On BPMN Process Fragment Auto-Completion

Oliver Kopp, Frank Leymann, David Schumm, and Tobias Unger

Institute of Architecture of Application Systems, University of Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

Abstract. Process fragments provide reusable granules of business processes to enable process modeling based on existing knowledge. Current verification tools cannot deal with BPMN process fragments and support complete BPMN processes only. To enable verification for BPMN process fragments, we sketch how a single BPMN fragment can be completed to a BPMN process, where additional gateways and start events are added. ¡

1 Introduction

A process fragment is intended as a reusable granule for business process design: Parts, which are reoccurring in multiple processes do not have to be modeled from scratch, but stored in a process fragment library, where they can be managed and retrieved. The main characteristics of process fragments are (i) control links with either no source or no target (called fragment entries and fragment exits or dangling edges) and (ii) place holders for variability (called regions). Regarding publicly available tool support for BPMN process verification, there is a Petri net semantics for BPMN available [5] and model checkers for Petri Nets (e.g. LoLa [18]). These tools, however, lack support for handling BPMN process fragments. To enable existing tooling to handle BPMN process fragments, there are two possibilities: (i) modify the tooling to support BPMN process fragments and (ii) to provide a method to add a minimum set of elements to a BPMN process fragment to form a complete business process without any non-standard elements. This enables any tooling to exploit the reuse concept of process fragments. This paper reasons on the second approach. The objective is to generate a BPMN process, where for each task there exists a process execution path. In other words, the resulting BPMN process has to be relaxed sound [3]. We opted for this criterion as it provides "an adequate correctness understanding" [4].

We present our concept of BPMN process fragments in Sect. 2 and discuss the auto-completion issues in Sect. 3. Section 4 presents related work and Sect. 5 concludes and discusses future work.

2 BPMN Process Fragments

In this paper, we follow the fragment definition by Schumm et al. [20]. There, a process fragment is defined as "connected graph, however with significantly relaxed completeness and consistency criteria compared to an executable process graph.

[...] A process fragment has to consist of at least one activity and there must be a way to complete it to an executable process graph". To ease understanding, we adopted the Business Process Model and Notation (BPMN) standard for the graphical representation of process fragments in [19]. Figure 1 shows basic BPMN constructs: A task represents a work item which has to be performed in a process by a human being or executed by a service or program invocation. Sequence flows are used for connecting and gateways for forking and joining the control flow. We extended the BPMN notation by a shape representing a region and by an icon representing constraints which can be annotated to process elements such as tasks, regions, and gateways. The usage of annotations is explained in [20].

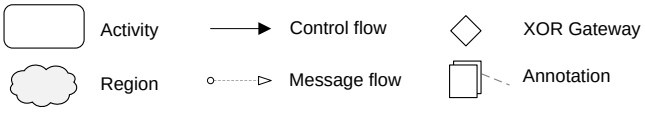

Fig. 1. Fragments in BPMN 2.0

Figure 2 presents an example BPMN fragment. The fragment models a part of a loan approval process. A form is checked for completeness. If it is not complete, the control flow has to leave the fragment. The fragment may also be started by a complete form (and thus the completeness check may be skipped). Using the complete form, the overall credit is assessed. One of the two fragment exits is taken dependent on the assessed risk.

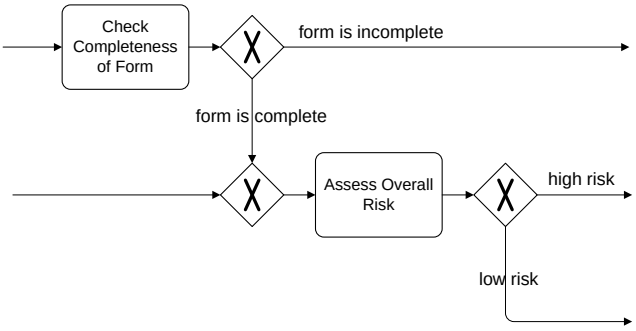

Fig. 2. Example BPMN fragment

3 Auto-completion of Process Fragments

Figure 3 presents a naive auto-completion: Each fragment entry is connected to a message start event. This approach works if mutually exclusive process entries

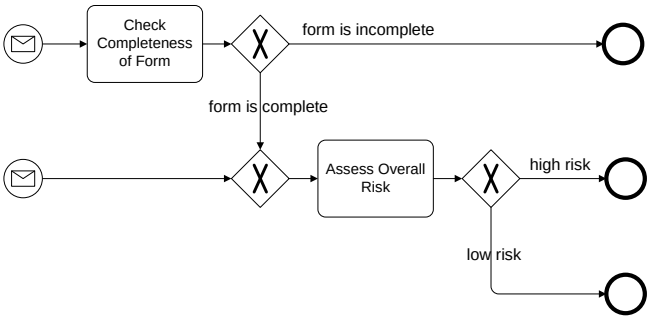

Fig. 3. Naive auto-completion

lead to a proper process execution. In case of the example, this approach is valid. In case the control flow of entries is joined via an AND join, this approach is not valid as the process execution will be stuck at the AND join as there will never be tokens on each incoming link.

Theorem 5.1 by Kiepuszewski et al. [9] shows that "every sound model with multiple end nodes can be transformed into an equivalent sound model with a unique end node" [16]. Polyvyanyy et al. [16] state that "The reverse technique can be applied to models with multiple start nodes". They, however, provide no proof for this claim. For an auto-completion for fragment entries we need such a technique. The approach by Kiepuszewski et al. requires the model to consist of AND and XOR joins only. Mendling et al. [15] showed that all OR joins in (acyclic) EPC process models can be converted to a process model containing only XOR and AND splits and joins.

We intend to apply the technique by Kiepuszewski et al. by using the following steps:

1. Merge all end nodes to a single end node.
2. Convert the BPMN model to a Petri net.
3. Reverse the edges in the process.
4. Apply the technique by Mendling et al. to eliminate all OR joins in the process model.
5. Apply the technique by Kiepuszewski et al.
6. Convert the Petri net back to a BPMN model.
7. Change the newly created end node to a message start event.
8. Reverse the edges again.
9. Undo the merging of all end nodes.

Regarding step 1, this merging requires that the end nodes are mutually exclusive. Figure 4 presents the auto-completed fragment.

It is not proven whether this approach is valid. The reversal of the edges might change properties of the Petri net [12]. For instance, the reversal of a free choice Petri net does not preserve the property of free choice.

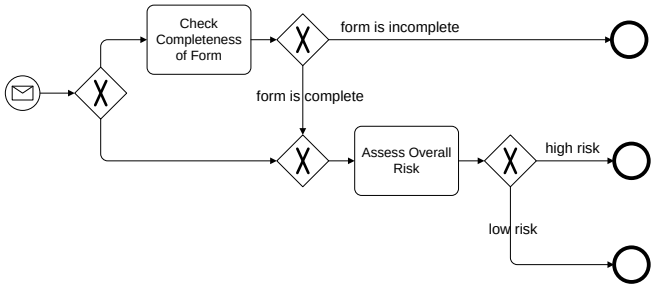

Fig. 4. Auto-completed fragment

We claim that it is *not* possible to automatically deduct concrete conditions on the sequence flows going out from the new root activity as we cannot guess the intention of the fragment designer. The fragment does not state how the fragment entries have to be reached and which conditions decide which fragment entry has to be taken. As current process verification abstracts from data, the concrete conditions are not necessary for the completion.

4 Related Work

Polyvyanyy et al. [16] showed how a subset of an unstructured BPMN models can be converted to a structured BPMN model. They require the model to contain activities, control links, and XOR and AND forks and joins only. The Refined Process Structure Tree (PST [21]) is used to (i) classify a process models which can be structured and (ii) structure the process model. A "sound and safe acyclic process model is inherently unstructured if its RPST has a rigid component for which the modular decomposition of its ordering relations contains a primitive [module]." [16]. This technique requires the process model having one entry and one exit. They rely on the technique presented by Kiepuszewski et al. [9], which we also do.

Other auto-completion approaches rely on existing business processes or process variants stored at a repository. During editing [2, 8, 11] or runtime [13], fragments matching the current process model/instance are selected. In our approach, however, we do not rely on existing process models and process variants, but do a purely syntactical extension. Syntactical autocompletion is offered by Mazanek and Minas in the case of BPMN [14]. The authors, however, rely on block-structured business process models. In this paper, we discussed arbitrary structured process models.

Generating adapters might also be a solution to auto-complete a process fragment. "An adapter is an artifact acting as mediator between services." [6]. A service here needs to be executable or at least model the behavior of the implementation. A fragment is not executable and the behavior of the fragment cannot be directly derived as the relation of the entries to each other is not directly specified by the fragment. In other words, a fragment does not state

if the entries are mutually exclusive or if all entries have to be taken for each instance of the fragment.

Gschwind et al. [7] presented a technique for process modeling by using patterns during the design of a process. One starts with a plain process and may only add activities if this inclusion conforms to a pattern. That implies that each intermediate process model is a structured process model. The fragment approach, however, enables creating and storing unstructured parts of processes.

Fragments as reusable parts have been investigated by Avrilionis et al. [1] in the case of Petri nets and by Rolland et al. [17] in the case of a process meta-model intended for artificial intelligence. Our work applies the idea of process fragments to modern business process modeling languages. Subprocesses are also a unit of reuse, but restrict the logical form to a single logical entry and a single logical exit and are not intended to be copied into a process, but used as separate process [10].

5 Conclusion and Outlook

In this paper, we reasoned on auto-completion of process fragments to either gain nearly-executable process models or nearly-complete fragments. A nearly complete fragment can be used as a part of a process model without the need of additional activities to correctly trigger the fragment entries. The concrete proposal is to apply the techniques by Kiepuszewski et al. [9] and Mendling et al. [15] to BPMN process models. This enables a proper auto-completion of fragments, where the types of the joins are AND, OR, and XOR. A proof of the presented technique is not provided in this paper and left as future work. In case arbitrary join conditions are used, the proposed approach will not work. Thus, future work is to classify arbitrary join conditions into (i) those which can be converted to XOR and AND gateways and into (ii) those which cannot.

We did not implement the proposed transformation. Thus, we did not proof whether BPMN process fragments can be really verified using a model checking tool. In case the auto-completed fragment is presented to the modeler of the fragment, the modeler might be surprised that the fragment looks differently than he has modeled it. This might lead to confusion. Therefore, we propose to show verification results on the original fragment and not on the auto-completed one.

In the discussion, we did not define what a valid fragment is and how to verify whether a fragment is valid. For instance, a fragment can be modeled, where an activity a is never reached. We would like to consider such a fragment as invalid fragment. As there is currently no criteria for valid fragments, our future work is to define such criteria.

Acknowledgments This work was supported by funds from the European Commission (contract no. 215175 for the FP7-ICT-2007-1 project COMPAS, `http://www.compas-ict.eu` and contract no. FP7-213339 for the FP7-ICT-2007-1 project ALLOW, `http://www.allow-project.eu/`).

References

1. Avrilionis, D., Cunin, P., Fernström, C.: OPSIS: a View Mechanism for Software Processes which Supports their Evolution and Reuse. In: ICSE. IEEE (1993)
2. Born, M., Brelage, C., Markovic, I., Pfeiffer, D., Weber, I.: Auto-completion for Executable Business Process Models. In: Business Process Management Workshops. LNBIP, vol. 17. Springer (2008)
3. Dehnert, J., Rittgen, P.: Relaxed Soundness of Business Processes. In: Advanced Information Systems Engineering. LNCS, vol. 2068. Springer (2001)
4. Dehnert, J., Van Der Aalst, W.M.P.: Bridging the gap between business models and workflow specifications. International Journal of Cooperative Information Systems 13(3), 289–332 (2004)
5. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information and Software Technology 50(12), 1281–1294 (2008)
6. Gierds, C.: Finding Cost-Efficient Adapters. In: AWPN. CEUR Workshop Proceedings, vol. 380. CEUR-WS.org (2008)
7. Gschwind, T., Koehler, J., Wong, J.: Applying Patterns during Business Process Modeling. In: BPM. LNCS, vol. 5240. Springer (2008)
8. Hornung, T., Koschmider, A., Oberweis, A.: Rule-based Autocompletion Of Business Process Models. In: CAiSE Forum 2007 (2007)
9. Kiepuszewski, B., ter Hofstede, A., van der Aalst, W.: Fundamentals of control flow in workflows. Acta Informatica 39(3), 143–209 (March 2003)
10. Kopp, O., Eberle, H., Leymann, F., Unger, T.: The Subprocess Spectrum. In: BPSC. LNI, vol. P-177. Gesellschaft für Informatik e. V. (2010)
11. Koschmider, A.: Ähnlichkeitsbasierte ModellierungsunterstÃijtzung fÃijr Geschäftsprozesse. Ph.D. thesis, Fakultät für Wirtschaftswissenschaften, Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB) (2007)
12. Lohmann, N.: Personal discussion at ZEUS 2011
13. Lu, R., Sadiq, S., Governatori, G.: On managing business processes variants. Data & Knowledge Engineering 68(7), 642 – 664 (2009)
14. Mazanek, S., Minas, M.: Business Process Models as a Showcase for Syntax-Based Assistance in Diagram Editors. In: Model Driven Engineering Languages and Systems. LNCS, vol. 5795. Springer (2009)
15. Mendling, J., van Dongen, B.F., van der Aalst:, W.M.P.: Getting Rid of OR-Joins and Multiple Start Events in Business Process Models. Enterprise Information Systems (EIS) 2(4), 403–419 (October 2008)
16. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring Acyclic Process Models. In: BPM. LNCS, vol. 6336. Springer Berlin / Heidelberg (2010)
17. Rolland, C., Prakash, N.: Reusable Process Chunks. In: DEXA. Springer (1993)
18. Schmidt, K.: LoLA: A Low Level Analyser. In: ICATPN. pp. 465–474 (2000)
19. Schumm, D., Karastoyanova, D., Leymann, F., Strauch, S.: Fragmento: Advanced Process Fragment Library. In: ISD. Springer (2010)
20. Schumm, D., et al.: Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls. In: MKWI (2010)
21. Vanhatalo, J., Völzer, H., Koehler, J.: The Refined Process Structure Tree. Data Knowl. Eng. 68(9), 793–818 (2009)