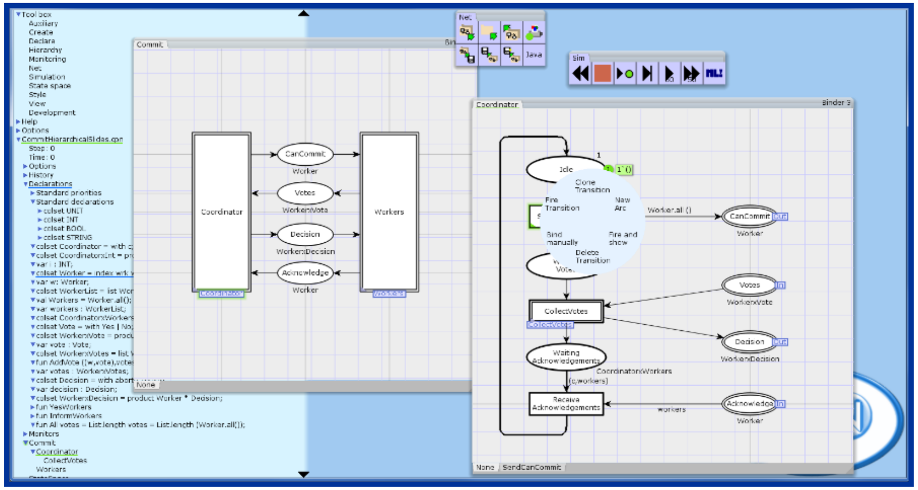
Lecture 1

Introduction to Coloured Petri Nets and CPN Tools

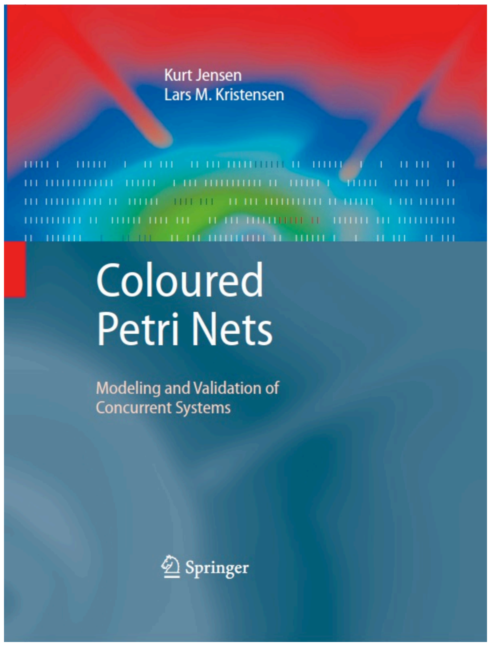


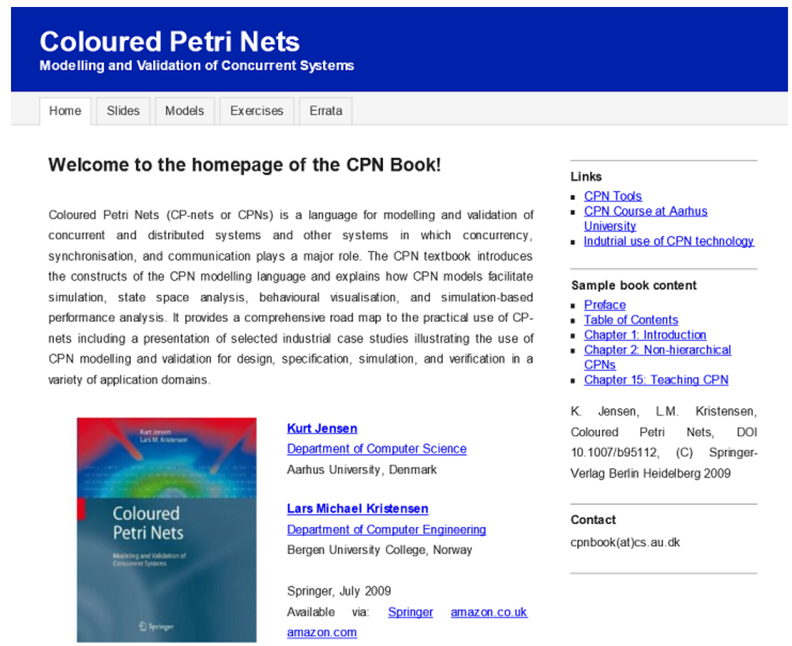
Lars Michael Kristensen
Department of Computing, Mathematics, and Physics
Western Norway University of Applied Sciences
Email: lmkr@hvl.no / WWW: home.hib.no/ansatte/lmkr

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**
  - **Real-time avionics missions systems**
- **2002-2009: Associate professor at Aarhus University**
  - **Capacity planning of web servers with Hewlett-Packard**
  - **Development of protocols for IPv6 with Ericsson Telebit**
- **Since 2009: Professor in 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

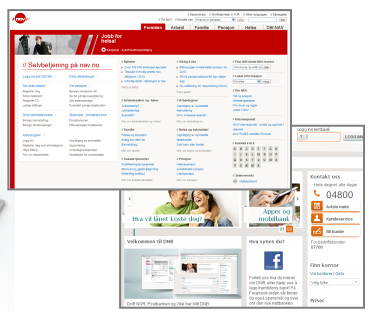




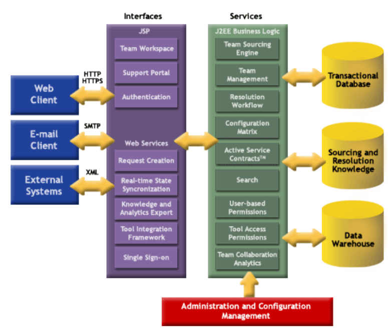
- **K. Jensen and L.M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, 2009.**
- **Book website: www.cpnbook.org**

Concurrent Systems

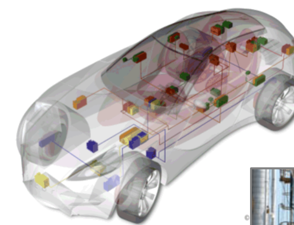
- **The vast majority of software systems today can be characterised as concurrent systems:**
  - Structured as a collection of concurrently executing software components and applications (parallelism).
  - Operation relies on communication, synchronisation, and resource sharing.

Internet protocols, cloud, IoT, web-based applications



Multi-core platforms and multi-threaded software




Automation systems and networked control systems

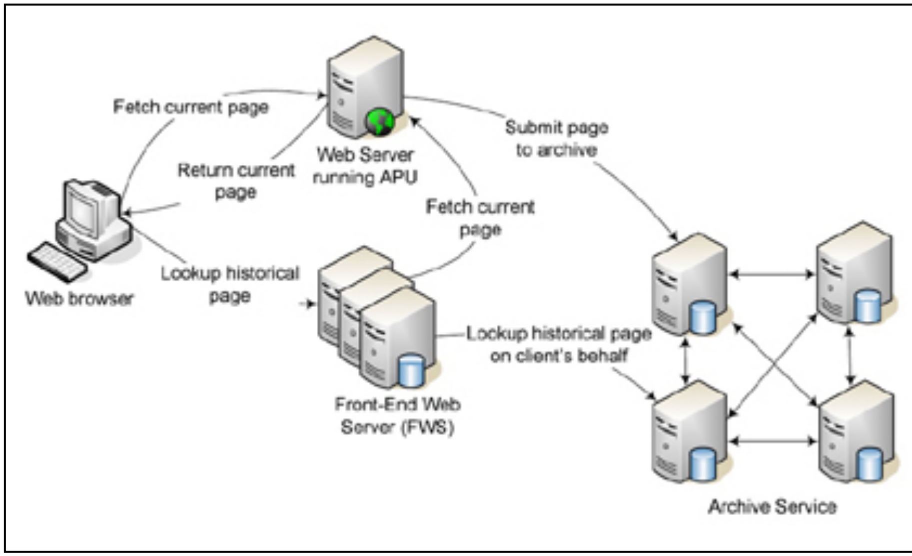
Concurrent Systems

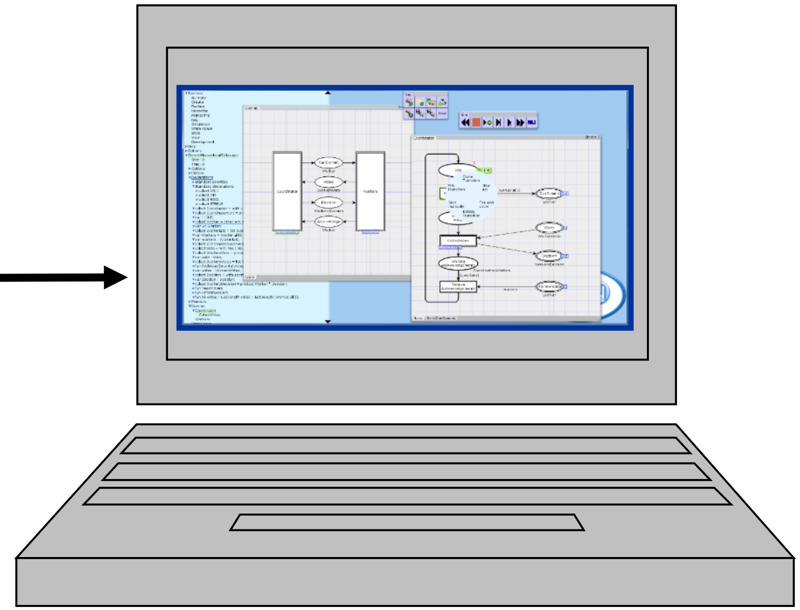
- **Most software development projects are concerned with concurrent software systems.**
- **The engineering of concurrent systems is challenging due to their complex behaviour:**
  - Concurrently executing and independently scheduled software components.
  - Non-deterministic and asynchronous behaviour (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:





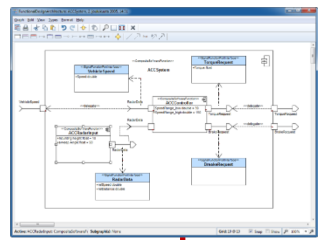
- Explore the design and undertake testing of the system prior to deployment and implementation.

Model-driven Engineering

Problem domain
(vision and requirements)

Validate

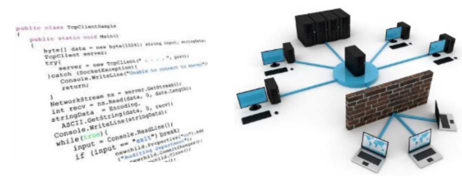
Software engineering
based on models



Code
generation

Verify

Technical domain
(implementation)



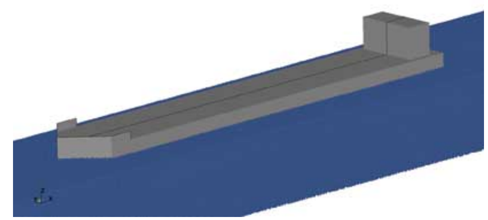
Validation:
Are we building the right software?

Verification:
Are we building the software right ?

- **Advantages:**
  - **Abstraction:** Use of high-level and domain-specific languages in the development of systems.
  - **Productivity:** Automated code generation for a wide range of platforms based on the same model.
  - **Reliability:** Verification and testing prior to implementation and deployment.

Modelling ...

- **Used in most engineering disciplines:**
  - **When engineers construct a bridge.**
  - **When architects design a building.**



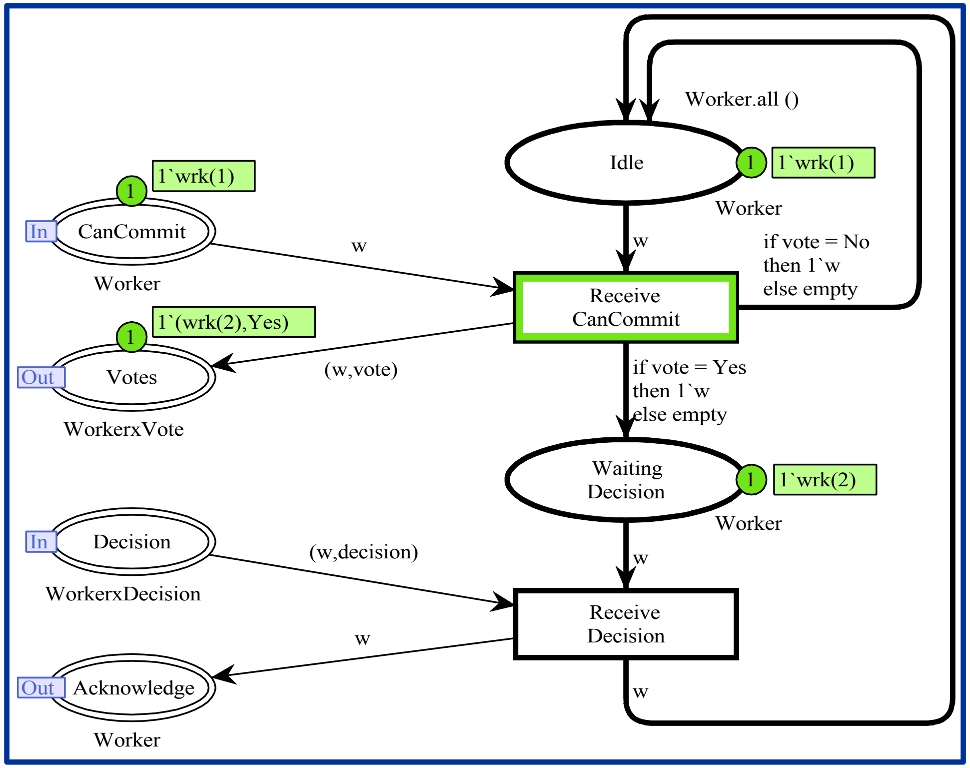
Western Norway University of Applied Sciences

Why Modelling?

- **Supports abstraction and the use of domain-specific concepts in software development.**
- **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.
- **Properties can be validated and verified prior to implementation and deployment:**
  - Functional properties (e.g., deadlocks, timing requirements,…).
  - Performance properties (e.g., delay, throughout, scalability,…).
- **The software models can be used as a basis for automated generation of implementations.**

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

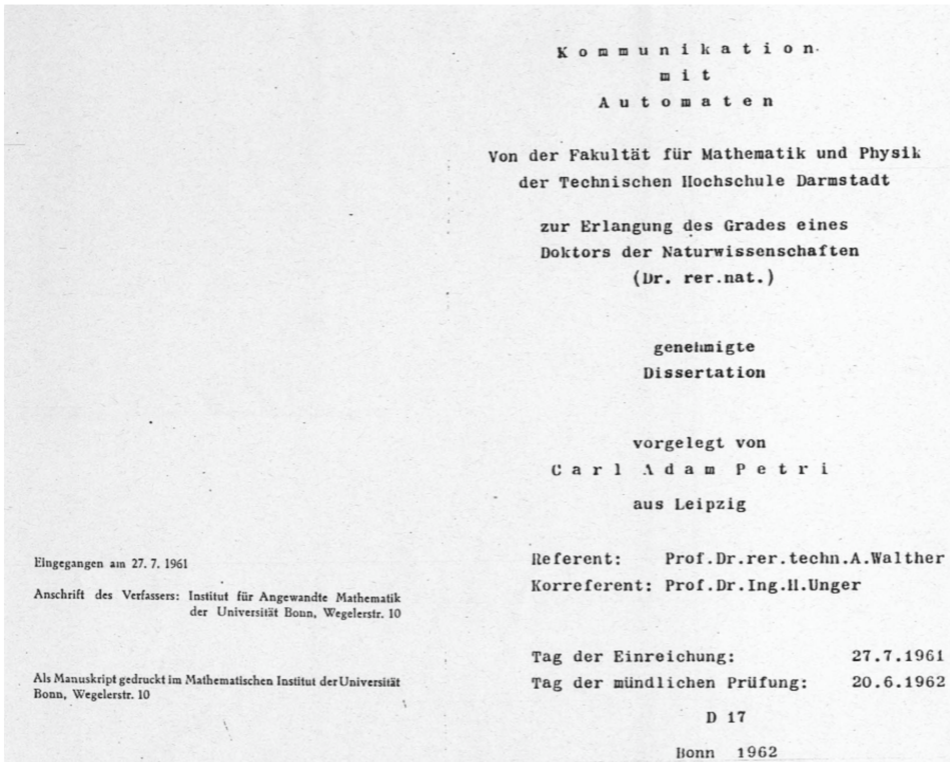
CPN ML (Standard ML)

- data and data manipulation
- compact modelling
- parameterisable models

High-Level Petri Net

Petri Nets

- **Originates from the PhD dissertation of Carl Adam Petri (1926 – 2010):**



Kommunikation mit Automaten

Von der Fakultät für Mathematik und Physik der Technischen Hochschule Darmstadt

zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer.nat.)

genehmigte Dissertation

vorgelegt von Carl Adam Petri aus Leipzig

Eingegangen am 27. 7. 1961

Anschrift des Verfassers: Institut für Angewandte Mathematik der Universität Bonn, Wegelerstr. 10

Als Manuskript gedruckt im Mathematischen Institut der Universität Bonn, Wegelerstr. 10

Referent: Prof.Dr.rer.techn.A.Walther
Korreferent: Prof.Dr.Ing.H.Unger

Tag der Einreichung: 27.7.1961
Tag der mündlichen Prüfung: 20.6.1962

D 17

Bonn 1962

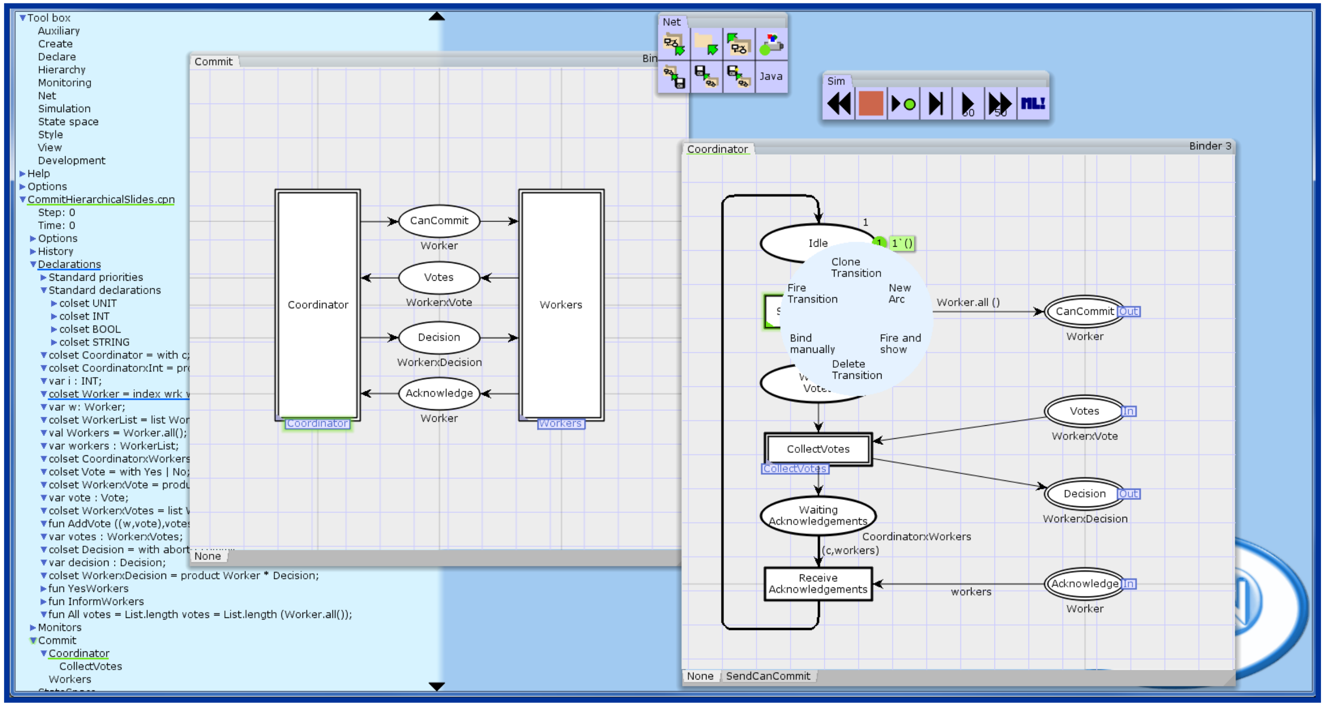
High-level Petri Nets

- 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.**

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

Asia

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

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

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

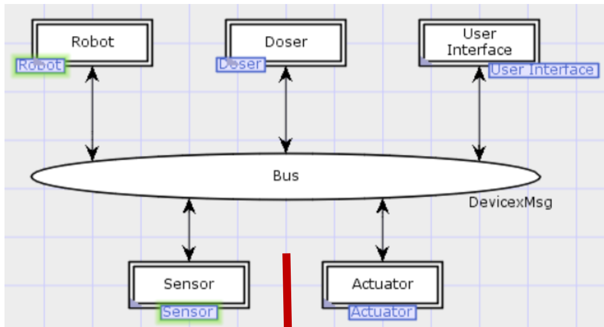
CPN Tools Demo

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

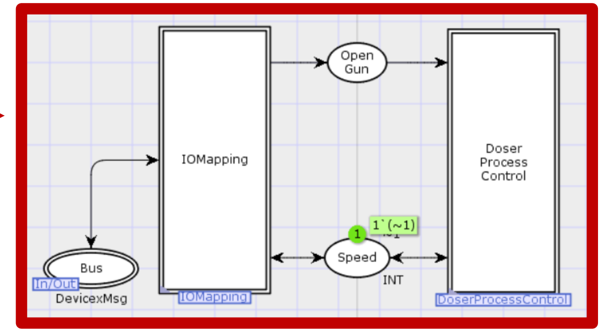
CPNs @ Atlas Copco

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

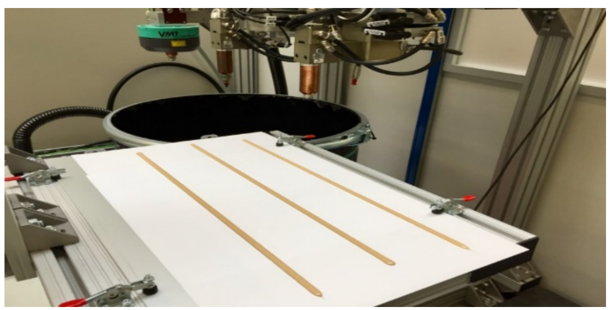
CPN Tools: editing, validation, and verification (design time)

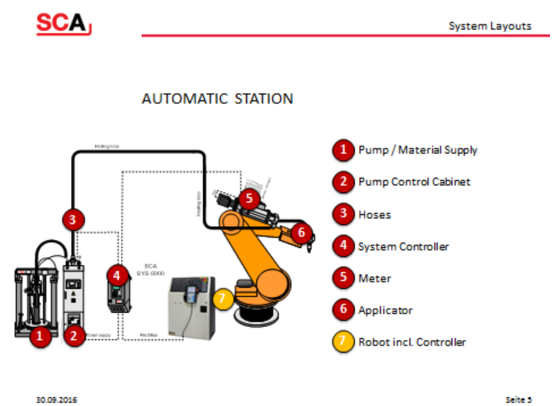


Environment modelling for (non-site) software testing



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

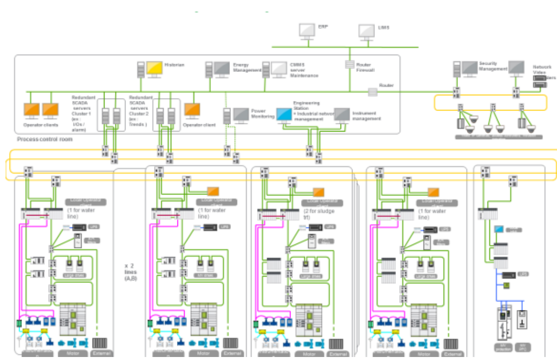




- **The CPN model is directly used an the controller software implementation.**

CPN @ Schneider Electric

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

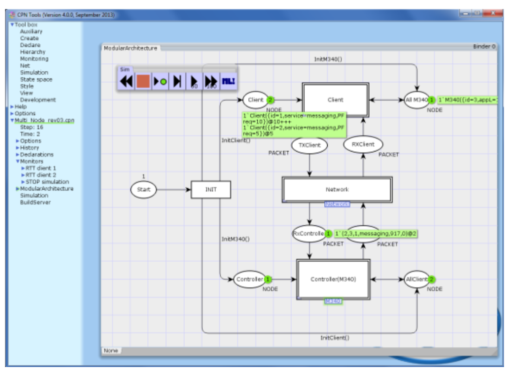


Dependability analysis software tools

Performance - Reliability
Availability - Safety

Modelling

Automated code generation



Tools for modelling driven engineering

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. $C : P \to \Sigma$ is a c
7. $G : T \to EXPR_V$
   that $Type[G(t)] =$
8. $E : A \to EXPR_V$
   each arc a such t
   arc a.
9. $I : P \to EXPR_\emptyset$ 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. $\forall p \in P : \quad {}_{MS}\overset{++}{\sum}_{(t,b)\in Y} E(p,t)\langle b \rangle \ll\!= M(p)$.

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

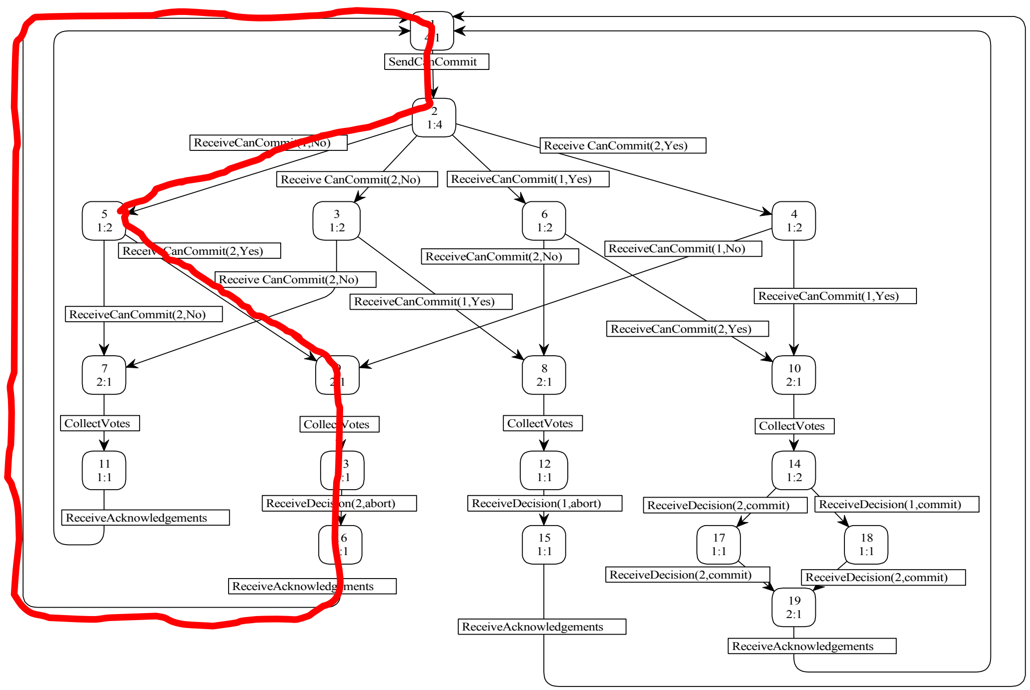
3. $\forall p \in P : M'(p) = (M(p) \; -\!- \; {}_{MS}\overset{++}{\sum}_{(t,b)\in Y} E(p,t)\langle b \rangle) \; +\!+ \; {}_{MS}\overset{++}{\sum}_{(t,b)\in Y} E(t,p)\langle b \rangle$.

$\square$

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

Verification and Model Checking

- **Formal verification of CPN models can be conducted using explicit state space exploration:**

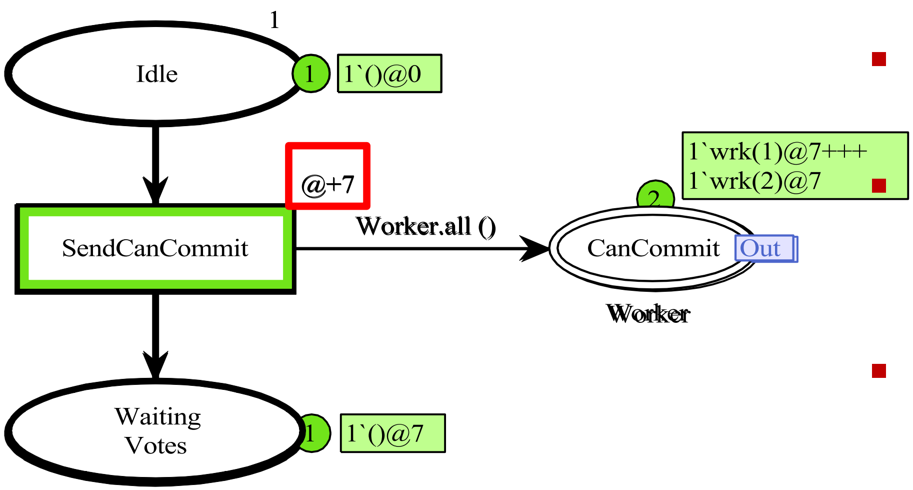


- **A state space represents all possible executions of the CPN model.**
- **Standard behavioural properties can be investigated using the state space report.**
- **Model-specific properties can be verified using queries and temporal logic model checking.**

- **Several advanced techniques available to alleviate the inherent state explosion problem.**

Performance Analysis

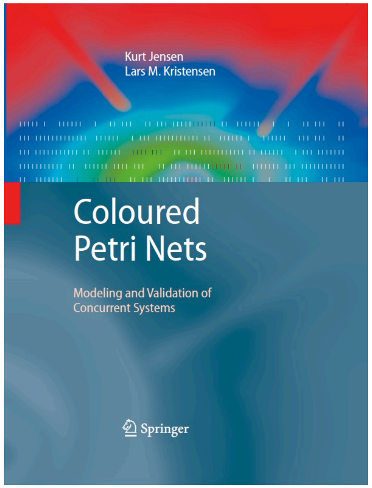
- CPNs include a concept of time that can be used to model the timed taken by activities:



- A global clock representing the current model time.
- Tokens carry time stamps describing the earliest possible model time at which they can be removed.
- Time inscriptions on transitions and arcs are used to give time stamps to the tokens produced on output places.

- Random distribution functions can be used in arc expressions (delays, packet loss, ...).
- Data collection monitors and batch simulations can be used to compute performance figures.

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

- **Shorter 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.

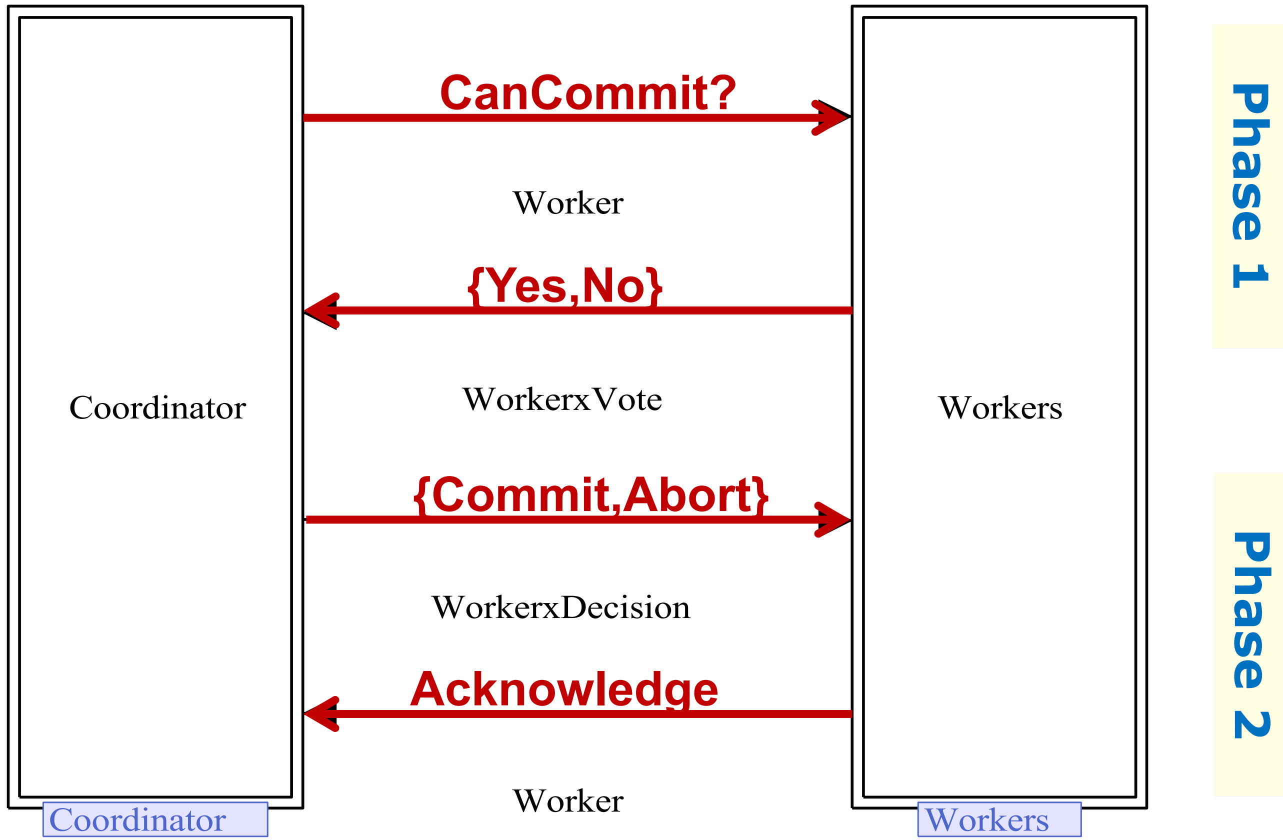
Outline

- **Part 1: Overview and Basic Concepts of Petri Nets**
  - Overview of Petri Nets and Coloured Petri Nets
  - Modelling with Place/Transition Nets
- **Part 2: Coloured Petri Nets**
  - Extending Petri nets with a functional programming language
  - Structuring large CPN models into modules
- **Part 4: Hands-on session with CPN Tools**
  - Creating and simulating a Place/Transition net model
  - Building and simulating a CPN model
- **Part 4: Pump Controller and Doser CPN models**
  - Demonstration of the CPN models
  - Q&A and discussions

Do not hesitate to ask questions along the way!

Two-phase Commit Transaction Protocol

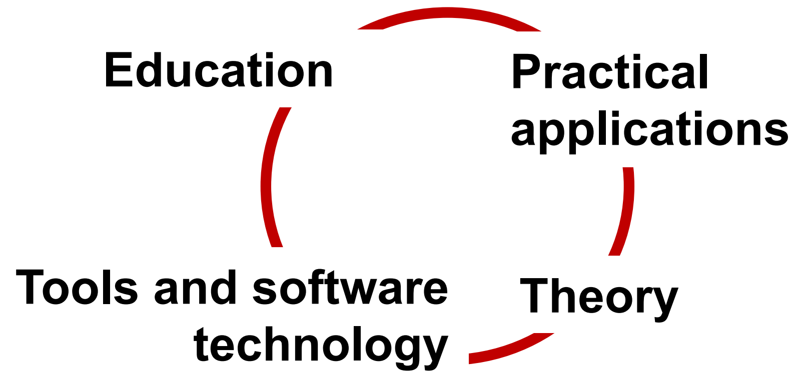
- A concurrent system consisting of a coordinator process and a number of worker processes:



Western Norway University of Applied Sciences

Perspectives on CPNs

- **Modelling language combining Petri Nets with a programming language.**
- **The development has been driven by an application-oriented research agenda**



- **Key characteristics:**
  - **Few but still powerful and expressive modelling constructs.**
  - **Implicit concurrency inherited from Petri nets: everything is concurrent unless explicit synchronised.**
  - **Verification and performance analysis supported by the same modelling language.**