Model-based Software Engineering for/with Petri nets

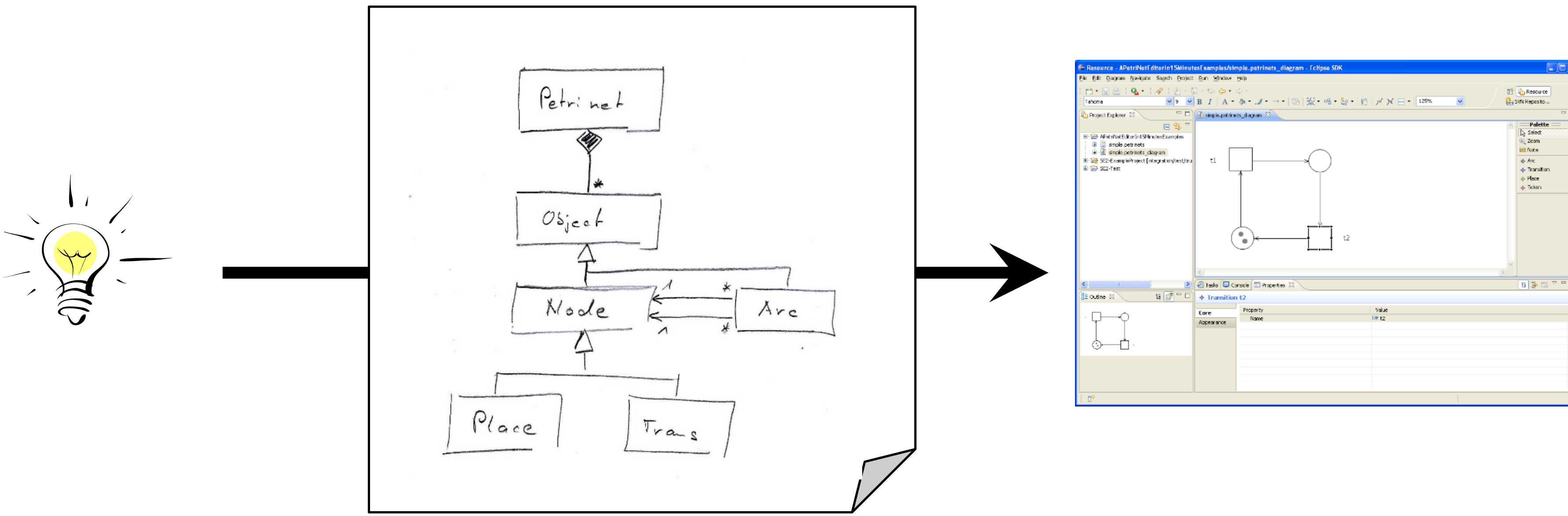
Behaviour Modelling (and its challenges)

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science

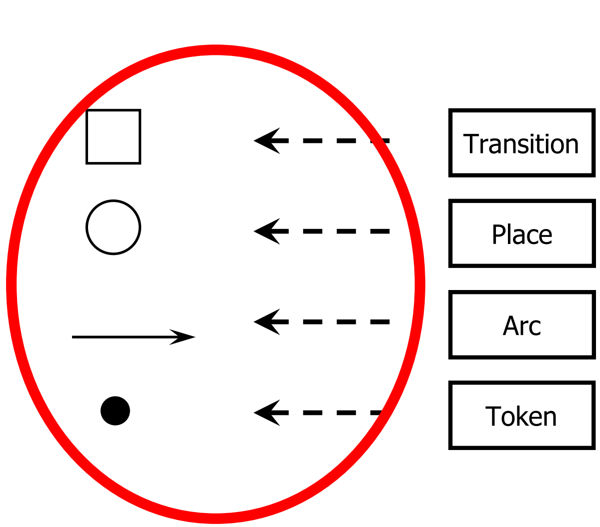
Motivation

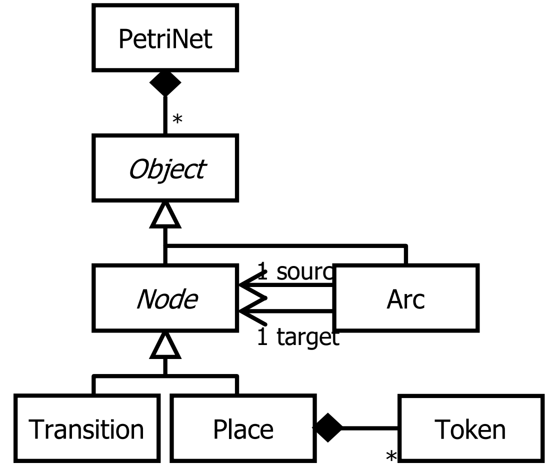
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

meta model

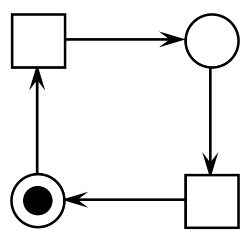


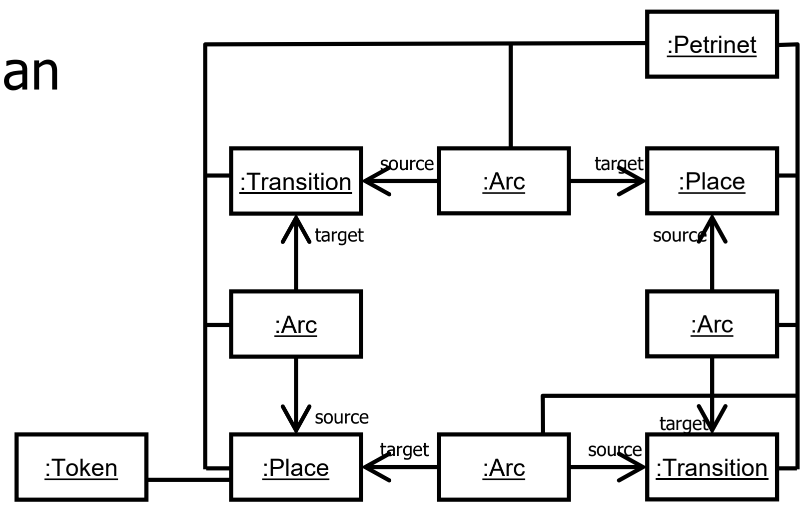


is instance of

generate an editor

model





concrete syntax

abstract syntax

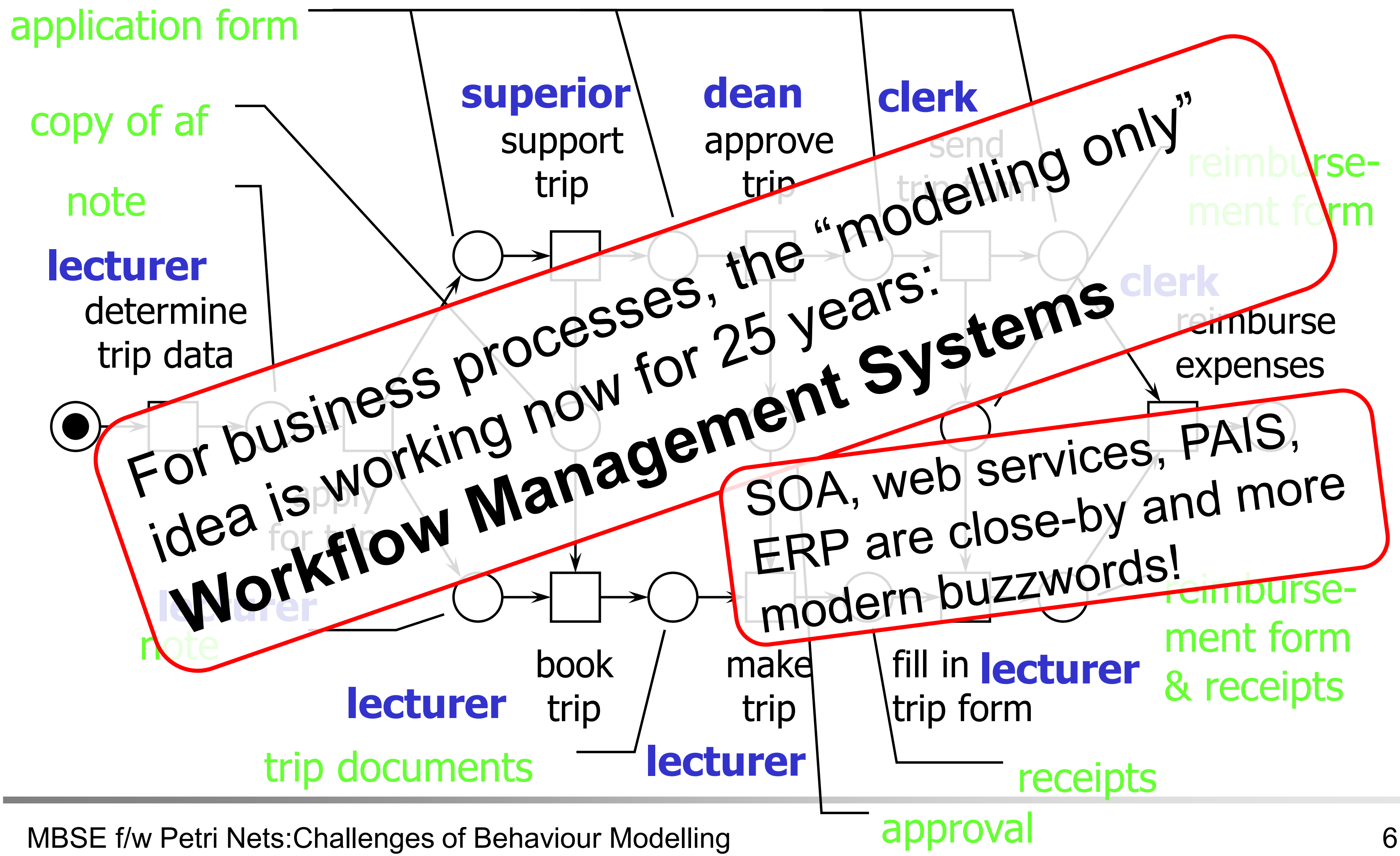
Benefits of Modelling (cntd.)

- Better Understanding
- Mapping of instances to XML syntax (XMI)
- Automatic Code Generation
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)
  - Editors and GUIs

But, all this is "standard functionality"! Where is "real functionality" or non-standard behaviour?

Table of Contents

Example: Business Trip



DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Where does the “modelling only” idea work?

Graphical editors

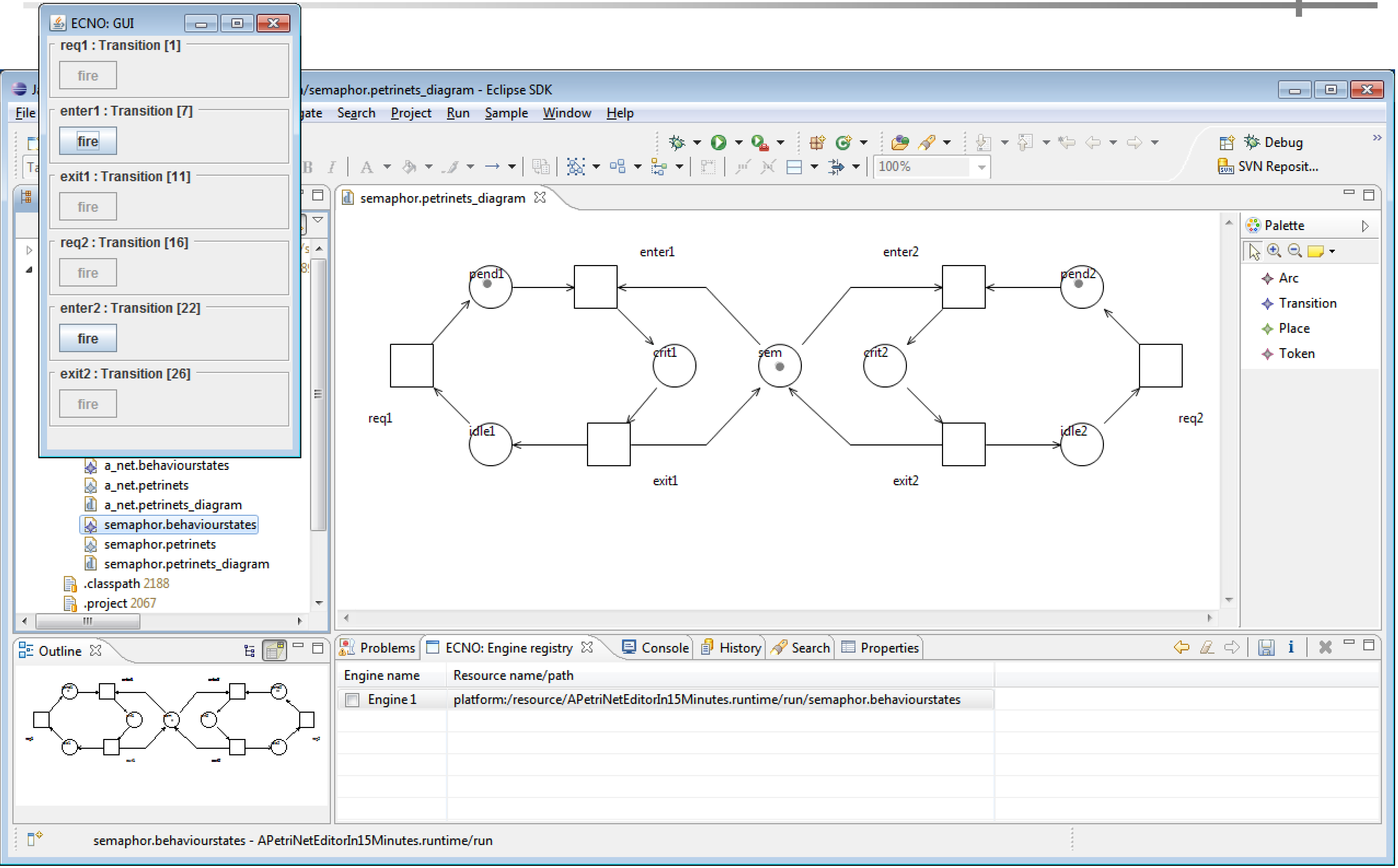
- graphics only
- standard functions only
- no business logic

Workflow management

- standard GUI only
- business logic explicitly modelled
- dedicated modelling notation

e.g. a Petri net simulator?

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Challenges

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

~~"There are no notations for modelling behaviour!"~~

This claim is actually as wrong as it can get!
There are (too?) many such notations!

Challenges

- Adequate modelling methodologies
  - Coarse grain behaviour
  - Fine grain behaviour
- Mechanism for integrating and coordinating behaviour beyond invocation (calls of procedure, function, method, or service)
- Integration with
  - existing software (legacy, manually created, generated)
  - other models (structural & behavioural)
- Change mentality (change culture)
  - Stuck with thread- and invocation-based thinking
  - Software engineering is programming thinking (→ model interpretation vs. code generation)

Moreover: Today's modelling technologies are not very "agile"!

Table of Contents

- Motivation
  - Model-based Software Engineering
  - Business process modelling
- Challenges and Problems
- Some Ideas
  - modelling behaviour
  - integration and coordination of behaviour
    (a vision: Event Coordination NOtation ECNO)
- Discussion

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Motivation

- Given some object oriented software with (or without) explicit domain model,
- model behaviour on top of it – and make these models executable.
- Model behaviour on a high level of abstraction (domain): coordination of behaviour

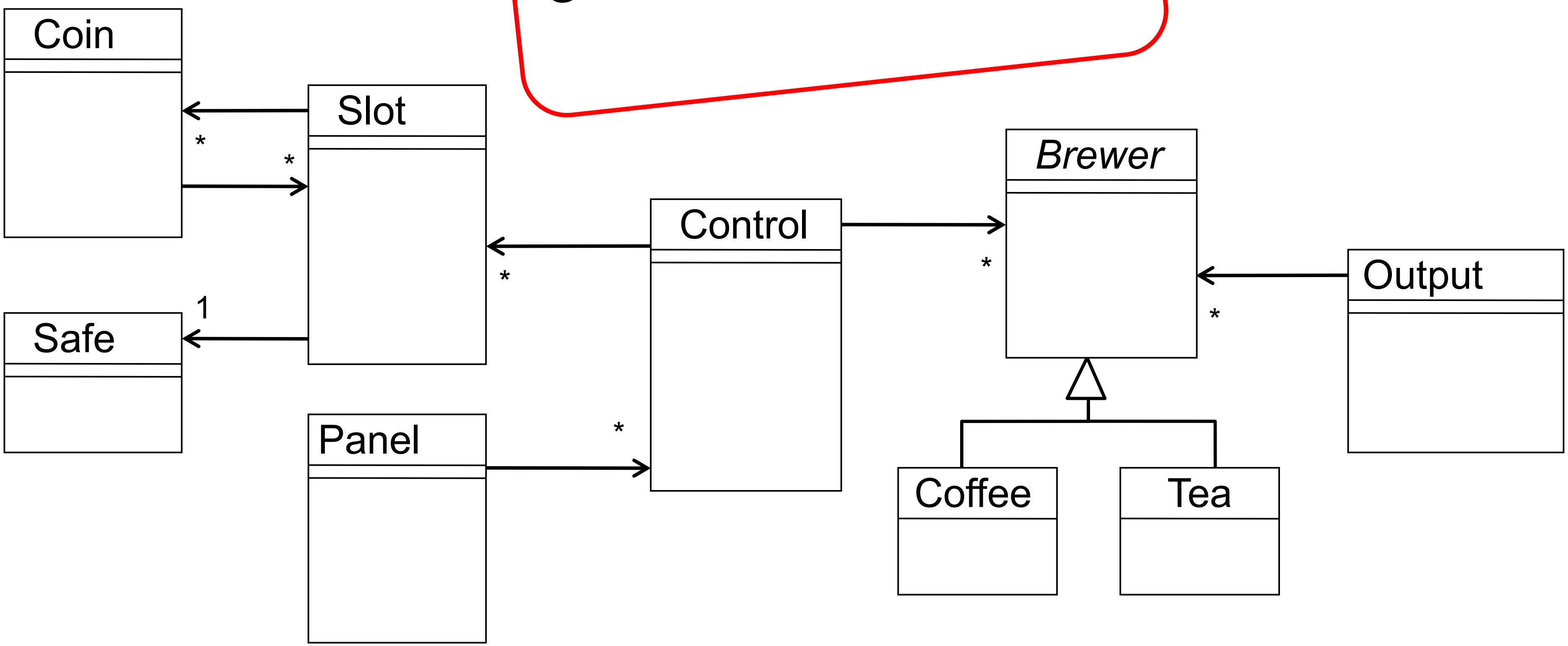
Meta-models / domain models including behaviour!

→ Integrate behaviour models with structural models

→ Integrate different structural models and manually written code (or code generated by different technologies)

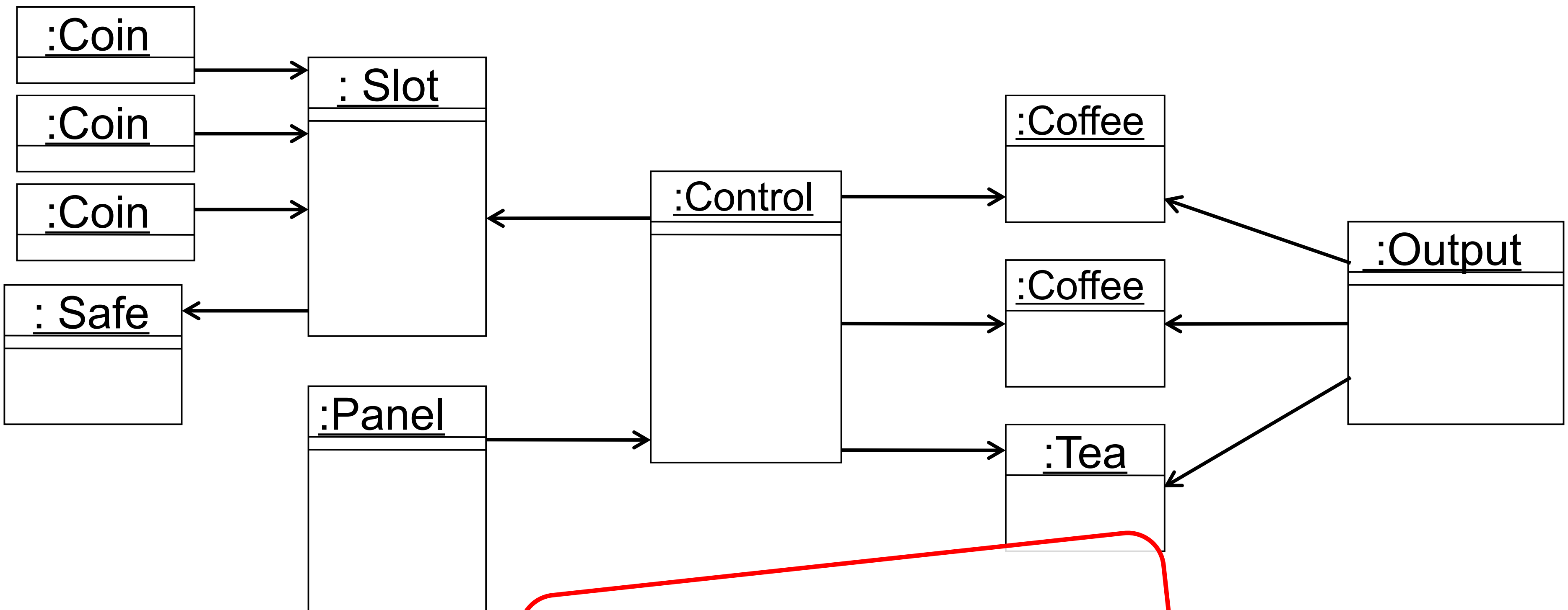
2.1 Example: Vending machine

Class diagram as usual



Instance: Object Diagram

Initial configuration, current situation



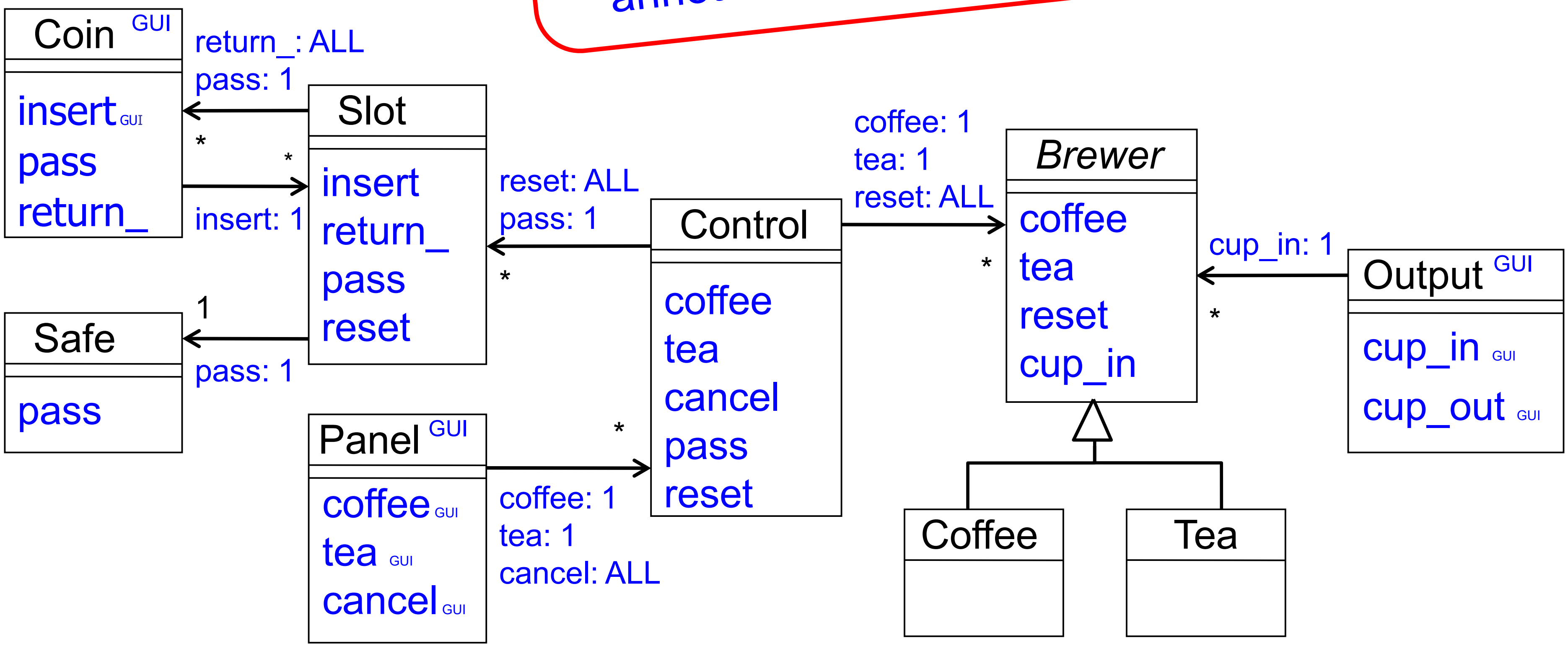
Object diagram as usual

Coordination Diagram

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- We call objects elements now!

- Events (event types)
- Coordination annotations: event type + quantification annotation



... + Event declaration

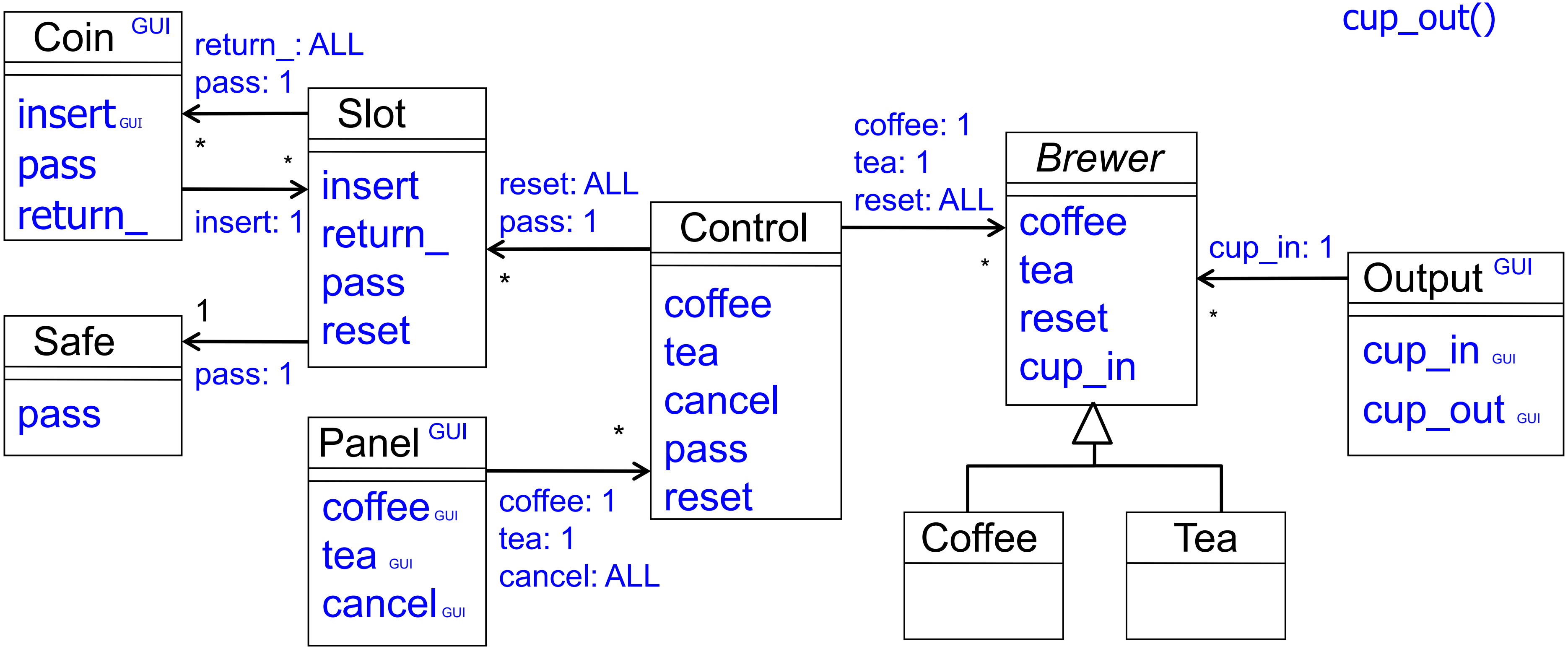
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Event (type) declaration
- Parameters

insert(Coin coin, Slot slot)
pass(Coin coin, Slot slot)
return(Slot slot)
reset_()

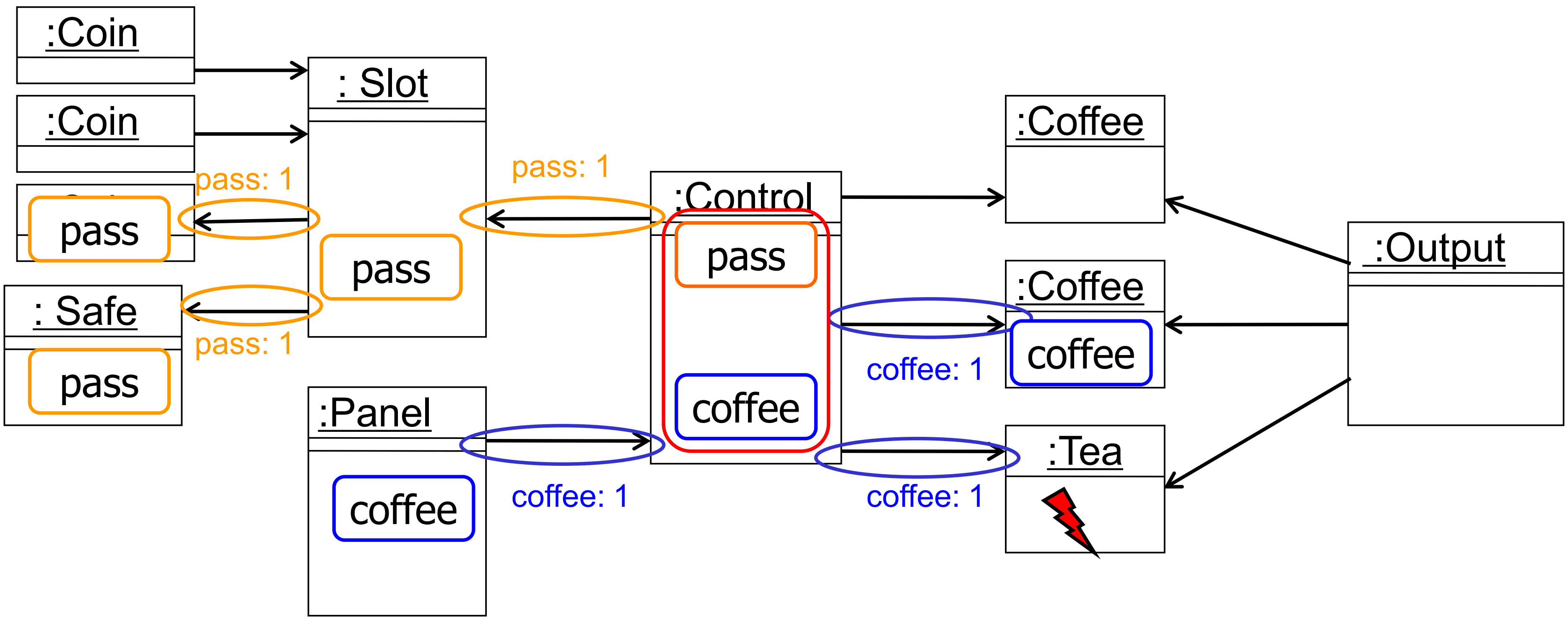
coffee()
tea()
cancel()

cup_in()
cup_out()



Interaction

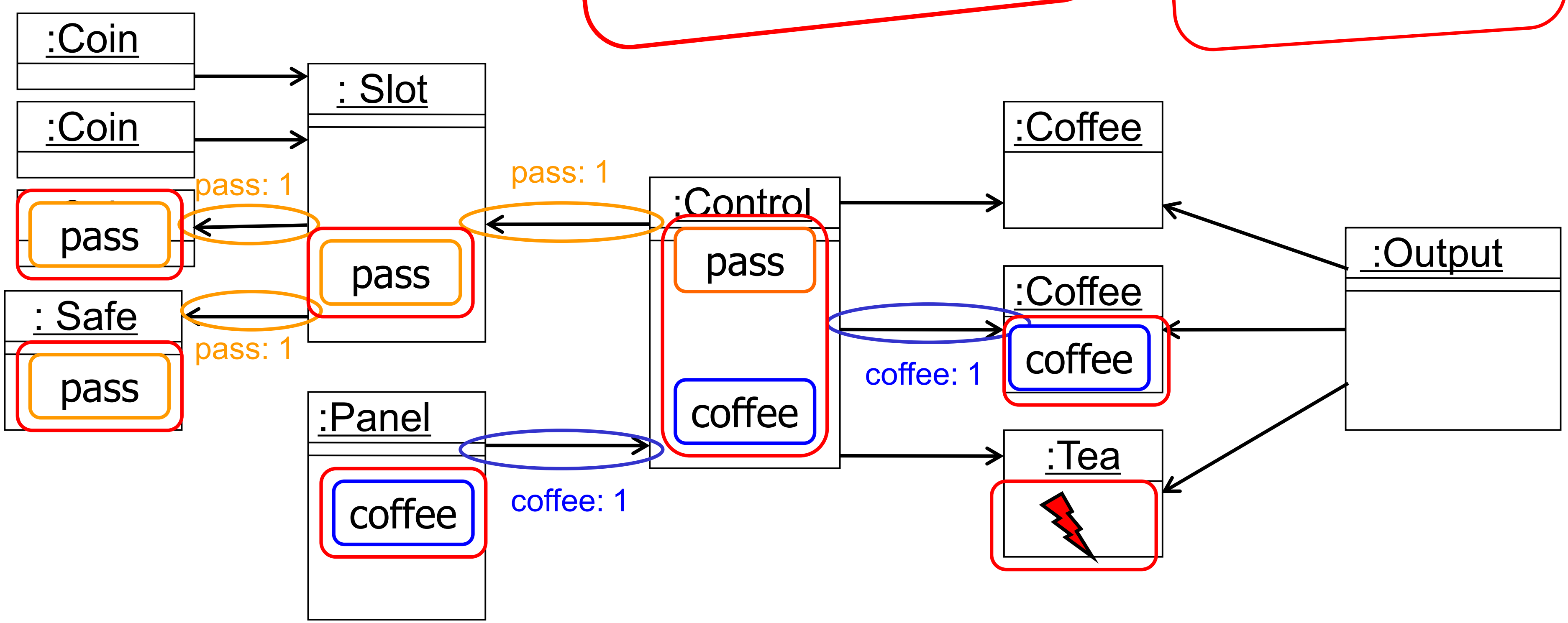
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



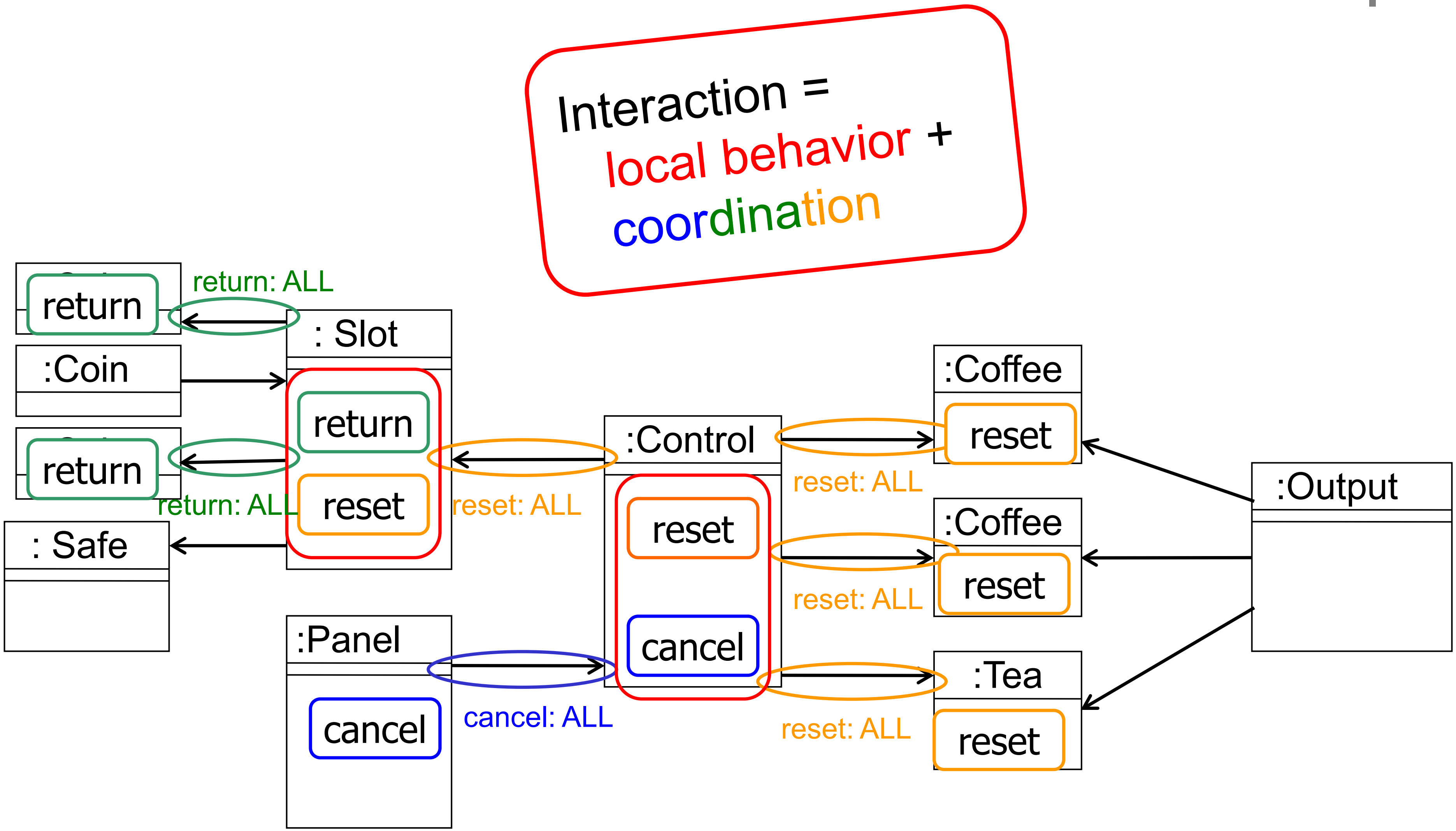
Interaction

Interaction =
local behavior +
coordination

We come back to
local behaviour!
→ slide 20



Another Interaction

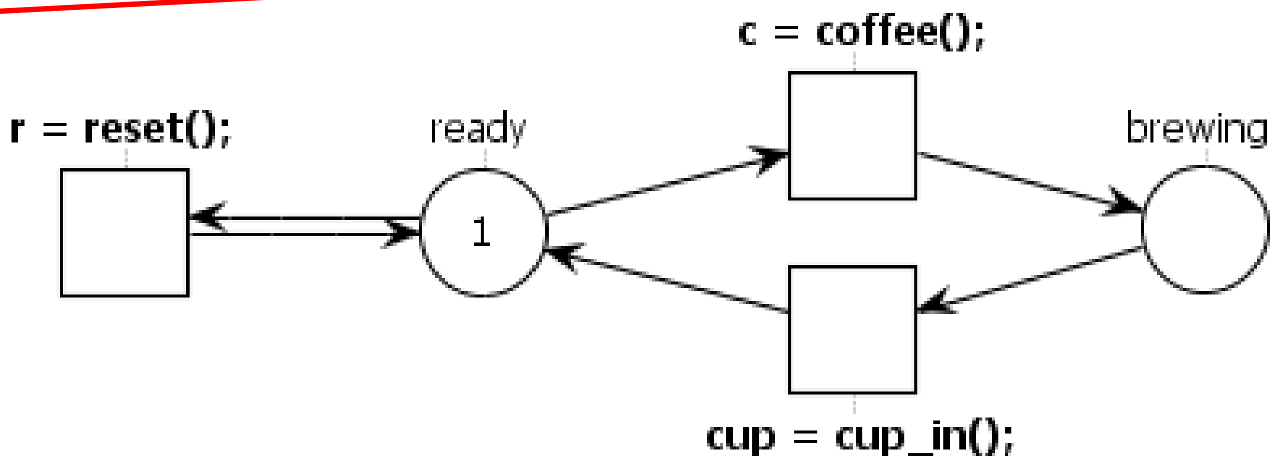


Local behaviour: Coffee

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

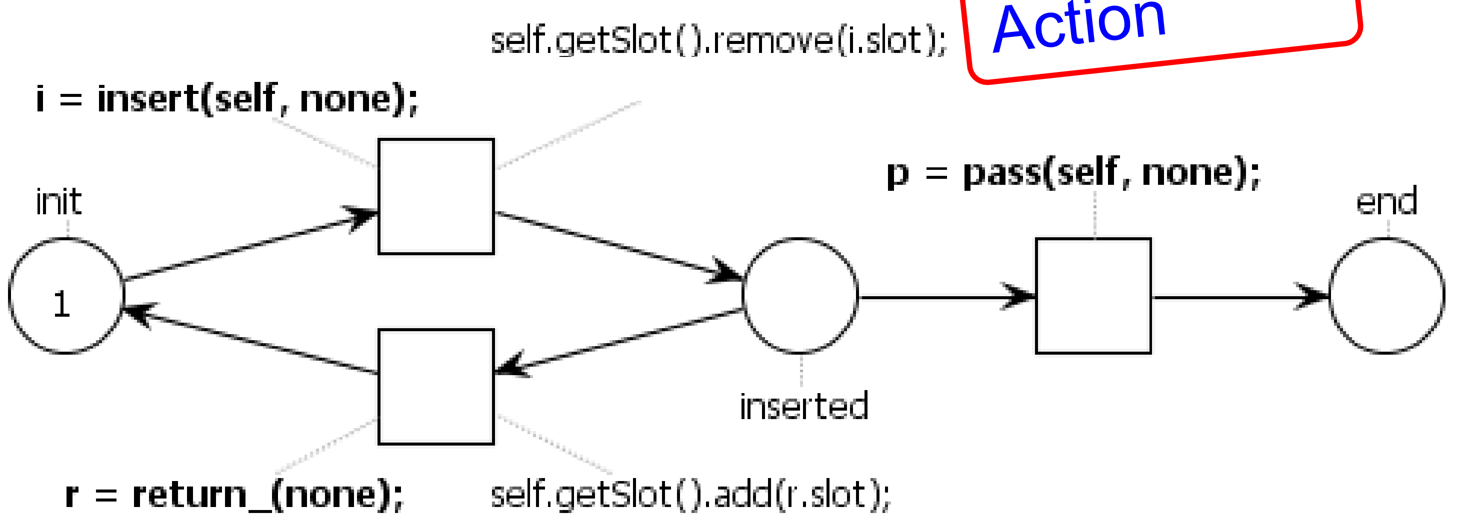
Elements are objects with an explicitly modelled **life-cycle**

Event binding



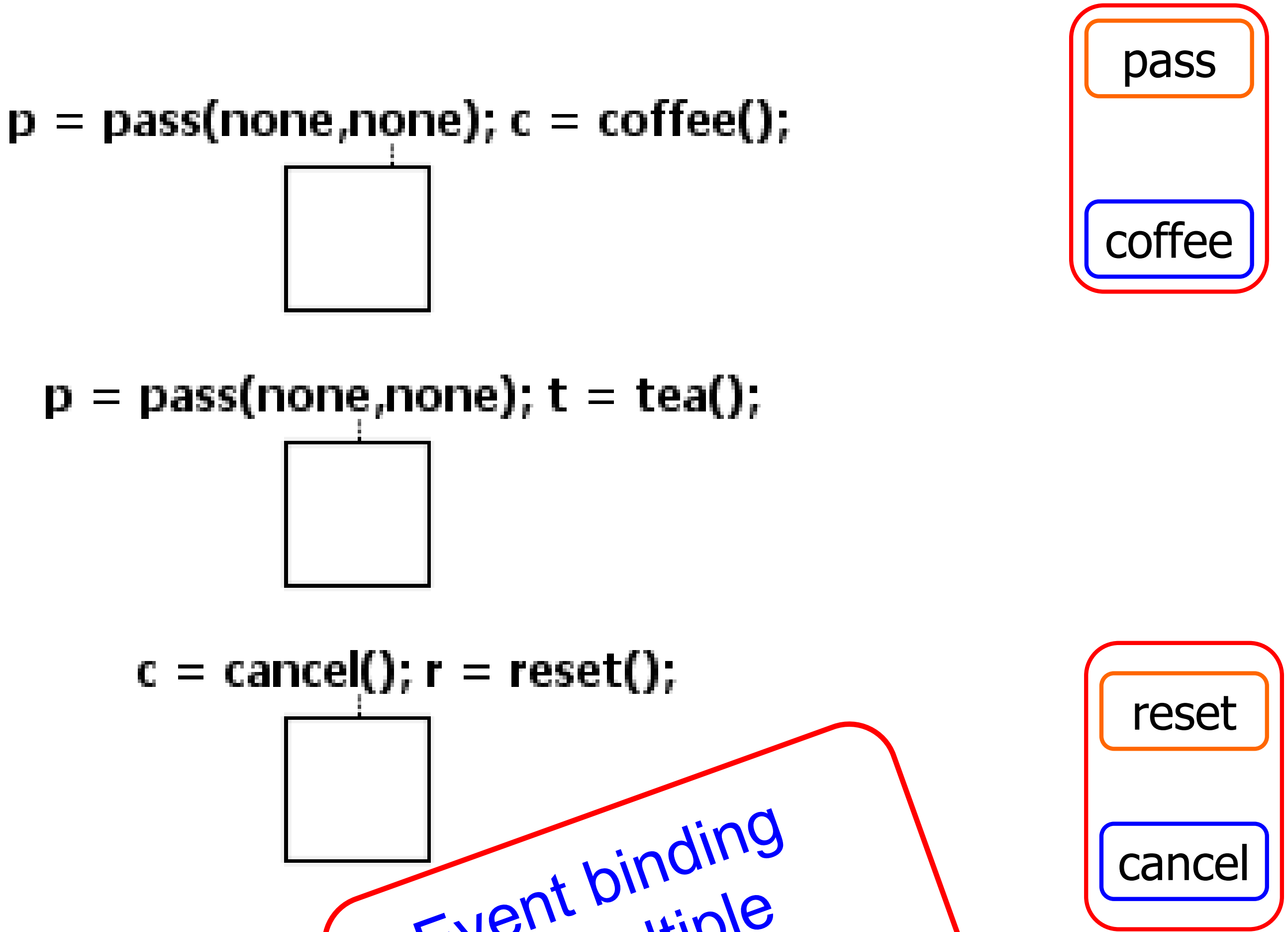
Local behaviour: Coin

Action



- Event binding
- Parameter assignment

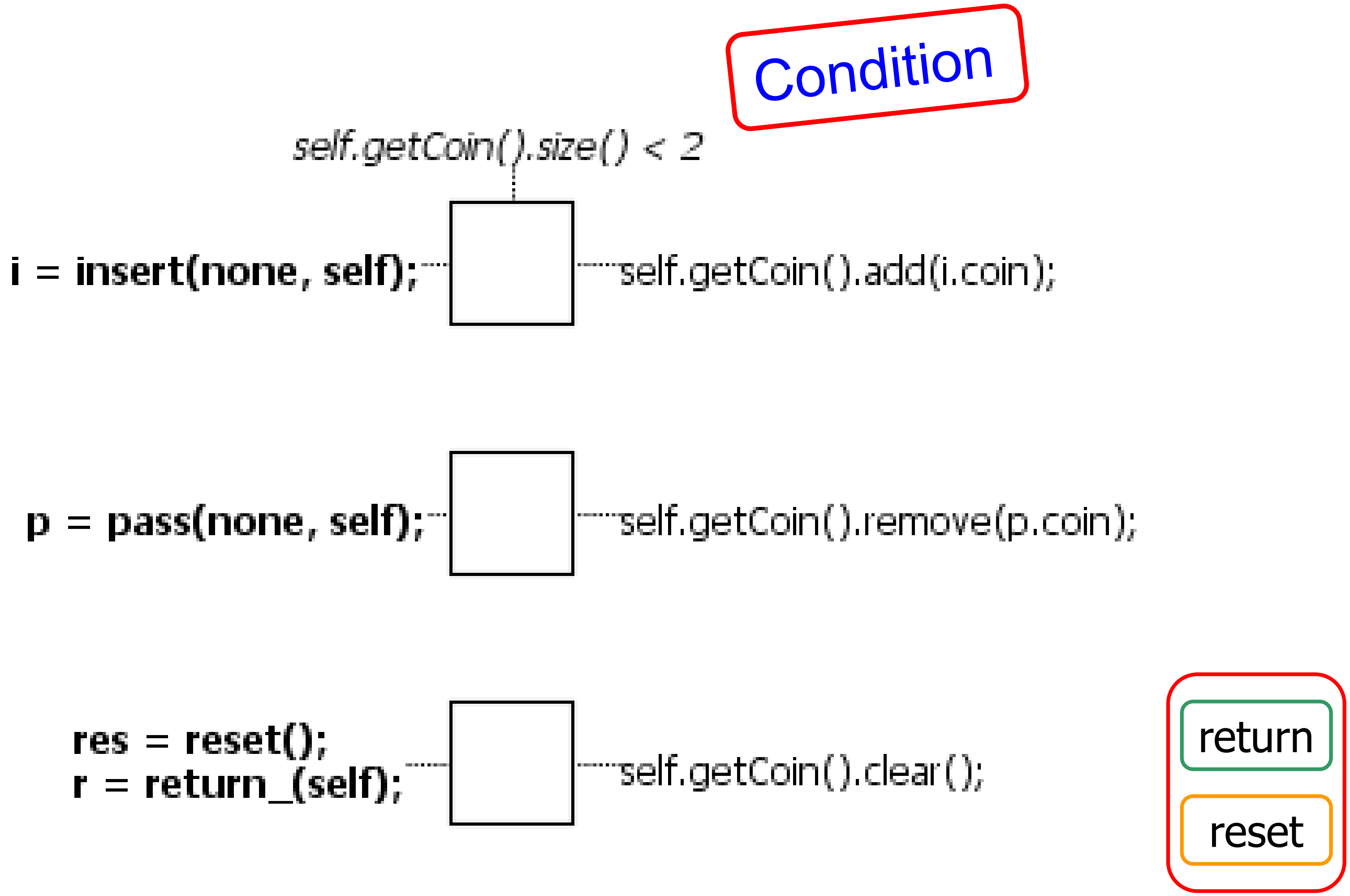
Local behaviour: Control



- Event binding with multiple event types!

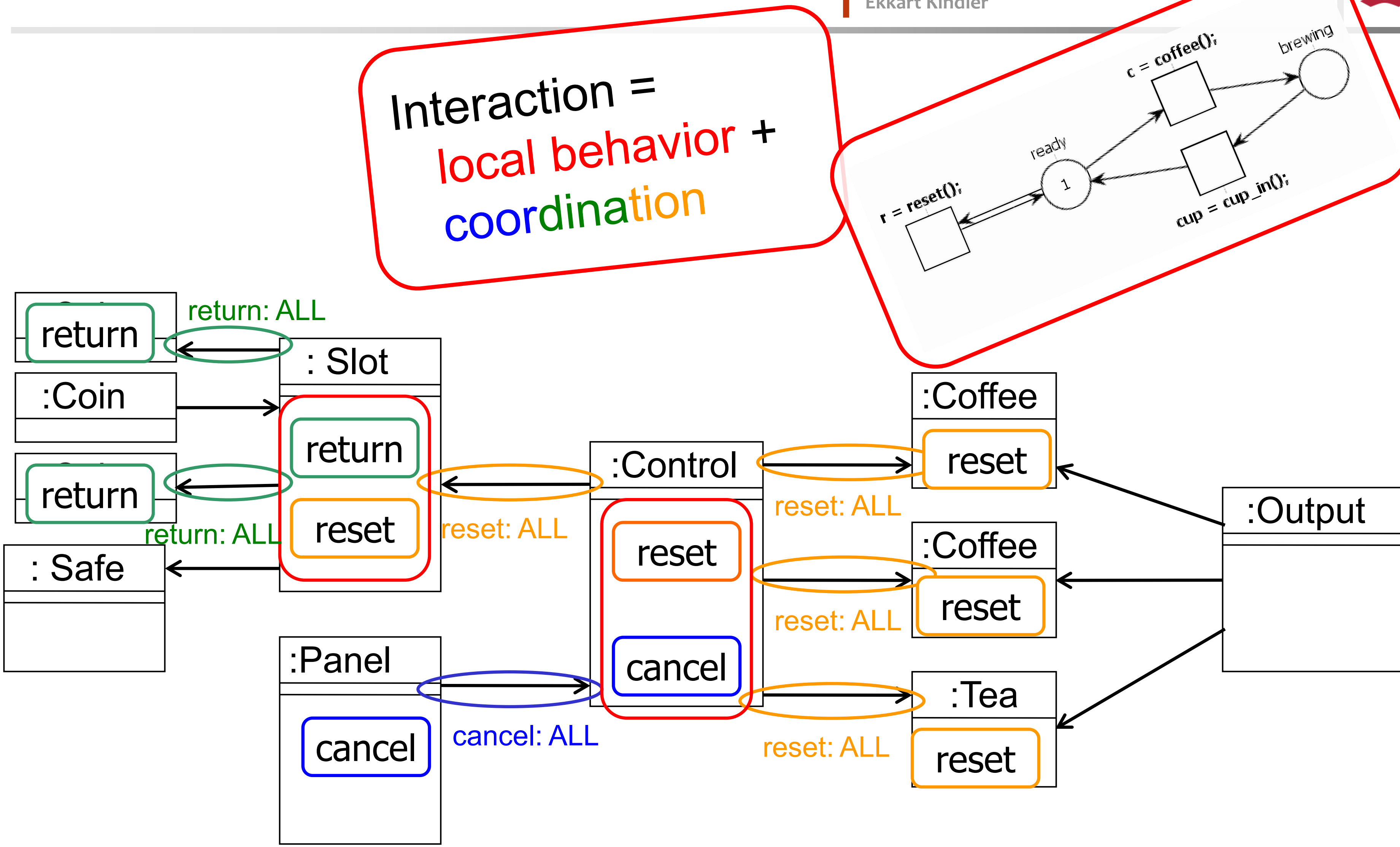
Local behaviour: Slot

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Interaction

DTU Compute
Department of Applied Mathematics and Comput
Ekkart Kindler

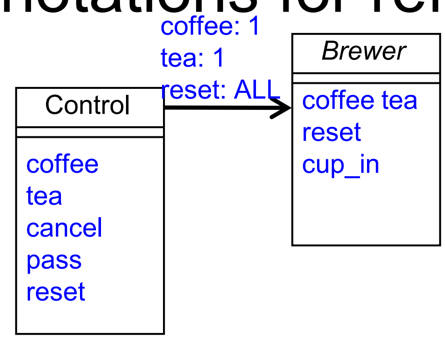


2.2 ECNO: Basic Concepts

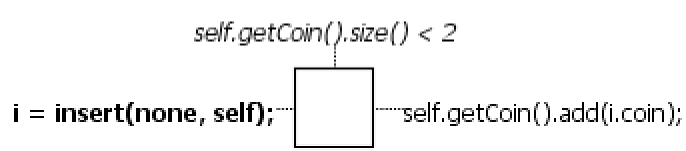
- ElementTypes (Classes)
- EventTypes with
  - parameters

insert(Coin coin, Slot slot)

- Global Behaviour: Coordination annotations for references
  - Event type
  - Quantification (1 or ALL)



- Local behaviour (life-cycle): ECNO nets (or something else)
  - Event binding (with parameter assignment)
  - Condition
  - Action

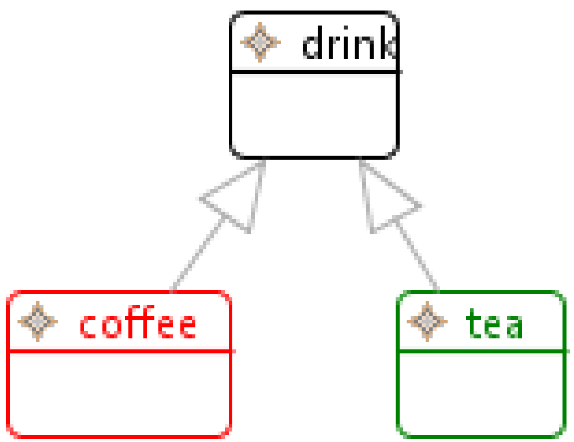


DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

ECNO with its basic concepts has some limitations, which makes modelling things **in an adequate way** a bit painful. ECNO has some additional concepts to make modelling more convenient. E.g.

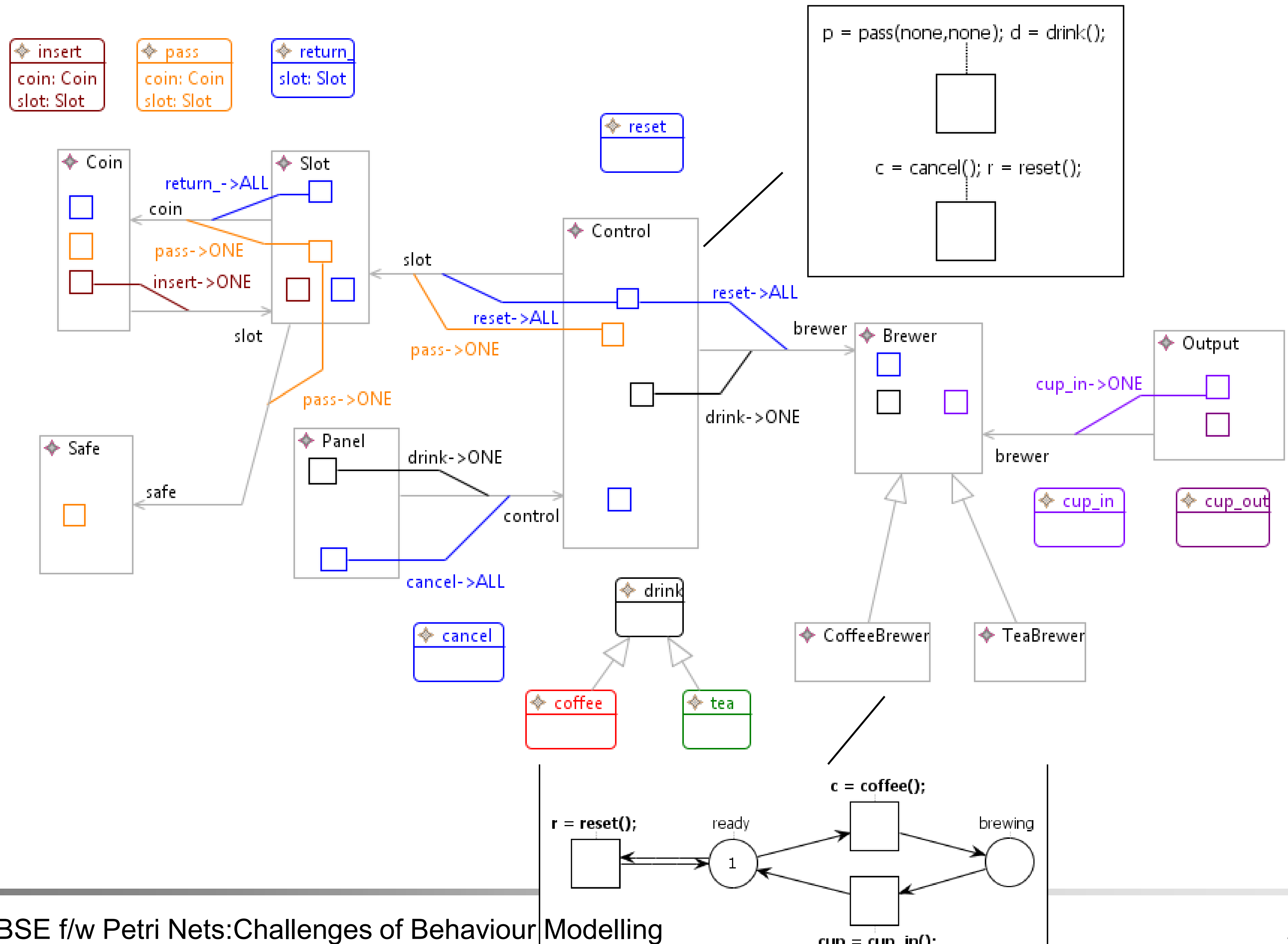
Just a glimpse …

- Inheritants on events



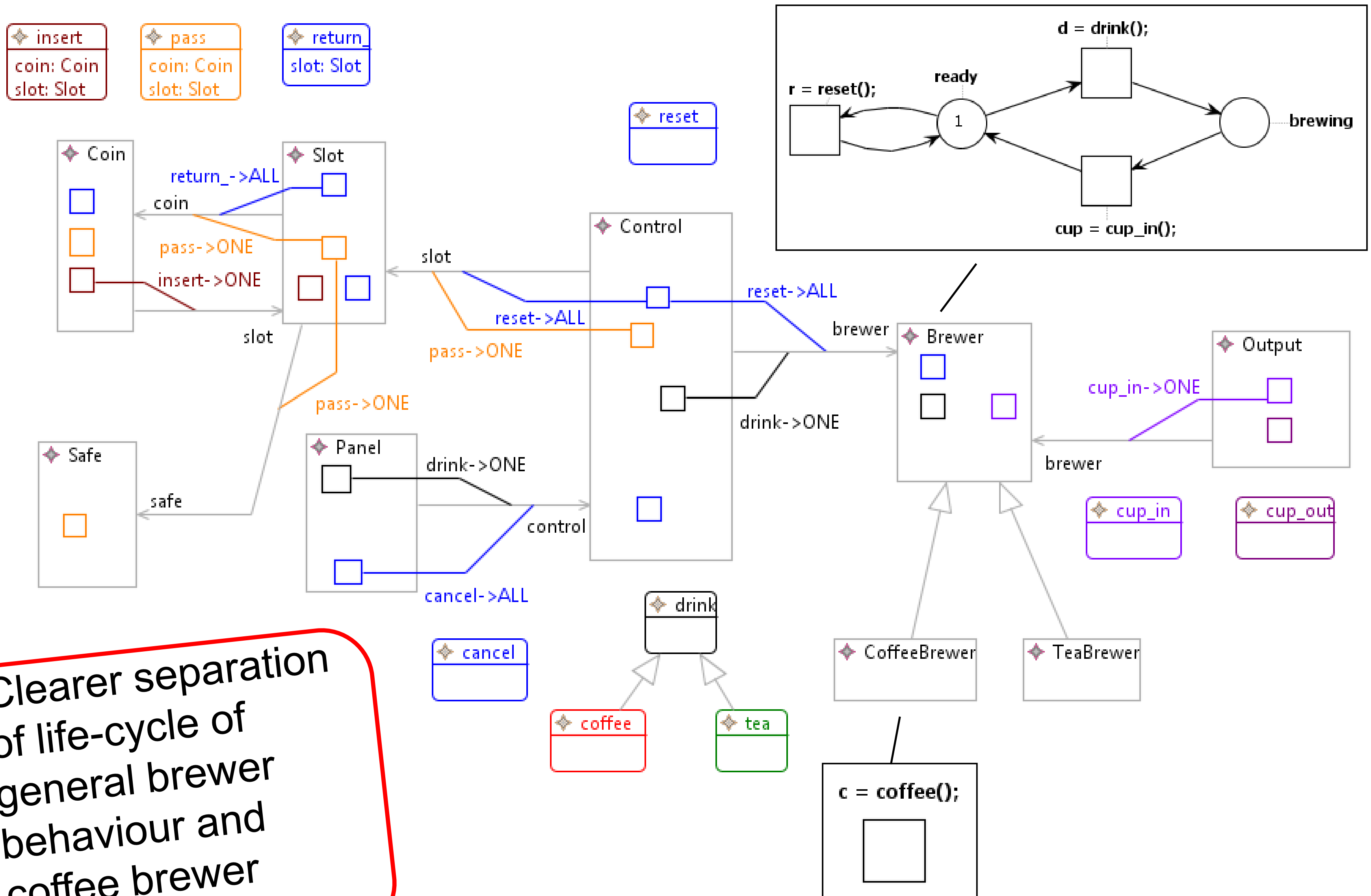
”Nicer Vendingmachine”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



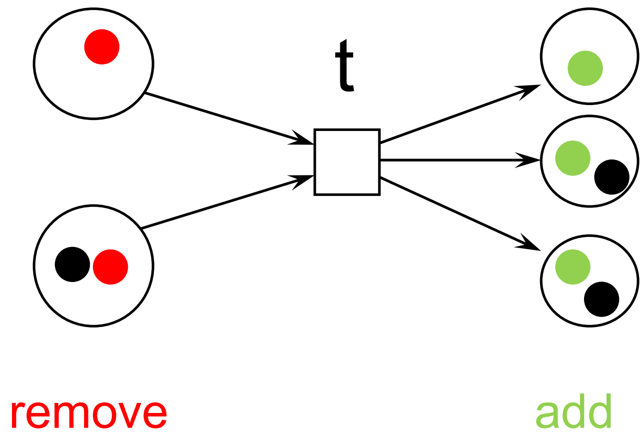
Behaviour inheritance

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Clearer separation of life-cycle of general brewer behaviour and coffee brewer specifics.

2.4 Example 2: Petri nets



How can we model that behaviour in ECNO nets?

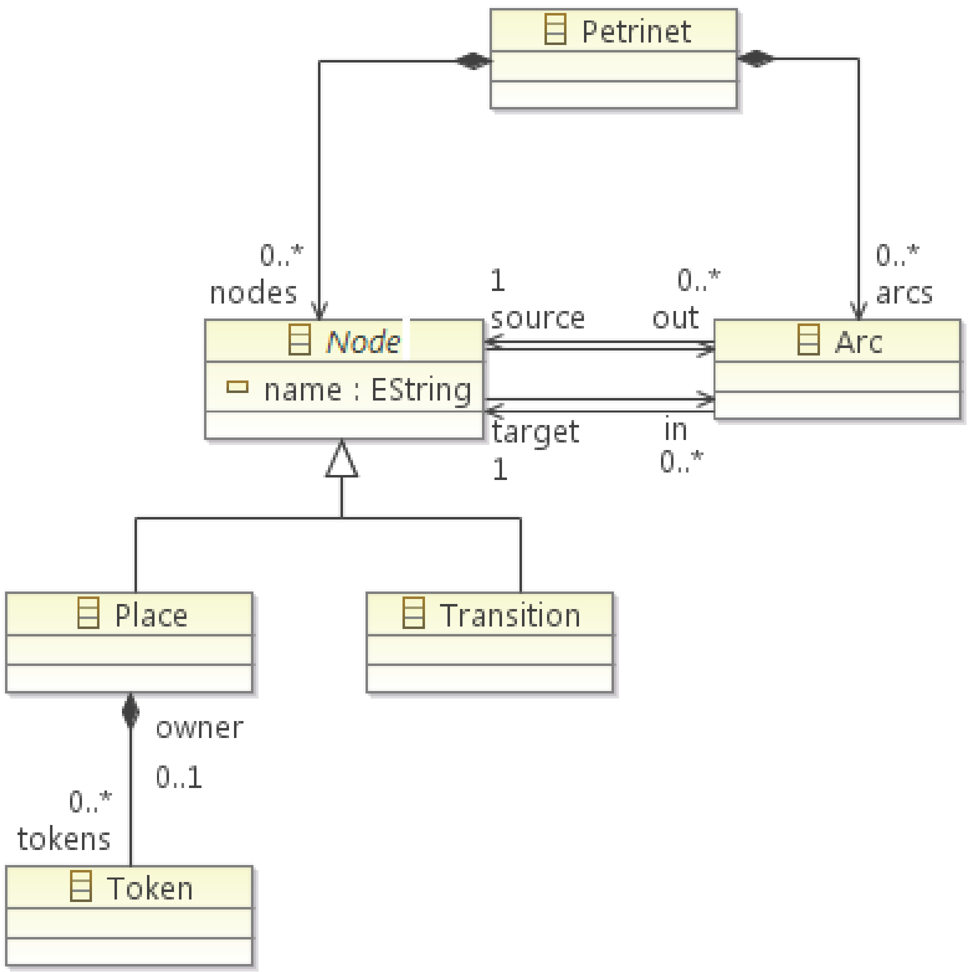
Transition t **enabled**:
  for ALL incoming Arcs a:
    for ONE source Place p of Arc a:
      find a token

Fire Transition t:
  for ALL incoming Arcs a:
    for ONE source Place p of Arc a:
      find a token and remove it

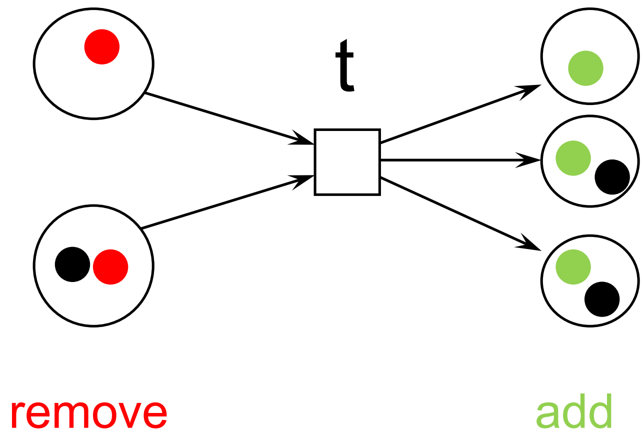
  for ALL outgoing arcs a:
    for ONE target Place p of Arc a:
      add a new Token

Petri net: Abstract Syntax

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Example 2: Petri nets

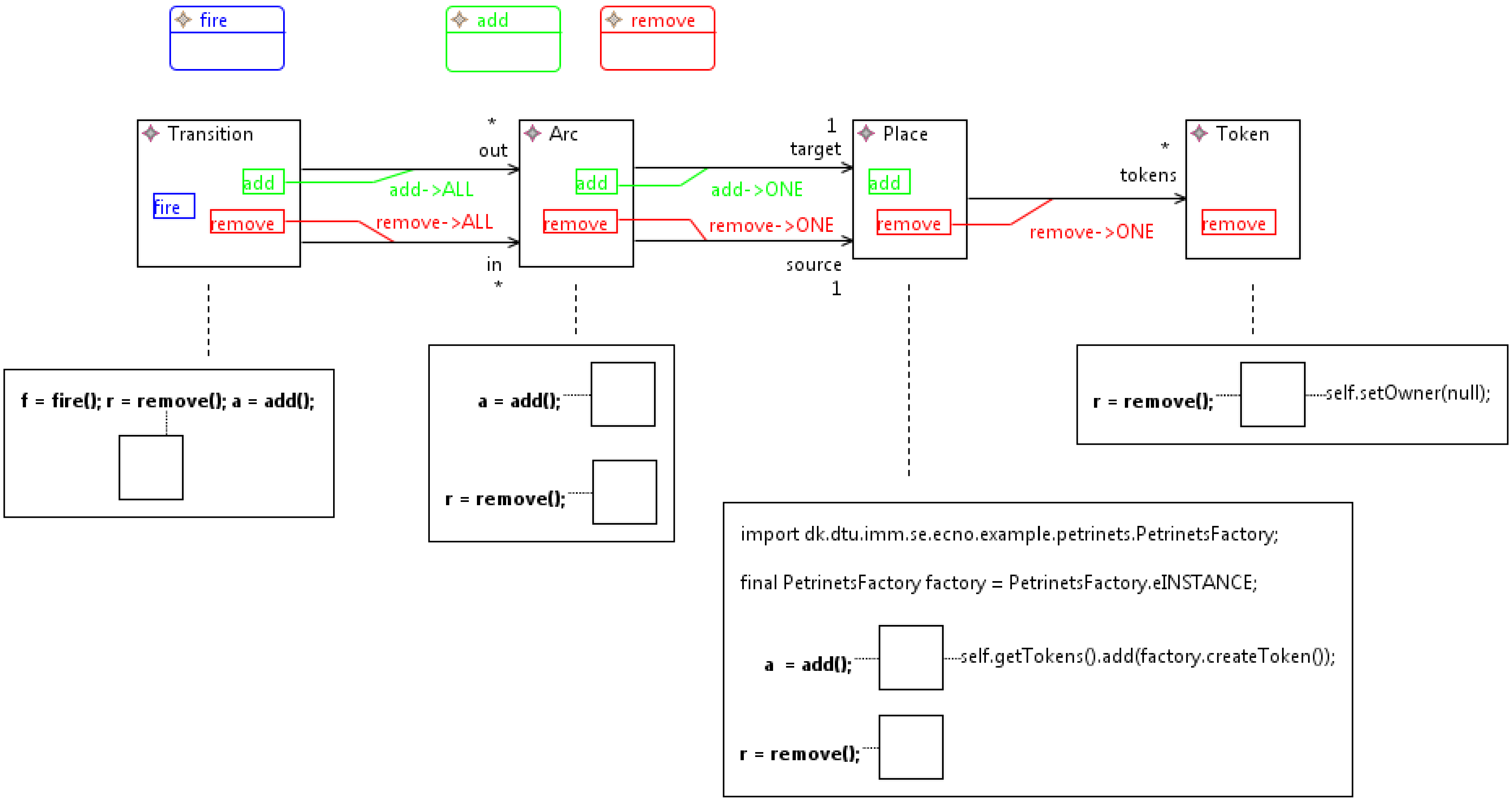


Transition t **enabled**:
  for **ALL** incoming Arcs a:
    for **ONE** source Place p of Arc a:
      find a token

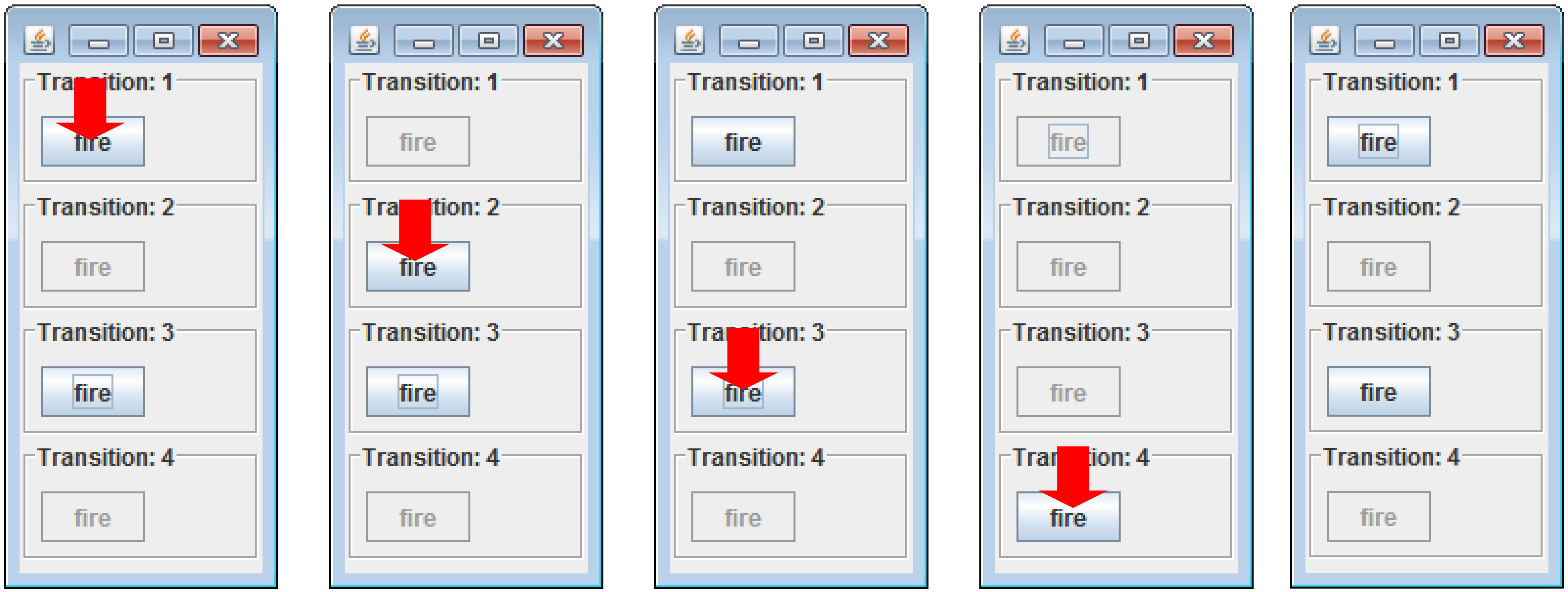
Fire Transition t:
  for **ALL** incoming Arcs a:
    for **ONE** source Place p of Arc a:
      find a token and remove it

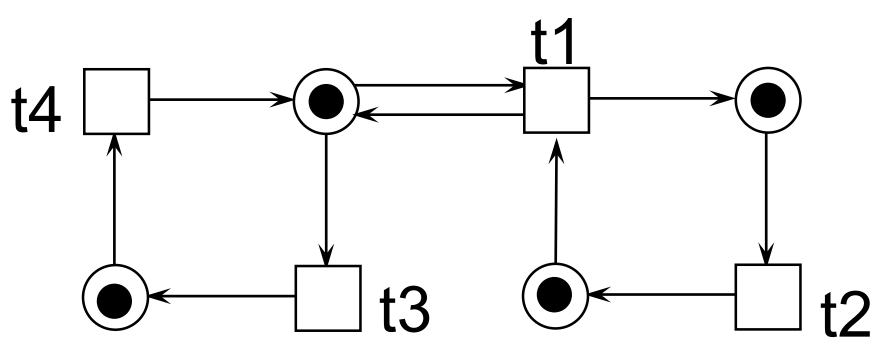
  for **ALL** outgoing arcs a:
    for **ONE** target Place p of Arc a:
      add a **new** Token

ECNO Semantics of PN



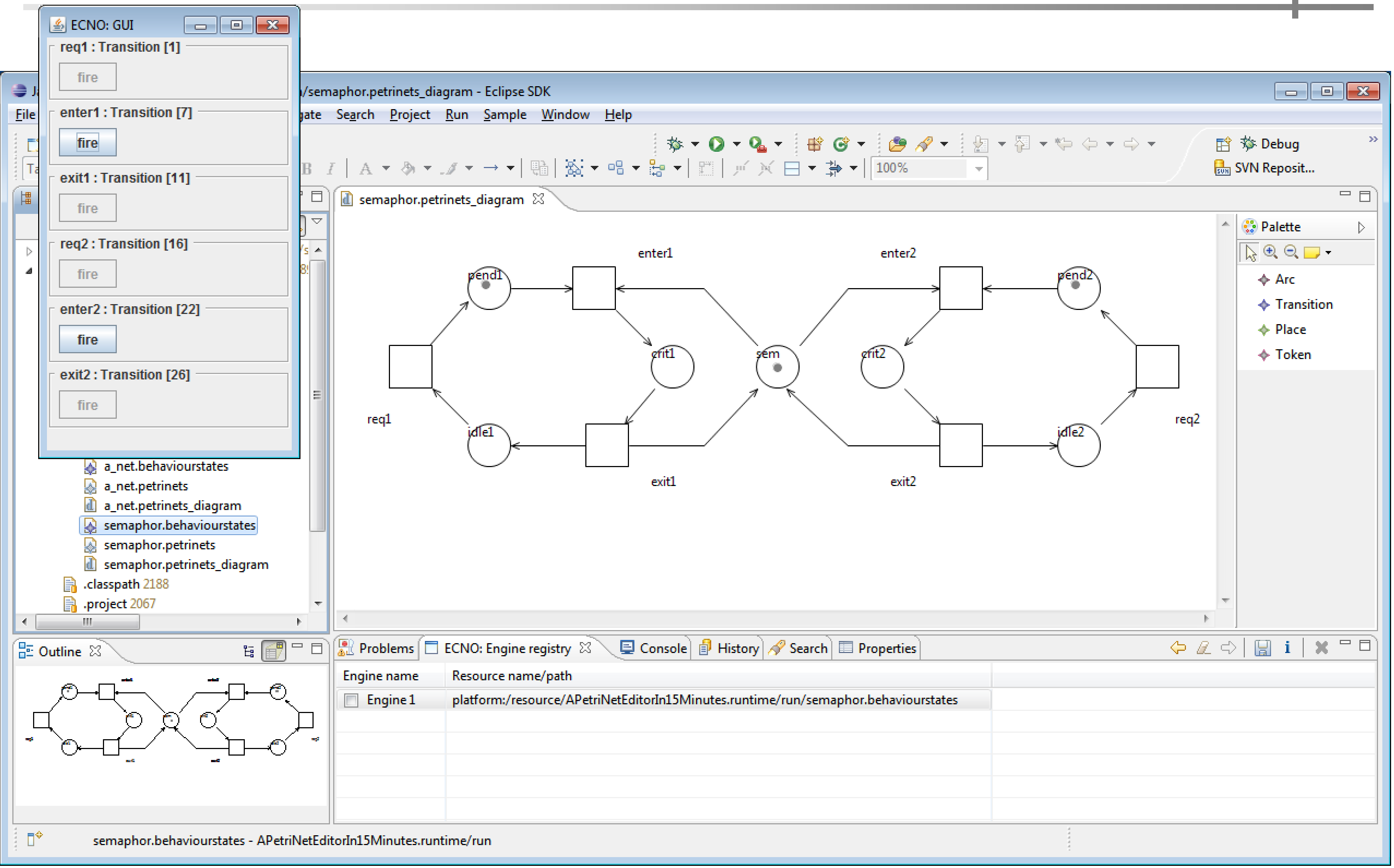
Result


Transition: 1
fire
Transition: 2
fire
Transition: 3
fire
Transition: 4
fire


t1
t4
t3
t2

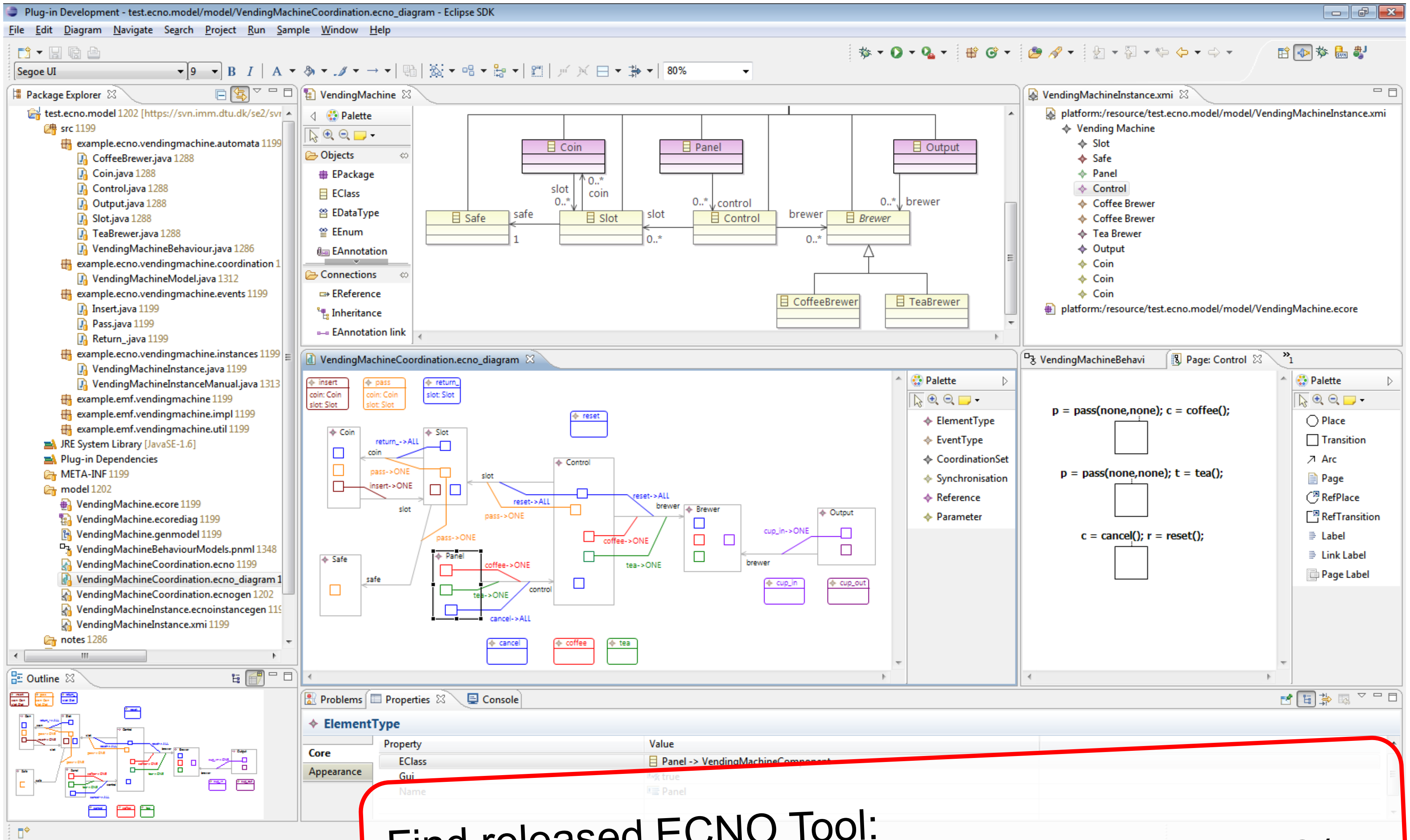
Petri net simulator

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



2.5 ECNO: Summary

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Find released ECNO Tool:
http://www2.compute.dtu.dk/~ekki/projects/ECNO/

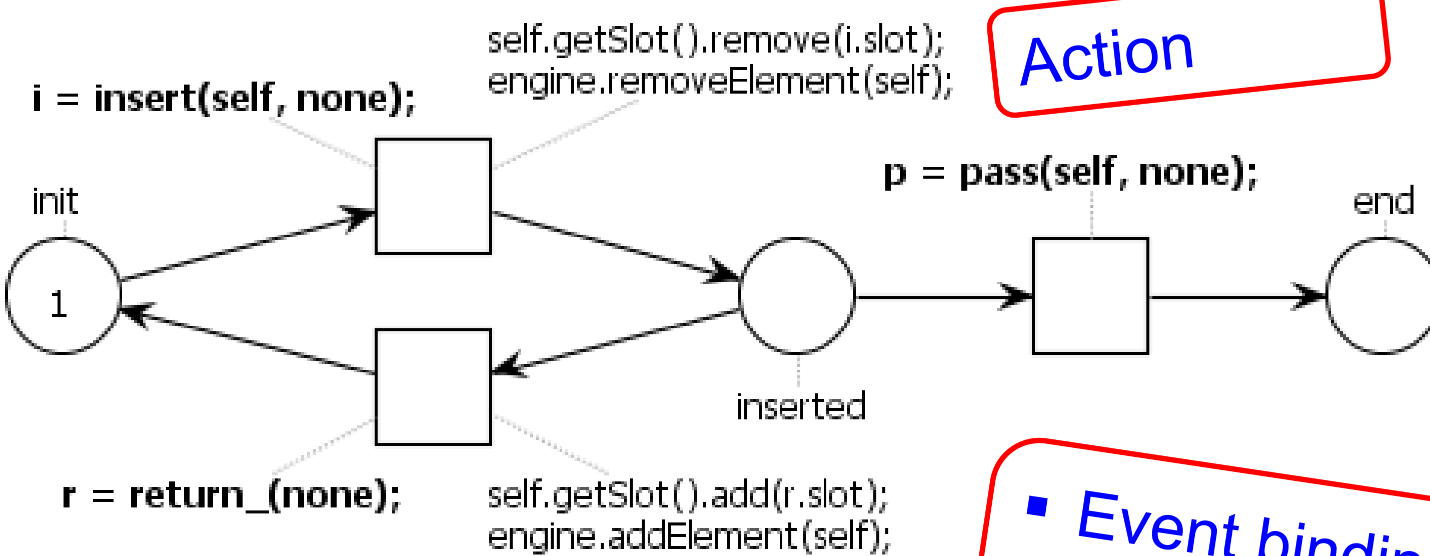
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- ECNO nets and the code generator are probably the largest application of the ePNK!

Declarations and imports

```
import dk.dtu.imm.se.ecno.engine.ExecutionEngine;

final ExecutionEngine engine = ExecutionEngine.getInstance();
```

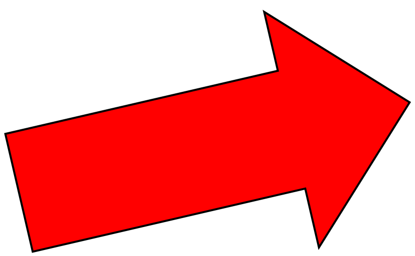


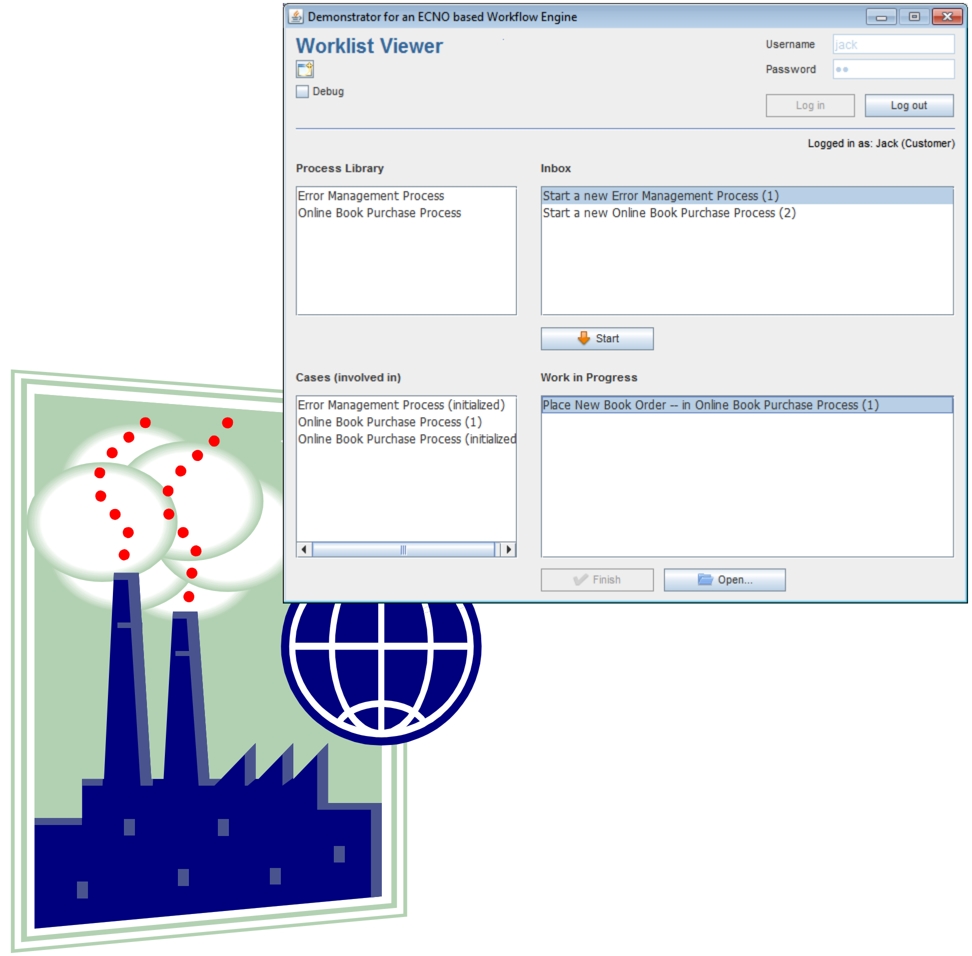
Action

- Event binding
- Parameter assignment

Beyond Mickey Mouse

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





Discussion

- Models of software on higher level of abstraction
  → concise / adequate
- Domain model including semantics
- Coordination! Not invocation!

UML is too invocation oriented!

- Idea: Define semantics of ECNO in ECNO itself (truely "meta")

UML does that (→ MOF)
– but for structure only!
How about behaviour?

ECNO: Focus

- Software Engineers
  (vs. end-users)
- Domain level
  (vs. low-level programming)
- Coordination
  (vs. algorithmics/invocation/sequential flow)
- Behaviour
  (vs. GUI)

But, there is a project were standard GUI generators could be coupled with ECNO, to model the complete software including behaviour and GUI!

Comment of a non-IT friend

”Oh, that’s great. Then I can also develop my own software now.”

My first reaction:
Nooo?!

ECNO is for software engineers, helping them focussing on the domain and not on technical details!

On second thought:

Can’t we help end-users develop their own programs? At least in simple cases.