

Generating Event Logs with CPN IDE

Eric Verbeek and Dirk Fahland

Eindhoven University of Technology, Groene Loper 5, 5612 AE Eindhoven, The Netherlands

Abstract

This extended abstract introduces the event log generation facility of CPN IDE. CPN IDE has replaced CPN Tools as a tool for editing and simulating (Coloured) Petri Net models. The main advantage of generating event logs with CPN IDE is that it typically will work on existing models by providing some additional information, and that typically the models themselves do not require any change. Basically, the only change that may be required is the consistent use of a single variable for the case identifier.

Keywords

Coloured Petri Net (CPN), CPN IDE, Event Logs, XES

1. CPN IDE

CPN IDE [1] was introduced at ICPM 2021 as a replacement for CPN Tools [2], a tool that is well-known in the Petri net community. CPN IDE offers a new JavaScript-based CPN editor on top of the Access/CPN library [3], which wraps the simulator of CPN Tools in a Java-based controller. As a result, CPN IDE can be extended either on the level of the CPN editor (using JavaScript) or on the level of the controller (using Java).

2. Generating event logs

Although it is possible to create event logs using CPN Tools [4], this is not a simple endeavor. First, it requires the user to incorporate special log-creating annotations into the CPN model, forcing to change the model itself. Second, it requires the user to create specific Standard ML scripts for the model at hand. As a result, a lot of manual labor is involved when using CPN Tools to create event logs. In CPN IDE, the user can simply load a model, run a simulation (which results in raw simulation records) and then configure the event log to be created. The only prerequisite for this is that a single variable is used throughout the model for the case identifier.

Figure 1 shows a screenshot of CPN IDE where a model has been loaded and has been simulated for 500 steps while recording events and their timestamps. In the main toolbar, at the top, the box preceding the *Record events* item has been selected, indicating that events are being recorded, and the box preceding the *Record event times* has been checked, indicating that timestamps are being recorded for the events. To get to such a state from which we can export an event log, the user only needed to check both these items as additional actions to performing the simulation as usual.


If timestamps are recorded, then two events are recorded in the back-end for every transition that (1) was fired and (2) does not have black fill: A start event and a complete event. If a transition has a black fill, no events will be recorded. The timestamp for the start event simply equals the current simulation time when the transition fired, but the timestamp of the complete event is more involved. For this, we compare the token values in the new marking (after the transition has fired) with the token values in the old marking (before the transition fired) and take the lowest time of any token value in the new marking that is not in the old marking. If no

ICPM 2023: Tool demonstrations, October 23–27, 2023, Rome, Italy

✉ h.m.w.verbeek@tue.nl (E. Verbeek); d.fahland@tue.nl (D. Fahland)

ORCID 0000-0002-1658-9679 (E. Verbeek); 0000-0002-1993-9363 (D. Fahland)

© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

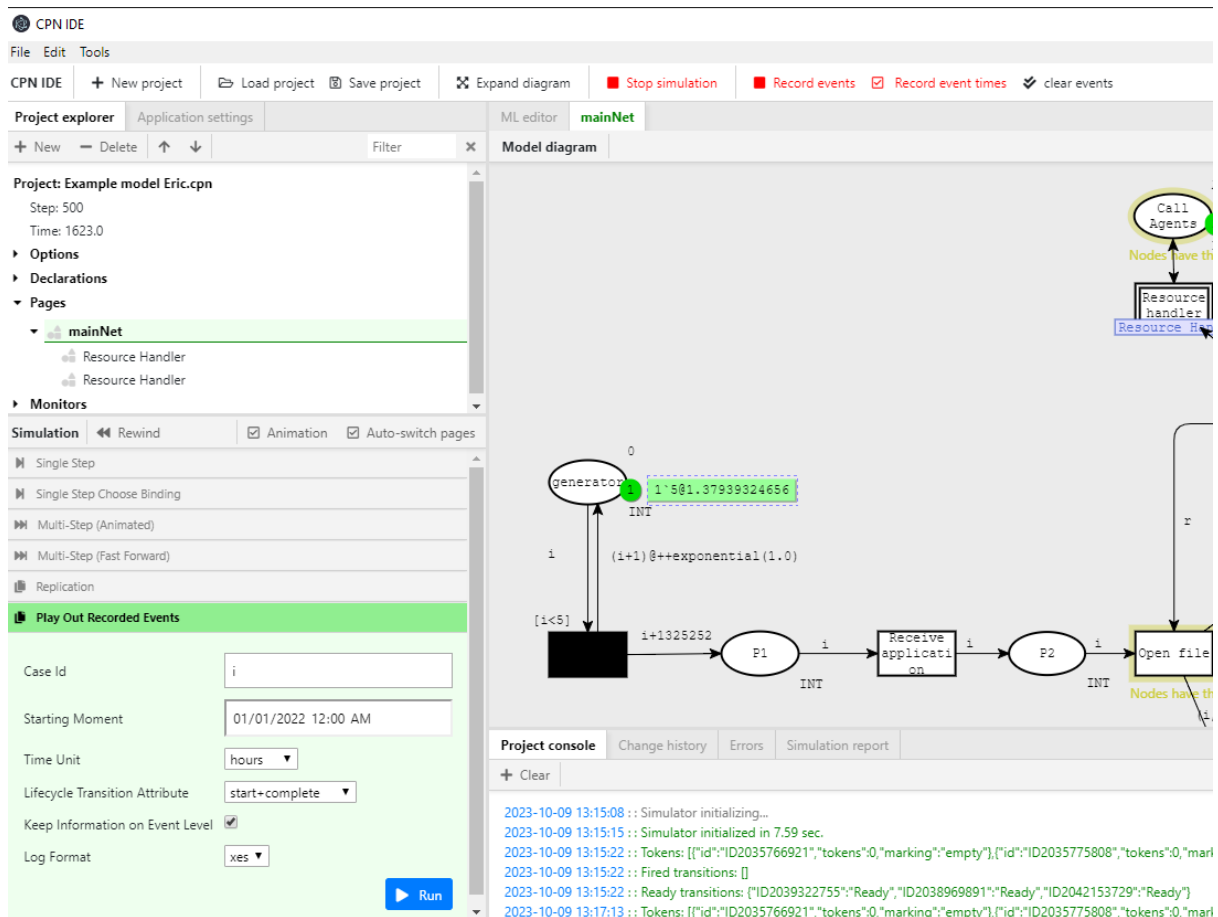


Figure 1: The facility for exporting the recorded events to an event log.

such token value exists, the current simulation time is used. As an example, if a transition fires at time t and produces tokens at times $t + 1$, $t + 2$, and $t + 3$, then the start event will have timestamp t and the complete event will have timestamp $t + 1$.

The recorded events can now be exported to an event log using the *Play Out Recorded Events* form in the lower left corner. For this export, the user needs to provide the following configuration settings:

- *Case Id*: This should be the name of the variable that is used as the case identifier of an event. Only recorded events that have this variable as an input in their binding will be exported.
- *Starting Moment*: The date and time that coincides with the simulation time 0.0.
- *Time Unit*: The relation between wall clock time and simulation time. As an example, if set to “hours”, then a simulation time of 1.0 means one hour.
- *Lifecycle Transition Attribute*: Which events will be exported to the event log:
 - *start*: Only start events will be exported.
 - *complete*: Only complete events will be exported.
 - *start+complete*: Both start and complete events will be exported.
 - *in transition name*: Only start events will be exported, but the lifecycle transition *start* will be replaced by the lifecycle transition as found in the transition name. As an example, if the transition name equals *A+resume*, then *start* will be replaced by *resume*, which effectively replaces the start event with a resume event. If no valid lifecycle transition is found in the transition name, *start* will be replaced by the empty string.
- *Keep Information on Event Level*: Has no use at the moment.

- *Log Format*: The format for the exported log, either *xes* (for XES [5] files) or *csv* (for CSV files).

After having provided these settings, the user can select the *Run* button in the form, which will ask the user for the name of the log file. After having provided this name, the log will be exported to a file with that name (with either the extension *.xes* or *.csv* added). The exact location of the file will be shown in the *Project console*.

3. Links

From <https://cpnide.org/> you can download the latest releases for CPN IDE. The latest release is CPN IDE 1.23.0725.

From the *Latest downloads* section on <https://cpnide.org/> you can also download a screencast on how to generate an event log in CPN IDE. In this screencast, we load an example model (which can also be downloaded from the *Latest downloads* section), start the simulator, select to record events together with their timestamps, and then simulate the model for 500 steps. After this, we select how the events that are recorded in the back by the controller should be exported to an event log, and have the controller export them accordingly.

On <https://github.com/cpn-io/cpn-js/> you will find the sources for CPN IDE. CPN IDE is Open Source.

4. Conclusion

CPN IDE has been extended with functionality that allows the user to generate event logs from (new or existing) CPN models, provided that a single variable is used for the case identifier throughout the model. The user can simply load the model into CPN IDE, select the option to record events (and possibly also event times), simulate the model for some time, and then to export an event log from the events that have been recorded.

The variable for the case identifier plays an important role in this extension, as a recorded event is exported to the event log if and only if the event has this variable as input. This also means that from the same collection of recorded events event logs from different perspectives can be exported. As an example, if a single variable is used throughout the model for a resource, then this variable could be used as well, leading to an event log containing a trace for every different resource.

For future work we are thinking of adding an option to import and/or export models from and to PNML [6] files, adding an option to create object-centric event data logs and adding an option to replay an event log on the model:

- *Import/export from/to PNML* [6]. CPN IDE (and CPN Tools as well) can be used together with ProM because ProM can import/export P/T nets from/to CPNXML files², which is the native file format for both CPN IDE and CPN Tools. However, if CPN IDE could import/export workflow nets from/to PNML, then other tools could also be used together with CPN IDE.
- *Creation of object-centric event data logs*. Every log that can be created by CPN IDE is now limited to a single notion of a case, like an order number or a client name. However, processes in real-life are often more complex and not limited to this single case notion. For this reason, work is being done on a standard for capturing Object-Centric Event Data (OCED) in a log³.
- *Replay of event log on model*. In a way, this is the counterpart of the facility to generate event logs as presented by this abstract. Instead of exporting the events that resulted from a simulation to an event log, we then import an event log and then 'guide' the simulation as best as possible using the imported events.

² As from ProM 6.10, the *CPNXML export (Petri net)* plug-in exports a P/T net to the CPNXML format and the *Import Petri net from CPNXML* file plug-in imports a P/T net from a CPNXML file.

³ See <https://www.tf-pm.org/resources/oced-standard> (accessed on October 10, 2023).

Acknowledgements

The authors would like to thank Tom Verberk for the work on his master project, which resulted in this facility to generate event logs in CPN IDE [7]. We also would like to thank Mitchel Brunings and Boudewijn van Dongen for their help and supervision on this project.

References

- [1] H. M. W. Verbeek and 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), ser. CEUR Workshop Proceedings, M. Jans, G. Janssenswillen, A. Kalenkova, and F. M. Maggi, Eds. CEUR-WS.org, 2021, pp. 29–30. [Online]. Available: http://ceur-ws.org/Vol-3098/demo_197.pdf
- [2] M. Westergaard and H. M. W. Verbeek. (2018) CPN Tools. [Online]. Available: <https://cpntools.org/>
- [3] M. Westergaard and L. M. Kristensen, The Access/CPN framework: A tool for interacting with the CPN Tools simulator, in Applications and Theory of Petri Nets, G. Franceschinis and K. Wolf, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 313–322.
- [4] A.K.A.d. Medeiros, C.W. Günther, Process mining: using CPN tools to create test logs for mining algorithms, K. Jensen (Ed.), Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005), DAIMI, vol. 576, University of Aarhus, Aarhus, Denmark (2005), pp. 177-190
- [5] G. Acampora, A. Vitiello, B. Di Stefano, W. M. P. v. d. Aalst, C. W. Günther, and H. M. W. Verbeek, IEEE 1849TM: The XES standard: The second IEEE standard sponsored by IEEE Computational Intelligence Society, IEEE Computational Intelligence Magazine, pp. 4–8, May 2017.
- [6] J. Billington and et. al., The Petri Net Markup Language: Concepts, Technology, and Tools,” in Application and Theory of Petri Nets 2003, W. Aalst and E. Best, Eds., vol. 2679, 2003, pp. 483–506.
- [7] T. Verberk, Generating Realistic Logs using a Colored Petri Net simulator, 30 Aug 2022, Master thesis, Eindhoven University of Technology. [Online]. Available: https://research.tue.nl/files/292963154/Verberk_T.pdf.