petrinet-io: An Extensible Library for Petri Net Modeling and Simulation

Ricardo Perez Marin, Andrej Jurco, Andrea Burattin and Andrey Rivkin

DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark

Abstract

Petri nets represent a fundamental pillar of Business Process Management, necessary for modeling, analyzing, and explaining behavioral aspects of processes. Surprisingly, however, the current tool landscape for Petri net modeling is quite outdated, especially in terms of open-source and web-based applications. As a consequence, there is a clear limitation in terms of tool prototyping and extension for BPM-related tasks as well as in teaching environments. This paper presents `petrinet-io`, an open-source JavaScript library for Petri net modeling, simulation, and net import/export. Built on top of the diagram-js framework, `petrinet-io` provides an easy to instantiate Petri net editor, offering PNML import/export and visual simulation, as well as a modular architecture designed for extension, including future support for class- and task-level extensions.

Keywords

Petri net, Modelling, PNML, Simulation, JavaScript

1. Introduction

Petri nets [1] are one of the key formalisms for representing and analysing process behavior. In Business Process Management [2] (BPM), they provide a formal way to describe concurrency, synchronization, choice, sequences, and all other standard workflow patterns [3]. Their formal nature makes them suitable not only for modeling processes, but also for executing and analyzing them. Moreover, due to their formal nature, Petri nets are broadly used in teaching disciplines which span beyond BPM.

Despite this relevance, the state of the art in Petri net tooling reveals a practical and important gap. Many tools support Petri net editing, simulation, or import/export, but they are typically standalone and/or commercial applications, and often outdated. The first two aspects can create adoption complications, both by end-users (for example, in teaching settings) and developers (for example, researchers). For this reason, instead of reusing well-defined components, in many cases, Petri net developers often need to build a custom editor or adapt a monolithic tool.

The goal of `petrinet-io` is to address this gap by focusing on Petri net modeling as a reusable software library. While `petrinet-io` comes with an editor, that is instantiated with a few lines of code, this represents just a proof of concept of the capabilities of the underlying library. The broader contribution of our approach is a modular, embeddable, open-source infrastructure for Petri nets on the web, which also helps to foster broader research on Petri nets, including related empirical aspects [4].

`petrinet-io` emerged from the well known diagram-js ecosystem (https://github.com/bpmn-io/diagram-js), developed by Camunda of which the bpmn-io project (https://bpmn.io/) is another instance. This is an important design decision: rather than implementing low-level diagramming functionality from scratch, `petrinet-io` builds on a very mature foundation already used by widely adopted BPM-related modeling tools. As a result, the library benefits from existing modules for canvas management, editing, serialization, and graphical rendering (as SVG). In addition, from a programming standpoint, the system comes with element registries, dependency injection, modeling services, palettes, context pads, and event handling. In `petrinet-io` we tailored these foundations for Petri nets by adding Petri net-specific syntax, rendering, simulation, model import/export.

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

andbur@dtu.dk (A. Burattin); ariv@dtu.dk (A. Rivkin)

https://andrea.burattin.net/ (A. Burattin)

0000-0002-0837-0183 (A. Burattin); 0000-0001-8425-2309 (A. Rivkin)

Table 1
Comparison of Petri net tools according to the requirements. We use the symbol ✓̸ to indicate partial fulfillment. A '?' indicates an unclear situation.

Tool	R1	R2	R3	R4	R5	R6	R7	R8	R9
SAP Signavio	✓			✓	✓	✓			✓
WoPeD	✓	✓	✓	✓	✓		✓		
I ♡ Petri Nets	✓	✓	✓	✓			✓		✓
VSB Petri Net Editor	✓	✓	✓	✓	✓̸				✓
CPN Tools/CPN IDE	✓	✓		✓			?		
TAPAAL	✓	✓	✓	✓	✓		✓		
YAPNE	✓	✓	✓	✓	✓	✓̸	✓		✓
Renew	✓	✓	✓	✓	✓		✓		
OCPN studio	✓	✓	✓	✓	✓		✓		✓
`petrinet-io`	✓	✓	✓	✓	✓	✓	✓	✓	✓

In the rest of this paper, we present `petrinet-io`, an extendible web-based tool for standard P/T nets [1]. The contribution is relevant for BPM researchers, educators, and tool builders who need lightweight Petri net functionality that can be embedded, extended, and reused.

2. Innovation and Requirements

The main innovation of `petrinet-io` is that it is not conceived only as a Petri net editor, but as a *reusable (and extendible) library* for Petri net modeling. The need for such a library emerges directly from the current state of the art: many tools exist for Petri net modeling, but no well-established and standardized web-based editors that are open-source and extensible for the developer community. The aim of `petrinet-io` is therefore to address this lack. To clarify these gaps, we compare `petrinet-io` with some of the existing Petri net tools according to the following requirements:

R1: **Petri net modeling.** The user should be able to model a Petri net while enforcing its structural correctness and configure its marking.
R2: **Basic simulation capabilities.** Enabled transitions should be highlighted, and the user should be able to fire them to simulate the Petri net. Visual cues should indicate already fired transitions.
R3: **Autolayout capabilities.** As Petri nets can be used to model very large and complex systems, the ability to automatically lay out the model, thus improving its readability, becomes important.
R4: **PNML export/import.** The user should be able to load and save Petri nets designed on the canvas as in the PNML file format (https://www.pnml.org/).
R5: **High quality graphical export.** When it comes to Petri net research and education, the ability to produce high-quality graphical outputs from Petri net models means having "vector graphic" output, such as SVG or PDF.
R6: **Modern and standard modeling approach** (use of *context pads*). In the BPMN community, tools have greatly evolved in terms of usability, and some features have become commodities. For example, *context pads* present only the palette items relevant to the currently selected element. The editor should leverage these.
R7: **Open-source availability.** Other developers should be able to access the source code easily. The code should not be provided according to a free and open source software license.
R8: **Available as a library.** Other developers should be able to easily integrate Petri net capabilities into other applications using the supplied library. This includes reusable components or modules that can be reused or extended.
R9: **Web-based.** The user should be able to access the Petri net editor through a modern browser. The application should not require the installation of any additional software.

Several existing tools partially satisfy these requirements. While non-exhaustive, this overview aims to highlight the gap between the aforementioned requirements and the current availability

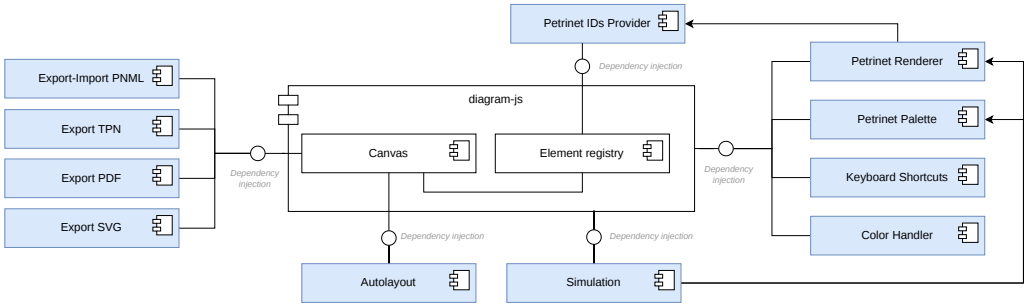

Figure 1: Component diagram of `petrinet-io`. At the center, there is the core of diagram-js and, around it, the different customized components of `petrinet-io` (in light blue background) are presented.

of tools. SAP Signavio (https://www.signavio.com/) is a professional web-based BPMM tool with Petri net simulation support. The tool is proprietary, and its PNML functionality was not found to work reliably. WoPeD [5] and CPN Tools [6]/CPN IDE [7] are very rich tools, providing simulation and PNML import/export, but they are desktop tools rather than web-based libraries; in addition, CPN Tools/IDE comprise several components, which come with different licenses. OCPN Studio (https://github.com/ELTE-DSED/ocpn-studio) is a recently introduced single-page web application which partially re-implements CPN Tools. Notably, the re-implementation does not come with a reusable library interface. I ♡ Petri Nets (https://www.fernuni-hagen.de/ilovepetrinets/) is a web-based project comprising a family of open source tools. Among them, some support simulation and PNML import/export. TAPAAL (https://www.tapaal.net/) is another open-source and feature-rich tool for modelling, simulation, and verification of Timed-Arc Petri nets and their extensions. The tool does not have a web interface. YAPNE (https://chimenkamp.github.io/YAPNE-Yet-Another-Petri-Net-Editor/) is a web-based tool for the modeling, analysis, and simulation of Petri nets and their data-aware extensions. However, the tool does not provide as a reusable library. Lastly, Renew (www.renew.de) is a Java-based tool for development, execution, and simulation of reference nets. The tool has no web interface. Table 1 summarizes the comparison of the tools with respect to the aforementioned requirements.

The tool presented in this paper, `petrinet-io`, aims to mitigate the fragmentation of tools currently existing by creating a library in which Petri net editors can be extended and modified, eliminating the need to develop a new tool every time.

3. Implementation

`petrinet-io` is built on top of diagram-js (https://github.com/bpmn-io/diagram-js) and relies on its solid foundations. diagram-js is an open-source JavaScript library for building interactive diagram editors in the browser. It already provides modules for canvas management, element manipulation, visual editing tools such as palettes and context pads, connection routing, and other editor functionality. These modules are provided as *services* and can be plugged in through the dependency injection system. Consequently, `petrinet-io` does not reimplement low-level diagramming functionality from scratch, but specializes an established modeling framework with Petri-net-specific syntax, semantics, rendering, simulation, and import/export.

Figure 1 presents a component diagram of the functionalities achieved by `petrinet-io`. In particular, everything revolves around the core of diagram-js. The dependency injection-based architecture makes it easy to plug in new modules or change modules individually. This makes the code very robust and *open for extension and closed for modification*, following the open-closed principle. This is achieved through a clear API that allows the library to operate independently and is not tied to any specific tool. This also makes it possible to extend `petrinet-io`'s data model toward richer Petri net variants and algorithms. For example, Colored Petri nets could be introduced by extending the token representation, arc inscriptions, transition guards, PNML serialization, and firing semantics, while reusing the existing canvas, palette, context pad, event bus, modeling services, and dependency injection infrastructure.

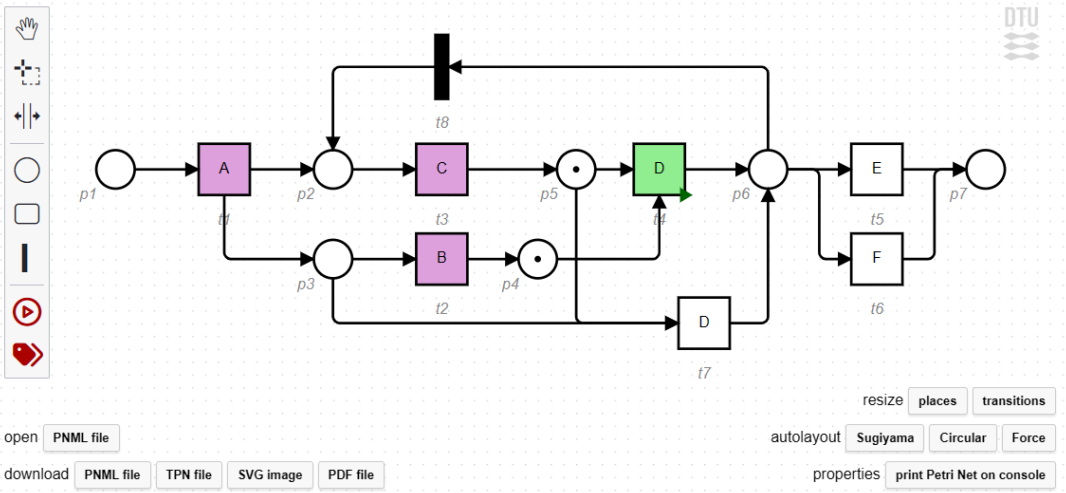

Figure 2: A screenshot of an editor instantiated for the `petrinet-io` library. Currently, simulation mode is active, as indicated by the red icon ⏵. Purple transitions are those that have already been executed, green transitions are the enabled ones.

Instantiating `petrinet-io` into a new application requires just a few lines of code. After installing it with 'npm install petrinet-io', a new Petri net editor is created with:

```
import { PetriNetIO } from "petrinet-io";
const pn = new PetriNetIO({
    container: "#canvas" // this is the ID of the element where the Petri net will be loaded
});
```

A containerized and fully working instantiation of the library is available at https://codesandbox.io/p/sandbox/elastic-lake-z2mlg8.

4. Maturity and Availability

The current maturity of `petrinet-io` can be assessed with respect to the requirements introduced in Section 2 and summarized in Table 1. The library comes with a simple instantiation that exposes its functionalities. A screenshot of such instantiation is in Fig. 2

With respect to R1, `petrinet-io` supports the modeling of ordinary place/transition Petri nets. The system enforces the bipartite structure of Petri nets by preventing invalid connections. Tokens are treated as properties of places, ensuring that markings are consistently represented with the underlying Petri net semantics. Regarding R2, the tool provides visual simulation capabilities which can be activated by enabling the "simulation mode" (⏵). Enabled transitions are highlighted in green, and the user can fire them by clicking them. After firing, the marking is updated by consuming tokens from input places and producing tokens in output places. Previously fired transitions are colored in purple. Regarding R3, the tool includes autolayout capabilities; select the corresponding tool at the bottom right. Currently, the Sugiyama, Circular, and Force Directed algorithms [8] are supported. With respect to R4, `petrinet-io` supports PNML import and export. Regarding R5, the library supports high-quality graphical export as SVG and PDF files. These can be relevant for research papers, teaching material, documentation, and presentations. With respect to R6, `petrinet-io` adopts a modern modeling interface inspired by established BPMN modeling tools based on a palette, context pads (see Fig. 1), keyboard shortcuts, and element-specific actions. With respect to R7, `petrinet-io` is open-sourced, with an Apache-2.0 license. Being a JavaScript library by design, makes `petrinet-io` fulfilling both R8 and R9.

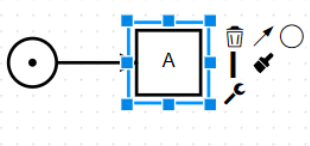

Figure 3: Context pad for a selected transition.

The tool's maturity has also been evaluated via testing of the two most important behavioral aspects: simulation and import/export. Simulation tests focus on basic control-flow patterns (i.e., sequences, AND-splits, AND-joins, XOR-splits, and XOR-joins). For these structures, the PNML input/output and token game have been unit-tested. Based on the above, we believe that `petrinet-io` should be considered a mature and practical library, making it well suited for use in teaching environments and in research prototypes that require Petri net functionality as a reusable component.

The library is available via npm at https://www.npmjs.com/package/petrinet-io. Its source code is available at https://github.com/processintelligence/petrinet-io, and an editor showcasing the capabilities is available at https://processintelligence.github.io/petrinet-io/. A screencast of the basic features is at https://youtu.be/woO0UveV9so.

5. Conclusion and Future Work

Petri nets remain highly relevant in BPM, but the current tool landscape is fragmented. Many existing tools provide useful functionality, but they are typically standalone applications rather than reusable libraries. This limits their adoption in settings where Petri net functionality needs to be embedded into larger systems. `petrinet-io` addresses this gap by providing a reusable and modern JavaScript library built on top of the mature diagram-js ecosystem. The library is open-source and easily instantiable (see https://processintelligence.github.io/petrinet-io/ for an example).

Future work will focus on extending the tool's analytical capabilities. Petri nets are not only a modeling notation, but also a formalism for behavioral analysis. Future versions of `petrinet-io` may therefore include reachability analysis, boundedness checking, liveness analysis, soundness checking, and other verification features relevant for BPM research and education. Additionally, we plan to support additional variants of Petri nets, such as Colored Petri nets and Data Petri nets.

Declaration on Generative AI

During the preparation of this work, the authors used generative AI tools in order to check grammar and spelling and improve phrasing. After using this service, the authors reviewed the content and take full responsibility for the publication's content.

References

[1] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (1989) 541–580.

[2] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, 2 ed., Springer, 2018.

[3] W. M. van der Aalst, A. H. ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow Patterns, Distributed and Parallel Databases 14 (2003) 5–51.

[4] J. Mendling, B. Depaire, H. Leopold, Empirical Research on Petri Nets, in: In Proc. of PETRI NETS, Springer, 2026, pp. 3–18.

[5] T. Freytag, P. Allgaier, A. Burattin, A. Danek-Bulius, WoPeD - A "Proof-of-Concept" platform for experimental BPM research projects, in: CEUR Workshop Proceedings, volume 1920, 2017.

[6] K. Jensen, L. M. Kristensen, L. Wells, Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems, Int. J. on STTT 9 (2007) 213–254.

[7] E. Verbeek, D. Fahland, CPN IDE: An Extensible Replacement for CPN Tools That Uses Access/CPN, in: Proceedings of the ICPM Doctoral Consortium and Demo Track 2021 co-located with 10th International Conference on Process Mining (ICPM 2021), 2021.

[8] S. Di Bartolomeo, T. Crnovrsanin, D. Saffo, E. Puerta, C. Wilson, C. Dunne, Evaluating Graph Layout Algorithms: A Systematic Review of Methods and Best Practices, Computer Graphics Forum 43 (2024).