CPN Tools 4: A Process Modeling Tool Combining Declarative and Imperative Paradigms

Michael Westergaard[1,2⋆] and Tijs Slaats[3,4⋆⋆]

[1] Department of Mathematics and Computer Science,
Eindhoven University of Technology, The Netherlands
[2] National Research University Higher School of Economics,
Moscow, 101000, Russia
[3] IT University of Copenhagen
Rued Langgaardsvej 7, 2300 Copenhagen, Denmark
[4] Exformatics A/S, Lautrupsgade 13, 2100 Copenhagen, Denmark
m.westergaard@tue.nl, tslaats@itu.dk

Abstract. CPN Tools is a tool for modeling, simulating, and analyzing colored Petri nets. The latest iteration of the tool, CPN Tools 4, extends this with constraints known from declarative languages such as Declare and DCR Graphs. Furthermore, this version introduces an explicit process perspective, powerful extensibility allowing third parties to extend the tools capabilities, and a visualization perspective making it possible to make high-level visualizations of executions directly in the tool.
In our demonstration, we show how it is possible to create models incorporating declarative and imperative constructs and how to use these models to generate simulation logs that can be directly imported into ProM. We show off the new process perspective on top of colored Petri nets, exemplify the use of the perspective to generate readable Java code directly from models, and show how the visualization perspective makes it possible to show the formal underlying model alongside an easier-to-grasp for non-experts high-level visualization.
Our intended audience comprise current users of CPN Tools interested in recent developments and practitioners interested in colored Petri nets and hybrid models. We expect to tailor each demonstration to the wishes of the audience.

Standard imperative languages are suitable for the description of well-structured and well-understood processes. On the other hand, processes that are less well-understood or less well-structured, are often easier modeled using a declarative approach, where instead of specifying the next task to execute, constraints between tasks are described. Popular languages for imperative specifications include BPMN and (colored) Petri nets. Declarative modeling is a more recent

⋆ Support from the Basic Research Program of the National Research University Higher School of Economics is gratefully acknowledged.
⋆⋆ This research is supported by the Danish Agency for Science, Technology and Innovation through an industrial PhD Grant.

and less matured approach which has so far not found widespread application in industry yet, however the two declarative languages Declare [6] and DCR Graphs [2, 3] have been studied extensively in academia over the last decade. Declarative languages do not explicitly specify flow of control, but instead specifies constraints between actions; examples of such constraints are init(A), meaning that any execution has to start by executing A, and response(A, B), meaning that after executing A, B has to be executed at some point. Other constraints deal with choices and variations of the response constraint.

Hybrid modeling. Recently interest has emerged in hybrid approaches, where some aspects of a process are specified directly using imperative constructs and other aspects declaratively. This is useful if part of the process is well-structured and part is more free, or for going from an abstract, little-understood process, often modeled more naturally using declarative constraints, to a more concrete implementation which by nature is often more imperative. One such hybrid approach is implemented in CPN Tools 4 [5, 7]. This approach combines the places and transitions of colored Petri nets with the constraints of the Declare and DCR Graphs languages. Fig. 1 shows an example of a mixed declarative and imperative model. In the upper half of the screen we describe the registration of a patient using an electronic patient record, which is basically form-filling and well-suited for an imperative approach. In the bottom half we describe the treatment of the patient which is strongly knowledge-based, therefore more flexible and hence modeled using a Declarative approach. While these two aspects could have been modelled as separate processes (one imperative and the other declarative), using the hybrid approach allows us to join the two and show how they interact. Initially, only Receive Patient is required due to the declarative constraint init. After executing Disinfect Wound, Stitch Wound has to be executed because of a response between them. Registration and treatment can happen in parallel, but prescription of antibiotics is dependent on the patient data.

The model can be extended with time information and exercises to obtain simulation-based performance information. It is also possible to obtain a simulation log from CPN Tools, which can be imported directly into ProM 6.3 for analysis using a known process. CPN Tools also offers state-space analysis for ensuring the absence of errors such as dead-locks in the process. For more information about hybrid modeling, we refer the interested reader to [7].

Domain-specific visualization. While colored Petri net models are graphical, they are also complex to understand for non-experts. Previously, CPN Tools supported visualizations of such models by means of an external tool, but with version 4 such visualizations are internalized, making it possible to show model and visualization side-by-side without requiring external tools. In Fig. 2, we see two simple visualizations of the model from Fig. 1. The sequence diagram (left) shows a single patient interaction and is updated when simulation is conducted. The visualization is driven purely by the model, and as CPN Tools allows users full control over the simulation, can be used to demonstrate complex scenarios

in a simple way. The bar chart (Fig. 2 (right)) shows aggregated statistics over multiple simulations.

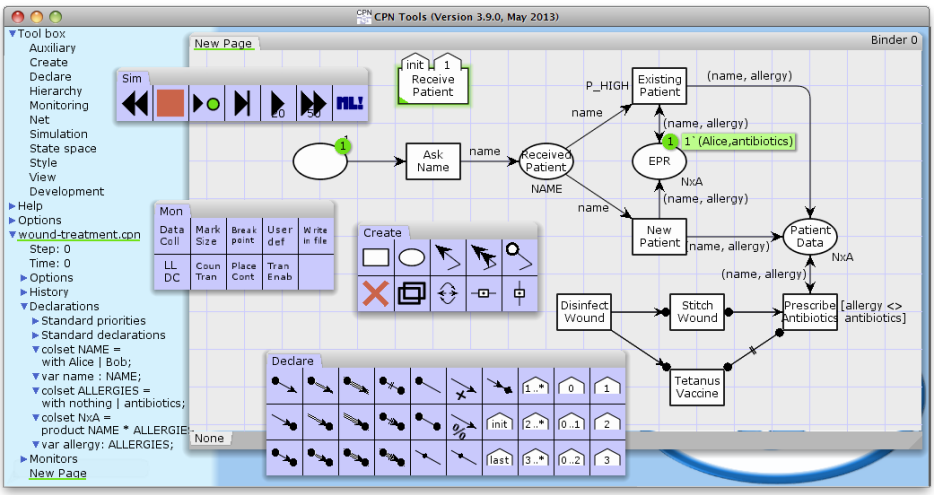

Fig. 1: Overview of CPN Tools with an example hybrid model for a hospital loaded.

Process-partitioned colored Petri nets. Colored Petri nets allow modelers a lot of freedom. Most importantly, it is very hard to separate the flow of data

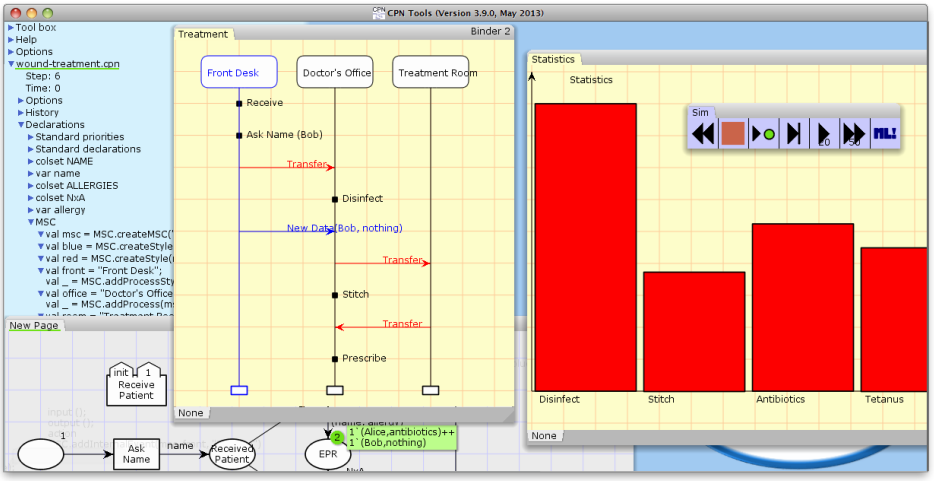

Fig. 2: Two visualizations of the simple model from Fig. 1. The model itself is just visible below the visualizations.

from the flow of control, which makes models hard to understand and analyze. Workflow Nets solved this problem for standard Petri nets, but some of the restrictions are too severe for efficient use of the higher-level formalism. Colored Workflow Nets [1] generalize Workflow Nets to colored Petri nets, but impose some restrictions that make models unnatural. Instead, CPN Tools implements Process-partitioned colored Petri nets (PP-CPNs) [4], which allow more flexibility and more natural models. PP-CPNs explicitly separate the flow of control and data, separating places into process places, local and shared places (for data), resource places, and communication (buffer) places.

PP-CPNs allow multiple instances of multiple process types to communicate, and hence supports an artifact-centric modeling style. Of course, classical Workflow Nets are recognized as PP-CPNs as one would expect. An example PP-CPN model of a simple producer/consumer system can be seen in Fig. 3 (top). Here, we have two kinds of processes communicating over a buffer place;

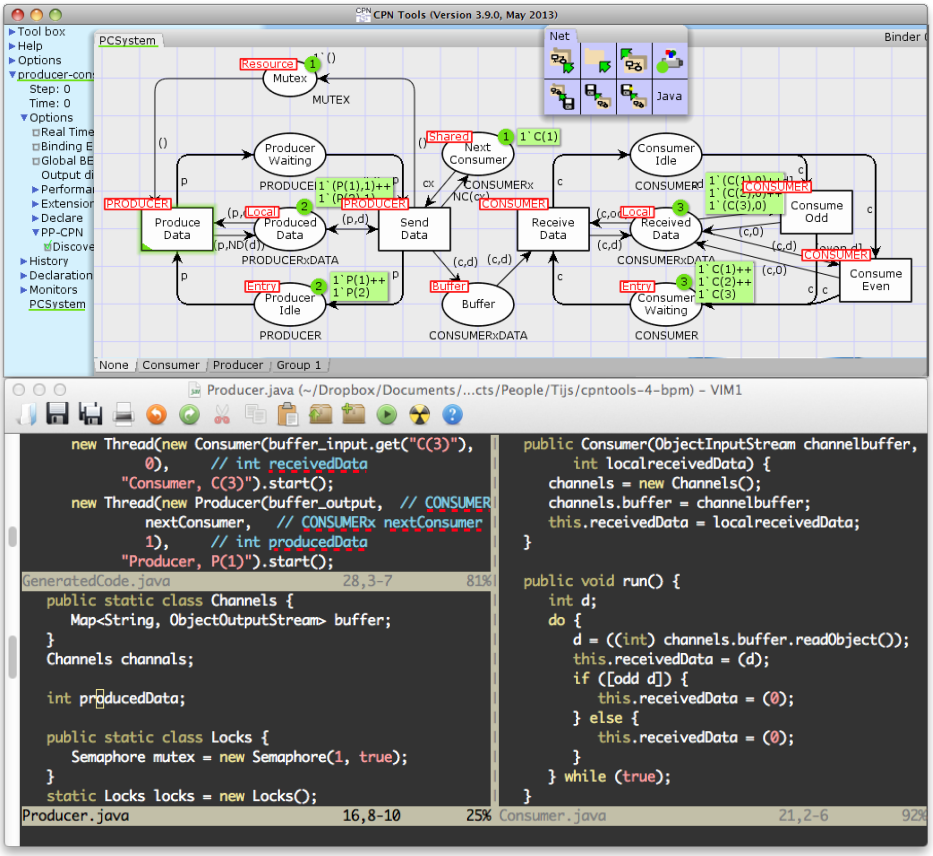

Fig. 3: A colored Petri net model with an explicit process perspective (top) and (some of the) generated Java code from the model (bottom).

producers produce items (integers), store them locally, and transmit them. They use a mutex (a single resource) to prevent race conditions. Initially there are two producers. Consumers receive data from producers, store it locally and dispatch depending on the data.

An advantage of PP-CPNs is that it is possible to generate them automatically from code and to generate running Java code from such models; an example of code generated from the model in Fig. 3 (top) is shown in Fig. 3 (bottom).

Maturity, availability, screencast. CPN Tools is a very mature tool and has been in active use for over 10 years. It enjoyed approximately 5500 downloads in the period May 1, 2012–May 1, 2013. It is used in teaching in several universities, used by companies, and a large number of case studies in several fields are available from `http://cs.au.dk/cpnets/industrial-use/` and on our own homepage we showcase models from industrial case studies at `http://cpntools.org/documentation/examples/`. We are currently conducting case studies using the new declarative constraints, but these are on-going and not yet ready for publication. The implementation of the Declare language is an optimized version of the Declare tool [6].

CPN Tools is open source and available for free for everybody at `http://cpntools.org/`. On this page, we also have a comprehensive getting started guide including screencasts for beginners. In the future, we plan to extend CPN Tools with timed and process-aware versions of Declare.

References

1. van der Aalst, W.M.P., Jørgensen, J.B., Lassen, K.B.: Let's Go All the Way: From Requirements Via Colored Workflow Nets to a BPEL Implementation of a New Bank System. In: Proc. of OTM Conferences (1). LNCS, vol. 3760, pp. 22–39. Springer (2005)
2. Hildebrandt, T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: Post-proc.of PLACES 2010 (2010)
3. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Nested dynamic condition response graphs. In: Proc. of Fundamentals of Software Engineering (FSEN) (April 2011)
4. Kristensen, L.M., Westergaard, M.: Automatic Structure-Based Code Generation from Coloured Petri Nets: A Proof of Concept. In: Proc. of FMICS. pp. 215–230. LNCS, Springer (2010)
5. Westergaard, M.: CPN Tools 4: Multi-formalism and Extensibility. In: Proc. of ATPN. LNCS, vol. 7927, pp. 400–409. Springer (2013)
6. Westergaard, M., Maggi, F.M.: Declare: A Tool Suite for Declarative Workflow Modeling and Enactment. In: Business Process Management Demonstration Track (BPMDemos 2011). CEUR Workshop Proceedings, vol. 820. CEUR-WS.org (2011)
7. Westergaard, M., Slaats, T.: Mixing Paradigms for More Comprehensible Models. In: Proc. of BPM. LNCS, vol. 8094. Springer (2013)