# Business Process Model Abstraction: Theory and Practice

Sergey Smirnov[1], Hajo A. Reijers[2], Thijs Nugteren[2], and Mathias Weske[1]

[1] Hasso Plattner Institute, Potsdam, Germany
{sergey.smirnov, mathias.weske}@hpi.uni-potsdam.de
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
h.a.reijers@tue.nl,g.m.nugteren@student.tue.nl

**Abstract.** Business process management aims at capturing, understanding, and improving work in organizations. The central artifacts are process models, which serve different purposes. Detailed process models are used to analyze concrete working procedures, while high-level models show, for instance, handovers between departments. To provide different views on process models, business process model abstraction has emerged. While several approaches have been proposed, a number of abstraction use case that are both relevant for industry and scientifically challenging are yet to be addressed. In this paper we systematically develop, classify, and consolidate different use cases for business process model abstraction. The reported work is based on a study with BPM users in the health insurance sector and validated with a BPM consultancy company and a large BPM vendor. The identified fifteen abstraction use cases reflect the industry demand. The related work on business process model abstraction is evaluated against the use cases, which leads to a research agenda.

## 1 Introduction

With an ever increasing interest to BPM, organizations report to have hundreds or even thousands of process models at their disposal. Often such process models are reused long after their initial creation, since they can be employed for a multitude of purposes [1]. The models are of value, for example, to train new employees, to identify performance improvement opportunities, to align conflicting views of stakeholders on business operations, and to demonstrate an organization's compliance with external regulations.

Along with the different usages of a process model, the need emerges to bring important parts of the process model to the foreground, while parts less relevant for the present purpose should be hidden. This principle, which we shall refer to as Business Process Model Abstraction (BPMA), is especially wholesome in those situations, where process models become very large. After all, it has been argued that it becomes difficult for end users to comprehend process models containing more than 50 elements [14].

Although work on BPMA exists, e.g., see [5, 20, 29], this paper is motivated by two open problems. First of all, we note considerable confusion about what

BPMA actually entails. Different terms are in use for similar concepts and there is little reflection on related work, which makes it hard to distinguish between existing BPMA techniques. Secondly, given this fragmented state of the art, it is hard to determine how existing techniques address actual end user needs.

The contribution of this work is a comprehensive and precise view on BPMA. While the discussion of different levels of decision making in process model abstraction relates to the comprehensiveness, the formalization of the involved operations contributes to the precision. In addition, the paper presents a catalog of fifteen use cases. The use cases have been gathered and validated in close cooperation with industrial partners from the end user, consulting, and software vendor domains. The catalog (1) illustrates the value of BPMA, (2) helps to categorize existing BPMA techniques, and (3) displays the mismatches between the available BPMA techniques and the industrial demand.

The structure of the paper is as follows. Section 2 formalizes the domain of BPMA. Section 3 empirically approaches BPMA presenting a catalog of use cases validated by BPM experts. In Section 4 the catalog is used to match the existing research against the industry demand in BPMA. Section 5 concludes the paper.

## 2 Towards BPMA Formalization

Although BPMA has been discussed by several research papers, the term still requires formalization. This section starts with an informal BPMA introduction and concludes with its formal definition and properties discussion. As BPMA is an engineering problem dealing with models, we refer to the MetaObject Facility (MOF)—a standard for model-driven engineering which organizes (meta-) modeling artifacts into 4 levels [18]. We use MOF to illustrate the roles of main BPMA artifacts. Further, we propose a framework for BPMA, organizing the related methods and questions.

### 2.1 BPMA and (Meta-) Modeling

Informally, business process model abstraction is an operation on a business process model preserving essential process properties and leaving out insignificant details in order to retain information relevant for a particular purpose. We postulate a finite non-empty set of process models $M$ and an infinite non-empty set of process instances $I$. A mapping $inst : M \rightarrow \mathcal{P}(I)$ sets up a correspondence between a process model and the set of instances it describes. Sets of instances $inst(m)$ partition the set of process models $M$ into equivalence classes: models of one class describe the set of instances $inst(m)$. For a process model $m \in M$ there is a set of abstract process models, where each model describes the set of instances $inst(m)$, but with less detail: $abstr : M \rightarrow \mathcal{P}(M)$. Hence, model $m$ and its abstractions $abstr(m)$ belong to one equivalence class of models describing $inst(m)$. If the user possesses model $m$, any abstract model $m_a \in abstr(m)$ provides no new information about $inst(m)$. Although one process model may

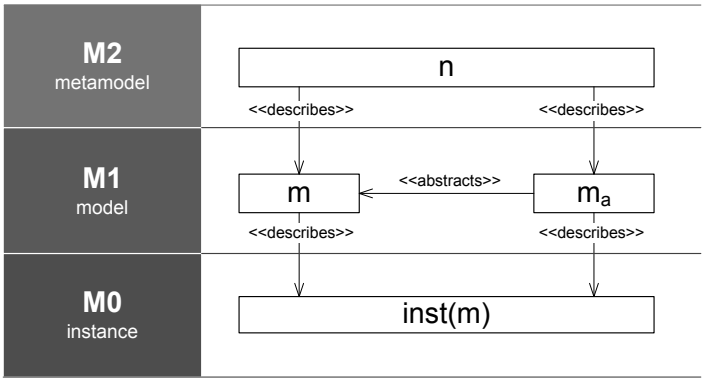


**Fig. 1.** Allocation of BPMA concepts on MOF levels

have many abstractions, in the further discussion we refer to a process model $m$ and its one counterpart, abstract process model $m_a \in abstr(m)$.

To give the reader a better insight into the relations between the described BPMA artifacts, we allocate them on different levels of MOF and show their relations (see Fig. 1). In this way we reuse the established vocabulary and the formalism of MOF. A set of process instances $inst(m)$ related to process model $m$ is allocated to level $M0$. The business process model $m$ is put on level *M1*, as it describes/models set of instances $inst(m)$. Process model $m$ conforms to the modeling notation in which it is described—metamodel $n$. The result of abstraction, an abstract process model $m_a \in abstr(m)$, belongs to *M1*. Model $m_a$ describes the set of instances $inst(m)$. Notice that we require models $m$ and $m_a$ to conform to one metamodel. For instance, if the detailed process model is created using Business Process Modeling Notation (BPMN) [19], an abstract process model conforms to BPMN as well. In the general case, models $m$ and $m_a$ may adhere to different notations. To achieve this, the BPMA must not only abstract insignificant information, but also guarantee that the abstract model $m_a$ adheres to its metamodel.

One can observe that BPMA is a modeling act itself. In this case $m$ is the system and $m_a$ is the model of $m$. According to [28], a model has three features: a mapping feature, a reduction feature, and a pragmatic feature. These three features are essential to BPMA:
*Mapping feature* BPMA is an operation on an initial process model,
*Reduction feature* BPMA leaves out insignificant process details,
*Pragmatic feature* BPMA retains information relevant for a particular purpose.

### 2.2 BPMA Framework

By now we have identified the main artifacts of BPMA and the relations between them. However, we did not describe why, when, and how an abstraction operation is performed. These questions have been partially studied in [20]. In the current paper we propose a BPMA framework systematically organizing these questions

and enabling their formal discussion. Rather than creating the framework from scratch, we reuse the knowledge of cartographic generalization, a discipline existing for centuries. Cartographic generalization is the process of selecting and representing information of a map in a way that adapts to the scale of the display medium. Hence, cartographic generalization copes with the problem resembling the BPMA problem.

There exist several cartographic generalization models, e.g., [6, 13, 17]. We adopt the overall structure of the first comprehensive generalization model focused on digital generalization as proposed by McMaster and Shea in [13]. McMaster and Shea claim that cartographic generalization consists of three components: a consideration of objectives of *why* to generalize; a cartometric evaluation of the conditions that indicate *when* to generalize; a selection of spatial and attribute transformations providing techniques on *how* to generalize.

**Why.** The why aspect of BPMA considers the reason why a process model should be abstracted, i.e., the goal of a process model abstraction. The abstraction goal is driven by the purpose of an abstract process model and its intended audience. On the one hand, BPMA stakeholders vary from a technical specialist, interested in a particular technical perspective of a process, to a manager, seeking for a high-level business process overview. On the other hand, even one user may demand a whole spectrum of abstraction scenarios. For instance, a manager may be interested in activities which have a high execution cost or in the paths in the model that are executed most often. The purposes and the audience of these scenarios are different, and, so the goals are.

Depending on an abstraction goal, different objects attract the user's attention. For instance, if the user wants to see how a process utilizes a data object, an abstraction technique has to analyze all model elements and present only elements accessing this data object. In another scenario the user may want to observe "expensive" process instances by means of a model. In this scenario an abstraction mechanism has to analyze all the paths in a process model and select those, which depict expensive instances. Disregarding a concrete scenario, each BPMA focuses on a set of objects of one type, where each object is treated as an atomic entity during abstraction. Atomicity means that the whole object is either relevant or irrelevant. While relevant objects are preserved, irrelevant are abstracted. We refer to these objects as *abstraction objects* and postulate an alphabet of all abstraction objects $\Omega$. For the two given examples abstraction objects are model elements and process instances. An abstraction goal defines an *abstraction criterion*—a property of an abstraction object that enables object comparison and allows the identification of objects relevant for the task at hand. Examples of abstraction criteria are the activity execution cost and process instance frequency.

**When.** The next question of BPMA is when to abstract objects. An abstraction criterion allows for a comparison of abstraction objects. Subsequently, an abstraction criterion classifies abstraction objects of model $m$ into *significant* and *insignificant*. We formalize this classification with the function $sign : \Omega \rightarrow \{true, false\}$.

If an abstraction criterion has an ordinal scale, the classification into significant and insignificant elements can be realized by an *abstraction threshold value*. The threshold value partitions the set of model elements into two classes: elements with a criterion value greater or equal to the threshold, and the rest. One of these classes is considered to be significant, while the other—insignificant (the choice depends on the concrete abstraction goal). [20] proposes an *abstraction slider*, which is an implementation of the function *sign*.

**How.** The how question of BPMA explores methods that enable transformation of an initial process model into a more abstract process representation. We distinguish *basic abstraction operations*. A basic abstraction operation allows to abstract from one insignificant abstraction object. While the basic abstraction operation is defined on the model level, Definition 1 makes use of an auxiliary function $\alpha'_o$. The auxiliary function sets up correspondences between abstraction objects of $m$ and $m_a$ and allows to judge about the properties of basic abstraction operations. Further we use $O \subset \Omega$ and $O_a \subset \Omega$ to reference the sets of abstraction objects in models $m$ and $m_a$, respectively.

**Definition 1 (Basic abstraction operation).** A function $\alpha_o : M \to M$ transforming process model $m$ into model $m_a$ is a *basic abstraction operation*, if it abstracts from an insignificant abstraction object $o \in O \wedge sign(o) = false$, so that:

- $|O| > |O_a|$, where $O, O_a \subset \Omega$ are the sets of abstraction objects in models $m$ and $m_a$, respectively;
- $\alpha_o$ is associated with an auxiliary function $\alpha'_o : O \backslash \{o\} \to O_a$;
- $\alpha'_o$ is a surjection.

Two prominent examples of basic abstraction operations are *elimination* ($\pi$) and *aggregation* ($\sigma$).

**Definition 2 (Elimination operation).** A basic abstraction operation $\pi_o : M \to M$ is an *elimination operation*, if $|O| = |O_a| + 1$ and auxiliary function $\pi'_o$ is a bijection.

Elimination is a transitive, asymmetric, and, hence, irreflexive operation. Elimination produces a model containing no information about the omitted abstraction object $o$, while other abstraction objects are preserved. In contrast, aggregation preserves information about the abstraction object $o$.

**Definition 3 (Aggregation operation).** A basic abstraction operation $\sigma_o : M \to M$ is an *aggregation operation*, if an extension of auxiliary function $\sigma'_o$ to set $O$ is a non-injective surjection.

Aggregation is transitive, asymmetric, and irreflexive. Aggregation produces an abstract model, where an insignificant abstraction object $o$, together with several other abstraction objects, is represented with a newly introduced abstraction object $o'$. Object $o'$ inherits the properties of objects it aggregates. For instance, if two sequential activities are aggregated into one activity, properties of the new activity comprise properties of the aggregated activities: the execution cost of an

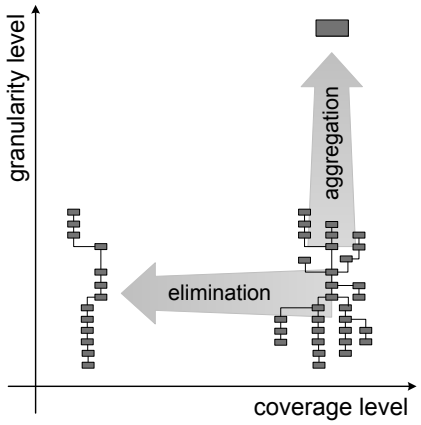


**Fig. 2.** Comparison of aggregation and elimination

aggregating activity can be defined as the sum of execution costs of aggregated activities.

Fig. 2 compares the effects of elimination and aggregation operations. Aggregation increases the granularity of model elements. In the ultimate case the whole business process can be described with one high-level activity. Elimination omits model elements, but does not change their granularity level. Hence, elimination and aggregation enable navigation along two orthogonal (independent) axes: the granularity level of model elements and the coverage level of a business process by a model.

A BPMA is realized as a composition of basic abstraction operations. Basic abstraction operations are applied, until every insignificant abstraction object is handled.

**Definition 4 (Business process model abstraction).** Business process model abstraction is an operation $\alpha : M \rightarrow M$ transforming process model $m$ into model $m_a$ such that $\alpha = \alpha_{o_l} \circ \alpha_{o_{l-1}} \circ \ldots \circ \alpha_{o_1}$ is the function composition, where:

- $\forall o \in O_a : sign(o) = true \wedge (\nexists k < l, \forall o \in O_k : sign(o) = true)$,
- $\alpha_{o_1}$ is a basic abstraction operation $\alpha_{o_1}(m) = m_2, o_1 \in O \wedge sign(o_1) = false$,
- for $k = 2 \ldots (l-1)$, $\alpha_{o_k}$ *is a basic abstraction operation* $\alpha_{o_k}(m_k) = m_{k+1}$, $o_k \in O_k \wedge sign(o_k) = false$,
- $\alpha_{o_l}$ is a basic abstraction operation $\alpha_{o_l}(m_l) = m_a, o_l \in O_l \wedge sign(o_l) = false$.

Notice that Definition 4 implicitly expresses the abstraction goal in the abstraction objects and the significance function. If the BPMA is a composition of elimination and aggregation, it is a transitive, asymmetric, and irreflexive operation. We distinguish several properties of BPMA that have to be considered when the abstraction method is selected. As an intrinsic property of BPMA is information loss, an abstract model contains less ordering constraints than its detailed counterpart. At the same time, each abstraction use case and the underlying

abstraction goal define a tolerance level to the loss of ordering constraints. While there are order-preserving abstractions, localizing the lost ordering constraints within an abstracted fragment, others are more tolerant to ordering constraints loss (see [22]).

Companies often use process models to analyze operational business processes, for example to analyze their cost or bottlenecks. Model elements of the models supporting such an analysis are annotated with additional information, e.g., activity execution time, hand-off times, and activity execution probabilities. We refer to this additional information as *non-functional properties*. If the user considers abstract models in the analysis, a BPMA has to ensure that the analysis of model $m_a$ delivers the same results as the analysis of model $m$. If the BPMA fulfills this requirement, we call it an abstraction that preserves non-functional properties.

## 3 BPMA Use Case Catalog

In this section, we discuss a catalog of BPMA use cases which are identified with the help of BPM experts. First, we explain the method that has been applied to derive and validate the use cases. Next, we present the initial version of the catalog used as the input for the validation stage. Then, we discuss the feedback that we received during the validation stage and summarize the modified use case catalog.

### 3.1 Catalog Design

In order to understand the user demand for BPMA techniques we referred to the expertise of our industry partners. As the problem of BPMA is relatively new, we followed an exploratory approach and conducted a series of semi-structured interviews with BPM experts. The study was separated into the two phases of (1) generation and (2) validation, which overall involved three categories of stakeholders, i.e., end users, consultants, and software developers.

In the *first phase* we considered BPMA use cases that emerged out of a joint project with a large German health insurance company, AOK. The goal of the project was to develop BPMA techniques enabling a fast comprehension of large business process specifications containing, for example, more than 300 nodes. The BPMA use cases were retrieved and elaborated in interviews with various AOK employees: a business process leader, a coordinator of IT infrastructure for process management, a BP knowledge manager and three process modelers. All these employees are interested in BPMA as end users of a set of over 4,000 process models. The use cases derived from the interviews were complemented by use cases from the literature. The outcome of the first phase were 14 use cases organized into four groups.

In the *second phase* the use cases were validated by ten professionals at Infosys, an Indian information technology services company with a specific focus on BPM, and eight employees of Pallas Athena, a Dutch software vendor producing BPM

systems. All Infosys employees fulfill a role as BP consultant and their experience with BPM had an average value of 6.5 years. The spectrum of job descriptions of the interviewees at Pallas Athena varied from software engineer to chief executive officer. The BPM experience of the participants within this group had an average value of 11.5 years. The primary goal in this phase was to reflect on the relevance of the initial set of use cases. Secondly, we encouraged the interviewees to generate new use cases. The output of the second phase was a validated use case catalog. In comparison with the initial set, one use case was dropped and two new use cases were added.

### 3.2 Initial Use Cases

The set of initial use cases that were derived from the first phase of our exploratory approach will be discussed in this section by distinguishing four groups, each of which contains use cases that have similar properties.

**Group 1: Preserving Relevant Activities** The user analyzes a business process captured by a process model. The model specifies numerous activities. However, the user wants to focus on activities that are significant for the task at hand. The distinction between what are significant and insignificant activities is based on the threshold value of a non-functional property of these activities. All the activities with a value for this property that is lower than the threshold are insignificant and these are eliminated. The use cases in this group share that they have the *activity* as abstraction object and *elimination* as a basic abstraction operation. The ordering constraints between the significant activities are *preserved*, while the use of elimination leads to a *change* of the non-functional properties of the overall process. We distinguish four BPMA use cases that belong to this group.

**Use Case 1: Preserve Pricey Activities** The user optimizes a business process and is interested in the activities with a high execution cost.

**Use Case 2: Preserve Frequent Activities** The user improves a business process and focuses on frequently executed activities.

**Use Case 3: Preserve Long Activities** The user is interested in process optimization and focuses on activities with a high duration.

**Use Case 4: Show High Hand-off Times** The user optimizes a business process and focuses on activities with high hand-off times.

**Group 2: Preserving Relevant Process Instances** The user analyzes a business process described by a precise model specifying the life cycle for a wide variety of process instances. The user does not want to know about each process instance, but needs to focus on a specific subset of instances. We call such instances significant. The significant process instances are visualized in the process model as paths. A BPMA eliminates the paths corresponding to insignificant process instances and preserves the paths describing significant ones. To summarize, the use cases in this group have *process instances* as an

abstraction object, have *elimination* as a basic abstraction operation, *preserve* the ordering constraints among the significant abstraction objects, and *do not* allow to preserve the non-functional properties of the overall process. We have encountered the following use cases.

**Use Case 5: Preserve Pricey Instances** The user optimizes a process and considers costly process instances as significant. She specifies a cost threshold, distinguishing significant process instances from insignificant ones: process instances with an execution cost that is higher than the threshold value are significant, the rest are not.

**Use Case 6: Preserve Frequent Instances** The user performs process optimization and considers frequent process instances as significant. By means of an instance execution frequency threshold, the user distinguishes significant instances from insignificant ones. The instances with an execution frequency higher than the threshold are considered to be significant, while the rest are insignificant.

**Use Case 7: Preserve Instances with Long Duration** The user optimizes the process and considers paths with long durations as significant. She specifies a path execution duration threshold value, distinguishing significant instances from insignificant ones: the instances with execution times higher than the threshold are important, while instances with lower execution times are unimportant.

**Use Case 8: Trace a Case** The user is interested in the question how special cases evolve in a business process. For instance, she wants to know how orders with a cost higher than 1000 euros unfold. Hence, the user specifies a case to be traced and obtains a model capturing only the significant process evolutions.

**Group 3: Filtering of Model Elements** The process model in possession of the user is overspecified for the task at hand. Only a subset of model elements is relevant and have to be disclosed. In contrast to the use cases of Group 1, the significance of model elements is determined according to their qualitative properties. To simplify model comprehension, irrelevant model elements are eliminated. The relevant elements are preserved, as well as the ordering constraints between them. The use cases of this group exhibit common properties: abstraction objects are *model elements* and a basic abstraction operation is *elimination*. The ordering constraints between significant model elements are *preserved*, while non-functional properties of the overall process are *changed*.

**Use Case 9: Adapt Process Model for an External Partner** The user adapts an existing business process model for the presentation to an external partner. The available model either captures confidential, internal process details, or details which are of no interest to the partner. The user manually marks model elements, which are relevant for inter-organizational collaboration and which are significant.

**Use Case 10: Trace Data Dependencies** The user modifies a data object interface. Beforehand she needs to know which data dependencies exist in

the business process. Hence, the significant model elements are those that access the data object of interest.

**Use Case 11: Trace a Task** The user evaluates the effect of an activity in a process model. To achieve this, a transitive closure of model elements dependent on this activity has to evaluated. Model elements of this closure are significant, while other model elements are not.

**Group 4: Obtaining a Process Quick View** The user needs a business process overview for fast process comprehension. The available model is a process specification formalizing every minor detail. A study of this model is time consuming and is not necessary for the ongoing work. The user needs a representation of this business process on a higher level, capturing more coarse-grained activities and overall information about the ordering constraints. For all of the use cases in this group, *activities* are abstraction objects. *Aggregation* is the basic abstraction operation. While Use Case 12 and 13 aim to *preserve* the ordering constraints, Use Case 14 does *not* consider the ordering constraints. Similarly, as the non-functional properties of the process are *preserved* by Use Case 12 and Use Case 13, Use Case 14 does *not* allow to preserve them. The following use cases belong to this group.

**Use Case 12: Get Process Quick View Respecting Ordering Constraints** The user needs a process specification, capturing coarse-grained activities, as well as the ordering constraints between them. She does not know in advance which abstraction level is sufficient and wants to control this level gradually. The user wants to preserve non-functional properties of the process.

**Use Case 13: Get Process Quick View Respecting Roles** Activities performed by a special role, e.g., *Manager*, are considered to be significant. The rest of activities are not. Insignificant activities are aggregated into coarse-grained ones, significant activities are preserved as is, and the ordering constraints are preserved where possible. Non-functional properties of the process, e.g., execution time or execution cost, should be preserved.

**Use Case 14: Retrieve Coarse-grained Activities** The user wants to grasp the coarse-grained activities that appear in the business process. She does not require an abstraction mechanism to deliver the ordering constraints between the high level activities: once these activities are available, she can manually order them.

### 3.3 Use Case Validation

During the validation phase of the catalog design, each participant received a booklet that described the initial set of use cases. The participants were asked to study these descriptions and the researchers were available for clarification. Each participating BPM expert expressed her demand for each of the presented use cases. To express her opinion, the participant had three options. If the participant found the use case important and the intended abstraction approach helpful, she could mark the use case with a *yes*. If the participant saw no value in the

presented use case, she could answer *no*. If the participant had doubts about the relevance of the use case, she was able to respond with *undecided*. For the evaluation we encoded the responses: positive responses correspond to *1*, negative responses were encoded with *-1*, whilst neutral answers—with *0*. Participants had the opportunity to give comments and discuss the use cases with the researchers.

**Relevance and Completeness.** Table 1 presents the aggregated values of the response codes. As can be seen, the table differentiates between the two groups of stakeholders: consultants and (software) vendors. However, the opinions of the two groups are highly consistent, with the notable exception of "Use Case 8: Trace a Case". The latter use case is favorably perceived by consultants (total score of 7), while vendors are neutral to it (total score of 0). Overall, the use cases "Use Case 6: Preserve Frequent Instances" and "Use Case 12: Get Process Quick View Respecting Ordering Constraints" find the most outspoken support. The former is associated with finding a so-called "happy path" in the process or its "sunny day scenario". The latter use case is interpreted by most participants as the type of abstraction that is most in demand.

Surprisingly, the participants seem to differentiate between use cases that exploit the *same* abstraction technique, but operate with *different* non-functional properties of model elements. This is most vividly illustrated by the contrast between the values for "Use Case 1: Preserve Pricey Activities" and "Use Case 2: Preserve Frequent Activities". Whilst the former use case is of no interest for interviewees (score of 0), the latter is in high demand (score of 13). A less pronounced differentiation can be observed for "Use Case 5: Preserve Pricey Instances" and "Use Case 6: Preserve Frequent Instances". We conclude that *frequency* is perceived as a more natural abstraction criterion by users. Furthermore, these observations highlight the importance of an explicit abstraction criterion choice.

A study of Table 1 also reveals that use case 14 is a clear outlier. This use case is the only one that completely neglects control flow: it exclusively delivers a set of activities to the user. We deduce that for the BPMA stakeholders *ordering constrains* are of vital importance and belong to the essential model information to be preserved. Hence, we interpret "Use Case 14: Retrieve Coarse-grained Activities" as an example of a *false* BPMA use case and drop it from the final catalog.

| Category \ Use case ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Consultant (10) | 1 | 6 | 4 | 5 | 4 | 8 | 4 | 7 | 5 | 3 | 3 | 8 | 6 | -1 |
| Vendor (8) | -1 | 7 | 1 | 7 | 7 | 8 | 6 | 0 | 3 | 3 | 5 | 8 | 5 | -1 |
| Total | 0 | 13 | 5 | 12 | 11 | 16 | 10 | 7 | 8 | 6 | 8 | 16 | 11 | -2 |

**Table 1.** Support of BPMA use cases by interviewees

During the evaluation of use cases that belong to Group 1 the participants noticed that the elimination of insignificant activities often leads to unacceptable information loss. Instead of eliminating insignificant activities, the interviewees saw benefits of aggregating them. We summarize these user requests in a new use case.

**Use Case 15: Preserve Frequent Activities Summarizing Rare Activities** The user analyzes a process captured in a detailed process model. She has to focus on activities relevant for the current analysis. The distinction between significant and insignificant activities bases on the threshold value of an activity frequency: the activities with a frequency value lower than the threshold are insignificant. Significant activities are preserved as-is, while insignificant activities are aggregated, when possible.

The introduction of this use case raises the issue whether a whole new family of use cases should be created that is based on the initial members of Group 1. However, despite the external similarity to the use cases of Group 1, such new use cases would heavily rely on the technique needed for "Use Case 13: Get Process Quick View Respecting Roles". As such, we decided not to pursue this larger extension.

Interviewees also pointed to BPMA scenarios where only model elements relevant for a certain perspective, e.g., a business perspective or a data flow perspective, are presented to the user. Notice that this abstraction depends on the existence of information that is relevant to make this distinction in the initial process model. Abstractions of this type belong to Group 4: Filter Model Elements. We formulate the user demand in the following use case:

**Use Case 16: Get Particular Process Perspective** The user analyzes a process model captured in a detailed process model. She wants to see a particular process perspective. Model elements which belong to the desired perspective are significant and preserved in the model as-is. Model elements which do not belong to this perspective are insignificant and are eliminated.

No needs for further use cases were found. In sum, this leads us to a final set of $14 - 1 + 2 = 15$ use cases.

### 3.4 Additional Insights.

While the second phase in our validation approach mainly aimed at the relevance and completeness of the use case catalog, the discussions with the involved participants raised additional insights. First of all, other visualization techniques came forward as important *alternatives* to deal with some of the use cases. In particular, we can distinguish the following techniques that were discussed:

**Highlighting:** Instead of completely abstracting from model objects that do not need to be visualized, it is also possible to *highlight* the objects that deserve attention, for example by coloring these or changing their shape. The main advantage is that it provides the context of the highlighted objects. A good example where this could be useful is "Use Case 6: Preserve Frequent Instances", where a "happy path" is highlighted *within* the process model.

**Tagging:** Depending on the exact use case, it may be important to see *more* rather than less information in a process model, which is the objective of BPMA. Such additional information could be presented as tags, annotations or even icons that are added to existing process model elements. For instance, in the context of "Use Case 13: Get Process Quick View Respecting Roles" it could also be useful to see relevant role information along with tasks in the model.

**Animation:** While BPMA is focused on the static representation of process model content, for some use cases a more *dynamic* representation mode is desirable. Specifically, for the use cases in Group 2 (Preserving Relevant Process Instances) it is useful to see how a particular process evolution unfolds step-by-step.

**Textual Reporting:** For the considered use cases, it is not always important to obtain the information that one seeks in the form of a process model. Instead, a textual or tabular enumeration can suffice. Recall that we dropped "Use Case 14: Retrieve Coarse-grained Activities" as a use case for BPMA, even though the participants can imagine the intended overview to be relevant in the form of a tabular visualization.

This overview is by no means meant as comprehensive, but it puts the importance of BPMA into the right perspective. After all, it would be improper to consider BPMA as the only viable way to present relevant information in a process model. Yet, we argue that the value of BPMA in comparison with other techniques can be explicitly found in use cases that involve *very large* process models. For all the alternatives we discussed, one can foresee a range of problems in such cases. For example, if highlighting is applied in an extremely large process model, it will become difficult to distinguish let alone focus on the emphasized objects.

A final insight relates to the specific feedback of one of the participants, who argued that he did not see value in BPMA for any of the proposed use cases. He explained that in his environment a strictly hierarchical modeling approach is employed, such that each process is modeled on five different levels of granularity (using subprocesses). Therefore, the BPMA techniques add limited additional value with respect to navigating through these levels. Clearly, it is open to debate whether switching between subprocesses can provide exactly the same insights as the BPMA techniques do. Yet, it is important to realize that built-in features of process models can already greatly contribute to an improvement of large process model understanding. This is also in line with our earlier work on the value of modularity [24].

## 4 Use Case Catalog as a Research Compass

The derived catalog of use cases systematically describes BPMA from an application perspective. However, it neither specifies the algorithms that enable the realization of the use cases, nor shows how these algorithms can be employed for BPMA. In this section, we reflect on how the discussed use cases are supported by existing techniques. Furthermore, by considering the available gaps we identify which use cases and which aspects are calling for further research.

| Use case name | Why | When | How | |
|---|---|---|---|---|
| | | | Model Transformation | Abstraction Algorithm |
| Use Case 1–4 | | | [2, 3, 8, 9, 15, 16, 25] | [4, 5, 12, 26] |
| Use Case 5–8 | | | [2, 3, 8, 9, 15, 16, 25] | |
| Use Case 9 | [10] | [10] | [10] | [10] |
| | | | [2, 3, 8, 9, 15, 16, 25, 30, 31] | [4, 5, 12, 26] |
| Use Case 10–11 | | | [2, 3, 8, 9, 15, 16, 25, 30, 31] | [4, 5, 12, 26] |
| Use Case 12 | | | [7] | [7] |
| | [11, 21] | [11, 21] | [11, 21] | [11, 21] |
| | | | [2, 3, 8, 9, 15, 16, 25, 30, 31] | [4, 5, 12, 26, 22, 23, 27] |
| Use Case 13,15 | | | [2, 3, 7–9, 11, 15, 16, 21, 25, 30, 31] | [4, 5, 12, 26, 27] |
| Use Case 16 | | | [2, 3, 8, 9, 15, 16, 25] | [4, 5, 12, 26] |

**Table 2.** Existing techniques related to identified BPMA use cases

### 4.1 Retrospective

Table 2 provides correspondences between the use cases from the catalog and existing techniques for BPMA. A table row specifies the papers related to one or several use cases from the catalog. The columns distinguish the papers according to the three BPMA aspects: *why*, *when*, and *how*. The *how* aspect further refines the classification into papers on process model transformation and papers specifying how to apply abstraction algorithms for BPMA.

Among the papers on process model transformation, work is available on reduction rules and process model decomposition that typically do not specifically target BPMA. However, these papers do provide a theoretical basis for the realization of abstraction. As Table 2 expresses, we argue that both *reduction* techniques, see [2, 3, 8, 9, 15, 16, 25], and *decomposition* techniques, see [30, 31], have the potential to support elimination and aggregation as the most prominent forms of abstraction. In the context of BPMA, for every set of reduction rules it is necessary to show that one of the following statements holds:

- The set of reduction rules is complete to abstract a process model of an arbitrary structure into one node.
- There is a description of the class of models that can be reduced to one node by this set of rules.

As in practice process models have an arbitrary, non-compositional structure, the above requirements are highly relevant to reflect on the applicability of a set of reduction rules. Decomposition approaches, see [30, 31], developed by Vanhatallo et al. are free of this limitation.

The papers that describe how to use model transformation techniques for BPMA by no means always use this label, but rather refer to developing *process views*, see [4, 5, 10], or focus on *process simplification*, see [11]. However, the essential purpose of these techniques is in line with the way we characterized

BPMA in this paper. While in a number of papers, e.g., [4, 5, 12, 26], generic BPMA techniques are discussed, others address concrete use cases, see [10].

Table 2 allocates several papers on dedicated rows by which we emphasize the role of these works. [10, 11, 21] propose *comprehensive* solutions for particular BPMA use cases. By this, we mean that the papers discuss all the three aspects of BPMA. We emphasize [7], as it discusses not only a structural perspective on model transformations, but also the principles for the evaluation of non-functional properties.

### 4.2 Perspective

Another look at Table 2 reveals a disproportion in the related work: as the *how* is thoroughly investigated, the *why* and *when* are hardly touched. The *why* calls for research on how the user can formulate the abstraction goal and what is the frontier of BPMA application. The *when* question is concerned with the definition of the *sign* function, which can be non-trivial, e.g., consider Use Case 9.

Furthermore, even the referenced techniques provide only partial support for some of the use cases. For instance, although [10] proposes a BPMA approach covering all the aspects of abstraction, the approach is only capable of handling block-structured process models. Similarly, a BPMA technique that is developed in [21] is restricted by a set of rules that enable this abstraction. Finally, in [11] Günther and Van der Aalst propose an approach that supports Use Case 12, but it is only capable of handling process models in a very simplistic notation. While the contributions of all these papers are acknowledged, it is also apparent that their applicability can be enhanced.

Table 2 illustrates that BPMA has been studied by a number of researchers and that various techniques have become available in the past years. Yet, there is still considerable room for improvement and extension, on the one hand by tackling the almost unexplored *why* and *when* aspects and one the other hand by extending the range of advanced techniques addressing the *how*.

## 5 Conclusions and Future Work

The contribution of this paper is twofold. First of all, it provides a clarification and formalization of the domain of business process model abstraction. To do so, we explained the basic concepts behind BPMA as well as their inter-relations using the terms of model-driven engineering. Also, we proposed a framework organizing BPMA aspects.

Secondly, the paper provides an insight into the industry demand for BPMA techniques. The main deliverable here is the BPMA use case catalog. The catalog reflects the demand for BPMA from practice and captures 15 use cases identified through a study of the literature, our own practical experiences, and additional interactions with industry experts. We demonstrated how the catalog can be used to relate the identified use cases against the related work. The comparison reveals well studied areas, as well as unexplored fields. Among other benefits,

this comparison enabled us to formulate a research agenda. Overall, the paper delivers an integral view on BPMA in the sense that it is precise, comprehensive, and relevant.

We identify several directions of the future work. Naturally, *why* and *when* are on the agenda. While for the *why* it is important to investigate goals of abstraction and properties of abstraction objects, the *when* misses a study of the *sign* function. As we argued earlier, the *how* also has its white spots. For instance, a high user demand for Use Case 12 is a strong motivation to develop techniques delivering aggregations of activities that semantically belong together. Here, by semantics we mean the domain semantics of the model elements. A related problem is how to label an aggregating activity delivered by the abstraction. Finally, it is interesting to determine when BPMA techniques are preferable over alternative visualization techniques and textual reports.

## Acknowledgments

The authors acknowledge the support of the following industry partners: AOK Brandenburg in Teltow, Germany; Infosys in Bangalore, India; and Pallas Athena, Apeldoorn, The Netherlands. The authors thank Tassilo Glander for sharing his research expertise on map visualization.

## References

1. J. Becker, M. Kugeler, and M. Rosemann. *Process Management: A Guide for the Design of Business Processes*. Springer Verlag, Berlin, Germany, 2003.
2. G. Berthelot. Checking Properties of Nets using Transformation. In *Advances in Petri Nets 1985*, pages 19–40, London, UK, 1986. Springer.
3. G. Berthelot. Transformations and Decompositions of Nets. In *Advances in Petri nets 1986*, pages 359–376, London, UK, 1987. Springer.
4. R. Bobrik, Th. Bauer, and M. Reichert. Proviado—Personalized and Configurable Visualizations of Business Processes. In *EC-Web*, pages 61–71, 2006.
5. R. Bobrik, M. Reichert, and Th. Bauer. View-Based Process Visualization. In *BPM 2007*, volume 4714 of *LNCS*, pages 88–95, Berlin, 2007. Springer.
6. K. E. Brassel and R. Weibel. A Review and Conceptual Framework of Automated Map Generalization. *International Journal of Geographical Information Science*, 2(3):229–244, 1988.
7. J. Cardoso, J. Miller, A. Sheth, and J. Arnold. Modeling Quality of Service for Workflows and Web Service Processes. Technical report, University of Georgia, 2002. Web Services.
8. J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, New York, NY, USA, 1995.
9. B. van Dongen, M. Jansen-Vullers, H. Verbeek, and W. M. P. van der Aalst. Verification of the SAP Reference Models Using EPC Reduction, State-space Analysis, and Invariants. *Comput. Ind.*, 58(6):578–601, 2007.
10. R. Eshuis and P. Grefen. Constructing Customized Process Views. *Data Knowl. Eng.*, 64(2):419–438, 2008.

11. C. W. Günther and W. M. P. van der Aalst. Fuzzy Mining—Adaptive Process Simplification Based on Multi-perspective Metrics. In *BPM 2007*, volume 4714 of *LNCS*, pages 328–343, Berlin, 2007. Springer.
12. D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *Information Systems*, 28(6):505–532, 2003.
13. R. B. McMaster and S. K. Shea. Generalization in Digital Cartography. In *Resource Publication of the Association of American Geographers*, Washington D.C., USA, 1992.
14. J. Mendling, H. A. Reijers, and W. M. P. van der Aalst. Seven Process Modeling Guidelines (7pmg). *Information and Software Technology*, 52(2):127–136, 2010.
15. J. Mendling, H. Verbeek, B. van Dongen, W. M. P. van der Aalst, and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data Knowl. Eng.*, 64(1):312–329, 2008.
16. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
17. B. G. Nickerson and H.R. Freeman. Development of a Rule-based System for Automatic Map Generalization. In *ISSDH*, pages 537–556, Seattle, Washington, USA, January 1986.
18. OMG. *Meta Object Facility (MOF) Core Specification*, 2.0 edition, January 2006.
19. OMG. *Business Process Modeling Notation*, 1.2 edition, January 2009.
20. A. Polyvyanyy, S. Smirnov, and M. Weske. Process Model Abstraction: A Slider Approach. In *EDOC 2008*, pages 325–331, 2008.
21. A. Polyvyanyy, S. Smirnov, and M. Weske. Reducing Complexity of Large EPCs. In *EPK'08 GI-Workshop*, Saarbrücken, Germany, 11 2008.
22. A. Polyvyanyy, S. Smirnov, and M. Weske. On Application of Structural Decomposition for Process Model Abstraction. In *BPSC 2009*, pages 110–122, Leipzig, 2009.
23. A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *BPM 2009*, pages 229–244, Ulm, Germany, 2009. Springer.
24. H. A. Reijers and J. Mendling. Modularity in Process Models: Review and Effects. In *BPM 2008*, pages 20–35, Milan, Italy, 2008. Springer.
25. W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.
26. M. Shen and D. Liu. Discovering Role-Relevant Process-Views for Recommending Workflow Information. In *DEXA*, pages 836–845, 2003.
27. S. Smirnov. Structural Aspects of Business Process Diagram Abstraction. In *International Workshop on BPMN*, pages 375–382, Vienna, Austria, July 2009.
28. H. Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973.
29. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. *LNCS*, 3649:205, 2005.
30. J. Vanhatalo, H. Völzer, and J. Koehler. The Refined Process Structure Tree. In *BPM 2008*, pages 100–115, Milan, Italy, September 2008. Springer.
31. J. Vanhatalo, H. Völzer, and F. Leymann. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In *ICSOC 2007*, volume 4749 of *LNCS*, pages 43–55. Springer, 2007.