SMC4PEP: Stochastic Model Checking of Product Engineering Processes

Hassan Hage[1,2](✉), Emmanouil Seferis[1,3], Vahid Hashemi[2], and Frank Mantwill[1]

[1] Helmut-Schmidt-University, Holstenhofweg 85, 22043 Hamburg, Germany
hassan.hage@hsu-hh.de
[2] AUDI AG, Auto-Union-Straße 1, 85057 Ingolstadt, Germany
[3] Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany

Abstract. Product Engineering Processes (PEPs) are used for describing complex product developments in big enterprises such as automotive and avionics industries. The Business Process Model Notation (BPMN) is a widely used language to encode interactions among several participants in such PEPs. In this paper, we present SMC4PEP as a tool to convert graphical representations of a business process using the BPMN standard to an equivalent discrete-time stochastic control process called Markov Decision Process (MDP). To this aim, we first follow the approach described in an earlier investigation to generate a semantically equivalent business process which is more capable of handling the PEP complexity. In particular, the interaction between different levels of abstraction is realized by events rather than direct message flows. Afterwards, SMC4PEP converts the generated process to an MDP model described by the syntax of the probabilistic model checking tool PRISM. As such, SMC4PEP provides a framework for automatic verification and validation of business processes in particular with respect to requirements from legal standards such as Automotive SPICE. Moreover, our experimental results confirm a faster verification routine due to smaller MDP models generated from the alternative event-based BPMN models.

Keywords: Product Engineering Processes · Verification and validation · Probabilistic model checking · Markov decision processes · Probabilistic reward CTL.

1 Introduction

The ever-increasing technical challenges in products, for instance autonomous driving in automotive industries, requires *Original Equipment Manufacturers (OEMs)* to restructure their *Product Engineering Process (PEP)* from a mechanical-oriented to a system-oriented development to enable a rigorous verification and validation of its processes with respect to safety and non-safety requirements [5]. Additionally, legal authorities oblige OEMs to address consistency and traceability in their PEPs through compliance with standards such as *Automotive Software Process Improvement and Capability Determination (A-SPICE)* [21]. As the quality of a product is dependent on its processes's quality [17], consistent and qualitative processes are required for adequately addressing technical challenges, legal compliance and customer satisfaction.

E. B. Johnsen and M. Wimmer (Eds.): FASE 2022, LNCS 13241, pp. 155–162, 2022.
https://doi.org/10.1007/978-3-030-99429-7_9

A well known and most common modelling language of processes in industrial PEPs is *Business Process Model and Notation (BPMN)* [7] which we refer to as *pool-based BPMN (*pBPMN*)*. pBPMNs provide different users with their internal process workflows in a graphical notation and show the communication and dependency between different organization within the PEP. With the aim of facing the above mentioned challenges, the previous work in [8] shows the need for a revision of the BPMN language which is called *event-based BPMN (*eBPMN*)* in this paper. The processes, which are modelled according to the BPMN guidelines, are enriched with events and time symbols while message-flows of all processes are removed. On that way we ensure to capture time aspects like milestones of PEPs, to enable a communication between processes on different levels of abstraction by means of events, to determine the logical dependencies between processes and finally to remove process redundancies for ensuring consistency and traceability in PEPs. These argumentations on the process design motivated us to consider eBPMNs as a better design language in SMC4PEP. We discuss later that the eBPMN is more beneficial than its pBPMN counterpart in generating smaller MDPs and hence, enabling faster verification routine. The core part of the SMC4PEP relies on converting pBPMNs to eBPMNs while implicitly reducing the model size which is in turn done by removing redundant processes without losing information. As a bi-product, it realizes consistency in PEPs by message passing on different levels of abstraction which is not the case if pBPMN is used as a design language. Then, SMC4PEP converts the generated eBPMN to an equivalent MDP described in the syntax of the probabilistic model checking tool PRISM [15]. SMC4PEP ensures not only the consistency in PEPs but also allows for automated verification of generated MDPs against formal description of requirements from legal standards such as A-SPICE.

2 Related Tools

There exist different tools for analyzing business processes. Due to the wide industrial use of the pBPMN standard, the most common tools for analyzing business processes use this graphical representation of processes as an initial model.

The work of Ou-Yang and Lin in [19] provides an approach to translate pBPMNs to the Modified BPEL4WS representation and then to the Colored Petri-net XML (CPNXML) that can finally be verified by using CPN tools. This approach has restrictions in the support of split and merge conditions. The approach of Daclin et al. in [1] or Mendoza Morales in [18] realize a conversion of pBPMNs to a set of Timed Automata (TA) that uses Clocked Computation Tree Logic (CCTL) for the verification. In the work of Lam in [16] pBPMNs are converted to the New Symbolic Model Verifier (NuSMV) language. Then NuSMV enables an analysis of the processes using model checking techniques and verifying properties by the Computation Tree Logic (CTL). The approaches discussed in [1, 16, 18, 19] do not consider probability distributions and non-deterministic choices of processes which are required for complex processes such as PEP. Duran et al. [3] develop the approach of Rewriting Logic to enrich pBPMNs with timing and probabilistic properties. They verify stochastic properties such as synchronization time, probability distributions by means of the Parallel Statistical Model Checking And Quantitative Analysis (PVeStA) tool. However, mes-

sage passing between different processes especially on different levels of abstraction is not considered. Finally, Herbert in [14] develop an algorithm for converting pBPMNs into MDPs, where resources like timing and probabilities are considered while message passing is performed between sub-processes. Nevertheless, the size of investigated processes is small and limited and hence, message passing between large processes in particular with different levels of abstraction is not considered. Moreover, the process model is designed with less message passing and complexity to avoid the already known state-space explosion in the generated MDP model which consequently means that this approach is not applicable on complex processes like PEPs.

3 SMC4PEP **Architecture and Workflow**

As shown in Fig. 1, SMC4PEP consists of three modules, namely: (I) Differentiator, (II) Converter and (III) Generator. The Differentiator determines if the input model is a pBPMN or eBPMN. In case it is a pBPMN, the Converter converts the process model automatically to an eBPMN and moves then to the Generator. Otherwise with an eBPMN as input, SMC4PEP skips the Converter and moves automatically to the Generator. Finally, the Generator converts the eBPMN into an MDP described in the PRISM syntax which can directly be analyzed in PRISM. The process of generating the output PRISM model consists of three steps discussed as follows.

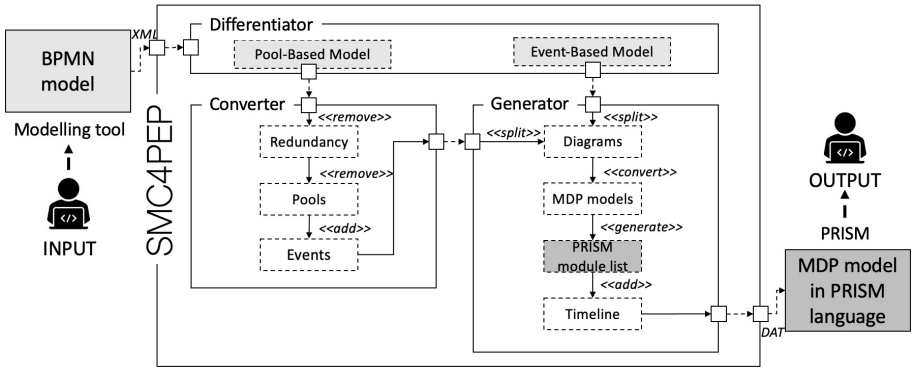

Fig. 1. Architecture of the tool SMC4PEP.

Input. SMC4PEP requires a business process model as input with no limitation of abstraction levels. Process models can be designed either according to the guidelines in [7] or [8] with different modelling tools such as Enterprise Architect [4]. Each process model needs to be exported as an XML document for the readability of SMC4PEP.

SMC4PEP**.** The Differentiator of SMC4PEP receives the input document and checks the content of the BPMN model based on the syntactic and semantic differences between eBPMN and pBPMN. According to [7] message passing between processes is

performed by message flows from tasks to task of the associated sub-processes, while each sub-process obtains its own boundary called pool. In the eBPMN approach message flows and pools are eliminated [8] and each sub-process obtains its own diagram. Then the process is enriched by events to enable message passing between each process. In case of a detected pBPMN, the Differentiator triggers the Converter, otherwise the Converter will be skipped and SMC4PEP starts automatically the Generator.

The Converter of SMC4PEP analyzes the number of identical processes within the whole process model to remove first redundant processes of pBPMN that may occur on different levels of abstraction. Redundant processes are determined when one process is equal to a second process in all elements of the model. That means in all number and content of tasks, number and content of events, number and content of gateways, role/responsible person of the process as well as number and order of sequence flows. The definition of these elements is available in [7]. When equal processes are detected, SMC4PEP eliminates all equal processes apart from one. Afterwards, all pools of the process models are removed and each sub-process obtains its own diagram. Finally, message flows are eliminated and replaced with events to ensure message passing and logical dependencies between the processes on different levels of abstraction. Note that message passing of the removed processes are also considered so that only one process enables a communication between different levels of abstraction. Finally, the pBPMN initial model is converted into an eBPMN and the Converter triggers the Generator.

The Generator requires an eBPMN which is provided either from the Differentiator or Converter. Then the process model is split into its number of diagrams. Afterwards, the Generator converts each diagram to an MDP taking into account message passing on different levels of abstraction by events, probability distributions and non-deterministic choices. Followed by the next step, the Generator of SMC4PEP generates for each MDP model a PRISM module list which are then combined to one main PRISM module list. Finally, in case of an available timeline [8] in the process model, the PRISM module list is enriched by the values of the timeline to consider time aspects and process execution costs as rewards in the MDP model described in the PRISM syntax.

Output. SMC4PEP saves the generated MDP model described in the syntax of PRISM as a DAT document which can be uploaded into the probabilistic model checker PRISM. It is worthwhile to mention that there are quite a number of tools which are able to read the PRISM modelling language. Among others, model checkers Storm [2], PARAM [10], ePMC [11] and Modest [12] can read our generated PRISM model for doing model checking various properties of interests.

4 Case Studies

For the evaluation of SMC4PEP, we converted two different use cases with SMC4PEP. Before, we developed an algorithm inspired by the work of [14] to convert a pBPMN directly into an MDP. Note that this conversion is not applicable on complex processes with different levels of abstraction. Complexity means a higher number of message passing between processes, probability distributions and non-deterministic choices. Therefore, for the evaluation we assumed that in pBPMN a communication between

different levels of abstraction is possible by merging all diagrams to one main diagram, although in real processes it is not the case. This assumption is met to obtain the MDP sizes of the pBPMN. On that way MDP sizes generated through a pBPMN and eBPMN model can be compared and the effectiveness of the eBPMN can be approved. The first use case describes the process of testing an autonomous park pilot with three levels of abstraction and includes five roles where each role performs its associated task of the process. The second use case handles a more complex process of an urgent request for a change of the vehicle construction during the PEP. In total this use case extends over four levels of abstraction and includes eleven roles. Both use cases are provided by an automotive OEM. We run all experiments on an Core i7 laptop running Windows 10.

Table 1 provides promising results generated based on SMC4PEP. The generated MDP model of the first use case with two levels of abstraction is for the eBPMN 33.8% in states and 40.7% in transitions less than for the pBPMN. Moreover, the generated MDP model in the third level of abstraction is in the eBPMN 67.78% in states and 73.11% in transitions less than in the pBPMN. The build time of the MDP model for the eBPMN with three levels of abstraction is higher compared to the pBPMN. Note that the MDP model is built only once which has no impact on the run-time of model checking MDPs. This is indeed the case for generating a formalism like MDP from giant BPMN models and use it several times for model checking various properties. The generated MDP models of use case two with four levels of abstraction are large compared to the first use case due to the high number of activities, probability distributions and non-deterministic choices of the processes. Nevertheless, the effectiveness of the eBPMN for complex processes is strongly confirmed by the generated MDP size of the second use case on four levels of abstraction which is far less than the MDP size of pBPMN. Finally, our generated MDP models from eBPMN have much smaller sizes compared to the approach discussed in [14]. In particular, for the second use case we got several order of magnitudes reduction in model size which is significant for an efficient model checking routine. However, similar to [14] we also realize the state space explosion problem which can be alleviated using bisimulation minimization techniques [6,9,13].

Table 1. Results of the analyzed processes.

BPMN model	Use case	Abstraction level	MDP model States	MDP model Transitions	Built time (s)
pBPMN	1	2	423	1143	0.071
eBPMN	1	2	280	685	0.037
pBPMN	1	3	5276	21503	0.170
eBPMN	1	3	1700	5782	0.551
pBPMN	2	4	$93x10^{16}$	$14x10^{16}$	4.263
eBPMN	2	4	$17x10^{10}$	$19x10^{11}$	0.871

At the end, we take the PRISM tool for model checking some properties of interest described in the *Probabilistic Reward Computation Tree Logic (PRCTL)* [20]. It is worthwhile to note that for SMC4PEP we provide the first use case as an eBPMN to

capture time and cost aspects of the PEP by a timeline while the second use case is described first in pBPMN and then converted to eBPMN. Firstly, we verify some properties based on the A-SPICE guidelines [21] by φ_1, φ_2 and φ_3. The properties are taken from the *Generic Practice (GP)* of A-SPICE Level 2 [21] where each level of A-SPICE determines the quality of the processes. The property GP 2.1.7 of A-SPICE denoted as φ_1 which requires ensuring no deadlocks in the processes and reaching the final state of the process with the probability of 100%. Additionally by φ_2 we denote the property GP 2.1.2 which ensures the ability of performing the process to fulfil the identified objectives similar to φ_1. Moreover, the GP 2.1.3 is denoted by φ_3 through which we ensure that our process does not deviate from its original setting according to A-SPICE. Finally for use case one, the non-functional properties are denoted by φ_4 which delivers the minimum days (d) for performing the whole process, and by φ_5 which enables the expected cost estimation of the process obtained in accumulated working days (wd). We have to note here that φ_4 is obtained by the GUI simulator of PRISM. The results of the property verification obtained from PRISM are depicted in Table 2.

Table 2. Model checking of eBPMN processes.

Abstraction level	Use Case	MDP model States	MDP model Transitions	Properties φ_1	φ_2	φ_3	φ_4(d)	φ_5(wd)
2	1	280	685	✓	✓	✓	78	267.9
3	1	1700	5782	✓	✓	✓	110	346.5
4	2	$17x10^{10}$	$19x10^{11}$	✓	✓	✓	-	-

5 Conclusion

In this paper we presented the new tool SMC4PEP to enable in the first phase an automated conversion of complex process models such as PEPs that are modelled according to the BPMN standard [7] into revised process models based on the modelling approach of [8]. This conversion paves the way for consistency and traceability of complex PEPs by removing redundant processes and enabling an exchange between different levels of process abstraction. In the second phase, SMC4PEP converts the new process model into an MDP to capture stochastic properties of a PEP and to enable an automated verification of the MDP using PRISM against formal descriptions of requirements. In case of designing a new PEP based on [8], SMC4PEP considers also the timeline of processes to capture time and cost aspects of a PEP that are essential for developing a new product in particular in automotive and avionics industries. Finally, we approved the effectiveness of our tool in an automotive case study where we compared pBPMNs with eBPMNs and verified some properties of interest such as legal regulations from A-SPICE.

Acknowledgments. This work is supported by the Helmut-Schmidt-University in Hamburg and by the AVAI project at AUDI AG in Ingolstadt.

References

1. Daclin, N., Vallespir, B., Vincent, C.: Enabling model checking for collaborative process analysis: from bpmn to 'network of timed automata'. In: Enterprise Inforation Systems. vol. 9, pp. 279–299. Taylor and Francis (2015)
2. Dehnert, C., Junges, S., Katoen, J., Volk, M.: The probabilistic model checker storm (extended abstract). CoRR **abs/1610.08713** (2016), http://arxiv.org/abs/1610.08713
3. Duran, F., Rocha, C., Salaün, G.: Stochastic analysis of bpmn with time in rewriting logic. In: Science of Computer Programming. pp. 168, pp. 1–17. Elsevier (2018)
4. Europe, S.S.C.: Enterprise Architect 15.2 [Software] (2021), https://www.sparxsystems.de
5. Gausemeier, J., Dumitrescu, R., Steffen, D., Czaja, A., Wiederkehr, O., Tschirner, C.: Systems engineering in der industriellen praxis. Heinz Nixdorf Institut, Frauenhofer Institut, UNITY AG (2013)
6. Gebler, D., Hashemi, V., Turrini, A.: Computing behavioral relations for probabilistic concurrent systems. In: ROCKS 2012. pp. 117–155. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
7. Group, O.O.M.: Business process model and notation (bpmn). Website (2014), https://www.omg.org/spec/BPMN
8. Hage, H., Hashemi, V., Mantwill, F.: Towards a systems engineering based automotive product engineering process. In: Software Architecture - 14th European Conference. Communications in Computer and Information Science, vol. 1269, pp. 527–541. Springer (2020)
9. Hahn, E.M., Hashemi, V., Hermanns, H., Turrini, A.: Exploiting robust optimization for interval probabilistic bisimulation. In: Agha, G., Van Houdt, B. (eds.) Quantitative Evaluation of Systems. pp. 55–71. Springer International Publishing, Cham (2016)
10. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Param: A model checker for parametric markov models. In: CAV. pp. 660–664 (2010)
11. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: iscas m c: a web-based probabilistic model checker. In: International Symposium on Formal Methods. pp. 312–317. Springer (2014)
12. Hartmanns, A., Hermanns, H.: The modest toolset: An integrated environment for quantitative modelling and verification. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 593–598. Springer (2014)
13. Hashemi, V., Hermanns, H., Song, L., Subramani, K., Turrini, A., Wojciechowski, P.: Compositional bisimulation minimization for interval markov decision processes. In: Language and Automata Theory and Applications. pp. 114–126. Springer (2016)
14. Hebert, L.: Specification, verification and optimisation of business process. Technical University of Denmark (2014)
15. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic symbolic model checking with prism: A hybrid approach. In: Katoen, J.P., Stevens, P. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 52–66. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
16. Lam, V.S.W.: Formal analysis of BPMN models: a nusmv-based approach. Int. J. Softw. Eng. Knowl. Eng. **20**(7), 987–1023 (2010), https://doi.org/10.1142/S0218194010005079
17. Martin Glinz, S.F.: Software quality selected chapter, chapter 7, process quality. University of Zürich, Institut for Informatics (2007)
18. Mendoza Morales, L.: Business process verification: The application of model checking and timed automata. CLEI Electronic Journal **17**, 3–3 (08 2014)
19. Ou-Yang, C., Lin, Y.D.: BPMN-based business process model feasibility analysis: a Petri net approach. vol. 46, pp. 3763–3781. Taylor and Francis (2008), https://doi.org/10.1080/00207540701199677

20. Parker, D.: Lecture 14 model checking for MDPs. University of Oxford, Department Science (2011)
21. SIG, V.Q.W.G...A.: Automotive SPICE Process Assessment / Reference Model. Automotive SPICE (2017)