YAPNE: A Tool for Modeling and Automated Verification of Data Petri Nets

Christian Imenkamp[1,*], Joscha Grüger[2,3], Martin Kuhn[3], Christoph Matheja[4], Andrey Rivkin[5] and Agnes Koschmider[1]

[1]University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany
[2]Trier University, Universitätsring 15, 54296 Trier, Germany
[3]German Research Center for Artificial Intelligence (DFKI), Behringstraße 21, 54296 Trier, Germany
[4]University of Oldenburg, Ammerländer Heerstraße 114-118, 26129 Oldenburg, Germany
[5]Technical University of Denmark, Kgs. Lyngby, Denmark

Abstract

Data Petri Nets (DPNs) offer a robust formalism for modeling data-dependent processes. Despite their utility, a notable gap persists in available tools that seamlessly integrate a modeling environment with automated analysis techniques, such as soundness checks. Furthermore, there is a need for a modeling environment that makes modeling DPNs efficient and user-friendly. To cover this gap, the paper proposes Yet Another Petri Net Editor (YAPNE), a web-based tool for DPN modeling and soundness verification. YAPNE's intuitive graphical user interface (GUI) streamlines modeling and analysis activities, thereby enhancing the accessibility of formal models such as Data Petri Nets (DPNs) for both academics and practitioners.

Keywords

Data Petri Nets, DPN Modeling, Soundness Verification, Web-based Editor

1. Introduction

In the realm of Business Process Management (BPM), precise and effective modeling of business processes is of high importance. Among the various formalisms employed for this purpose, Petri nets have long served as a foundational tool for process representation and analysis. Traditional Petri nets excel at representing control flow, routing decisions, and task executions. However, in real scenarios, compliance checks often depend on the data variables created during the process execution [1]. Data Petri Nets (DPNs) extend classical Petri nets by incorporating data variables and guards on transitions that operate over such variables, allowing for a more powerful representation of complex, data-aware processes [2, 3]. Intuitively, DPNs combine control flow with global variables and guards. This enables the modeling of both control-flow and data-flow conditions within one formalism.

Modeling and automatically analyzing DPNs, however, presents unique challenges. This is particularly concerning the availability of comprehensive and accessible tool support. Many existing solutions either lack dedicated features for data-aware process modeling or do not seamlessly integrate the modeling environment with formal analysis capabilities. Powerful, feature-rich tools like CPN Tools provide mature modeling environments but are often complex, platform-dependent desktop applications with a steep learning curve [4]. Educational tools like WoPeD are more accessible but typically lack native support for data-aware modeling features and advanced analysis techniques, like data-aware soundness and model checking [5]. This forces users seeking automated analysis techniques for data-aware processes to employ separate, highly specialized verification engines like Ada. Such engines focus purely on algorithmic analysis (specifically, data-aware soundness and model checking) and do not

ICPM Doctoral Consortium and Demo Track 2025, October 20-24, 2025, Montevideo, Uruguay
*Corresponding author.
christian.imenkamp@uni-bayreuth.de (C. Imenkamp); grueger@uni-trier.de (J. Grüger); martin.kuhn@dfki.de (M. Kuhn); christoph.matheja@uni-oldenburg.de (C. Matheja); ariv@dtu.dk (A. Rivkin); agnes.koschmider@uni-bayreuth.de (A. Koschmider)
0009-0007-4295-1268 (C. Imenkamp); 0000-0001-7538-1248 (J. Grüger); 0000-0002-3242-1251 (M. Kuhn); 0000-0001-9151-0441 (C. Matheja); 0000-0001-8425-2309 (A. Rivkin); 0000-0001-8206-7636 (A. Koschmider)

offer a GUI for DPN modeling [6]. In contrast, frameworks like ProM [7] include support for modeling of DPNs, but their primary focus remains on process mining tasks such as discovery and conformance checking [1], with little to no support for the aforementioned automated verification tasks. Moreover, ProM is designed as a platform for integrating a wide range of process mining plug-ins, which can compromise the user experience of individual tools. This is particularly evident in the case of the plug-in for DPN modeling: the modeling process is unintuitive and lacks usability. As a result, users who require both modeling and verification capabilities for DPNs within a single tool are often forced into an inefficient and fragmented workflow. Users must rely on separate applications or non-user-friendly modules for visual modeling and formal verification, which creates a high barrier to entry.

To address these challenges, we propose YAPNE (Yet Another Petri Net Editor), a tool for modeling and analyzing Data Petri Nets. A key feature of YAPNE is its minimalist modeling environment, which enables fast creation and simulation of DPNs. As a fully client-side, open-source web application, it runs in any modern browser without requiring additional dependencies. The tool also supports the export and import of DPNs using the standard PNML format. Furthermore, the tool offers support for integrated and interactive verification of data-aware soundness, a well-established correctness criterion studied for DPNs in works such as [8, 9]. This paper demonstrates how all the aforementioned features are seamlessly accessible within a single environment and illustrates their relevance to the BPM community.

2. Tool Features

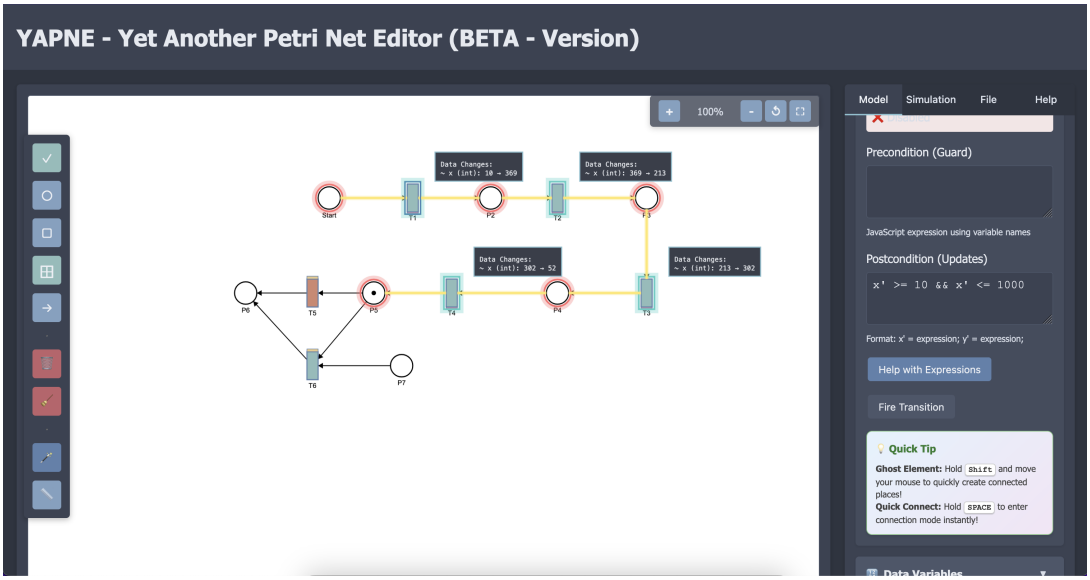

Figure 1: Screenshot of YAPNE's graphical user interface

The main goal of YAPNE is to provide a modeling tool for DPNs, thereby addressing the lack of a web-based modeling environment for faster creation and modification of DPN models. As shown in Figure 1, the main interface comprises a minimalistic canvas augmented by contextual toolbars and property panels. The core modeling tool comes with the following functionalities:

- **Graphical Modeling Environment:** A core design principle of YAPNE is to facilitate an efficient and intuitive modeling experience. The graphical editor provides a clean, uncluttered GUI with standard elements, implemented with a drag-and-drop interface with snap-to-grid functionality for creating and arranging net graph elements. Variable declarations, as well as transition guards, are defined in the dedicated panel.

- **Support for data import and export:** The tool supports import and export of models in the PNML format.[1]
- **Modeling modes:** The tool supports two modeling modes. The first mode enables users to add multiple elements of one single type (e.g., places). The second mode provides a potentially faster modeling experience by suggesting the next element type based on the currently selected graph node.
- **Accessible, Open, and Interoperable Architecture:** Implemented entirely in client-side JavaScript, YAPNE requires no server components, plugin installations, or user authentication. Its modular codebase, released under a permissive open-source license, invites extension by researchers and practitioners.

YAPNE comes with an embedded parser that performs real-time validation, detecting and reporting errors such as type incompatibility (when data variables are assigned values of incompatible types) and syntactic errors in expressions used in transition pre- and post-conditions. Detected errors are highlighted immediately, which prevents the propagation of malformed constraints through the model. In addition, the tool offers a user manual to support the DPN modeling experience.

As many other Petri net modeling tools, YAPNE comes with a net simulator. Currently, the tool supports two types of simulations. The first type is fully automatic (that is, enabled transitions and variable assignments are selected non-deterministically using the internal tool logic) and executes the net until no further transitions are enabled. The second is interactive, allowing users to manually select among multiple enabled transitions and, when applicable, specify corresponding variable assignments required by the chosen transition's guard.

Moreover, the tool includes functionalities supporting automated verification of data-aware soundness for DPNs. In summary, data-aware soundness can be boiled down to checking the following properties: (i) it must be always possible to reach the final control state (while the actual data variable assignment of this state is left unspecified), (ii) the final state must be always reached in a "clean" way (i.e., no tokens remain in places that are not part of the final marking), and (iii) it should be possible, for any transition, to have a reachable state in which it gets enabled (i.e., no dead transitions). Formally, data-aware soundness has been defined (with slight variations) in multiple papers, such as [3, 9, 8].

The aforementioned functionalities are as follows:

- **Algorithmic support:** The tool implements an algorithm presented in [8]. The algorithm enables the verification of data-aware soundness of DPNs, which may exhibit unsoundness due to issues at the control-flow level, the data-flow level, or their interplay.
- **Counterexample visualization:** When one of the aforementioned soundness sub-properties is violated, the tool provides counterexamples (finite runs on a DPN) that demonstrate the violation. The counterexamples can then be visualized on the net, which should help the users to understand the reason for the soundness violation.

It is important to note that, although existing papers on data-aware soundness provide tool support, the architectural choices made in YAPNE rendered it infeasible to directly integrate the algorithm presented in [8]. Consequently, the soundness algorithm was re-implemented from scratch and extended with counterexample highlighting to showcase the tool's visual capabilities. However, at the time of writing, the algorithm still requires a thorough validation. The scalability and broader user studies remain topics for future work.

These features make YAPNE a powerful tool for DPN modeling. The key innovation lies not just in the individual functionalities but in their tight integration. By allowing modelers to define data constraints and verify temporal properties within the same interface where the process structure is defined, YAPNE creates a fluid and continuous workflow. This immediate feedback loop, where the

[1]The guard syntax follows the format used in [10]. Alternative guard formats, such as the one proposed in [9], are not currently supported.

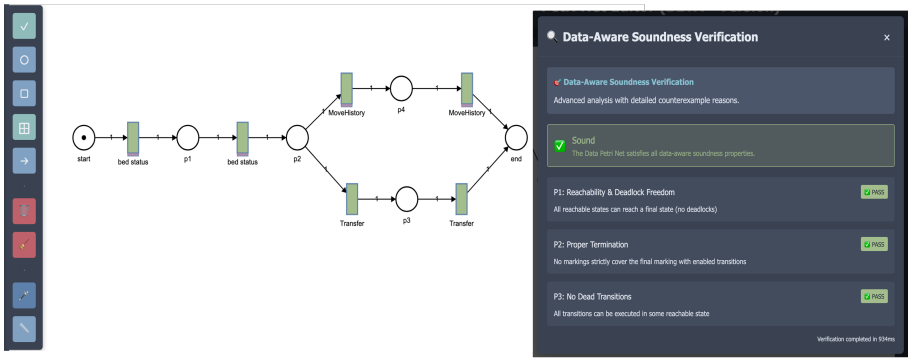

Figure 2: Digital whiteboard: discharge DPN

impact of a change can be verified instantly, empowers users to identify and correct logical flaws early in the design phase, significantly improving model quality and reliability.

YAPNE's source code is available on GitHub at https://github.com/chimenkamp/YAPNE-Yet-Another-Petri-Net-Editor. The web version can be found on GitHub Pages https://chimenkamp.github.io/YAPNE-Yet-Another-Petri-Net-Editor/. A "User Manual" guide and a video showcasing the main functionalities can be found here https://github.com/chimenkamp/YAPNE-Yet-Another-Petri-Net-Editor/tree/main/docs. The GitHub repository also includes example DPNs to demonstrate the capabilities of YAPNE. The examples can also be accessed directly in the editor (`Menu` (right) → `File` → `Examples`).

3. Tool Maturity

In this section, we discuss the maturity of YAPNE from two perspectives. First, we outline its technical stability, implementation choices, and evaluation through representative DPNs. Second, we compare YAPNE with established tools for Petri net modeling and verification, highlighting how it fills the gap left by existing tools by combining usability, accessibility, and native support for DPNs in one tool.

3.1. Technical Maturity

YAPNE has reached production-grade stability; the current release is fully web-based, its part related to modeling and simulation is feature-complete, and it can be used in both research and teaching projects. Extensive functional tests covering graphical modeling and model simulation were conducted to validate correctness before the tool release. The code base is implemented in pure JavaScript, a design choice that minimizes external attack surfaces, simplifies maintenance, and sustains high performance in modern browsers. The source code is released under the permissive MIT license, encouraging community contributions and extensions.

To further evaluate the capabilities of YAPNE we analyzed two DPNs also used in the evaluation of [8]. We selected a Petri net that is not data-aware sound and another that is data-aware sound. (1) The Digital whiteboard: discharge (see Figure 2) Petri net is data-aware sound because there are no control-flow or data-flow structures that could prevent it from eventually reaching the final marking from any reachable state. (2) Digital transfer: discharge (see Figure 3) is not data-aware sound. The transition "bed status" sets the value of **org1** to a random integer from $[1, \infty)$. Furthermore, the transition "Transfer" can only be fired if the value of **org1** is lower than 0 (i.e., $org1 \leq 0$). Hence, there exist no traces where the final state can be reached.

Although the tool has not undergone a formal user evaluation, it was tested by several academic experts familiar with DPNs, who confirmed its suitability for modeling and simulation tasks. In addition to academic tests, the tool has been used in teaching.

As mentioned in the previous section, the soundness checking component of the tool still requires

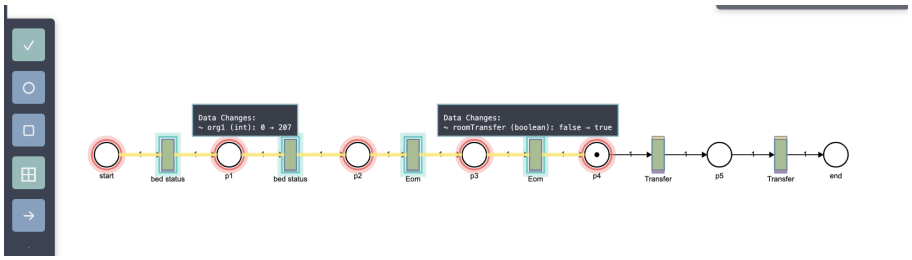

Figure 3: Digital transfer: discharge DPN including a counterexample trace that would lead to a deadlock

thorough validation. Nevertheless, the current implementation already offers the valuable feature of counterexample highlighting, which is not available in existing data-aware soundness checking tools.

3.2. Comparison with Existing Tools

Several established Petri net modeling tools support various data types and provide integrated verification support.

CPN Tools enables modeling, simulation, and verification of colored Petri nets and their extensions [4]. In [11], it was shown how DPNs can be modeled and analyzed using this tool. However, CPN Tools is desktop-only, has a rather steep learning curve, lacks cross-platform compatibility, and does not natively support volatile data or rapid guard-level data updates.

The ProM framework (https://promtools.org/) includes a DPN modeling plugin developed in [1]. Although the plugin is not very intuitive and is limited to modeling (e.g., conformance checking of DPNs requires separate plugins), it supports DPN import and export in PNML format, a feature missing in other tools mentioned in this section.

TAPAAL 4.0 is a mature platform for modeling and verifying Petri nets extended with time, color, and stochastic components [12]. While it has not yet been applied to DPNs, its support for colored extensions suggests it could be used similarly to CPN Tools [11]. This, however, again suggests that the tool provides no direct support for the DPN formalism.

Finally, when it comes to automated verification of temporal properties or more specific properties like data-aware soundness in DPNs, Ada offers a rich set of implemented analysis techniques [6]. At the same time, Ada cannot be used as a stand-alone tool as it lacks any modeling environment for DPNs.

YAPNE advances the current state-of-the-art by directly combining graphical modeling, interactive simulation, and data-aware soundness checking within a single environment. In contrast to established tools like CPN Tools or TAPAAL 4.0, it natively encodes DPN semantics. Furthermore, YAPNE integrates modeling with verification rather than distributing them across separate plug-ins or tools, allowing for a better user experience if both graphical modeling and verification are needed in one workflow. Thus, YAPNE fills a clear gap in tool support by offering a lightweight, web-based solution that prioritizes usability, accessibility, and seamless integration of modeling and verification tasks, simplifying modeling workflows for practitioners. Unlike earlier tools, YAPNE directly integrates modeling, simulation, and verification in a lightweight browser-based environment, making it uniquely accessible to researchers and students.

4. Conclusion and Future Work

This paper introduced YAPNE, a novel web-based tool that offers an integrated environment for the modeling and analysis of Data Petri Nets. By providing an intuitive graphical user interface and client-side accessibility, YAPNE can be used for both teaching and research activities. Moreover, its support for counterexample highlighting, facilitates automated analysis tasks such as data-aware soundness checking.

Future work will proceed in three main directions. First, we aim to conduct an extensive validation of the tool's functionalities, with particular emphasis on the data-aware soundness checking algorithm. Second, we plan to expand the tool's verification capabilities by enabling model checking for first-order extensions of temporal logics (e.g., [6, 9]). Lastly, we intend to enhance the simulation component by integrating various simulation engines (e.g., [10]), thus supporting more advanced performance analysis, hypothetical scenarios, and the generation of synthetic event logs based on simulation outputs.

Acknowledgments

This work received funding by the Deutsche Forschungsgemeinschaft (DFG), grant 496119880.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, and LanguageTool in order to: Paraphrase and reword, improve writing style, and Grammar and spelling check. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] F. Mannhardt, Multi-perspective process mining, Ph.D. thesis, Technische Universiteit Eindhoven, 2018.

[2] M. de Leoni, J. Munoz-Gama, J. Carmona, W. M. P. van der Aalst, Decomposing alignment-based conformance checking of data-aware process models, in: Proc. of OTM, volume 8841 of *LNCS*, Springer, 2014, pp. 3–20. doi:10.1007/978-3-662-45563-0_1.

[3] P. Felli, M. de Leoni, M. Montali, Soundness verification of data-aware process models with variable-to-variable conditions, Fundam. Informaticae 182 (2021) 1–29. doi:10.3233/FI-2021-2064.

[4] K. Jensen, L. M. Kristensen, Coloured Petri Nets - Modelling and Validation of Concurrent Systems, Springer, 2009.

[5] T. Freytag, M. Sänger, Woped - an educational tool for workflow nets, in: Proc. of BPM Demos, volume 1295 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2014, p. 31.

[6] P. Felli, M. Montali, S. Winkler, Ctl model checking for data-aware dynamic systems with arithmetic, in: Proc. of IJCAR, Springer-Verlag, Berlin, Heidelberg, 2022, p. 36–56.

[7] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, W. M. P. van der Aalst, The prom framework: A new era in process mining tool support, in: Applications and Theory of Petri Nets 2005, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 444–454.

[8] N. M. Suvorov, I. A. Lomazova, Verification of data-aware process models: Checking soundness of data petri nets, J. Log. Algebraic Methods Program. 138 (2024) 100953. doi:10.1016/J.JLAMP.2024.100953.

[9] P. Felli, M. Montali, S. Winkler, Linear-time verification of data-aware dynamic systems with arithmetic, in: Proc. of AAAI, AAAI Press, 2022, pp. 5642–5650.

[10] M. Kuhn, J. Grüger, C. Matheja, A. Rivkin, Data petri nets meet probabilistic programming, in: Business Process Management, Springer Nature Switzerland, Cham, 2024, pp. 21–38.

[11] M. de Leoni, P. Felli, M. Montali, A holistic approach for soundness verification of decision-aware process models, in: J. Trujillo, K. C. Davis, X. Du, Z. Li, T. W. Ling, G. Li, M. Lee (Eds.), Proc. of ER, volume 11157 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 219–235. doi:10.1007/978-3-030-00847-5_17.

[12] T. Dubois, K. G. Larsen, J. Srba, Statistical model checking of stochastic timed-arc petri nets, in: E. Amparore, Ł. Mikulski (Eds.), Application and Theory of Petri Nets and Concurrency, Springer Nature Switzerland, Cham, 2025, pp. 174–196.