

Model Driven SW-Development for Embedded Systems with Eclipse eTrice



Thomas Schütz
Protos Software GmbH



The Eclipse **etrice** Project

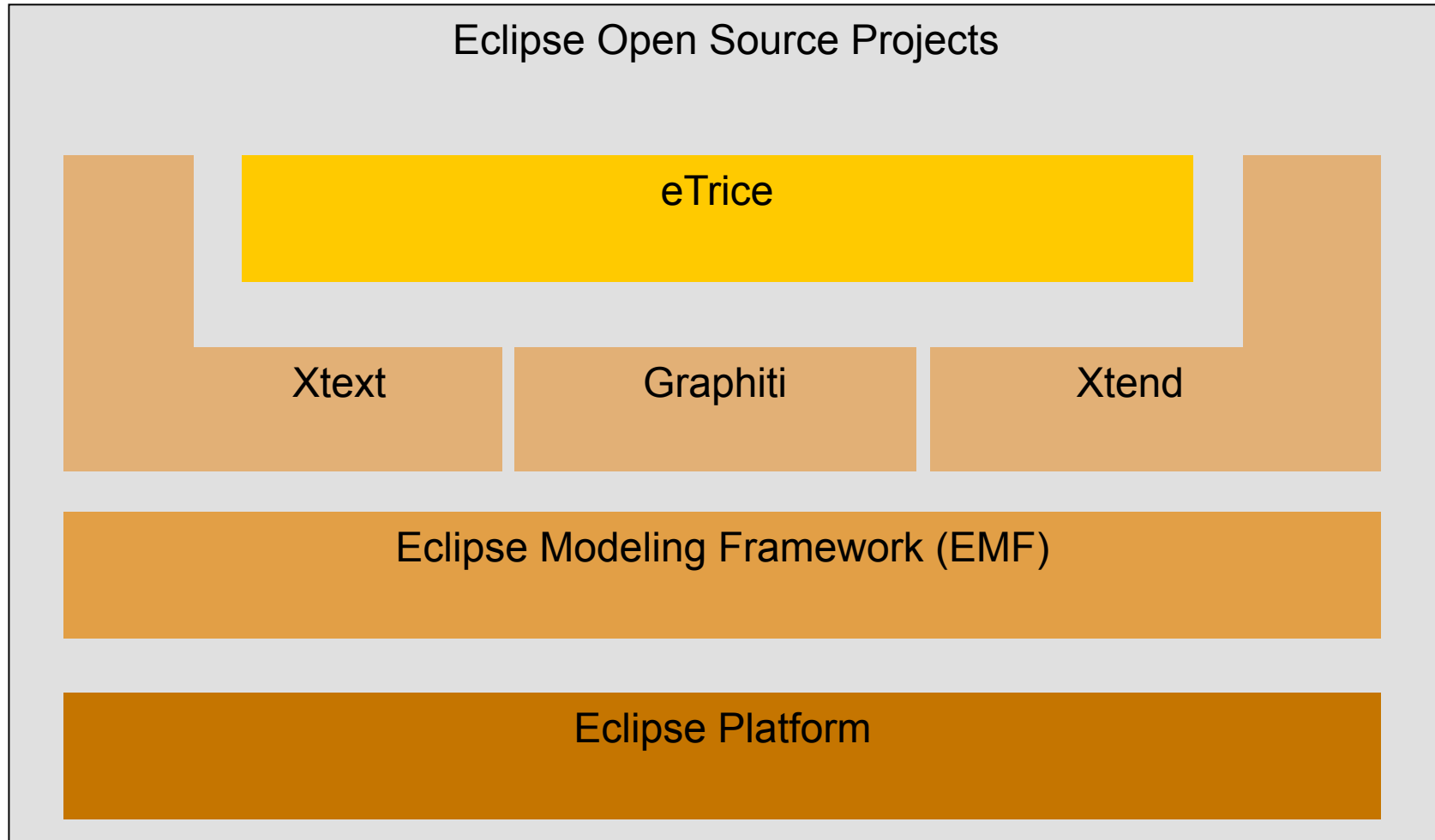


- **etrice** provides an implementation of the ROOM modeling language (Realtime Object Oriented Modeling)
- codegenerators and runtime for Java, C and C++
- based on industrial proven standard technologies (Eclipse, EMF, Xtext, Graphiti, ...)
- guiding principles
 - extensibility
 - conceptual integrity
 - **simplicity**

➤ ***etrice is a Modeling Toolset for eventdriven, concurrent realtime systems***

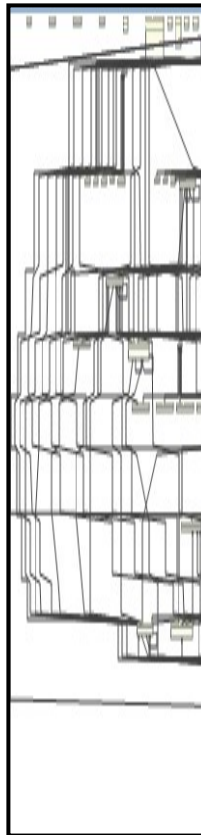


Eclipse **etrice** Software Stack

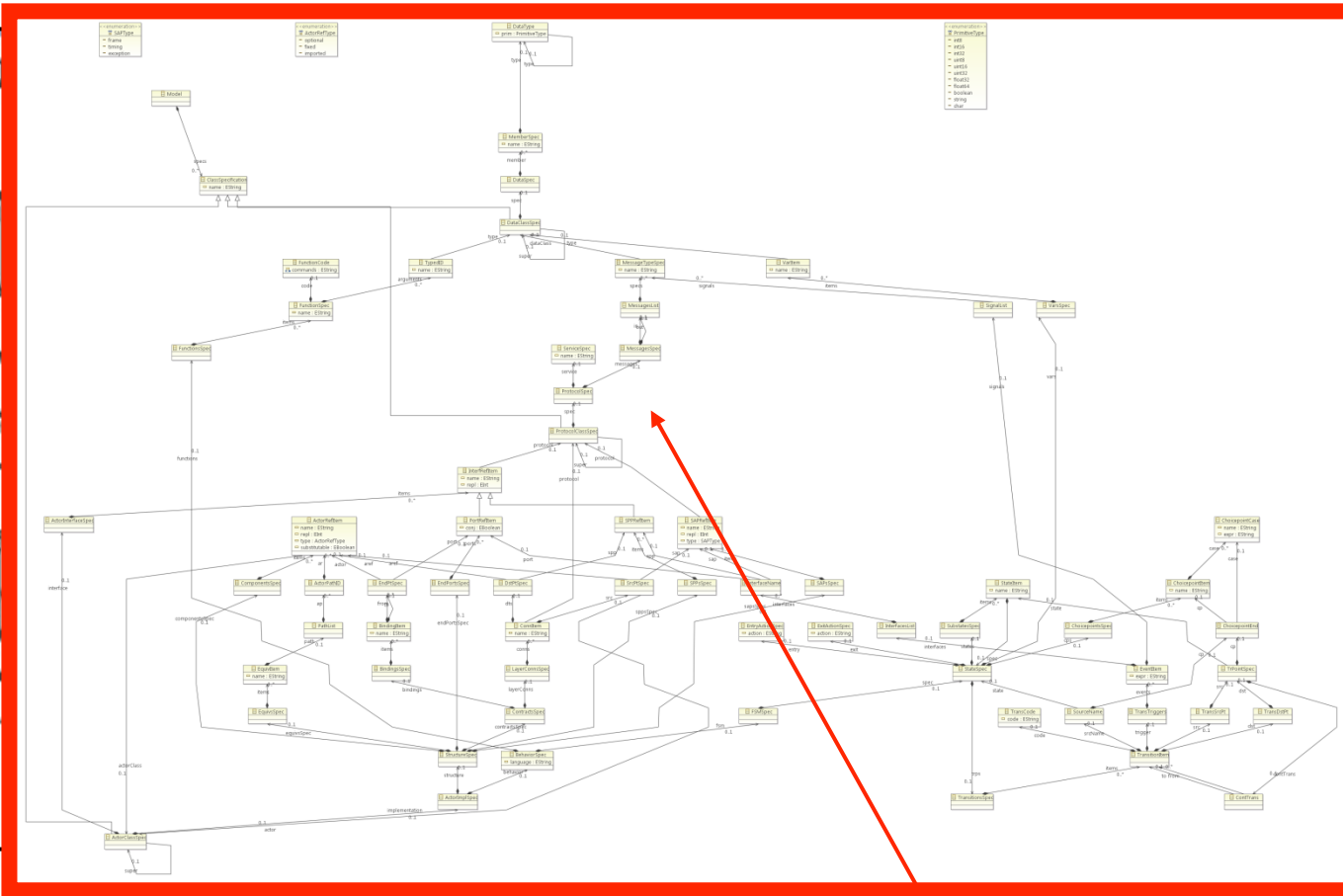


Why ROOM and not UML2?

➤ it's all about reduction of complexity !!!

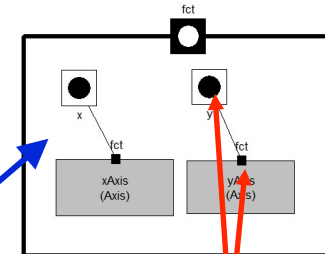
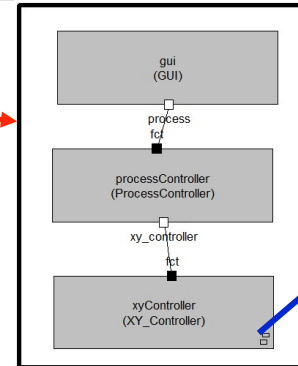


UML2 Meta Model

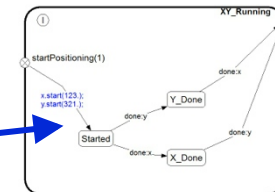
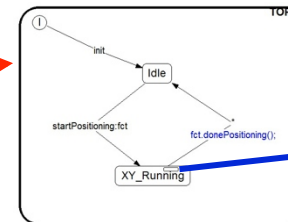


ROOM Meta Model

- Structure Modeling: Hierarchical **Actor** structures supporting component building
 - containment
 - layering
- Communication/Interfaces: **Protocols** and **Ports**
- Behavior Modeling: Hierarchical finite state machines (**FSMs**) for the event driven behavior
- Deployment: Decoupling of Actors by Ports enable free **deployment** of Actors on Threads and Nodes
- Reuse/Variants: Inheritance for Structure, Behavior (FSMs) and Protocols

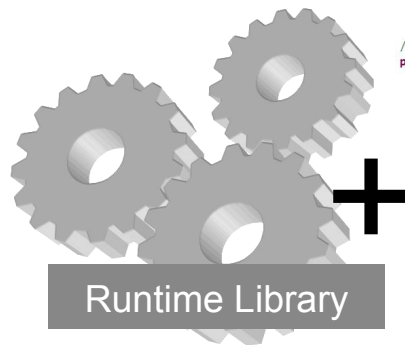
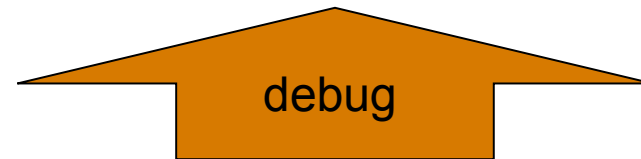
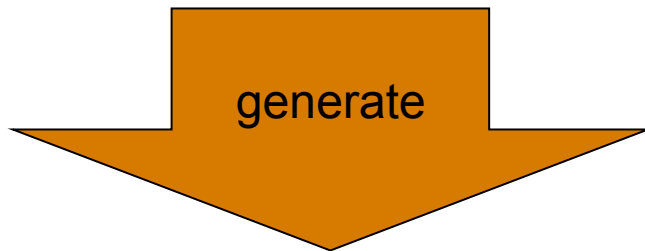
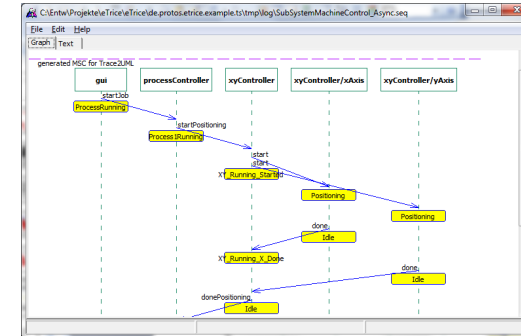
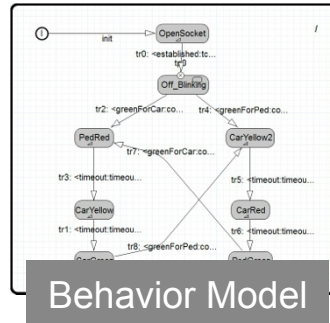
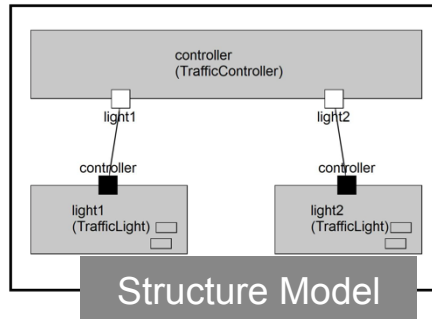


```
eventdriven ProtocolClass P_ServoDrive {
  incoming {
    Message start(position: float32)
    Message stop()
  }
  outgoing {
    Message done()
  }
}
```



```
LogicalThread defaultThread
LogicalThread processThread
LogicalThread servoThread

ActorInstanceMapping gui -> defaultThread
ActorInstanceMapping processController -> processThread
ActorInstanceMapping xyController -> servoThread
```



```

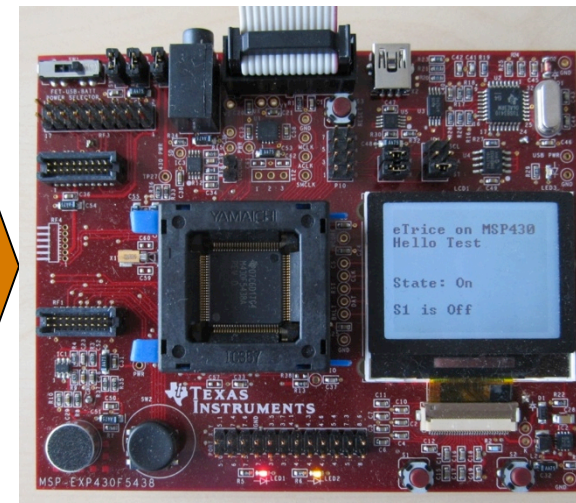
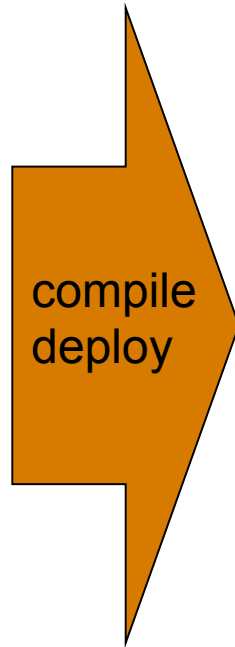
/* receiveEvent contains the main implementation of the FSM */
public void receiveEvent(InterfaceItemBase ifitem, int evt, Obj:
int trigger = ifitem.getLocalId() + EVT_SHIFT*evt;
int chain = NOT_CAUGHT;
int catching_state = NO_STATE;
boolean is_handler = false;
boolean skip_entry = false;

if (!handleSystemEvent(ifitem, evt, generic_data)) {
switch (this.state) {
case STATE_OpenSocket:
switch(trigger) {
case TRIG_tcpCtrl_established:
{
chain = CHAIN_TRANS_tr0_FROM_OpenSo

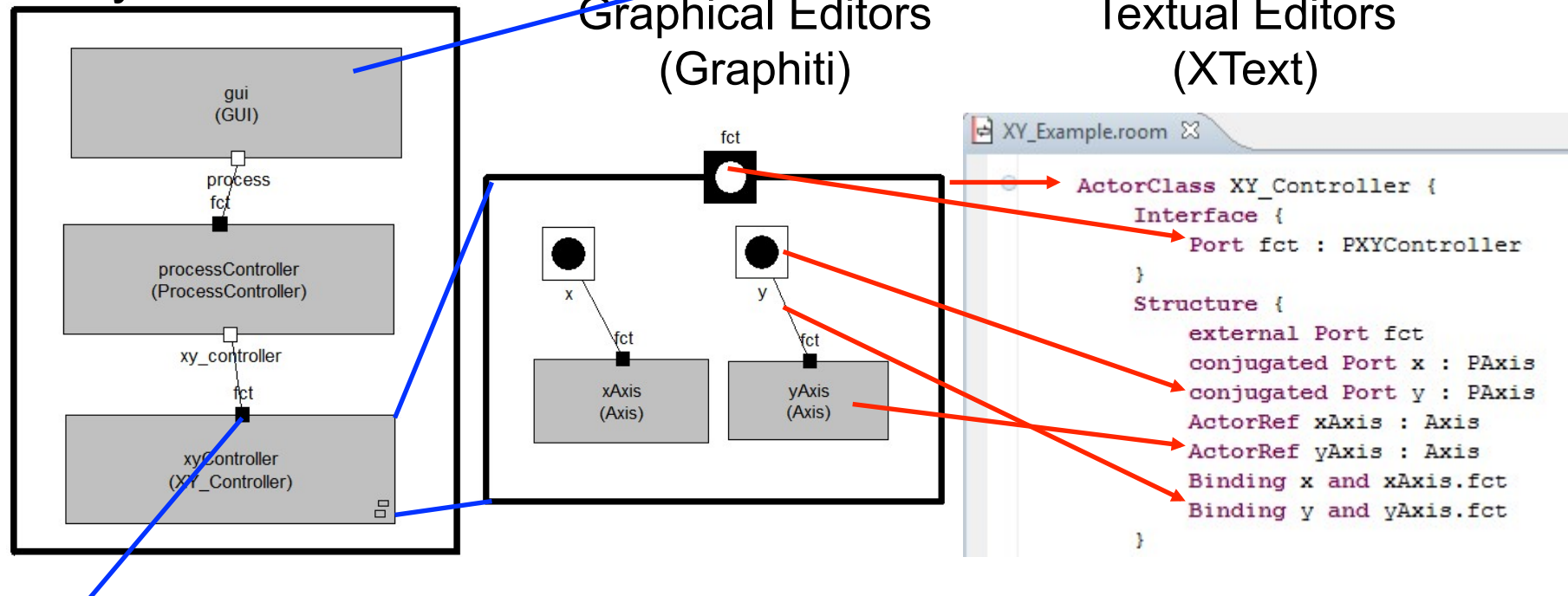
break;
case STATE_CarYellow:

```

Source Code



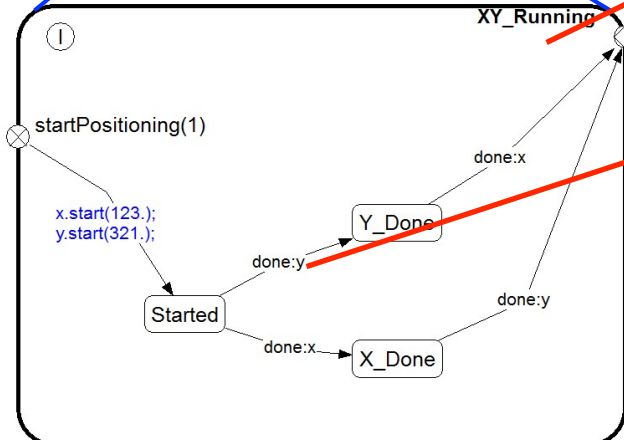
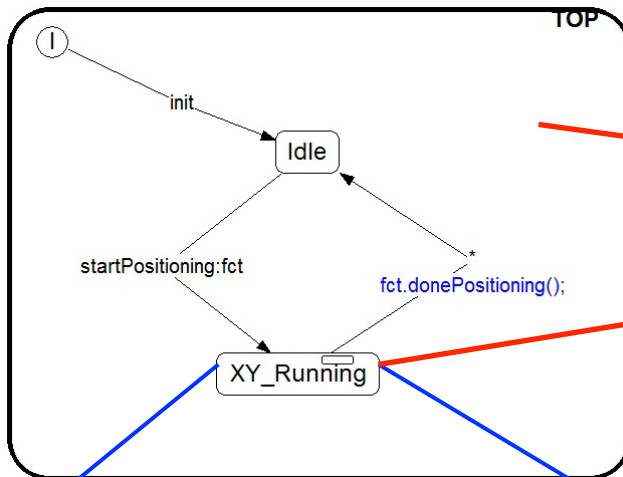
hierarchical components called **Actors** define the structure of a system



Ports are the only Interfaces of an actor and define a specific role in its environment.

➔ Models can be edited with graphical or textual editors

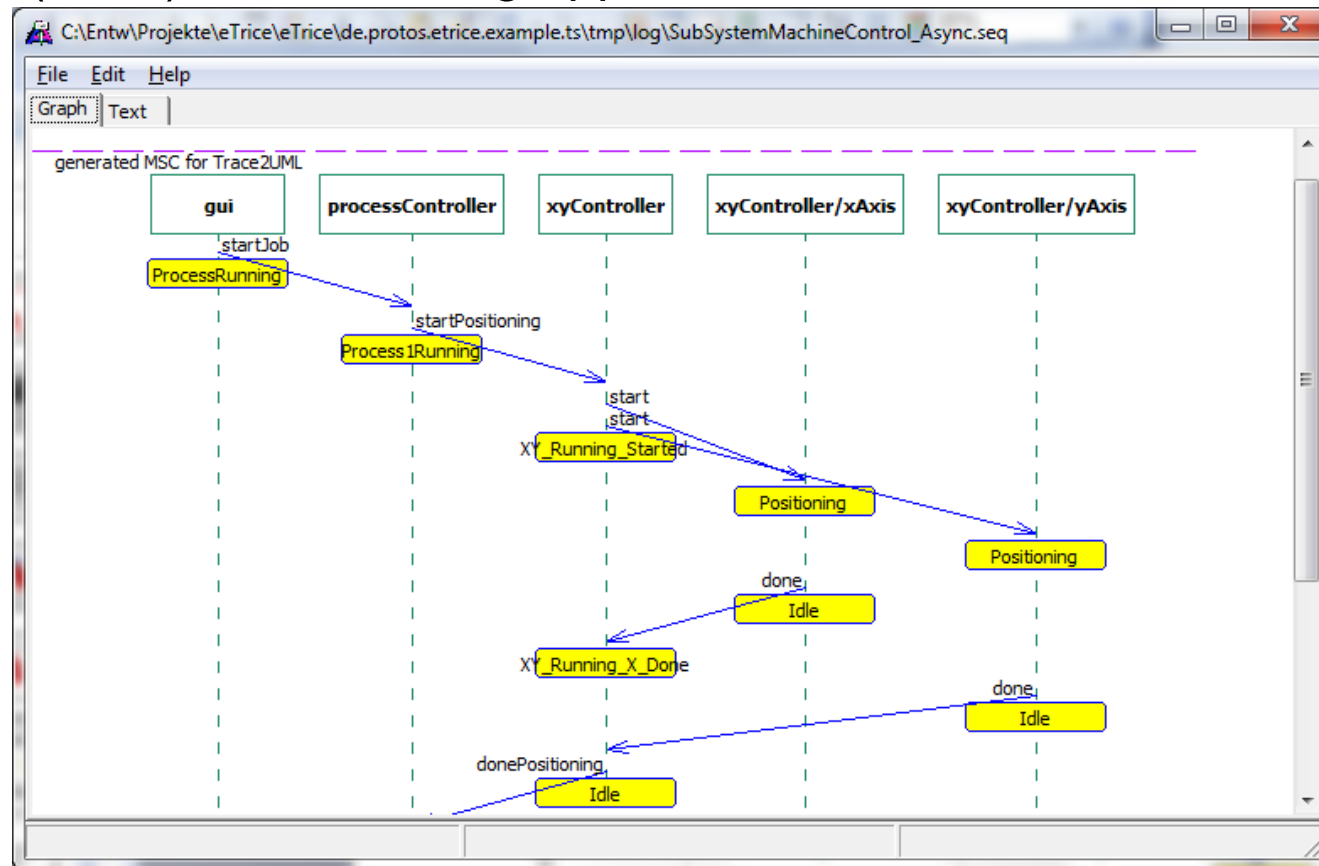
hierarchical **Statemachines** define the dynamical behaviour of Actors



```

*XY_Example.room X
Behavior {
  StateMachine {
    State Idle {}
    State XY_Running {
      subgraph {
        State Started {}
        State X_Done {}
        State Y_Done {}
        EntryPoint tp0
        ExitPoint tp1
        Transition tr1: Started -> Y_Done {
          triggers {
            <done:y>
          }
        }
        Transition tr2: Started -> X_Done {
          triggers {
            <done:x>
          }
        }
      }
    }
  }
}
  
```


The events can be logged on the target to create Message Sequence Charts (MSC) of the running application



Logs can be viewed with Trace2UML (<http://trace2uml.tigris.org/>)

- Model information can be generated as LaTeX documentation
- Manual documents can be integrated
- LaTeX documents can be converted into PDF, OpenDoc, Word-Doc, HTML, ...

3 Protocol Class Description

3.1 PTrafficLight

3.1.1 Incoming Messages

Message	Data	Description
greenForCar		trigger green for car
greenForPed		trigger green for pedestrians

3.1.2 Outgoing Messages

Message	Data	Description
greenForCarDone		positive response for greenForCar - is sent when switch is over
greenForPedDone		positive response for greenForPed - is sent when switch is over

4 Actor Class Description

4.1 TrafficlightExampleApplication

Toplevel Actor of the Trafficlight Example Application.

4.1.1 Structure

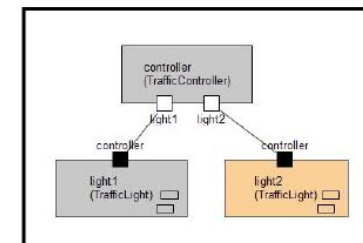
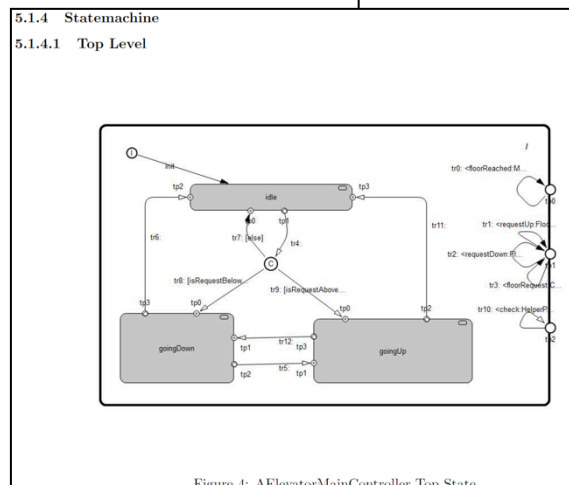
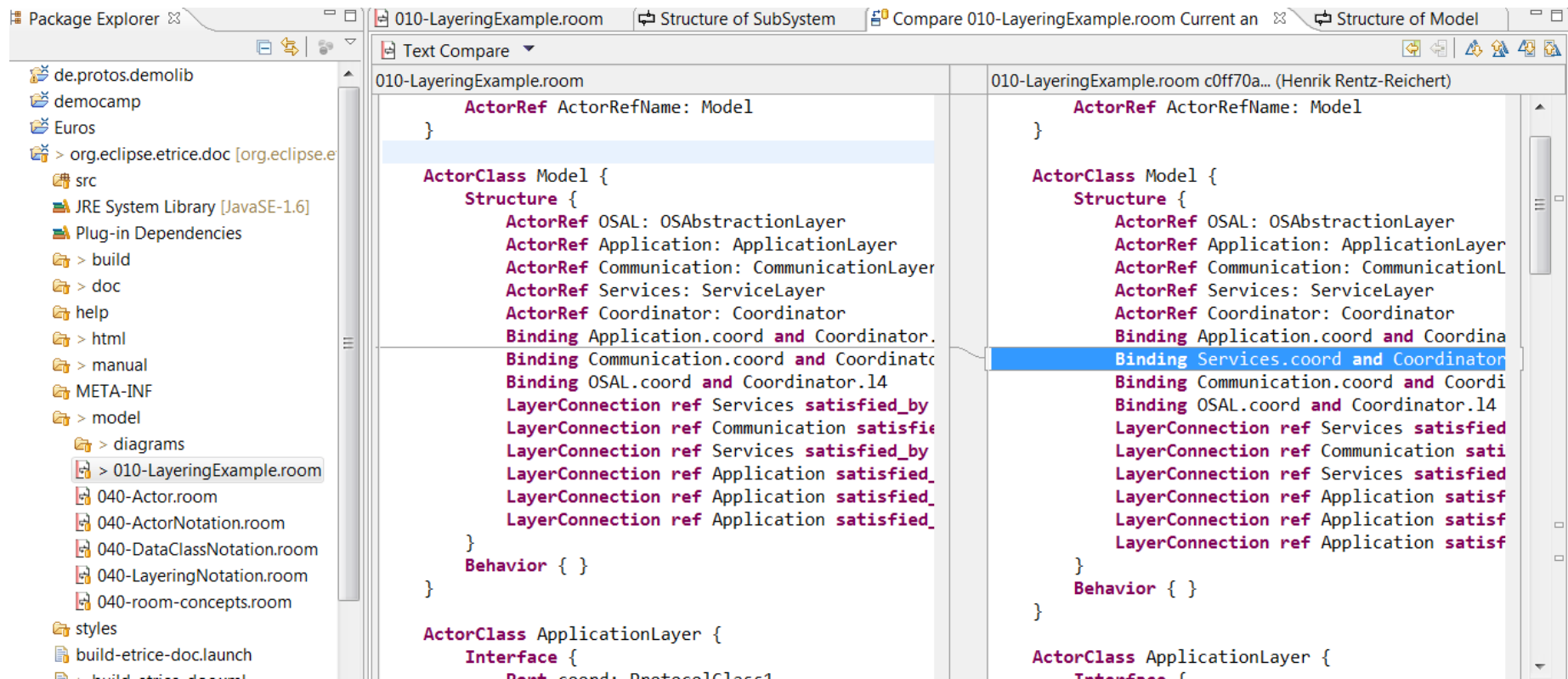


Figure 3: TrafficlightExampleApplication Structure



- Persistency based on Ascii Files enables simple Model Diff and Merge
- No special tooling necessary



The screenshot shows the Eclipse IDE with a 'Text Compare' window open. The window compares two versions of the file '010-LayeringExample.room'. The left pane shows the original file, and the right pane shows a modified version. The modified version has several changes highlighted in blue, including the addition of a 'Binding Services.coord and Coordinator.14' and the removal of 'Binding OSAL.coord and Coordinator.14'. The Package Explorer on the left shows the project structure, including 'de.protos.demolib', 'democamp', 'Euros', and 'org.eclipse.etrice.doc'.

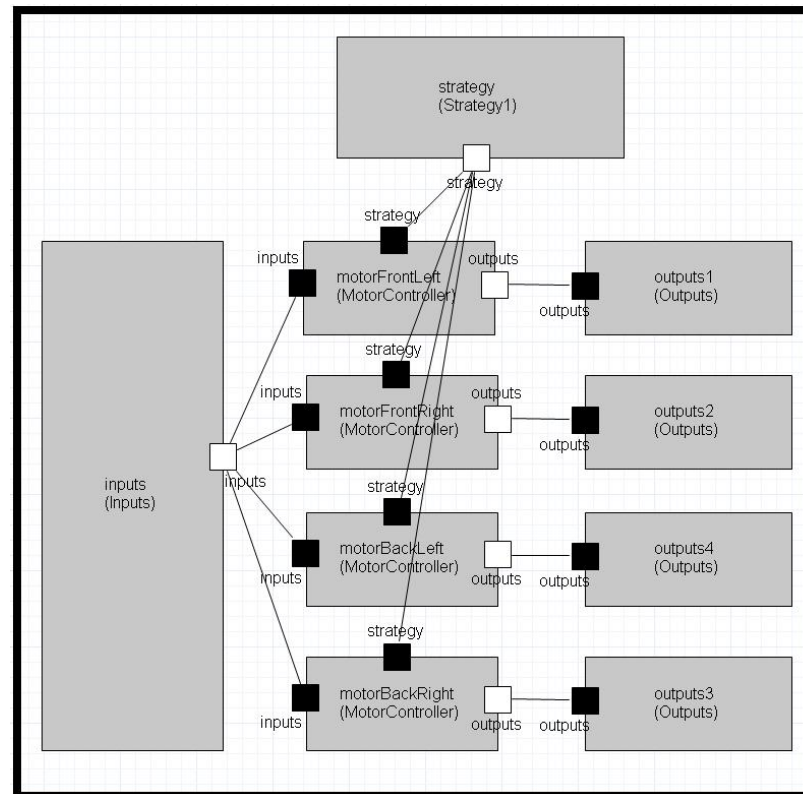
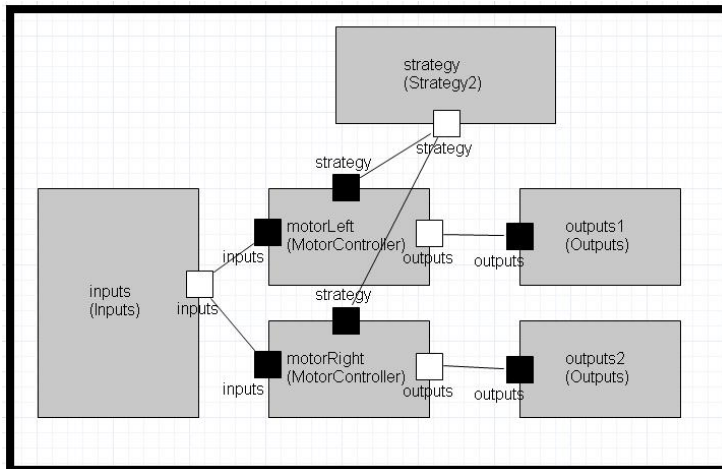
```

ActorRef ActorRefName: Model
}

ActorClass Model {
  Structure {
    ActorRef OSAL: OSAbstractionLayer
    ActorRef Application: ApplicationLayer
    ActorRef Communication: CommunicationLayer
    ActorRef Services: ServiceLayer
    ActorRef Coordinator: Coordinator
    Binding Application.coord and Coordinator.14
    Binding OSAL.coord and Coordinator.14
    LayerConnection ref Services satisfied_by
    LayerConnection ref Communication satisfie
    LayerConnection ref Services satisfied_by
    LayerConnection ref Application satisfied_
    LayerConnection ref Application satisfied_
  }
  Behavior { }
}

ActorClass ApplicationLayer {
  Interface {
    next coord: ProtocolClass1
  }
}
  
```

- existing Actors can be combined to different applications
- Actors with same Ports can replace each other

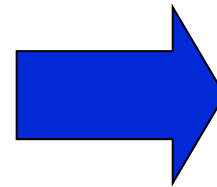




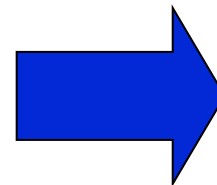
Example: PariTec Automation Control

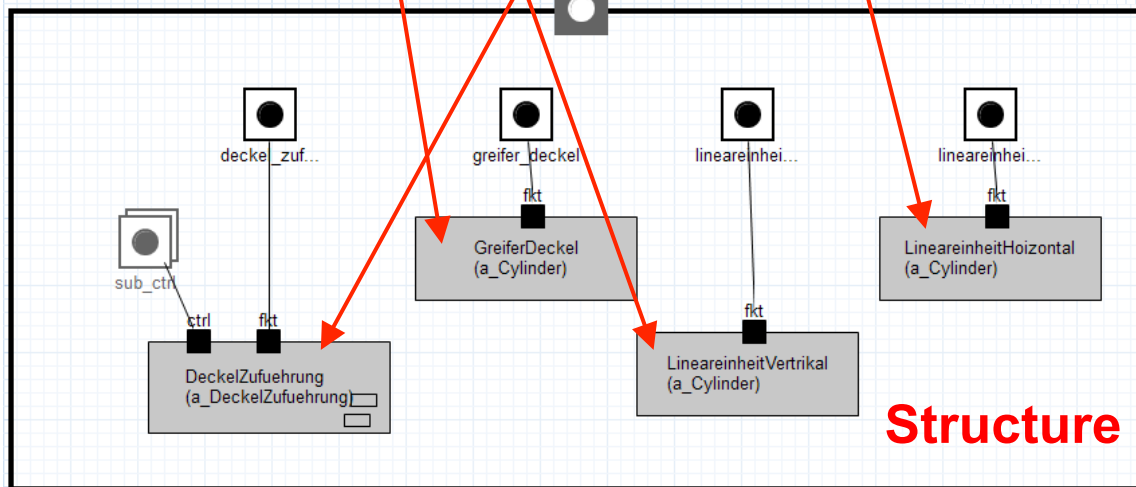
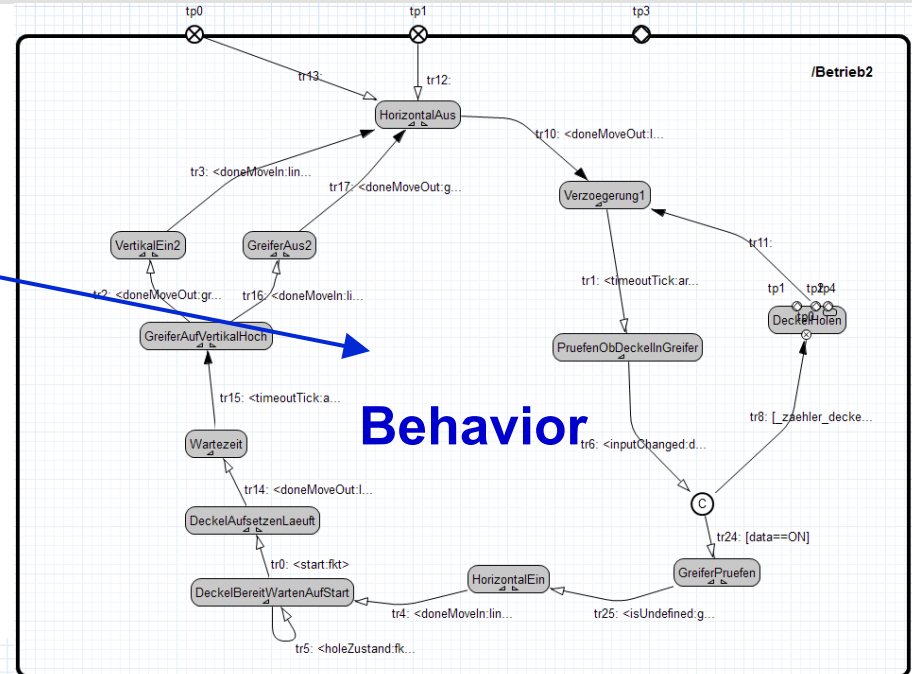
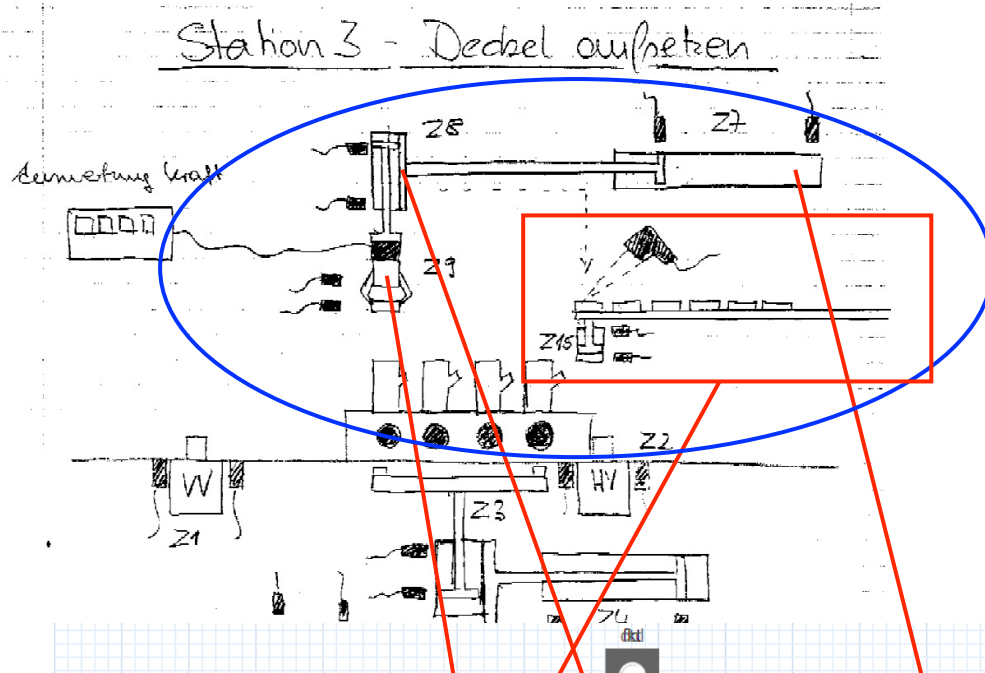


1. smaller production system for nebulisers

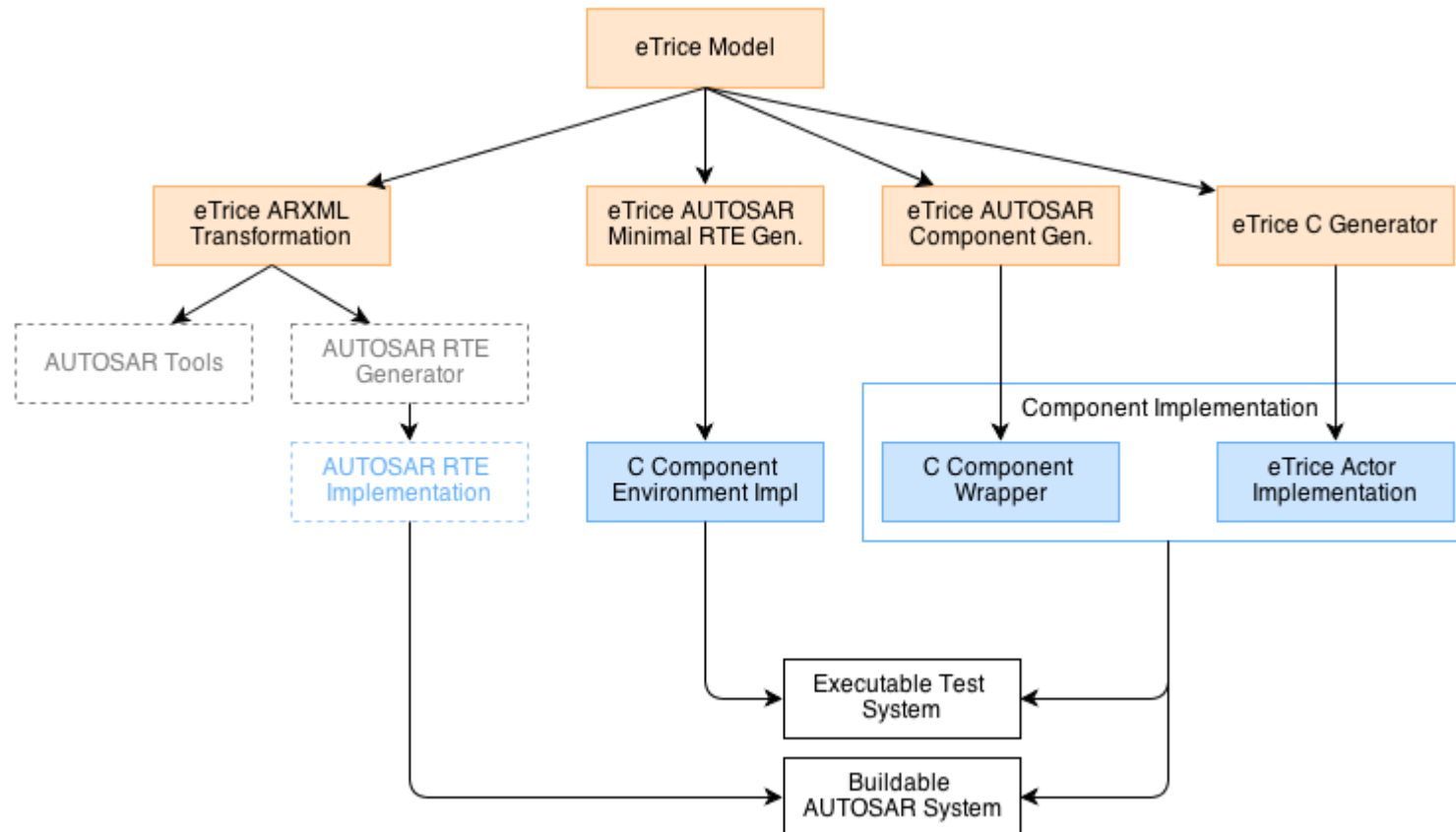


2. bigger production system for compressors





- eTrice for rapid prototyping of AUTOSAR systems (structure and behavior)
- eTrice as simple component editor for a subset of AUTOSAR
- First working prototype of AUTOSAR transformation: Generators for minimal RTE, component wrappers, ARXML
- Will probably be published under the AUTOSAR license





Domains: Automation



- eTrice for development of full blown automation control systems
- Existing automation component library (basic software) will be published most likely under the eTrice project
 - Basic services: IO (Analog, Digital, Fieldbus Wrappers), Communication (TCP/IP, M2M, Serial), Timing
 - Atomic control components: cylinders, drives, feedback controllers
 - Architecture components: control structures for distributed systems, inline systems, error handling and reporting
 - Simulation components for unit testing, virtual commissioning, Model/HW/ Software in the loop



Community



Committers

- Protos (3), Tieto ES (1), Dräger Medical (1)

Contributors

- Contributors of Tieto, Mixed Mode
- 2012: 3 Google Summer of Code (GSOC) Projects with Students in India and Singapore
- 2013: 1 GSOC Project with Student in India

Projects

- Automotive
- Automation
- Banking (starting)
- ?

- **Release 0.3 planned for Q3 2013**
 - First prototype of C++ generator
 - Data configuration model
 - Integration of GSOC projects:
 - Detail level (expression) language
 - Model checking for Statemachines
 - Layouter for graphical editors
 - First version of physical- and mapping model (deployment)
 - Consolidation of current features
- **Further releases**
 - more model level debugging (state machine back animation, data inspection and manipulation and message injection)
 - model level support for distributed systems
 - more detailed documentation generator
 - Abstract execution for ROOM and expression language
 - ...



... any questions?

Thomas Schütz

ts@protos.de

<http://www.eclipse.org/etrice>

<http://www.protos.de>