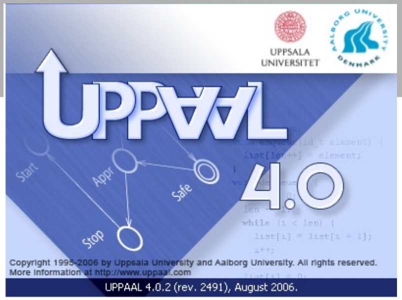
UPPAAL

Model Checking, Performance Analysis and Testing of Real Time Systems



Kim G. Larsen

CISS – Aalborg University

DENMARK

CISS – Center For Embedded Software Systems

Regional ICT Center (2002-)

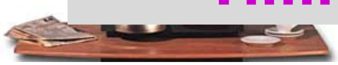
- 3 research groups
  - Computer Science
  - Control Theory
  - Hardware
  - Wireless Communication

- **20** Employed
- **25** Associated
- **20** PhD Students
- **50** Industrial projects
- **10** Elite-students
- **140+** MDKK

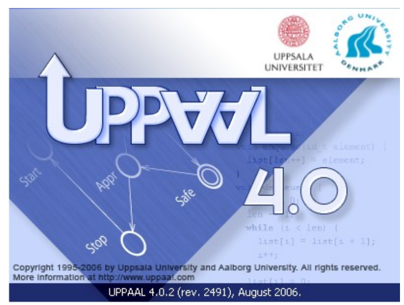
- ARTIST Design
- ARTEMIS
- ... ...

Characteristica:

- Dedicated function
- Complex environment
- SW/HW/Mechanics
- Networked
- Autonomous
- Ressource constrained
  : Energy
  : Bandwidth
  : Memory
  : ...
- **Timing constraints**



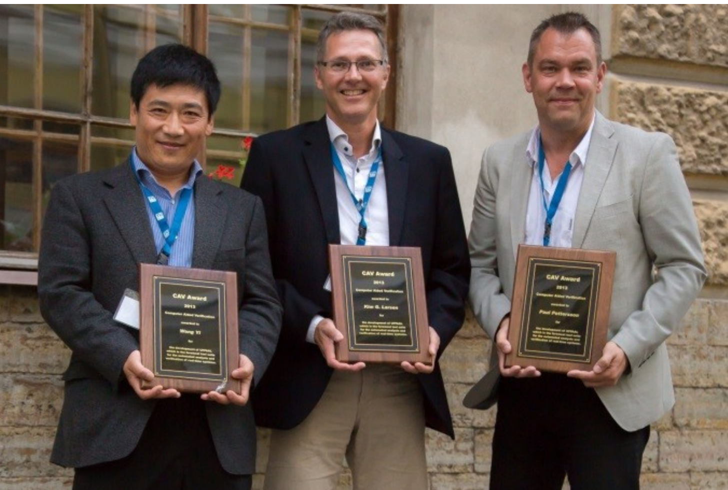
Model Checking &
Performance Analysis



UP4ALL

Origin of UPPAAL

TAU
CCS & Modal Transition Systems
Refinements
Modal Mu-Calculus
Explicit State Representation
Prolog



1995

UPPAAL
Timed Automata
TCTL
Zones
C++ & Java

2007

2013

EPSILON
TCCS
Timed Refinements
Timed Mu-Calculus
Regions
Prolog<

UP4ALL

CAV Award

Contributors

@UPPsala

- **Wang Yi**
- **Paul Pettersson**
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

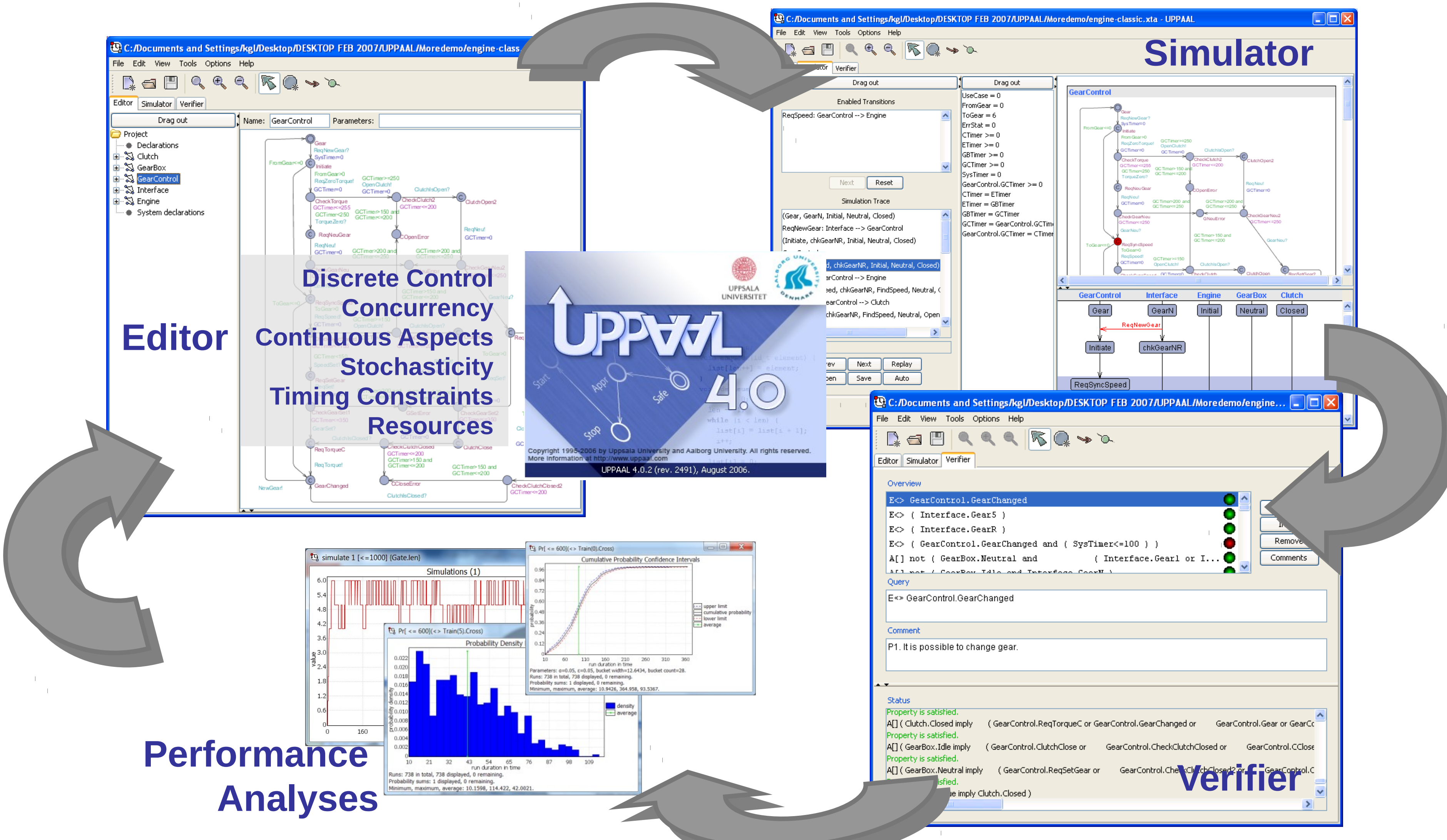
@AALborg

- **Kim G Larsen**
- **Alexandre David**
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Jacob I. Rasmussen
- Marius Mikucionis
- Thomas Chatain

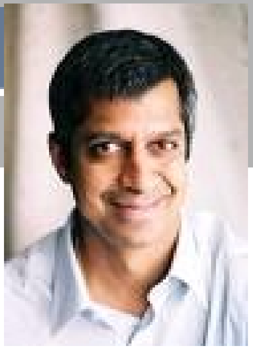
@Elsewhere

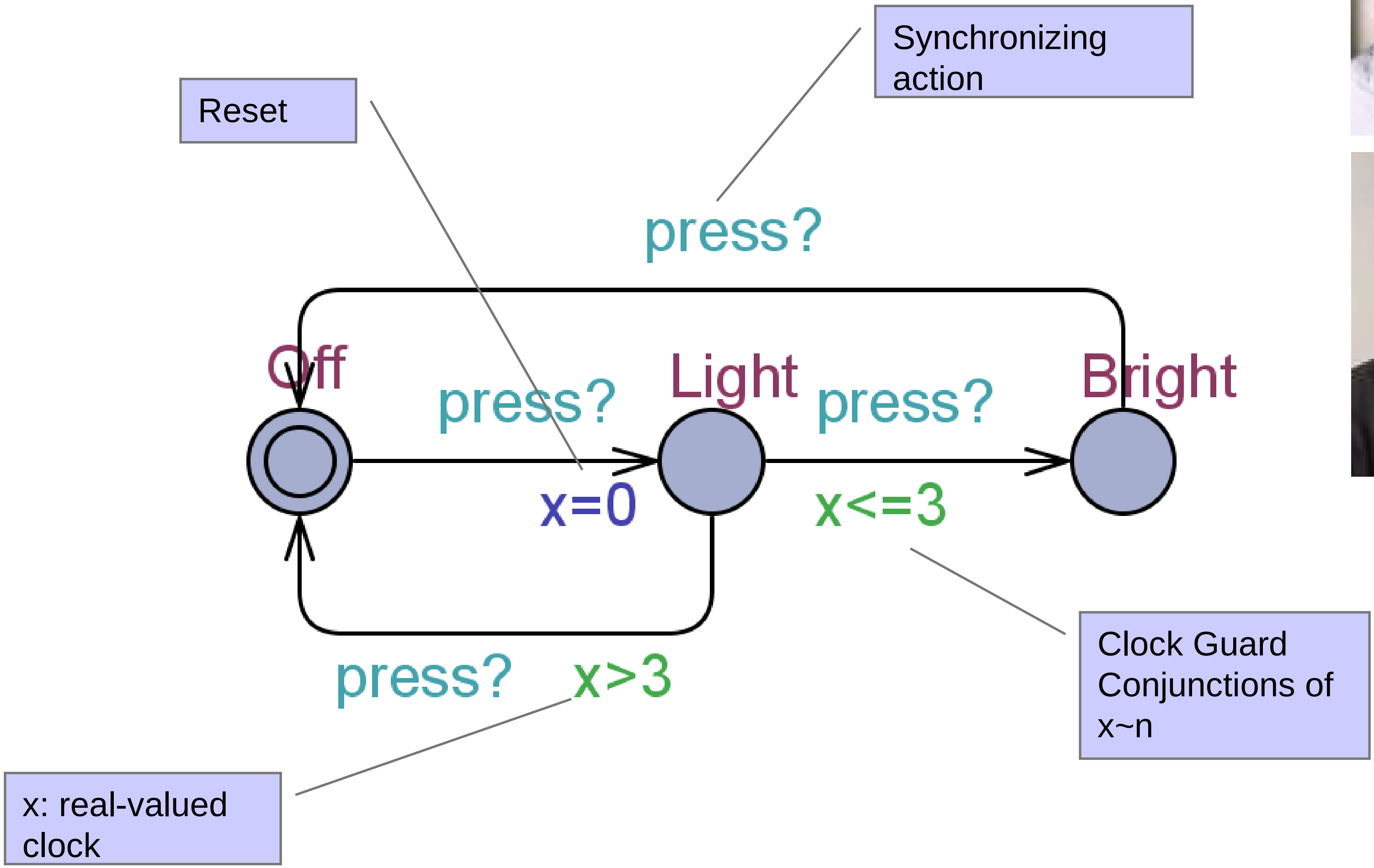
– Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...

UPPAAL Model Checker



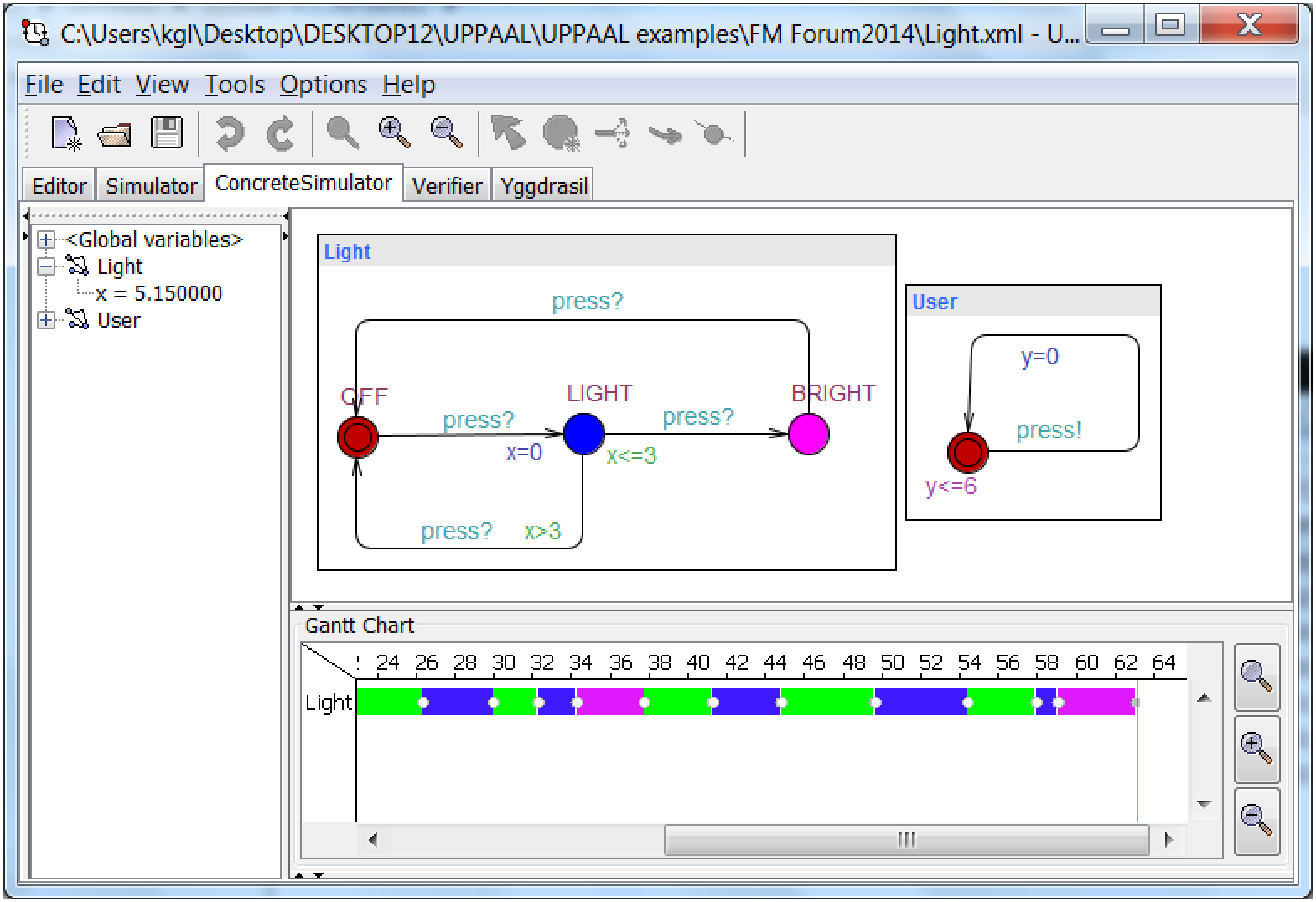
Timed Automata [Alur & Dill'89]



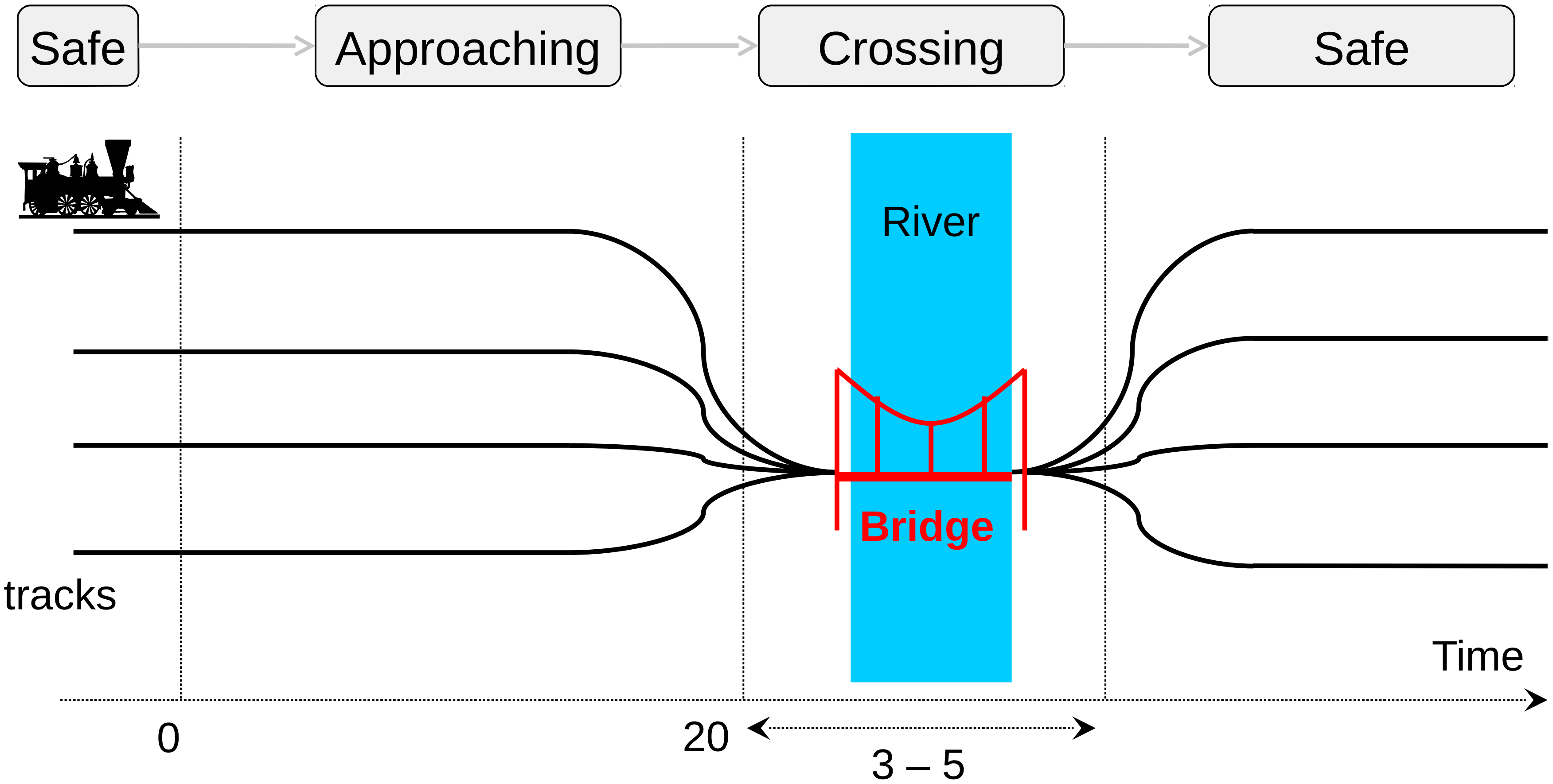


ADD a clock x

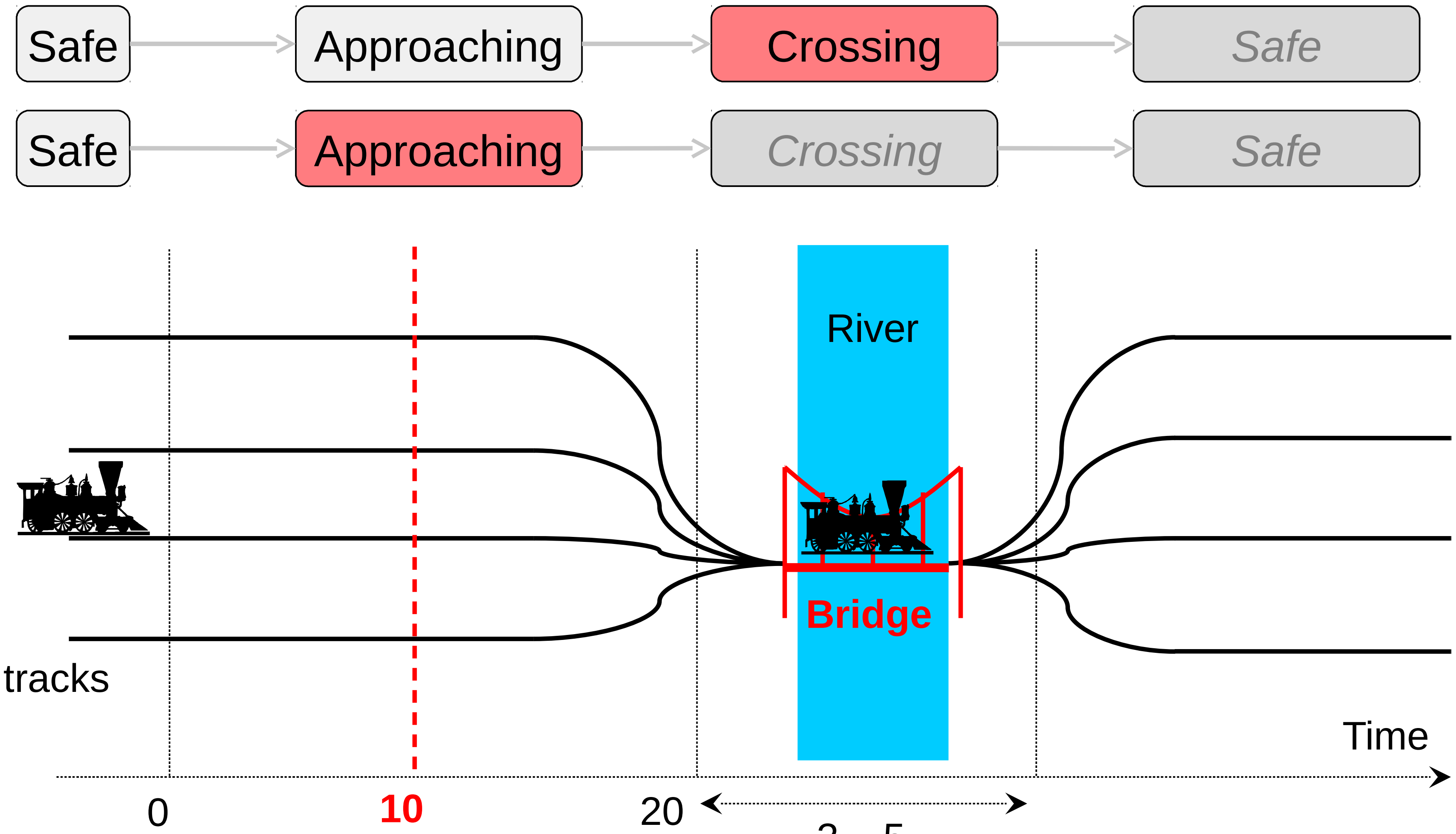
Semantics in UPPAAL



Train Crossing

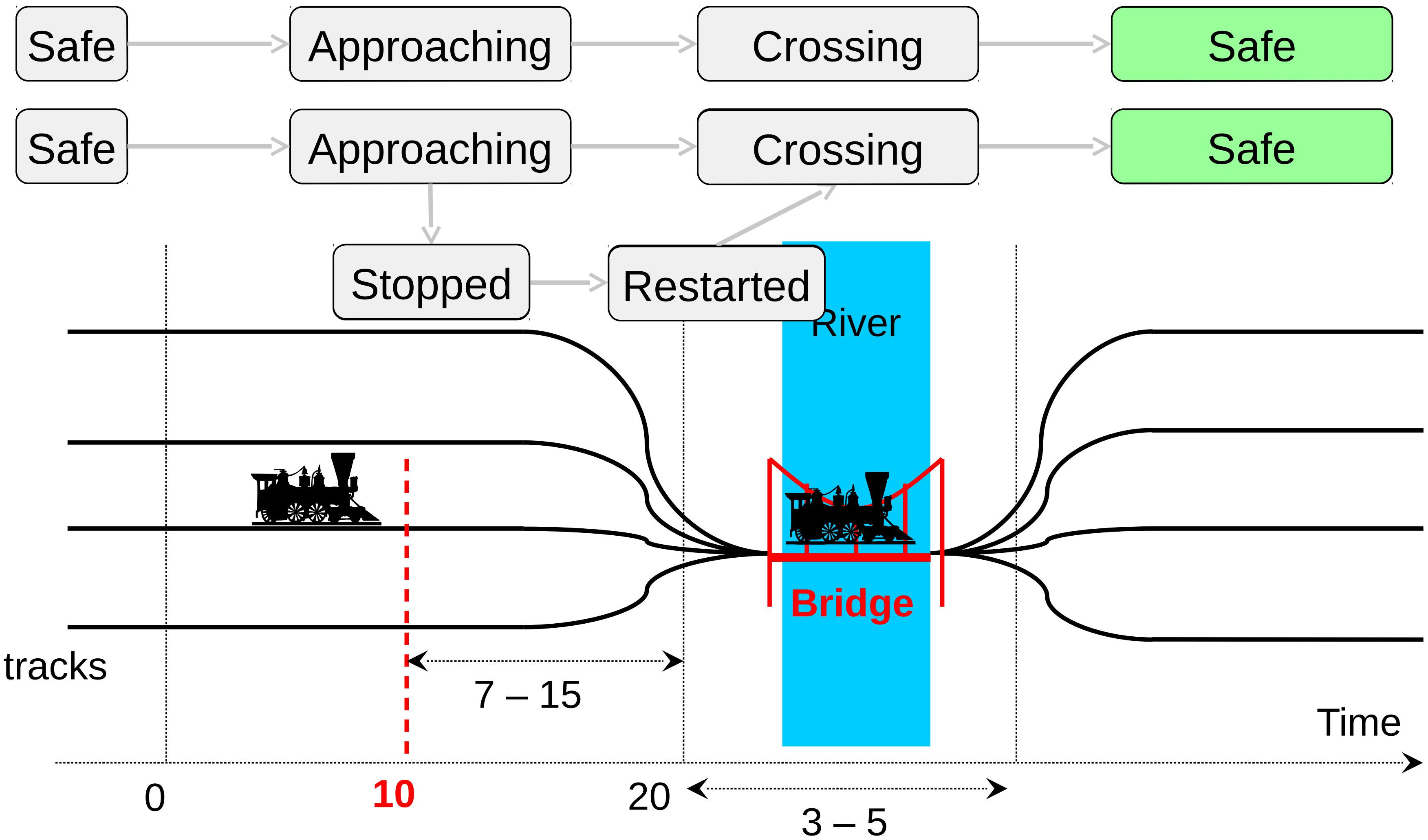


Train Crossing

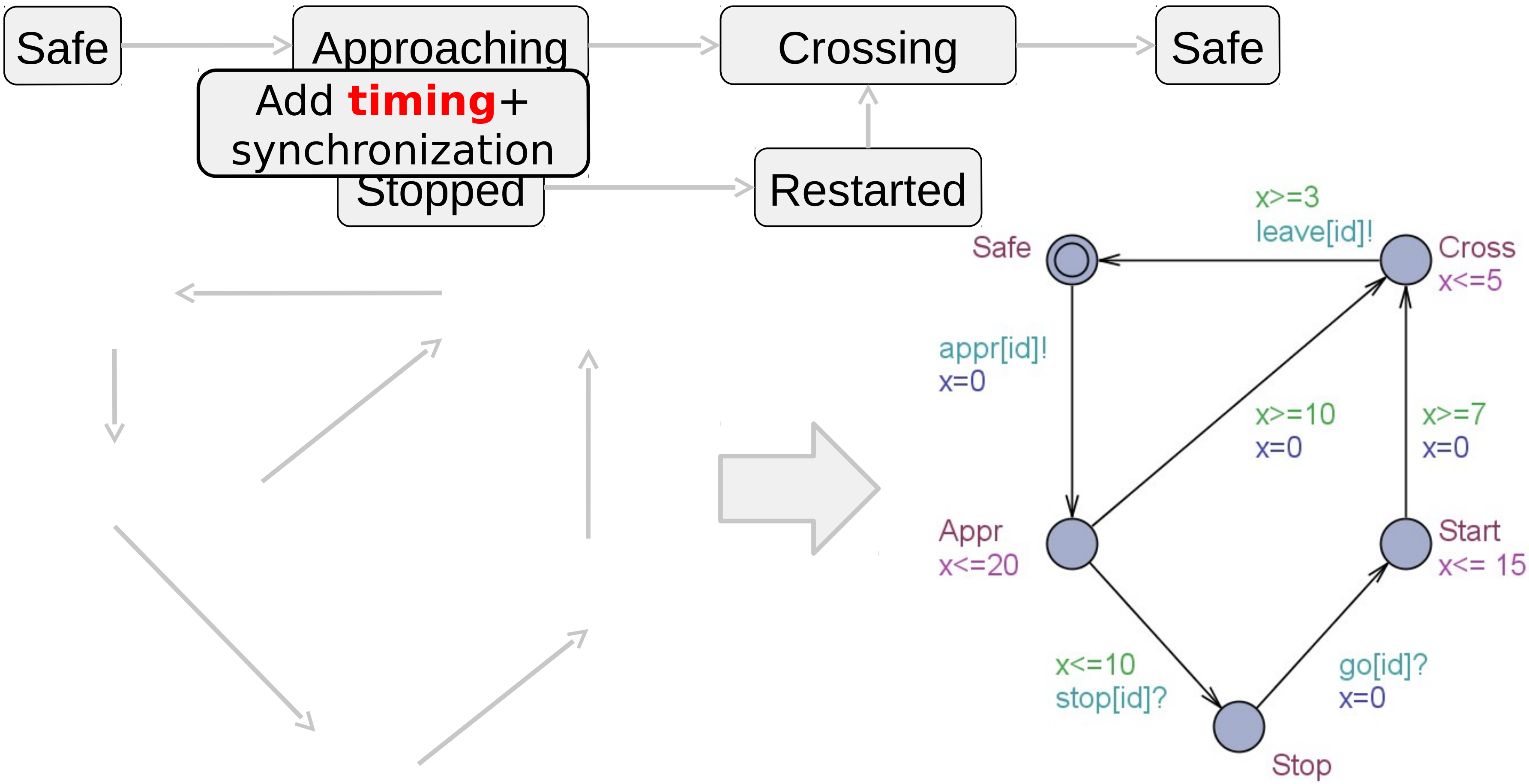


Stop the train while it still stoppable!

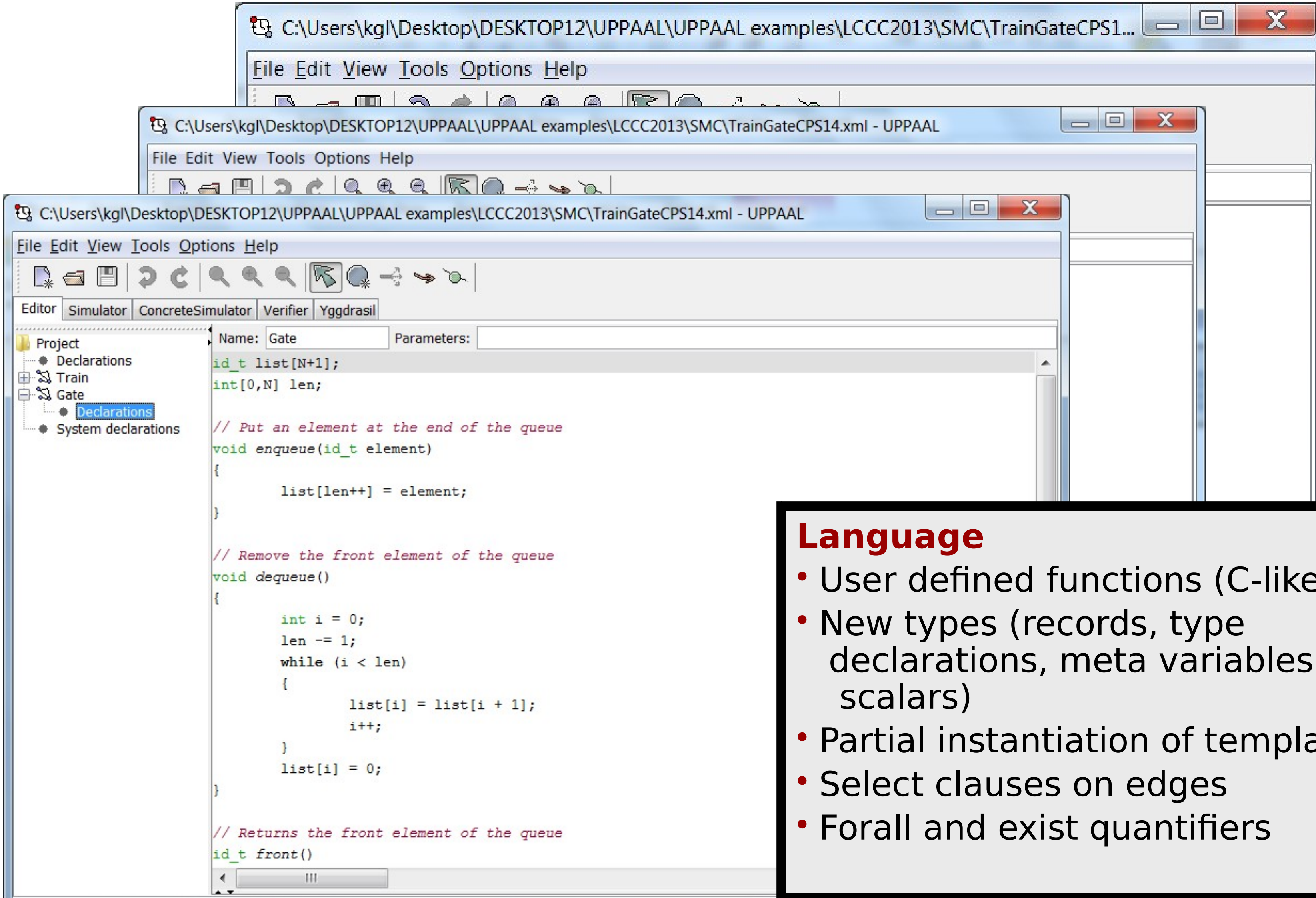
Train Crossing



Train Crossing



Editor



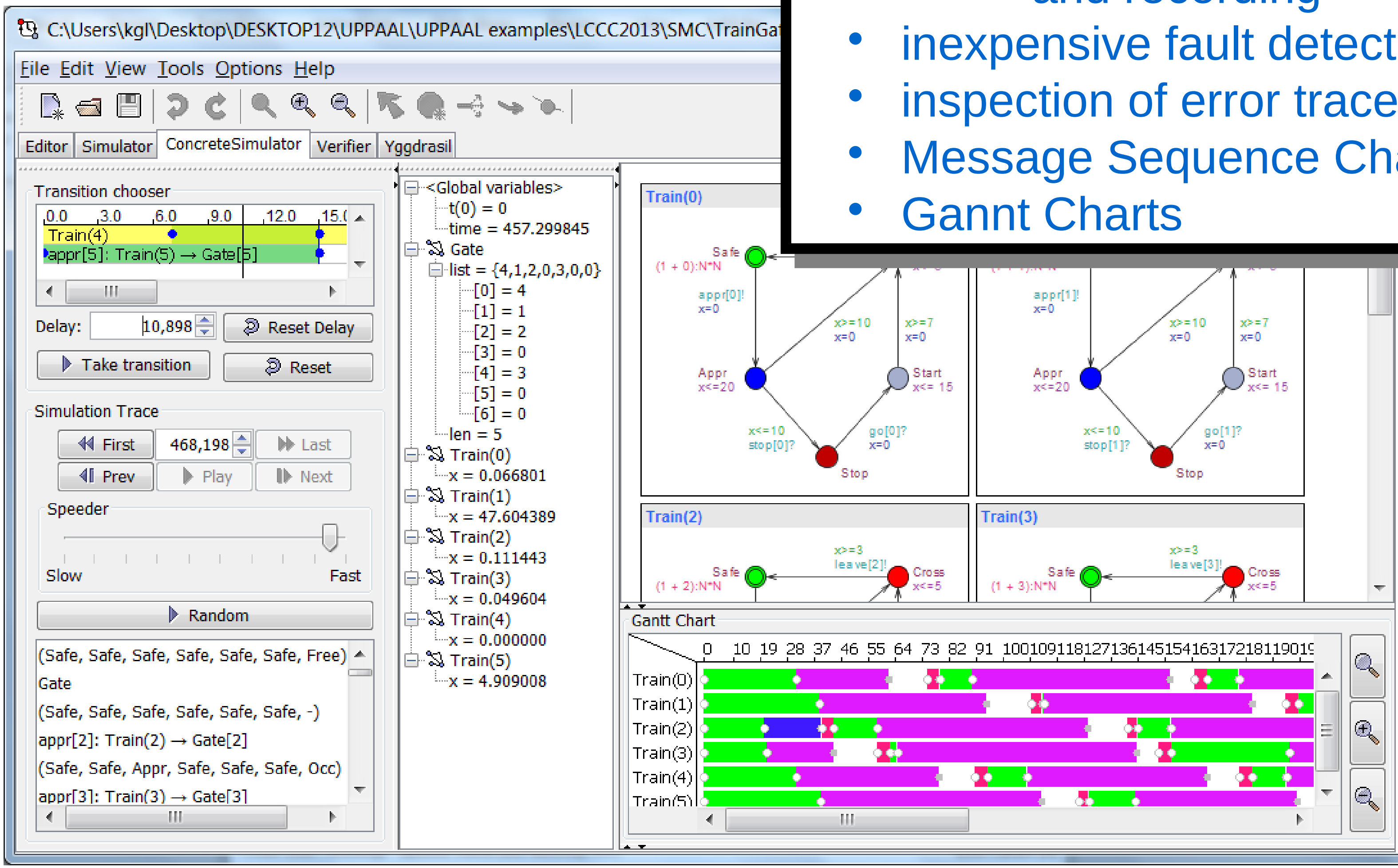
Language

- User defined functions (C-like)
- New types (records, type declarations, meta variables, scalars)
- Partial instantiation of templates
- Select clauses on edges
- Forall and exist quantifiers

Concrete Simulator

Graphical Simulator

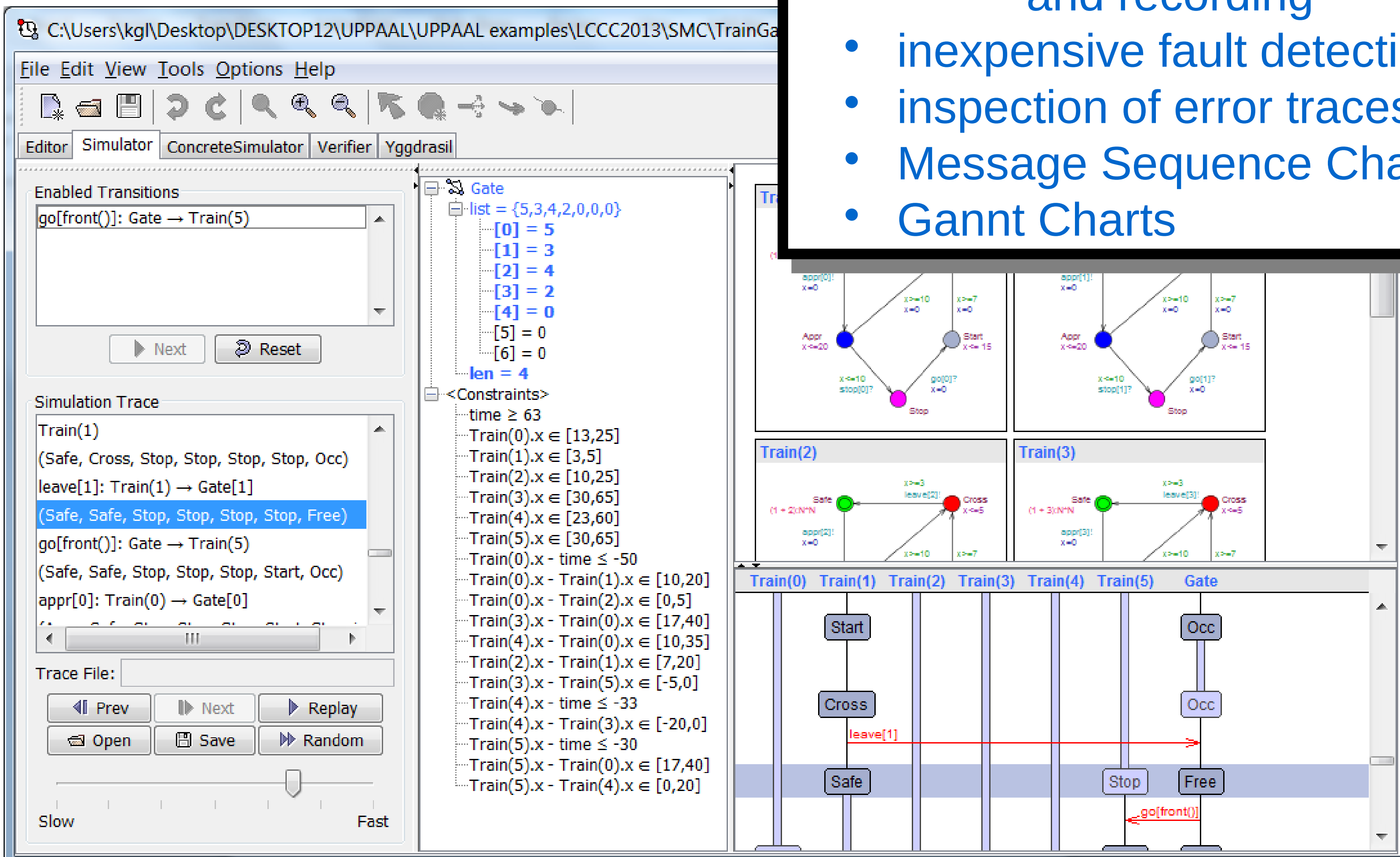
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gannt Charts

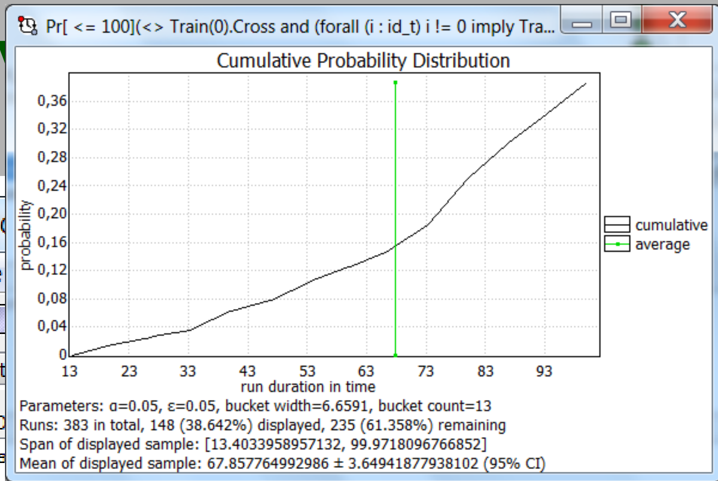


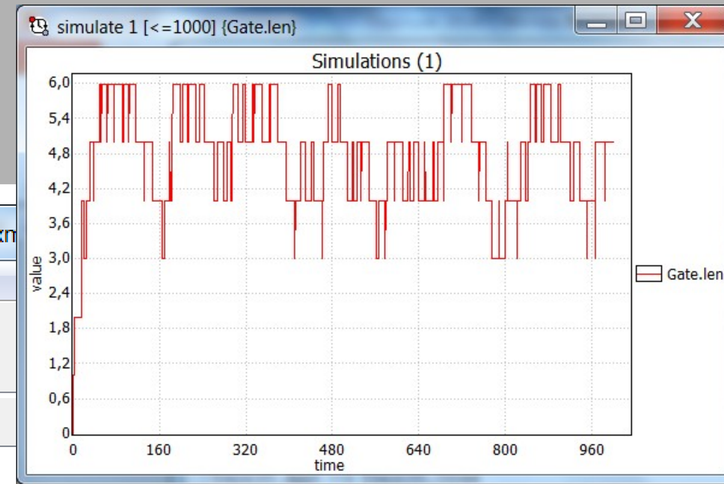
Symbolic Simulator

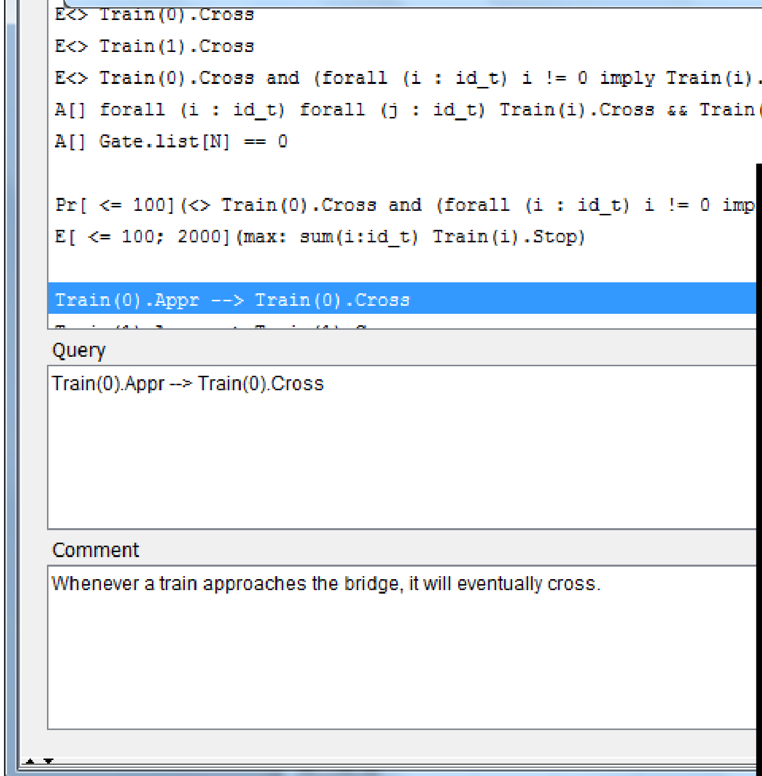
Graphical Simulator

- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gannt Charts





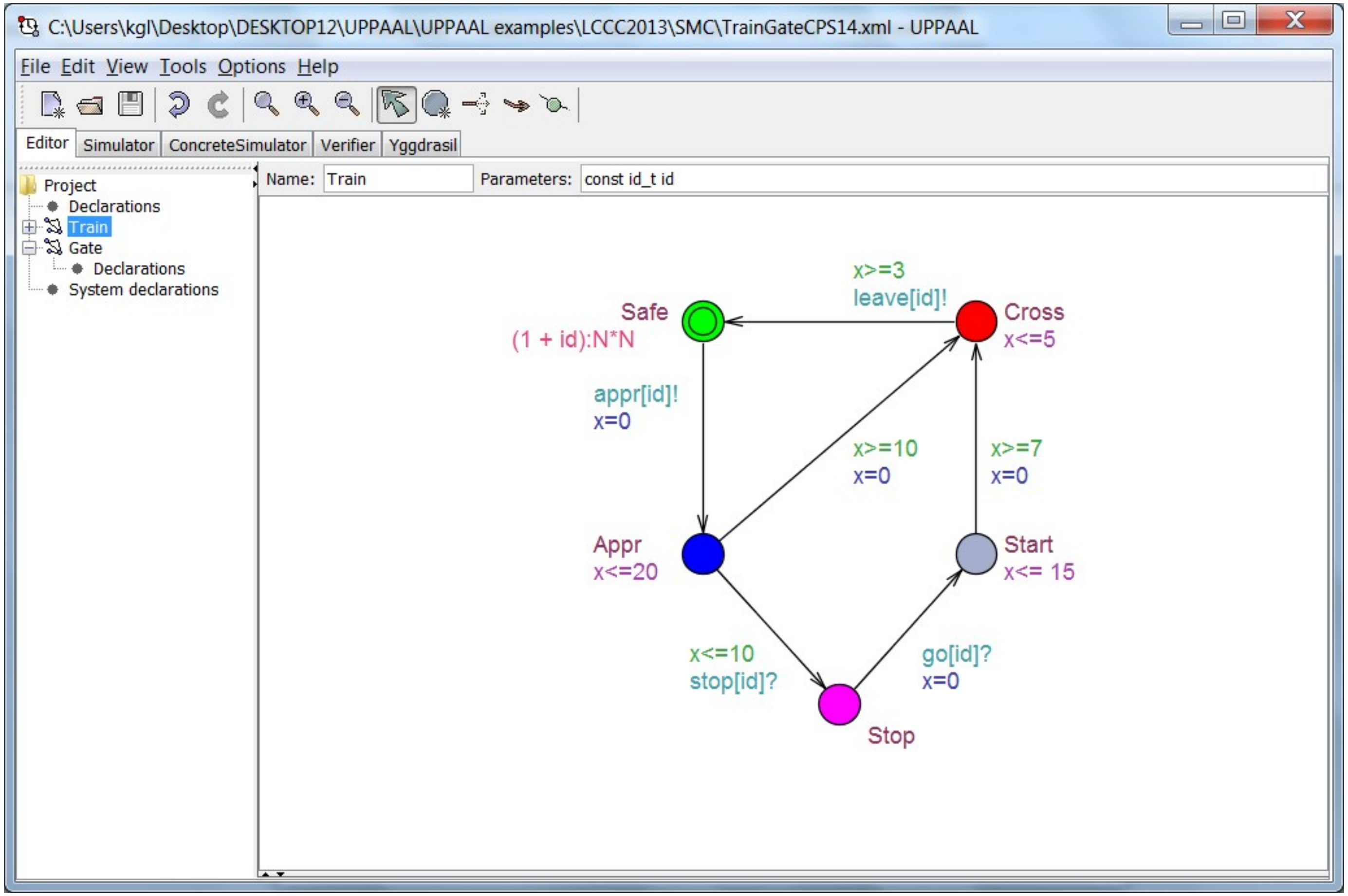




Verifier

- Exhaustive & automatic checking of requirements
- .. including validating, safety, liveness, bounded liveness and response properties
- .. performance properties, e.g probabilistic and expectation.
- .. generation of debugging information for visualisation in simulator.
- .. plot composer

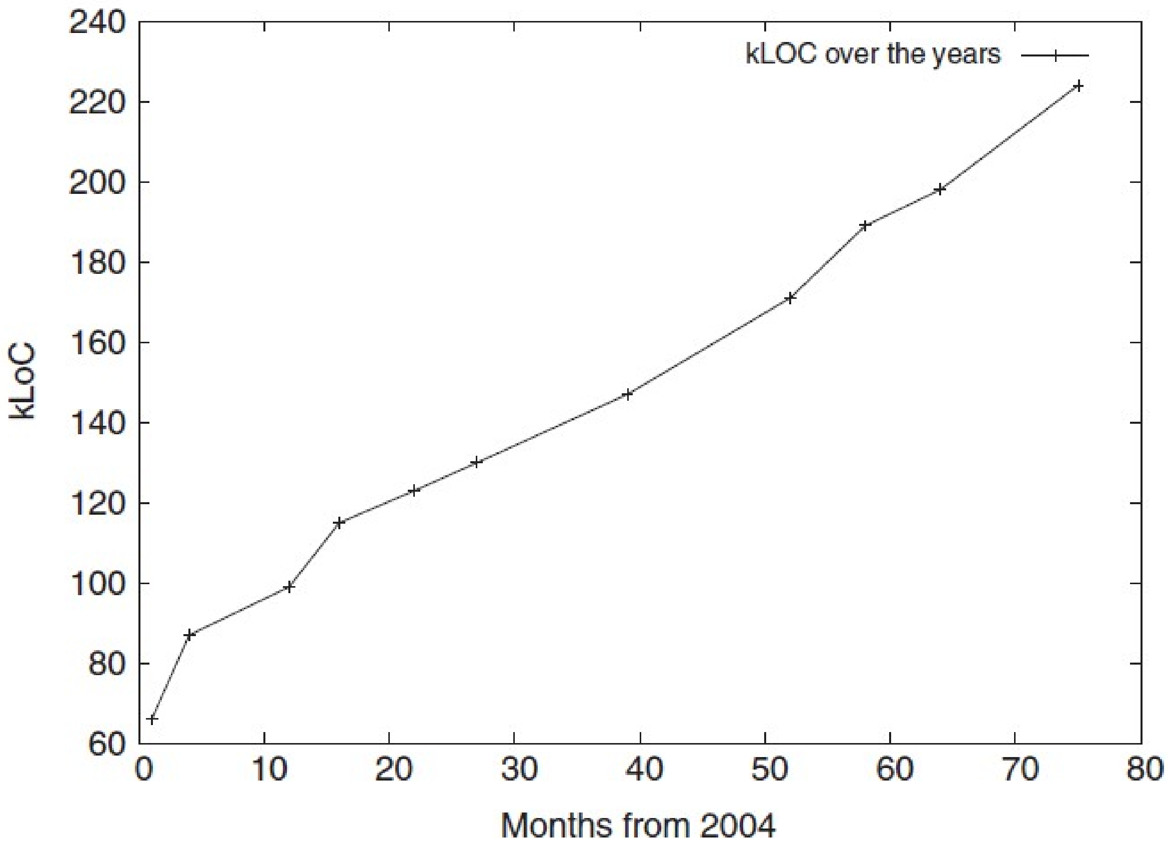
Demo


C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier Yggdrasil
Project
Declarations
Train
Gate
Declarations
System declarations
Name: Train
Parameters: const id_t id
x>=3
leave[id]!
Safe
(1 + id):N*N
Cross
x<=5
appr[id]!
x=0
x>=10
x=0
x>=7
x=0
Appr
x<=20
Start
x<= 15
x<=10
stop[id]?
go[id]?
x=0
Stop

Evolution of Performance

Version	CSMA5	CSMA7	CSMA12	Fischer5	Fischer7	Fischer12	HDDI7	HDDI12
3.0.39	*8.4* s	—	—	*4.2* s	—	—	*36.3* s	—
	7.2 MB	—	—	10.6 MB	—	—	20.1 MB	—
3.2.12	*0.3* s	*417* s	—	*1.6* s	—	—	*7.2* s	—
	3.8 MB	145 MB	—	6.8 MB	—	—	11.9 MB	—
3.3.25	*0.2* s	*198* s	—	*1.1* s	—	—	*3.2* s	—
	3.4 MB	113 MB	—	6 MB	—	—	8.4 MB	—
3.4.6	<*0.1* s	*40.7* s	—	*0.3* s	*4706* s	—	*0.1* s	*5.3* s
	3.1 MB	34.5 MB	—	4.9 MB	267 MB	—	1.6 MB	14.1 MB
4.0.11	<*0.1* s	*0.2* s	*33.8* s	<*0.1* s	*0.4* s	*418* s	<*0.1* s	*0.4* s
	1.6 MB	38 MB	115 MB	1.6 MB	38.1 MB	300 MB	1.6 MB	38 MB
4.1.2	<*0.1* s	*0.2* s	*41.9* s	<*0.1* s	*0.3* s	*341* s	*0.05* s	*0.2* s
	1.6 MB	21.6 MB	99 MB	21 MB	21.6 MB	248 MB	1.6 MB	22.9 MB

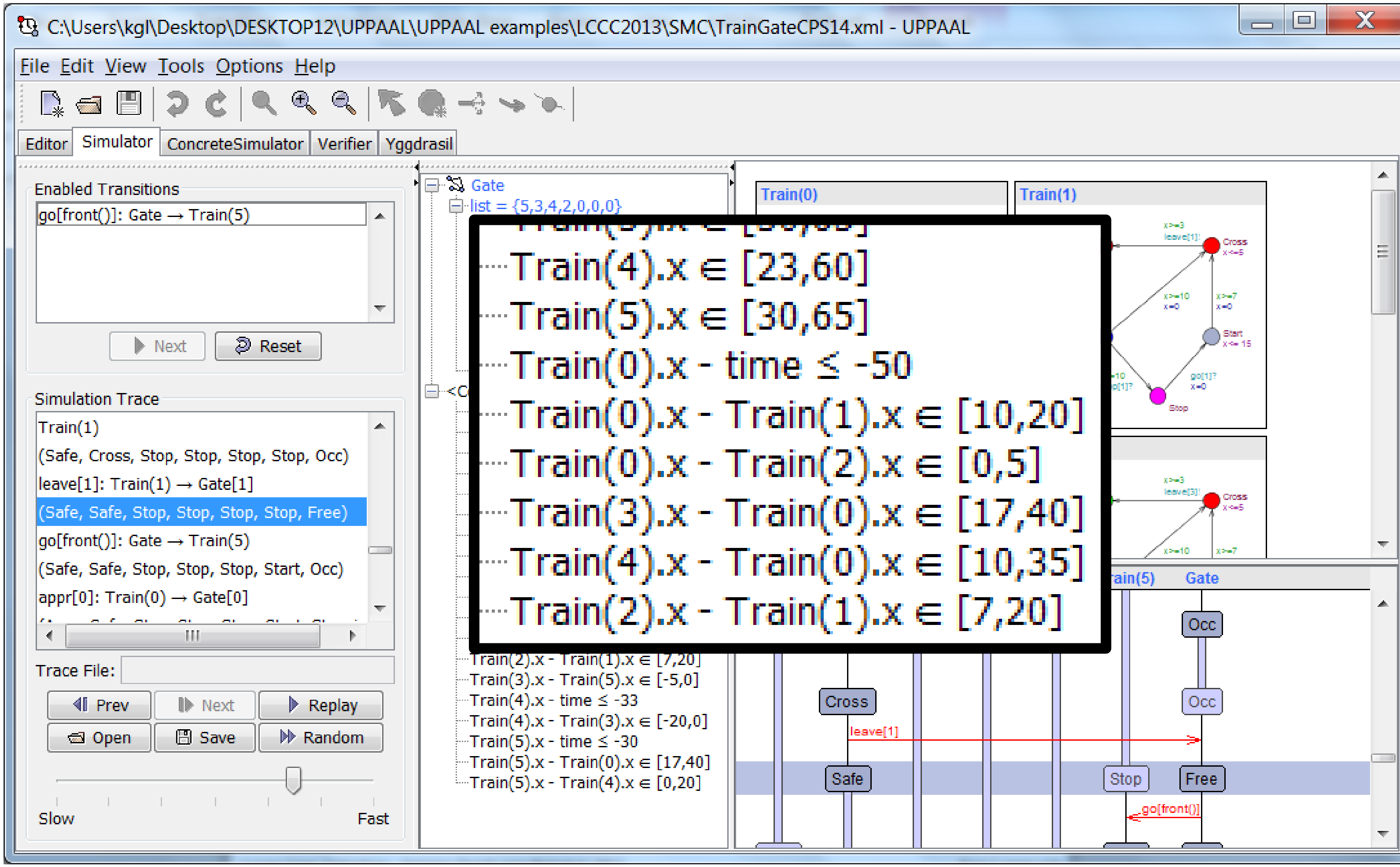
Evolution of Code Base



Client-Server Architecture
GUI: Java
Engine: C++
Platforms:
Linux, MacOS, Solaris, Windows

3 major cycles.

THE "secret" of UPPAAL



Zones & DBMs
THE "secret" UPPAAL

RELATED SITES: UPPAAL

UPPAAL DBM Library

The library used to manipulate DBMs in UPPAAL

Main Page | **Download** | **Ruby Binding** | **Help** | **Contact us**

Welcome!

DBMs [dill89, rokicki93, lpw:fct95, bengtsson02] are efficient data structures to represent clock constraints in timed automata [ad90]. They are used in UPPAAL [lpy97, by04, bdl04] as the core data structure to represent time. The library features all the common operations such as up (delay, or future), down (past), general updates, different extrapolation functions, etc.. on DBMs and federations. The library also supports subtractions. The API is in C and C++. The C++ part uses active clocks and hides (to some extent) memory management.

References

- [dill89] David L. Dill. *Timing Assumptions and Verification of Finite-State Concurrent Systems.* LNCS 407. Springer Berlin 1989, pp 197-212.
- [rokicki93] Tomas Gerhard Rokicki. *Representing and Modeling Digital Circuits.* Ph.D. thesis, Standford University 1993.
- [lpw:fct95] Kim G. Larsen, Paul Pettersson, and Wang Yi. *Model-Checking for Real-Time Systems.* Fundamentals of Computation Theory 1995, LNCS 965 pages 62-88.
- [bengtsson02] Johan Bengtsson. *Clocks, DBM, and States in Timed Systems.* Ph.D. thesis, Uppsala University 2002.
- [ad90] Rajeev Alur and David L. Dill. *Automata for Modeling Real-Time Systems.* International Colloquium on Algorithms, Languages, and Programming 1990, LNCS 443 pages 322-335.
- [lpy97] Kim G. Larsen, Paul Pettersson, and Wang Yi. *UPPAAL in a Nutshell.* International Journal on Software Tools for Technology Transfer , October 1997, number 1-2 pages 134-152.
- [by04] Johan Bengtsson and Wang Yi. *Timed Automata: Semantics, Algorithms and Tools.* Concurrency and Petri Nets 2004, LNCS 3098

Latest News

Draft manual available.
23 Oct 2006

The current work-in-progress manual of the DBM library is now available for download. It is still incomplete but it may be useful so we release it.

Ruby binding version 0.7 released
30 Jun 2006

Added displaying of points in the viewer. Re-compiled against 2.0.5.

Version 2.0.5 released
30 Jun 2006

Fixed bug in getValuation, added hooks to mingraph from the C++ API, drastically improved partition_t, improved subtractions, added a hasZero method, new print format of DBMs and federations to be more compatible with the Ruby binding, fixed doxygen

Done

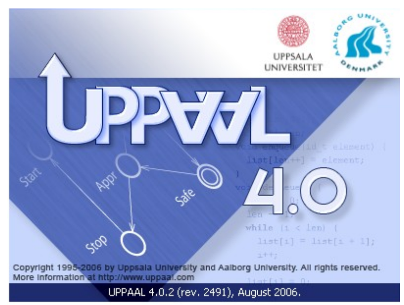
True

UPPAAL as a back-end

- Vooduu: verification of object-oriented designs using Uppaal, 2004.
- Moby/RT: A Tool for Specification and Verification of Real-Time Systems, 2000.
- Formalising the ARTS MPSOC Model in UPPAAL, 2007
- Marte UML ✉ UPPAAL , 2003.
- Yggdrasil: Statechart ✉ UPPAAL, 2003
- Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
- Verification of COMDES-II Systems Using UPPAAL with Model Transformation, 2008
- METAMOC: Modular WCET Analysis Using UPPAAL, 2010.
- … …

Industrial Usage

some examples



Bang & Olufsen (1997)

- Bug known to exist for 10 years
- Ill-described:

  2.800 loc +
  3 flowchart +
  1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.
- Follow-up project.

Arne Skou, Klaus Havelund

Bang & Olufsen (2001)

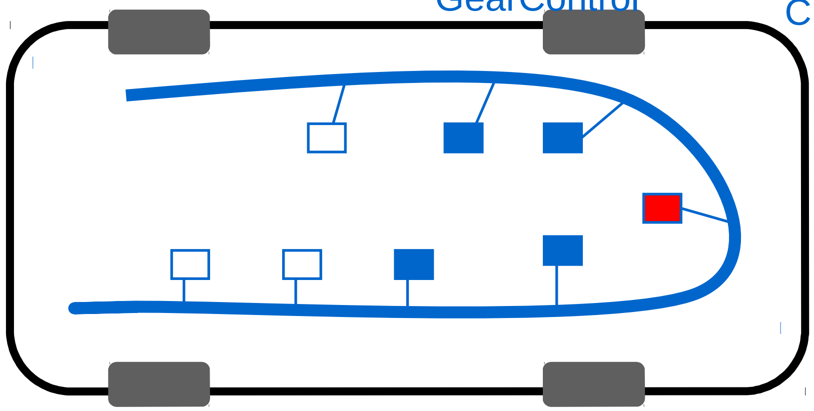
Conclusions

- It *is* possible to make a feasible abstraction of the existing system
- Bugs were found during model development and simulation
- A timing problem was identified during model checking. B&O changed their desing to remove the problem
- Time slicing and interrupt priorities can be modelled by timed automatons
- B&O obtained more confidence in the design before starting their implementation work. The design was robust in the sense that it did not have to be changed during the implementation phase

MECEL AB (1998)

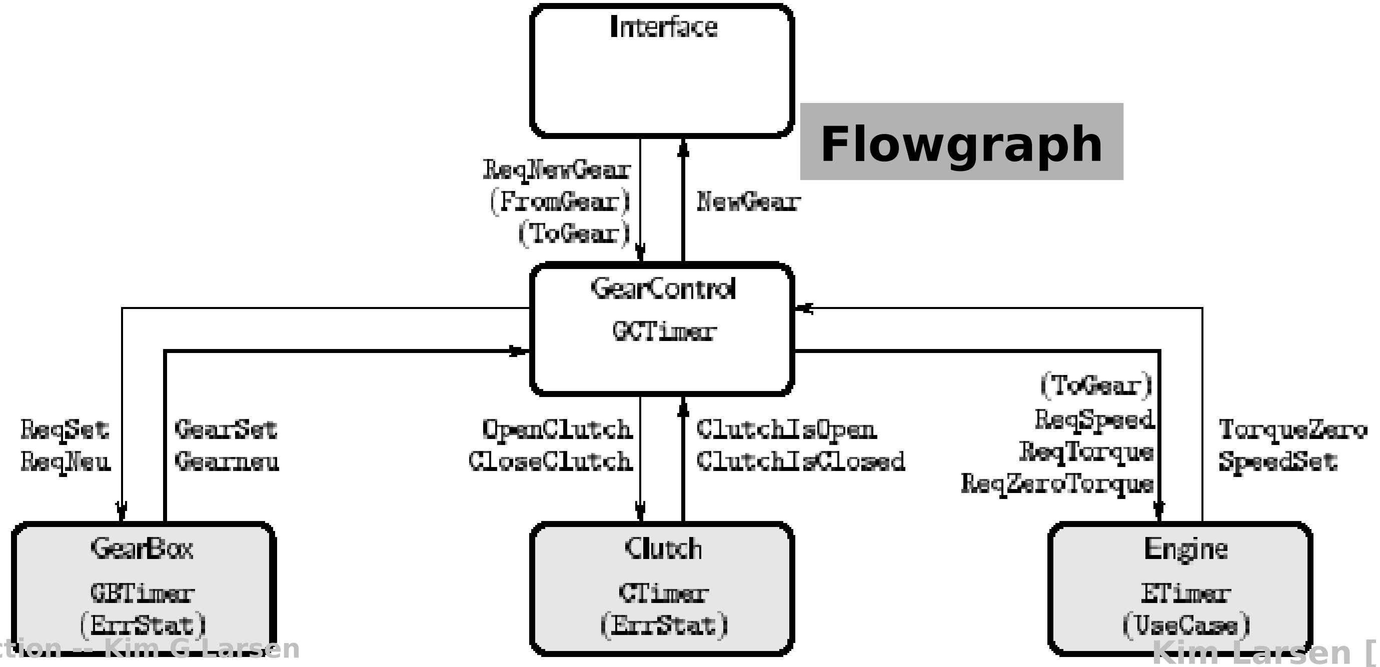
Gear Controller

Lindahl, Pettersson, Yi 1998



Paul Pettersson

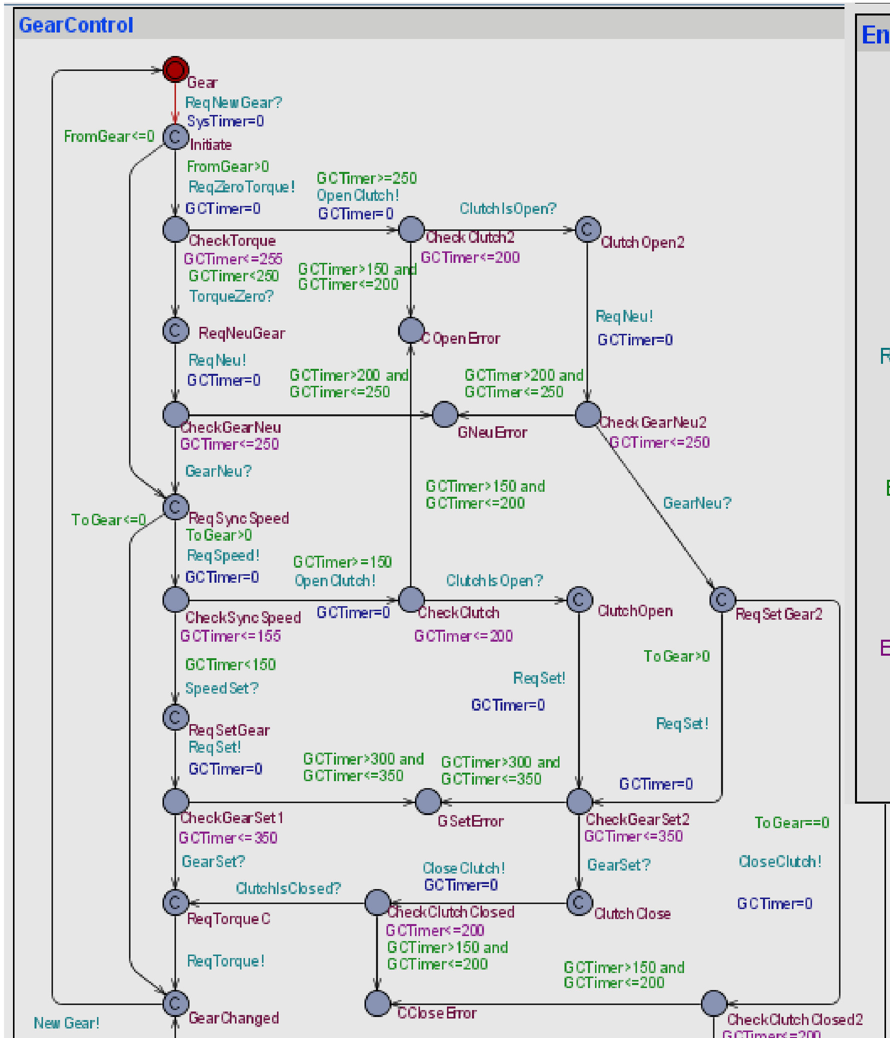
Flowgraph

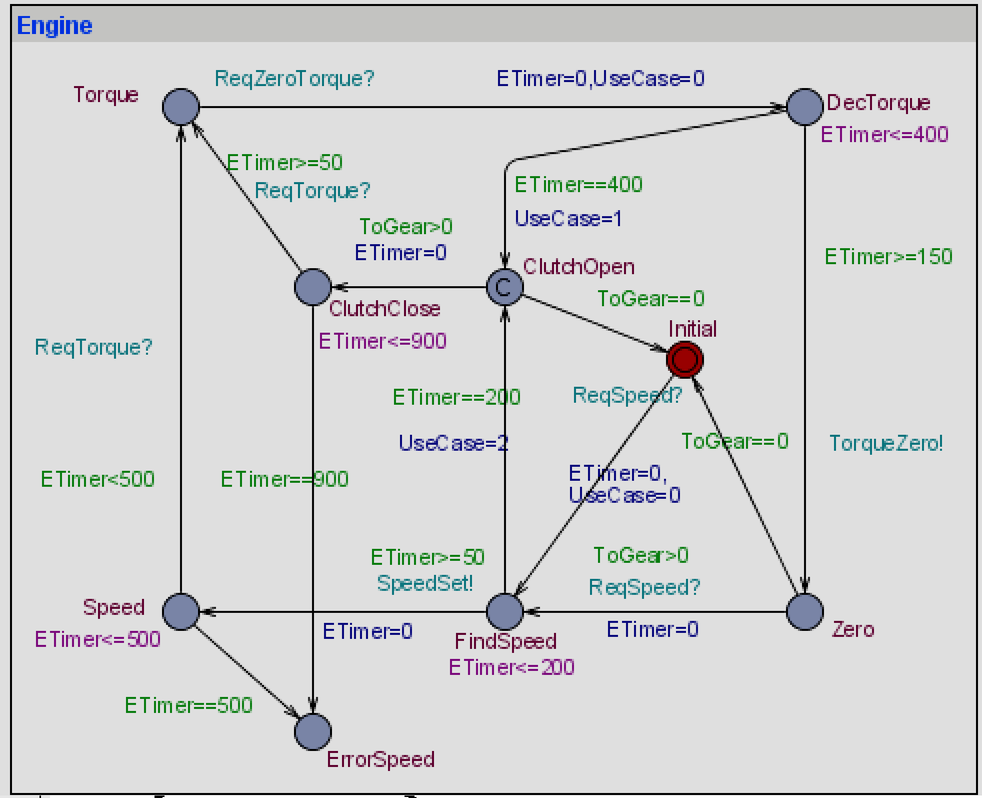


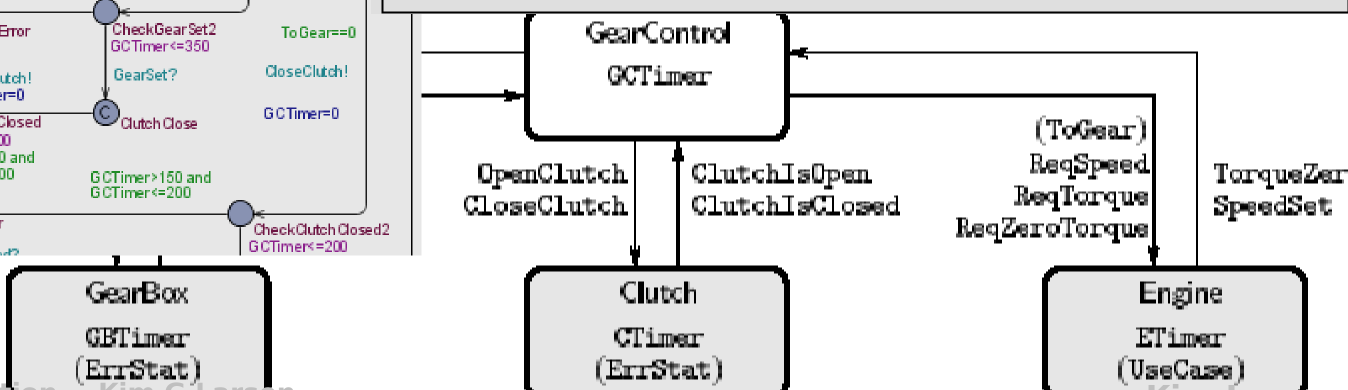
MECEL AB (1998)

Gear Controller

Lindahl, Pettersson, Yi 1998



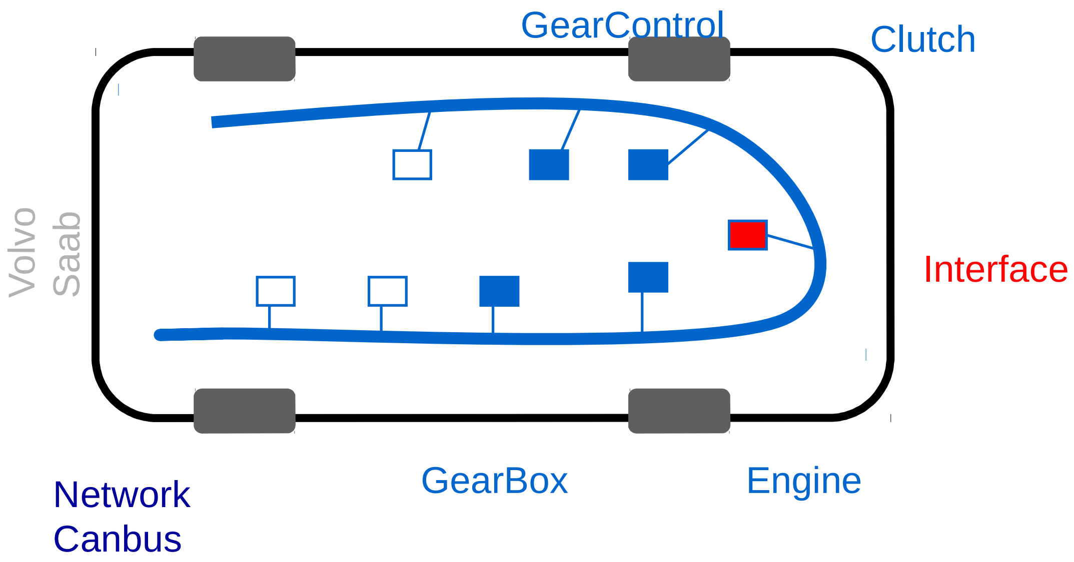


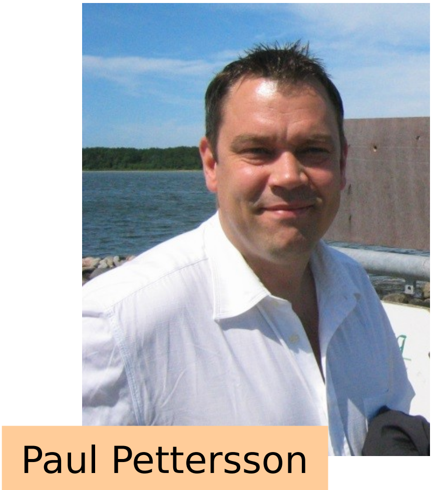


MECEL AB (1998)

Gear Controller

Lindahl, Pettersson, Yi 1998





$$\mathsf{GearControl@Initiate} \leadsto_{\leq 1500} ((\mathtt{ErrStat} = 0) \Rightarrow \mathsf{GearControl@GearChanged}) \quad (1)$$

$$\mathsf{GearControl@Initiate} \leadsto_{\leq 1000} ((\mathtt{ErrStat} = 0 \wedge \mathtt{UseCase} = 0) \Rightarrow \mathsf{GearControl@GearChanged}) \quad (2)$$

$$\mathsf{Clutch@ErrorClose} \leadsto_{\leq 200} \mathsf{GearControl@CCloseError} \quad (3)$$

$$\mathsf{Clutch@ErrorOpen} \leadsto_{\leq 200} \mathsf{GearControl@COpenError} \quad (4)$$

$$\mathsf{GearBox@ErrorIdle} \leadsto_{\leq 350} \mathsf{GearControl@GSetError} \quad (5)$$

$$\mathsf{GearBox@ErrorNeu} \leadsto_{\leq 200} \mathsf{GearControl@GNeuError} \quad (6)$$

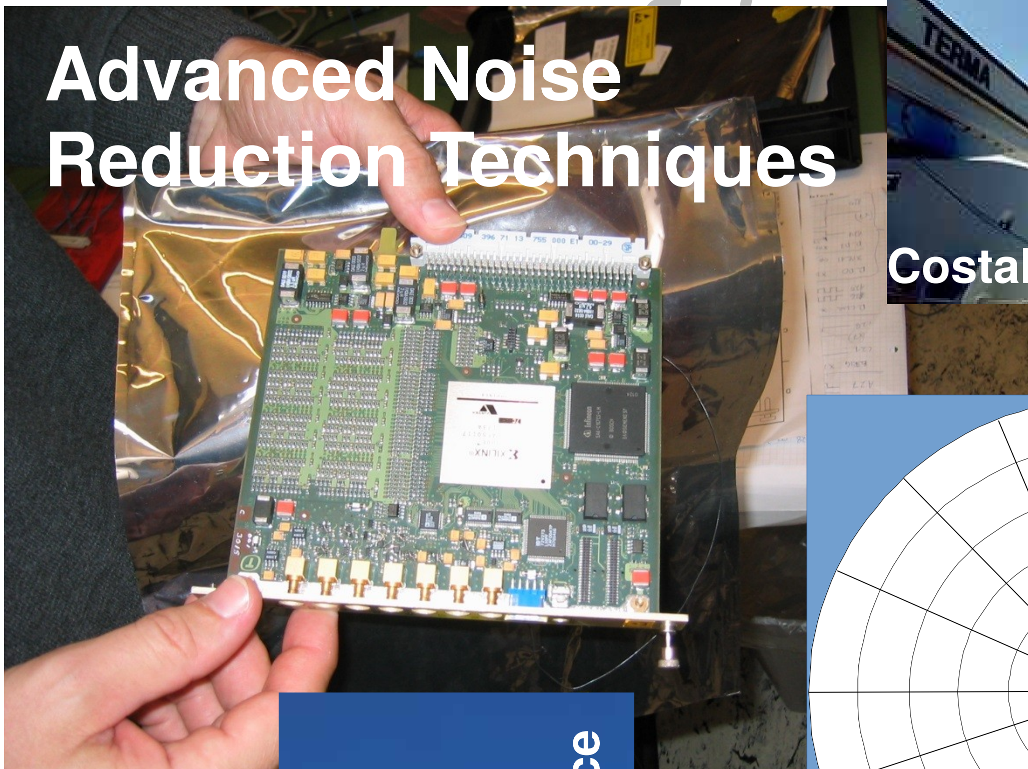
$$Inv\,(\mathsf{GearControl@CCloseError} \Rightarrow \mathsf{Clutch@ErrorClose}) \quad (7)$$

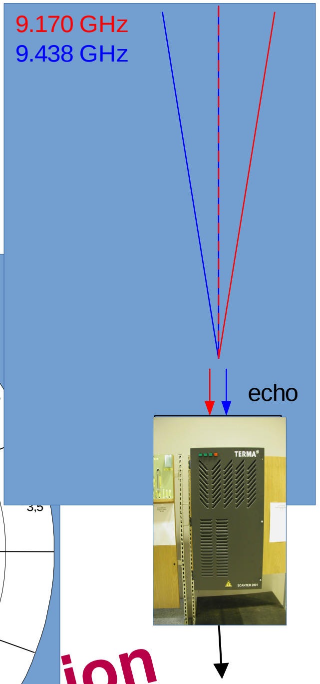
$$Inv\,(\mathsf{GearControl@COpenError} \Rightarrow \mathsf{Clutch@ErrorOpen}) \quad (8)$$

TERMA A/S (2004)

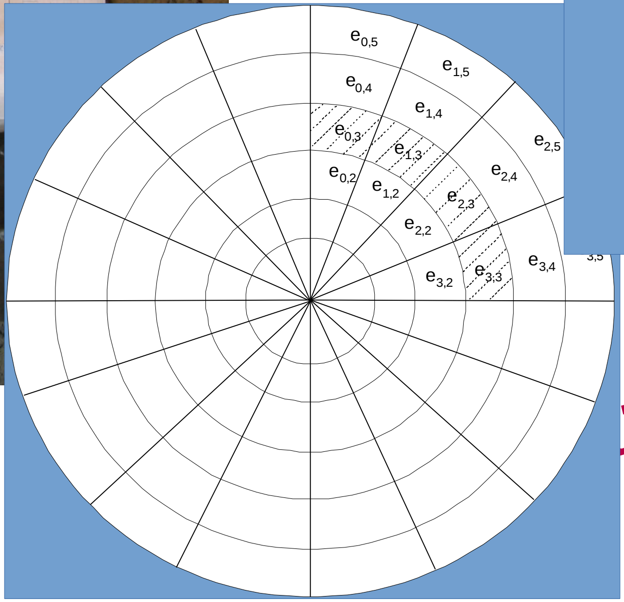
Memory Management for Radars

TERMA®





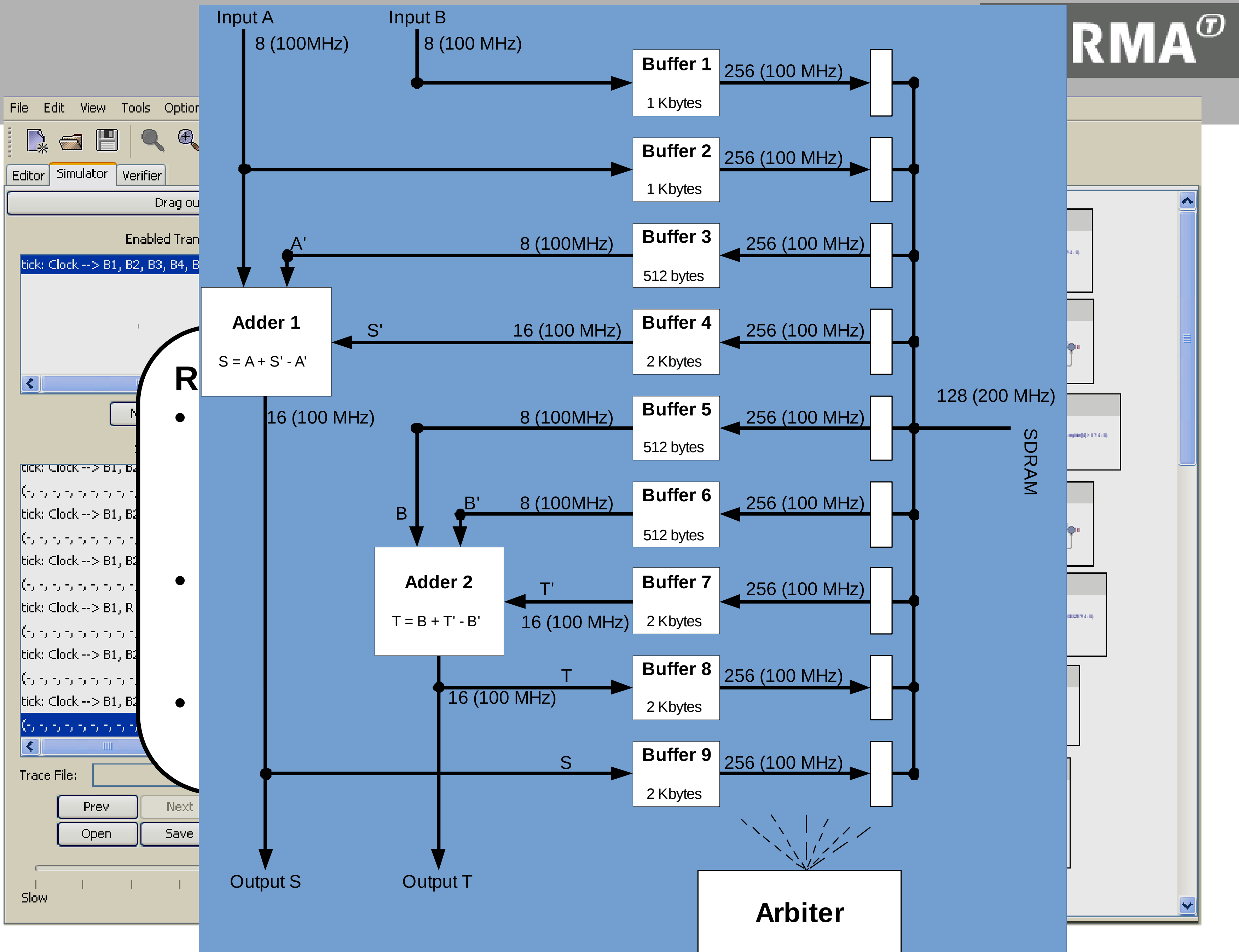
Frequency Diversity



ion

AMETIST

advanced methods for timed systems

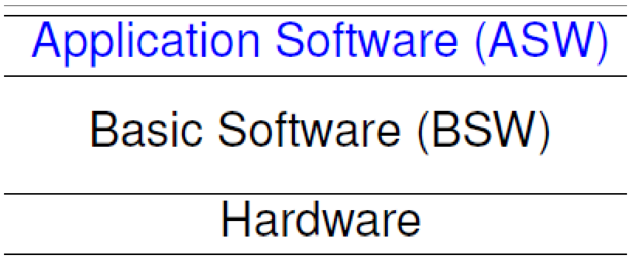

Input A
8 (100MHz)
Input B
8 (100 MHz)
Buffer 1
1 Kbytes
256 (100 MHz)
Buffer 2
1 Kbytes
256 (100 MHz)
A'
8 (100MHz)
Buffer 3
512 bytes
256 (100 MHz)
Adder 1
S = A + S' - A'
S'
16 (100 MHz)
Buffer 4
2 Kbytes
256 (100 MHz)
128 (200 MHz)
16 (100 MHz)
8 (100MHz)
Buffer 5
512 bytes
256 (100 MHz)
SDRAM
B
B'
8 (100MHz)
Buffer 6
512 bytes
256 (100 MHz)
Adder 2
T = B + T' - B'
T'
16 (100 MHz)
Buffer 7
2 Kbytes
256 (100 MHz)
T
16 (100 MHz)
Buffer 8
2 Kbytes
256 (100 MHz)
S
Buffer 9
2 Kbytes
256 (100 MHz)
Output S
Output T
Arbiter
File Edit View Tools Options
Editor Simulator Verifier
Trace File:
Prev Next
Open Save
Slow

TERMA A/S (2011)
Herschel-Planck Scientific Mission at ESA

Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard

TERMA A/S (2011)
Herschel-Planck Scientific Mission at ESA

- **Application software (ASW)**
  - built and tested by Terma:
  - does attitude and orbit control, tele-commanding, fault detection isolation and recovery.
- Basic software (BSW)
  - low level communication and scheduling periodic events.
- Real-time operating system (RTEMS)
  - Priority Ceiling for ASW,
  - Priority Inheritance for BSW
- Hardware
  - single processor, a few
    buses, sensors and ac



Requirements:

Software tasks should be schedulable.

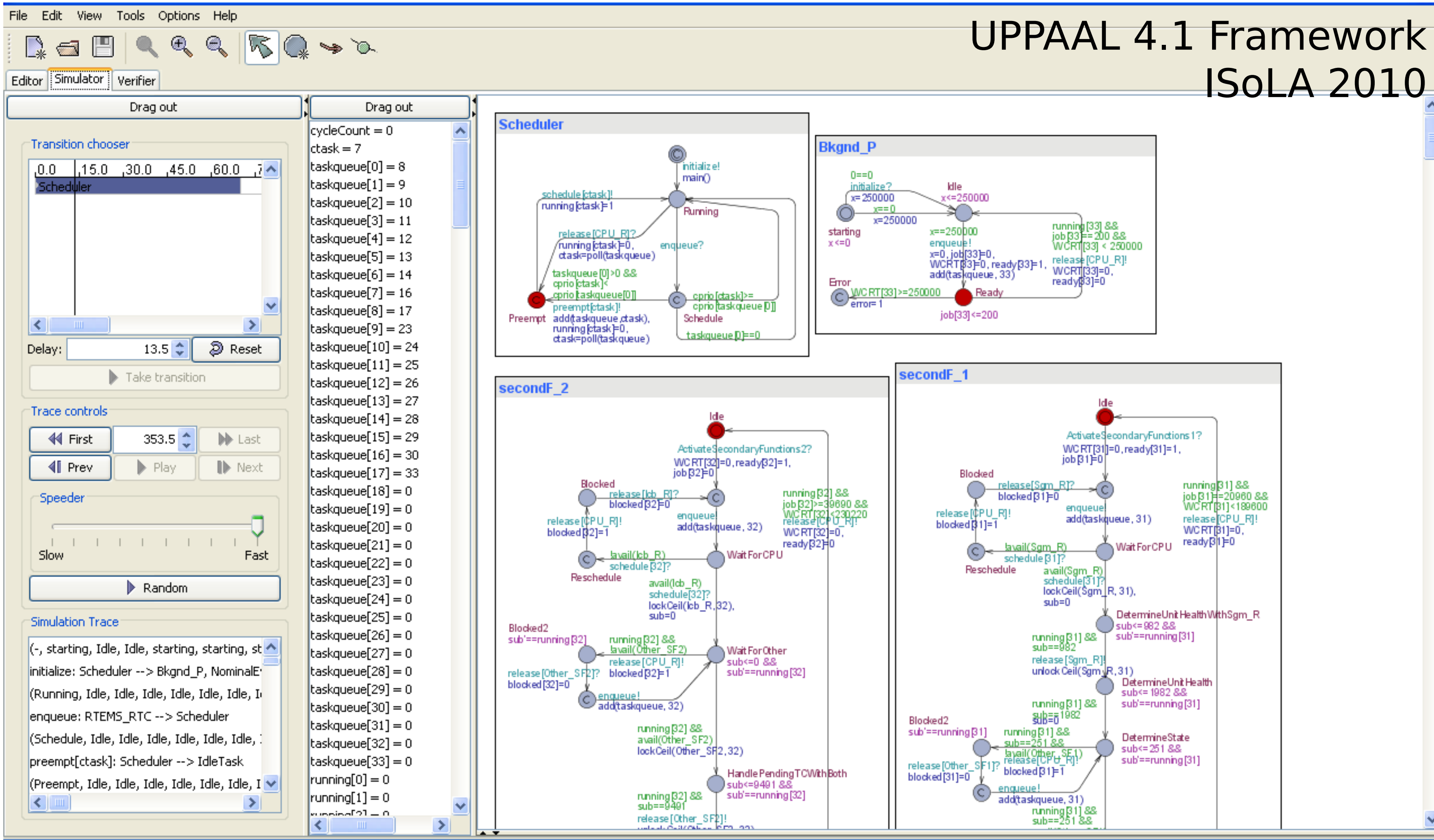
CPU utilization should not exceed 50% load

TERMA A/S (2011)

Herschel-Planck Scientific Mission at ESA

TERMA

UPPAAL 4.1 Framework
ISoLA 2010



TERMA A/S (2011)
Herschel-Planck Scientific Mission at ESA

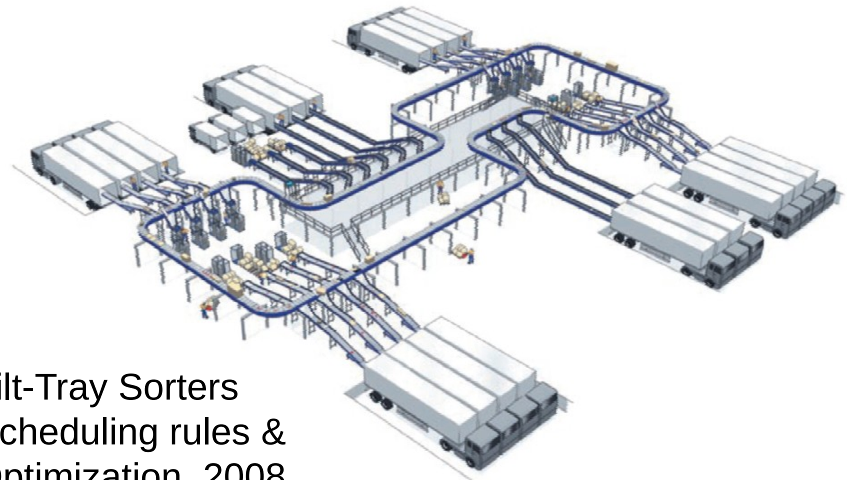
ID	Task	Specification			Blocking times			WCRT		
		Period	WCET	Deadline	Terma	UPPAAL	Diff	Terma	UPPAAL	Diff
1	RTEMS_RTC	10.000	0.013	1.000	0.035	0	0.035	0.050	0.013	0.037
2	AswSync_SyncPulseIsr	250.000	0.070	1.000	0.035	0	0.035	0.120	0.083	0.037
3	Hk_SamplerIsr	125.000	0.070	1.000	0.035	0	0.035	0.120	0.070	0.050
4	SwCyc_CycStartIsr	250.000	0.200	1.000	0.035	0	0.035	0.320	0.103	0.217
5	SwCyc_CycEndIsr	250.000	0.100	1.000	0.035	0	0.035	0.220	0.113	0.107
6	Rt1553_Isr	15.625	0.070	1.000	0.035	0	0.035	0.290	0.173	0.117
7	Bc1553_Isr	20.000	0.070	1.000	0.035	0	0.035	0.360	0.243	0.117
8	Spw_Isr	39.000	0.070	2.000	0.035	0	0.035	0.430	0.313	0.117
9	Obdh_Isr	250.000	0.070	2.000	0.035	0	0.035	0.500	0.383	0.117
10	RtSdb_P_1	15.625	0.150	15.625	3.650	0	3.650	4.330	0.533	3.797
11	RtSdb_P_2	125.000	0.400	15.625	3.650	0	3.650	4.870	0.933	3.937
12	RtSdb_P_3	250.000	0.170	15.625	3.650	0	3.650	5.110	1.103	4.007
14	**FdirEvents**	250.000	5.000	230.220	0.720	0	0.720	7.180	5.153	2.027
15	**NominalEvents_1**	250.000	0.720	230.220	0.720	0	0.720	7.900	5.873	2.027
16	**MainCycle**	250.000	0.400	230.220	0.720	0	0.720	8.370	6.273	2.097
17	HkSampler_P_2	125.000	0.500	62.500	3.650	0	3.650	11.960	5.380	6.580
18	HkSampler_P_1	250.000	6.000	62.500	3.650	0	3.650	18.460	11.615	6.845
19	Acb_P	250.000	6.000	50.000	3.650	0	3.650	24.680	6.473	18.207
20	IoCyc_P	250.000	3.000	50.000	3.650	0	3.650	27.820	9.473	18.347
21	**PrimaryF**	250.000	34.050	59.600	5.770	0.966	4.804	65.470	54.115	11.355
22	**RCSControlF**	250.000	4.070	239.600	12.120	0	12.120	76.040	53.994	22.046
23	Obt_P	1000.000	1.100	100.000	9.630	0	9.630	74.720	2.503	72.217
24	Hk_P	250.000	2.750	250.000	1.035	0	1.035	6.800	4.953	1.847
25	StsMon_P	250.000	3.300	125.000	16.070	0.822	15.248	85.050	17.863	67.187
26	TmGen_P	250.000	4.860	250.000	4.260	0	4.260	77.650	9.813	67.837
27	Sgm_P	250.000	4.020	250.000	1.040	0	1.040	18.680	14.796	3.884
28	TcRouter_P	250.000	0.500	250.000	1.035	0	1.035	19.310	11.896	7.414
29	Cmd_P	250.000	14.000	250.000	26.110	1.262	24.848	114.920	94.346	20.574
30	**NominalEvents_2**	250.000	1.780	230.220	12.480	0	12.480	102.760	65.177	37.583
31	**SecondaryF_1**	250.000	20.960	189.600	27.650	0	27.650	141.550	110.666	30.884
32	**SecondaryF_2**	250.000	39.690	230.220	48.450	0	48.450	204.050	154.556	49.494
33	Bkgnd_P	250.000	0.200	250.000	0.000	0	0.000	154.090	15.046	139.044

Marius Micusionis

CONCLUSION

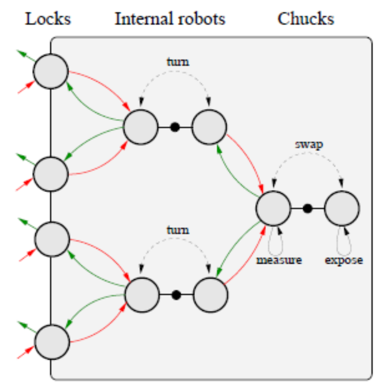
- Schedulability framework made available in UPPAAL
- Provides more exact analysis than classical methods
  - Depending on WCET information the task set is schedulable or not.
- Performance:
  - 1-2 minutes: BCET=WCET or BCET/WCET < 0.5
  - 1 day: 0.5 < BCET/WCET < 0.8
- Work on domain specific notation in order to be fully taken up by company.

& NASA



Tilt-Tray Sorters
Scheduling rules &
Optimization, 2008

Océ Datapath, 2012



ASML, 2004:
Wafer Scanners
Optimization of
Throughput

Philips: Indoor Lighting systems, 2014

Within the Prisma project of TNO-ESI and Philips Lighting, research is done into the robustness and reliability of large-scale indoor lighting systems. The focus is on the robustness of the lighting control system. To analyse control system robustness, model checking is used. Timed automata models of lighting control systems have been created and checked with the model checker Uppaal. To validate

UPPAAL Outside Europe

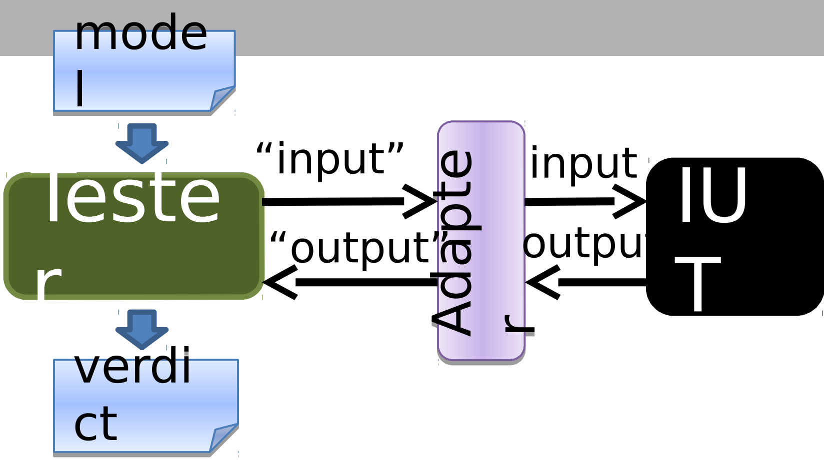
Testing

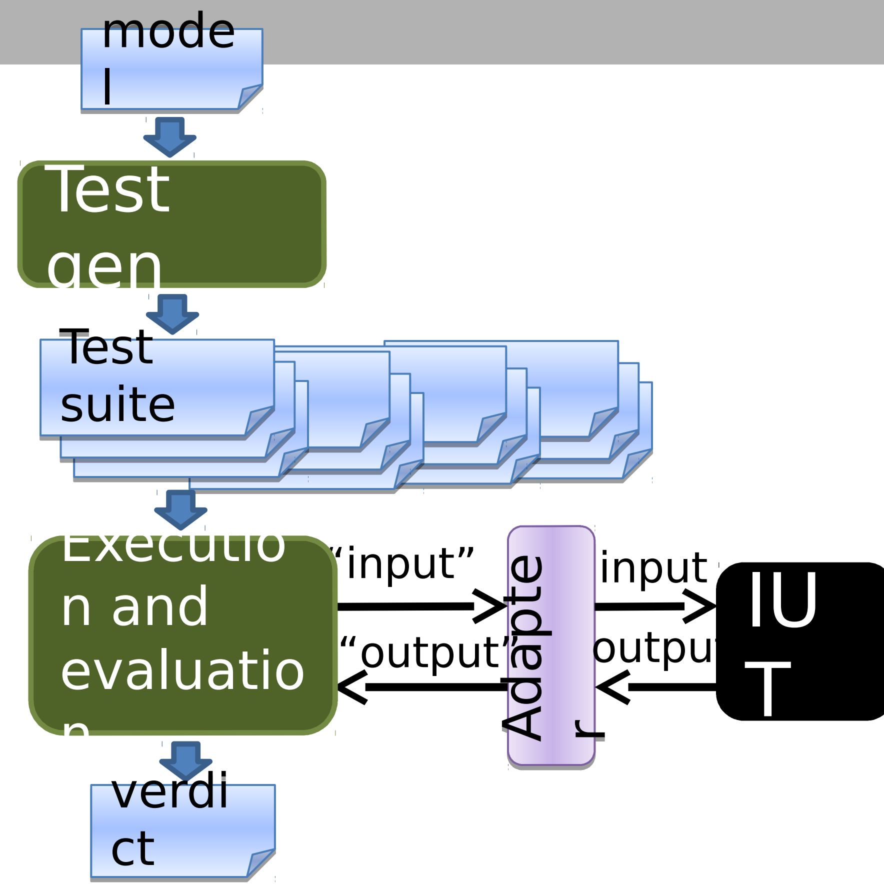
TRON & YGGDRASIL

AALBORG UNIVERSITET

UP4ALL

Online vs. Offline





nline testing:

ros:

- Abstract system-level behavior
- Realistic setup, many components
- Adaptive, explores only relevant states
- Allows concurrency, non-determinism
- Long and intricate interactions
- Automatic check against model

ons:

- Does not guarantee coverage
- Interpreting model can be slow
- Can be difficult to replicate
- Does not replace offline testing

Offline testing:

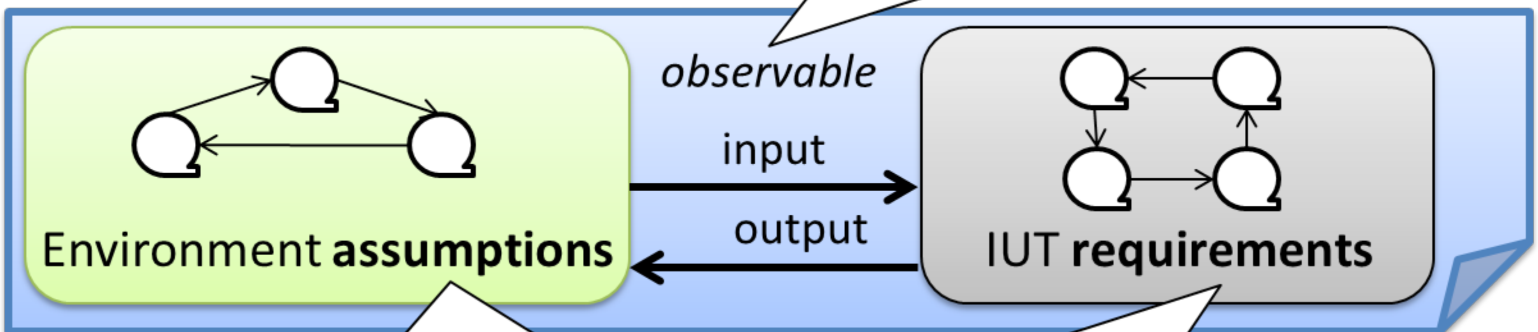
- Real-time systems are inherently non-determir
- Non-determinism yields exponentially large tes
- Few or no concurrent components
- Short and specific interactions
- Evaluation requires careful assertion programr

Model Interpretation

Test execution

Inputs translated as test stimuli

Test responses recognized as outputs



Test generation

How the **tester** should behave:

- Anything is possible (stress testing)
- Emulate physical processes
- Specific use-case scenario
- Replay a previous test trace (regression)

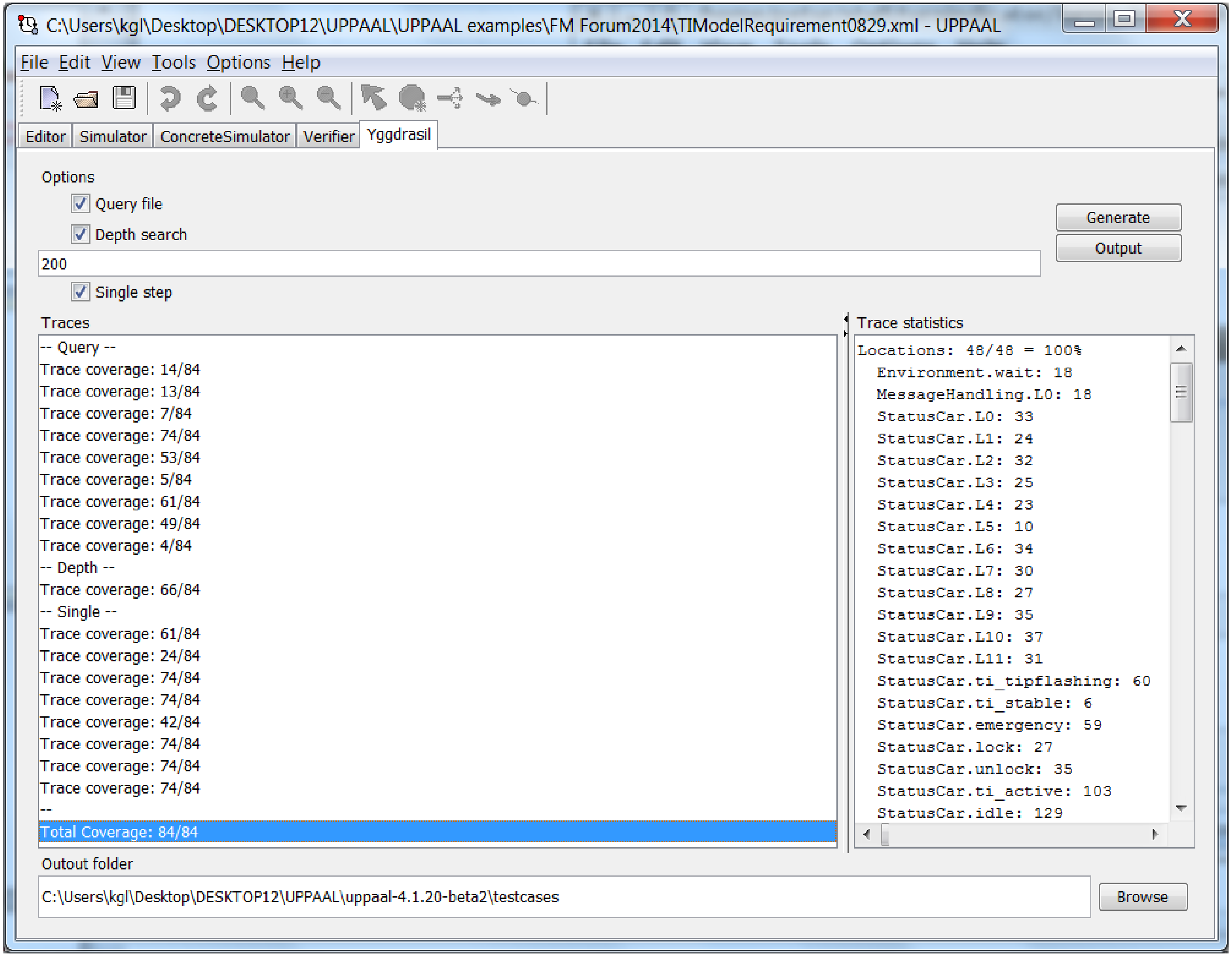
Test evaluation/monitoring

How the **IUT** should behave:

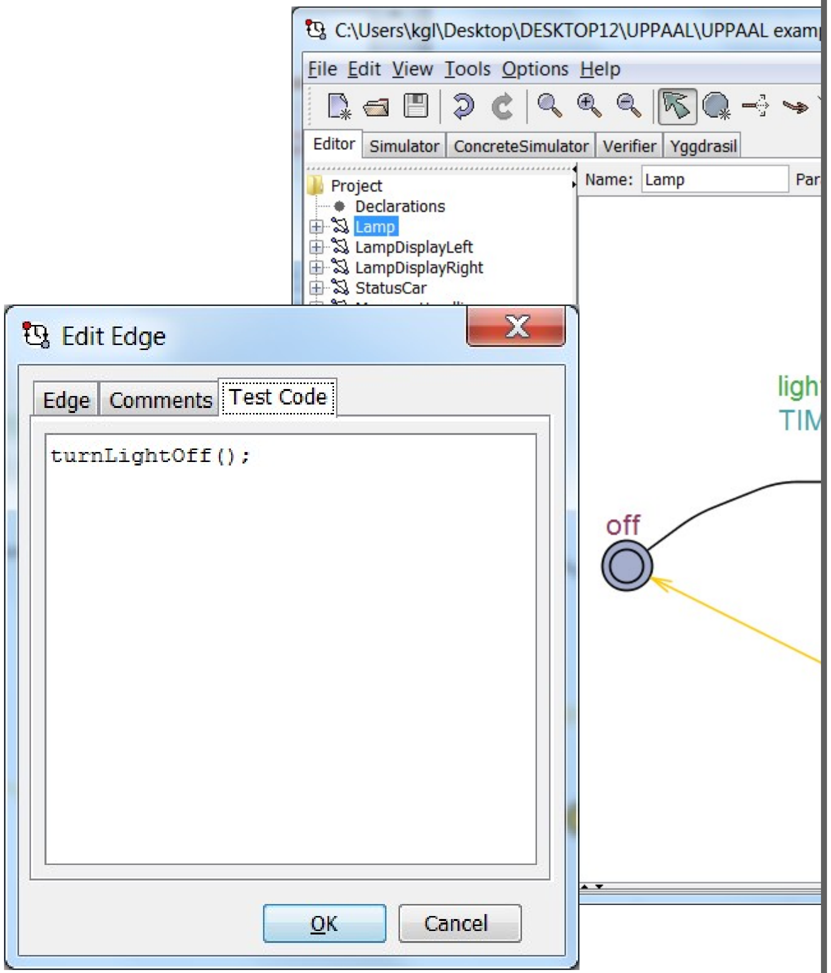
- Consume any input at anytime
- Produce outputs expected by model
- Neither too late nor too early
- Non-deterministic:
  - multiple outcomes
  - imprecision of timing
  - concurrency

Yggdrasil (offline)

MBAT Daimler Case (2014)



Test Code & Output



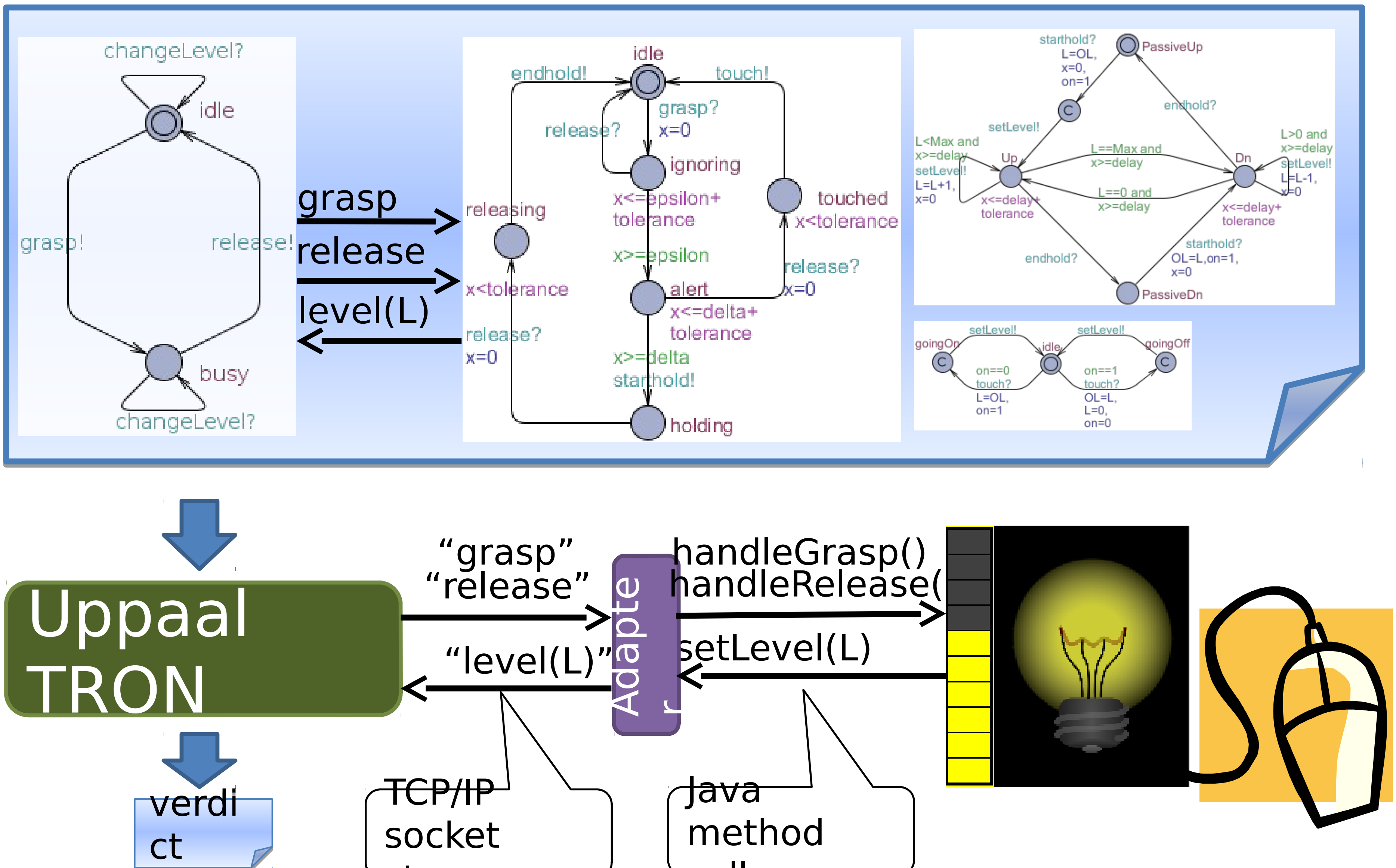

```
DrumScaleOneWeighing_IsInTopPos = 2;

DrumScaleOneWeighing_StartSTM( &me->itsDrumScaleOneWeighing );

HouseSettings_SetRamChar( &me->itsHouseSettings,
HS_CURRENT_ACTIVE_SILO, (UCHAR)0);

me->itsDrumScaleWeighingStable.IsReady_Return = 1;

me->itsDrumScaleRollDrum.IsReady_Return = 1;

Test_Validate( &me->itsTest, "DrumScaleOneWeighing_running",
(UCHAR)IS_IN( &me->itsDrumScaleOneWeighing,
DrumScaleOneWeighing_running ) );

Test_Validate( &me->itsTest,
"DrumScaleOneWeighing_StartOneWeighingState", (UCHAR)IS_IN(
&me->itsDrumScaleOneWeighing,
DrumScaleOneWeighing_StartOneWeighingState ) );

PrintCurrentState(me);

DrumScaleOneWeighing_StartOneWeighing(&me->
itsDrumScaleOneWeighing, (FLOAT32)12.0,(void (*)(void * const,
FLOAT32))OneWeighingFinishedCb,(UCHAR (*)(void * const))
OkToRollDrumCb, me);

Test_Validate( &me->itsTest,
"DrumScaleOneWeighing_WeighingDrumEmpty", (UCHAR)IS_IN( &me->
itsDrumScaleOneWeighing,
DrumScaleOneWeighing_WeighingDrumEmpty ) );

PrintCurrentState(me);

Test_Comment( &me->itsTest, "DrumScaleRollDrum_IsInTopPos is %
ld", (UCHAR)DrumScaleRollDrum_IsInTopPos(&me->
itsDrumScaleRollDrum) );

me->itsDrumScaleWeighingStable.StableWeighingFinishedCb(me->
itsDrumScaleWeighingStable.owner, 0.6F, 9.62F );
```

Yggdrasil Industrial Use

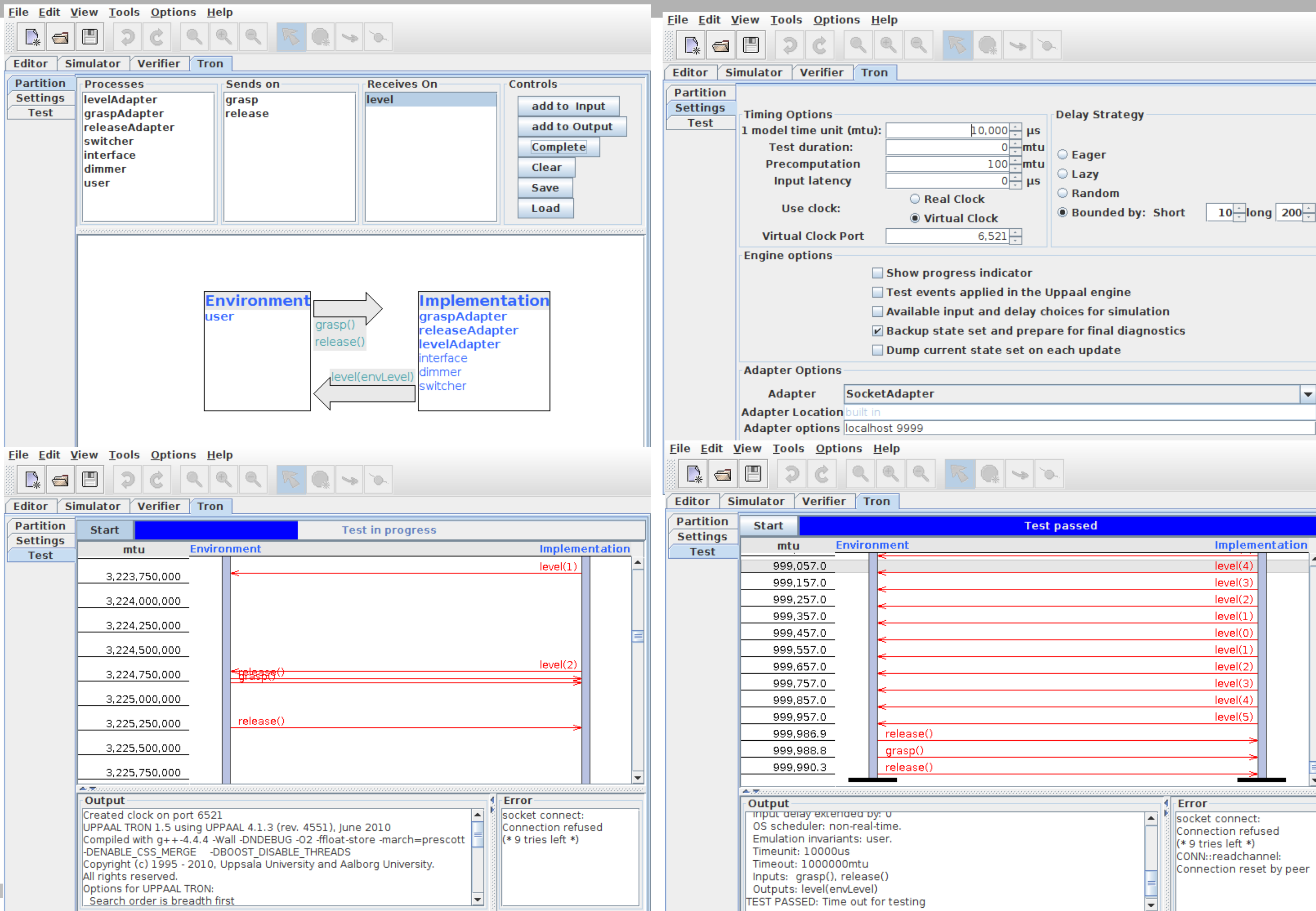
- Novo Nordisk
  - Reduction in time for testing a module 30 days 30 days ✉ 2 days
- Skov A/S
- TK Validate
  - Ambitios business plan

- Evaluation at
  - Daimler
  - Infinion Austria
  - EADS
  - Bombardier
    - Cov. Inc 40%
    - Reduced test time 20% (80% for unit test)

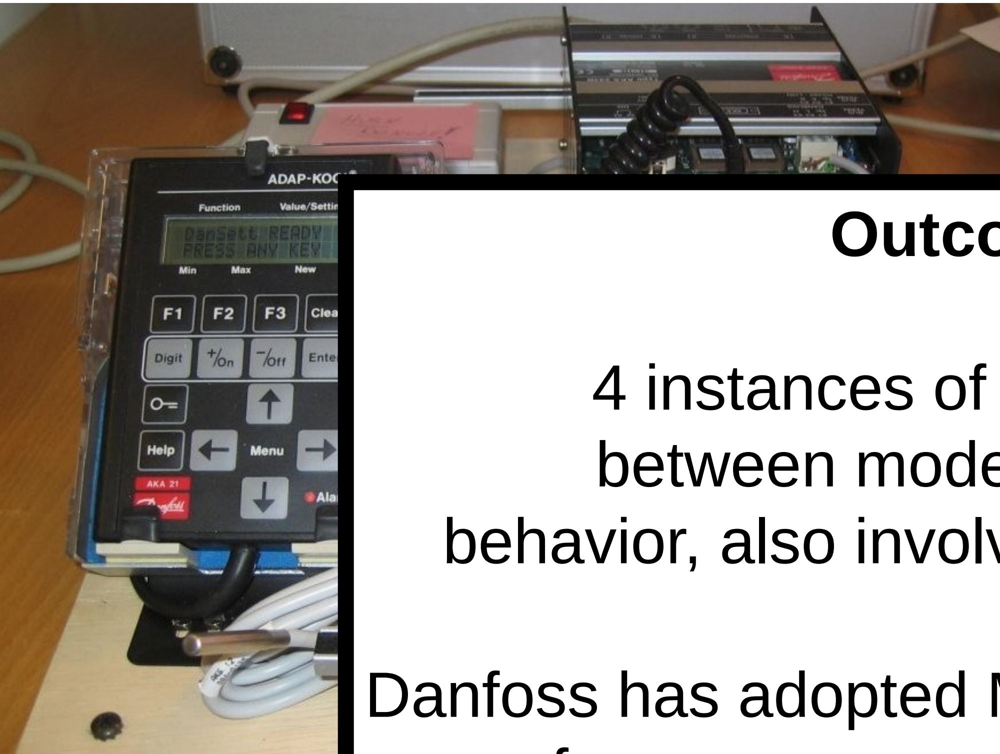
TRON (online)


changeLevel?
idle
grasp!
release!
busy
changeLevel?
grasp
release
level(L)
idle
endhold!
touch!
grasp?
x=0
release?
ignoring
x<=epsilon+
tolerance
releasing
touched
x<tolerance
x>=epsilon
release?
x=0
x<tolerance
alert
x<=delta+
tolerance
release?
x=0
x>=delta
starthold!
holding
starthold?
L=OL,
x=0,
on=1
PassiveUp
endhold?
setLevel!
L<Max and
x>=delay
setLevel!
L=L+1,
x=0
Up
L==Max and
x>=delay
Dn
L>0 and
x>=delay
setLevel!
L=L-1,
x=0
L==0 and
x>=delay
x<=delay+
tolerance
x<=delay+
tolerance
starthold?
OL=L,on=1,
x=0
endhold?
PassiveDn
setLevel!
setLevel!
goingOn
idle
goingOff
on==0
touch?
L=OL,
on=1
on==1
touch?
OL=L,
L=0,
on=0
Uppaal
TRON
verdict
"grasp"
"release"
"level(L)"
Adapter
handleGrasp()
handleRelease()
setLevel(L)
TCP/IP
socket
streams
Java
method
calls

TRON GUI


File Edit View Tools Options Help
Editor Simulator Verifier Tron
Partition
Settings
Test
Processes
levelAdapter
graspAdapter
releaseAdapter
switcher
interface
dimmer
user
Sends on
grasp
release
Receives On
level
Controls
add to Input
add to Output
Complete
Clear
Save
Load
Environment
user
grasp()
release()
level(envLevel)
Implementation
graspAdapter
releaseAdapter
levelAdapter
interface
dimmer
switcher
Timing Options
1 model time unit (mtu): 10,000 µs
Test duration: 0 mtu
Precomputation 100 mtu
Input latency 0 µs
Use clock:
Real Clock
Virtual Clock
Virtual Clock Port 6,521
Delay Strategy
Eager
Lazy
Random
Bounded by: Short 10 long 200
Engine options
Show progress indicator
Test events applied in the Uppaal engine
Available input and delay choices for simulation
Backup state set and prepare for final diagnostics
Dump current state set on each update
Adapter Options
Adapter SocketAdapter
Adapter Location built in
Adapter options localhost 9999
Start
Test in progress
mtu
Environment
Implementation
3,223,750,000
3,224,000,000
3,224,250,000
3,224,500,000
3,224,750,000
3,225,000,000
3,225,250,000
3,225,500,000
3,225,750,000
level(1)
level(2)
release()
grasp()
release()
Output
Created clock on port 6521
UPPAAL TRON 1.5 using UPPAAL 4.1.3 (rev. 4551), June 2010
Compiled with g++-4.4.4 -Wall -DNDEBUG -O2 -ffloat-store -march=prescott
-DENABLE_CSS_MERGE -DBOOST_DISABLE_THREADS
Copyright (c) 1995 - 2010, Uppsala University and Aalborg University.
All rights reserved.
Options for UPPAAL TRON:
Search order is breadth first
Error
socket connect:
Connection refused
(* 9 tries left *)
Test passed
999,057.0
999,157.0
999,257.0
999,357.0
999,457.0
999,557.0
999,657.0
999,757.0
999,857.0
999,957.0
999,986.9
999,988.8
999,990.3
level(4)
level(3)
level(2)
level(1)
level(0)
level(1)
level(2)
level(3)
level(4)
level(5)
release()
grasp()
release()
OS scheduler: non-real-time.
Emulation invariants: user.
Timeunit: 10000us
Timeout: 1000000mtu
Inputs: grasp(), release()
Outputs: level(envLevel)
TEST PASSED: Time out for testing
socket connect:
Connection refused
(* 9 tries left *)
CONN::readchannel:
Connection reset by peer

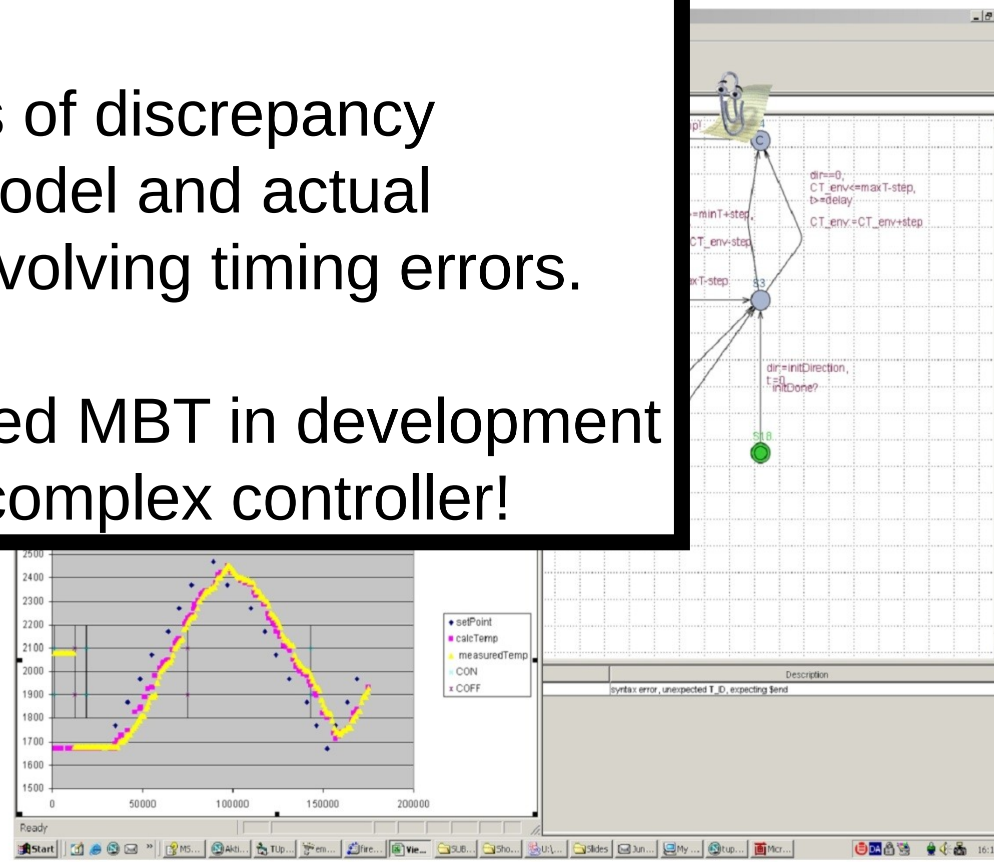
Danfoss (2008)
Cooling Control for Industrial Fridges



Outcome

4 instances of discrepancy between model and actual behavior, also involving timing errors.

Danfoss has adopted MBT in development of new more complex controller!



- Sequanto SeqZap test harness
- Programmable Logic Controllers (PLC)

Advantages of MBT

- Engineer focus on **what** to test at a high level of abstraction
- Avoids cost of making scripts
  - As much test code as production code
  - Maintenance nightmare
- Heard of, but is still considered an **advanced technique** by industry
- Industry is very motivated, MB A&T will give
  - **10% cost reduction**
  - **20% quality improvement**

Model Checking & Testing

Model Checking

- Abstract models
- Exhaustive “proof”
- Many mature tools
- Early detection of errors
- State space expl

Testing

- Checks the actual implementation
- Only few executions checked
- But is the most direct method

How to effectively ***combine*** the different model checking and testing techniques?

www.uppaal.{org,com}