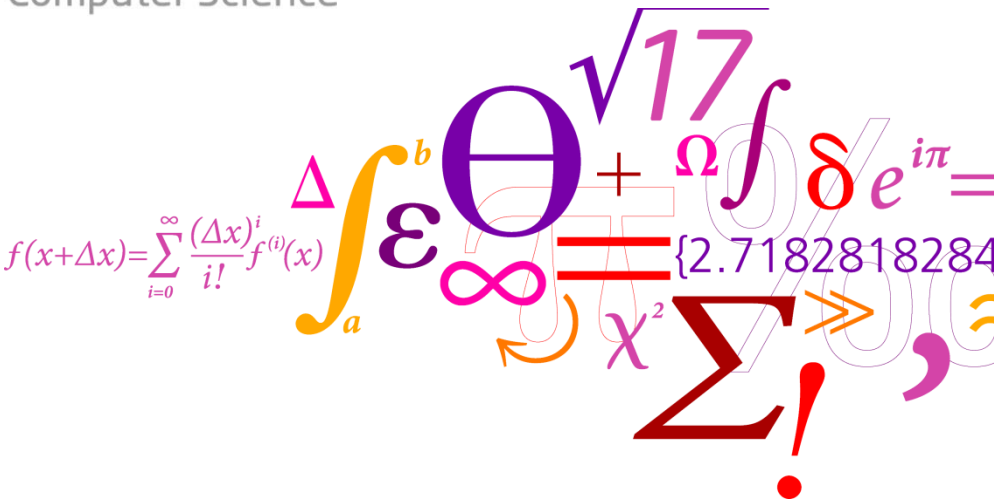
Model-based Software Engineering
(02341, spring 2016)

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



Project (again)

DTU Compute
Department of Applied Mathematics and Computer Science

1. Project Task

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

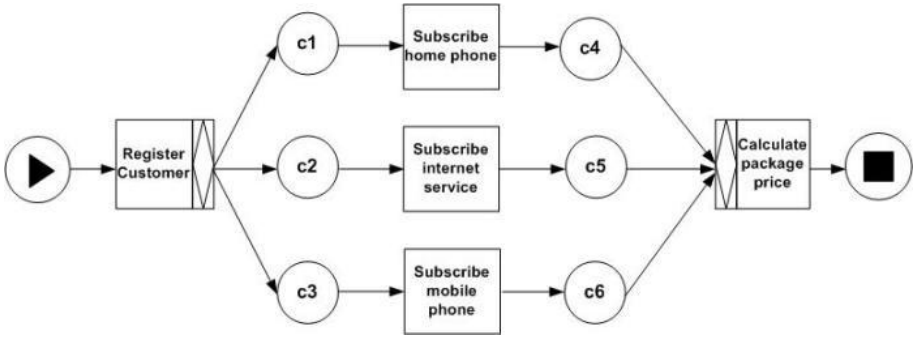
- Implement a graphical editor for a subset* of YAWL and a simulator on top of that graphical editor visualizing the behaviour of a YAWL process
- This editor and simulator must be implemented based on the ePNK**

*) 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)

YAWL

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



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

ePNK

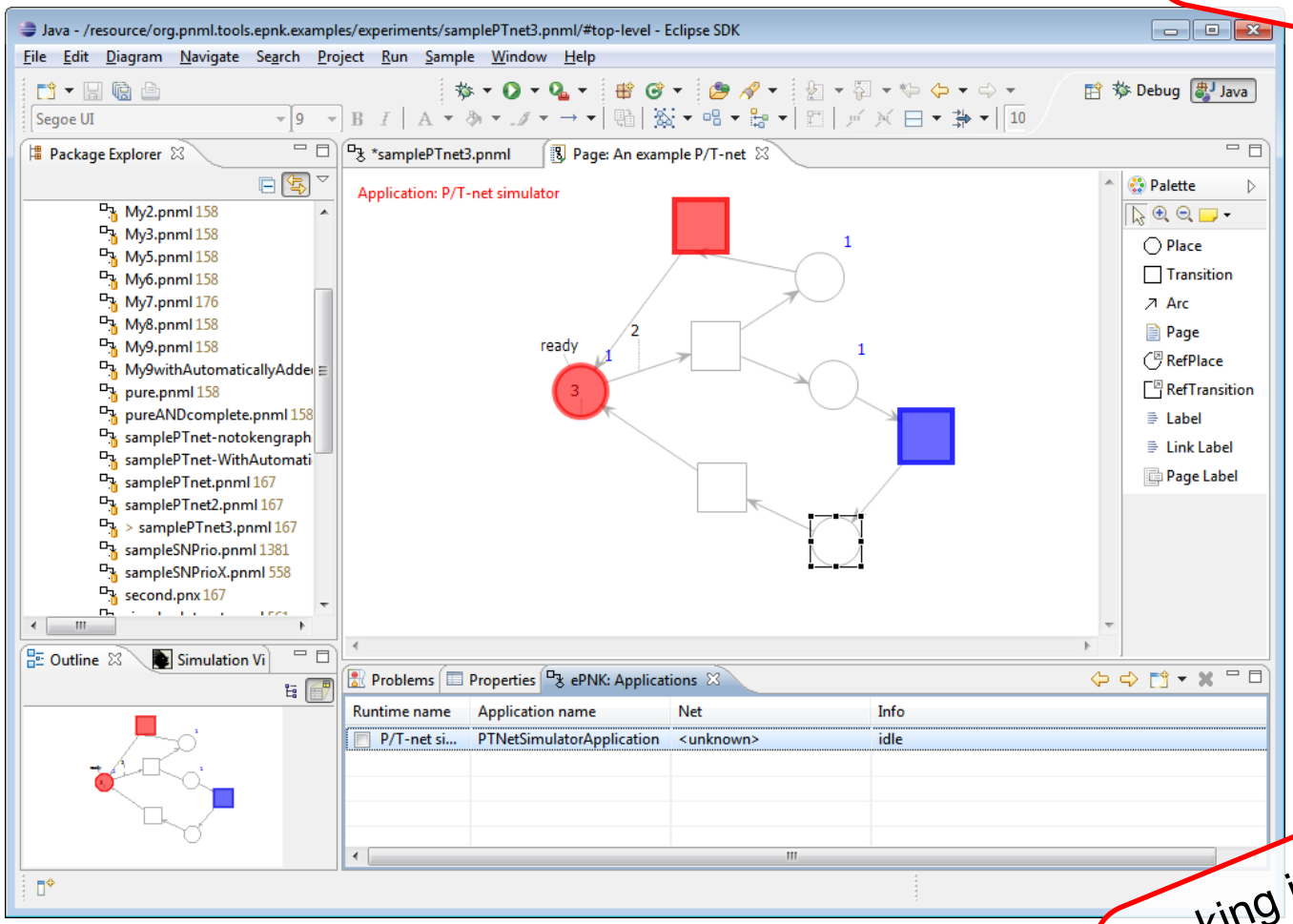
”The ePNK is a platform for Petri net tools based on the PNML transfer format. Its main idea is to provide generic Petri net types, which can be easily plugged into it, and to provide a simple generic GMF editor, which can be used for graphically editing nets of any plugged in type.” [ePNK Homepage]

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

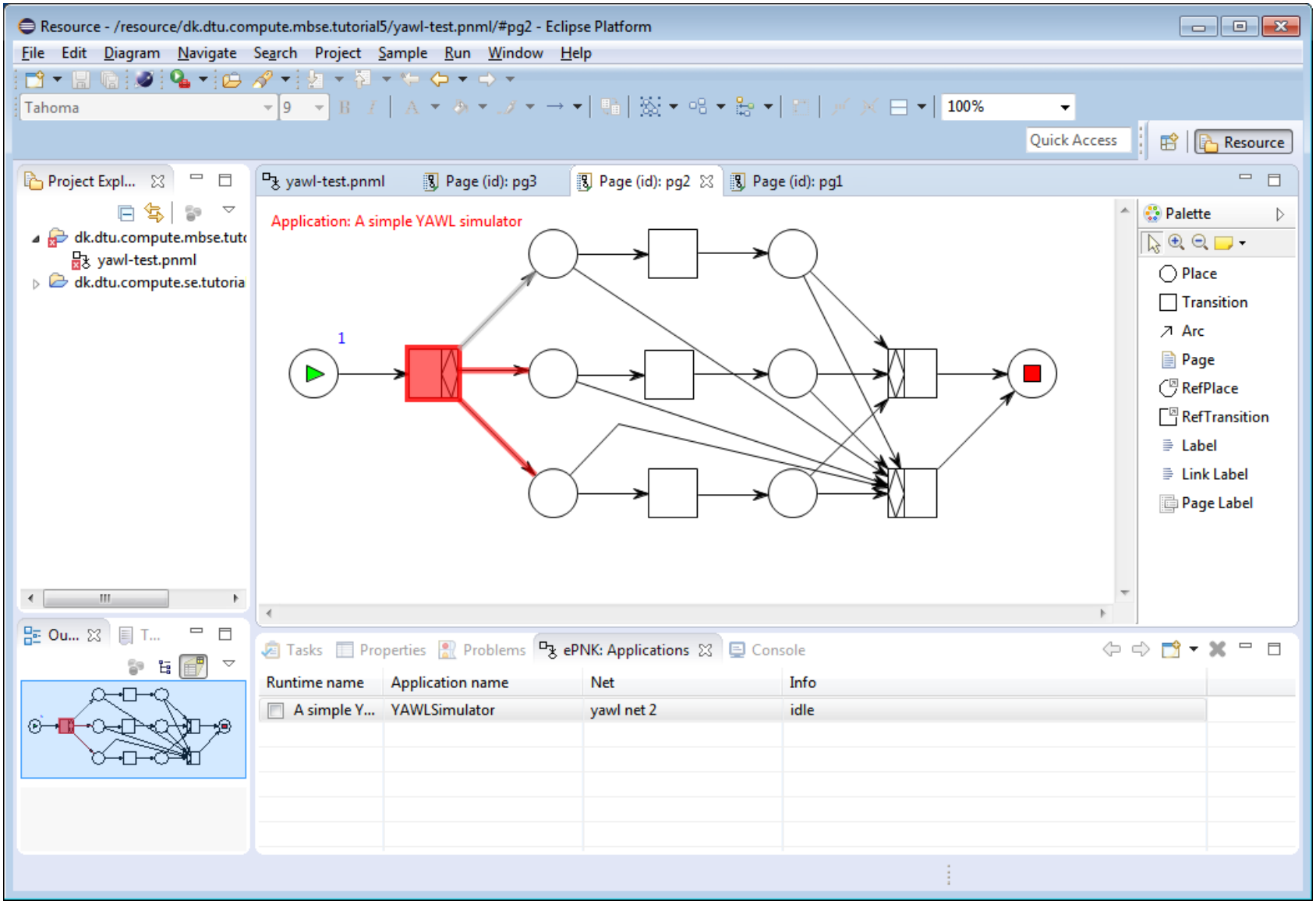
Screen shot of a simulator for ordinary Petri nets, which is deployed with the ePNK (with source code).



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

YAWL Tool (after Tutorial 8)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

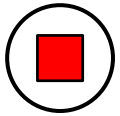


Scope

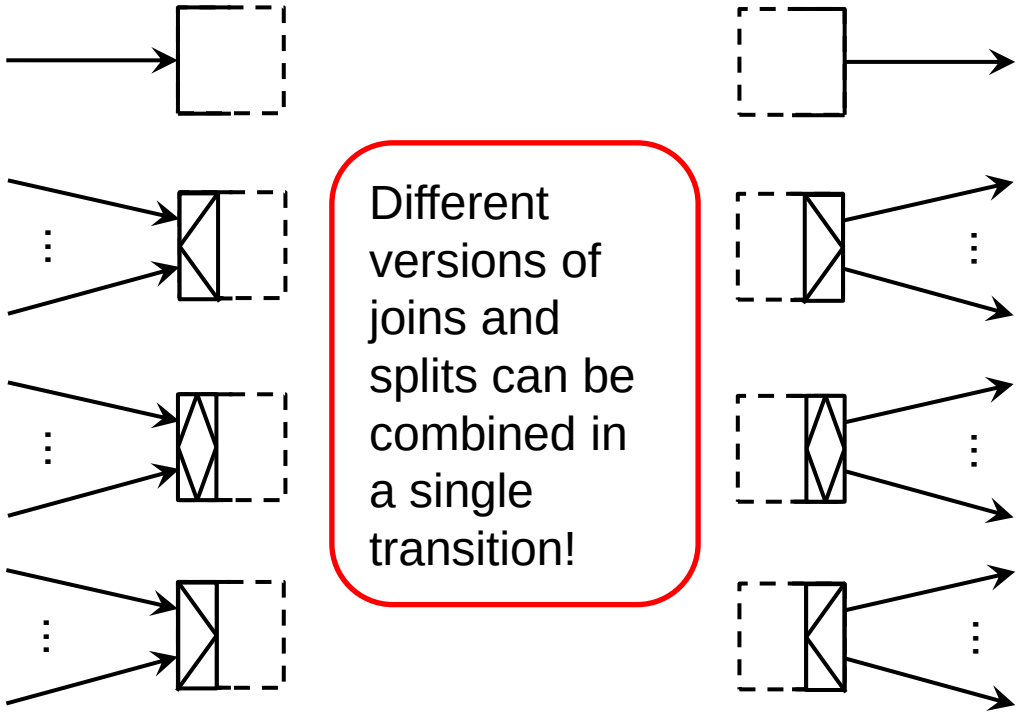
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

The tool must support the following YAWL features

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



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



single

XOR

OR

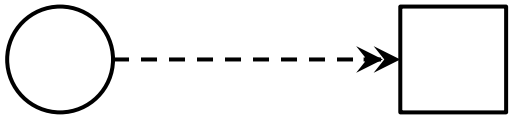
AND

Scope (cntd.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- 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)

2. YAWL (concepts)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

In this section, we introduce some additional notation for modelling business processes with Petri nets in a slightly more user friendly way: YAWL* (Yet another Workflow Language)

- XOR-split / XOR-join
- AND-split / AND-join

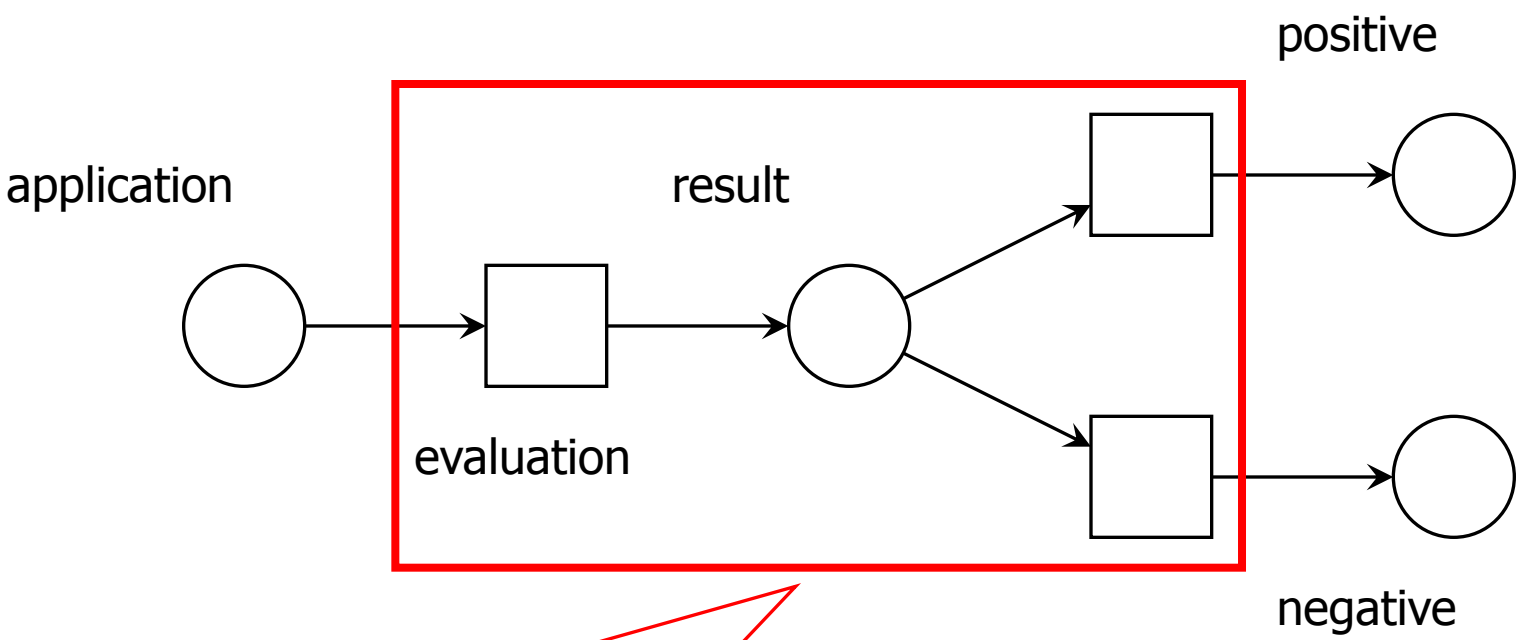
- OR-split / OR-join
- Reset-arcs

*) A slightly academic workflow notation!

In your project, you implement a graphical editor and a simulator for YAWL (the subset presented here and in the project slides)

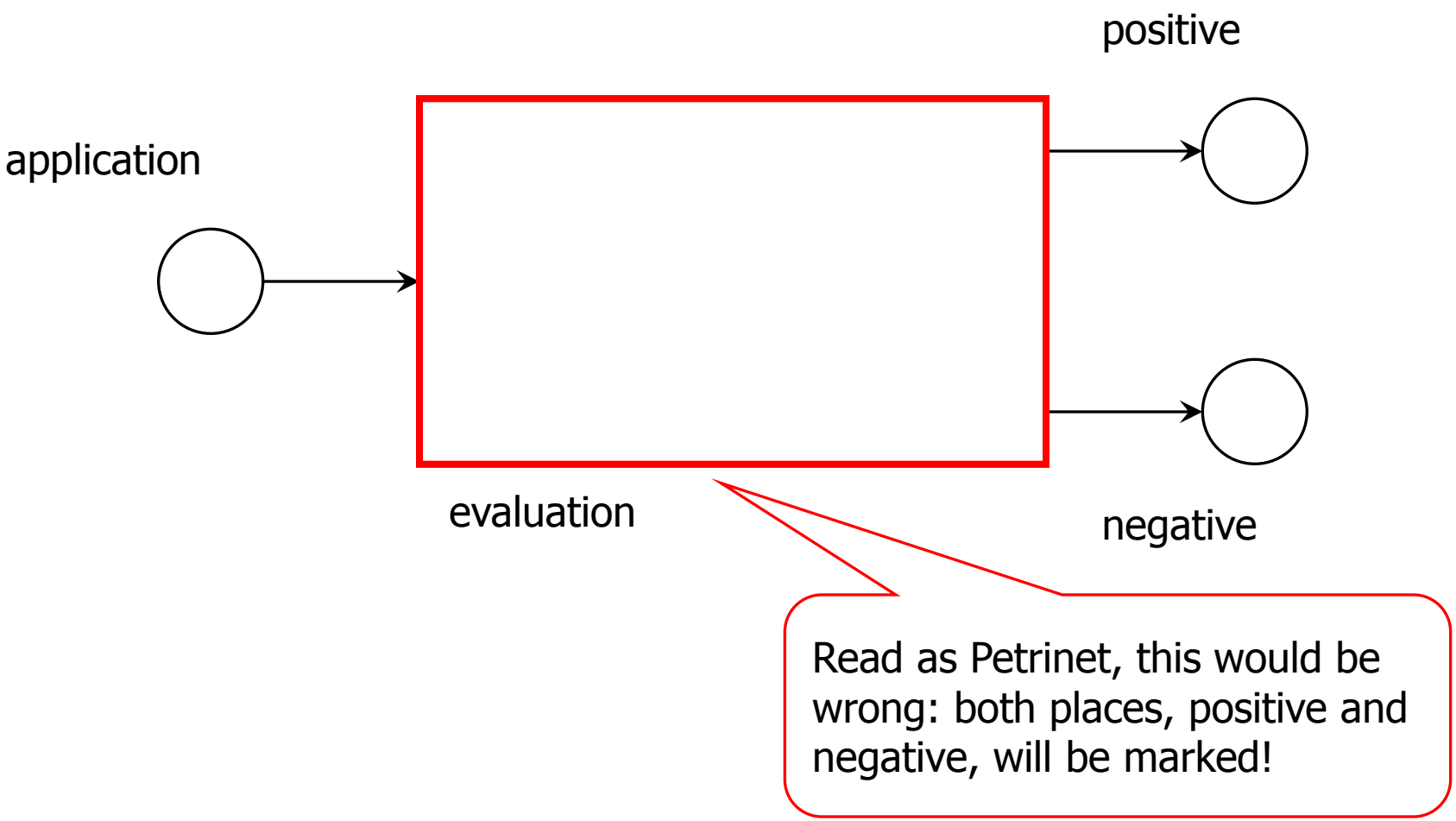
Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



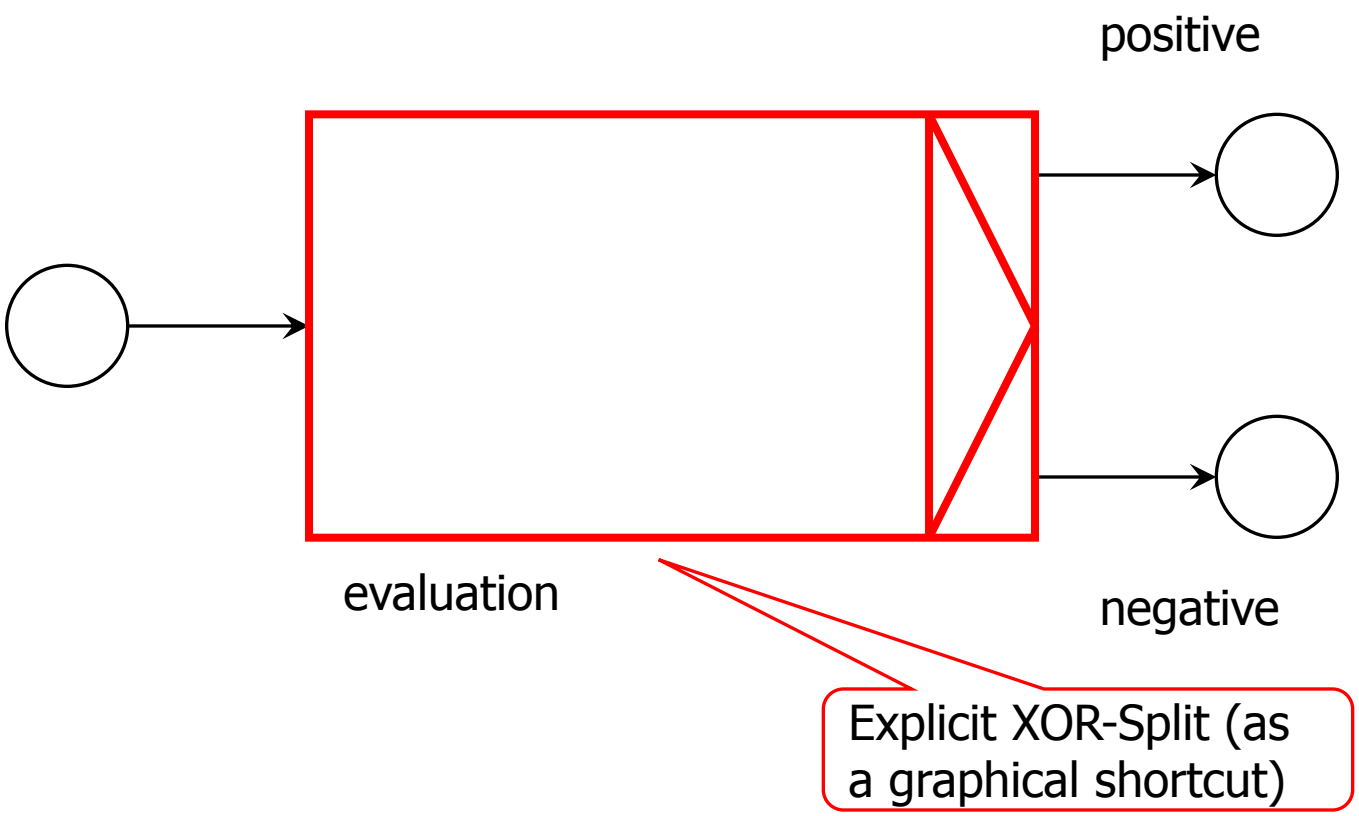
One task: two different outcomes: We would like to consider this as a single activity with two possible outcomes!

Problem: One Task – Two Outcomes



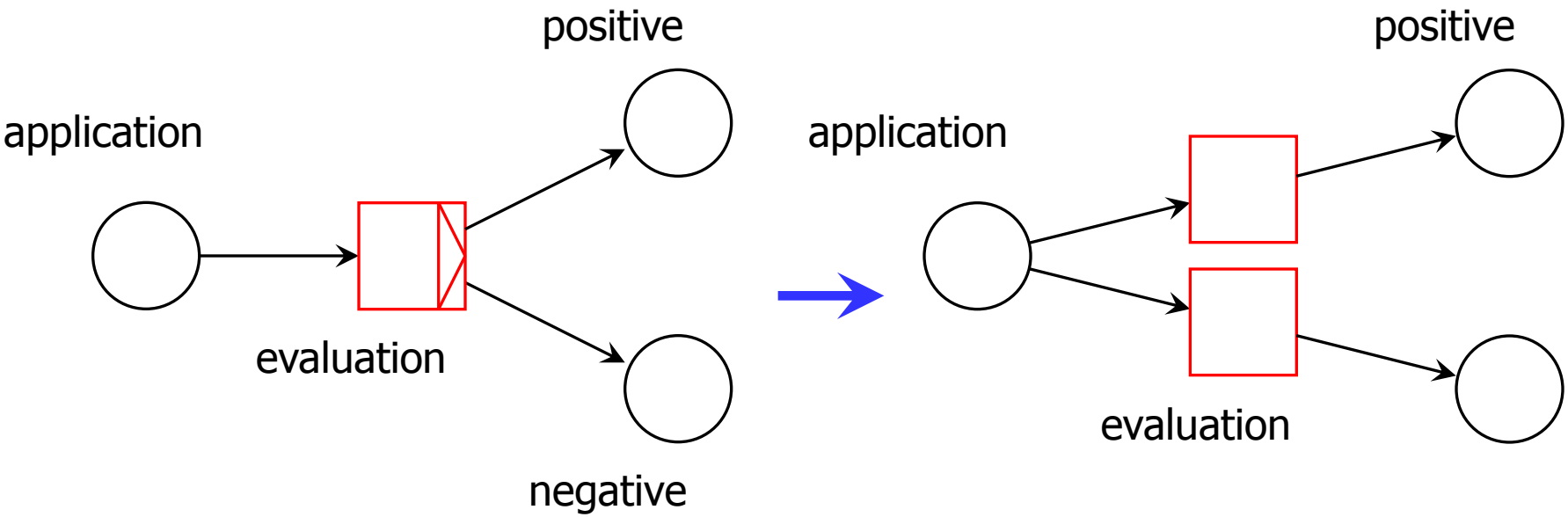
Solution: Explicit XOR-Split

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Petri Net Semantics for XOR-Split

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

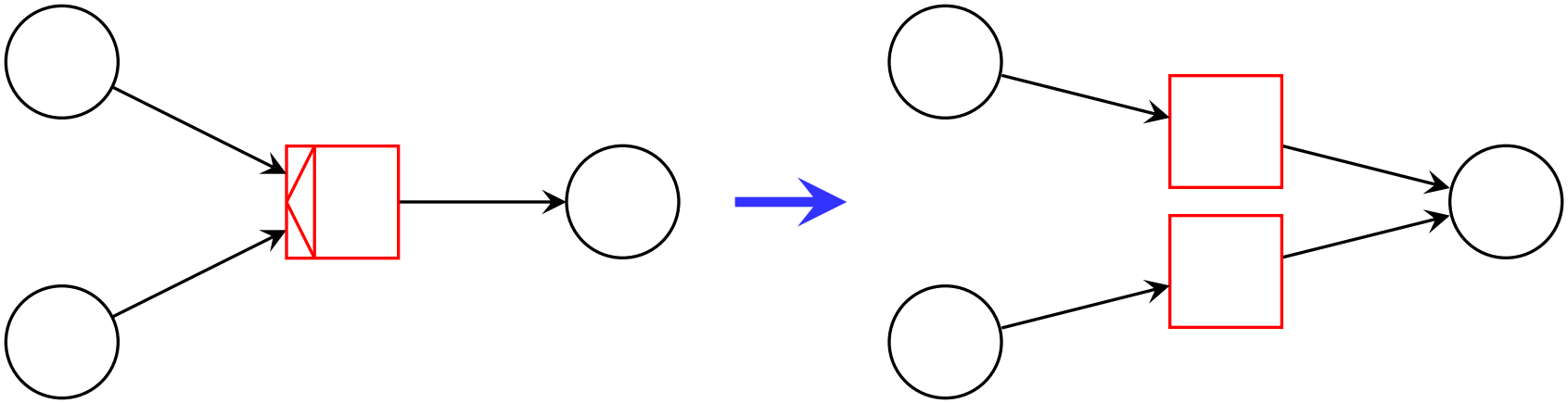


XOR-split

its Petri net interpretation

Petri Net Semantics for XOR-Join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

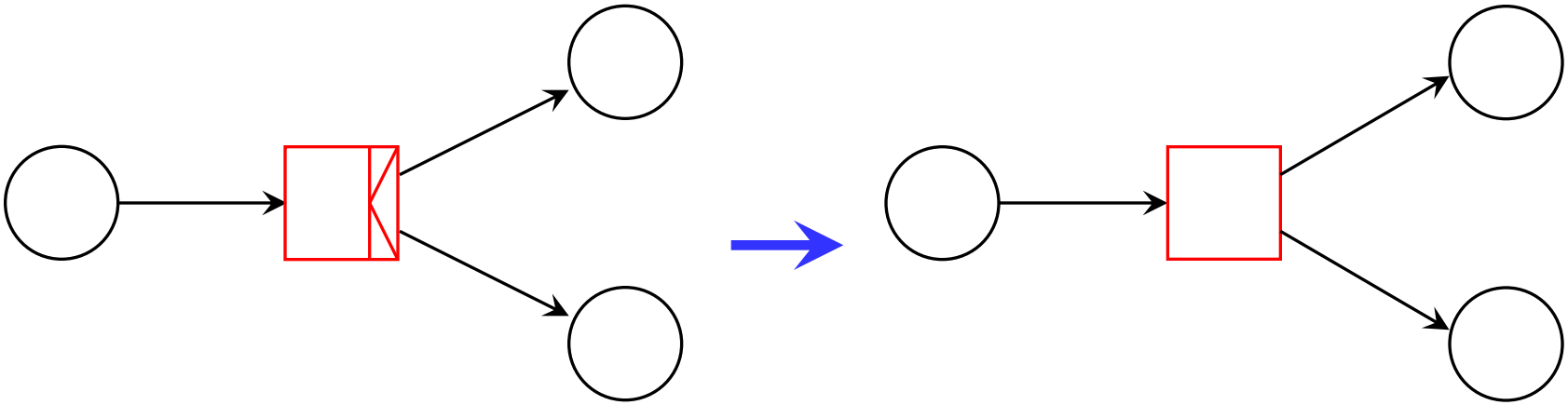


XOR-join

its Petri net interpretation

Petri Net Semantics for AND-Split

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



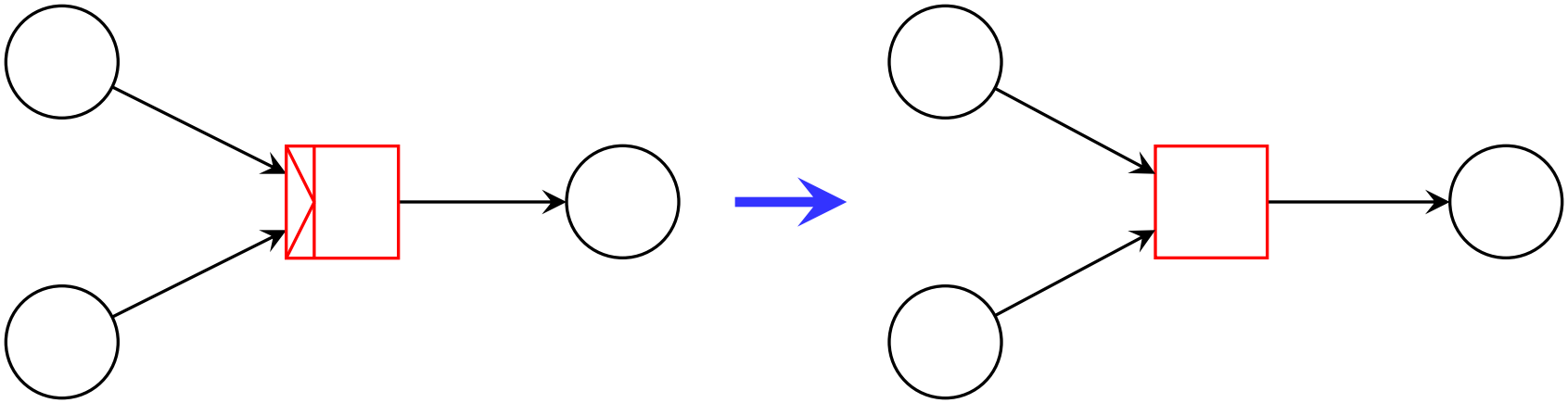
AND-split

its Petri net "interpretation";

a transition

Petri Net Semantics for AND-Join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



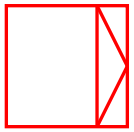
AND-join

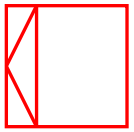
its Petri net "interpretation";

a transition

XOR-Split/Join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

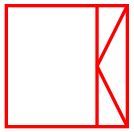


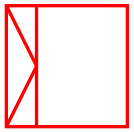


- An XOR-splits allows us to model an activity with different outcomes as a single „transition“
- An XOR-join allows us to model an activity with different preconditions as a single „transition“
- XOR-joins and XOR-splits correspond to conditional routing.

AND-Split/Join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





- AND-split and AND-join correspond to the usual Petri net transitions;
- They have been introduced for symmetry reasons only.
- AND-join and AND-splits correspond to parallel routing.

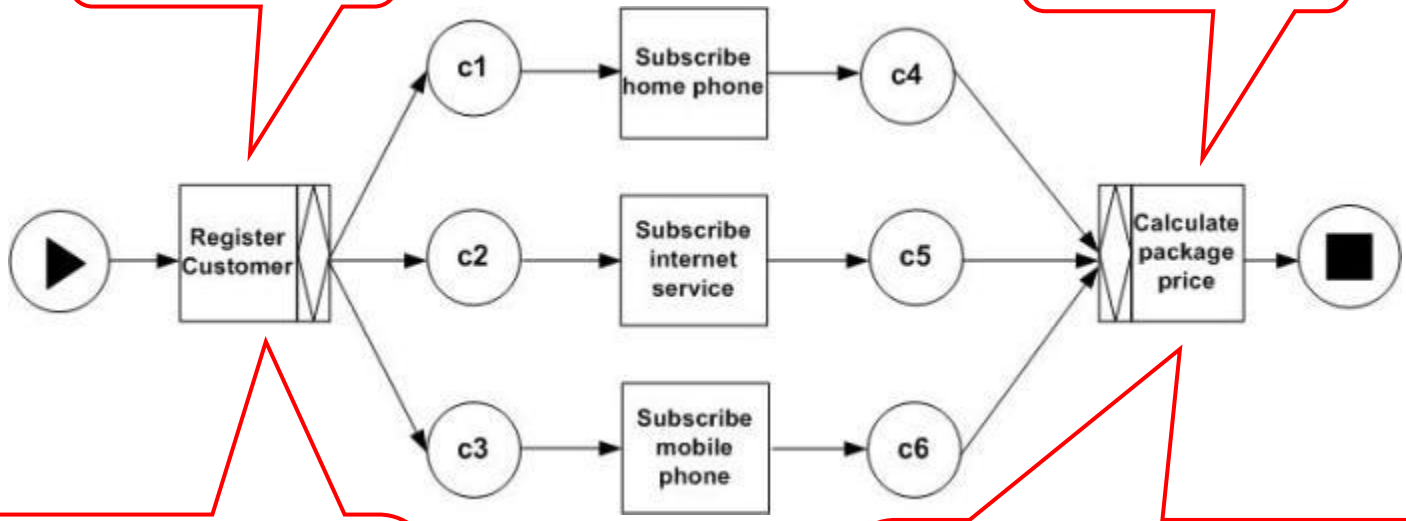
Example: OR-split/join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

OR-split

OR-join

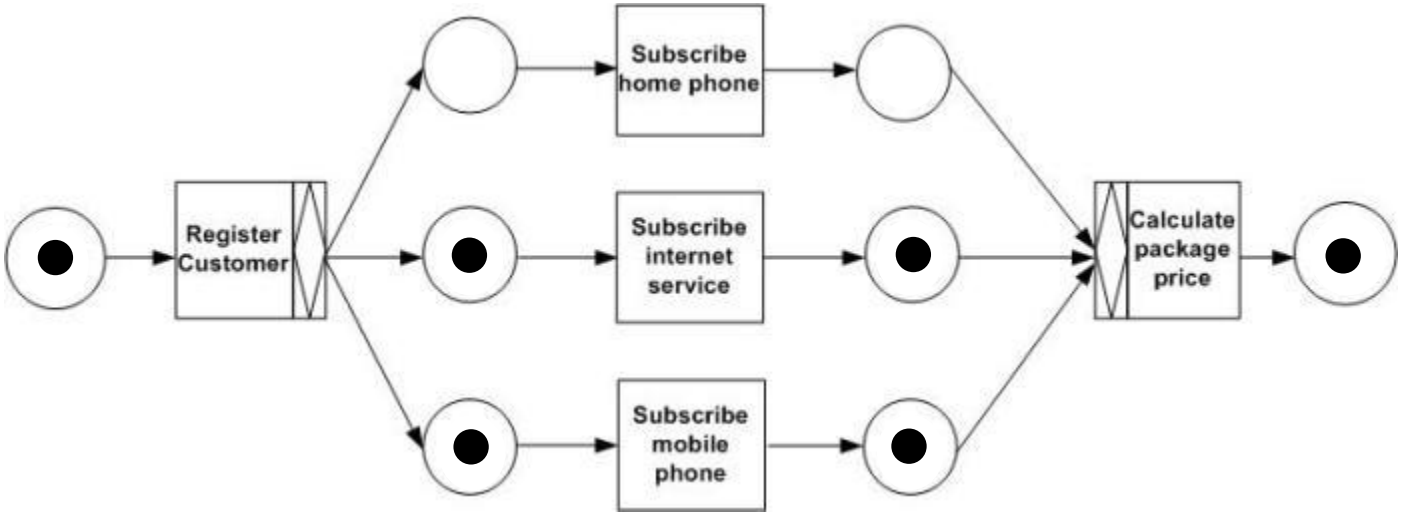


Adds token to some output places (at least one)

Needs tokens on at least one input place; removes one token for each place that has a token; but → slides 24-27

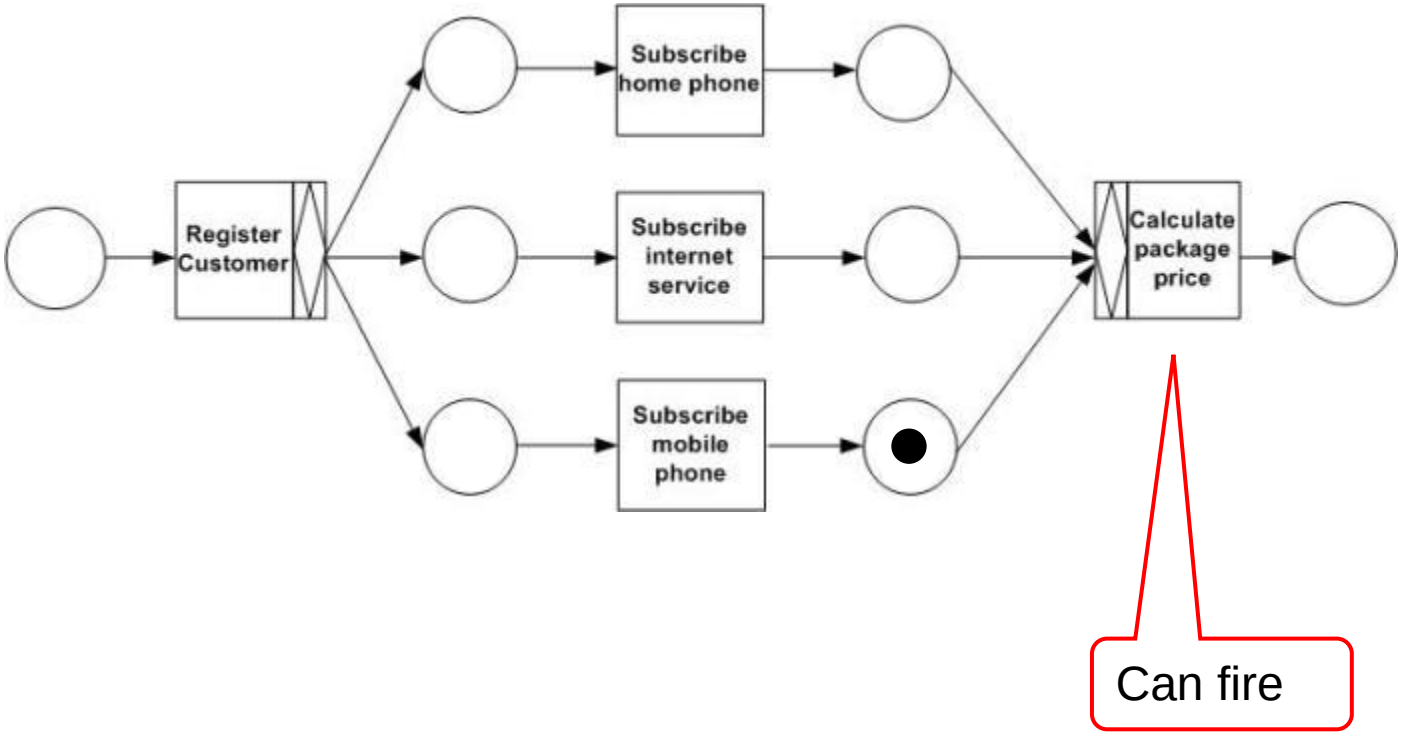
Example: OR-split/join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Example: OR-split/join

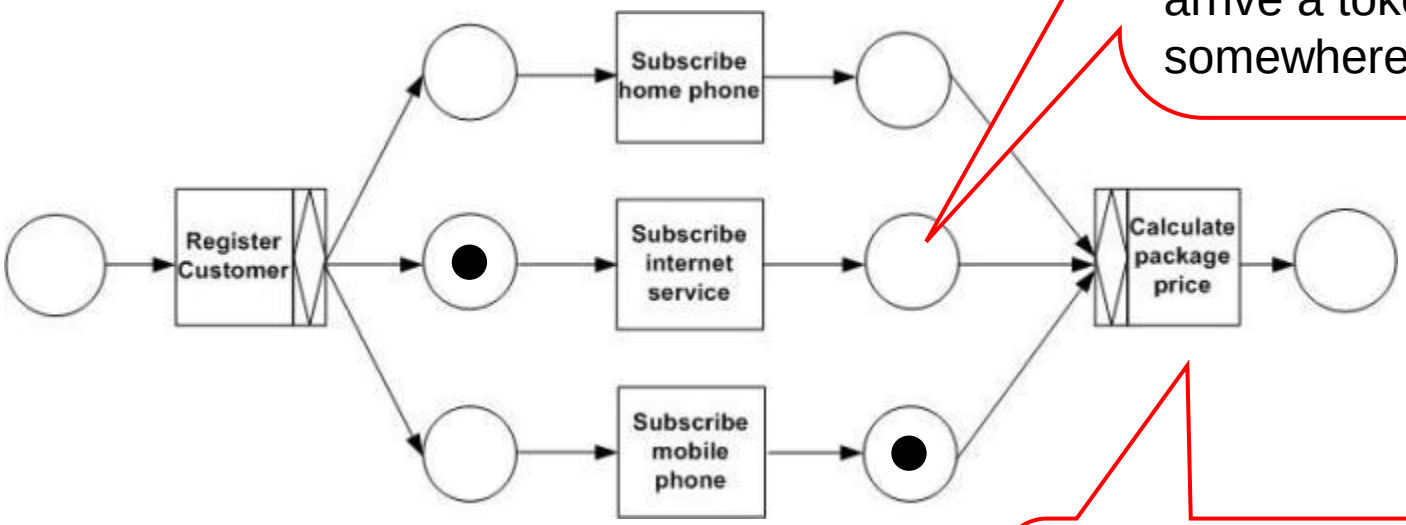
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Example: OR-split/join

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

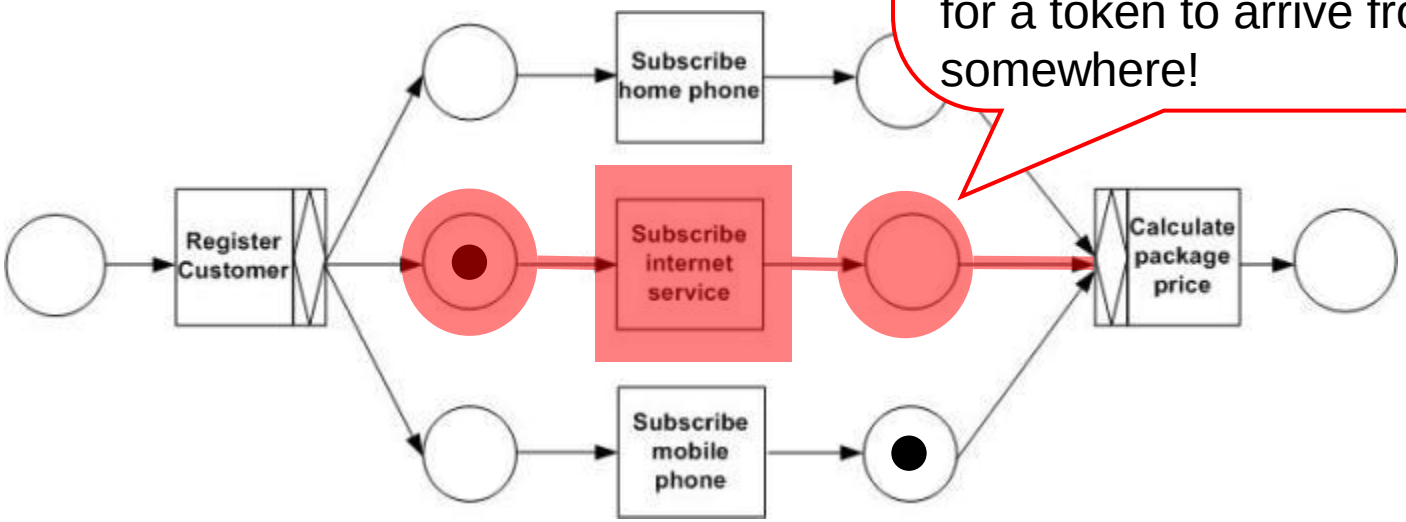
When an input place of an OR-join transition is not marked, the OR-join should not fire, if there still could arrive a token from somewhere!



Cannot fire! The transition should wait until the other token has arrived!

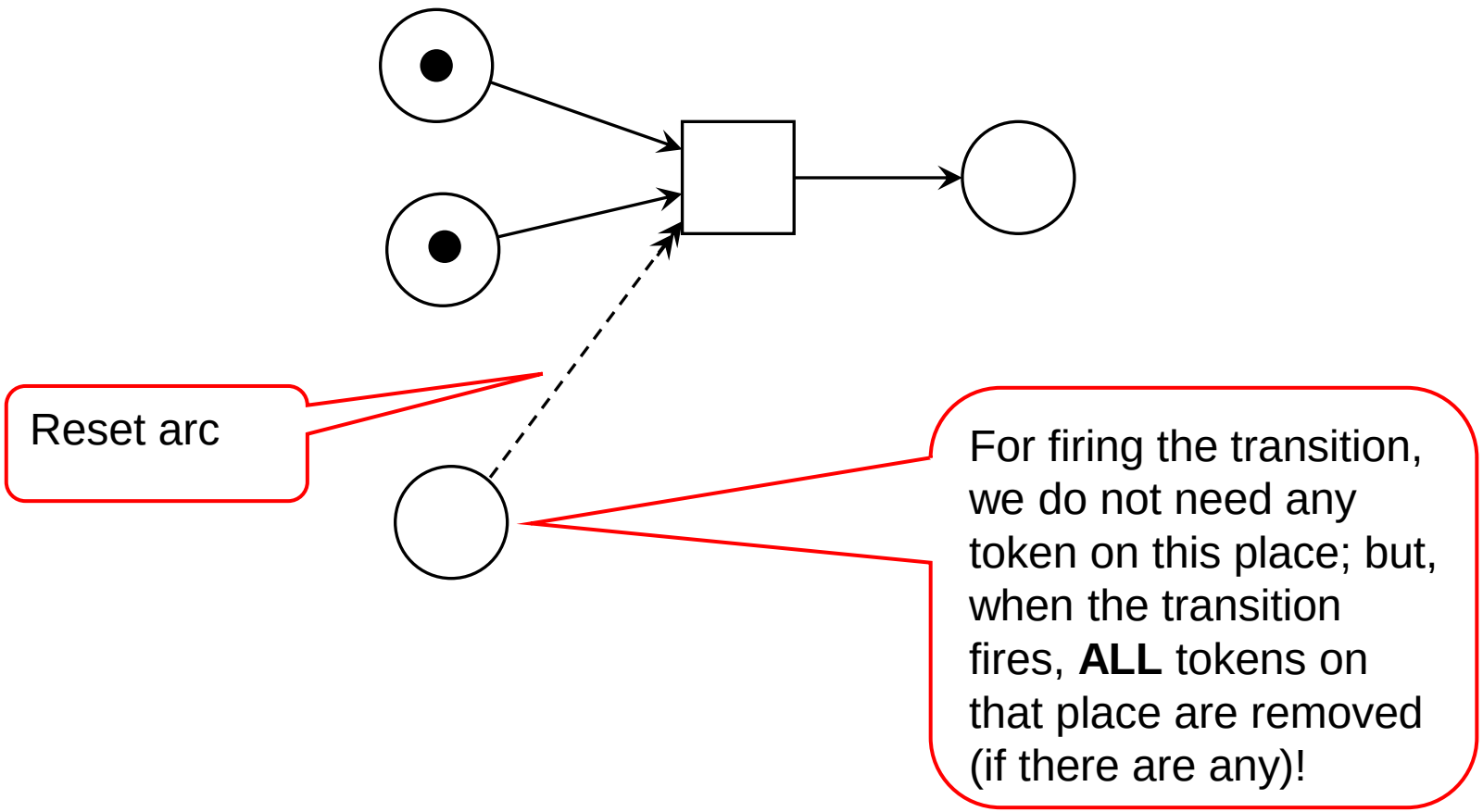
Example: OR-split/join

In complex examples, it is “a bit tricky” to decide whether a token could arrive at some place! But, a warning can be issued when firing the transition, that there is the potential for a token to arrive from somewhere!



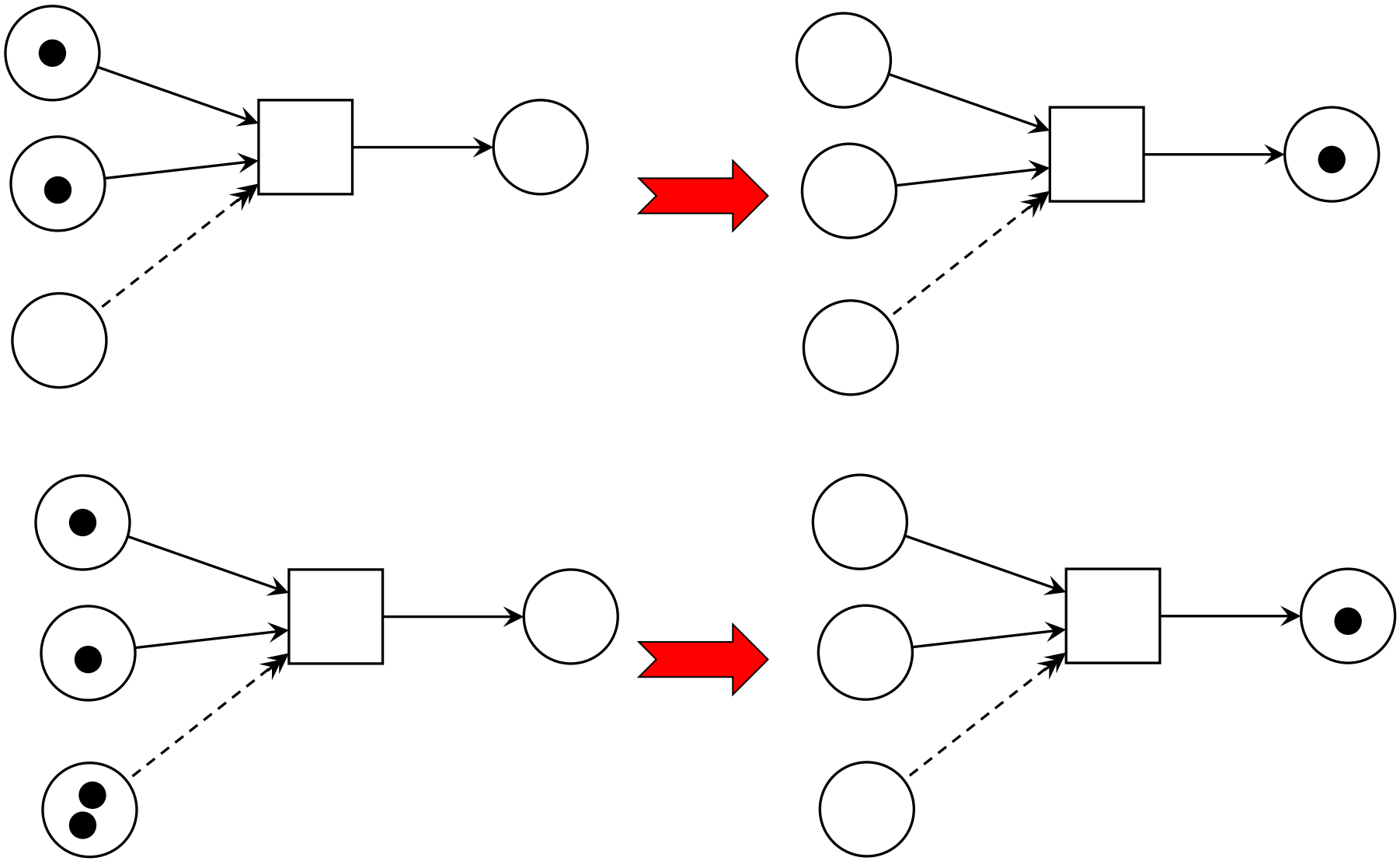
Reset-arcs

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Reset-arcs: Semantics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



On Writing Well

Headline “borrowed” from the book
William Zinsser: On Writing Well
(ed. from 1976 - 1998)

We will discuss some example
project definitions together!

Motivation

- Writing good texts is hard work!

- Most of it can be learned and is more about the writer's attitude than about talent:
  - What is the purpose?
  - What do I want to achieve?
  - Who is the reader?
  - How do I achieve my goals?

Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Problems

- The readers can't ask the writer
- The writer must foresee possible questions and misunderstandings
  (and take care of them)

- The writer should not assume too much
- The writer should not make implicit assumptions or conclusions

Rule of thumb:
Don't assume anything. But, don't tell the reader that he is stupid.

Comprehensibility

When is a text comprehensibility?

Are there criteria for comprehensibility?

Langer, Schulz von Thun, Tausch:
„Sich verständlich ausdrücken!“

Criteria

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Simplicity (-- - 0 + ++)
  - simple words
  - simple sentences
  - short sentences
  - concrete (e.g. by example)

→ Inductive vs. deductive

- Structuring (-- - 0 + ++)
  - one idea after the other
  - form and content are coherent
  - conclusive

Criteria

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Conciseness (-- - 0 + ++)
  - shortness
  - focussed on essentials
  - no empty words and sentences

- Inspiring Additions (-- - 0 + ++)
  - motivating
  - interesting
  - diversified

Important issues

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Set the scene / context:
  Don't assume anything (except readers pragmatics) for granted
- Different levels of abstraction:
  Typical student mistake: always on the lowest level!!
- Guide the reader:
  Why do you say what you are saying
- Bring the point (argument) home – **completely**!

- "**Spiralform writing**": → blackboard
  Writing linearly about a complex network of concepts

More rules (of thumb)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Important stuff first / high-lighted
- strong verbs (avoid adjectives / adverbs)
- short sentences
- use singular whenever possible
- familiar terms and expressions
- use “active” wherever possible
- clear headlines
- …

Be critical about your own texts!

Comprehensibility

- The above criteria hold for almost all texts
- For scientific texts:
  - consistent terminology (same term for same concept throughout the text):

    My favourite **counter example** „Deutscher Fußballreporter“:
    Ball, Rund, Kulle, Leder, Ding, …
  - Same structure for alike structured content

Architecture (discussed on blackboard)

DTU Compute
Department of Applied Mathematics and Computer Science

MVC

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler