Process Model Abstraction: A Slider Approach

Artem Polyvyanyy, Sergey Smirnov, Mathias Weske
Business Process Technology Group
Hasso Plattner Institute at the University of Potsdam
D-14482 Potsdam, Germany
{Artem.Polyvyanyy, Sergey.Smirnov, Mathias.Weske}@hpi.uni-potsdam.de

Abstract

Process models provide companies an efficient means for managing their business processes. Tasks where process models are employed are different by nature and require models of various abstraction levels. However, maintaining several models for one business process requires a lot of synchronization effort and is erroneous. Business process model abstraction assumes a detailed model of a process to be available and derives coarse grained models from it. The task of abstraction is to tell significant model elements from insignificant ones and reduce the latter. In this paper we argue that process model abstraction can be driven by different abstraction criteria. Criterion choice depends on the task which abstraction facilitates. We propose an abstraction slider—an approach providing a user control over the level of model abstraction. The slider can be used with different abstraction criteria. Finally, we show an example of combining the slider with the set of process model transformation rules.

1 Introduction

Business process models are the main instrument facilitating business process management tasks in modern companies. Every model is a representation of a business process used by a certain group of stakeholders. The desired level of model granularity depends on a stakeholder and a current task. Top level management prefers coarse grained process descriptions facilitating fast and correct business decisions, while employees directly executing processes appreciate fine granular specifications of working procedures [2, 3, 5, 9, 13]. Thus, it is a common situation when a company maintains several models for one business process. To ease the maintenance modeling notations, like Business Process Modeling Notation [4] (BPMN) or Yet Another Workflow Language [1, 6] (YAWL), enable hierarchical model structuring. A model hierarchy allows to organize process details into different abstraction levels. Unfortunately, these approaches require considerable effort when a process model is changed: keeping separate models consistent as well as preserving inter subprocess dependencies is laborious. An alternative approach is to derive coarse grained process models from the existing detailed models on demand. This technique can be referenced as process model abstraction. Abstraction goal is to produce a process model containing only significant information while the excessive details are omitted.

This paper discusses an approach capable of automated business process model abstraction and focuses on the method providing a user control over the abstraction—an abstraction slider. An abstraction slider is a mechanism enabling a user to change model abstraction level continuously. The slider assumes that a user has a detailed process model and provides functionality that allows adjusting the desired level of model details to the current task.

The results presented in this work were obtained in a joint research project with the health insurance company AOK Brandenburg in Teltow, Germany. The operational processes of the company are captured in about 4 000 EPCs. Process models are enriched with information about the time required to complete each function and probabilities of process edge transitions.

The paper has the following structure. Section 2 provides definition for the key concepts and motivates process model abstraction by providing several abstraction scenarios. Section 3 introduces a slider control and explains how it is employed for process model abstraction. Examples of process model abstraction illustrate the capabilities of the abstraction slider. The work continues with section 4, presenting process model transformation rules. The purpose of the rules is to reduce insignificant model elements. The paper concludes with section 5 summarizing the contributions of the work.
2 Process Model Abstraction

In this section we define the basic terms used in the paper and motivate why process model abstraction should be flexible. After providing the basic definitions, we describe several motivating abstraction scenarios. Their purpose is to illustrate the demand for process model abstraction and to show that a business user needs different abstraction mechanisms. From the scenarios we derive several properties of process model elements helping to tell significant model fragments from insignificant.

2.1 Key Concepts

Let us start with the definition of a process model introduced in [15].

Definition 1 Let \( F \) be a set of control flow constructs. \((N, E, \text{type})\) is a process model if:

- \( N \) is a set of nodes, \( N \neq \emptyset \)
- \( E \) is a set of edges, \( E \neq \emptyset \)
- \( N = N_A \cup N_E \cup N_G \), where \( N_A \) is a set of activity models, \( N_E \) is a set of event models and \( N_G \) is a set of gateway models; the sets are mutually disjoint
- \( E \) is a set of directed edges between nodes, representing the control flow: \( E \subseteq N \times N \)
- \( (N, E) \) is a connected graph
- \( \text{type} : N_G \rightarrow F \) assigns to each gateway model a control flow construct.

Abstraction is generalization, reducing the undesired details in order to retain only information relevant for a particular task. Abstraction mechanisms are used in many domains where users suffer from information overload. One of the most well-known examples is cartography, where geographical maps visualize landscapes on different scales. While a map of the particular town provides detailed information on houses and side streets, the world map captures the shape of continents, main river contours and marks the locations of the largest cities. To stay useful to a reader, large scale geographical maps reduce the level of details, but are based on the information derived from the detailed maps. By performing a business process model abstraction we aim at reduction of the insignificant process model elements, e.g., activities, events, or gateways. Given Definition 1, business process model abstraction can be described as a function performing process model transformation.

Definition 2 A business process model abstraction is the function \( A : P \times S \rightarrow P \), such that:

- \( P \) is a set of all process models
- \( S \) is an abstraction setting, \( S \subseteq C \times \mathbb{R} \):
  - \( C \) is a finite set of abstraction criteria, \( C \subseteq T \times \{\text{asc, desc}\} \), where \( T \) is a set of criteria types
    - \( \text{asc} \) indicates that higher criterion values are of higher significance
    - \( \text{desc} \) indicates that lower criterion values are of higher significance
  - \( \mathbb{R} \) is the set of real numbers; an element of this set is the criterion value dividing significant elements from insignificant
  - if \( p' = A(p, s) \), where \( p, p' \in P, s \in S, p = (N, E, \text{type}), p' = (N', E', \text{type}') \), then \(|N'| \leq |N|\).

The parameters of the abstraction function are a process model and an abstraction setting. An abstraction setting defines a subspace of abstraction criteria values, putting restrictions on the properties of elements which should present in the abstracted process model. Only model elements conforming to the abstraction setting are left in the resulting model. An abstraction criterion is a pair, where the first element is a criterion type and the second element—a hint specifying the relation between the element property value and the element significance. Examples of criterion types are activity probabilities, activities execution cost, and activities duration. Example of an abstraction criterion is a pair \((\text{activity probability, asc})\), denoting that significance of activities depends on their probabilities and the higher the probability, the higher activity’s significance.

Conceptually, abstraction task implies answering its what and how:

- What parts of a process model are of low significance?
- How to transform a process model so that insignificant parts are removed?

The choice of the abstraction setting answers the what question. An answer to the how question depends on the transformation rules used within an abstraction. However, answers to both questions should address the current abstraction context, i.e., a business task a user solves at the moment. For instance, if a user optimizes a process, “typical” process execution scenarios might be of interest. A helpful tool in this case is an abstraction preserving model nodes with high frequencies and suppressing the nodes which occur seldom.

2.2 Abstraction Scenarios

Possessing several process models at different levels of detail a user is flexible to choose the one which is most
suitable for accomplishing a task. In this paper we aim at learning the common principles of process model abstraction. Let us introduce several process model abstraction use cases. These use cases are the starting point for analysis of abstraction problem and their goal is to facilitate understanding of the problem context.

A business process model analyst might be interested in activities which are executed most frequently in a process. Such activities are of high importance, since they noticeably influence execution time and cost of a business process. Consequently, these activities play an important role in such tasks as business process optimization and reengineering.

Alternatively, an analyst can be interested in activities that consume more time in comparison to other process activities. These activities contribute a large share to the overall process execution time and are natural candidates for being studied during process improvement task. Once such an activity is optimized, the overall process execution time might drop considerably. Besides, in some situations the execution cost is proportional to the execution time.

Activity execution cost and overall process execution cost are crucial properties of a business process. Since an activity cost has a direct influence on the process cost, identification of activities with high costs is another scenario.

Abstractions reducing insignificant process instances constitute another set of abstraction scenarios. In these scenarios properties of process instances are used as abstraction criteria. For example, one might be interested in “typical” executions of a business process model. A typical execution means that among all possible ways of a business process completion it is the one that is executed most often. Abstractions of this type result in process models describing only process instances which are often observed. Similarly, process instances with the highest duration or cost may be in the focus of process abstraction task. These abstractions result in a process model representing either most time consuming or most “expensive” process instances.

Application of different abstraction methodologies to one process model leads to different abstracted models. Figure 1 presents three models of one operational process of AOK health insurance company. The process is formalized in EPC modeling notation. The figure provides one original detailed process model and its two abstracted versions. The original model shown in Figure 1(a) contains 336 nodes: 132 functions, 137 events and 67 connectors. Figure 1(b) presents the result of a process model abstraction using activity execution time as abstraction criterion (this case corresponds to the second abstraction scenario). The abstracted EPC contains only functions taking five or more minutes to execute and consists of 159 nodes: 44 functions, 73 events
and 42 connectors. Figure 1(c) shows a model corresponding to the most typical process execution. The model is obtained using the probabilities of EPC connection transitions. The resulting model is considerably reduced and consists of 29 nodes only. Figure 1 emphasizes clear-cut distinctions between the results different abstractions produce: each abstraction reduces different sets of nodes. It illustrates how important it is for a user to have an abstraction method optimally suiting a current task.

### 2.3 Abstraction Criteria

Abstraction leads to reduction of insignificant process model details. As we have already mentioned abstraction criteria help to tell significant process model elements from insignificant. Abstraction criteria are properties of model elements or model fragments that enable elements comparison and allow identifying information relevant for the task at hand.

Analysis of the business scenarios shows that different abstraction criteria can be used for the task of business process model abstraction. A choice of an abstraction criterion or a set of criteria is problem specific. The following abstraction criteria can be derived from the aforementioned scenarios.

**Definition 3** Relative probability \( (p_r) \) of reaching a process node \( n \) from its direct predecessors \( n_p \) is the probability of an edge transition from \( n_p \) to \( n \):

\[
\langle (n_p, n) \in E \mid n_p \in N, n \in N \rangle \rightarrow [0,1].
\]

**Definition 4** Mean occurrence number of a node \( (n_i) \) is the mean number that the node \( i \) will occur in a process instance.

The mean occurrence number of a node takes into account occurrences of a node in all process instances and shows how often on average a node appears in the process.

**Definition 5** Relative effort of a process activity \( (e_r) \) is time required to execute the activity: \( e_r : N_A \rightarrow \mathbb{R}^+ \).

As it follows from the definition, relative effort of an activity is measured in time units (e.g. minutes or hours) and quantitatively coincides with the activity duration. However, semantically the effort concept is close to the concept of cost. For instance, if two activities are executed in parallel their total effort is the sum of efforts of both activities.

**Definition 6** Absolute effort of a process activity \( (e_a) \) is the mean effort contributed to the execution of the activity in a process instance: \( e_a : N_A \rightarrow \mathbb{R}^+ \). Absolute effort can be obtained as the product of relative effort and the mean occurrence number of the activity.

In addition to properties of process model activities, properties of other model elements can be used as abstraction criteria. One can use properties of process execution paths in a process model as abstraction criteria. Such a path leads from a process start event to an end event and semantically corresponds to a process instance. A model abstraction based on such a criterion identifies significant execution paths in a process model. In the following we define abstraction criteria relevant to process execution paths.

**Definition 7** Probability of a process instance \( (P_i) \) is the probability of a concrete process instance \( i \) to happen within process execution.

**Definition 8** Effort of a process instance \( (E_i) \) is the effort to be invested in execution of a concrete process instance \( i \) and can be found as the sum of efforts of all the activities executed within this instance.

The proposed list of abstraction criteria does not claim to be complete. It can be extended once there is a demand for new abstraction scenarios.

Each abstraction assumes that a process model contains information required for the abstraction process or data from which this information can be derived. For instance, an abstraction using relative probability as abstraction criterion requires a process model to possess information about edge transitions. Most process modeling notations, such as EPC [8] or BPMN, can be extended to allow enriching models with such concepts as probabilities of edge transitions, activity execution time, or activity mean occurrence number.

Information required to execute process model abstraction can be obtained in various ways. Process execution logs provide information about process execution statistics, e.g. probabilities of edge transitions or activity execution frequencies. Alternatively, property values can be estimated by domain experts or process analysts. Finally, some properties can be derived from the data already available in a business process model. For instance, the cost of process execution can be found using costs of separate activities [10].

### 3 Process Model Abstraction Slider

In the previous sections we have motivated why business process model abstraction should provide maximal flexibility to a user. The abstraction task has been decomposed into two subtasks: its what and how. In this section we focus on the what problem. We use a slider metaphor to explain an approach enabling flexible control over process model abstraction. In the beginning the slider concept is introduced. It is shown how the slider can be employed for distinguishing significant process model elements from
insignificant ones. We provide examples demonstrating the approach and illustrating that the slider functions effectively with different abstraction criteria.

3.1 Slider Metaphor

Once an abstraction criterion is selected, the required level of abstraction should be specified. As soon as the desired level of detail cannot be predicted without a priori knowledge about the abstraction context, the decision about the suitable abstraction level is postponed to the moment a concrete model is demanded. Ideally, the user should be able to change an abstraction level continuously within the whole range from an initial process model with every detail to a process model containing one activity. This activity, bounding the abstraction level above, semantically corresponds to the whole process. A model abstraction exhibiting such a behavior is controlled by an abstraction slider.

The slider concept is employed in many engineering systems, where a controlled parameter has to be changed smoothly. Numerous examples of a slider can be found in IT systems. For instance, this control is used in modern geographic information systems (GIS), where a user controls map scale by means of a zoom slider. Let us formalize the slider concept.

Definition 9 A slider is an object that can be described by:

- \([S_{\text{min}}, S_{\text{max}}]\) — a slider interval with a minimum value \(S_{\text{min}}\) and a maximum value \(S_{\text{max}}\)
- \(s \in [S_{\text{min}}, S_{\text{max}}]\) — a slider state.

How a slider can be implemented in the business process abstraction task? Every abstraction criterion discussed in this paper has a quantitative measurement. Therefore, a partial order relation holds for property values. Since properties describe objects of a process model, these objects can be ordered according to the selected property. As a result, for each abstraction criterion the set of model objects can be ordered. For instance, if activity relative effort is an abstraction criterion, an activity taking two minutes precedes an activity taking four minutes. The partial order relation enables object classification. One can choose a value splitting the set into two classes: objects which property is less than the specified value and objects which property value exceeds it. Objects of the first class are assumed to be insignificant and should be omitted in the abstracted model, while objects of the other class are significant and should be preserved. A variable according to which the objects are classified is called abstraction threshold. In the example, an abstraction threshold of three minutes results in a two minutes activity to be assumed insignificant and to be reduced, while the four minutes activity is significant and is preserved in the abstracted process model. Thus, a process model abstraction slider is a function which for a given process model fragment and a specified threshold value tells if this fragment is significant or not. According to the slider definition, an abstraction slider is a slider, such that:

- its slider interval is defined on an interval of abstraction criterion values
- its slider state is associated with the current threshold and it belongs to the slider interval.

Figure 2 illustrates application of a slider to controlling a business process model abstraction. In the example the abstraction criterion is activity absolute effort (the criterion type is activity absolute effort and it is defined on the set \(\mathbb{R}\)). Activities with higher absolute efforts are considered to be more significant, i.e. `asc`! ordering is used. The business process is captured in EPC notation and the initial process model is presented in Figure 2(a). The business process model corresponds to the case when the slider state is 0.00, i.e. no activities are reduced. If the slider state changes to 0.37, the model shown in Figure 2(b) is produced. As a result of abstraction more than 50% of the nodes are reduced. When the slider state is set to 1.00, the process model degenerates into one activity (see Figure 2(c)). Every abstracted business process model contains only objects which properties exceed the specified threshold. Therefore, elements of an abstracted model are more homogeneous in relation to a used abstraction criterion. This fact illustrates that the abstracted model contains objects that have closer properties and thus, belonging to one abstraction level.

From a user perspective a slider control regulates the amount of objects preserved in a business process model. The slider state is directly associated with the threshold value, classifying model objects into significant and insignificant. In the simplest case the user specifies an arbitrary value used as a threshold (which means that the slider interval is \([-\infty, +\infty]\)). An obvious drawback of this approach is that a user has to study a process model thoroughly in order to provide a helpful threshold value. A low threshold value makes all the objects in a process model to be treated as significant, i.e. no nodes or edges are reduced. Therefore, an abstraction does not bring added value in this case. On the other hand, a threshold which is too high may lead to reduction of the whole process model to one activity. A process model containing one activity provides such a small amount of information about a business process that the abstracted model becomes useless. To avoid confusing situations, the user should be guided by an interval in which all the values of abstraction criteria lie. However, the abstraction slider directly associated with the absolute threshold value is the easiest from the implementation point of view.
The abstraction slider can control the threshold value of an abstraction criterion indirectly. In this case, the slider specifies a share of nodes to be preserved in the model. Since abstraction mechanism possesses information about the model object properties, it is always possible to estimate the threshold value which results in the reduction of the specified share of the process model.

As we have mentioned, an abstraction slider can manage abstraction process based on various criteria. Depending on the chosen criteria and the current slider state, abstraction results in different process model. Consider the EPC fragment shown in Figure 3. It presents an exclusive choice taking place during execution of an operational process in health insurance industry. The process model is enriched with the information about the probabilities of connection transitions. Each function has two labels: function relative effort and function absolute effort (in italic). According to the process model this fragment always takes place upon process execution. Figure 4 shows what parts of this fragment are considered to be insignificant depending on the selected abstraction criterion and the slider state. Three different criteria are used: relative probability (Figure 4(a)), activity absolute effort (Figure 4(b)) and activity relative effort (Figure 4(c)). Color coding is used to show correspondence between the range of the slider state change and fragment elements which are considered to be insignificant within this range. For instance, if activity absolute effort is considered as criterion and the slider state changes in the range between 0 and 0.04 (colored with dark gray), only “Document the results” (filled with the same color) is considered insignificant. Figure 4 vividly visualizes the importance of abstraction criteria choice: the coloring of process fragments substantially differs from one case to another.

3.2 Implementation Issues

There is a number of ways to implement an abstraction slider. The most straightforward way is to compute an abstracted process model on the fly, i.e. each time the user asks for it. In this case a process model corresponding to the desired abstraction level is obtained from the initial detailed model every time. This approach is very flexible, since the user may specify an arbitrary abstraction level and receive a corresponding model.

Another approach is based on the lessons learned from GIS applications in the World Wide Web. Since these systems provide services to a huge amount of users, the computational effort of performing all the abstractions simultaneously is huge. Therefore, maps of different scale are not generated on the fly, but are precomputed. Among the precomputed maps of different scale a user chooses a map suiting the current task best. Conceptually this means that a slider interval is divided into a predefined number of subintervals, i.e. the slider is not continuous any more, but discrete. Adaptation of this approach to process model abstraction means that for each abstraction criterion a slider interval is split into subintervals and corresponding abstracted models are precomputed. The precomputed models are stored and provided to the user on demand. An obvious drawback of the approach is that only the fixed set of precomputed process models is available for the user. Furthermore, if a detailed model is changed, all the dependent process models have to be updated. On the other hand, if process model abstraction is computationally intensive an overall system performance may become more important than the flexibility issue. If a modeling environment is a multiuser system this approach may turn out to be the only possible alternative.

Between the first and the second approaches there is a space for various optimization techniques. For instance,
caching of the abstracted process models increases the system performance, while preserving all the advantages of a slider concept.

Finally, it is possible to use a process model of the current abstraction level as input for abstraction algorithm. As usual the user starts with a detailed model and comes up with its abstracted version. However, a reference to the initial model is not maintained—all the subsequent models are derived from the current process model. This approach works well when the abstraction level is increased, since the available model provides all the necessary information. As soon as abstraction leads to loss of information, a reverse operation, i.e. increasing the number of model details, is not possible. To make decreasing of abstraction level feasible, the information lost during the abstraction should be kept by the system.

4 Process Model Transformation Rules

The abstraction slider answers the what question of the abstraction task. However, it does not address the how of abstraction. In this section we would like to describe the process model transformation rules solving this problem. Graph reduction rules used for learning the properties of process models are studied in [7, 12, 14]. First, two classes of abstraction rules are introduced: elimination and aggregation. Afterwards, requirements for abstraction and their influence on the transformation rules are discussed. We argue when each of the techniques is appropriate. Finally, an example of an abstraction approach is presented.

4.1 Model Transformation Techniques

Once it is known which elements of a process model are insignificant, they have to be abstracted from. Different techniques can be used to reduce insignificant elements. We distinguish two main approaches: elimination and aggregation.

Elimination means that an insignificant process model element is omitted in the abstracted process model. As a result of elimination a model contains no information about the omitted model element. Although elimination can be seen as the simplest abstraction method, it still requires rules assuring that the process model is well-formed and preserving the ordering constraints of the initial model after an element is omitted. For instance, if an activity is removed, its incoming and outgoing edges should be removed from the process model. At the same time a new edge leading from the removed activity predecessor to its successor should be introduced in the model.

Aggregation implies that insignificant elements of a process model are aggregated with other elements. In contrast to elimination, aggregation allows preserving information about the abstracted element in the model. If two sequential activities are aggregated into one activity, the properties of the new activity comprise properties of the aggregated ac-
activities. For instance, one can define an aggregation rule so that the name of the aggregating activity is the concatenation of the aggregated activities names. The execution cost of an aggregating activity can be defined as the sum of execution costs of aggregated activities.

An abstraction approach can be based on the exclusive usage of elimination or aggregation; combination of both techniques in one approach is also possible. Elimination can be seen as the simplest technique, since it requires only the rules of correct elements omitting. However, elimination is insufficient in many cases. Aggregation requires more sophisticated specification of how the properties of the aggregated elements influence the properties of the aggregating elements. The choice of an abstraction methodology is dependent on the requirements imposed on the abstraction.

4.2 Transformation Requirements

An essential requirement for a process model abstraction is preserving the process execution logic. This requirement can be met by an abstraction preserving the ordering constraints of the initial model: neither new ordering constraints should be introduced, nor the existing ones should be changed. Process transformation rules that satisfy this requirement were discussed in [9]. Elimination as well as aggregation allows an abstracted model to preserve the ordering constraints of the detailed model.

Further, one may formulate additional requirements on abstraction rules. If a company uses process models for estimation of the workforce required to execute business processes, information about the absolute effort of process execution should be preserved in a process model. Abstractions which preserve process properties are called property preserving abstractions. In this particular case effort preserving abstraction is discussed. If an abstraction must be property preserving elimination is not sufficient: once a model element is omitted all the information about its properties is lost. Within property preserving abstraction elimination can be applied only to those elements which do not possess the preserved property. Objects enriched with the information about a property to be preserved must be aggregated.

Modeling notations possess additional requirements on process model transformation rules. Depending on a notation, several types of nodes can be distinguished in a model—each type having its own semantics. Nodes may represent activities, events or control flow structures. Consequently, nodes of different types should be treated differently by abstraction mechanisms. If a gateway is omitted within an abstraction process, the ordering constraints can be broken. Since we aim an abstraction to produce well-formed process models, features of a modeling notation should be taken into account by transformation rules. As a consequence, we can expect different rules to be used, e.g. for EPC and BPMN.

Every requirement which is imposed on an abstraction restricts the transformation rules. It could be the case that an insignificant model element cannot be reduced, because of the too restrictive set of rules. Assume an effort preserving abstraction should be performed. If there is an activity to be reduced and the abstraction does not specify a rule how to handle the given activity (so that the process absolute effort is preserved), this activity should be not reduced.
In this relation an important finding is to show which class of process models can be abstracted to one activity by a given set of rules. As we have argued not every set of rules enables this. An abstraction which is not capable of reducing a well-formed process model to one function is called best effort abstraction, since it tries to assure that a given process model is abstracted to the requested level using the given set of rules.

4.3 Transformation Rules Example

In [11] a process model abstraction approach is presented. Its cornerstone is the set of abstraction rules. We would like to use these rules as an illustration of the concepts discussed in the previous parts of this section and demonstrate how these rules can function together with the abstraction slider and activity absolute effort as abstraction criterion.

The approach presented in [11] is capable of abstracting process models captured in EPC. Two requirements are imposed on abstraction: it should preserve ordering constraints of a process model and an absolute process effort. While the former requirement is essential, the latter originates from the fact that process models have to be used in head counting task (where overall process effort is important). The approach is based on the set of transformation rules called elementary abstractions. Four types of elementary abstractions are proposed:

- sequential abstraction
- block abstraction
- dead end abstraction
- loop abstraction

Each elementary abstraction is associated with a certain type of EPC fragment and defines how this fragment is transformed. For instance, sequential abstraction describes how two sequential functions of an EPC can be aggregated into one function. The design of elementary abstractions allows preserving ordering constraints as well as process absolute effort. Elementary abstractions use both elimination and aggregation techniques. For instance, an event taking place between two sequential functions is eliminated as a result of sequential abstraction. On the other hand, abstracted functions can not be eliminated, since it leads to effort leaks. Therefore, within effort preserving approach functions are always aggregated. Single application of an elementary abstraction does not bring much added value. That is why elementary abstractions are organized into abstraction strategies. An abstraction strategy describes rules of elementary abstraction composition, e.g. their order.

As soon as each elementary abstraction handles only a specific process fragment, a class of EPCs which are reduced to one function is implicitly defined by the elementary abstractions. Once a process model contains a fragment, which can not be handled by the given elementary abstractions, abstraction can not proceed. Therefore, abstraction based on the named set of elementary abstractions is best effort.

As it has already been mentioned, elementary abstractions can be employed in the slider approach. Let us select activity absolute effort as an abstraction criterion. Once insignificant process model elements are learnt, elementary abstractions can be applied. Assume that we have used an abstraction strategy that specifies the following order of elementary abstractions: sequential, block, dead end and loop elementary abstraction. For every insignificant function to be abstracted, starting from the one with the lowest effort, abstraction algorithm tries to apply transformation rules. If one of the elementary abstraction can be employed, a function is aggregated. The aggregating function has to be tested if it is significant or not. If it turns out that the aggregating function is insignificant (its absolute effort is lower than the threshold specified by the slider) it has to be abstracted in the subsequent steps. The algorithm works till all the functions in the model are significant. It could also be the case that there are insignificant functions, but they can not be reduced, since elementary abstractions can not handle them. This is the effect of the best effort feature. Therefore, the described abstraction mechanism guarantees that it abstracts a process model at best effort, bringing the process model either to the level specified by the slider or to the state when no elementary abstraction can be applied.

The described abstraction approach is implemented in Atlas tool. Atlas supports abstraction of process models captured in EPC notation. The process examples presented in this paper were abstracted using Atlas.

5 Conclusion

Business process model abstraction is an automated way to derive high level process models from the detailed ones. This paper discusses a slider as the mean for controlling model abstraction level. We argued that the abstraction task can be decomposed into two independent subtasks: learning process model elements which are insignificant and abstracting from those elements. The abstraction problem is decoupled into two steps of answering of what and how to abstract. The work reported primarily focuses on the former problem.

Several abstraction scenarios were provided to motivate the task of business process model abstraction. These scenarios were further reused to extract model properties that can serve as abstraction criteria. It is then proposed to adopt
a slider concept in order to manage abstraction criterion interval and specify desired abstraction level. The principles of abstraction slider were explained as well as examples of its work were provided. In addition, hints on possible implementation of the abstraction slider infrastructure were discussed.

In the last section we have discussed process model transformation rules which can be employed together with the slider approach for abstraction of insignificant model elements. We have distinguished two abstraction techniques: elimination and aggregation. The properties of transformation rules were discussed. Finally, we explained how abstraction slider and the process transformation rules proposed in [11] can be used together to perform abstraction. Proposed transformation rules were implemented in the process model abstraction tool called Atlas. The results presented in this paper were obtained as the output of the Atlas work.

As the direct continuation of this work we foresee development of transformation rules which can be used together with the abstraction slider concept. As future steps we identify implementation of additional abstraction methods and generalization of already implemented ones to handle other process modeling notations, currently EPC is supported.

Acknowledgment

The authors acknowledge the support of the project partner AOK Brandenburg in Teltow, Germany, in particular Dr. Anke-Britt Möhr, Norbert Sandau and Anja Niedersält.

References