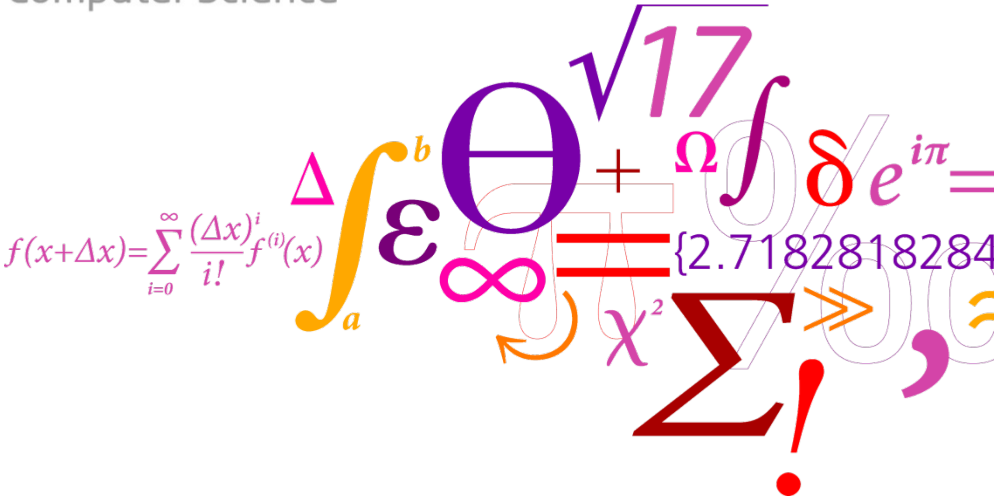
Software Engineering 2
A practical course in software engineering

Ekkart Kindler

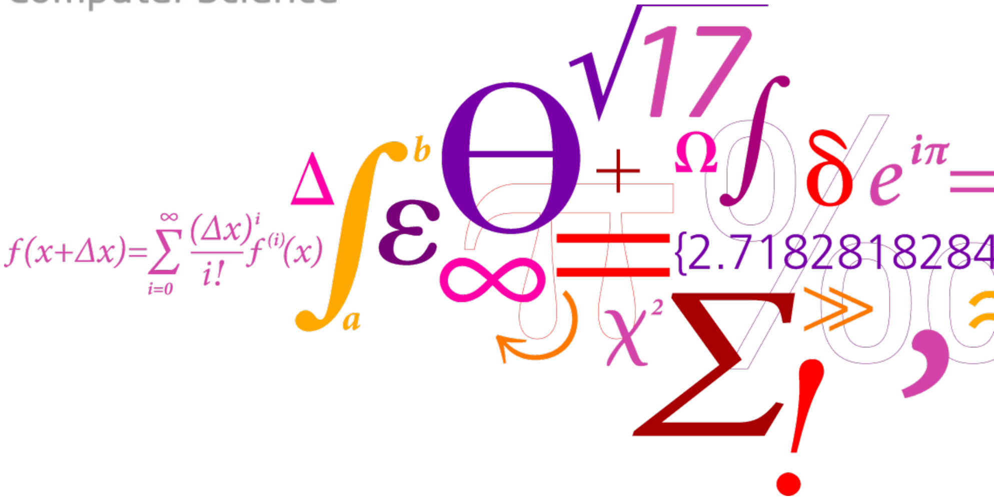
DTU Compute
Department of Applied Mathematics and Computer Science


$f(x+\Delta x)=\sum_{i=0}^{\infty}\frac{(\Delta x)^i}{i!}f^{(i)}(x)$
{2.7182818284

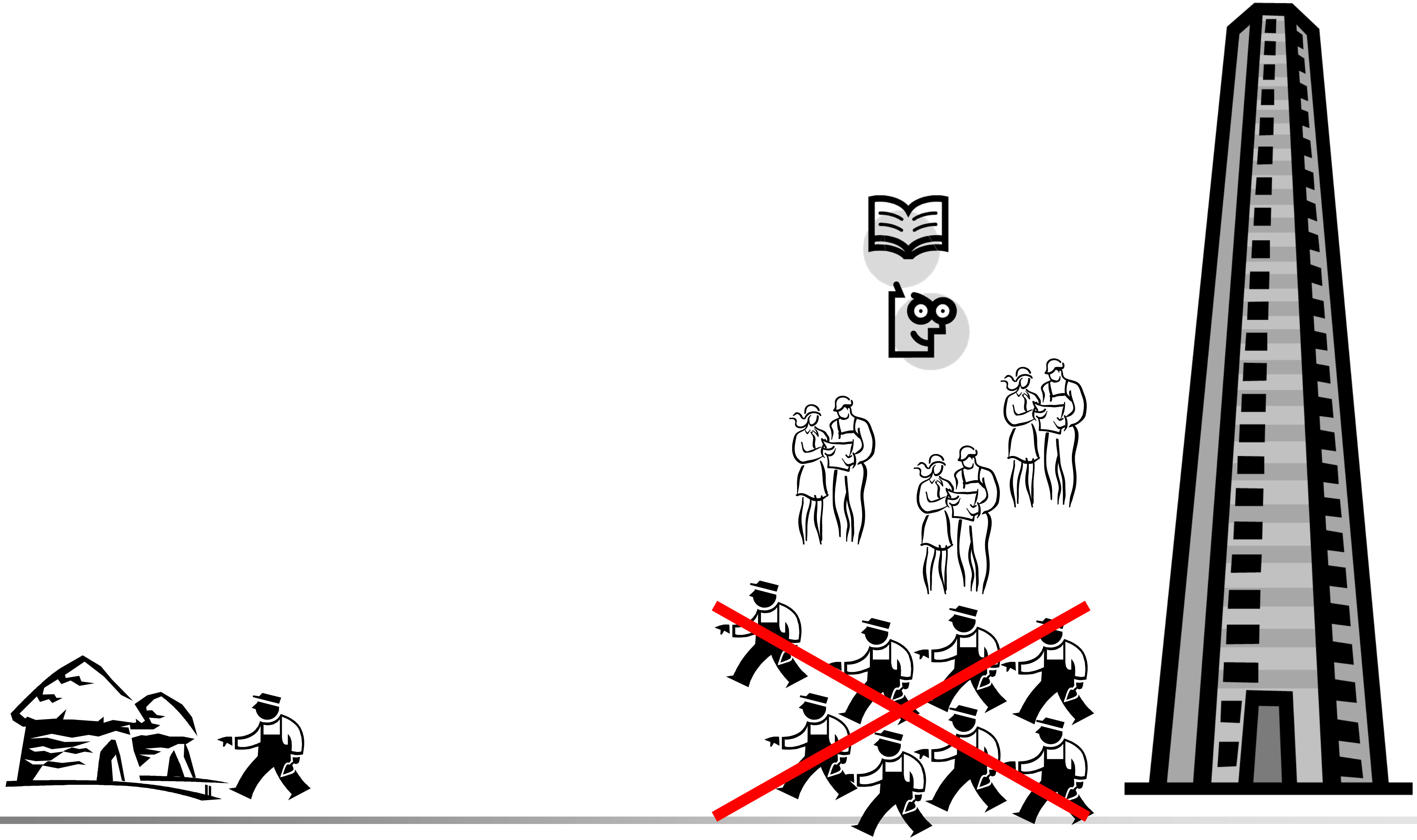
The course in hindsight

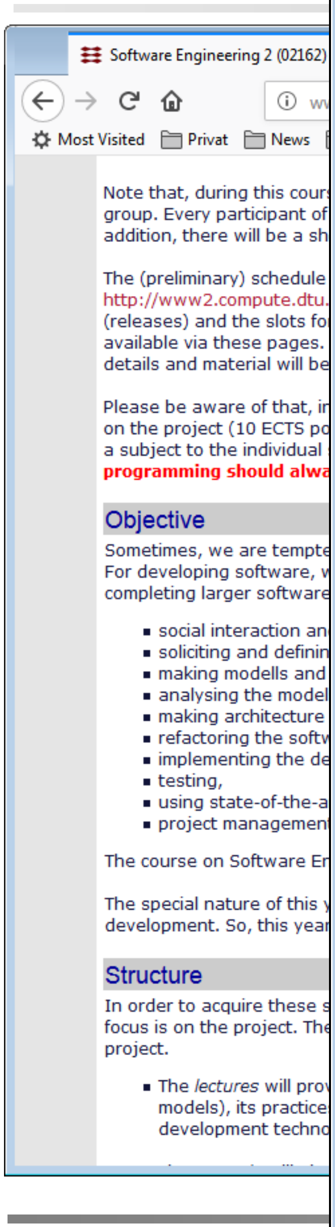
Conclusion, Outlook, Discussion

DTU Compute
Department of Applied Mathematics and Computer Science



Motivation





Note that, during this cour
group. Every participant of
addition, there will be a sh

The (preliminary) schedule
http://www2.compute.dtu.
(releases) and the slots fo
available via these pages.
details and material will be

Please be aware of that, in
on the project (10 ECTS po
a subject to the individual
programming should alwa

Objective
Sometimes, we are tempte
For developing software, w
completing larger software

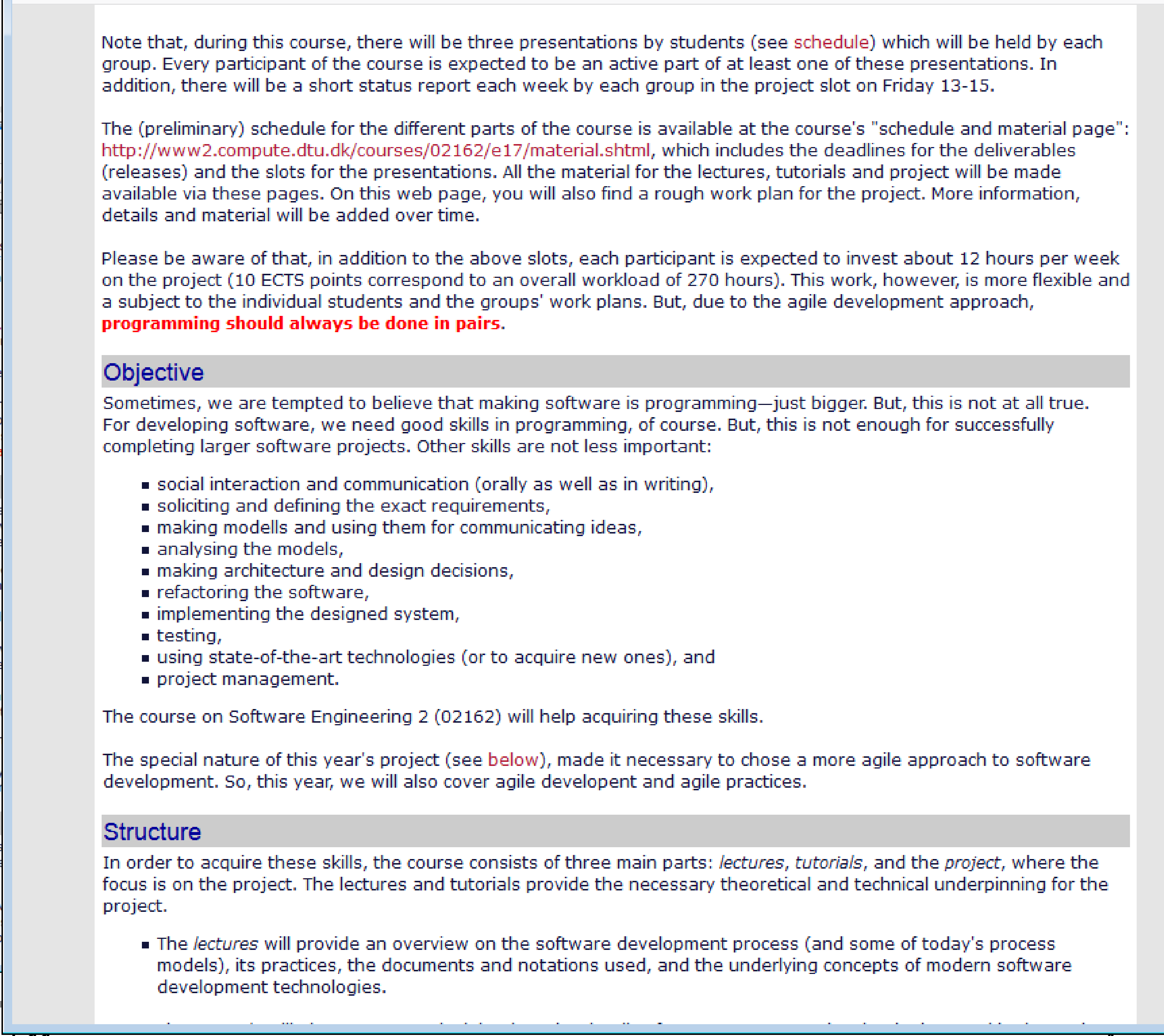
- social interaction an
- soliciting and definin
- making modells and
- analysing the model
- making architecture
- refactoring the softw
- implementing the de
- testing,
- using state-of-the-a
- project managemen

The course on Software Er

The special nature of this y
development. So, this year

Structure
In order to acquire these s
focus is on the project. The
project.

- The *lectures* will prov
models), its practice
development techno



Note that, during this course, there will be three presentations by students (see schedule) which will be held by each group. Every participant of the course is expected to be an active part of at least one of these presentations. In addition, there will be a short status report each week by each group in the project slot on Friday 13-15.

The (preliminary) schedule for the different parts of the course is available at the course's "schedule and material page": http://www2.compute.dtu.dk/courses/02162/e17/material.shtml, which includes the deadlines for the deliverables (releases) and the slots for the presentations. All the material for the lectures, tutorials and project will be made available via these pages. On this web page, you will also find a rough work plan for the project. More information, details and material will be added over time.

Please be aware of that, in addition to the above slots, each participant is expected to invest about 12 hours per week on the project (10 ECTS points correspond to an overall workload of 270 hours). This work, however, is more flexible and a subject to the individual students and the groups' work plans. But, due to the agile development approach, **programming should always be done in pairs**.

Objective

Sometimes, we are tempted to believe that making software is programming—just bigger. But, this is not at all true. For developing software, we need good skills in programming, of course. But, this is not enough for successfully completing larger software projects. Other skills are not less important:

- social interaction and communication (orally as well as in writing),
- soliciting and defining the exact requirements,
- making modells and using them for communicating ideas,
- analysing the models,
- making architecture and design decisions,
- refactoring the software,
- implementing the designed system,
- testing,
- using state-of-the-art technologies (or to acquire new ones), and
- project management.

The course on Software Engineering 2 (02162) will help acquiring these skills.

The special nature of this year's project (see below), made it necessary to chose a more agile approach to software development. So, this year, we will also cover agile developent and agile practices.

Structure

In order to acquire these skills, the course consists of three main parts: *lectures*, *tutorials*, and the *project*, where the focus is on the project. The lectures and tutorials provide the necessary theoretical and technical underpinning for the project.

- The *lectures* will provide an overview on the software development process (and some of today's process models), its practices, the documents and notations used, and the underlying concepts of modern software development technologies.

1. What did you learn?

- Objectives of this course:
  Basic skills in software engineering!

What did you learn?

What is important?

Software Engineering is

… much more than programming!

… listening and understanding!
… analytic and conceptual work!
… communication!
… a social process!
…
… acquiring new technologies!

… a discipline with proven concepts, methods, notations, and tools!
… and ever new technologies emerging!

Experience

Software Engineering requires much experience!

This experience

- can not be taught theoretically!
- will be provided in this course!

→ tutorial
→ project
→ and (only) backed by the lectures

Issues

How to make useful UML models

- Domain model / analysis OOA
- Architecture and design OOD

- Learn to use new technologies **YOURSELF**!

  ASP.NET / Entity Framework / React /
  GitLab / Jenkins / …

You did it!?

Issues

- Get requirements straight

- Write a systems specification

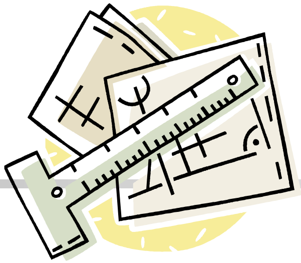
Important: Have another look at your documents now and ask yourself: What should have been in there? And fix it for final submission!

Specifying Software

And "why"! (objective)

what

- Project Definition
- Requirements Specification
  - rough
  - detailed
- Systems specification
- Complete Models
- Implementation, Documentation Handbook



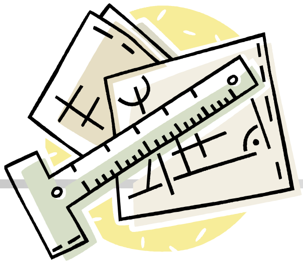
how

Specifying Software

- Project Definition
- Requirements Specification
  - rough
  - detailed
- Systems specification
- Complete Models
- Implementation, Documentation Handbook

rough

detailed

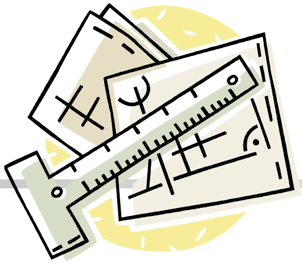


Specifying Software

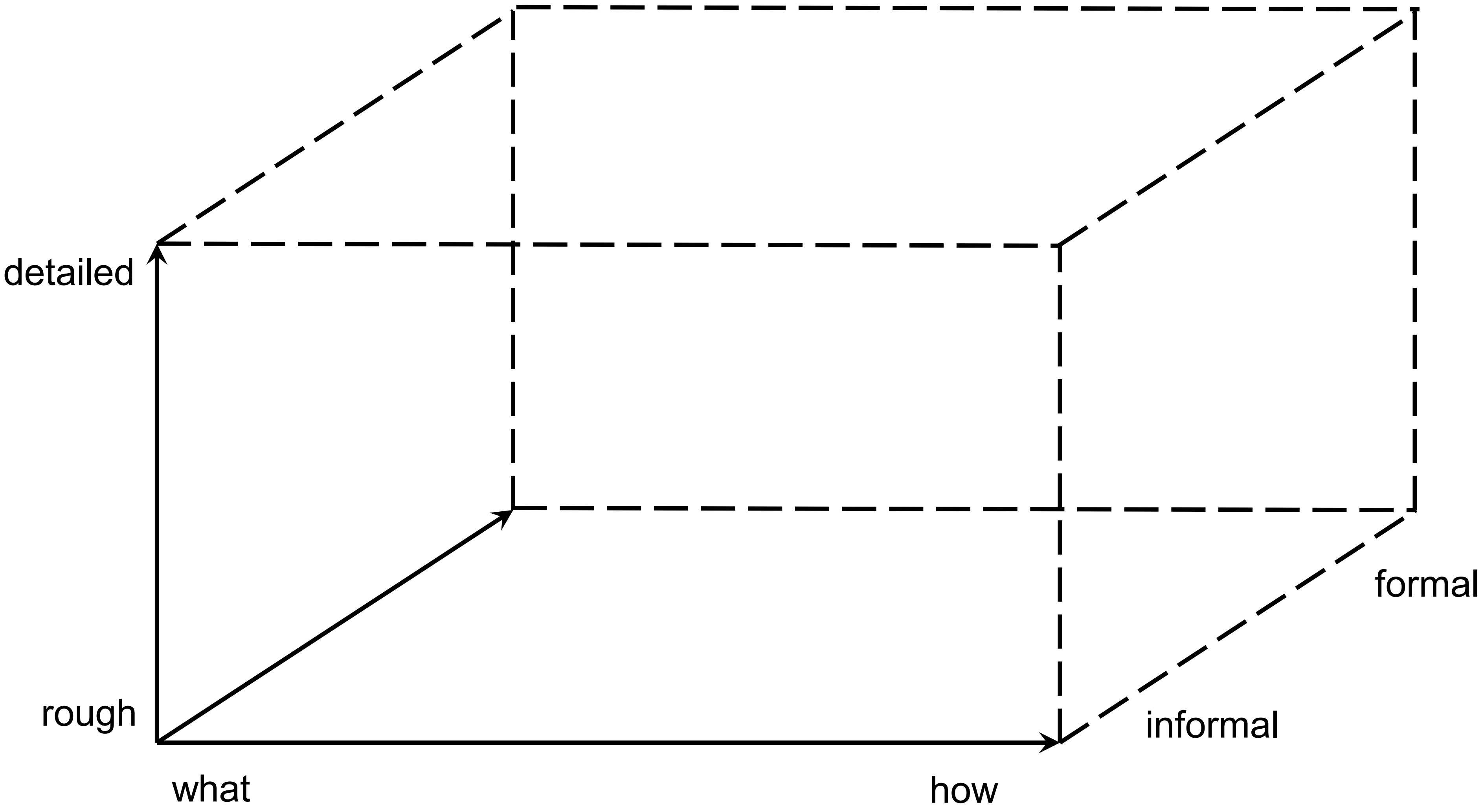
- Project Definition
- Requirements Specification
  - rough
  - detailed
- Systems specification
- Complete Models
- Implementation, Documentation Handbook

low cost

high cost



Dimensions



Issues

- writing,
- talking,
- communicating, and
- organizing yourself

- work together
- version management
- Collaborative Development Environments

Explicit and concrete communication!

Issues

- Quality
  - Management
  - Testing
  - Reviews

Note: In today's group meetings, Tobias and I will do another informal code walkthrough!

- Many practical issues on programming and solving problems

databases docker

Rest Jenkins JUnit component

model

YAML C# debugger controller React JS

Issues

- Integration and extension
  - Integrating features in existing software (PlugIn Mechanisms, …)
  - Developing parts in parallel (based on a common model)
  - Separating concerns
  - **Stepwise extension** (prototyping, agile)
  - …

Topics

- Software Specifications (incl. writing)
- Modelling & Meta modelling
- Quality mangament (incl. testing)
- Code generation
- Working together
- Management

The main point of this course is NOT on the acquired knowledge!

It is on APPLYING it (in a meaningful way): acquiring SKILLS!

Agile manifesto

„We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.“

Kent Beck et al. 2001

Agile Values

- Communication
- Simplicity
- Feedback
- Courage

Agile (fundamental) Principals

- Rapid feedback
- Assume Simplicity
- Incremental change
- Embracing change
- Quality work

Agile Practices (overview)

- **On-site customer**
- **Small/short releases** 2-3 week
- **Planning game**
- **Coding standards**
- **Testing**
- **Continuous integration**
- **Pair programming**
- **Simple design**
- **Refactoring**
- **Collective ownership**
- **...**

It is the combination or practices, that makes agile work!

2. What did you not learn?

Concerning technology and complexity of real software, you have just seen the tip of the iceberg!

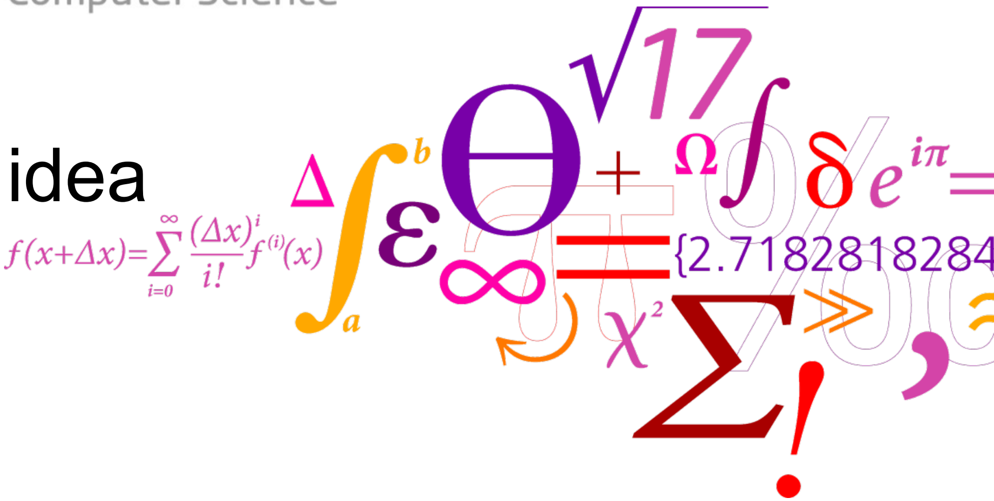
Technology issues

- Meta modelling (MOF Meta Object Facility)
- Software without Programming (EMF and more) / code generation technologies
- Other technologies: other application servers, cloud, IoT, databases, service technologies, …
- Analyse, validate, verify software, …
- Other programming and modelling paradigms: e.g. Aspect Oriented Modelling
- ...

Advanced Topics

DTU Compute
Department of Applied Mathematics and Computer Science

Just to give you some idea



$f(x+\Delta x)=\sum_{i=0}^{\infty}\frac{(\Delta x)^i}{i!}f^{(i)}(x)$
$\Delta\int_a^b \varepsilon\,\Theta^{\sqrt{17}}+\Omega\int\delta e^{i\pi}=$
∞
{2.7182818284
χ^2
Σ!

Future technologies

- Modelling dynamic behaviour
  (and generating code from that)
- Get completely rid of programming?!
- More IoT and cloud technologies

More details & experience

- Master courses
  - Systems integration (H.B.)
  - Web Services (H.B.)
  - Formal methods (A.H.)
  - Special courses (E.K.)

- Bachelor and master projects

  (last slides)

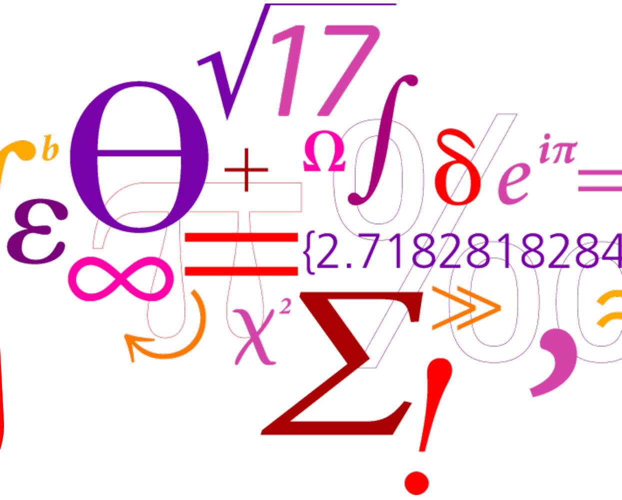
Coordinating Interactions
The Event Coordination Notation

Ekkart Kindler

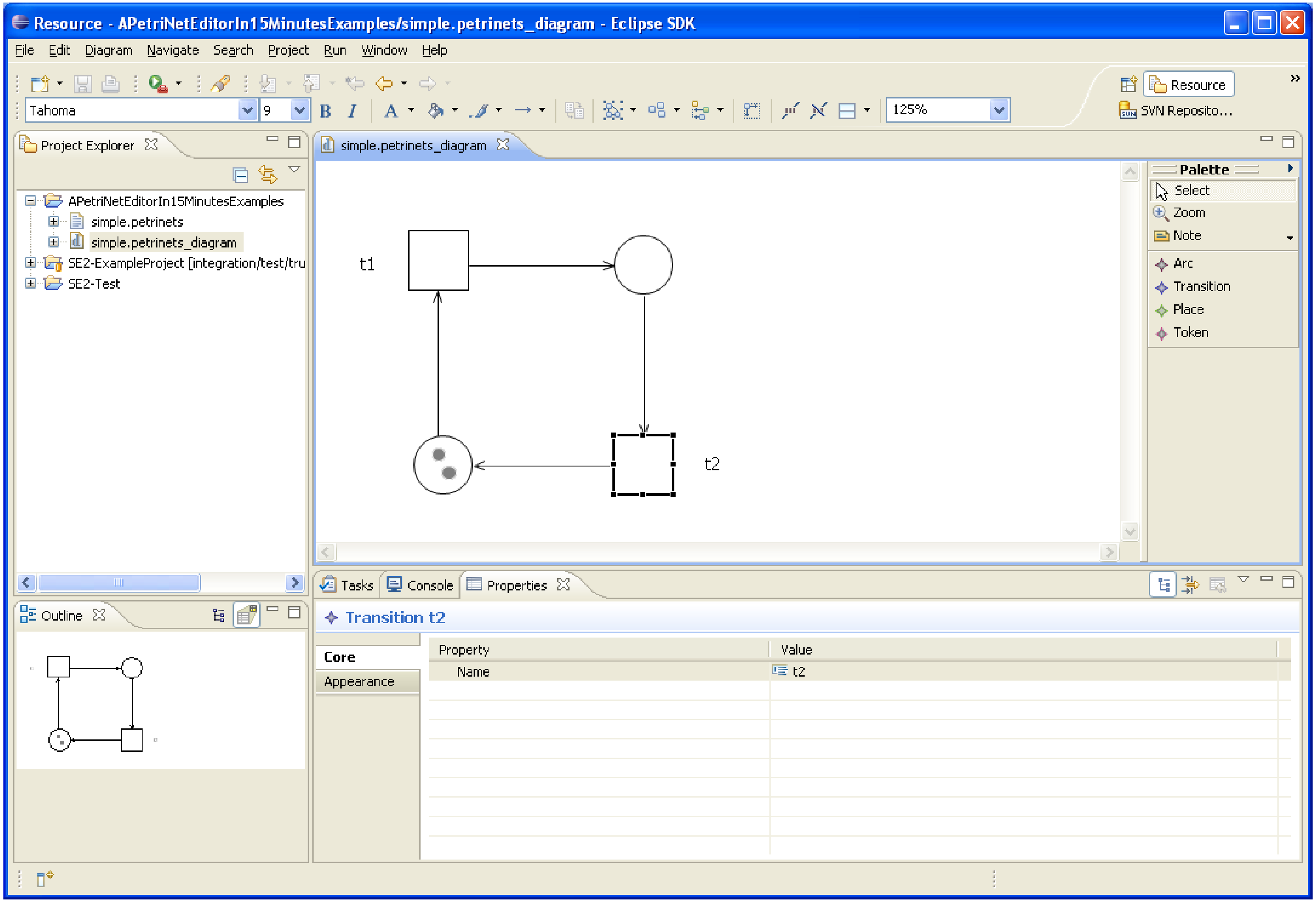
DTU Compute
Department of Applied Mathematics and Computer Science

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

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



From lecture 1: Example

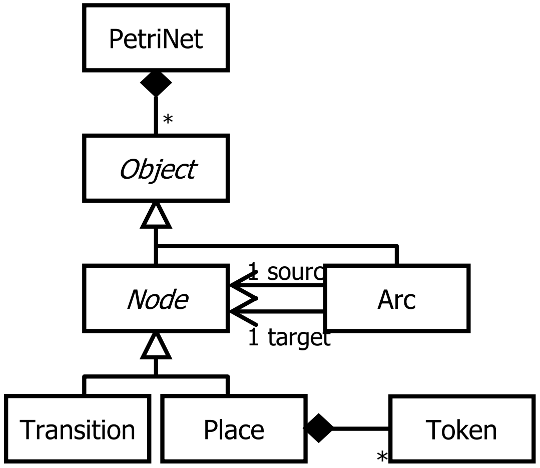


Property	Value
Name	t2

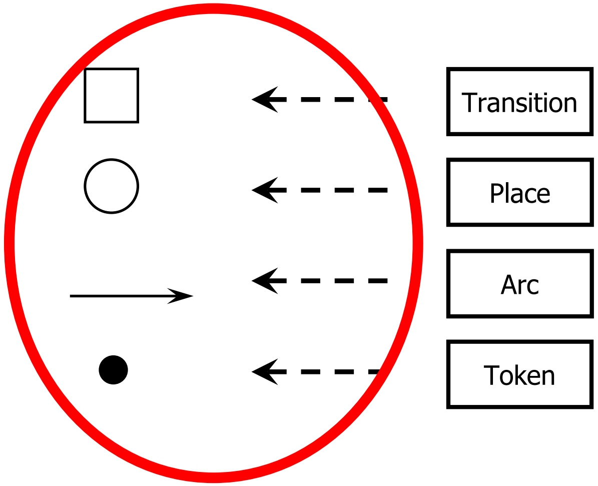
From lecture 1: Example

From this (EMF) model for Petri nets:
Generation of (Java) code for

- all classes
- methods for changing the Petri net
- loading and saving the Petri net as XML files (→XMI)



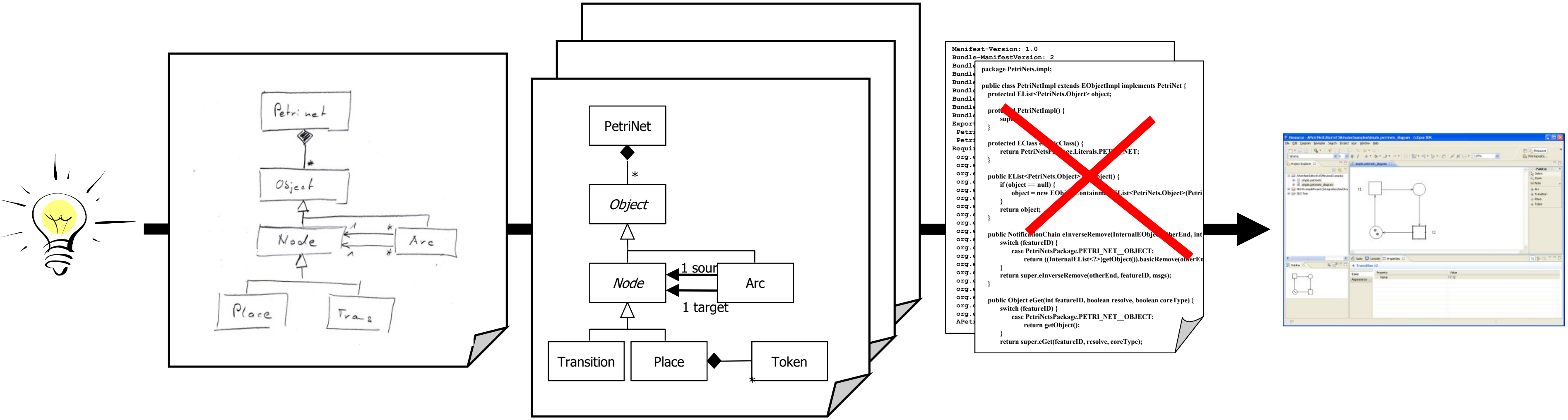
With this and some more GMF information:
Generation of the Java code of a graphical complete editor (with many fancy features). No programming at all.



Almost all you need to say about a Petri net editor.

From lecture 1

How about behaviour ?
(non-standard behaviour)



```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle
Bundle
Bundle
Bundle
Bundle
Bundle
Bundle
Export
 Petri
 Petri
Requir
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 org.e
 APetr
```

```
package PetriNets.impl;

public class PetriNetImpl extends EObjectImpl implements PetriNet {
   protected EList<PetriNets.Object> object;

   prote   PetriNetImpl() {
         supe
   }

   protected EClass e    icClass() {
         return PetriNetsP    ge.Literals.PET    NET;
   }

   public EList<PetriNets.Object>     bject() {
         if (object == null) {
               object = new EObj    Containme    EList<PetriNets.Object>(Petri
         }
         return object;
   }

   public NotificationChain eInverseRemove(InternalEObje    therEnd, int
         switch (featureID) {
               case PetriNetsPackage.PETRI_NET__OBJECT:
                     return ((InternalEList<?>)getObject()).basicRemove(otherEn
         }
         return super.eInverseRemove(otherEnd, featureID, msgs);
   }

   public Object eGet(int featureID, boolean resolve, boolean coreType) {
         switch (featureID) {
               case PetriNetsPackage.PETRI_NET__OBJECT:
                     return getObject();
         }
         return super.eGet(featureID, resolve, coreType);
```

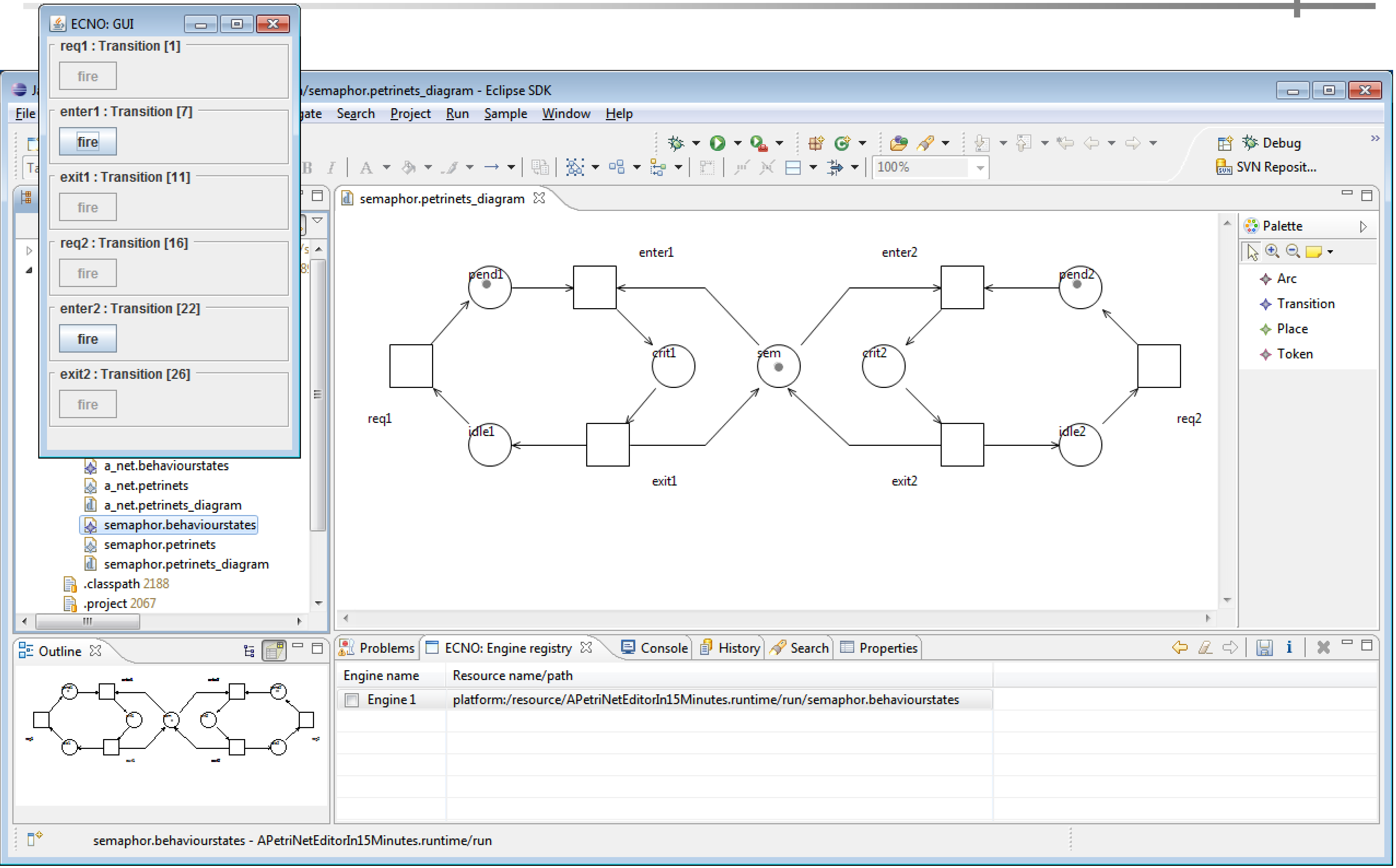
Analysis

Design

Implementation

~~Coding~~

e.g. a Petri net simulator?



Engine name	Resource name/path
□ Engine 1	platform:/resource/APetriNetEditorIn15Minutes.runtime/run/semaphor.behaviourstates

semaphor.behaviourstates - APetriNetEditorIn15Minutes.runtime/run

Motivation

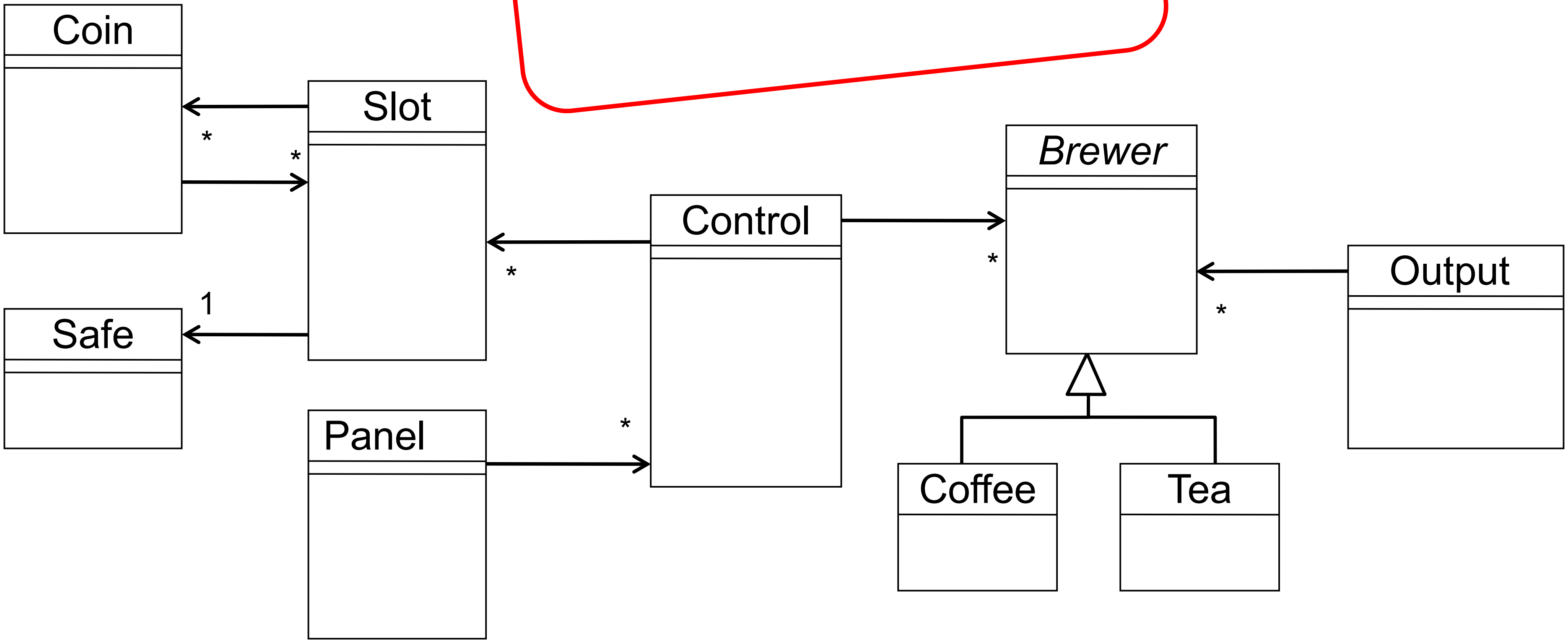
- Given some object oriented software with (or without) explicit domain model
- Model behaviour on top of it – and make these models executable
- Model behaviour on a high level of abstraction (domain): coordination of behaviour

→ Integrate behaviour models with structural models

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

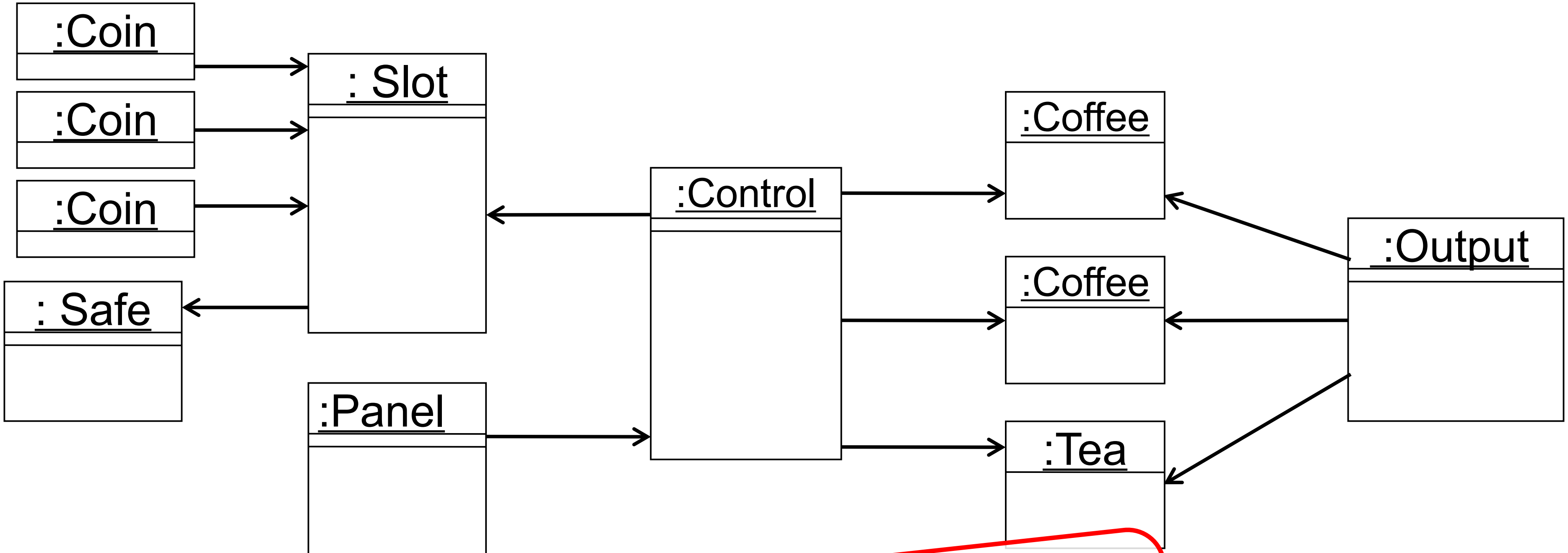
2.1 Example 1: Vending machine

Class diagram as usual



Instance: Object Diagram

Initial configuration, current situation

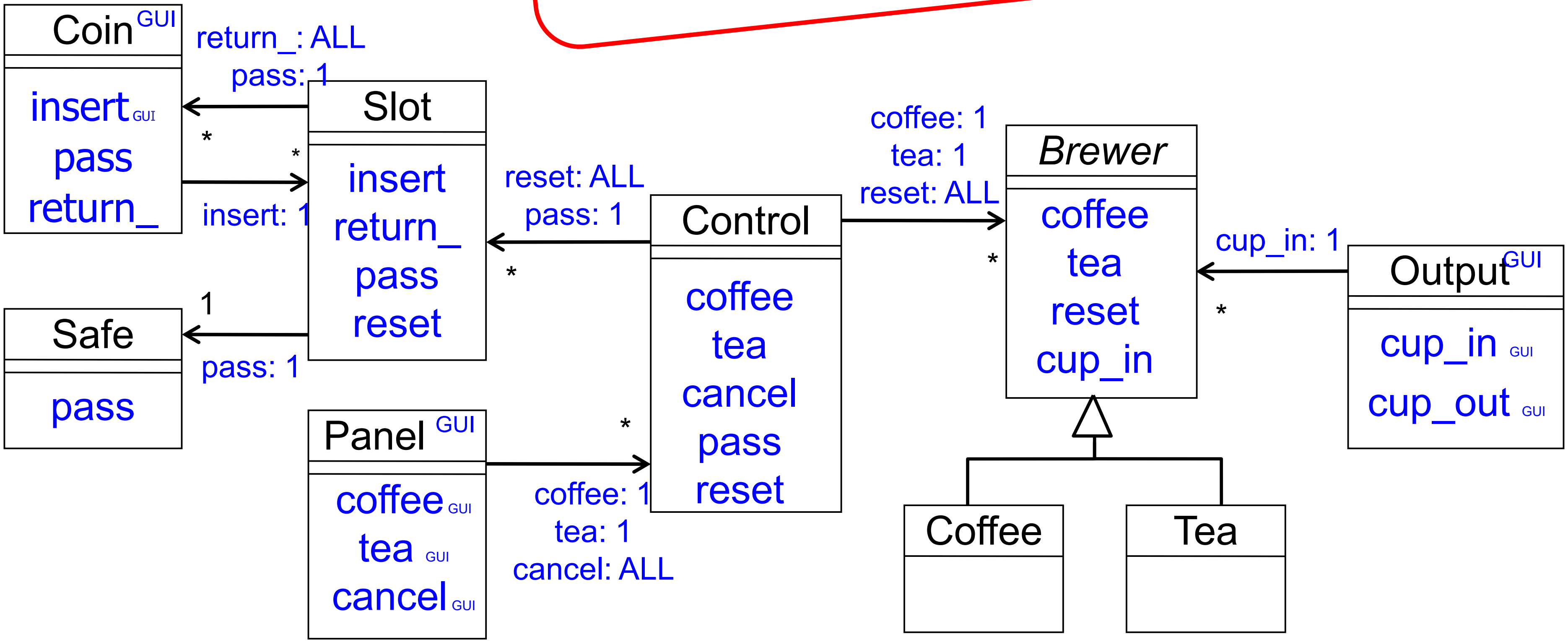


Object diagram as usual

Coordination Diagram

- We call objects elements now!

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



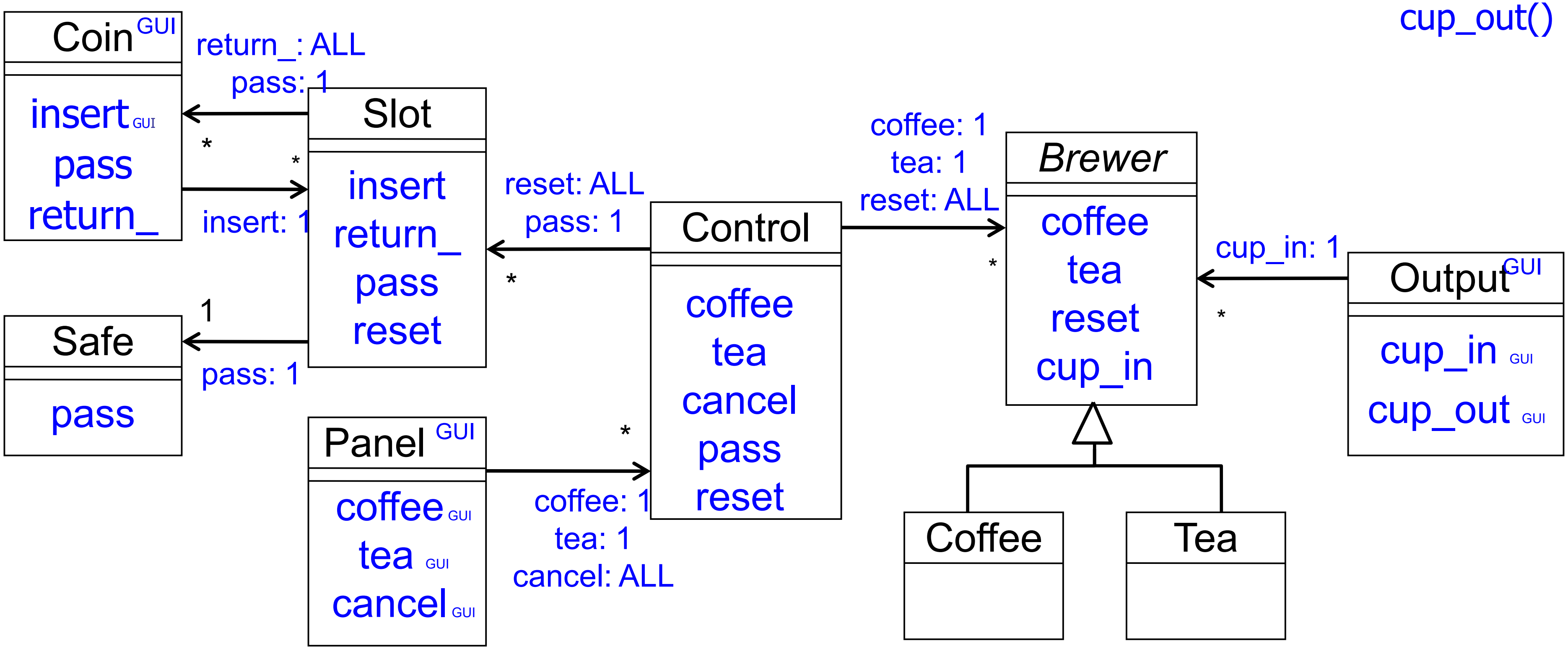
... + Event declaration

- Event (type) declaration
  - Parameters

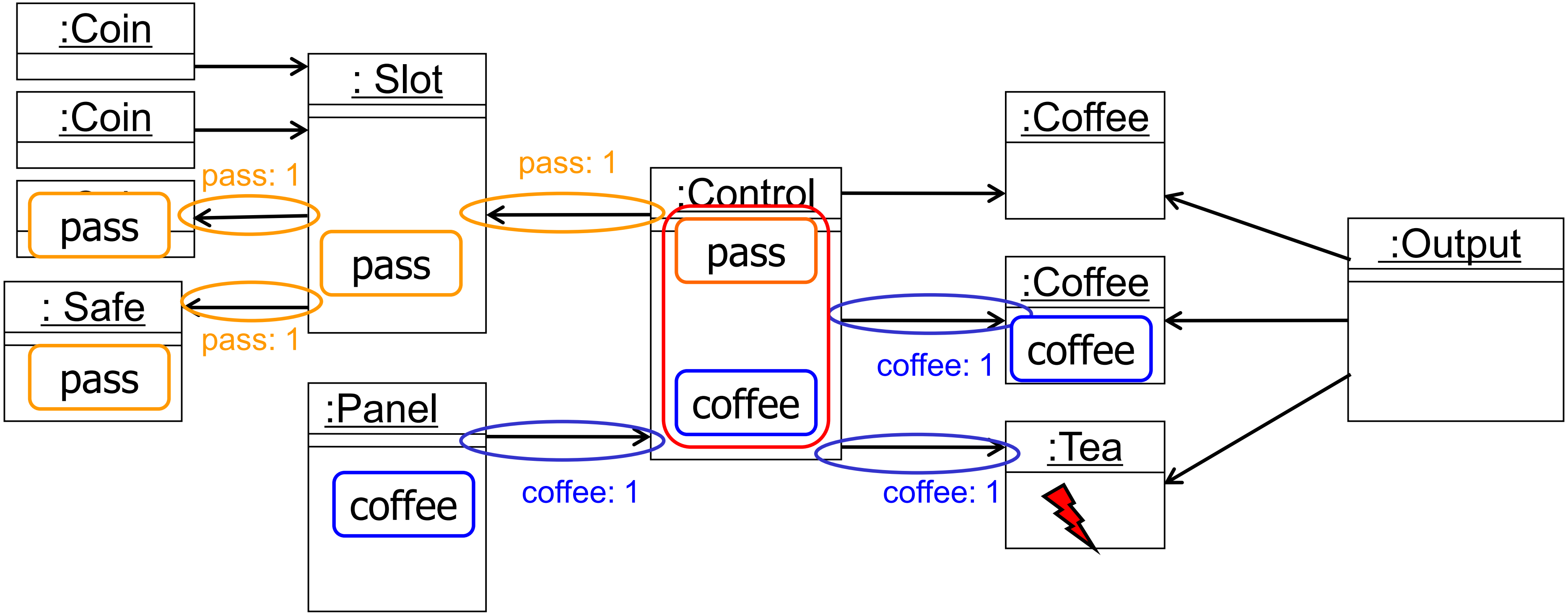
insert(Coin coin, Slot slot)
pass(Coin coin, Slot slot)
return(Slot slot)
reset_()

coffee()
tea()
cancel()

cup_in()
cup_out()

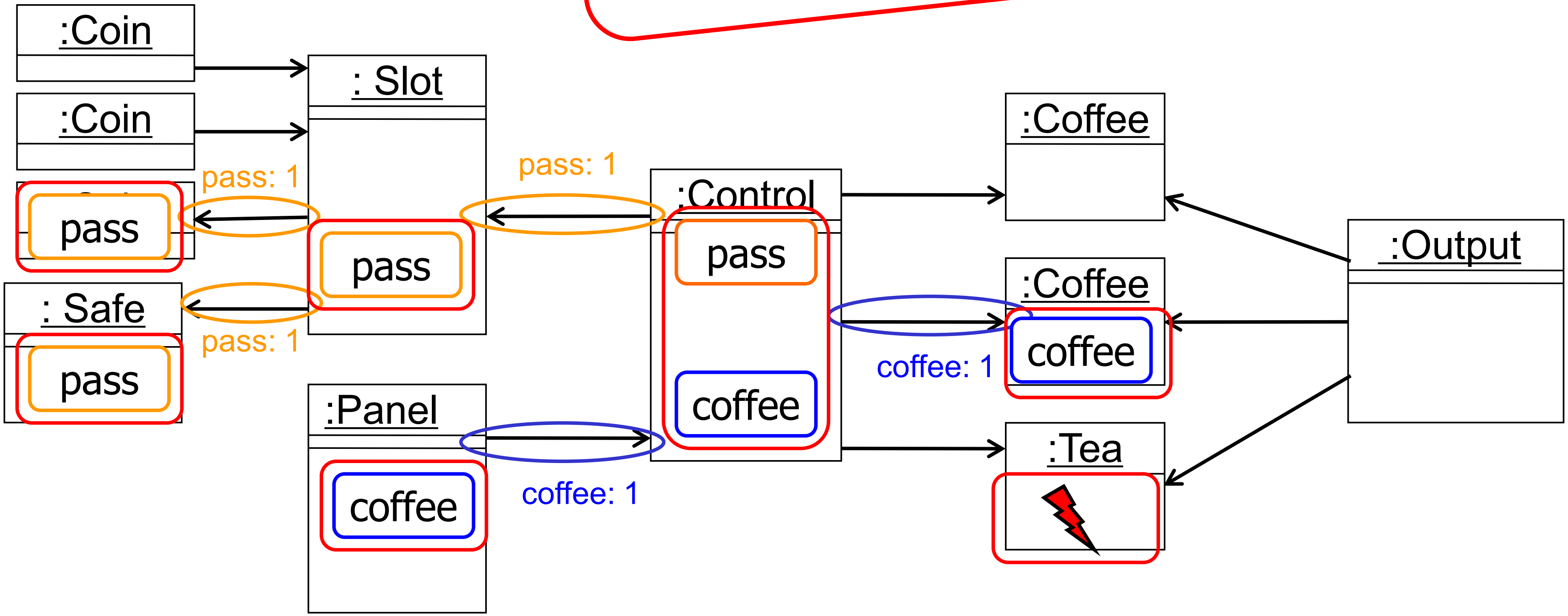


Interactions



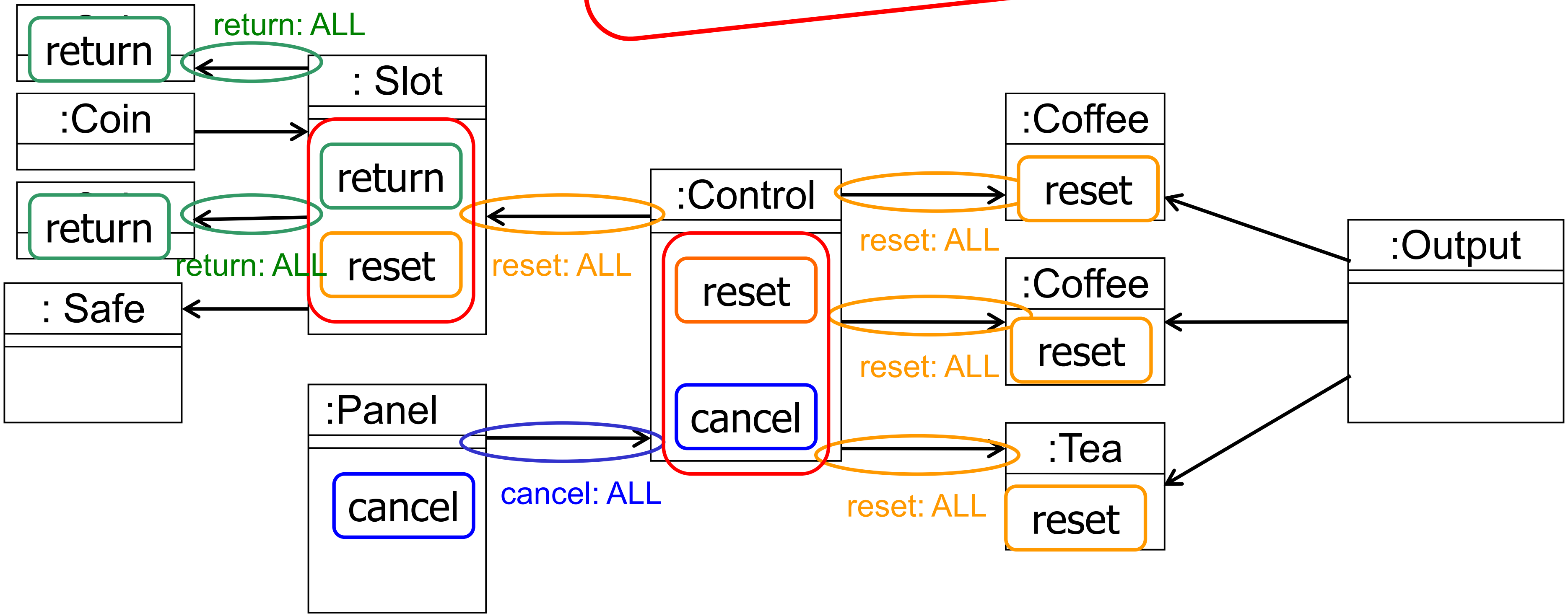
Interactions

Interaction =
local behavior +
coordination



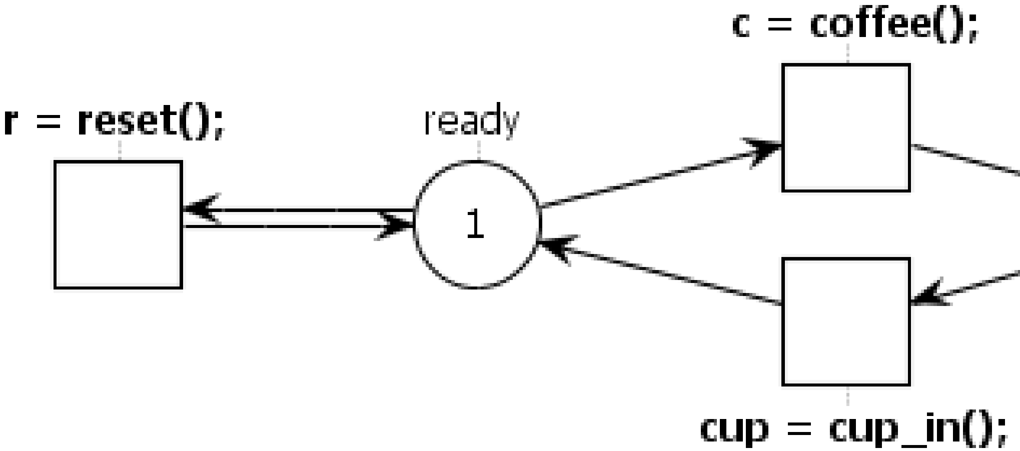
Another Interaction

Interaction =
local behavior +
coordination



Local behaviour: Coffee

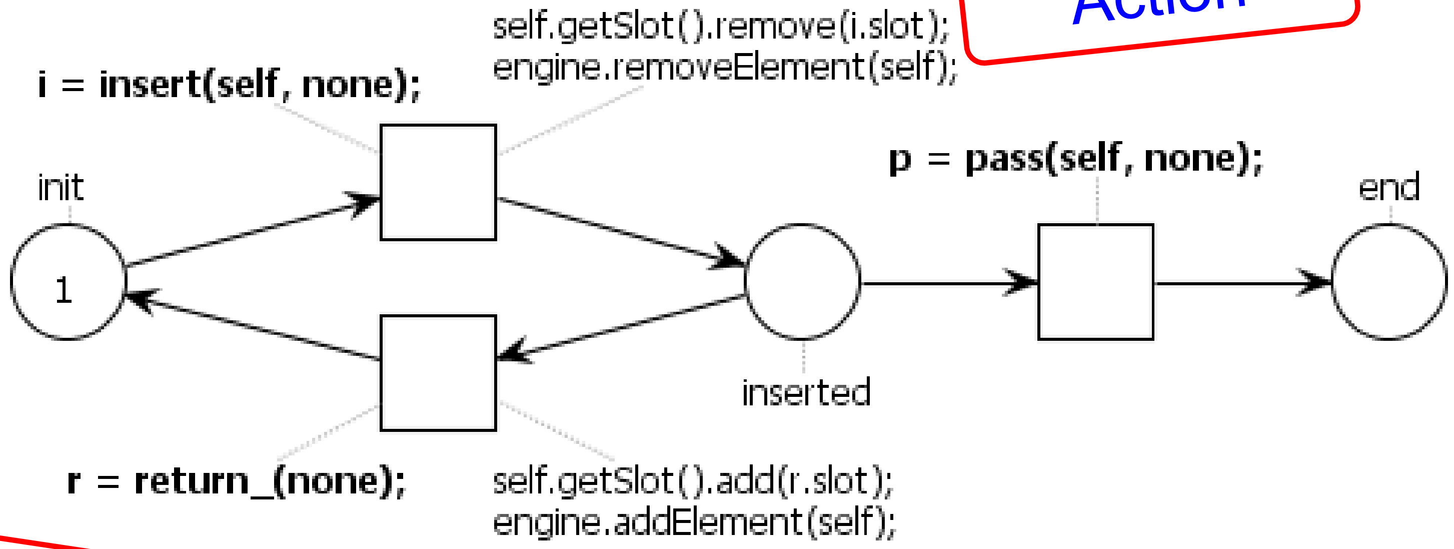
Event binding



Local behaviour: Coin

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

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



```
i = insert(self, none);
```

```
self.getSlot().remove(i.slot);
engine.removeElement(self);
```

Action

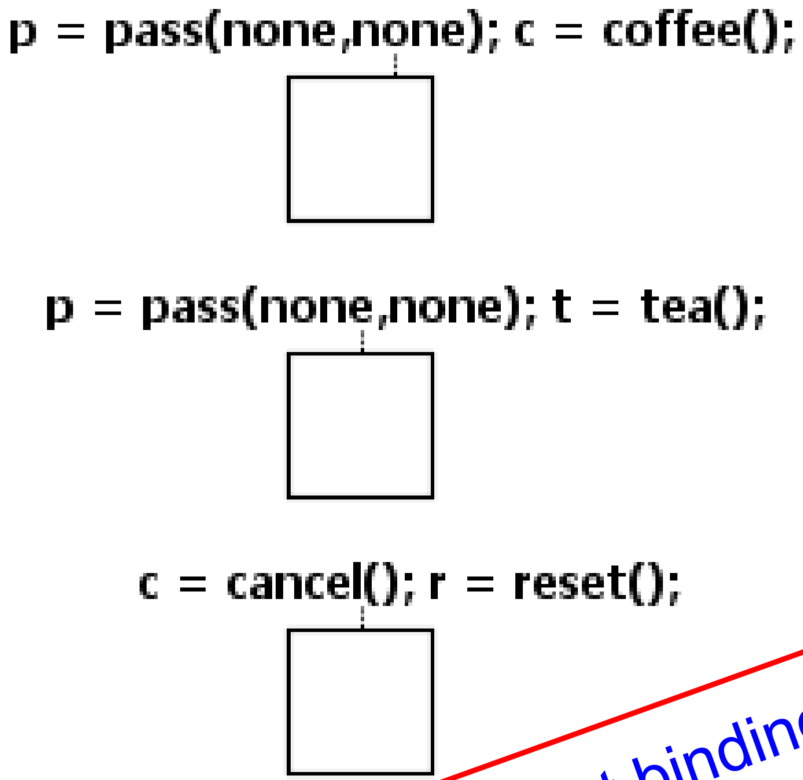
```
p = pass(self, none);
```

```
r = return_(none);
```

```
self.getSlot().add(r.slot);
engine.addElement(self);
```

- Event binding
- Parameter assignment

Local behaviour: Control



p = pass(none,none); c = coffee();

p = pass(none,none); t = tea();

c = cancel(); r = reset();

pass

coffee

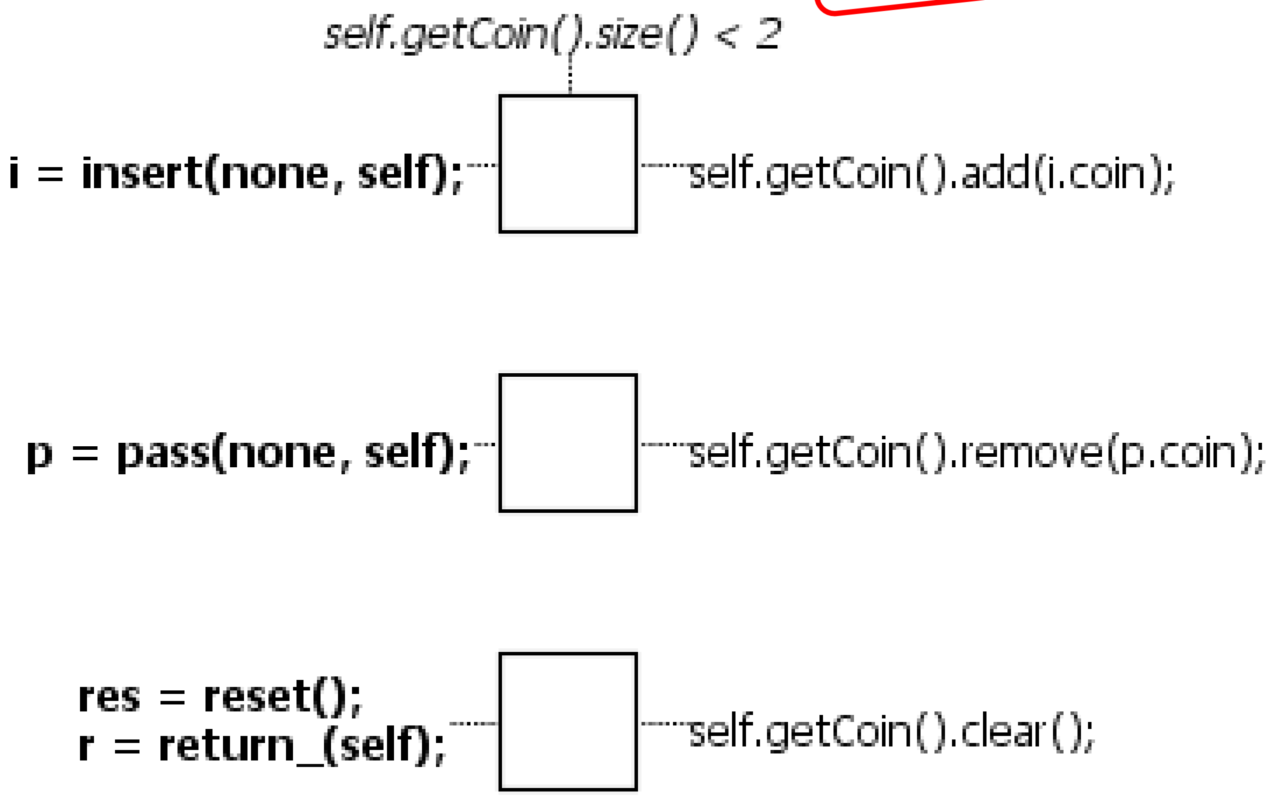
reset

cancel

- Event binding with multiple event types!

Local behaviour: Slot

Condition



self.getCoin().size() < 2

i = insert(none, self); self.getCoin().add(i.coin);

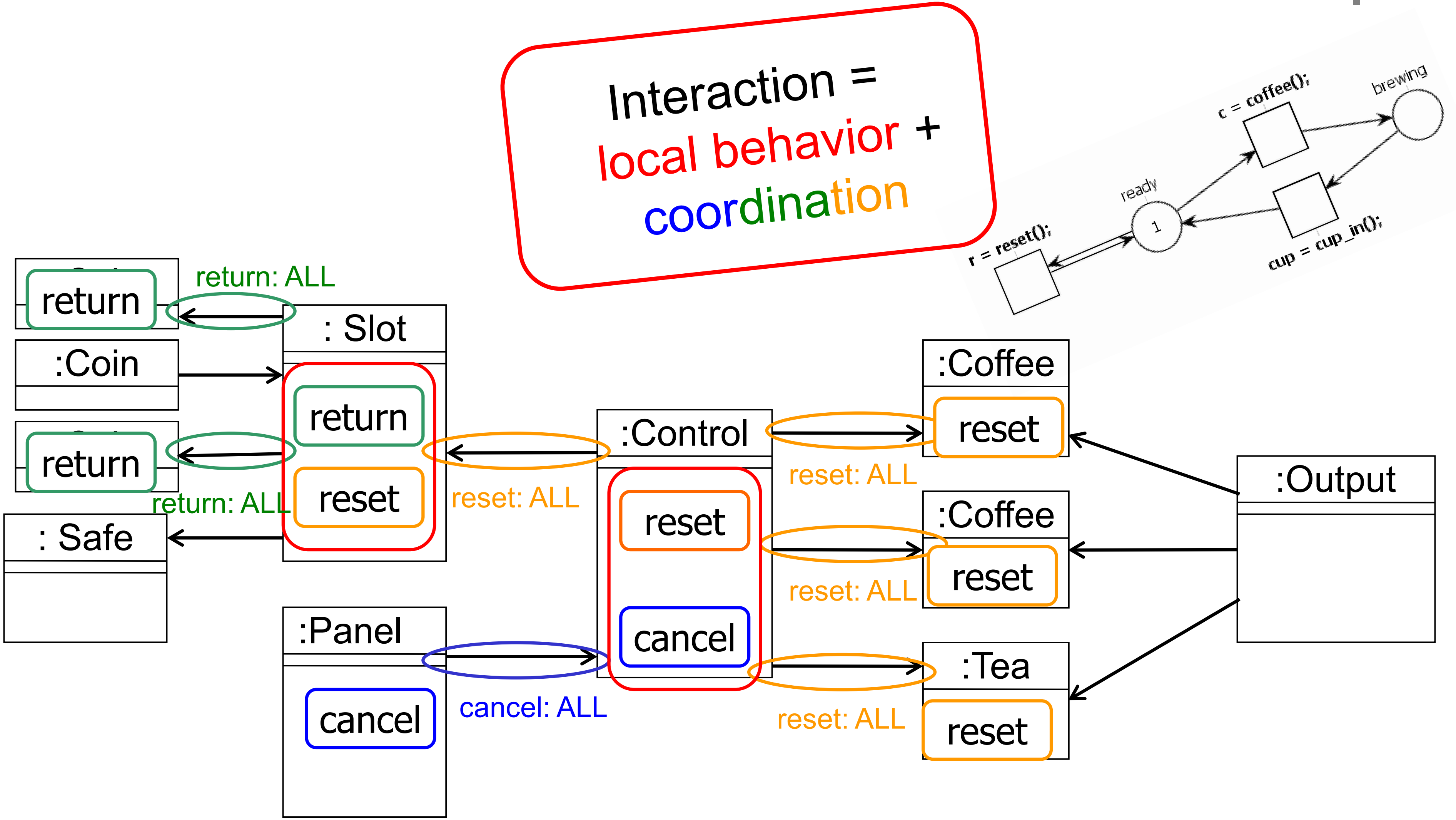
p = pass(none, self); self.getCoin().remove(p.coin);

res = reset();
r = return_(self); self.getCoin().clear();

return

reset

Interactions



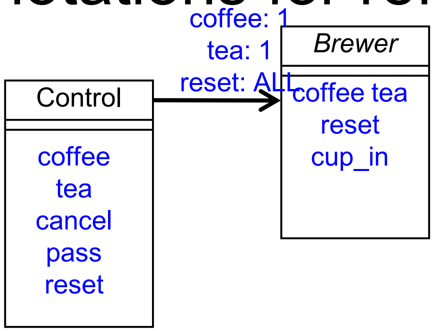
Interaction =
local behavior +
coordination

2.2 ECNO: Basic Concepts

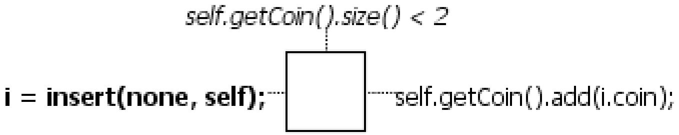
- ElementTypes (Classes)
- EventTypes with
  - parameters

insert(Coin coin, Slot slot)

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



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



self.getCoin().size() < 2

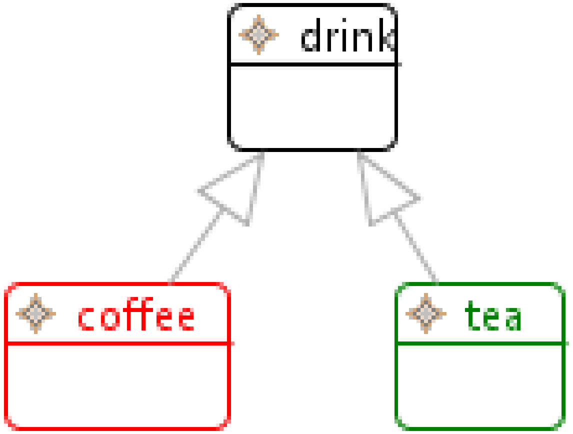
i = insert(none, self);

self.getCoin().add(i.coin);

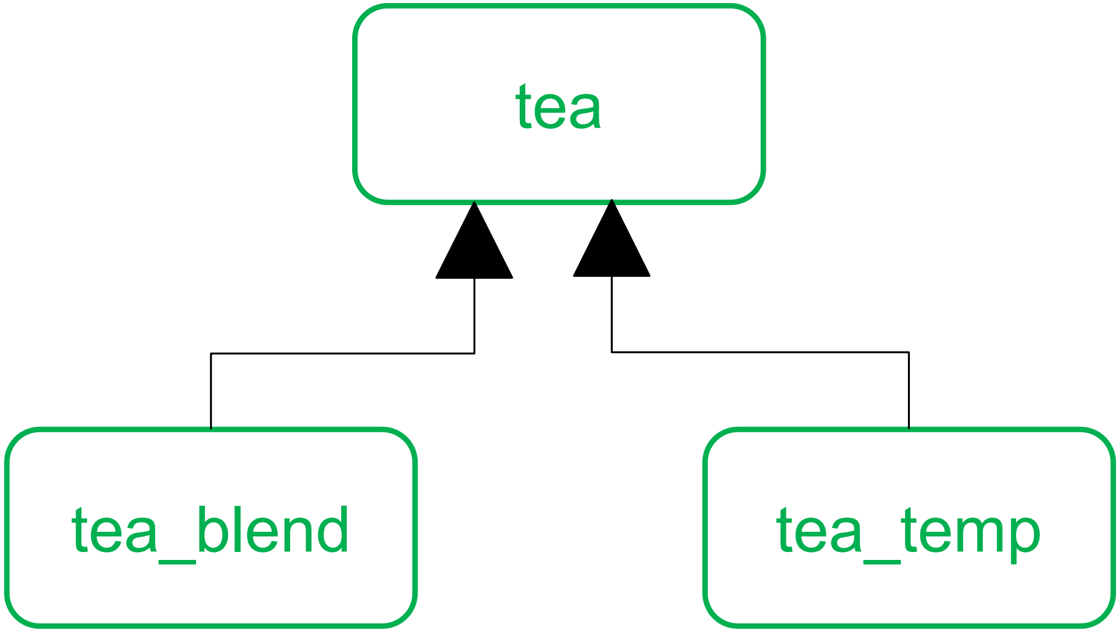
2.3 Extensions

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



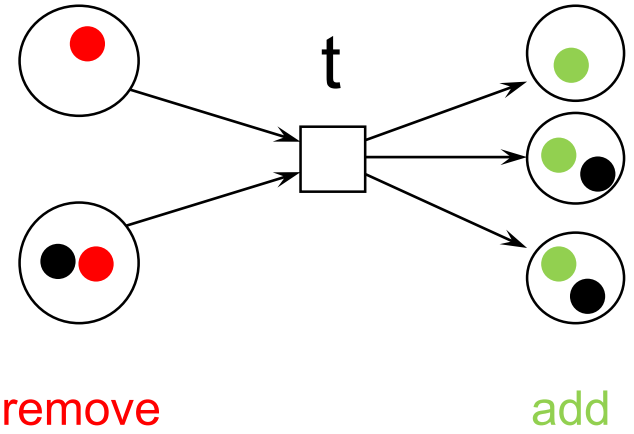
Event Extension



→ Two forms of inheritance on event types:

- specialization (previous slide)
- extension

2.4 Example 2: Petri nets



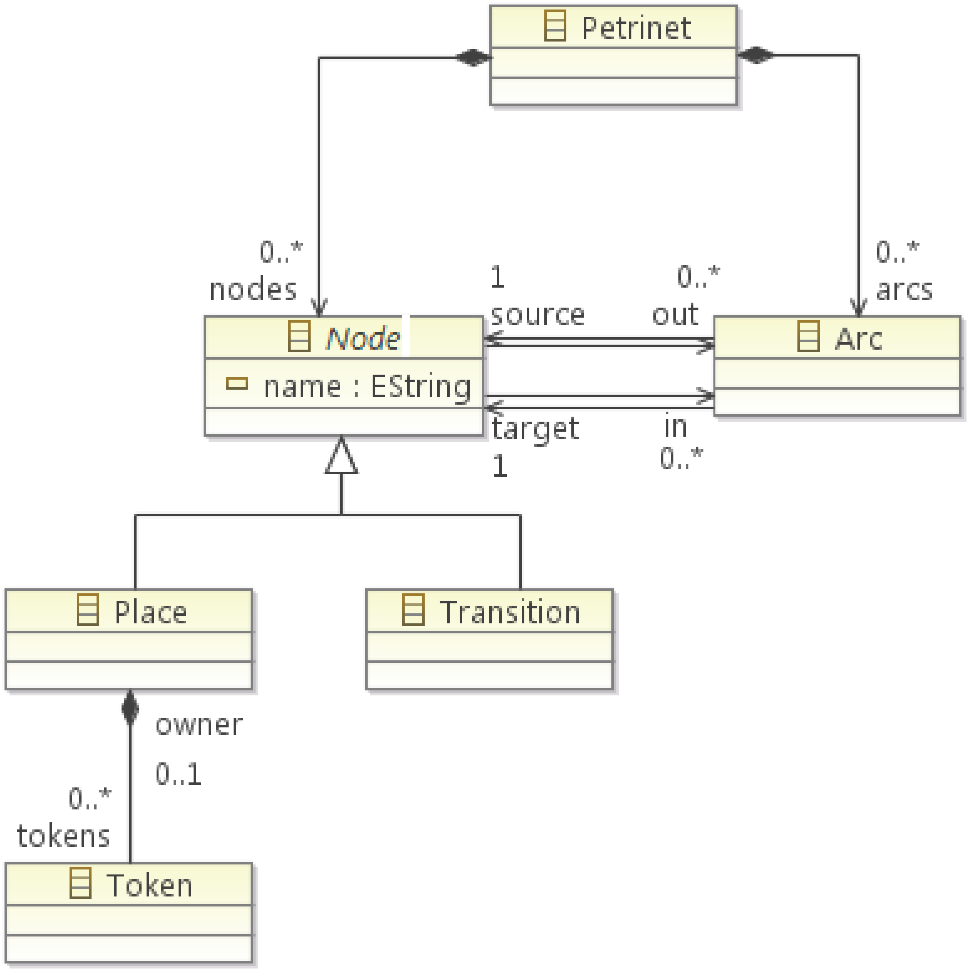
How can we model that behaviour in ECNO nets?

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

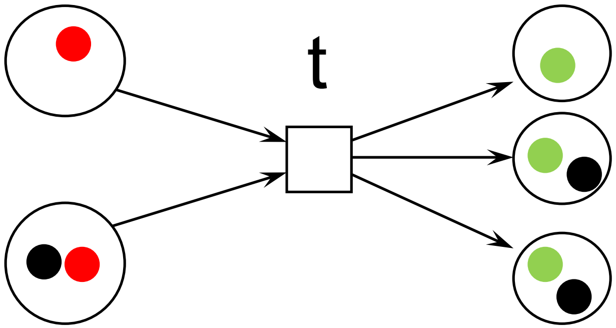
```
Fire Transition t:
  for ALL incoming Arcs a:
    for ONE source Place p of Arc a:
      find a token and remove it

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

Petri net: Abstract Syntax



Example2: Petri nets



remove

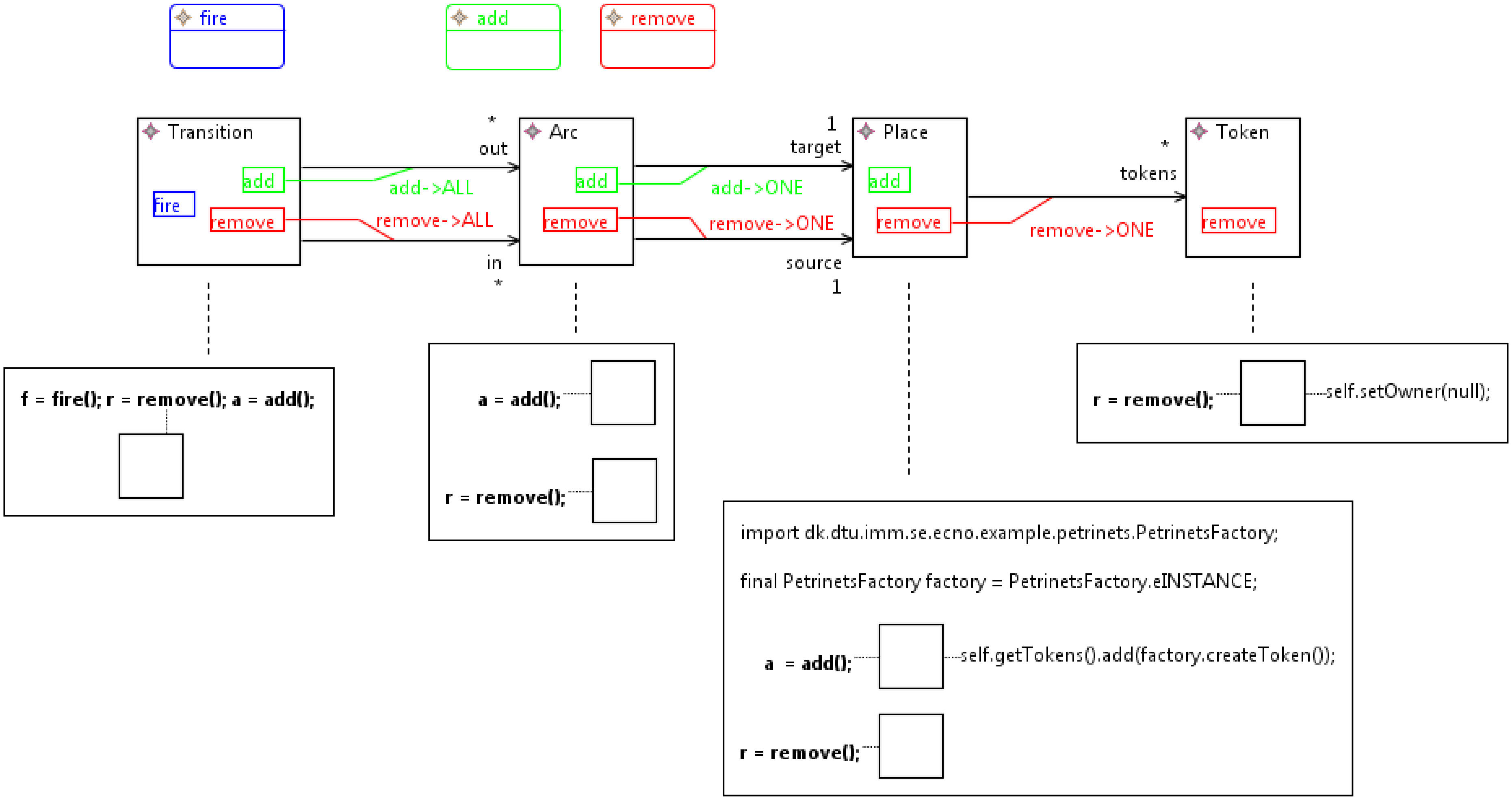
add

```
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



f = fire(); r = remove(); a = add();

a = add();

r = remove();

r = remove(); self.setOwner(null);

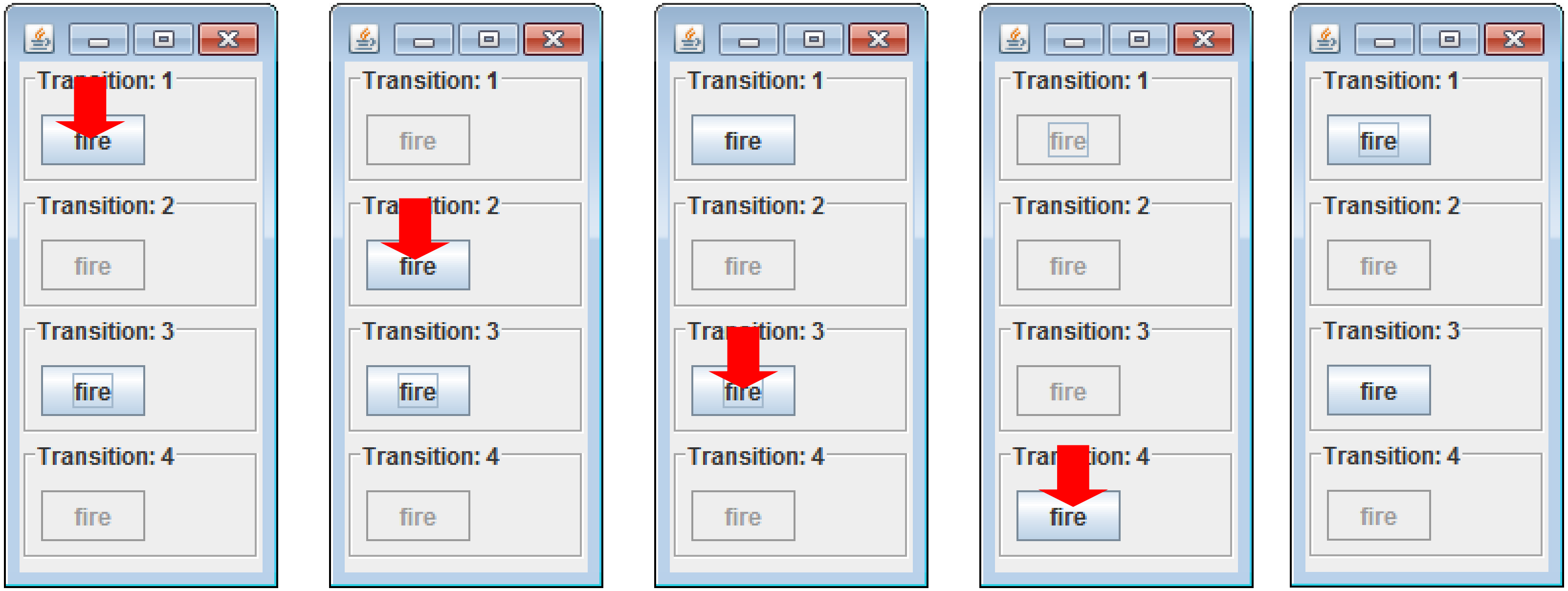
```
import dk.dtu.imm.se.ecno.example.petrinets.PetrinetsFactory;

final PetrinetsFactory factory = PetrinetsFactory.eINSTANCE;
```

a = add(); self.getTokens().add(factory.createToken());

r = remove();

Result



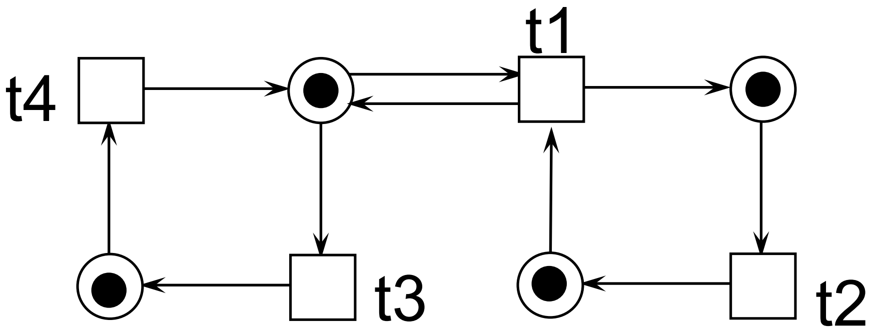
Transition: 1
fire
Transition: 2
fire
Transition: 3
fire
Transition: 4
fire

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

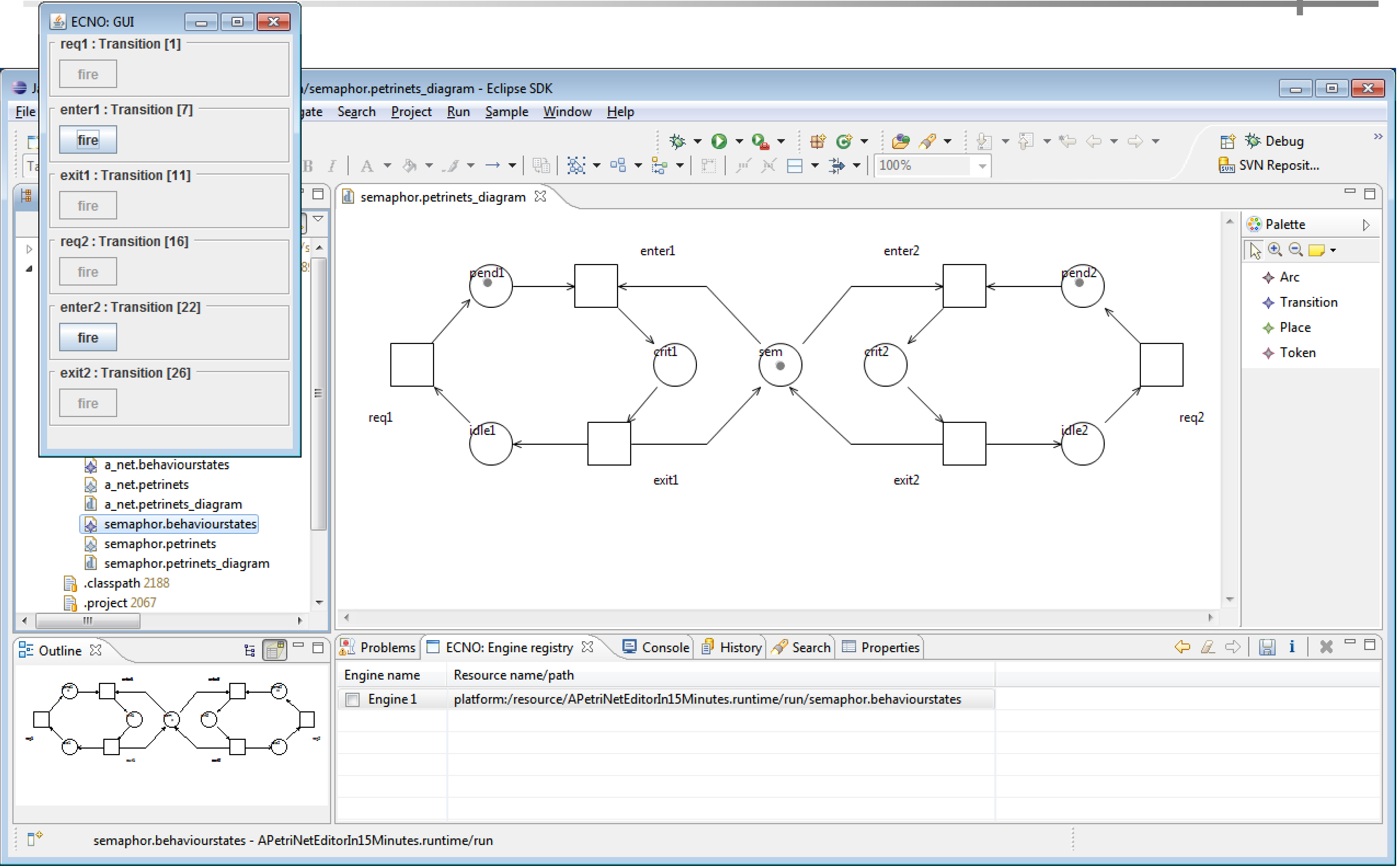
Transition: 1
fire
Transition: 2
fire
Transition: 3
fire
Transition: 4
fire

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

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



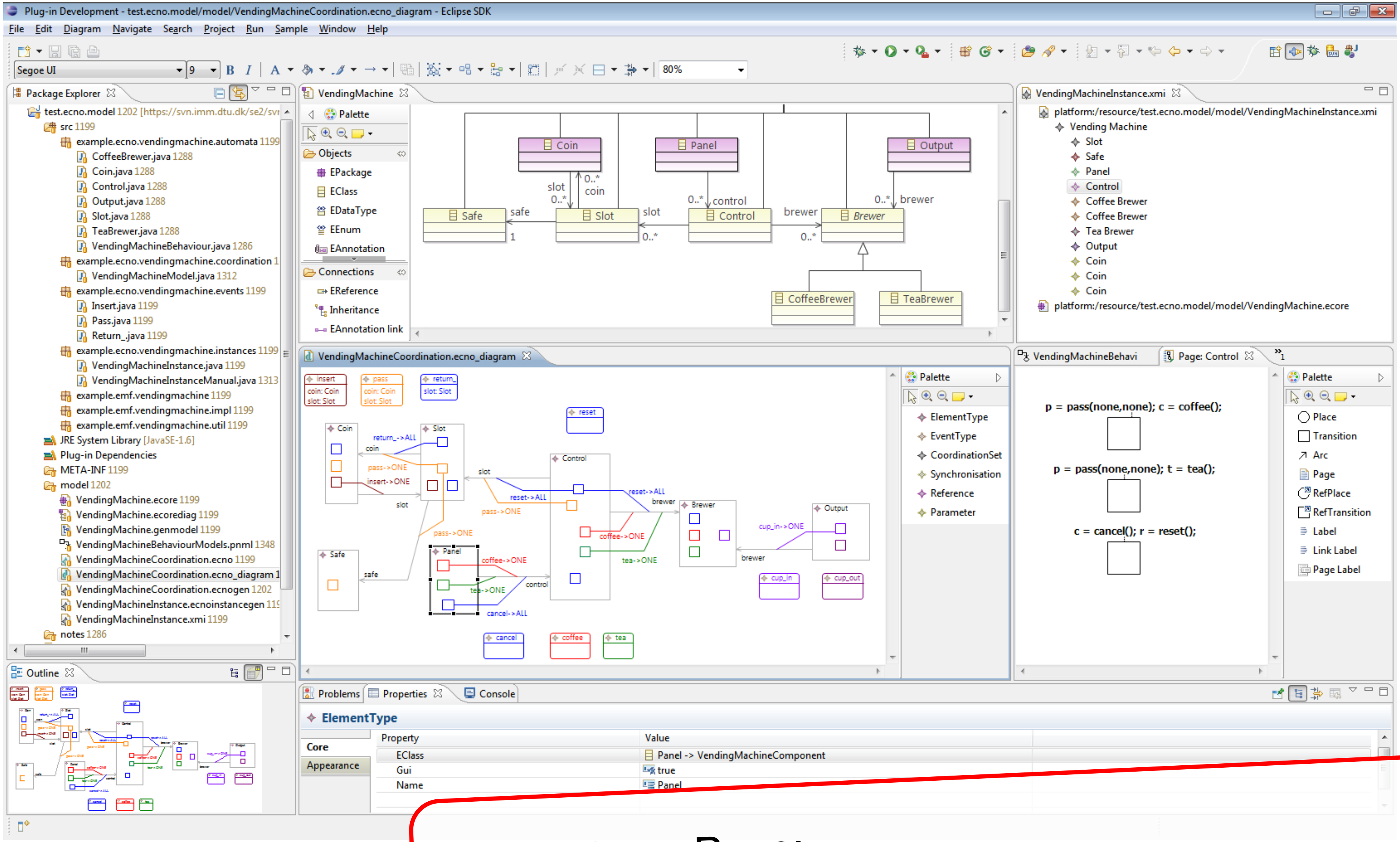
Petri net simulator



Engine name	Resource name/path
□ Engine 1	platform:/resource/APetriNetEditorIn15Minutes.runtime/run/semaphor.behaviourstates

semaphor.behaviourstates - APetriNetEditorIn15Minutes.runtime/run

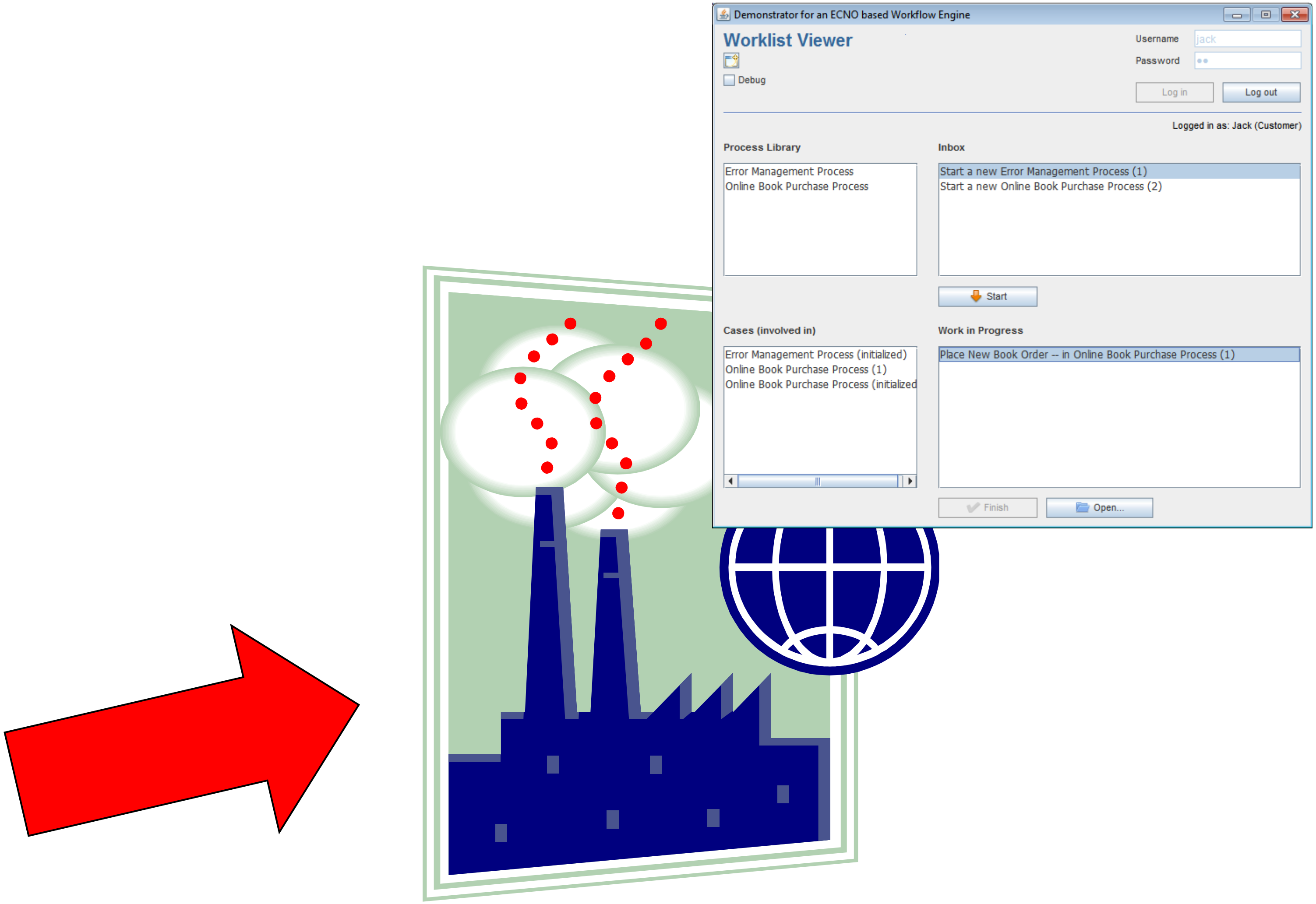
3. Conclusion


Plug-in Development - test.ecno.model/model/VendingMachineCoordination.ecno_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project Run Sample Window Help
Segoe UI
9
80%
Package Explorer
test.ecno.model 1202 [https://svn.imm.dtu.dk/se2/svn
src 1199
example.ecno.vendingmachine.automata 1199
CoffeeBrewer.java 1288
Coin.java 1288
Control.java 1288
Output.java 1288
Slot.java 1288
TeaBrewer.java 1288
VendingMachineBehaviour.java 1286
example.ecno.vendingmachine.coordination 1
VendingMachineModel.java 1312
example.ecno.vendingmachine.events 1199
Insert.java 1199
Pass.java 1199
Return_.java 1199
example.ecno.vendingmachine.instances 1199
VendingMachineInstance.java 1199
VendingMachineInstanceManual.java 1313
example.emf.vendingmachine 1199
example.emf.vendingmachine.impl 1199
example.emf.vendingmachine.util 1199
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
META-INF 1199
model 1202
VendingMachine.ecore 1199
VendingMachine.ecorediag 1199
VendingMachine.genmodel 1199
VendingMachineBehaviourModels.pnml 1348
VendingMachineCoordination.ecno 1199
VendingMachineCoordination.ecno_diagram 1
VendingMachineCoordination.ecnogen 1202
VendingMachineInstance.ecnoinstancegen 11
VendingMachineInstance.xmi 1199
notes 1286
Outline
VendingMachine
Palette
Objects
EPackage
EClass
EDataType
EEnum
EAnnotation
Connections
EReference
Inheritance
EAnnotation link
Coin
Panel
Output
Safe
Slot
Control
Brewer
CoffeeBrewer
TeaBrewer
slot 0..*
0..* coin
safe 1
slot 0..*
0..* control
brewer 0..*
0..* brewer
VendingMachineInstance.xmi
platform:/resource/test.ecno.model/model/VendingMachineInstance.xmi
Vending Machine
Slot
Safe
Panel
Control
Coffee Brewer
Coffee Brewer
Tea Brewer
Output
Coin
Coin
Coin
platform:/resource/test.ecno.model/model/VendingMachine.ecore
VendingMachineCoordination.ecno_diagram
insert coin: Coin slot: Slot
pass coin: Coin slot: Slot
return_ slot: Slot
reset
Coin
Slot
return_->ALL
coin
pass->ONE
insert->ONE
slot
Control
reset->ALL
pass->ONE
reset->ALL
brewer
Brewer
Output
cup_in->ONE
Safe
safe
Panel
coffee->ONE
coffee->ONE
tea->ONE
tea->ONE
control
cancel->ALL
brewer
cup_in
cup_out
cancel
coffee
tea
Palette
ElementType
EventType
CoordinationSet
Synchronisation
Reference
Parameter
VendingMachineBehavi
Page: Control
p = pass(none,none); c = coffee();
p = pass(none,none); t = tea();
c = cancel(); r = reset();
Palette
Place
Transition
Arc
Page
RefPlace
RefTransition
Label
Link Label
Page Label
Problems
Properties
Console
ElementType
Core
Appearance

Property	Value
EClass	Panel -> VendingMachineComponent
Gui	true
Name	Panel

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

Master Project: WfMS



Demonstrator for an ECNO based Workflow Engine

Worklist Viewer

☐ Debug

Username: jack

Password: ••

Logged in as: Jack (Customer)

Process Library

- Error Management Process
- Online Book Purchase Process

Inbox

- Start a new Error Management Process (1)
- Start a new Online Book Purchase Process (2)

Cases (involved in)

- Error Management Process (initialized)
- Online Book Purchase Process (1)
- Online Book Purchase Process (initialized

Work in Progress

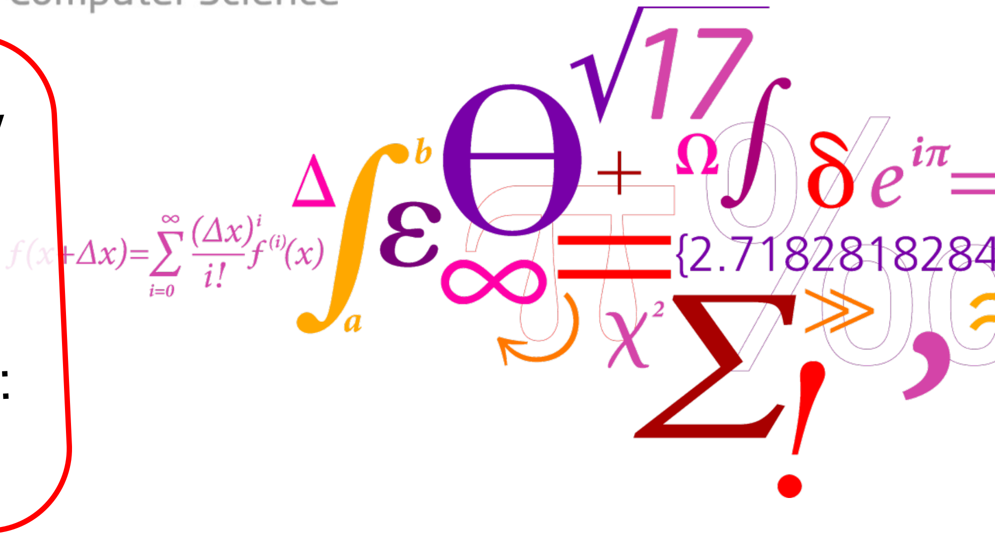
- Place New Book Order -- in Online Book Purchase Process (1)

MSc and BSc Projects

DTU Compute
Department of Applied Mathematics and Computer Science

MSc and BSc projects or not only "programming something"!

There needs to be a conceptual/scientific contribution: Design Science!



Topics / areas

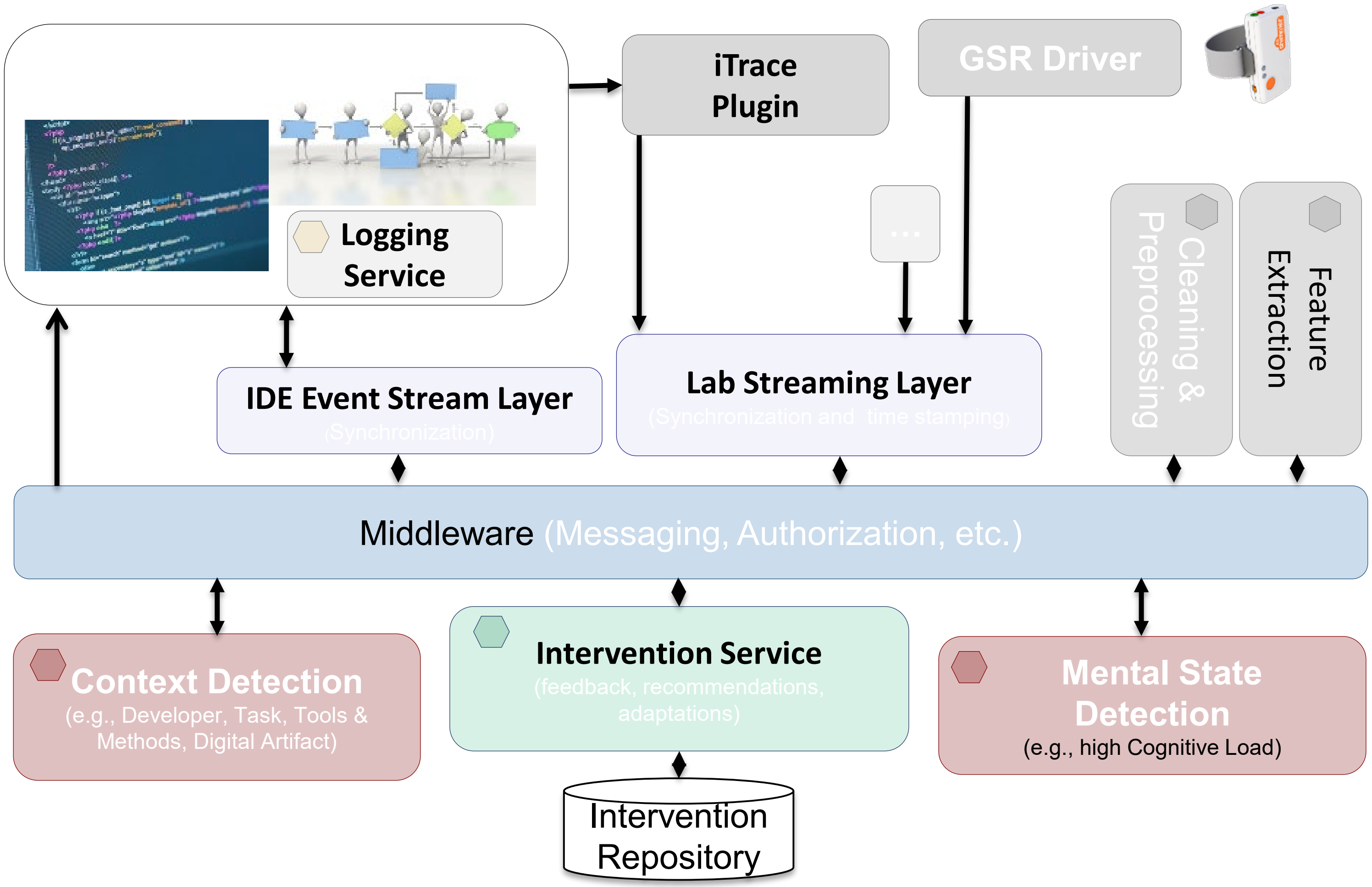
- CITIES, skoleklima/climify
  - Solidify skoleklima/climify
  - Create a secure version, which might run in production
  - Generic backend for collecting data from IoT devices

There are also payed student jobs in Climify++

- Flexible, configurable and generic data collection from all kinds of devices (e.g. Fitbit)
  - health care
  - energy
  - →EmpiRes
  - →CITIES

Topics / areas

- Empirical Research Platform: EmpiRes



iTrace Plugin

GSR Driver

Logging Service

...

Cleaning & Preprocessing

Feature Extraction

IDE Event Stream Layer (Synchronization)

Lab Streaming Layer (Synchronization and time stamping)

Middleware (Messaging, Authorization, etc.)

Context Detection (e.g., Developer, Task, Tools & Methods, Digital Artifact)

Intervention Service (feedback, recommendations, adaptations)

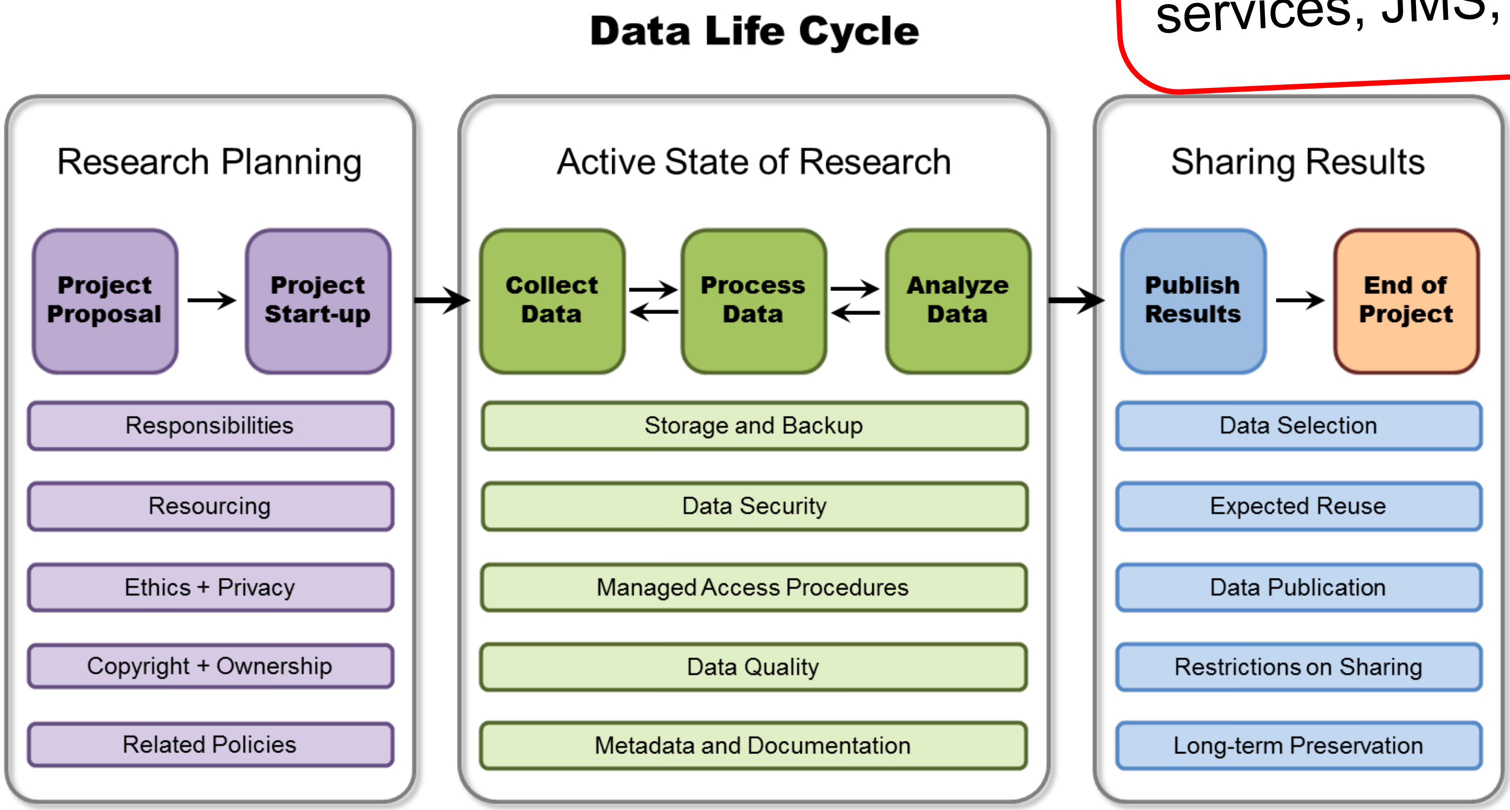
Mental State Detection (e.g., high Cognitive Load)

Intervention Repository

Topics / areas

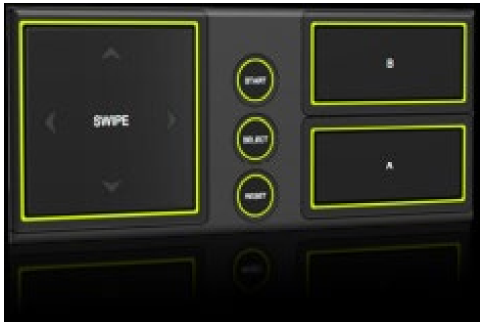
- Empirical Research Platform (cntd.)

In this project: plethora of technologies (cloud, services and micro services, JMS, IoT, …)



Topic / areas

- TacTile Pinpoint®: A software that imitates the feel of a physical control. It allows touch screens to be operated without watching where the user's fingers are going and give a 3D user experience.



CAMAR

- Topic 1: Porting SW from Android platform to C++
- Topic 2: Demonstrators of technology for gaming

There are also payed student jobs related to these projects

There might also be payed student jobs in LiRA

Live Road Assessment (LiRA) based on modern cars' sensors

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science

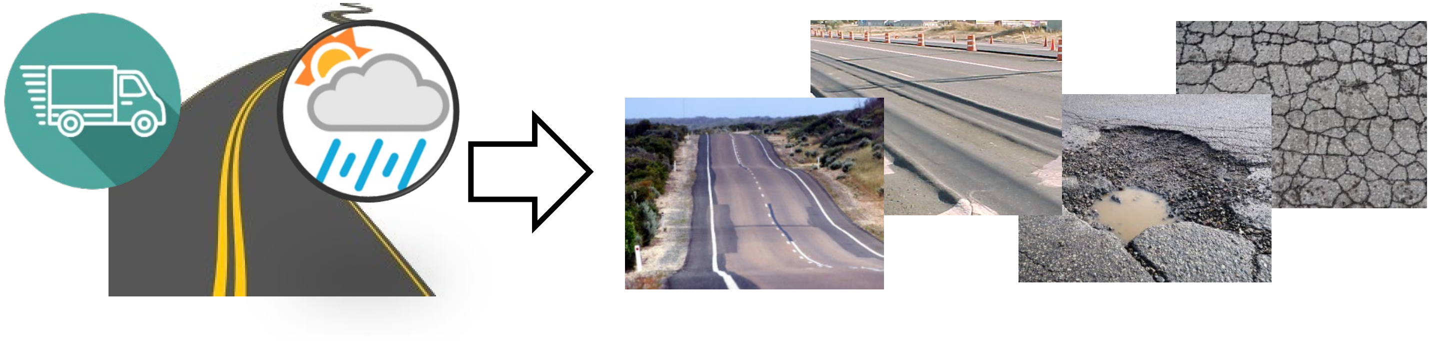
Innovation Fund Project (Grand Solutions)
Danish Road Authority (Vejdirektoratet, DRD)
DTU Byg
DTU Compute (SPE & CogSys)
Green Mobility
SWECO

Duration: 3½ years
Budget: 18 mio DKK (2.4 mio EUR)
IF funding: 12 mio DKK (1.6 mio EUR)

Some slides "borrowed" from
Matteo Pettinari (Projektleader, DRD)

Background

Roads make a crucial contribution to economic development and growth and bring **important** social benefits.



Background

Standard road measures have been developed to guarantee proper road conditions and to optimize maintenance strategies focusing on (DRD operational costs 5 million DKK per year – do not include Env. Emissions) :

- Safety
- Comfort
- Durability
- Environmental emissions (noise and CO_2)

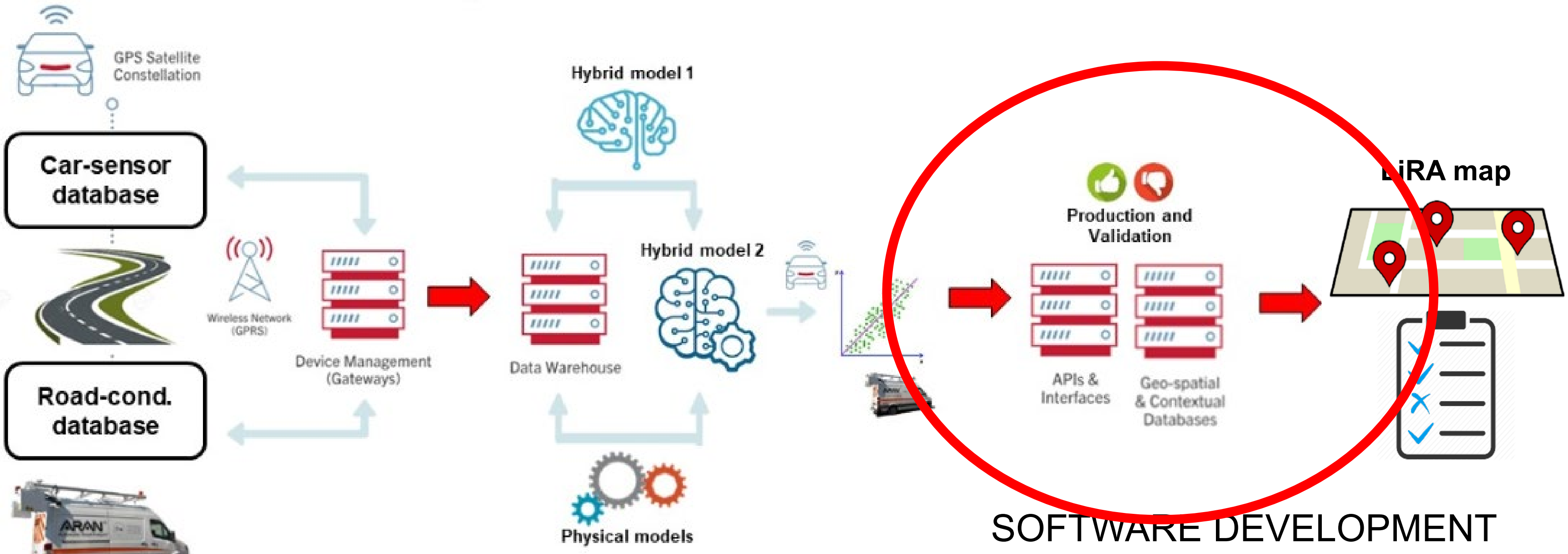
Project idea

Can we find a more efficient and faster way to **monitor**, maintain and manage the roads?

Modern cars are equipped with many sensors and can also provide further data including energy consumption.

Can car sensors data be used to measure road conditions?

Overview



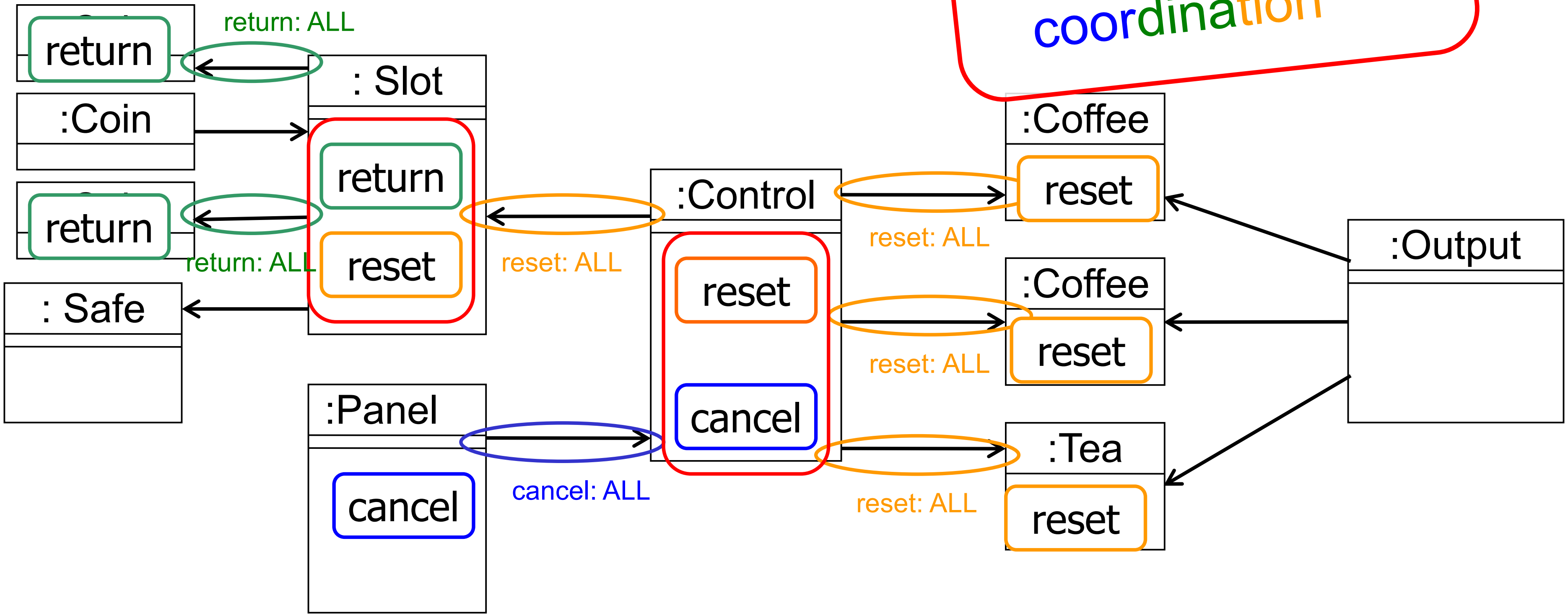
SOFTWARE DEVELOPMENT

MODELING

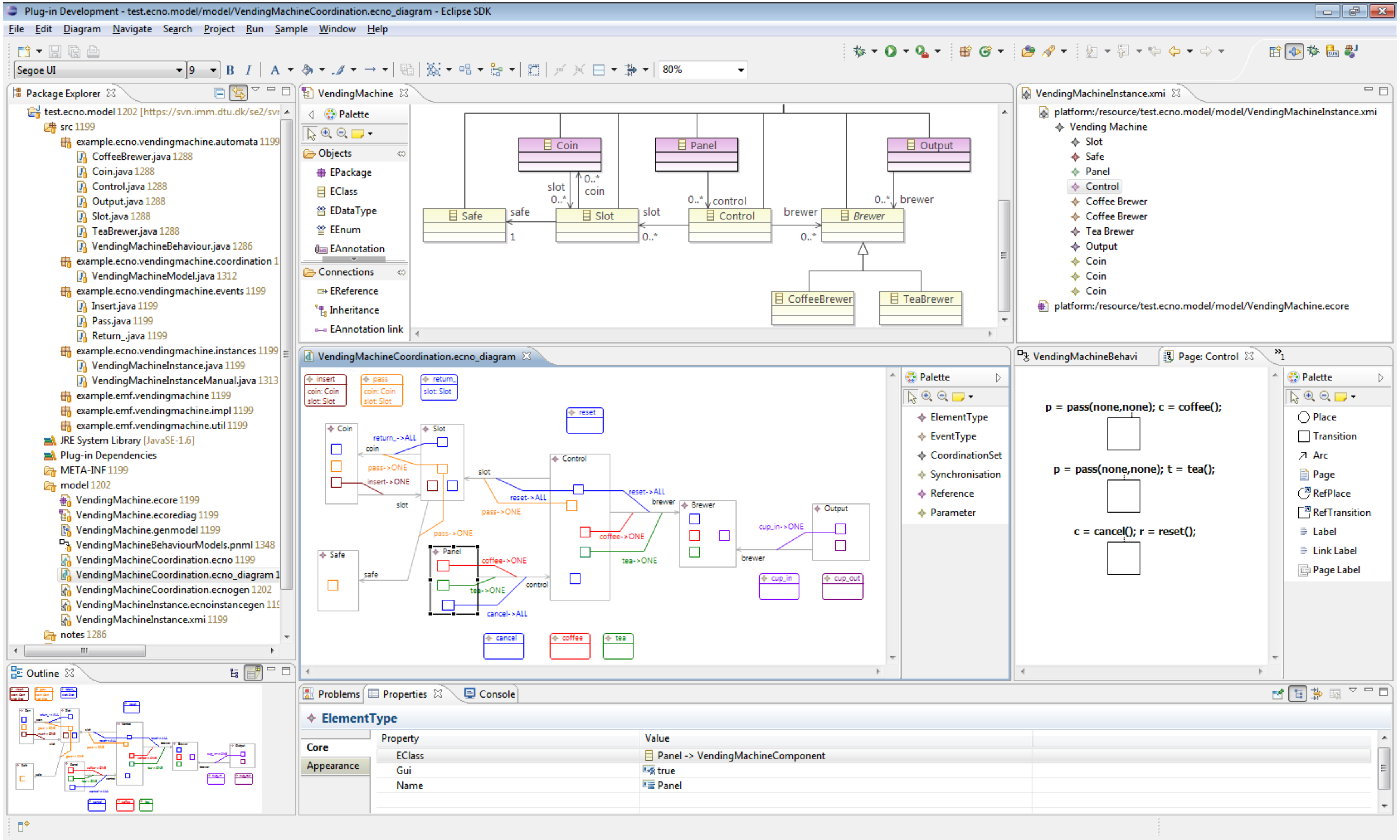
ECNO

- Behaviour modelling: ECNO

Interaction =
local behavior
coordination



ECNO Tool



Plug-in Development - test.ecno.model/model/VendingMachineCoordination.ecno_diagram - Eclipse SDK

Package Explorer

- test.ecno.model 1202 [https://svn.imm.dtu.dk/se2/svn
  - src 1199
    - example.ecno.vendingmachine.automata 1199
      - CoffeeBrewer.java 1288
      - Coin.java 1288
      - Control.java 1288
      - Output.java 1288
      - Slot.java 1288
      - TeaBrewer.java 1288
      - VendingMachineBehaviour.java 1286
    - example.ecno.vendingmachine.coordination 1
      - VendingMachineModel.java 1312
    - example.ecno.vendingmachine.events 1199
      - Insert.java 1199
      - Pass.java 1199
      - Return_.java 1199
    - example.ecno.vendingmachine.instances 1199
      - VendingMachineInstance.java 1199
      - VendingMachineInstanceManual.java 1313
    - example.emf.vendingmachine 1199
    - example.emf.vendingmachine.impl 1199
    - example.emf.vendingmachine.util 1199
  - JRE System Library [JavaSE-1.6]
  - Plug-in Dependencies
  - META-INF 1199
  - model 1202
    - VendingMachine.ecore 1199
    - VendingMachine.ecorediag 1199
    - VendingMachine.genmodel 1199
    - VendingMachineBehaviourModels.pnml 1348
    - VendingMachineCoordination.ecno 1199
    - VendingMachineCoordination.ecno_diagram 1
    - VendingMachineCoordination.ecnogen 1202
    - VendingMachineInstance.ecnoinstancegen 11
    - VendingMachineInstance.xmi 1199
  - notes 1286

VendingMachineInstance.xmi

- platform:/resource/test.ecno.model/VendingMachineInstance.xmi
  - Vending Machine
    - Slot
    - Safe
    - Panel
    - Control
    - Coffee Brewer
    - Coffee Brewer
    - Tea Brewer
    - Output
    - Coin
    - Coin
    - Coin
- platform:/resource/test.ecno.model/VendingMachine.ecore

Page: Control

p = pass(none,none); c = coffee();

p = pass(none,none); t = tea();

c = cancel(); r = reset();

Properties

ElementType

Property	Value
EClass	Panel -> VendingMachineComponent
Gui	true
Name	Panel

ECNO Projects

- Improved IDE Integration & debugging
- Smooth database integration (cntd.)
- Case studies / larger examples (e.g. games)
- Semantics of ECNO in ECNO
- …

Other topics

- Analysis and design of lighting in buildings

- Model-based Software Engineering
- Domain Specific Languages (DSL)
- Automating the SE process
- Tools support for development process

In a nutshell, everything that helps improving and speeding up the software development process!