

# Formal Verification of UML Statecharts with Real-Time Extensions

<sup>1</sup>Alexandre David      <sup>2</sup>M. Oliver Möller      <sup>1</sup>Wang Yi

<sup>1</sup> **Uppsala University**

<sup>2</sup>  **BRICS Århus**

{adavid,yi}@docs.uu.se      omoeller@brics.dk

## Outline:

- 1** UML, Statecharts, and Time
- 2** Semantics for Formal Verification
- 3** Verifying a Pacemaker with UPPAAL

# Unified Modeling Language (UML)

---

Born from unification of other methods (*Booch, OMT, OOSE*)

Different *views* of a system:

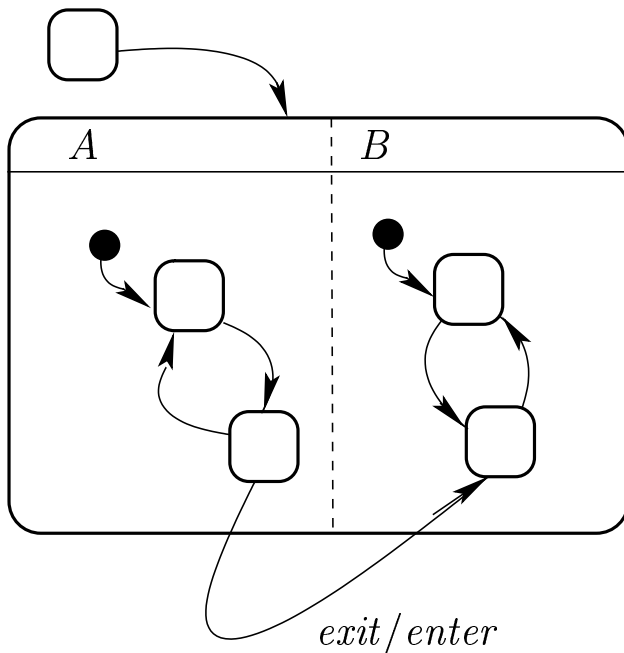
- A) user view - *use case diagrams*
- B) structural view - *class diagrams*
- C) behavioral view - *statecharts*
- D) environmental view - *deployment diagrams*
- E) implementation view - *component diagrams*

An *evolving standard*:

1.3	finished 2000
1.4	finished 2001
2.0	work in progress (4 RFP issued May/Sept)

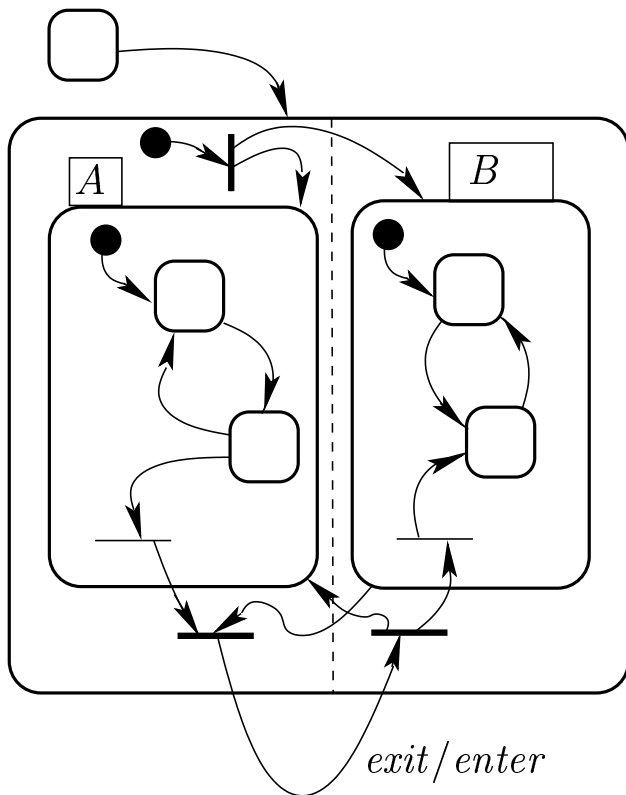
# The Statechart Formalism

## Features



- hierarchical state machines
- parallelism (on any level)
- history
- event communication
- powerful synchronization mechanisms
- inter-level transitions
- actions that are dependent on states
- actions on entry/exit
- ...

# Restricted Statechart Formalism



## Current restricted features

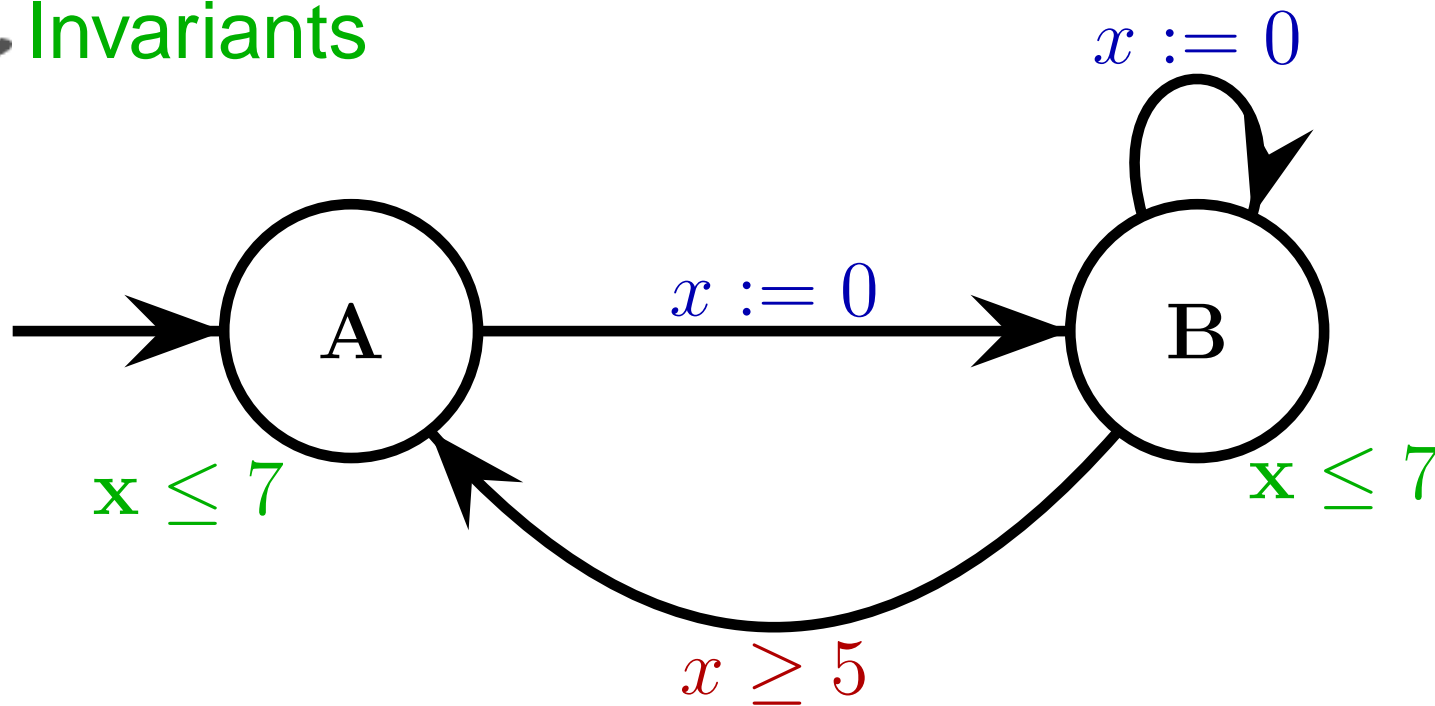
- hierarchical state machines ✓
- parallelism (on any level) ✓
- history ✓
- **no event communication**
- **no sync states**
- **no inter-level transitions**
- **no actions that are dependent on states**
- **no actions on entry/exit**

## instead:

- hand-shake style synchronization
- shared variables

# Real-Time Extensions

- Clocks
- (timed) **Guards**
- **Invariants**





# A Word on Semantics

---

## UML-statecharts:

- informal (textual) semantic statements
- ambiguity of *text*
- variations over 1.3 / 1.4 / 2.0
- implementations make user-driven choices

## our formalism:

- rule-based, formal semantic
- unambiguous
- ➔ *not* identical, makes clear choices
- ➔ any given formal statechart semantic should be “easy” to translate into it

# Semantic Rules (example)

**configuration:**  $\langle \rho, \mu, \nu, \theta \rangle$  with  $\rho$  : control locations  
 $\mu$  : valuation of integer variables  
 $\nu$  : valuation of clocks  
 $\theta$  : history

**operation:**

$t : l \xrightarrow{g,s,r,u} l', \rho, \mu, \nu$  a transition

$$\frac{g(\mu, \nu) \quad \text{JoinEnabled}(\rho, \mu, \nu, l) \quad \text{Inv}(\rho^{\mathcal{I}_t}, \nu^{\mathcal{I}_t}) \quad \neg \text{EXIT}(l')}{(\rho, \mu, \nu, \theta) \xrightarrow{t} \mathcal{I}_t(\rho, \mu, \nu, \theta)} \text{action}$$

# Model Checking

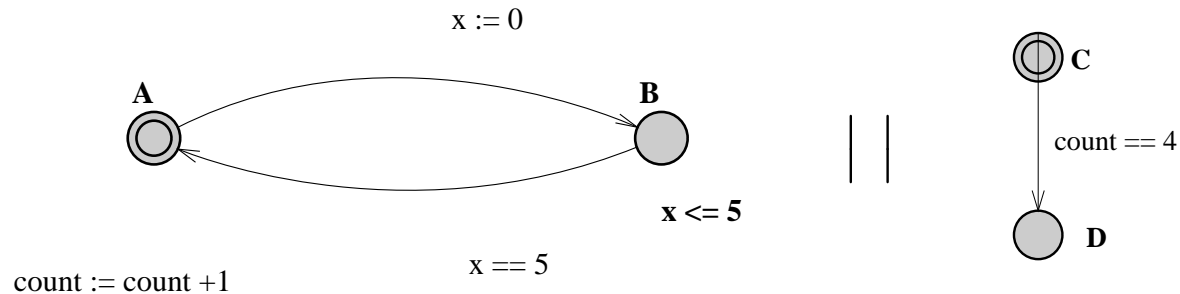
$$M \stackrel{?}{\models} \varphi$$

$M$  : description of the system

$\varphi$  : desired property

- easier than proving a general theorem
- completely automatic ('yes' or counterexample)
- *efficient* algorithms tailored for classes of problems

# Real-Time Model Checking with UPPAAL



```
clock x; int count
```

Only subset of TCTL supported:

- $E \langle \rangle \varphi$                       reachability
- $A [] \varphi$                         safety (invariantly  $\varphi$ )
- $E [] \varphi$                         possibly always  $\varphi$
- $A \langle \rangle \varphi$                         inevitably  $\varphi$
- $A [] \varphi \Rightarrow A \langle \rangle \psi$     unbounded response

$\varphi, \psi$  : propositional formula over locations and (existing) clocks

# From (timed) Statecharts to UPPAAL

*Rhapsody timed Statechart*

hierarchical model  
informal description



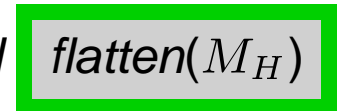
*HTA model*



TA-close hierarchy  
formal semantics

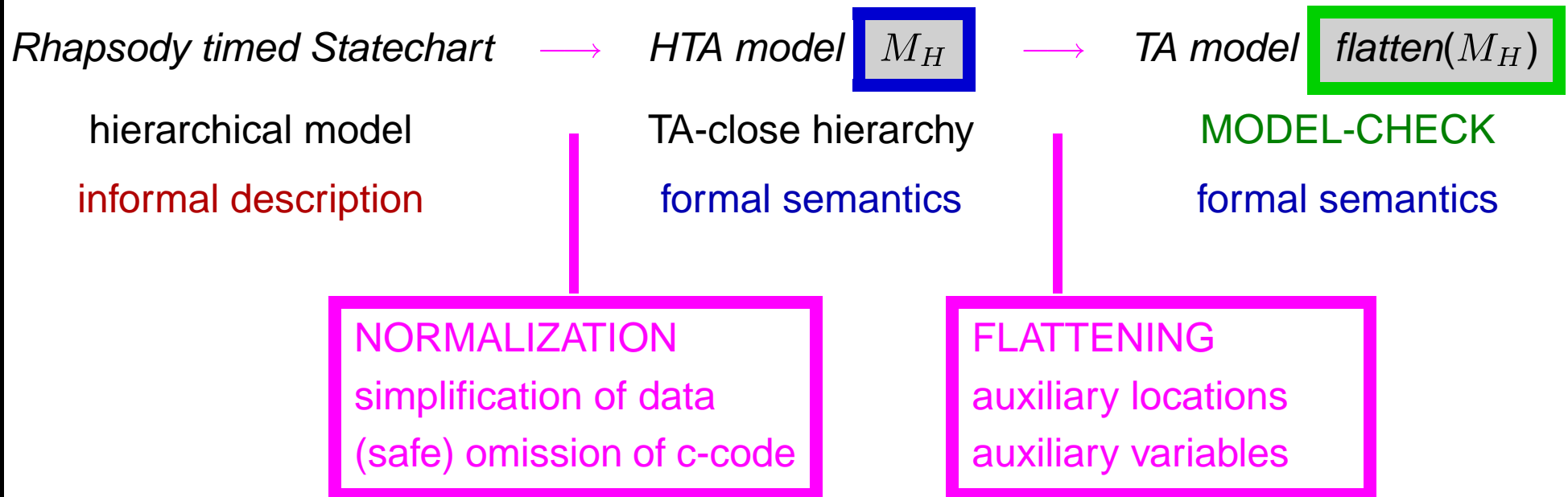


*TA model*



MODEL-CHECK  
formal semantics

# From (timed) Statecharts to UPPAAL



**Guiding Principle:** Make it easy to adjust to small changes

# Soundness & Correctness

Translations introduce slack. Thus

$$\boxed{M_H} \models \varphi \not\leftrightarrow \boxed{\text{flatten}(M_H)} \models \text{flatten}(\varphi)$$

but

$$M_H \models \varphi \Leftrightarrow \text{flatten}(M_H) \models_{\text{project}(M)} \text{flatten}(\varphi)$$

timed transition system

↓ give rise to

timed  $M_H$  traces

timed  $\text{flatten}(M)$  traces

↓ project to  $M_H$

timed  $M_H$  traces

*match*



# Outline of the Flattening

3 phases to flatten a hierarchical structure:

## 1. **Collect instantiations**

every superstate becomes one (flat) timed automaton

## 2. **Compute global joins**

mimic synchronization-on-exit in the the flat automata

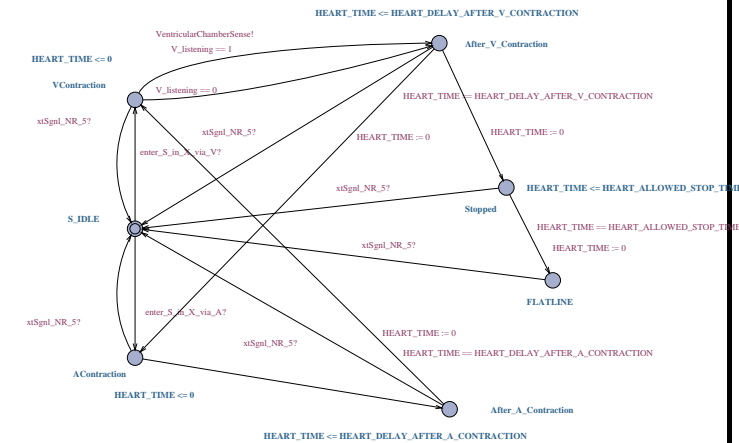
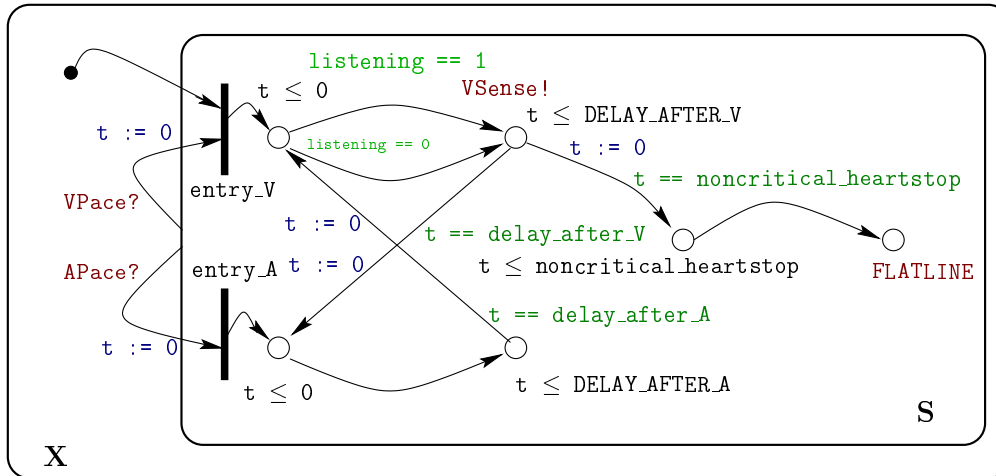
*principle:* use counters & and add threshold-guard

## 3. **Post-process channel communication**

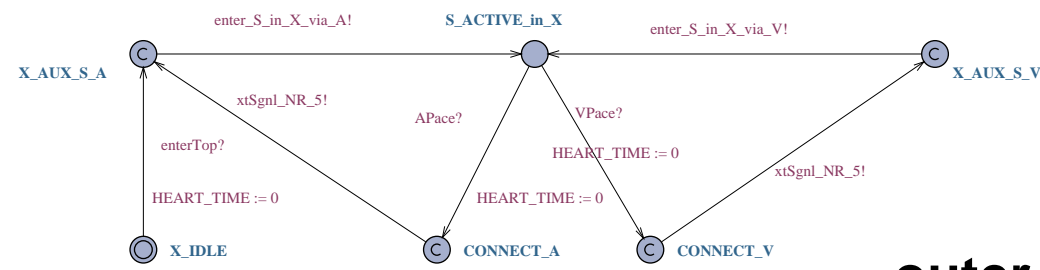
a transitions may not synchronize with its own superstate

*principle:* duplicate channels & restrict scope

# Example: Flattening the Model of a Human Heart

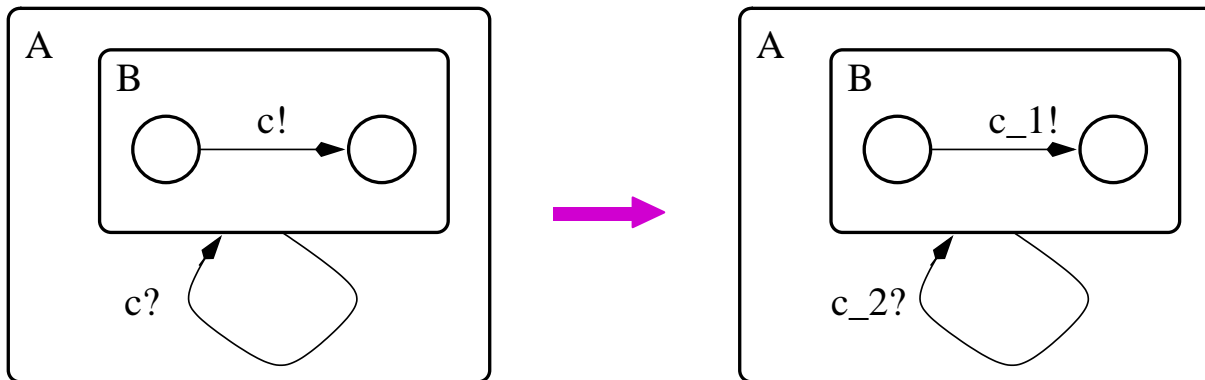


**inner superstate**



**outer superstate**

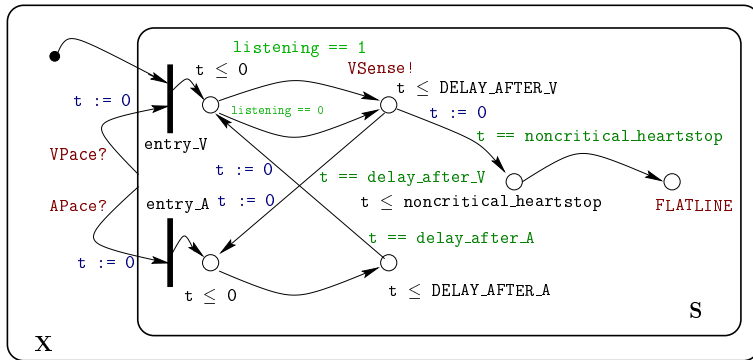
# Communication Conflict



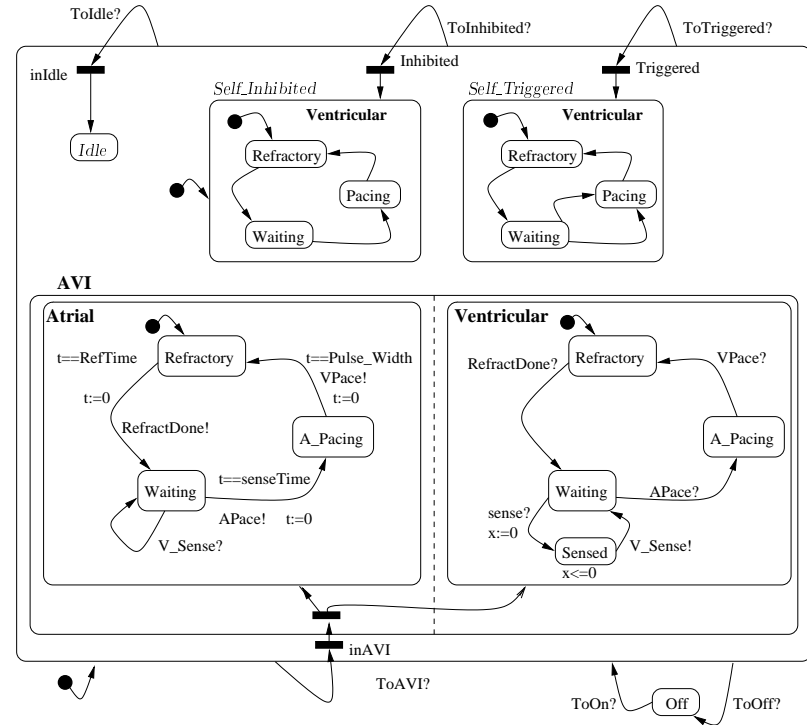
- cannot keep **c**
- cannot remove **c**

- rename **c** inside
- rename **c** outside
- modify other transitions:  
    either choose one of  $c_1$ ,  $c_2$   
    or duplicate transition (allow both)

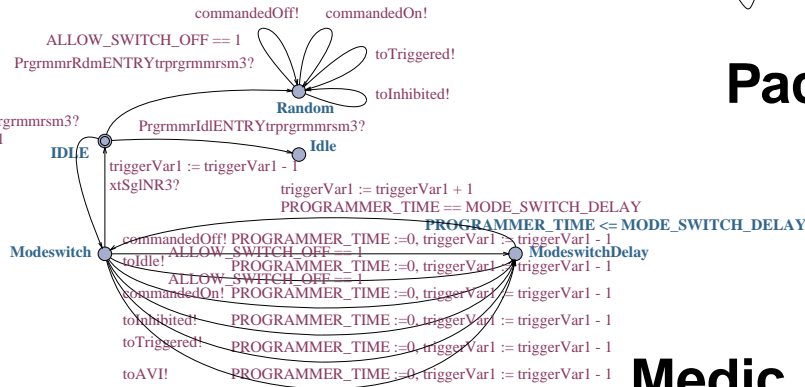
# Model-Checking a Pacemaker



Human Heart



Pacemaker



Medic

# Flattening of the Pacemaker Model

	HTA model	UPPAAL model
# XML tags	564	1191
# proper control locations	35	45
# pseudo-states / committed locations	33	63
# transitions	47	177
# variables and constants	33	72
# formal clocks	6	6

# Model-Checking the Pacemaker

---

- DEADLOCK:  
possible (if heart stops)
- SAFETY:  
 $A[] \neg \text{heart stops}$   
only true for 'good' medic
- LIVENESS:  
 $A[] V\text{contract} \Rightarrow A\langle\rangle A\text{contract}$

# Model-Checking the Pacemaker

- DEADLOCK:

possible (if heart stops)

- SAFETY:

$A[] \neg \text{heart stops}$

only true for 'good' medic

- LIVENESS:

$A[] V\text{contract} \Rightarrow A\langle\rangle A\text{contract}$

## Parameters:

REFRACTORY\_TIME = 50

SENSE\_TIMEOUT = 15

DELAY\_AFTER\_V = 50

DELAY\_AFTER\_A = 5

HEART\_ALLOWED\_STOP\_TIME = 135

MODE\_SWITCH\_DELAY = 66

E.g. for  $\text{MODE\_SWITCH\_DELAY} = 65$ ,  $A[] \neg \text{heart stops}$  is violated

## Related Work

---

- Variations of the statechart formalism  
e.g., in 1994, von der Beeck lists 21 different statecharts and distinguishes them in 26 criteria
- *Timed* extension of statecharts  
e.g., work of Kesten/Pnueli, Petersohn, and others
- UML profile for *Schedulability, Performance and Time*  
general time model, both discrete and continuous  
no progress notion with invariants
- realizations of UML, that extend the standard  
e.g., the Rhapsody tool has timers

# Our Formalism in the European WOODDES Project

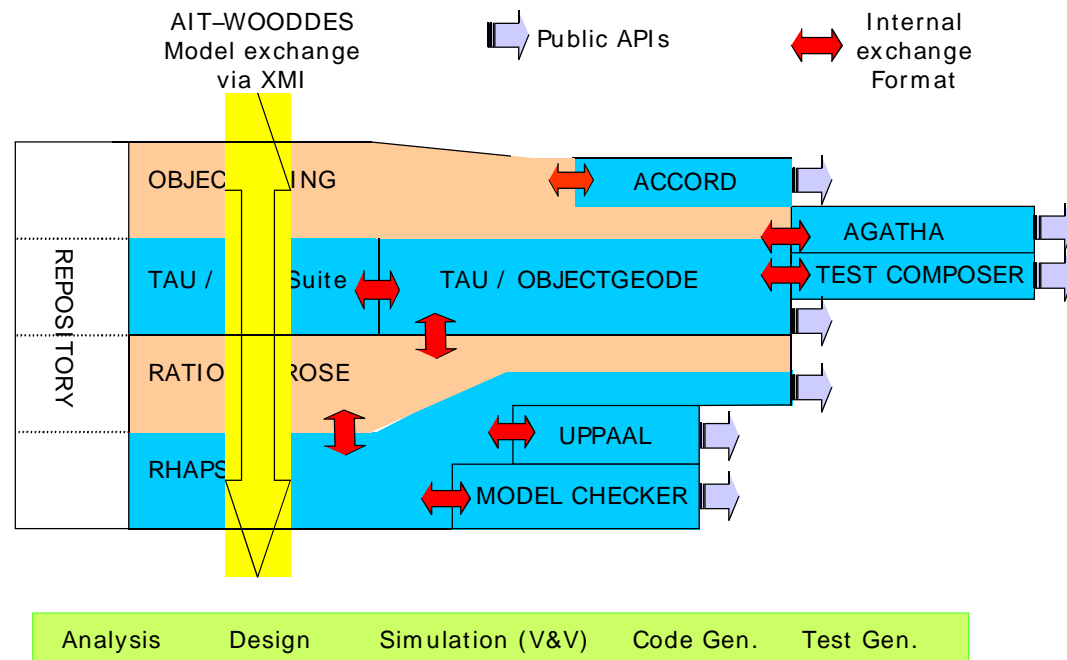
Workshop for Object-Oriented Design and Development of Embedded Systems

## Partners:

-  PSA
-  Mecel
-  CEA
-  I-Logix
-  Intracom
-  Offis
-  Uppsala
-  Aalborg

## Objectives:

- UML Real-Time profile
- WOODDES methodology & tool platform



# Conclusions & Future Work

---

## Status

- ✓ XML grammar
- ✓ semantics
- ✓ flattening

## Future Work

- formal proof for semantic correspondence
- implementation of an hierarchical editor
- integrate HTAs in the UPPAAL tool

# References

---

- [AD94] R. Alur and D.L. dill. A Theory of Timed Automata. In *Theoretical Computer Science*, number 125, 1994
- [vdB94] Michael von der Beeck. A Comparison of Statechart Variants. In de Roever Langmaack and Vytopil, editors, *Formal Techniques in RealTime and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 128–148. Springer-Verlag, 1994.
- [D99] Bruce Powel Douglass. Real-Time UML, Second Edition - Developing Efficient Objects for Embedded Systems. *Addison-Wesley*, 1999
- [DM01] Alexandre David and M. Oliver Möller. From Hierarichcal Timed Automata to UPPAAL. Research Series RS-01-11, BRICS, Department of Computer Science, University of Aarhus, March 2001. see <http://www.brics.dk/RS/01/11/index.html>.
- [OMG] Unified Modeling Language, version 1.4. Download from <http://www.omg.org>
- [WOODDES] WOODDES web page: <http://wooddes.intranet.gr>