Formal Verification of UML Statecharts with Real-Time Extensions

[1] Alexandre David   [2] M. Oliver Möller   [1] Wang Yi

[1] **Uppsala University**   [2] 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*

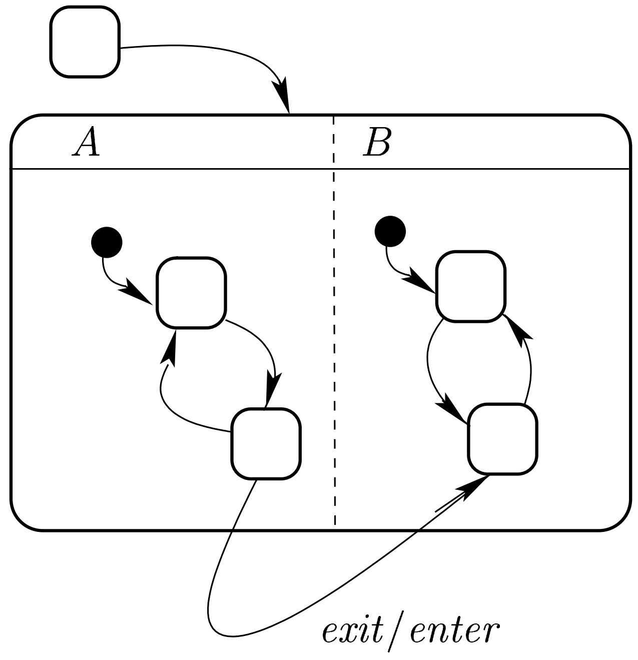
C) behavioral view - *statecharts*

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)

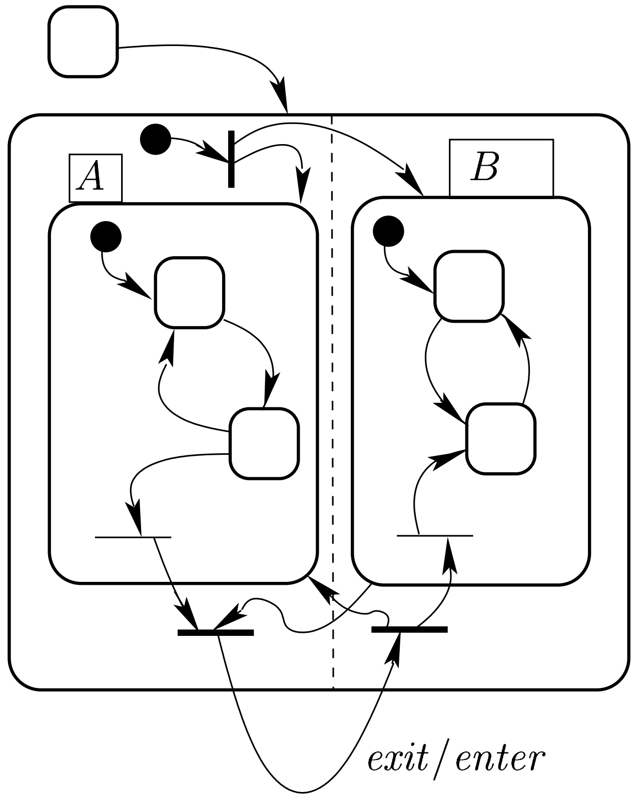
The Statechart Formalism



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

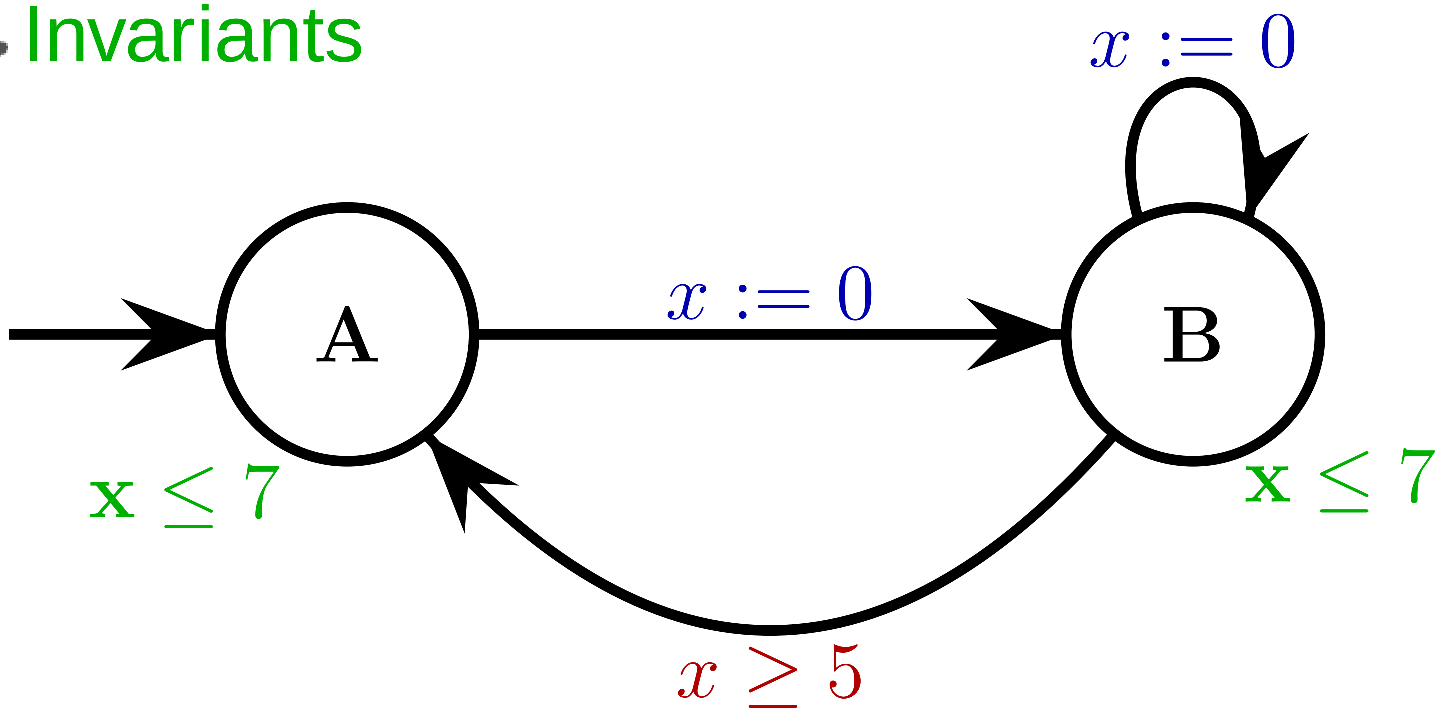
- hierarchical state machines ✓
- parallelism (on any level) ✓
- history ✓
- **no** event communication
- **no** sync states
- **no** inter-level transitions
- **no** actions that are dependent on states
- **no** actions on entry/exit

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

ρ	:	control locations
μ	:	valuation of integer variables
ν	:	valuation of clocks
θ	:	history

operation:

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

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

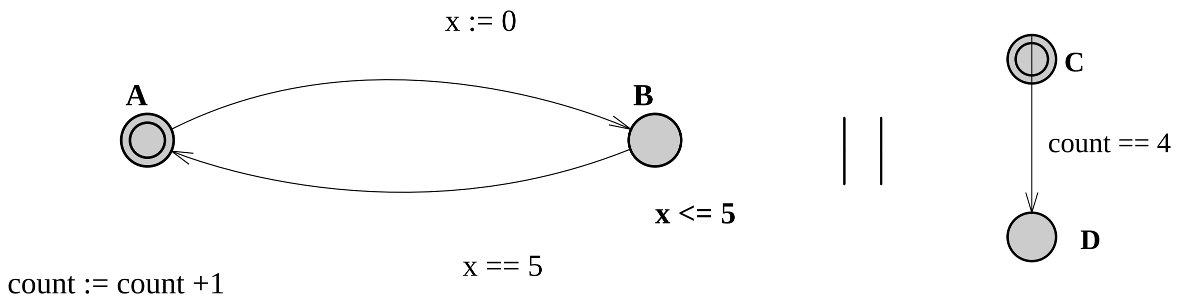
Model Checking

$$\boxed{M \overset{?}{\models} \varphi}$$

M	:	description of the system
φ	:	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<>` φ	reachability
`A[]` φ	safety (invariantly φ)
`E[]` φ	possibly always φ
`A<>` φ	inevitably φ
`A[]` $\varphi \Rightarrow$ `A<>` ψ	unbounded response

φ, ψ : propositional formula over locations and (existing) clocks

From (timed) Statecharts to UPPAAL

Rhapsody timed Statechart	$\longrightarrow$	*HTA model* $\boxed{M_H}$	$\longrightarrow$	*TA model* $\boxed{\textit{flatten}(M_H)}$
hierarchical model		TA-close hierarchy		MODEL-CHECK
informal description		formal semantics		formal semantics

From (timed) Statecharts to UPPAAL

Rhapsody timed Statechart $\longrightarrow$ *HTA model* $\boxed{M_H}$ $\longrightarrow$ *TA model* $\boxed{\textit{flatten}(M_H)}$

Rhapsody timed Statechart	*HTA model* M_H	*TA model* *flatten*(M_H)
hierarchical model	TA-close hierarchy	MODEL-CHECK
informal description	formal semantics	formal semantics

NORMALIZATION
simplification of data
(safe) omission of c-code

FLATTENING
auxiliary locations
auxiliary variables

Guiding Principle: Make it easy to adjust to small changes

Soundness &Correctness

Translations introduce slack. Thus

$$\boxed{M_H} \models \varphi \quad \not\Leftrightarrow \quad \boxed{\mathit{flatten}(M_H)} \models \mathit{flatten}(\varphi)$$

but

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

timed transition system | timed *flatten*(M) traces

$\downarrow$ give rise to | $\downarrow$ project to M_H

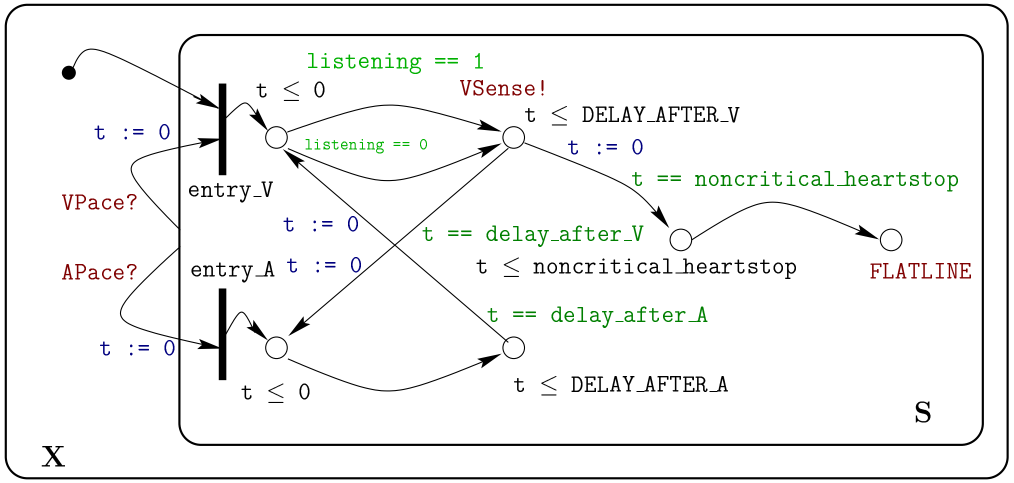
timed M_H traces *match* timed M_H traces

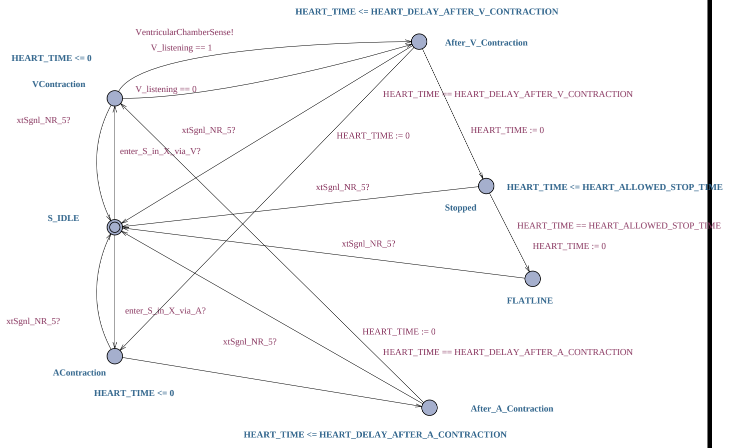
Outline of the Flattening

3 phases to flatten a hierarchical structure:

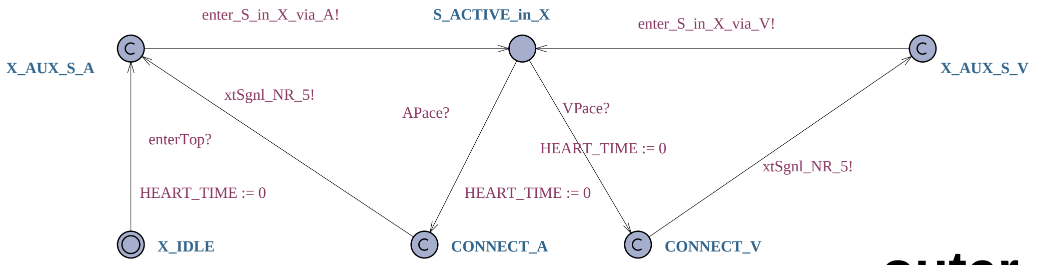
1. **Collect instantiations**
   every superstate becomes one (flat) timed automaton
2. **Compute global joins**
   mimic synchronization-on-exit in the the flat automata
   principle: use counters & and add threshold-guard
3. **Post-process channel communication**
   a transitions may not synchronize with its own superstate
   principle: duplicate channels & restrict scope

Example: Flattening the Model of a Human Heart



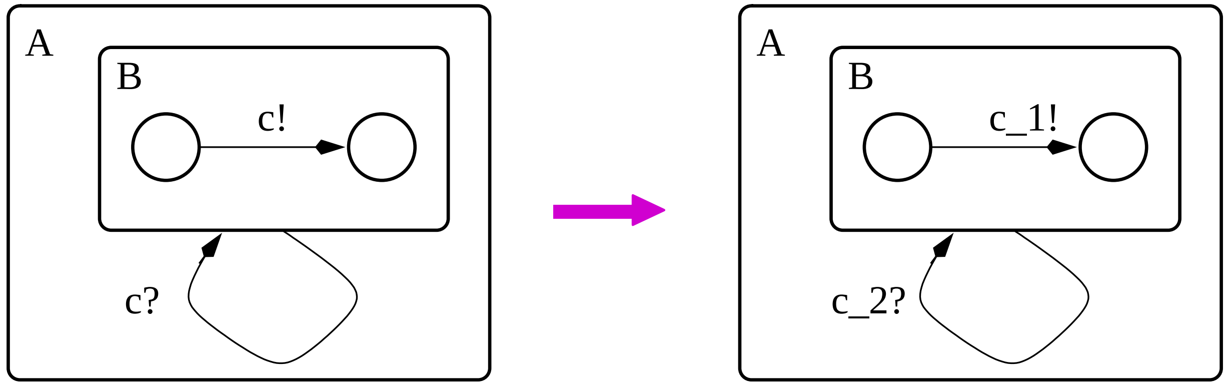


inner superstate



outer superstate

Communication Conflict



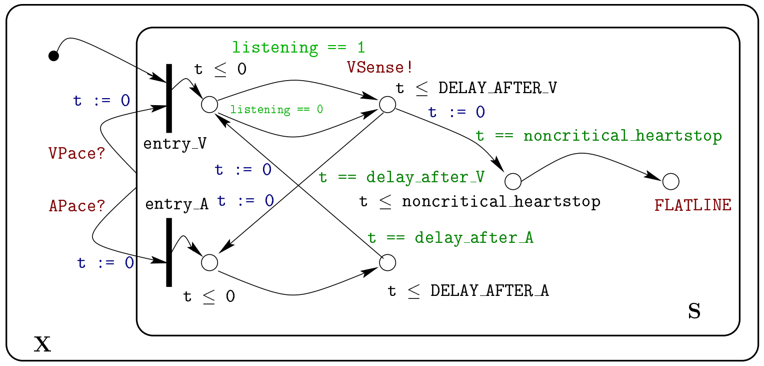
- cannot keep **c**
- cannot remove **c**

- rename **c** inside
- rename **c** outside
- modify other transitions:

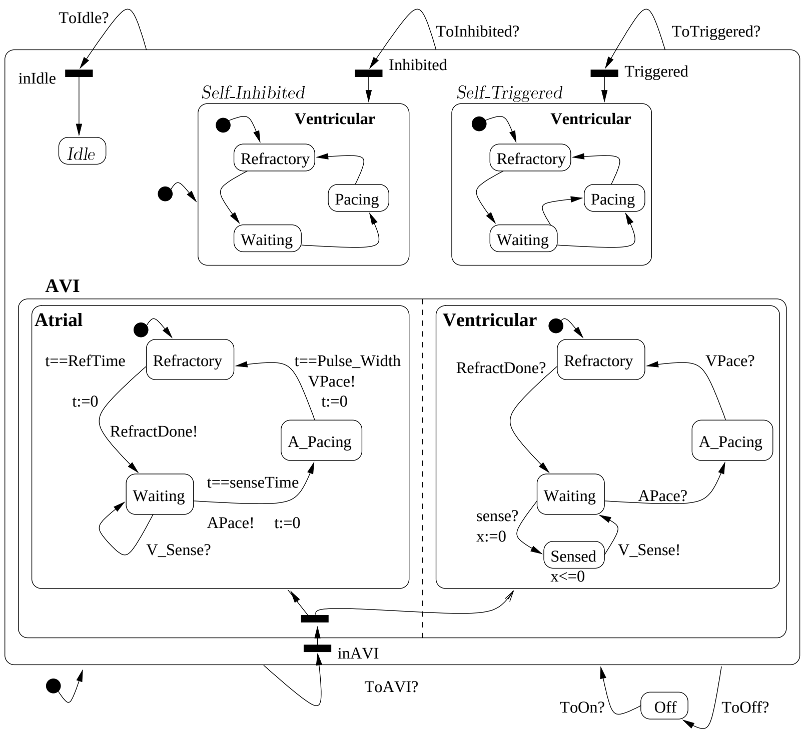
  either choose one of c_1, c_2

  or duplicate transition (allow both)

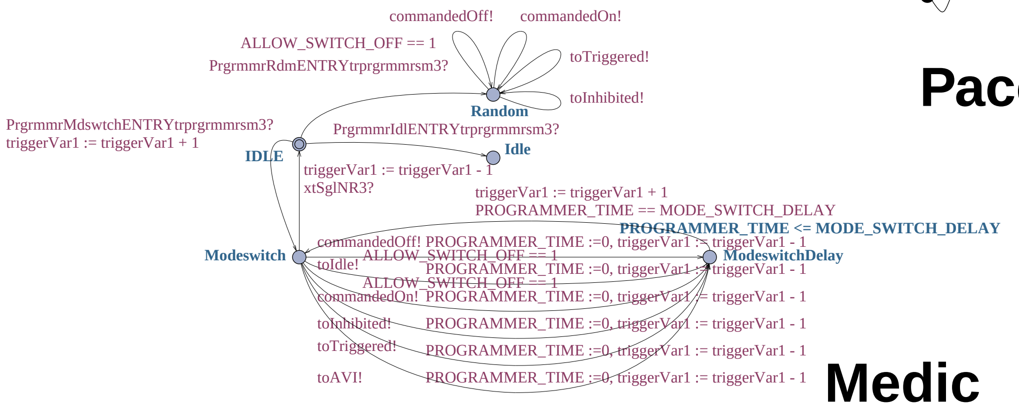
Model-Checking a Pacemaker



Human Heart



Pacemaker


commandedOff!
commandedOn!
ALLOW_SWITCH_OFF == 1
PrgrmmrRdmENTRYtrprgrmmrsm3?
toTriggered!
toInhibited!
Random
PrgrmmrMdswtchENTRYtrprgrmmrsm3?
triggerVar1 := triggerVar1 + 1
PrgrmmrIdlENTRYtrprgrmmrsm3?
IDLE
Idle
triggerVar1 := triggerVar1 - 1
xtSglNR3?
triggerVar1 := triggerVar1 + 1
PROGRAMMER_TIME == MODE_SWITCH_DELAY
PROGRAMMER_TIME <= MODE_SWITCH_DELAY
commandedOff! PROGRAMMER_TIME :=0, triggerVar1 := triggerVar1 - 1
Modeswitch
ALLOW_SWITCH_OFF == 1
ModeswitchDelay
toIdle!
PROGRAMMER_TIME :=0, triggerVar1 := triggerVar1 - 1
ALLOW_SWITCH_OFF == 1
commandedOn! PROGRAMMER_TIME :=0, triggerVar1 := triggerVar1 - 1
toInhibited! PROGRAMMER_TIME :=0, triggerVar1 := triggerVar1 - 1
toTriggered! PROGRAMMER_TIME :=0, triggerVar1 := triggerVar1 - 1
toAVI! PROGRAMMER_TIME :=0, triggerVar1 := triggerVar1 - 1

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[] ¬heart stops`
  only true for ’good’ medic
- LIVENESS:
  `A[] Vcontract => A<> Acontract`

Model-Checking the Pacemaker

- DEADLOCK:
  possible (if heart stops)
- SAFETY:
  `A[] ¬heart stops`
  only true for 'good' medic
- LIVENESS:
  `A[] Vcontract => A<> Acontract`

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 `MODE_SWITCH_DELAY   = 65`, `A[] ¬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

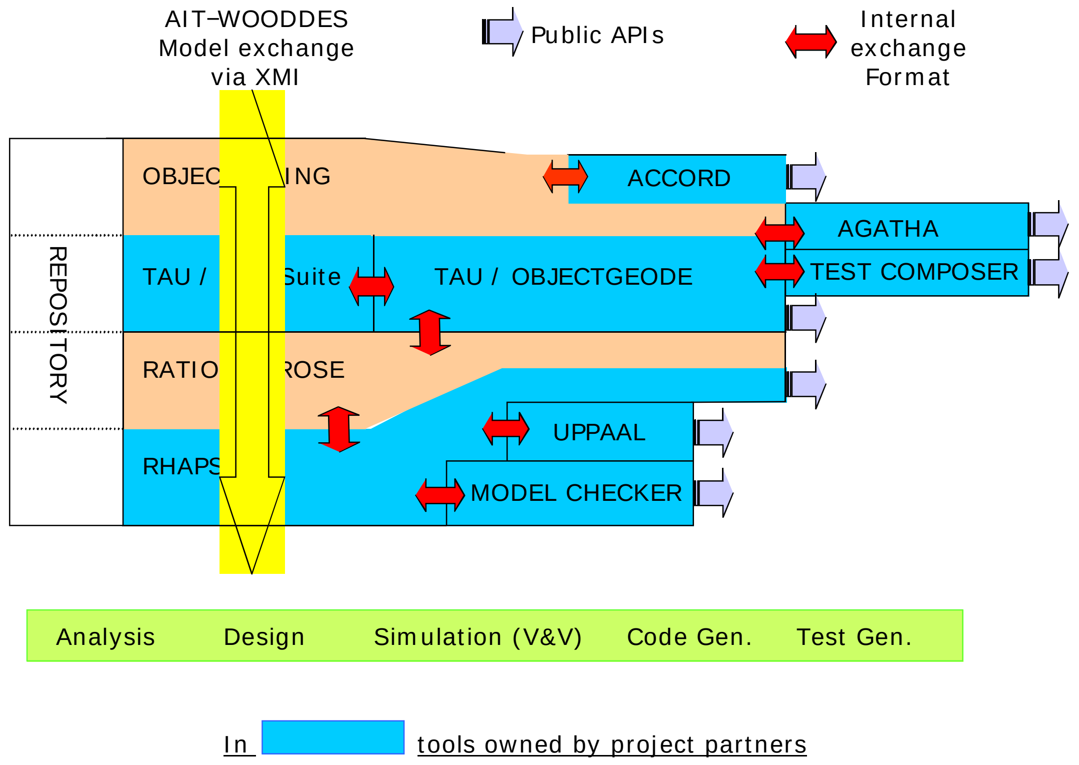
Workshop for Object-Oriented Design and Development of Embedded Systems

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*