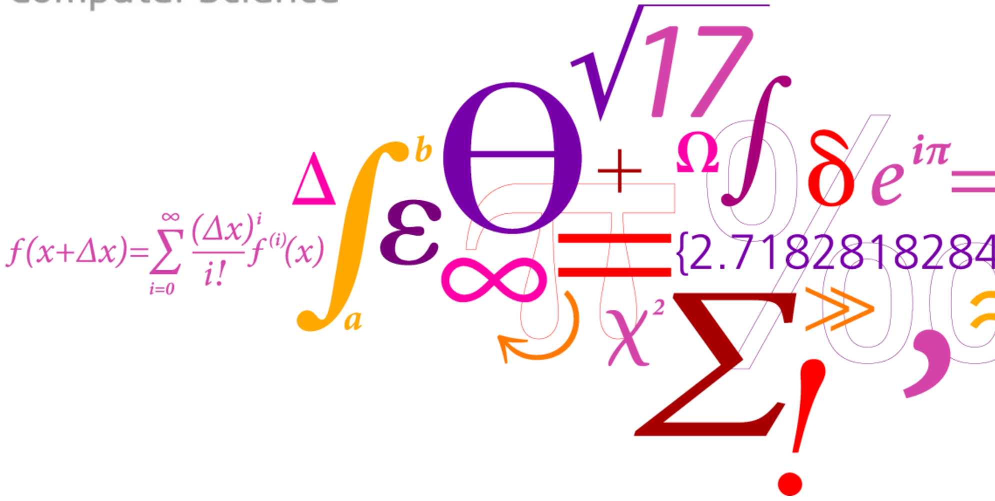
Model-based Software Engineering
(02341, spring 2016)

Ekkart Kindler

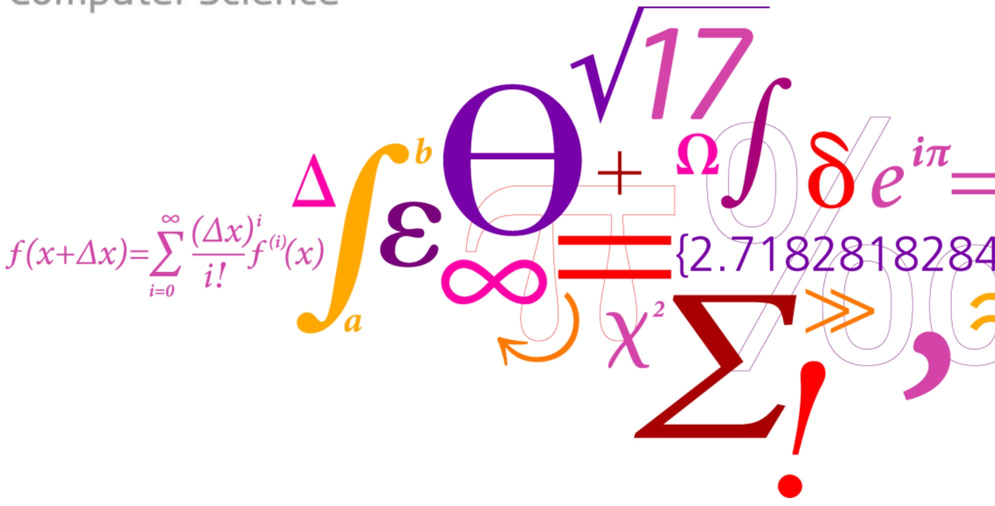
DTU Compute
Department of Applied Mathematics and Computer Science



The Project

A YAWL editor and simulator

DTU Compute
Department of Applied Mathematics and Computer Science



Course Structure

The course consists of three main parts:

- Lectures:
  Understand, ideas, principles, concepts, and technologies
- Tutorials:
  Accquire experience with technologies and tools in a small and controlled setting
- Project:
  Demonstrate that you can use ideas, concepts, technologies, and tools in (slightly) larger context.

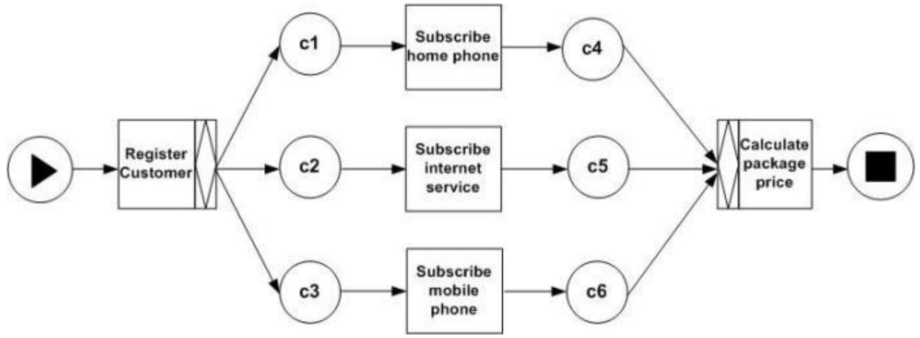
Lectures and tutorials are necessary prerequisites for properly doing the project!

Evaluation will be based on project: submitted software, written report, and the final presentation

YAWL

See lecture 4 for references and more details!

YAWL (Yet another Workflow Language) a graphical notation for modelling and enacting business processes



[from: http://www.yawlfoundation.org/pages/research/orjoin.html]

Task

- Implement a graphical editor for a subset* of YAWL and a simulator on top of that graphical editor visualizing the behaviour of a YAWL process
- This editor and simulator must be implemented based on the ePNK**

*) see scope for the specific subset your tool of YAWL thar must be supported

**) the ePNK will be discussed in tutorials 5-8 (see idea on next slides)

ePNK

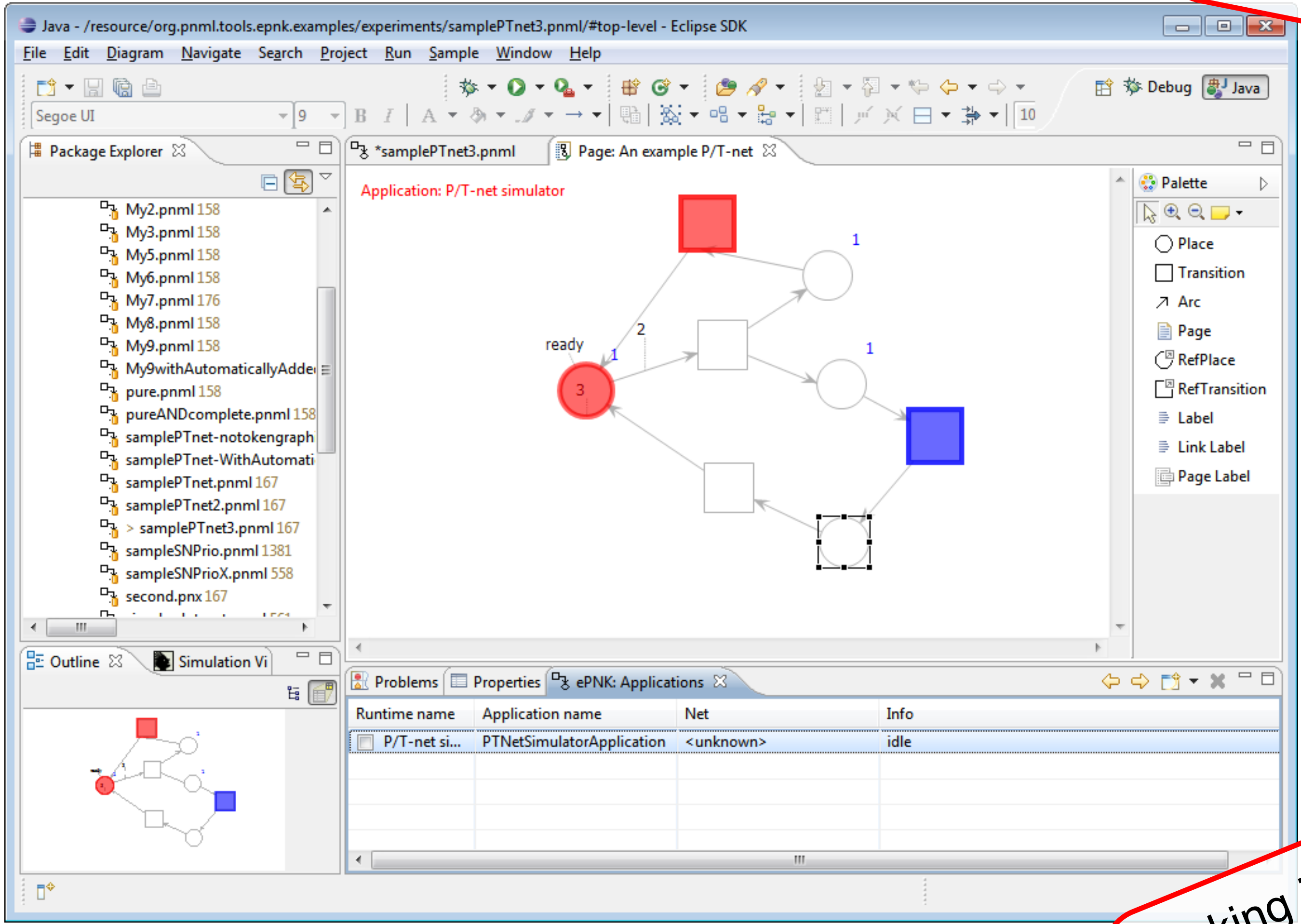
”The ePNK is a platform for Petri net tools based on the PNML transfer format. Its main idea is to provide generic Petri net types, which can be easily plugged into it, and to provide a simple generic GMF editor, which can be used for graphically editing nets of any plugged in type.” [ePNK Homepage]

ePNK

- A new Petri net type is defined by an EMF model, which, basically, can be plugged in to the ePNK (t5)
- ePNK provides a simple graphical editor, which can be customized (by programming) to feature specific graphical representation of the new net type (t6)
- Additional consistency conditions on the Petri net type can be plugged in too, as OCL or Java constraints (t7)
- Applications like simulators with graphical feedback can also be plugged in to the ePNK for some net types (t8)

ePNK

Screen shot of a simulator for ordinary Petri nets, which is deployed with the ePNK (with source code).



Java - /resource/org.pnml.tools.epnk.examples/experiments/samplePTnet3.pnml/#top-level - Eclipse SDK

File Edit Diagram Navigate Search Project Run Sample Window Help

Package Explorer

My2.pnml 158
My3.pnml 158
My5.pnml 158
My6.pnml 158
My7.pnml 176
My8.pnml 158
My9.pnml 158
My9withAutomaticallyAdde
pure.pnml 158
pureANDcomplete.pnml 158
samplePTnet-notokengraph
samplePTnet-WithAutomati
samplePTnet.pnml 167
samplePTnet2.pnml 167
> samplePTnet3.pnml 167
sampleSNPrio.pnml 1381
sampleSNPrioX.pnml 558
second.pnx 167

*samplePTnet3.pnml | Page: An example P/T-net

Application: P/T-net simulator

Problems | Properties | ePNK: Applications

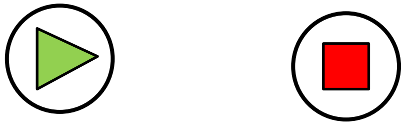
Runtime name	Application name	Net	Info
P/T-net si...	PTNetSimulatorApplication	<unknown>	idle

Looking into the source code and models of these projects might give you a lot of inspiration!

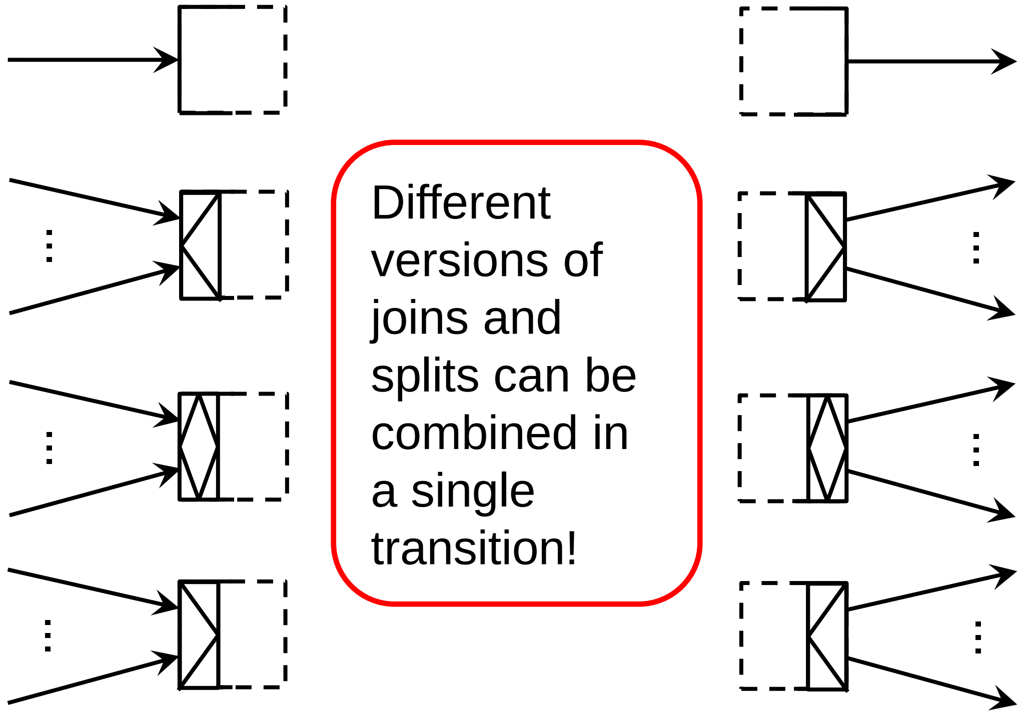
Scope

The tool must support the following YAWL features

- Start and end conditions (exactly on of each kind)



- Transition input/output: single, AND, XOR, OR



Different versions of joins and splits can be combined in a single transition!

single

XOR

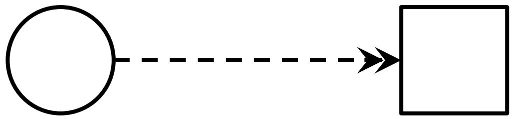
OR

AND

Scope (cntd.)

The tool must support the following YAWL features (cntd.)

- Reset arcs



All tokens removed, when transition fires (does not require a token on this place)

- Support the page concept of ePNK (flattening discussed in lectures/tutorial)

Data and organisation concepts do **not** need to be supported

Scope (cntd.)

The simulator must

- Provide graphical feedback on the current state of the process (marking)
- Visually indicate the enabled transitions/actions, and allow the user to select a transition to fire
- For XOR-joins and -splits allow the user to select from which place a token should be consumed and to which place the token should be produced
- For OR-splits allow the user to chose to which places a token should be produced
- For OR-joins indicate (give a warning) that on some unmarked input places a token might still arrive (and graphically indicate from where)

Submission

- The software as source code (exported Eclipse plugin projects)
- At least two YAWL examples (with reasonable processes)
- A report documenting your software (underlying models and design), including (but not limited to):
  - Intro and overiew of your project and ePNK extension
  - Domain models (EMF modles) with detailed discussion
  - Discussion of how your extension works together with the ePNK (software models, interfaces, interactions)
  - A brief handbook explaining the use of all features of your software (for an end user) using your examples (standard features of the ePNK as documented in the ePNK handbook do not need to be explained in detail)

Project

All submissions as groups via CampusNet (detailed instructions later)

Submissions / presentations:

- **February 12: Group members** of each group (3 to **4**)
- **March 11: Project definition**
- **April 18** (optional) : preliminary **submission of project** (including software and written report)
- **May 18: Final submission of project** (including software and report)
- **May 30: Presentation of final submission**
  - **Presentation**
  - **Questions and answers**

BTW: Due to the number of students, we might need an additional day for the final presentations! Which one would be good?