History of Colored Petri Nets

The basic ideas of Petri nets were introduced by Carl Adam Petri in his doctoral thesis [1] published in 1962. This was far ahead of the time where distributed systems were invented and computers started to have parallel processes. At that time programs and processing were considered to be sequential and deterministic. Hence, it was extremely visionary of Carl Adam Petri to predict the importance of being able to understand and characterize the basic concepts of concurrency.

Over the next decade, Petri nets were widely accepted as one of the most well-founded theories to describe important behavioral concepts such as concurrency, synchronization and resource sharing. Petri nets were also used to model and analyze small practical systems. However, the practical use soon revealed a serious shortcoming. Petri nets (in their basic form) is not well-suited to model systems in which data play a crucial part, and since this is the case for most computing systems (and many other kinds of systems) the use of Petri nets for practical modeling were staggering. To remedy this situation many modelers proposed different ad-hoc extensions to Petri nets. This created a large “zoo” of different Petri net languages. Many ad-hoc extensions were not well-defined, and even when they were, they introduced a fundamental problem. Whenever a new ad-hoc extension was introduced, all the basic concepts and analysis methods had to be redefined – to apply for the extended Petri net language (with the ad-hoc extension).

With the invention of the first (text-based) computer tools to support the analysis of Petri net models, the situation became acute. Whenever a new ad-hoc extension was introduced (to handle a modeling problem) all existing computer tools became void, and could only be used after time-consuming (and error-prone) reprogramming. Hence, there was an urgent need to develop a class of Petri nets which were general enough to handle a large variety of different application areas without the need of making ad-hoc extensions.

The first successful step towards a common more powerful class of Petri nets were taken by Hartmann Genrich and Kurt Lautenbach in 1979 with the introduction of Predicate/Transion Nets (PrT nets) [2]. Their work was inspired by earlier work on “Transition Nets with Colored Tokens” (by Kurt Lautenbach and M. Schiffers in 1977) and “Transition Nets with Complex Conditions” (by Robert M. Shapiro in 1979). The basic idea behind PrT nets was to introduce a set of colored tokens which can be distinguished from each other (in contrast to the indistinguishable black tokens in basic Petri nets). In this way it became possible to model different processes in a single subnet. As an example consider the workers in the CPN model of Sect. 2. In the basic Petri net model, it would be necessary to have a subnet for each worker (even though they behave in exactly the same way), but in PrT nets they can be modeled by a single subnet, because we can use tokens with one color to model the state of the first worker, tokens with another color to model the state of the second worker, and so on. This means that we for W workers can have a single WorkerIdle place (which may contain tokens of W different colors) instead of having a separate Idle place for each worker. PrT nets uses arc expressions to define how transitions can occur in different ways (occurrence modes) depending of the colors of the involved input and output tokens. This works in a similar way as described for CPNs in Sect. 2.

The invention of colored distinguishable tokens in PrT nets was a gigantic step forward – but it still had some limitations. PrT nets only had one set of token colors, and all places had to use this set (or Cartesian products of this set with itself). For the example in Sect. 2 this means that the identity of the consumer, the identity of the producers, Yes/no votes and abort/commit decisions all had to be modeled by colors in this single unstructured set of token colors.

The second step towards a more general class of Petri nets was taken by Kurt Jensen in his PhD thesis 1981 with the introduction of the first kind of Colored Petri Nets [3]. This net model allowed the modeler to use a number of different color sets (e.g. one color set for the coordinator, a second for the workers, a third for Yes/no votes and a fourth for abort/commit decisions). This made it possible to represent data values in a more intuitive way instead of having to encode all data into a single shared set. It later turned out to be convenient to define the color sets by means of data types known from programming languages, such as products, records, lists, enumerations, etc. The use of types had three implications: Token colors became structured (and hence much more powerful). Type checking became possible (making it much easier to locate modeling errors). Color sets, arc expressions and guards could be specified by the well-known and powerful syntax and semantics known from programming languages. This gave the modeler a convenient way to handle complex data and specify the often complex interaction between data and system behavior.

A third step forward was taken by Peter Hubert, Kurt Jensen and Robert M. Shapiro in 1990 with the introduction of Hierarchical CPNs [4]. Their work was heavily inspired by the hierarchy concepts in the Structured Analysis and Design Technique (SADT) developed by D.A. Marca and C.L. McGowan. It was Robert Shapiro who got the idea to port the SADT hierarchy concepts to CPN. Hierarchical CPNs allows the modeler to split a large model into a number of interacting and re-usable modules (subsomponents) – in a similar way as used in many programming languages. The basic ideas are illustrated by the modules, substitution transitions and port/socket places in Sect. 2. The introduction of hierarchies in CPNs has several implications: Petri net models of large systems become much more tractable, since they can be split into modules of a reasonable size (instead of working with a single monolithic net which has to be glued to a large wall (or laid out on a football field). The human modeler can concentrate on a few details (in a single module) at a time – instead of being overwhelmed with the full details of the complete model. CPN modules can be seen as black boxes, where modelers, when they desire, can forget about the details within the modules. This makes it possible to work at different abstraction levels – and hence we talk about hierarchical CPNs. Finally there are often system components that are used repeatedly. It would be inefficient to model these components several times. Instead a module can be defined once and used repeatedly. In this way there is only one description to read, and one description to modify, when changes are necessary.

The fourth step was taken by the creation of graphical computer tools to support the modeling and analysis by means of hierarchical CPNs. The Design/CPN was created at Meta Software, Cambridge, Massachusetts, USA starting in 1988 [5]. The main architects behind the tool was Kurt Jensen, Robert Shapiro and Peter Hubert and the implementation was made together with a large international group of people including Jawahar Malhotra (who got the brilliant idea to use the Standard ML language for type definitions and net inscriptions), Ole Bach Andersen (who implemented the graphical interface), Søren Christensen (who implemented the complex algorithms for automatic binding of free variables during simulations) and Hartmann Genrich (who contributed with knowledge and experience from PrT nets). The first version of Design/CPN supported modeling, syntax cheek and interactive simulation. Later versions added timed CPNs (based on ideas by Wil van der Aalst), fast automatic simulations (e.g. for performance analysis) and state space analysis (to investigate behavioral properties).

Steps 2-4 above are closely related. It was during the design of the Design/CPN tool that the need of modules was discovered and it was also in this phase that it became clear that the type concept from programming languages was adequate for type definitions and net inscriptions (instead of using more ad-hoc notations). If we compare the development of CPNs with the development of programming languages, we find many similarities. The first step from black tokens to colored tokens corresponds to the step from bits to simple data types. The second step corresponds to the invention of structured data types and type checking. The third step corresponds to the introduction of concepts like modules, procedures, functions and subroutines. The fourth step corresponds to the construction of compilers. Wihtout these programming languages are of no practical use. Analogously, a Petri net language without tools is useless for practical modeling work.

The extensions of Petri nets and programing languages described above add modeling power. They make it easier for the user to model/program complex systems. They do not add computational power. Anything which can be programming in Java can (in theory) also be programmed in assembler code. It is just much more time-consuming and much more error-prone. Analogously it can be proved (quite easily) that each hierarchical CPN can be unfolded to a (much larger) basic Petri net with exactly the same dynamic behavior. Furthermore, each basic Petri net can be folded into a CPN with a single module consisting of one place and one transition. In practice such a folding is totally uninteresting, because the arc expressions will be extremely complex and totally non-interpretable for a human being. However, the fact that the unfolding and folding exists shows that hierarchical CPNs has the same theoretical properties as basic Petri nets – in particular that CPNs are a solid model for concurrency, synchronization and resource sharing

The introduction of hierarchical CPNs supported by the Design/CPN tool made a dramatic change to the practical used of Petri nets. The new modeling language and its tool support were general and powerful enough to eliminate the need of making ad-hoc extensions. A common platform for practical modeling had been established and this was used by most practical modelers. The use of the platform was supported by a three volume monograph on Colored Petri Nets was published by Kurt Jensen in 1992-1997 [6]. In addition a large number of research papers were published. Some of these describe new analysis methods, while others describes experiences from practical modeling and analysis. References to more more than hundred published describing large-scale modeling and analysis projects can be found in [7]. Many of the projects have been carried out in an industrial environment.

Starting from year 2000 a second generation of tool support, called CPN Tools, was designed and implemented at Aarhus University, Denmark. The main architects behind the new tool were Kurt Jensen, Søren Christensen and Michael Westergaard. The graphical user interface was designed together with Michel Beaudouin-Lafon and Wendy McKay from the international HCI research community. It is based on empirical studies of the use of Design/CPN and much easier and efficient to use. There is also a much faster simulation engine (developed by Torben B. Haagh and Tommy R. Hansen). Many models run one thousand times faster allowing complex automatic simulations to be executed within seconds instead of hours. Many different kinds of state space analysis are supported (designed and implemented by Lars M. Kristensen, Michael Westergaard and Thomas Mailund). There is high-level support for defining and collecting data from simulation-based performance analysis (developed by Lisa M. Wells and Bo Lindstrøm). Finally there is improved support for creating graphical feedback from ongoing simulations (developed by Michael Westergaard).

An international standard for high-level Petri nets was developed and approved in 2004 [8]. The standardization work was headed by Jonathan billington and the standard is heavily based on CPNs (which adhere to the standard).

A new monograph on Colored Petri Nets has been published by Kurt Jensen and Lars M. Kristensen in 2009 [9]. It provides an in-depth, but yet compact introduction to modeling and validation of concurrent systems by means of CPNs. The book introduces the constructs of the CPN modeling language, presents its analysis methods, and provides a comprehensive road map to the practical use of CPNs. Furthermore, the book presents some selected industrial case studies illustrating the practical use of CPN modeling and validation for design, specification, simulation, and verification in a variety of application domains. The book is aimed at use both in university courses and for self-study.

In 2010 CPN Tools had 10.000 licenses in 150 countries. At this time the development and maintenance of the tool set were transferred to the group of Wil van der Aalst at the Technical University of Eindhoven, The Netherlands [10]. New updates with improved functionality are made at a regular basis.

**References**

[1] C.A. Petri: Kommunikation mit Automaten. Institut für Instrumentelle Mathematik, Bonn, 1962. English translation: Technical Report RADC-TR-65-377, Griffiths Air Force Base, New York, Vol. 1, Suppl. 1, 1966.

[2] H.J. Genrich, K. Lautenbach: System Modelling with High level Petri Nets. Theoretical Computer Science 13 (1981), North-Holland, 109–136.

[3] K. Jensen: *Coloured Petri Nets and the Invariant Method.* Theoretical Computer Science 14 (1981), North-Holland, 317–336.

[4] P. Huber, K. Jensen, R.M. Shapiro: *Hierarchies in Coloured Petri Nets.* In: G. Rozen­berg (ed.): Advances in Petri Nets 1990, Lecture Notes in Computer Science Vol. 483, Springer-Verlag 1991, 313–341.

[5] K. Jensen, et. al: Design/CPN. A Reference Manual. Meta Software and Computer Science Department, University of Aarhus, Denmark. On-line version: [www.daimi.au.dk/designCPN/](http://www.daimi.au.dk/designCPN/).

[6] K. Jensen. Coloured Petri Nets; Vol. 1: Basic Concepts, Vol. 2: Analysis Methods, Vol. 3: Practical Use. Monographs in Theoretical Computer Science. Springer, 1992, 1994 and 1997.

[7] Industrial Use of Coloured Petri Nets, [www.cs.au.dk/CPnets/industrialex/](http://www.cs.au.dk/CPnets/industrialex/).

[8] Jonathan Billington: ISO/IEC 15909-1:2004, Software and System Engineering – High-level Petri Nets - Part 1: Concepts, Definitions and Graphical notation.

[9] K. Jensen and L.M. Kristensen: Coloured Petri Nets. Modelling and Validation of Concurrent Systems. Springer 2009. Web-page: [www.cs.au.dk/CPnets/cpnbook/](http://www.cs.au.dk/CPnets/cpnbook/).

[10] CPN Tools. Online documentation: <http://cpntools.org/>