SVERTS 2004

Workshop associated with UML 2004

Susanne Graf Verimag, Grenoble, France
Oystein Haugen University of Oslo, Norway
Ileana Ober Verimag, Grenoble, France
Bran Selic IBM/Rational, Canada

Motivation

The IST-2001-33522 Omega project on *Correct Development of Real-Time Embedded Systems*

- **Duration**: January 2002 –February 2005
- **Budget**: 2.8 KEuro
- **Aim**: Definition of a development methodology in UML for embedded and real-time systems integrating formal validation techniques
- **Coordinator**: Verimag
- **Partners**: next slide

Partners and supporters

Academic (tool and technology providers)

- **Verimag**, France – coordinator
- Christian-Albrechts University **Kiel**, Germany
- **CWI** (Centrum voor Wiskunde en Informatica), Netherlands
- University of **Nijmegen**, Netherlands
- **OFFIS**, Germany
- **Weizmann** Institute, Israel

Users

- **EADS** Launch Vehicles, France
- **France Telecom R&D**, France
- **Israeli Aircraft Industries**, Israel
- **NLR** (Nationaal Lucht- en Ruimtevaartlaboratorium), Netherlands

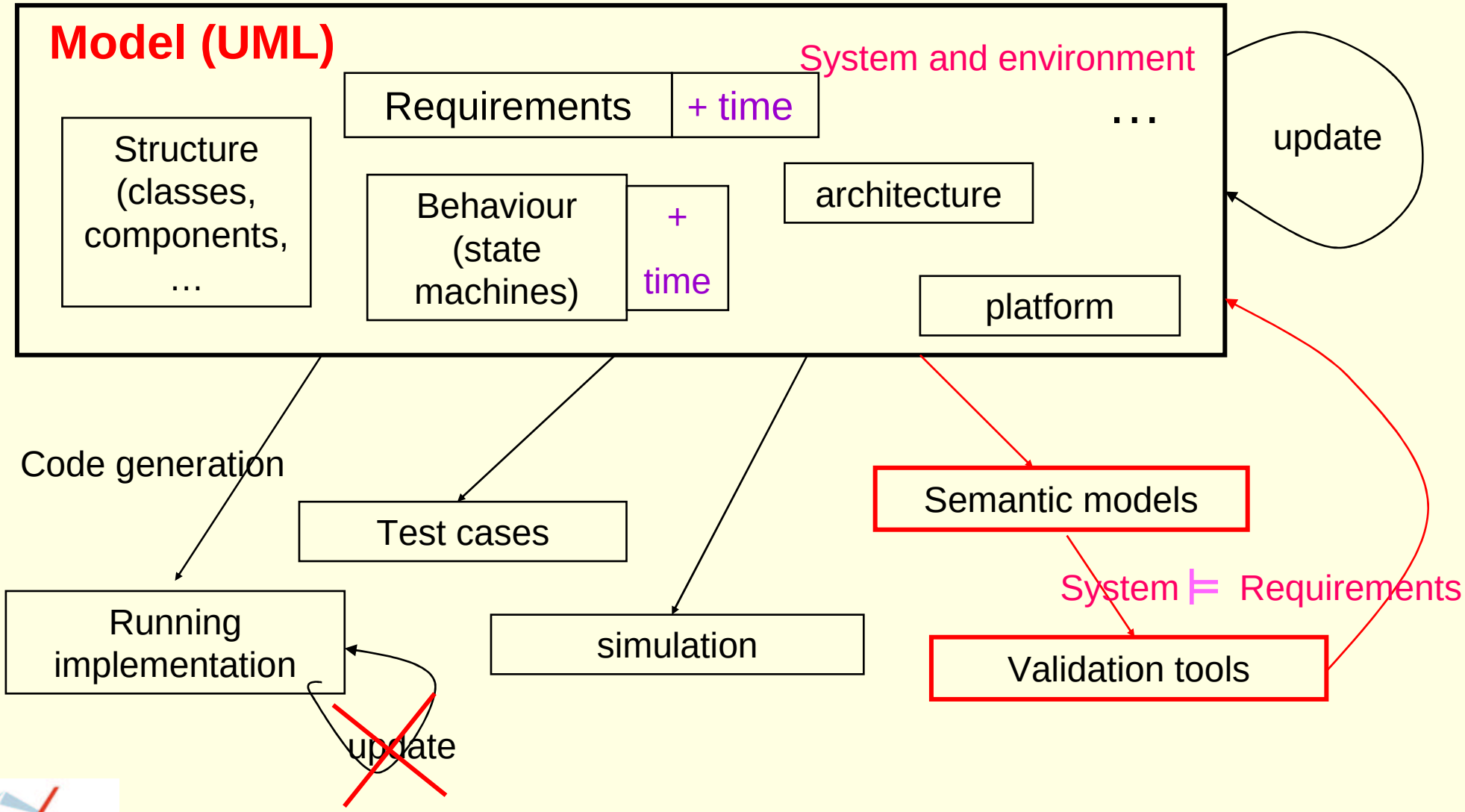
Supporters (UML tool providers)

I-Logix --- Rational Software, IBM --- **Telelogic**

Overview

- Motivation: how to apply formal validation in a UML based approach to system development
- Overview on the results developed in OMEGA
  - UML semantic based profile for the expression of real-time properties
  - Validation tools for real-time properties
  - The IF toolset and its connection with UML
  - Case studies and some preliminary conclusions
- Problems encountered
- What next?

Model based development and validation for real-time systems



How well does UML fit ?

Strong points of UML

- Support of requirement level and design level notations, including architecture and components, which made their proofs
- User acceptance
- Integration in development cycle possible

Weak points of UML (for validation of dynamics)

- Concepts are defined at syntax level, no well defined (dynamic) semantics and no framework for defining one
- **No clear concepts, it's up to the tools to chose and fix them**
- **Weak support of real-time concepts (improved by UML 2.0)**

Choices of Omega

- Fact: validation is only one aspect
  - Do not restrict the considered UML profile to make it just fit to the validation tools

- Fact: validation is an expensive task
  - Reuse existing state-the-art methods and tools
  - Be open to any UML tool: use standard model exchange format (XMI) and UML standard extension mechanisms
  - Be open to a variation of semantics
    - Chose a level of granularity which allows to adapt to different semantic frameworks by restrictions on non deterministic choices
  - Be open to different methodologies

Omega real-time profile for real-time systems

All extensions made using UML extension mechanisms

➔ models can be edited by “any” UML support

- Structure
  - Class diagrams distinguishing active objects (mono-threaded processes) and passive objects (local data)
  - Architecture and components (not available in UML 1.4 ; some work on components and connectors)

- Requirements
  - Live Sequence Charts and Observers express (global) constraints on the behavior (not only a step); they represent a generalization and formalization of use cases
  - OCL for the expression of structural invariants and invariants on event histories

Omega real-time profile for real-time systems

- Behavior (focusing on coordination)
  - Object behavior specifications using State machines with Action language (compatible to UML1.4 A.S.)
  - Some concepts for communication & concurrency
    - active/passive objects ➔ activity groups (run-to-completion)
    - interactions: primitive/triggered operations, asynchronous signals
- Timing constraints (in requirements, structure and design)

A semantics has been formally defined for this subset and implemented in several tools

Omega real time profile: Timing

Compatible SPT profile and UML 2.0

- Basics
  - A notion of *global time, external* to the system
  - Time primitive types: Time, Duration with operations
  - Timed Events: history of occurrence times of identified state changes
- Operational time access: *time dependent behavior*
  - Mechanisms for measuring durations: timers, clocks

Omega real time profile: Timing

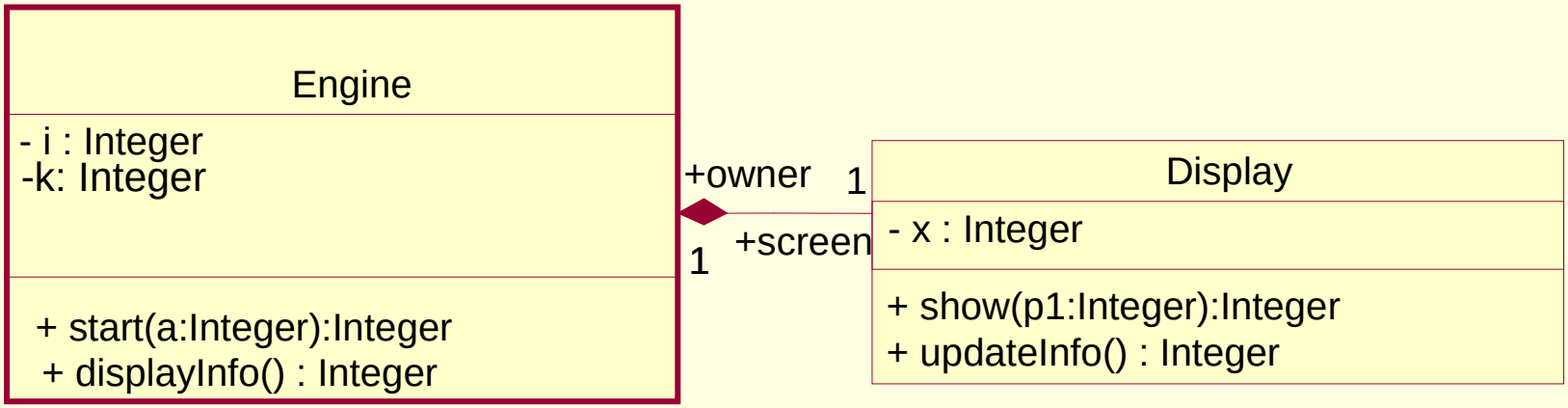
Compatible SPT profile and UML 2.0

- Basics
  - A notion of global time, *external* to the system
  - Time primitive types: Time, Duration with operations
  - ***Timed Events***: sequence of instants of occurrences of identified state changes in each execution:
    - "send signal", "receive signal", "consume signal"
    - "invoke method", ….
    - "enter state", "exit state"
    - "start action", "end action"
    - ….
- Operational time access (as in UML 2.0): *time dependent behavior*
  - Mechanisms for measuring durations: timers, clocks
  - And corresponding actions: set, reset,…

Time profile

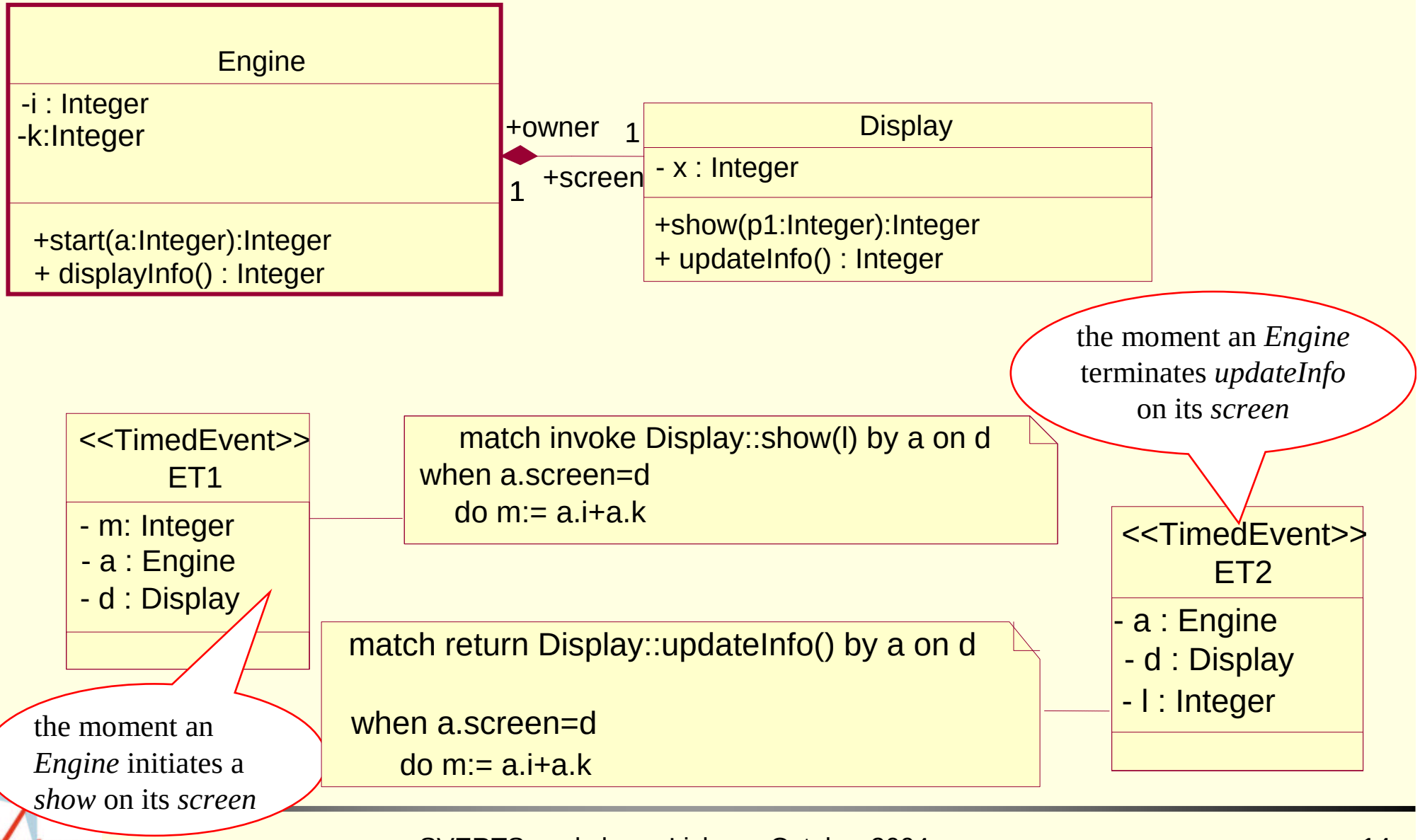
- Time constraints: *orthogonal to the behaviour*
  - Constraints on durations between *occurrences of events* (OCL based)
    - Temporal patterns for constraining occurrences of 2 events
    - Derived patterns associated with syntactic entities
      - response time,
      - duration of actions → deadline constraints,
      - duration in state,
      - delay of channel,...
  - Observers with time constraints (local or global) for the expression of properties implying more than 2 events
- Scheduling related
  - Resources accessed in mut. excl. and consuming execution time
  - Execution time of actions
  - *Dynamic priorities* for expressing scheduling policies

Time profile: example

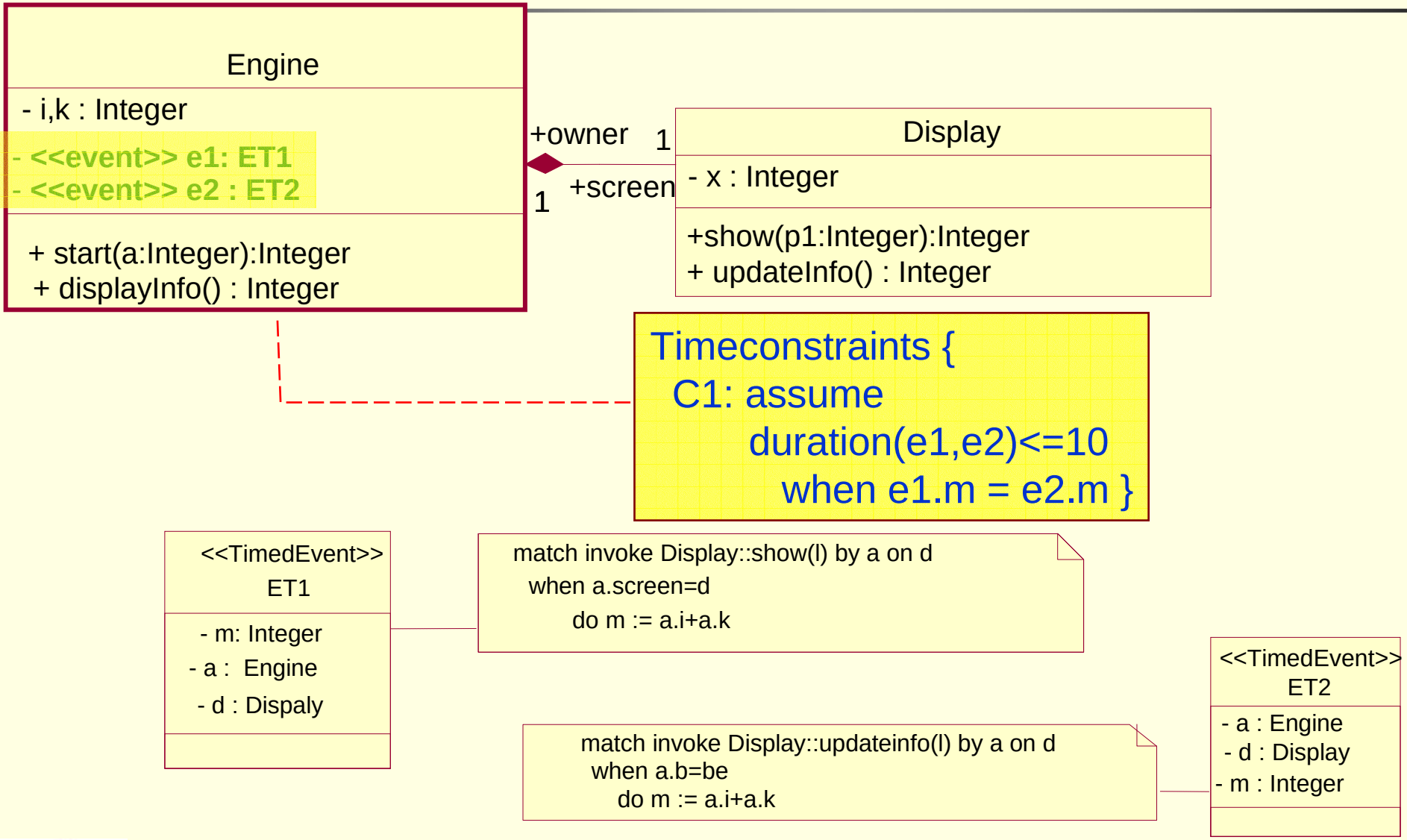


An informal time constraint:
Between the *moment an Engine initiates a show on its screen* and the *moment the same Engine has updated the information (finishes the call updateInfo) on its screen* less than 10 time units pass, if the sum i+k has not changed.

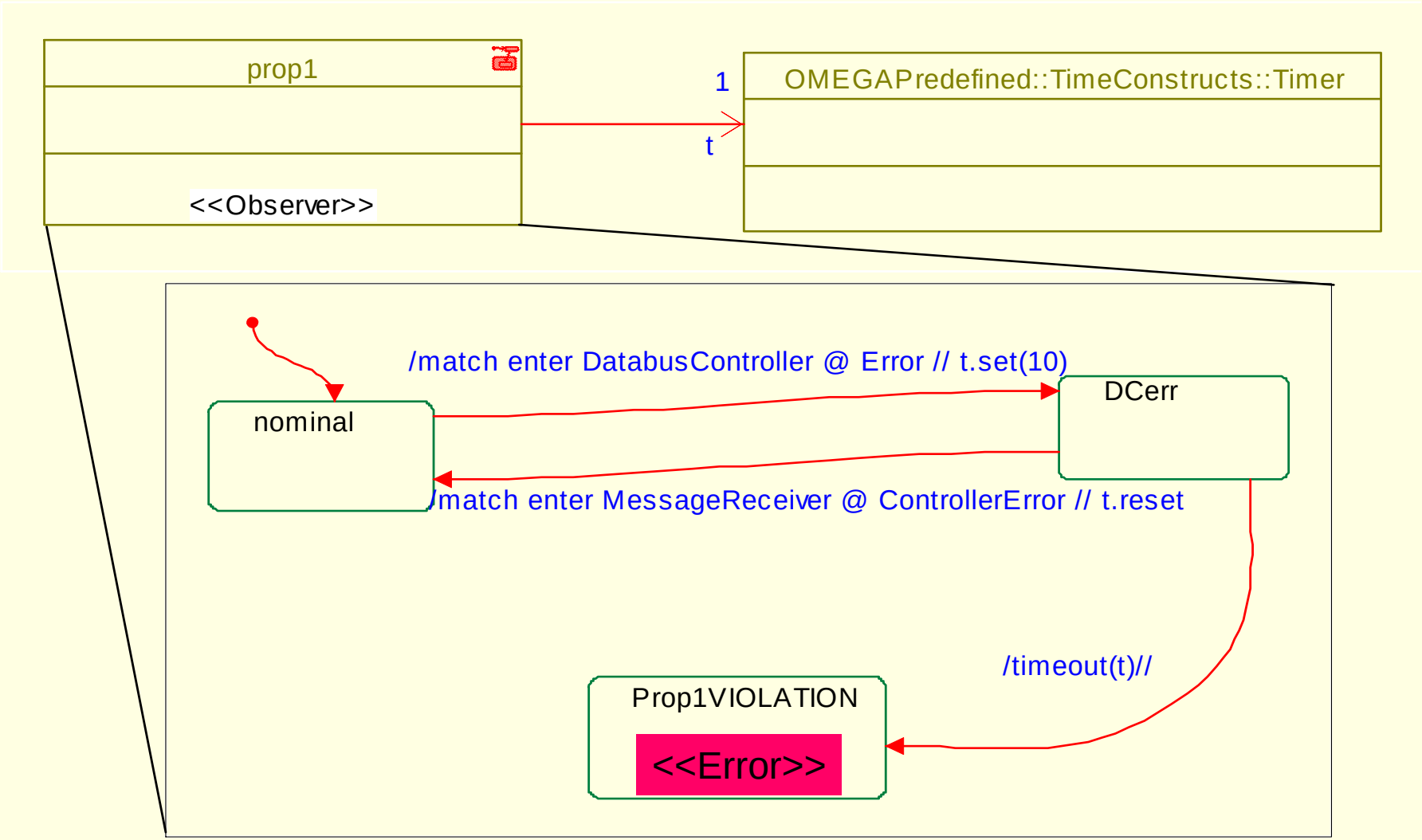
Time profile: events (example)



Time profile: constraints



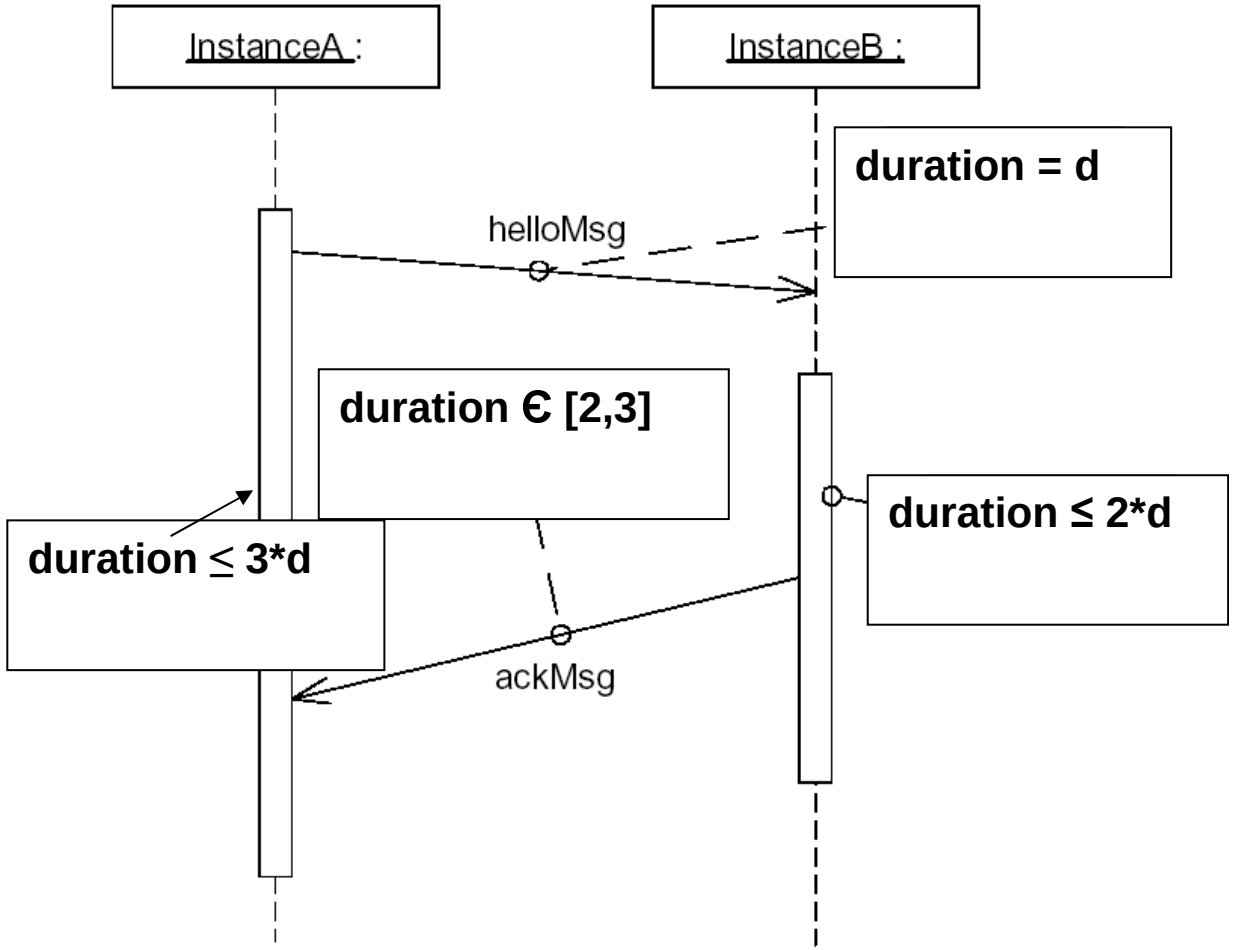
Time profile: observers



Time profiles: Sequence Diagrams vs observers

Problems:

- mainly argue on consistency of a subset of (important) event sequences
- expression of choices is non trivial
- express properties at instance level



Live Sequence charts avoid some of the problems !

Conclusions on profile

- Untimed behaviour:
  - Semantics with a fine grained granularity and non determinism
  - Different frameworks = different restrictions of non-determinism
- Timed extension:
  - Expression of time constraints in existing frameworks
  - Introduce a general naming scheme for events
  - Possibility to define a semantic framework

Overview

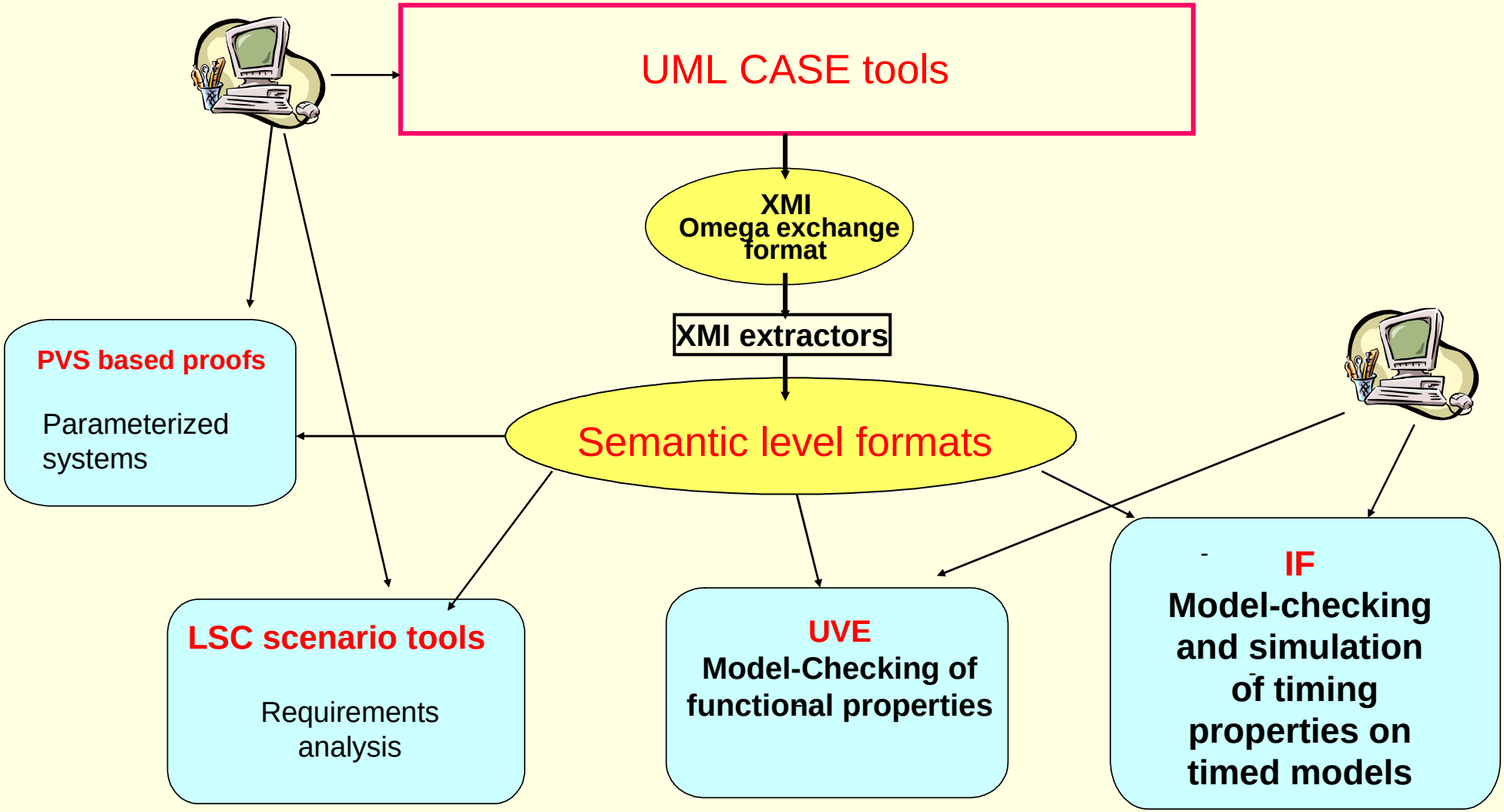
- Motivation: how to apply formal validation in a UML based approach to system development
- Overview on the results developed in OMEGA
  - UML profile for the expression of real-time properties
  - Validation tools for real-time properties
  - The IF toolset and its connection with UML
  - Case studies and some preliminary conclusions
- Problems encountered
- What next?

Validation tools for the Omega profile

Some challenges:

- Possibility to implement several semantics
- Avoid introduction of state explosion due to the translation
- Which high level concepts to keep at semantic level and which ones to map into simpler ones ?

Omega tool-set



IF tool-set: the IF notation

Processes (dynamic behavior)

extended timed automata

(non-determinism, dynamic creation)

Interactions

asynchronous channels

shared variables

Data

predefined data types (basic types, arrays, records)

abstract data types

Execution control

Priority rules

“resources” for the definition of mutex constraints

Toolset: UML-to-IF

Structure:

- class → process type
- attributes & associations → variables
- inheritance → replication of features
- signals, basic data types → direct mapping

Behavior

- state machines (with restrictions) → IF hierarchical automata
- action language → IF actions, automaton encoding
- operations:
  - operation call/return → signal exchange
  - procedure activations → process creation
  - polymorphism → untyped PIDs
  - dynamic binding → destination object automaton determines the executed procedure

Toolset: UML-to-IF

Time

- Timers and clocks $\rightarrow$ direct mapping
- Time constraints and observers $\rightarrow$ additional clocks and guards, observers

Execution modes

- use the dynamic priorities of IF to define the run-to-completion execution
  - ➔ Needs only 2 rules

$$\forall x,y.\ (x.active = y\ ) \Rightarrow x < y$$

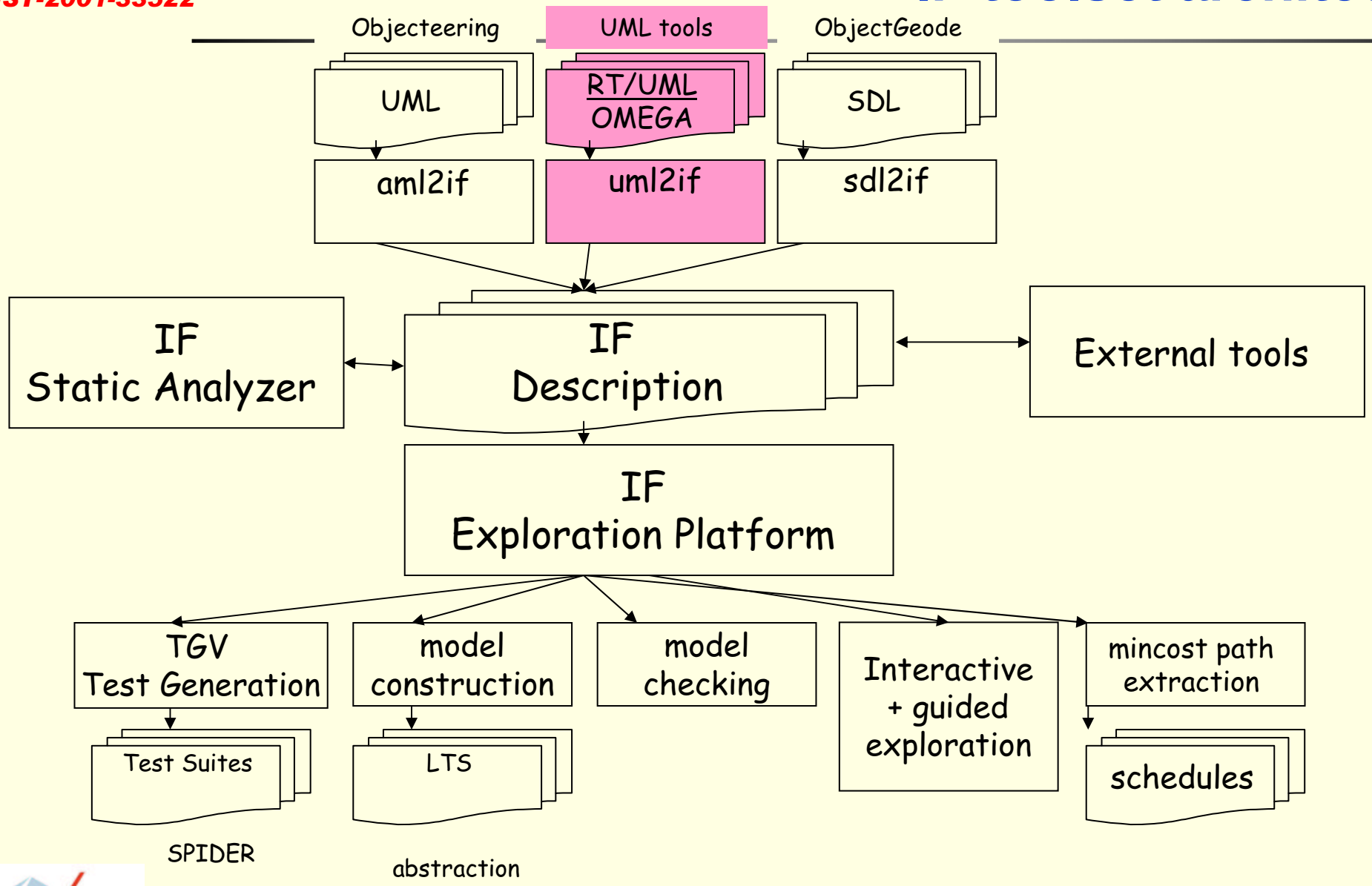
$$\forall x,y.\ (x = y.active.running \wedge x \neq y) \Rightarrow x < y$$

Conclusions

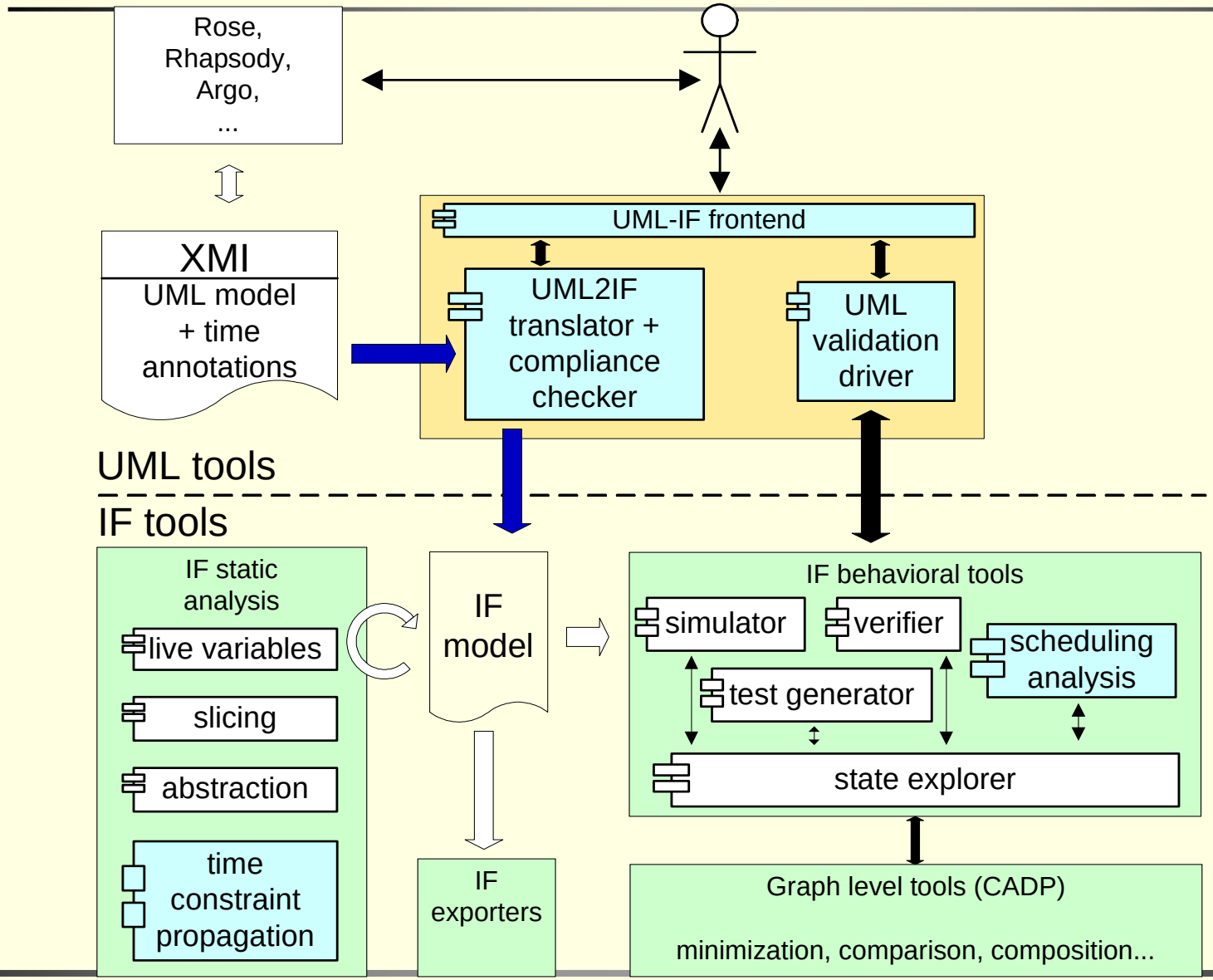
In the IF validation tool, the mapping from UML to the tool language has been designed for flexibility (play with semantic variations):

- Can be adapted to different dynamic semantics easily
- Architectural constraints (restrictions on communications, number of instances,…) are captured by attributes and guards
- A notion of thread is defined by IF level rules giving the priority to internal actions over acceptance of communications from outside.

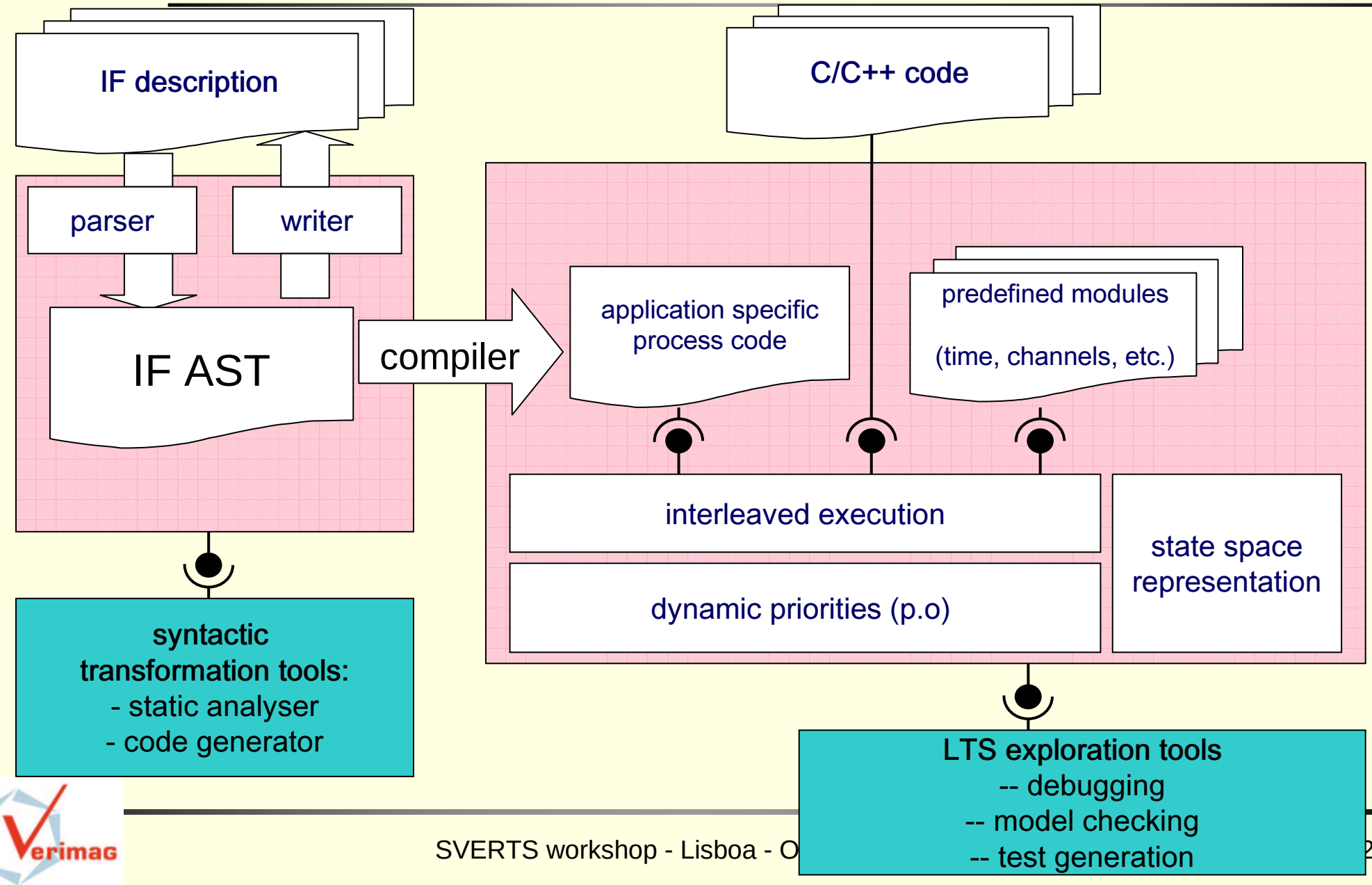
IF toolset architecture



IF toolset architecture



IF core components



IF toolset: time implementation

Dedicated module

- including clock variables
- handling dynamic clock allocation (set, reset)
- checking timing constraints (timed guards)
- computing time progress conditions w.r.t. actual deadlines and
- fires timed transitions, if enabled

Two implementations for discrete and continuous time (others can be easily added)

i) discrete time

- clock valuations represented as varying size integer vectors
- time progress is explicit and computed w.r.t. the next enabled deadline

ii) continuous time

- clock valuations represented using varying size difference bound matrices (DBMs)
- time progress represented symbolically
- non-convex time zones may arise because of deadlines: they are represented implicitly as unions of DBMs

Overview

- Motivation: how to apply formal validation in a UML based approach to system development
- Overview on the results developed in OMEGA
  - UML profile for the expression of real-time properties
  - Validation tools for real-time properties
  - The IF toolset and its connection with UML
  - Case studies and some preliminary conclusions
- Problems encountered
- What next?

Case studies

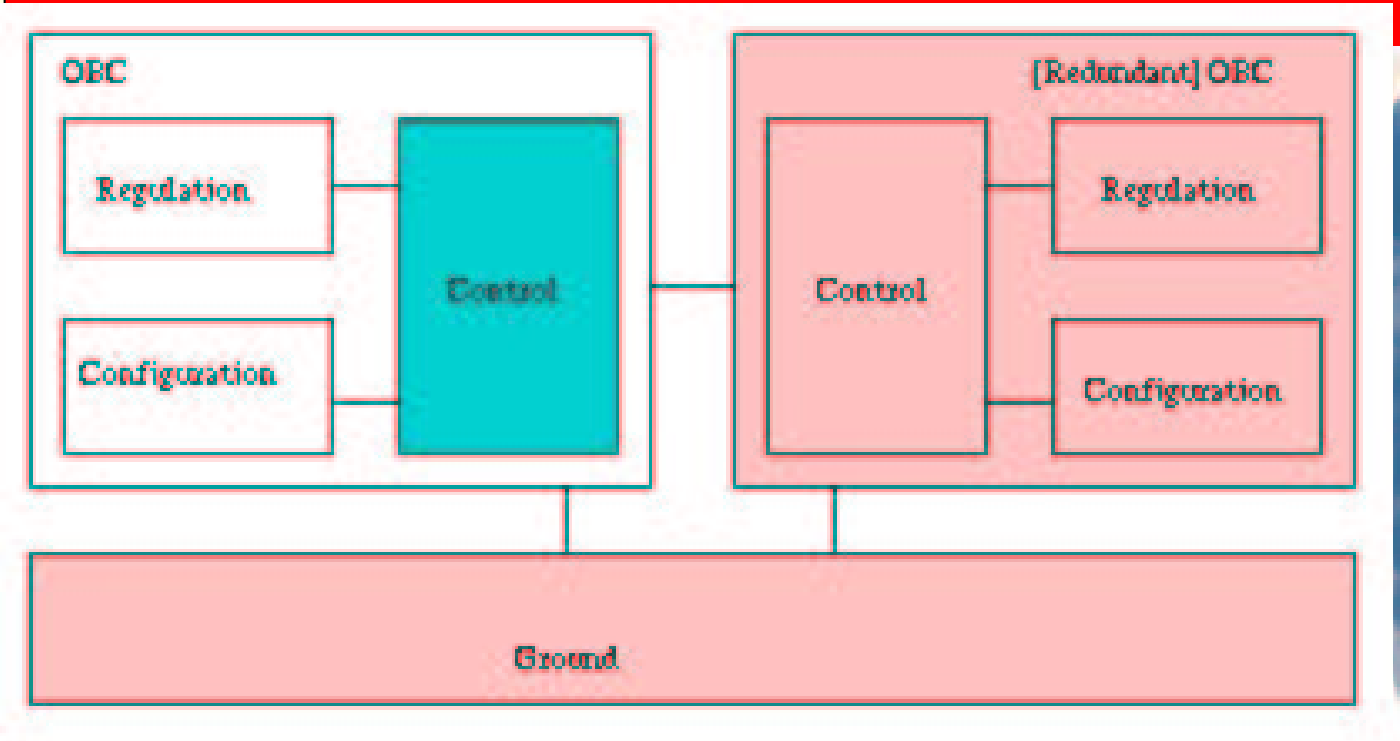
4 Case studies – 3 from aeronautics – 1 from Telecom

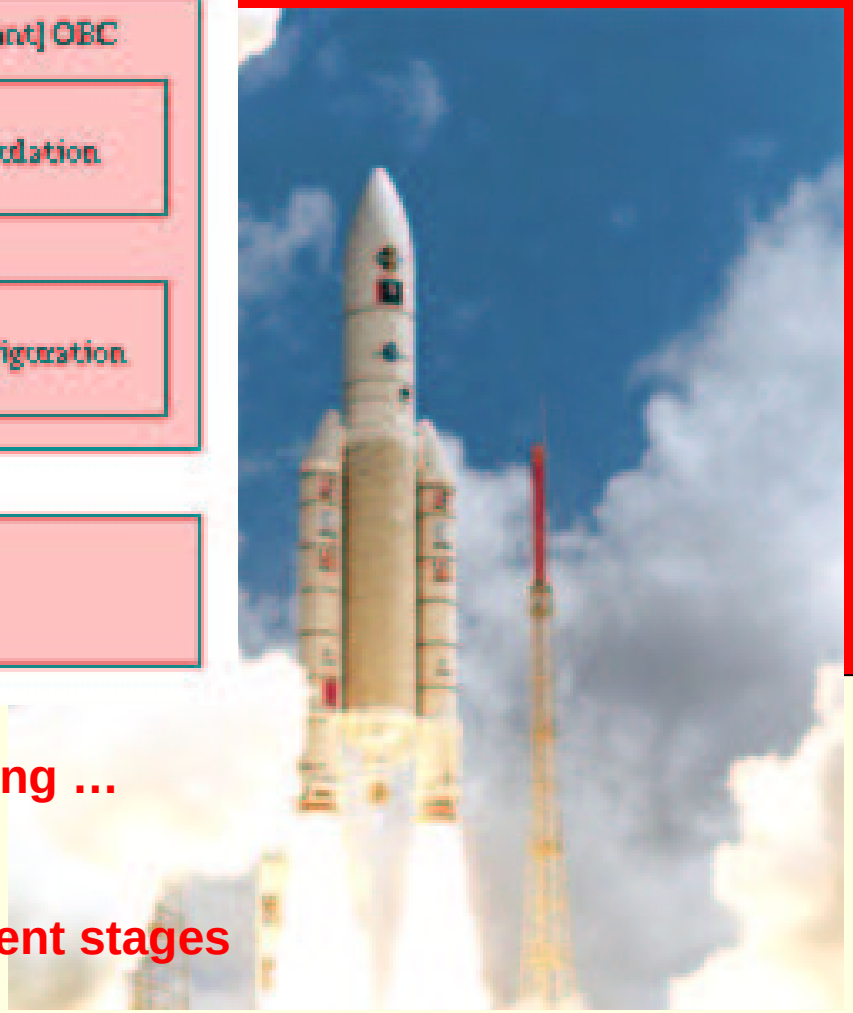
- An ATV: Replicated voting algorithm (2 voters, 3 sensors) (IAI)
  - Correctness of voting algorithm (functional correctness)
  - Timing correctness (under work, encouraging preliminary results)
- Ariane 5: flight schedule and control (EADS)
  - Correct ordering of sub-phases (slides below)
  - Interaction between synchronous and asynchronous part
  - Scheduling issues (under work)
- Mars: Bus controller for two desynchronized data sources (NLR)
  - Timely recognition of a bus error (slides below)
- Components for a depannage service (just started) (FT RD)
  - Timely occurrence of events
  - Constraints on event occurrence orders

Ariane 5 flight controller: 40 minutes flight

- many timers (smallest with 70ms rate)
- 31 objects

Ariane 5 flight controller





Control: navigation, guidance, global timing …

Regulation: firing control of different stages

Configuration: manages separation of the different stages

Ariane 5: validation steps

- Translation to IF : only few elementary data variables, but complex timing aspects (transmission of timer values through signals)
- Static analysis
  - **Clock reduction**: 1. Description: 143 timers reduced to 41 clocks

    2. Description: 55 timers, no more reduction
  - **Dead variable reduction**: 20% of all variables are dead in each state
  - **Slicing**: elimination of passive processes (without outputs)
- Model generation
  - **Appropriate environment restrictions**
  - **Partial order reduction** (31 processes)
- Verification
  - Construction and visualization of bisimulation reduced models
  - Evaluation of μ-calculus formulas
  - Out of 16 properties, 3 did not hold in the initial, carefully debugged model

Ariane 5: model generation results

	reduction strategy	states	transitions	time
1	none	too large !	too large !	?
2	live	2 200 000	18 700 000	2h07 m
3	**p.o.**	**2 000**	**2040**	**3.21 s**
4	live + p.o.	1 600	1642	1.68 s

Model generation after application of clock reduction and slicing of passive processes

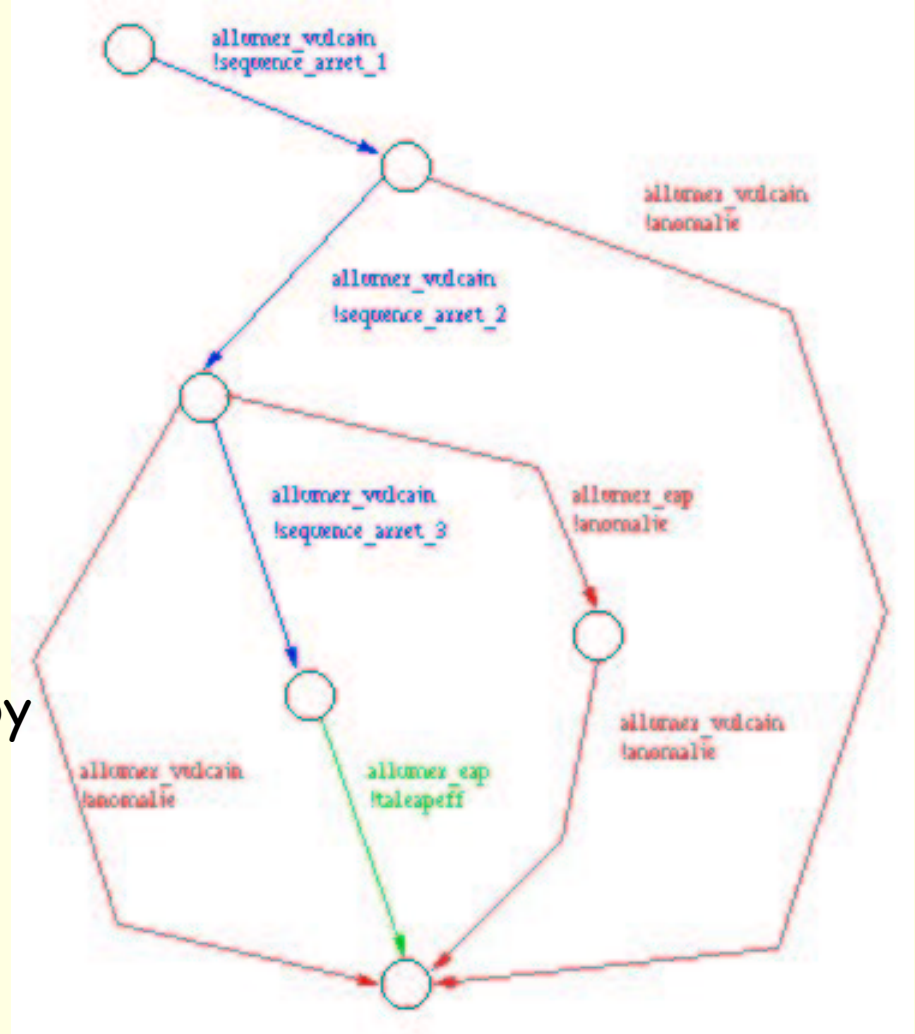
Ariane 5 case study : verification result

Property:

whenever a problem is detected before the ignition command to the main Vulcan engine is sent, then the ignition of the main engine and the secondary engine is stopped

Graph obtained from the global model by weak bisimulation minimisation

➔ Allows visual inspection of the graph

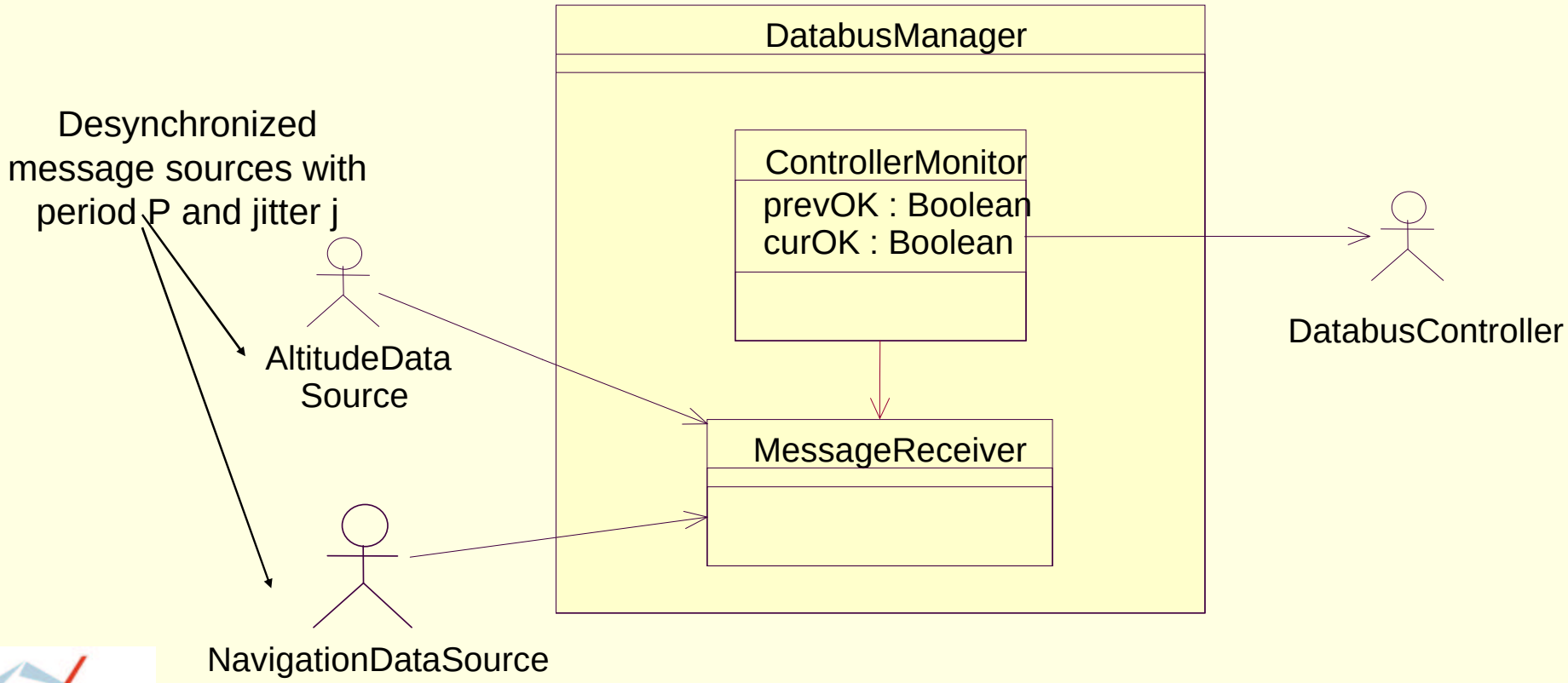


MARS Bus Controller

DatabusManager

Informal spec: timely recognition of bus error

bus error state for 3 periods no NavMsg or 3 times no AltMsg
ok again if 2 consecutive NavMsg and 2 consecutive AltMsg



Overview on the tool set capabilities

Concerning Mars case study

- system verifiable for 2 desynchronized data sources and fixed period and jitter, 3 sources → state explosion
- Verification of a parameterized version (n data sources, period p and jitter j) under work using PVS
- Found 2 subtle errors (counters reset at a bad moment + timing)
- When trying to remodel the system to separate the timed part (recognition of a signal/nosignal in each period) and the logic part (counting of messages) we made the system less reactive.
  - Error found (in a sequence of 3000 steps) and corrected counter examples and interactive simulation in ½ h
  - Hard to find error by testing

Overview on the tool set capabilities

General issues

- The translation from UML to IF introduces syntactic overhead, but no additional state explosion
- Manipulation of model-checker by users with some initial assistance from the tool builders
- Abstraction of data part: possible in a restricted way → separation of data and control part
- Property depending extraction of subsystems: slicing

➔ Proof of feasibility of integration of formal verification into the development process

Conclusions

A feasability proof of the integration of state-of-the-art verification tools in a UML based development by providing tools for a large subset of UML, ….

… but, also some problems:

- Exhaustive verification possible for small parts or abstract models

  ➔ apply more systematically component based validation
- XMI a standard ?
- Case tools make lots of restrictions making the definition of a profile open to several case tools hard (some do not export XMI)

Some references:

http://www-verimag.imag.fr/~async/IF

http://www-omega.imag.fr

Discussion points for the workshop

- **Expression of other non functional and probabilistic constraints**
  - How to get the necessary platform dependent estimations?
- **Modelling of real time systems**: what are the **right paradigms for implementation *and* validation?**
  - Integration of the synchronous approach in UML, extensions of it?
  - High level communication modes
- **Semantic level integration of different modeling tools**
- **Case tool independence vs Case tool integrated validation**? Tighter integration allows
  - Taking into account modeling patterns in a more optimal way
  - Coincidence of semantics in both tools: profit from CASE tool simulation facilities (problem is time)
- **Object orientation makes validation harder**
  - Are the advantages of OO worth this inconvenience (in RTES)?
  - More effort on static analysis of such systems
- **How to provide an "*intuitive semantics*" to the user?**
  - Semantic variation points:
    - Naming scheme for semantic level state changes (timed events)
    - In the tool: unconstraint object behaviors + global priorities
    - Is this appropriate also as a user level semantics
  - Concepts defined as stereotypes:
    - Map into more primitive concepts, but which ones