BPMNinvest: Feedback on Change in Process Models

Thomas M. Prinz[1,*], Yongsun Choi[2]

[1]Course Evaluation Service, Friedrich Schiller University Jena, 07743 Jena Germany
[2]Department of Industrial and Management Engineering, Inje University, Gimhae, Republic of Korea

Abstract

Developing sound business process models is not trivial, even for experts. Although new tools, such as the BPMN Analyzer 2.0, are evolving, the majority of them are based on model checking or state-space exploration leading to known drawbacks in software testing such as fault blocking and fault masking. This paper presents *BPMNinvest* as an extension of the *bpmn.io* framework that provides fault feedback on every change of the model through structural approaches of the state of the art. This feedback contains a detailed explanation and localization of the fault and the possibility to manifest the fault as error during execution. For this reason, it is also suitable for teaching and quality training purposes. Currently, it detects 11 kinds of flaws and faults and was developed with the challenges of *coverage*, *immediacy*, and *consumability* in mind.

Keywords

Business process model, Analysis, Modeling, Soundness, Loop decomposition.

1. Introduction

Business Process Management (BPM) involves investigating process models that describe the flow of various tasks to achieve specific business goals [1]. Although process models are typically created by experts, such models can contain structural errors [2, 3]. *Soundness* [4] is considered a minimum quality criterion [5] of process models guaranteeing the *option to complete*, a *proper completion*, and *no dead activities* [1]. Soundness further corresponds to the absence of *deadlocks* (in which the execution blocks locally) and *lacks of synchronization* (where the same task is executed multiple times unintentionally) [2].

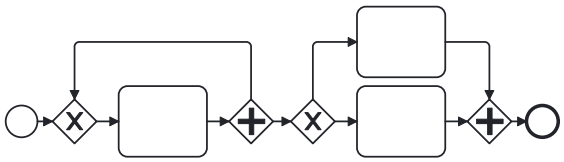
In [6], we presented our tool *Mojo*[1] for static analysis of business process models to investigate soundness providing immediate feedback with detailed diagnostic information. Besides its extendable core module, *Mojo* provided a plugin for the *Activiti BPMN 2.0 Designer*[2]. Inspired by the new tool *BPMN Analyzer 2.0* [7], we have reimplemented the algorithms in the *bpmn.io*[3] framework as a plugin called *BPMNinvest*. *BPMNinvest* leverages new insights from previous work [8, 9, 3] to accelerate and simplify the implementation and to improve the quality of diagnostic information. As a consequence, all analyses can be done directly in the browser using JavaScript. *BPMNinvest* is available as a demo and as a video demonstration[4]. It is open source under the MIT license, and its implementation is available[5].

BPMN Analyzer 2.0 [7] utilizes an impressive interface, but exposes the shortcomings of analysis based solely on state space exploration. From a software testing theory perspective, such drawbacks have been intensively studied in previous work [10]. The main disadvantages are: *fault distance* (a modeling fault incurs an unsound behavior, e. g., a deadlock, somewhere else "downstream" the process model); *fault masking* (an unsound behavior makes another unsound behavior disappear); *fault illusion* (a substantive unsound behavior leading to another kind of behavior that is not expected); and *fault blocking*

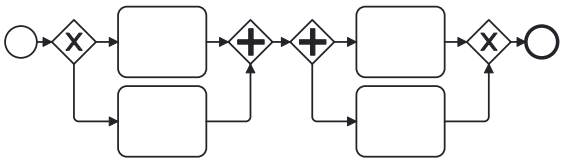
Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 23rd International Conference on Business Process Management (BPM 2025), Seville, Spain, August 31st to September 5th, 2025.

*Corresponding author.
Thomas.Prinz@uni-jena.de (T. M. Prinz); yschoi@inje.edu (Y. Choi)
0000-0001-9602-3482 (T. M. Prinz); 0000-0002-8605-4055 (Y. Choi)

[1]https://github.com/guybrushPrince/mojo.core, visited July 2025
[2]https://www.activiti.org/, visited July 2025
[3]https://bpmn.io, visited July 2025
[4]https://guybrushprince.github.io/bpmninvest/ and https://youtu.be/fHmVLcOeMZI, visited July 2025
[5]https://github.com/guybrushPrince/bpmninvest, visited July 2025

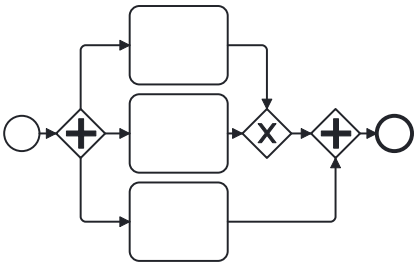


(a) *Fault masking:* Unbounded number of tokens produced by the left loop hinders a deadlock at the right structure. The browser crashes when to analyze this model with *BPMN Analyzer 2.0*.

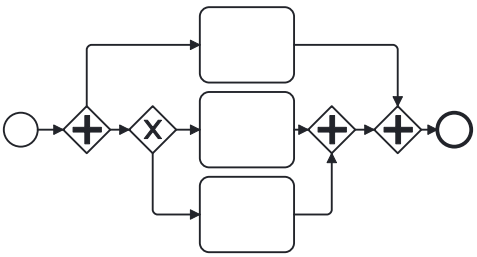


(b) *Fault blocking: BPMN Analyzer 2.0* detects the deadlock situation at the left-most parallel gateway but not the lack of synchronization at the right-most exclusive gateway.

Figure 1: Process models showing (a) *fault masking* and (b) *fault blocking* when tested with *BPMN Analyzer 2.0*.



(a) *Fault illusion: BPMN Analyzer 2.0* detects a deadlock situation at the right-most parallel gateway but it is caused by the previous lack of synchronization at the exclusive gateway.

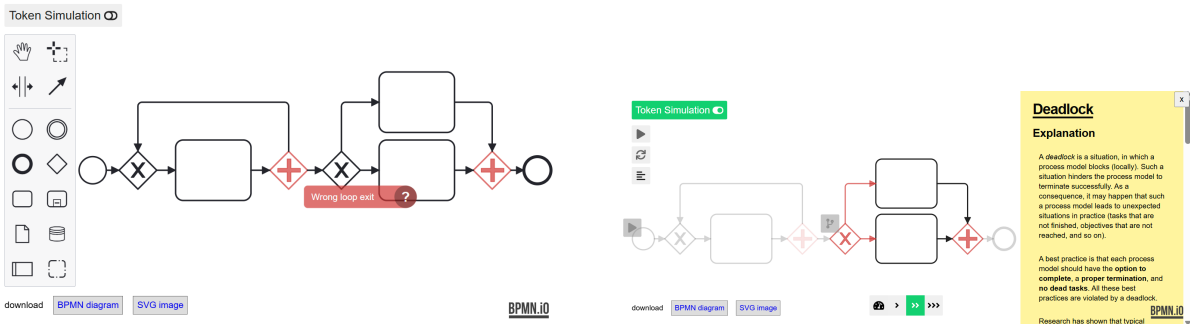


(b) Although *BPMN Analyzer 2.0* states that this model is not guaranteed to terminate, no visual feedback is provided.

Figure 2: Process models showing (a) *fault illusion* and (b) no visual feedback when tested with *BPMN Analyzer 2.0*.

(an unsound behavior, such as deadlock, prevents other unsound behaviors from being reached). To verify such disadvantages, example cases in [10] have been analyzed with *BPMN Analyzer 2.0*. Figure 1 (a) illustrates a BPMN process model with *fault masking*: The left loop produces an unbounded number of tokens (a lack of synchronization), which hinders the right deadlock-incurring structure to encounter a deadlock. Surprisingly, this simple process model crashes the browser while using *BPMN Analyzer 2.0*. We expect that the unbounded number of tokens leads the algorithms behind *BPMN Analyzer 2.0* not to terminate. Figure 1 (b) shows a BPMN process model with *fault blocking* for which the *BPMN Analyzer 2.0* can only show hints for the deadlock in the left-most parallel gateway. The potential lack of synchronization at the right-most exclusive gateway remains undetected. A *fault illusion* is illustrated in Figure 2 (a) where the *BPMN Analyzer 2.0* detects a deadlock in the right-most parallel gateway although the "real" fault is the non-synchronizing exclusive gateway causing the right-most parallel gateway to have a remaining token. Unfortunately, in some situations, as illustrated in Figure 2 (b), *BPMN Analyzer 2.0* does not provide any visual feedback except a statement that the model is not guaranteed to terminate. *At this point, we want to emphasize that these drawbacks are* not *the result of a faulty implementation of the* BPMN Analyzer 2.0. The tool is well implemented and the authors have invested much effort to provide easy and valuable modeling support. *The core of the drawbacks lay in the used methodology.*

This paper describes our new tool *BPMNinvest*, which orientates on the usability aspects and features of *BPMN Analyzer 2.0*. After any change of a process model, the detected faults are illustrated directly in the model without any noticeable delay (cf. Figure 3 (a)). The methodology of loop decomposition in [8] and of compiler-based algorithms for detecting deadlocks and lacks of synchronization in acyclic process models [3] are utilized for an *on the fly* analysis of control flow faults. One strong benefit of those methodologies is that they find those faults without state-space exploration, i. e., they are fast (with a cubic computational worst-case time complexity) and they find faults instead of errors (fault distance, masking, blocking, and illusion do not manifest). In addition, the illustrated faults in the process model are *comprehensible* as a detailed explanation and localization are provided attached with literature references



(a) Faults are highlighted with colored elements and info boxes while hovering over the element with the mouse pointer.

(b) Each fault is explained in detail by *BPMNinvest* and a simulation is provided that tries to force a faulty situation.

Figure 3: Two screenshots of *BPMNinvest* with (a) immediate fault feedback in the process model and (b) an explanation of the fault with the possibility of simulation.

(a) Inclusive gateways are not supported by *BPMN Analyzer 2.0* but in *BPMNinvest*.

(b) Boundary events are not analyzed by *BPMN Analyzer 2.0* but in *BPMNinvest*.

Figure 4: BPMN elements supported in *BPMNinvest* but not in *BPMN Analyzer 2.0*.

and the possibility to force the fault by a simulation. Besides those features, *coverage*, *immediacy*, and *consumability* are the main challenges for standard users (e. g., developers) being confronted with analysis results from formal analysis [2]. We addressed these challenges during development of *BPMNinvest*.

The rest of this paper is structured as follow: Section 2 describes the innovative features of *BPMNinvest* whereas Section 3 gives a short report of its maturity. The paper is concluded by Section 4.

2. Innovations

Regarding the main innovations of an *instantaneous* and *comprehensible* control flow error detection of *BPMN Analyzer 2.0*, *BPMNinvest* extend those innovations with a *broad coverage of BPMN elements*, an *analysis on change*, and a very *detailed fault feedback*.

2.1. Broad Coverage of BPMN Elements

BPMNinvest covers most elements available in the *bpmn.io* modeler except *complex gateways* and *message flows*. It fully supports *inclusive gateways* regarding [11] (cf. Figure 4 (a)) and *boundary events* (even non-interrupting boundary events, cf. Figure 4 (b)). Both are not covered by the *BPMN Analyzer 2.0*. Expanded sub-processes as well as different lanes and pools are supported as well. The model can further have multiple (implicit) start events and multiple (implicit) end events. For this reason, *BPMNinvest* has a broad coverage of elements of the BPMN specification used in practice.

2.2. Analysis on Change

Although Fahland et al. [2] stated an instantaneous feedback as 500ms or less, our analyses only require in 95% of considered cases less than 2ms [3, 9]. For this reason, feedback is not just instantaneous, it is

on change. Of course, *BPMNinvest* is implemented in JavaScript and, therefore, interpreted and compiled at runtime. Thus, the implementation is slower than the Java implementation of *Mojo*. However, even for big process models, we do not recognize any obvious delay during modeling. In the future, collecting detailed diagnostic information for explanations and simulations could be performed on demand if there should be drawbacks for big models.

2.3. Comprehensible and Detailed Fault Feedback

Instead of showing errors, which occur during execution (as returned by a state-space exploration approach), *BPMNinvest* collects 11 kinds of flaws and *faults* in the entire model, i. e., fault masking, fault illusion, and fault blocking are not a problem. Regarding best practices, e. g., stated in [1], *BPMNinvest* reports the flaws: *no (implicit) start event*, *no (implicit) end event*, *usage of implicit start*, *usage of implicit end*, and *wrongly structured gateway*. As faults, it detects *parallel or inclusive gateways as loop exits*, *parallel gateways as loop entries*, *deadlocks during loop initialization*, *deadlocks*, *endless loops*, and *lack of synchronization*. Since it is based on research results of [8] and [3], it is complete regarding *soundness*.

Figure 3 (a) illustrates *BPMNinvest* how it highlights BPMN elements with a fault. Overall, informative hints are highlighted blue (such as implicit starts and ends), wrongly structured gateways are highlighted yellow, and faults are colored red. If the user hovers the elements, a short explanation of the fault/flaw is displayed. Clicking on the element and on the question mark opens a detailed fault explanation panel. Furthermore, the fault with the related BPMN elements are focused in the model (as illustrated in Figure 3 (b)). It gives a general explanation of the fault, a model-oriented explanation, some repair suggestions, and references to the literature if more detailed information is required. Most flaws and faults provide a *simulation*, i. e., the attempt to manifest the fault as an error during execution. This is not trivial as a fault is not directly associated to a trace of execution or a similar concept. Since fault blocking, fault illusion, and fault masking may appear during simulation, forcing the error is not guaranteed but usually works fine. The simulation is done via a token game being based on the *bpmn.io Token simulation* extension. If the error is reached, the simulation automatically pauses and provides a further hint about the error.

3. Maturity of the Tool

BPMNinvest is the result of ongoing refinements on the algorithms of our business process analysis tool *Mojo*. Those algorithms have been tested multiple times and were leading with regard to other approaches in both diagnostic information and performance regarding running time [3]. New insights about loops and their relevance during soundness checking lead to even more efficient, accurate, and localized fault detection algorithms [8]. The tool is open source and everyone can help to further improve the quality of code although we take much effort to have a high code quality while staying efficient. Furthermore, the tool is practical enough to be integrated with the *Progression* engine [12]. *BPMNinvest* will be further extended with new research results, e. g., with a "reachability checker" that will allow users to ask if a specific state is reachable from an initial state [13]. The tool will be further improved regarding usability after collecting feedback from users.

4. Conclusion

Modeling correct business process models is not trivial. *BPMNinvest* supports users during modeling by giving feedback on change directly in the model with a detailed explanation and a simulation of the faults. Currently, it detects 11 kinds of flaws and faults. As it is based on structural instead of behavioral analysis, it is able to detect faults behind other faults and remove difficulties such as fault masking, illusion, and blocking. *BPMNinvest* addresses *coverage*, *immediacy*, and *consumability* of the analysis results.

In the future, we plan to extend *BPMNinvest* to handle further flaws and faults such as loops without entry, loops without exits, complex gateways, and message flows. Furthermore, *BPMNinvest* will get a "reachability checker" allowing to ask if a specific state can be reached from an initial state.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.

[2] D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, K. Wolf, Analysis on demand: Instantaneous soundness checking of industrial business process models, Data Knowl. Eng. 70 (2011) 448–466. (Mar. 2024).

[3] T. M. Prinz, W. Amme, Control-flow-based methods to support the development of sound workflows, Complex Syst. Informatics Model. Q. 27 (2021) 1–44.

[4] W. M. P. van der Aalst, Verification of workflow nets, in: P. Azéma, G. Balbo (Eds.), Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings, volume 1248 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 407–426. doi:10.1007/3-540-63139-9_48.

[5] B. F. van Dongen, J. Mendling, W. M. P. van der Aalst, Structural patterns for soundness of business process models, in: Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China, IEEE Computer Society, 2006, pp. 116–128. doi:10.1109/EDOC.2006.56.

[6] T. M. Prinz, N. Spieß, W. Amme, A first step towards a compiler for business processes, in: A. Cohen (Ed.), Compiler Construction - 23rd International Conference, CC 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings, volume 8409 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 238–243.

[7] T. Kräuter, P. Stünkel, A. Rutle, Y. Lamo, H. König, BPMN analyzer 2.0: Instantaneous, comprehensible, and fixable control flow analysis for realistic BPMN models, in: A. del-Río-Ortega et al. (Ed.), Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum at BPM 2024 co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024, volume 3758 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 66–70. URL: https://ceur-ws.org/Vol-3758/paper-11.pdf.

[8] T. M. Prinz, Y. Choi, N. L. Ha, Soundness unknotted: An efficient soundness checking algorithm for arbitrary cyclic process models by loosening loops, Inf. Syst. 128 (2025) 102476.

[9] T. M. Prinz, Y. Choi, N. L. Ha, Understanding and decomposing control-flow loops in business process models, in: C. Di Ciccio et al. (Ed.), Business Process Management - 20th International Conference, BPM 2022, Münster, Germany, September 11-16, 2022, Proceedings, volume 13420 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 307–323.

[10] T. M. Prinz, W. Amme, Why We Need Static Analyses of Service Compositions — Fault vs. Error Analysis of Soundness, International Journal on Advances in Intelligent Systems 10 (2017) 458–473. ISSN 1942-2679.

[11] T. M. Prinz, N. L. Ha, Y. Choi, Transformation of cyclic process models with inclusive gateways to be executable on state-of-the-art engines, in: J. Filipe et al. (Ed.), Proceedings of the 27th International Conference on Enterprise Information Systems, ICEIS 2025, Porto, Portugal, April 4-6, 2025, Volume 2, SCITEPRESS, 2025, pp. 280–291. doi:10.5220/0013386400003929.

[12] T. M. Prinz, Y. Choi, A. Vetterlein, Progression: A lightweight BPMN engine simplifying the execution and monitoring of process models, in: [To be published at the 23rd International Conference on Business Process Management (BPM 2025) Forum], Springer, 2025.

[13] T. M. Prinz, C. T. Schwanen, W. M. P. van der Aalst, Deciding (sub-marking) reachability in $O(P^2 + T^2)$ for sound acyclic free-choice workflow nets, in: E. G. Amparore, L. Mikulski (Eds.), Application and Theory of Petri Nets and Concurrency - 46th International Conference, PETRI

NETS 2025, Paris, France, June 22-27, 2025, Proceedings, volume 15714 of *Lecture Notes in Computer Science*, Springer, 2025, pp. 366–387. doi:10.1007/978-3-031-94634-9_18.