A Modeling Tool for Object-Centric Petri Net Variants

Maximilian König[1], Nils Schmitt[1] and Mathias Weske[1]

[1]Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

Abstract

Petri nets have long been used as a formal underpinning for business process models. With the advent of object-centric process management, process representations must capture complex object-interaction behavior instead of control flow, resulting in additional requirements for formalisms that lead to several object-centric extensions for Petri nets. Currently, these languages lack dedicated design-time tool support. Therefore, this paper introduces a modeling tool supporting several typed Petri net variants used in the object-centric context. Besides modeling, the tool provides structural verification and token simulation implementing the approaches' execution semantics. For broad availability, it is integrated into the OpenBPT platform.

Keywords

Petri nets, Object-centric, Synchronization, Modeling Tool, Token Simulation, PNID, OCPN, OPID, DOPID

1. Introduction

Traditionally, processes are considered from a fixed case notion with a specific start and end. Object-centric process management upends that assumption by considering all entities involved in a process as objects with individual behavior and observing their interaction [1, 2]. Objects are categorized into object types, and observed events store a reference to all involved objects. Supporting this paradigm requires the development of dedicated methods for the various tasks along the business process lifecycle [2].

Petri nets have proven to be a suitable formal underpinning for process modeling and mining. Specifically, a subclass of Petri nets called workflow nets has become a common formalism to evaluate properties such as soundness [3] while also serving as a common target language for process mining tasks, e.g., discovery and conformance checking [4]. In the object-centric context, the assumption of a singular start and end (place) as required by workflow nets is not applicable anymore. To provide a formal underpinning for object-centric processes, several Petri-net-based approaches have been developed that are used for object-centric process mining tasks such as discovery [1, 5] and conformance checking [6, 7].

While there exists extensive design-time support for traditional Petri net variants[1], current implementations do not capture the concepts required for object-centric process mining. On the other end of the spectrum, colored Petri nets (CPNs) [8] are more expressive and come with tool support. However, they are too generic to allow for meaningful object-centric analyses [2] and lack dedicated support for the required features. Thus, this demo presents an open-source modeling tool tailored to several kinds of typed Petri nets used for object-centric process mining. It supports the modeling of all features of Petri nets with identifiers (PNIDs) [9], object-centric Petri nets (OCPNs) [1], object-centric Petri nets with identifiers (OPIDs) [6], and a subset of the features introduced for data-aware OPIDs (DOPIDs) [7]. To enhance the design-time support, the tool additionally provides capabilities commonly implemented in tools dedicated to traditional Petri nets, namely automated structural verification during modeling and a manual token simulation implementing the execution semantics of all supported features. In the remainder of this paper, we briefly introduce the Petri-net-based approaches covered by the tool before explaining its main features in more detail and discussing its maturity.

Proceedings of the BPM 2026 Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum (BPM 2026), Toronto, Canada, September 27th to October 2nd, 2026.

maximilian.koenig@hpi.de (M. König); nils.schmitt@student.hpi.de (N. Schmitt); mathias.weske@hpi.de (M. Weske)

0000-0002-2244-1179 (M. König); 0009-0008-9971-3727 (N. Schmitt); 0000-0002-3346-2442 (M. Weske)

[1]https://www2.informatik.uni-hamburg.de/tgi/PetriNets/tools/quick.html

2. Typed Petri Net Variants

Object-centric Petri nets (OCPNs) have been developed [1] as one of the first approaches addressing the needs of object-centric process mining. They extend Petri nets by assigning each place an object type, and tokens in a place represent individual objects of the respective type (e.g., orders and products). Additionally, *variable arcs* can carry an arbitrary, unspecifiable number of tokens with a single transition firing. While this enables the capturing of tasks involving variable amounts of objects, OCPNs lack a representation of object relationships, e.g., the products belonging to an order. However, to analyze object-centric behavior and perform, e.g., more advanced conformance checking, information on individual objects and their relationships is required. To that end, Gianola et al. propose object-centric Petri nets with identifiers (OPIDs) [6], which build on OCPNs and extend them with features from Petri nets with identifiers (PNIDs) [9], namely tokens with identifiers representing concrete objects (e.g., order $o1$, product $p1$) and the option to assign places a set of object types (e.g., *order* × *product*) to represent and utilize links between objects to restrict transition firing. Object-centric event data typically comprises attribute values for objects. To account for that, data-aware object-centric Petri nets with identifiers (DOPIDs) [7] extend OPIDs to include complex data types for tokens, which were previously limited to carrying object identifiers. Further, transitions may specify guards based on these token values. Finally, a differentiation between exact and subset synchronization is formalized, meaning that a transition may require all linked objects for a reference object to be available for enablement (e.g., all products belonging to an order) or an arbitrary, unspecifiable subset. Table 1 summarizes the features supported by each approach and by our tool, which will be elaborated in the subsequent section.

Table 1
Overview of modeling concepts defined for the supported typed Petri net variants and our modeler. We use ✓to indicate support and – for its absence. *Sync.* abbreviates *Synchronization*.

	Typed Places	Complex Place Types	Inhibitor Arcs	Variable Arcs	Subset Sync.	Exact Sync.	Complex Data Types	Transition Guards
OCPN [1]	✓	–	–	✓	–	–	–	–
PNID [9]	✓	✓	✓	–	–	–	–	–
OPID [6]	✓	✓	–	✓	✓	–	–	–
DOPID [7]	✓	✓	–	✓	✓	✓	✓	✓
Tool Support	✓	✓	✓	✓	✓	✓	–	–

An example net for an order management process utilizing most of the introduced concepts is depicted in Figure 1. It includes two object types, namely orders and products. Initial transitions create new identifiers for both types, before transition *confirm order* selects an arbitrary amount of products through a variable arc and links them to an order in the *order* × *product* place. Transition *ship products* uses subset synchronization to ship an arbitrary subset of products related to the same order (indicated by the $\subseteq$ suffix on variable P). Finally, *archive order* requires exact synchronization (indicated by the $=$ suffix) between orders and products. Hence, transition *ship products* is enabled in the depicted marking while *archive order* must wait for products $p1$ and $p3$ to be shipped.

3. Overview & Features

The presented tool is a web application mainly written in JavaScript. The code is publicly available on GitHub[2] alongside detailed setup instructions. A demonstration of the tool is available via screencast[3], and a ready-to-use version is freely accessible either standalone[4] or integrated into the OpenBPT [5] platform [10]. In the following, we describe the main features of our tool, namely typed Petri net modeling, structural model verification, and token simulation.

[2]https://github.com/bptlab/openbpt-modeler-typed-pn-variants
[3]https://youtu.be/ZCfXzlpiemc
[4]https://bptlab.github.io/openbpt-modeler-dev/
[5]https://app.openbpt.org/ (Select *Typed Petri Net* when creating a new model after logging in.)

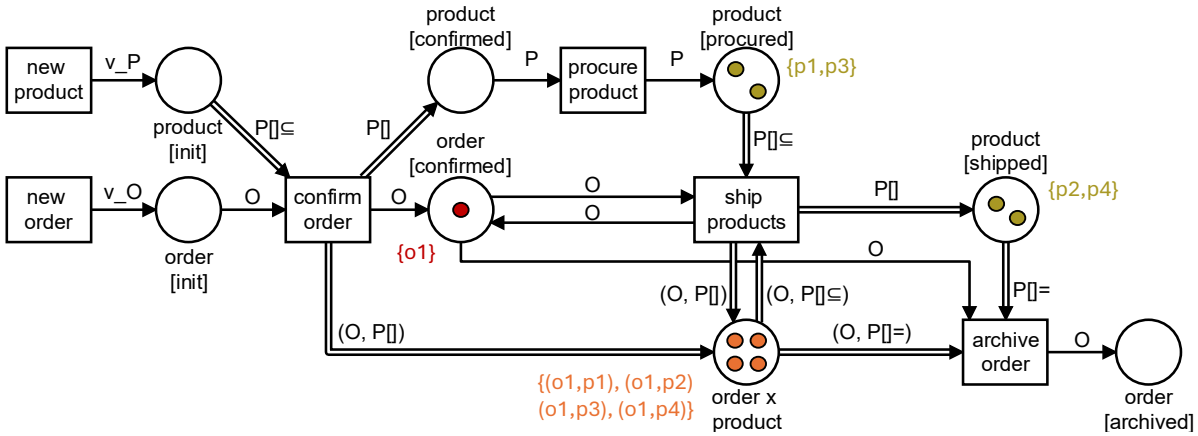


Figure 1: Object-centric Petri net with identifiers [6] and exact synchronization [7] representing an order management process with orders (O) and products (P). Prefix $v\_$ indicates identifier creation, double-lined arcs represent variable arcs, suffix [] signals the variable type of an arc, and $\subseteq$ and $=$ specify subset and exact synchronization, respectively.

3.1. Modeling

The modeling functionality is implemented based on the *diagram-js* library[6], which is an extensible toolbox that provides basic modeling features such as SVG-based diagrams, drag-and-drop visual modeling, copy and paste, import and export, and an internal event bus to throw and intercept events on user interaction. Besides modeling Petri nets with transitions, places, and arcs, several features from the typed Petri net variants previously introduced are supported. An overview can be found in Table 1, and a metamodel is provided in the repository. Object types with default variable names can be defined. Places can be assigned one or multiple object types. Variable names on arcs can be edited, and arcs directed towards a place may specify the creation of newly generated tokens. Tokens with editable values can be added to places. Currently, only data type ID is supported, i.e., the value of a token references the object it represents. However, the underlying metamodel allows for an extension with complex data types. Arcs can be configured to be inhibitor arcs or variable arcs, with the latter requiring a single object type to be specified as a variable type following the definition in [6]. Additionally, subset and exact synchronization can be differentiated. Transition guards have not yet been implemented. In summary, we fully support the modeling of OCPNs, PNIDs, OPIDs, and, due to the integration of exact synchronization, we partially support the modeling of DOPIDs. Created models can be exported as PNML files with compatibility to CoCoMoT[7] for conformance checking.

3.2. Structural Verification

To compensate for the inherent modeling complexity, several structural verification checks are executed after each update to the model. Violations are displayed as icons in the modeler next to the affected transitions, alongside an explanation on hover. The following properties are evaluated:

1. All places must be assigned at least one object type.
2. All variables on outgoing arcs must be defined, i.e., they must either be a variable creating a new identifier or they must also occur on an ingoing arc that is not an inhibitor arc.
3. There may not be multiple arcs with the same direction between a place-transition pair.

Note that the verification is purely structural and therefore independent of the current marking.

3.3. Token Simulation

To validate a model's behavior, its formal execution semantics are implemented and can be explored via a token-simulation mode. Entering the mode stores the currently configured marking and disables any

[6]https://github.com/bpmn-io/diagram-js
[7]https://github.com/bytekid/cocomot

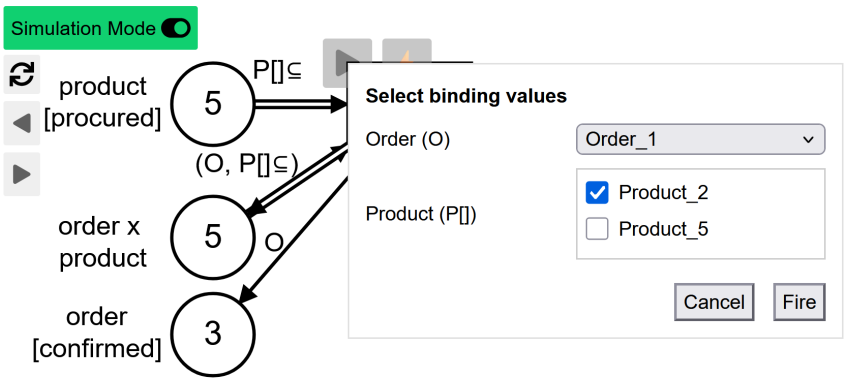


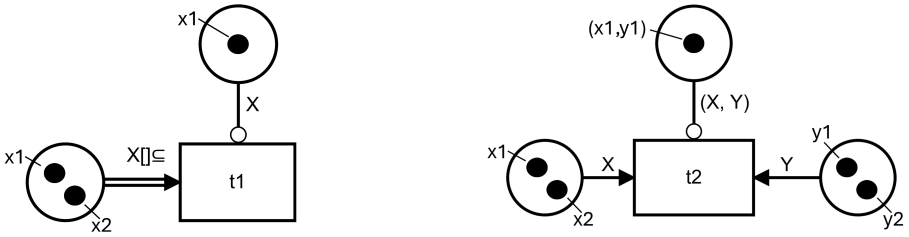
Figure 2: Screenshot from the tool in simulation mode. The buttons below the simulation mode toggle can reset the simulation, and undo or redo the last transition firing. The opened modal allows users to select the binding for firing the next transition.

changes to the model apart from the firing of transitions. Leaving the simulation mode reverts the net to the initial marking. During the simulation, the tool highlights all enabled transitions for the current marking. Users have two options for firing a transition. The quick-fire option randomly selects a valid binding, i.e., an assignment of token values to variables, and executes it for the selected transition. Alternatively, the user can manually configure the binding by selecting values for each variable. The selection options are dynamically adapted whenever a value is assigned. For example, the selection of *Order_1* in Figure 2 restricts the selectable products to those linked to the order. The tool internally stores the execution trace to enable undo and redo operations for individual transition firings.

The semantics are implemented according to the firing rules specified in [6] with exact synchronization as formalized in [7]. In addition, we adapt the semantics of inhibitor arcs presented in [9] to account for their interplay with variable arcs and links. For inhibitor arcs connected to places with a single object type, bindings are only valid if the annotated variable is not bound to any token value existing in that place. For example, the inhibitor arc in Figure 3a prohibits any bindings where variable X includes token value $x1$. For inhibitor arcs connected to places with at least two assigned object types, we only inhibit bindings where the variables on the inhibitor arc are assigned values matched by any single token in the place. For example, the inhibitor arc in Figure 3b prevents the firing of $t2$ only for bindings where X is bound to $x1$ and Y is bound to $y1$, due to token $(x1, y1)$. The other three combinations in the example are valid. With this logic, we enable more differentiated constraints compared to [9], whose semantics can still be replicated by creating a place for each object type of the connected place.

4. Discussion & Conclusion

Maturity. The current state of the tool is stable regarding modeling, structural verification, and token simulation. It fully supports OCPNs, PNIDs, and OPIDs and a subset of the concepts introduced for DOPIDs while lifting inhibitor semantics to DOPIDs. While CPN Tools [8] can technically support all mentioned features, the presented tool is specifically tailored to them, thus reducing complexity. In



(a) Inhibitor arc impacting a variable arc.

(b) Inhibitor arc on linking place.

Figure 3: Minimal examples for inhibitor arcs interacting with variable arcs and places with several object types assigned to them.

addition, the integration into OpenBPT makes the tool broadly available and allows for the development of services processing these models [10]. The integration into such a platform may also be helpful in teaching efforts on typed Petri nets. The export to PNML enables their use in conformance checking.

Future Work. To complement the current state, several lines of future work are planned. On the one hand, next steps should extend the tool with complex data types and transition guards to fully support DOPIDs. In addition, further model verification properties would help to better support users and improve understandability. Besides structural properties, that should include behavioral properties such as identifier soundness [11] extended to OPIDs and DOPIDs. Finally, the token simulation based on the implemented execution semantics lays the foundation for an automated instance simulation, thereby enabling object-centric event log generation.

Acknowledgments

We thank Pit Buttchereit and Tom Lichtenstein for their contributions to the tool development.

Declaration on Generative AI

During the preparation of this work, the authors used Gemini and LanguageTool in order to: Grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] W. M. P. van der Aalst, A. Berti, Discovering object-centric petri nets, Fundam. Informaticae 175 (2020) 1–40. doi:10.3233/FI-2020-1946.

[2] A. Seidel, M. Weske, M. Montali, et al., Object-centric process management: A research manifesto, Information Systems 141 (2026) 102728. doi:10.1016/j.is.2026.102728.

[3] W. M. P. van der Aalst, Verification of workflow nets, in: ICATPN 1997, volume 1248 of *LNCS*, Springer, 1997, pp. 407–426. doi:10.1007/3-540-63139-9_48.

[4] W. M. P. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, volume 448 of *LNBIP*, Springer, 2022. doi:10.1007/978-3-031-08848-3.

[5] A. Seidel, S. Winkler, A. Gianola, M. Montali, M. Weske, To bind or not to bind? Discovering stable relationships in object-centric processes, in: ER 2025, LNCS, Springer, 2025, pp. 223–241. doi:10.1007/978-3-032-08623-5_12.

[6] A. Gianola, M. Montali, S. Winkler, Object-centric conformance alignments with synchronization, in: CAiSE 2024, LNCS, Springer, 2024, pp. 3–19. doi:10.1007/978-3-031-61057-8_1.

[7] A. Gianola, M. Montali, S. Winkler, Object-centric processes with structured data and exact synchronization - formal modelling and conformance checking, in: CAiSE 2025, LNCS, Springer, 2025, pp. 185–202. doi:doi.org/10.1007/978-3-031-94571-7_11.

[8] K. Jensen, L. M. Kristensen, L. Wells, Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems, Int. J. Softw. Tools Technol. Transf. 9 (2007) 213–254. doi:10.1007/S10009-007-0038-X.

[9] K. M. van Hee, N. Sidorova, M. Voorhoeve, J. M. E. M. van der Werf, Generation of database transactions with petri nets, Fundam. Informaticae 93 (2009) 171–184. doi:10.3233/FI-2009-0095.

[10] T. Lichtenstein, M. König, M. Körner, A. Seidel, A. Ghasemi, M. Weske, OpenBPT: An extensible platform for conceptual modeling and analysis, in: CAiSE 2025 Forum, LNBIP, Springer, 2025, pp. 189–196. doi:10.1007/978-3-031-94590-8_23.

[11] J. M. E. M. van der Werf, A. Rivkin, M. Montali, A. Polyvyanyy, Correctness notions for petri nets with identifiers, Fundam. Informaticae 190 (2024) 159–207. doi:10.3233/FI-242169.