Yakindu Tools & Domain-Specific Statecharts

Axel Terfloth
itemis AG

YAKINDU

is a ***modular toolkit***

for ***model-based development***

of embedded systems

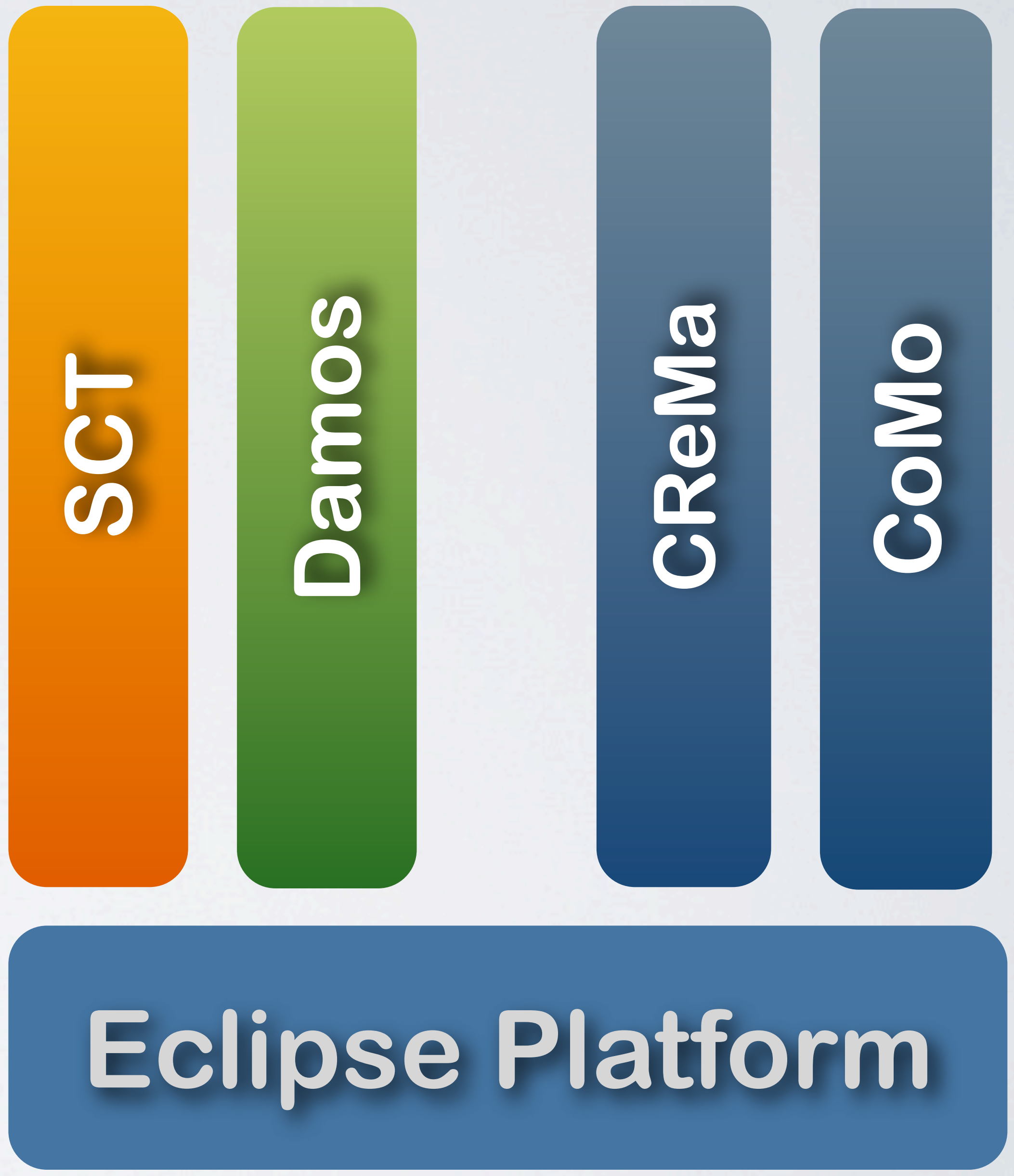
YAKINDU Modules

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

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

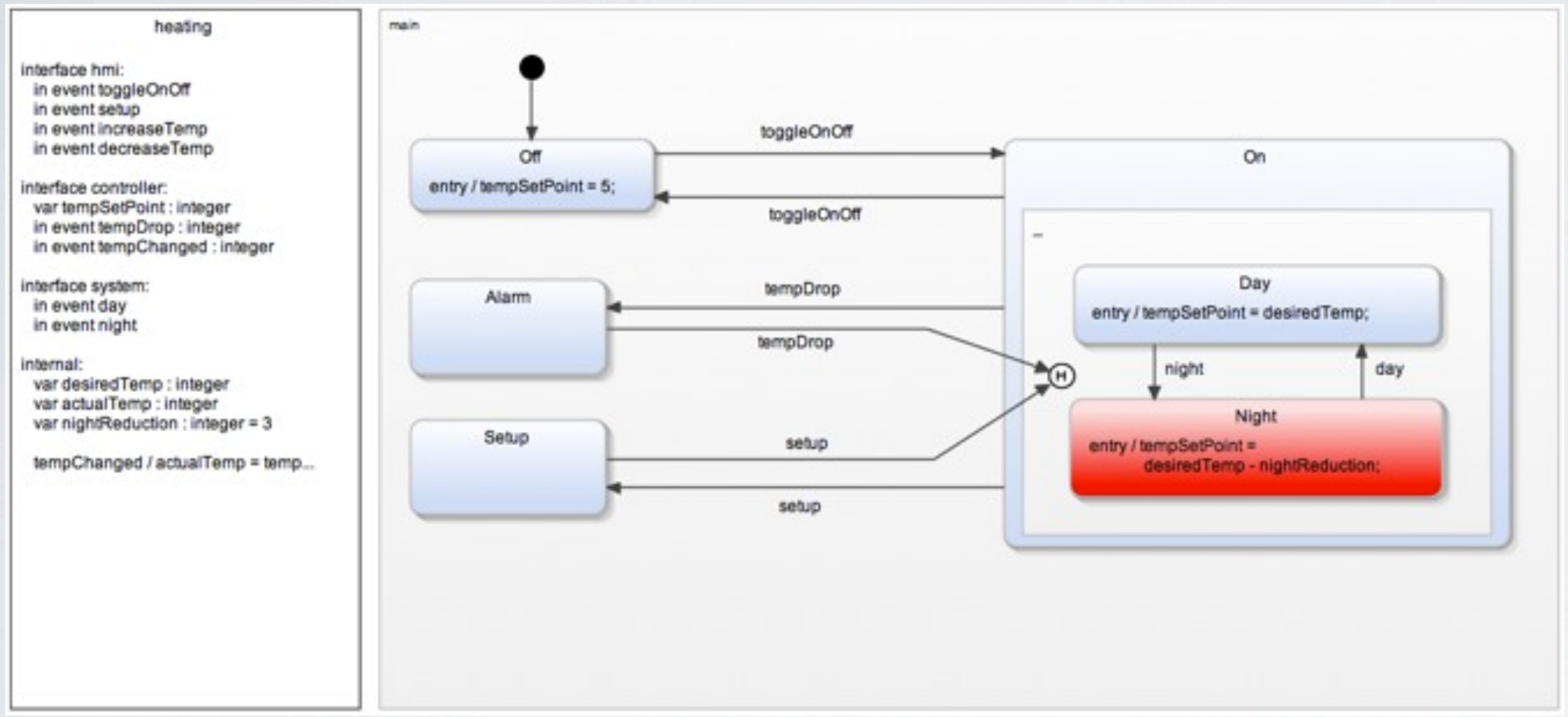
➡ **Reuse of**

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

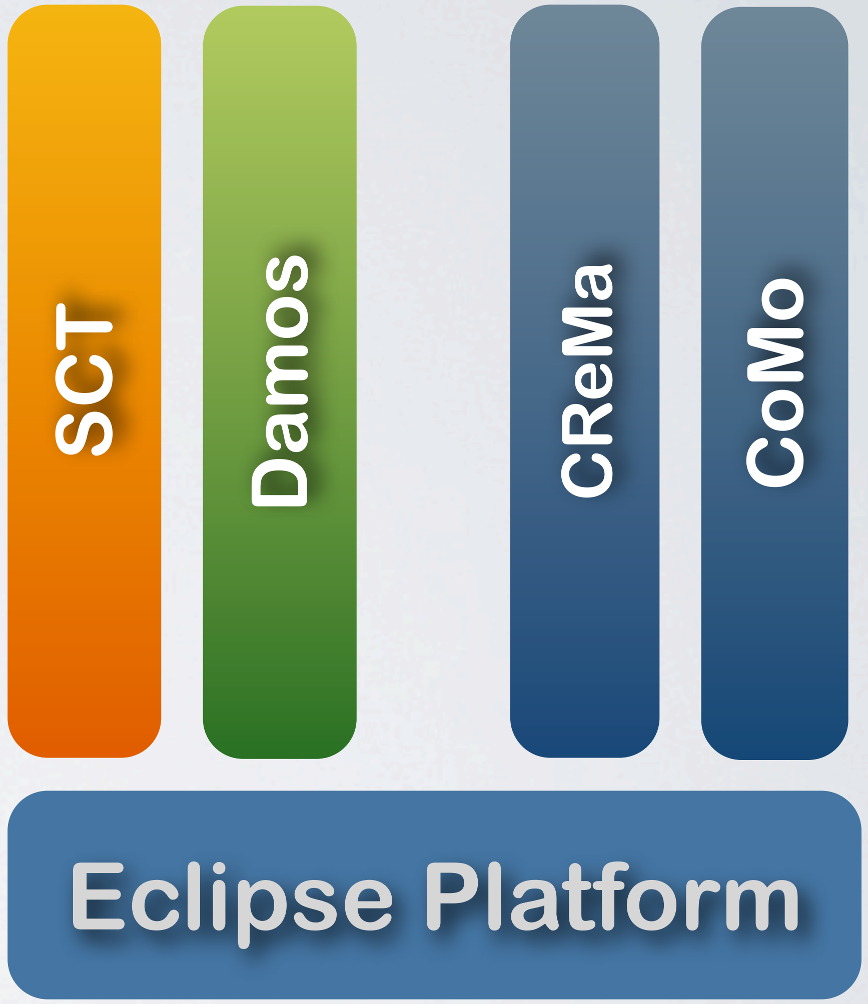


YAKINDU SCT

Statechart Tools

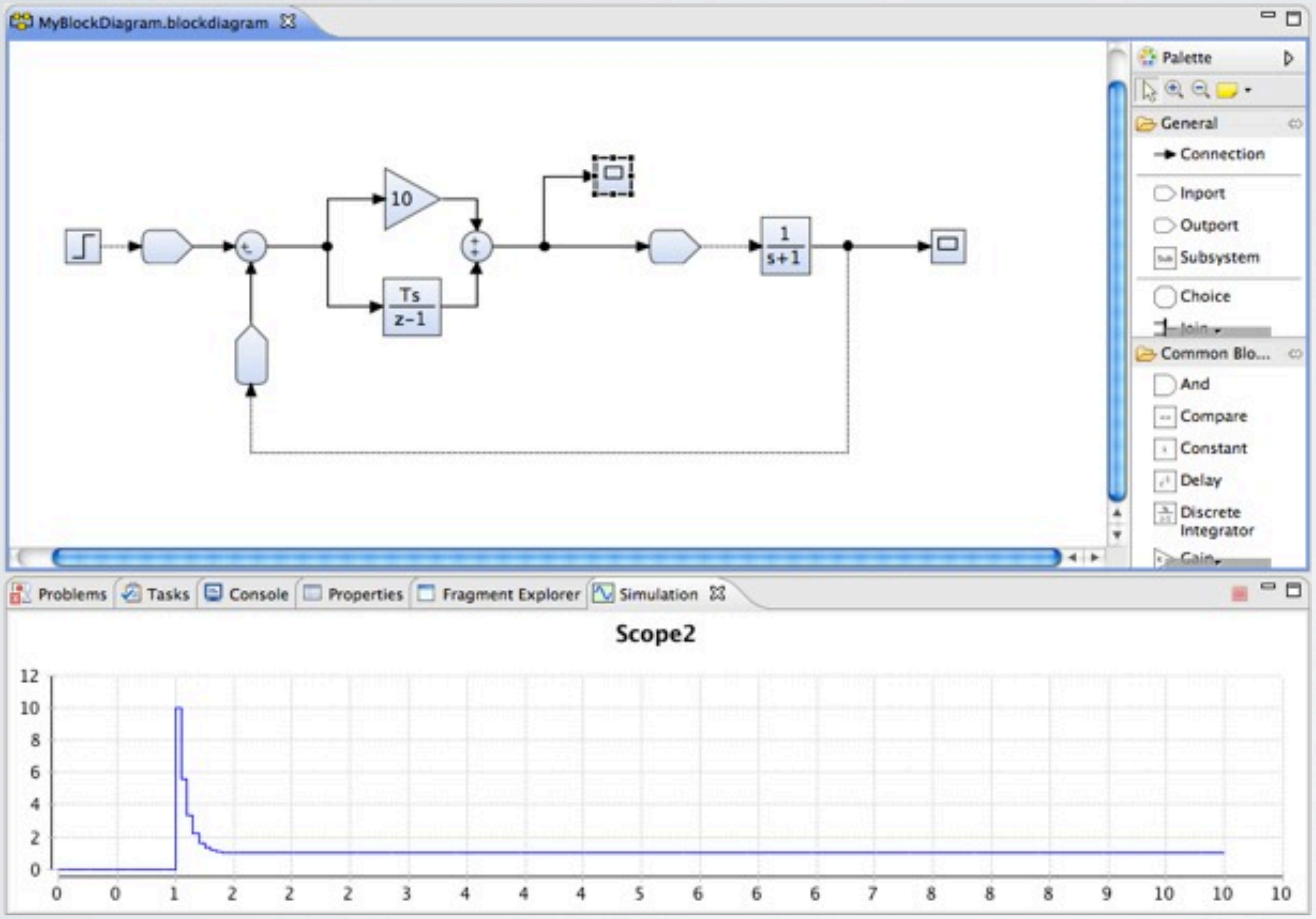


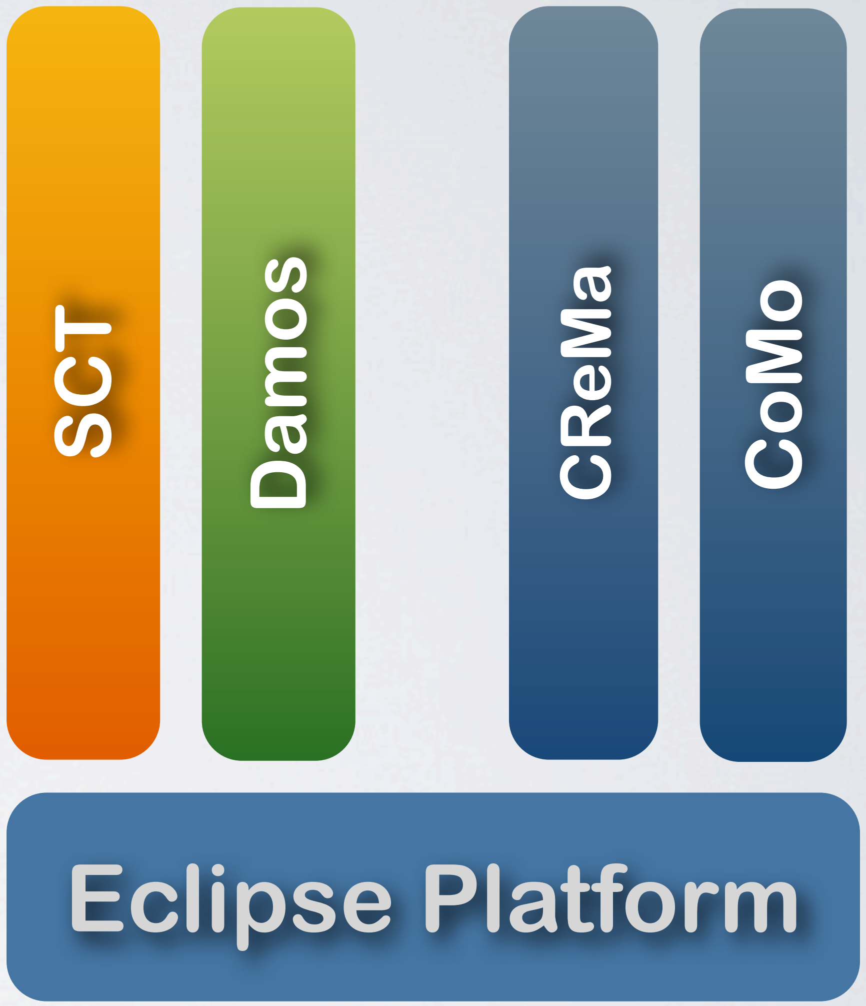
```
heating

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...
```



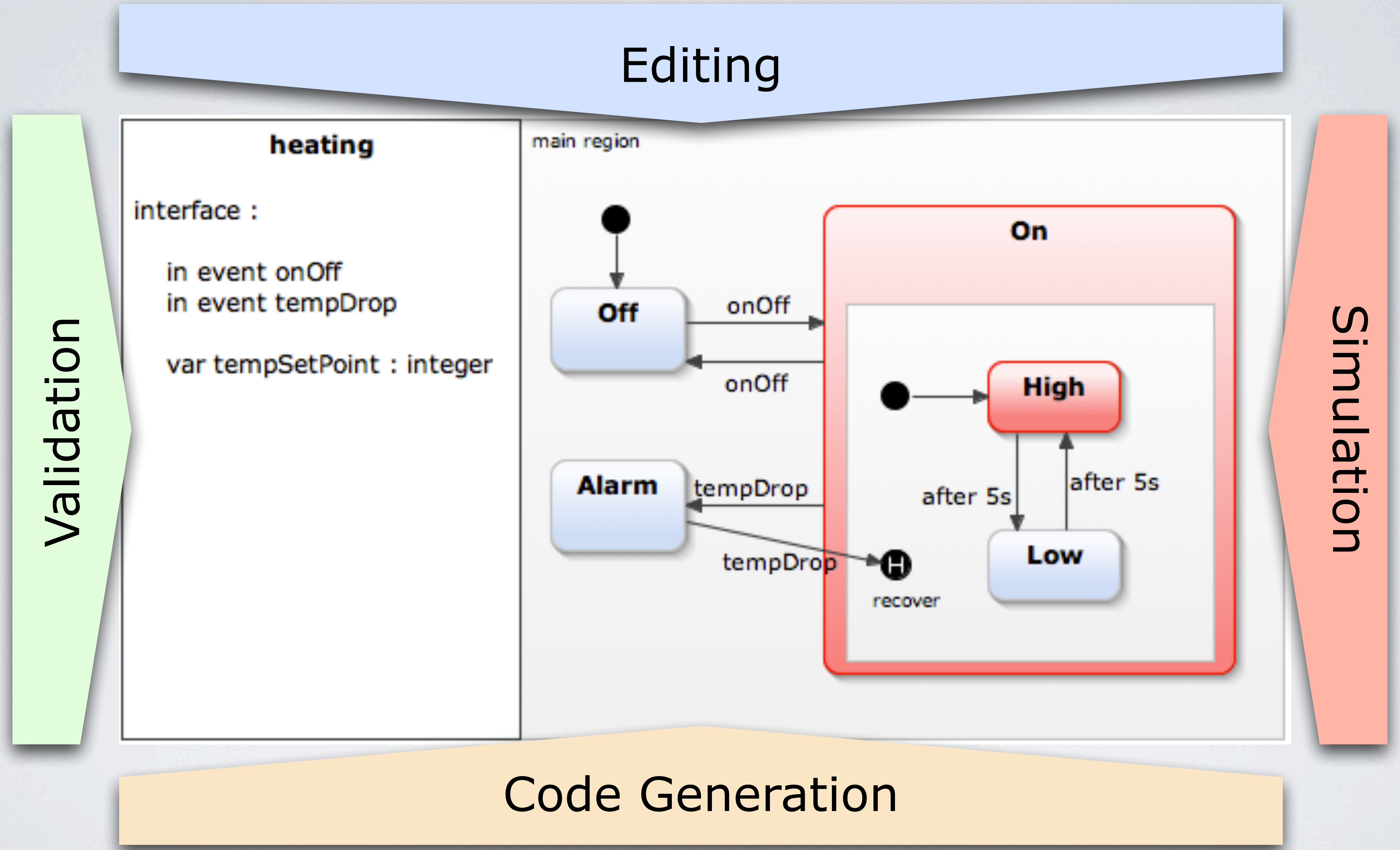
YAKINDU Damos

Dataflow-oriented Modeling





YAKINDU Statechart Tools (SCT)



```
interface :

    in event onOff
    in event tempDrop

    var tempSetPoint : integer
```

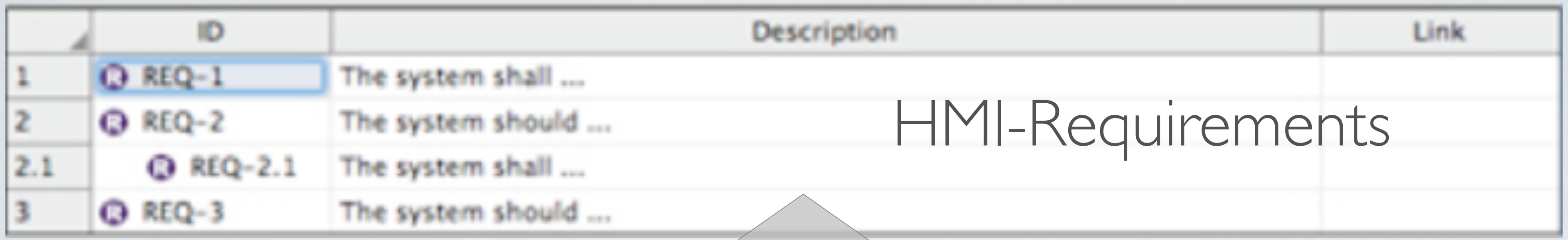
The Statechart Application Gap

State-based modeling
is useful
in many domains

Typically, statecharts
are independent
of any domain

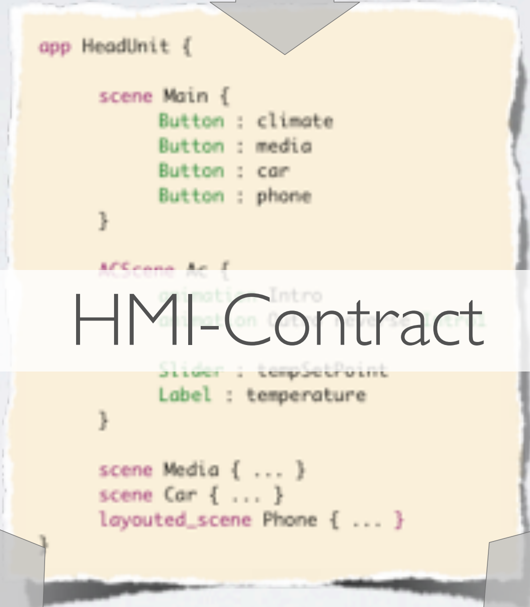
- How can statecharts be adopted to different domains?
- How can tools support this adoption?

HMI / UI Statecharts



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

HMI-Requirements

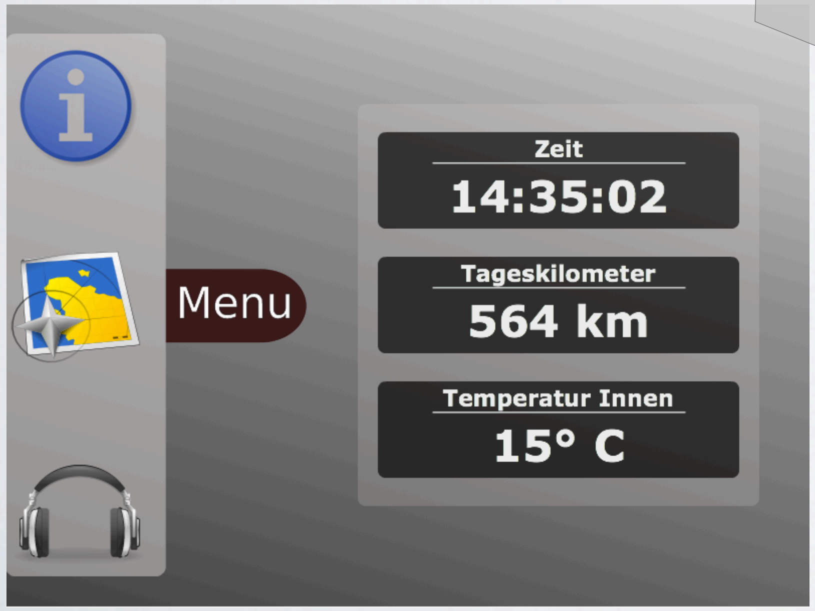


```
app HeadUnit {

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

    ACScene Ac {
        animation Intro
        animation Outro reverse Intro

        Slider : tempSetPoint
        Label : temperature
    }

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

HMI-Contract

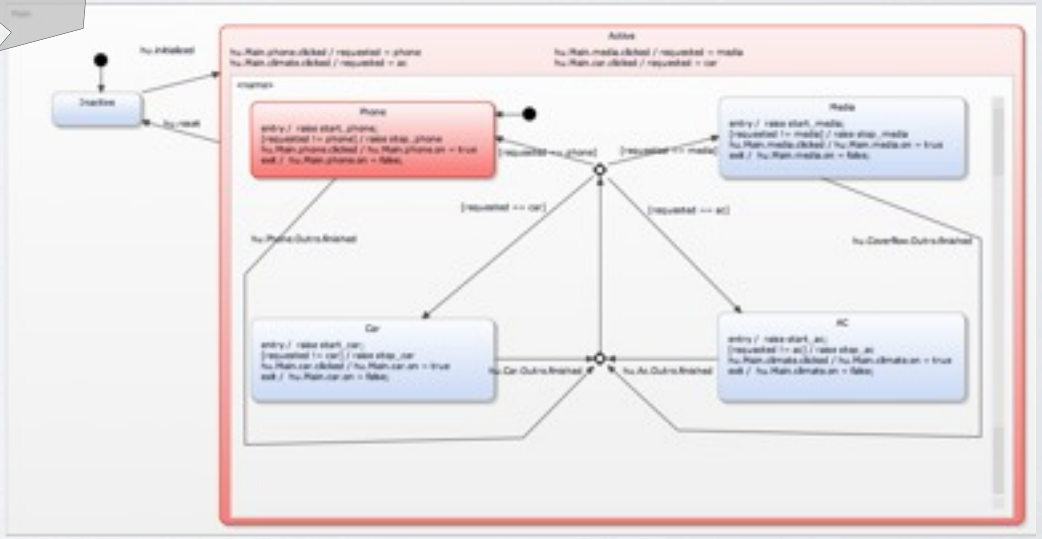
Visualization

(CGI-Studio)

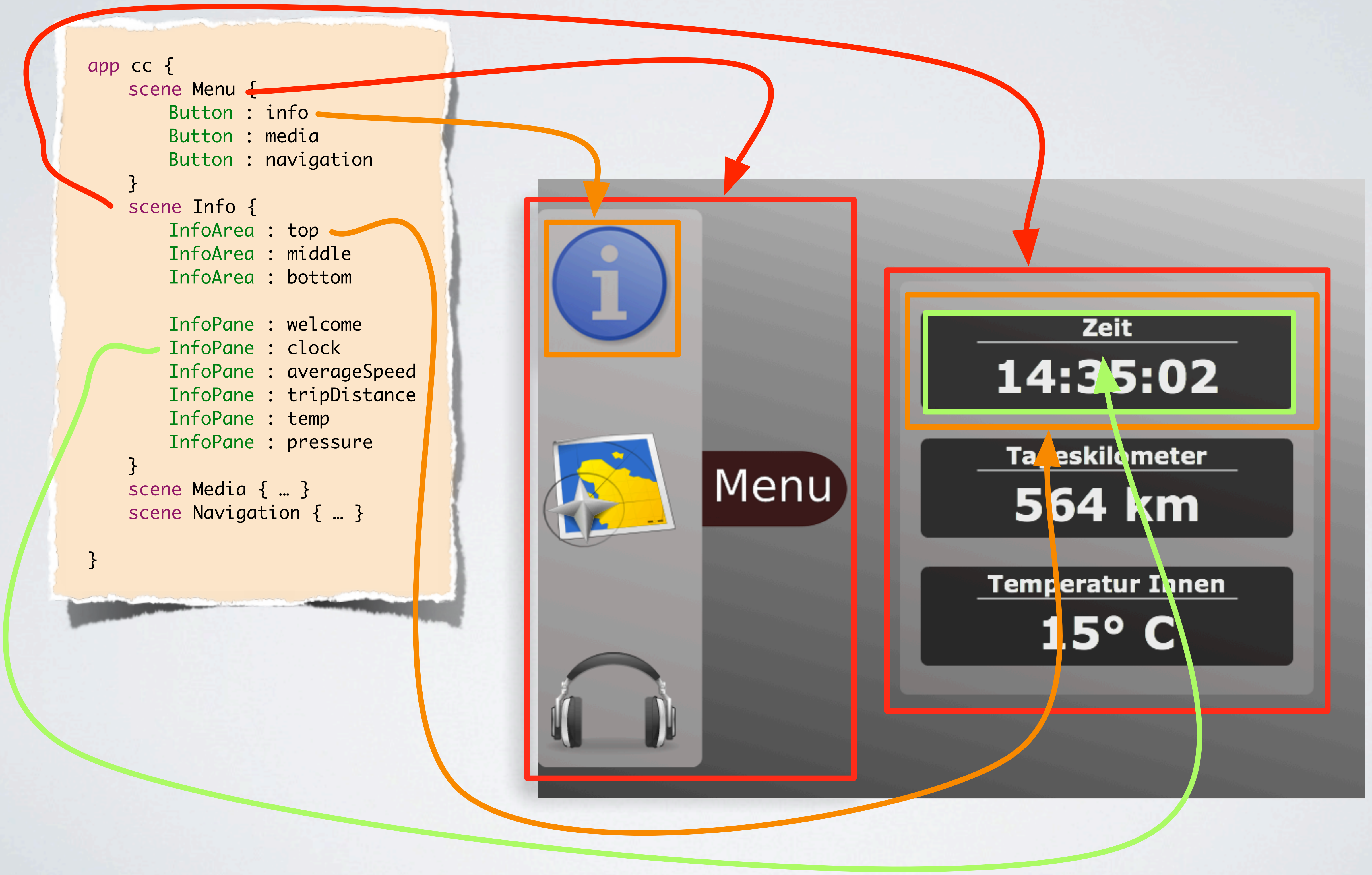


Behavior

HMI-Statecharts



Example: Domain Concepts - HMI



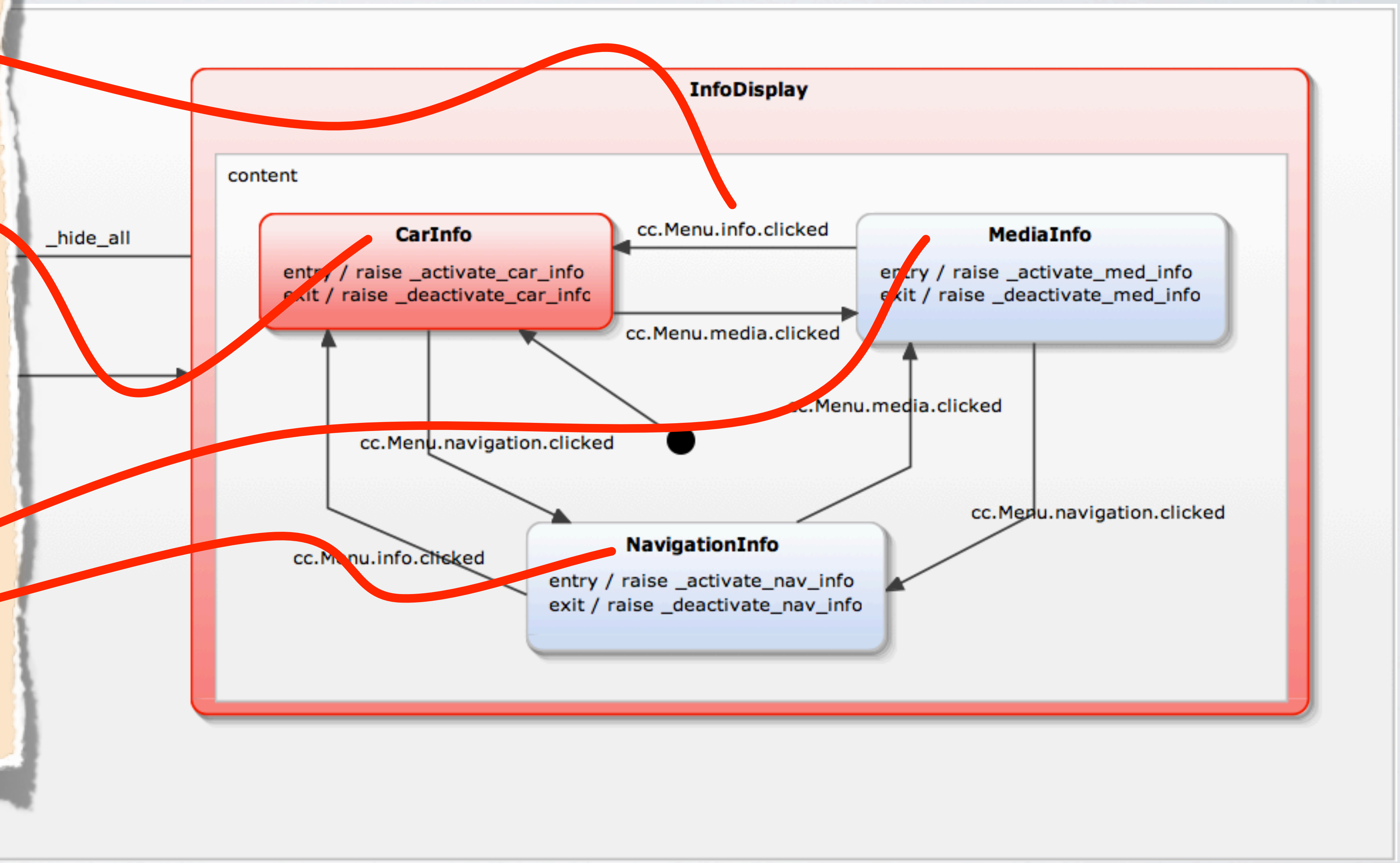
```
app cc {
    scene Menu {
        Button : info
        Button : media
        Button : navigation
    }
    scene Info {
        InfoArea : top
        InfoArea : middle
        InfoArea : bottom

        InfoPane : welcome
        InfoPane : clock
        InfoPane : averageSpeed
        InfoPane : tripDistance
        InfoPane : temp
        InfoPane : pressure
    }
    scene Media { … }
    scene Navigation { … }

}
```

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,...

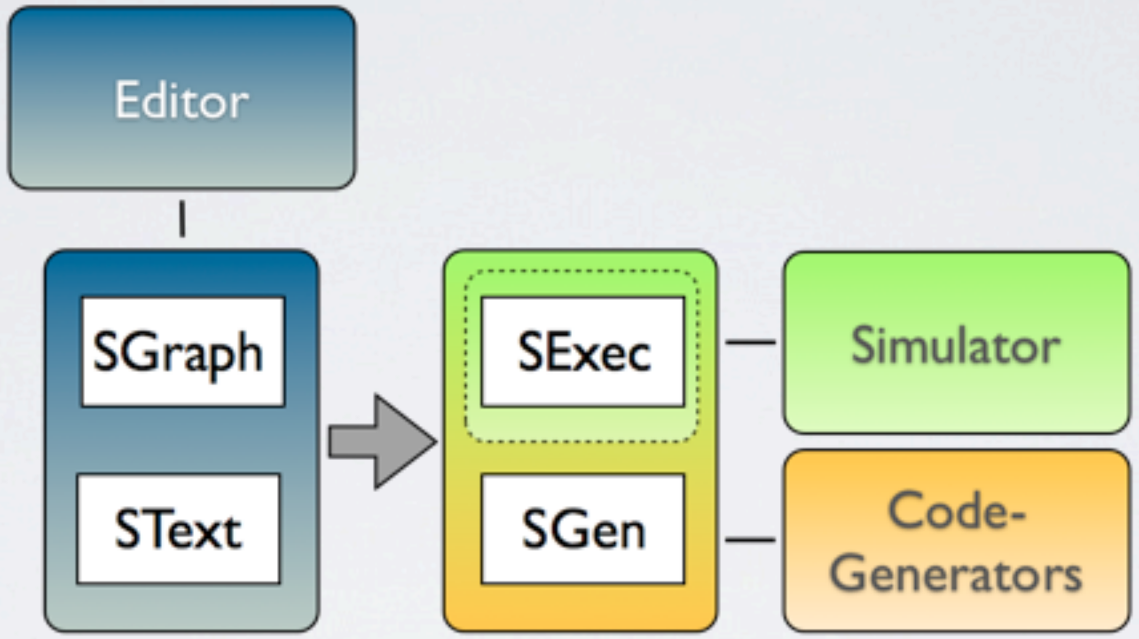
Integration of HMI Concepts

```
app cc {
    scene Menu {
        Button : info
        Button : media
        Button : navigation
    }
    scene Info {
        InfoArea : top
        InfoArea : middle
        InfoArea : bottom

        InfoPane : welcome
        InfoPane : clock
        InfoPane : averageSpeed
        InfoPane : tripDistance
        InfoPane : temp
        InfoPane : pressure
    }
    scene Media { … }
    scene Navigation { … }

}
```


InfoDisplay
content
_hide_all
CarInfo
entry / raise _activate_car_info
exit / raise _deactivate_car_info
MediaInfo
entry / raise _activate_med_info
exit / raise _deactivate_med_info
NavigationInfo
entry / raise _activate_nav_info
exit / raise _deactivate_nav_info
cc.Menu.info.clicked
cc.Menu.media.clicked
cc.Menu.media.clicked
cc.Menu.navigation.clicked
cc.Menu.navigation.clicked
cc.Menu.info.clicked

Yakindu SCT - Extensibility

- 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

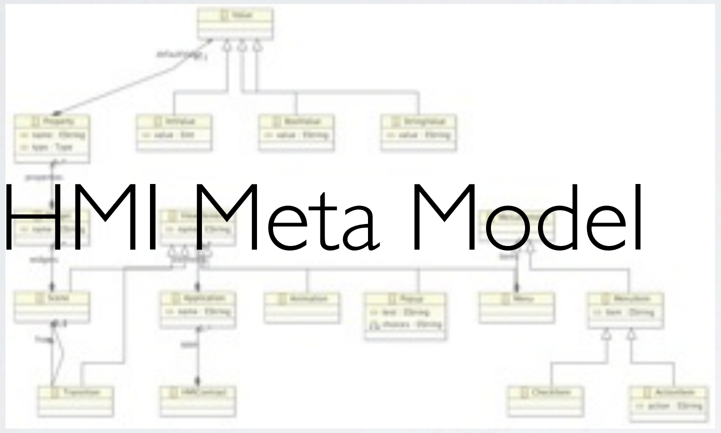
- Restriction of structural concepts (SGraph)
- Customization of declarations & expressions (SText)
- Adoption of the execution semantics (SExec)
- Adoption of existing or integration of custom code generators
- Integration of custom type system, augmentation by application types
- Integration of additional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

YAKINDU SCT Approach

specialization

Domain-Specific

HMI Meta Model



references

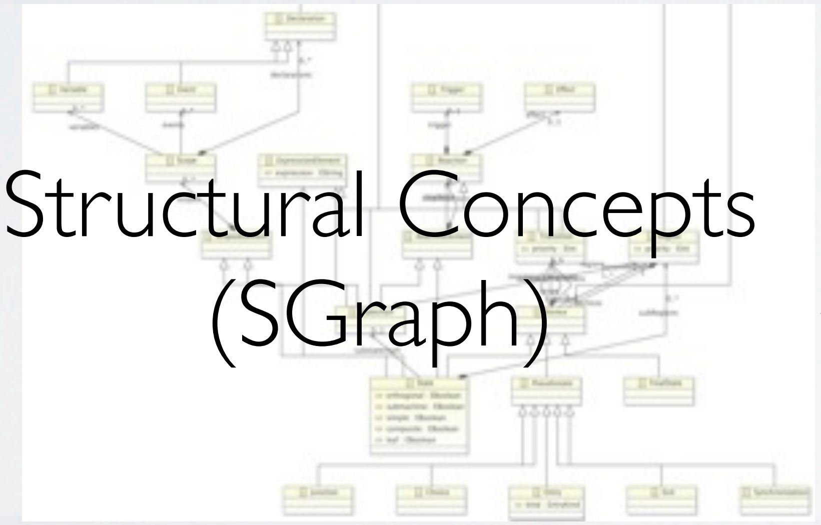
HMI Declarations

```
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[illegible]::Animation|QID];
```

extends

Generic

Structural Concepts (SGraph)



extends

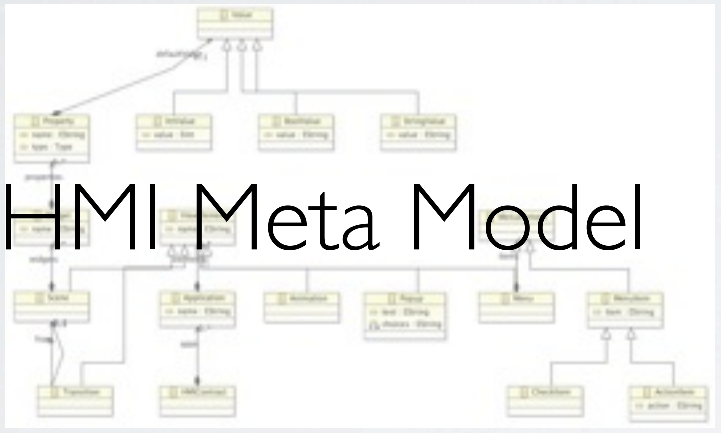
Declarations & Expressions (SText)

```
grammar org.yakindu.sct.model.stext.[illegible]ext 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:
```

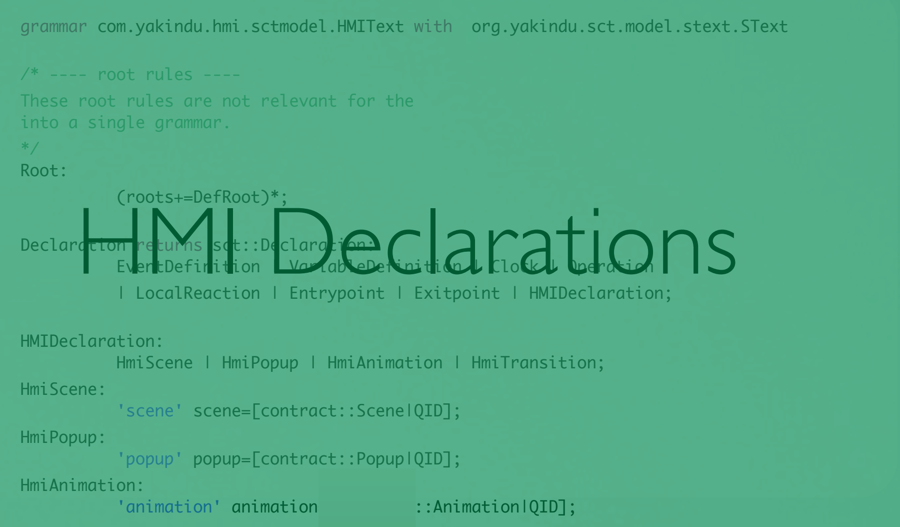
YAKINDU SCT Approach

specialization

Domain-Specific



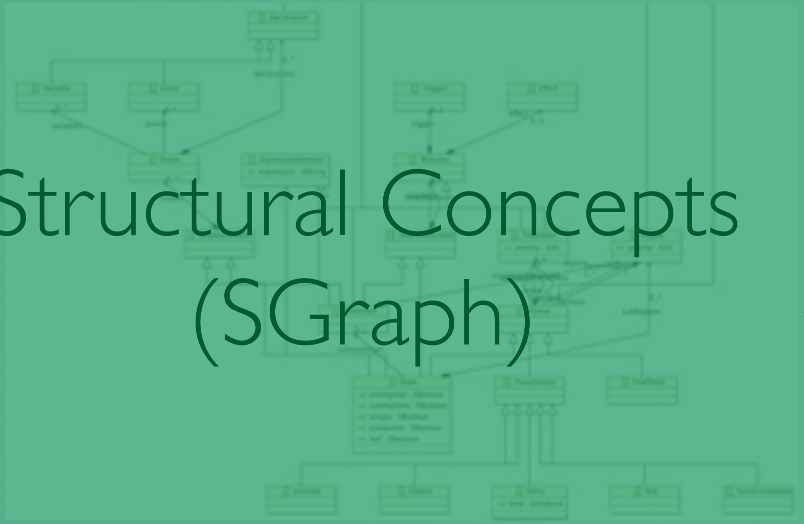
references



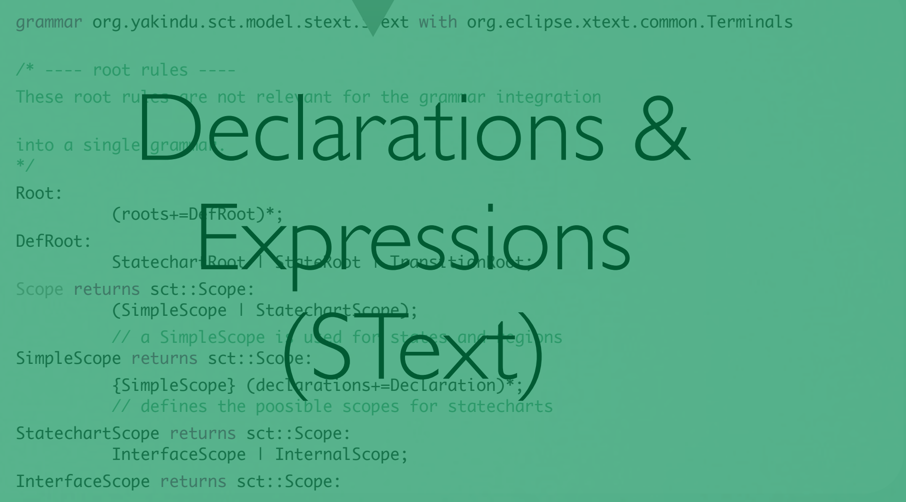
```
grammar com.yakindu.hmi.sctmodel.HMIText with  org.yakindu.sct.model.stext.SText

/* ---- 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];
```

Domain Specific Statechart

extends



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:
```

Generic

Yakindu SCT

- Open Source / EPL
- Hosted at EclipseLabs
- SCT Eclipse-Proposals planned for 2012
    - Damos already sumitted
    - 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?