YAKINDU
Statechart Tools (SCT)

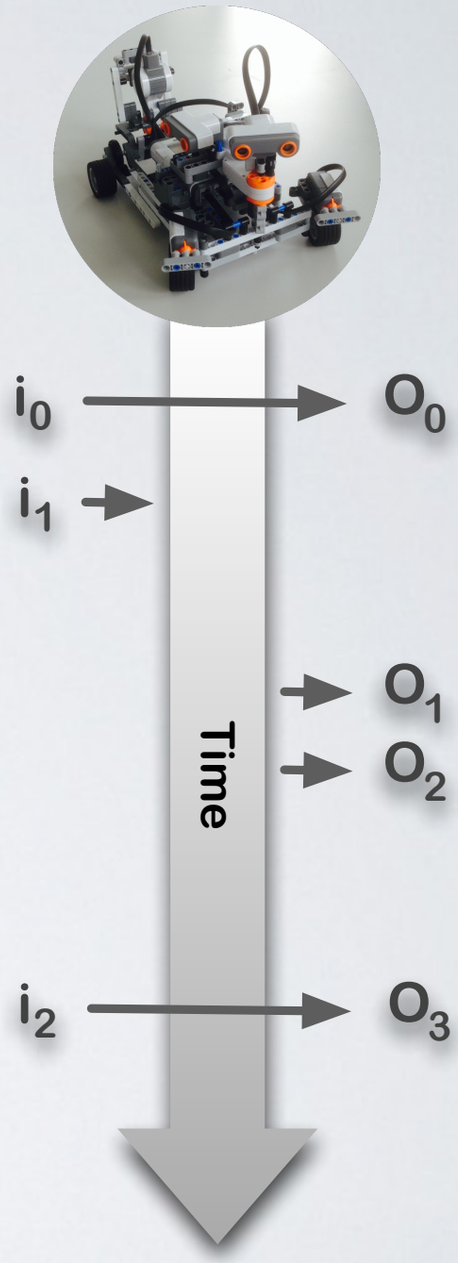
statecharts made easy

Eclipse Demo Camp Mars - Zürich

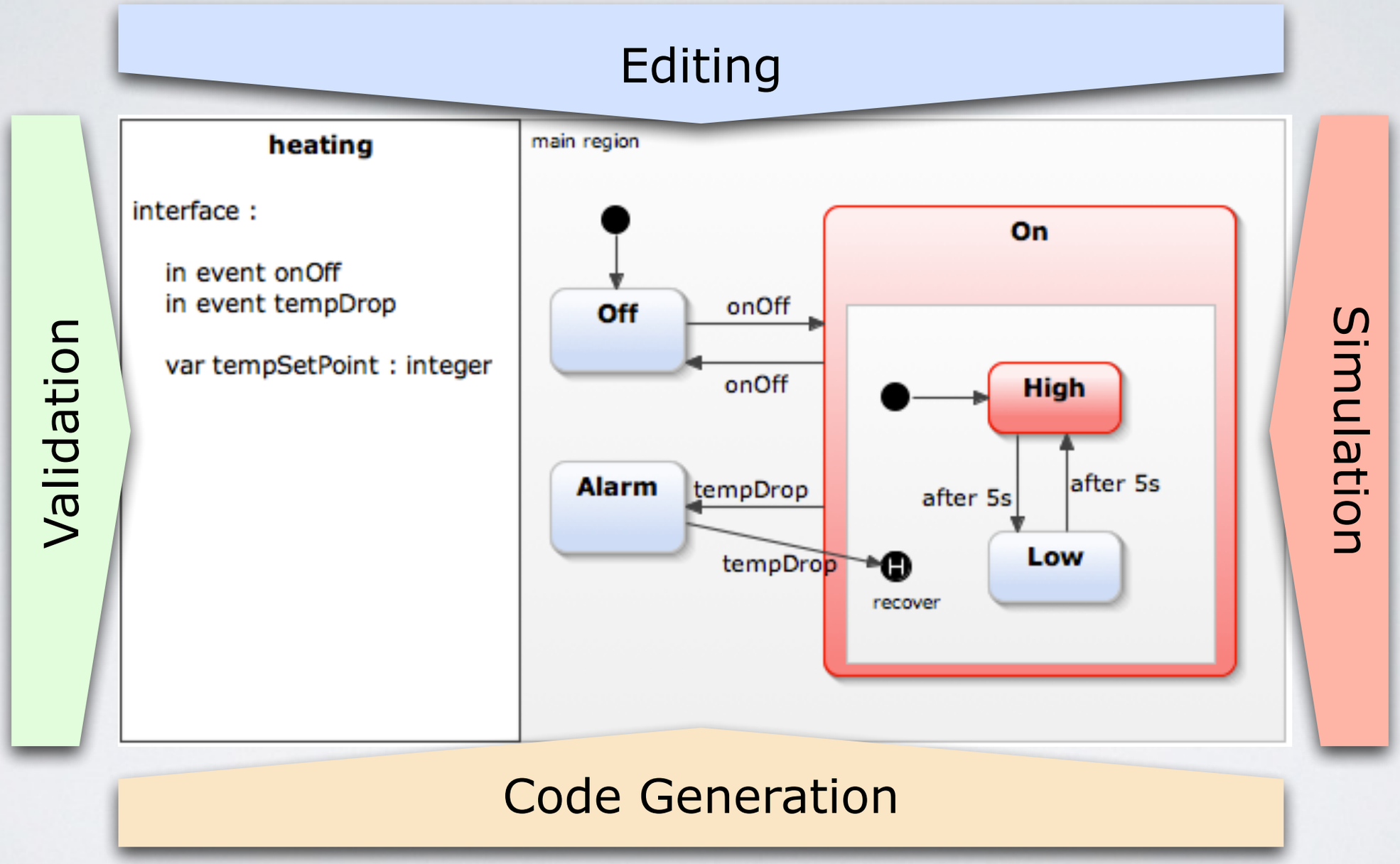
Axel Terfloth, itemis AG
terfloth@itemis.de

state machines (aka state charts)

- model reactive systems
- continuously interacts with the environment
- event driven > typical for technical/embedded systems
- 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
- state chart is a specific state machine formalism defined by David Harel
- state charts have a graphical notation

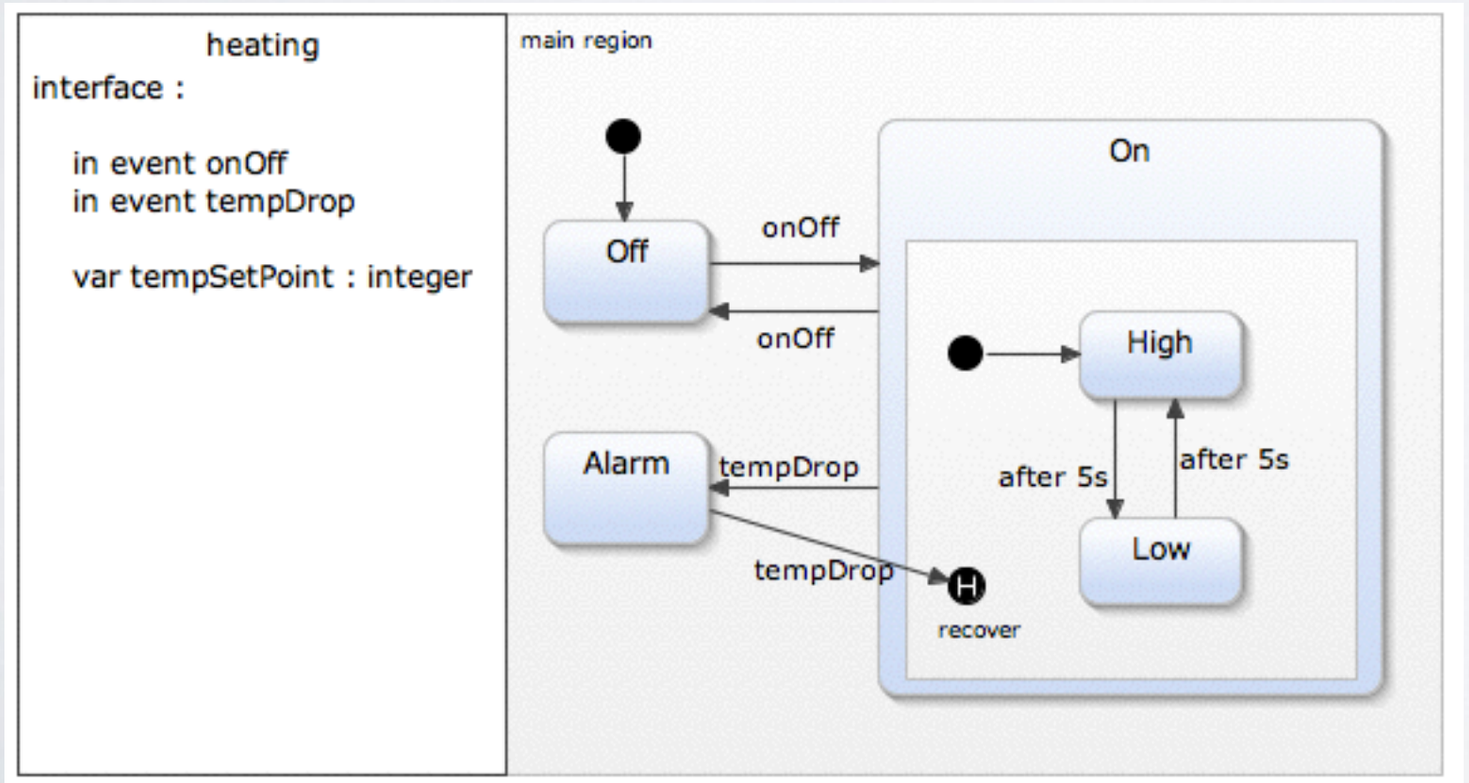


Yakindu *State**Chart**Tools*



YAKINDU Statechart

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



- time control is delegated to the environment
- model interpreter and different flavors of generated code follow the same core semantics

Demo...

Code Generation

Yakindu SCT - Code Generation

- Yakindu comprises code generators for C, C++, Java
- Plain code approach by default
- 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

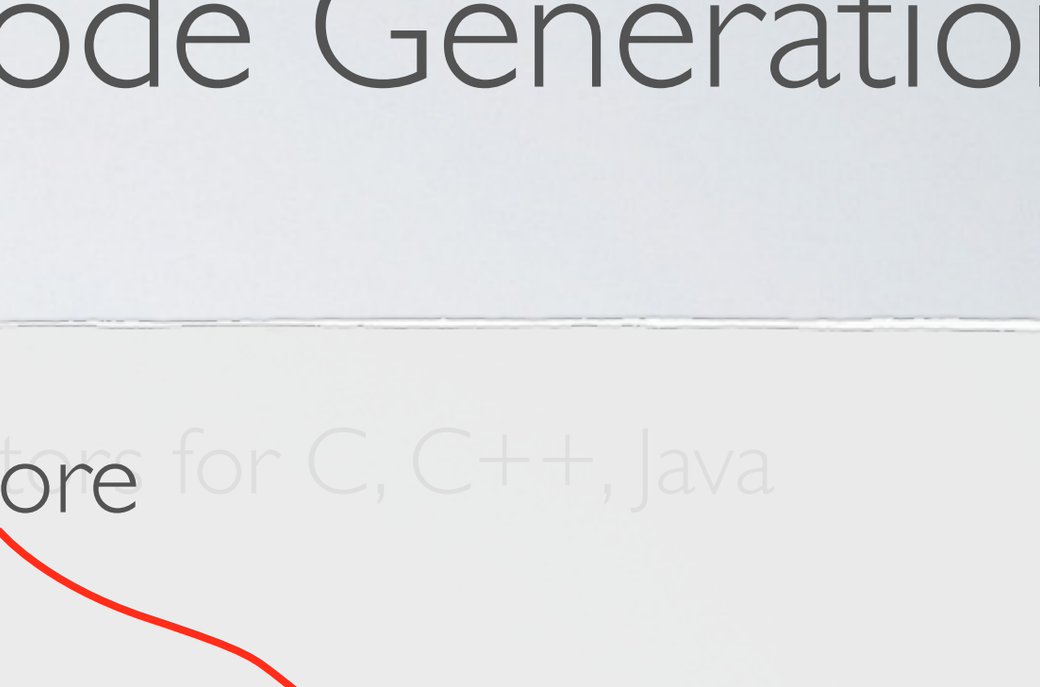
Yakindu SCT - Code Generation

- Ya
- Pl
- Al
- Cu
  ea

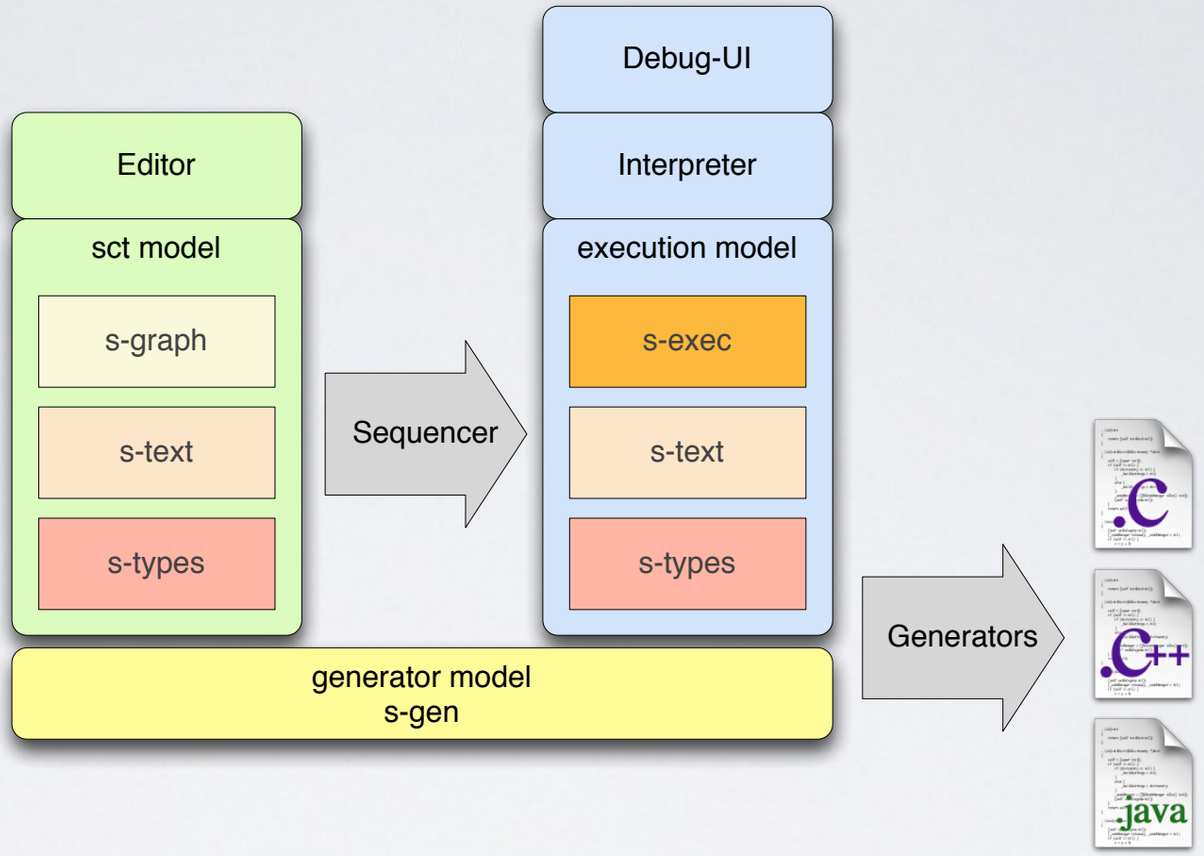
platform agnostic core

Framework RTOS

... platform specific extensions



Yakindu Languages Modules within Generators



- SCT generators take execution model and translate it into code
- all generators conform to the same execution semantic
- generator model can capture specific configurations

Extending & Customizing Yakindu SCT

Yakindu SCT

- is a generic statechart MDD toolkit
  - editing
  - validation
  - simulation
  - code generation
- typically used ‚off the shelf' / as it is
  - self contained
  - no dependencies to further modeling concepts

… but

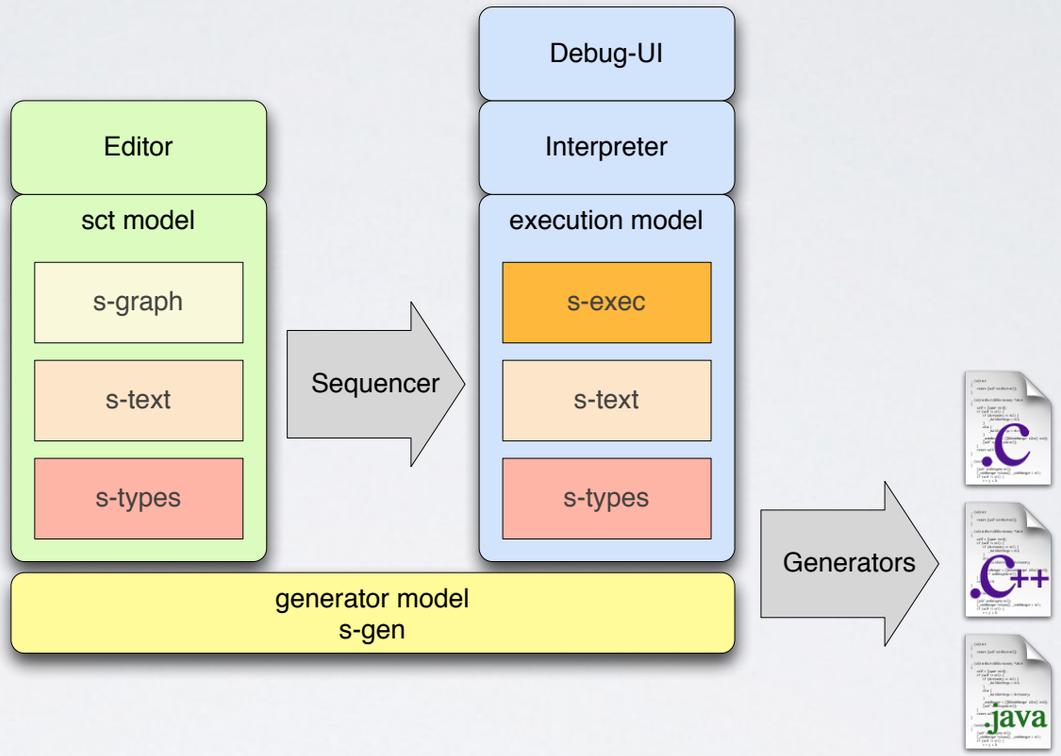
- statecharts are only one of many modeling concepts
- may be used in various combinations with other modeling concepts (i.e. structural models)

… *thus YAKINDU SCT also*

- *provides statecharts as a reusable ‚language module'*
- *is an open framework that provides various ‚integration points'*

Yakindu SCT - Extensibility

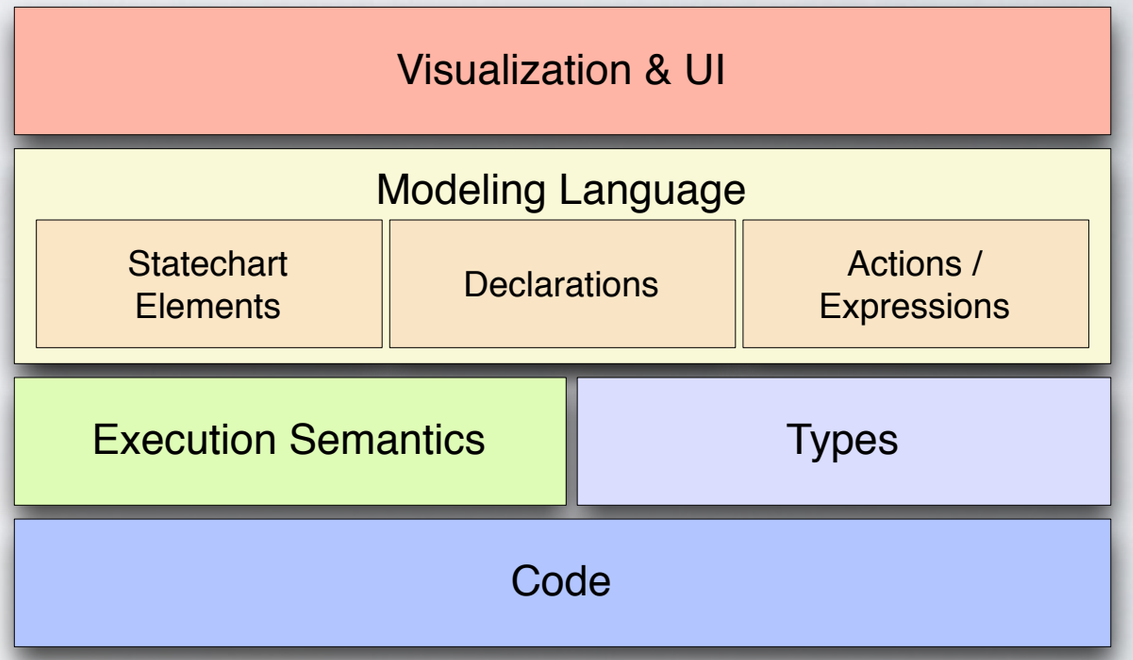
- Recap: different models are used around the Statechart formalism



- **s-graph** (EMF): specification of graphical structures
- **s-text** (Xtext): textual specification of declarations & expressions
- **s-type** (EMF): specification of types and type semantics
- **s-exec** (EMF): sequentialized statechart execution
- **s-gen** (Xtext): code generator parameterization

Integration based on SCT Extensions

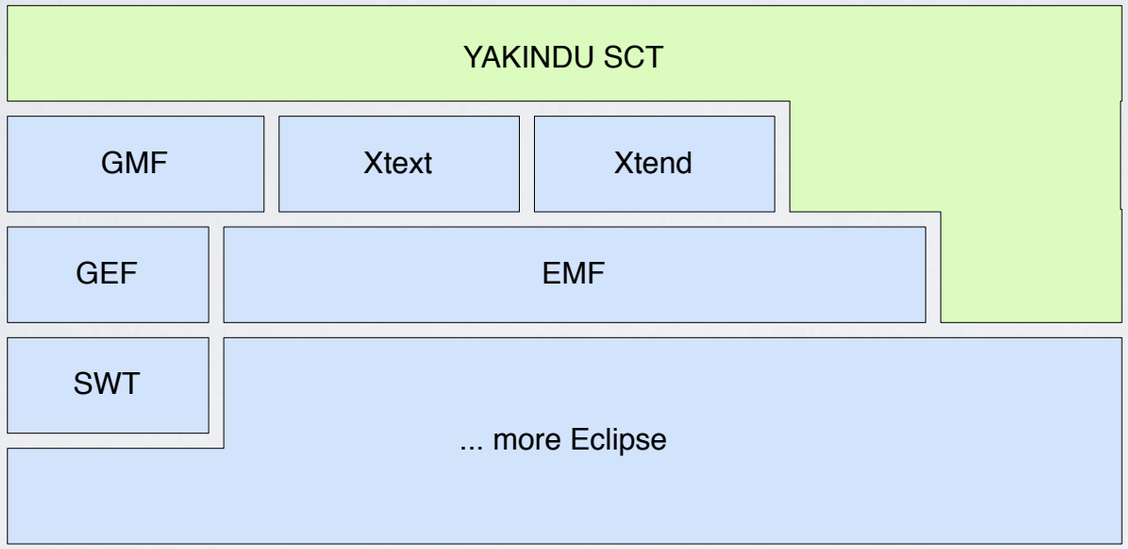
Integration & extension can be done on four levels (bottom up):



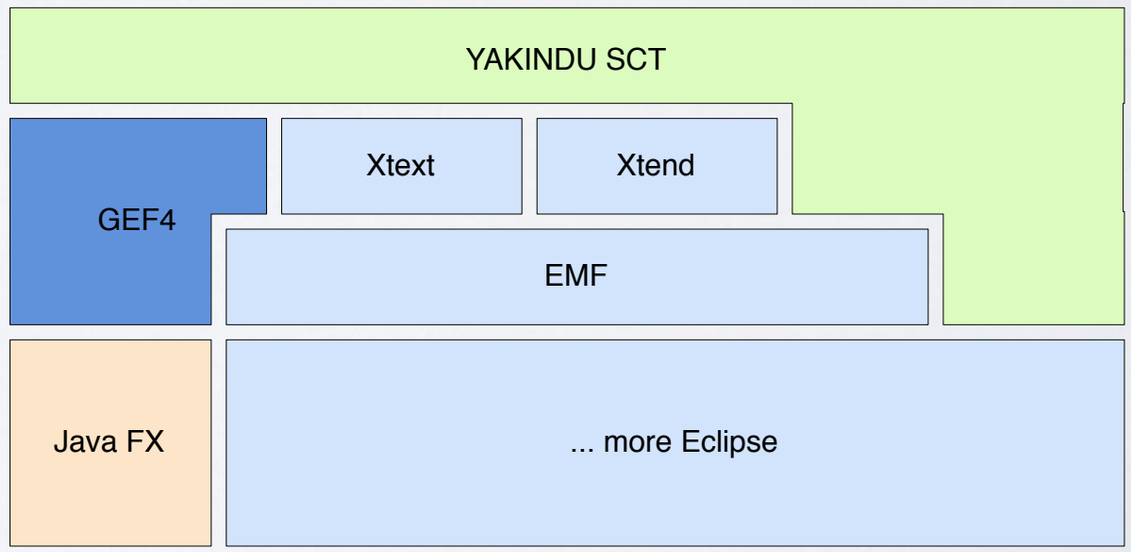
- **Code Level** - integrating with target platform
  - extending existing code generators
  - contributing own code generators
- **Semantic Level**
  - customize statechart execution semantics
  - provide types & type semantics
- **Language Level** - extend the statechart language itself
  - Meta model is modular and allows exchanging and extending (graphical) statechart elements, (textual) declarations and expressions
- **UI Level** - extend the modeling tool
  - e.g. extensions of graphical notation & tool extensions

Eclipse Technologies

Today:



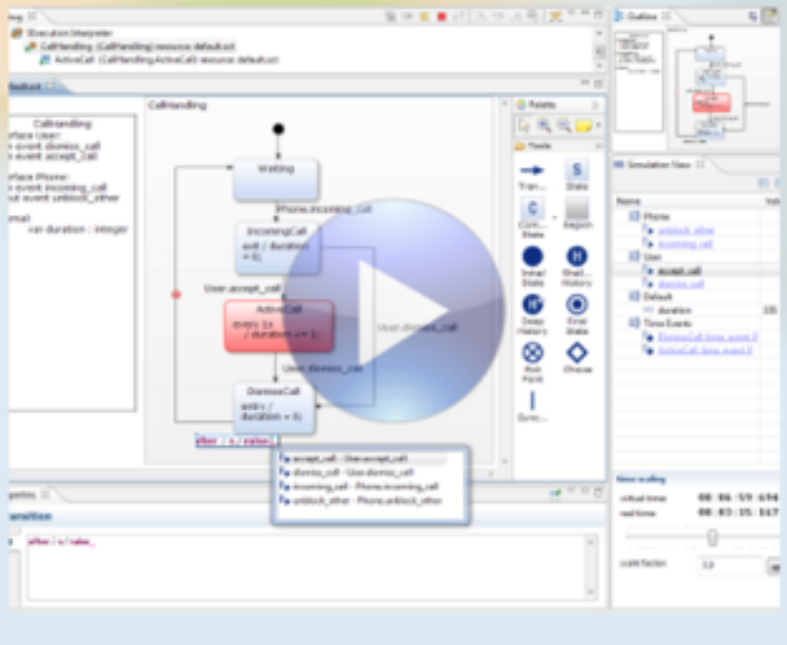
Tomorrow (likely…):



Yakindu Statechart Tools

The free to use, open source toolkit **YAKINDU Statechart Tools** (SCT) provides an integrated modeling environment for the specification and development of reactive, event-driven systems based on the concept of statecharts.

Download Documentation Tutorials



- Open Source / EPL: http://statecharts.org
- Hosted at GitHub: https://github.com/Yakindu/statecharts

Questions & Comments