Persiform:
Performance Engineering Based on Simulation of Formal Functional Models

Olivier Constant, Marius Bozga, Susanne Graf -- Verimag, Grenoble
Nicolas Moteau, Wei Monin -- France Telecom R&D

2007 April, 20th

research & development

orange™

The **PerSiForm** project : Towards the integration of performance simulation in functional design

- French national project (RNRT)
- Started Nov. 2004, ends August 2007
- Partners
  - 3 academic labs

  - 2 companies

(leader)

summary

1. **Persiform methodology and tool chain: Performance models from annotated Functional Models**
   - Objectives and Constraints: methodology
   - Persiform Modeling concepts
   - Methodology implementation by systematic model transformation
   - Example
2. **Application of the Persiform Tool Chain and Industrial Perspective**
   - Case study description and demo
   - Simulation’s results vs. measures realized on the application
   - Perspectives for France Telecom

1

Performance Models from Annotated Formal Functional Models through Systematic Model Transformation

Outline

- Objectives and constraints

  A methodology for performance evaluation throughout the development of performance critical service oriented systems
- Contribution
  - PerSiForm modeling concepts
  - Implementing the methodology by systematic model transformations
- Example
- Conclusion & perspectives

Objectives : *a methodology for performance evaluation throughout the development of performance critical service oriented systems*

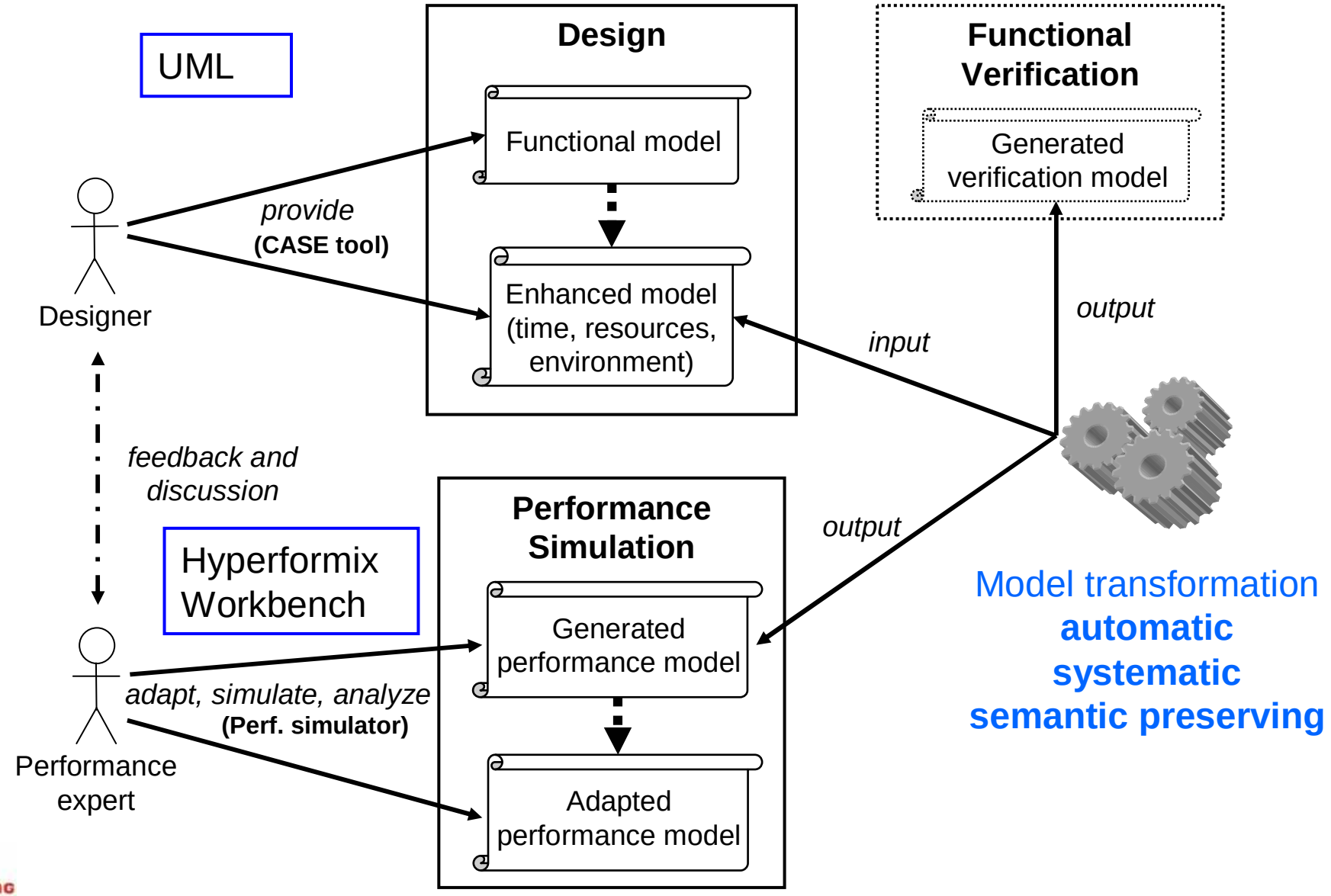
Performance of complex systems

- Complex **service-oriented** system specifications
  - Distributed
  - Multiple usages of services / components
  - Large number of users
- Criticity of their **performance**
  - Controlling end-to-end response time, throughput
  - Controlling the cost of deployment
- Need for performance analysis throughout the **entire** design process
  - Early: evaluate architectural decisions
  - Late: determine an acceptable infrastructure

Performance of complex systems (2)

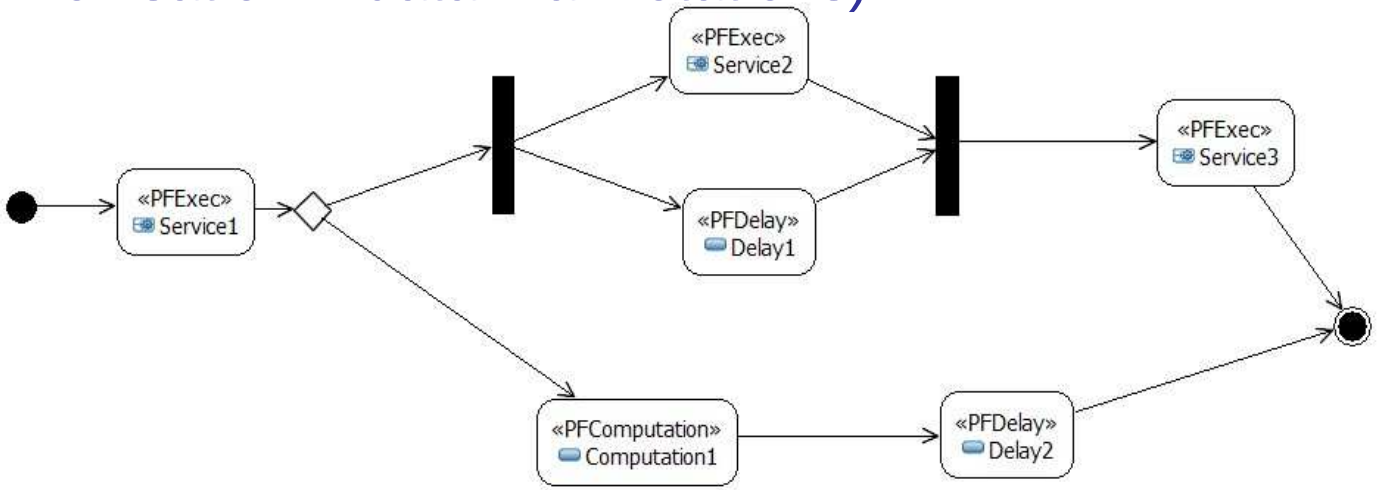
- Existing analysis techniques
  - Analytical methods
  - **Simulation** (expressiveness, scalability, applicability)
  - Test
- Problems
  - Consistency between functional and performance models
    - Functional verification vs. performance analysis
  - Performance modeling is an activity for experts
    - Requires dedicated professional tools

Methodology

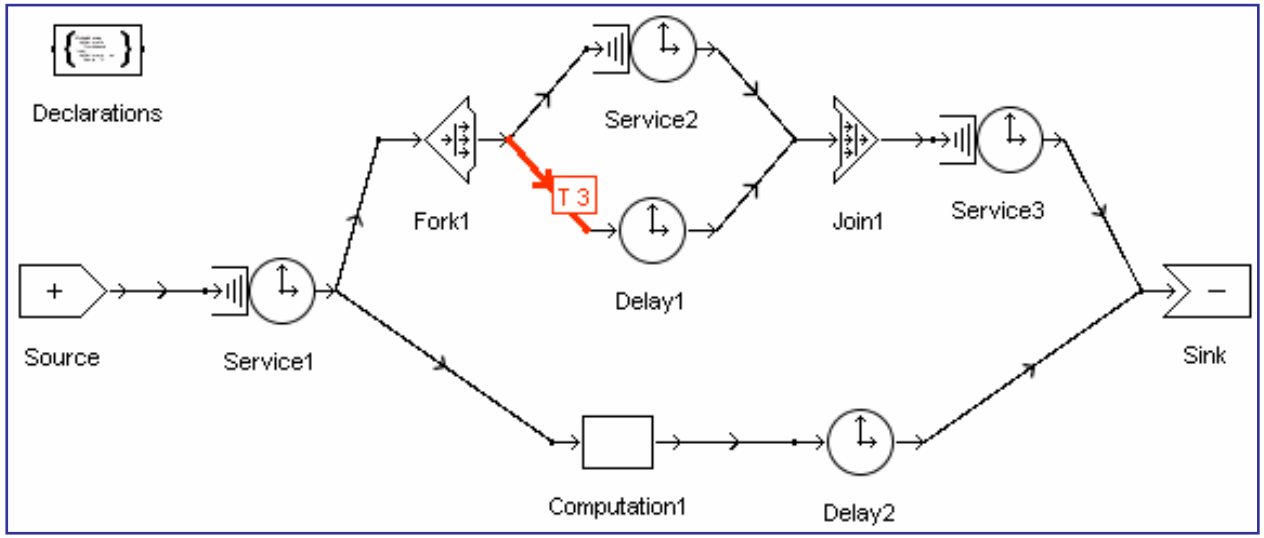


Models in Design and Performance Case Tools

- A UML activity Diagram: functional view (causality + synchronisation + data + annotations)



- A queuing network in *HyPerformix Workbench:*



focusing on resource consumption, characteristics of resources, environment (arrival of service requests)

PerSiForm modeling concepts

Categories of modeling concepts

- Structure
  - its impact on the behaviour (deployment)
- Behaviour
  - Causal flows
  - Synchronizations
  - Behaviour decomposition
  - Data
  - Resource consumptions
  - Time consumptions

Activity Diagrams

+

Data, Actions

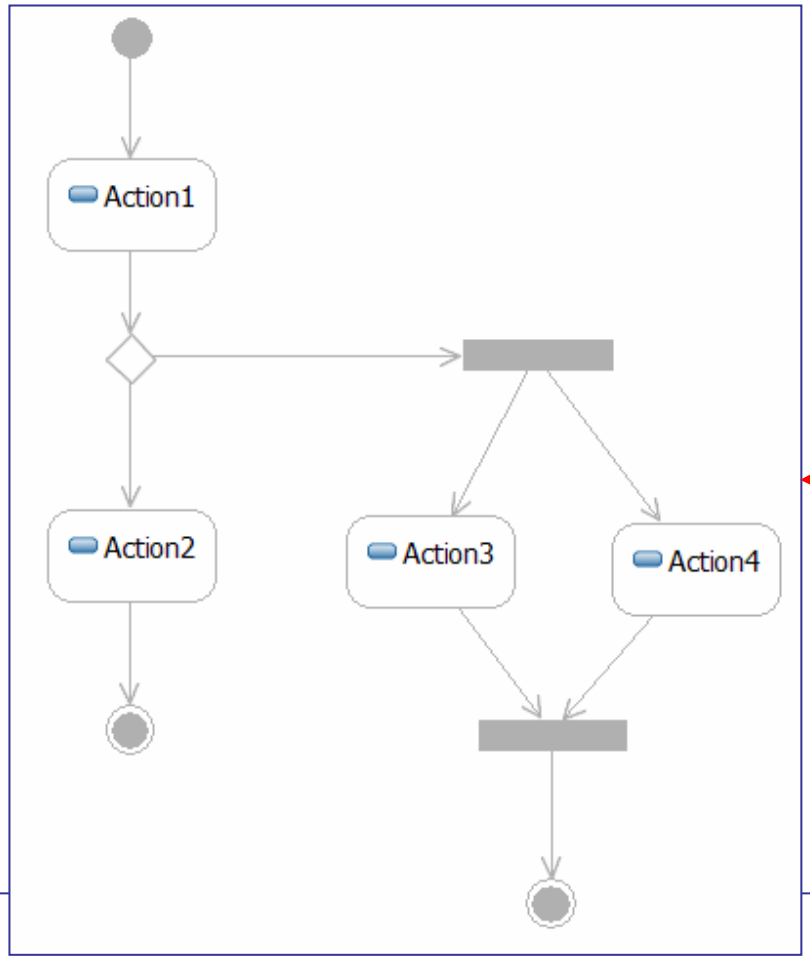
+

Annotations

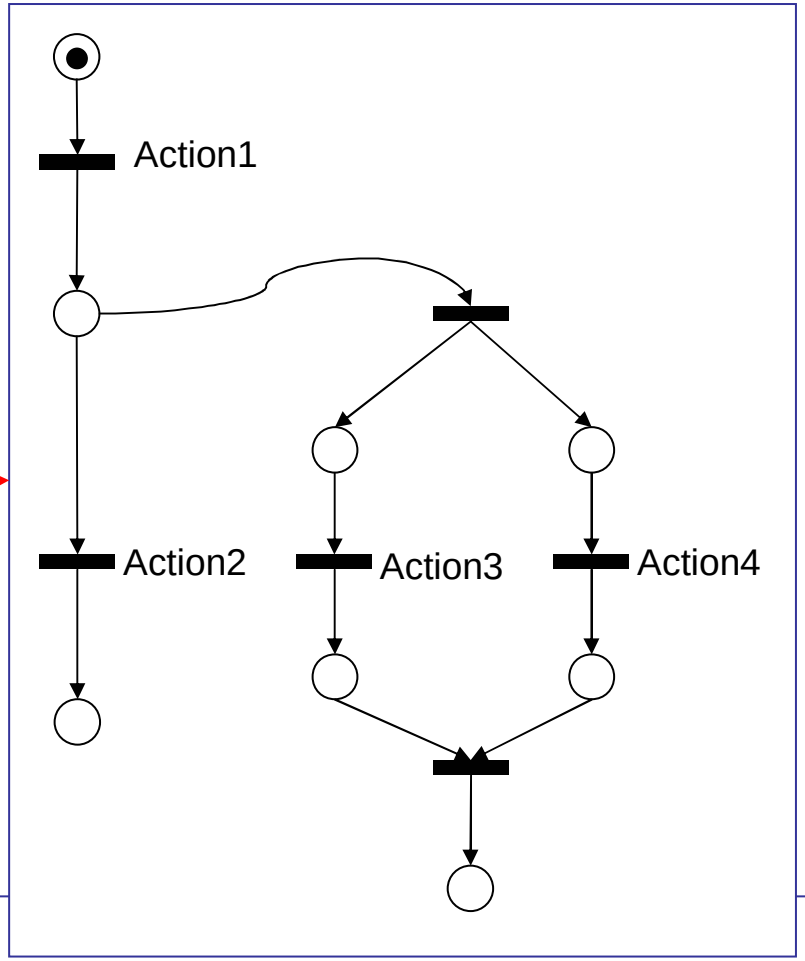
- Environment (request arrival patterns)

Behaviour: Causal flows

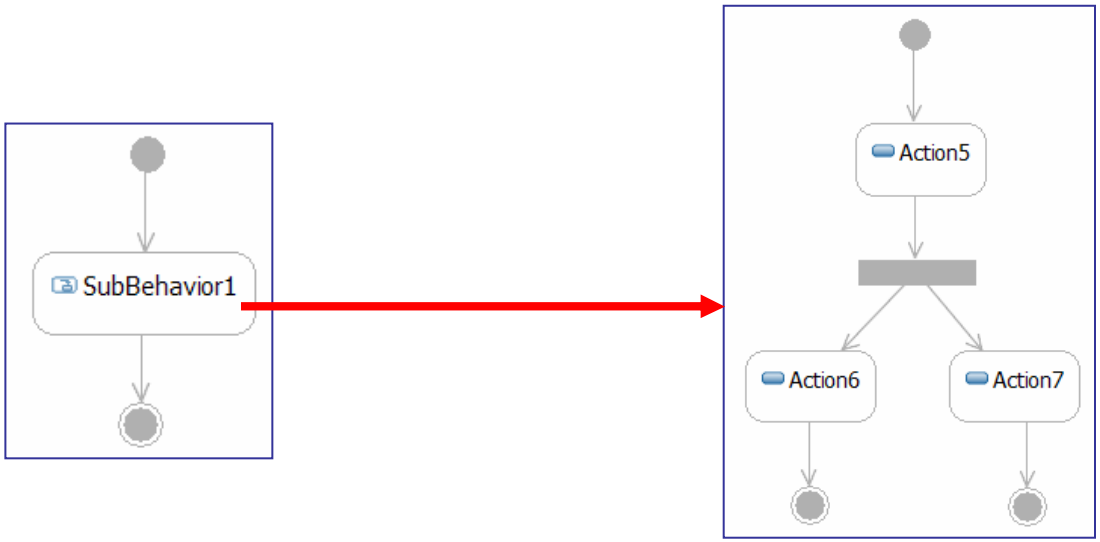
A possible representation
(UML Activity Diagrams)



A formal definition
(**Petri Nets**)



Decomposition

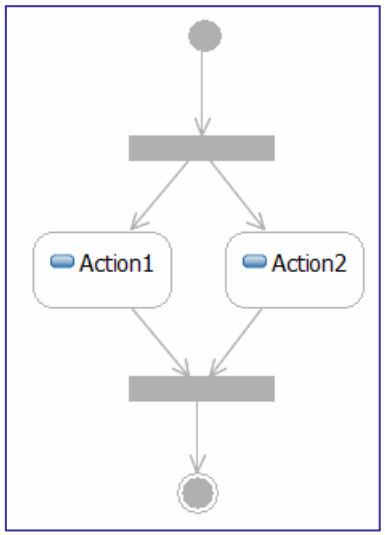


- Reusable sub-behaviours with parameters
  - ‘‘Procedural’’ Petri Nets without recursion
- A sub-behaviour may have **concurrent executions**

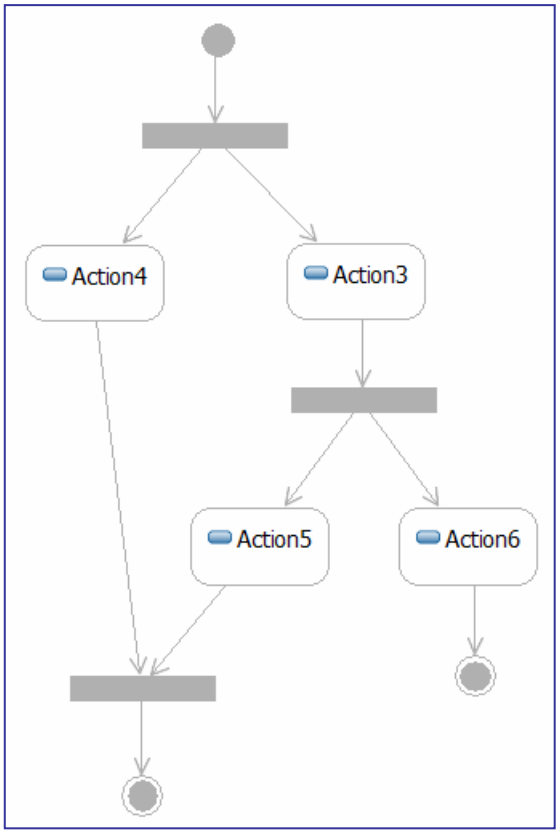
Decomposition (2)

- Possible sub-behaviour terminations
  -  Return — Equivalent to sub-behaviour inlining
  -  Stop — Thread destruction
  -  Interrupt — Return + destruction of all threads in the same execution
  -  Wait — Return once per execution, when all threads in the same execution have been destroyed

Causal flows and synchronizations (defined by designers)

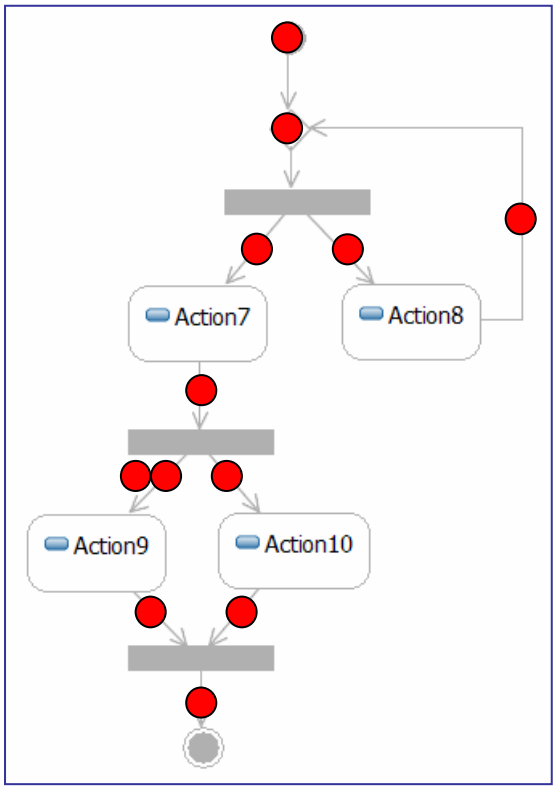


classical fork / join pattern



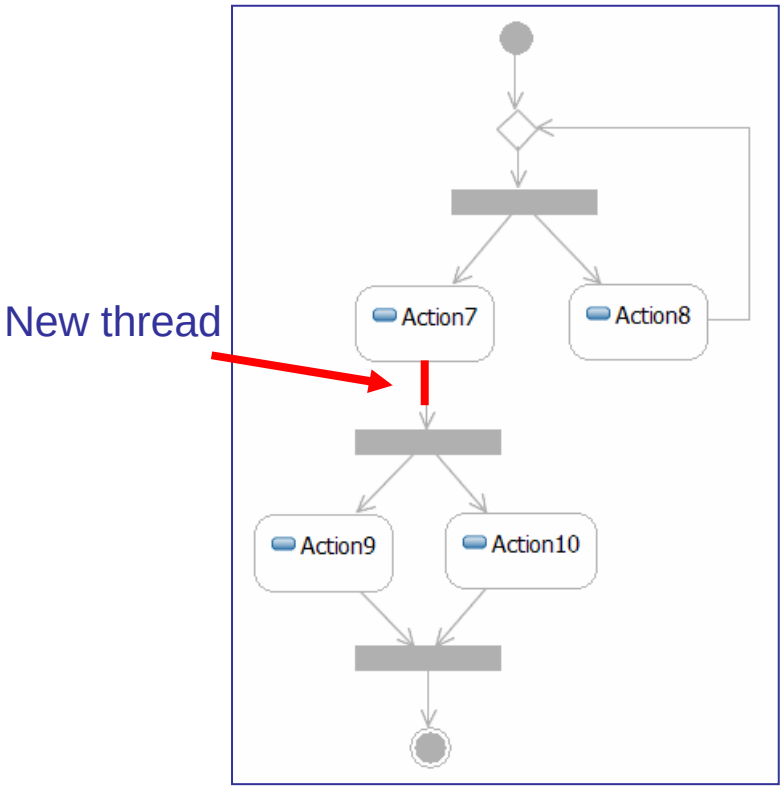
arbitrary fork / join patterns

Causal flows and synchronizations (2)



Loss of causality thread!

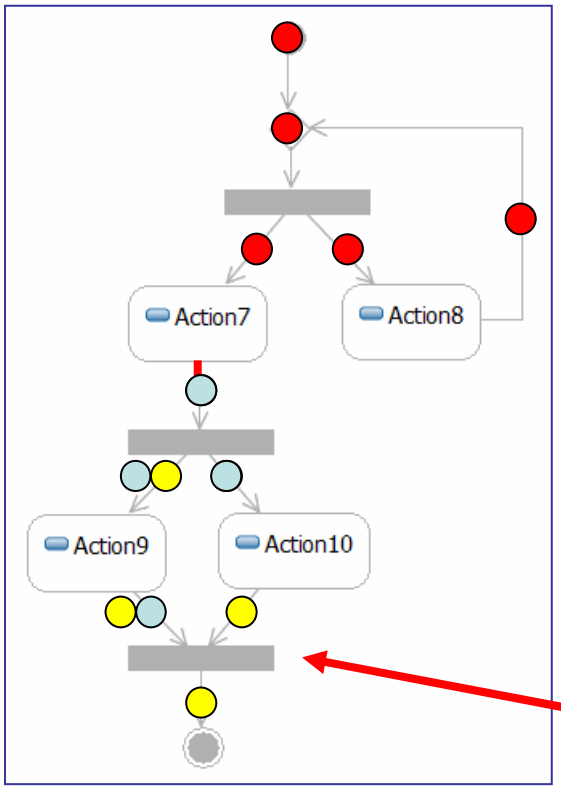
Causal flows and synchronizations (3)



Solution

- Use colours (coloured Petri Nets)
- Allow edges to be marked as ‘‘**new thread**’’
- Synchronize on colours
- But also support ‘‘weak’’ synchronization

Causal flows and synchronizations (4)

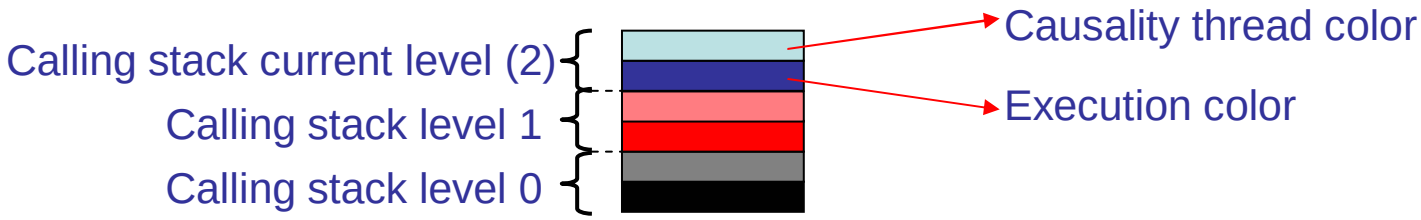


Strong synchronization

Causality is preserved

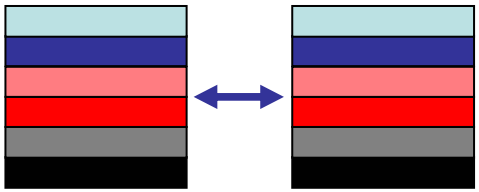
Decomposition (3)

- Semantics: each token owns a **stack of colours**
  - Entering a sub-behavior means **pushing 2 fresh colors**
  - Exiting a sub-behavior means **popping twice**
  - "New color" changes the topmost color only
- Rationale
  - The topmost color identifies the **causality thread**, as introduced for flat behaviors
  - The second color identifies the **execution** of the sub-behavior that the thread belongs to

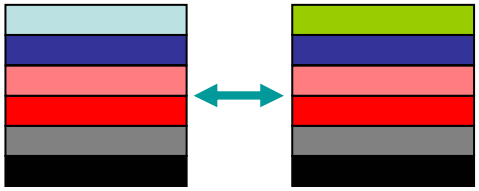


Behavior: Decomposition (4)

- Termination semantics clarified
  - Tokens belong to an execution E iff their stack contains the color of E (independently of further sub-executions)
- Synchronization rules redefined
  - Strong synchronization requires stack equality



  - Weak synchronization requires stack prefix equality



Resource consumptions (defined by performance experts)

ProcessingUnit
- Queuing discipline
- Nb of processors
- Max threads nb
(- Max priority)
(- Preemptible or not)

deployment → Action → execute (*duration*, *time unit* [, *priority*])

Memory
- Size

deployment → Action → allocate / free (*size*)

TokenPool
- Queuing discipline
- Capacity

deployment → Action → take / release (*number*)

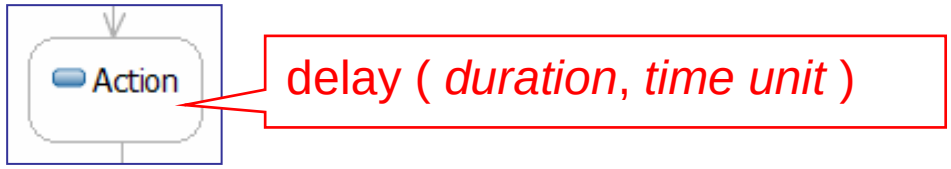
Network link
- Queuing discipline
- Delay
- Throughput

deployment → Action → emit (*quantity*)

Semantics comes from Queuing Networks

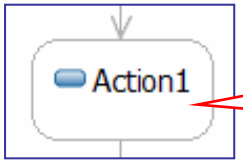
Behavior: Time consumption

- Usage
  - Functional delays
  - Abstraction of external processing
  - Abstraction of internal processing



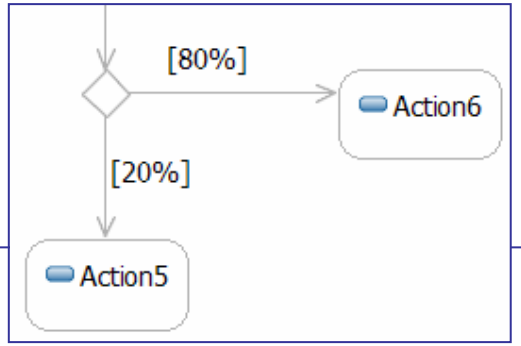
Behaviour: Data

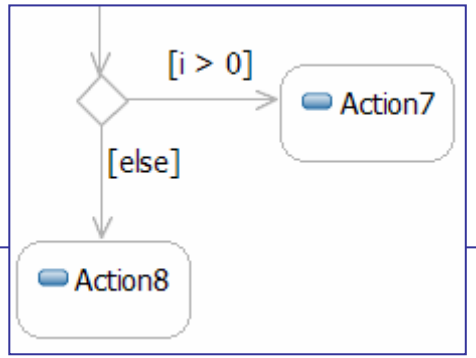
- Variables
  - Scope: sub-behavior
  - Kind: parameter / **shared** / **private** to tokens
  - Type: Boolean, Integer, Real, String
- Computations
  - C / Workbench strings



i = i + 3 + *iuniform*(1, 10);

- Probabilistic / deterministic choices





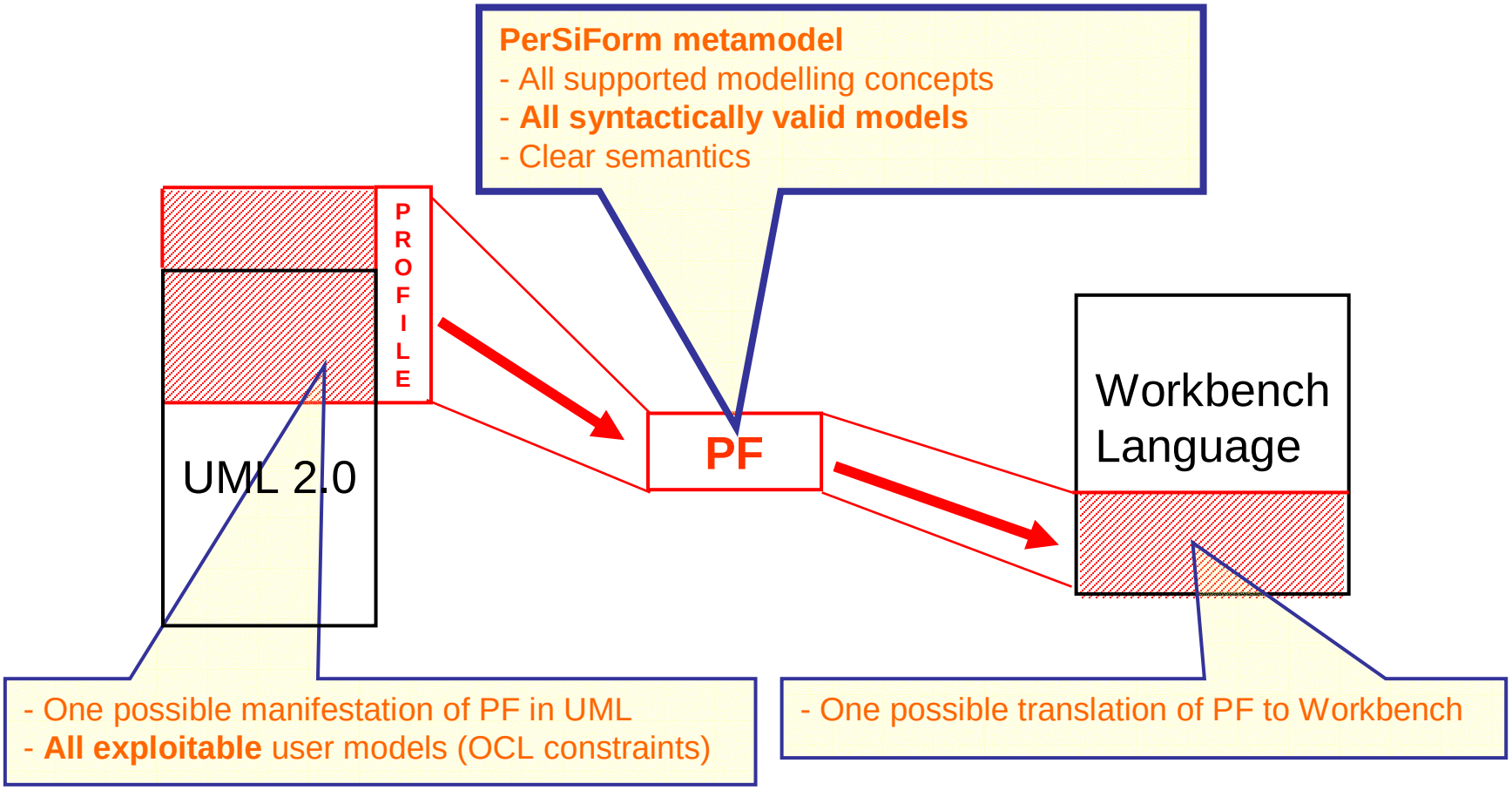
Environment

- Triggers the execution of the system's behaviours
- Reflects the expected behaviour of the clients of the system

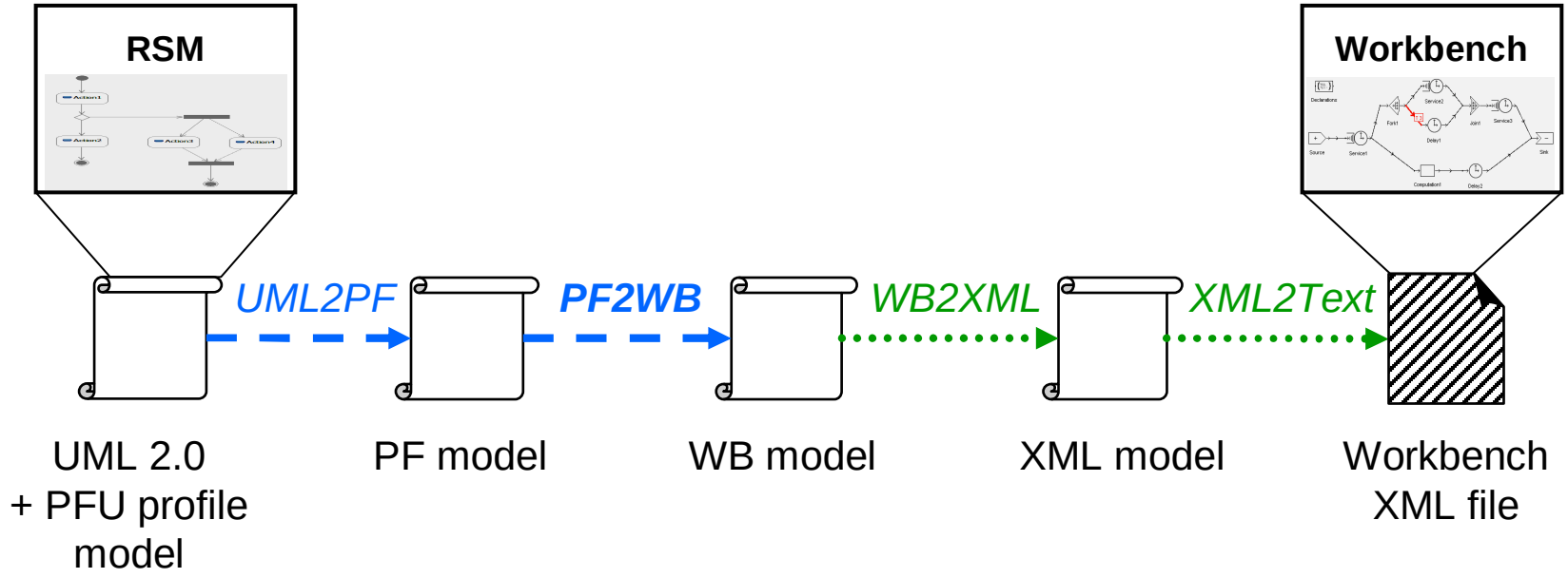
Environment
- Executed behaviour
- Inter arrival delay
- Time unit
- Nb executions
- Values passed as parameters

Implementing the approach

Transformation process: Conceptual view



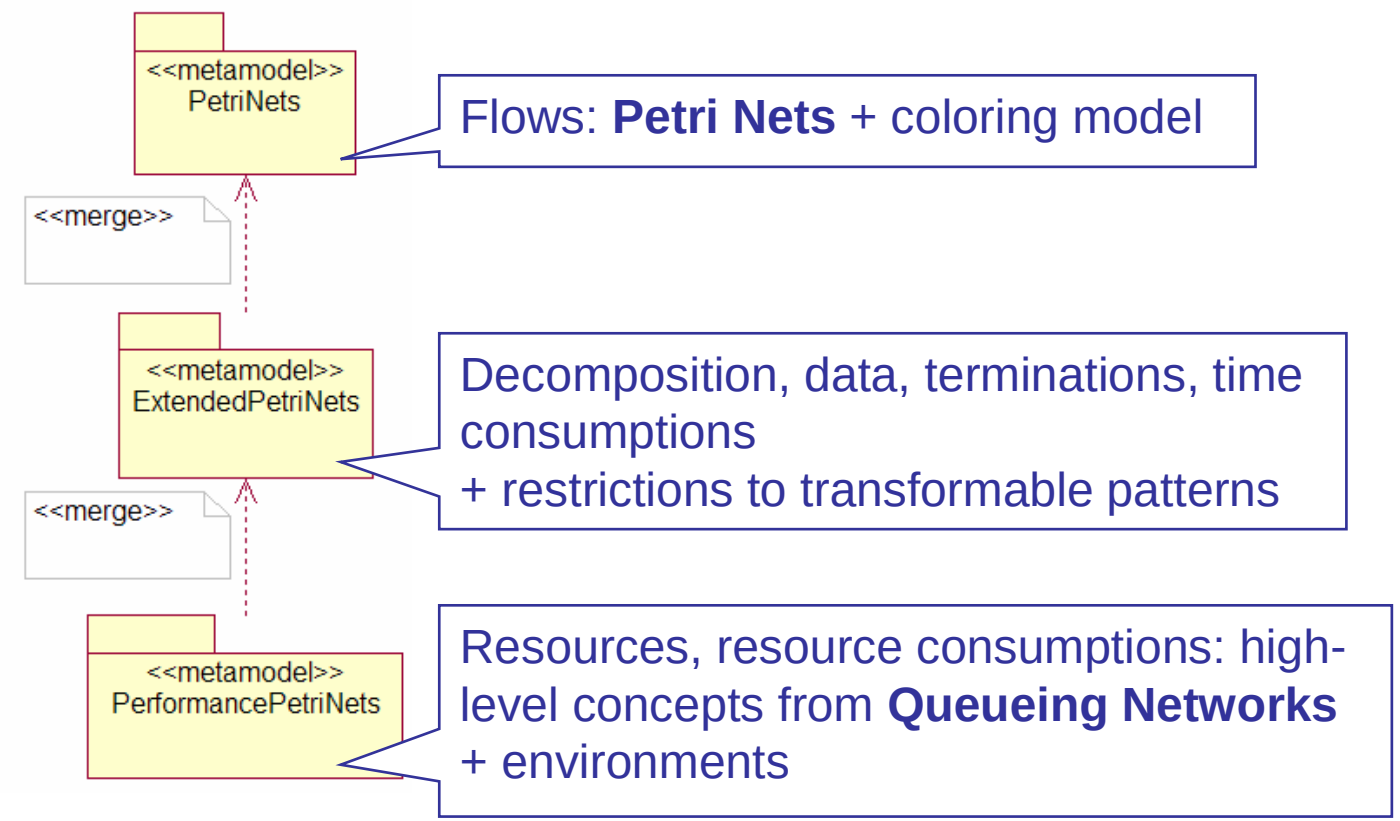
Transformation process: Technical view



Profile & Metamodels

- Profile
  - 40 stereotypes (12 abstract) on 20 UML metaclasses
  - 34 OCL constraints
- PF
  - 64 metaclasses (28 abstract)
  - 31 OCL constraints
- Our Workbench metamodel
  - 47 metaclasses (8 abstract)

Overview of PF



Transformations implementation

- Transformation process implemented in ATL (Inria Nantes)
  - Tried to stick to a declarative style (rule-based)
  - Allowed informal proofs

Query

```
helper context UML!ActivityEdge def:
mustProducePlace() : Boolean =
    not self.producesTransition()
        and
    self.source.transformsToTransition()
        and
    self.target.transformsToTransition();
```

Rule based on query

```
rule Edge2SimplePlace {
    from e : UML!ActivityEdge
        (e.mustProducePlace())
    to
        p : PF!SimplePlace (
            name <- e.name.newPlaceName(),
            net <- e.getPNProducer(),
            isColoring <-
                if (e.hasStereotype('PFArc')) then
                    e.stereotypeValue('PFArc', 'isColoring')
                else
                    false
                endif
        )
}
```

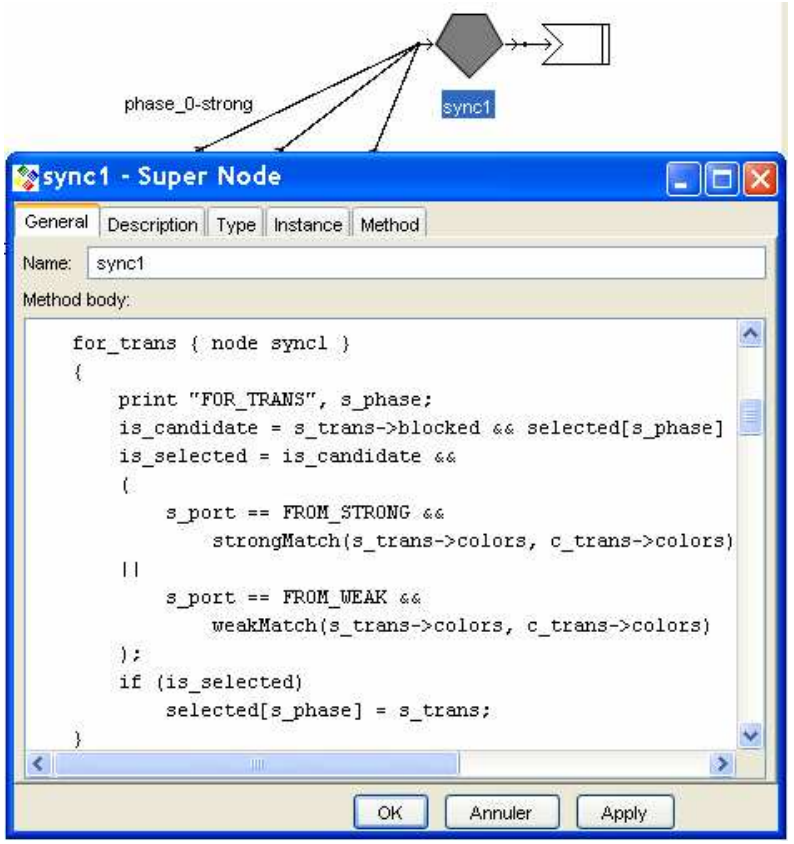
Transformations implementation (2)

- Overview of transformation code
  - Fully declarative except for ‘‘called rules’’

	UML2PF	*PF2WB*	*WB2XML*	*XML2Text*
Query helpers	46	55	14	7
Matched rules	24	36	28	0
Lazy rules	13	1	30	0
Called rules	7	1	1	0
LOCs	1100	1700	1600	50

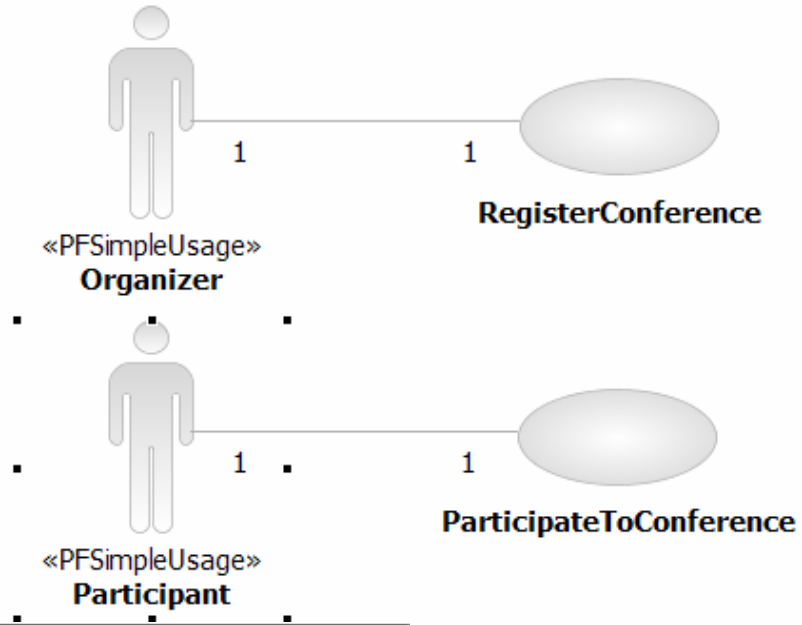
PF to Workbench transformation

- Principles
  - Standard concepts to high-level Workbench concepts
  - Complex concepts to Workbench code
    - Arbitrary synchronization, terminations, etc.
  - Try to obtain a clear graph representation
    - Apply a last transformation that interfaces with DOT to generate layout information for Workbench



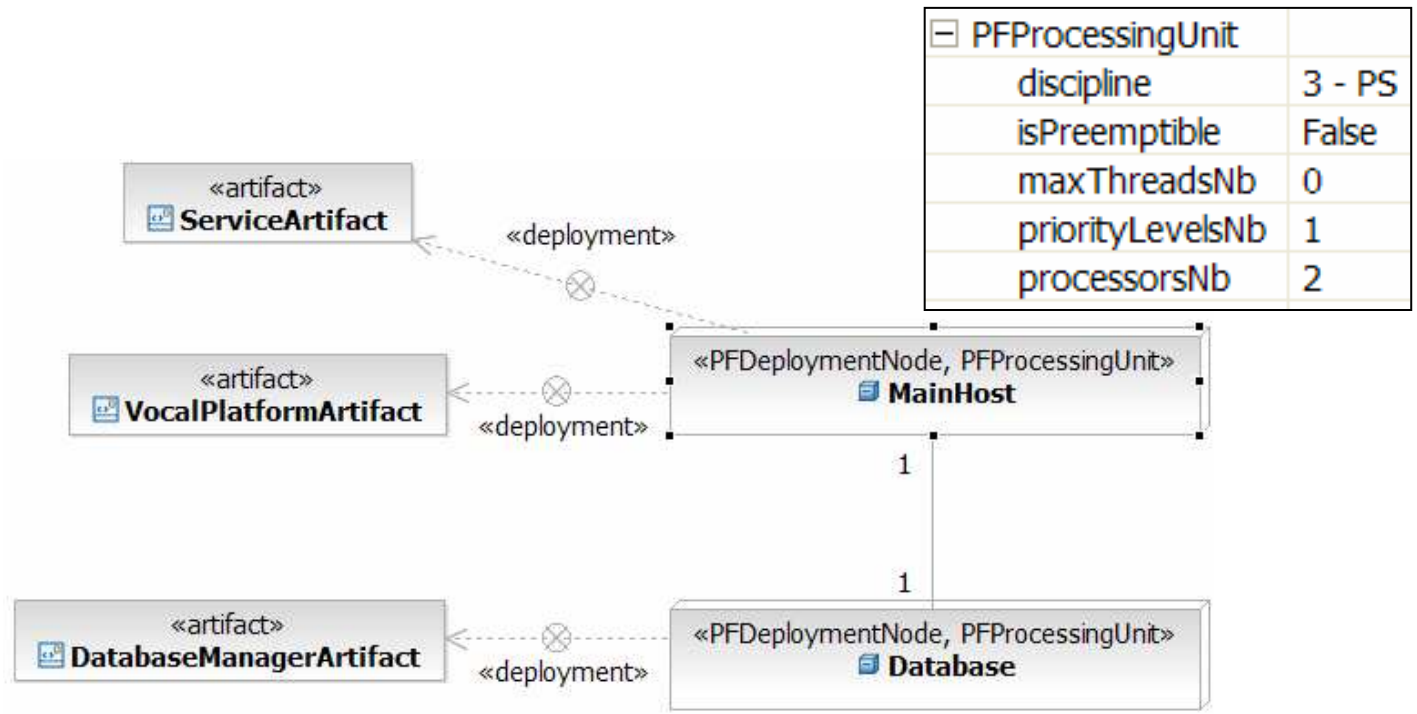
Example: Audio Conference System

Environment



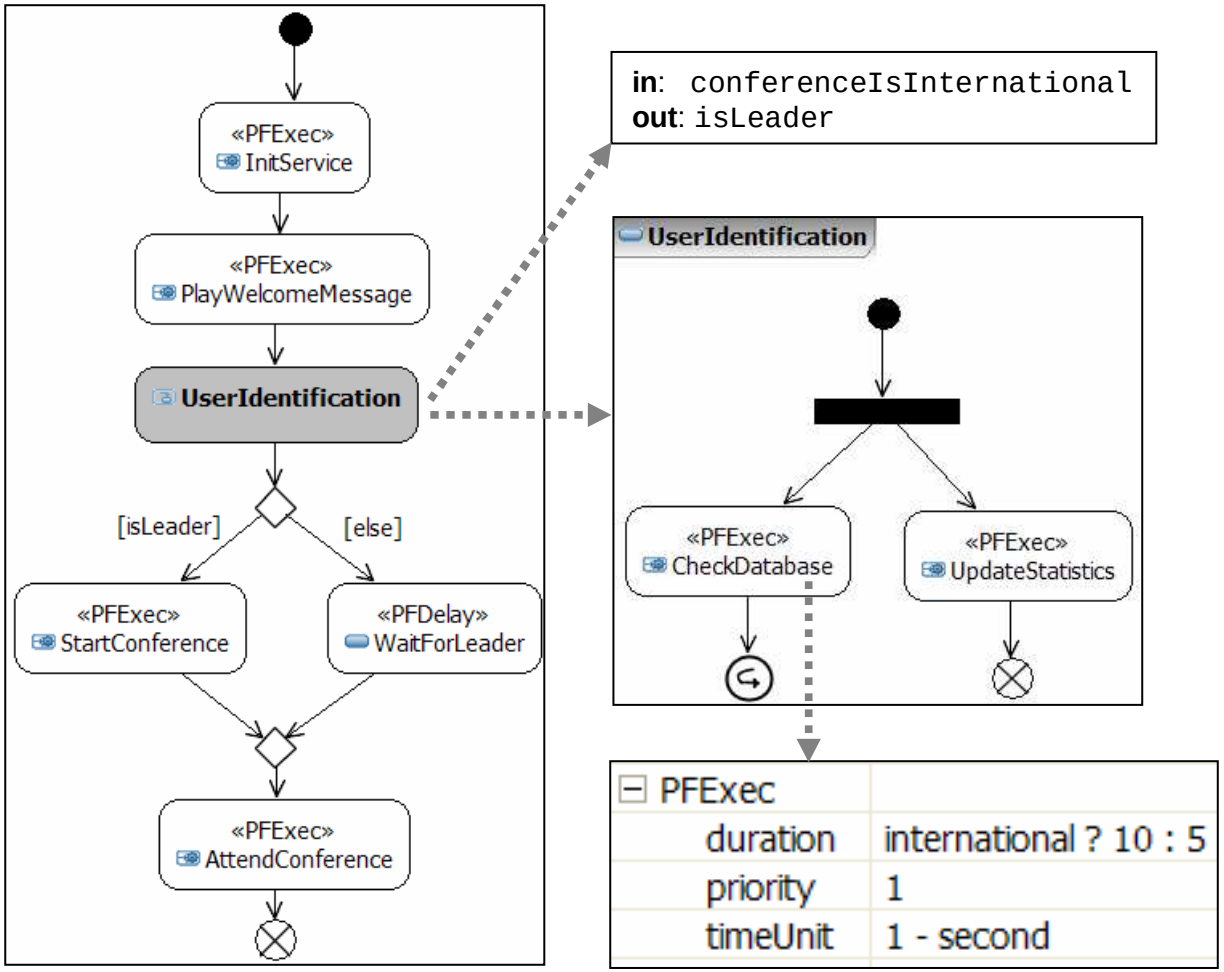
PFSimpleUsage	
created	1
interarrivalDelay	uniform(0, 5)
passedValue	Entries: 1
timeUnit	1 - second

Deployment

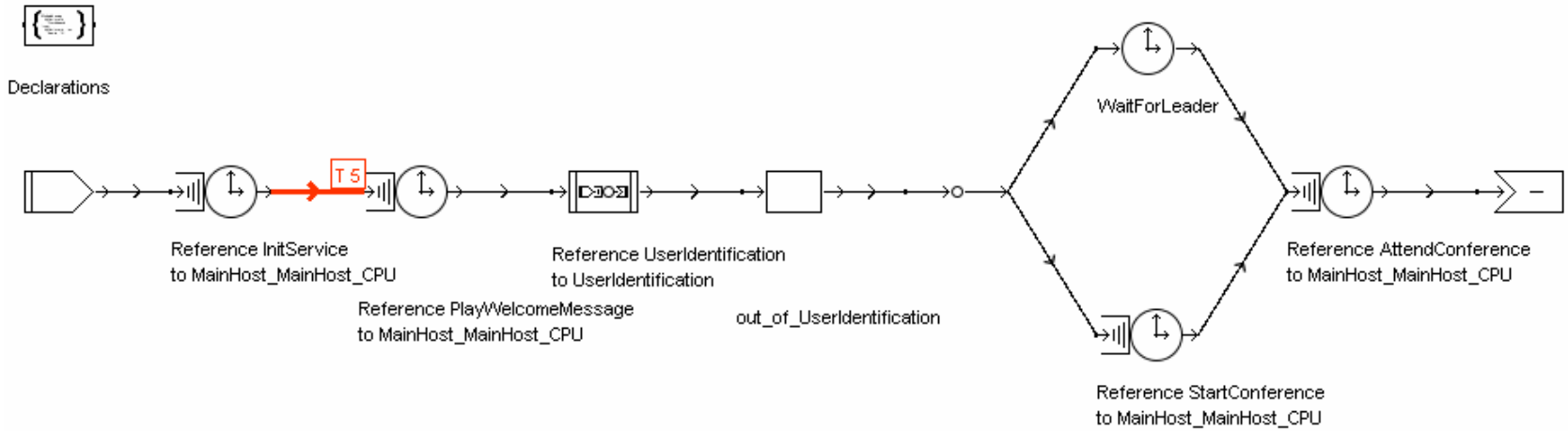

PFProcessingUnit
discipline 3 - PS
isPreemptible False
maxThreadsNb 0
priorityLevelsNb 1
processorsNb 2
«artifact»
ServiceArtifact
«deployment»
«artifact»
VocalPlatformArtifact
«deployment»
«PFDeploymentNode, PFProcessingUnit»
MainHost
1
1
«artifact»
DatabaseManagerArtifact
«deployment»
«PFDeploymentNode, PFProcessingUnit»
Database

Behavior

ParticipateToConference



Resulting Workbench model (extract)



Case studies

- Several small examples
- Orpheus Radar map system (under work)
  - Service providing actual radar pictures to ships
  - Purpose: calibrate memory, computation and network capacities
- France Télécom IOS-W (demonstrated application)
  - Plat-form-to-platform communication bus using web services
- France Télécom ADSL
  - ADSL registration service
  - 17 activity diagrams

Conclusion & perspectives

- Shown the feasibility of the approach
  - **Systematic** transformations
  - High expressivity & complex models
  - Involvement of **industrial tools** used by the specialists

- Forthcoming …
  - Progress on investigating support for functional validation, in particular absence of feature interaction (Promela, IF, BIP),
  - More case studies
  - code generation from detailed models ? (BIP)

2

Application of the Persiform Tool Chain and Industrial Perspective

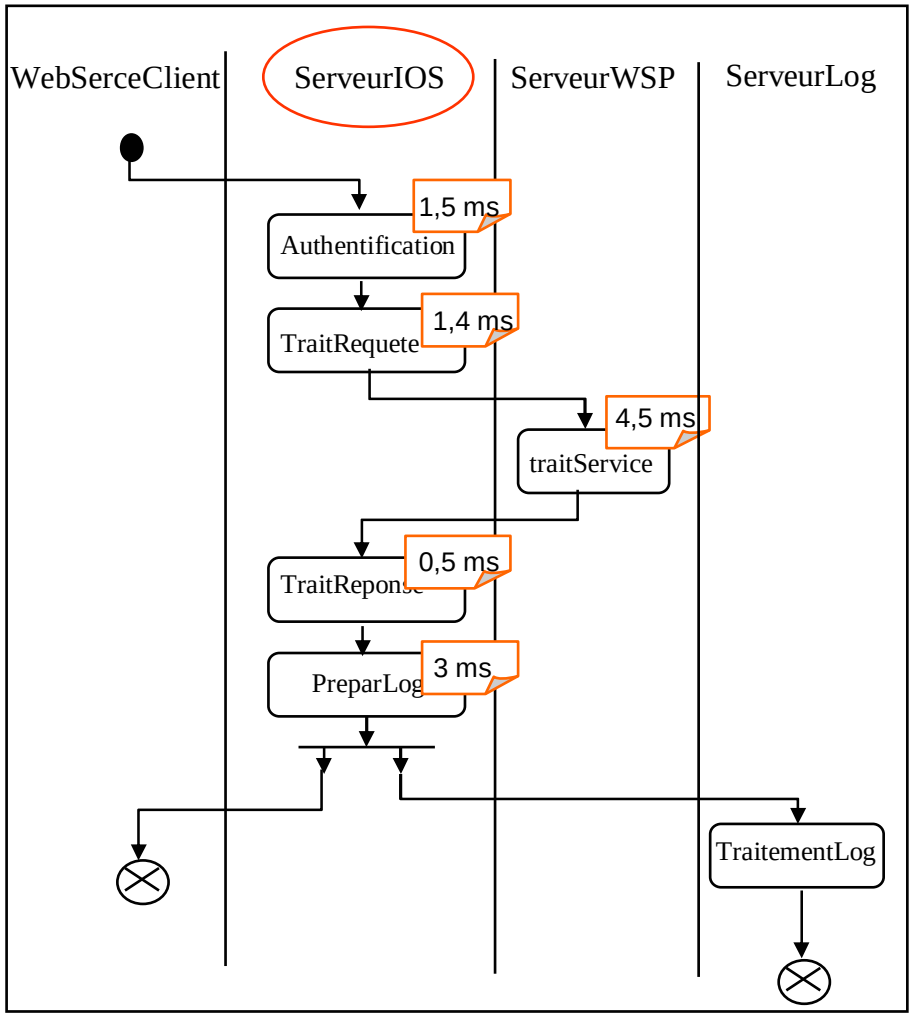
- Case study description and demo
- Simulation's results vs. measures realized on the application
- Perspectives for France Telecom

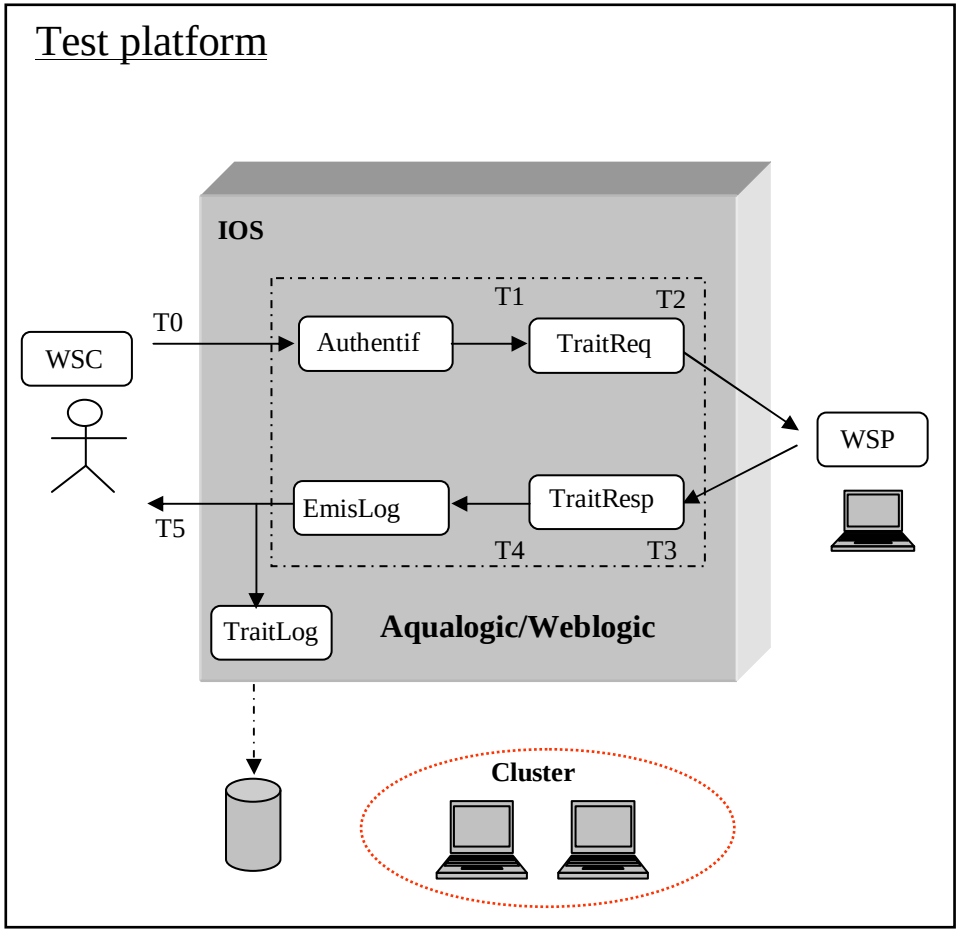
case study: IOS-W

- why this case study and not the ADSL delivery?
  - IOS-W fits well
  - ADSL delivery case study is a too big case study for a short talk
    - 17 activity diagrams
  - IOS-W system is developed and real measurements are available
    - Comparisons of simulation results with these measurements

- what is IOS-W?
  - a platform-to-platform communication bus using web services
  - used by France Telecom's Information System
  - developed on BEA Aqualogic (SOA platform)

informal description





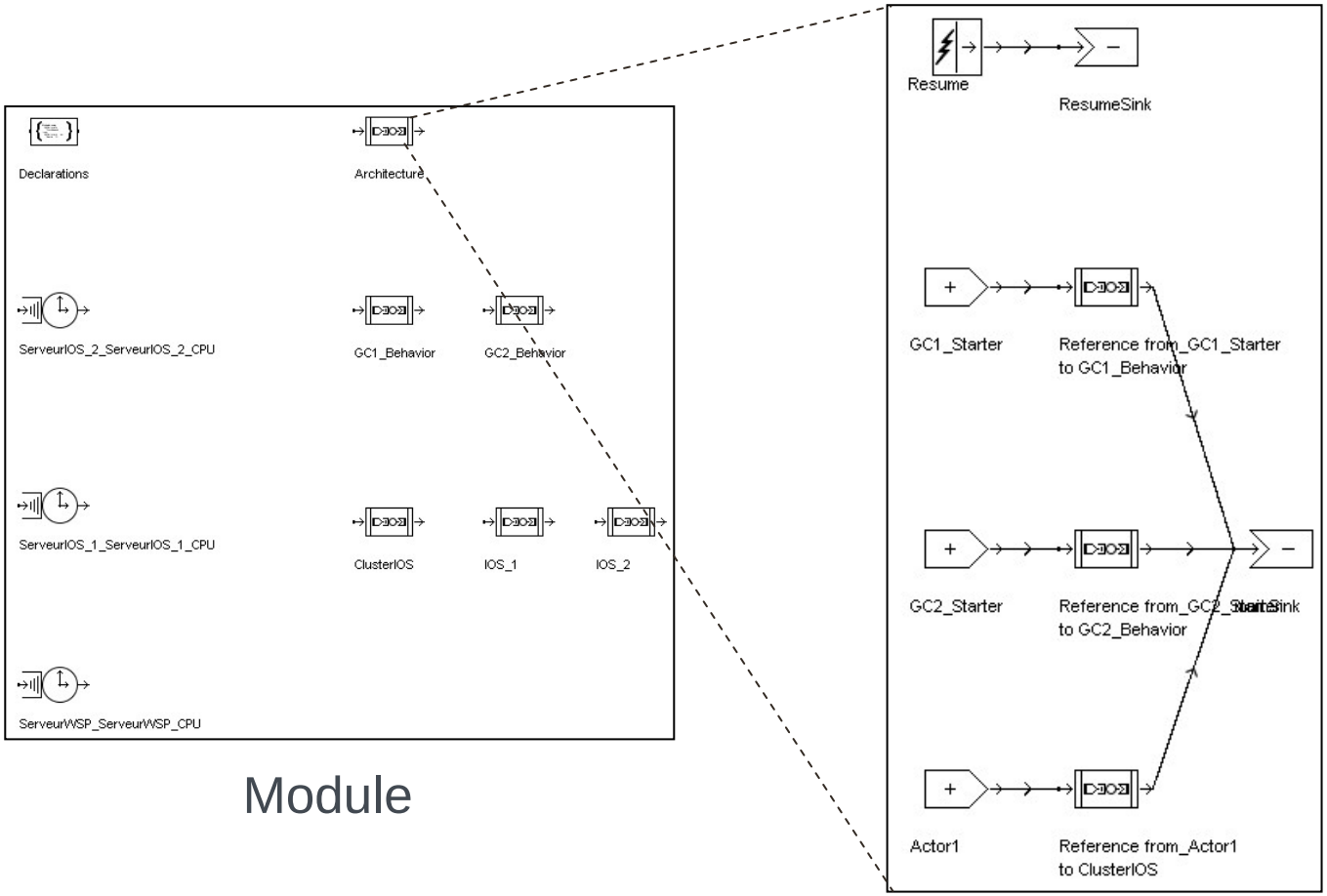
IOS-W ’s PerSiForm description

- 7 diagrams for IOS-W:
  - 1 use case diagram
    - describes client's behavior
  - 5 Activity diagrams
    - describe the dynamic of the system, the cluster and garbage collectors
  - 1 deployment diagram
    - describe how the system is deployed on the hardware

Demo

- A look at some of the diagrams with RSM
- running model transformations
  - UML to Petri net
  - Petri net to Workbench
  - Workbench to XML
  - XML to text (to produce a file for Hyperformix/Workbench)

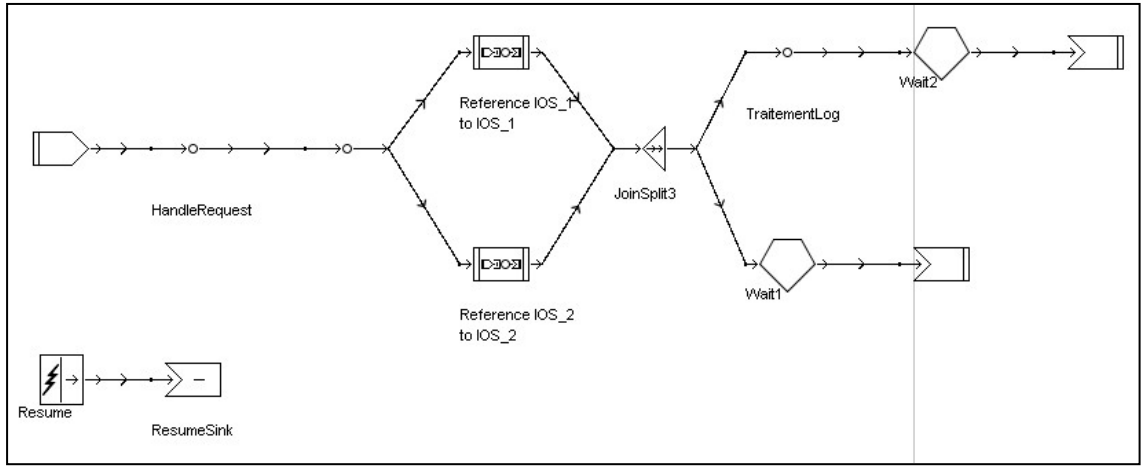
Generated simulation model



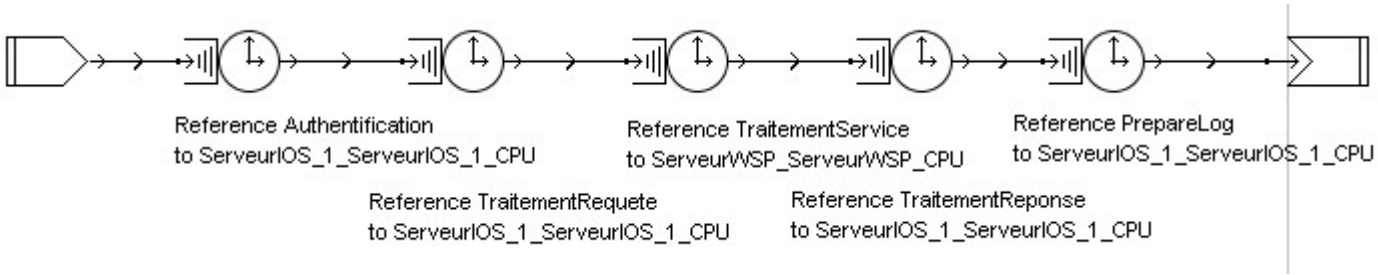
Module

Architecture

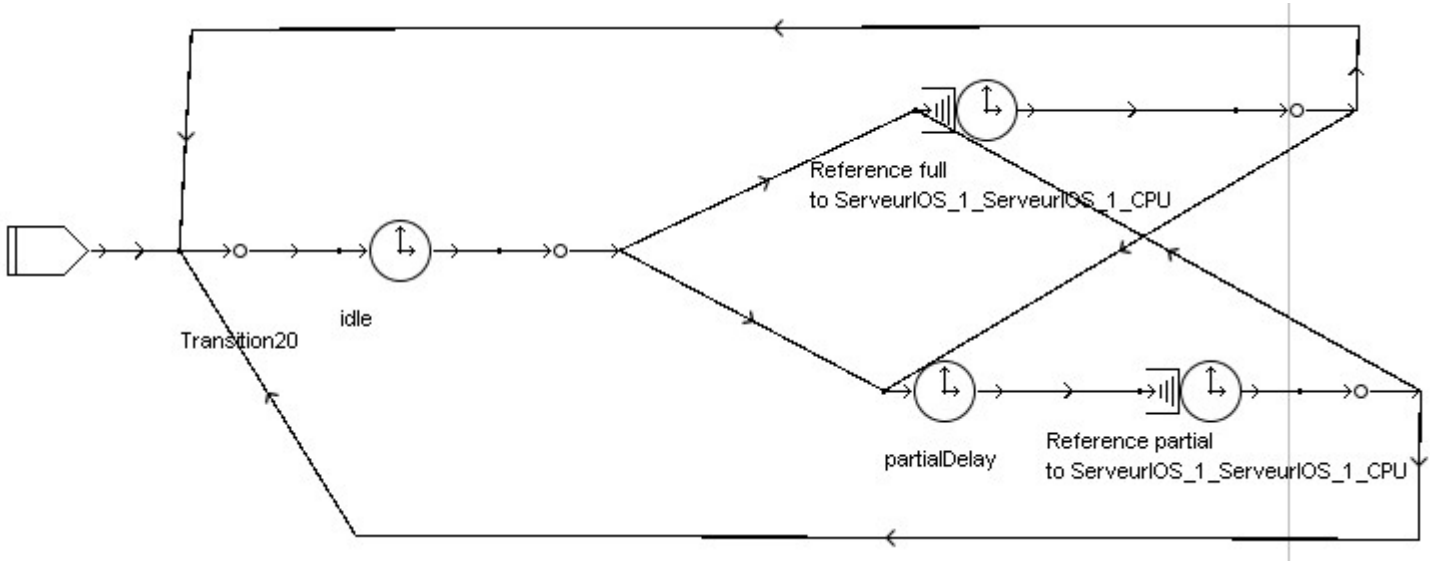
Cluster's behavior:



GC's behavior:



IOS's behavior:

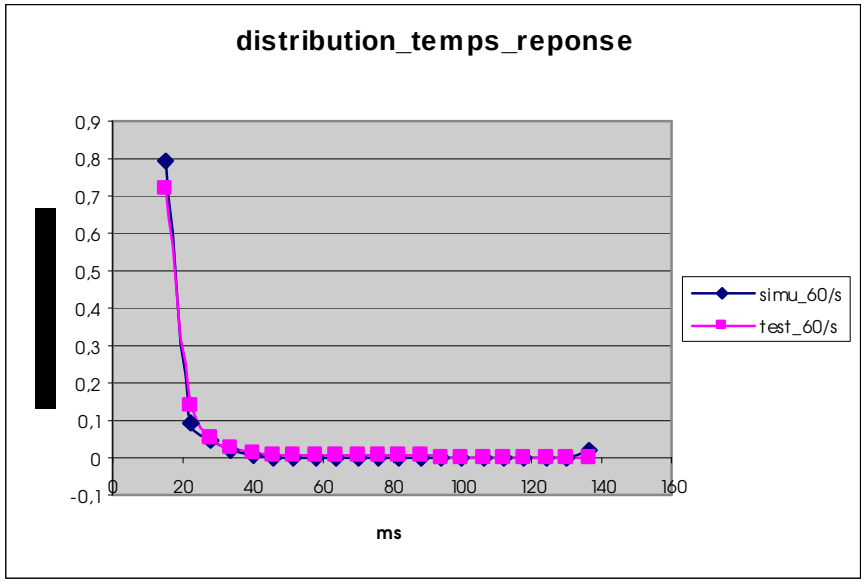


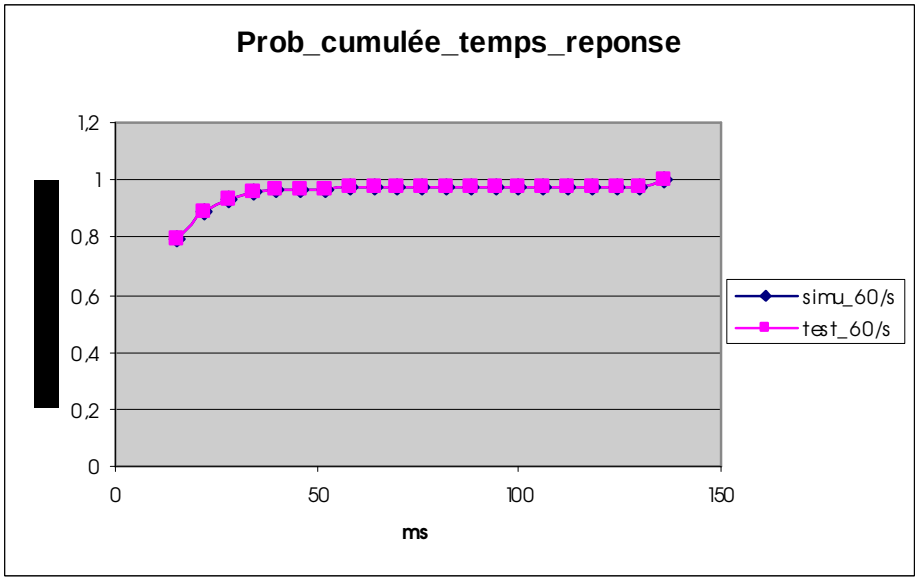
Simulation results vs. measurements

- original study in 2006 by
  - Wei Monin, Cyrille Puget and Guy Vachet (France Telecom R&D)
  - open source load injection framework: CLIF (ObjectWeb)
- Some comments

Simulation results vs. measurements (1/4)

- end-to-end response time under a load of 60 requests/sec

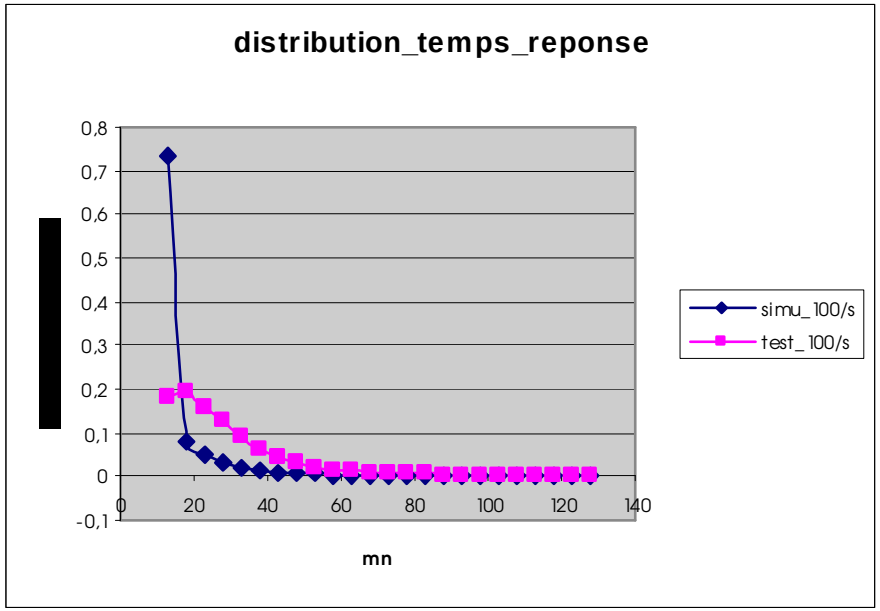


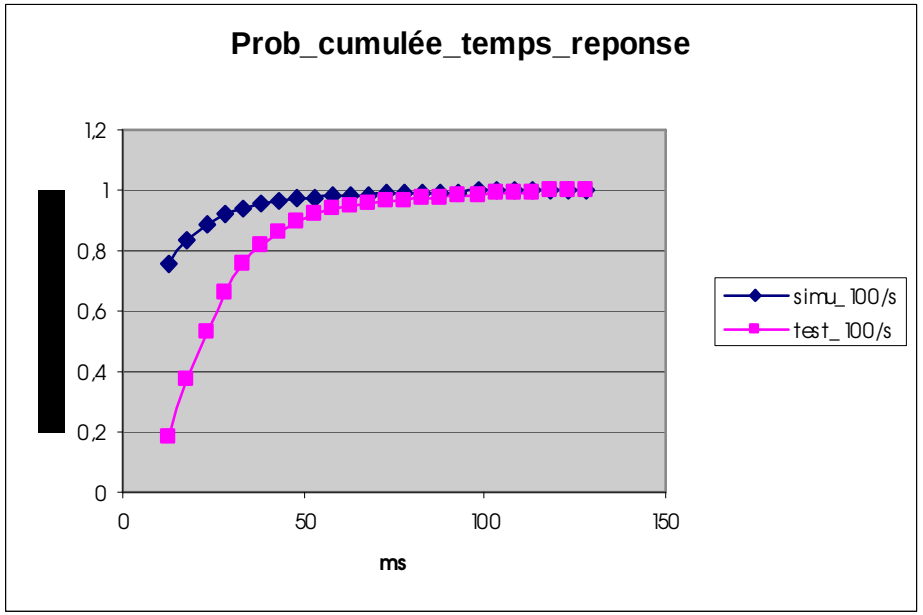


- simulation is very close to what is observed by test

Simulation results vs. measurements (2/4)

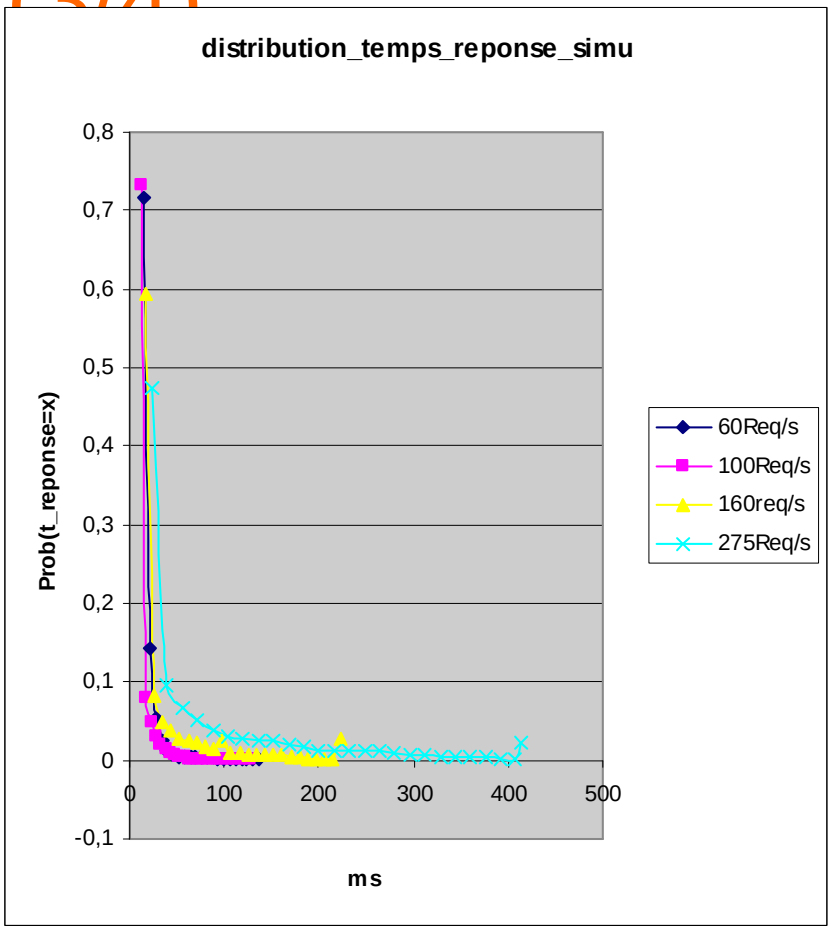
- end-to-end response time under a load of 100 requests/sec

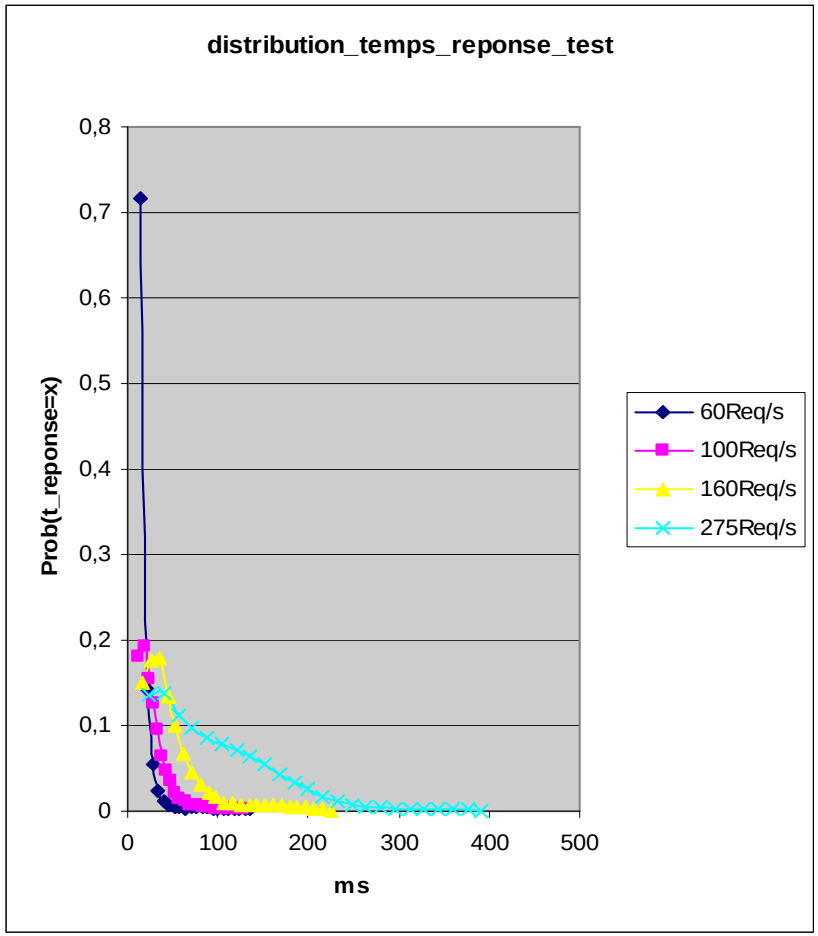




- a distance between simulation and test appears

Simulation results vs. measurements (3/4)





- and the gap grows with the load…

Simulation results vs. measurements (4/4)

- under somewhat heavy loads, a request's end-to-end response time is longer in test than in simulation…
- a possible reason:

```
"ExecuteThread: '0' for queue: 'weblogic.socket.Muxer'" id=18 idx=0x28 tid=25208
    -- Blocked trying to get lock: java/lang/String@0xc2842c8[fat lock]
    at jrockit/vm/Threads.waitForSignal()V(Native Method)
    at jrockit/vm/Locks.fatLockBlockOrSpin(ILjrockit/vm/ObjectMonitor;II)V(Unknov
    at jrockit/vm/Locks.lockFat(Ljava/lang/Object;ILjrockit/vm/ObjectMonitor;Z)Lj
    at jrockit/vm/Locks.monitorEnterSecondStage(Ljava/lang/Object;I)Ljava/lang/Ob
    at jrockit/vm/Locks.monitorEnter(Ljava/lang/Object;)Ljava/lang/Object;(Unknov
    at weblogic/socket/EPollSocketMuxer.processSockets()V(EPollSocketMuxer.java:
    at weblogic/socket/SocketReaderRequest.run()V(SocketReaderRequest.java:29)
    at weblogic/socket/SocketReaderRequest.execute(Lweblogic/kernel/ExecuteThrea
    at weblogic/kernel/ExecuteThread.execute(Lweblogic/kernel/ExecuteRequest;)V(I
    at weblogic/kernel/ExecuteThread.run()V(ExecuteThread.java:117)
    at jrockit/vm/RNI.c2java(IIII)V(Native Method)
    -- end of trace

"ExecuteThread: '1' for queue: 'weblogic.socket.Muxer'" id=19 idx=0x2a tid=25209
    -- Blocked trying to get lock: java/lang/String@0xc2842c8[fat lock]
    at jrockit/vm/Threads.waitForSignal()V(Native Method)
    at jrockit/vm/Locks.fatLockBlockOrSpin(ILjrockit/vm/ObjectMonitor;II)V(Unknov
    at jrockit/vm/Locks.lockFat(Ljava/lang/Object;ILjrockit/vm/ObjectMonitor;Z)Lj
    at jrockit/vm/Locks.monitorEnterSecondStage(Ljava/lang/Object;I)Ljava/lang/Ob
    at jrockit/vm/Locks.monitorEnter(Ljava/lang/Object;)Ljava/lang/Object;(Unknov
    at weblogic/socket/EPollSocketMuxer.processSockets()V(EPollSocketMuxer.java:
    at weblogic/socket/SocketReaderRequest.run()V(SocketReaderRequest.java:29)
    at weblogic/socket/SocketReaderRequest.execute(Lweblogic/kernel/ExecuteThrea
    at weblogic/kernel/ExecuteThread.execute(Lweblogic/kernel/ExecuteRequest;)V(I
    at weblogic/kernel/ExecuteThread.run()V(ExecuteThread.java:117)
    at jrockit/vm/RNI.c2java(IIII)V(Native Method)
    -- end of trace
```

- need to describe middleware's behavior for finer simulation results (thread pool, connection pool, component pool,…)

Conclusion and perspectives

- relevant simulation results when the behavior is well known
  - kind of results: response time, number of concurrent accesses, risks of bottleneck, impact of parameter values, configuration's comparison,…
  - sizing resources for large scale deployment
- possible transfer to anticipation and/or development projects
  - main targeted domains: Information System, service platforms,…
- use theses techniques on more applications within pilot projects
  - user feed back

Conclusion and perspectives

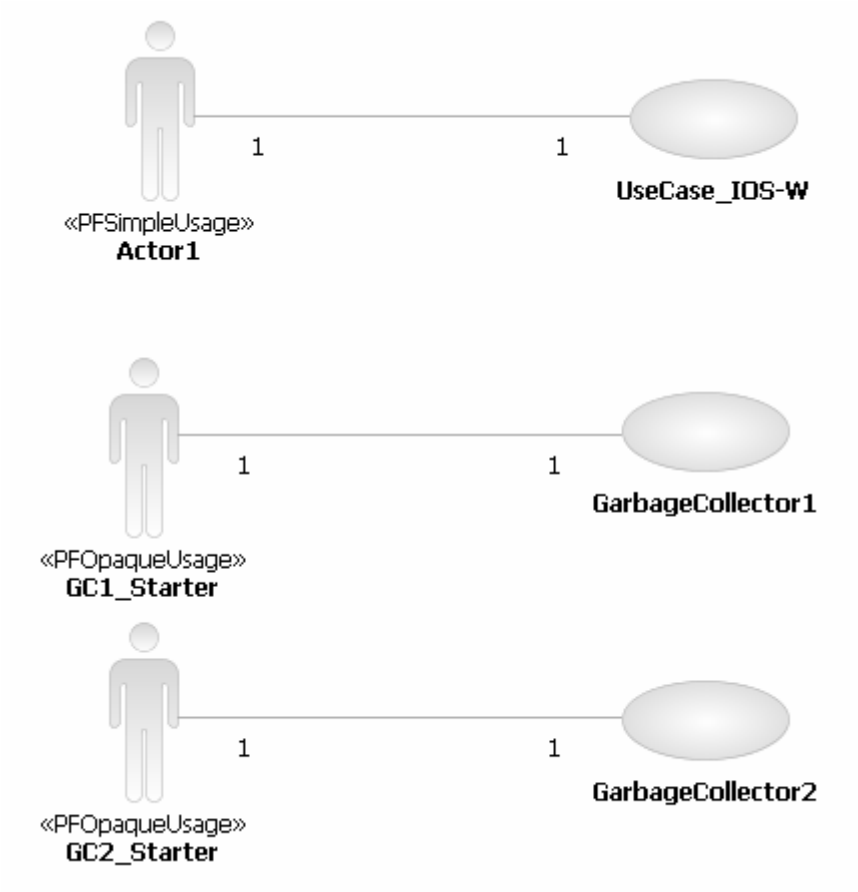
- Information System edits recommendations to handle performance requirements earlier in the life cycle.
  - 4 main complementary approaches
    - intuition: based on experience
    - measures: expensive, difficult interpretation, system needs to exist,…
    - model for performance: flexible (modifications are easy), reliable, but no specific tool recommended yet. Not widespread
    - peer reviews
- PerSiForm is legitimate to contribute to the 'Model for performance' approach
  - brings added value with it's formalism (strong semantic + specific OCL constraints) and it's tool chain
  - make performance study more accessible (description of the model based on UML)

Questions & discussion

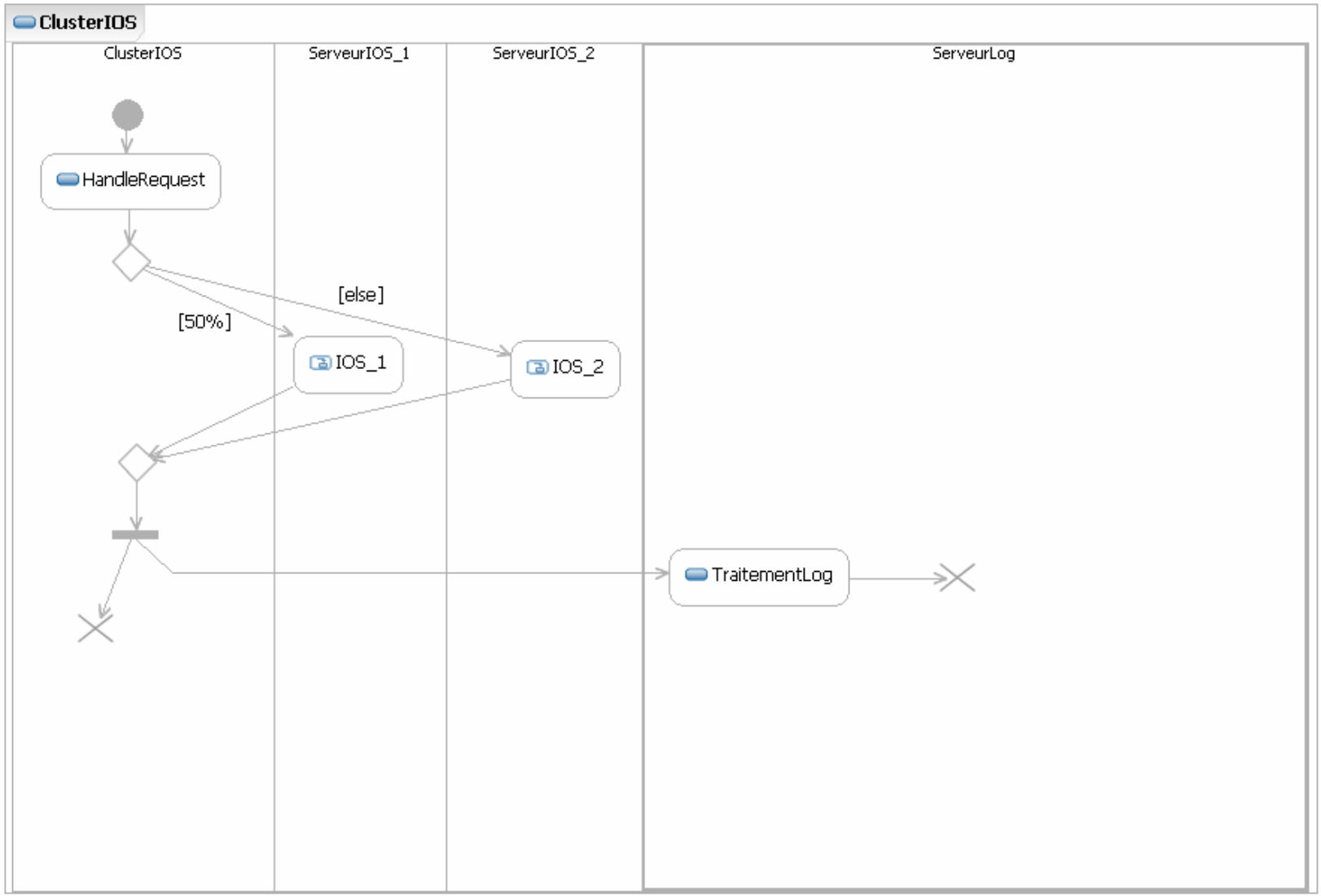
Annexes

- UML diagrams screenshots

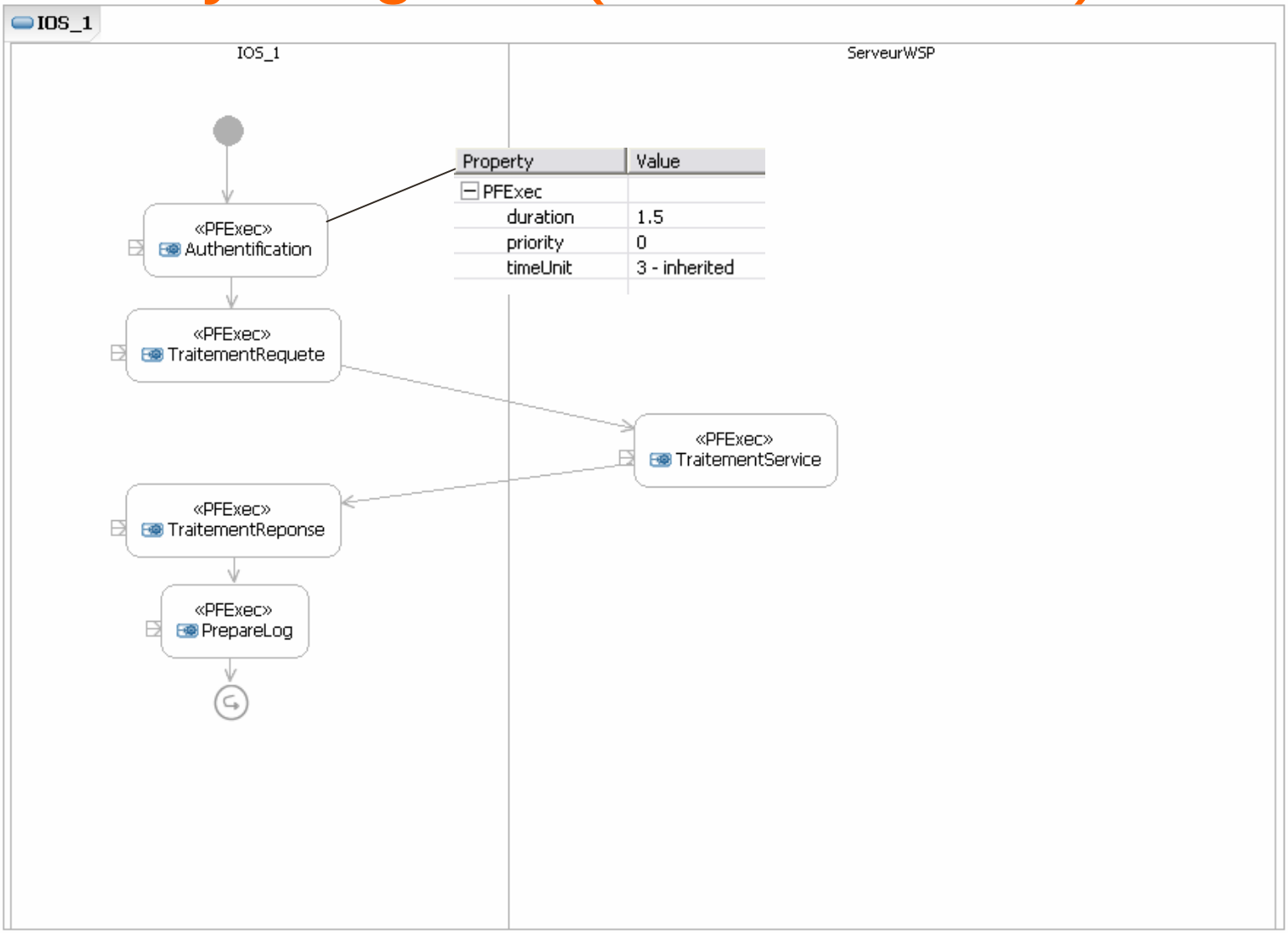
Use Case diagram



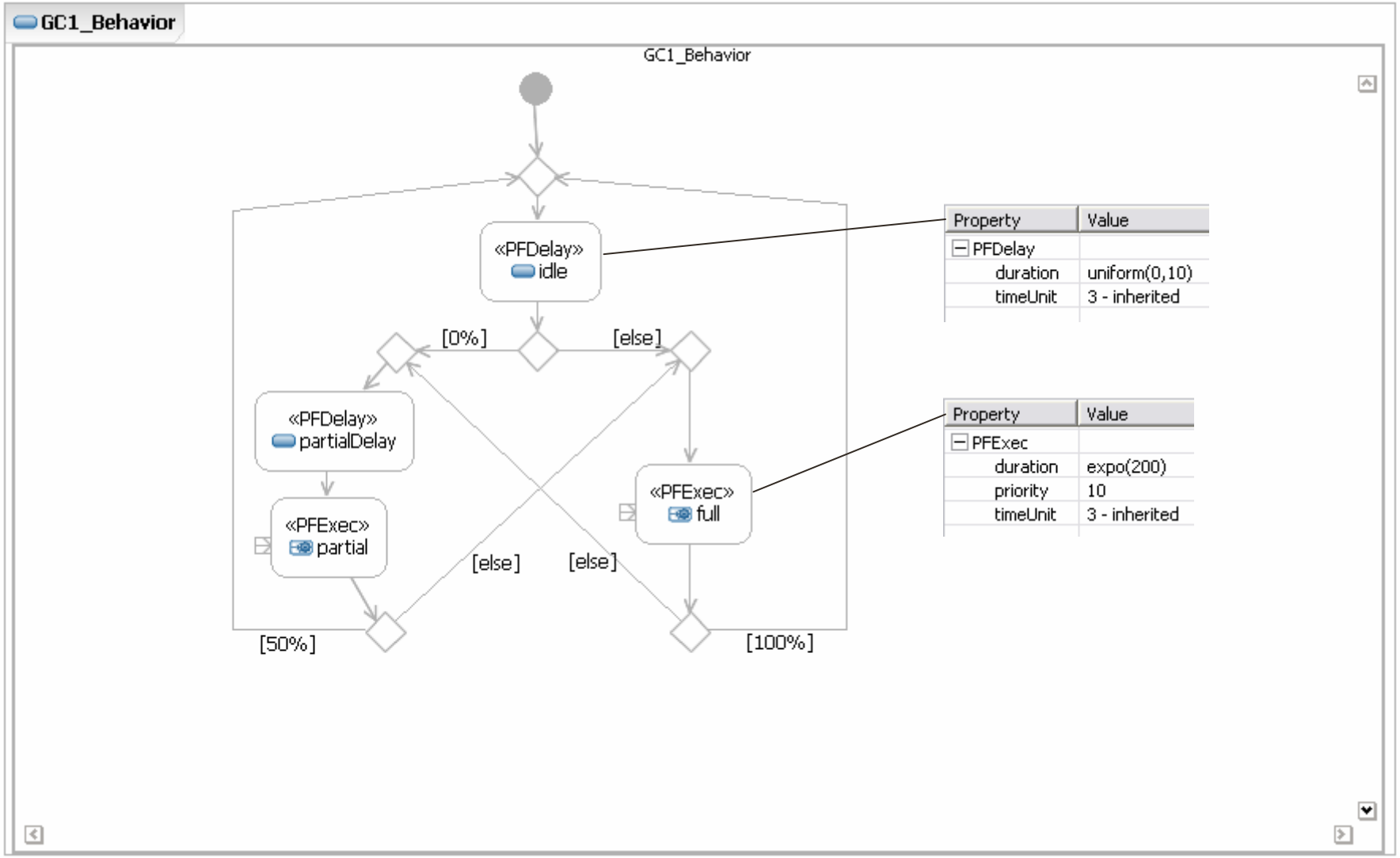
Activity diagram (cluster behavior)



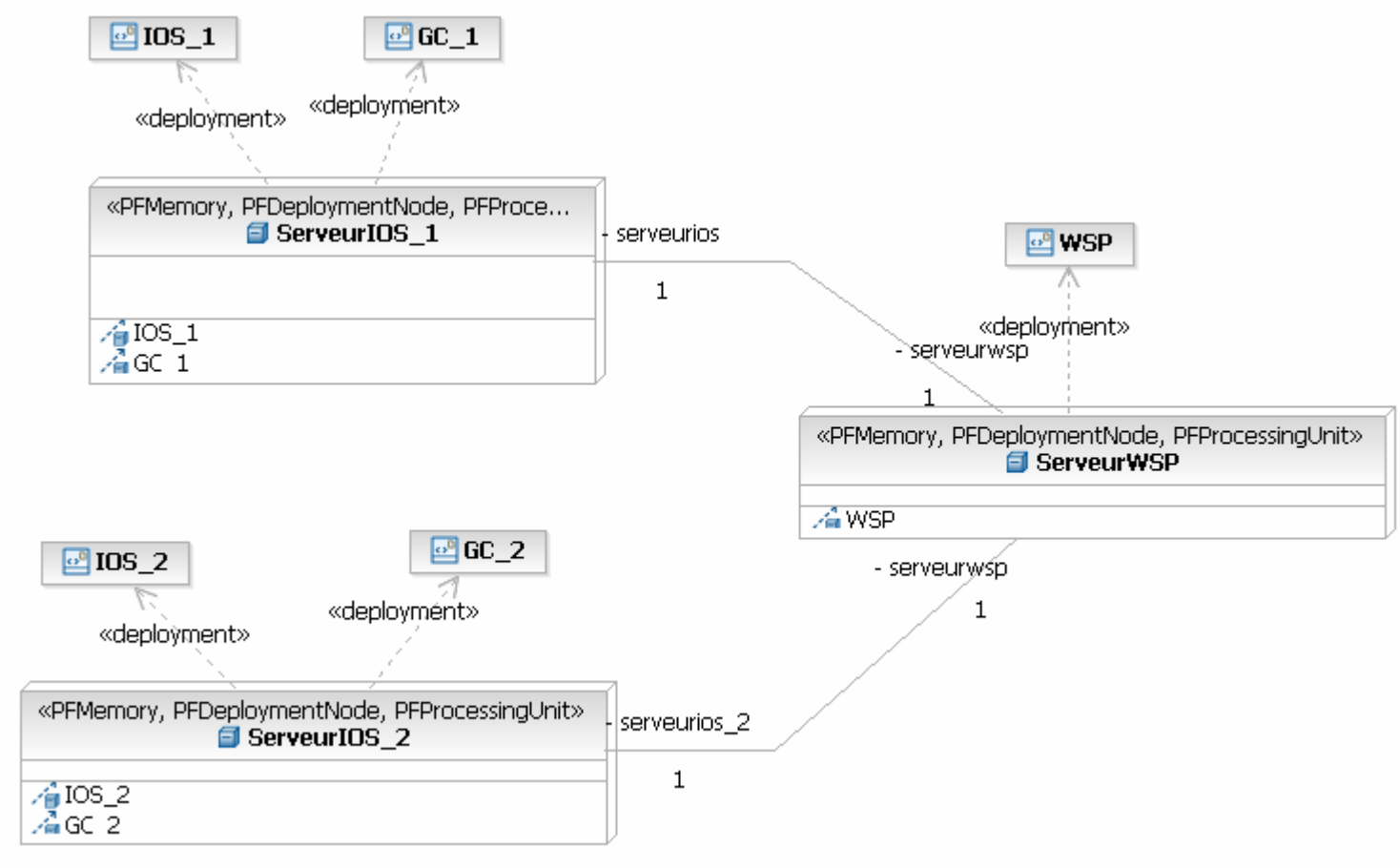
Activity diagram (IOS behavior)



Activity diagram (GC behavior)



Deployment diagram



Model used for the simulation