSDs take up the concept presented in [11]. They are used to describe split gateway conditions by relating affecting elements to absolute or relative conditions (e.g., “A = 5” or “A < B”). Thus, branch-conditional BSDs represent environmental conditions that co-determine which tasks are enacted.
[none]	Task-requisite BSDs	The BSD concept is also used to describe enactment preconditions attached to tasks. Semantically, we assume that an enabled task is enacted if and only if <i>all</i> task-requisite BSDs are fulfilled. Task-requisite BSDs may relate to resources that just need to be available (e.g., “information system available = true”), or to resources actually spent (e.g., “working time available > 1h”).
[none]	State operations	State operations related to tasks are used to model effects on affected elements as functions (e.g., “A = A + B”). We assume that if a task is enacted, <i>all</i> related state operations are executed, and that state operations related to tasks are the only elements of business process models with an impact on affected elements.

Table 2. Required Terminology

In BPMN, the modeler is generally free with respect to the level of granularity regarding tasks and activities as atomic or aggregate constructs. In our context, however, we need to limit this degree of freedom to obtain stricter execution semantics. Accordingly, we require tasks to be enacted atomically, i.e., either in total or not at all. Thus, tasks do not have internal execution semantics. Trivially, this can be achieved by sufficiently refining tasks during modeling.

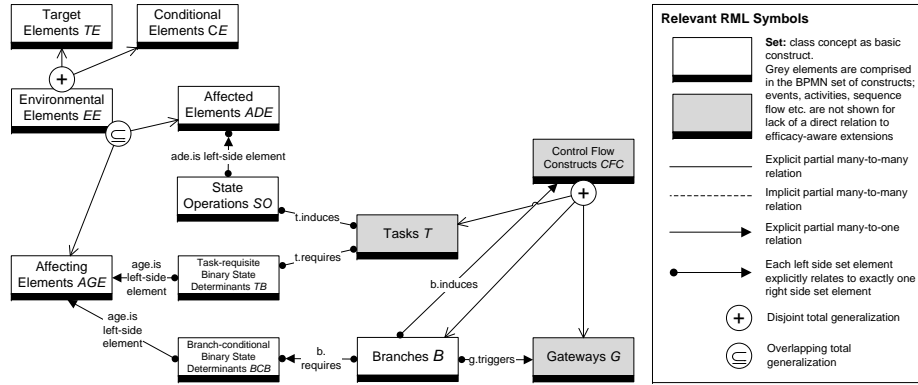


Fig. 6. Efficacy-aware Business Process Meta-model

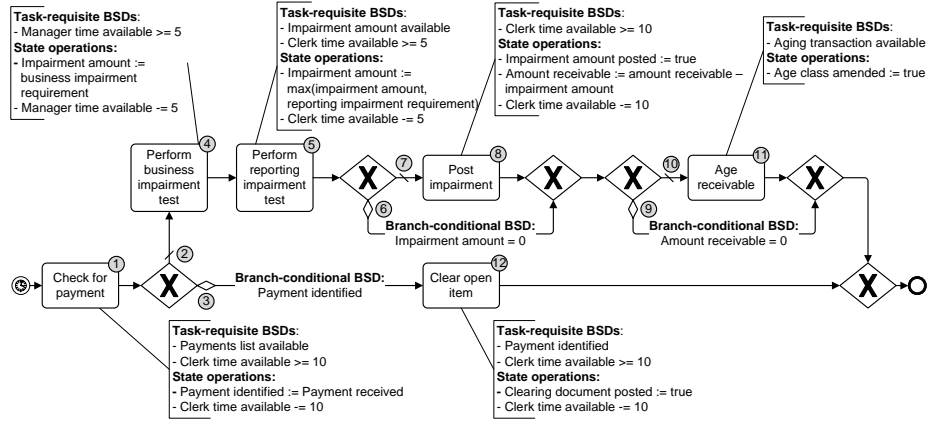


Fig. 7. Exemplary Business Process: Year-end Receivables Processing

Figure 7 illustrates a process modeled in terms of BPMN and amended with additional information according to Table 2. Its semantics are described in Example 4.

Example 4 (Sample Business Process). Figure 7 depicts a sample process model which corresponds to the business objective model from Figure 4. Relevant control flow elements have been annotated with reference numbers 1-12.

Business objective and business process relate to the management of receivables during year-end closing. Receivables are first matched against unallocated payments (1). If payment has been identified (3), the receivable is cleared (12). Otherwise (2), it is assessed in an impairment test based on management’s appraisal (4) and formal criteria (5). If an impairment amount has been identified (7), the impairment is posted (8). If an open item remains (10), it is allocated

to an age class (11). The latter task, for instance, can be enacted if it is enabled and the aging transaction is available (task-requisite BSD). If it is enacted, the age class is amended (state operation).

6 Efficacy Assessment Method and Sample Validation

Since our approach towards an efficacy-aware business process models extends BPMN with a small set of additional terms, we may assume the second effectiveness criterion (i.e., integration with common modeling languages) to be fulfilled. Accordingly, our validation focuses on the functional requirement of enabling to assess business process efficacy. Figure 5 shows which information must be available to allow assessing efficacy. This section discusses, by means of the sample process described in Example 4, how this information can be derived from an efficacy-aware business process model and used for efficacy assessment. It thus demonstrates a *method* as discussed in Section 2.

To enable efficacy assessment for a business process model, we require information about the outer conditional environment of target BSDs as described in Figure 5. This information is obtained by executing three steps presented in the following. The fourth step constitutes the actual efficacy assessment.

Step 1 (Matching Target BSDs, State Operations, and Control Flow Paths). State operations describe the actions carried out on environmental elements when enacting tasks (cf. Table 2), and are required to fulfil target BSDs. Table 3 therefore matches target BSDs against relevant state operations and possible control flow paths to enact the state operations.

Note that, for the third target BSD (*Impairment document NOT posted*), the relevant state operation *must not* be executed to fulfill the target BSD. This issue generally occurs for fully determinate bivalent target BSDs and for de-composed trivalent target BSDs (cf. Section 3.2).

Building relevant control flow paths necessitates traversing the process model. This is trivial for our simple example, but may get more complex in other cases. Respective algorithms are discussed in, for instance, [24]. Loops with an unspecified number of iterations mostly reflect sets of uniform target elements to be managed. As an example, consider lists of documents to be processed. In line with common modeling approaches [25], this issue is best addressed by using a corresponding structure of super- and sub-processes. Efficacy assessment is then executed separately for each level.

Step 2 (Consolidating Control Flow Paths). To determine the outer conditional environment required to fulfill a target BSD, we need to consolidate the BSDs comprised in relevant alternative control flow paths (cf. Table 3) considering the respective state operations. This can be achieved by “merging” subsequent control flow path elements until each relevant control path has been consolidated into one set of conditional BSDs. The required operations are shortly

Target BSD (cf. Fig. 4)	Relevant State Operation (Task No.)	Control Flow Path Alternatives (cf. Fig. 7 for reference numbers)
Clearing document posted	Clearing document posted = true (12)	(1-3-12)
Impairment document posted	Impairment document posted = true (8)	(1-2-4-5-7-8)
Impairment document NOT posted	Impairment document posted = true (8)	NOT (1-2-4-5-7-8)
Age class amended	Age class amended = true (11)	(1-2-4-5-7-8-10-11) OR (1-2-4-5-6-10-11)

Table 3. Target BSDs, State Operations, and Control Flow Paths

described in this step, but formalized in the appendix. Two subsequent control flow elements are merged as follows:

- (a) Apply the state operations of the first element to the BSDs of the second element. This is necessary to consider that state operations of the first element might affect BSDs of the second element.
- (b) Merge the resulting BSDs with the first element's BSDs.
- (c) Merge the first element's with the second element's state operations.

This provides us with new sets of BSDs and state operations, which jointly describe a new *virtual control flow element*.

Example 5 (Control Flow Path Consolidation). The results of recursively following through the consolidation procedure for the first relevant control flow path of the *Age class amended* target BSD are presented in Figure 8. The top line depicts the relevant control flow path alternative extracted from the process model (cf. Figure 7). In the second line, the merge operation has been executed for the first two control flow elements. In this case, the respective sets of BSDs do not address common environmental elements. Accordingly, the branch-conditional BSD of (2) has simply been added to the merged set of BSDs of the new virtual control flow element. The third line shows the results of following through the merge procedure for the entire control flow path alternative, so that only one virtual control flow element remains. This element bundles all BSDs and state operations as though they would be enacted in a single task.

For more complex processes, the consolidation of control flow paths can be structured along sub-processes that occur multiple times. Virtual control flow elements can then be re-used. In our example, this applies to 1-2-4-5-7-8: this sub-path is relevant to both *Impairment document posted* and *Age class amended*. Note that parallel execution paths can be handled by using block-structured process models and recursively consolidating parallel paths into activities. As an additional consistency condition, this requires that the affected elements of neither parallel path are affecting elements of the other one (cf. Table 2).

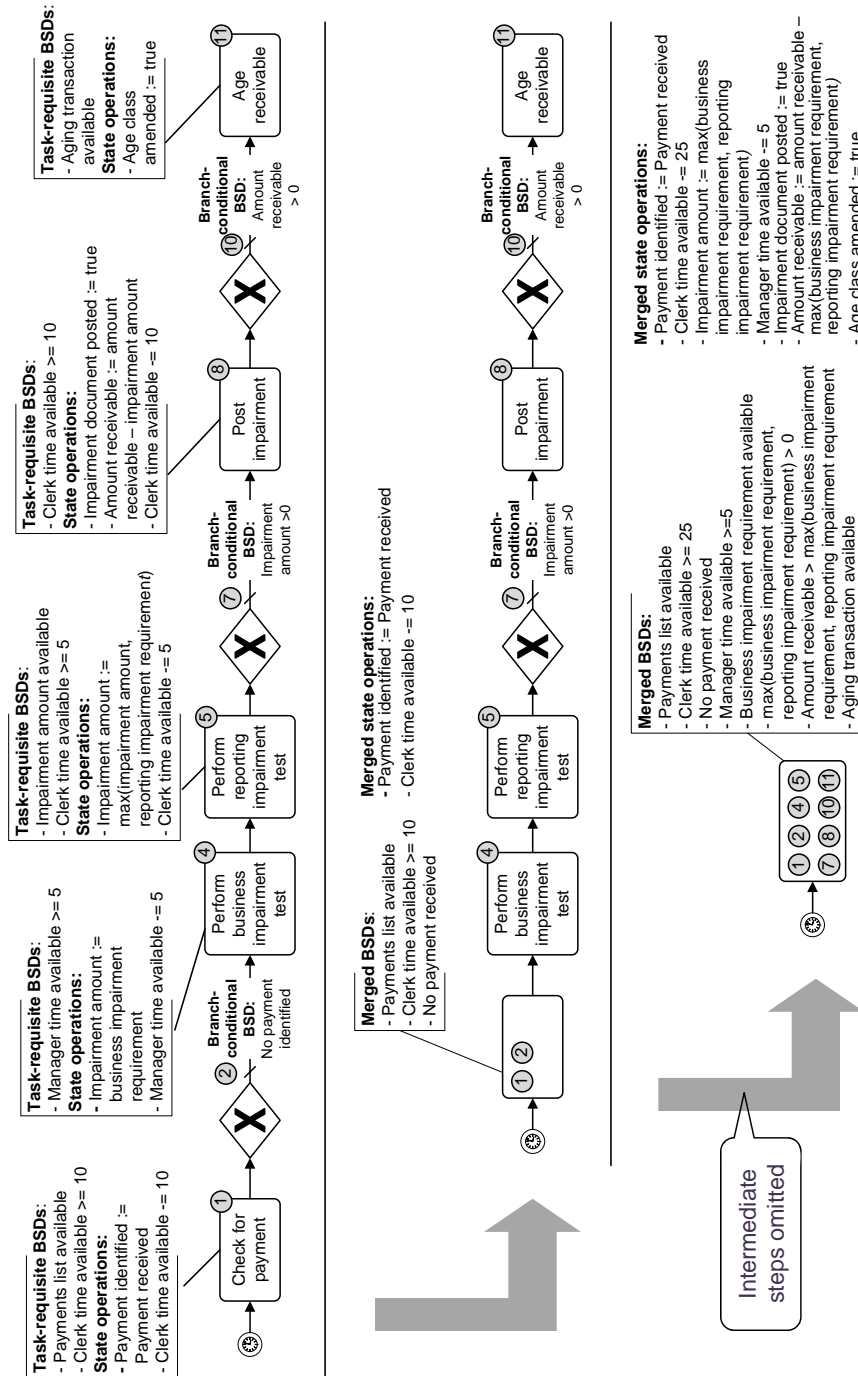


Fig. 8. Control Flow Path Consolidation Example (cf. Fig. 7)

Step 3 (Building Necessary and Sufficient Sub-conditions). Necessary and sufficient outer sub-conditions for a target BSD as defined in Figure 5 can now be derived according to a simple schema:

- Each set of merged BSDs of a consolidated control flow path constitutes a sufficient outer sub-condition since fulfillment of the BSDs is sufficient to enable the target BSD. Accordingly, there are as many sufficient outer sub-conditions as there are control flow path alternatives for a target BSD (cf. Table 3)
- Any merged BSD that occurs in each control flow alternative constitutes a necessary outer sub-condition. Note that, for the purpose of necessary outer sub-conditions, BSDs where the same set of affecting elements is covered in each control flow alternative are represented by their most “relaxed” form (in our case, this applies to the available clerk time).
- If the state operation inducing the target BSD has affecting elements, an additional necessary outer sub-condition is derived from the image function describing the operation’s content (cf. Def. 3 in the appendix): the target presumption as included in Figure 5. To this end, we substitute the function’s affected element in the target BSD by the image function. The resulting term yields the target presumption. It represents environmental conditions not caused by control flow, but by the final state operation itself.

The resulting sub-conditions are then compared to the respective necessary sub-conditions of the target BSD as per the objective model. Results for *Age class amended* are shown in Table 4.

Binary State Determinants	Outer Sub-conditions			Objective Model: Necessary Sub-conditions
	Sufficient: Path 1	Sufficient: Path 2	Necessary	
Payments list available	X	X	X	
Clerk time available ≥ 25		X		
Clerk time available ≥ 15		X	X	
Payment received = false	X	X	X	X
Manager time available ≥ 5	X	X	X	
Business impairment req’ment available	X	X	X	
$\max(\text{impairment requirements}) > 0$	X			
$\max(\text{impairment requirements}) = 0$		X		
Amount receivable $> \max(\text{imp. req's})$	X	X	X	
Aging transaction available	X	X	X	

Table 4. Target BSDs and the Conditional Environment for *Age class amended*

Step 4 (Assessing Efficacy). To compare the outer conditional environment as defined by a process model to the conditional environment of a business objective, we must mainly consider the inconsistencies. According to Def. 1, Table 4 enables to conclude:

- Target BSDs included in the business objective but not covered by state operations signify that the business process is *not formally efficacious*, because the business process alone is not sufficient to fulfill all target BSDs. This is not the case in our example.
- Necessary sub-conditions of the business objective not covered by the process indicate that the business process is *not formally efficacious*, because it may induce target BSDs without considering relevant constraints. Again, this is not the case in our example.
- Necessary outer sub-conditions of the process model with regard to a Target BSD that do not correspond to necessary sub-conditions of the business objective indicate resources required by the business process to induce a target BSD. It needs to be judged whether these are considered as reasonable – the process may be *not fully efficacious* even if it is *formally efficacious*.

For our sample process, we can conclude that the process is formally efficacious. Whether it is fully efficacious will mainly depend on whether the associated requirements regarding available labor resources are deemed as reasonable by subject matter experts. Our sample validation has thus shown that our business process meta-model (cf. Figure 6) enables efficacy assessment. As a next step, we might use the objective model and the assessment method to evaluate alternative implementation options for business process optimization.

7 Conclusion

In this paper, we built an approach towards efficacy-aware business process models based on the design science paradigm, insights on related work, and available results on business objectives modeling. We derived required terminology from functional requirements, and integrated the results into a meta-model extending BPMN. We described a method to assess the efficacy of a corresponding process model, and demonstrated the validity of our approach by application to a sample case, thus addressing our functional effectiveness criterion.

Together with the business objectives modeling approach presented in [11], the results presented yield the capabilities required for application scenarios described in Section 1: the ability to model business objectives independently from business processes, and the ability to formally assess efficacy. Seamless integration with established BPM methods is warranted. As an example of how application scenarios might be further pursued, reconsider our first scenario:

Scenario 1 (Automated Business Process Optimization). Formal efficacy assessment as demonstrated in Section 6 permits to determine whether process adaptations compromise business objective achievement. It thus becomes possible to automatically propose adaptations aimed at cost or time optimization and test their efficacy. Propositions might either be derived from empirically assessing “wasteful” execution paths, or based on random selection and subject to subsequent statistical assessment. A prospective approach would require preliminary determination of the efficacious scope of action. In this respect, the procedure of consolidating efficacy-aware process models (cf. Section 6) would have to be inverted.

In addition to adoption of the results presented into practical application scenarios, future work will also address integration of the business objectives modeling approach with requirements engineering frameworks such as KAOS [21]. Corresponding to the integration into the BPM field of knowledge, this will further improve the practical appeal of the approach.

References

1. Davenport, T.J., Short, J.E.: The new industrial engineering: Information technology and business process redesign. *Sloan Mgmt. Rev.* (4) (1990) 11–27
2. Krogstie, J., Sindre, G., Jørgensen, H.: Process models representing knowledge for action: a revised quality framework. *Europ. J. of Inf. Syst.* **15**(1) (2006) 91–102
3. The Object Management Group: Business Process Model and Notation: Version 2.0 (2011) <http://www.omg.org/spec/BPMN/2.0>.
4. Scheer, A.W., Thomas, O., Adam, O.: Process Modeling Using Event-Driven Process Chains. In: *Process-aware Information Systems*. Wiley (2005) 119–145
5. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer (2012) to appear.
6. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. *J. Sw. Mnt. Ev. Res. Pract.* **22**(6-7) (2010) 519–546
7. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. *Data & Knowl. Eng.* **70**(5) (2011) 409–434
8. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. *Comput. Ind.* **62**(5) (2011) 467–486
9. Camp, R.C.: *Benchmarking: the search for industry best practices that lead to superior performance*. Quality Press (1989)
10. IDS Scheer: Process intelligence white paper: What is process intelligence? (2009) <http://www.process-intelligence.com>.
11. Lohrmann, M., Reichert, M.: Modeling business objectives. In: *Proc. 4th S-BPM ONE – Scientific Research*. LNBP 104 (2012) 106–126
12. Simon, H.A.: *The Sciences of the Artificial*. 3rd edn. MIT Press (1996)
13. March, S.T., Smith, G.F.: Design and natural science research on information technology. *Decis. Support Syst.* **15**(4) (1995) 251–266
14. Kueng, P., Kawalek, P.: Goal-based business process models: creation and evaluation. *Bus. Process Manag. J.* **3**(1) (1997) 17–38
15. Neiger, D., Churilov, L.: Goal-oriented business process modeling with EPCs and value-focused thinking. In: *Proc. 2nd BPM*. LNCS 3080 (2004) 98–115
16. Markovic, I., Kowalkiewicz, M.: Linking business goals to process models in semantic business process modeling. In: *Proc. 12th EDOC, IEEE* (2008) 332–338
17. Soffer, P., Wand, Y.: On the notion of soft-goals in business process modeling. *Bus. Process Manag. J.* **11**(6) (2005) 663–679
18. Lin, Y., Sølvsberg, A.: Goal annotation of process models for semantic enrichment of process knowledge. In: *Proc. 19th CAiSE*. LNCS 4495 (2007) 355–369
19. Engelsman, W., Quartel, D., Jonkers, H., van Sinderen, M.: Extending enterprise architecture modelling with business goals and requirements. *Ent. Inf. Sys.* **5**(1) (2011) 9–36
20. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: *Proc. 3rd Int’l Symp. on Requirements Engineering, IEEE* (1997) 226–235

21. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. *Sc. of Comp. Programming* **20**(1-2) (1993) 3–50
22. Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Göser, K., Pfeifer, H., Reichert, M., Dadam, P.: Seaflows toolset – compliance verification made easy for process-aware information systems. In: *Proc. CAiSE'10 Forum. LNBIP 72* (2010) 76–91
23. Sølvsberg, A.: Data and what they refer to. In: *Conceptual Modeling. Springer* (1999) 211–226
24. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*. 3rd edn. MIT Press (2009)
25. Weske, M.: *Business Process Management*. Springer (2007)

Appendix: Formal Definitions

Since control flow elements (task and branches, cf. Table 2) are described through BSDs and state operations, we require formal representations of both to enable consolidating control flow paths as described in Section 6.

Let E be the set of environmental elements of a business process model, and let $e \in E$ be an individual environmental element with its domain $\mathcal{D}(e)$ as the range of possible values it may assume. BSDs³ and state operations are then defined as follows:

Definition 2 (Binary State Determinant). *A Binary State Determinant (BSD) $\gamma = \langle E_\gamma, \Lambda_\gamma \rangle$ is defined by a sequence of environmental elements $E_\gamma = (e_{\gamma_1}, \dots, e_{\gamma_n})$, $e_{\gamma_i} \in E$, which may not be empty, and a sub-set of the cartesian product of their domains $\Lambda_\gamma \subseteq \mathcal{D}(e_{\gamma_1}) \times \dots \times \mathcal{D}(e_{\gamma_n})$. Λ_γ describes the set of value tuples or states of affecting elements for which the BSD is fulfilled.*

Note that ordering the affecting elements of BSDs is necessary to maintain the semantics of the value tuples comprised in Λ_γ : each individual value obtains its semantic meaning out of the association with the environmental element at the respective position in the sequence of affecting elements.

Definition 3 (State Operation). *A state operation $\delta = \langle \theta_\delta, E_\delta, i_\delta \rangle$ is defined by an affected element $\theta_\delta \in E$, a sequence of affecting elements $E_\delta = (e_{\delta_1}, \dots, e_{\delta_n})$, $e_{\delta_i} \in E$, which may be empty, and an image function*

$$i_\delta : \prod_{e_i \in E_\delta} \mathcal{D}(e_i) \rightarrow \mathcal{I}(\delta)$$

with $\mathcal{I}(\delta) \subseteq \mathcal{D}(\theta_\delta)$ as its co-domain. A state operation is reflective iff its affected element is comprised in its set of affecting elements.

The image function describes how the new state of the affected element is deduced from the affecting elements of the state operation. Note that reflective state operations often occur in conjunction with consumable resources being decremented during process enactment. Moreover, state operations enacted within the same task may affect neither the same environmental element nor each other (i.e., no affected element of a state operation may be an affecting element of another one) since their sequence is not defined. Therefore, the following stipulation applies:

Definition 4 (Semantic Consistency of Tasks). *Let Δ_t be the set of state operations associated with a task t . The task is semantically consistent iff:*

$$\nexists \langle \delta_1, \delta_2 \rangle \in \Delta_t^2 : \delta_1 \neq \delta_2 \wedge (\theta_{\delta_1} = \theta_{\delta_2} \vee \theta_{\delta_1} \in E_{\delta_2})$$

³ In comparison to [11], the definition of BSDs has been slightly amended to allow combining an arbitrary number of environmental elements into one BSD.

On that basis, we can define a summary merge operation for relevant control flow elements, i.e. tasks and branches (cf. Table 2), which needs to be supplemented with operations to apply state operations to BSDs, to merge sets of BSDs, and to merge sets of state operations. These operations reflect the procedure lined out in Step 2 in Section 6: to merge two subsequent control flow elements, we first apply the first's state operations to the BSDs of the second one. Then, both sets of BSDs and both sets of state operations are merged, respectively. The resulting tuple consisting of a set of BSDs and a set of state operations describes the resulting virtual control flow element.

Definition 5 (Merging Control Flow Elements). *Let p_1 and p_2 be two subsequent control flow elements, i.e. tasks or branches where p_2 is the direct successor of p_1 , as part of a control flow path enabled by a process model. Let Γ_p and Δ_p be the sets of BSDs and state operations associated with a control flow element $p = \langle \Gamma_p, \Delta_p \rangle$, respectively. For branches, the set of state operations is empty. The control flow elements p_1 and p_2 are then merged as follows:*

$$p_1 \diamond p_2 \equiv \langle \Gamma_{p_1} \bullet (\Delta_{p_1} \triangleright \Gamma_{p_2}), \Delta_{p_1} \dagger \Delta_{p_2} \rangle$$

When a state operation is applied to a BSD, the BSD remains unchanged if the affected element of the state operation is not an affecting element of the BSD. If it is, the affected element as an element of the sequence of affecting elements of the BSD is replaced by the affecting elements of the state operation. To fulfill the BSD, the new sequence of affecting elements must be in a state where the result of the state operation's image function together with the state of the BSD's "remaining" affecting elements fulfills the BSD. Note that this operation can lead to individual environmental elements' occurring multiple time in the sequence of affecting elements of a BSD. In this case, note that value tuples of a BSD γ where contradictory values are required for the same affecting element cannot be fulfilled. Accordingly, the respective value tuples may be eliminated from A_γ without loss of semantic content.

A set of state operations (of a preceding task) is then applied to a BSD by subsequently applying each individual state operation. The order in which state operations are applied is of no concern since the semantic consistency of tasks (cf. Def. 4) requires state operations within one task not to affect one another or have the same affected element.

Definition 6 (Applying State Operations to BSDs). *A state operation $\delta = \langle \theta_\delta, E_\delta, i_\delta \rangle$ is applied to a BSD $\gamma = \langle E_\gamma, A_\gamma \rangle$ as follows:*

$$\delta \triangleright \gamma \equiv \begin{cases} \langle (e_{\gamma_1}, \dots, e_{\gamma_{k-1}}, E_\delta, e_{\gamma_{k+1}}, \dots, e_{\gamma_n}), A_{\gamma'} \rangle & \text{if } \theta_\delta = e_{\gamma_k} \\ \gamma & \text{else} \end{cases}$$

$$\text{with } A_{\gamma'} = \{ \lambda \mid \lambda = \langle x_1, \dots, x_{k-1}, y_1, \dots, y_l, x_{k+1}, \dots, x_n \rangle, \\ \langle x_1, \dots, x_{k-1}, i_\delta(y_1, \dots, y_l), x_{k+1}, \dots, x_n \rangle \in A_\gamma \}$$

A set of state operations Δ is applied to a BSD γ as follows:

$$\Delta \triangleright \gamma \equiv \delta_n \triangleright \left(\dots (\delta_2 \triangleright (\delta_1 \triangleright \gamma)) \right) \text{ with } \bigcup_{\delta_i \in \{\delta_1, \dots, \delta_n\}} \delta_i = \Delta \text{ and } n = |\Delta|$$

A set of state operations Δ is then applied to a set of BSDs Γ as follows:

$$\Delta \triangleright \Gamma \equiv \bigcup_{\gamma_i \in \Gamma} \Delta \triangleright \gamma_i$$

Two BSDs can be merged into one if their sets of affecting elements are equal. This operation is commutative.

Definition 7 (Merging BSDs). Two BSDs γ_1 and γ_2 are merged as follows:

$$\gamma_1 \bullet \gamma_2 \equiv \begin{cases} \langle E_{\gamma_1}, A_{\gamma_1} \cap A_{\gamma_2} \rangle & \text{if } E_{\gamma_1} = E_{\gamma_2} \\ \{\gamma_1, \gamma_2\} & \text{else} \end{cases}$$

Two sets of BSDs Γ_1 and Γ_2 are then merged as follows:

$$\Gamma_1 \bullet \Gamma_2 \equiv \bigcup_{\langle \gamma_i, \gamma_j \rangle \in \{\Gamma_1 \times \Gamma_2\}} (\gamma_i \bullet \gamma_j)$$

To merge two state operations, we need to consider whether they have the same affected element, and whether the first state operation's affected element is part of the affecting elements of the second one:

- If the first characteristic applies, the state operations are merged into one.
- If the second characteristic applies, the first state operation's affecting elements substitute its affected element in the set of affecting elements of the second operation, and the second state operation's image function is replaced by a composition of both image functions.

Note that both characteristics might apply at the same time. If neither characteristic applies, both state operations remain unchanged.

Definition 8 (Merging State Operations). Two state operations δ_1 and δ_2 with δ_1 preceding δ_2 are merged as follows:

$$\delta_1 \dagger \delta_2 \equiv \begin{cases} \delta_2 & \text{if } \theta_{\delta_2} = \theta_{\delta_1} \wedge \theta_{\delta_1} \notin E_{\delta_2} \\ \{\delta_1, \delta_2\} & \text{if } \theta_{\delta_2} \neq \theta_{\delta_1} \wedge \theta_{\delta_1} \notin E_{\delta_2} \\ \langle \theta_{\delta_1}, E', i_{\delta_2} \circ i_{\delta_1} \rangle & \text{if } \theta_{\delta_2} = \theta_{\delta_1} \wedge \theta_{\delta_1} \in E_{\delta_2} \\ \{\delta_1, \langle \theta_{\delta_2}, E', i_{\delta_2} \circ i_{\delta_1} \rangle\} & \text{if } \theta_{\delta_2} \neq \theta_{\delta_1} \wedge \theta_{\delta_1} \in E_{\delta_2} \end{cases}$$

with $E' = (e_{\delta_{2_1}}, \dots, e_{\delta_{2_{k-1}}}, E_{\delta_1}, e_{\delta_{2_{k+1}}}, \dots, e_{\delta_{2_n}})$ where $\theta_{\delta_1} = e_{\delta_{2_k}}$

Two sets of state operations Δ_1 and Δ_2 with Δ_1 preceding Δ_2 are then merged as follows:

$$\Delta_1 \dagger \Delta_2 \equiv \bigcup_{\langle \delta_i, \delta_j \rangle \in \{\Delta_1 \times \Delta_2\}} (\delta_i \dagger \delta_j)$$