FLEXAB – Flexible Business Process Model Abstraction

Matthias Weidlich, Sergey Smirnov, Christian Wiggert, and Mathias Weske

Hasso Plattner Institute, Potsdam, Germany
{matthias.weidlich,sergey.smirnov,mathias.weske}@hpi.uni-potsdam.de,
christian.wiggert@student.hpi.uni-potsdam.de

Abstract. Process models are a widely established means to capture business processes. Large organizations maintain process model collections with hundreds of process models. Maintenance of these collections can be supported by business process model abstraction. Given a detailed model, an abstraction technique derives a coarse grained process model that preserves the essential process properties. In this paper, we introduce FLEXAB, a tool that realizes flexible process model abstraction. Arbitrary groups of activities may be selected for abstraction. FLEXAB is realized in a mashup environment, which allows for creating different abstracted versions of a process model and comparing them on a single screen.

Keywords: Process Model Abstraction, Model Synthesis.

1 Introduction

In the last decades, there has been a remarkable uptake of business process management (BPM). This trend emerged largely independent of any business domain or organizational background. Organizations that adopt BPM often manage the knowledge about their business processes by means of process models. These models define how business activities are performed in coordination to achieve a certain goal [16]. Large organizations maintain collections of hundreds of process models. The sheer number along with potential overlap of process models are challenges regarding the maintenance of such model collections.

Business process model abstraction (BPMA) emerged as a technique to support the management of large model collections. Given a very detailed model, it abstracts the process model by preserving essential process properties and leaving out insignificant details. In this way, maintenance of model collections can be centered around the most fine-grained model – more abstract models are generated by an abstraction approach.

In this paper, we present FLEXAB, a tool for flexible business process model abstraction. The tool is based on the abstraction approach introduced in [13]. In contrast to other work on process model abstraction, e.g., [3, 7, 9, 10], it does not impose structural restrictions when selecting activities that should be grouped into more coarse-grained ones. Instead, it is flexible in the sense that arbitrary

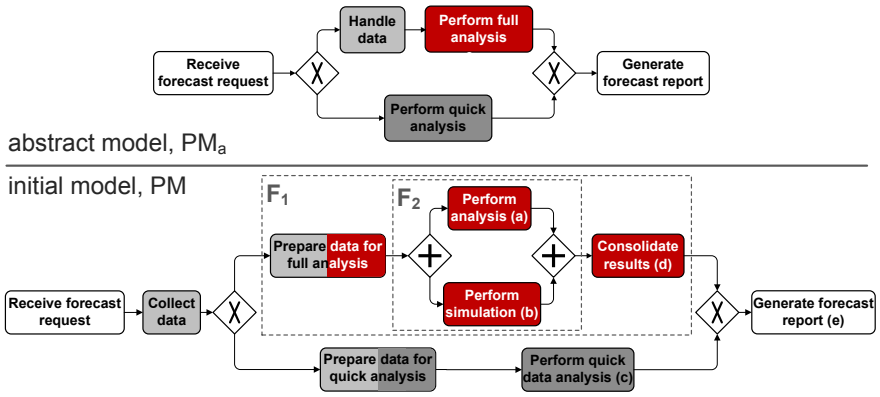

Fig. 1. Motivating example: initial model and activity grouping

groups of activities may be selected for abstraction. The question of how to define control flow dependencies for these arbitrary groupings in the abstracted model has been addressed in [13] using behavioral profiles. These profiles capture control flow relations between pairs of activities. FLEXAB implements the approach in a web-based environment. Using the Oryx framework [6], we created a mashup environment. This environment features gadgets for the visualization of process models and for providing the abstraction functionality. Using the FLEXAB gadgets, different abstracted versions of a common process model can be created and compared on a single screen. As such, our tool allows for different abstract views on a detailed process model at the same time.

The remainder of this paper is structured accordingly. The next section summarizes our approach to flexible abstraction of process models. Then, Section 3 introduces the implementation of this approach. We elaborate on the system architecture and explain the realization of all steps of the abstraction in detail. Finally, Section 4 reviews related work, before we conclude in Section 5.

2 Business Process Model Abstraction Approach

This section summarizes the approach to flexible abstraction of process models that was introduced in [13]. This approach focuses on the control flow perspective and has been defined for a generic graph model. The latter captures the commonalities of process modeling languages, i.e., a process model is a graph consisting of activities and control nodes that realize the routing behavior (aka gateways in BPMN and connectors in EPCs).

We illustrate the approach using the example depicted in Fig. 1. The lower model PM represents a detailed model of a forecasting process. This models contains several semantically related activities, indicated by the coloring in Fig. 1. These activities may be grouped to arrive at an abstract process model. Our abstraction approach allows for arbitrary grouping of activities, which may even be overlapping (indicated by a two-colored activity background in Fig. 1). This

flexibility is not offered by existing approaches, which allow to aggregate only fragments, such as the groups $F1$ or $F2$ illustrated in Fig. 1.

Our abstraction approach comprises four steps. In the remainder of this section, we explain each of these steps.

1. derive the behavioral profile BP_{PM} of the process model PM
2. construct the behavioral profile BP_{PM_a} for the abstract process model PM_a
3. **if** a well-structured model with profile BP_{PM_a} exists
4. **then** create PM_a, **else** report to user.

1. Derivation of the Behavioral Profile BP_{PM}. The approach leverages the notion of a behavioral profile. Such a profile captures behavioral characteristics of a process model by means of relations between pairs of activities. Two activities are said to be in strict order, if one occurs always before the other in every trace of the process model that contains both activities (e.g., (d) and (e) in Fig. 1). Activities that never occur together in a single trace are exclusive according to the behavioral profile (e.g., (c) and (d)). If two activities may occur in any order in a trace, then they are in interleaving order (e.g., (a) and (b)). For the class of process models considered by our approach (assuming the absence of behavioral anomalies such as deadlocks), the relations of the behavioral profile are computed in low polynomial time to the size of the model [15]. With BP_{PM}, we refer to the behavioral profile comprising the aforementioned relations for the model PM.

2. Construction of Behavioral Profile BP_{PM_a}. As the next step, we require a user to select groups of activities in the detailed process model that should be aggregated in the abstracted model. For our example in Fig. 1, a user defines several aggregations for activities, such as the aggregation of activities *Prepare data for quick analysis* and *Perform quick data analysis* that yields an activity *Perform quick analysis*. Once aggregation dependencies have been defined, we leverage the behavioral profile BP_{PM} of PM to construct a behavioral profile BP_{PM_a} for the abstract model PM_a. This works as follows. For each pair of coarse-grained activities x, y in PM_a, we study the relations of the activities in PM that are aggregated into activities x and y. As a result, we obtain a dominating behavioral relation between the activities that are aggregated. This approach has the advantage that behavioral relations between activity pairs of PM_a are discovered independently of each other. For the setting in Fig. 1, for instance, we observe that both activities *Prepare data for quick analysis* and *Perform quick data analysis* are in strict order with *Generate forecast report*. Hence, the aggregated activity *Perform quick analysis* and activity *Generate forecast report* are in strict order in the behavioral profile BP_{PM_a}.

3. Behavioral Profile Well-Structuredness Validation. The creation of the behavioral profile for the abstract model may yield an inconsistent profile. That is, we may obtain a behavioral profile for which there does not exist a process model that satisfies certain requirements, e.g., that is free of behavioral anomalies and free of duplicated activities, and shows the relations of this profile. An example for an inconsistency would be a cyclic strict order dependency between activities

(x before y before z before x). The implementation in FLEXAB deviates from the synthesis proposed in [13], which is underspecified. Within FLEXAB, we analyze the behavioral profile BP_{PM_a} following an approach proposed for different behavioral relations to restructure process models [11]. Based on the profile relations, we create a graph that represents the different behavioral dependencies between activities. Then, a modular decomposition is applied to this graph. It identifies a hierarchy of modules, groups of activities that have equal dependencies with the remaining activities. The behavioral profile is well-structured if the decomposition yields a hierarchy of modules and none of them is unstructured. If the behavioral profile is well-structured, there exists a well-structured process model that is free of behavioral anomalies and shows the respective profile.

4. Abstract Model Synthesis from BP_{PM_a}. Given a well-structured behavioral profile for the abstract model, we create the abstract process model. All modules, groups of activities that have equal relations to all remaining activities, identified in the previous step directly translate into process model fragments. For instance, a module comprising activities that are all pairwise exclusive to each other is represented by an XOR-block containing the respective activities. Since the modular decomposition yields a hierarchy of modules, we are able to stepwise synthesis the process model. For our example, Fig. 1 illustrates the abstract model PM_a derived from the initial model PM. The model PM_a, for instance, reflects the strict order relation between activities *Perform quick analysis* and *Generate forecast report* derived before.

3 Process Model Abstraction using FLEXAB

In this section, we elaborate on FLEXAB—an application enabling process model abstraction. FLEXAB extends the Oryx framework, which we introduce first. Then, we describe the FLEXAB architecture and illustrate the usage to demonstrate the capabilities of FLEXAB.

Oryx. We implemented the business process model abstraction approach described in Section 2 within the Oryx Framework. Oryx is an extensible modeling framework bringing Web 2.0 technologies to business process designers. It allows for web-based modeling following a zero-installation approach. Oryx identifies each model by a URL, so that models can be shared by passing references rather than by exchanging model documents in email attachments. The framework can be extended in various directions. New languages are added by stencil sets that define explicit model element typing, rules of the composition and connection of elements, and the visualization of elements. Further, Oryx features a plugin infrastructure to add new functionality.

Oryx is organized into client and server components. The client component, the Oryx editor, realizes the modeling functionality. The editor is a JavaScript application running in a web-browser. The server component, the Oryx backend, stores process models, stencil sets, and fulfills other tasks, e.g., user management and rendering of various model representations (SVG, PNG, or PDF). The backend is implemented in Java.

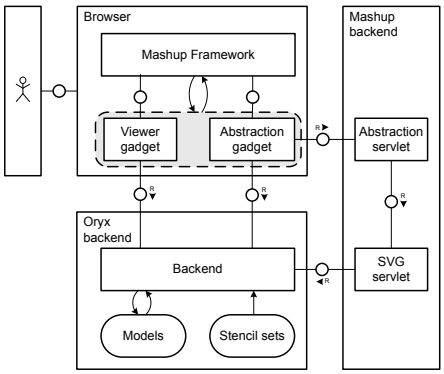

Fig. 2. FLEXAB architecture overview (FMC notation)

Oryx Mashup Framework. The Oryx editor addresses use cases that center around a single model, i.e., a designer edits one model at a time and does not need to trace dependencies with other models. However, several use cases, and process model abstraction is one of them, require the designer to observe several models simultaneously. The Oryx Mashup Framework provides an API for developing applications in which several models are manipulated on one screen. Similar to the Oryx Editor, the Mashup Framework is written in JavaScript and runs within a browser. The framework organizes functionality by *gadgets* and provides means to support communication between different gadgets. Each gadget not only accumulates business logic, but also has a UI representation. The UI components of gadgets are allocated on a dashboard. Typical gadgets provide model viewing functionality or enable selection of model elements. Hence, the Oryx Mashup Framework enables developers to create mashups for analyzing existing Oryx models and for concurrent interaction with several models.

FLEXAB. We have used the Oryx Mashup Framework as the basis for FLEXAB. Logically, the application is decomposed into the client-side and server-side components. The client-side component is built as an extension of the Oryx Mashup Framework. The server-side component is further distributed into the Oryx backend and Mashup backend, see Fig. 2. The communication between these three components is established by HTTP requests. The client-side component renders the user interface of the application. A viewer gadget presents the initial model that should be abstracted. The abstraction gadget, in turn, enables the user to define activity groups. This is supported by the viewer gadget to allow for populating groups with activities by simply selecting the activities in the viewer. Finally, another instance of a viewer gadget is used to show the abstract model.

Once the abstraction is triggered, the abstraction gadget sends the user-defined activity groups along with the initial process model to the abstraction servlet on the server side. Given this input, the abstraction servlet performs the abstraction algorithm and produces an abstract model. The abstraction servlet is supported by an SVG servlet that is responsible for the generation of a

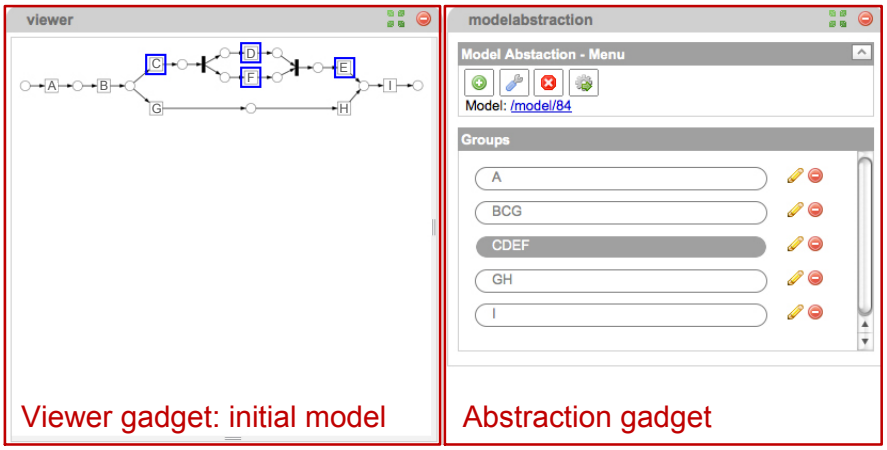

Fig. 3. Screenshot of FLEXAB at the stage of activity group creation

SVG representation of the abstract model. To this end, it needs to retrieve the respective stencil set from the Oryx backend.

From a user perspective, abstracting a process model in FLEXAB works as follows. The user starts selecting the model to be abstracted. In response, the application caters two gadgets: a viewer gadget and an abstraction gadget, see Fig. 3. The user creates named activity groups, edits, and deletes the groups using the controls of the abstraction gadget. The viewer gadget not only renders the process model and provides zoom functionality, but also supports activity group creation: the user populates groups selecting activities directly in the model. Once the groups are finalized, the user initiates model transformation clicking the abstraction button in the abstraction gadget. Then, FLEXAB abstracts the model in the background and instantiates a new viewer gadget to visualize the result of abstraction. Fig. 4 presents the UI constellation in terms of the complete Mashup dashboard once model abstraction completes.

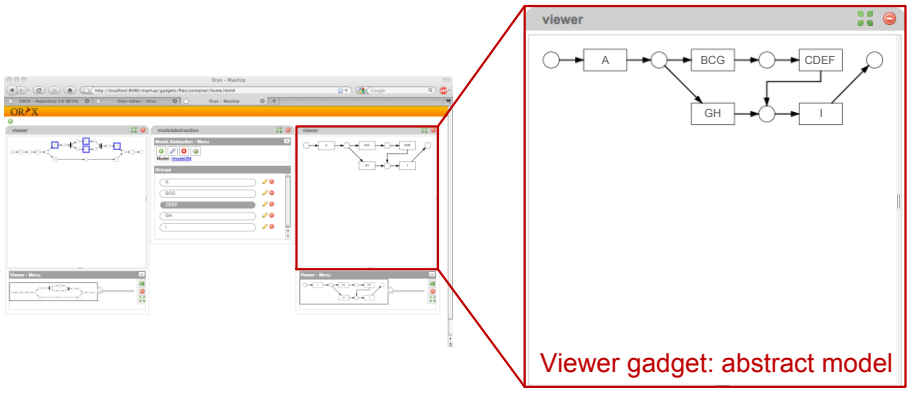

Fig. 4. FLEXAB presents the process model emerging from abstraction

4 Related Work

FLEXAB supports the user in the creation of an abstract process model given a detailed model. We identify three streams of related work, theoretical foundations of process model abstraction, applications implementing abstraction functionality, and research on process model generation.

In the recent years, a number of techniques for business process model abstraction emerged, e.g., [3, 4, 7–9, 12, 13]. All of these works investigate the theoretical principles of process model abstraction, while some address the implementation aspects as well. In [3, 4, 7–9, 13] the primary challenge of process model abstraction is addressed, i.e., structural model transformation. While the approaches of [3, 4, 9] build on an explicit definition of a fragment to aggregate, in [2, 10, 14] the fragments are discovered according to their properties. The abstraction approach implemented in this work equips the user with the most flexible way of activity aggregation. The question of how to identify model elements that are candidates for abstraction has been tackled in [7, 8, 12].

A few ideas on business process model abstraction found their way into implementations. The contribution of Bobrik, Reichert, and Bauer is realized in the Proviado system [3] and the approach presented in [9] has also been implemented in a prototype. As mentioned earlier, both approaches impose restrictions on the selection of the activities that are avoided by our approach. In the context of process mining, a mechanism for process simplification has been realized as a ProM plugin [8]. In contrast to our work, this simplification is guided by the occurrence frequency of activities in event logs. A system architecture for an application realizing model abstraction has been presented in [7].

The employed method for the synthesis of abstract process models from behavioral profiles belongs to the family of process model synthesis techniques. Most prominently, the alpha-algorithm constructs a process model given an event log [1]. The relations used in this algorithm differ to ours, since they are grounded on direct successorship of activities. A number of approaches based on Petri net formalism take the state space as an input for process model synthesis, e.g., [5].

5 Conclusion and Future Work

The theoretical aspects of business process model abstraction have been described in numerous papers. Up until now, however, very few implementations of these approaches have been presented. This paper showcases FLEXAB—an implementation of the business process model abstraction developed in [13]. FLEXAB builds on the Oryx framework. Hence, it brings together the functionality of process model abstraction and the Web 2.0 features of the Oryx framework including an extensible Mashup framework.

We have to reflect on some limitations of FLEXAB. The abstraction is currently restricted to Petri net models. Further, FLEXAB does not address the challenge of naming activities in abstract process models. In future work, we want to extend FLEXAB towards automation of process model abstraction. Since the current

version of the tool requires the modeler to group model elements manually, the natural next step is to develop functionality for the automatic discovery of activity groups in process models.

Acknowledgments

The authors acknowledge the technical support of Gero Decker and Philipp Maschke from Signavio, a BPM company based in Berlin.

References

1. W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE TKDE*, 16(9):1128–1142, 2004.
2. A. Basu and R.W. Blanning. Synthesis and Decomposition of Processes in Organizations. *ISR*, 14(4):337–355, 2003.
3. R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007*, volume 4714 of *LNCS*, pages 88–95, Berlin, 2007. Springer.
4. 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.
5. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE TC*, 47(8):859–882, August 1998.
6. G. Decker, H. Overdick, and M. Weske. Oryx - Sharing Conceptual Models on the Web. In *ER*, volume 5231 of *LNCS*, pages 536–537. Springer, 2008.
7. R. Eshuis and P. Grefen. Constructing Customized Process Views. *DKE*, 64(2):419–438, 2008.
8. 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.
9. D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *ISJ*, 28(6):505–532, 2003.
10. A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *BPM 2009*, pages 229–244, Ulm, Germany, 2009. Springer.
11. A. Polyvyanyy, L. García-Bañuelos, and M. Dumas. Structuring acyclic process models. In *BPM 2010*, volume 6336 of *LNCS*, pages 276–293. Springer, 2010.
12. S. Smirnov, H. Reijers, Th. Nugteren, and M. Weske. Business Process Model Abstraction: Theory and Practice. Technical report, Hasso Plattner Institute, 2010. `http://bpt.hpi.uni-potsdam.de/pub/Public/SergeySmirnov/abstractionUseCases.pdf`.
13. S. Smirnov, M. Weidlich, and J. Mendling. Business Process Model Abstraction Based on Behavioral Profiles. In *ICSOC 2010*, volume 6470 of *LNCS*, pages 1–16, 2010.
14. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. *LNCS*, 3649:205–219, 2005.
15. M. Weidlich, J. Mendling, and M. Weske. Efficient Consistency Measurement based on Behavioural Profiles of Process Models. *IEEE TSE*, DOI: 10.1109/TSE.2010.96, 2010. In press.
16. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.