Coordinating Interactions
The Event Coordination Notation

Ekkart Kindler

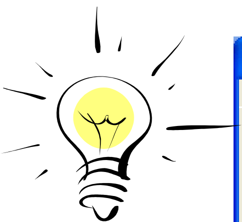
DTU Compute
Department of Applied Mathematics and Computer Science

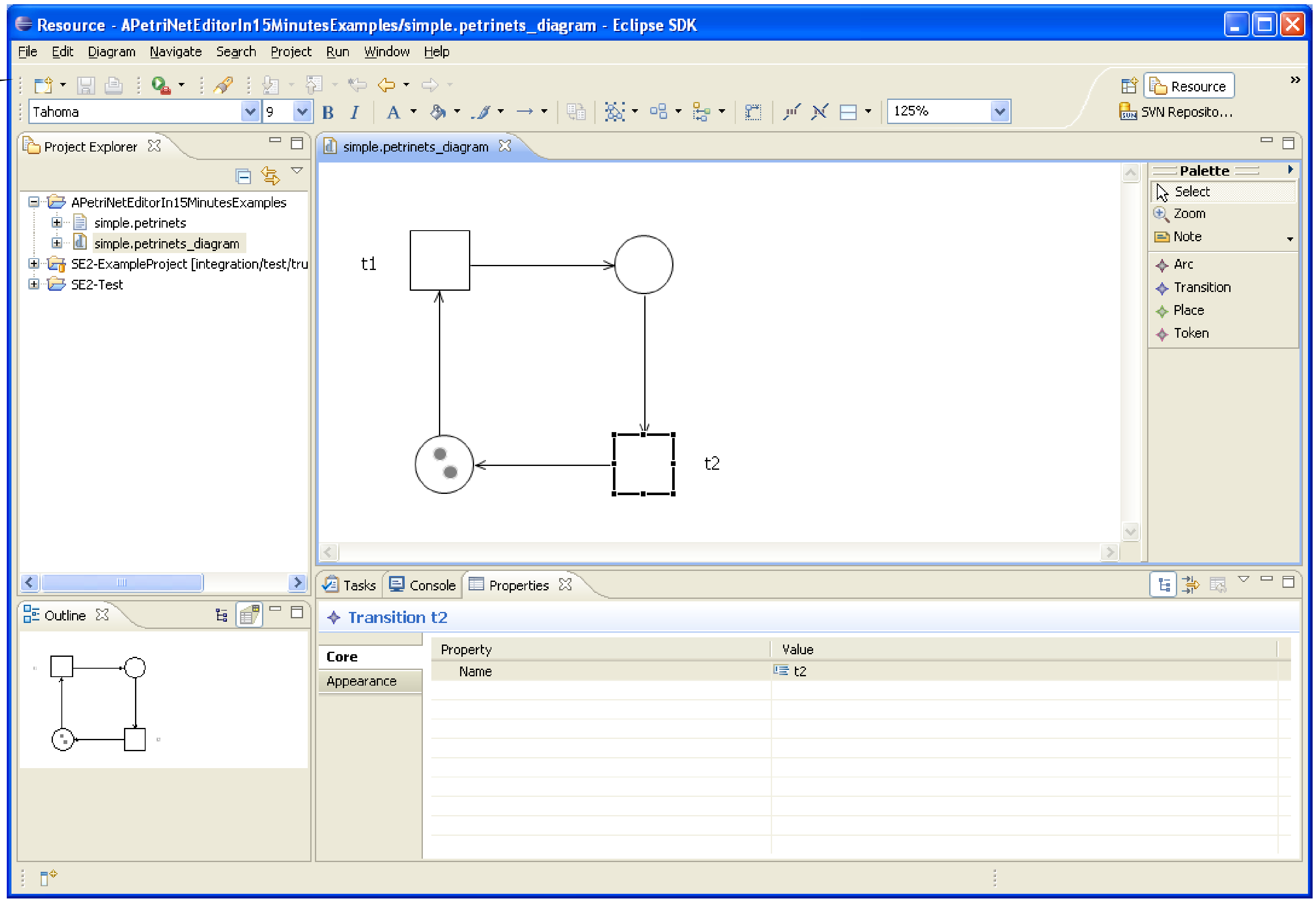
Ekkart Kindler: Coordinating Interactions:
The Event Coordination Notation.
DTU Compute Technical Report 2014-05,
May 2014

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

1. Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

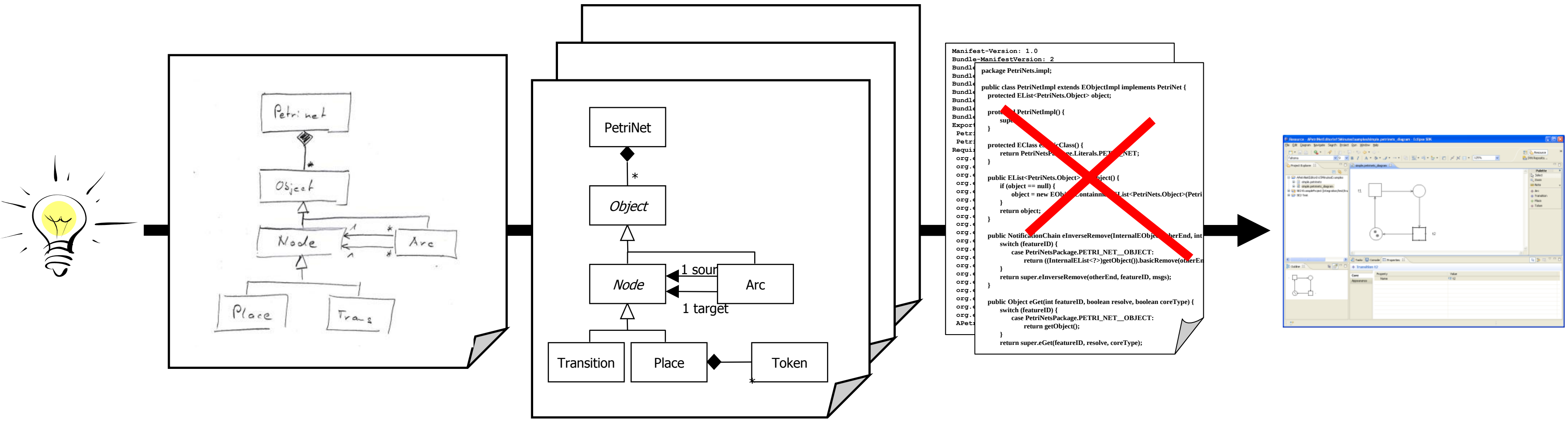




DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

MBSE: Today

How about behaviour ?
(non-standard behaviour)

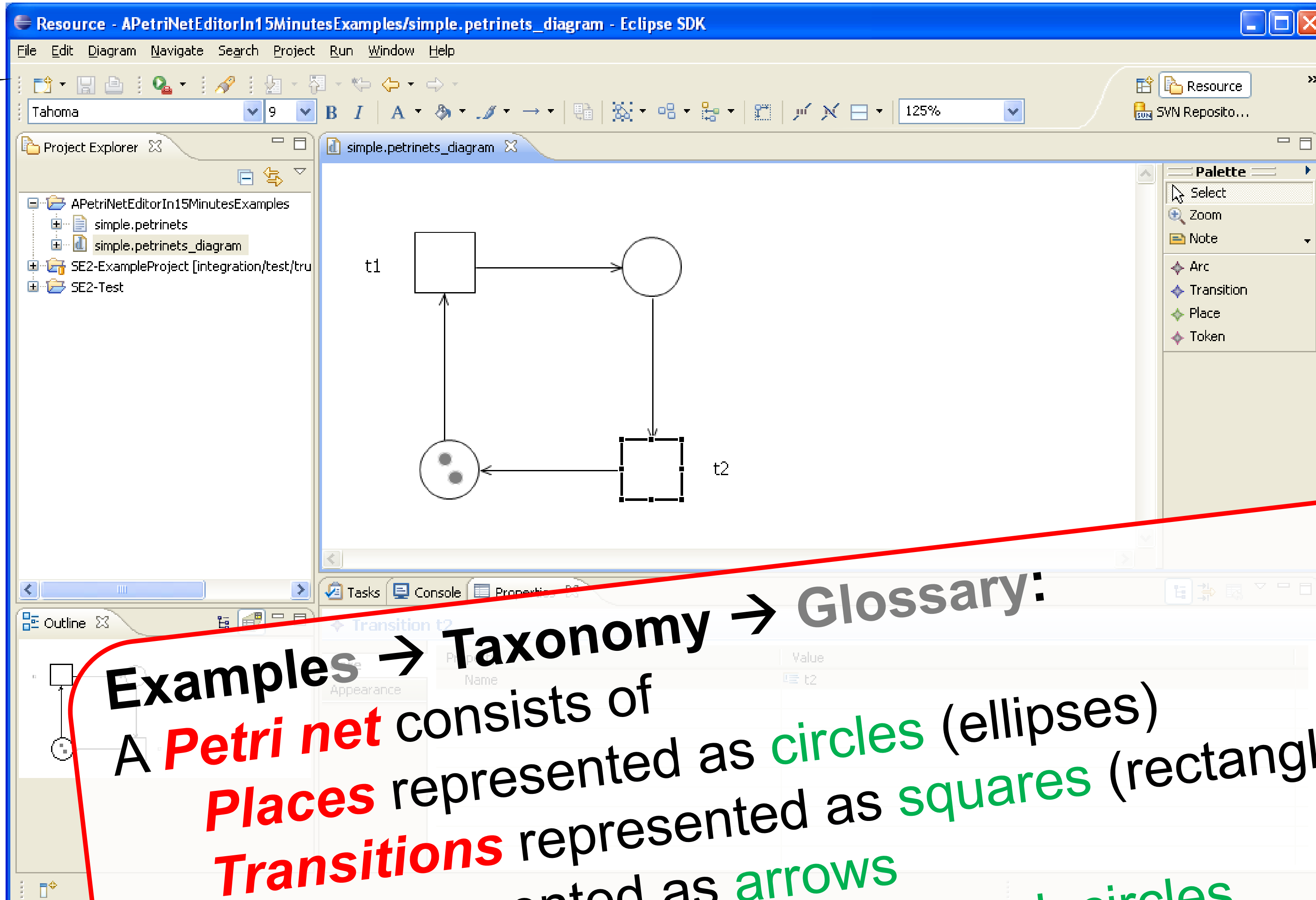


Analysis
Design
Implementation
Coding

Some code is generated

Modelling on domain level

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



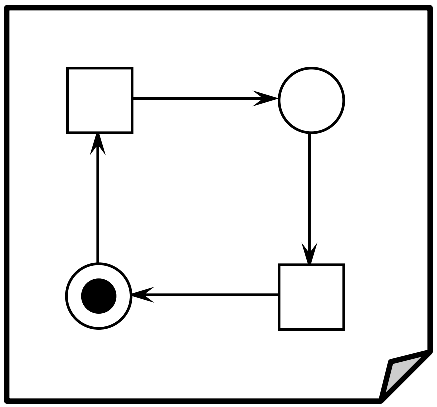
Examples → Taxonomy → Glossary:

A ***Petri net*** consists of

- ***Places*** represented as circles (ellipses)
- ***Transitions*** represented as squares (rectangles)
- ***Arcs*** represented as arrows
- ***Tokens*** represented as filled black circles

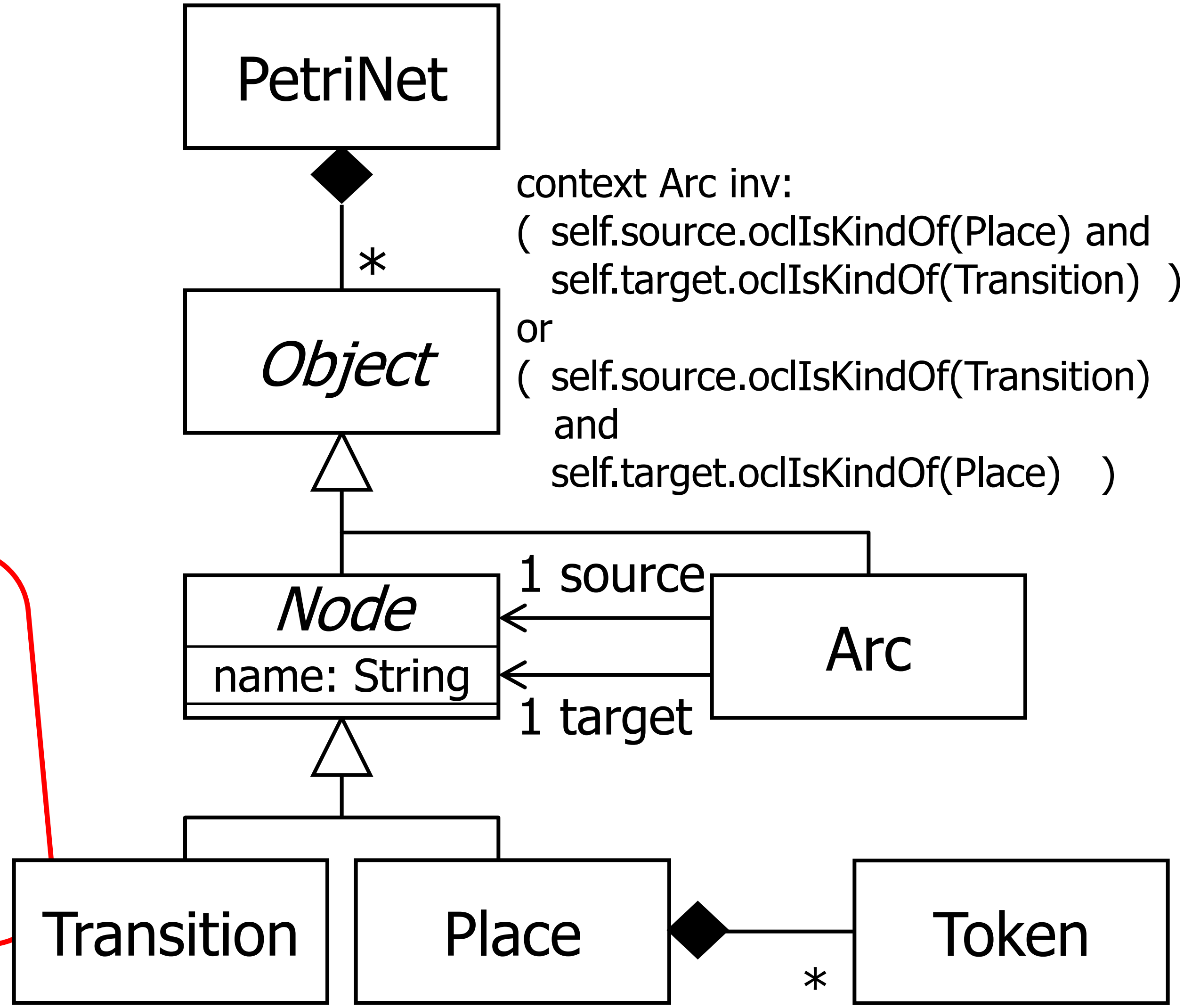
Domain model of PN

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



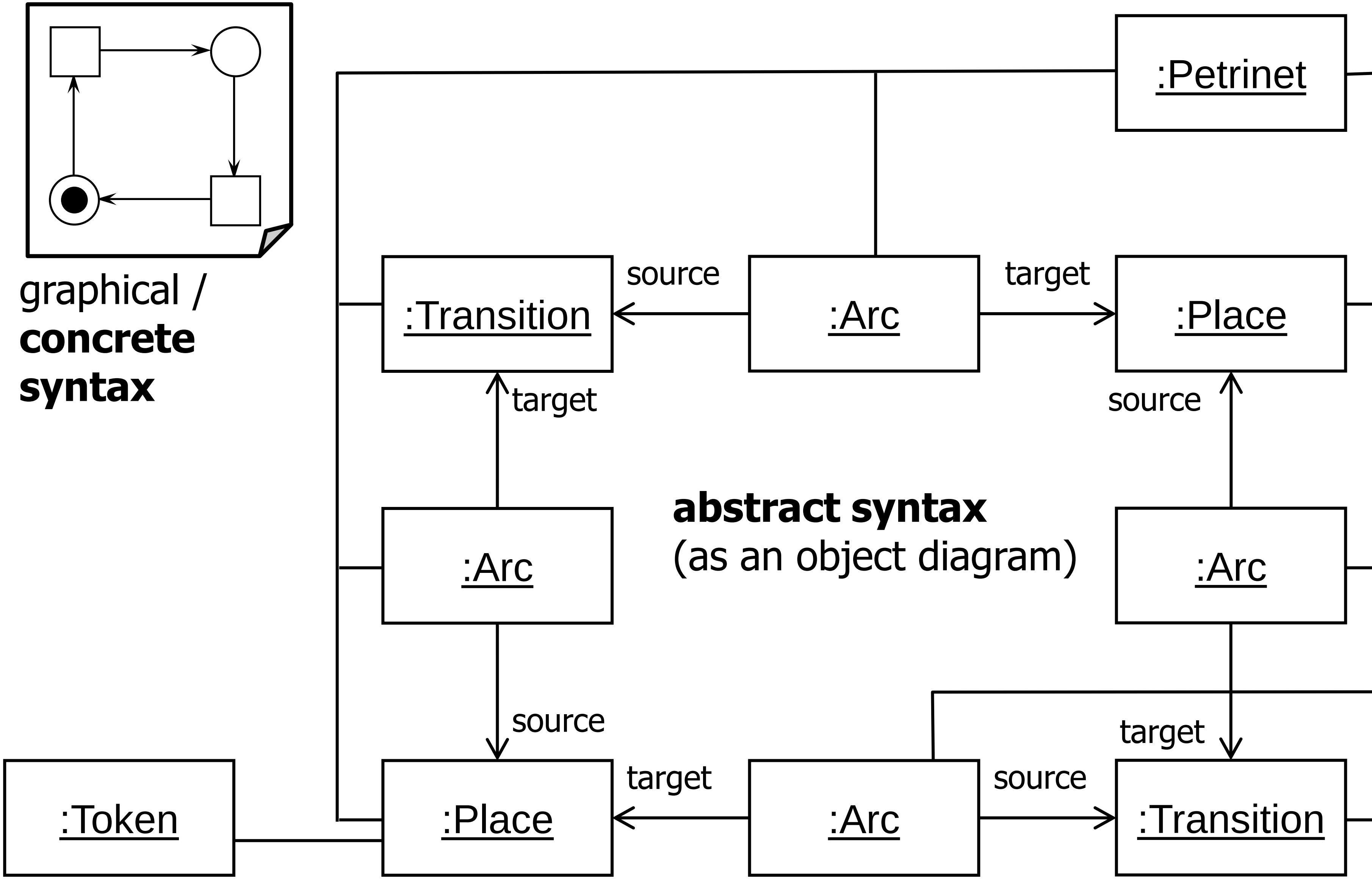
Petri net model

Glossary:
A ***Petri net*** consists of
Places
Transitions
Arcs
Tokens



"Meta model" for Petri nets
(as a UML class diagram + OCL)

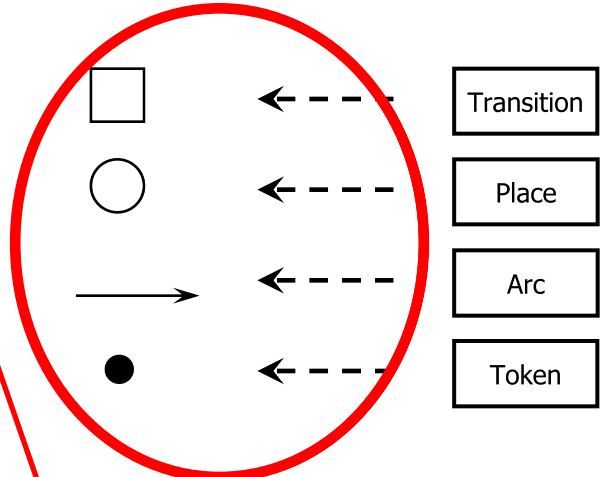
Syntax (abstract and concrete)

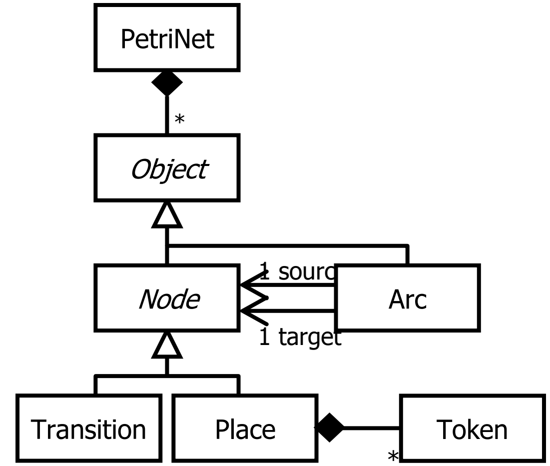


Graphical Syntax/Editor?

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

meta model



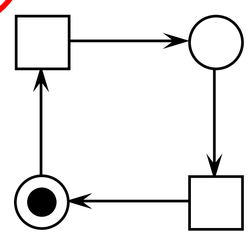


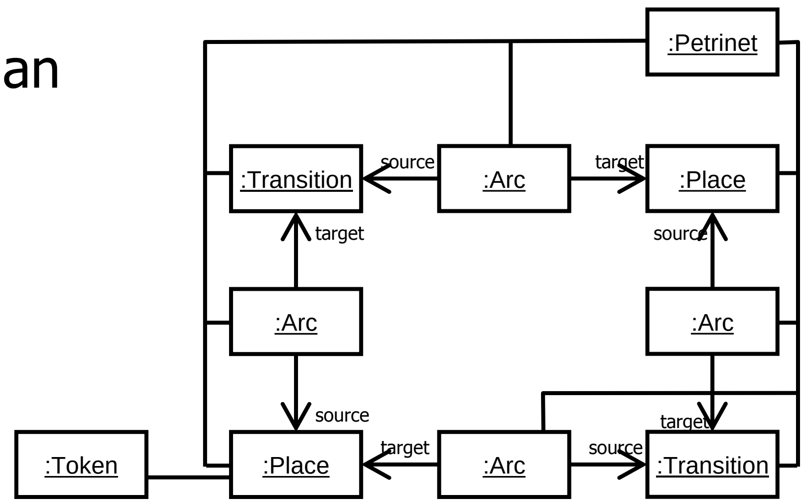
is instance of

This is about all there is to say about a Petri net editor – conceptually!

generate an editor

model



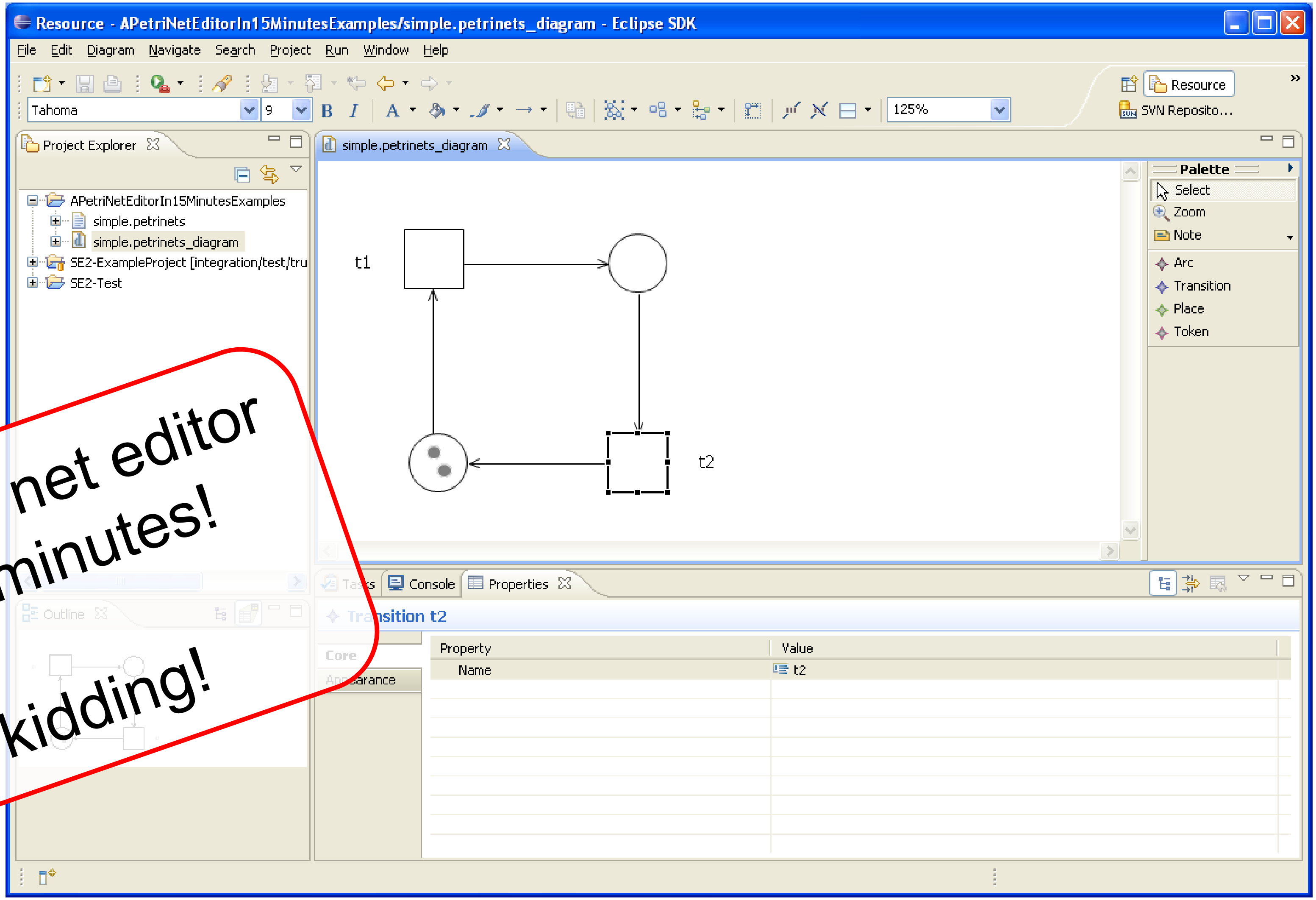


concrete syntax

abstract syntax

The Result

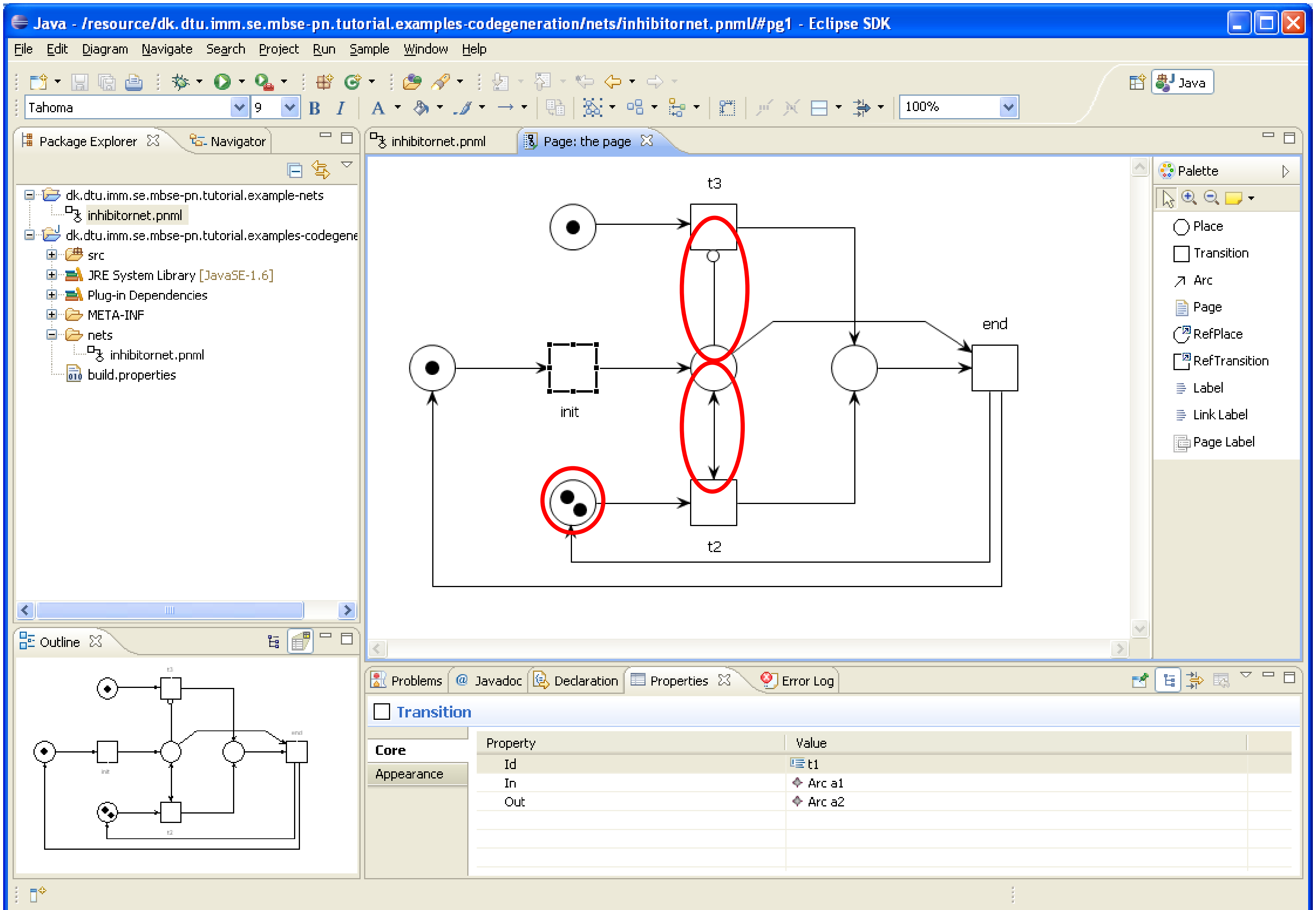
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



A Petri net editor in 15 minutes!

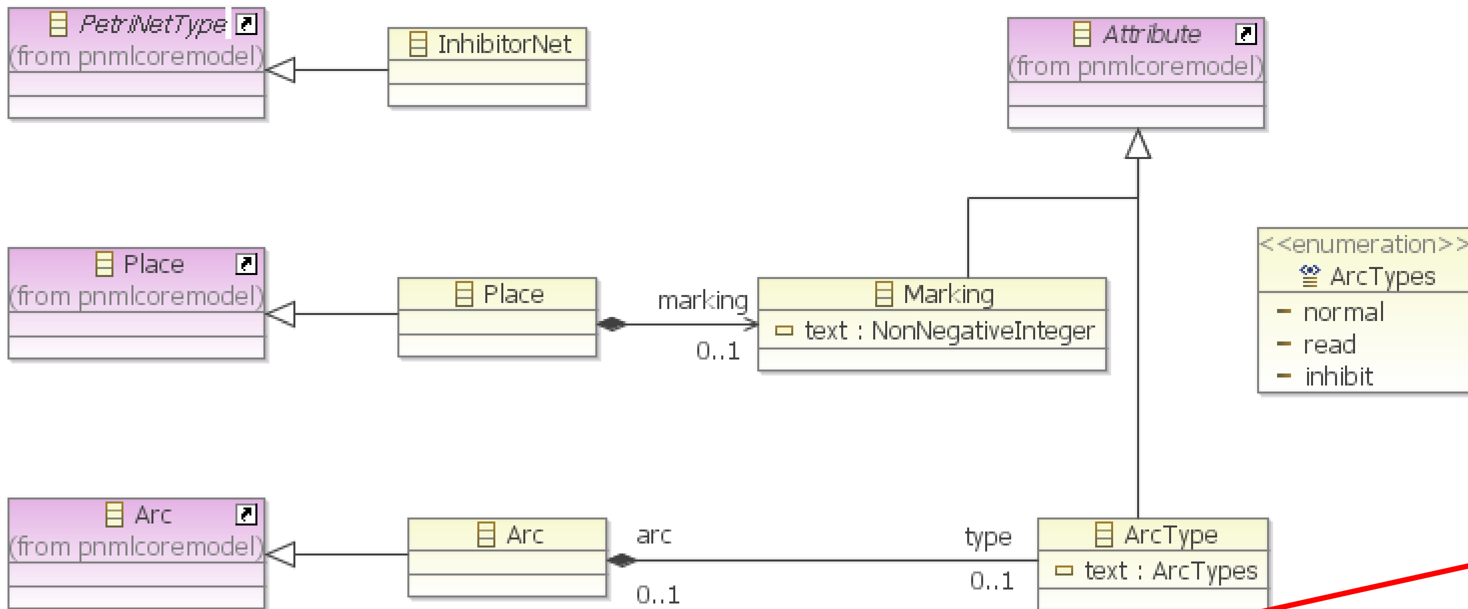
Not kidding!

In the large: ePNK



Define a Petri Net Type

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Additional constraints of inhibitor nets (defined in OLC).

```
( self.source.oclIsKindOf(pnmlcoremodel::PlaceNode) and
  self.target.oclIsKindOf(pnmlcoremodel::TransitionNode) )
or
( self.source.oclIsKindOf(pnmlcoremodel::TransitionNode) and
  self.target.oclIsKindOf(pnmlcoremodel::PlaceNode) and
  not ( self.type.text = ArcTypes::inhibit ) )
```

Program Graphics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public class InhibitornetsArcFigure extends ArcFigure {
[...]
  private void setGraphics() {
    RotatableDecoration targetDecorator = null;
    RotatableDecoration sourceDecorator = null;

    if (type.equals(ArcTypes.NORMAL)) {
      targetDecorator = new ReisigsArrowHeadDecoration();

    } else if (type.equals(ArcTypes.INHIBIT)) {
      CircleDecoration circleDecoration =
        new CircleDecoration();
      circleDecoration.setLineWidth(this.getLineWidth());
      targetDecorator = circleDecoration;

    } else if (type.equals(ArcTypes.READ)) {
      targetDecorator = new ReisigsArrowHeadDecoration();
      sourceDecorator = new ReisigsArrowHeadDecoration();
[...]
```

Just a glimpse!

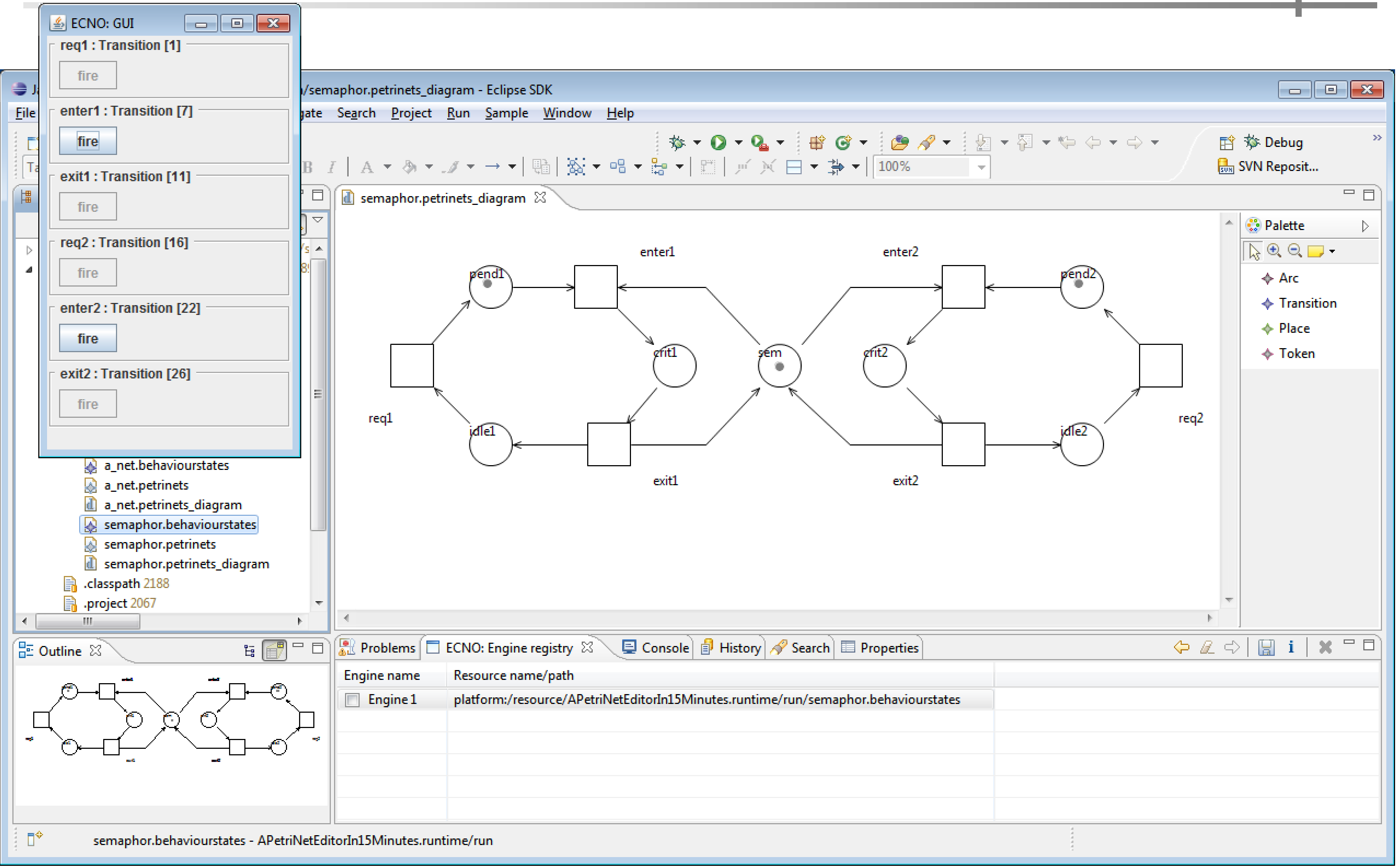
Benefits of Modelling

- Better understanding
- Mapping of instances to XML syntax (XMI) (canonical exchange format for instances)
- Automatic code generation for
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)
  - Transactional changes (ACI)
  - Database access / persistence (D)
  - Graphical editors

BUT:
All this is "standard functionality"!
How about "real" functionality or behaviour?

e.g. a Petri net simulator?

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Challenges

~~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"!

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

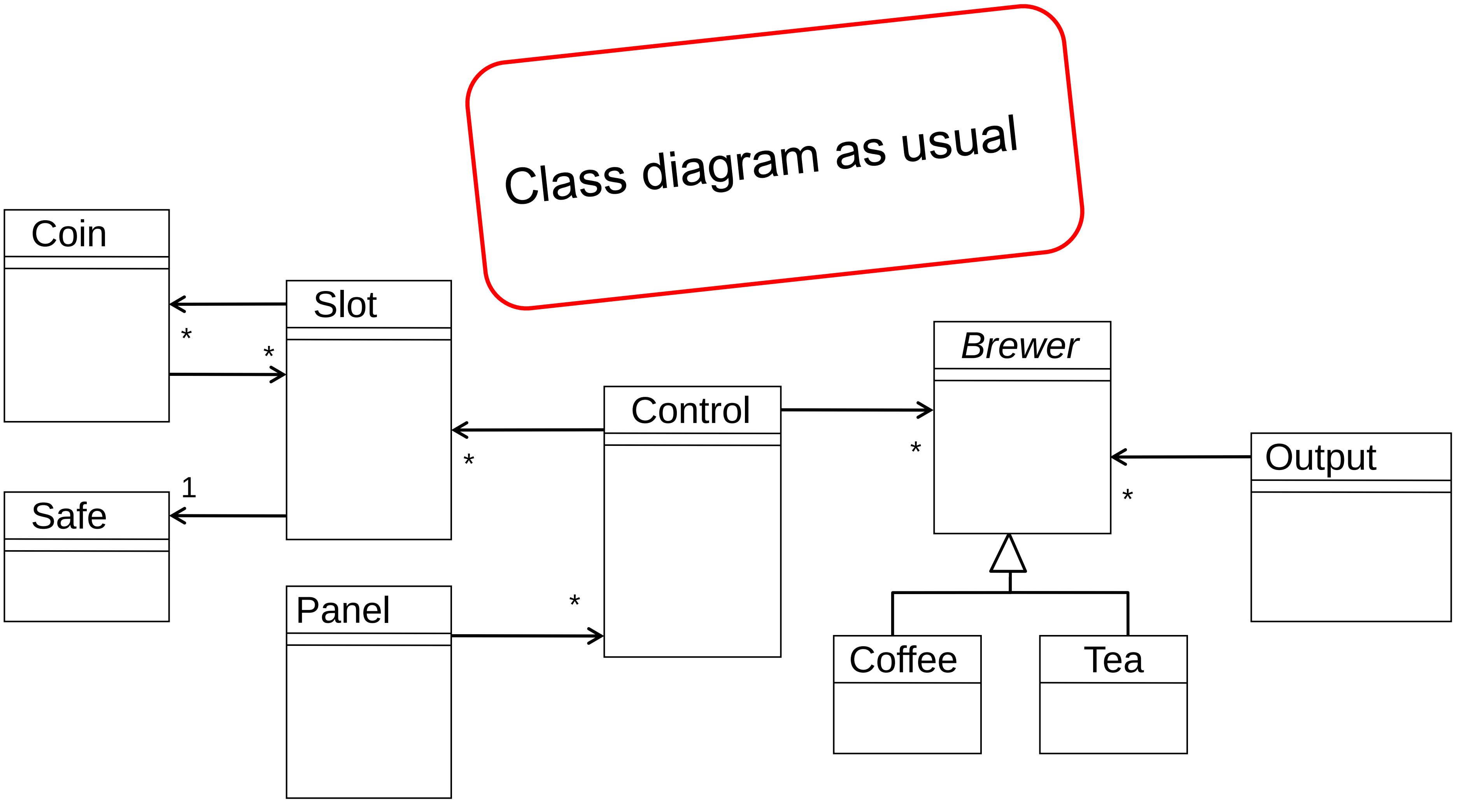
Motivation

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

Meta-modelling / domain modelling **including behaviour!**

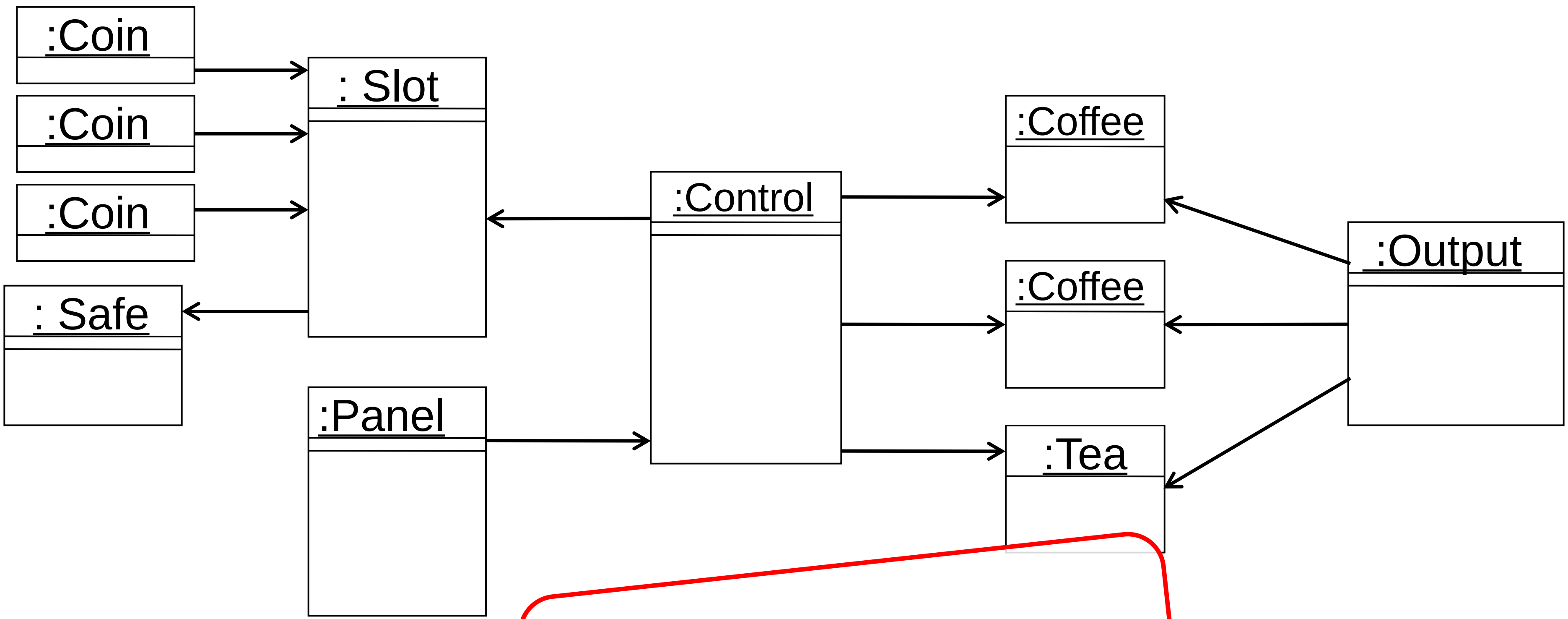
→ Integrate behaviour models with structural models
→ Integrate different structural models and manually written code (or code generated by different technologies)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Instance: Object Diagram

Initial configuration, current situation

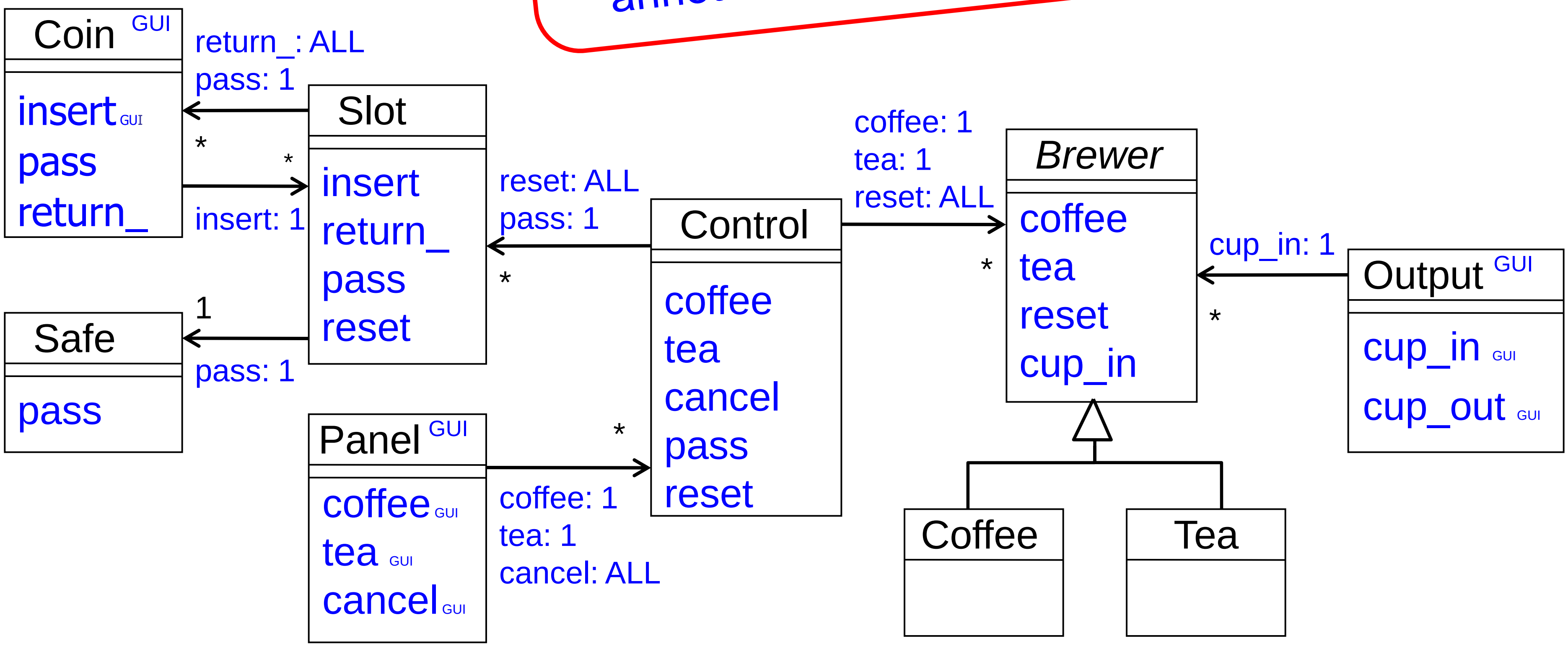


Object diagram as usual

Coordination Diagram

- We call objects elements now!

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



... + Event declaration

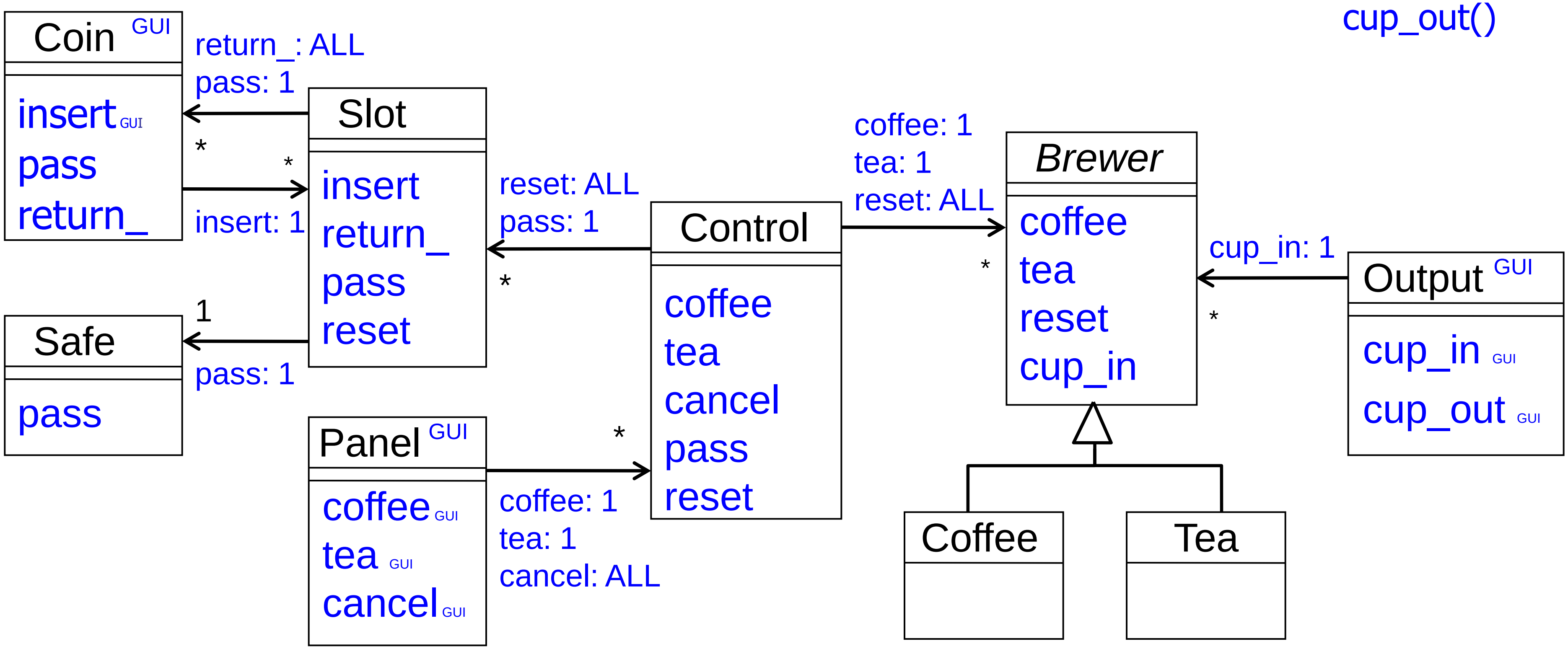
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Event (type) declaration
- Parameters

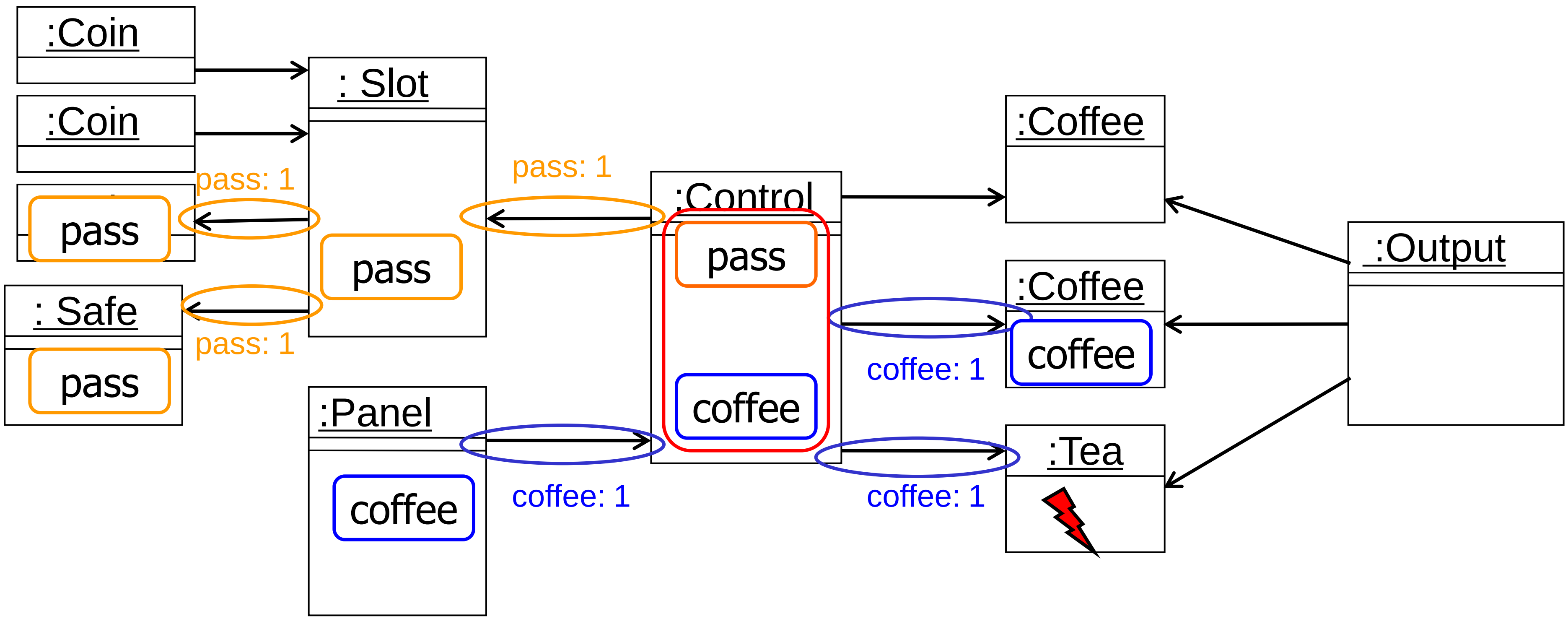
insert(Coin coin, Slot slot)
pass(Coin coin, Slot slot)
return(Slot slot)
reset_()

coffee()
tea()
cancel()

cup_in()
cup_out()

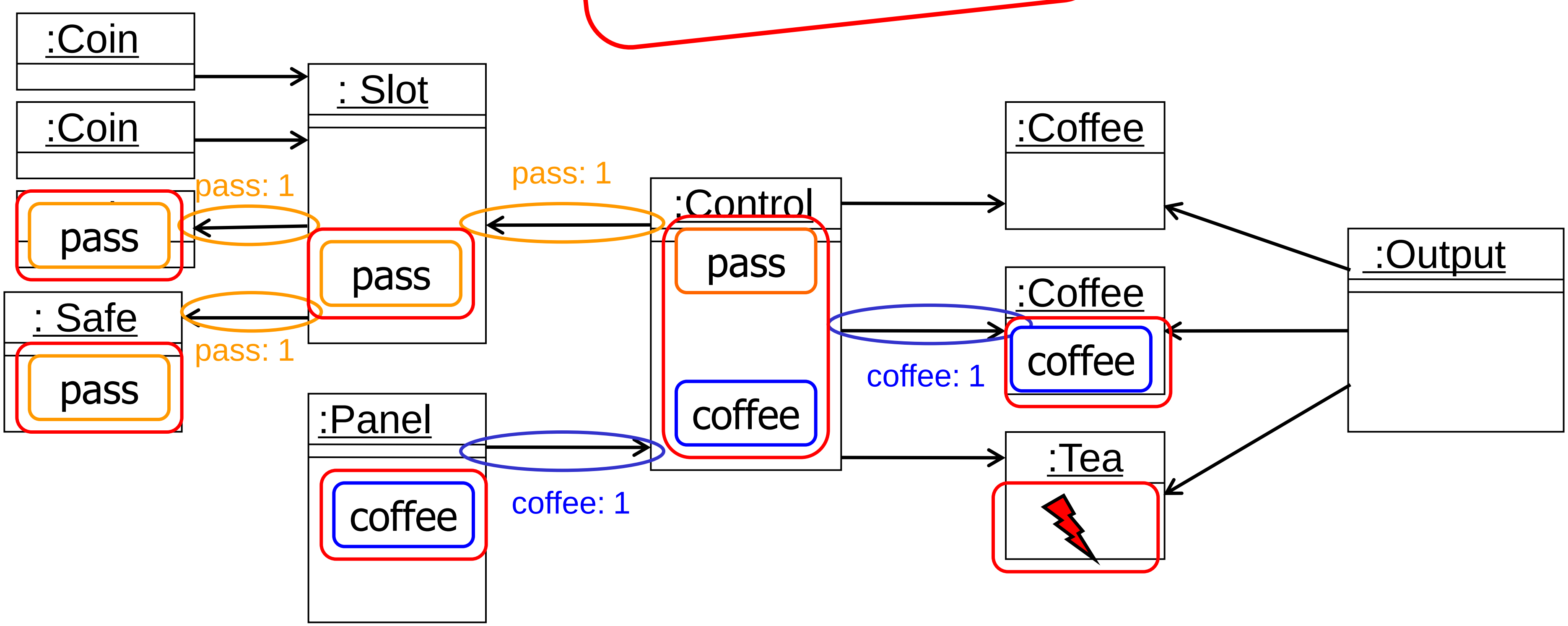


Interactions



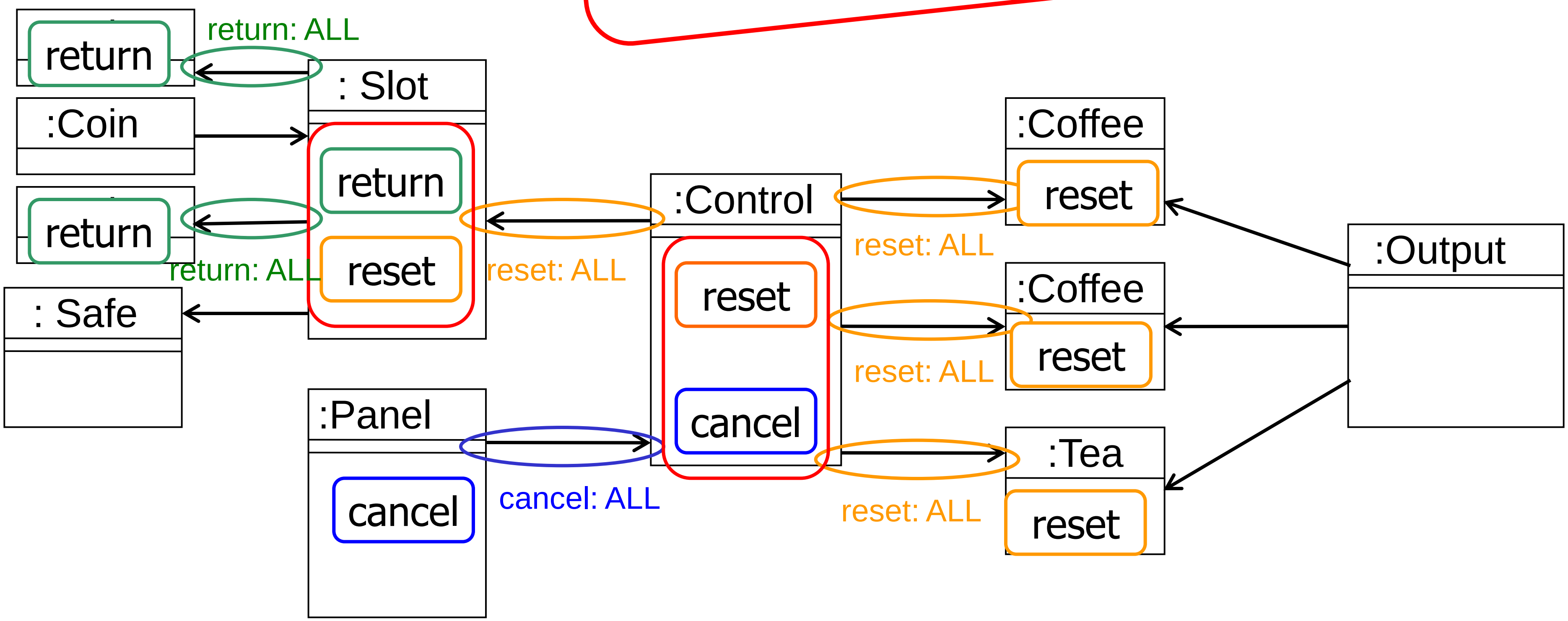
Interactions

Interaction =
local behavior +
coordination



Another Interaction

Interaction =
local behavior +
coordination

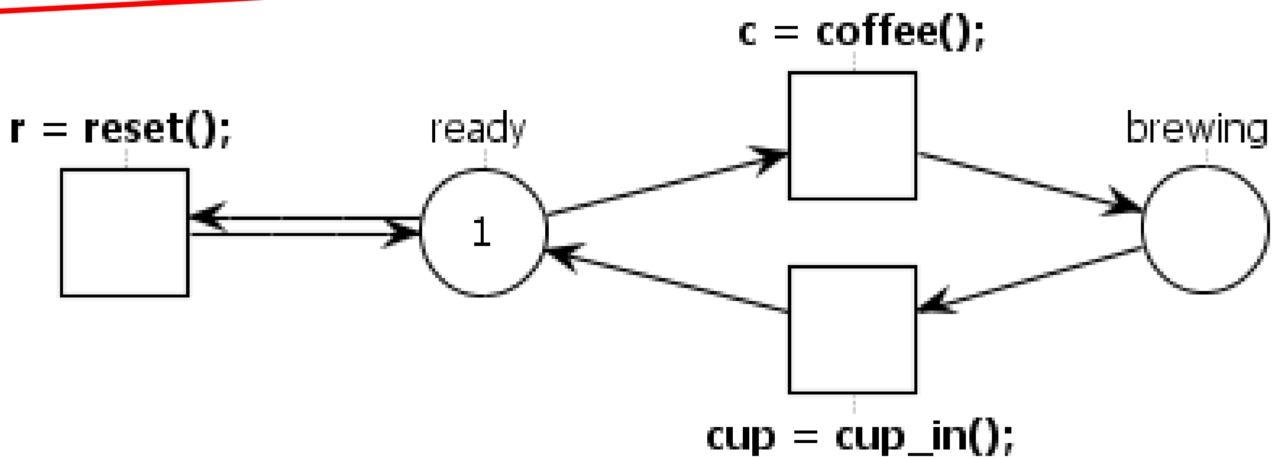


Local behaviour: Coffee

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

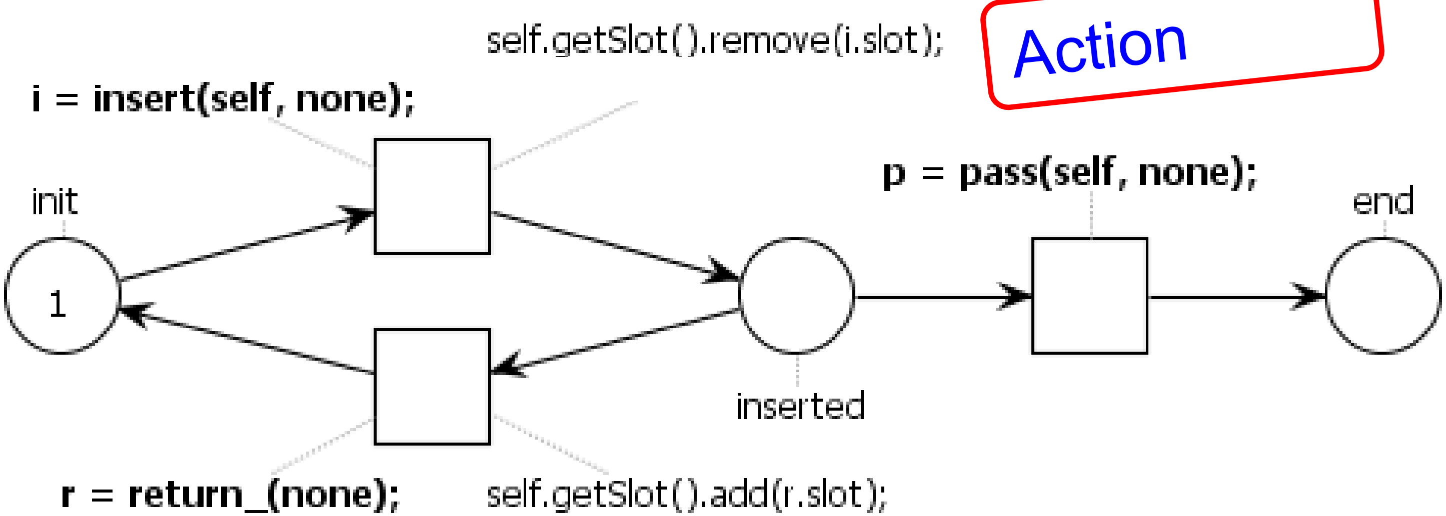
Event binding



Local behaviour: Coin

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

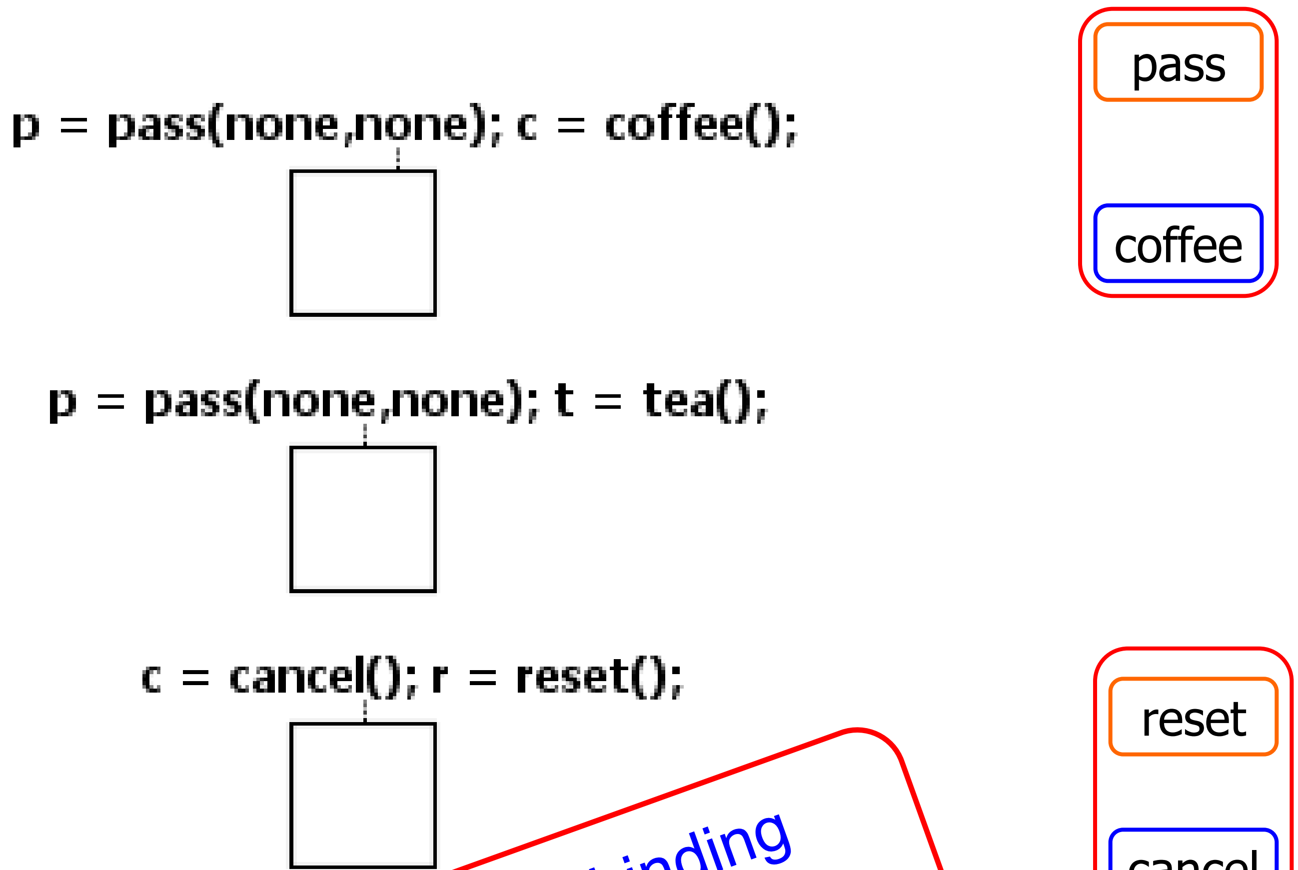
```
import dk.dtu.imm.se.ecno.engine.ExecutionEngine;

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



Action

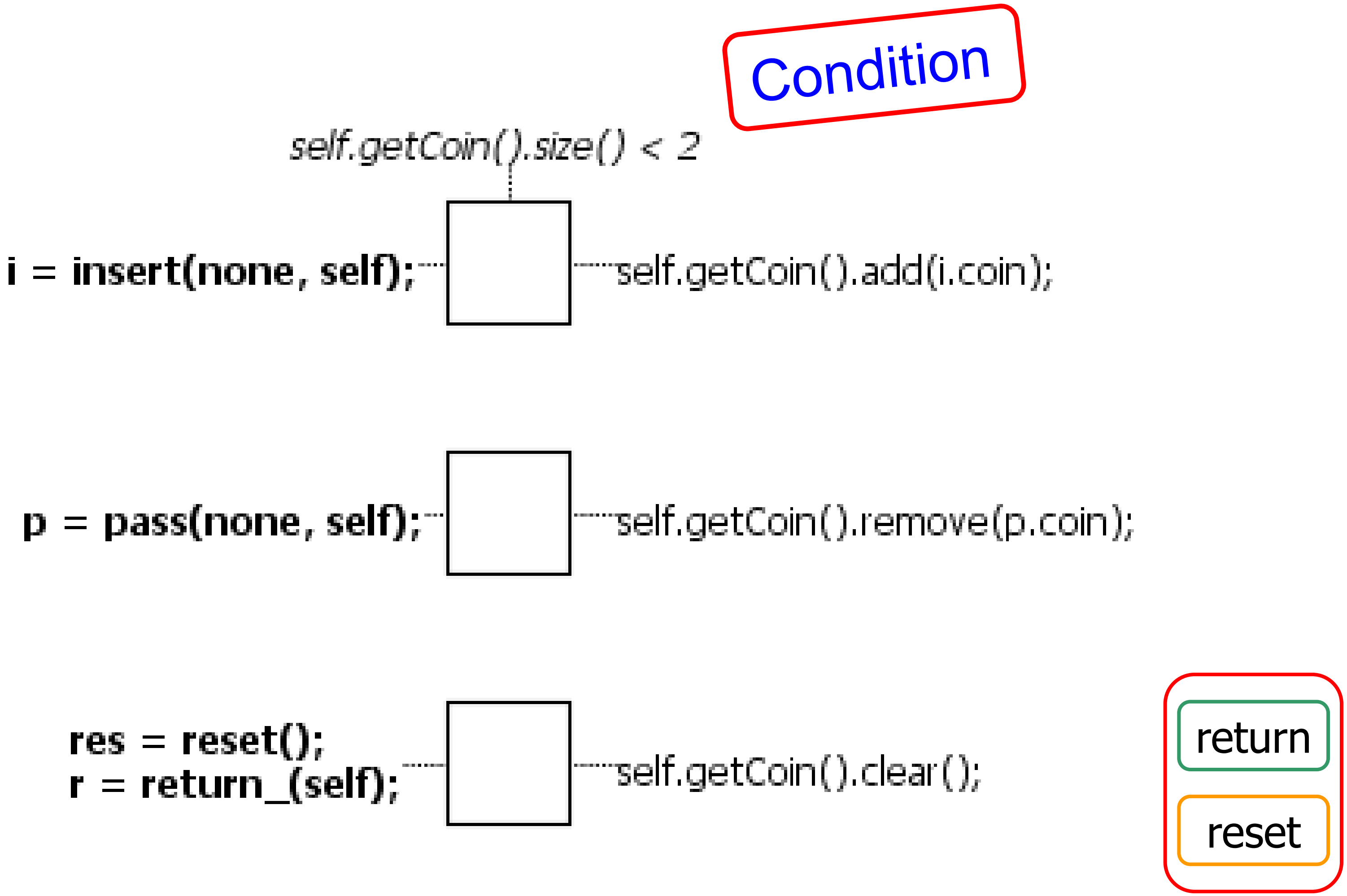
- Event binding
- Parameter assignment

Local behaviour: Control



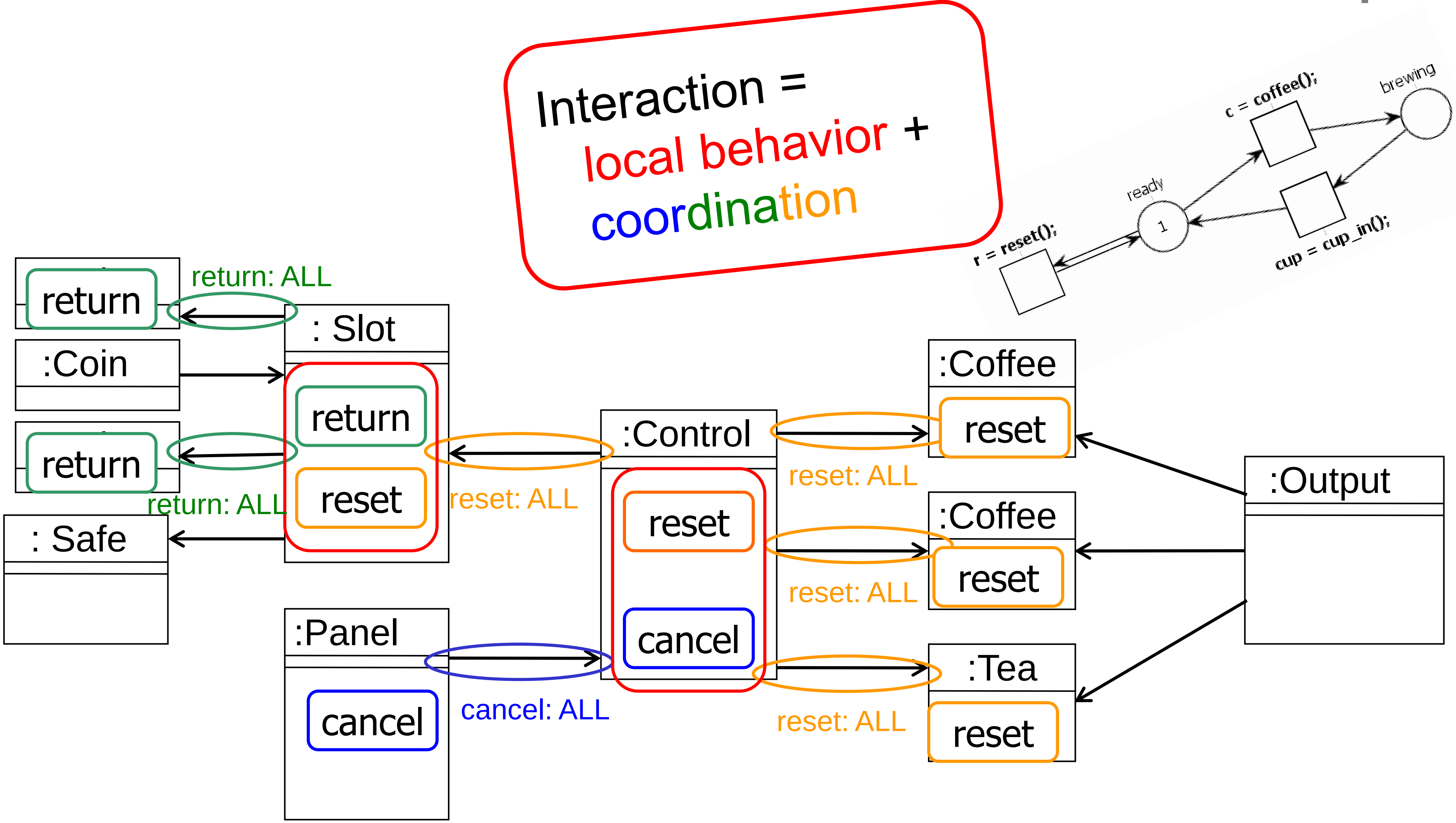
- Event binding with multiple event types!

Local behaviour: Slot



DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Interaction =
local behavior +
coordination



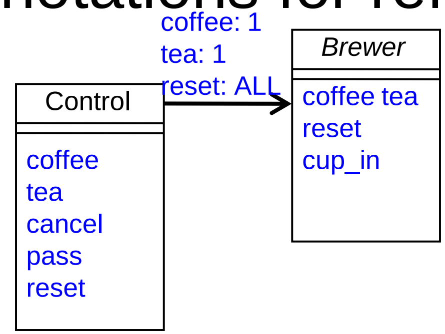
2.2 ECNO: Basic Concepts

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

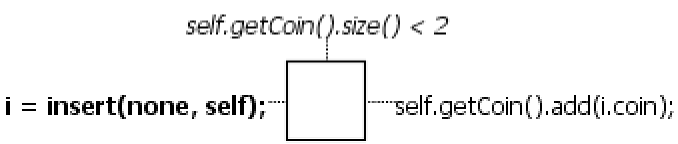
- ElementTypes (Classes)
- EventTypes with
  - parameters

insert(Coin coin, Slot slot)

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



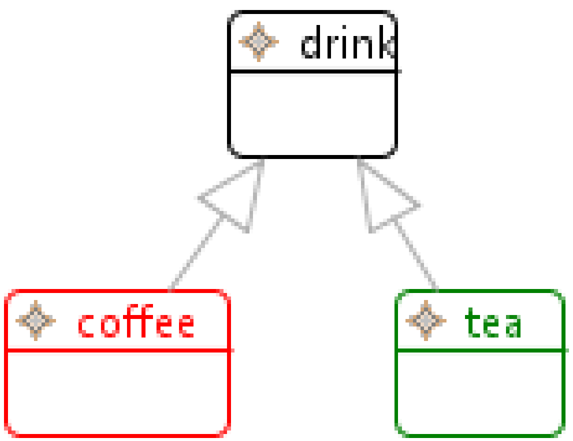
- Local behaviour (life-cycle): ECNO nets (or something else)
  - Event binding (with parameter assignment)
  - Condition
  - Action



2.3 Extensions (cntd.)

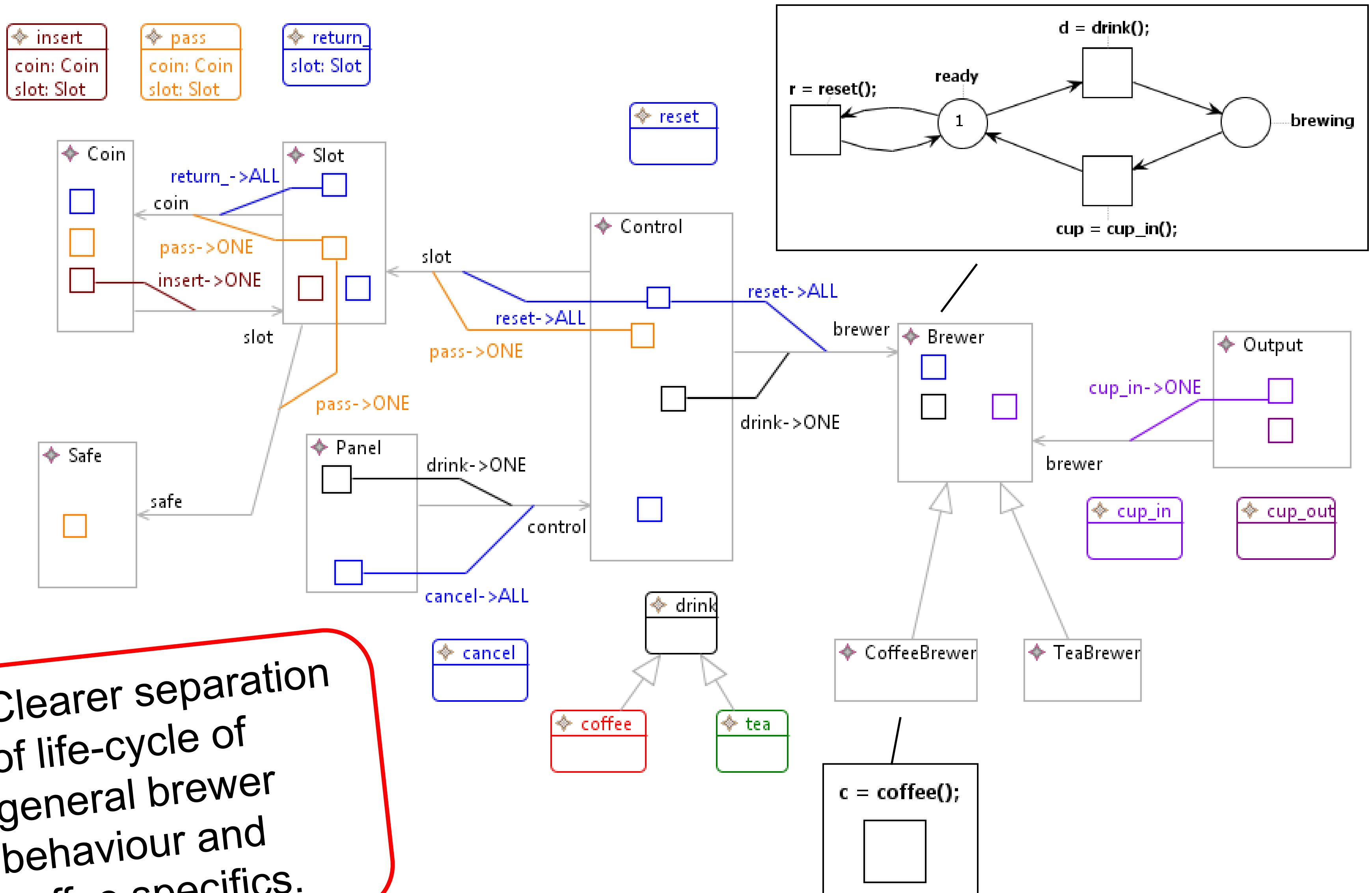
ECNO with its basic concepts has some limitations, which makes modelling things **in an adequate way** a bit painful.

- Sometimes, we want to extend event types later



Behaviour inheritance

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



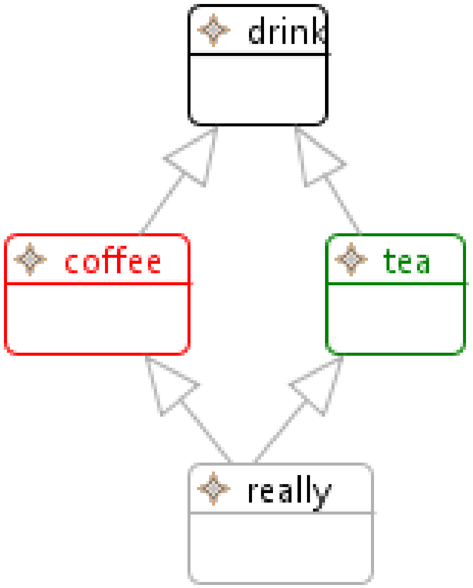
Clearer separation of life-cycle of general brewer behaviour and coffee specifics.

Event Inheritance

Question: Would we like to have multiple inheritance on event types?

Problems:

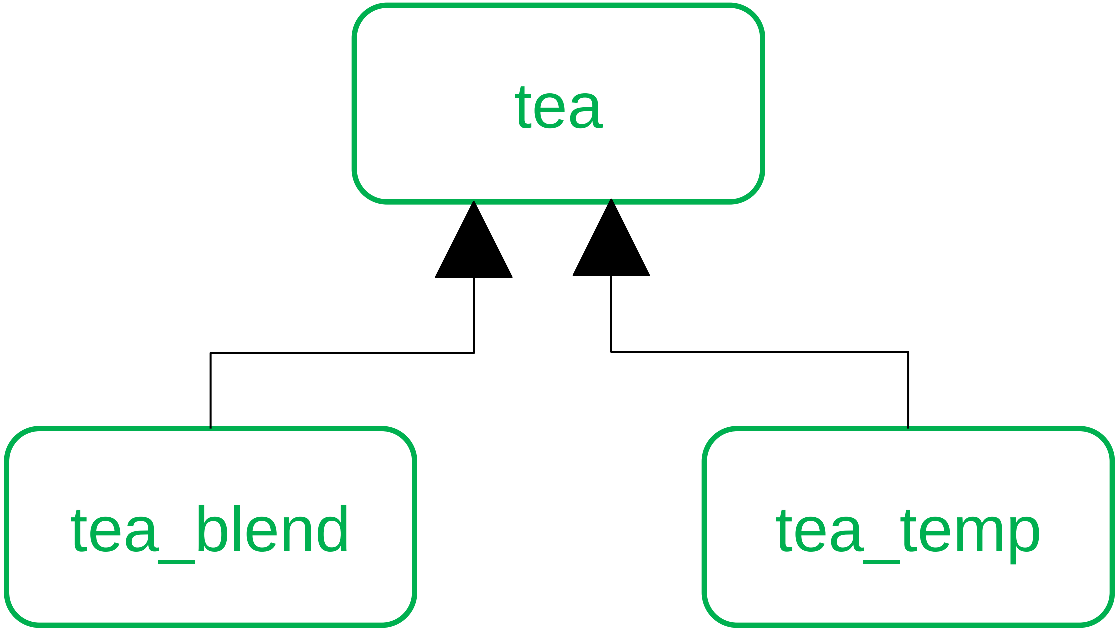
- We could never be sure that two event types that were meant to be different are different!
- We would not know which event type an instance of subtype would represent!



Avoid confusion! Without a really compelling argument, do not introduce multiple inheritance on event types.

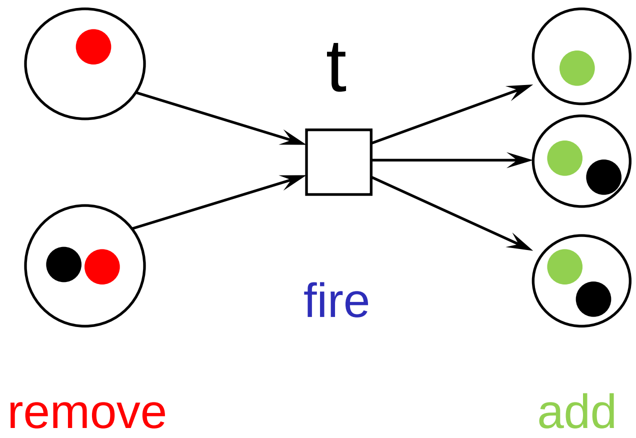
Event Extension

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



=> Two forms of inheritance on event types!

2.4 Example2: Petri nets



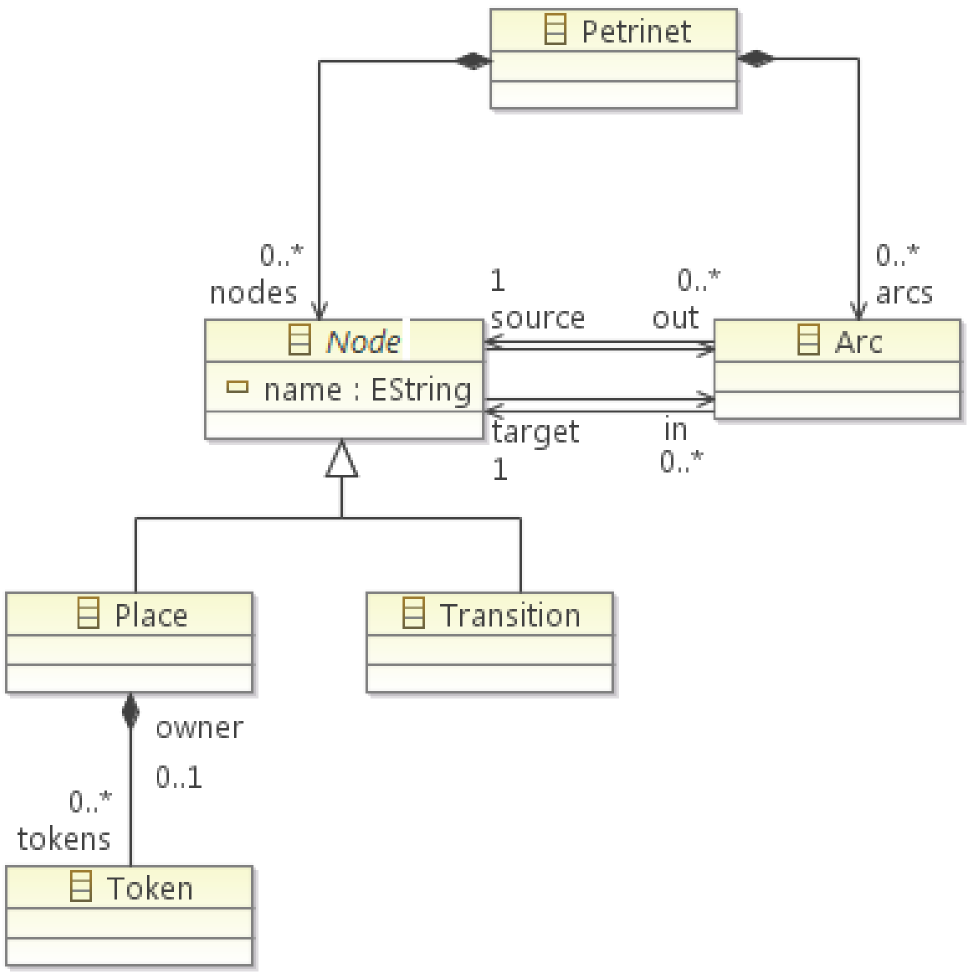
How can we model that behaviour in ECNOs?

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

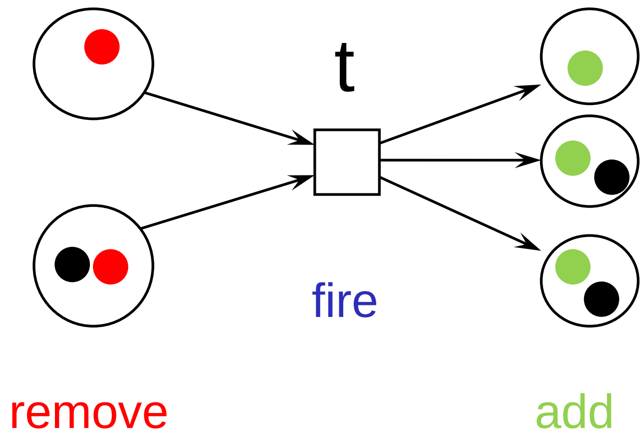
Fire Transition t:
  for ALL incoming Arcs a:
    for ONE source Place p of Arc a:
      find a token and remove it

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

Petri net: Abstract Syntax



Example2: Petri nets



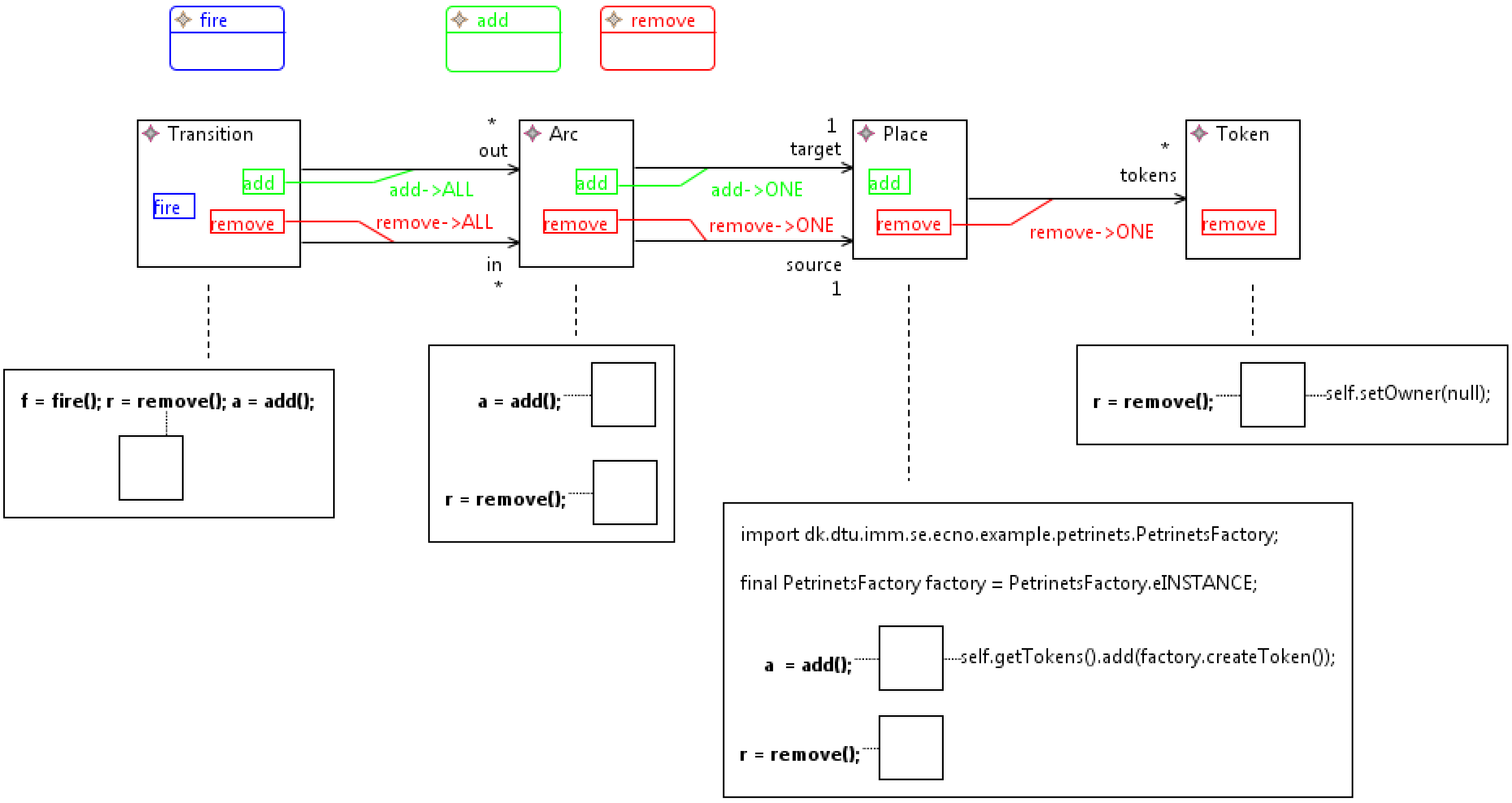
Transition t **enabled**:
  for **ALL** incoming Arcs a:
    for **ONE** source Place p of Arc a:
      find a token

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

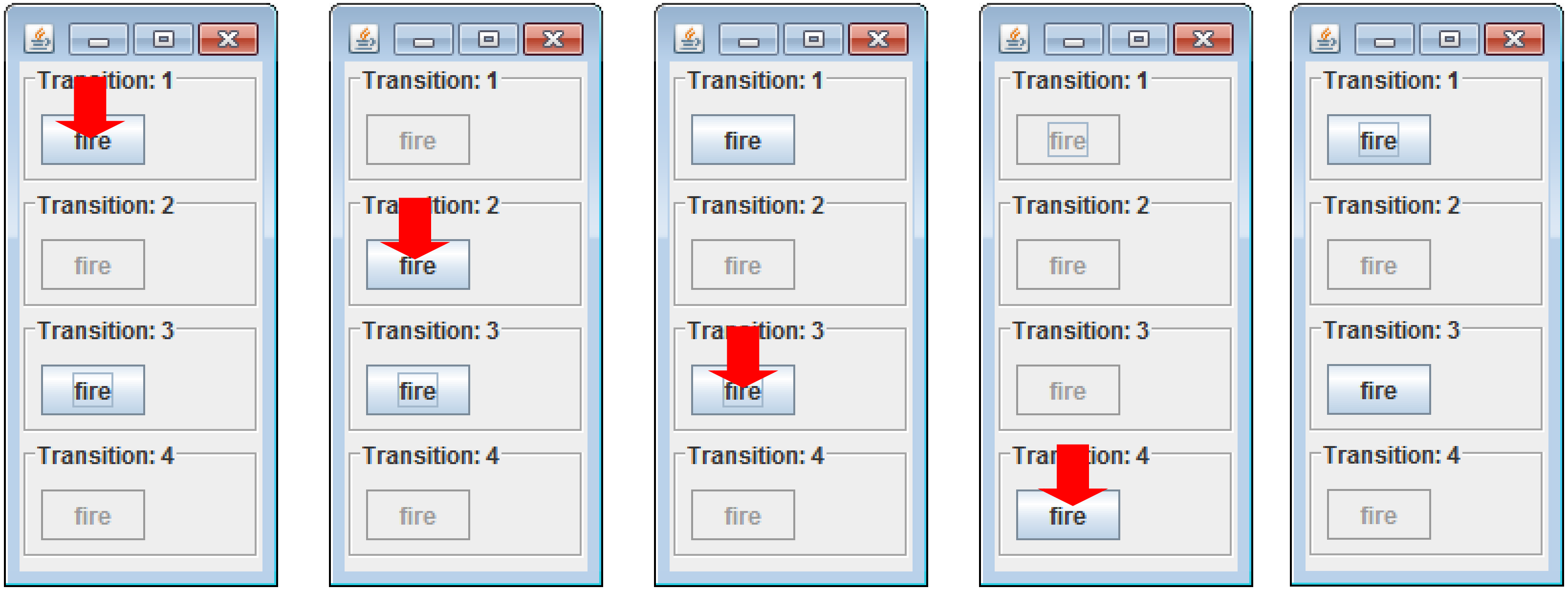
ECNO Semantics of PN

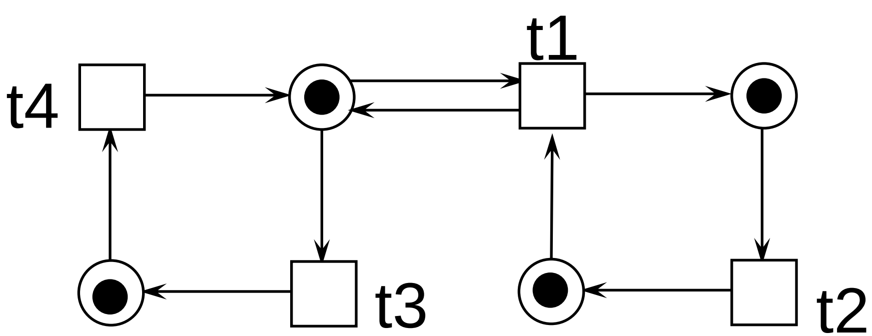
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Result

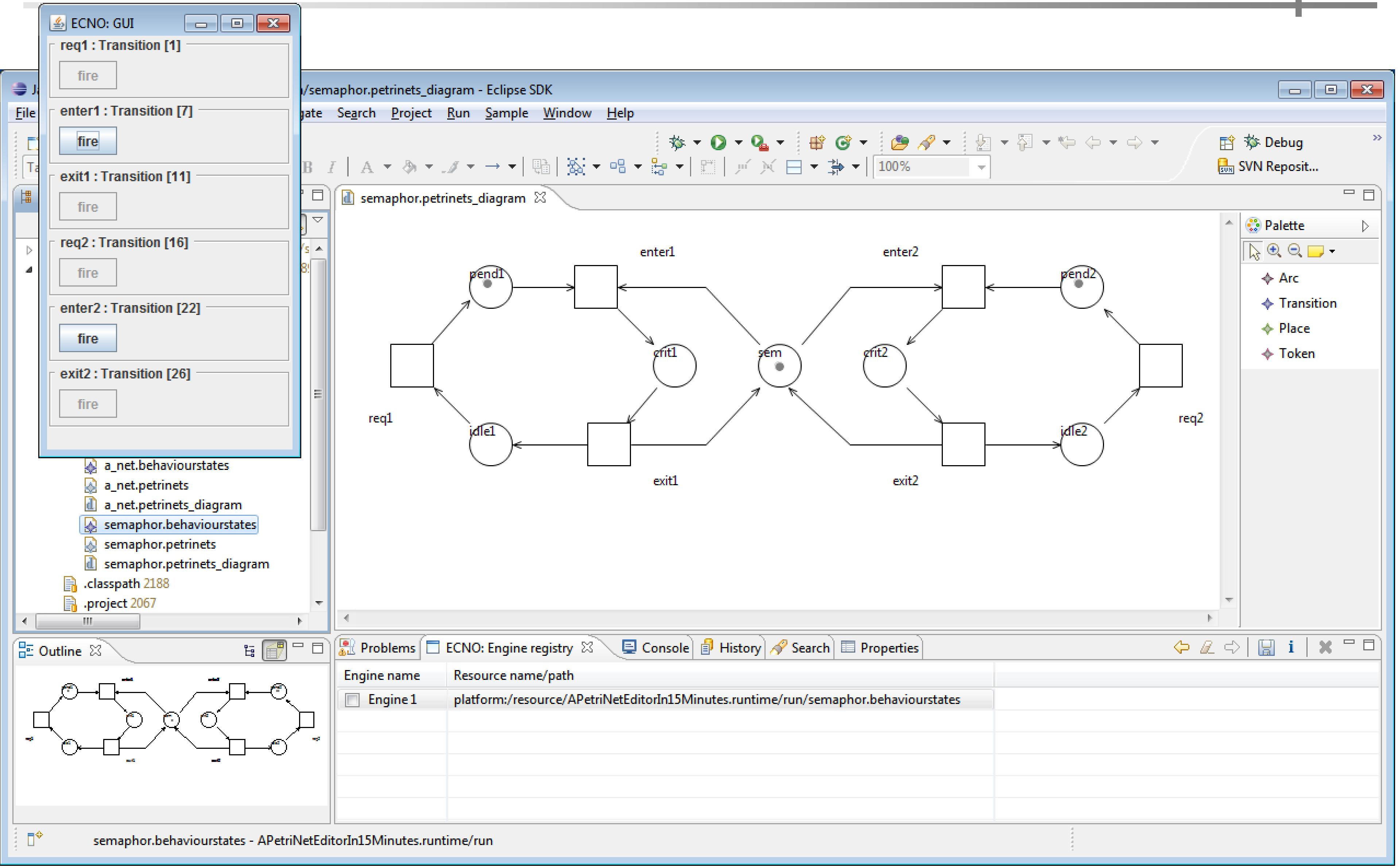
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





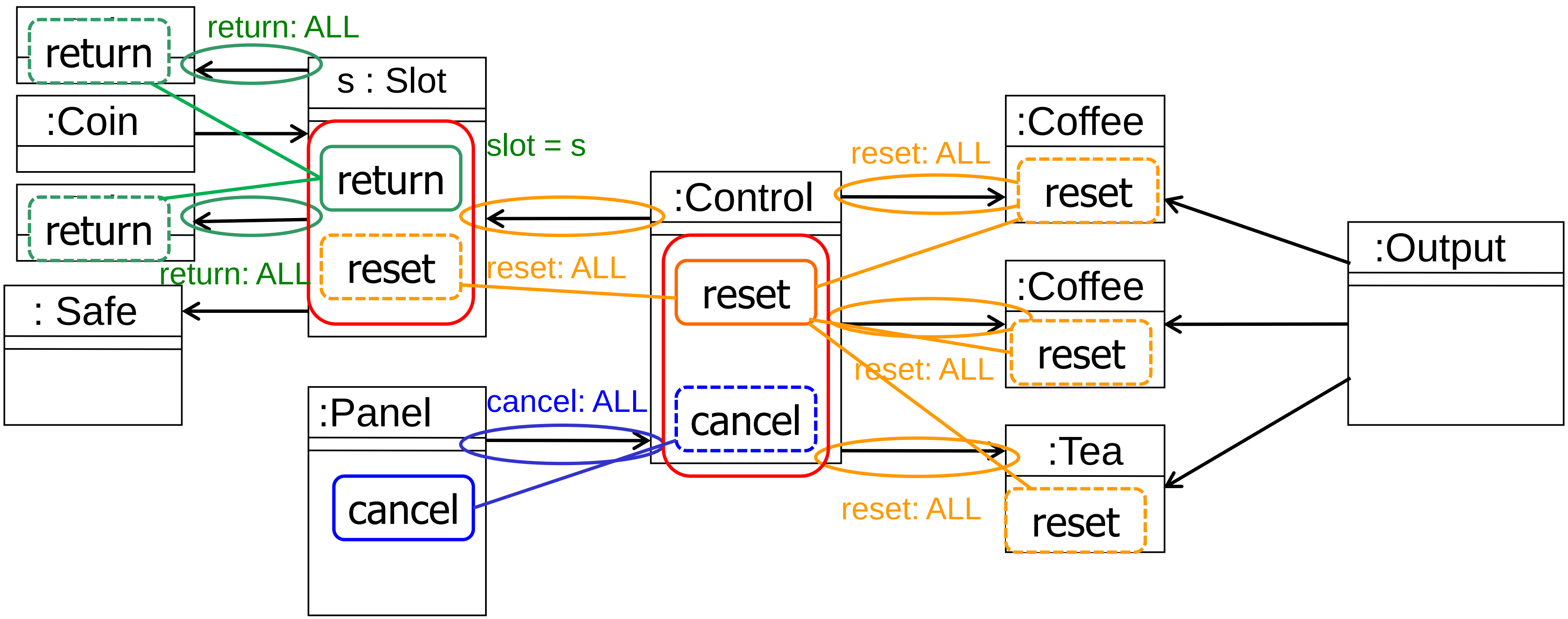
Petri net simulator

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



3. ECNO: Semantics

- Can be formalized in mathematics
  (done for core concepts → ECNO Technical Report)



"Meta ECNO"

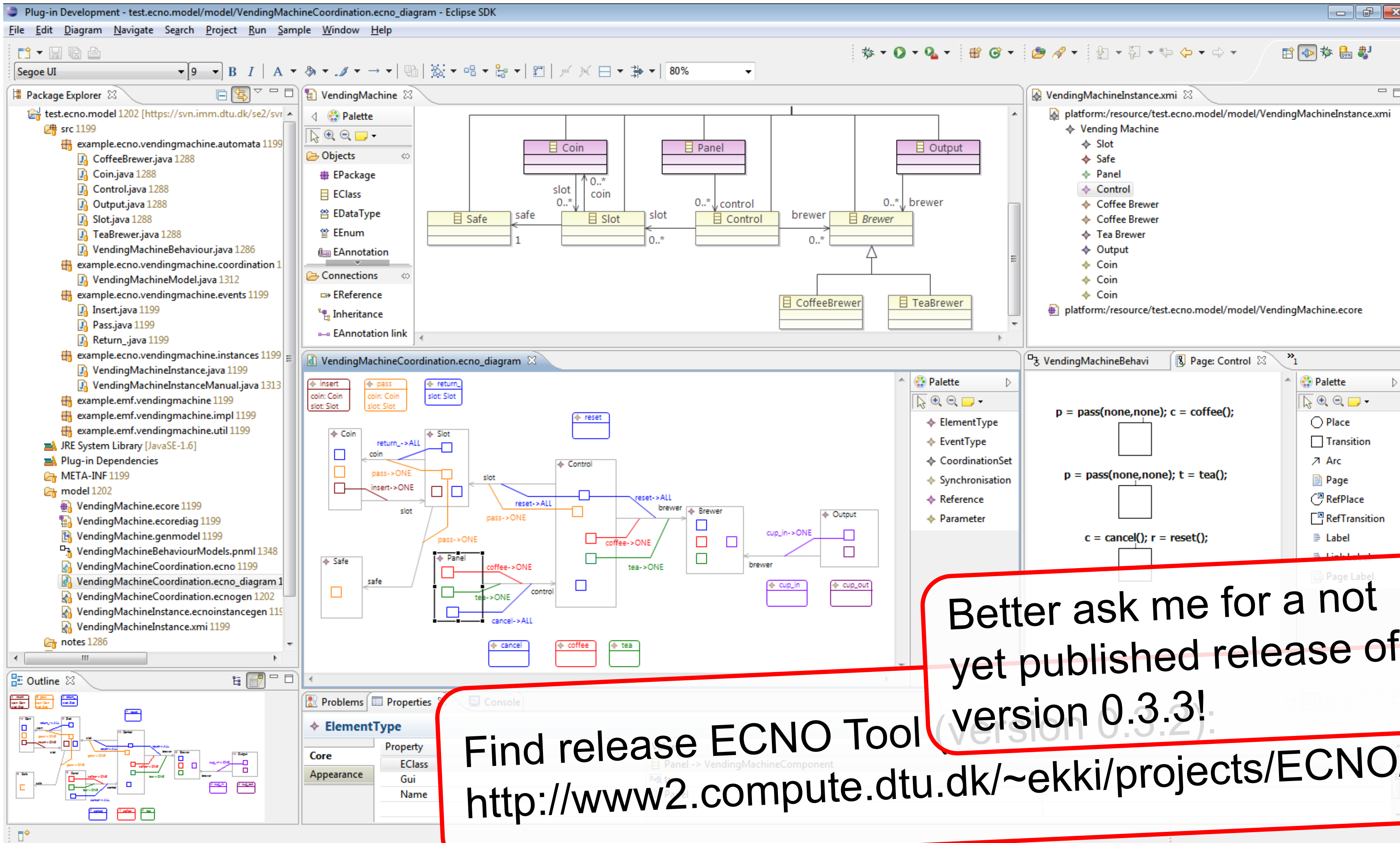
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- The semantics of ECNO could, probably, be expressed in ECNO itself!

I have a sketch (but not fully worked out yet).

Idea similar to PN semantics; needs most of the advanced concepts (except inheritance).

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

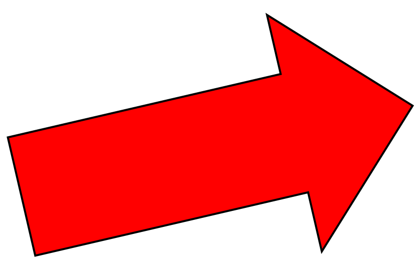


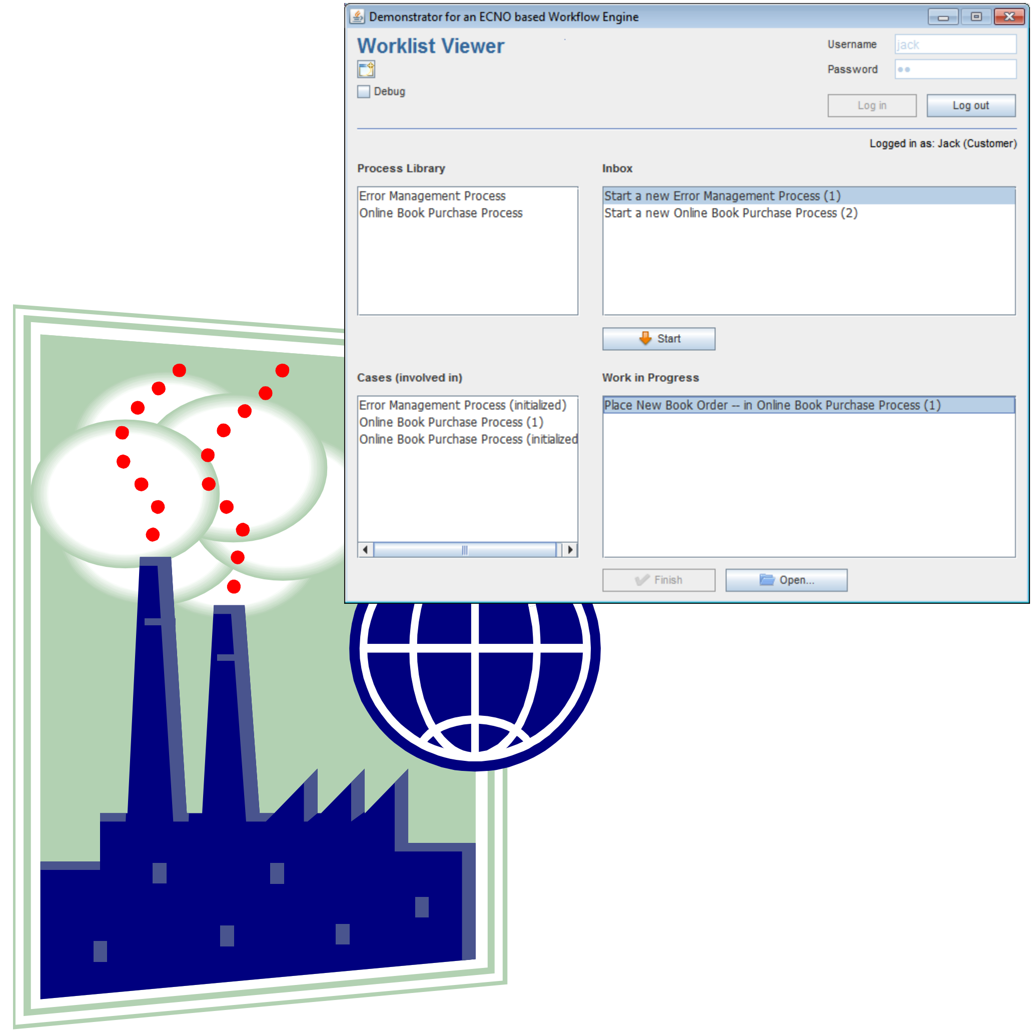
Find release ECNO Tool (version 0.3.2):
http://www2.compute.dtu.dk/~ekki/projects/ECNO/

Better ask me for a not yet published release of version 0.3.3!

Next Steps

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





Next Steps (cntd.)

- Documentation
- Database integration (e.g. Hibernate/Teneo)

- GUI modelling (DSL for GUI for ECNO applications)
- Better IDE support & debugger

- Case studies
- Efficiency / distributed execution
- Standard adapters for other technologies

- **Methodology & education**