Spécification objet UML et vérification formelle de systèmes critiques

Mise en œuvre sur deux applications industrielles

Bernard Dion
Esterel Technologies

Journées Systèmes et Logiciels Critiques
Grenoble
14 novembre 2000

Contact: Bernard.Dion@simulog.fr

Contents

- Objectives
- Reactive vs algorithmic parts
- UML extensions
- Esterel Studio for UML
- Example 1: Fuel management
- Example 2: Trusted Third Party (TTP) protocol
- Conclusions: Key features & benefits

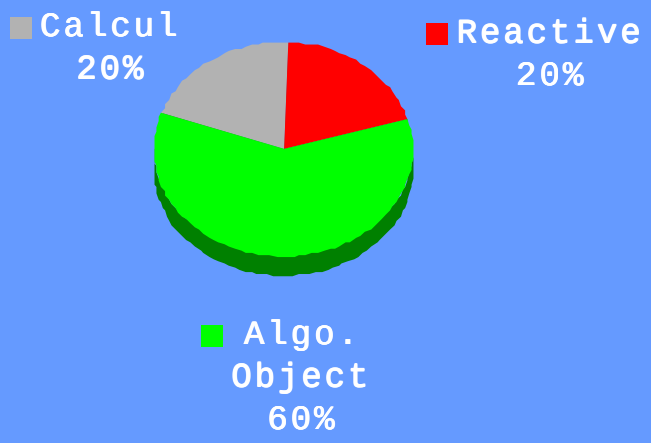
Objectives
Increase development productivity

- Object Oriented approach
  - interfaces separation / internals representations / behaviors
  - formalization of similarities / specialization
- Graphical specification
  - readability, ease of use
- Graphical simulation
  - animation of the specification
  - co-simulation
- Formal Verification
  - conformity to specification
- Code Generators:
  - time saving
  - no programming (coding) error

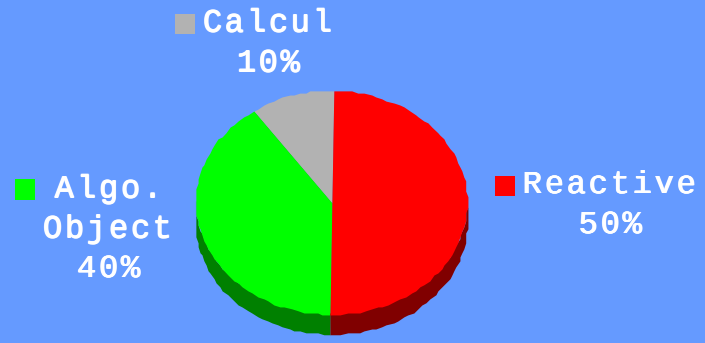
Reactive vs *Algorithmic (1/3)*
Complexity

- The reactive part is smaller in size
- However, it concentrates much of the complexity in the specification

Code Size



Evolution and Modification



(source: Dassault Aviation)

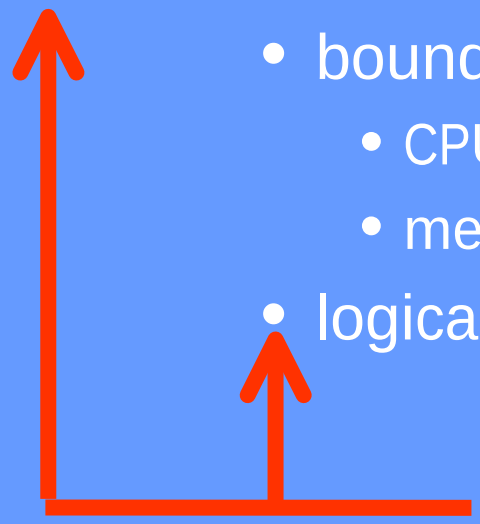
Reactive vs *Algorithmic (2/3)*
Expected properties

Reactive part

- **Interaction through events**
- **Parallelism**
  - **between objects**
  - **within objects**

Algorithmic part

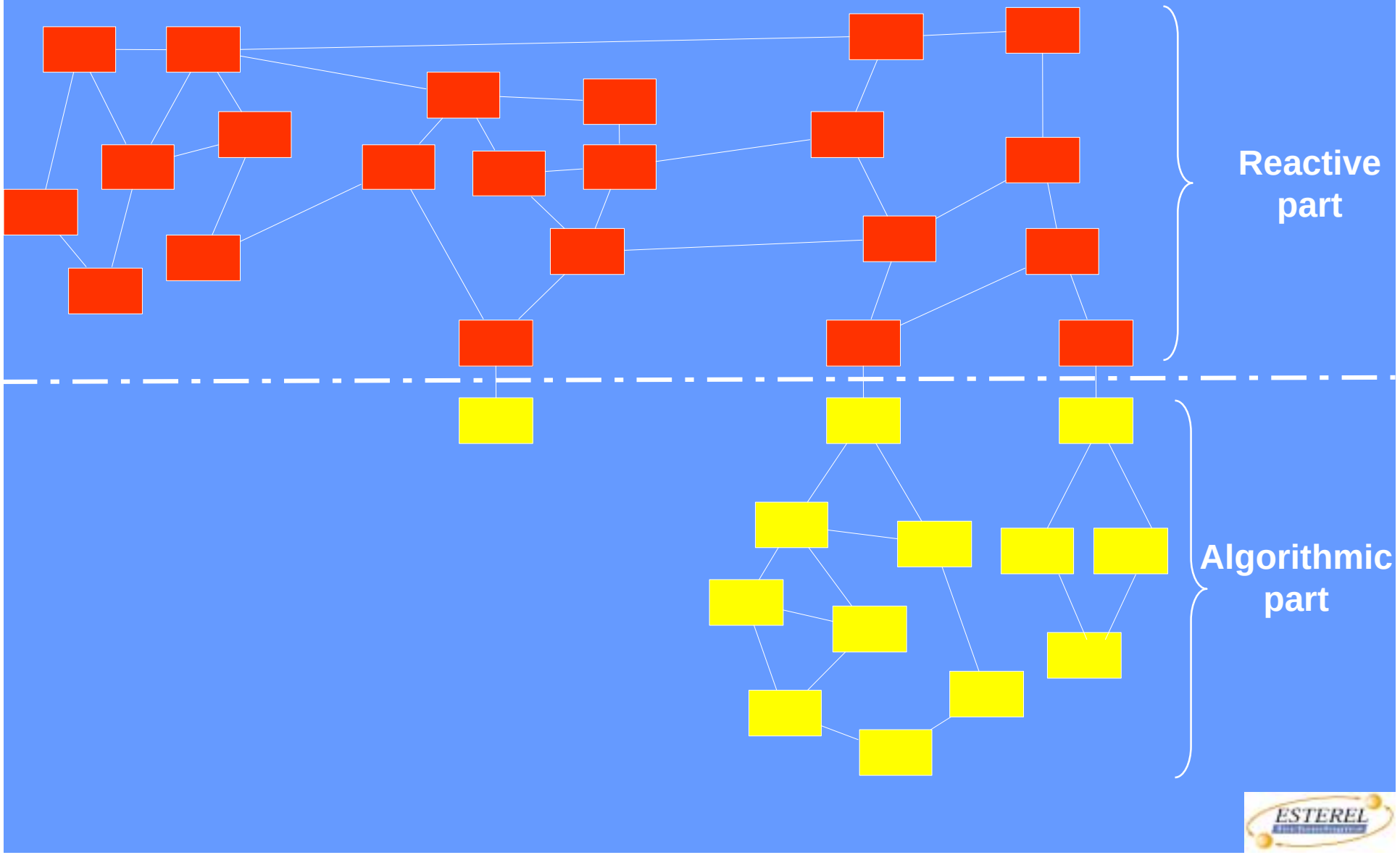
- **Interaction through messages**
- **Messages are handled sequentially**



- bounded consumption of resources
  - CPU
  - memory
- logical and temporal predictability

Esterel Studio

Reactive vs *Algorithmic (3/3)*
A clear separation



UML for Real Time applications

- The UML standard allows extensions
  - stereotypes
- UML for Real Time background work
  - Objectime and Rational proposal
  - stereotypes and approach coming from ROOM method
- Esterel Studio for Rational ROSE
  - is a prototype, commercially available soon
  - adopts UML for Real Time stereotypes
  - *+ Synchronous semantics*

UML extensions for Reactive Systems (1/2)

UML Entity	Corresponding UML-RT Stereotypes	Constraints
class	<<capsule>> *reactive class* <<protocole>> events interface	Compulsory stereotype for reactif/algorithmique distinction
operation	<<input>> *events* <<output>> <<port>> typed by a *protocol* <<port~>> <<plug-in>> *explicit associations connection*	Compoulsory Stereotype
association	Standard semantics	Fixed cardinality Unidirectional navigation
aggregation	Standard semantics *or* <<port>> *graphical feed-back (opt.)*	Fixed cardinality Unidirectional navigation

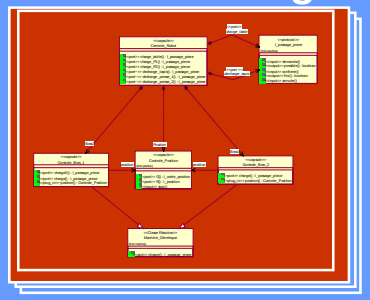
ESTEREL

UML extensions for Reactive Systems (2/2)

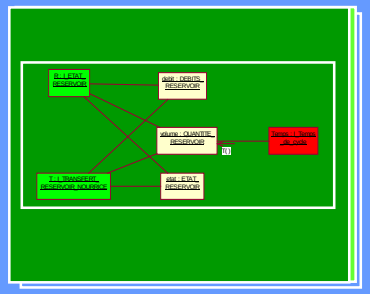
- A structure diagram for each reactive class
  - ➔ static construction of the graph of instances
    - ➔ parameters
    - ➔ interconnections
- SyncCharts replace StateCharts
  - ➔ weak and strong preemption
  - ➔ possibility to exploit simultaneity of events
  - ➔ mandatory priorities on transitions
  - ➔ power of expression of textual Esterel for actions

Esterel Studio for Rational ROSE (UML)
How Does It Work?

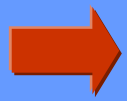
ROSE class diagram

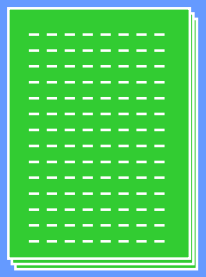


Structure diagram
(now in Rose, soon in Esterel Studio)

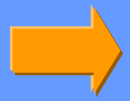


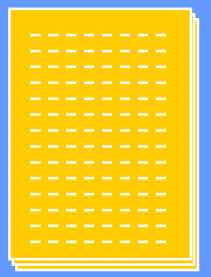
Esterel Studio SyncCharts





Esterel



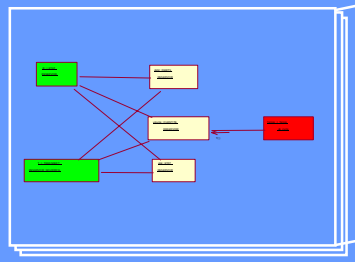


C

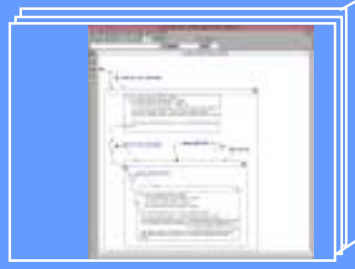
Esterel Studio for Rational ROSE (UML)
How Does It Work?

Rose Class Diagram

Structure diagram



Esterel Studio SyncCharts



```
module reservoir :
input commande;
output niveau : float;
...
signal local__0, … in
run controle_niveau […]
||
run controle_vannes […]
||
…
||
loop
  await init;
  …
end loop
end signal
end module
```

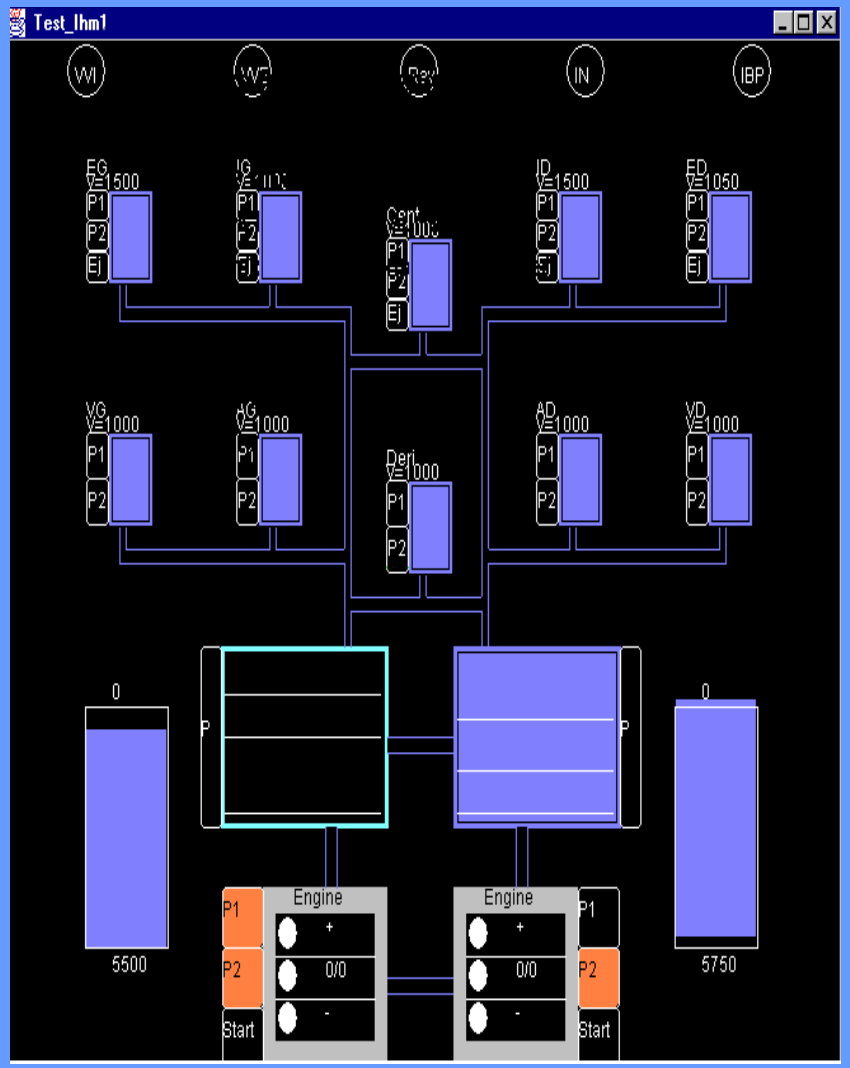
Capsule name and its interface

Subcapsuls activation and connexion

Behavior traduction in textual form

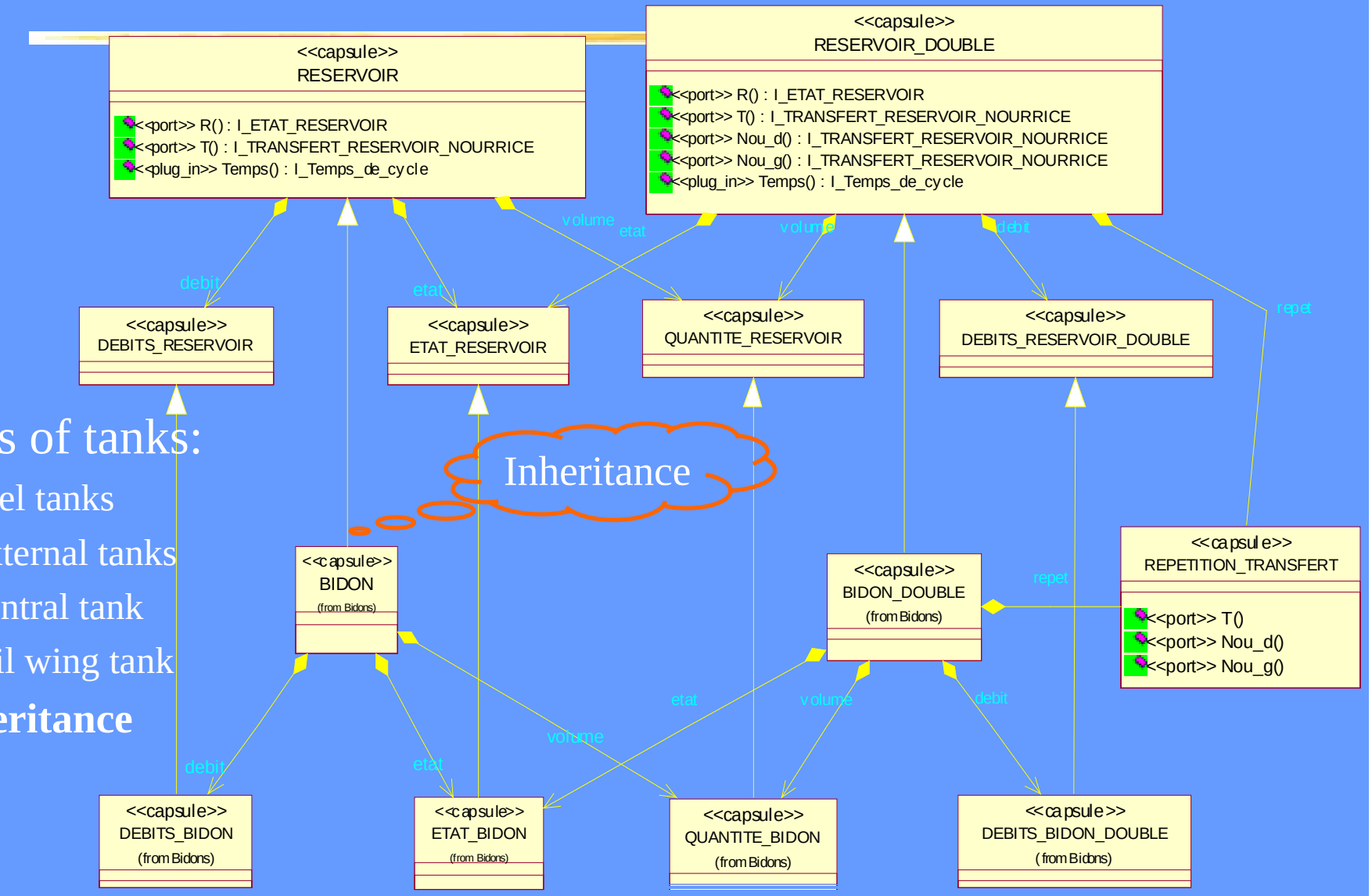
Example 1
Fuel Control (Dassault Aviation)

- Military aircraft
  - 12 fuel tanks (7 internal and 5 external)
  - 2 jet engines
- Main Functions:
  - Choose the tank for engine fuelling
  - Emergency fuel dump
  - Aerial refuelling
  - Failure monitoring
    - tank failures (pressurisation)
    - pump failures (transfer or booster)
    - engine failures
  - Pilot information
    - fuel flow monitoring
    - fuel remaining



Click on the image
to launch the application (1)

Fuel Control Inheritance

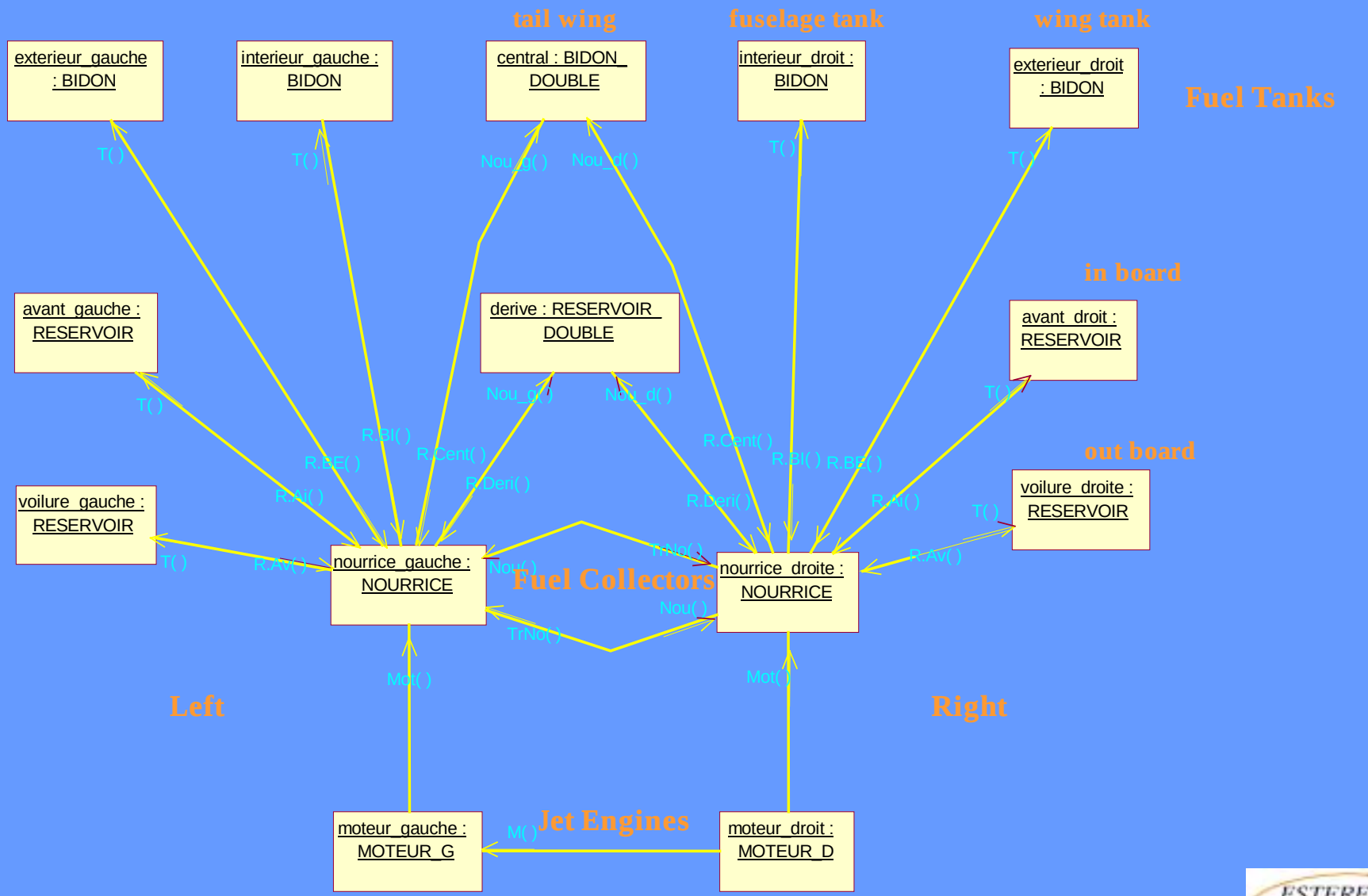


4 types of tanks:

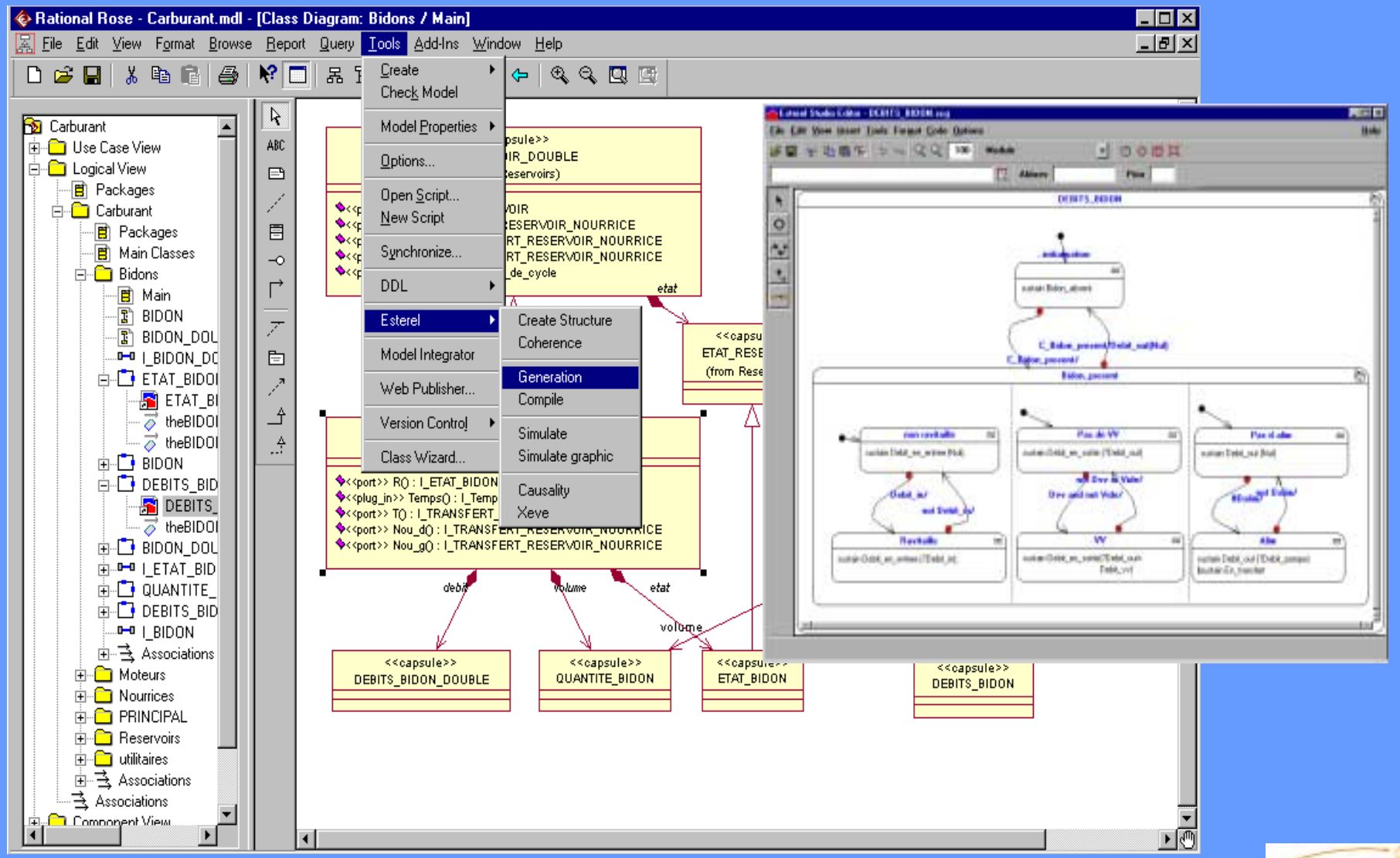
- fuel tanks
- external tanks
- central tank
- tail wing tank

=> **inheritance**

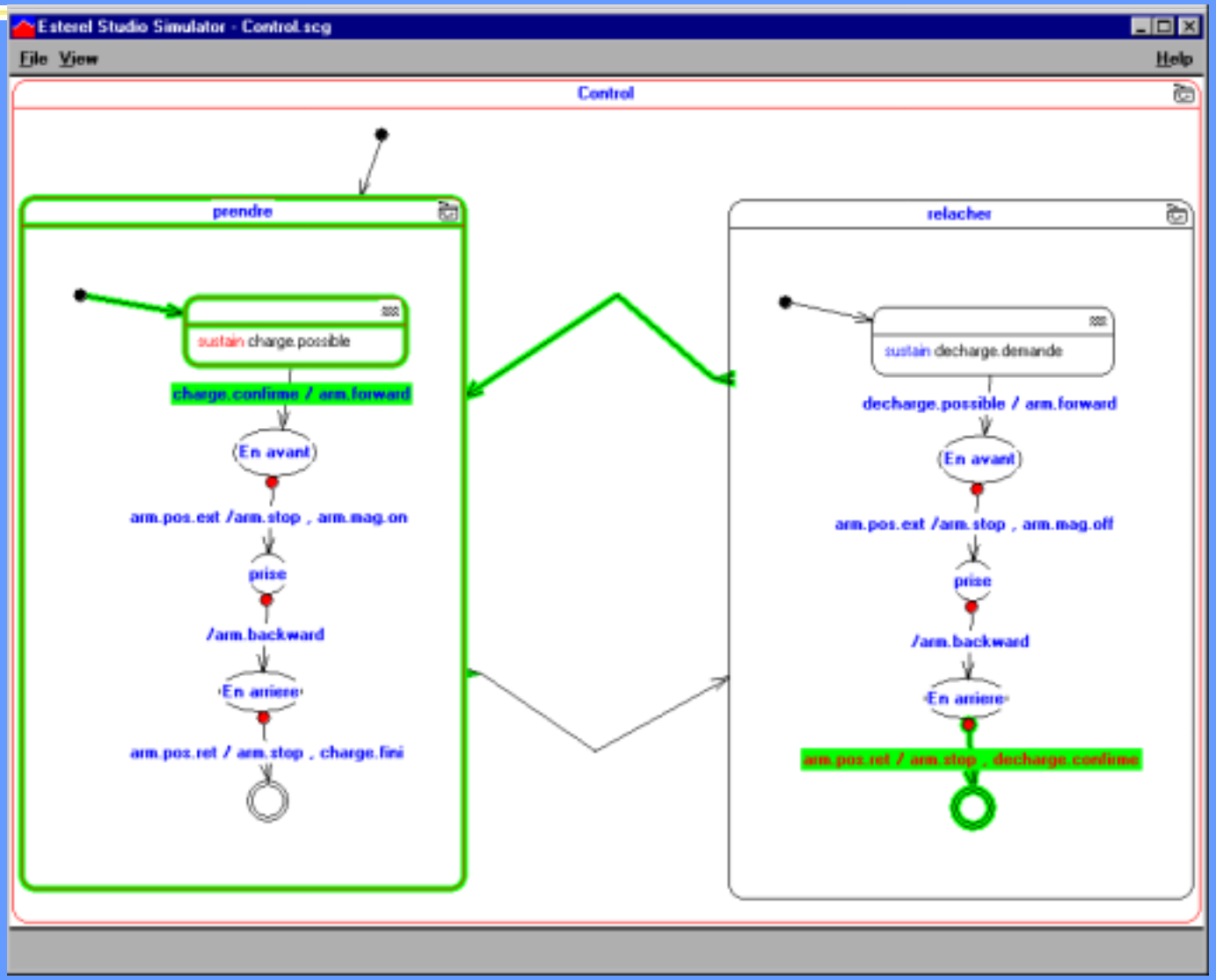
Fuel Control
Structure Diagram



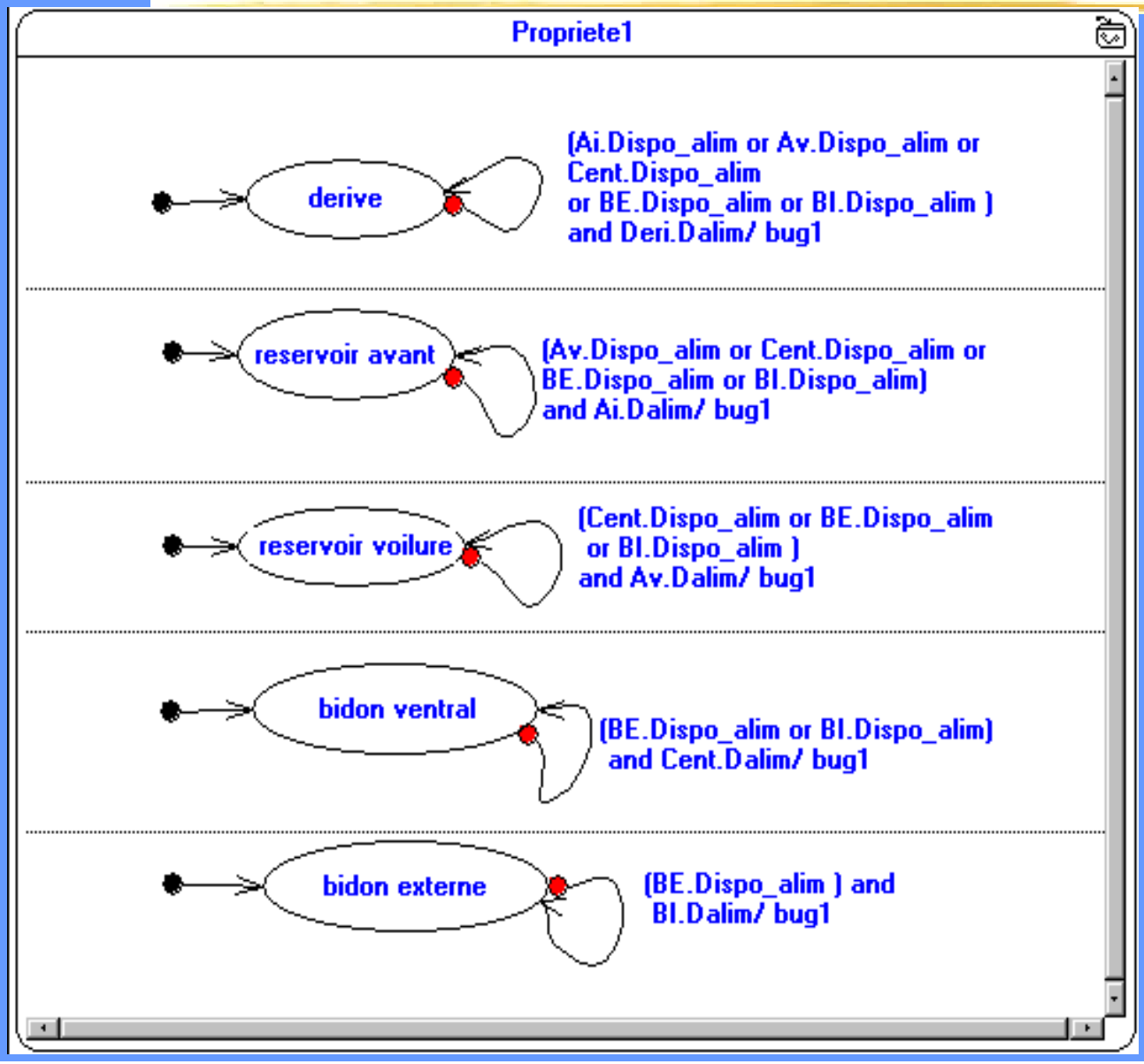
Fuel Control
Generation & Compilation



Fuel Control Simulator



Fuel Control
Proving a Property



Lowest priority

Observer is to check property:

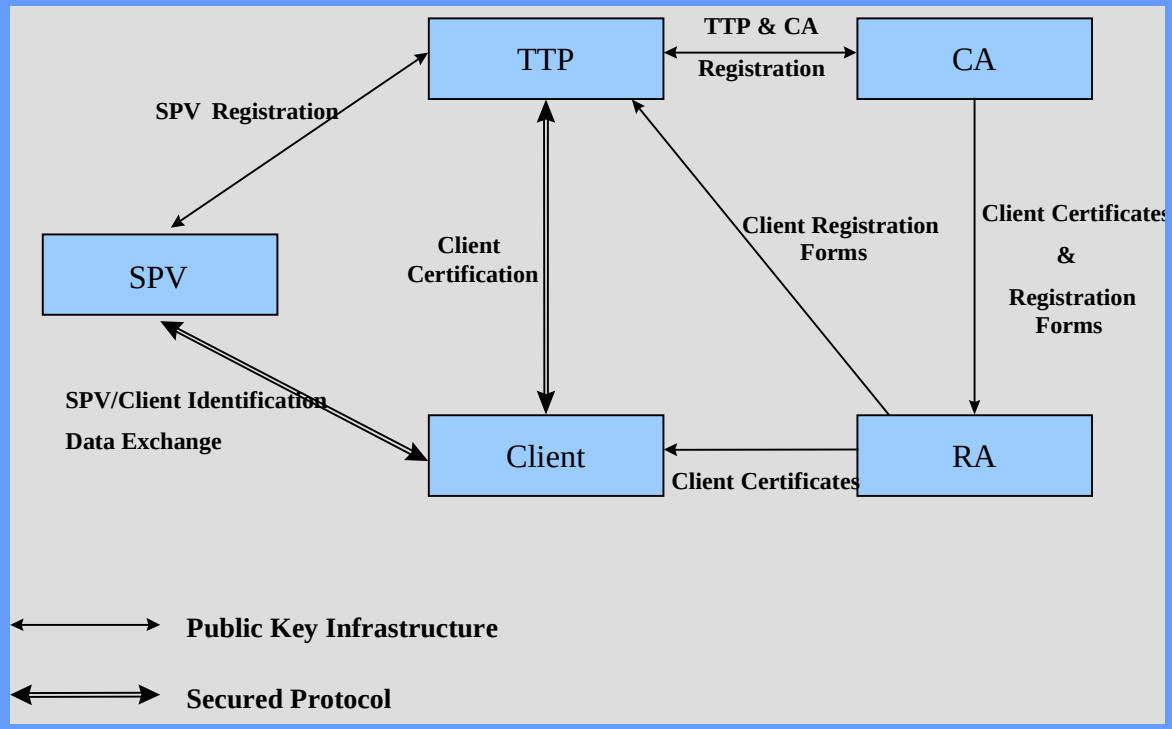
« Fuel collector asks fuel transfer from first available tank in a given order »

Highest priority

Fuel management
Project Achievements

- 1.5 person *x* month
  - specification
  - object structure design and SyncCharts
- The program
  - 17 SyncCharts
  - 4600 Esterel generated lines
- Efficient code generation (ADA on power PC 601 66Mhz):
  - 350 µs (using Esterel code generation optimizations)
- Used in operations at Dassault Aviation

Example 2
Trusted Third Party (Thomson CSF)



THOMSON-CSF
COMSYS

ESTEREL

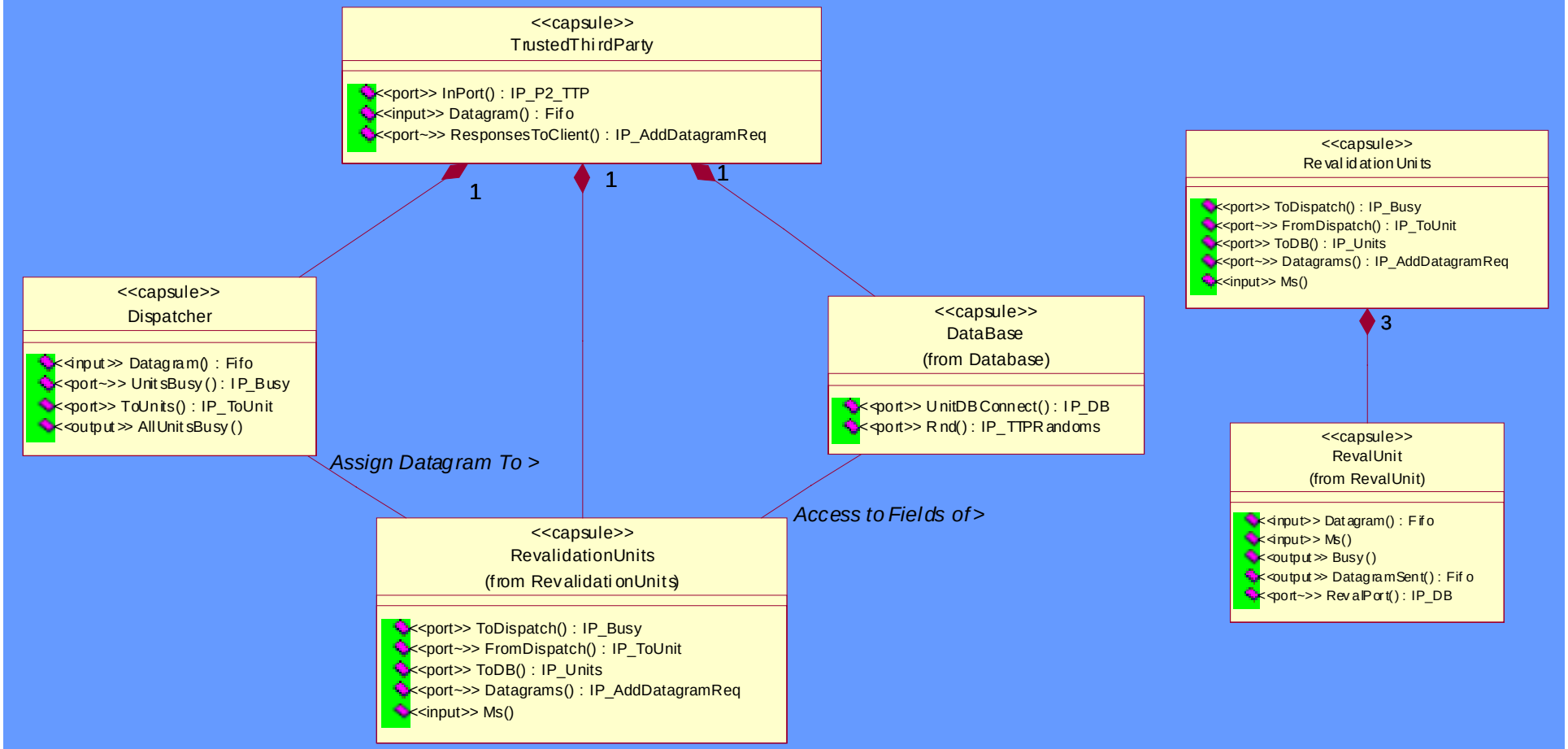
TTP Development

- Modeling
  - methodological support
    - with UML Approach : structuration
      - structure and behavior was mixed with dynamics in previous model
    - graphic formalism => introduction of //
      - Parallelism in cryptographic computations and database access
- Validation
  - Design patterns for observer, proof of properties
  - Co-simulation
- Integration
  - Coupling with C/C++ functions (algorithms)
  - Benchmarking (Esterel versus C/C++)

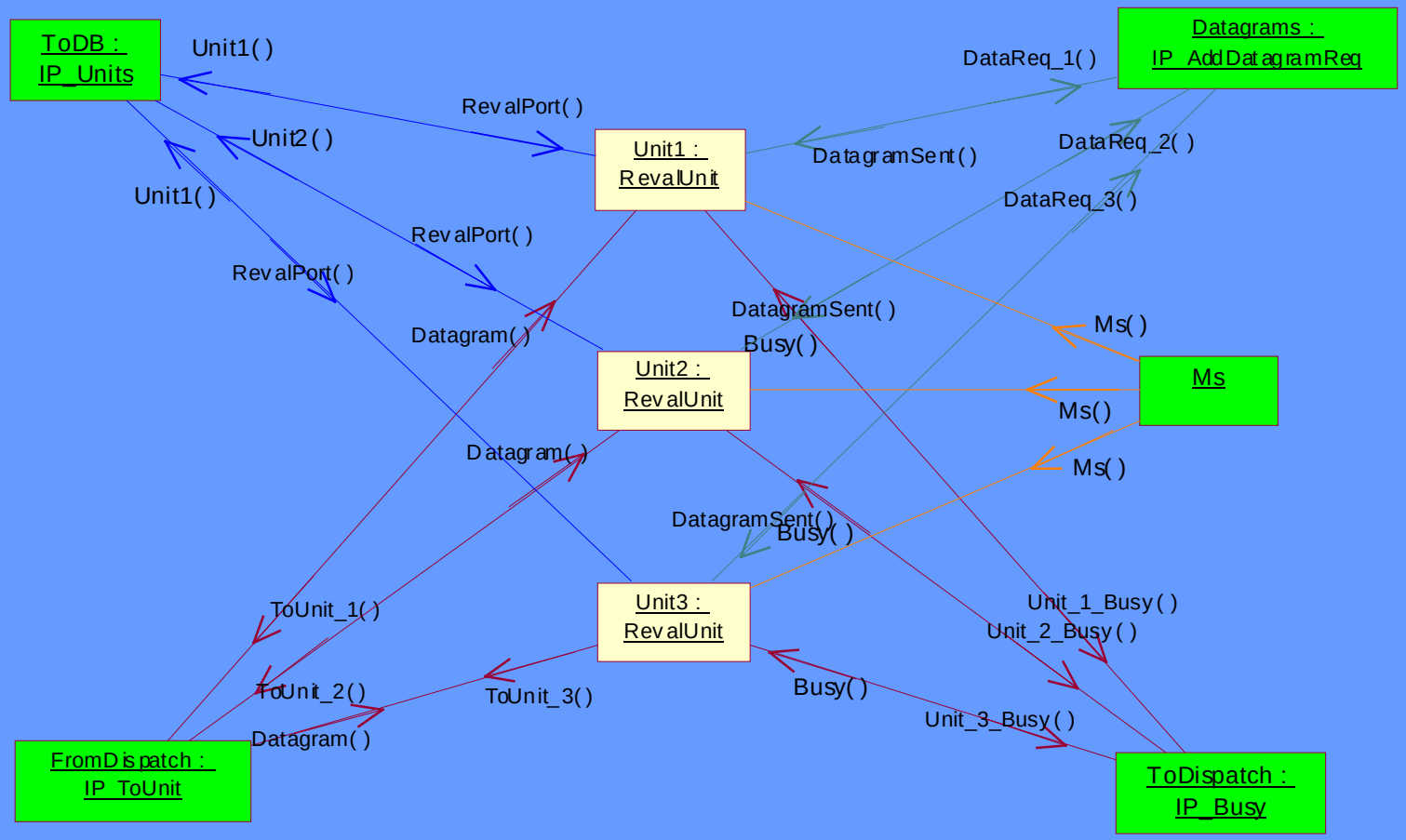
TTP Requirements

- Two main objectives
  - Security
  - Efficiency
- Requirements
  - to treat several users requests simultaneously
  - to allow non-blocking communication with Database
  - to maximize the utilization of time slots available

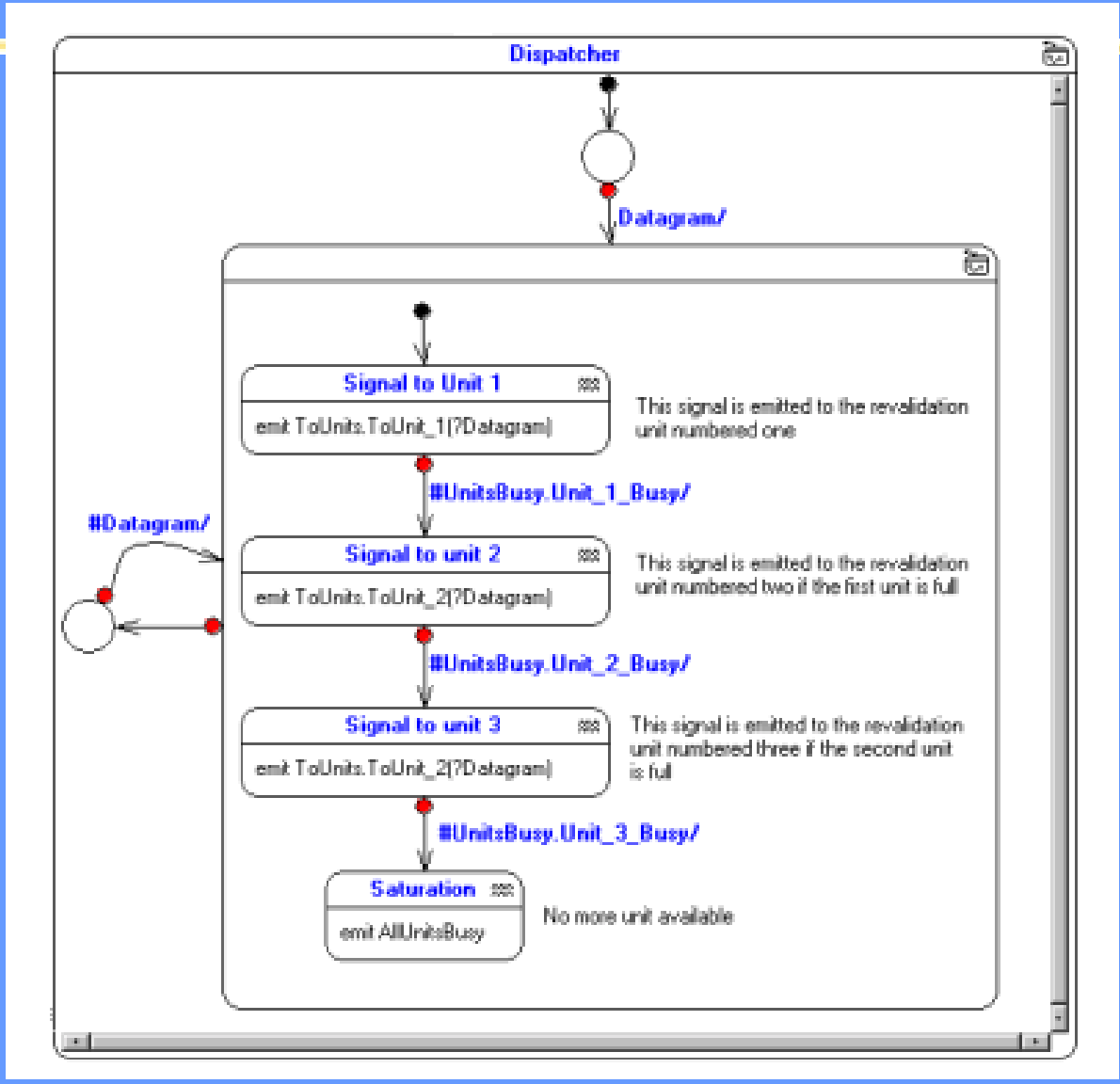
TTP
Class diagrams



TTP
Structure diagrams



TTP
SyncCharts

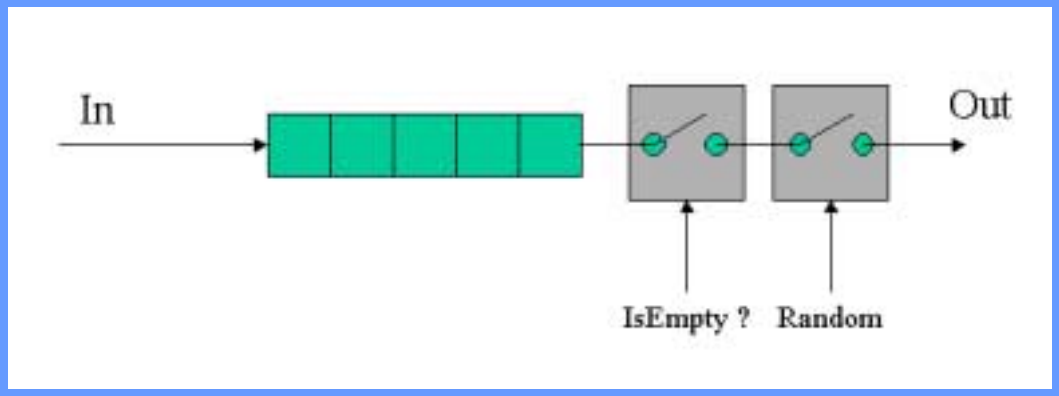


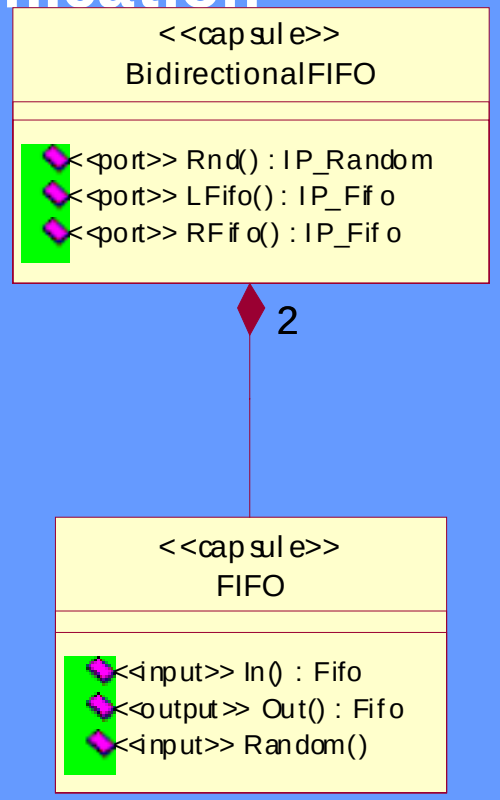
TTP
Data Base access

- Use of a Design Pattern :FIFO
  - **potential asynchrony of the communication**

  (more than 10 instances in TTP Modeling)

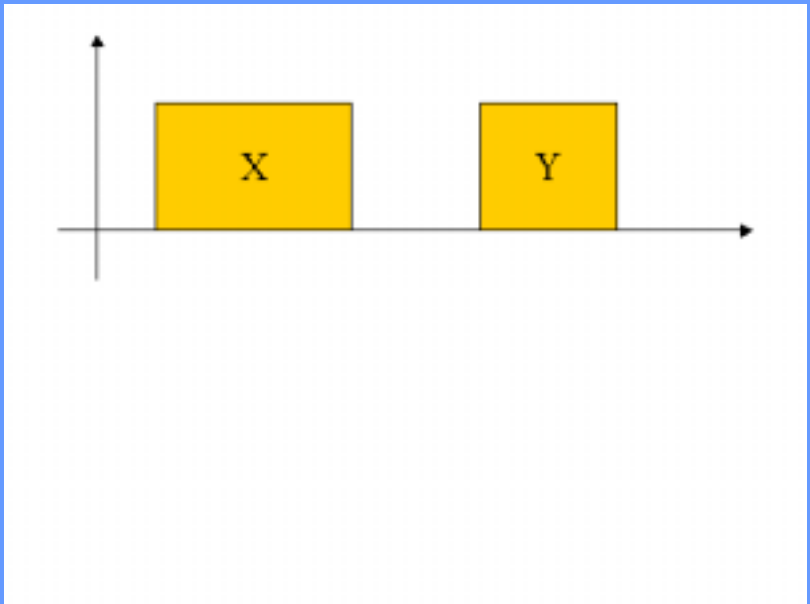
- One Channel package (2 FIFOs)

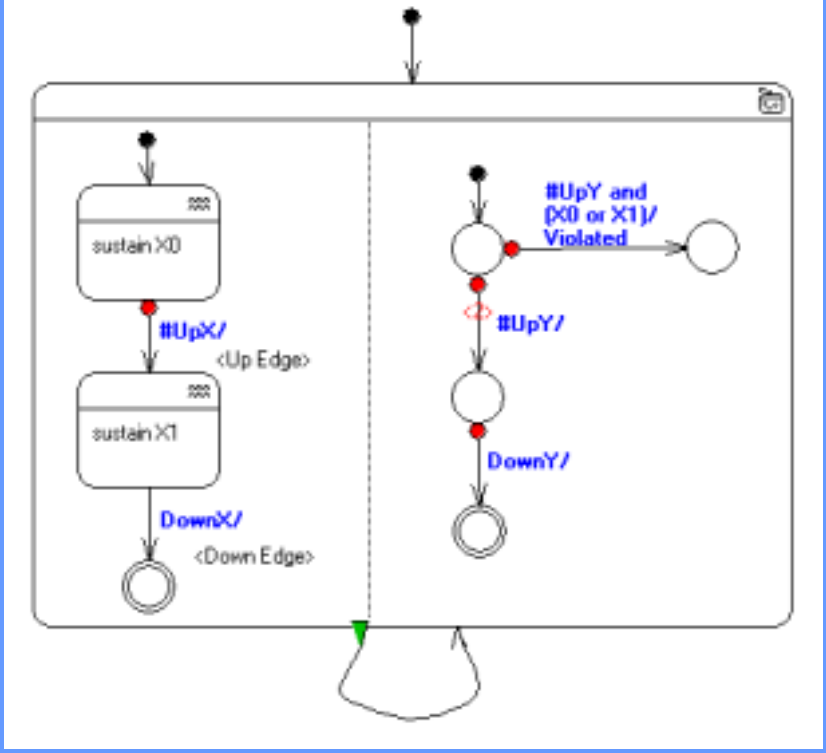




TTP
Verification patterns

X before Y pattern



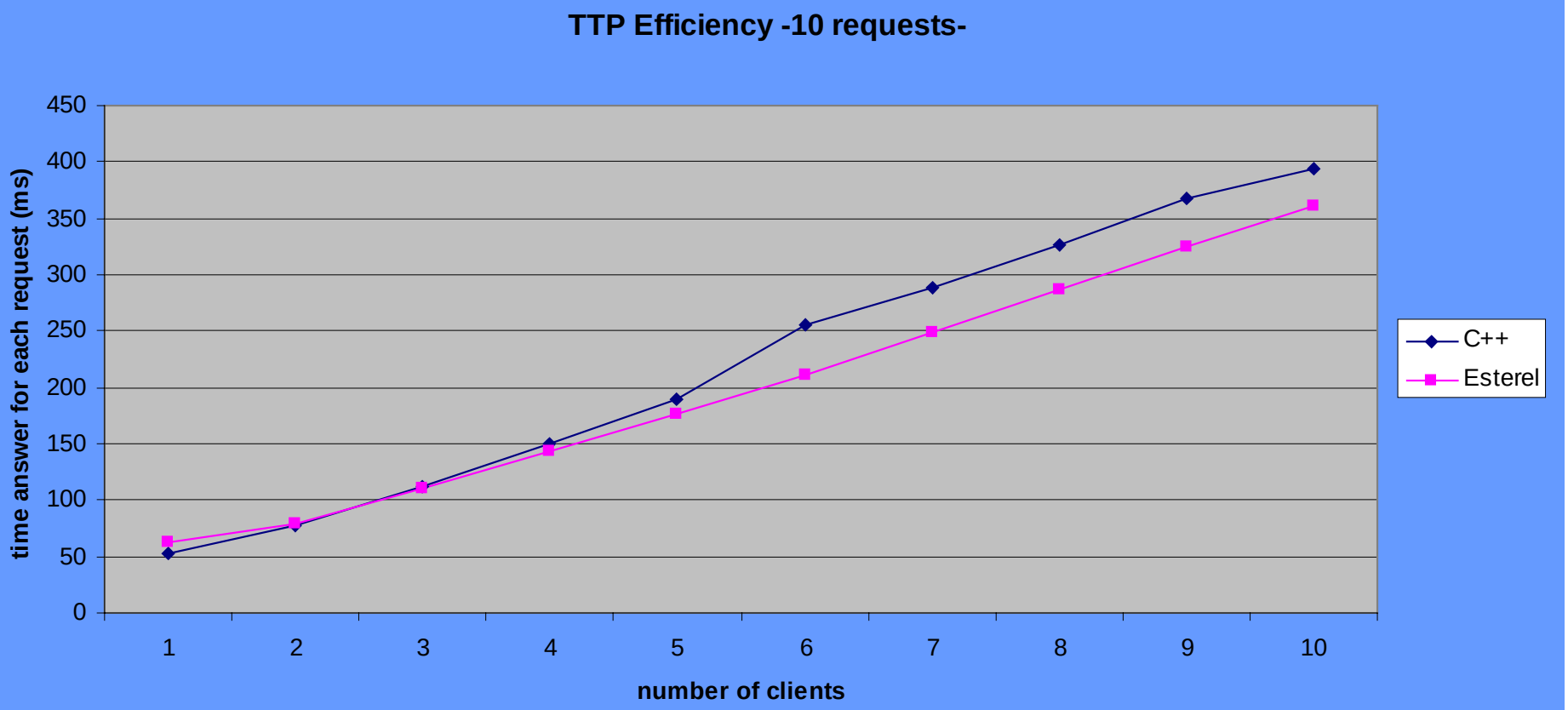


TTP
6 proven properties

- P1 : Does a revalidation unit always send a response datagram at the latest Z millisecond after the corresponding revalidation request ? *(30 minutes sur PC)*

- .....

TTP Performance *(no optimization)*



TTP
Project achievements

- Model and performances
  - 10 class diagrams
  - 10 structure diagrams
  - 9 synccharts
  - 45 input, 8 output
  - 1500 lines of esterel generated code
  - 8000 lines of C generated code
  - 2 000 000 reachable states
  - 4 person x months
  - performance identical to C++ handwritten code for 10 users (no optimisation)
  - co-simulation

Conclusions
Key Features & Benefits

- Uniform and standard object-oriented framework for analysis and design
- Clear separation between reactive and algorithmic parts
- Possibility to use libraries of available application oriented components
- Formal verification and automatic tests generation
- Automatic generation of very compact code

ESTEREL

Spécification objet UML et vérification formelle de systèmes critiques

Mise en œuvre sur deux applications industrielles

Bernard Dion
Esterel Technologies

Journées Systèmes et Logiciels Critiques
Grenoble
14 novembre 2000

Contact: Bernard.Dion@simulog.fr