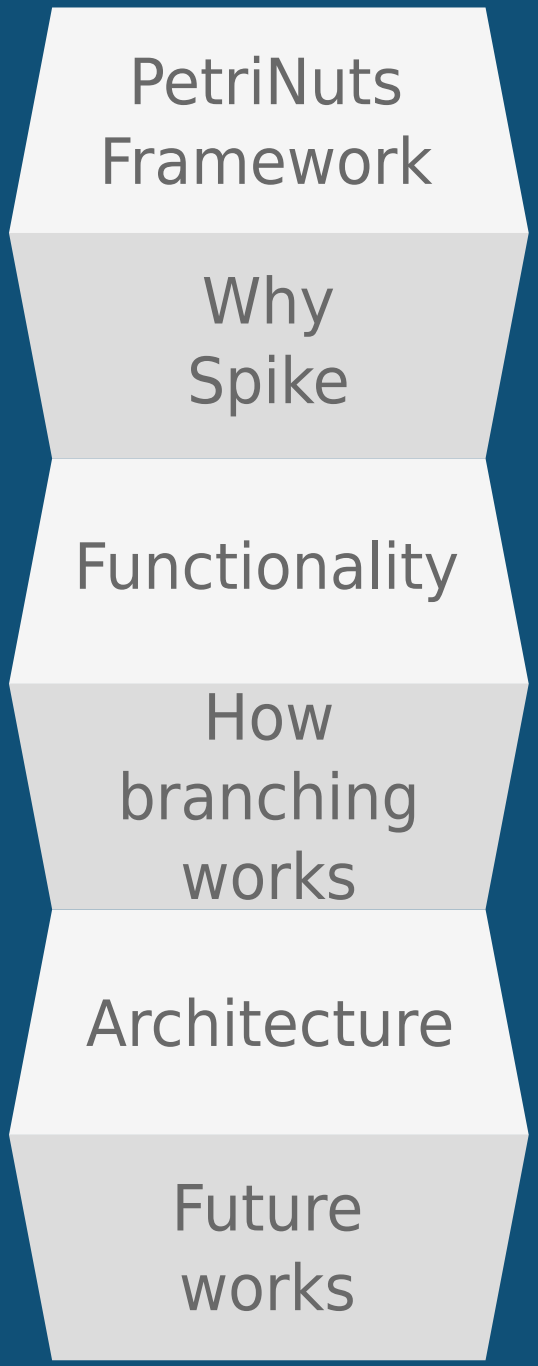
Spike – reproducible simulation experiments with configuration file branching

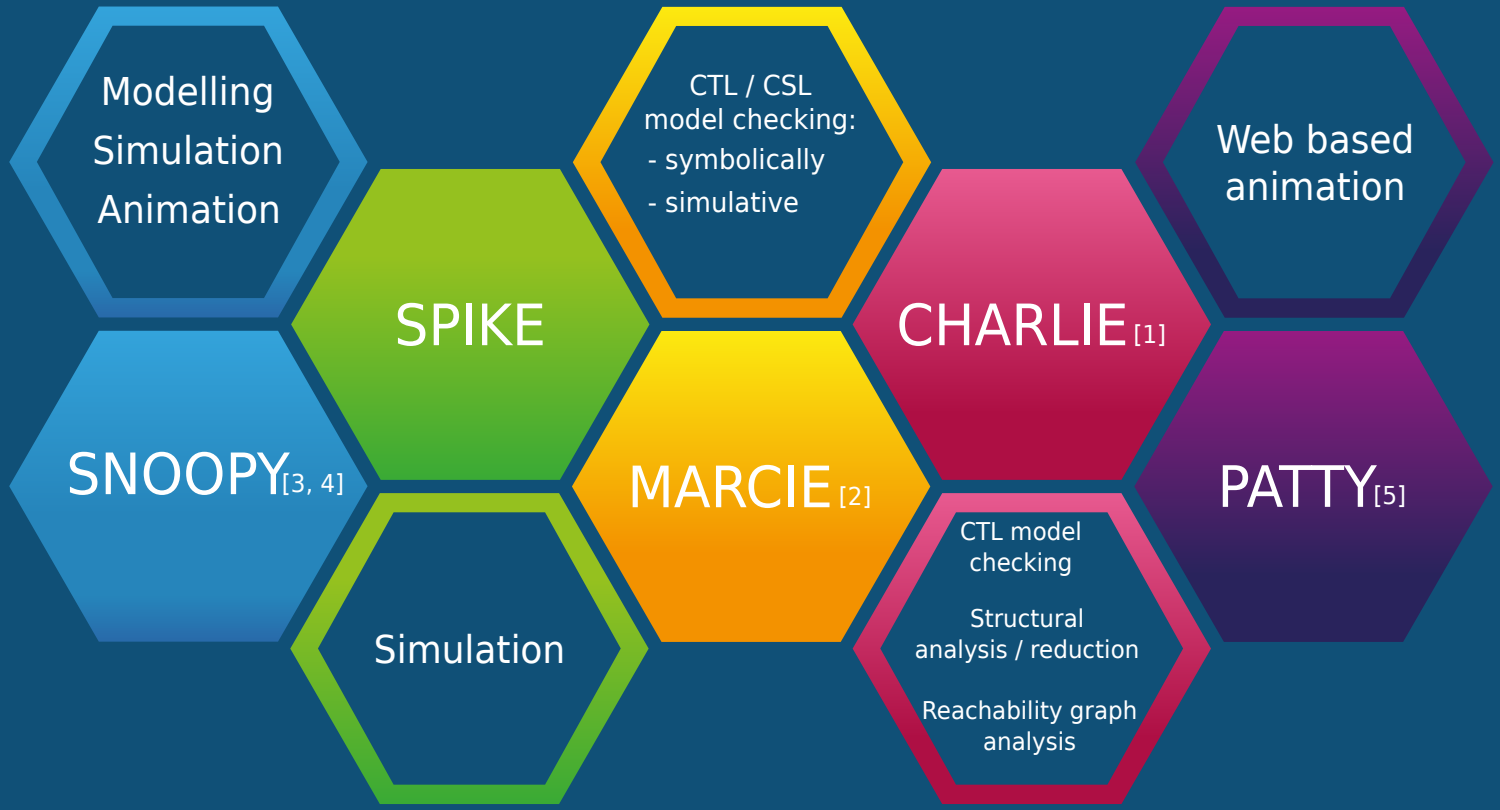
Jacek Chodak, Monika Heiner (supervisor)
Brandenburg University of Technology
Cottbus, Germany
jacek.chodak@b-tu.de, monika.heiner@b-tu.de
http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Spike

Agenda



jacek.chodak@b-tu.de

PetriNuts Framework



References:

- [1] M Heiner, M Schwarick and J Wegener: PETRI NETS 2015
- [2] M Heiner, C Rohr and M Schwarick: PETRI NETS 2013
- [3] M Herajy, F Liu, C Rohr and M Heiner: BMC Systems Biology 2017
- [4] C Rohr, W Marvan, M Heiner: Bioinformatics 2010
- [5] K Schulz, BA thesis, BTU Cottbus, CS department 2008

03

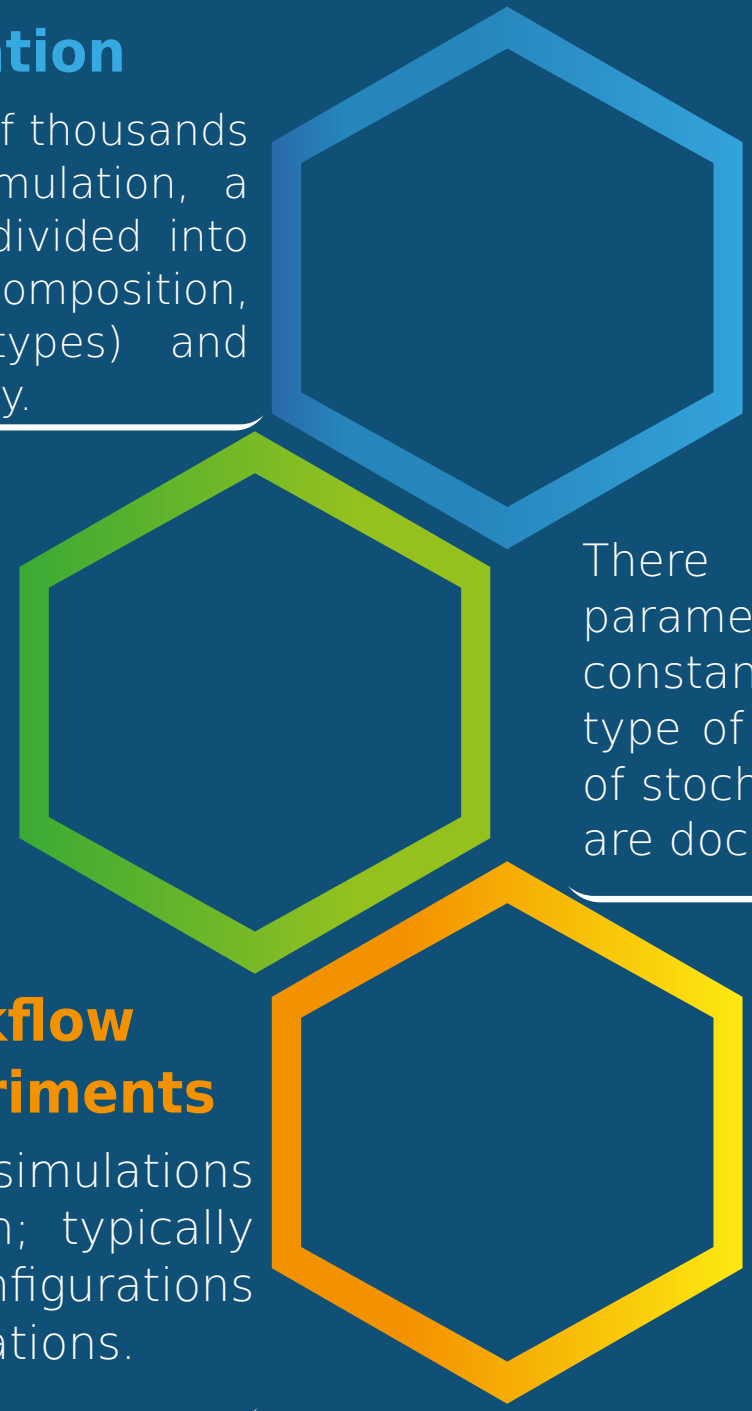
CSMB 2019

jacek.chodak@b-tu.de

Why Spike

Speed-up simulation

Models can contain dozens of thousands of nodes. To speed up simulation, a model can be reduced or divided into modules (spatial decomposition, decomposition by node types) and simulated in a distributed way.



Reproducibility

There are many parameters: model parameters (e.g., initial marking, kinetic constants); simulations parameters (e.g., type of algorithm, length of trace, number of stochastic runs). Experiments with Spike are documented by configuration files.

Simplifying workflow of simulation experiments

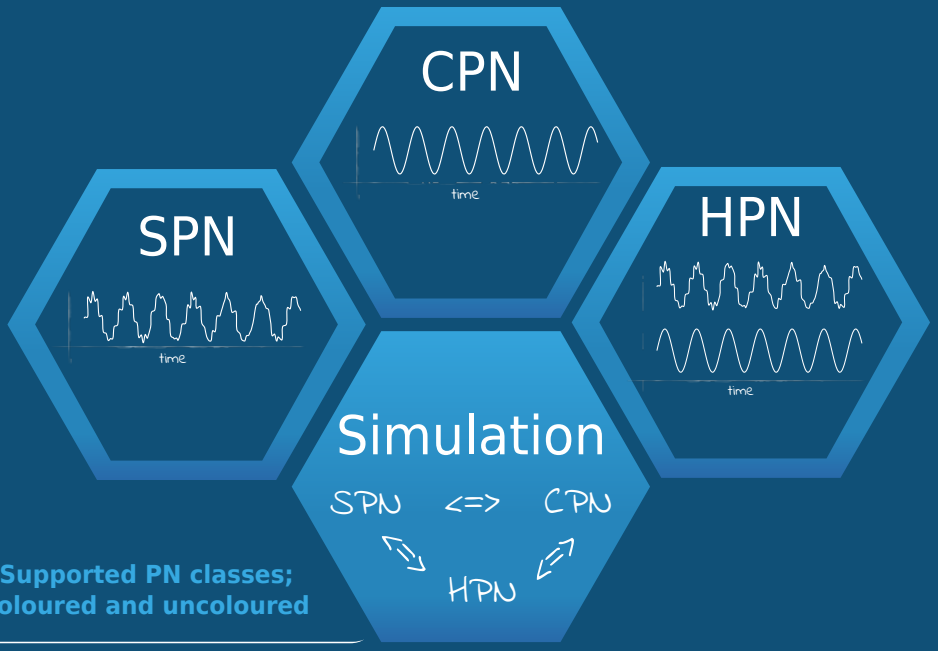
CLI controlling multiple simulations without user intervention; typically for different model configurations and/or simulator configurations.

04

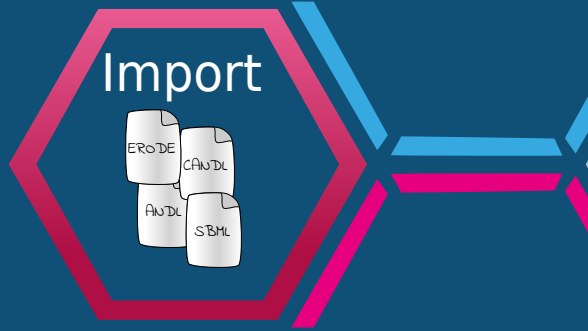
CSMB 2019

jacek.chodak@b-tu.de

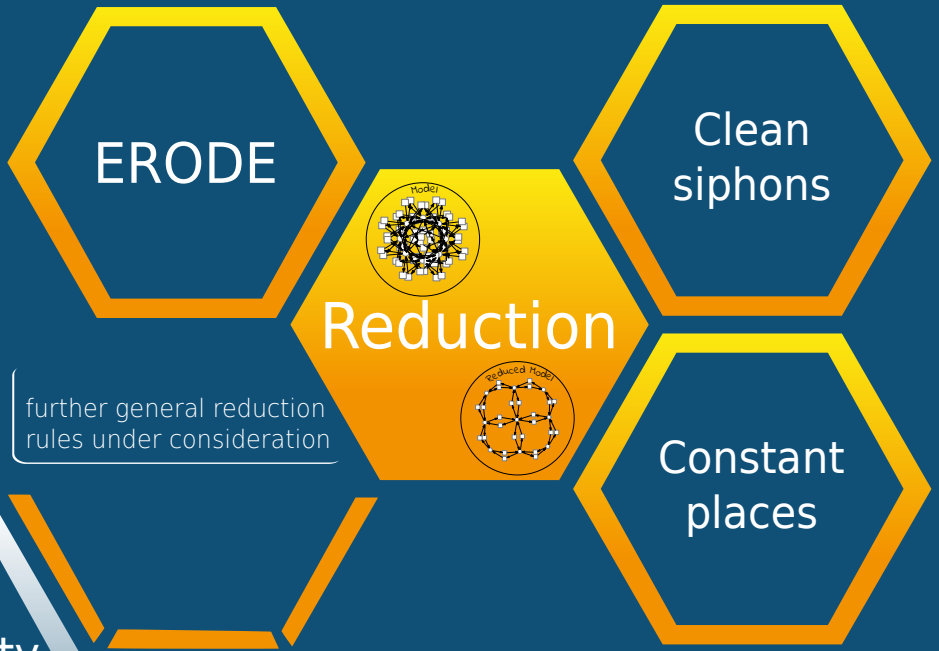
Functionality



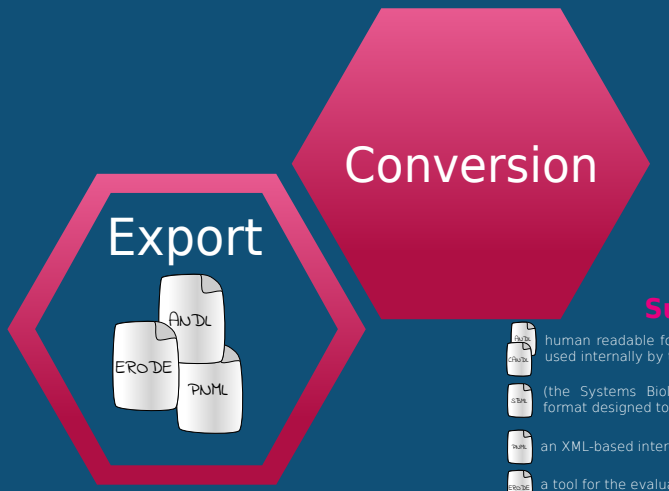
Supported PN classes; coloured and uncoloured



Functionality

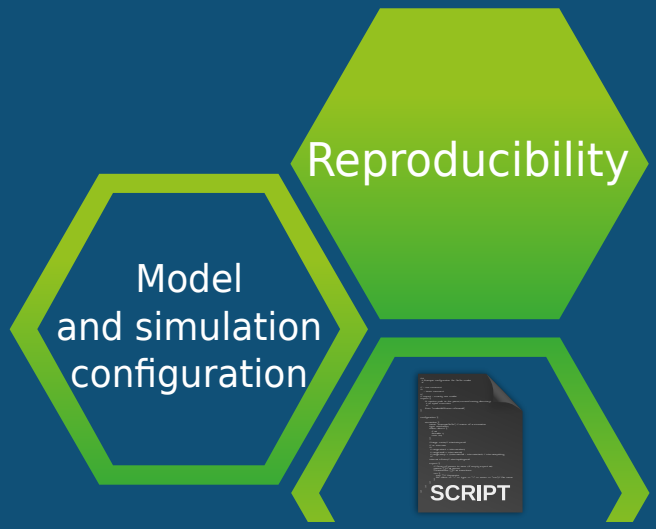


further general reduction rules under consideration



Supported formats

- human readable formats for Petri nets and Coloured Petri nets, respectively, used internally by the PetriNuts framework
- (the Systems Biology Markup Language) - an XML-based representation format designed to exchange computational models of biological processes
- an XML-based interchange format for Petri nets (no time)
- a tool for the evaluation and reduction of chemical reaction networks



05

CSMB 2019

jacek.chodak@b-tu.de

Configuration

definition of (named) constants

definition of observers (auxiliary variables)

```
configuration: {
  model: {
    constants: {
      ...
    }
    places: {
      ...
    }
    observers: {
      ...
    }
  }

  simulation : {
    ...
    export: {
      ...
    }
  }
  ...
}
```

specification of model parameters and simulator options

specification of multiple exports of simulation results

jacek.chodak@b-tu.de

Configuration

```
constants: {
  // name of a group
  all: {
    /* if constant does not exist
    * then it will be created and
    * can be used in the configuration ,
    * for example in defining a place marking
    */
    M: ”D/2 + 1”;
  }
  kineticParam: {
    Threshold: 1200;
  }
}
```

definition of (named) constants

which can get a specific value or a set of values,
either specified by a list or an interval;
supported data types: boolean, integer, real, string

jacek.chodak@b-tu.de

Configuration

```
places: {
  // example of use of the newly created constant M
  P: "1000`(M,M)";
  P_2_2: 500;
}
```

specification of model parameters

using either constants or (direct) values as arguments; values can be given as a single specific value or a set of values, specified by a list or an interval

jacek.chodak@b-tu.de

Configuration

```
observers: {
 place: {
  OP01: {
   function: "P_1_1 + P_2_3";
  }
 }
}
```

definition of observers (auxiliary variables)

allow for extra measures by defining numerical functions; depending on the type of observer, it can involve, beside constants, places, transitions or simultaneously places and transitions

jacek.chodak@b-tu.de

Configuration

```
simulation: {
  name: "exampleDiffusion_2D4"; // Name of a simulation
  type: stochastic; // continuous, stochastic, hybrid
  solver:
    direct: {
      // if 0 try to use as many thread as CPU cores
      threads: 0;
      runs: 10;
    }
    // range: 0:2:100;// start:step:end
    // or intervals
    /*
     * range.start = interval.start;
     * range.end = interval.end;
     * range.step = (interval.end - interval.start)
     *              / interval.splitting;
     */
     varSplit: [10:5:20];
     interval: 0:varSplit:100;// start:spliting:end
  ...
}
```

simulator options

using either constants or (direct) values as arguments; values can be given as a single specific value or a set of values, specified by a list or an interval

jacek.chodak@b-tu.de

Configuration

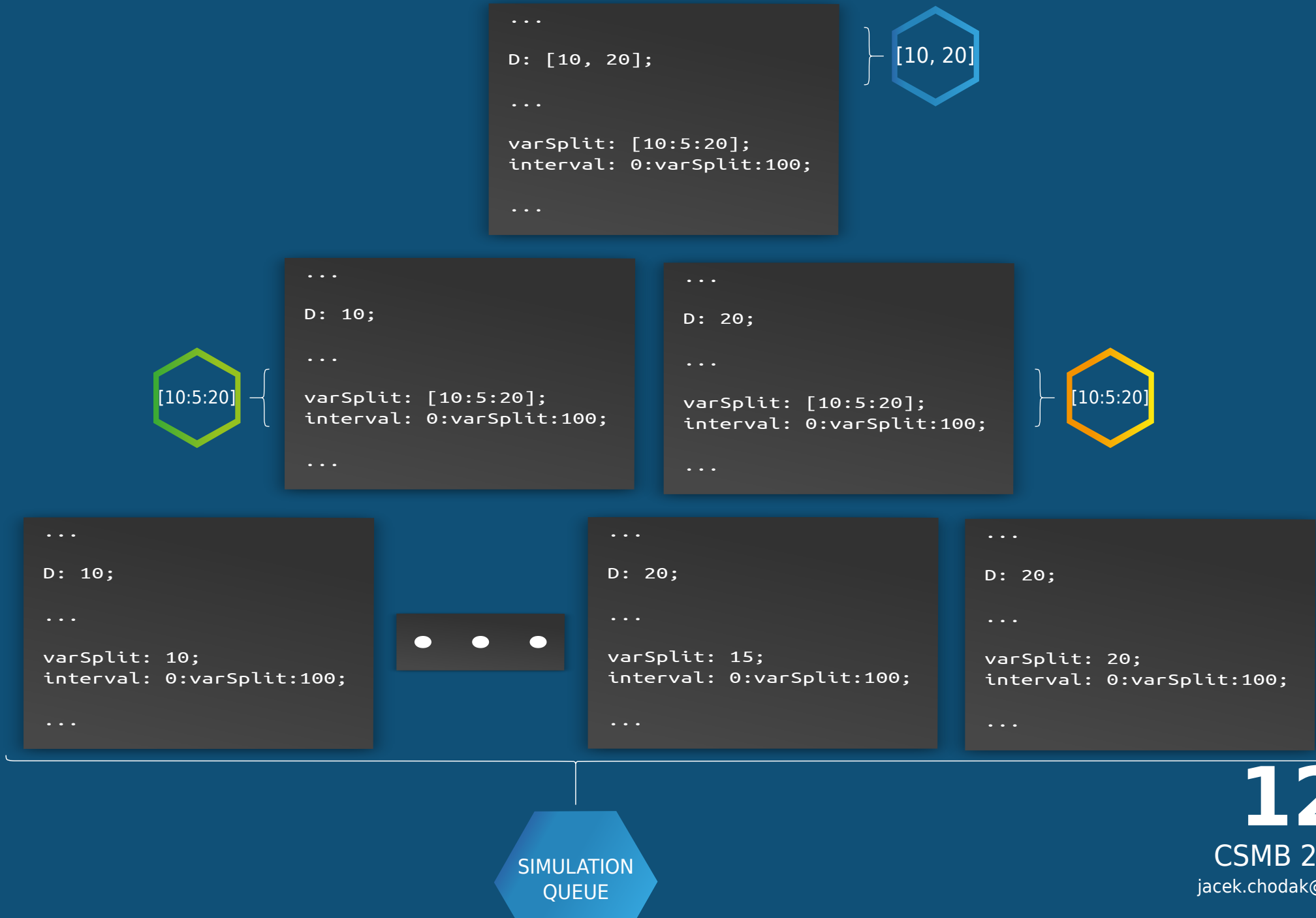
```
export: {
  /*
   * export places with given prefix P_1
   * or which belong to the colour set Grid2D
   */
   places: ["P_1.*", "Grid2D"];
   transitions: c["t3"];// given coloured transitions
   transitions: u["t3_1_1_1_2"];// given uncoloured transitions
   observers: [];// export all observers
   csv: {
     sep: ";";// Separator
     file: import.name
             << "_places"
             << "_" << type
             << "_" << solver
             << "_" << solver.runs
             << "x" << interval.end
             << "x" << model.constants.all.D
             << ".csv";// File name
   }
}

export: { ... }
```

specification of multiple exports of simulation results

by use of regular expressions over the nodes of which the simulation traces are to be recorded; it is possible to combine the results of places, transitions and observers, coloured and uncoloured, in one CSV file

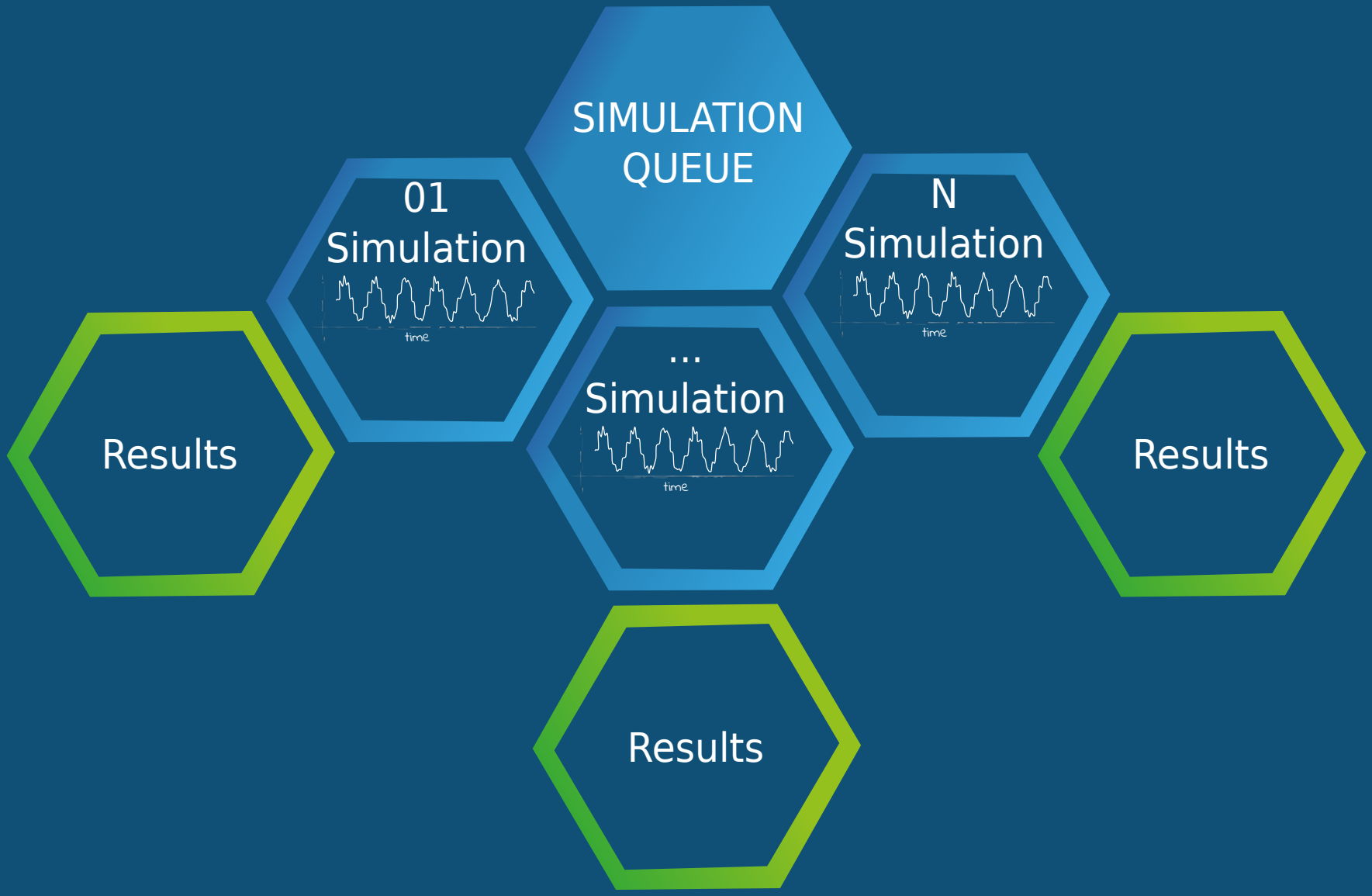
jacek.chodak@b-tu.de

How branching works



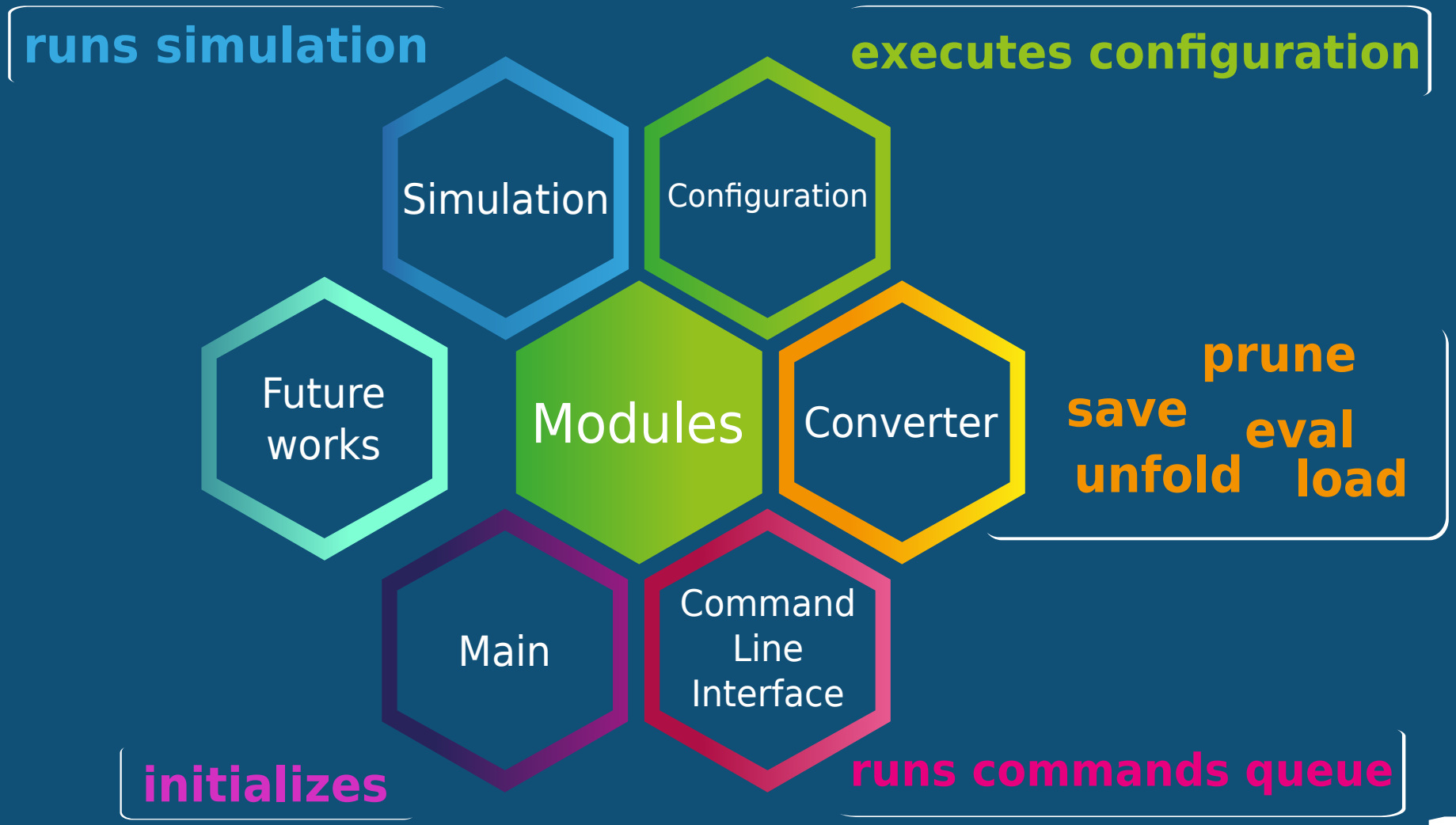
jacek.chodak@b-tu.de

How branching works



jacek.chodak@b-tu.de

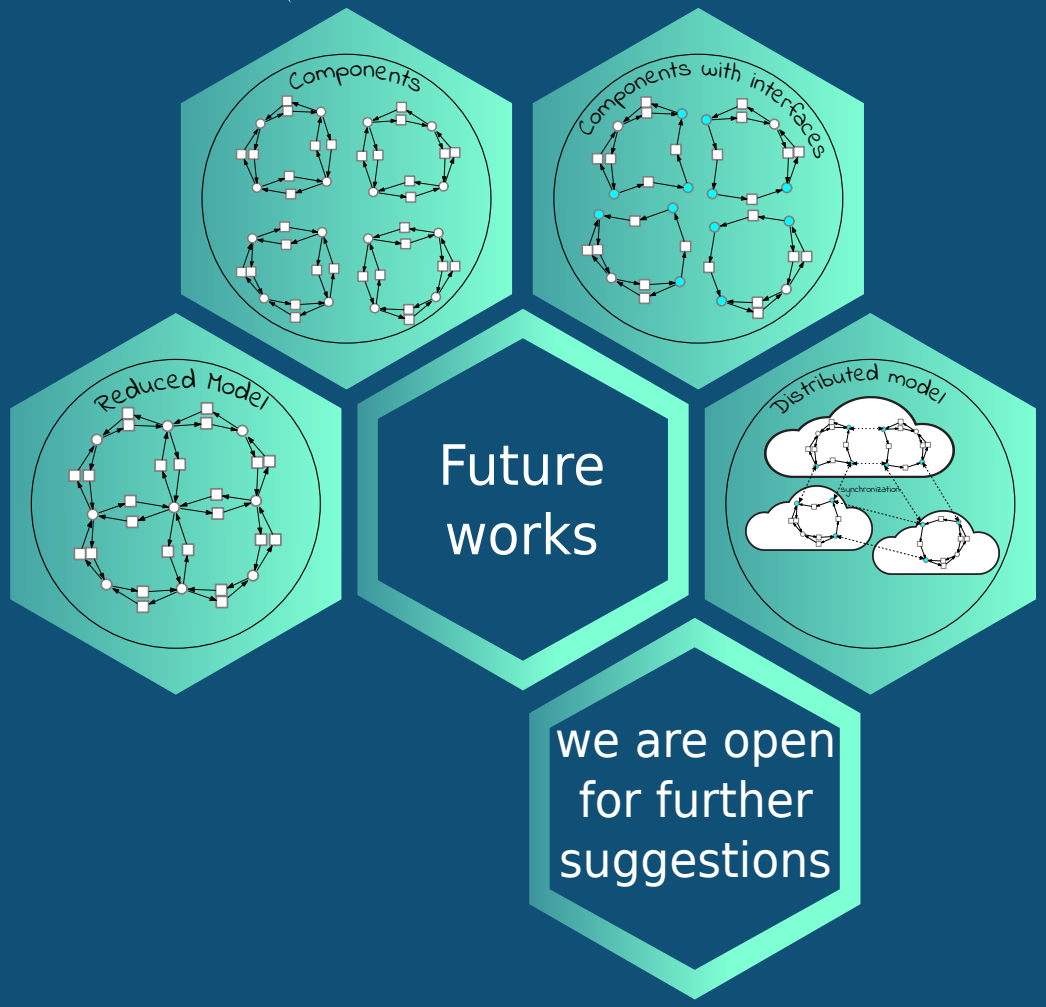
Architecture



jacek.chodak@b-tu.de

Future works

model decomposition



sophisticated model reduction

distributed simulation

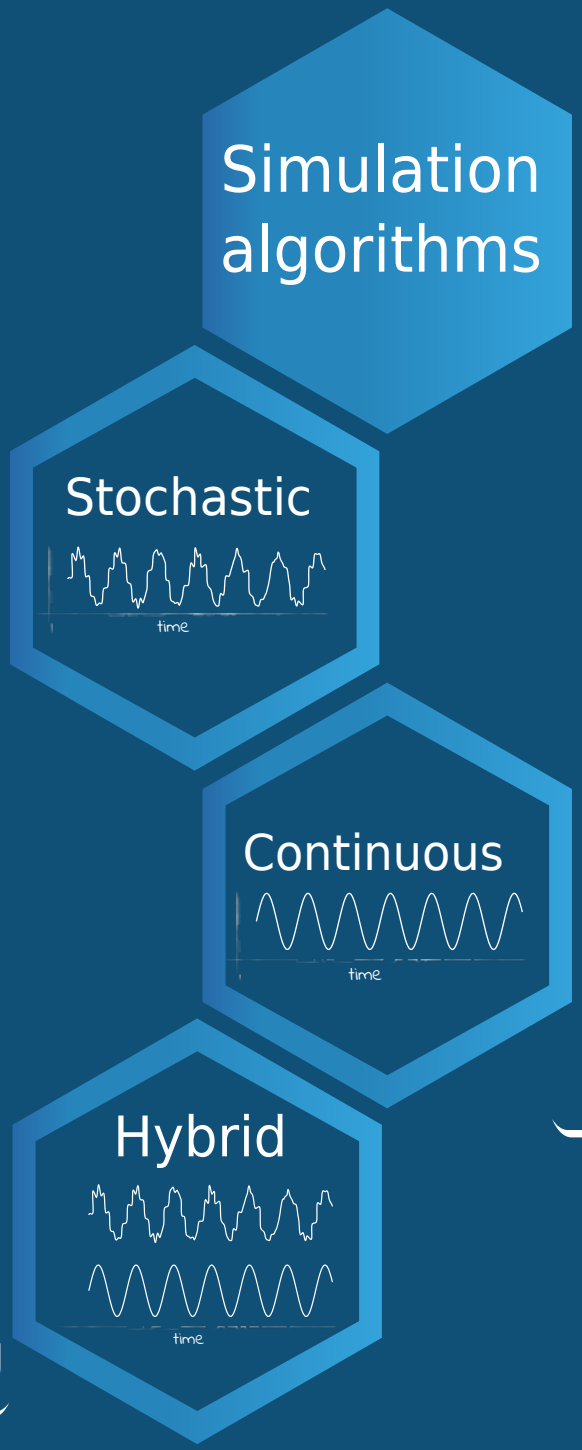
either for a set of simulations or a decomposed model

jacek.chodak@b-tu.de

Thank you
for attention

jacek.chodak@b-tu.de

Supported simulation algorithms



1. Gillespie
2. τ-leaping
3. δ-leaping
4. FAU

1. BDF
2. Adams-Moulton
3. Rosenbrock
4. classic Runge-Kutta
5. Euler
6. two-steps Euler

1. exact
2. accelerated exact
3. HRSSA
4. accelerated HRSSA
5. dynamic partitioning

jacek.chodak@b-tu.de