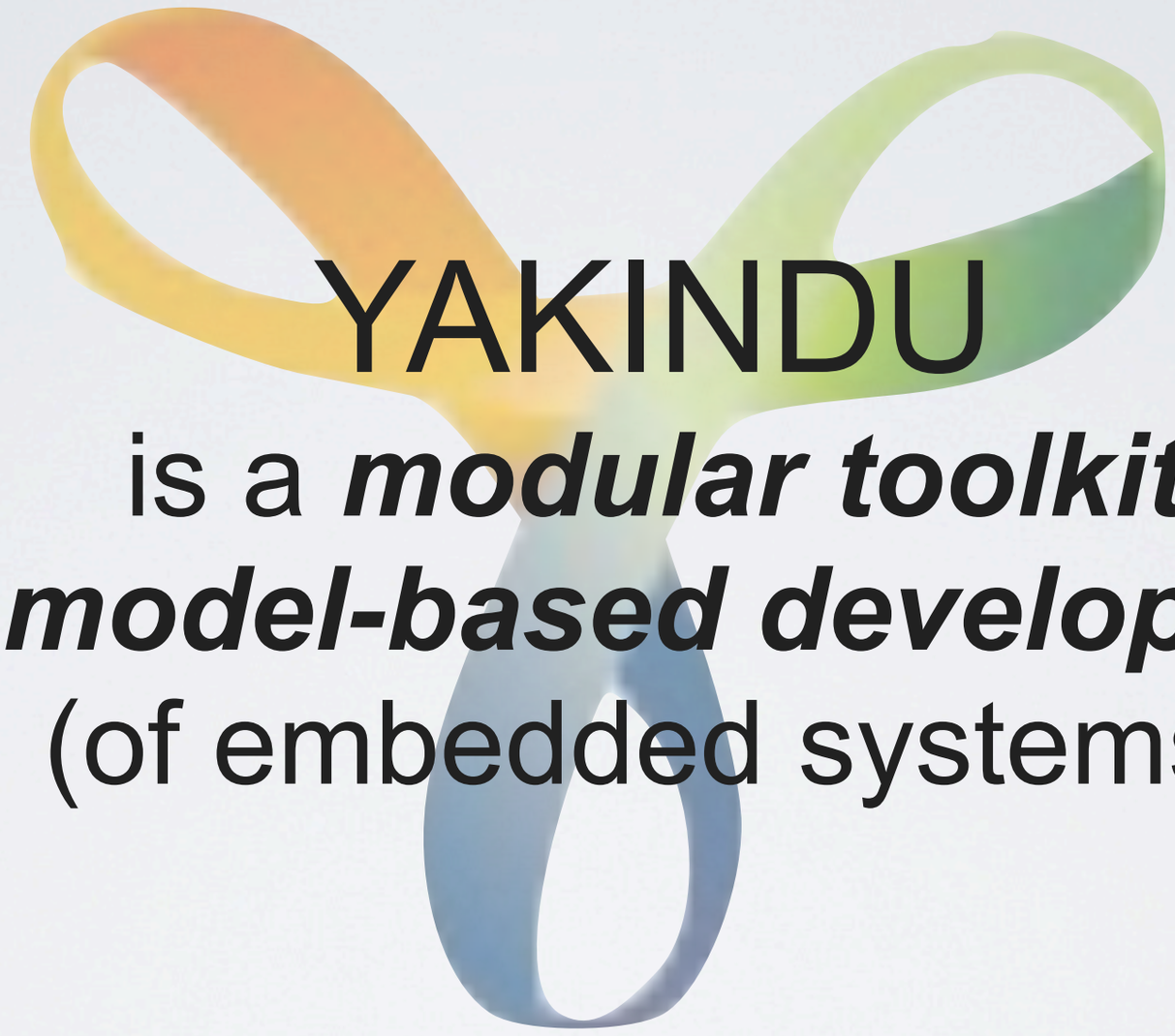
Yakindu SCT

Domain-Specific Statecharts

Alexander Nyßen, Axel Terfloth
itemis AG



YAKINDU

is a ***modular toolkit***

for ***model-based development***

(of embedded systems)

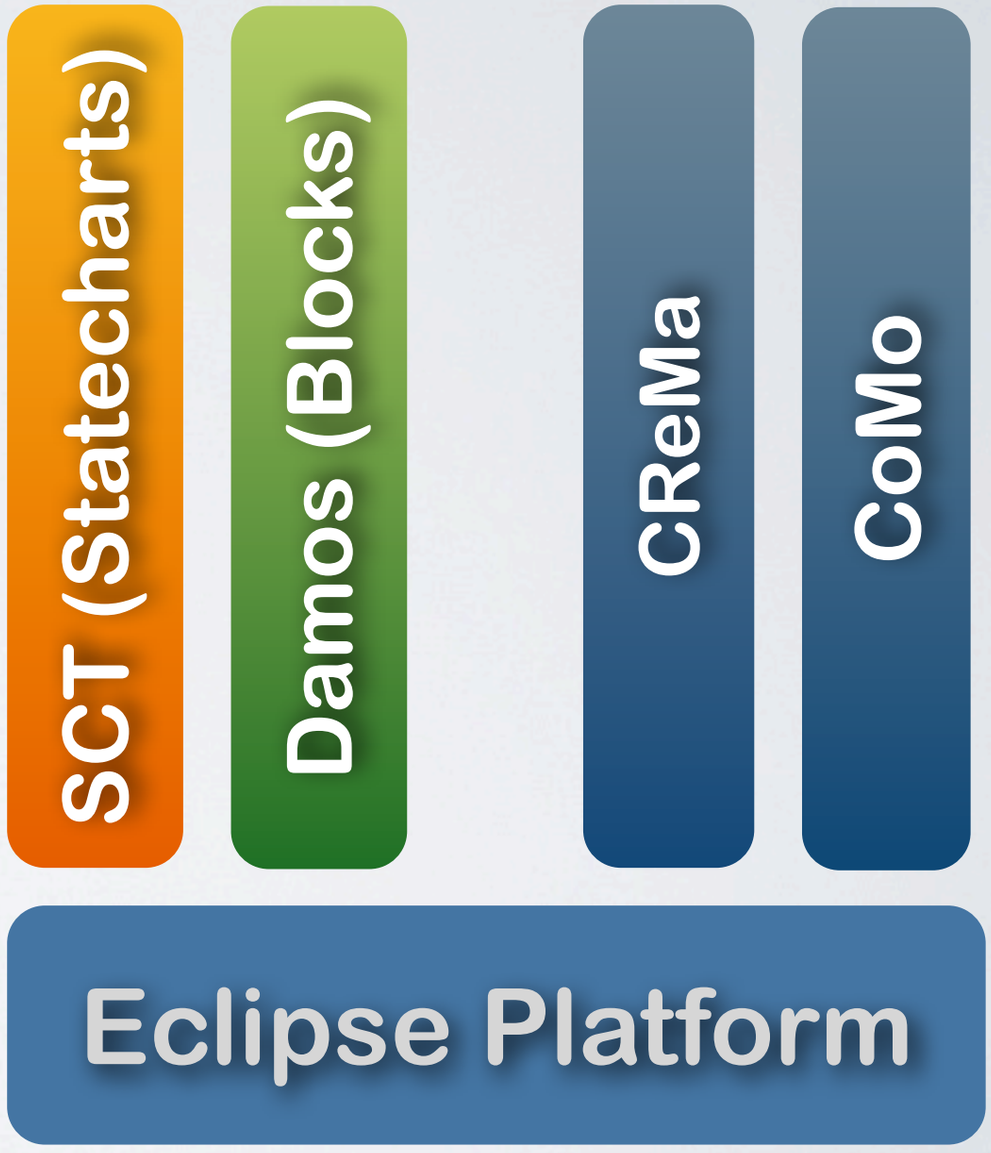
YAKINDU Modules

- independent and self-contained
- not bound to a specific methodology
- **usable on their own**

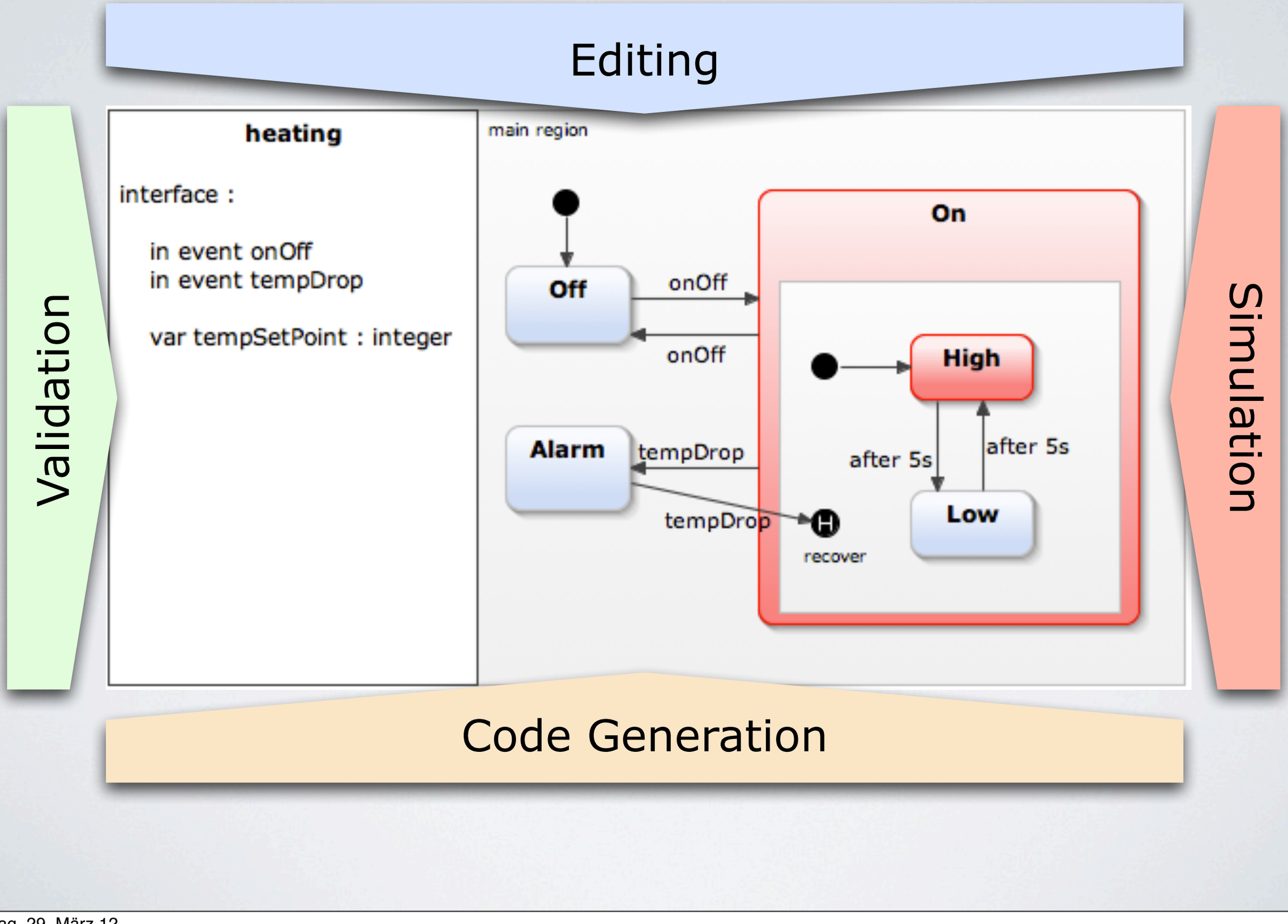
- open & extendable
- **composable to (domain-specific) language workbenches**

➡ **Reuse of**

- **modeling language**
- **tools**

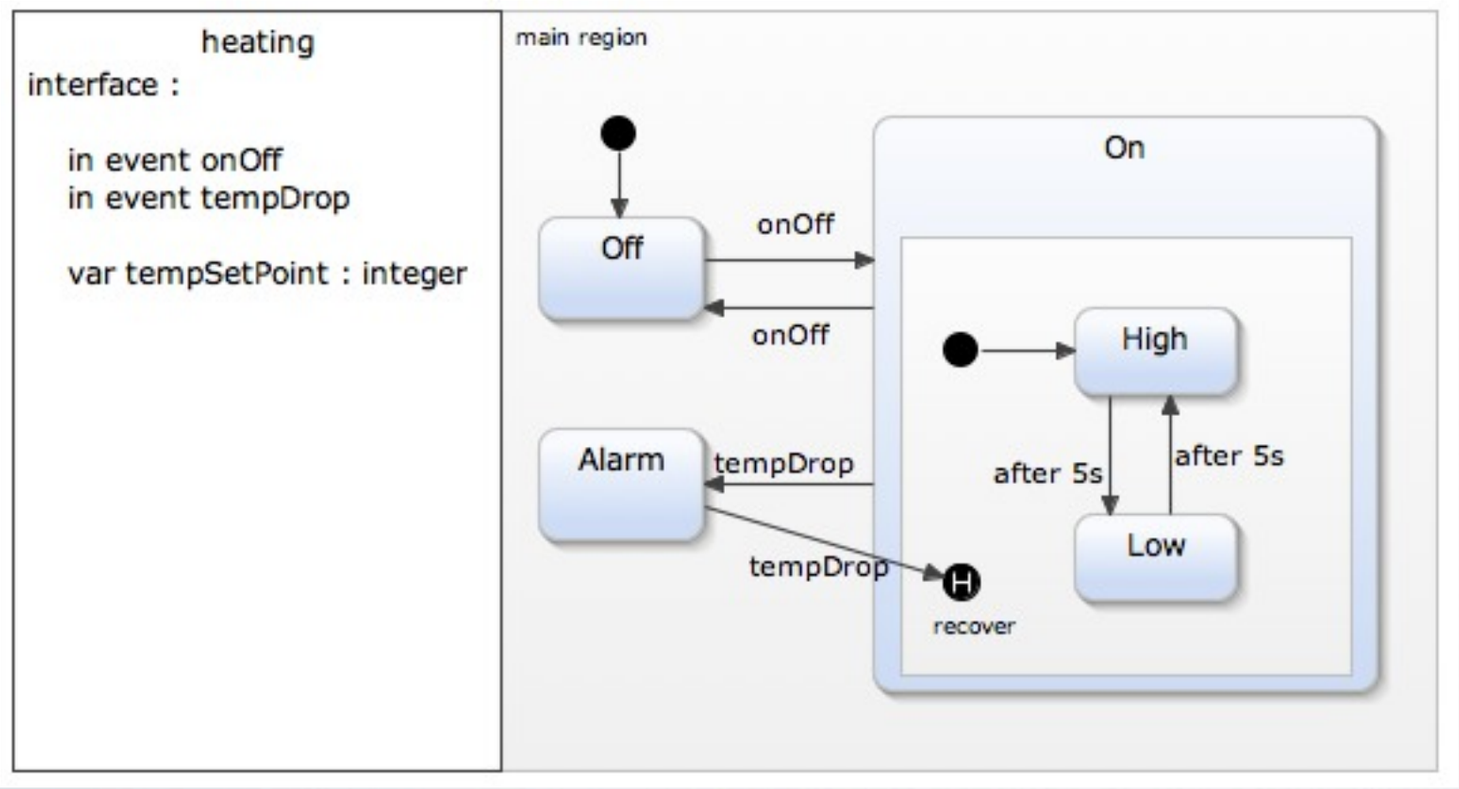


YAKINDU Statechart Tools (SCT)



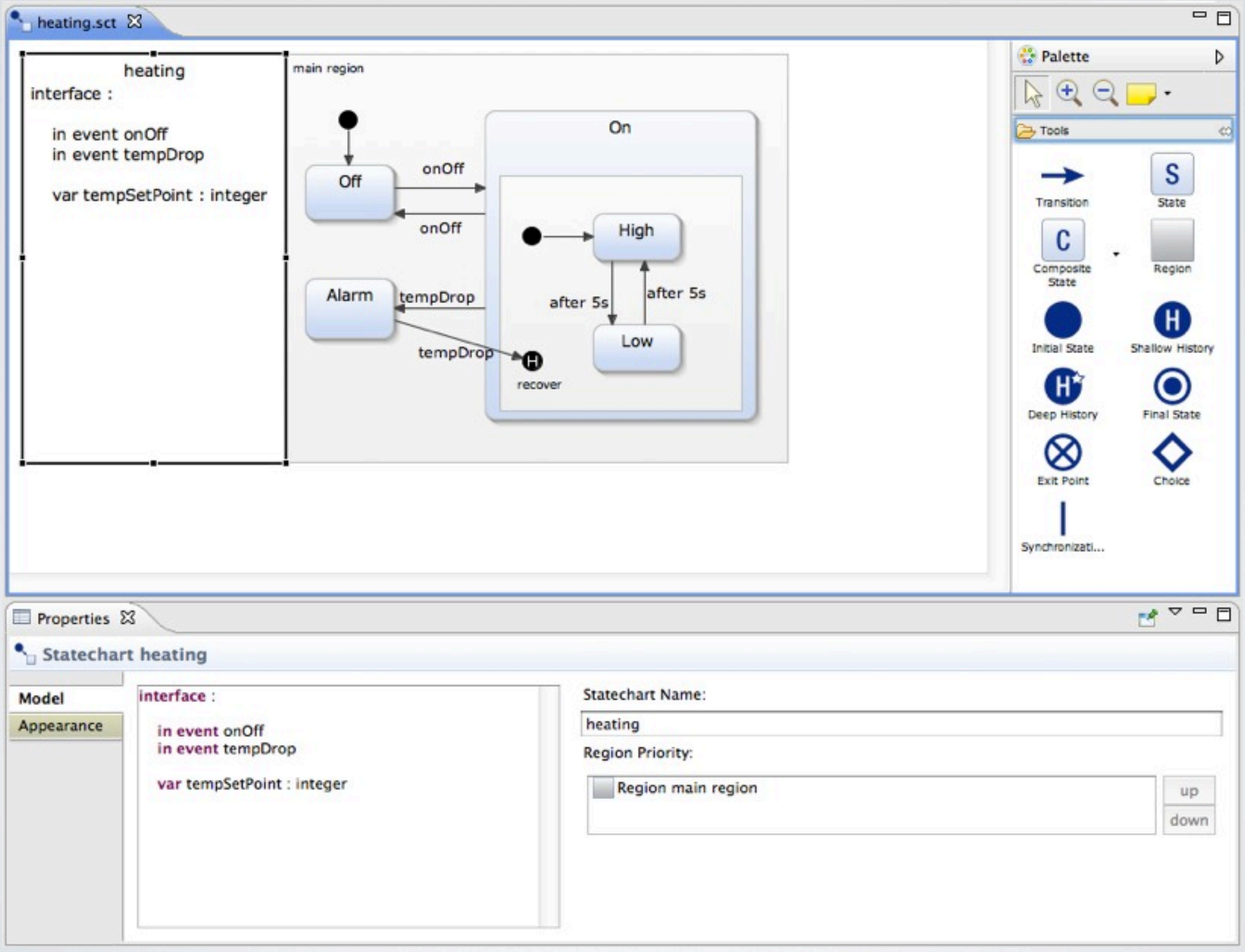
YAKINDU Statechart

- **Formalism** similar to **state machines** as defined by David Harel, but:
  - **self-contained** with a well defined **interface**
  - with a **cycle-based** execution semantics

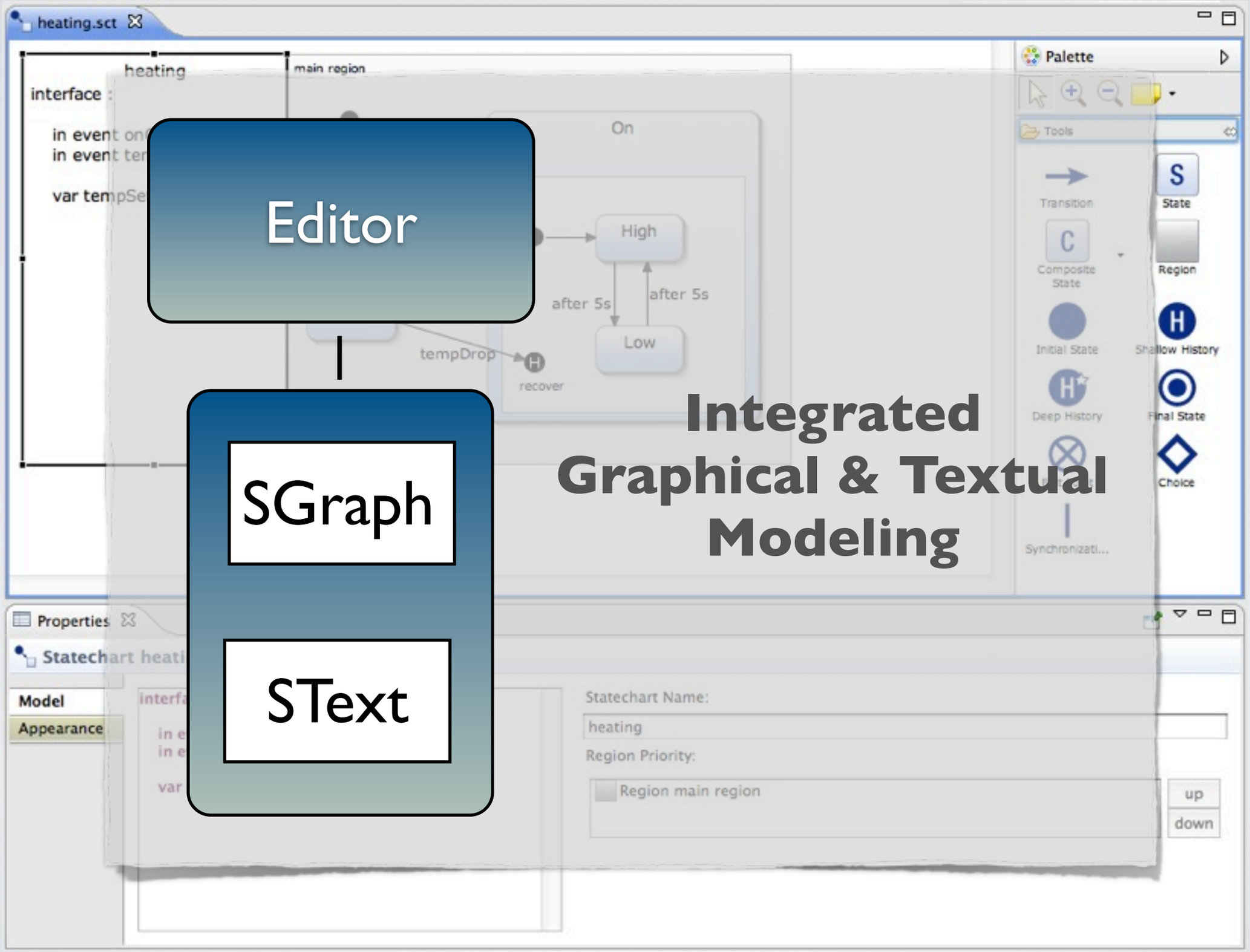


- allows processing concurrent events
- event-driven behavior can be defined on top
- time control is delegated to the environment

Yakindu SCT - Editing



Yakindu SCT - Editing



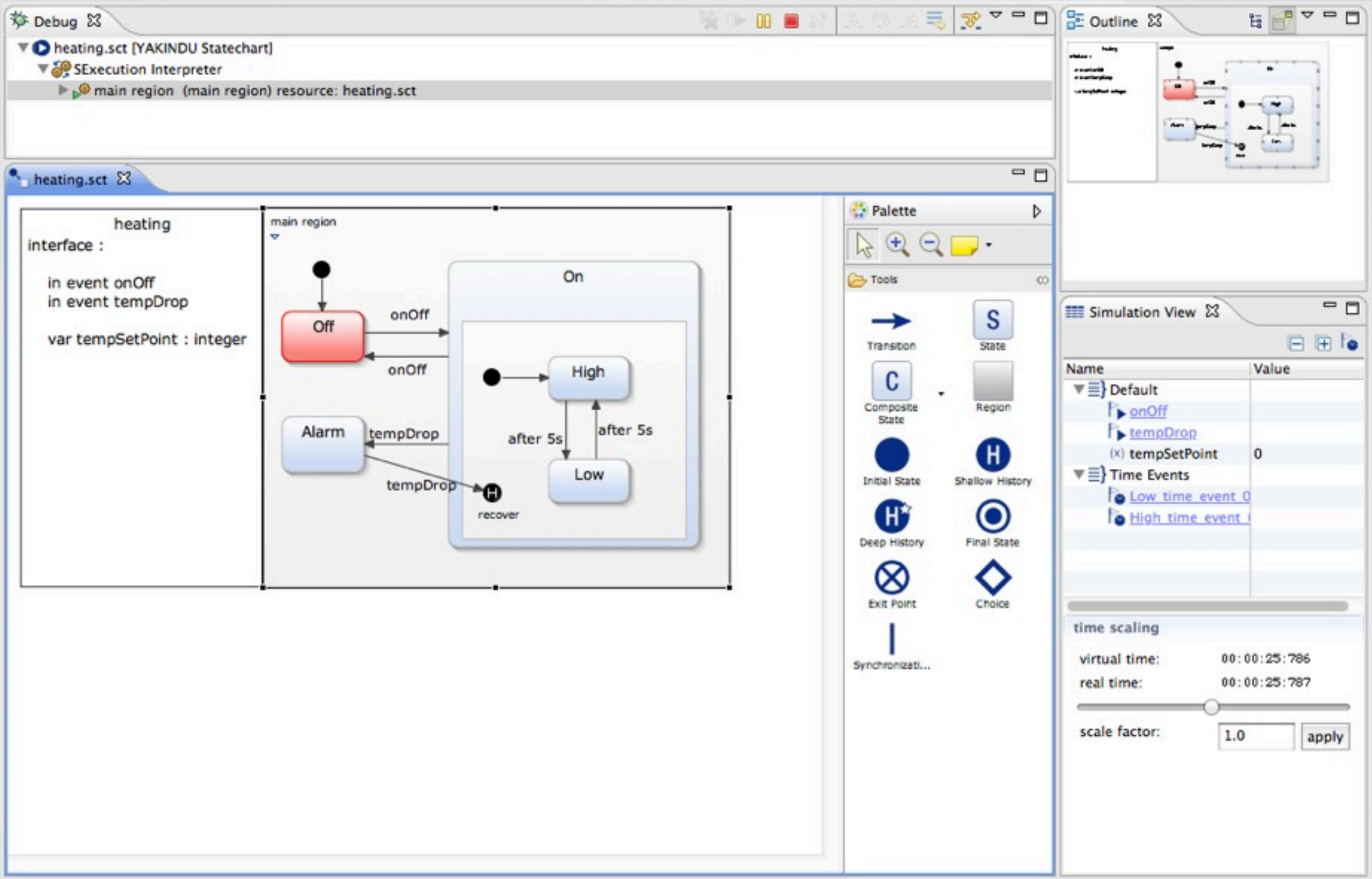
Editor

SGraph

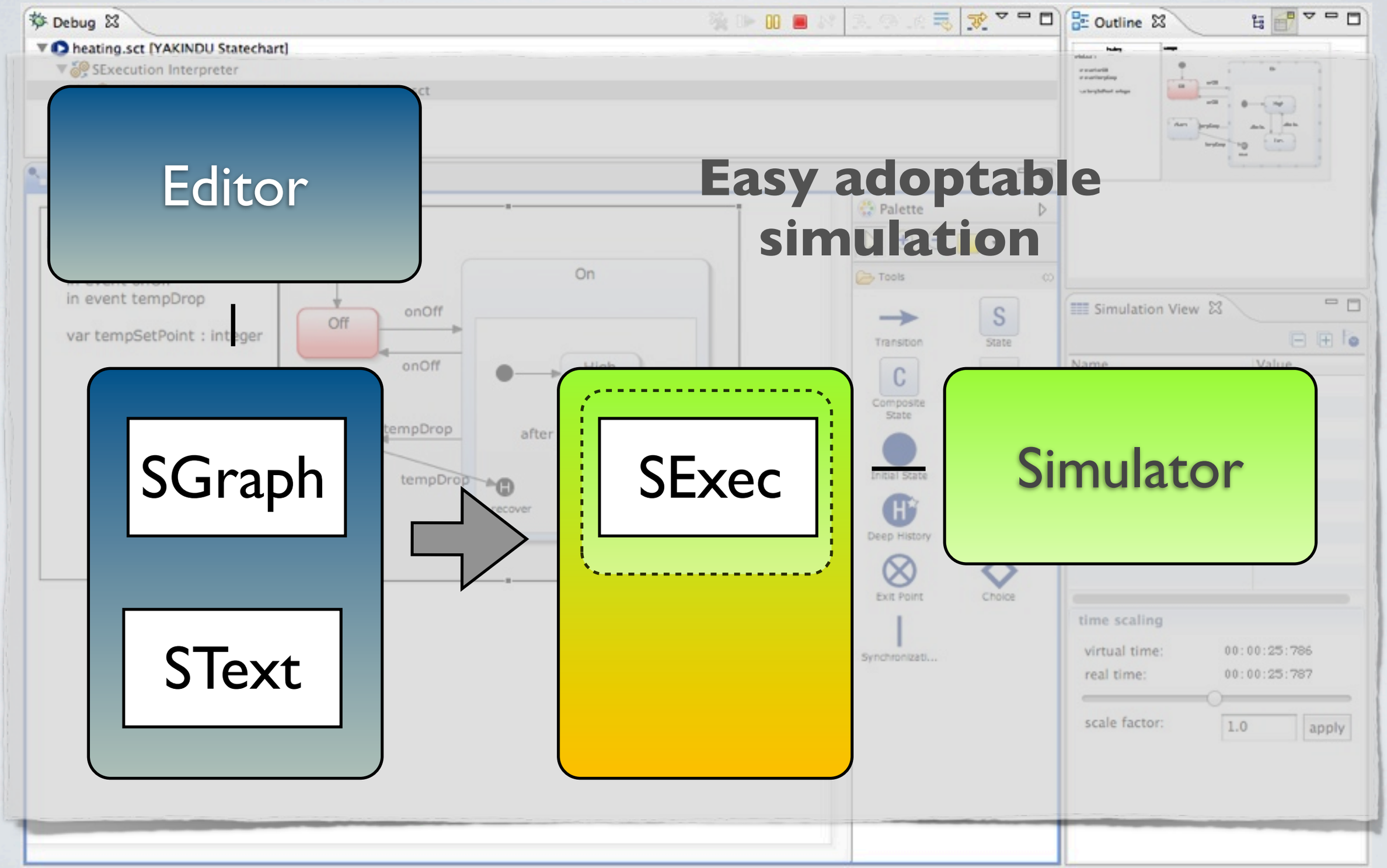
SText

Integrated Graphical & Textual Modeling

Yakindu SCT - Simulation



Yakindu SCT - Simulation



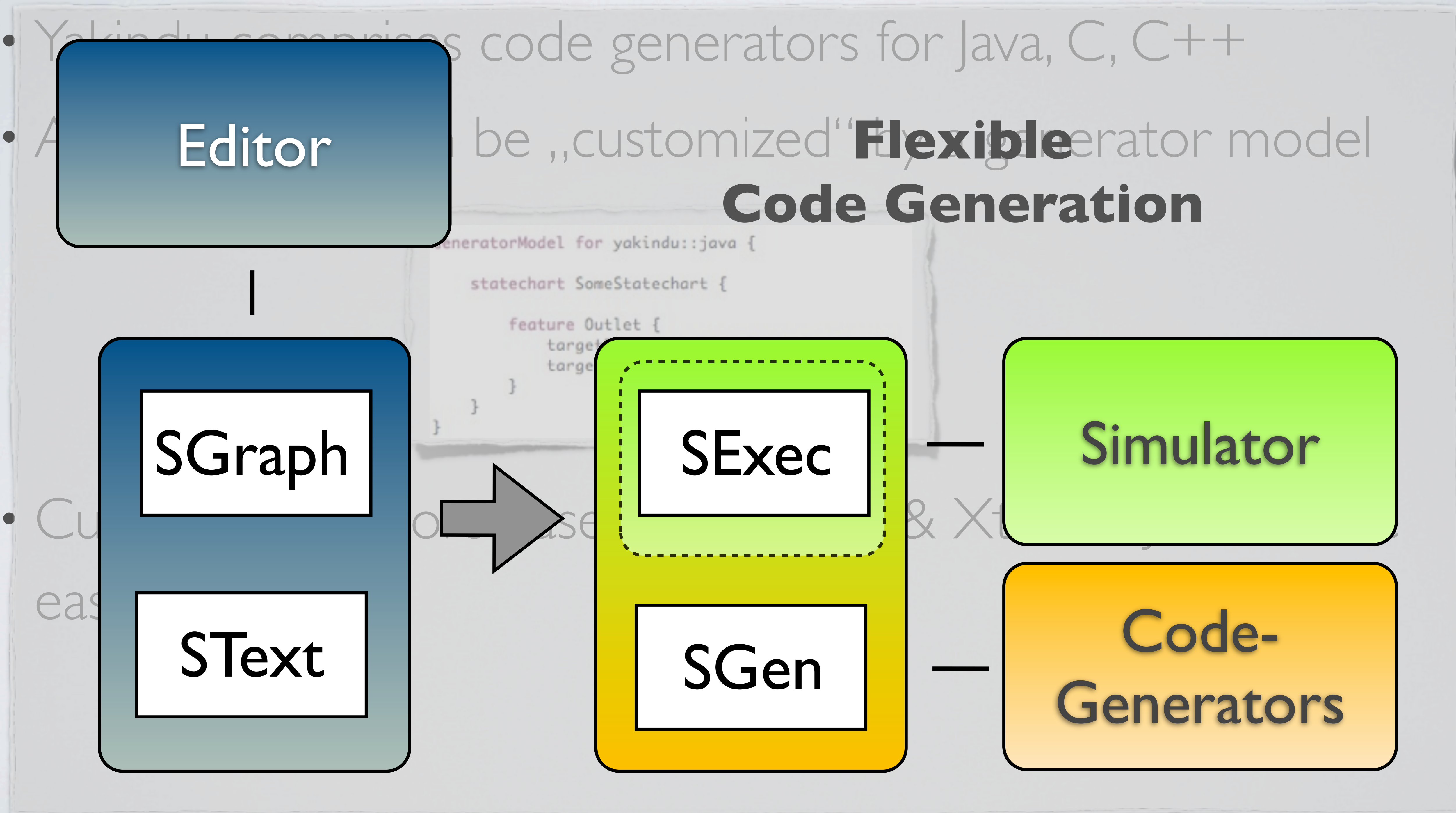
Yakindu SCT - Code Generation

- Yakindu comprises code generators for Java, C, C++
- All generators can be „customized“ by a generator model

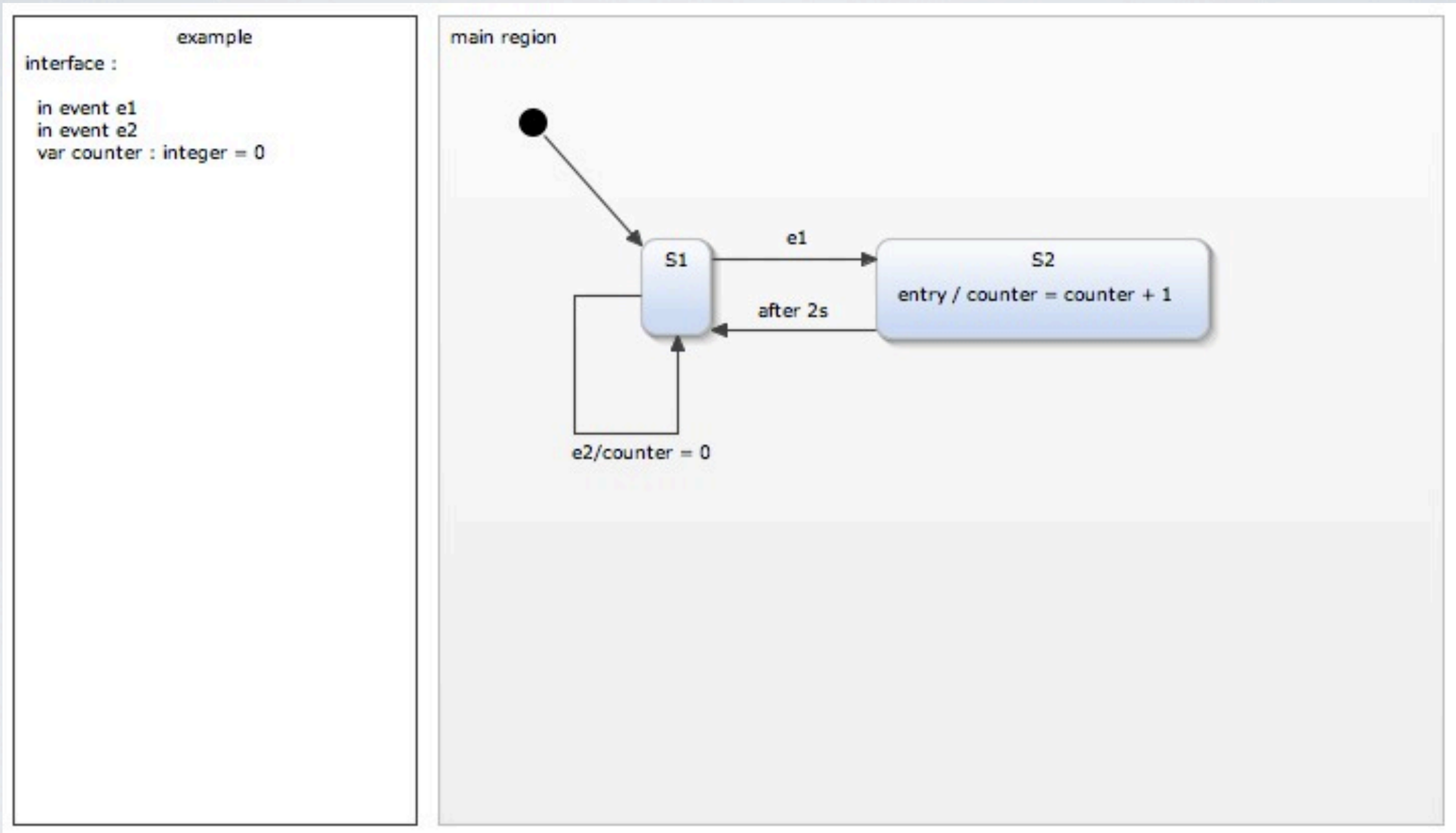
```
GeneratorModel for yakindu::java {

    statechart SomeStatechart {

        feature Outlet {
            targetProject = "SomeStatechartJava"
            targetFolder = "src-gen"
        }
    }
}
```

- Custom generators based on Xpand & Xtend2/Java can be easily integrated

Yakindu SCT - Code Generation

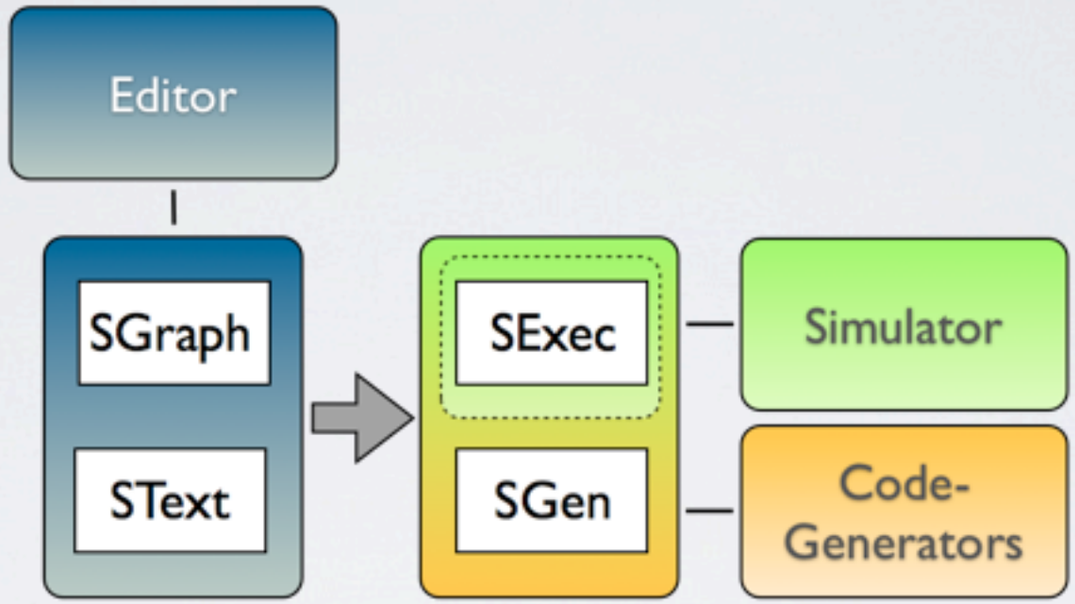


DEMO


example
interface :
in event e1
in event e2
var counter : integer = 0
main region
S1
e1
S2
entry / counter = counter + 1
after 2s
e2/counter = 0

Yakindu SCT - Extensibility

- Recap: different models are used around the Statechart formalism



- **SGraph** (EMF): specification of graphical structures
- **SText** (Xtext): textual specification of declarations & expressions
- **SExec** (EMF): sequentialized statechart execution
- **SGen** (Xtext): code generator parameterization

Built-In Extensibility

- Restriction of structural concepts (SGraph)
- Customization of declarations & expressions (SText)
- Adoption of the execution semantics (SExec)
- Adoption of existing or integration of custom code generators
- Integration of custom type system, augmentation by application types
- Integration of additional validation constraints

The Statechart Application Gap

The Statechart Application Gap

State-based modeling
is useful
in many domains

The Statechart Application Gap

State-based modeling
is useful
in many domains

Typically, statecharts
are independent
of any domain

The Statechart Application Gap

State-based modeling
is useful
in many domains

Typically, statecharts
are independent
of any domain

- How can statecharts be adopted to a specific domain?
- How can tools support this adoption?

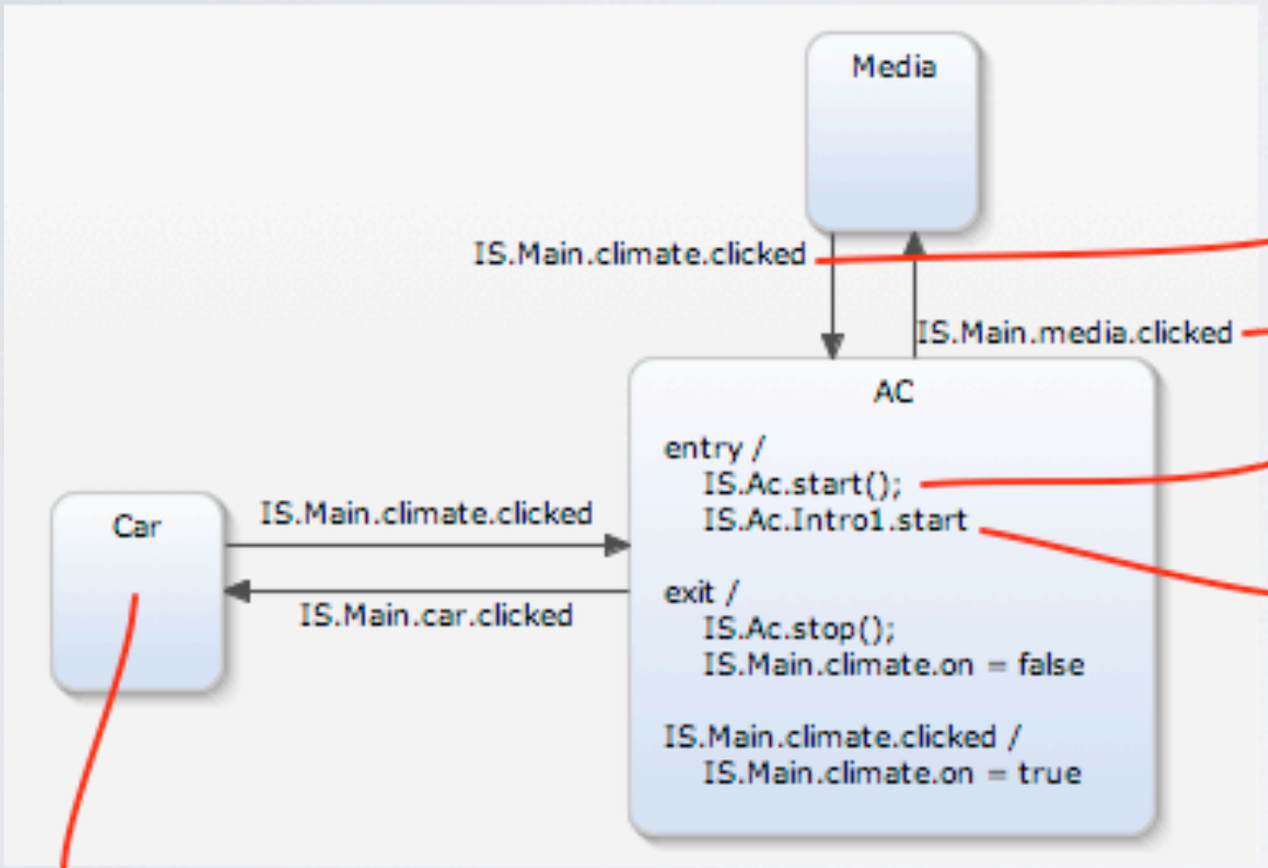
Example: Domain Concepts - HMI

```
app HeadUnit {

    ACScene Ac {
        animation Intro
        animation Outro reverse Intro1

        Slider : tempSetPoint
        Label : temperature
    }

    scene Main {
        Button : climate
        Button : media
        Button : car
        Button : phone
    }

    scene Media { ... }
    scene Car { ... }
    layouted_scene Phone { ... }
}
```

Domain Specific Statecharts

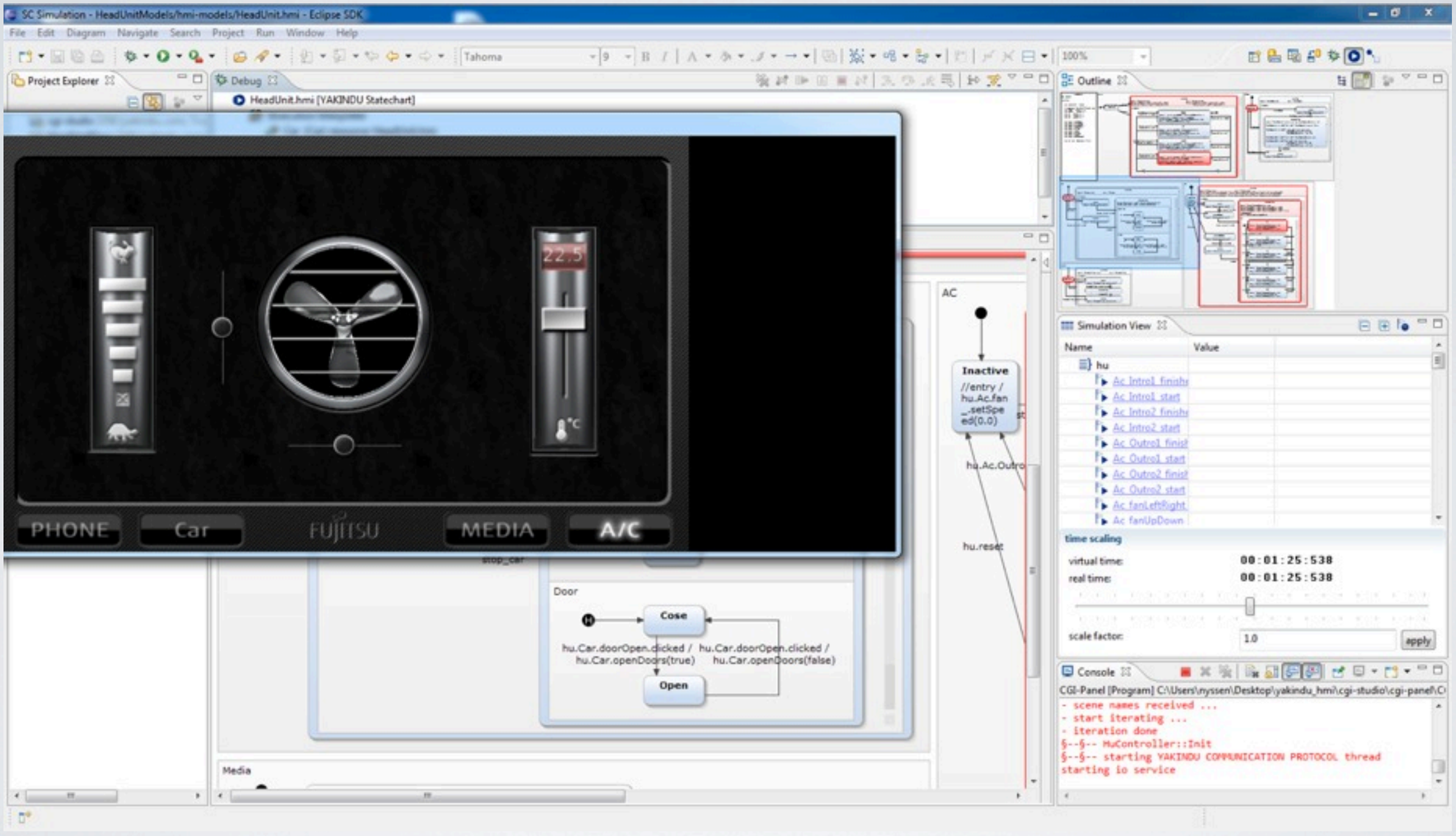
- Improving expressiveness and semantic integration by adopting domain concepts:
    - Refer to domain concepts within declarations (events, variables) and expressions (feature-calls)
    - Concepts from HMI domain: widget (button, label, etc.), scene, popup, animation, button-click, intro, outro,...

Integration of HMI Concepts



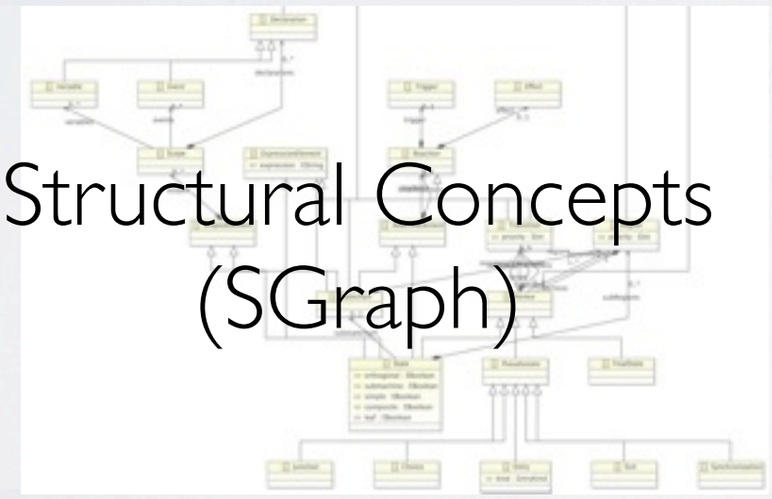

```
app HeadUnit {

    scene Main {
        Button : climate
        Button : media
        Button : car
        Button : phone
    }

    ACScene Ac {
        animation Intro
        animation Outro reverse Intro1

        Slider : tempSetPoint
        Label : temperature
    }

    scene Media { ... }
    scene Car { ... }
    layouted_scene Phone { ... }
}
```

DEMO


SC Simulation - HeadUnitModels/hmi-models/HeadUnit.hmi - Eclipse SDK
File Edit Diagram Navigate Search Project Run Window Help
Project Explorer
Debug
HeadUnit.hmi [YAKINDU Statechart]
Outline
Simulation View
Name
Value
hu
Ac_Intro1_finish
Ac_Intro1_start
Ac_Intro2_finish
Ac_Intro2_start
Ac_Outro1_finish
Ac_Outro1_start
Ac_Outro2_finish
Ac_Outro2_start
Ac_fanLeftRight
Ac_fanUpDown
time scaling
virtual time:
00:01:25:538
real time:
00:01:25:538
scale factor:
1.0
apply
Console
CGI-Panel [Program] C:\Users\nyssen\Desktop\yakindu_hmi\cgi-studio\cgi-panel\C
- scene names received ...
- start iterating ...
- iteration done
§--§-- HuController::Init
§--§-- starting YAKINDU COMMUNICATION PROTOCOL thread
starting io service
AC
Inactive
//entry /
hu.Ac.fan
_.setSpe
ed(0.0)
hu.Ac.Outro
hu.reset
Door
Cose
hu.Car.doorOpen.clicked /
hu.Car.openDoors(true)
hu.Car.doorOpen.clicked /
hu.Car.openDoors(false)
Open
Media
22,5
°C
PHONE
Car
FUJITSU
MEDIA
A/C

Domain-Specific Statechart Approach

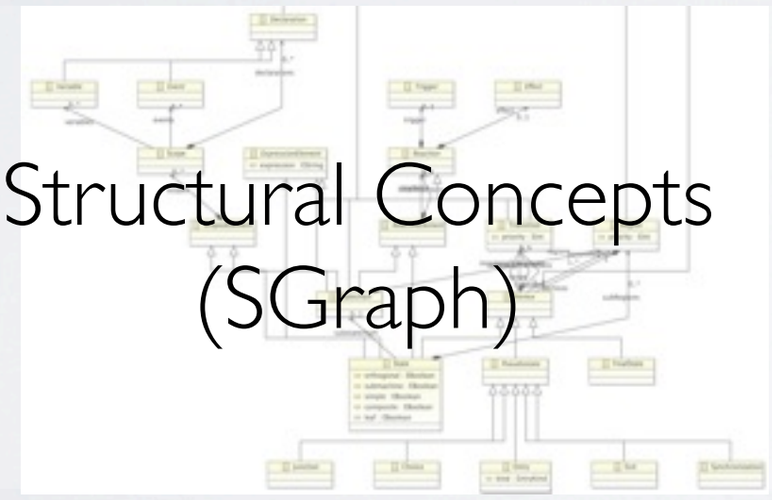
Domain-Specific Statechart Approach



Structural Concepts
(SGraph)

Generic

Domain-Specific Statechart Approach



Structural Concepts
(SGraph)

extends

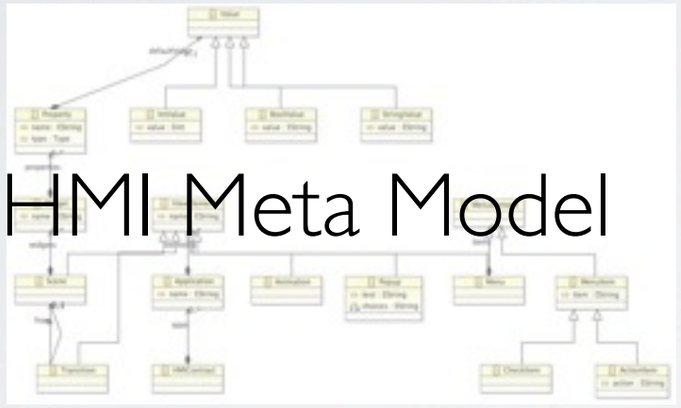
```
grammar org.yakindu.sct.model.stext.SText with org.eclipse.xtext.common.Terminals

/* ---- root rules ----
These root rules are not relevant for the grammar integration

into a single grammar
*/
Root:
        (roots+=DefRoot)*;
DefRoot:
        StatechartRoot | StateRoot | TransitionRoot;
Scope returns sct::Scope:
        (SimpleScope | StatechartScope);
        // a SimpleScope is used for states and regions
SimpleScope returns sct::Scope:
        {SimpleScope} (declarations+=Declaration)*;
        // defines the poosible scopes for statecharts
StatechartScope returns sct::Scope:
        InterfaceScope | InternalScope;
InterfaceScope returns sct::Scope:
```

Declarations &
Expressions
(SText)

Generic

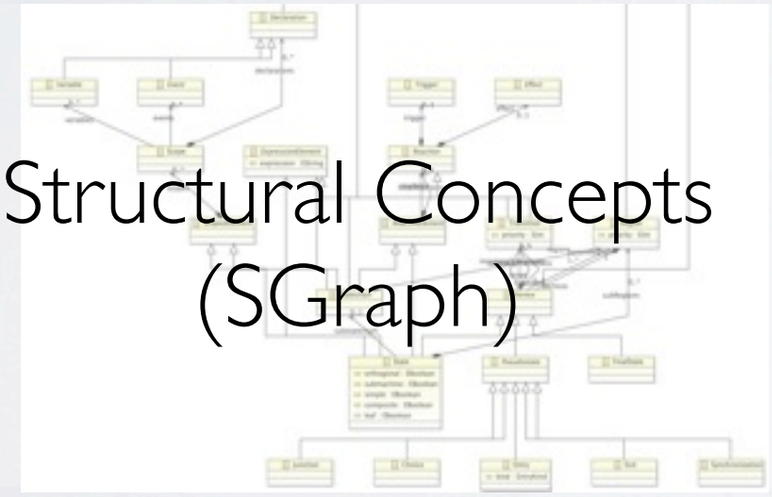
Domain-Specific Statechart Approach

Domain-Specific



HMI Meta Model

Generic



Structural Concepts (SGraph)

extends

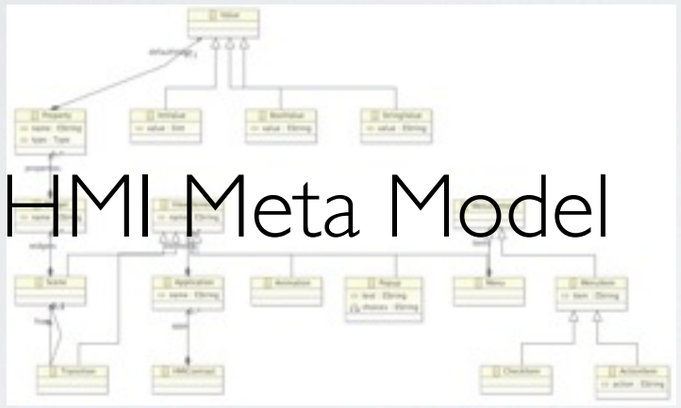
```
grammar org.yakindu.sct.model.stext.SText with org.eclipse.xtext.common.Terminals

/* ---- root rules ----
These root rules are not relevant for the grammar integration

into a single grammar
*/
Root:
        (roots+=DefRoot)*;
DefRoot:
        StatechartRoot | StateRoot | TransitionRoot;
Scope returns sct::Scope:
        (SimpleScope | StatechartScope);
        // a SimpleScope is used for states and regions
SimpleScope returns sct::Scope:
        {SimpleScope} (declarations+=Declaration)*;
        // defines the poosible scopes for statecharts
StatechartScope returns sct::Scope:
        InterfaceScope | InternalScope;
InterfaceScope returns sct::Scope:
```

Declarations & Expressions (SText)

Domain-Specific Statechart Approach

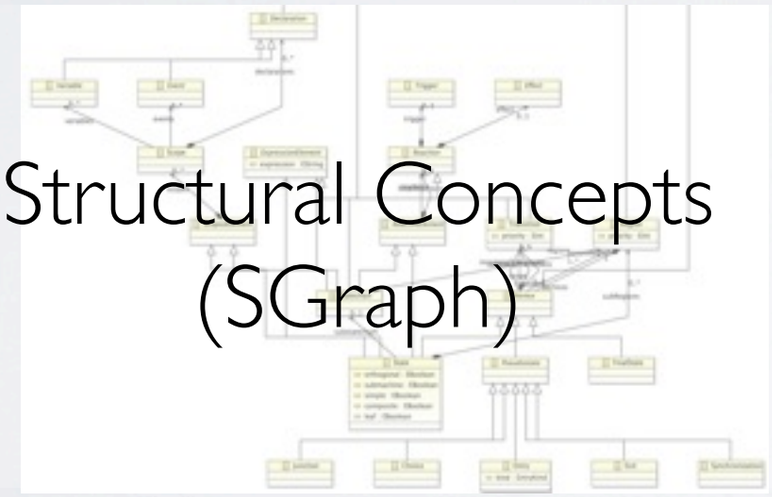
specialization

Domain-Specific



HMI Meta Model

Generic



Structural Concepts (SGraph)

extends

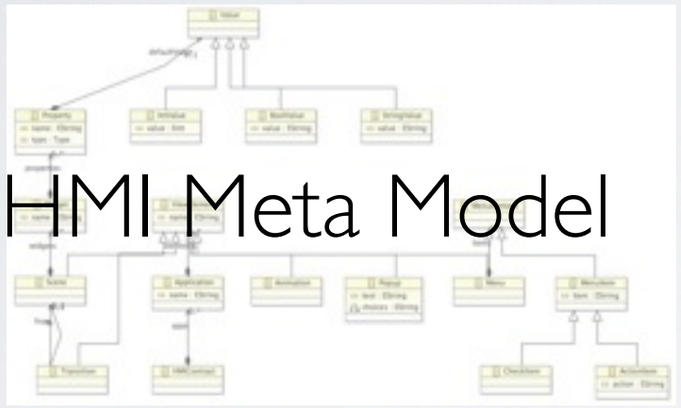
```
grammar org.yakindu.sct.model.stext.SText with org.eclipse.xtext.common.Terminals

/* ---- root rules ----
These root rules are not relevant for the grammar integration

into a single grammar.
*/
Root:
        (roots+=DefRoot)*;
DefRoot:
        StatechartRoot | StateRoot | TransitionRoot;
Scope returns sct::Scope:
        (SimpleScope | StatechartScope);
        // a SimpleScope is used for states and regions
SimpleScope returns sct::Scope:
        {SimpleScope} (declarations+=Declaration)*;
        // defines the poosible scopes for statecharts
StatechartScope returns sct::Scope:
        InterfaceScope | InternalScope;
InterfaceScope returns sct::Scope:
```

Declarations & Expressions (SText)

Domain-Specific Statechart Approach

specialization

Domain-Specific

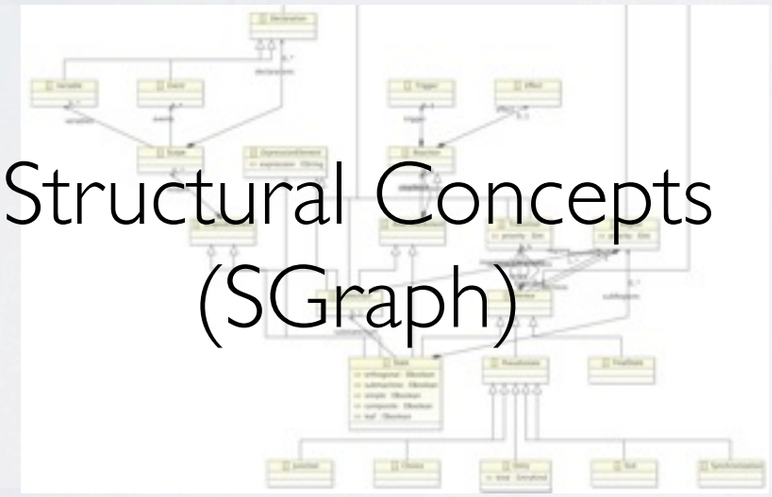


HMI Meta Model

```
grammar com.yakindu.hmi.sctmodel.HMIText with  org.yakindu.sct.model.stext.SText

/* ---- root rules ----
These root rules are not relevant for the
into a single grammar.
*/
Root:
        (roots+=DefRoot)*;

Declaration returns sct::Declaration:
        EventDefinition | VariableDefinition | Clock | Operation
        | LocalReaction | Entrypoint | Exitpoint | HMIDeclaration;

HMIDeclaration:
        HmiScene | HmiPopup | HmiAnimation | HmiTransition;
HmiScene:
        'scene' scene=[contract::Scene|QID];
HmiPopup:
        'popup' popup=[contract::Popup|QID];
HmiAnimation:
        'animation' animation=[contract::Animation|QID];
```

HMI Declarations

Generic



Structural Concepts (SGraph)

extends

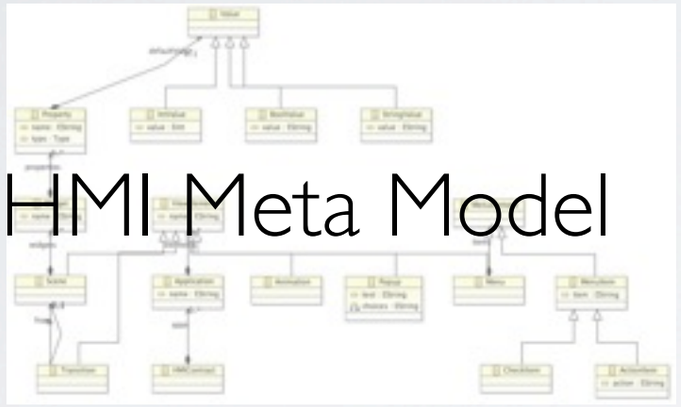
```
grammar org.yakindu.sct.model.stext.SText with org.eclipse.xtext.common.Terminals

/* ---- root rules ----
These root rules are not relevant for the grammar integration

into a single grammar.
*/
Root:
        (roots+=DefRoot)*;
DefRoot:
        StatechartRoot | StateRoot | TransitionRoot;
Scope returns sct::Scope:
        (SimpleScope | StatechartScope);
        // a SimpleScope is used for states and regions
SimpleScope returns sct::Scope:
        {SimpleScope} (declarations+=Declaration)*;
        // defines the poosible scopes for statecharts
StatechartScope returns sct::Scope:
        InterfaceScope | InternalScope;
InterfaceScope returns sct::Scope:
```

Declarations & Expressions (SText)

Domain-Specific Statechart Approach

specialization

Domain-Specific



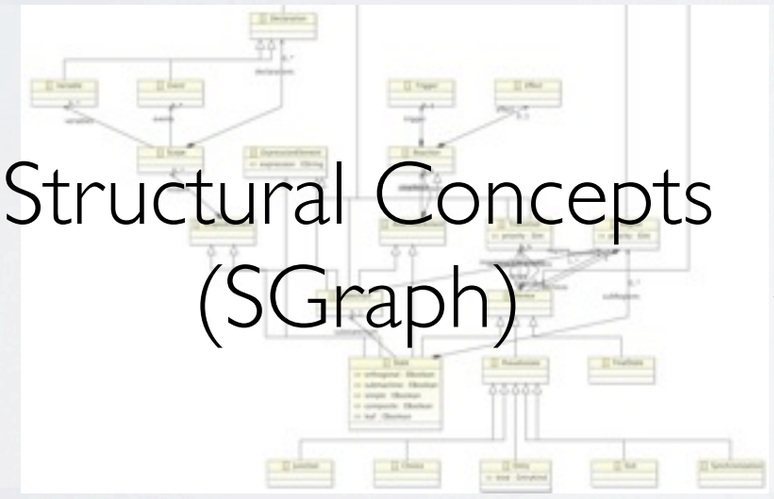
HMI Meta Model

references

```
grammar com.yakindu.hmi.sctmodel.HMIText with  org.yakindu.sct.model.stext.SText

/* ---- root rules ----
These root rules are not relevant for the
into a single grammar.
*/
Root:
        (roots+=DefRoot)*;

Declaration returns sct::Declaration:
        EventDefinition | VariableDefinition | Clock | Operation
        | LocalReaction | Entrypoint | Exitpoint | HMIDeclaration;

HMIDeclaration:
        HmiScene | HmiPopup | HmiAnimation | HmiTransition;
HmiScene:
        'scene' scene=[contract::Scene|QID];
HmiPopup:
        'popup' popup=[contract::Popup|QID];
HmiAnimation:
        'animation' animation          ::Animation|QID];
```

HMI Declarations

extends



Structural Concepts (SGraph)

extends

```
grammar org.yakindu.sct.model.stext.SText with org.eclipse.xtext.common.Terminals

/* ---- root rules ----
These root rules are not relevant for the grammar integration

into a single grammar.
*/
Root:
        (roots+=DefRoot)*;
DefRoot:
        StatechartRoot | StateRoot | TransitionRoot;
Scope returns sct::Scope:
        (SimpleScope | StatechartScope);
        // a SimpleScope is used for states and regions
SimpleScope returns sct::Scope:
        {SimpleScope} (declarations+=Declaration)*;
        // defines the poosible scopes for statecharts
StatechartScope returns sct::Scope:
        InterfaceScope | InternalScope;
InterfaceScope returns sct::Scope:
```

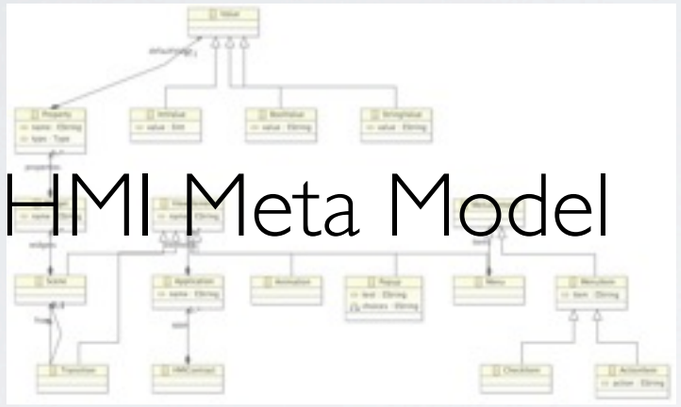
Declarations & Expressions (SText)

Generic

Domain-Specific Statechart Approach

specialization

Domain-Specific



HMI Meta Model

references

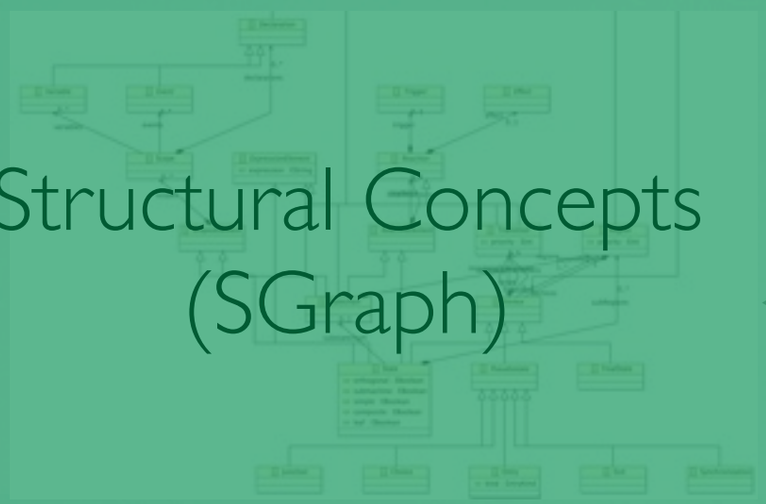
```
grammar com.yakindu.hmi.sctmodel.HMIText with  org.yakindu.sct.model.stext.SText

/* ---- root rules ----
These root rules are not relevant for the
into a single grammar.
*/
Root:
        (roots+=DefRoot)*;

Declaration returns sct::Declaration:
        EventDefinition | VariableDefinition | ClockI Operation
        | LocalReaction | Entrypoint | Exitpoint | HMIDeclaration;

HMIDeclaration:
        HmiScene | HmiPopup | HmiAnimation | HmiTransition;
HmiScene:
        'scene' scene=[contract::Scene|QID];
HmiPopup:
        'popup' popup=[contract::Popup|QID];
HmiAnimation:
        'animation' animation         ::Animation|QID];
```

HMI Declarations

Domain Specific Statechart

extends

Generic



Structural Concepts (SGraph)

```
grammar org.yakindu.sct.model.stext.SText with org.eclipse.xtext.common.Terminals

/* ---- root rules ----
These root rules are not relevant for the grammar integration

into a single grammar.
*/
Root:
        (roots+=DefRoot)*;
DefRoot:
        StatechartRoot | StateRoot | TransitionRoot;
Scope returns sct::Scope:
        (SimpleScope | StatechartScope);
        // a SimpleScope is used for states and regions
SimpleScope returns sct::Scope:
        {SimpleScope} (declarations+=Declaration)*;
        // defines the poosible scopes for statecharts
StatechartScope returns sct::Scope:
        InterfaceScope | InternalScope;
InterfaceScope returns sct::Scope:
```

Declarations & Expressions (SText)

Yakindu SCT

- Open Source / EPL
- Hosted at EclipseLabs
- Eclipse-Proposal planned for 2012
  - Interested parties welcome!

- Important Links:
  - Project Site: http://yakindu.org
  - Eclipse Labs Site: http://code.google.com/a/eclipselabs.org/p/yakindu/
  - Update Site: http://updates.yakindu.com/indigo/milestones/

Thank You! Questions?