Process Model Abstraction: A Slider Approach

Artem Polyvyanyy, Sergey Smirnov, and 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 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 of one business process involves 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 to reduce the latter. In this paper we argue that process model abstraction can be driven by different abstraction criteria. Criterion choice depends on a task which abstraction facilitates. We propose an abstraction slider—a mechanism that allows user control of the model abstraction level. We discuss examples of combining the slider with different abstraction criteria and sets of process model transformation rules.

1 Introduction

Business process models are the instrument facilitating business process management task 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. 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 (BPMN) [6] or Yet Another Workflow Language (YAWL) [1, 2], allow hierarchical model structuring. A model hierarchy permits organizing process details at 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. Different approaches of presenting significant process model information to a user are discussed in [3, 4, 5, 10, 13]. An alternative approach is to derive coarse grained process models from the existing detailed models on demand. This technique can be referred to as a process model abstraction.

Abstraction is generalization that reduces 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 a particular town provides detailed information on houses and side streets, the world map captures shapes of continents, main river contours, and marks 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. Process model abstraction goal is to produce a model containing significant information based on the detailed model specification.

The ideas presented in this paper emerged from a joint research project with AOK Brandenburg—the health insurance company in Teltow, Germany. The operational processes of the company are captured in about 4 000 event-driven process chains (EPC) [9]. The goal of the project is to derive methods of automated abstractions from process model details. In [11] we have discussed the developed abstraction mechanisms. This paper focuses on the method providing a user control over the abstraction—an abstraction slider.

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 can be employed for the task of process model abstraction. The work continues with section 4, presenting ideas of combining the slider with model transformation rules. The paper concludes with a contribution summary.

2 Process Model Abstraction

In this section we provide basic definitions, describe several motivating abstraction scenarios, and derive abstraction criteria that can be employed for controlling the process of model abstraction.

2.1 Fundamentals

Let us start with the definition of a process model adopted from [14].

Definition 1 $(N, E, type)$ is a *process model* if:

- $N = N_A \cup N_E \cup N_G$ is a set of nodes where $N_A \neq \emptyset$ is a set of activities, N_E is a set of events, and N_G is a set of gateways; the sets are mutually disjoint
- $E \subseteq N \times N$ is a set of directed edges between nodes representing the control flow
- (N, E) is a connected graph
- $type : N_G \rightarrow \{and, xor, or\}$ is a function that assigns to each gateway a control flow construct.

Given Definition 1 we define a business process model abstraction as a function performing a process model transformation.

Definition 2 A *business process model abstraction* is a function $A : P \times S \rightarrow P$, such that:

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

An abstraction process transformation must not increase the number of model nodes. The parameters of an abstraction function are a process model and an abstraction setting. An abstraction setting defines a subspace of abstraction criteria values and, thus, puts restriction on the elements which should appear in the abstracted process model. Only model elements conforming to the abstraction setting should remain in the resulting model. An abstraction criterion is a pair, where the first element is a criterion type and the second element is a hint specifying the relation between the element criterion value and the element significance. An example of the abstraction criterion can be a pair (*activity execution cost*, *asc*), i.e., the higher the execution cost, the higher the activity significance in the model.

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?

Answers to both questions should address the current abstraction context, i.e., a business task a user solves at the moment. The choice of an abstraction setting answers the *what* question. A concrete abstraction function implementation answers the *how* question.

2.2 Abstraction Scenarios

In this paper we aim at learning common principles of process model abstraction. Let us introduce several process model abstraction use cases. The use cases presented are the starting point for analysis and understanding of the abstraction problem.

A business process model analyst might be interested in activities which are executed 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 the task of process improvement. 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 overall process cost, identification of activities with high costs is another scenario.

Abstractions that reduce 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.

2.3 Abstraction Criteria

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.

Relative probability (p_r) of reaching a process node n from its direct predecessor n_p is the probability of an edge transition from n_p to n, $p_r : \{(n_p, n) \in E\} \rightarrow [0, 1]$.

Mean occurrence number of a node (m_i) is the mean number that the node i occurs in a process instance.

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

The 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.

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 instances as abstraction criteria. A model abstraction based on such a criterion preserves significant process instances in a model. Following, we define abstraction criteria relevant to process instances.

Probability of a process instance (P_i) is the probability of a process instance i to happen within a process execution.

Effort of a process instance (E_i) is the effort to be invested in the execution of a 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 a complete one. 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 procedure 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. However, most process modeling notations, such as EPC or BPMN, can be extended to allow enriching models with such concepts as probability of edge transition, activity execution time, or activity mean occurrence number.

3 Abstraction Slider

In this section we focus on the *what* question of the process abstraction. We use a slider metaphor to propose an approach enabling flexible control over process model abstraction. 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 works effectively with different abstraction criteria.

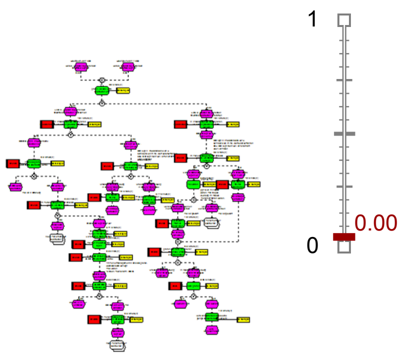
3.1 Slider Concept

Once an abstraction criterion is selected, the required level of abstraction should be specified. Since the desired level of detail cannot be predicted without a priori knowledge about the abstraction context, a decision about a suitable abstraction level is postponed to the moment when there is a demand for a concrete model. Ideally, a user should be able to change an abstraction level continuously within the whole range from an initial detailed process model to a process model containing only one activity. This activity, bounding the abstraction level above, semantically corresponds to the whole process. A model abstraction exhibiting such a behavior can be controlled by an *abstraction slider*.

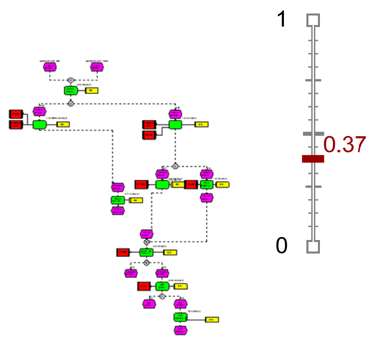
The slider concept is employed in many engineering systems, where a controlled parameter has to be changed smoothly. Numerous examples of a slider can be found in IT systems. For instance, this control is used in modern geographic information systems (GIS), where a user controls map scale by means of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

Definition 3 A *slider* is an object that can be described by:

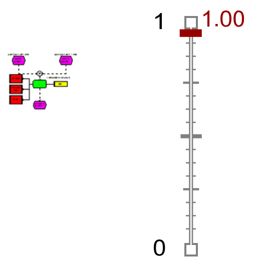
- $[S_{min}, S_{max}]$—a slider interval with a minimum value S_{min} and a maximum value S_{max}
- $s \in [S_{min}, S_{max}]$—a slider state.



(a) Initial process model



(b) Abstracted model with the slider set to 0.37



(c) Abstracted model with the slider set to 1.00

Figure 1. Process model abstraction slider (unreadability intended)

Every abstraction criterion discussed in this paper (see section 2.3) has a quantitative measurement. Therefore, a partial order relation holds for criterion values. Since criteria describe elements of a process model, these elements can be ordered according to the selected criterion. For instance, if activity relative effort is used, an activity taking two minutes precedes an activity taking four minutes. The partial order relation enables element classification. One can choose a value splitting the set into two classes: elements which criterion value is less than the specified value and elements which criterion value exceeds it. Elements of the first class are assumed to be insignificant and should be omitted in the abstracted model, while elements of the other class are significant and should be preserved. A value according to which elements are classified is called an *abstraction threshold*. In the example, an abstraction threshold of three minutes results in the 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 with the slider interval defined on an interval of abstraction criterion values and the slider state associated with the current threshold.

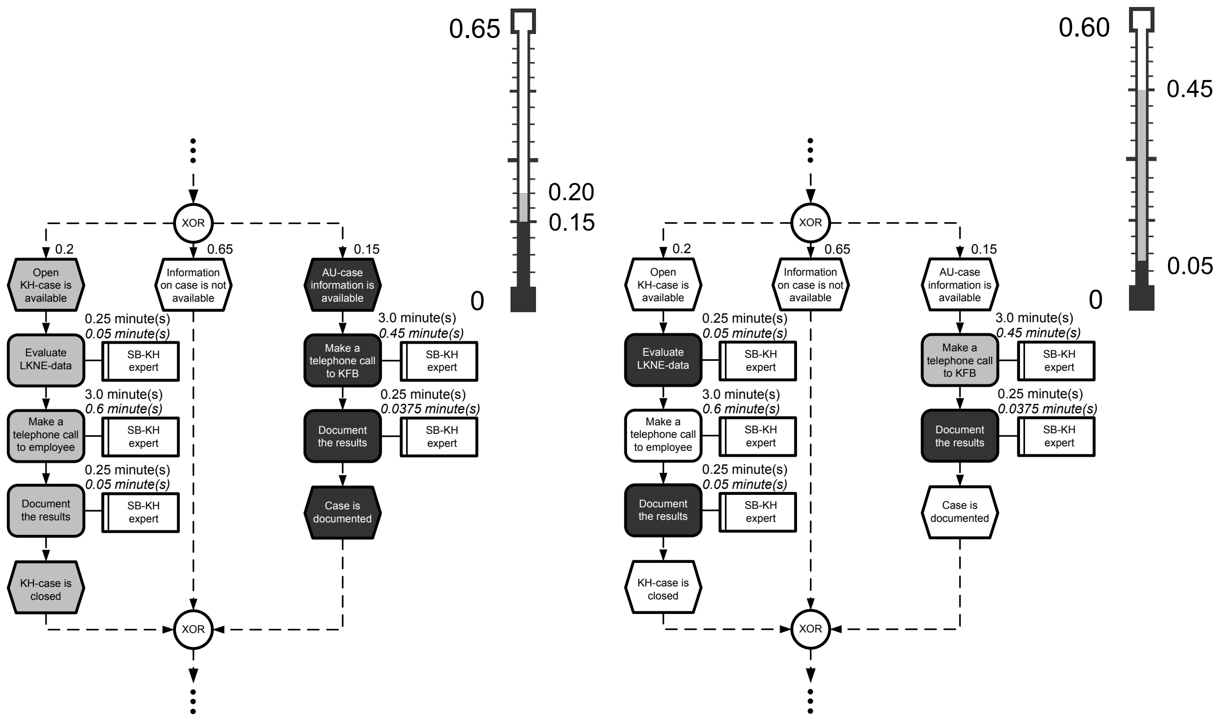
3.2 Abstraction Slider Examples

Figure 1 illustrates application of a slider to control a business process model abstraction. In the example the abstraction criterion is activity absolute effort. 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. Figure 1(a) presents the initial process model. 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 1(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 1(c)). Every abstracted business process model contains only elements which properties exceed the specified threshold. Therefore, elements of an abstracted model are more homogeneous in relation to a used abstraction criterion.

From a user perspective a slider control regulates the amount of elements preserved in a business process model. The slider state is directly associated with the threshold value, classifying model elements into significant and insignificant. In the simplest case a 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 elements in a process model to be treated as significant, i.e., no nodes or edges are reduced. 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 "useful" values of abstraction criteria lie.

Alternatively, the abstraction slider can control a share of nodes to be preserved in a model. Since abstraction mechanism possesses information about the model element 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



(a) Abstraction criterion is relative probability

(b) Abstraction criterion is absolute effort

Figure 2. Process model abstraction sliders with different abstraction criteria

results in different process models. Consider two examples from Figure 2. In both cases the same fragment of an EPC is shown. The fragment 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). Figure 2 shows what parts of this fragment are considered to be significant depending on the selected abstraction criterion and the slider state. Two different criteria are used: relative probability (Figure 2(a)) and activity absolute effort (Figure 2(b)). Color coding is used to show correspondence between the range of the slider state change and the elements which are considered to be significant within this range. Let us assume that activity absolute effort is considered as criterion and the slider state changes in the range between 0.05 and 0.45 (colored with light gray). Function “Make a telephone call to KFB” is considered significant in this range, while functions “Evaluate LKNE-data” and “Document the results” are insignificant. At the same time function “Make a telephone call to employee” is significant till slider state exceeds 0.60. Figure 2 vividly visualizes the importance of abstraction criteria choice: the coloring of process fragments substantially differs from one case to another.

4 Process Model Transformation

In this section we address the *how* question of the process model abstraction task. We base our solution on process model transformation and reduction rules. Reduction and transformation rules are widely used for analysis of process models and have been extensively studied in literature [7, 8, 12]. In this section 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 Elimination and Aggregation

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 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. Elimination can be seen as the simplest abstraction method. Although, it still requires rules assuring that the process model is well-formed and preserving the ordering constraints of the initial 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 activities. For instance, 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 is also possible. Elimination can be seen as the simplest technique, since it requires only the rules of correct elements dropping. However, elimination is insufficient in many cases. Aggregation requires more sophisticated specification of how the properties of the aggregated elements influence properties of aggregating elements. The choice of an abstraction methodology depends 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: neither new ordering constraints should be introduced, nor the existing ones should be changed. Process transformation rules that satisfy this requirement are discussed in [10].

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 a property preserving abstraction elimination can be applied only to those elements which do not influence the property being preserved.

It is an additional requirement for any abstraction to produce well-formed abstracted process models. Thus, features of modeling notations 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 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 preserved in the model. In this sense 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 allows this. An abstraction which is not capable of reducing a process model to one function is called *best effort abstraction*, since it only tries to assure that a given process model is abstracted to the requested level using the given set of rules.

4.3 Transformation Example

In [11] a process model abstraction approach is presented. Its cornerstone is a set of abstraction rules. We would like to use these rules as an illustration of the concepts discussed earlier and demonstrate how these rules can function together with the abstraction slider and activity absolute effort 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 abstracted 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, block, dead end, and 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, sequential abstraction eliminates an event between two sequential functions. On the other hand, functions are always aggregated, since their elimination leads to effort leaks. Elementary abstractions are organized into *abstraction strategies*. An abstraction strategy describes rules of elementary abstraction composition, e.g. their application order.

As soon as each elementary abstraction handles only a specific process fragment, a class of EPCs which can be 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, an abstraction based on the named set of elementary abstractions is the best effort abstraction.

The abstraction slider based on the activity absolute effort criterion can be used for identification of insignificant model elements. Afterwards, the 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 absolute effort, abstraction algorithm tries to apply transformation rules. If one of the elementary abstractions can be employed, a function is aggregated. Then, the aggregating function has to be tested if it should be abstracted. The algorithm works till all the functions in the model are significant. 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 a tool prototype. The tool supports abstraction of process models captured in EPC notation.

5 Conclusions

Business process model abstraction is a way to derive high level process models from the detailed ones. This paper proposed 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 (abstraction *what*) and abstracting from those elements (abstraction *how*). 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 abstraction criteria. We 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.

Finally, 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 the abstraction slider and the process transformation rules proposed in [11] can be used together to perform process model abstractions. However, it is also possible to employ rules discussed in [10, 12]. Transformation rules proposed in [11] are implemented in the process model abstraction tool prototype. The results presented in this paper were obtained as the output of the tool work.

As the direct continuation of this work we foresee development of transformation rules which can be used together with the abstraction slider concept. At the present moment the tool prototype supports abstraction of EPC models. As future steps we identify implementation of additional abstraction methods and generalization of already implemented ones to handle other process modeling notations.

Acknowledgments

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

References

[1] W. Aalst, L. Aldred, M. Dumas, and A. Hofstede. Design and Implementation of the YAWL System, 2004.

[2] W. Aalst and A. Hofstede. YAWL: Yet Another Workflow Language (Revised version). Technical Report FIT-TR-2003-04, Queensland University of Technology, Brisbane, 2003.

[3] R. Bobrik, T. Bauer, and M. Reichert. Proviado—Personalized and Configurable Visualizations of Business Processes. In *EC-Web*, pages 61–71, 2006.

[4] R. Bobrik, M. Reichert, and T. Bauer. Parameterizable Views for Process Visualization. Technical Report TR-CTIT-07-37, Enschede, April 2007.

[5] R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007, volume 4714 of LNCS*, pages 88–95, Berlin, 2007. Springer Verlag.

[6] BPMI.org. *Business Process Modeling Notation*, 1.0 edition, May 2004.

[7] B. Dongen, M. Jansen-Vullers, H. Verbeek, and W. Aalst. Verification of the SAP Reference Models Using EPC Reduction, State-space Analysis, and Invariants. *Comput. Ind.*, 58(6):578–601, 2007.

[8] C. Günther and W. Aalst. Fuzzy Mining–Adaptive Process Simplification Based on Multi-perspective Metrics. In *BPM 2007, volume 4714 of LNCS*, pages 328–343, Berlin, 2007. Springer Verlag.

[9] G. Keller, M. Nüttgens, and A. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Technical Report Heft 89 (in German), Veröffentlichungen des Instituts für Wirtschaftsinformatik University of Saarland, Saarbrücken, 1992.

[10] D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *Information Systems*, 28(6):505–532, 2003.

[11] A. Polyvyanyy, S. Smirnov, and M. Weske. Reducing the Complexity of Large EPCs. Technical Report 22, Hasso Plattner Institute at University of Potsdam, 2008.

[12] W. Sadiq and M. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.

[13] M. Shen and D. Liu. Discovering Role-Relevant Process-Views for Recommending Workflow Information. In *DEXA*, pages 836–845, 2003.

[14] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Verlag, 2007.