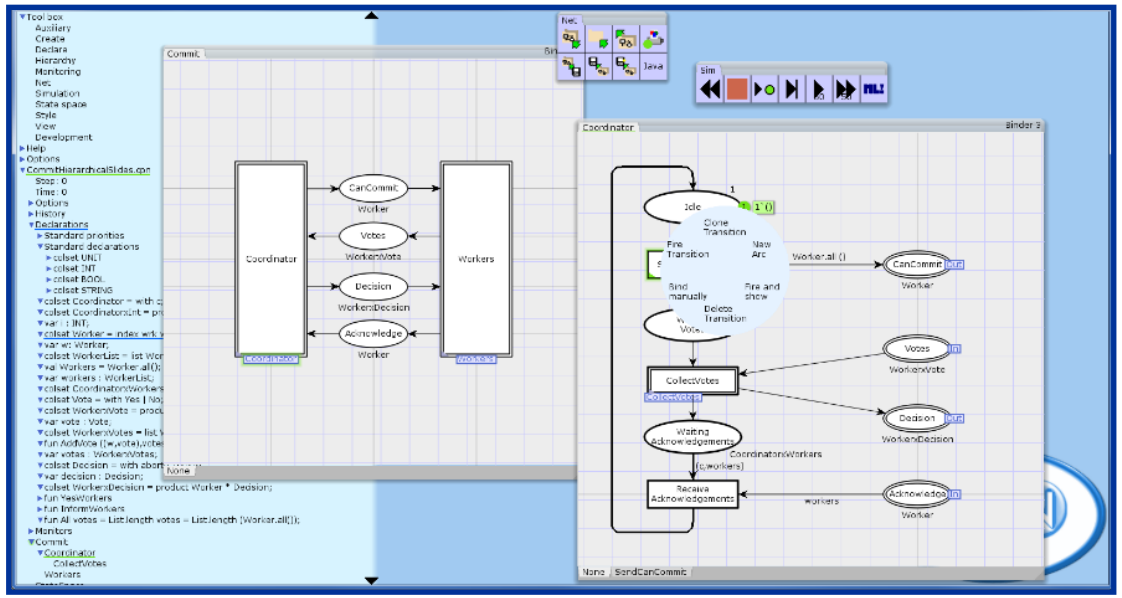
Lecture 1

Overview of Coloured Petri Nets and CPN Tools

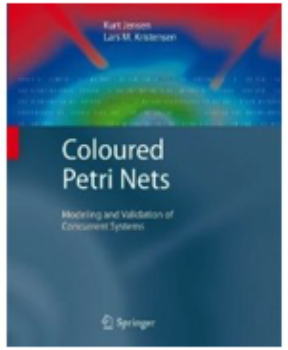


Lars Michael Kristensen
Department of Computer Science, Electrical Engineering, and Mathematical Scienes
Western Norway University of Applied Sciences
Email: lmkr@hvl.no / WWW: home.hib.no/ansatte/lmkr

CPN textbook

Coloured Petri Nets: Modelling and Validation of Concurrent Systems

Welcome to the homepage of the CPN Book



Springer, July 2009 - available via: Springer amazon.co.uk amazon.com

Authors

Kurt Jensen	Lars Michael Kristensen
Department of Computer Science	Department of Computing
Aarhus University, Denmark	Western Norway University of Applied Sciences

Introduction

Coloured Petri Nets (CP-nets or CPNs) is a language for modelling and validation of concurrent and distributed systems and other systems in which concurrency, synchronisation, and communication plays a major role. The CPN textbook

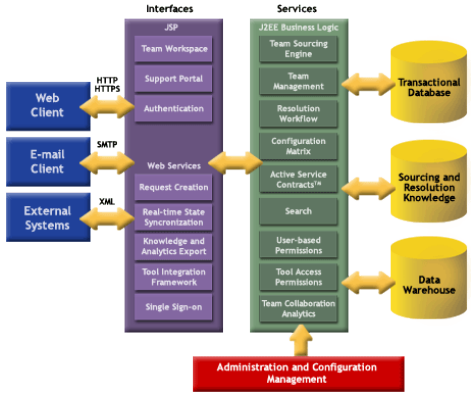
- **K. Jensen and L.M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, 2009.**
- **Book website: www.cpnbook.org**

Motivation - Application domain of Coloured Petri Nets

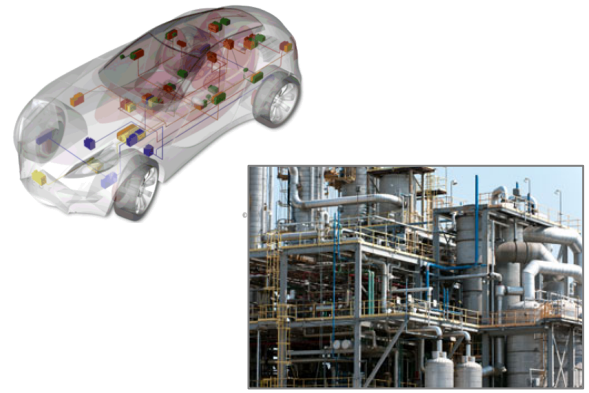
Concurrent systems

- **The vast majority of 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

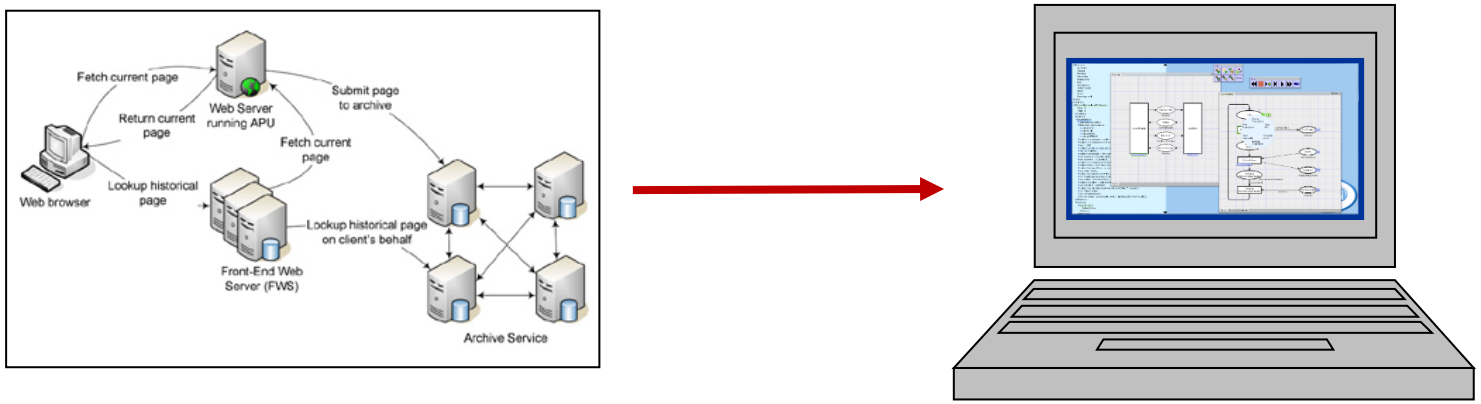
Complex behaviour

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

- **Methods to support the engineering of reliable concurrent systems are highly relevant**

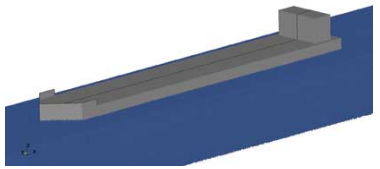
Modelling

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



- Modelling is used in most engineering disciplines



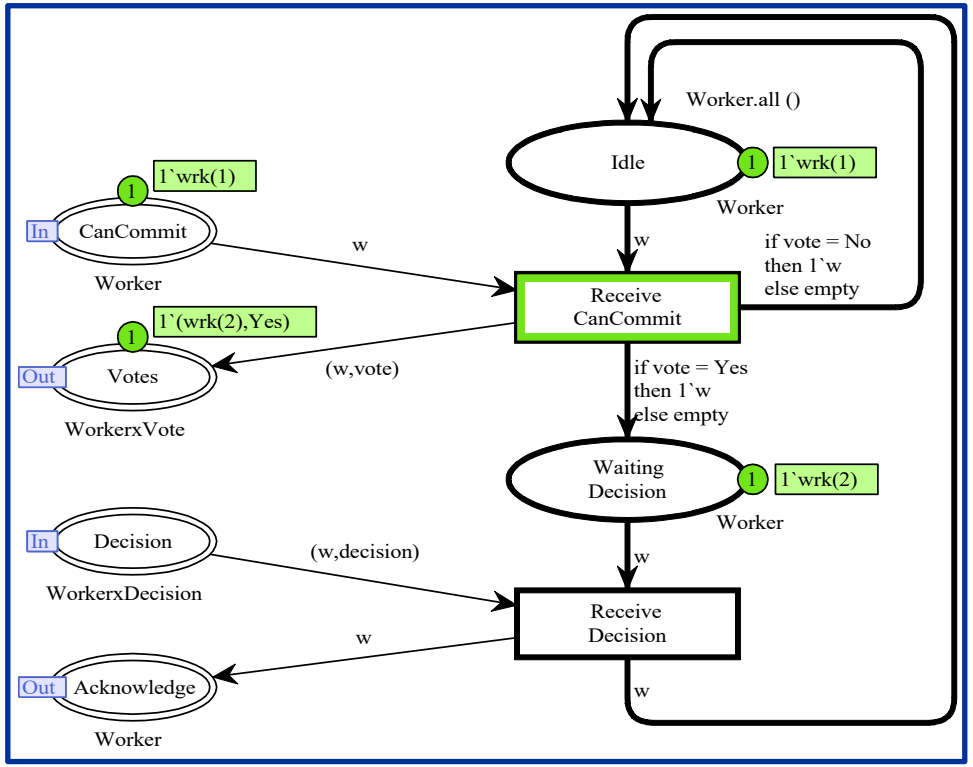



Why modelling?

- **Benefits of constructing executable models**
  - 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 and communication – validation using high-level and domain-specific concepts in 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 - models can be used (directly or indirectly) 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

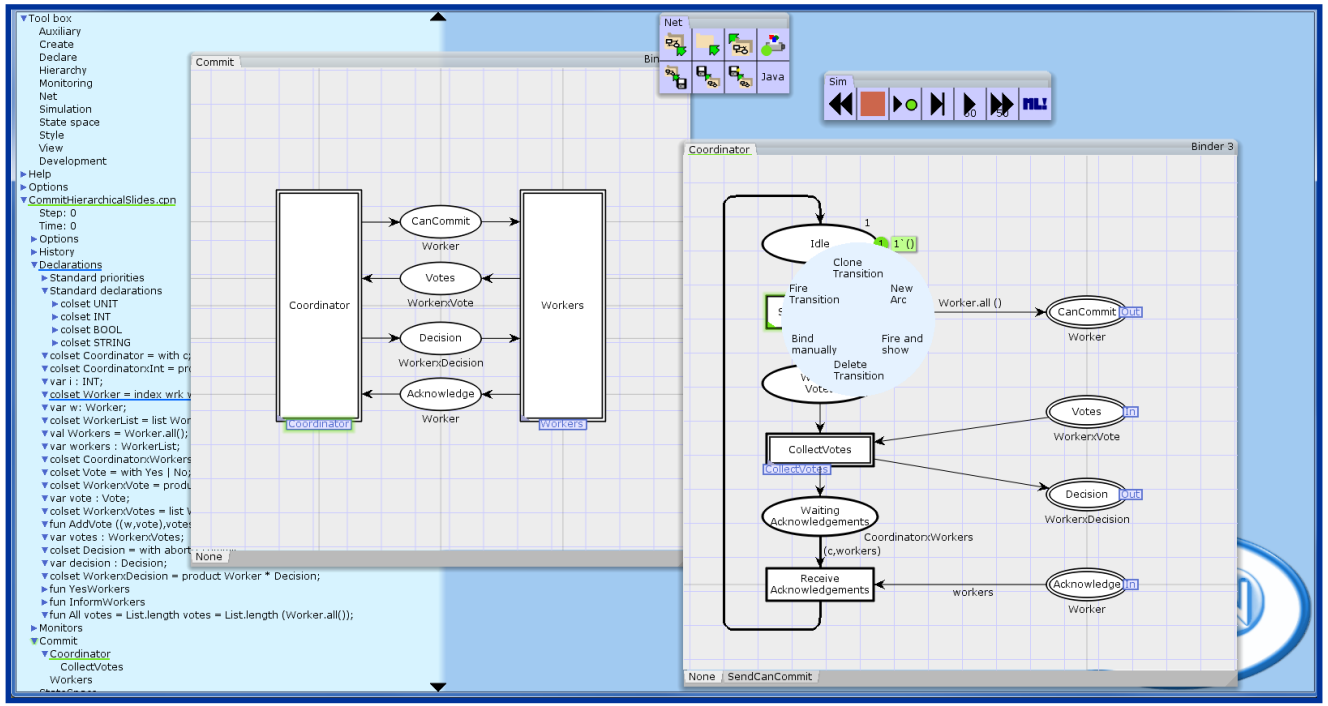
- graphical notation
- concurrency
- communication
- synchronisation
- resource sharing

CPN ML (Standard ML)

- data and data manipulation
- compact modelling
- parameterisable models

CPN Tools [www.cpntools.org]

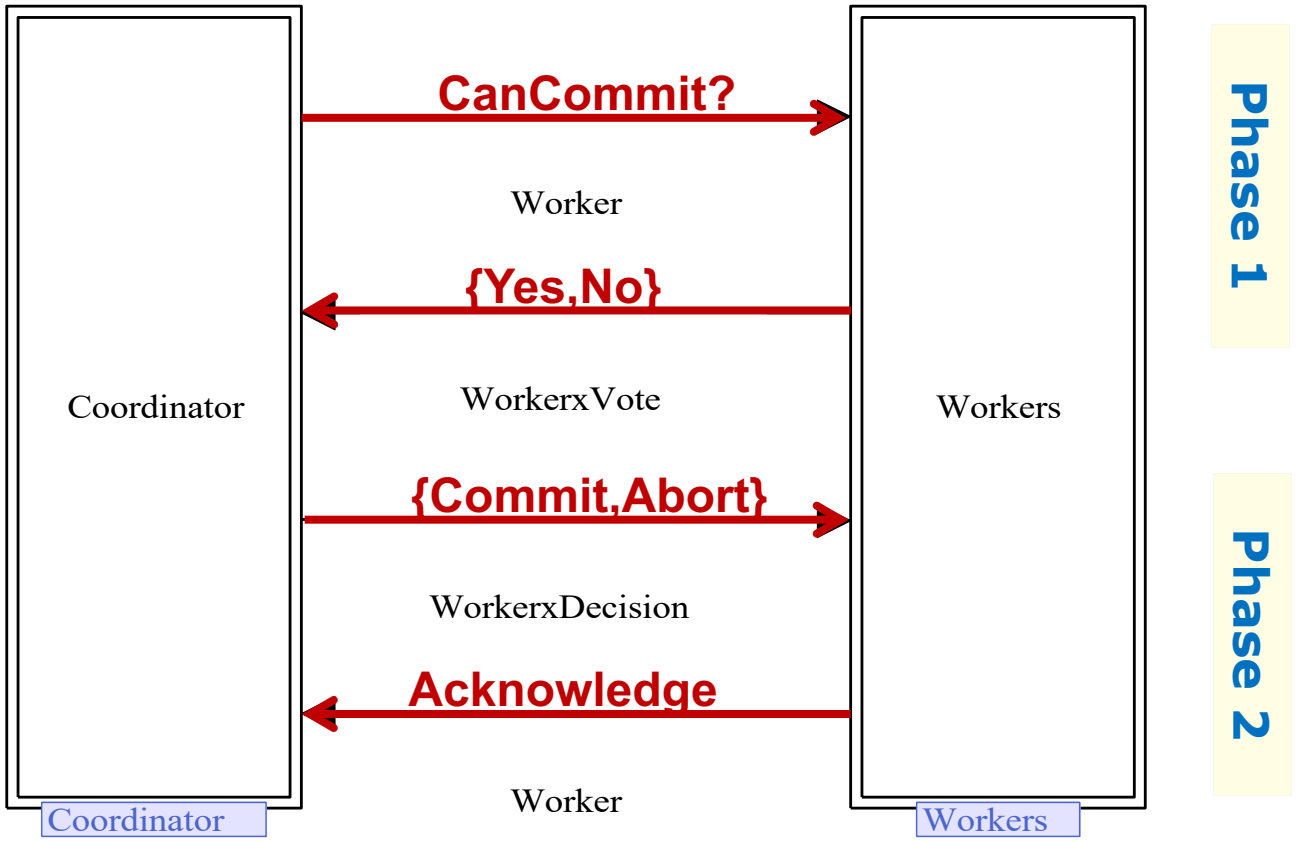
- **Practical use of CPNs is supported by CPN Tools**



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

Example: Two-phase Commit Transaction Protocol

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



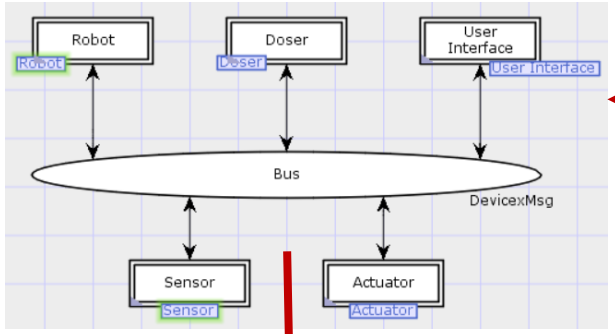
CPN Tools demo

- **User-interaction with CPN Tools**
  - **Index and workspace**
  - **Binders and tool palettes - drag-and-drop**
  - **Contextual menus - right click**
  - **No menu-bars or dialog-boxes**

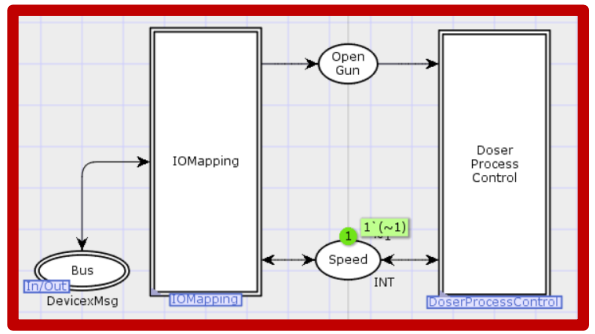
CPN @ Atlas Copco

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

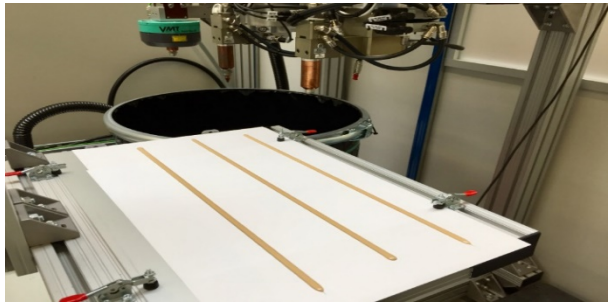
CPN Tools: editing, validation, and verification (design time)

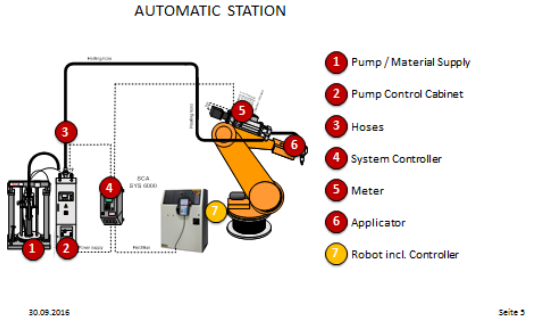


Environment modelling for (non-site) software testing



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

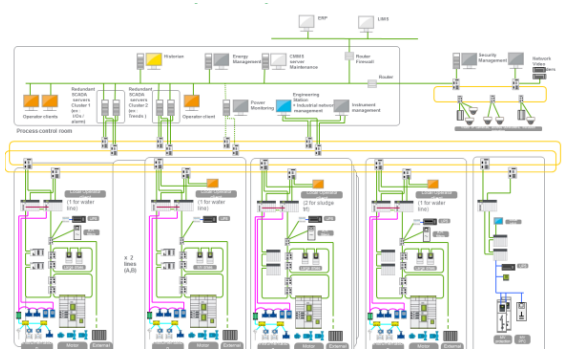




- **The CPN model is directly used as the pump controller software implementation**

CPN @ Schneider Electric

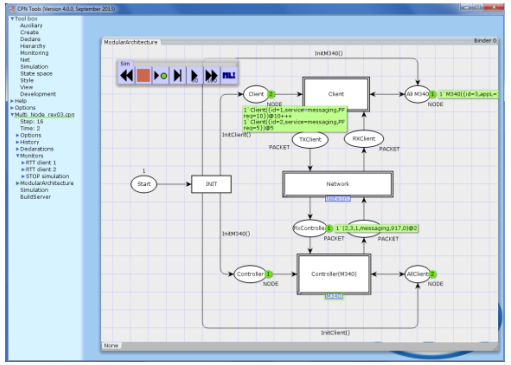
- **Dependability evaluation and capacity planning of large industrial automation architectures**



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

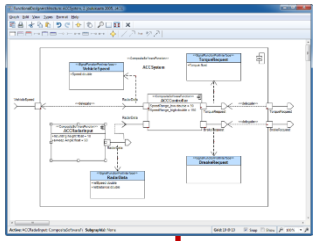
http://cs.au.dk/cpnets/industrial-use/

Model-driven engineering

Problem domain
(vision and requirements)

Validate

Software engineering
based on models



Verify

Model
execution

Code
generation

Technical domain
(implementation)

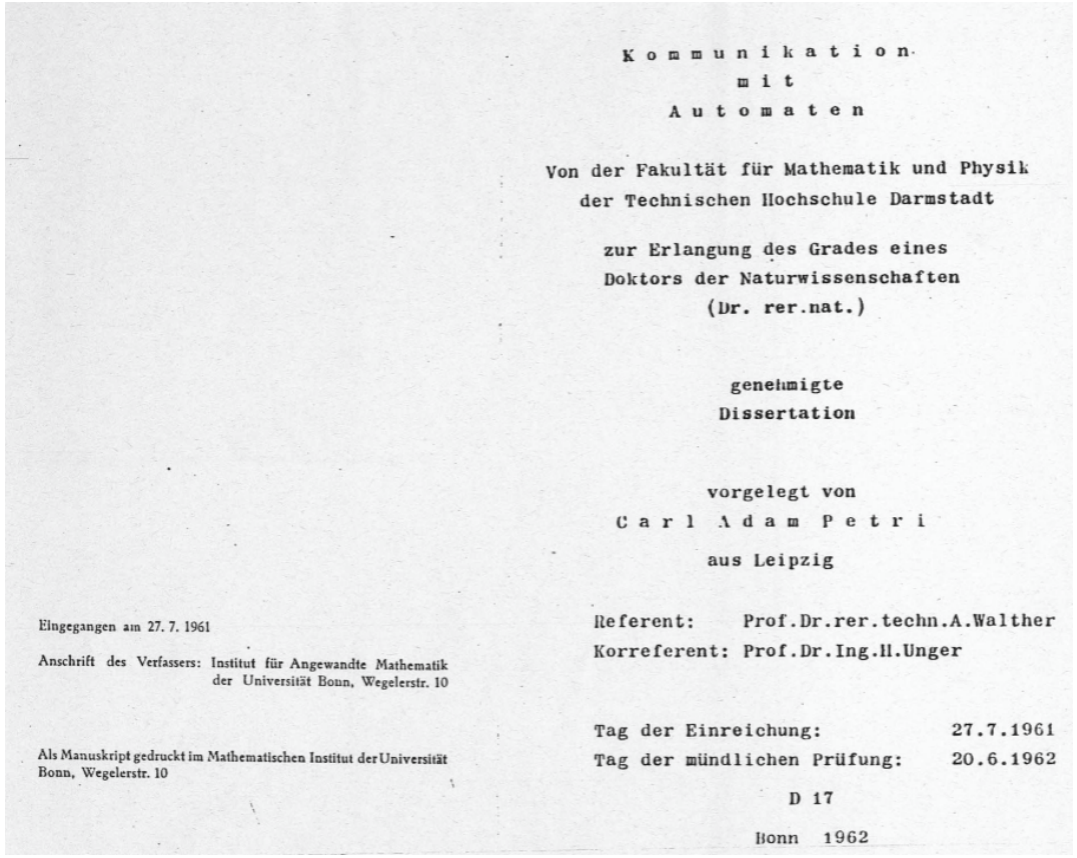
Validation:
Are we building the right software?

Verification:
Are we building the
software right ?

Background on Coloured Petri Nets

Petri Nets

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



K o m m u n i k a t i o n
m i t
A u t o m a t e n

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
C a r l A d a m P e t r i
aus Leipzig

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

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

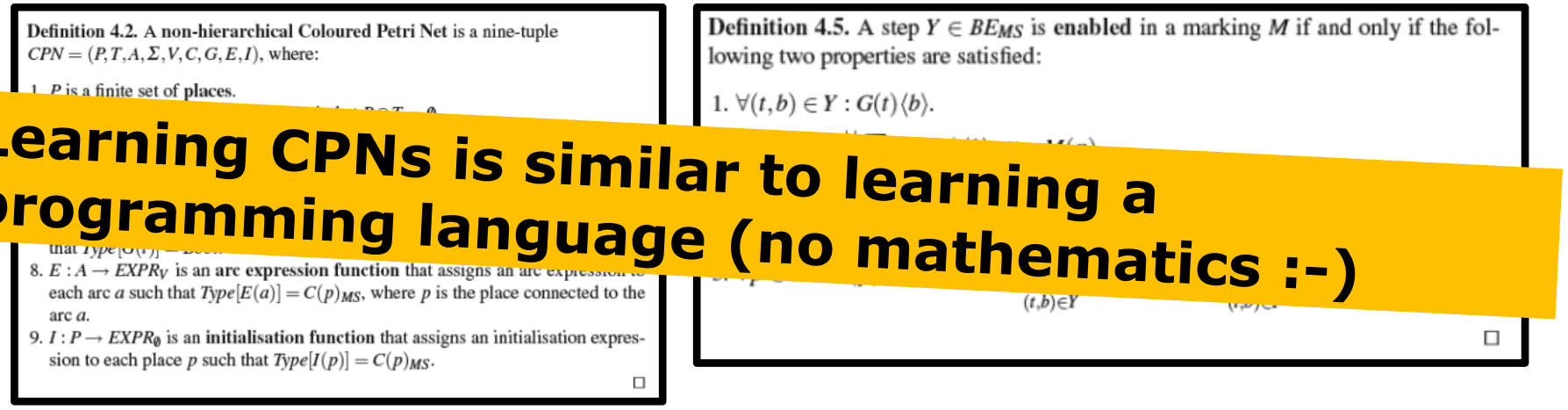
High-level Petri Nets

- 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*
  - The CPN modelling language and the supporting CPN Tools conform to this standard.

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

CPN models are formal

- **The CPN modelling language has a mathematical definition of both its syntax and semantics**
- **The formal representation is important**
  - Would have been impossible to develop a sound and powerful CPN language without it
  - Provides the foundation for the definition of the behavioural properties and for the formal analysis and verification methods



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**.

8. $E : A \rightarrow EXPR_V$ is an **arc expression function** that assigns an arc expression to each arc a such that $Type[E(a)] = C(p)_{MS}$, where p is the place connected to the arc a.
9. $I : P \rightarrow EXPR_\emptyset$ is an **initialisation function** that assigns an initialisation expression to each place p such that $Type[I(p)] = C(p)_{MS}$.

□

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. $\forall (t,b) \in Y : G(t)\langle b \rangle$.

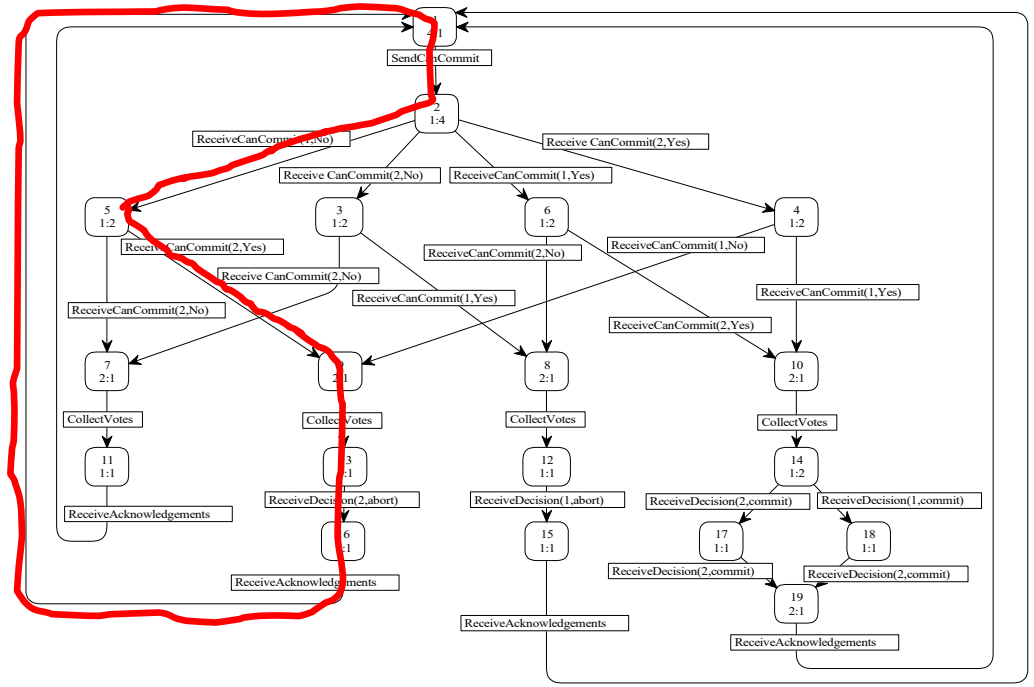
$(t,b) \in Y$

□

Learning 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**

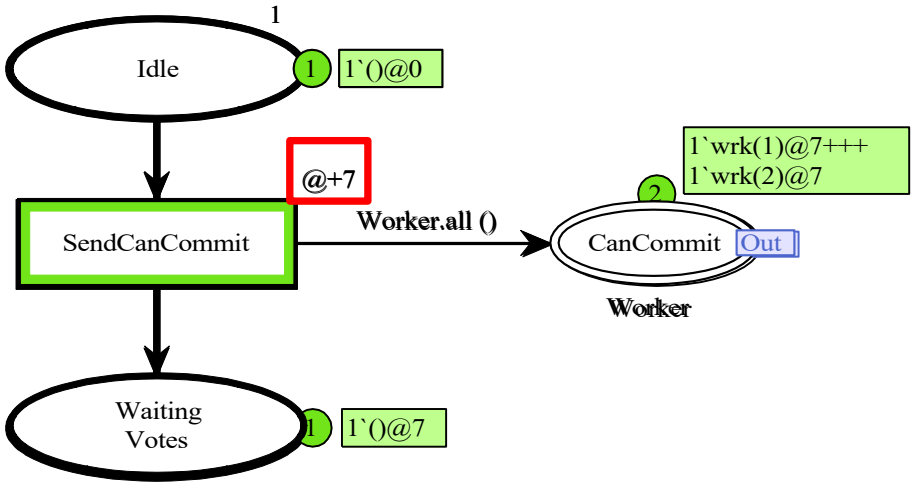


- A state space represents all possible executions of the CPN model
- Standard behavioural properties can be investigated using the state space report
- Model-specific properties can be verified using queries and temporal logic model checking

- **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**

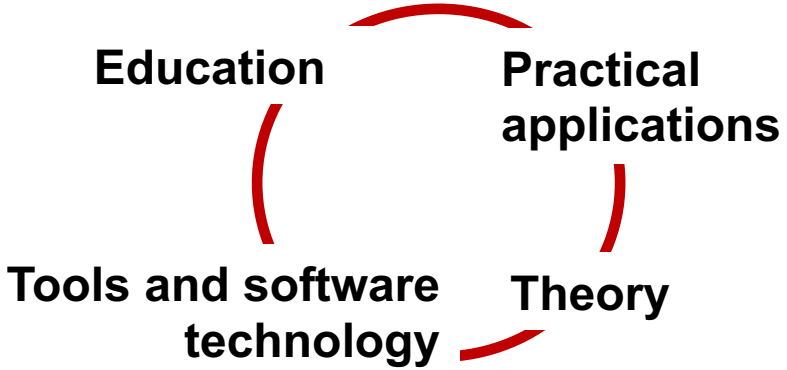


- **A global clock representing the current model time**
- **Tokens carry time stamps describing the earliest possible model time at which they can be removed**
- **Time inscriptions on transitions and arcs are used to give time stamps to the tokens produced on output places**

- **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**

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**



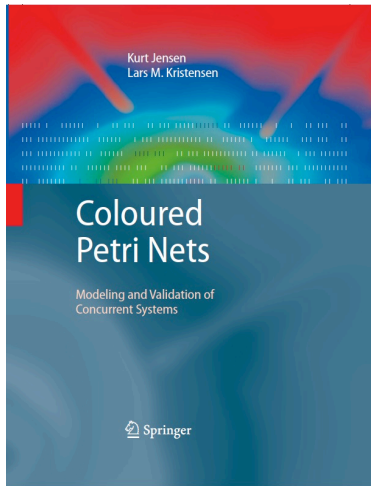
Practicalities

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 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
  - A larger example on the industrial use of CPNs and CPN Tools

Do not hesitate to ask questions along the way!

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

Coloured Petri Nets and CP
of concurrent systems

Colored Petri Nets:
Modeling and Val

ABSTRACT

Categories and Subject Descriptors

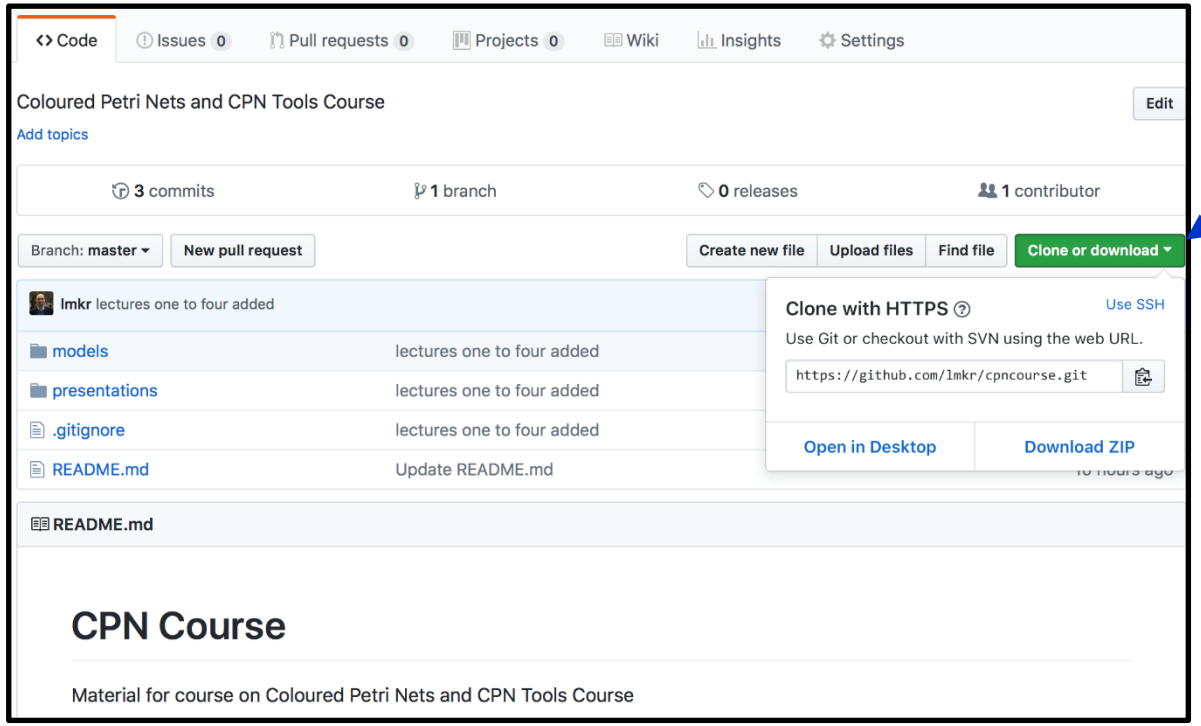
1. INTRODUCTION

1. Introduction

- **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.

Course material

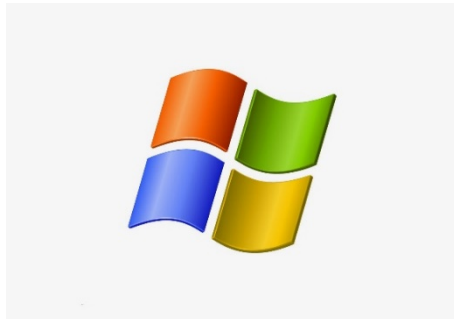
- **Slides, models, and papers are available via the github repository at** https://github.com/lmkr/cpncourse



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**
- **Hands-on with CPN Tools?**