BAIKEM *-Netzwerktreffen*

Embedded Systems

Modell-getriebene Entwicklung mit der YAKINDU-Workbench

itemis

Software | Consulting | Coaching

about me ...

Axel Terfloth
Head R&D Embedded Systems
axel.terfloth@itemis.de

itemis
Software | Consulting | Coaching

… work at itemis AG, Germany

… work on model driven development of embedded systems

… work on YAKINDU open source project

about itemis AG ...

founded 2003, ± 145 people

Training, Coaching, Consulting

- Model Driven Development (MDD)
- individual Tools and Toolchains
- Embedded Systems, Mobile Apps, Enterprise Systems

Open Source - Eclipse Strategic Member & Contributer

Eclipse Modeling

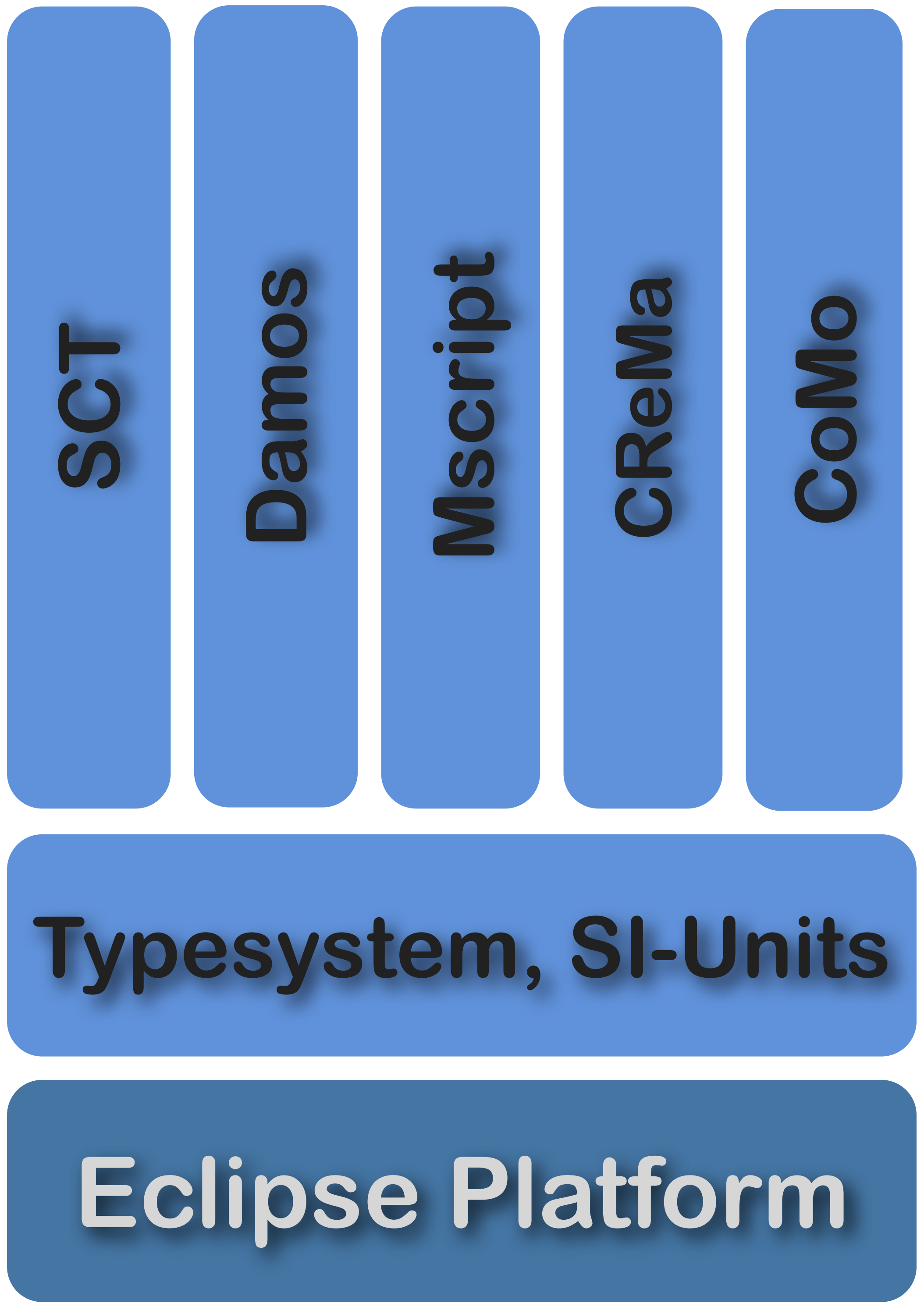
- EMF - Eclipse Modeling Framework
- Xtext - Textual Modeling Framework
- Xpand / Xtend - Code Generator Framework
- GEF - Grafical Editing Framework

YAKINDU

YAKINDU
is a ***modular toolkit***
for ***model driven development***
of embedded systems

Yakindu Language Modules

- SCT - statecharts
- Damos - data-flow oriented modeling
- Mscript - math oriented scripting
- CReMa - (requirements) traceability
- CoMo - component model (upcoming)



Yakindu Language Modules

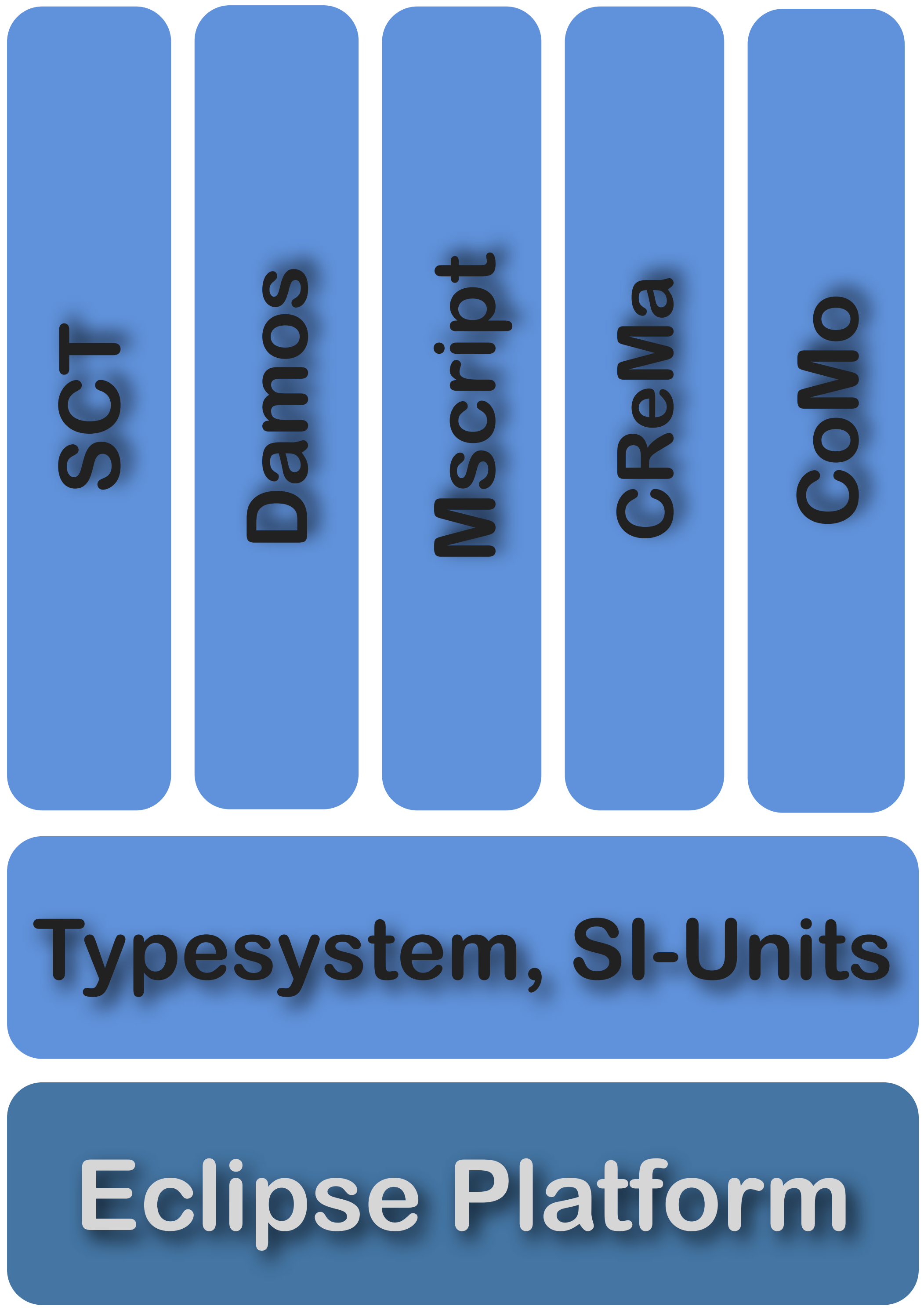
language modules are:

- independent
- not bound to a specific methodology
- self contained
- **can be used on their own**

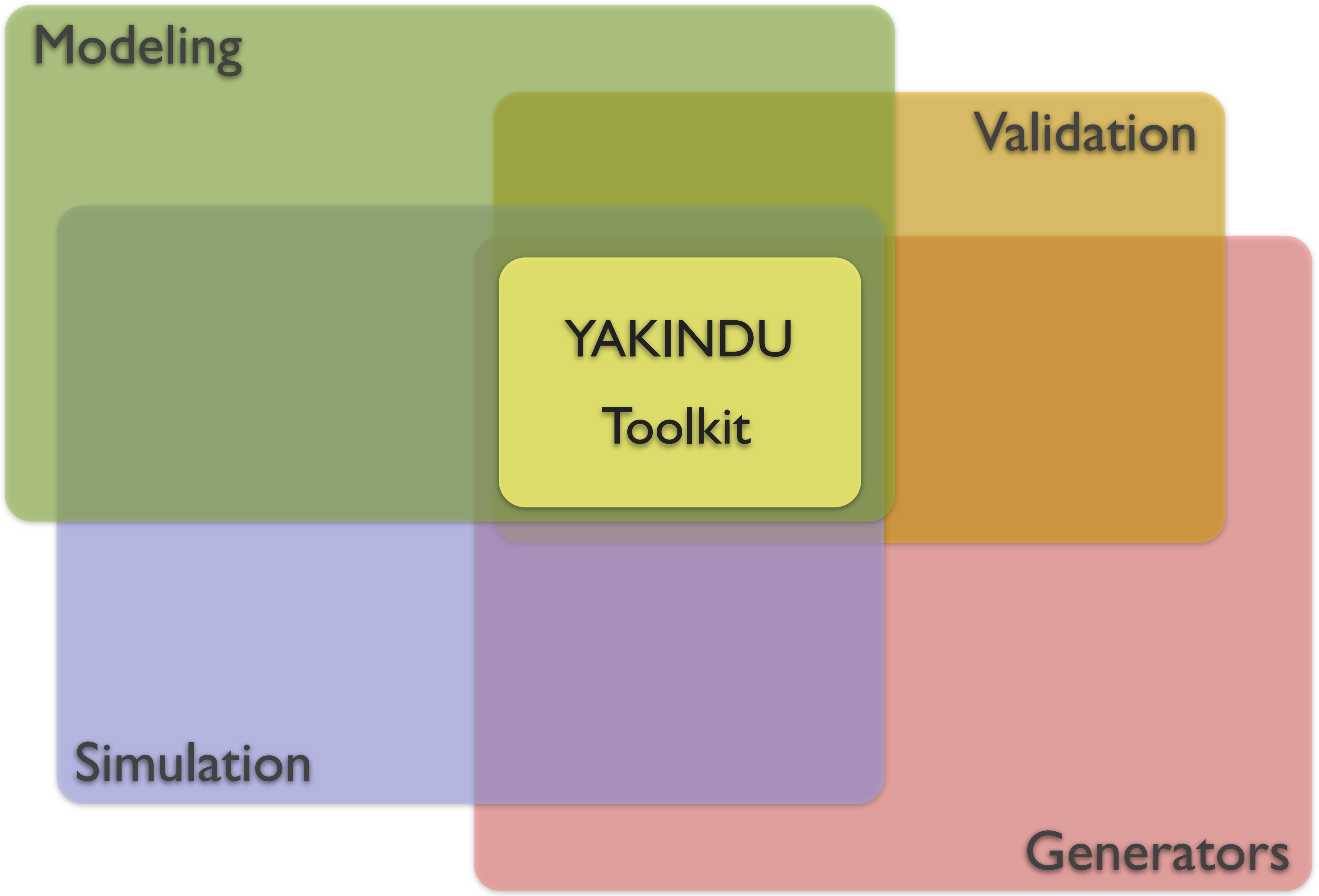
- open & extendable
- **can be composed to (domain) specific language workbenches**

➡ ***Reuse of***

- ***modeling language***
- ***Tools***



YAKINDU Tools consist of ...



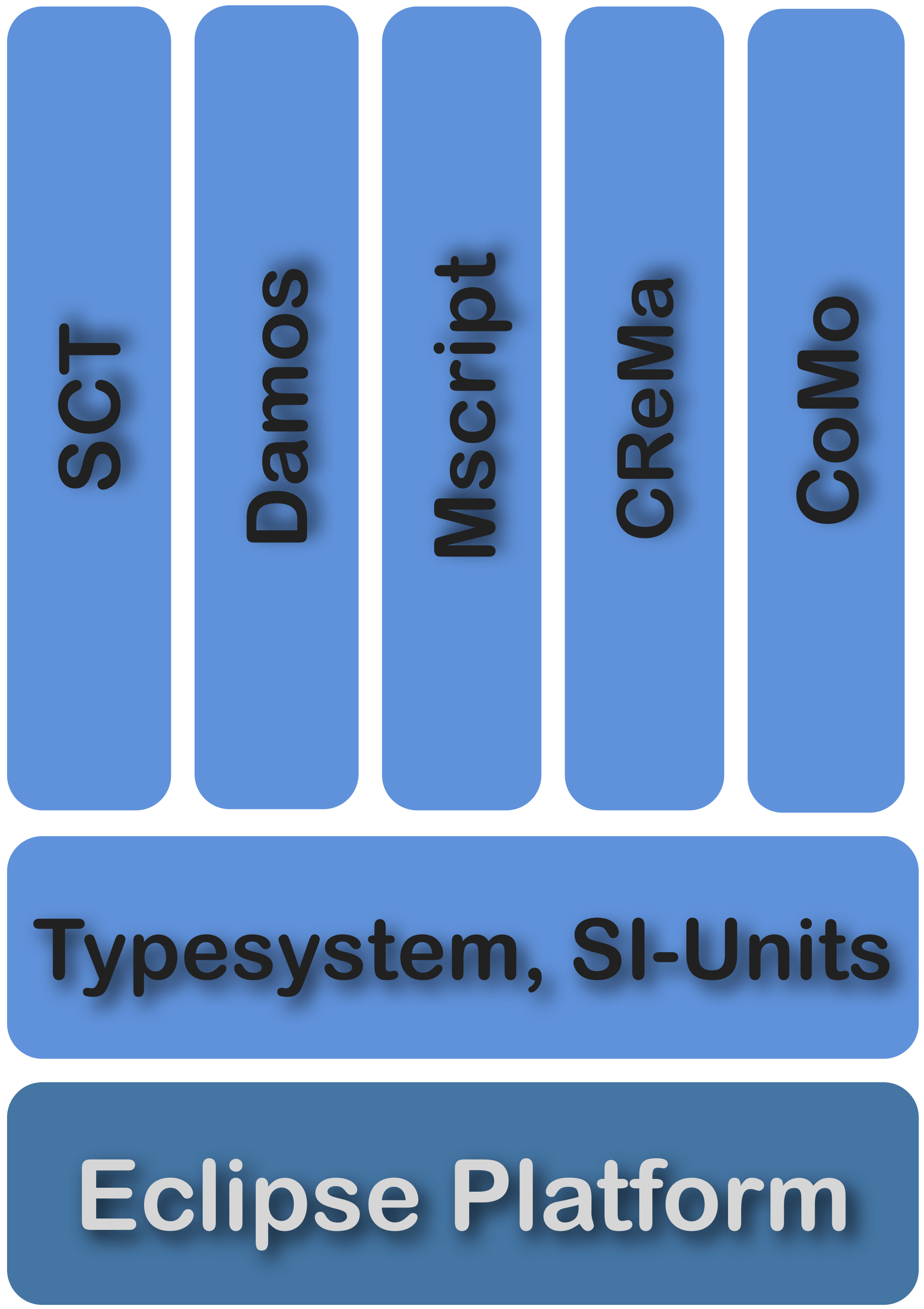
Yakindu is ...

- built on Eclipse
- open source
- available at Eclipse Labs

http://eclipselabs.org/p/yakindu

http://yakindu.org

Eclipse Project Proposal: 2011



What is Eclipse good for ?

Known as
Integrated Development Environment

Eclipse

- Application / Tool Platform
- Open Architecture
- Designed for Extensibility
- Strong Modeling Infrastructure
- Open Source
- Reduced Costs

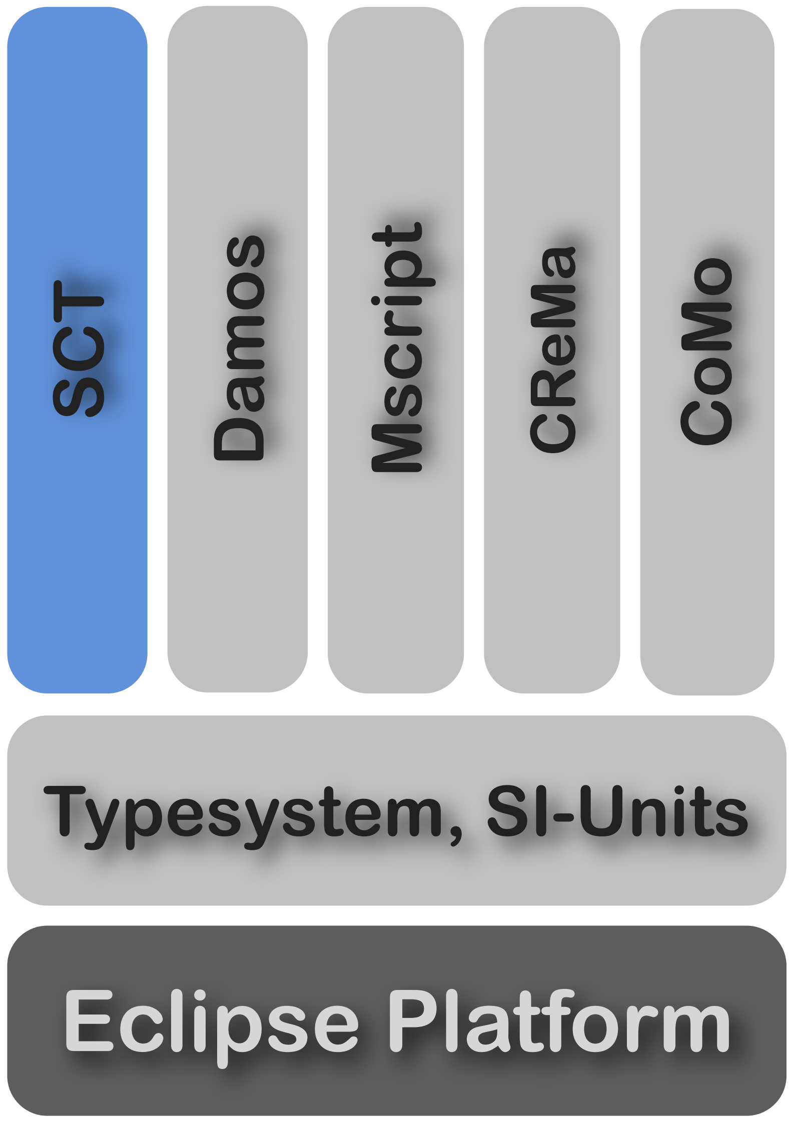
Eclipse

- Eclipse Modeling Framework
- Graphical Editing/Modeling Framework
- Xpand - Generator Framework
- Xtext - DSL Toolkit
- UML2 support
- ...

papyrus TOPCASED The Open Source toolkit for critical systems YAKINDU etrice.

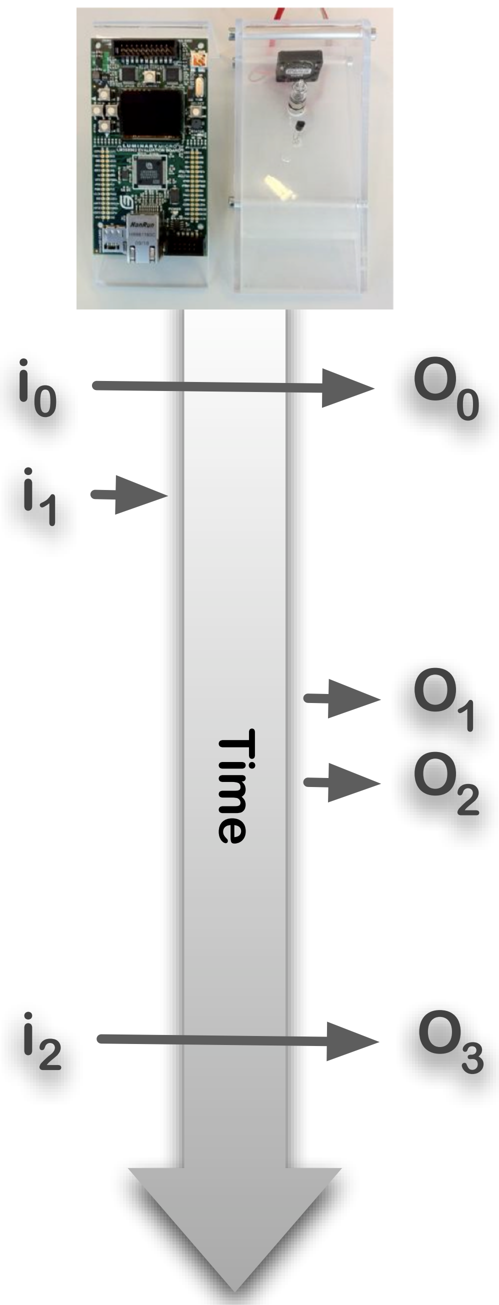
YAKINDU

SCT - Statecharts

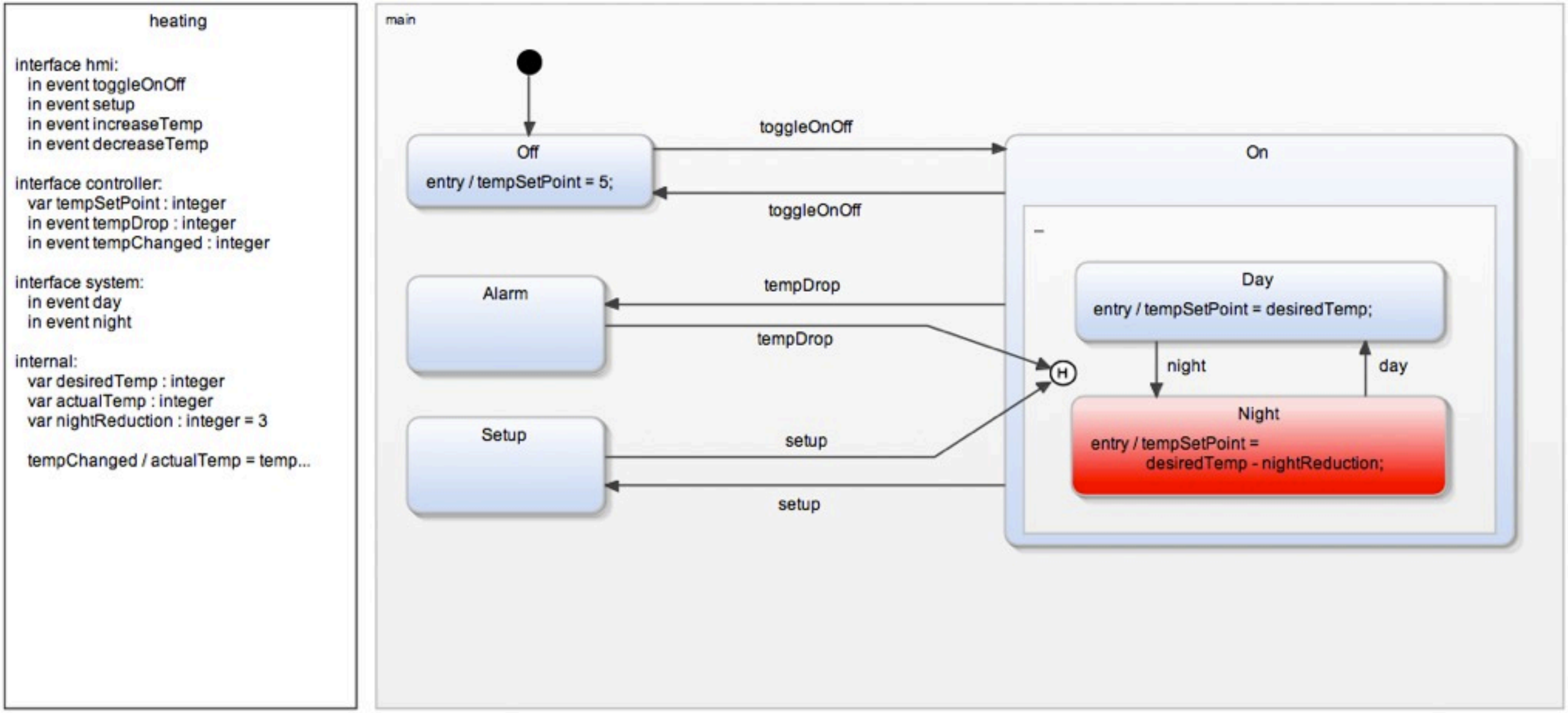


state machines (aka state charts)

- model reactive systems
- continuously interacts with the environment
- event driven
- focuses on transition of the systems state and it‘s reactions
- the state of the system evolves depending on previous inputs and time
- typically asynchronous



Yakindu Statechart Tools



```
heating

interface hmi:
  in event toggleOnOff
  in event setup
  in event increaseTemp
  in event decreaseTemp

interface controller:
  var tempSetPoint : integer
  in event tempDrop : integer
  in event tempChanged : integer

interface system:
  in event day
  in event night

internal:
  var desiredTemp : integer
  var actualTemp : integer
  var nightReduction : integer = 3

  tempChanged / actualTemp = temp...
```

main
Off
entry / tempSetPoint = 5;
toggleOnOff
toggleOnOff
On
Alarm
tempDrop
tempDrop
Setup
setup
setup
Day
entry / tempSetPoint = desiredTemp;
night
day
H
Night
entry / tempSetPoint = desiredTemp - nightReduction;

statechart properties

- based on statcharts as defined by David Harel
- close to UML state machines
- but:
  - YSCs are self contained with an interface well defined by events and variables
  - core execution semantics are cycle-driven and not event-driven
    - allows processing concurrent events
    - event driven behaviour can be defined on top
  - time is an abstract concept for statecharts
  - time control is delegated to the environment
- model interpreter and different flavours of generated code follow the same core semantics

Domain Specific Statecharts

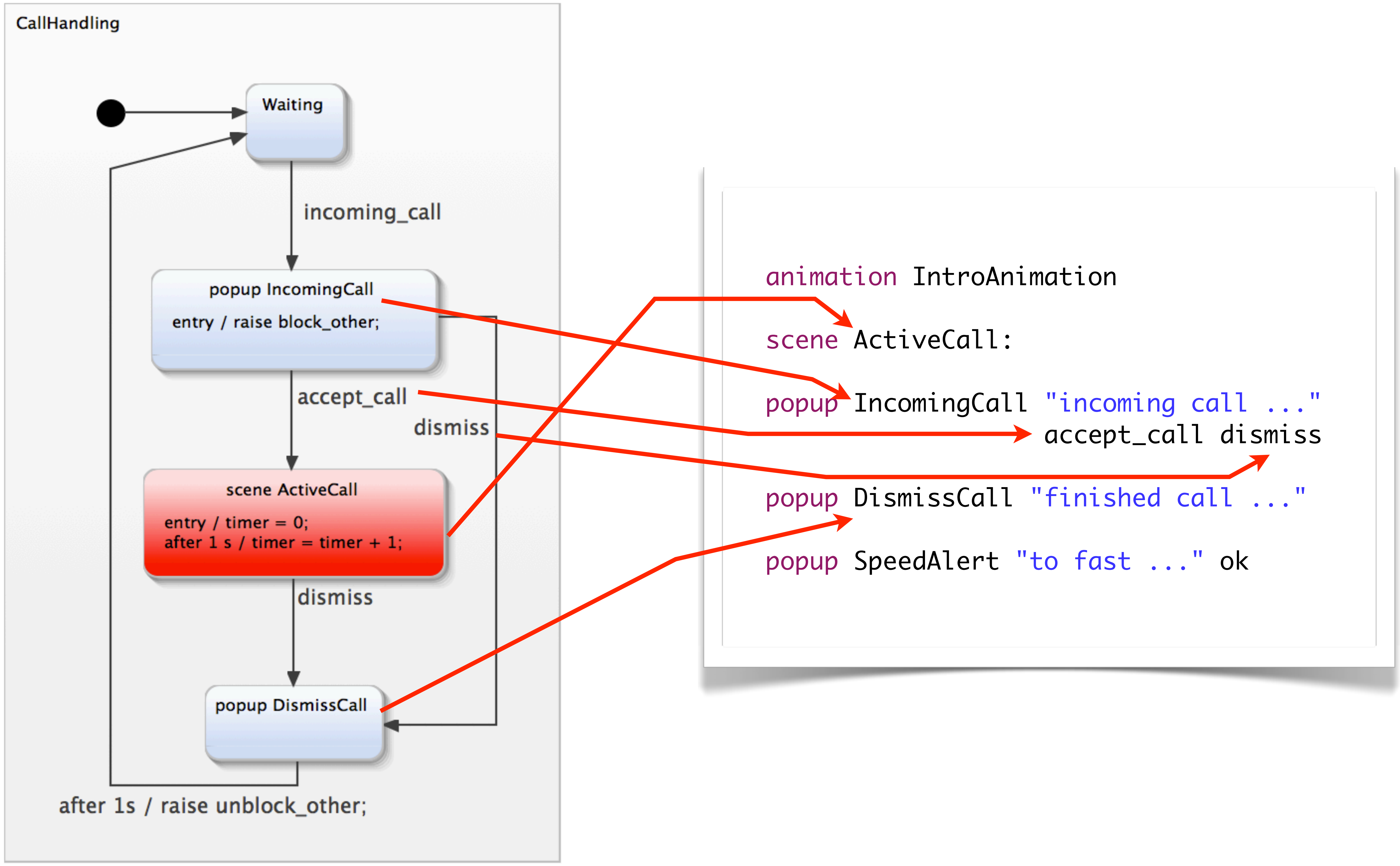
- Improving expressiveness and semantic integration by adopting domain concepts
- Integration of state based modeling with DSL workbenches
- SCT2 is built for extendability

- Example Domain: HMI Specification

Example DSL: HMI Contract

- Domain-Concepts:
  Scene, Transition, Animation, Popup
- Defined by a DSL:
  HMI-Contract
- HMI-Contract is a domain interface and supports technical decouplin

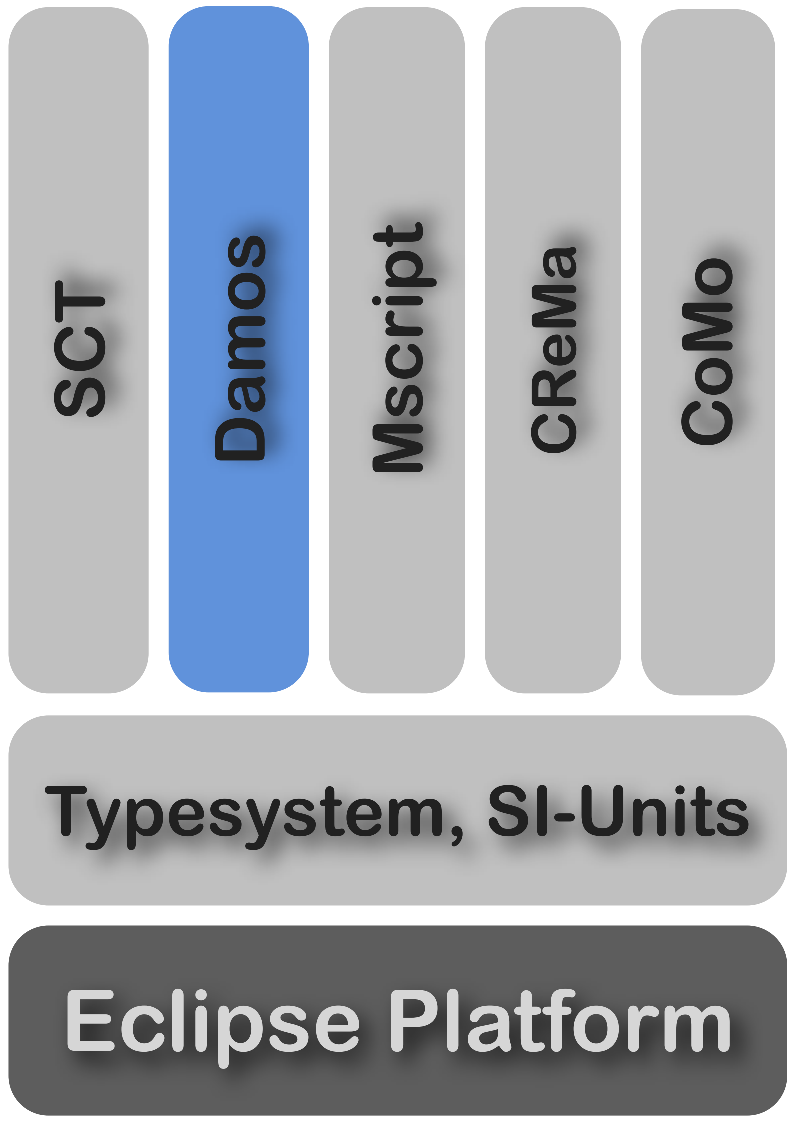
```
animation IntroAnimation

scene ActiveCall:

popup IncomingCall "incoming call ..." accept_call dismiss
popup DismissCall "finished call ..."

popup SpeedAlert "to fast ..." ok
```



```
animation IntroAnimation

scene ActiveCall:

popup IncomingCall "incoming call ..."
                    accept_call dismiss

popup DismissCall "finished call ..."

popup SpeedAlert "to fast ..." ok
```

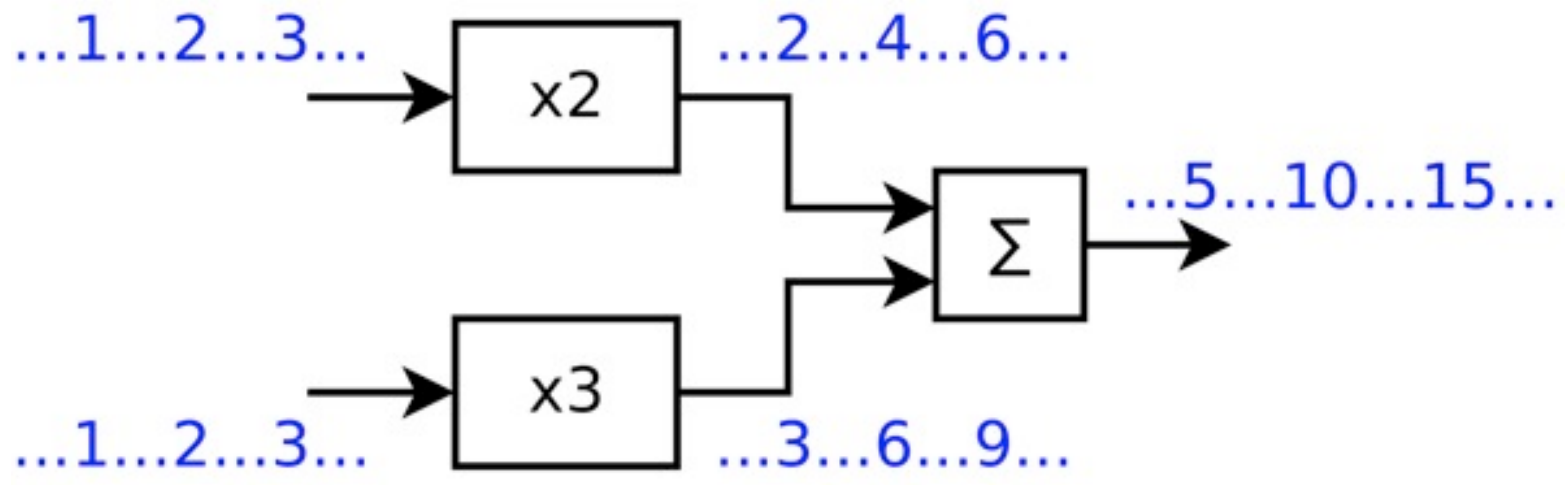
YAKINDU

Damos - block diagrams

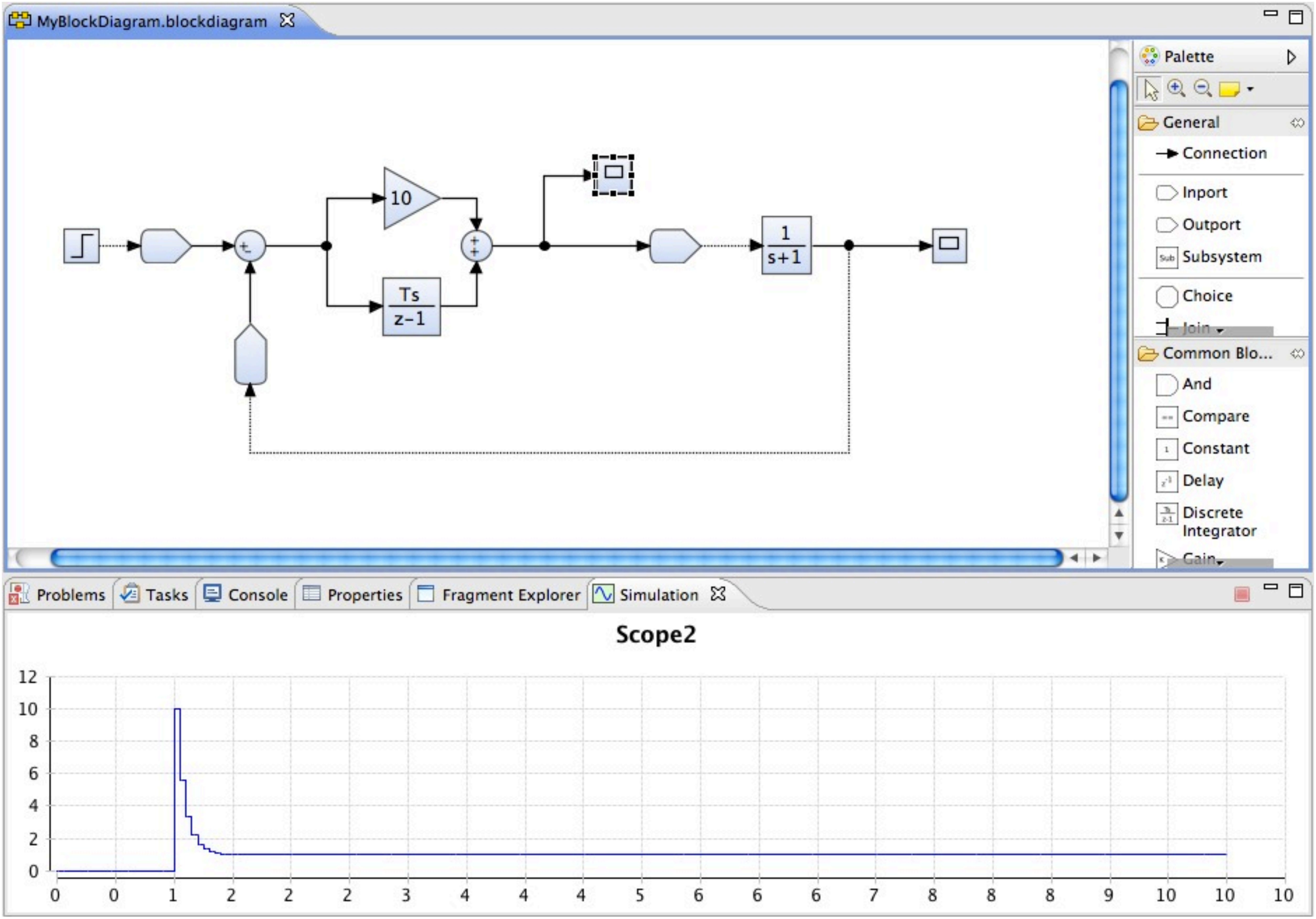


Data Flow-Oriented Modeling

- *Data* as main concept, instead of *state*
- Prevalent notation: Block diagrams
  - Block: System component's transfer function
  - Connection: Data flow (e.g. physical quantities)
- Technical applications: Control systems & digital signal processing



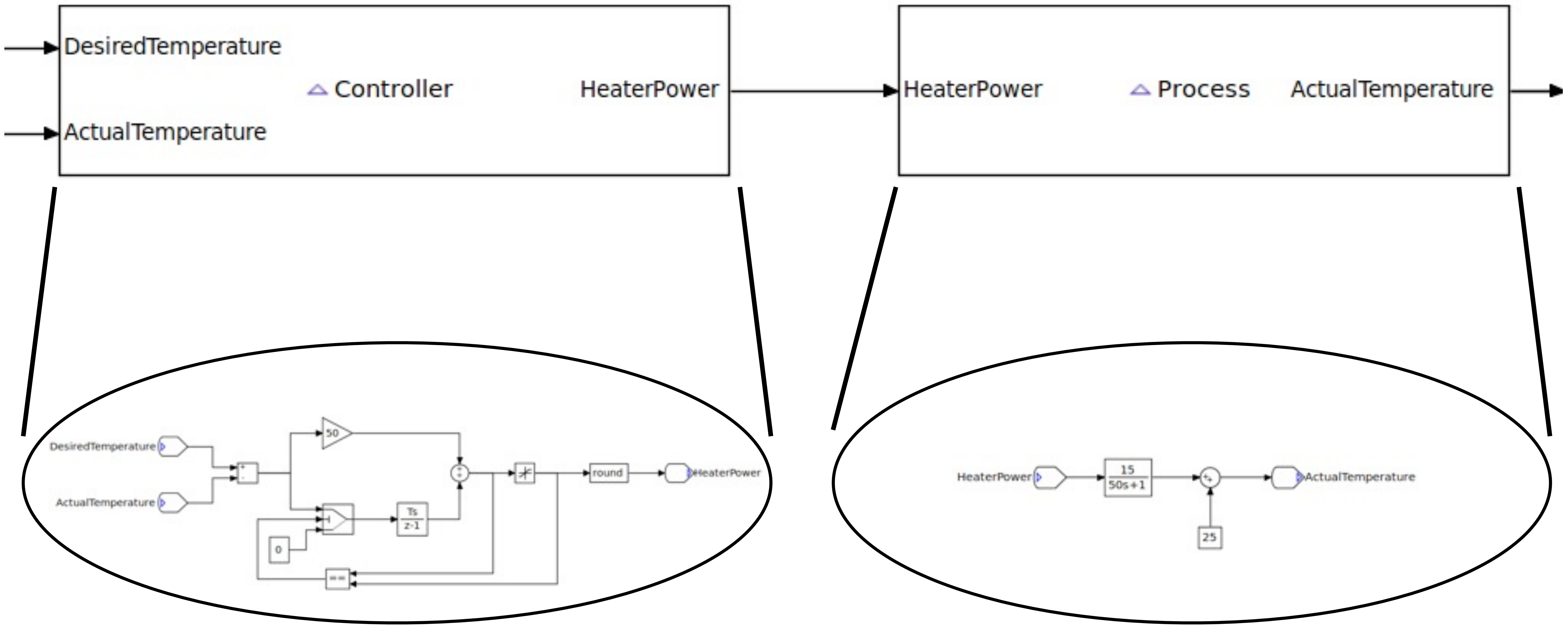
Damos Tooling



Structuring Models

- System components are divided into three categories
  - Device components (e.g. digital controller)
  - Environmental components (e.g. process)
  - Simulation interface components (e.g. step functions and scopes)
- Damos supports „two-dimensional“ structuring
  - Hierarchal structuring using subsystems
  - Cross-cutting structuring using system fragments

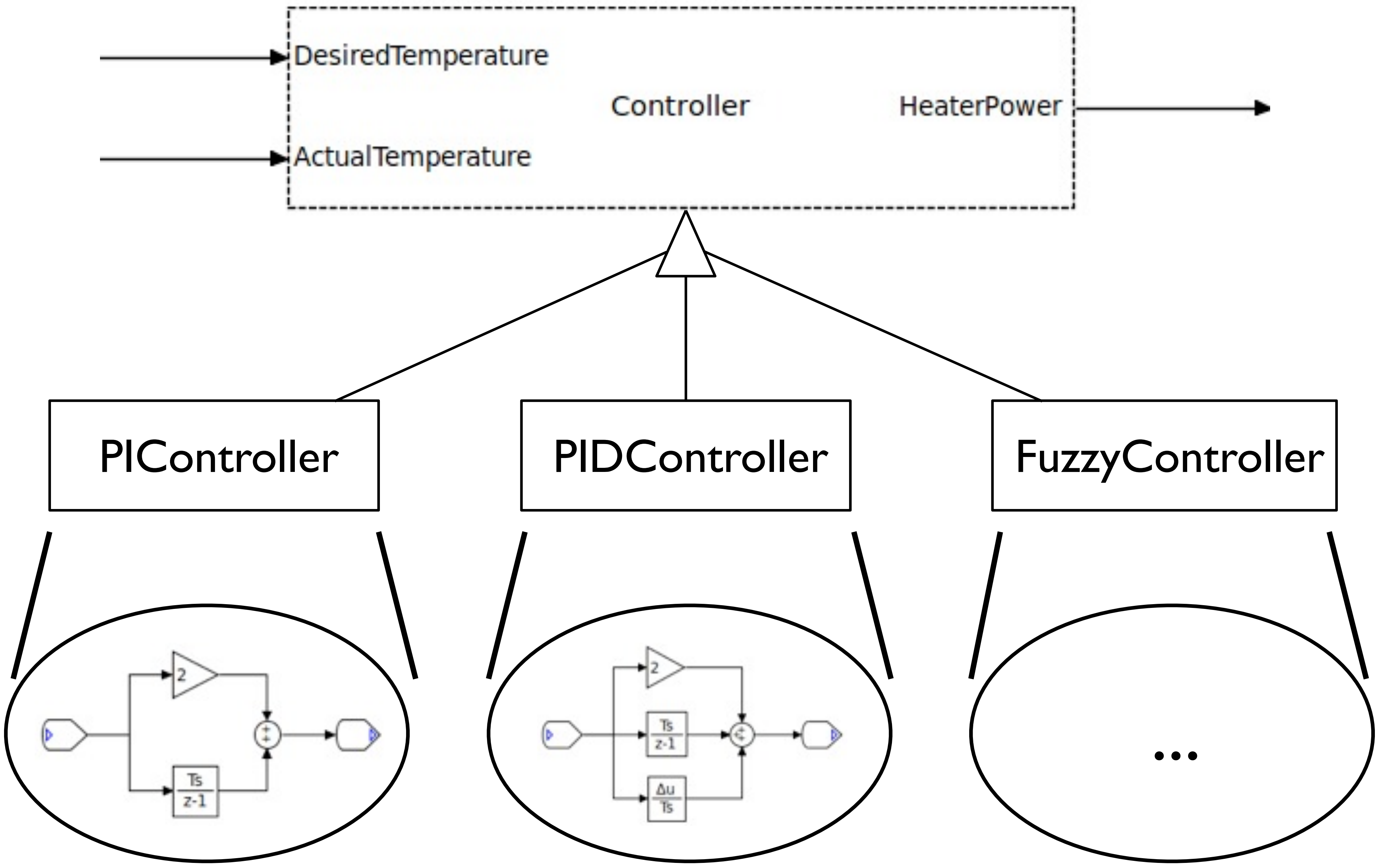
Subsystems


DesiredTemperature
ActualTemperature
Controller
HeaterPower
HeaterPower
Process
ActualTemperature
DesiredTemperature
ActualTemperature
50
0
==
Ts
z-1
round
HeaterPower
HeaterPower
15
50s+1
25
ActualTemperature

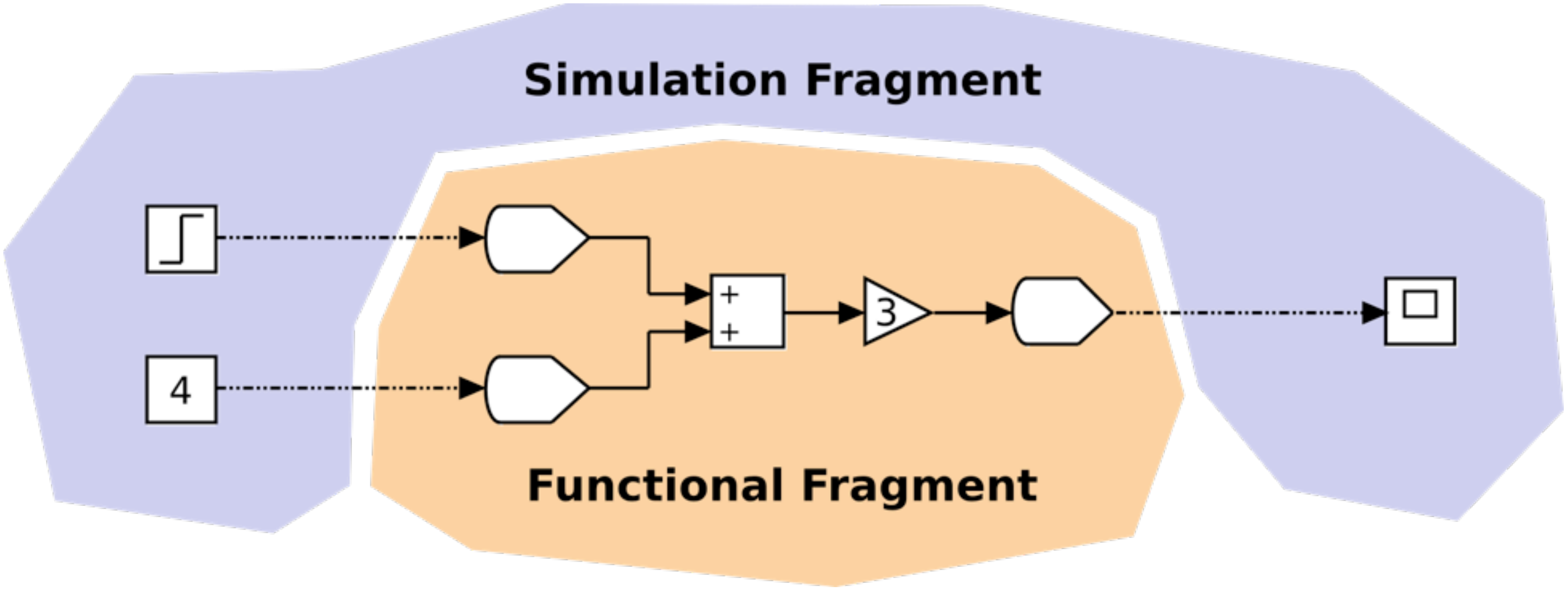
System Interfaces

- Defines system inlets and outlets, and their data types
- Subsystems specify their *provided interface*
- Realizing system is specified with *subsystem realization* model element
- Allows for specifying a subsystem without specifying a realization
- Can be used in product line engineering

Subsystem Realizations

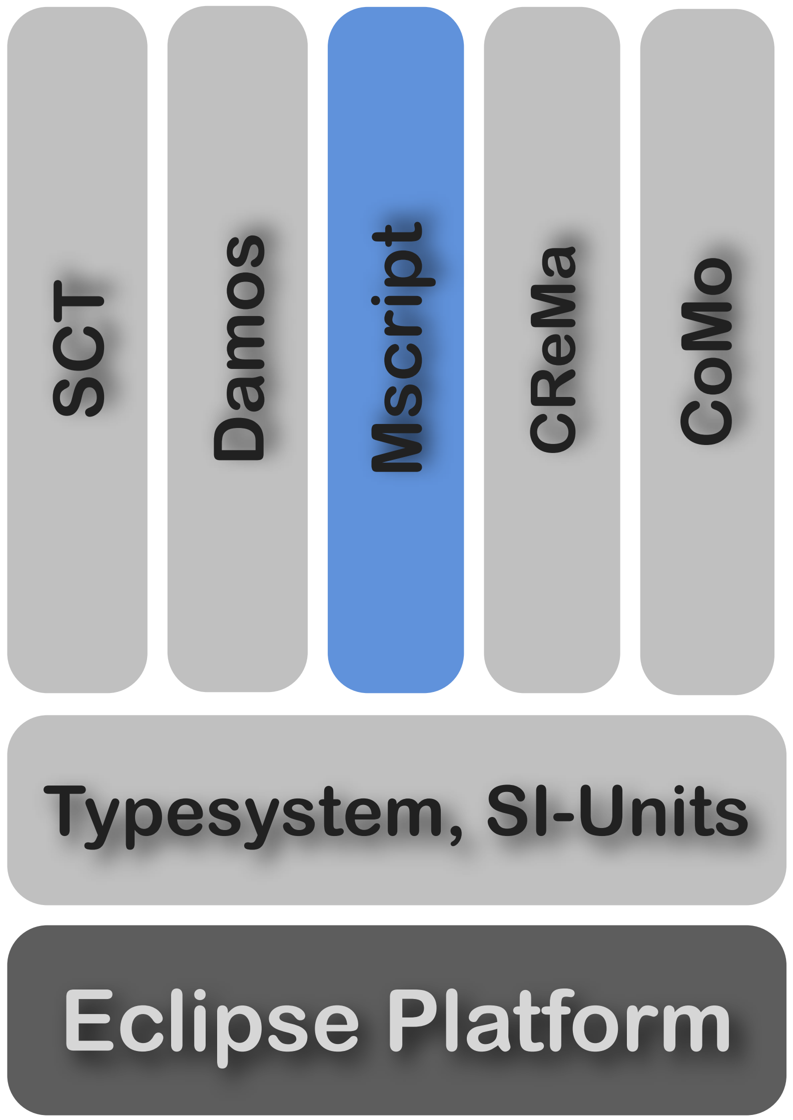


System Fragments



YAKINDU

Mscript



Mscript Example

Discrete derivative:

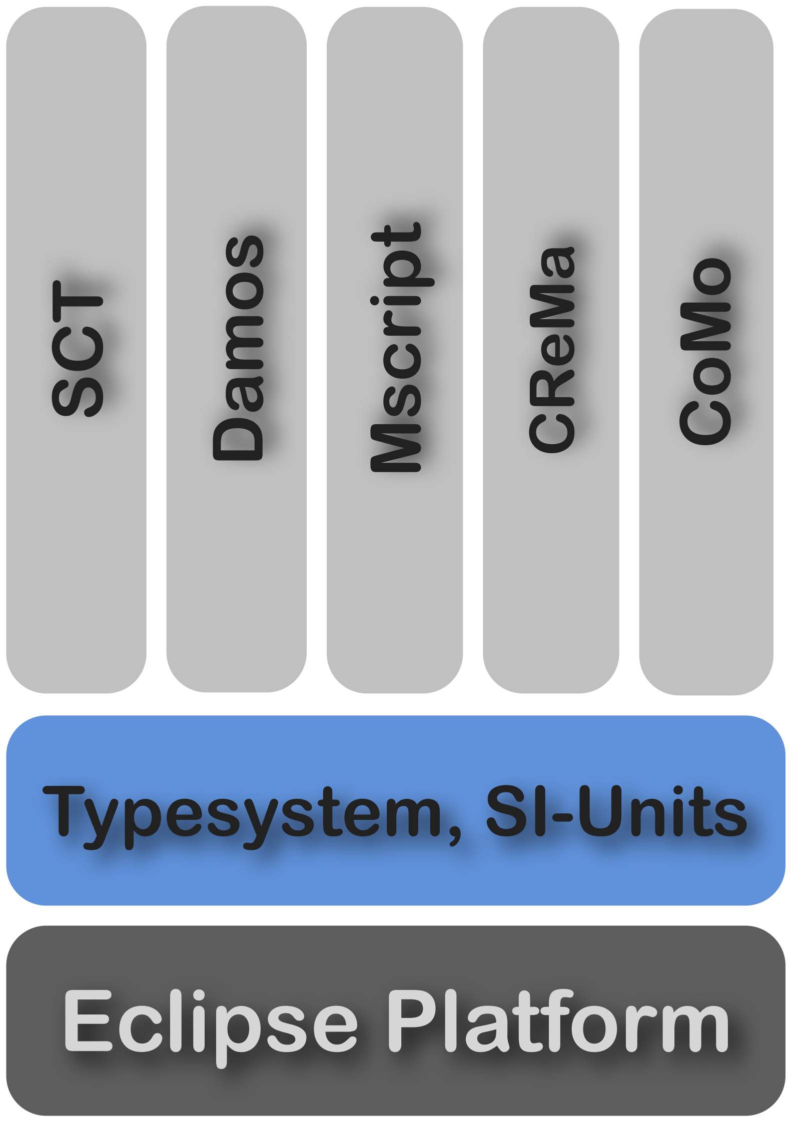
$$y(n) = \frac{x(n) - x(n-1)}{T_s}$$

$$\Downarrow$$

```
stateful func discreteDerivative<xinit, Ts>(x) -> y {
        x(-1) = xinit;
        y(n) = (x(n) - x(n-1)) / Ts;
}
```

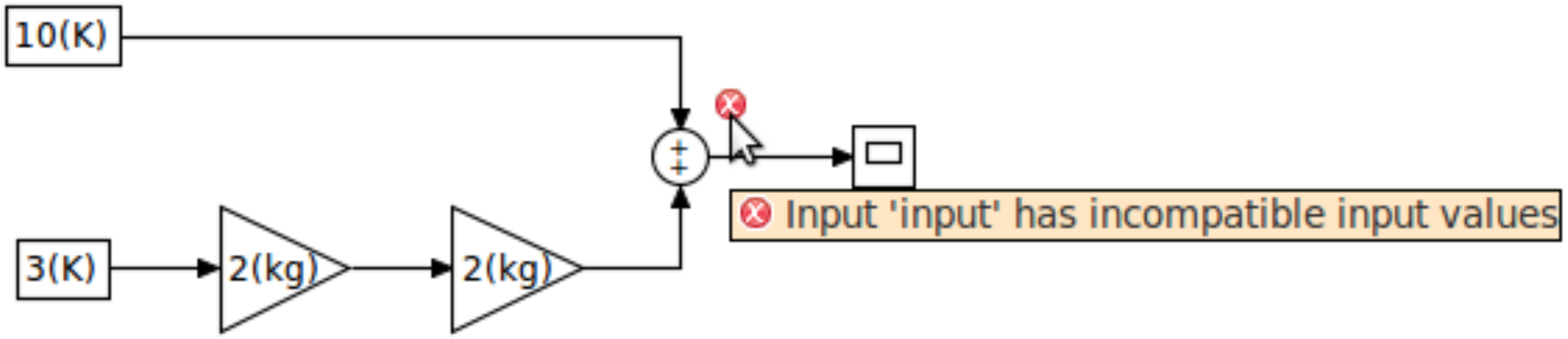
YAKINDU

Typesystem & SI-Units



Units of Measurement

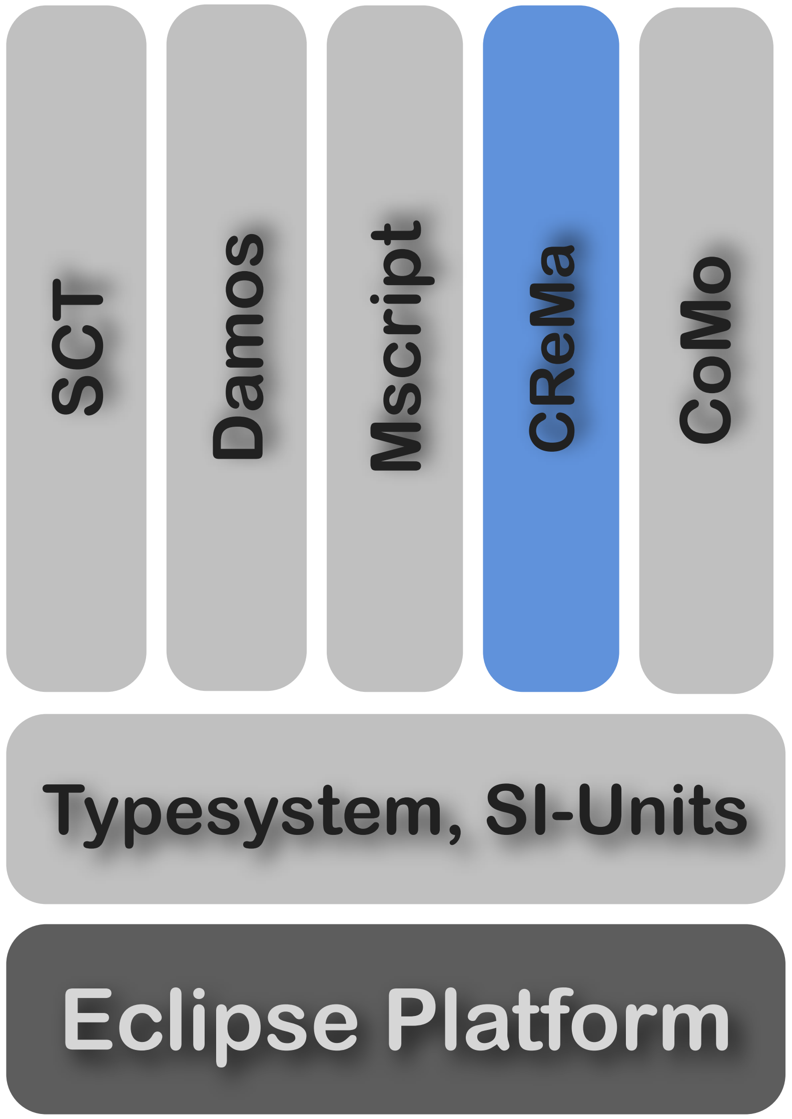
- Numeric data types incorporate unit of measurement
- When no unit is specified, dimensionless value is assumed
- Used for model validation



Input 'input' has incompatible input values

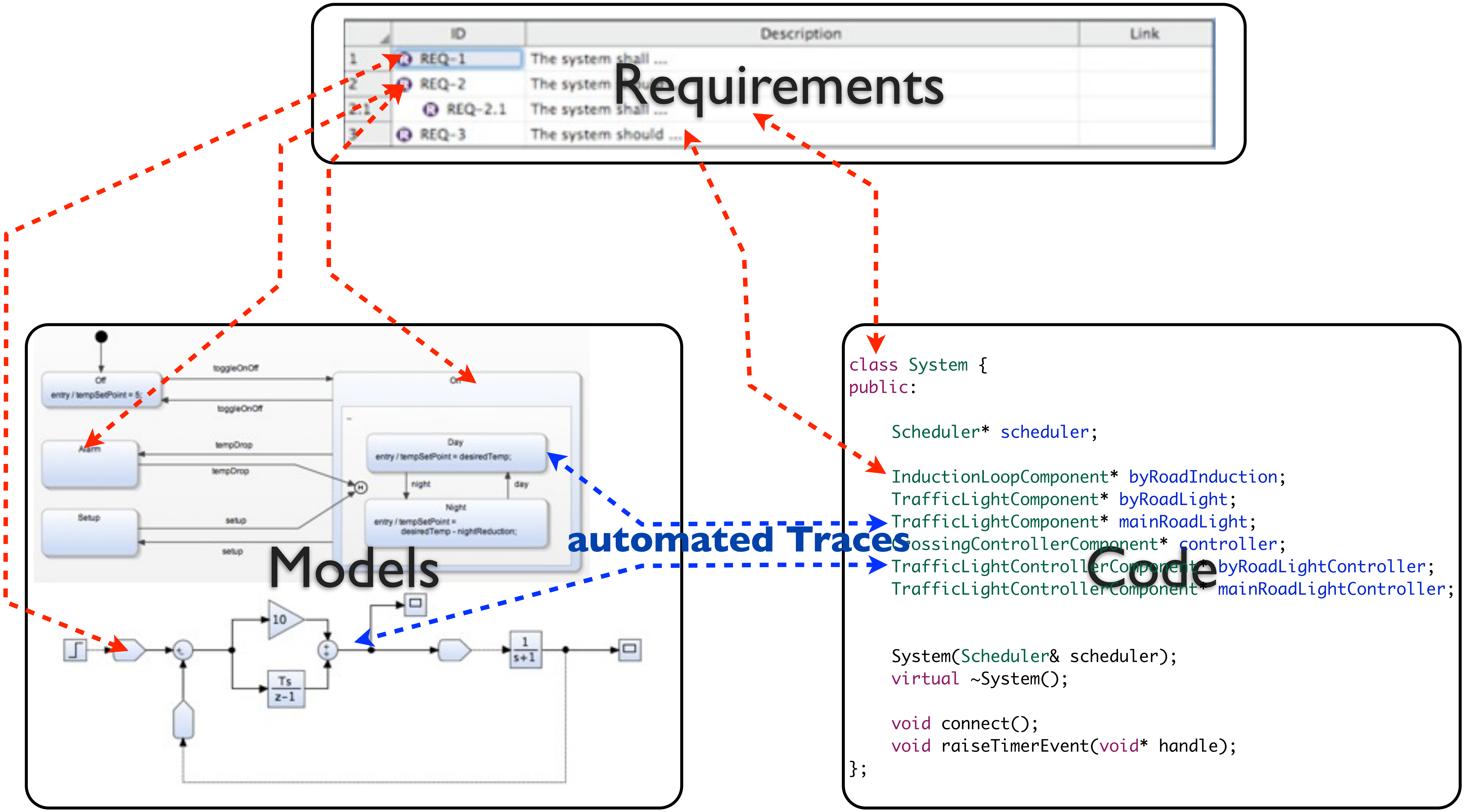
YAKINDU

Requirements & Traceability



Aim ...

- bring ***requirements*** into the ***development environment*** (Eclipse)
- provide ***tracing infrastructure***



	ID	Description	Link
1	REQ-1	The system shall ...	
2	REQ-2	The system shou...	
2.1	REQ-2.1	The system shall ...	
3	REQ-3	The system should ...	

```
class System {
public:

    Scheduler* scheduler;

    InductionLoopComponent* byRoadInduction;
    TrafficLightComponent* byRoadLight;
    TrafficLightComponent* mainRoadLight;
    CrossingControllerComponent* controller;
    TrafficLightControllerComponent* byRoadLightController;
    TrafficLightControllerComponent* mainRoadLightController;


    System(Scheduler& scheduler);
    virtual ~System();

    void connect();
    void raiseTimerEvent(void* handle);
};
```

RIF / ReqIF

- Ecore Model, including serialization to XML
  - RIF 1.1a - done
  - RIF 1.2 - done
  - ReqIF Beta2 - done

- Derived from the specification by model transformation
- Driven by itemis
- EPL (Eclipse Public License)
- Currently submitting an Eclipse project proposal

Traceablility with CReMa

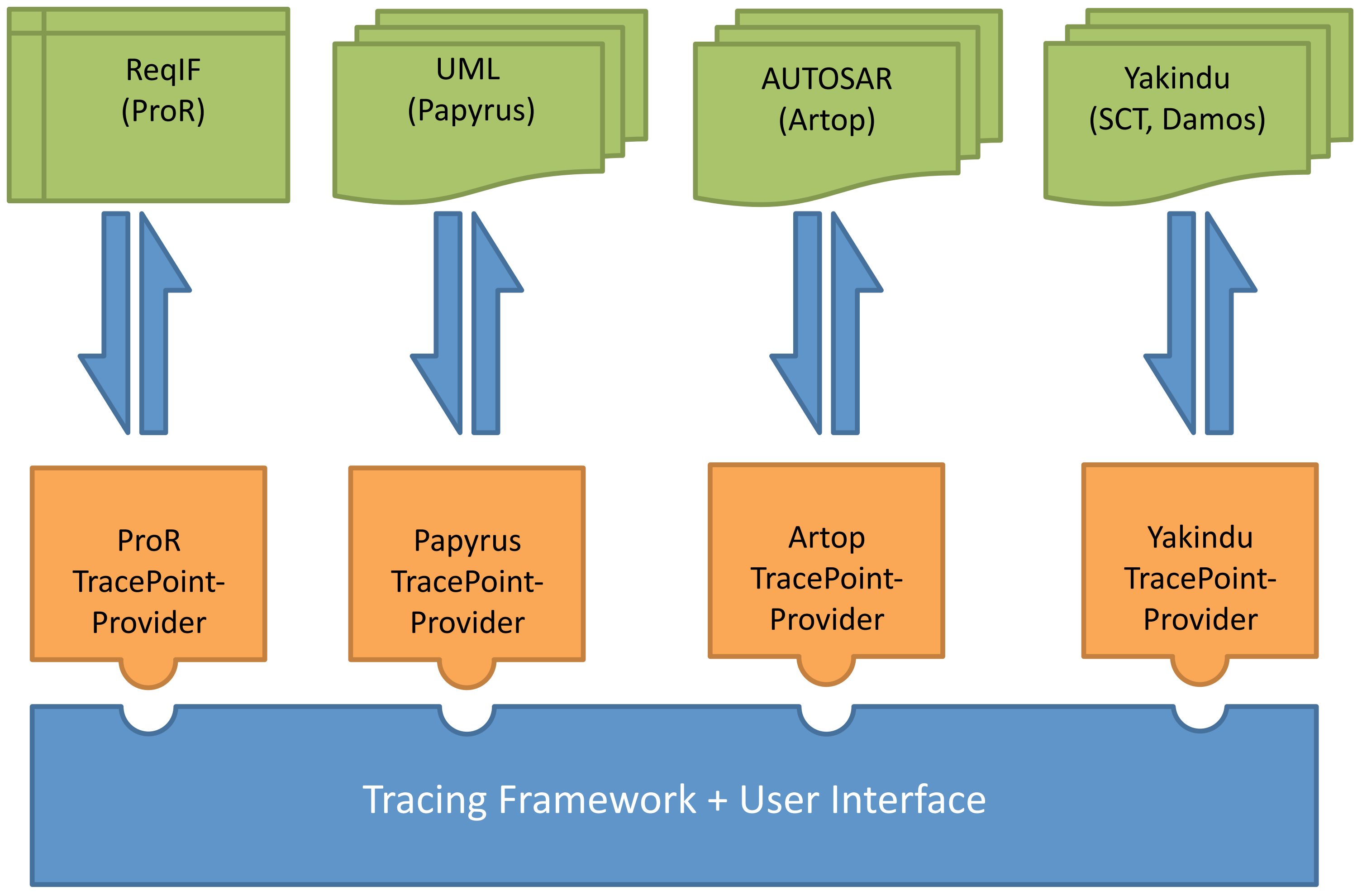
- CReMa - Cross Relational Manager

- Modular & extendable architecture
- Non invasive - don‘t change the target models

- Targets: requirements, models, code

- Result: any realtionsships in any context

YAKINDU CReMa


ReqIF
(ProR)
UML
(Papyrus)
AUTOSAR
(Artop)
Yakindu
(SCT, Damos)
ProR TracePoint-Provider
Papyrus TracePoint-Provider
Artop TracePoint-Provider
Yakindu TracePoint-Provider
Tracing Framework + User Interface

TracePoint A	TracePoint B
RIF://ID-238	UML://GUID-FF-AD-3M
RIF://ID-238	RES://model.uml
UML:://GUID-0B-CD-DD	SCT://_hb3fr1654h6

Questions & Comments