Distributed and Parallel Databases manuscript No.
(will be inserted by the editor)

Business Process Model Abstraction: A Definition, Catalog, and Survey

Sergey Smirnov · Hajo A. Reijers · Mathias Weske · Thijs Nugteren

the date of receipt and acceptance should be inserted later

Abstract The discipline of business process management aims at capturing, understanding, and improving work in organizations by using process models as central artifacts. Since business-oriented tasks require different information from such models to be highlighted, a range of abstraction techniques has been developed over the past years to manipulate overly detailed models. At this point, a clear understanding of what distinguishes these techniques and how they address real-world use cases has not yet been established. In this paper we systematically develop, classify, and consolidate the use cases for business process model abstraction and present a case study to illustrate the value of this technique. The catalog of use cases that we present is based on a thorough evaluation of the state of the art, as well as on our cooperation with end users in the health insurance sector. It has been subsequently validated by experts from the consultancy and tool vendor domains. Based on our findings, we evaluate how the existing business process model abstraction approaches support the discovered use cases and reveal which areas are not adequately covered, as such providing an agenda for further research in this area.

Keywords business process modeling · process model management · model abstraction · process model views

Sergey Smirnov · Mathias Weske
Hasso Plattner Institute, 2-3 Prof.-Dr.-Helmert-Strasse, Potsdam, D-14482, Germany
E-mail: {sergey.smirnov,mathias.weske}@hpi.uni-potsdam.de
Tel.: +49 (0)331-5509-194, Fax: +49(0)331-5509-189

Hajo A. Reijers
Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, PO Box 513, NL-5600 MB Eindhoven, The Netherlands
E-mail: h.a.reijers@tue.nl

Thijs Nugteren
Deloitte Consulting BV, PO Box 300, NL-1180 AH Amstelveen, Laan van Kronenburg 2, Amstelveen, The Netherlands
E-mail: tnugteren@deloitte.nl

1 Introduction

A modern business process is an inherently distributed system: its activities are performed by various employees, on different locations, using a heterogeneous set of IT systems. Furthermore, a business process typically crosses the borders of organizational departments and even companies. Hence, processes are complex artifacts that challenge managers in their efforts to properly govern them. Against this background, business process models are key artifacts to represent how work is performed in organizations. A broad spectrum of management tasks can be supported with such models, e.g., the documentation, evaluation, and improvement of business processes. While some models are useful to configure workflow technology [17], others are used to train new employees, identify performance improvement opportunities, align conflicting views of stakeholders on business operations, and demonstrate an organization's compliance with external regulations [2]. Obviously, this variety of modeling goals requires a modeler to focus on the relevant aspects of the distributed business process.

As, traditionally, for each modeling goal a *specific* process model was designed, companies maintain large process model repositories consisting of hundreds or even thousands of models. The stored models have complex interrelations: they may overlap, describe processes that subsume each other, or describe one process from different perspectives. Models that formalize the *same* business process typically vary in the level of abstraction, so that along with detailed models also more coarse-grained models are maintained. As such models are stored independently, it is hard to keep them in sync. Each change of the process need to be applied to all its models, which incurs a significant overhead.

To solve this problem, business process model abstraction (BPMA) has been proposed recently, e.g., see [7,15,33,43]. The general idea is to develop a detailed process model and to provide views on it using abstraction mechanisms. Hence, BPMA sets the scope of the model and focuses on the relevant aspects of the complex distributed system the model represents. For instance, BPMA may aim at delivering the model that reveals the process evolution within one organizational department. Another example is the model that visualizes how particular resources are used throughout the process. However, the work reported in the literature puts the focus mainly on abstraction techniques, rather than on the specific use cases for abstraction. As a result, the academic view on BPMA is quite narrow, overlooking a number of both challenging and relevant research questions.

Against this backdrop, the contributions of this paper are as follows. Firstly, this work provides a comprehensive and precise view on BPMA. In particular, our discussion of the different levels of decision making for the act of process model abstraction arguably results in a *comprehensive* treatment of the subject, while the formalization of the involved operations contributes to its *precision*. The second contribution of this paper is a catalog of fifteen use cases. The use cases have been gathered and validated in close cooperation with industrial partners from the health insurance, 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, which leads to research opportunities. The third and final contribution is a survey of the existing techniques available in the BPMA field.

The structure of the paper is as follows. Section 2 formalizes business process model abstraction. Section 3 empirically explores the application of BPMA and presents a catalog of use cases as validated by BPM professionals. To illustrate the use of BPMA in a real-life setting, we provide a case study in Section 4. In Section 5 the use case catalog is used to match the existing research against the industry demand in BPMA. Section 6 concludes the paper with a summary and discussion.

2 Towards BPMA Formalization

Although BPMA has been discussed in several research papers, the notion lacks a precise description. This section starts with an informal BPMA introduction and concludes with its formal definition and a discussion of its properties. As BPMA is an engineering problem dealing with models, we refer to the Meta Object Facility (MOF)—a standard for model-driven engineering which organizes (meta-) modeling artifacts into 4 levels [30]. We use the MOF to illustrate the relation between the main elements that play a role in BPMA. Further, we propose a framework for BPMA, organizing the related methods and questions.

2.1 BPMA and (Meta-) Modeling

Informally, business process model abstraction can be seen as an operation on a business process model that preserves process properties that are essential for a particular purpose, while it leaves out insignificant details. To further pin down this notion, we postulate a finite non-empty set of process models M and an infinite non-empty set of process instances I. A mapping $inst : M \to \mathcal{P}(I)$ sets up a correspondence between a process model and the set of instances it describes. 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: $abstr : M \to \mathcal{P}(M)$. 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 have many abstractions, in the further discussion we refer to a process model m and its single counterpart, abstract process model $m_a \in abstr(m)$.

To give the reader a better insight into the relations between the described aspects of BPMA, we allocate them to different levels of the MOF and show their relations, see Fig. 1. In this way we reuse the established vocabulary and the formalism of the 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

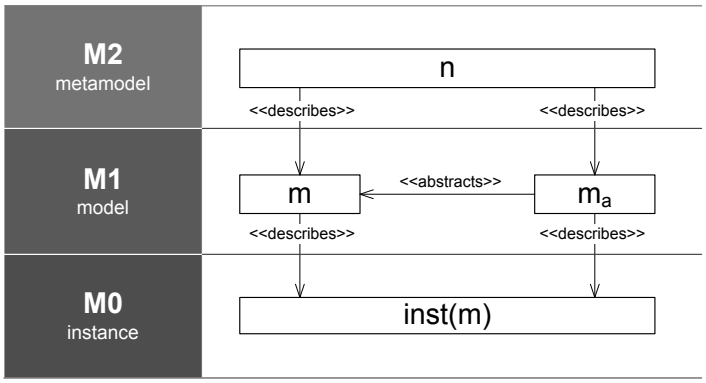


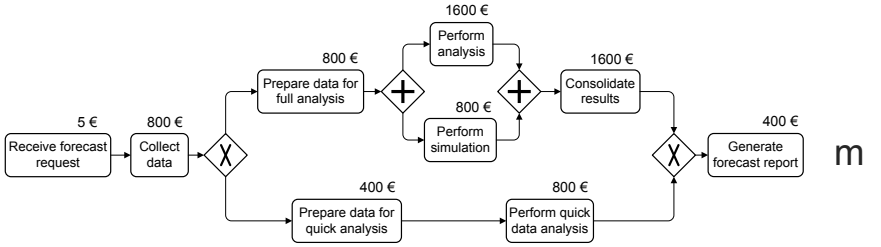
Fig. 1 Allocation of BPMA concepts on MOF levels

on level *M1*, as it describes/models a set of instances $inst(m)$. Process model m conforms to the modeling notation in which it is described—metamodel n. The process model $m_a \in abstr(m)$ is an abstraction of m and also belongs to level *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 the Business Process Modeling Notation (BPMN) [31], an abstract process model conforms to BPMN as well. However, in the general case, models m and m_a may adhere to different notations.

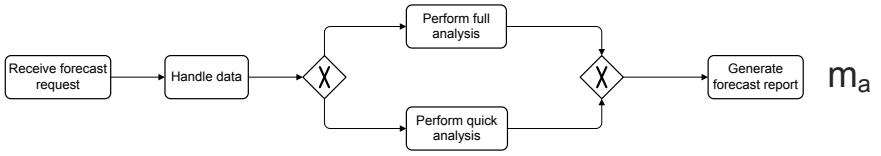
2.2 BPMA Framework

By now we have discussed the various aspects of BPMA in an abstract way. However, we did not address in detail what the goal of abstraction is, when it is to be applied, and how abstraction is exactly performed. These issues have been partially studied in [33]. In the current paper, we propose a BPMA framework systematically organizing these aspects 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 a problem that resembles that of BPMA.

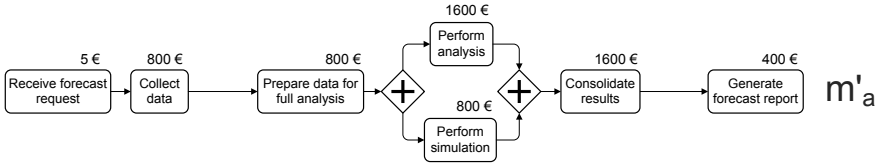
There exist several cartographic generalization models, e.g., [8,24,28]. We adopt the overall structure of the first comprehensive generalization model focused on digital generalization as proposed by McMaster and Shea in [24]. McMaster and Shea claim that cartographic generalization consists of three 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. We will consider these components in the context of BPMA, which will help us to arrive at a precise understanding of what BPMA entails.



(a) Initial model m



(b) Model with high-level activities m_a



(c) Model with the most expensive path m'_a

Fig. 2 Abstraction object examples

Why. The why aspect of BPMA considers the reasons for abstracting a process model, 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 may vary from technical specialists, interested in a particular technical perspective of a process, to managers, who are seeking a high-level business process overview. On the other hand, even one user alone may demand a whole spectrum of abstraction scenarios. For instance, a manager may both be interested in activities which have a high execution cost *and* in the paths through the model that are executed most often. The purposes and the stakeholders of these scenarios are different, and so are the goals.

Depending on an abstraction goal, different objects attract the user's attention. Consider a simplified illustrative example in Fig. 2. Model m describes a business process, where a forecast request is processed. Once a forecast request is received, the required data is collected. Then, there are two options: either to perform a full or a quick data analysis. The process concludes with a forecast report creation. Notice that each activity is annotated with an estimate of its execution cost. One abstraction scenario captures a user's demand for a high-level process outline, i.e., a model describing coarse-grained activities of the process as well as the ordering constraints between them. Model m_a is an example of such a process overview that is discoverable from model m. In

comparison to activities of model m, activities of m_a are more abstract and each of these comprises a set of activities from the initial model. For instance, activity *Perform quick analysis* corresponds to the set {*Prepare data for quick analysis, Perform quick data analysis*} in m. Thereby, in this scenario the user focuses on the granularity change of activities. In another abstraction scenario the user may want to observe the process instances that are likely to be expensive to execute by means of a model. In such a scenario an abstraction mechanism has to disclose all the paths in a process model and select those that represent costly instances. Model m'_a presents the result of abstraction addressing such a user demand: among two alternative paths the most expensive is preserved.

In general, it can be said that 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 ones are abstracted from. We refer to both types of objects as *abstraction objects* and postulate a finite non-empty set of all abstraction objects Ω. For the two given examples, we noted as abstraction objects activities and process instances respectively. 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. Abstraction criteria that appeared in the discussed examples are activity execution cost and process instance frequency respectively.

When. The next component of BPMA deals with the conditions under which abstraction objects are affected. An abstraction criterion allows for a comparison of abstraction objects. Subsequently, an abstraction criterion classifies abstraction objects of model m into *significant* and *insignificant* ones. We formalize this classification with the function $sign : \Omega \rightarrow \{true, false\}$.

If an abstraction criterion displays at least 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). [33] proposes an *abstraction slider*, which is an implementation of the function *sign*.

How. The how component of BPMA covers the method that enables the transformation of an initial process model into a more abstract process representation. At the lowest level, we distinguish *basic abstraction operations*. A basic abstraction operation allows to abstract from a single insignificant abstraction object. We will argue that any abstraction can be seen as a composition of basic abstraction operations. While the basic abstraction operation is defined on the model level, Definition 1 makes use of an auxiliary function α'_o as well. 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. Furthermore, we will 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;*
- *α_o is associated with an auxiliary function $\alpha'_o : O \backslash \{o\} \to O_a$;*
- *α'_o is a surjection.*

Two prominent, concrete examples of basic abstraction operations are *elimination* (π) and *aggregation* (σ).

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 π'_o is a bijection.*

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 σ'_o to set O is a non-injective surjection.*

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 aggregating activity can be defined as the sum of execution costs of aggregated activities.
Fig. 3 compares the effects of elimination and aggregation operations. Aggregation decreases the granularity of the process model, i.e., it makes a process model more coarse-grained. 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.

As stated, a concrete BPMA is then 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 $\alpha : M \to M$ 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:*

- *$\forall o \in O_a : sign(o) = true \wedge (\nexists k < l, \forall o \in O_k : sign(o) = true)$,*
- *α_{o_1} is a basic abstraction operation $\alpha_{o_1}(m) = m_2, o_1 \in O \wedge sign(o_1) = false$,*

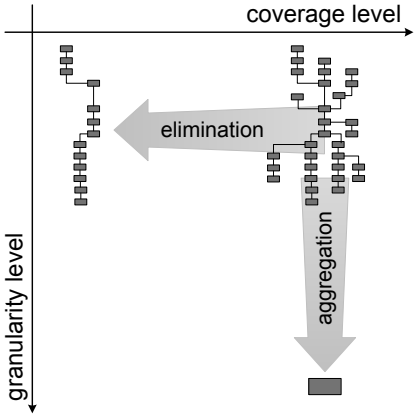


Fig. 3 Comparison of aggregation and elimination

- *for* $k = 2 \ldots (l-1)$, α_{o_k} is a basic abstraction operation $\alpha_{o_k}(m_k) = m_{k+1}$, $o_k \in O_k \wedge sign(o_k) = false$,
- α_{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 deals with the abstraction goal by referencing the abstraction objects and a significance function.

To complete our reflection on BPMA, we distinguish two additional aspects that have to be explicitly considered when an abstraction method is selected in practice. First of all, as an intrinsic property of BPMA is information loss, an abstract model contains fewer ordering constraints than its detailed counterpart. Depending on the exact abstraction use case and the underlying abstraction goal, the tolerance level for the loss of ordering constraints may differ. While there are order-preserving abstractions, localizing the lost ordering constraints within an abstracted fragment, others are more tolerant to ordering constraints loss (see [35]). In other words, the importance of control flow preservation must be taken into account.

Secondly, *non-functional properties* may be more or less important to be preserved when applying BPMA. 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. 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.

At this point, we have arrived at a formal and complete view of BPMA. Furthermore, we have stressed the context-specific importance of preserving the control-flow and non-functional properties in a process model when applying

BPMA. We will be using the various notions that have been introduced in this section in the structuring of a catalog of BPMA uses cases in the next section.

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 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 literature study includes papers from the reputable conferences on business process management and information systems, e.g., the *International Conference on Business Process Management* and the *International Conference on Service Oriented Computing,* as well as journals, e.g., *IEEE Transactions of Software Engineering* and *Information Systems* within the timespan of the last decade.

In the *second phase* the use cases were validated by involving two further companies: Infosys, an Indian information technology services company with a specific focus on BPM, and Pallas Athena, a Dutch software vendor developing BPM systems. From these parties, ten and eight professionals participated in this study respectively. All of the involved Infosys employees fulfill a role as BP consultant; 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 that of software engineer to the 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 of 14 cases, one use case was dropped and two new use cases were added leading to a total of 15 use cases in the end.

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. In this discussion, we will use the notions as introduced in Section 2 to characterize the various groups of use cases. Specifically, the description of each group contains the central abstraction object, the used abstraction criterion, the basic abstraction operation being involved, and the importance of preserving a model's control-flow and non-functional properties.

3.2.1 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 the significant and insignificant activities are 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.

3.2.2 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.

3.2.3 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.

3.2.4 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* aim 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, each 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.

3.3.1 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 the vendor representatives take a neutral stance (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 not much 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.

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

Furthermore, these observations highlight the importance of an explicit choice for the abstraction criterion in question.

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.

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 15 use cases, which is one of the contributions of this paper.

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 brought forward:

1. **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.
2. **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.
3. **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.
4. **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. At the same time, we do 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 of the alternatives we listed, 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, according to this participant, 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 of course greatly contribute to an improvement of large process model understanding. This is also in line with our earlier work on the value of modularity [38].

4 Case Study

Whilst the previous section explored the BPMA problematics in breadth, this section provides an in depth study. Such a shift in the argumentation reveals the technical aspects of BPMA that complement the high-level discussion of the earlier sections. We present a case study on the design of a BPMA technique motivated by a specific industry demand and supporting use case "Use Case 12: Get Process Quick View Respecting Ordering Constraints". In particular, we report on the solution developed to support the BPM initiative at AOK Brandenburg—a large health insurance company located in Teltow, Germany. We start by describing the project initial setting and the goal in Section 4.1. Furthermore, in Section 4.2 we outline the developed solution and illustrate it by the example. The detailed technical description of the discussed BPMA approach can be found in [34].

4.1 Case Motivation

AOK captures its operational processes in about 4 000 EPCs. The designers are not limited to creation of block-structured process models, i.e., the EPCs are arbitrarily structured. The models are enriched with information about the time required to complete each activity and probabilities of connection transitions from the source to the target. AOK uses such process models to evaluate the number of employees required to enact all process instances. However, some of the models contain exhaustive details impeding their comprehension by humans. Figure 4.a presents an example of a process model from the AOK repository. The process model is composed of 333 nodes: 130 functions, 137 events, and 66 connectors. The existence of this model and similar ones within AOK have created a demand for the use of techniques that can help deliver more abstract process representations. In particular, AOK business analysts are interested in models that contain more coarse-grained activities. An important constraint here is that the BPMA to be applied has to deliver process models containing high-level activities summarizing the content of the initial model.

Obviously, the BPMA technique to be selected is framed by the mentioned requirements. Specifically, the abstraction should preserve the ordering constraints of a process model and the time required to complete the process. While

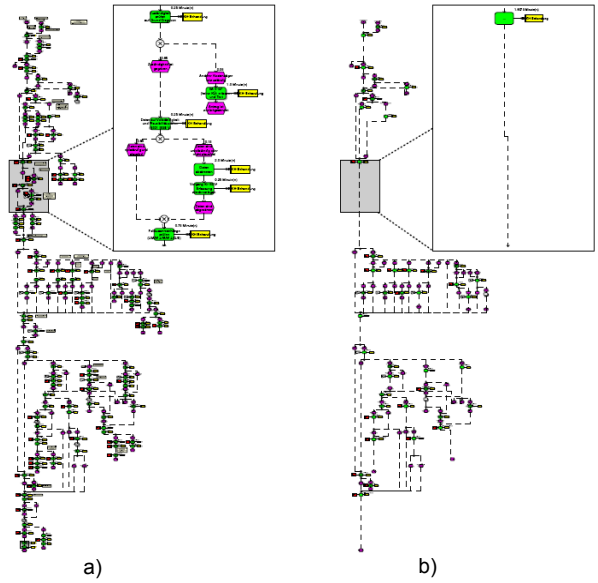


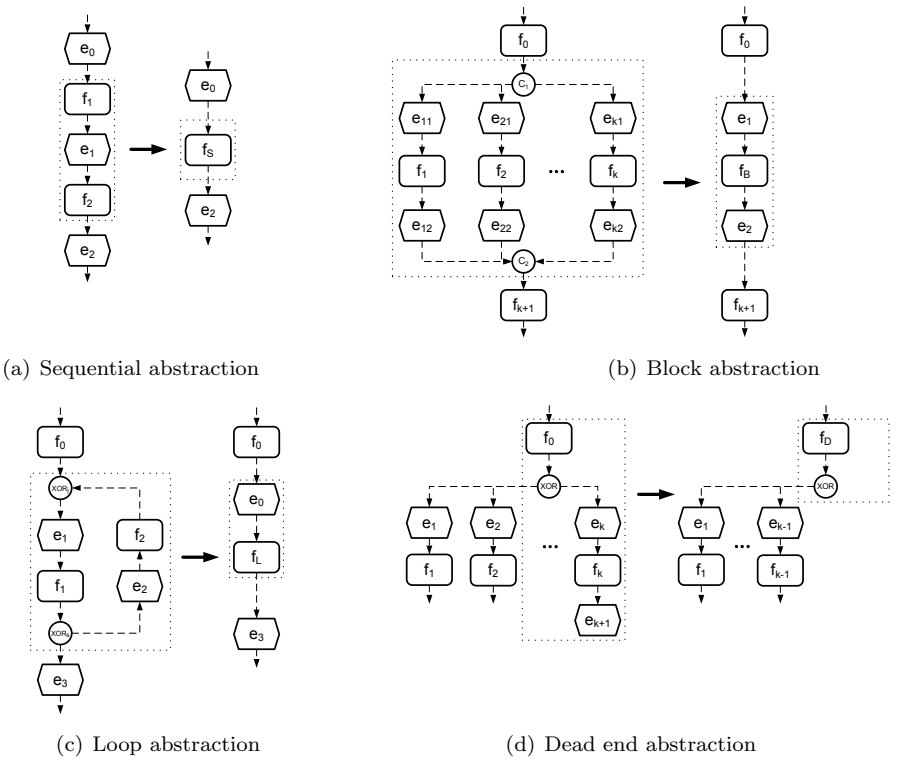
Fig. 4 Original (a) and abstracted (b) process models (label unreadability intended)

the former requirement is essential, the latter originates from the fact that process models are used in the head counting task, where process completion time is key. From a technical perspective, the BPMA should also deliver process models that are valid EPCs. The model's arbitrary structure motivated the development of a new BPMA technique: the available solutions handled only block-structured process models, see, e.g., [9,23]. The next section presents the designed solution.

4.2 Developed BPMA Technique

The developed BPMA considers activities as abstraction objects. The abstraction criterion is the activity completion time: activities consuming more time to complete are considered to be more significant to display. To preserve a meaningful notion of process completion during abstraction, aggregation is selected as the basic abstraction operation. The overall model transformation is then realized as a composition of aggregation operations. The aggregation operations are referenced as elementary abstractions and are distinguished into four types:

- sequential abstraction,
- block abstraction,
- dead end abstraction,
- loop abstraction.



(a) Sequential abstraction

(b) Block abstraction

(c) Loop abstraction

(d) Dead end abstraction

Fig. 5 Four elementary abstractions realizing the BPMA technique

Each elementary abstraction is associated with a certain type of EPC fragment and defines how this fragment is transformed. Fig. 5 illustrates the structural effect of elementary abstractions. The reader may wish to consult [34] for the detailed rules specifying how the process non-functional properties are re-evaluated. For instance, sequential abstraction describes how two sequential functions of an EPC can be aggregated into one function. The design of elementary abstractions allows to preserve the ordering constraints, as well as the process completion time. Given the elementary abstractions, the BPMA can be described as follows. First, insignificant process model elements are identified. Next, elementary abstractions are applied. For every insignificant activity to be abstracted, starting from the least significant one, the abstraction algorithm tries to apply transformation rules. If one of the elementary abstractions can be employed, an activity is aggregated. The aggregating activity is tested, whether it is significant or not. If it turns out that the aggregating activity is insignificant, it has to be abstracted in the subsequent steps. The algorithm works until all the activities in the model are significant, i.e., pass the threshold. It could also be the case that there are insignificant activities that cannot be reduced, as elementary abstractions can not handle them. This is the effect of the best effort principle. Therefore, the described abstraction mechanism

guarantees that it abstracts a process model at a best effort basis, bringing the process model either to the required abstraction level or to the state when no elementary abstraction operation can be applied.

Fig. 4.b shows the result of abstraction model shown in Fig. 4.a using the presented BPMA technique. After abstraction, the number of process model nodes was reduced to 167: 44 functions, 82 events, and 41 connectors. The overall reduction of process nodes is near 50% of its original size.

The proposed abstractions allow a company to deal with coarse-grained functions in business processes, while keeping the overall process logic intact. In terms of organization and management, these coarse-grained functions (with the associated execution effort being measured in minutes rather than of seconds) facilitate process improvement on a higher level. Tedious discussions on extremely low granularity functions are no longer required. Instead, process participants can apply improvements within the functions, keeping the overall process logic in sync with the process model.

The presented case study provides the insights into the technical details of BPMA. While the developed BPMA technique addresses use case "Use Case 12: Get Process Quick View Respecting Ordering Constraints", it also takes into account the specific requirements of the industry partner: order preservation and evaluation of non-functional properties in the process.

5 Related Work

The derived catalog of use cases as presented in Section 3 systematically describes BPMA from an application perspective. In the previous section, we have also illustrated how a specific BPMA is applied in a realistic setting and how the involved algorithm operates. However, a similar detailed understanding of all available techniques is missing at this point. In this section, we will reflect on how the use cases in our catalog are supported by the available techniques, building on our understanding of the state of the art. Furthermore, by providing the links between the use cases in the catalog and the available techniques we can identify which use cases and which aspects of BPMA are calling for further research.

5.1 BPMA State of the Art

Scientific papers that describe BPMA techniques by no means always use this exact label, but rather refer to developing *process views*, see [7,15], or focus on *process simplification*, see [18]. 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., [7,22,40], generic BPMA techniques are discussed, others address concrete use cases, see [15]. In this section we present an overview of the available BPMA techniques, which is preceded with a short technique summary.

Cardoso et al. [9] Propose a quality of service evaluation method for workflows enriched with information on transition probability and activity execution time/cost.

Liu and Shen [22,40] Suggest an order-preserving BPMA approach making use of reduction rules developed in [39].

Chiu et al. [10,11] Focuse on the BPMA in the context of cross-organizational interaction, where the generic model captures overall interaction, while abstract models are partner-specific.

Pankratius and Stucky [32] Adapt the principles of views in relational databases to the context of business process models delivering operations for constructing abstract process specification.

Günther and van der Aalst [18] Develop an abstraction technique for process models mined from logs, where the technique exploits metrics based on log information.

Bobrik et al. [5–7] Propose a BPMA approach with an emphasize on *how* aspect, specifying basic abstraction rules and their composition rules.

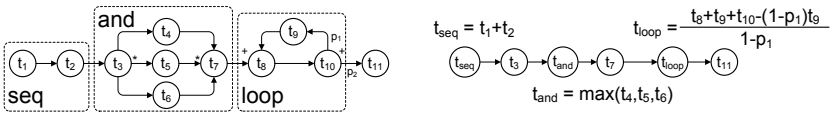
Eshuis and Grefen [15] Address the BPMA scenario, where an internal process model is adapted for an external partner in two steps: 1) the private details are concealed, 2) excessive information is hidden.

Polyvyanyy et al. [35,36] Argue that process model decomposition can be employed in BPMA and develop abstraction algorithms based on decomposition.

In the remainder, we will elaborate on each approach using the formalism introduced in Section 2. Also, each approach is positioned against the use cases elaborated in Section 3. To illustrate the approaches we employ examples in the notations of the original papers. After the discussion of each individual approach, we will provide a condensed view at the end of this section.

5.1.1 Cardoso et al.

In [9] Cardoso et al. evaluate the workflow quality of a service. The authors assume that every workflow activity is annotated with a non-functional property value, e.g., execution time or cost, and an execution probability. Quality of service for a workflow is evaluated through aggregation of workflow activities, where non-functional properties of an aggregating activity are determined by the properties of the aggregated ones. The paper considers activities as abstraction objects and utilizes aggregation as the basic abstraction operation. The proposed solution addresses the BPMA problem, in particular use cases of



(a) Initial model m (b) Abstract model m_a

Fig. 6 Illustration of the BPMA approach developed by Cardoso et al. in [9]

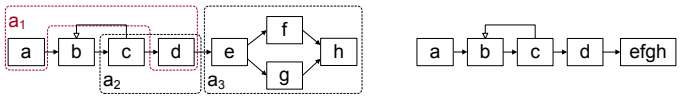
group "Group 1: Preserving Relevant Activities" and use cases "Use Case 12: Get Process Quick View Respecting Ordering Constraints" and "Use Case 13: Get Process Quick View Respecting Roles".

The paper particularly concentrates on the *how* component of BPMA. While the employed model is formalized as a graph, with nodes being tasks and edges being transitions between tasks, the process model is completely block-structured. In such a setting the authors specify the abstraction algorithm based on patterns and corresponding reduction rules. Once a workflow fragment is matched against a pattern, it is reduced according to the reduction rule. Four patterns are identified: sequence, AND block, XOR block, and two types of loop blocks. Fig. 6 demonstrates the application of these rules to model m in Fig. 6(a). The reduction rules for sequence, AND block, and the loop block are applied to fragments *seq*, *and*, and *loop*, respectively. The resulting abstract model is presented in Fig. 6(b). The reduction rules specify not only the structural transformations, but also the non-functional properties evaluation method.

5.1.2 Liu and Shen

In [22,40] Liu and Shen study the construction of process views and, in particular, order-preserving process views. The paper regards activities as the abstraction objects and employs aggregation as the basic abstraction operation. The proposed BPMA approach can support the use cases of group "Group 1: Preserving Relevant Activities" along with use cases "Use Case 9: Adapt Process Model for an External Partner", "Use Case 12: Get Process Quick View Respecting Ordering Constraints", "Use Case 13: Get Process Quick View Respecting Roles", and "Use Case 15: Preserve Frequent Activities Summarizing Rare Activities".

The authors provide a formal definition of a process model and specify a process execution semantics. The formalism allows for one type of model nodes, activities, that may realize the splitting and joining logic of ANDs and XORs. Loop dependency is considered as a special dependency type and process models may contain only single entry-single exit loops. Fig. 7 presents process model examples that adhere to the notation used in [22]. The paper elaborates on an algorithm for abstract process model construction based on the reduction rules proposed in [39]. The algorithm obeys three principles: activity membership, activity atomicity, and order preservation. The latter principle is of great practical importance and is thoroughly discussed in the



(a) Initial model m (b) Abstract model m_a

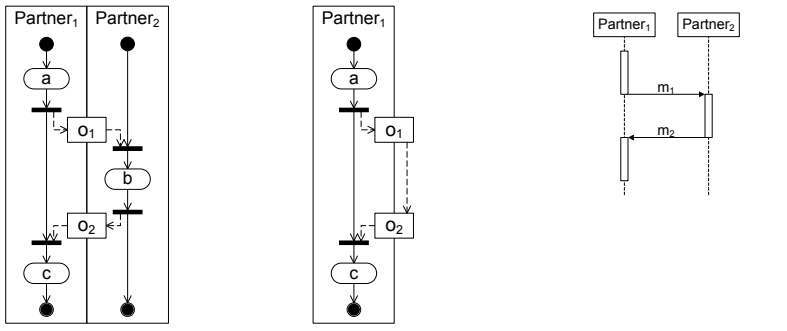
Fig. 7 Illustration of the BPMA approach developed by Liu and Shen in [22,40]

paper. The order preservation requires the BPMA to preserve the ordering constraints of the initial model in the abstract model. The example in Fig. 7(a) illustrates capabilities of the BPMA approach. While aggregations $a1$ and $a2$ violate the declared principles, $a3$ is valid. Fig. 7(b) presents the result of order-preserving abstraction: activities e, f, g, and h in the initial model are aggregated, see Fig. 7(a). While the paper elaborates on the BPMA *how*, it does not discuss the *why* and *when* questions.

5.1.3 Chiu et al.

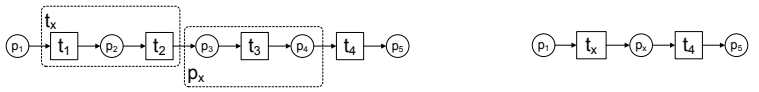
Chiu et al. discuss the BPMA in the context of cross-organizational interaction in a web service environment [10]. The authors concentrate on the use case, where an exhisting process model describes interorganizational interaction, while partner-specific models hiding confidential details are in demand. Effectively, this approach maps to "Use Case 9: Adapt Process Model for an External Partner". The BPMA approach selects activities as abstraction objects and uses elimination as the basic abstraction operation to deliver partner-specific views and narrow the scope within a party. The paper argues about the BPMA *how*, delegating the *why* and *when* questions to the human user.

Chiu et al. employ UML activity diagrams to capture processes and introduce a metamodel for workflow views. The metamodel defines a view as a graph with activities that can be organized into sequences, enriched with choice logic. Each activity can either be optional or iterative. Fig. 8 illustrates the approach: Model m in Fig. 8(a) is the initial model, while abstract model m_a in Fig. 8(b) captures the process from the point of view of one participant. The paper informally discusses the transition from the initial model to the abstract one. The authors argue that the partner interaction specified in the initial UML activity diagram can be captured on the high level by means of a sequence diagram. The paper gives the general idea of such a transformation. Fig. 8(c) presents an example of a sequence diagram that can be mined from



(a) Initial model m (b) Abstract model m_a (c) Abstract model m'_a

Fig. 8 Illustration of the BPMA approach developed by Chiu et al. in [10]



(a) Initial model m (b) Abstract model m_a

Fig. 9 Illustration of the BPMA approach developed by Pankratius and Stucky in [32]

the initial model m. Finally, the paper specifies consistency criteria for process model views with respect to the initial model.

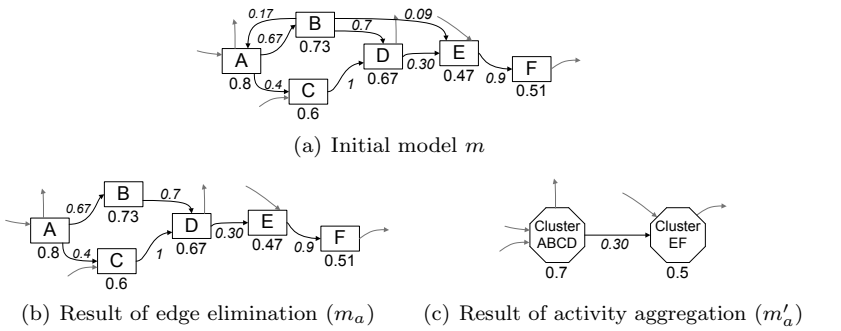
5.1.4 Pankratius and Stucky

Pankratius and Stucky relate process views to views in relational databases [32]. The authors define process models as Petri nets and adapt relational database operations to the Petri net formalism. This results in eight operations enabling process view creation: selection, difference, place projection, transition projection, place join, transition join, theta join, and union. The designed operations may support the BPMA approach, where activities and, potentially, events can be considered as abstraction objects. Elimination and aggregation can be realized through place projection and transition projection, respectively. Fig. 9 exemplifies an application of place projection and transition projection for a BPMA realization. Fig. 9(a) shows process model m, with two fragments, tx and px to be abstracted. Application of a transition projection to fragment tx and place projection to fragment px in m results in transition tx and place px in model m_a, respectively. Fig. 9(b) exhibits the abstract model m_a.

It should be noted that the developed operations consider only the structure of Petri nets, ignoring the execution semantics. As a consequence, important properties, e.g., soundness, of delivered process models may be violated [47]. In essence, the paper addresses the BPMA *how*. The developed BPMA technique supports use cases of group "Group 1: Preserving Relevant Activities" together with use cases "Use Case 9: Adapt Process Model for an External Partner", "Use Case 12: Get Process Quick View Respecting Ordering Constraints", "Use Case 13: Get Process Quick View Respecting Roles", "Use Case 15: Preserve Frequent Activities Summarizing Rare Activities", and "Use Case 16: Get Particular Process Perspective".

5.1.5 Günther and van der Aalst

In [18] Günther and van der Aalst investigate the simplification of "spaghetti-structured" process models as mined from event logs. The paper addresses all three aspects of BPMA: *why*, *when*, and *how*. The authors choose activities and edges as abstraction objects. The abstraction mechanism assumes the availability of substantial process logs enriched with activity and transition frequencies. The authors elaborate on how this information can be used within BPMA. They suggest metrics, e.g., activity frequency, distiniguishing significant abstraction objects from insignificant ones. The metrics are classified into



(a) Initial model m

(b) Result of edge elimination (m_a)

(c) Result of activity aggregation (m'_a)

Fig. 10 Illustration of the BPMA approach developed by Günther and van der Aalst in [18]

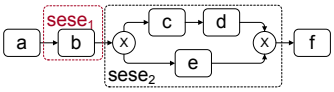
significance and *correlation* metrics and allow to orchestrate basic abstraction operations. Fig. 10 exemplifies the abstraction approach. The elements of initial model m are annotated with execution frequencies, see Fig. 10(a). In Fig. 10(b) the edges with low frequencies are eliminated, while Fig. 10(c) shows the aggregation of activity aggregation. The designed BPMA approach is potentially capable of supporting the use cases of groups "Group 1: Preserving Relevant Activities" and "Group 2: Preserving Relevant Process Instances" and use cases "Use Case 12: Get Process Quick View Respecting Ordering Constraints", "Use Case 13: Get Process Quick View Respecting Roles", "Use Case 15: Preserve Frequent Activities Summarizing Rare Activities", and "Use Case 16: Get Particular Process Perspective".

The authors formalize a process model as a graph, where nodes are activities, and edges—control flow relation. Unfortunately, such a simplistic model limits the applicability of the approach to some extent, as process modeling notations typically specify more than one node type. Moreover, the assumed availability of rich process logs is rather restrictive: Models are rarely enriched with such detailed execution information.

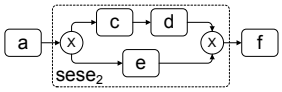
5.1.6 Bobrik et al.

Bobrik et al. study process views in [5–7]. The authors concentrate on the BPMA's *how* component, leaving the *why* and *when* out of scope. These works consider activities as the abstraction objects. The basic abstraction operations are aggregation and elimination. The developed BPMA method addresses use cases of group "Group 1: Preserving Relevant Activities" along with use cases "Use Case 9: Adapt Process Model for an External Partner", "Use Case 12: Get Process Quick View Respecting Ordering Constraints", "Use Case 13: Get Process Quick View Respecting Roles", "Use Case 15: Preserve Frequent Activities Summarizing Rare Activities", and "Use Case 16: Get Particular Process Perspective".

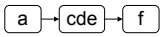
The process model is formalized as a graph with two node types, activities and gateways (subsequently typed to ORs, XORs, and ANDs), and the edges



(a) Initial model m



(b) Result of $sese1$ elimination (m_a)



(c) Result of $sese2$ aggregation (m'_a)

Fig. 11 Illustration of the BPMA approach developed by Bobrik et al. in [5–7]

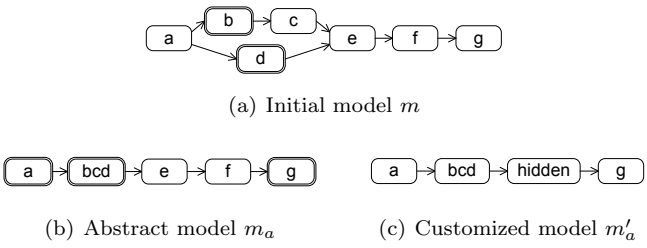
representing the control flow. Aggregation and elimination make use of the single entry single exit (SESE) fragments—fragments with exactly one incoming and exactly one outgoing edge. Elimination substitutes a SESE fragment with an edge, while aggregation—with an activity. The paper studies the BPMA properties, paying attention to control-flow preservation

Along with the model views, the authors discuss view construction for visualizations of process instances. The distinction of completed and not completed activities allows to extend basic abstraction operations beyond SESE fragment transformations: The completed activities are abstracted in a more flexible fashion. Fig. 11 illustrates the proposed BPMA approach. The initial model m is shown in Fig. 11(a). Two basic abstraction operations are sequentially applied. SESE fragment $sese1$ is eliminated, resulting in model m_a, see Fig. 11(b). Then, fragment $sese2$ is aggregated resulting model m'_a, see Fig. 11(c).

5.1.7 Eshuis and Grefen

In [15] Eshuis and Grefen are challenged by the adaptation of process models to support interorganizational communication. The designed approach has two steps: 1) the process owner specifies internal activities to be aggregated, 2) the process consumer omits and hides unnecessary activities. The selection of activities to be abstracted is manual. Thereby, the paper focuses on the BPMA *how*, ignoring *why* and *when*. The paper selects activities as the abstraction object. Aggregation and elimination are employed as basic abstraction operations. While the paper directly addresses use case "Use Case 9: Adapt Process Model for an External Partner", the developed BPMA approach indirectly supports the use cases of group "Group 1: Preserving Relevant Activities" and use cases "Use Case 9: Adapt Process Model for an External Partner", "Use Case 12: Get Process Quick View Respecting Ordering Constraints", "Use Case 13: Get Process Quick View Respecting Roles", "Use Case 15: Preserve Frequent Activities Summarizing Rare Activities", and "Use Case 16: Get Particular Process Perspective".

The processes are captured in UML Activity Diagrams. The formalization of a process model restricts models to block-structured ones, allowing AND and XOR blocks along with loops. The approach ensures that the resulting abstract models are order-preserving. The first phase is based solely on activity



(a) Initial model m

(b) Abstract model m_a (c) Customized model m'_a

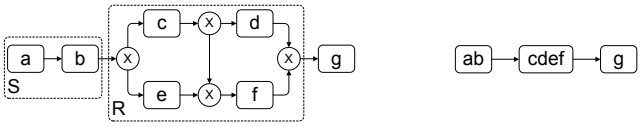
Fig. 12 Illustration of the BPMA approach developed by Eshuis and Grefen in [15]

aggregation, concealing the private activities. Fig. 12(a) captures the initial model m, where the user selects to aggregate activities b and d. Fig. 12(b) shows the abstraction result: activities b, c, and d are aggregated into bcd. Activity c is aggregated, as the BPMA approach constructs order-preserving views. The second phase, customization, employs aggregation and elimination to preserve only activities demanded by the consumer. In the example model m_a, see Fig. 12(b), the user preserves activities a, bcd, and g. Model m'_a in Fig. 12 shows the final result of abstraction, where irrelevant activities are hidden.

5.1.8 Polyvyanyy et al.

In [35,36] Polyvyanyy et al. study how process model decomposition supports BPMA. The developed BPMA approaches make use of aggregation as basic abstraction operation and choose activities as abstraction objects. The papers superficially discuss BPMA *why* and *when*, bringing *how* in the focus of the discussion. The developed BPMA technique supports use cases of group "Group 1: Preserving Relevant Activities", as well as use cases "Use Case 9: Adapt Process Model for an External Partner", "Use Case 12: Get Process Quick View Respecting Ordering Constraints", "Use Case 13: Get Process Quick View Respecting Roles", "Use Case 15: Preserve Frequent Activities Summarizing Rare Activities", and "Use Case 16: Get Particular Process Perspective".

The paper formalizes a process model as a graph, where nodes are activities and gateways (ANDs and XORs), and edges correspond to the control flow. However, the obtained results can be extended for BPMN, see [42]. While [35] decomposes the model into fragments with exactly one incoming and



(a) Initial model m (b) Abstarct model m_a

Fig. 13 Illustration of the BPMA approach developed by Polyvyanyy et al. in [35,36]

exactly one outgoing edge, [36] seeks for fragments having exactly one entry node and exactly one exit node. The latter approach results in more fine-grained decomposition making the BPMA more flexible. The proposed BPMAs are order-preserving. The approach is illustrated by the example in Fig. 13. Fig. 13(a) shows the initial model m, where fragments S and R can be aggregated. Fig. 13(b) presents the abstraction result with the given fragments being aggregated.

5.1.9 Summary

Table 2 summarizes the main properties of the aforementioned BPMA techniques. One can notice that the majority of BPMA approaches put the focus on the *how* component, leaving *why* and *when* mostly out of scope. However, even within the *how* aspect the focus is on the *structural* side of the model transformation. Typically, the decision which activities should be abstracted is delegated to the user. Furthermore, the business meaning of the coarse-grained activities created during abstraction has to be defined by the user as well. The question is whether without an more elaborate link with the *why* and *when* components it is possible for business users to adopt these techniques and link them to their needs.

As can be seen, activities dominate the "abstraction object" column of Table 2. This confirms that in practice the end users perceive activities to be in the center of BPMA. The observation is supported by the analysis of the use cases addressed by the existing techniques: The majority focuses on activities. Indeed, most approaches are capable of supporting use cases of group "Group 1: Preserving Relevant Activities", as well as use cases "Get Process Quick View Respecting Ordering Constraints" and "Get Process Quick View Respecting Roles".

Name	**Why**	**When**	**How**	π	σ	**Supported Use Cases**	**Abstraction Object**
Cardoso et al.	+	+	+	–	+	12, 13, 15	activity
Liu and Shen	–	–	+	–	+	1–4, 9, 12, 13, 15	activity
Chiu et al.	–	–	+	+	–	9	activity
Pankratius and Stucky	–	–	+	+	+	1–4, 9, 12, 13, 15, 16	model element
Günther and van der Aalst	+	+	+	+	+	1–8, 12, 13, 15, 16	edge, activity
Bobrik et al.	–	–	+	+	+	1–4, 9, 12, 13, 15, 16	activity
Eshuis and Grefen	–	–	+	+	+	1–4, 9, 12, 13, 15, 16	activity
Polyvyanyy et al.	–	–	+	–	+	1–4, 9, 12, 13, 15, 16	activity

Table 2 Existing BPMA techniques

Finally, Table 2 bears witness of the fact that the elimination π and aggregation σ are used relatively homogeneously. However, the aggregation prevails the table, being used in each BPMA technique, but one.

5.2 BPMA Related Papers

We recognize a series of papers that do not target BPMA specifically, yet notice that full-fledged BPMA can profit from their insights. In particular, we refer to the works on process model transformation, workflow inheritance, intermodel consistency, and model integration.

Among the papers on process model transformation, work is available on reduction rules and process model decomposition. Within decades the Petri net community studied reduction rule sets facilitating analysis of process models. In [3,4] Berthelot suggested a rule set capable of reducing live and bounded marked graphs to a single transition. Murata proposed reduction rules preserving the liveness, safeness, and boundedness properties in [27]. Desel and Esparza came up with a complete set of reduction rules for free-choice Petri nets, see [12]. [39] developed a set of graph reduction rules for identification of structural conflicts in process model. Recently, van Dongen and Mendling used reduction rule sets for analysis of process model soundness, see [14,26]. In the context of BPMA, for every set of reduction rules it is essential 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.

Process model decomposition approaches are free of this limitation: they enable unique decomposition of a process model into a hierarchy of fragments. The generic results for decomposition of graphs have been obtained by Johnson, Pearson, and Pingali in [19] and Tarjan and Valdes in [44]. Later Vanhatalo et al. adapted these decomposition techniques for business processes [46,45]. We argue that both reduction techniques and decomposition techniques have the potential to support elimination and aggregation as the most prominent forms of abstraction.

Preservation of process model behavior during model transformation can be defined in several ways. In [1] Van der Aalst and Basten use Petri nets as process formalism and define the notion of process inheritance, which relates to the behavioral aspect of the model. In particular, they identify four types of inheritance: protocol, projection, protocol/projection, and life-cycle inheritance. Further, the authors specify operations on models maintaining inheritance property. The reported work relates to BPMA problem, as models m and m_a can be seen as those belonging to one of the proposed inheritance relations.

While BPMA delivers an abstract process representation given a starting model, multiple models of one process might be already in place. Such models reflect the views of various stakeholders on one business process [20]. Typically, numerous relations exist between such models, which leads to new challenges in model management. Thereafter, the BPM community has thoroughly studied such models and their relations. In the remainder of this section we present the related work according to the type of considered intermodel relations and the associated model management task.

First, it is crucial to manage the consistency of multiple process specifications. In [13] Dijkman et al. addressed the problem of information system design, where multiple stakeholders contribute to the system creation. The authors developed a framework suggesting 1) the use of basic concepts shared by the stakeholders to be facilitate communication, 2) means to manage consistency between relations of system specifications. Further, the paper illustrates the applicability of the approach, establishing relations between structural and behavioral models. Recently, Weidlich et al. investigated the consistency relation of models formalizing one business process, see [48,49]. While [49] focuses on the behavioral aspects of process model, [48] studies methods for identification of correspondence relations between elements of different specifications. The existence of multiple models for one object is inherent not only for BPM, but also for software engineering and requirements engineering. Finkelstein et al. in [16] argued that model inconsistencies are inevitable and suggested a formal approach to deal with them. [29] suggested a framework for managing multiple views and their inconsistencies in the context of requirements engineering. The aforementioned papers study intra- and intermodel relations between model elements. We believe that their results can be useful in the context of BPMA *when* and *how*.

Second, the multiple models of a process can be seen as its "partial" models: each model presents one perspective on the subject. Integration of such partial models into one facilitates more comprehensive process understanding. Preuner, Conrad, and Schrefl designed a method for integration of models that capture business object life cycles [37]. Two integration types are distinguished: integration of type hierarchies and integration of behavior of object types. The integration makes use of generalization/specialization and extension/refinement operations. In [25] Mendling and Simon propose another approach for process view integration. The approach implies that one business process can have several specifications, each—an EPC model. Further, it assumes that the correspondences between elements of different models are known. Given two views of one business process and the element correspondences, the approach delivers an integrated process model. The relations between the integrated model and the initial models can be traced back to relation between m and m_a, respectively.

Use Case Name	Why	When	How	
			Model Transformation	Abstraction Algorithm
Use Case 1–4			[1,3–7,12,14,15,18,19, 25–27,32,35–37,39,44,48]	[7,22,40]
Use Case 5–8			[3,4,12,14,18,19,26,27,39, 44]	
Use Case 9	[15]	[15]	[15] [1,3–7,12,14,19,22,26,25, 27,32,35–37,39,40,44–46, 48]	[15] [7,22,40]
Use Case 10–11			[1,3,4,12,14,19,26,25,27, 37,39,44–46,48]	[7]
Use Case 12	[18,34]	[18,34]	[9] [18,34] [1,3–7,12,14,15,18,19,22, 26,25,27,32,35–37,39,40, 44–46,48]	[9] [18,34] [5–7,22,35,36,40,42]
Use Case 13,15			[1,3–7,9,12,14,15,18,19, 22,26,25,27,32,34–37,39, 40,44–46,48]	[5–7,22,40,42]
Use Case 16			[1,3–7,12,14,15,18,19,26, 25,27,32,35–37,39,44,48]	[5–7]

Table 3 Existing techniques related to identified BPMA use cases

5.3 Use Case Catalog as a Research Compass

5.3.1 Retrospective

Table 3 provides correspondences between the use cases from the catalog and the existing techniques for BPMA. Notice that the table refers to the works directly addressing BPMA along with the techniques potentially helpful in BPMA context. 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.

Table 3 allocates several papers on dedicated rows by which we emphasize the role of these works. [15,18,34] propose *comprehensive* solutions for particular BPMA use cases. By this, we mean that the papers discuss all the three aspects of BPMA. We emphasize [9], as it discusses not only a structural perspective on model transformations, but also the principles for the evaluation of non-functional properties.

5.3.2 Perspective

Another look at Table 3 reveals a disproportion in the related work: as the *how* is thoroughly investigated, the *why* and *when* are hardly touched upon. The *why* calls for research on how the user can formulate the abstraction goal and

what the frontier is of BPMA's application. The *when* question is concerned with the definition of the *sign* function, which can be non-trivial, e.g., consider Use Case 9. We conjecture that a more complete coverage of these components will simplify the uptake of the available techniques by industry.

As we argued earlier, the *how* also displays some white spots. For instance, a high user demand for Use Case 12 is a strong motivation to develop techniques that deliver aggregations of activities that belong together according to the domain semantics of the model elements. A related problem is how to label an aggregating activity as delivered by an abstraction. Finally, it seems highly interesting to determine when BPMA techniques are preferable over alternative visualization techniques and textual reports.

Furthermore, even the referenced techniques provide only partial support for some of the use cases. For instance, although [15] 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 [34] is restricted by a set of rules that enable this abstraction. Finally, in [18] 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 duly acknowledged, it is also apparent that their applicability can be enhanced.

Table 3 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 by tackling the almost unexplored *why* and *when* aspects on the one hand and by extending the range of advanced techniques addressing the *how* one the other.

6 Conclusions

This paper broadens the view on business process model abstraction based on a thorough investigation of real-world BPMA use cases. The paper's contribution is threefold. First, the paper facilitates a better understanding of what business process model abstraction entails by providing a formal framework for BPMA. Second, it elicits the current industry demand for model abstraction, as detailed in the BPMA use case catalog. The third and final contribution is an extensive survey of the state of the art in process model abstraction. For that purpose, the work that is related to the topic of BPMA was described in considerable detail using the framework introduced in this paper. We demonstrated how the use case catalog can be used to relate the identified use cases against the related work. The comparison reveals well studied areas as well as the unexplored fields and challenging opportunities for the future work in process model abstraction.

Our own interests for future work relate to the area where semantical knowledge of a particular domain can be exploited to enhance the outcomes of BPMA. Specifically, we refer here to our first steps to use knowledge external to a specific process model to determine the relatedness of the activities it contains [41]. Such an approach potentially enhances the quality of automated

aggregations considerably. Another line of work that interests us is more oriented to the development of a tool suite that provides integrated support for a whole range of process model manipulations, including BPMA [21].

To conclude, the work that is reported in this paper should be seen within the context of the process modeling discipline, where scientific advances and industrial demands traditionally go hand in hand. It is our hope that this paper will further help to streamline and unify the various academic endeavors to address the real-world needs of organizations relying on the use of business process models for a variety of purposes.

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

References

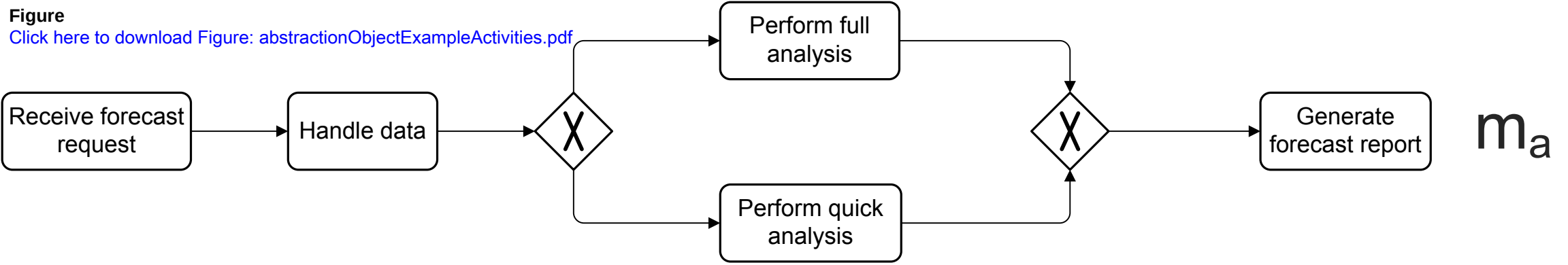
1. W. M. P. van der Aalst and T. Basten. Life-Cycle Inheritance: A Petri-Net-Based Approach. In *ICATPN 1997*, volume 1248 of *LNCS*, pages 62–81, London, UK, 1997. Springer.
2. J. Becker, M. Kugeler, and M. Rosemann. *Process Management: A Guide for the Design of Business Processes*. Springer Verlag, Berlin, Germany, 2003.
3. G. Berthelot. Checking Properties of Nets using Transformation. In *Advances in Petri Nets 1985*, pages 19–40, London, UK, 1986. Springer.
4. G. Berthelot. Transformations and Decompositions of Nets. In *Advances in Petri nets 1986*, pages 359–376, London, UK, 1987. Springer.
5. R. Bobrik, Th. Bauer, and M. Reichert. Proviado—Personalized and Configurable Visualizations of Business Processes. In *EC-Web*, pages 61–71, 2006.
6. R. Bobrik, M. Reichert, and T. Bauer. Parameterizable Views for Process Visualization. Technical Report TR-CTIT-07-37, Centre for Telematics and Information Technology, University of Twente, Enschede, April 2007.
7. R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007*, volume 4714 of *LNCS*, pages 88–95, Berlin, 2007. Springer.
8. 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.
9. 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.
10. D. K. W. Chiu, S. C. Cheung, S. Till, K. Karlapalem, Q. Li, and E. Kafeza. Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment. *Information Technology and Management*, 5(3–4):221–250, 2004.
11. D. K. W. Chiu, K. Karlapalem, Q. Li, and E. Kafeza. Workflow View Based E-Contracts in a Cross-Organizational E-Services Environment. *Distributed and Parallel Databases*, 12(2–3):193–216, 2002.
12. J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, New York, NY, USA, 1995.
13. R. M. Dijkman, D. A. C. Quartel, and M. J. van Sinderen. Consistency in Multi-Viewpoint Design of Enterprise Information Systems. *Information and Software Technology*, 50(7–8):737–752, 2008.
14. 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. *Computers in Industry*, 58(6):578–601, 2007.
15. R. Eshuis and P. Grefen. Constructing Customized Process Views. *Data and Knowledge Engineering*, 64(2):419–438, 2008.

16. A. C. W. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency Handling in Multiperspective Specifications. *IEEE Transactions on Software Engineering*, 20(8):569–578, 1994.
17. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: from Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, April 1995.
18. 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.
19. R. Johnson, D. Pearson, and K. Pingali. The Program Structure Tree: Computing Control Regions in Linear Time. In *ACM SIGPLAN PLDI 1994*, PLDI, pages 171–185. ACM Press, 1994.
20. J. Koehler, R. Hauser, J. M. Küster, K. Ryndina, J. Vanhatalo, and M. Wahler. The Role of Visual Modeling and Model Transformations in Business-driven Development. *Electronic Notes in Theoretical Computer Science*, 211:5–15, 2008.
21. M. La Rosa, H.A. Reijers, W.M.P. Aalst, R.M. Dijkman, J. Mendling, M. Dumas, and L. Garcia-Banuelos. Apromore: An Advanced Process Model Repository. *Expert Systems with Applications (accepted)*, 2011.
22. D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *Information Systems*, 28(6):505–532, 2003.
23. M. Magnani and D. Montesi. BPMN: How Much Does It Cost? An Incremental Approach. In *BPM 2007*, volume 4714 of *LNCS*, pages 80–87, Berlin, 2007. Springer.
24. 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.
25. J. Mendling and C. Simon. Business Process Design by View Integration. In *Business Process Management Workshops*, volume 4103 of *LNCS*, pages 55–64. Springer, 2006.
26. 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 and Knowledge Engineering*, 64(1):312–329, 2008.
27. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
28. 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.
29. B. Nuseibeh, J. Kramer, and A. Finkelstein. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994.
30. OMG. *Meta Object Facility (MOF) Core Specification*, 2.0 edition, January 2006.
31. OMG. *Business Process Modeling Notation*, 1.2 edition, January 2009.
32. V. Pankratius and W. Stucky. A Formal Foundation for Workflow Composition, Workflow View Definition, and Workflow Normalization based on Petri Nets. In *APCCM 2005*, volume 43 of *CRPIT*, pages 79–88, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.
33. A. Polyvyanyy, S. Smirnov, and M. Weske. Process Model Abstraction: A Slider Approach. In *EDOC 2008*, pages 325–331, 2008.
34. A. Polyvyanyy, S. Smirnov, and M. Weske. Reducing Complexity of Large EPCs. In *EPK'08 GI-Workshop*, volume 141 of *LNI*, pages 195–207, Saarbrücken, Germany, 11 2008. GI.
35. A. Polyvyanyy, S. Smirnov, and M. Weske. On Application of Structural Decomposition for Process Model Abstraction. In *BPSC 2009*, volume 147 of *LNI*, pages 110–122, Leipzig, 2009. GI.
36. A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *BPM 2009*, volume 5701 of *LNCS*, pages 229–244, Ulm, Germany, 2009. Springer.
37. G. Preuner, S. Conrad, and M. Schrefl. View Integration of Behavior in Object-Oriented Databases. *Data and Knowledge Engineering*, 36(2):153–183, 2001.
38. H. A. Reijers and J. Mendling. Modularity in Process Models: Review and Effects. In *BPM 2008*, volume 5240 of *LNCS*, pages 20–35, Milan, Italy, 2008. Springer.

39. W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.
40. M. Shen and D. Liu. Discovering Role-Relevant Process-Views for Recommending Workflow Information. In *DEXA*, pages 836–845, 2003.
41. S. Smirnov, R. Dijkman, J. Mendling, and M. Weske. Meronymy-Based Aggregation of Activities in Business Process Models. In *ER 2010*, volume 6412 of *LNCS*, pages 1–14. Springer, 2010.
42. S. Smirnov. Structural Aspects of Business Process Diagram Abstraction. In *International Workshop on BPMN*, pages 375–382, Vienna, Austria, July 2009. IEEE Computer Society.
43. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. In *BPM 2005*, volume 3649 of *LNCS*, pages 205–219. Springer, 2005.
44. R. E. Tarjan and J. Valdes. Prime Subprogram Parsing of a Program. In *POPL*, pages 95–105, New York, NY, USA, 1980. ACM.
45. J. Vanhatalo, H. Völzer, and J. Koehler. The Refined Process Structure Tree. In *BPM 2008*, volume 5240 of *LNCS*, pages 100–115, Milan, Italy, 2008. Springer.
46. 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.
47. I. Weber, J. Hoffmann, and J. Mendling. Beyond Soundness: on the Verification of Semantic Business Process Models. *Distributed and Parallel Databases*, 27:271–343, June 2010.
48. M. Weidlich, R. M. Dijkman, and J. Mendling. The ICoP Framework: Identification of Correspondences between Process Models. In *CAiSE 2010*, volume 6051 of *LNCS*, pages 483–498. Springer, 2010.
49. M. Weidlich, R. M. Dijkman, and M. Weske. Deciding Behaviour Compatibility of Complex Correspondences between Process Models. In *BPM 2010*, volume 6336 of *LNCS*, pages 78–94. Springer, 2010.

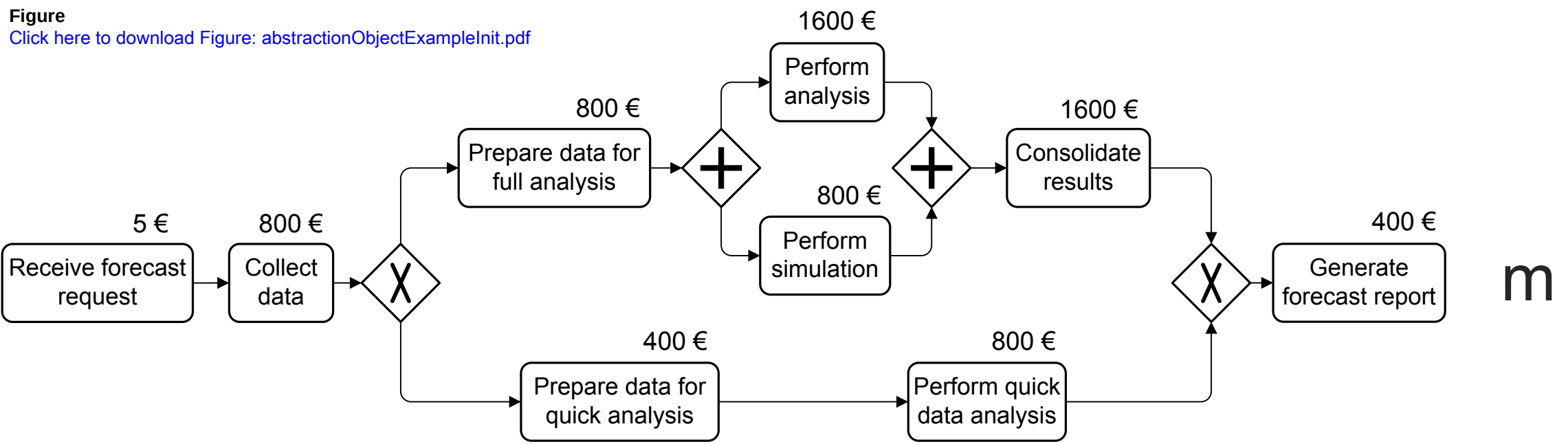
Figure

Click here to download Figure: abstractionObjectExampleActivities.pdf



Figure

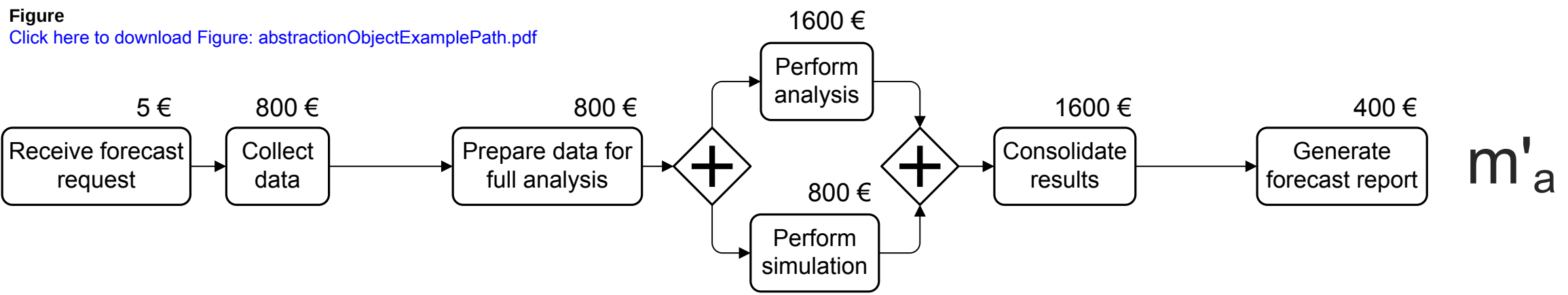
Click here to download Figure: abstractionObjectExampleInit.pdf



m

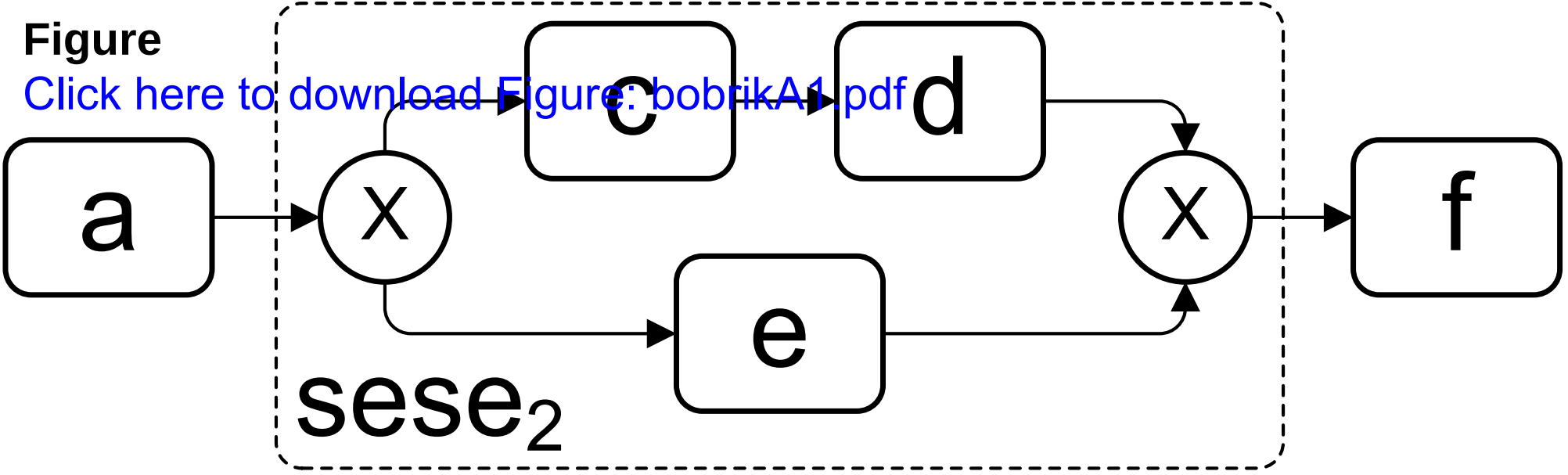
Figure

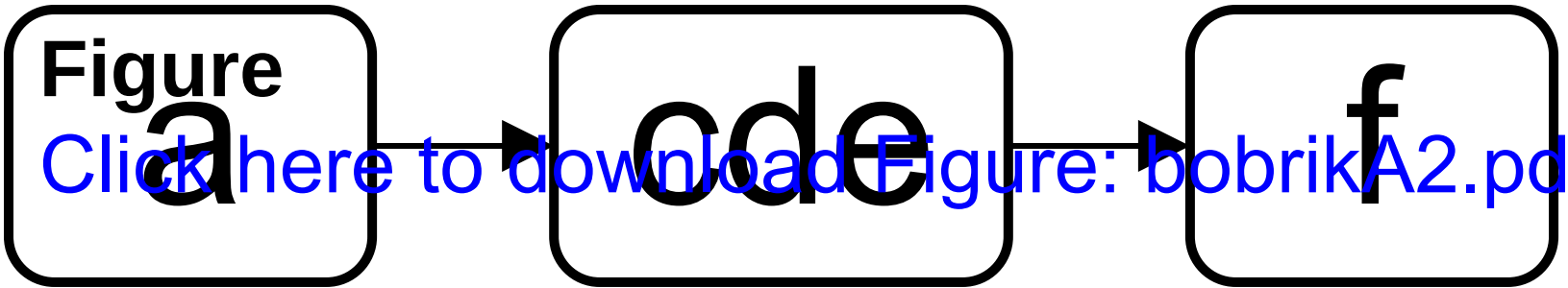
Click here to download Figure: abstractionObjectExamplePath.pdf



Figure

Click here to download Figure: bobrikA1.pdf



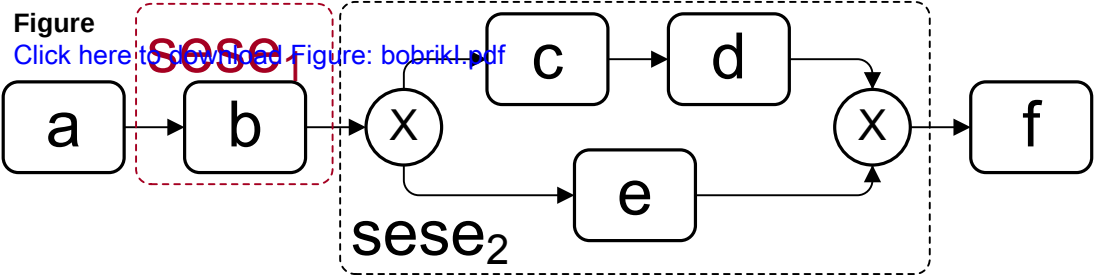


Figure

Click here to download Figure: bobrikA2.pd

Figure

Click here to download Figure: bobrikl.pdf

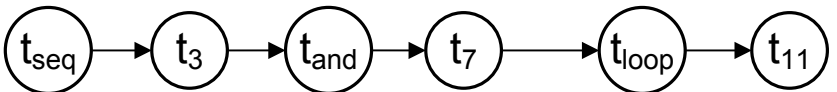


Figure

Click here to download Figure: cardosoA.pdf

$$t_{seq} = t_1 + t_2$$

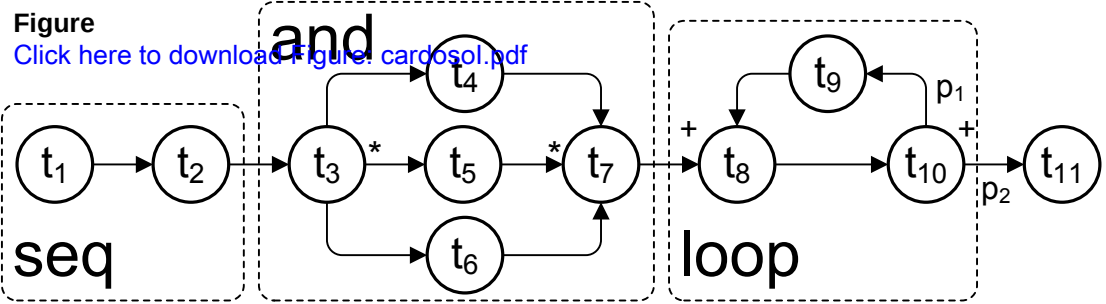
$$t_{loop} = \frac{t_8 + t_9 + t_{10} - (1-p_1)t_9}{1-p_1}$$



$$t_{and} = \max(t_4, t_5, t_6)$$

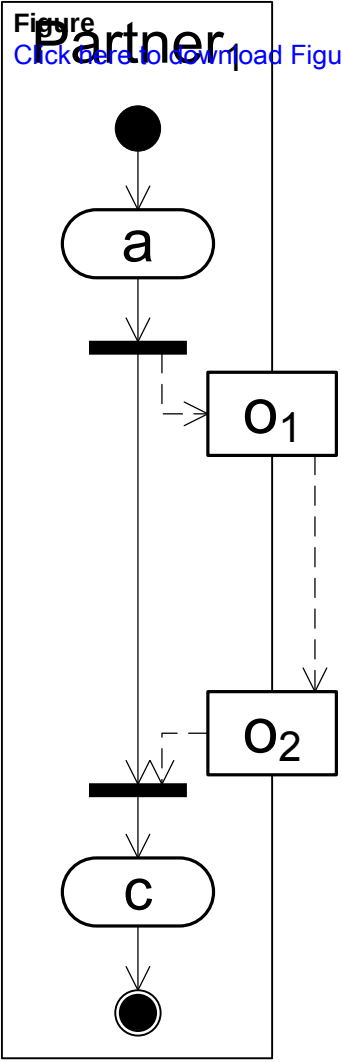
Figure

Click here to download Figure: cardosol.pdf



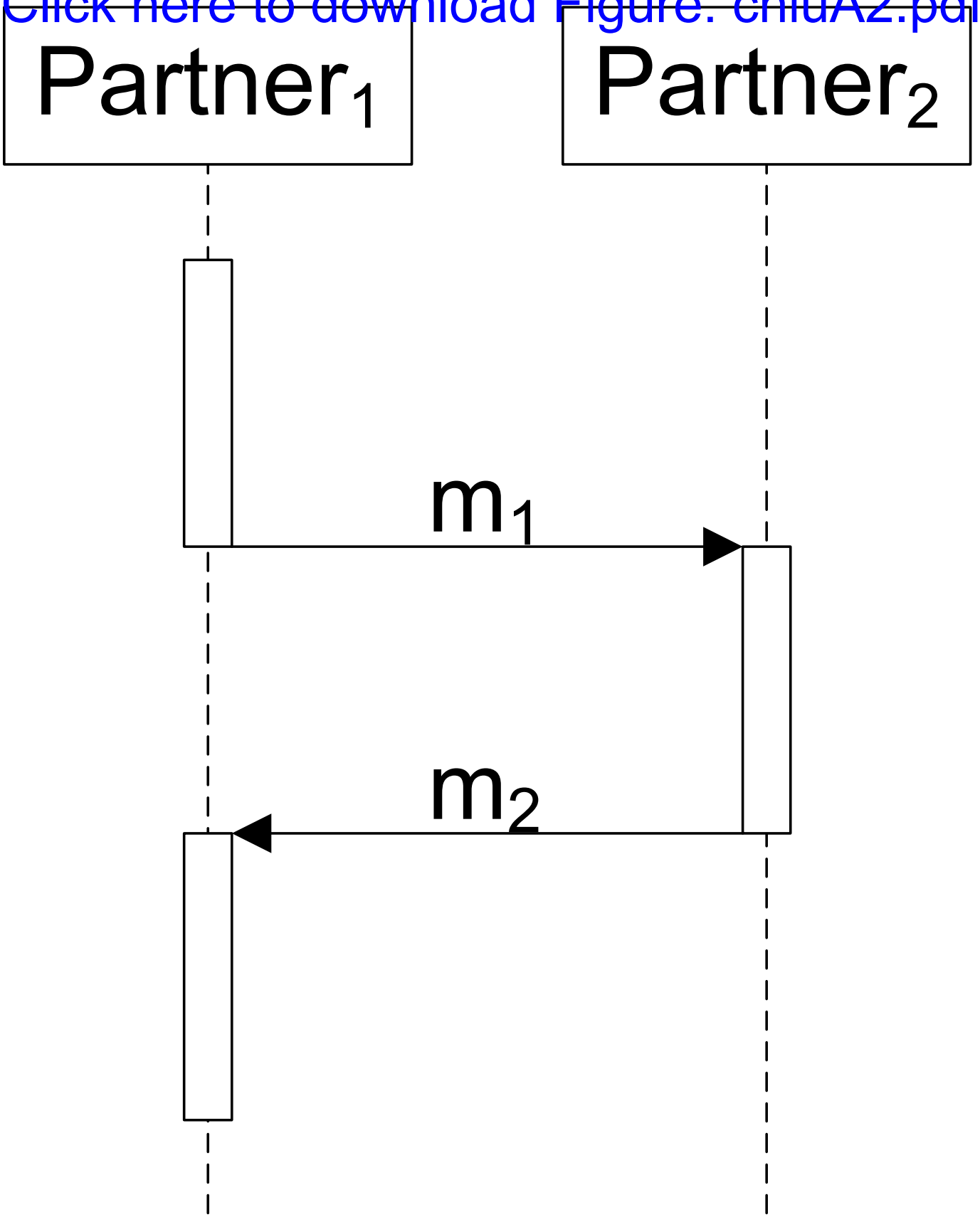
Figure

Click here to download Figure:

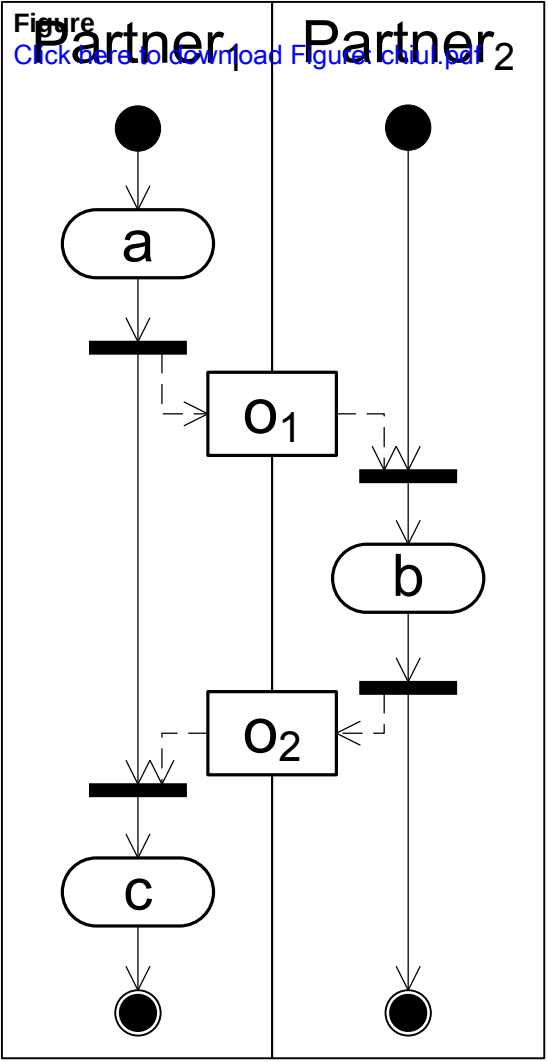


Figure

Click here to download Figure: chiuA2.pdf



Figure

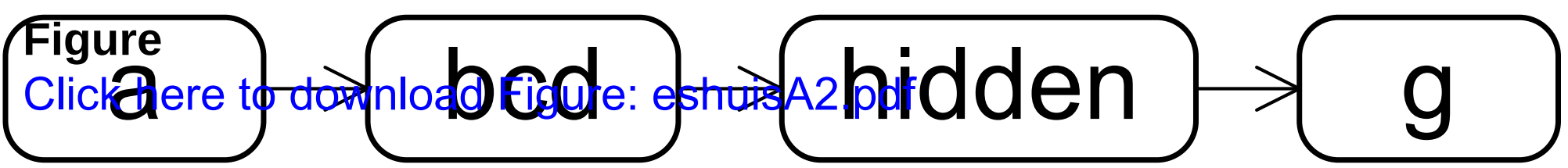


Figure

Click here to download Figure: eshuisA1.pdf

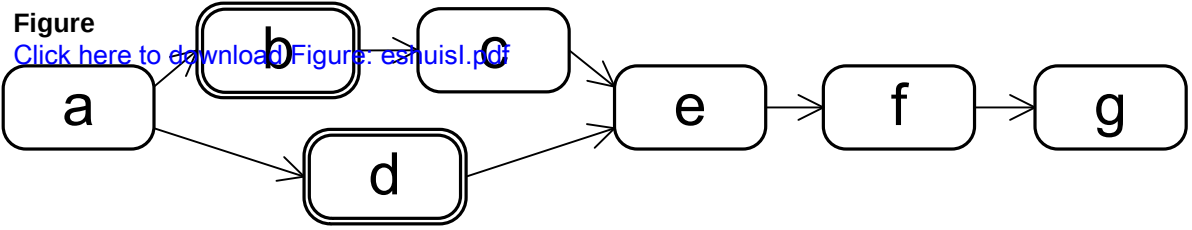
Figure

Click here to download Figure: eshuisA2.pdf



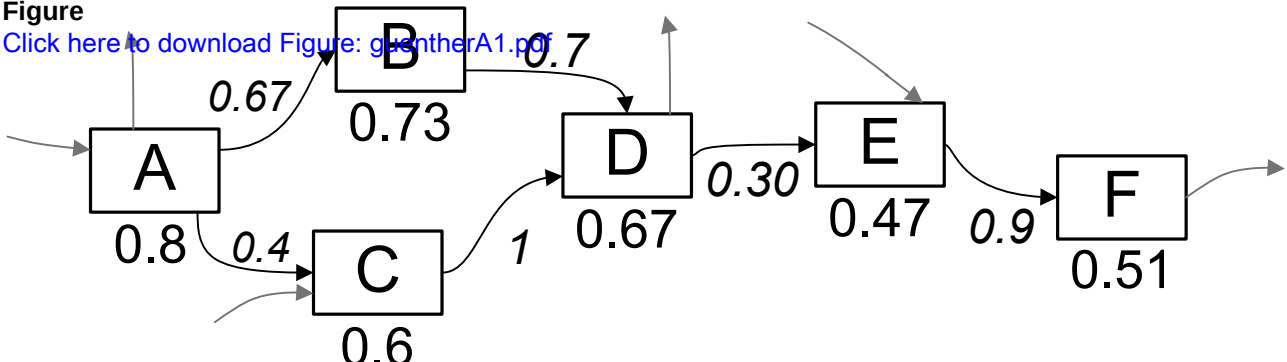
Figure

Click here to download Figure: eshuisl.pdf



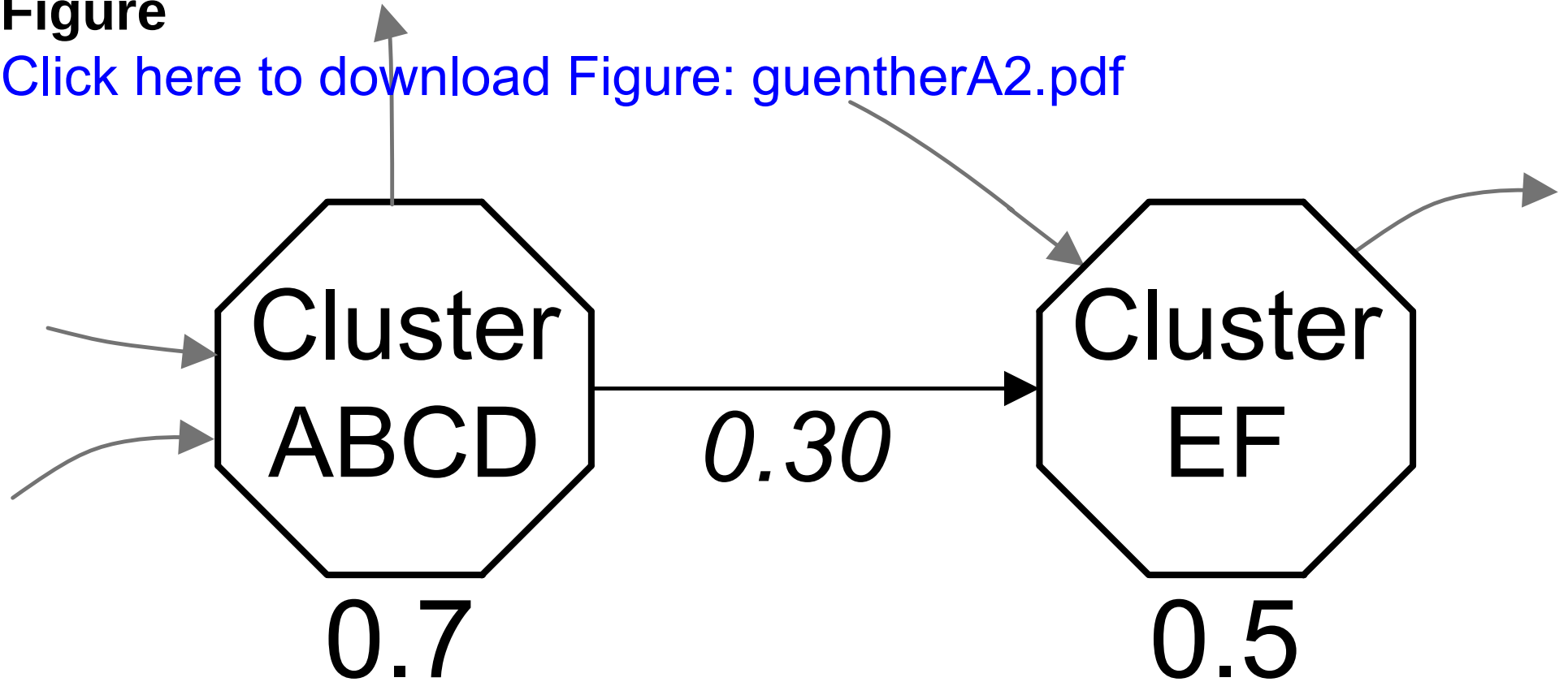
Figure

Click here to download Figure: guentherA1.pdf



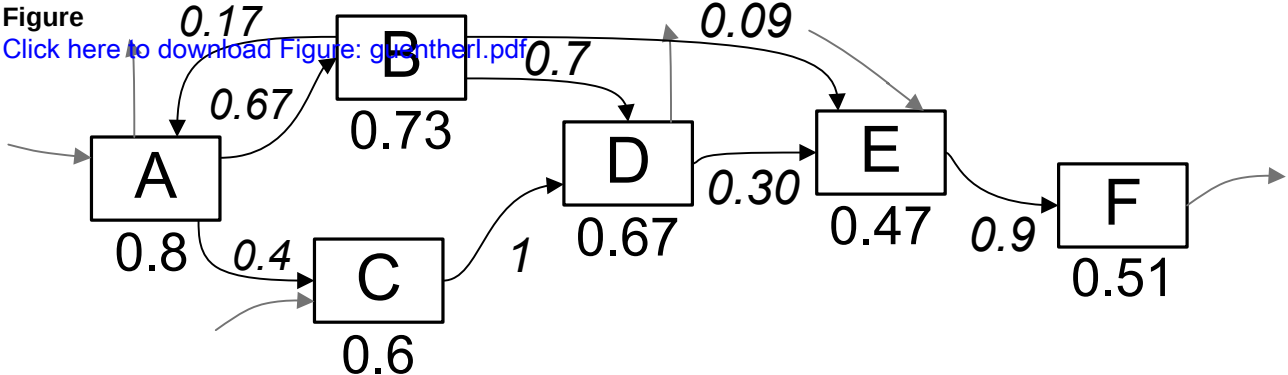
Figure

Click here to download Figure: guentherA2.pdf



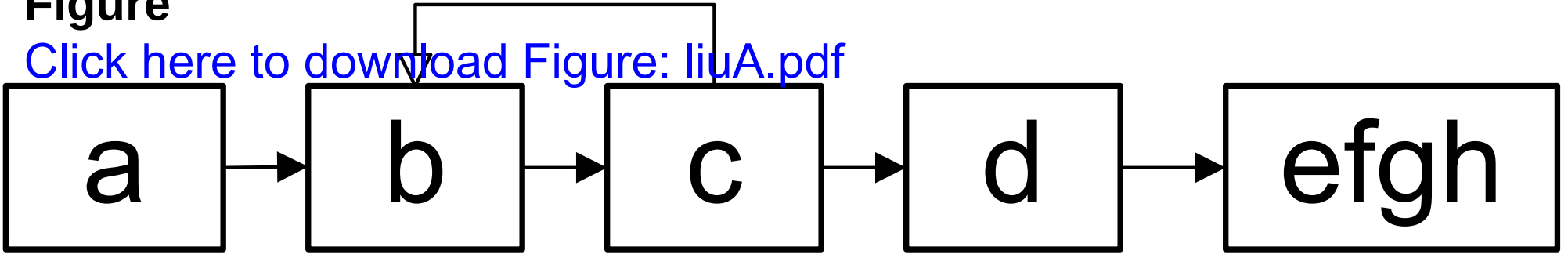
Figure

Click here to download Figure: guentherl.pdf



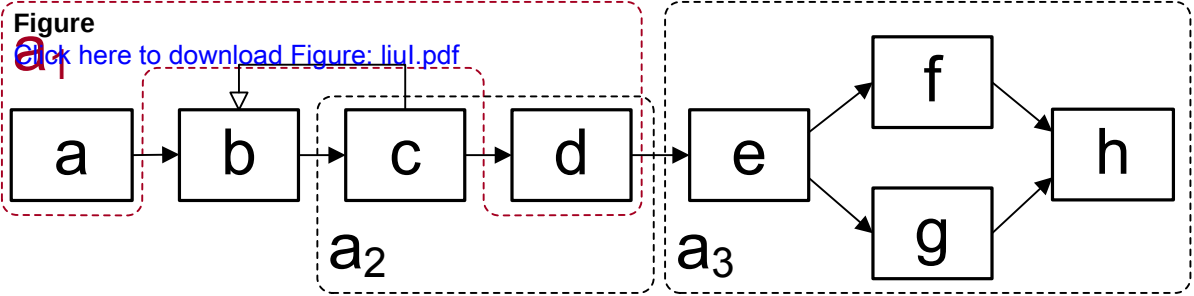
Figure

Click here to download Figure: liuA.pdf

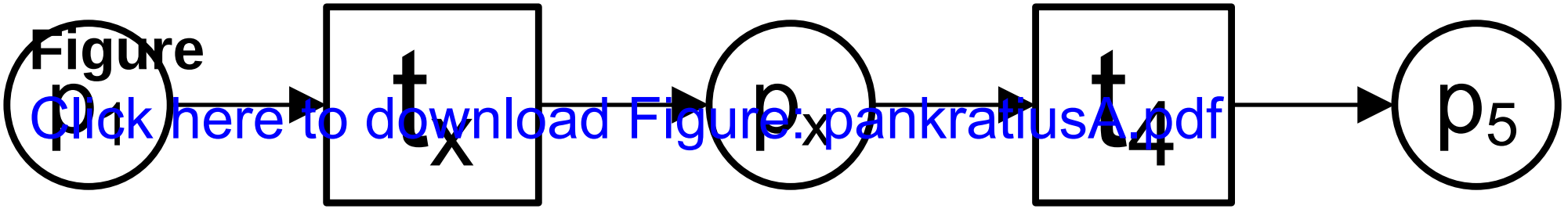


Figure

Click here to download Figure: liul.pdf



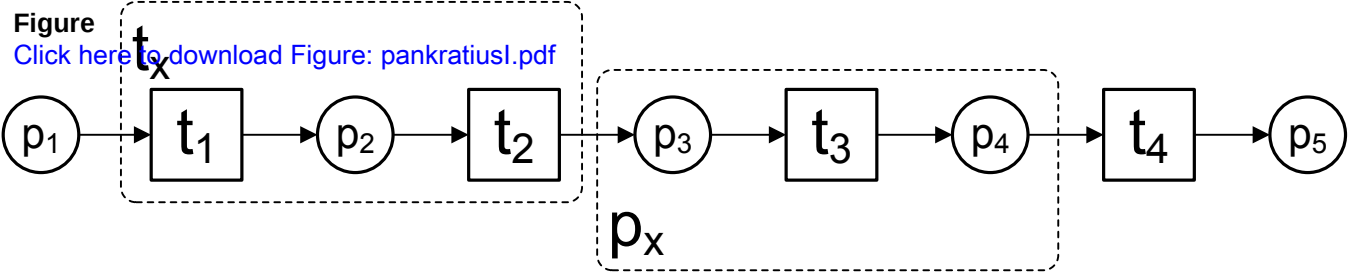
Figure



Click here to download Figure: pankratiusA.pdf

Figure

Click here to download Figure: pankratiusI.pdf

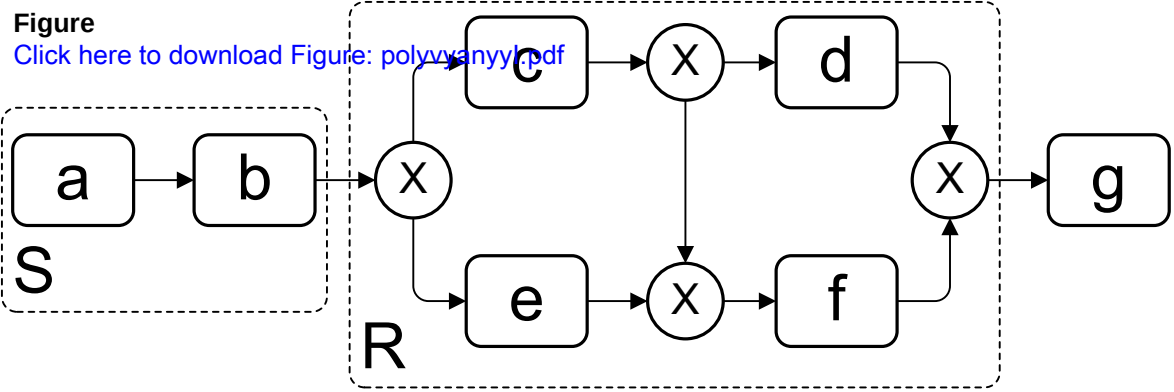


Figure

Click here to download Figure: polyvyanyyA.pdf

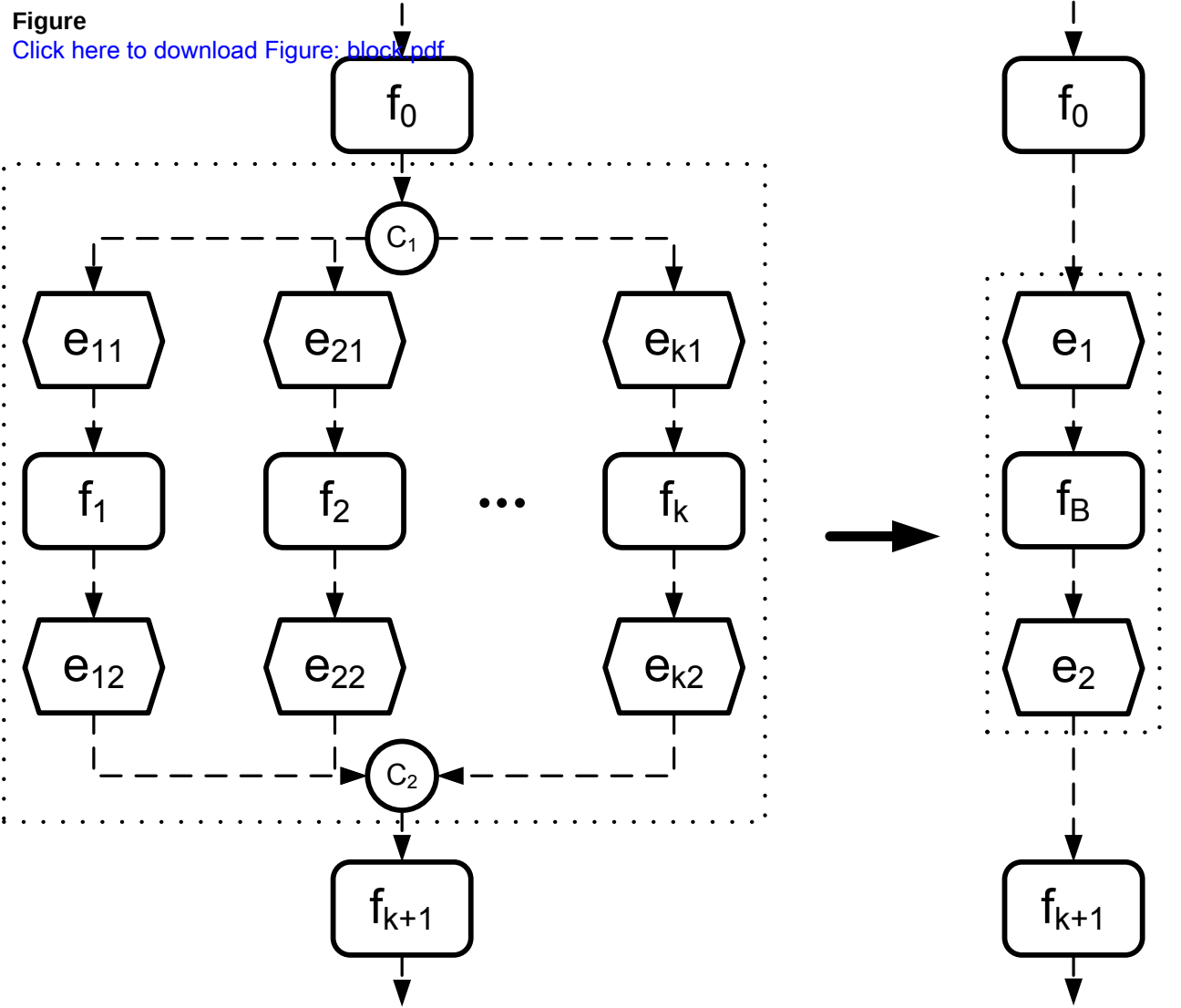
Figure

Click here to download Figure: polyvyanyyl.pdf



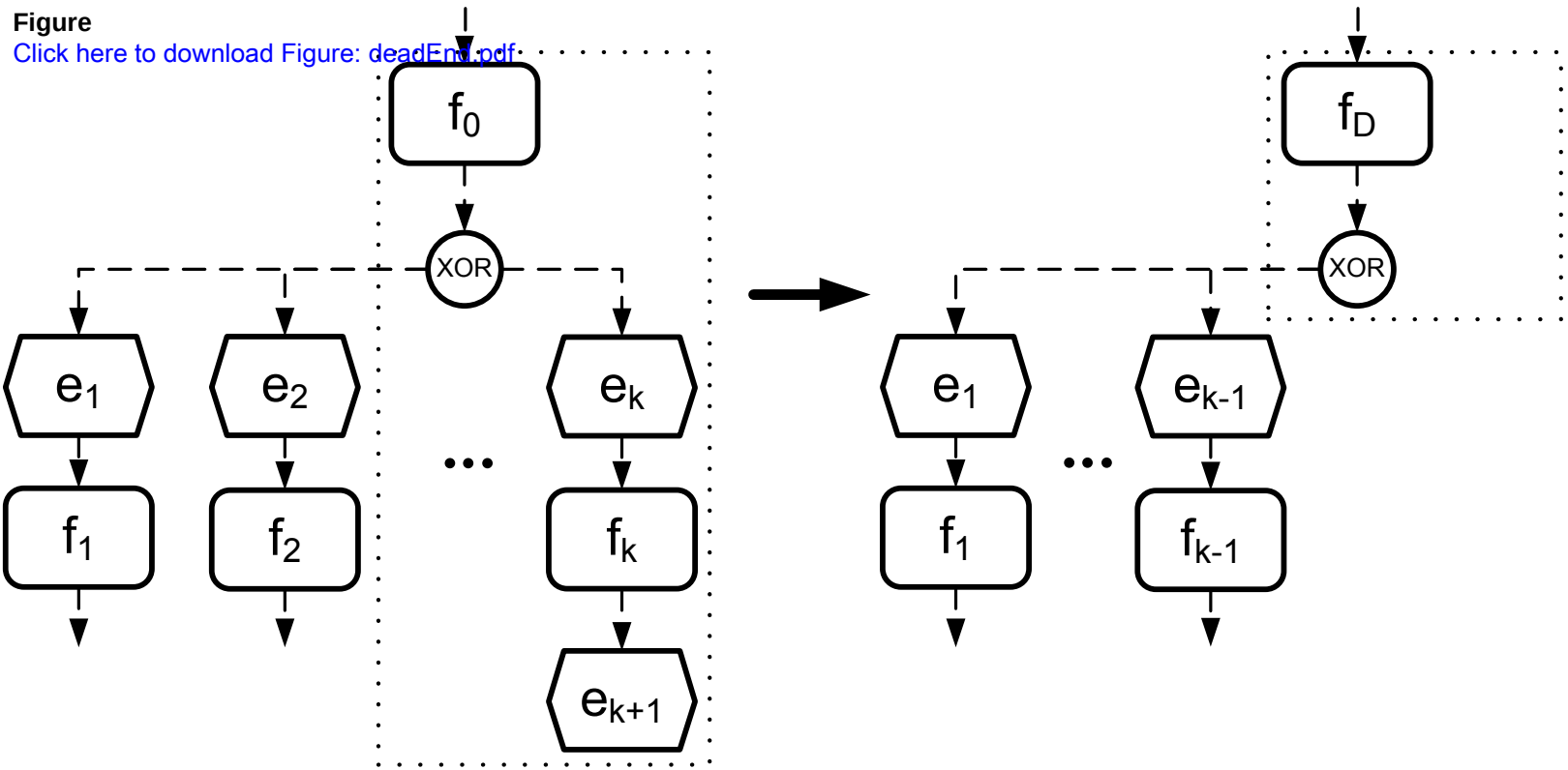
Figure

Click here to download Figure: block.pdf



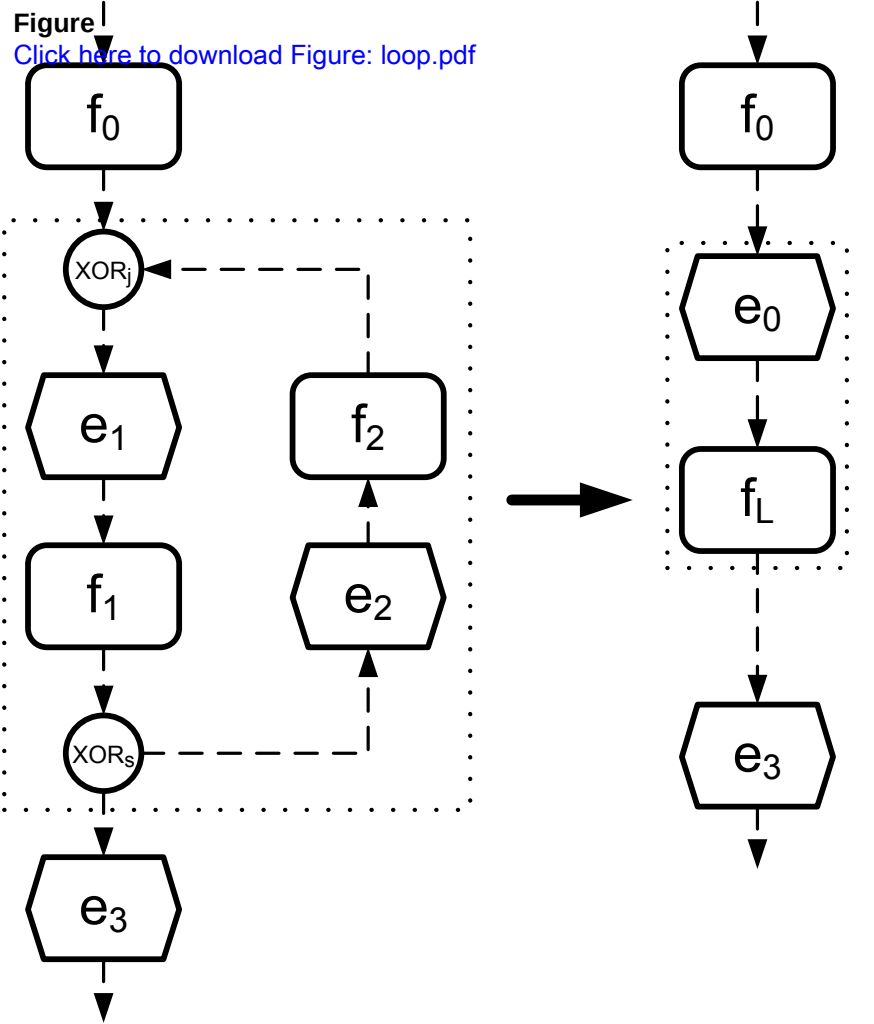
Figure

Click here to download Figure: deadEnd.pdf



Figure

Click here to download Figure: loop.pdf



Figure

Click here to download Figure: sequence.pdf