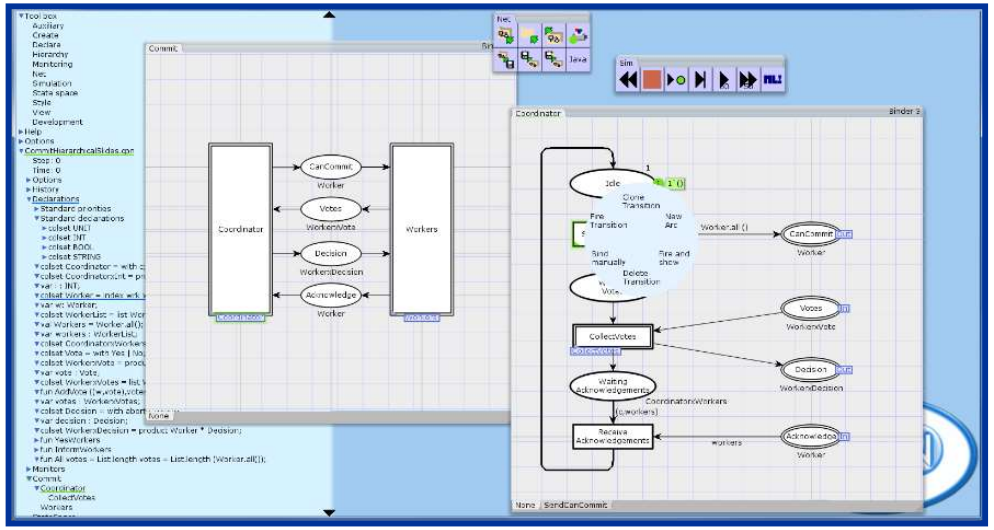
Lecture 1

Overview of Coloured Petri Nets and CPN Tools

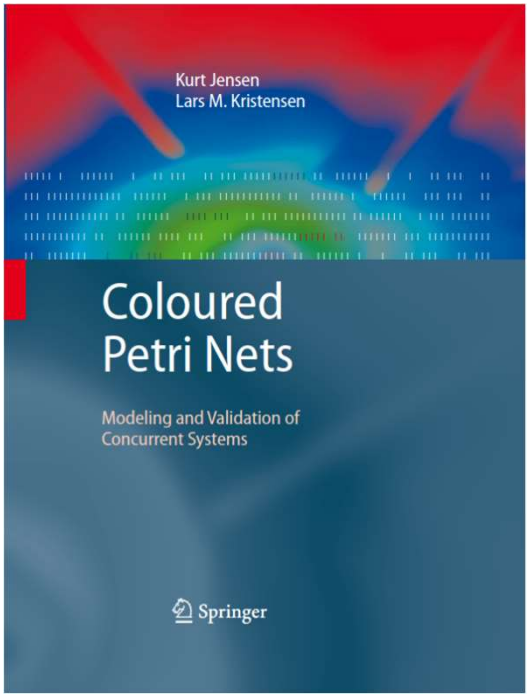


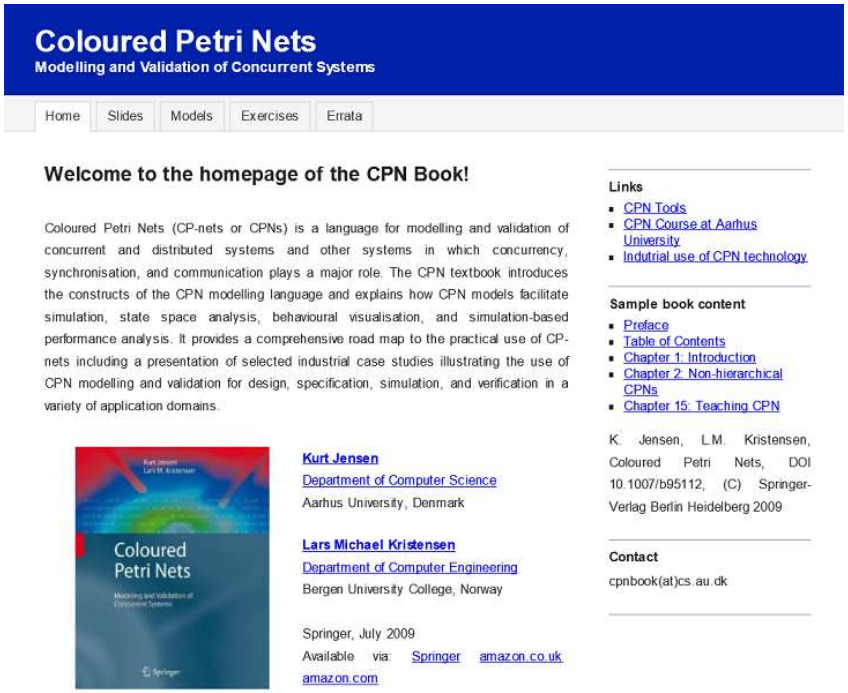
Lars Michael Kristensen
Department of Computing, Mathematics, and Physics
Western Norway University of Applied Sciences
Email: lmkr@hvl.no / WWW: home.hib.no/ansatte/lmkr

My Background

- **2000: PhD from the CPN research centre at Aarhus University (DK) on Coloured Petri Nets and software verification.**
- **2000-2002: Post-doctoral researcher at University of South Australia / Australian Defence and Technology Organisation**
  - **Software tool support for military command and control**
  - **Modelling and implementation of real-time avionics missions systems**
- **2002-2009: Associate professor at Aarhus University**
  - **Capacity planning for web servers with Hewlett-Packard**
  - **Development of protocols for IPv6 with Ericsson Telebit**
- **Since 2009: Professor of computer science and software engineering at Western Norway Univ. of Applied Sciences**
  - **Establishment of a PhD programme in Computer Science: Software Engineering, Sensor Networks and Engineering Computing [http://ict.hvl.no]**
  - **Teaching: programming, network technology and distributed systems, internet-of-things, model-driven software engineering and verification**

CPN Textbook

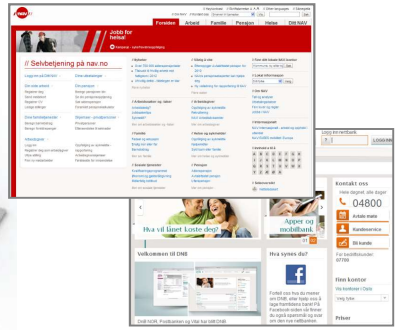


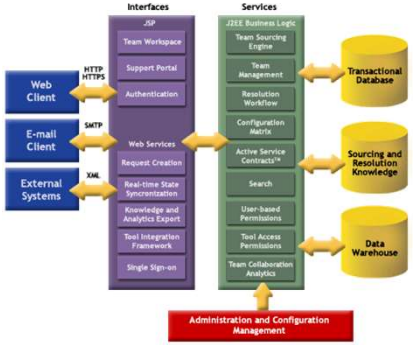


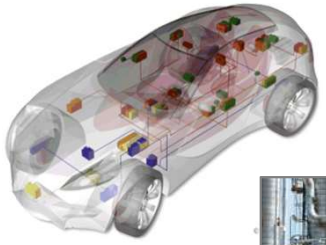
- **K. Jensen and L.M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, 2009.**
- **Book website: www.cpnbook.org**

Concurrent Systems

- **The vast majority of software systems today can be characterised as concurrent systems**
  - Structured as a collection of concurrently executing software components and applications (parallelism)
  - Operation relies on communication, synchronisation, and resource sharing







Internet protocols, cloud, IoT, web-based applications

Multi-core platforms and multi-threaded software

Automation systems and networked control systems

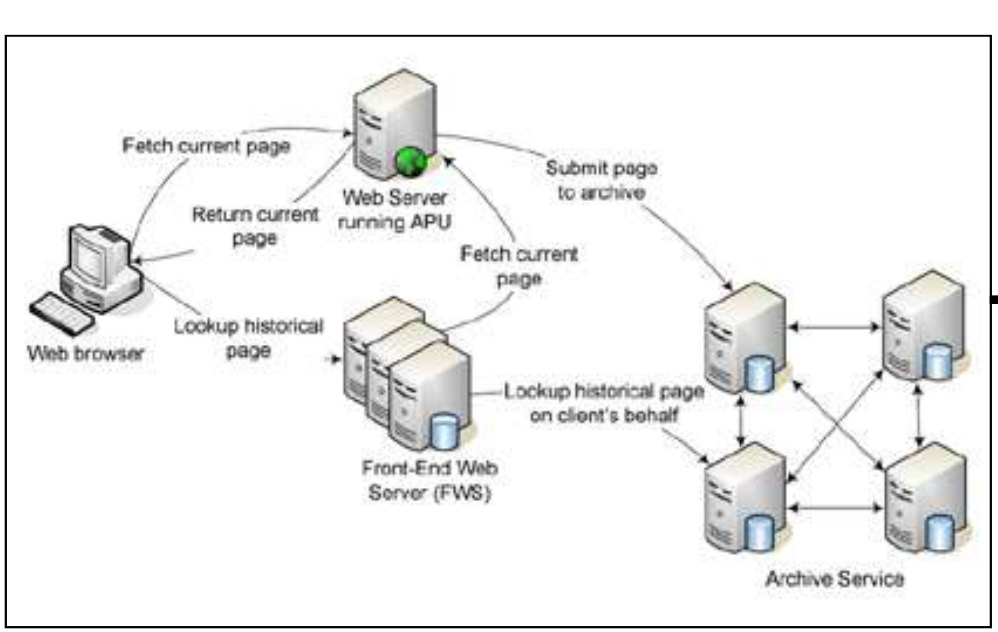
Complex Behaviour

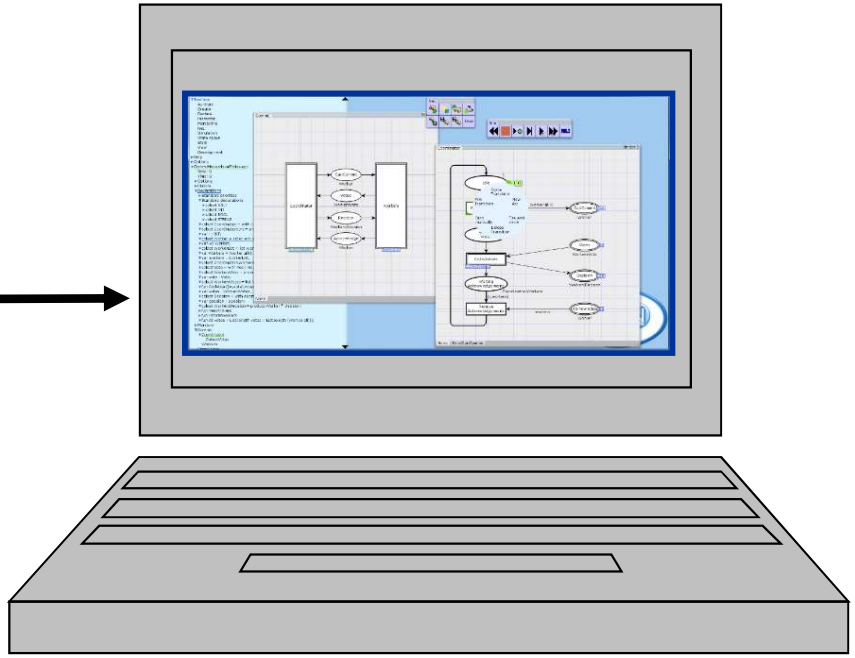
- **The engineering of concurrent systems is challenging due to their complex behaviour**
  - Concurrently executing and independently scheduled software components
  - Non-deterministic and asynchronous behaviour (e.g., timeouts, message loss, external events, …)
  - Almost impossible for software developers to have a complete understanding of the system behaviour
  - Software testing is challenging and reproducing errors is often difficult

- **Methods to support the engineering of reliable concurrent systems are important.**

Modelling

- One way to approach the challenges posed by concurrent systems is construction of models.
- A model is an abstract representation which can be manipulated by a computer software tool



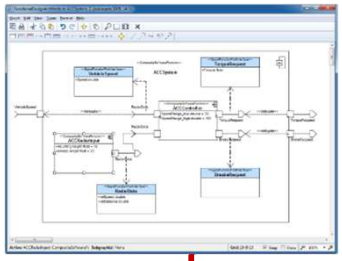


Model-driven Engineering

Problem domain
(vision and requirements)

Validate

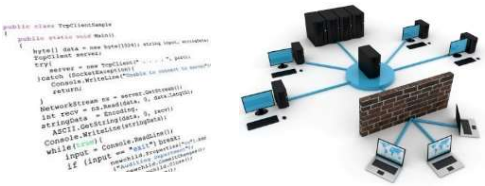
Software engineering
based on models



Model
execution

Code
generation

Technical domain
(implementation)

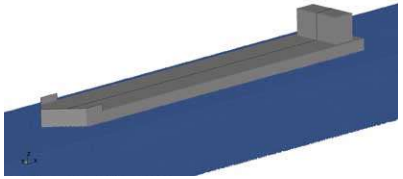


Validation:
Are we building the right software?

Verify

Verification:
Are we building the
software right ?

- **Used in most engineering disciplines**

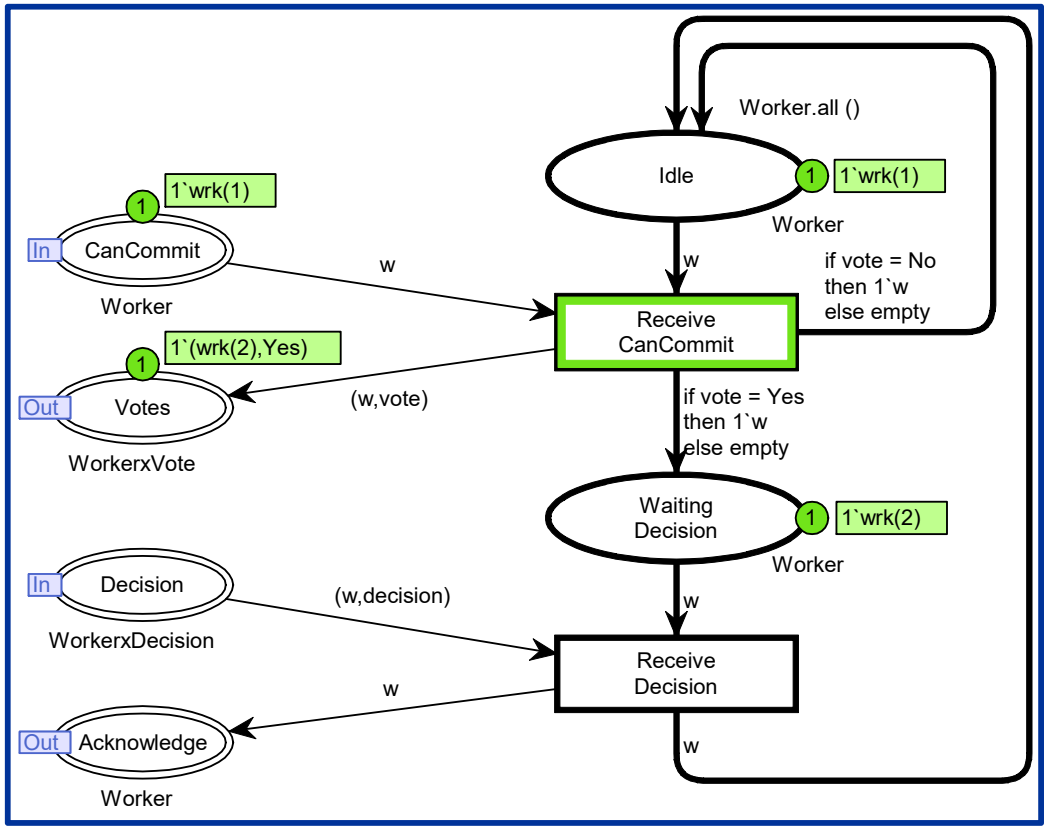


Why Modelling?

- **Benefits of constructing a formal model**
  - Insight into the design and operation of the system
  - Completeness: results in a more complete design
  - Correctness: reveal errors and ambiguities in the design phase
- **Abstraction - use of high-level and domain-specific concepts in software development.**
- **Reliability – Testing and verification and prior to implementation and deployment**
  - Functional properties (e.g., deadlocks, timing requirements,…)
  - Performance properties (e.g., delay, throughout, scalability,…)
- **Productivity - software models can be used as a basis for implementation.**

Coloured Petri Nets (CPNs)

- **General-purpose graphical modelling language for the engineering of concurrent systems.**
- **Combines Petri Nets and a programming language**



Petri Nets

- graphical notation
- concurrency
- communication
- synchronisation
- resource sharing

CPN ML (Standard ML)

- data and data manipulation
- compact modelling
- parameterisable models

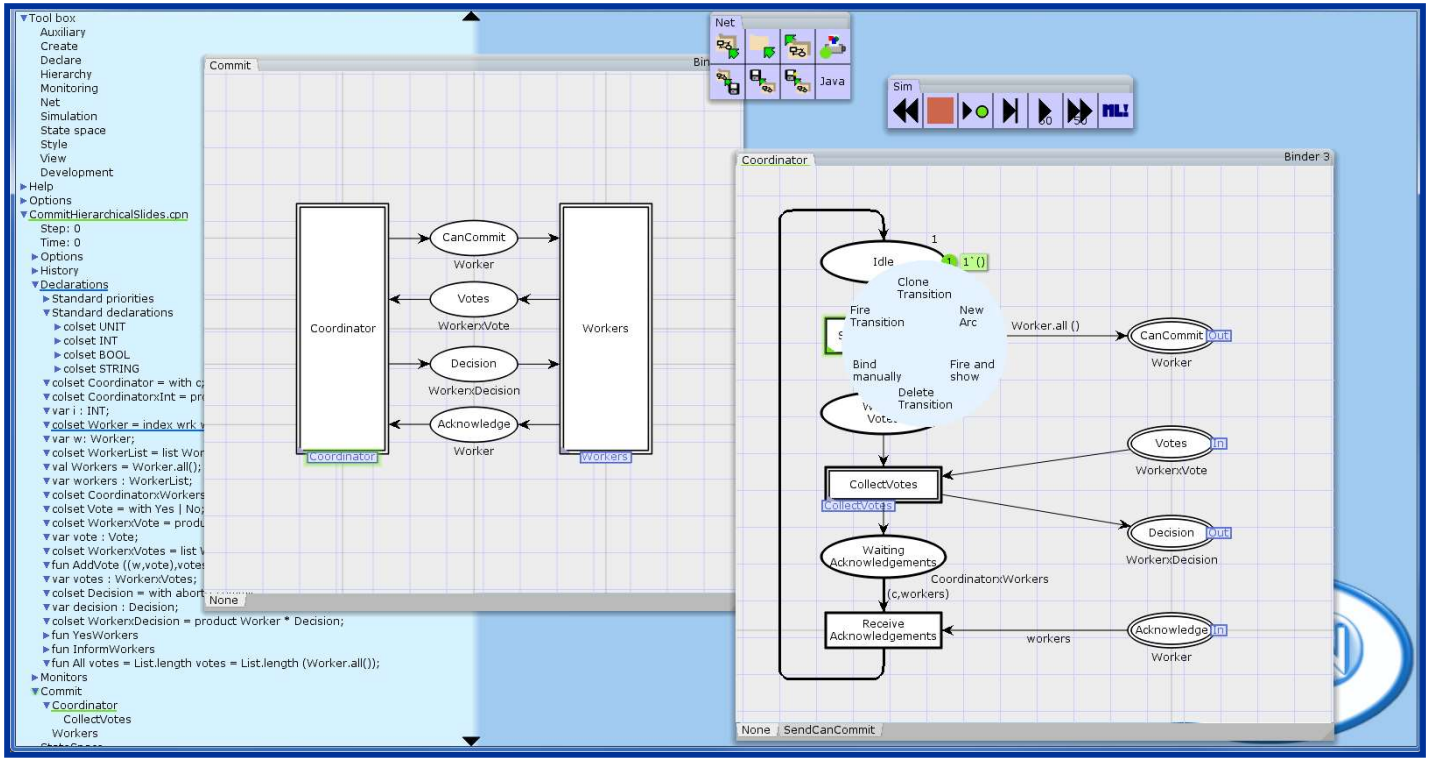
High-level Petri Nets

- **Petri Nets are divided into low-level and high-level Petri Nets:**
  - **Low-level Petri Nets (such as Place/Transitions Nets) are primarily suited as a theoretical model for concurrency, but are also applied for modelling and verification of hardware systems.**
  - **High-level Petri Nets (such as CP-nets and Predicate/Transitions Nets) are aimed at practical use, in particular because they allow for construction of compact and parameterised models.**
- **High-level Petri Nets is an ISO/IEC standard* and the CPN modelling language and supporting tools conform to this standard.**

* https://www.iso.org/standard/38225.html

CPN Tools [www.cpntools.org]

- **Practical use of CPNs is supported by CPN Tools**



- **Editing and syntax check**
- **Interactive- and automatic simulation**
- **Verification based on state space exploration**
- **Simulation-based performance analysis**

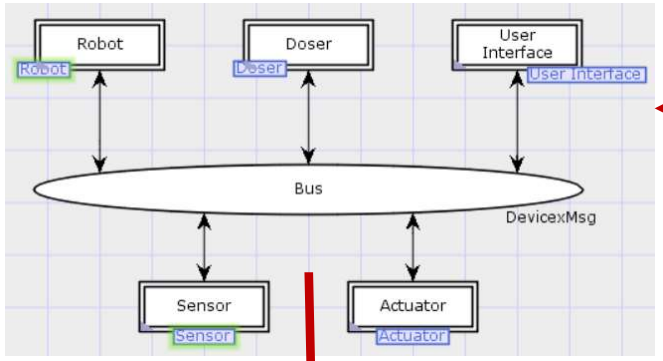
CPN Tools Demo

- **User-interaction with CPN Tools**
  - **Index and workspace**
  - **Binders and tool palettes (drag-and-drop)**
  - **Contextual menus (right click)**

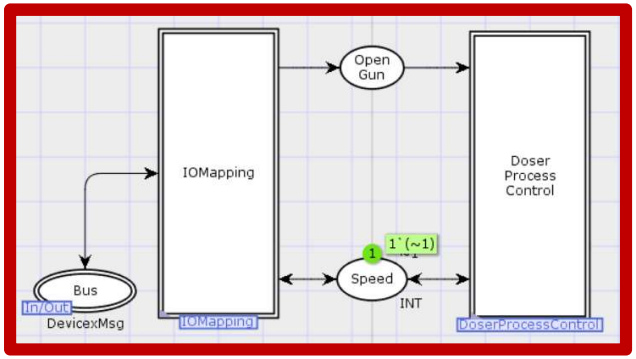
CPN @ Atlas Copco

- **Developing a model-driven software development approach and supporting infrastructure**

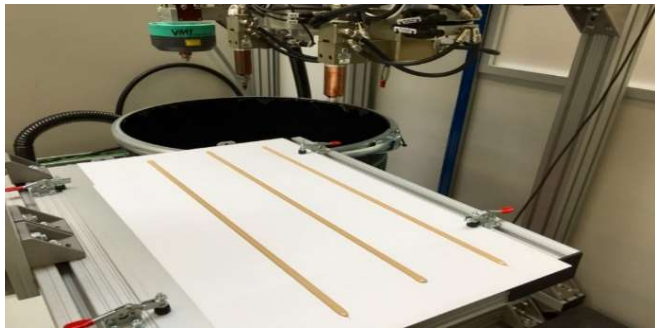
CPN Tools: editing, validation, and verification (design time)

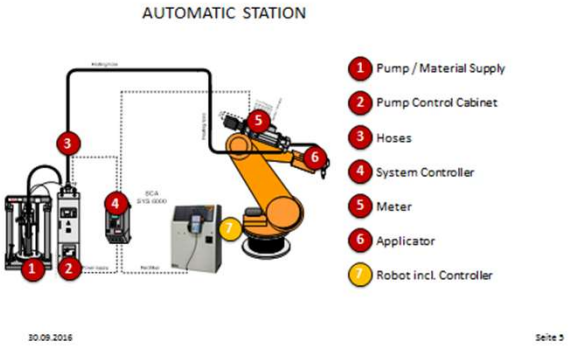


Environment modelling for (non-site) software testing



C++ execution engine for deployment and real-time execution (run-time)

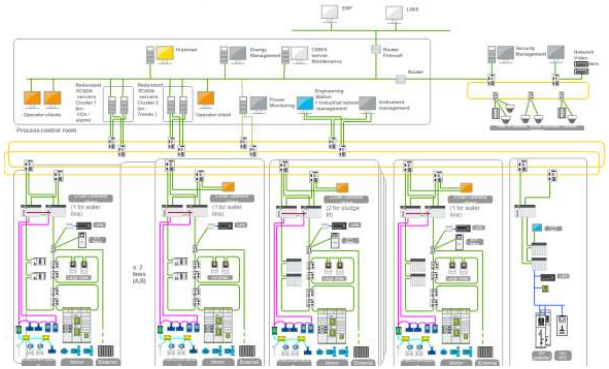




- **The CPN model is directly used as the controller software implementation.**

CPN @ Schneider Electric

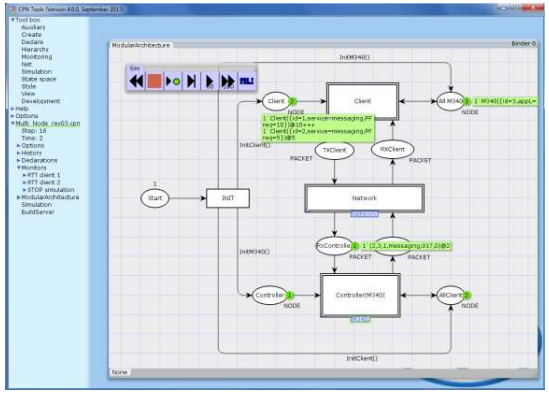
- **Dependability evaluation and capacity planning of large industrial automation architectures**



Dependability analysis software tools

Performance - Reliability
Availability - Safety

Modelling



Automated code generation

Tools for modelling driven engineering

Examples of CPN Tools users

North America

- Boeing
- Hewlett-Packard
- Samsung Information Systems
- National Semiconductor Corp.
- Fujitsu Computer Products
- Honeywell Inc.
- MITRE Corp.,
- Scalable Server Division
- E.I. DuPont de Nemours Inc.
- Federal Reserve System
- Bell Canada
- Nortel Technologies, Canada

Europe

- Alcatel Austria
- Siemens Austria
- Bang & Olufsen, Denmark
- Nokia, Finland
- Alcatel Business Systems, France
- Peugeot-Citroën, France
- Dornier Satellitensysteme, Germany
- SAP AG, Germany
- Volkswagen AG, Germany
- Alcatel Telecom, Netherlands
- Rank Xerox, Netherlands
- Sydkraft Konsult, Sweden
- Central Bank of Russia
- Siemens Switzerland
- Goldman Sachs, UK

Asia

- Mitsubishi Electric Corp., Japan
- Toshiba Corp., Japan
- SHARP Corp., Japan
- Nippon Steel Corp., Japan
- Hongkong Telecom Interactive Multimedia System

http://cs.au.dk/cpnets/industrial-use/

CPN models are formal

- **The CPN modelling language has a mathematical definition of both its syntax and semantics.**
- **The formal representation is important**
  - Provides the foundation for the definition of the different behavioural properties and the analysis methods
  - Would have been impossible to develop a sound and powerful CPN language without it
- **Formal models can be used to verify system properties such as**
  - Proving that certain desired properties are fulfilled
  - Proving that certain undesired properties are guaranteed to be avoided

Formal Definition

Definition 4.2. A **non-hierarchical Coloured Petri Net** is a nine-tuple $CPN = (P, T, A, \Sigma, V, C, G, E, I)$, where:

1. P is a finite set of **places**.
2. T is a finite set o
3. $A \subseteq P \times T \cup T \times$
4. Σ is a finite set o
5. V is a finite set o
6. $C : P \rightarrow \Sigma$ is a c
7. $G : T \rightarrow EXPR_V$ that $Type[G(t)] =$
8. $E : A \rightarrow EXPR_V$ each arc a such t arc a.
9. $I : P \rightarrow EXPR_\emptyset$ i sion to each plac

Definition 4.5. A step $Y \in BE_{MS}$ is **enabled** in a marking M if and only if the following two properties are satisfied:

1. $\forall (t,b) \in Y : G(t)\langle b \rangle$.
2. $\forall p \in P : \quad \overset{++}{\underset{(t,b)\in Y}{\underset{MS}{\sum}}} E(p,t)\langle b \rangle \ll= M(p)$.

When Y is enabled in M, it may **occur**, leading to the marking M' defined by:

3. $\forall p \in P : M'(p) = (M(p) -- \overset{++}{\underset{(t,b)\in Y}{\underset{MS}{\sum}}} E(p,t)\langle b \rangle) ++ \overset{++}{\underset{(t,b)\in Y}{\underset{MS}{\sum}}} E(t,p)\langle b \rangle$.

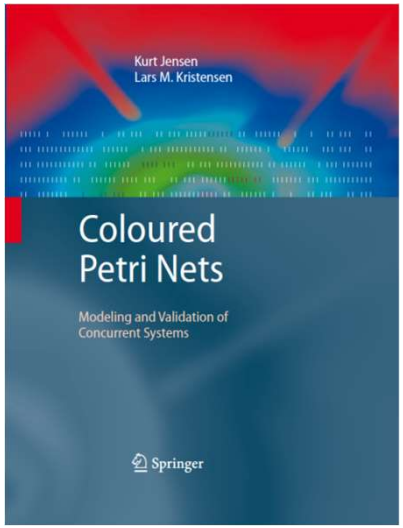
□

- **Learning to use CPNs is similar to learning a programming language (no mathematics :-)**

Outline

- **Module I: Modelling and CPN Tools [today]**
  - Motivation and overview of Coloured Petri Nets
  - The syntax and semantics of the basic constructs of the Coloured Petri Nets (CPNs) modelling language
  - Modules for hierarchical structuring of large CPN models
  - Application of CPN Tools for construction and simulation of medium-sized CPN models
- **Module II: Verification and Applications [tomorrow]**
  - The basic concepts of state spaces and how they are computed
  - Introduce standard behavioural properties of CPNs
  - Checking standard behavioural properties using state spaces
  - Application of state spaces for verification of medium-sized CPN models
  - A larger example on the industrial use of CPNs and CPN Tools

Resources



K. Jensen and L.M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, 2009.

www.cpnbook.org

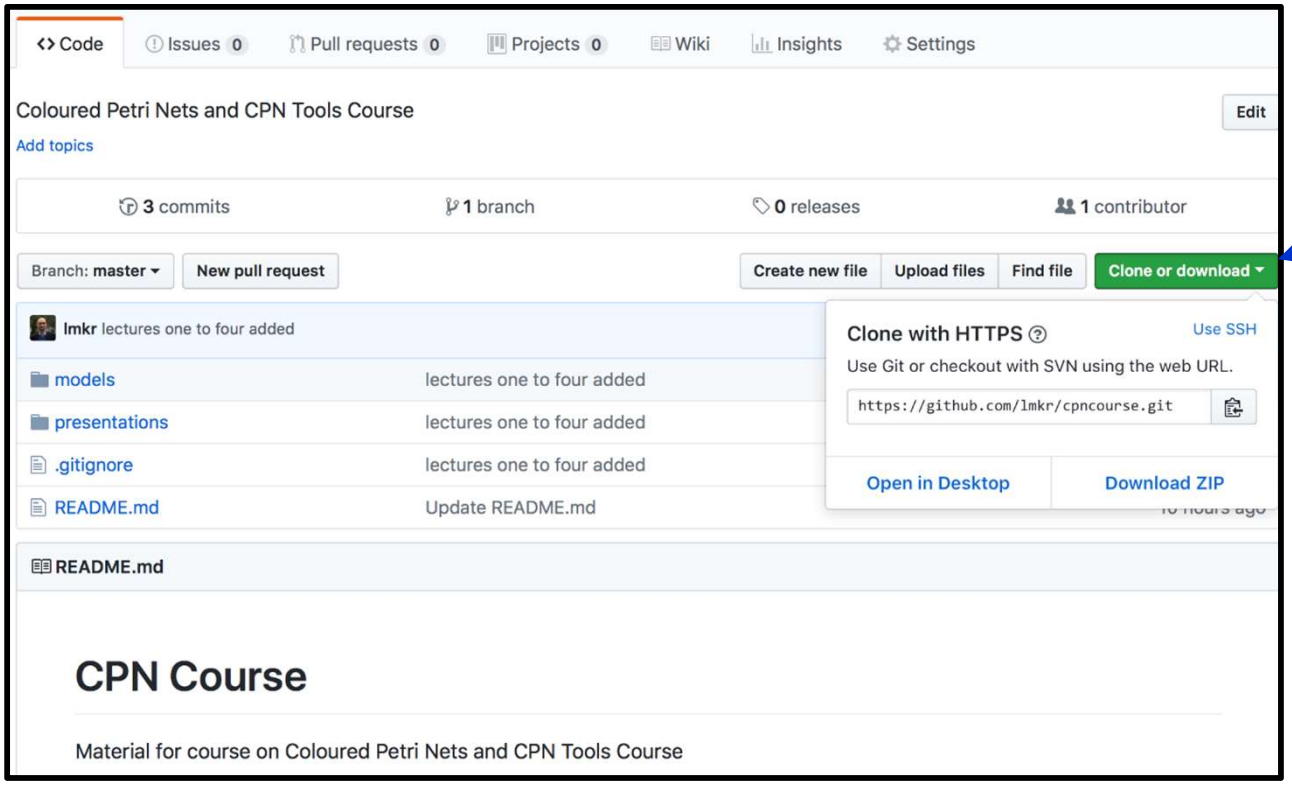
Practical use of CPN Tools is extensively documented at www.cpntools.org


Implementing Coloured Petri Nets
Using a Functional Programming Language

- **Research papers on Coloured Petri Nets**
  - K. Jensen and L.M. Kristensen. Coloured Petri Nets: A Graphical Language for Modelling and Validation of Concurrent Systems. Communications of the ACM, Vol. 58, No. 6, pp. 61-70, 2015.
  - K. Jensen, L.M. Kristensen, L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. Intl. Journal on Software Tools for Technology Transfer, Vol. 9, pp. 213-254, Springer, 2007.
  - L.M. Kristensen and S. Christensen: Implementing Coloured Petri Nets using a Functional Programming Language. In Higher-order and Symbolic Computation, Vol. 17, pp. 207-243, 2004.

Material

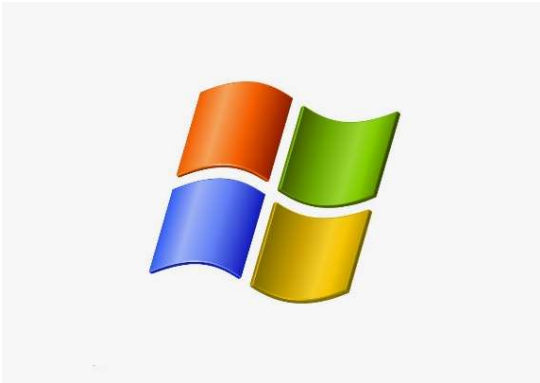
- **Models and presentations are available via the github repository at** https://github.com/lmkr/cpncourse



Clone the git-repository or download as a zip-file

CPN Tools Installation

- **CPN Tools can be downloaded and installed via www.cpntools.org**



Running on Mac OS / Linux via a virtual machine or emulator.

- **Some installations of windows required the application to be run as administrator.**