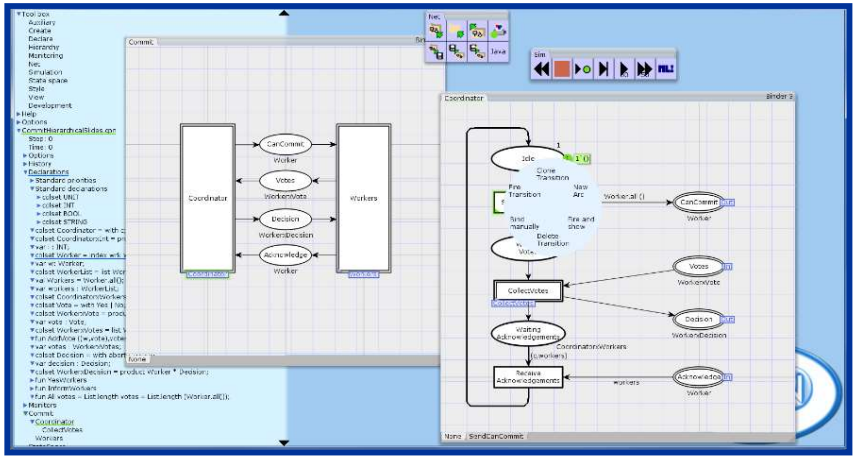
## Lecture 5

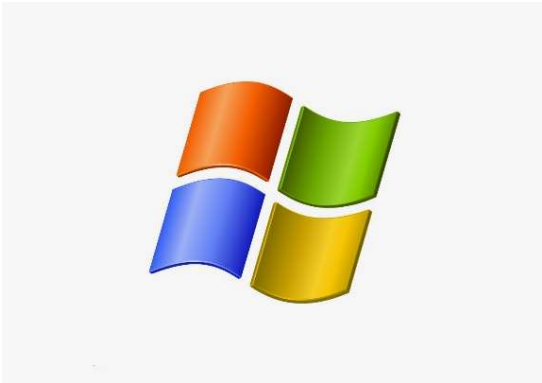
# Hands-on with CPN Tools



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

# Installation

- **CPN Tools can be downloaded and installed via [www.cpntools.org](http://www.cpntools.org)**

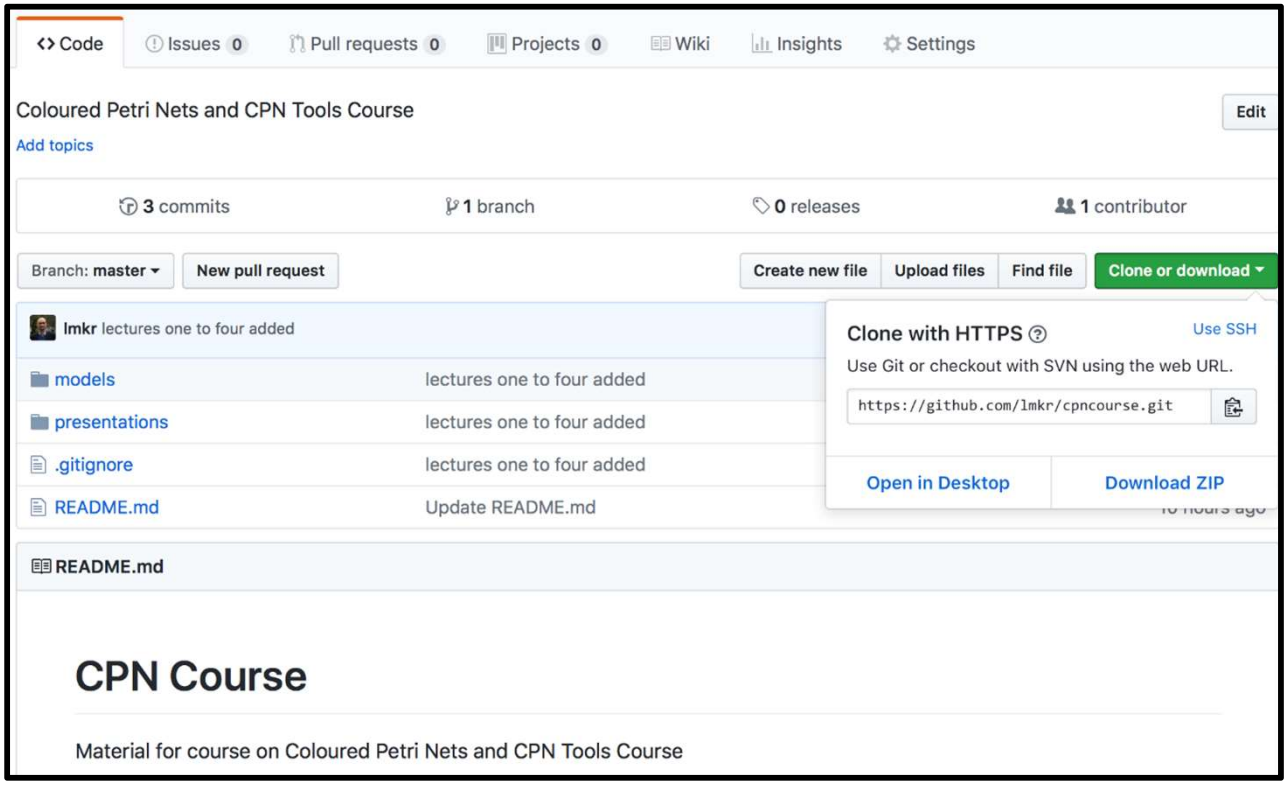
**Running on Mac OS / Linux via a virtual machine or emulator.**

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

# Material

- **Models and presentations are available via the public github repository at**

  **https://github.com/lmkr/cpncourse**




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

# Hands-on Overview

- **Hands-on 1: Navigate and simulate CPN models**
- **Hands-on 2: Editing of CPN models**
- **Hands-on 3: Building a controller PT-net model**
- **Hands-on 4: Building a controller CPN model**

# Hands-on 1: Simulation

- The main aim is to become familiar with how to navigate and simulate CPN models
- Open the two-phase-commit protocol CPN model located in the hands-on folder.
- Use the tools in the Simulation tool palette to conduct interactive and automatic simulations.
- Investigate scenarios where some workers vote No and a scenario where all workers vote Yes.

# Hands-on 2: Editing

- **The main aim is to become familiar with how to edit and modify CPN models**
- **Modify the two-phase commit protocol CPN model from hands-on 1 such that**
  1. The coordinator and the workers no longer return to their initial idle state upon completion of the protocol.
  2. The coordinator terminates in a place CoordinatorCompleted and the workers terminate in a place WorkerCompleted
  3. The coordinator puts an Abort-token or a Commit-token on a new place Result to indicate the result of the transaction
  4. Each worked records it vote locally by putting a token (w,vote) in a new place Votes
- **Use simulation to validate the revised model.**

# Case Study: Controller

- **Consider a concurrent system consisting of a controller communicating over a bus with**
  - An actuator – a motor for moving up/down a follower plate and which can be moving up, moving down or be stopped.
  - Two sensors – sends a signal if an UPPER limit / LOWER limit of the follower plate has been reached.
- **Model a controller that will move the follower plate up and down between the two limits.**
- **The controller should start when it receives a START command and stop on a STOP command.**

# Hands-on 3: PT-net Controller

- **Use Place/Transition nets to model (parts) of the controller**
- **The model should consists of**
  - An environment / plant part modelling the motor, the follower plate, and the two sensors
  - One part modelling the application logic of the controller
  - One part (places) for communication between the plant and the controller
  - One part for receiving the start and stop commands
- **For now – assume that the motor (follower plate) can be in three states STOPPED, MOVINGUP, MOVINGDOWN (we ignore exact position)**

# Hands-on 4: CPN Controller

- **Use CPNs to model the controller**
  - Model the communication across the bus as a single place with an enumeration colour set.
  - Model the state of the motor using an enumeration colour set consisting of the values MOVING_UP, MOVING_DOWN, STOPPED.
- **Split the model into modules**
  - One module for the motor and sensors
  - One module for the user behaviour
  - One module for the controller logic