Simulation Modeling of UML Software Architectures

Moreno Marzolla

Dipartimento di Informatica
Università “Ca' Foscari” di Venezia
marzolla@dsi.unive.it

Talk Outline

- Motivations and General Principles
- Contribution of this work
- Introduction to UML
- The Software Performance Model
- The Simulation Model
- Conclusions

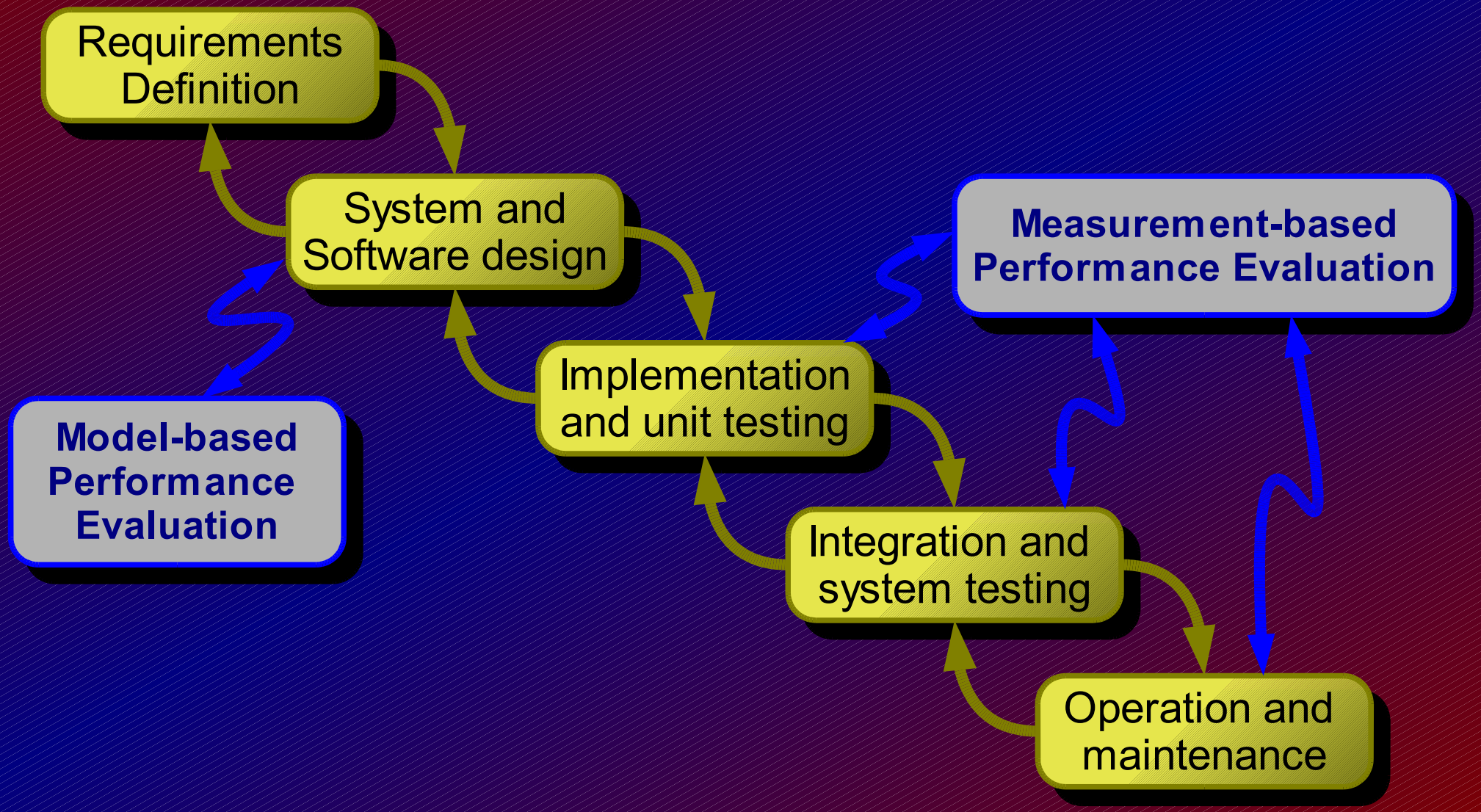
Motivations

- Large software systems are complex and expensive artifacts
- Their success depends on whether they deliver a satisfactory performance level
- Two questions:
  - How to evaluate performances during the software development process?
  - When are performances to be evaluated?

How performances can be evaluated?

- ***Measurement-based approach***
  - Perform direct measures on a running system or a prototype; use these measurements to identify bottlenecks
- ***Model-based approach***
  - Develop a performance model of the software system; use the model to mimic the behavior of the system and predict its performances

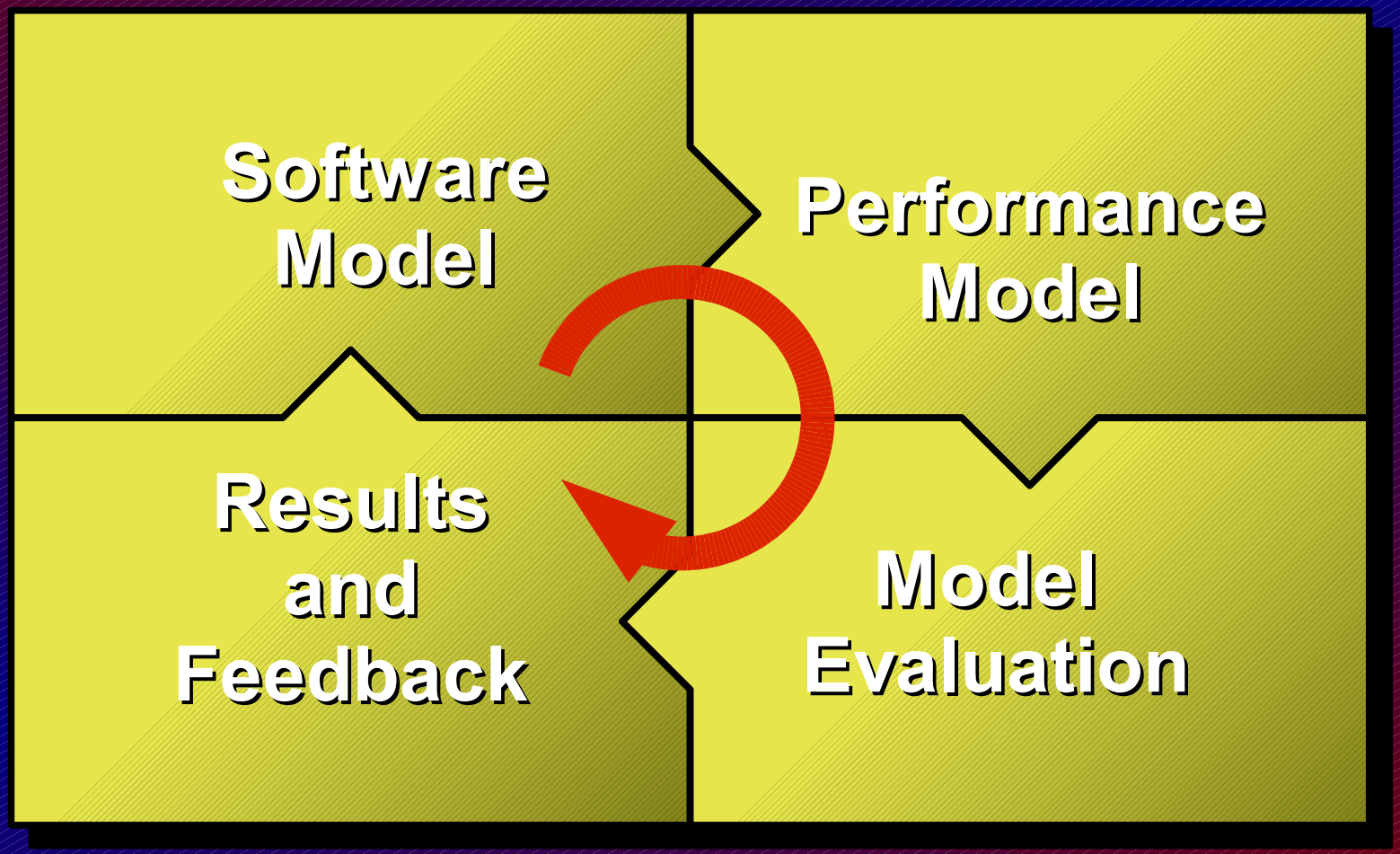
Where performances can be evaluated?



Model-Based Performance Evaluation

- **Pro**
  - Does not require a running system
  - Hence, can be applied from the early design stages
- **Cons**:
  - Accuracy of the performance prediction depends on the accuracy of the software model (100% accuracy not needed/reasonable anyway)
  - May be difficult to report feedback due to different structure of the performance model wrt the software model

General Software Performance Modeling



Background / 1

- SPE approach by Smith
  - C. U. Smith, *Performance Engineering of Software Systems*, Addison-Wesley 1990
  - C. U. Smith and L. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software,* Addison-Wesley, 2002
- Many approaches deriving analytical models
  - S. Bernardi, S. Donatelli and J. Merseguer. *From UML sequence diagrams and statecharts to analysable Petri net models.* Proc. WOSP 2000
  - M. Bernardo, P. Ciancarini and L. Donatiello. *AEmpa: A process algebraic description language for the performance analysis of software architectures*. Proc. WOSP 2000
  - F. Aquilani, S. Balsamo and P. Inverardi. *Performance Analysis at the Software Architectural Design level*. Performance Evaluation, 2001, no.45, 147-178.
  - C. M. Woodside et al.. *The stochastic rendezvous network for performance of synchronous client-server-like distributed systems*. IEEE Trans. Comp, 44:20-34, 1995

Background / 2

- Some works deriving simulation models
  - L. B. Arief and M. Speirs. *A UML tool for an automatic generation of simulation programs*. Proc. WOSP 2000
  - P. Kähkipuro. *UML-based performance modeling framework for component-based distributed systems. Performance Engineering*, vol 2047 of LNCS.
  - M. De Miguel et al. *UML extensions for the specifications and evaluation of latency constraints in architectural models.* Proc. WOSP 2000
  - A. Hennig et al. *Performance prototyping - generating and simulating a distributed IT-system from UML models*. Proc. ESM'03

Our contribution / 1

- We choose UML as a software specification notation
  - De facto standard for SA description, widely used in the Software Engineering community
  - Well supported by CASE tools
  - Extension mechanisms can be used to extend the UML metamodel
  - *UML Profile for Schedulability, Performance and Time specification* released in 2002 as an OMG standard

Our contribution / 2

- We develope a UML profile for specifying performance-oriented annotations on UML models
  - Based on the UML Performance Profile, with some modifications
  - Different way to model workloads
  - We modified the performance model in some points

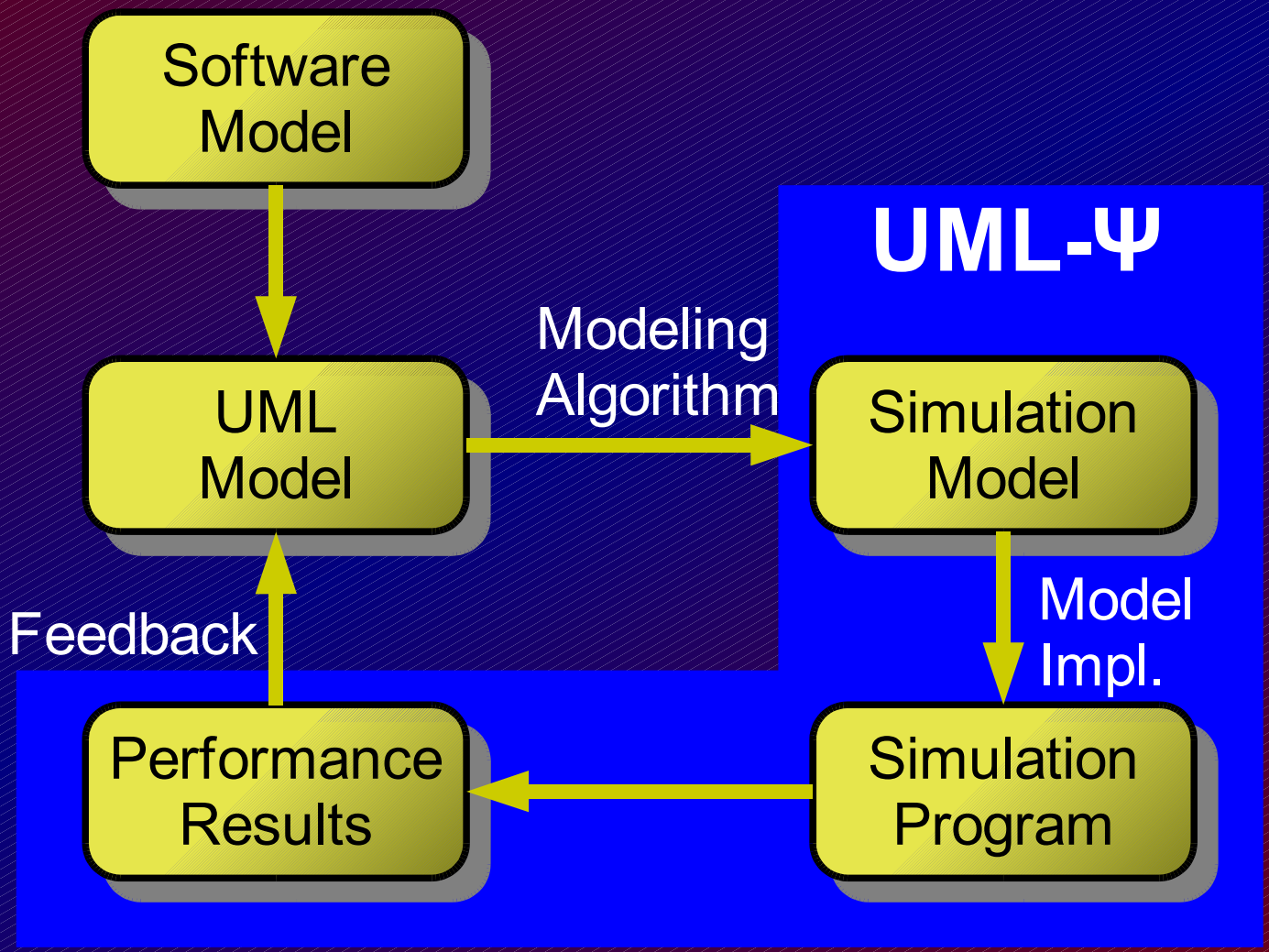
Our contribution / 3

- We define a process-oriented simulation model of UML SA
  - General modeling technique
  - Allows unconstrained model representation (fork/join, simultaneous resource possession, general scheduling policies, general time distributions...)
  - Model derivation is very easy...
  - ...and so reporting feedback

Our contribution / 4

- We develope a prototype tool (UML-Ψ)
  - Accepts XMI representations of annotated UML models produced by the ArgoUML CASE Tool
  - Automatically derives the simulation model
  - Creates a simulation program using a C++ process-oriented simulation library
  - Executes the simulation and returns the results into the UML model as tagged values
- We extend the approach to unified UML-based performance and mobility modeling

The approach

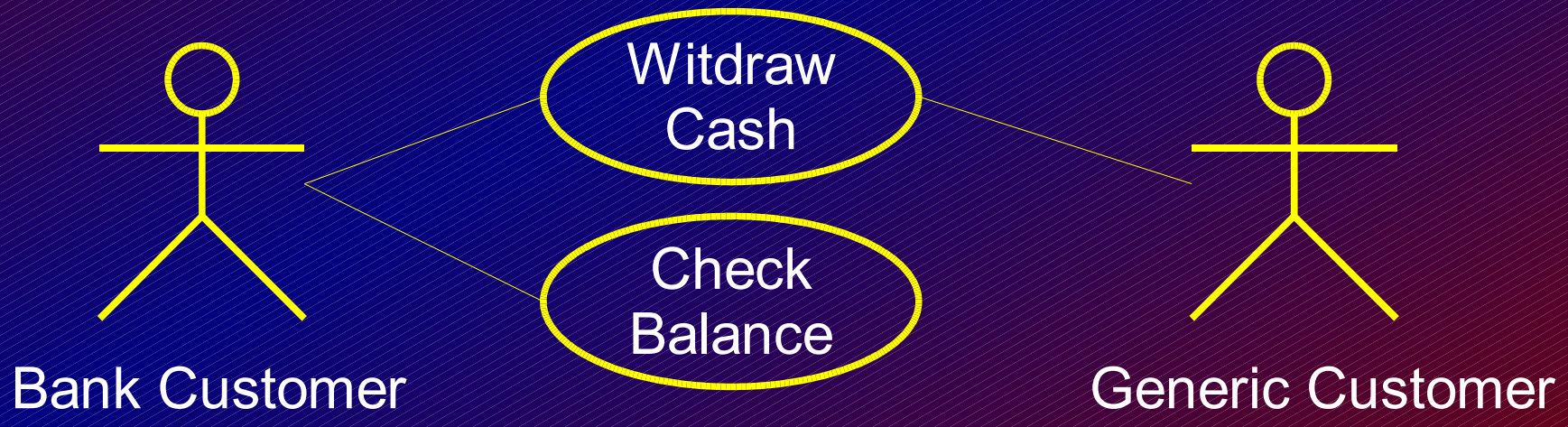


Introduction to UML

- UML is a graphical modeling notation
- Widely used to describe Object-Oriented software systems
- Informally specified
- Provides several types of diagrams
  - Use Case
  - Deployment
  - Activity
  - State
  - Class / Package
  - Collaboration
  - Sequence

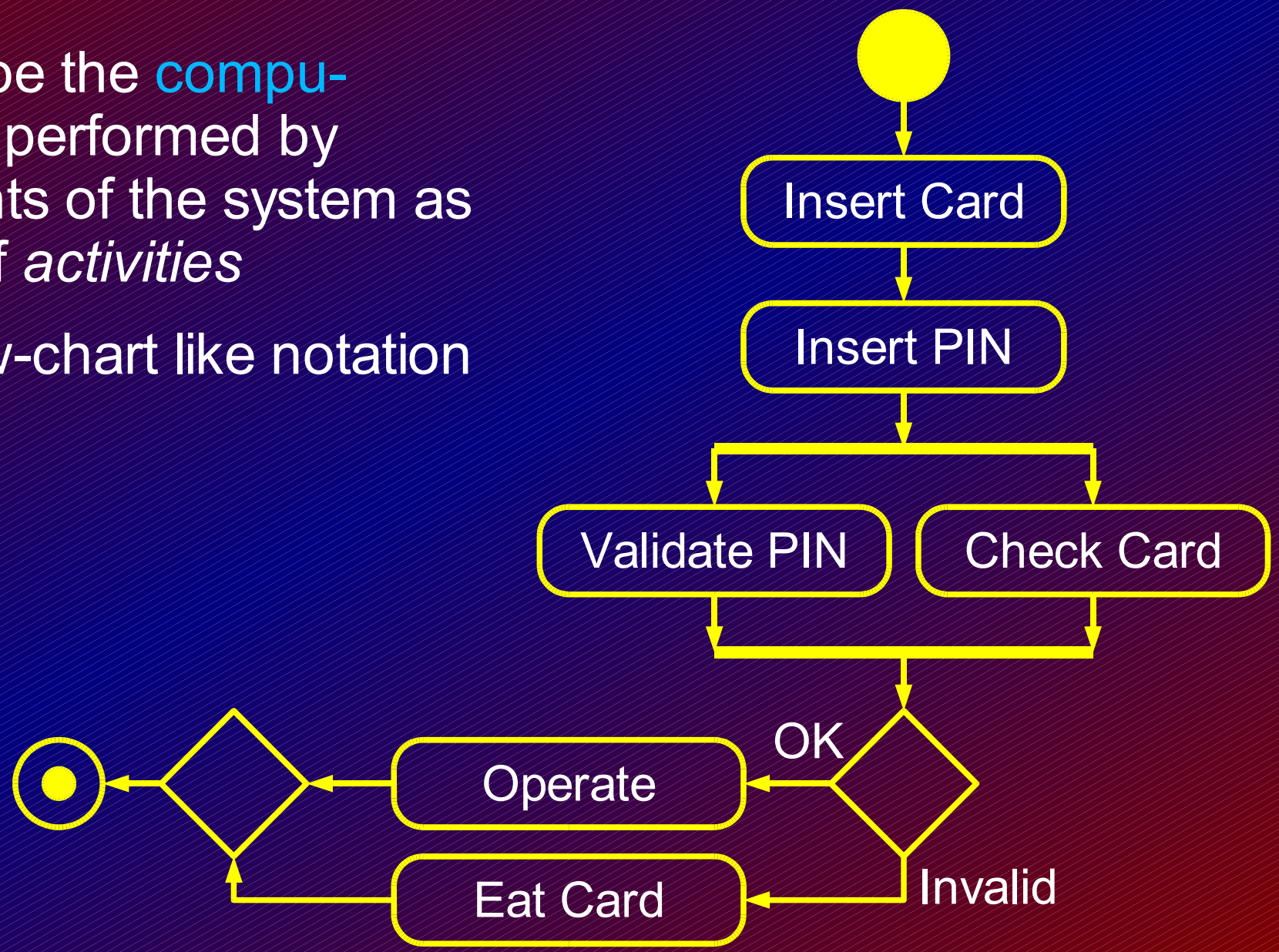
Use Case Diagram

- Contains Actors and Use Cases
  - Actors represent entities (physical or logical) which may interact with the system
  - Use Cases represent usage scenarios of the system



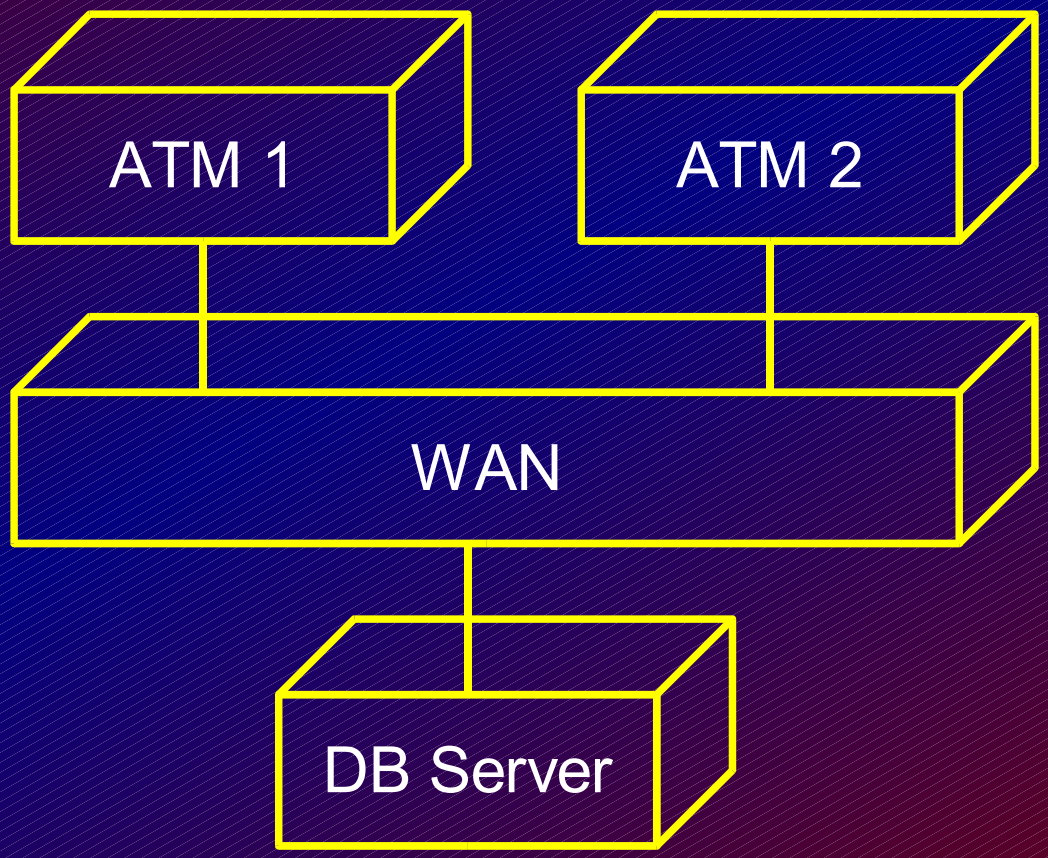
Activity Diagram

- Describe the computations performed by elements of the system as a set of *activities*
  - Flow-chart like notation

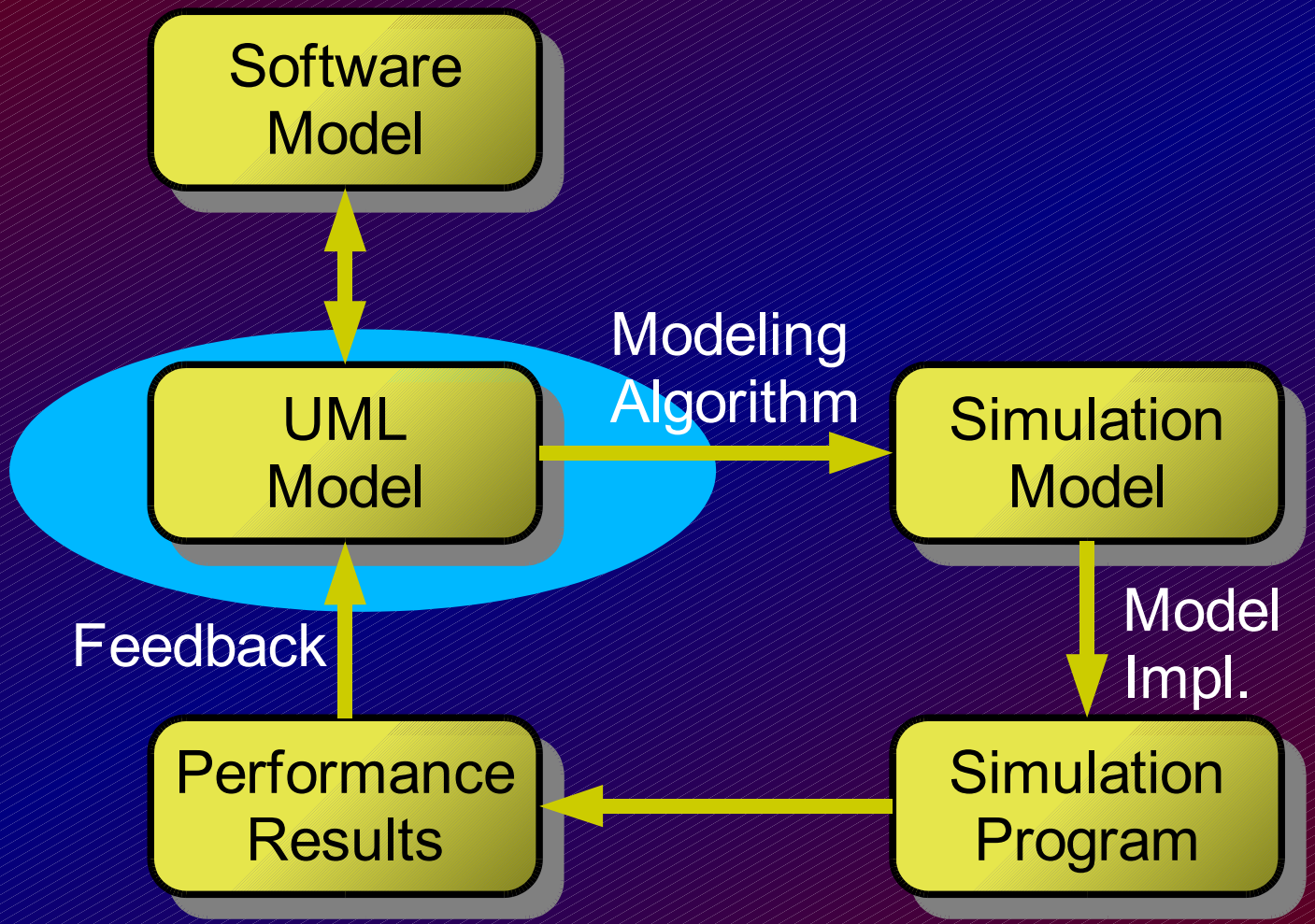


Deployment Diagram

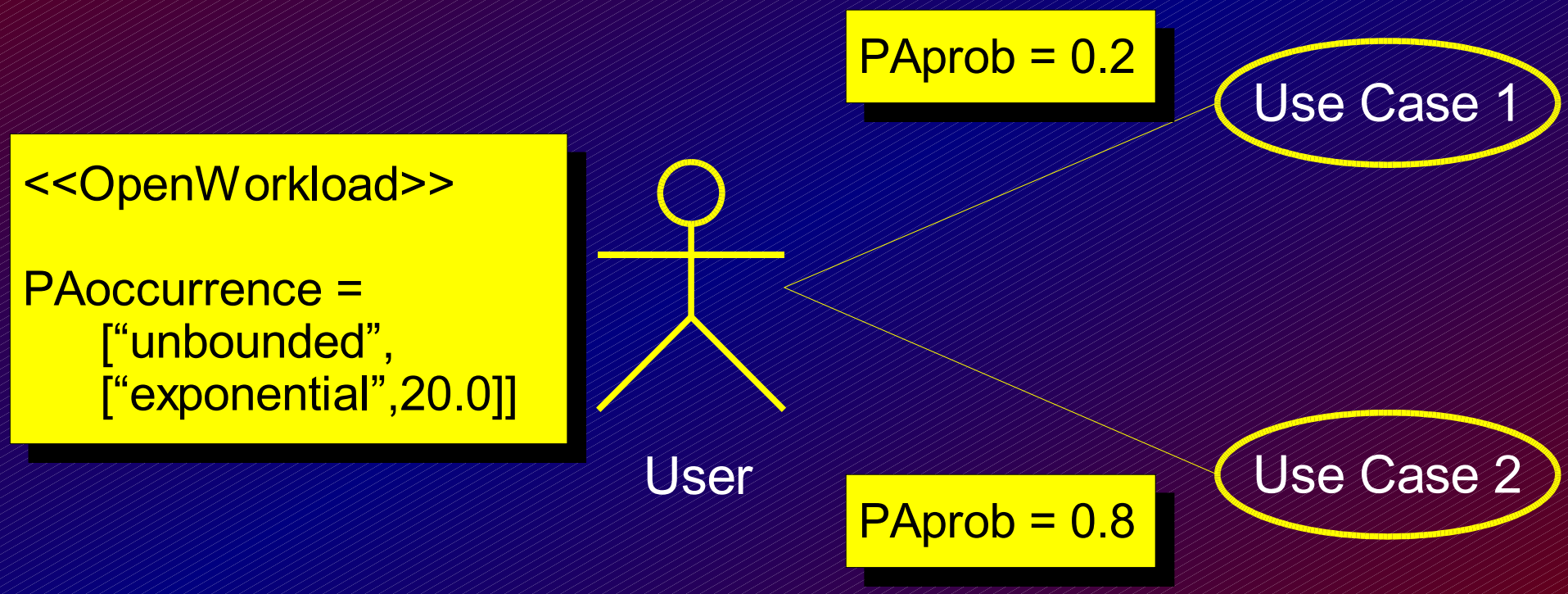
- Describe the resources available in the system



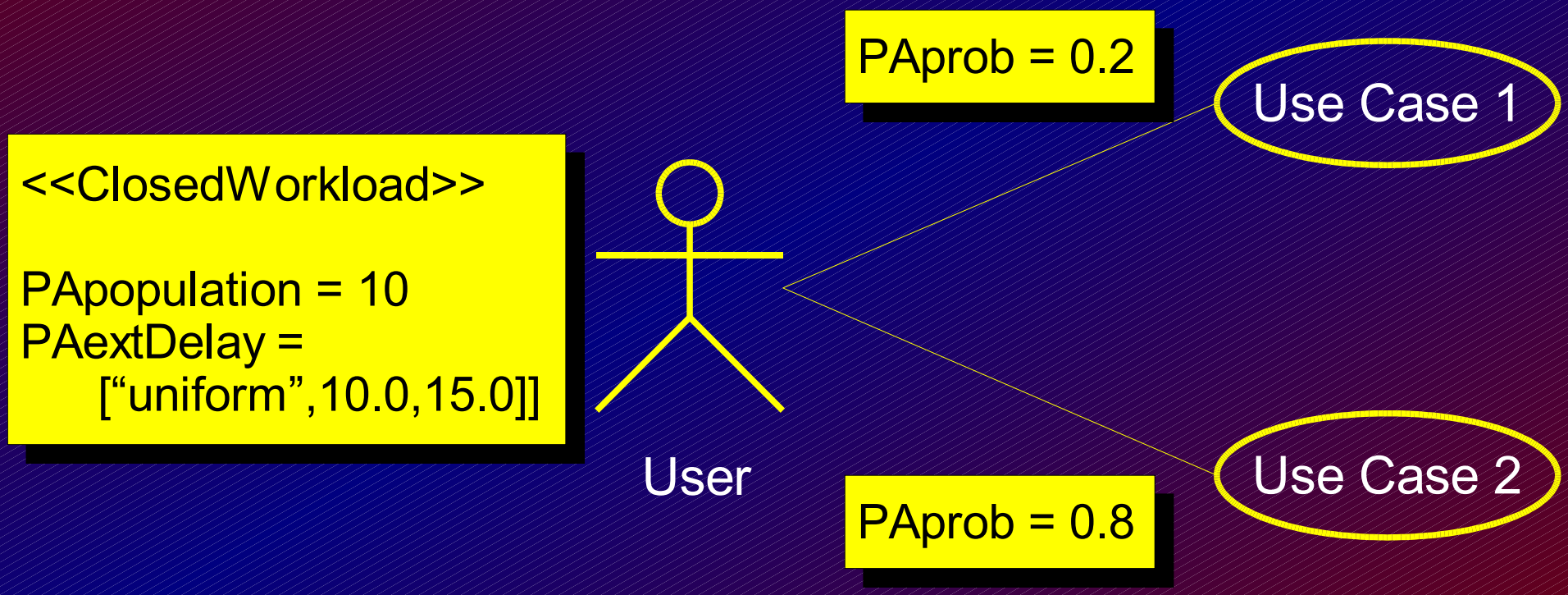
The approach



Modeling Workloads / 1

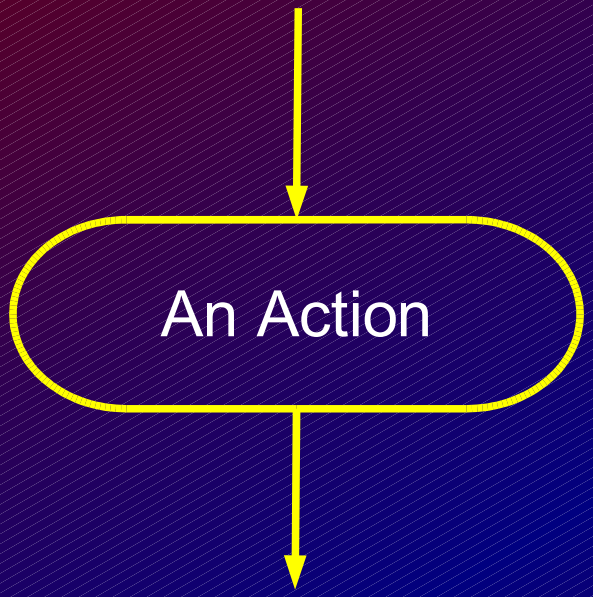


Modeling Workloads / 2



Modeling Scenarios / 1

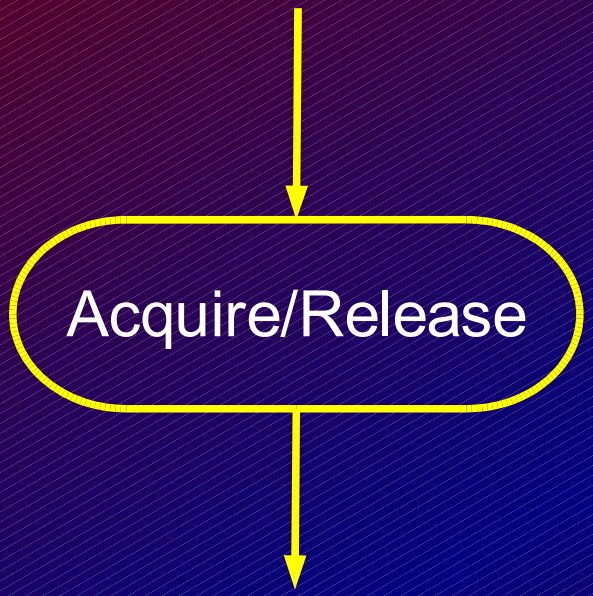
- Simple Actions




```
<<PAstep>>

PArep = 5
PAinterval = [“assm”,”dist”, [“exponential”, 0.1]]
PAdemand = [“msrd”,”dist”, [“exponential”, 0.2]]
PAhost = “Workstation”
PAdelay = [“msrd”,“dist”, [“constant”,0.1]]
PArespTime = @xx@
```

Modeling Scenarios / 2

- Resource Acquire/Release



<<GRMacquire>> / <<GRMrelease>>

PAresource = “Memory”
PAquantity = [“assm”,”dist”,[“constant”,2]]

Modeling Resources / 1

- Active Resource (processor)

<<PAhost>>

PAutilization = @xx@
PAthroughput = @xx@
PAschedPolicy = “FIFO”
PActxSwT = [“constant”, 0.1]
PArate = 2.0

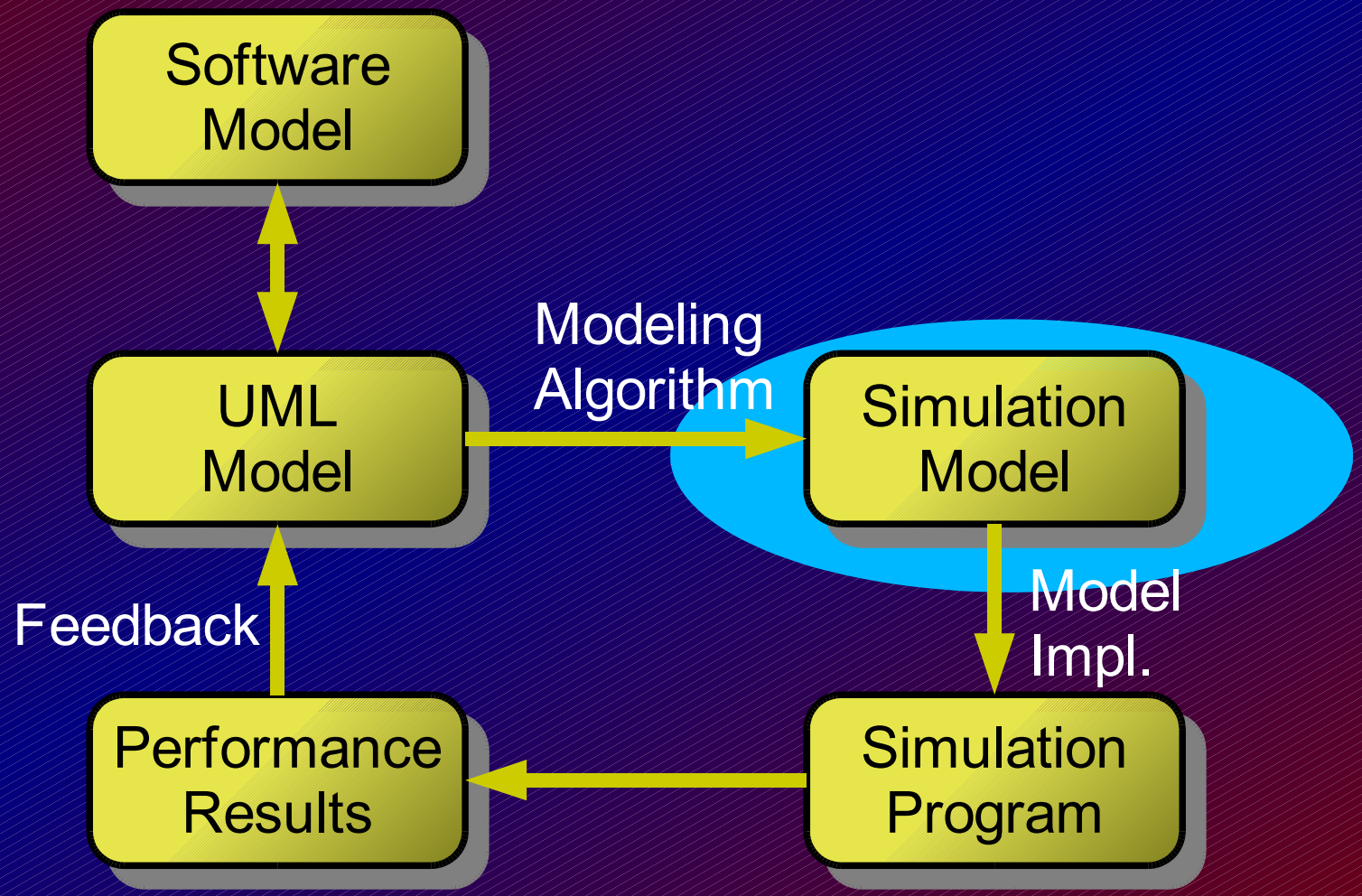
Modeling Resources / 2

- Passive Resource

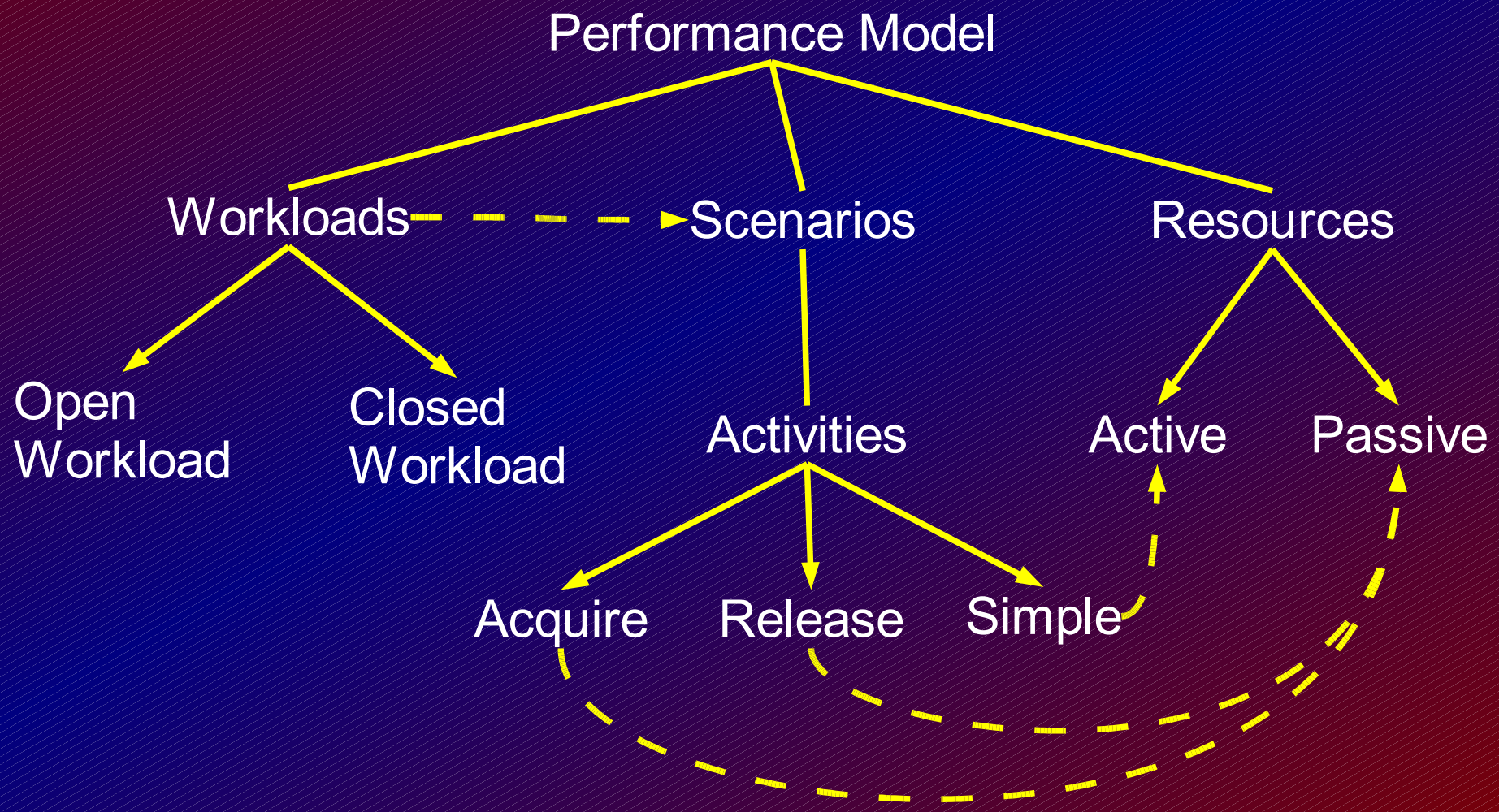
<<PAresource>>

PAutilization = @xx@
PAthroughput = @xx@
PAcapacity = 100
PAaxTime = [“uniform”,
10.0, 20.0]

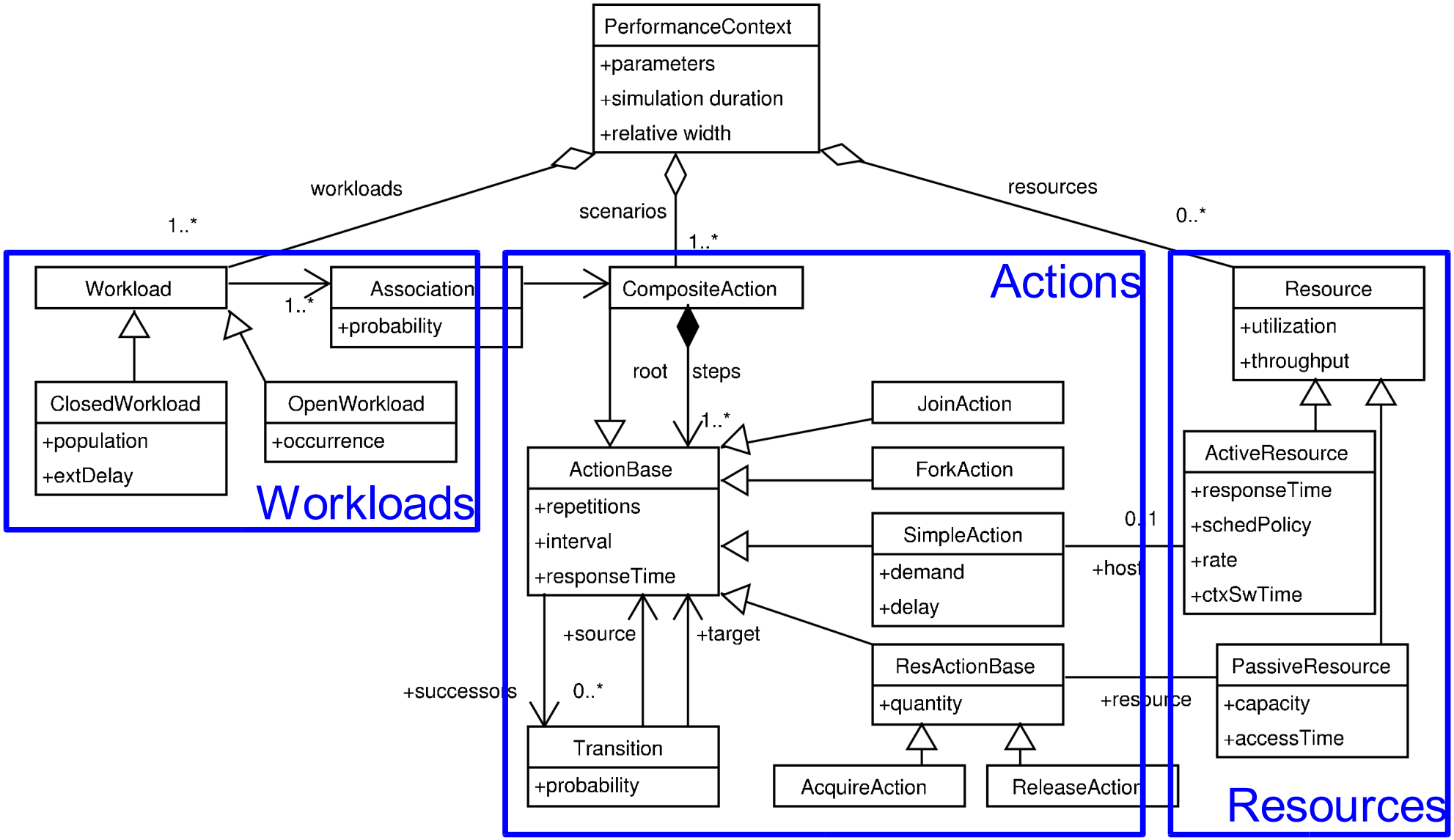
The approach



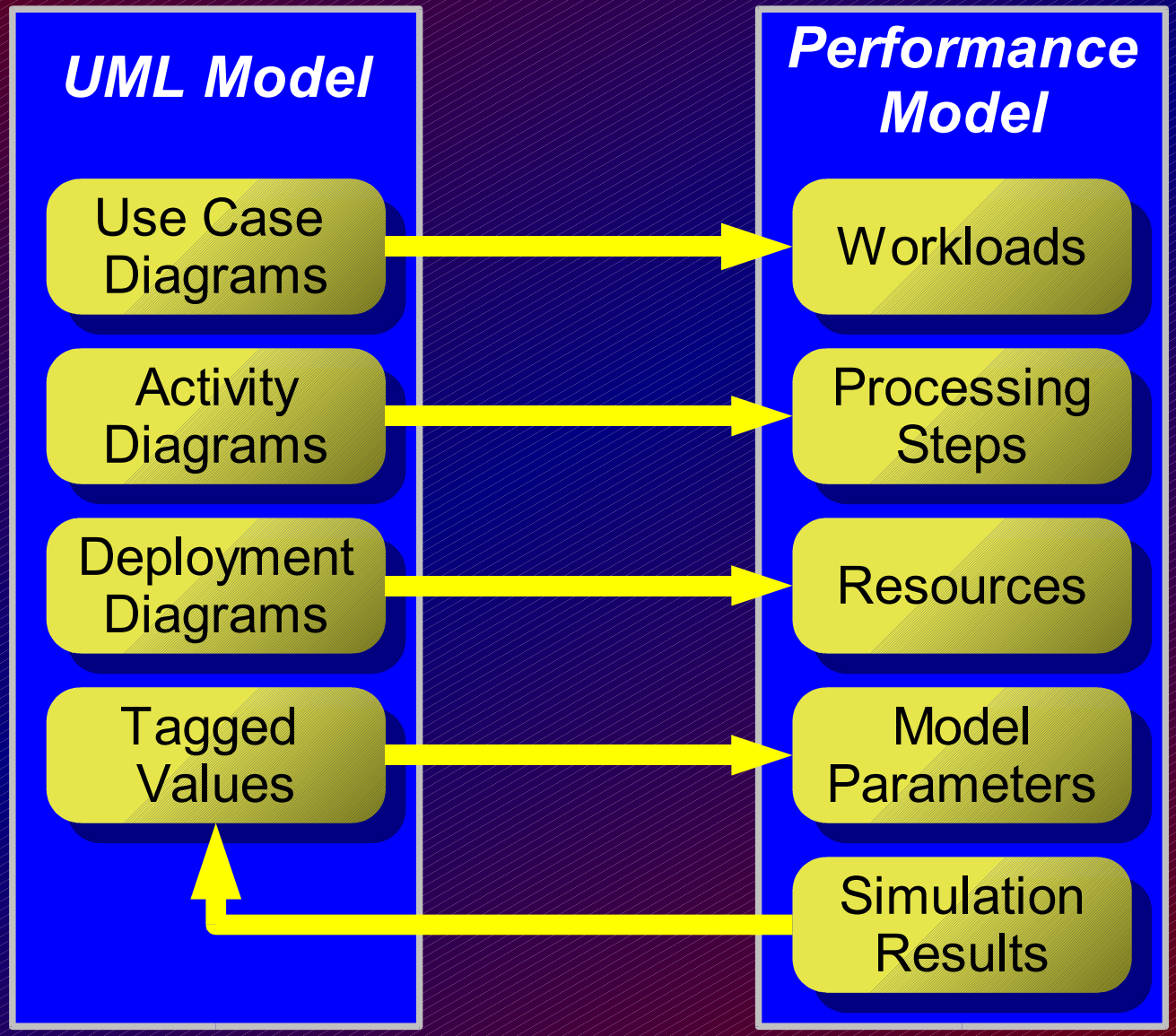
The Performance Model



Performance Model in UML



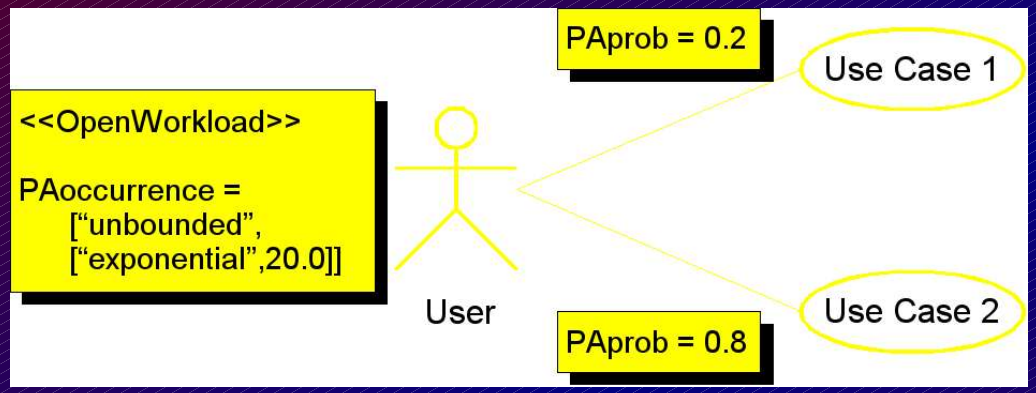
Model Generation



The Modeling Algorithm

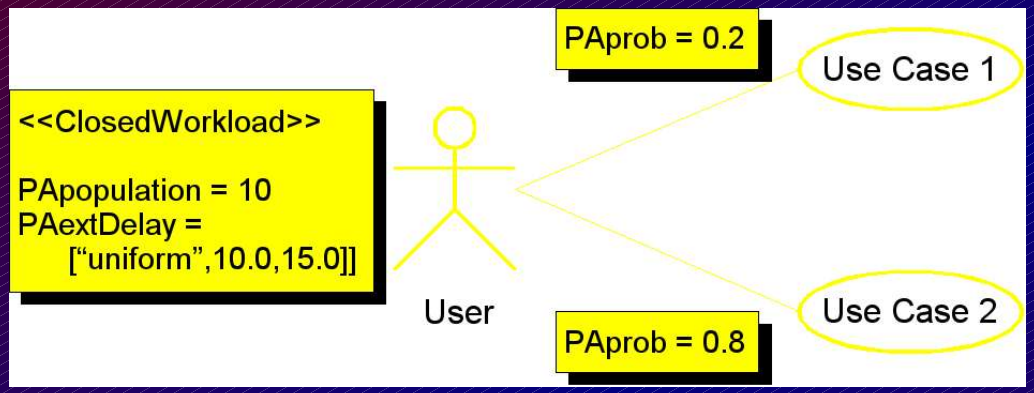
- For each Use Case diagram *U*
  - Make a simulation process of type `Open_Workload` or `Closed_Workload`, depending on the stereotype of *U*
- For each Action *A*
  - Make a simulation process of the appropriate type (`Simple_Action`, `Acquire_Action`, `Release_Action`) according with the stereotype of *A*
  - Link the action in the same predecessor-successor relationship as in the UML model
- For each Deployment diagram node *N*
  - Make a simulation process of type `Active_Resource` or `Passive_Resource`, depending on the stereotype of *N*

Translating Open Workloads



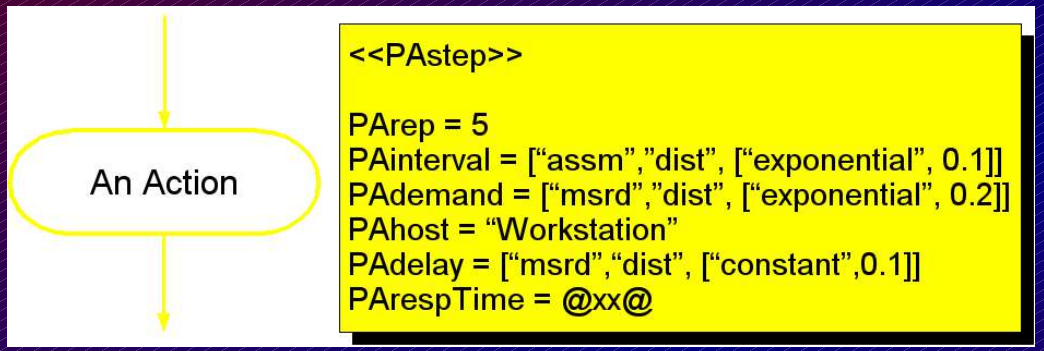

```
process OpenWorkload( Actor A );
var
   u : OpenWorkloadUser;
begin
   while ( true ) do begin
      hold( A.PAoccurrence );
      u := new( OpenWorkloadUser( A ) );
      activate( u );
   end;
end;

process OpenWorkloadUser( Actor A );
var
   i  : integer;
   uc : UseCase;
begin
   { choose i in [1..n] with
   probability [A.p[1]..A.p[n]] }
   uc := new( CompositeAction( UC[i] ) );
   activate( uc );
end;
```

Translating Closed Workloads



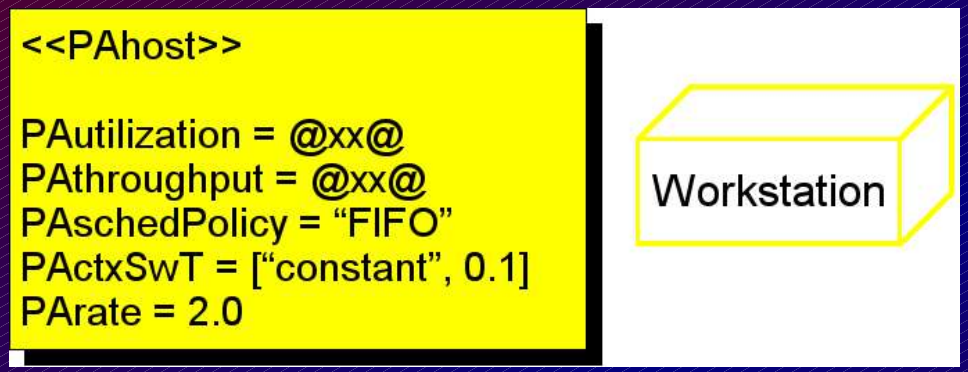

```
process ClosedWorkload( A: Actor )
var
   i : integer;
   u : ClosedWorkloadUser;
begin
   for i:=1 to A.pop do begin
      u := new( ClosedWorkloadUser( A ) );
      activate( u );
   end;
end;

process ClosedWorkloadUser( A: Actor )
var
   i  : integer;
   uc : UseCase;
begin
   while ( true ) do begin
      hold( A.PAextDelay );
      { choose i in [1..n] with
      probability [ A.p[1]..A.p[n] ] }
      uc := new( CompositeAction( UC[i] ) );
      activate( uc );
      { Wait for uc to terminate }
   end;
end;
```

Translating Simple Actions



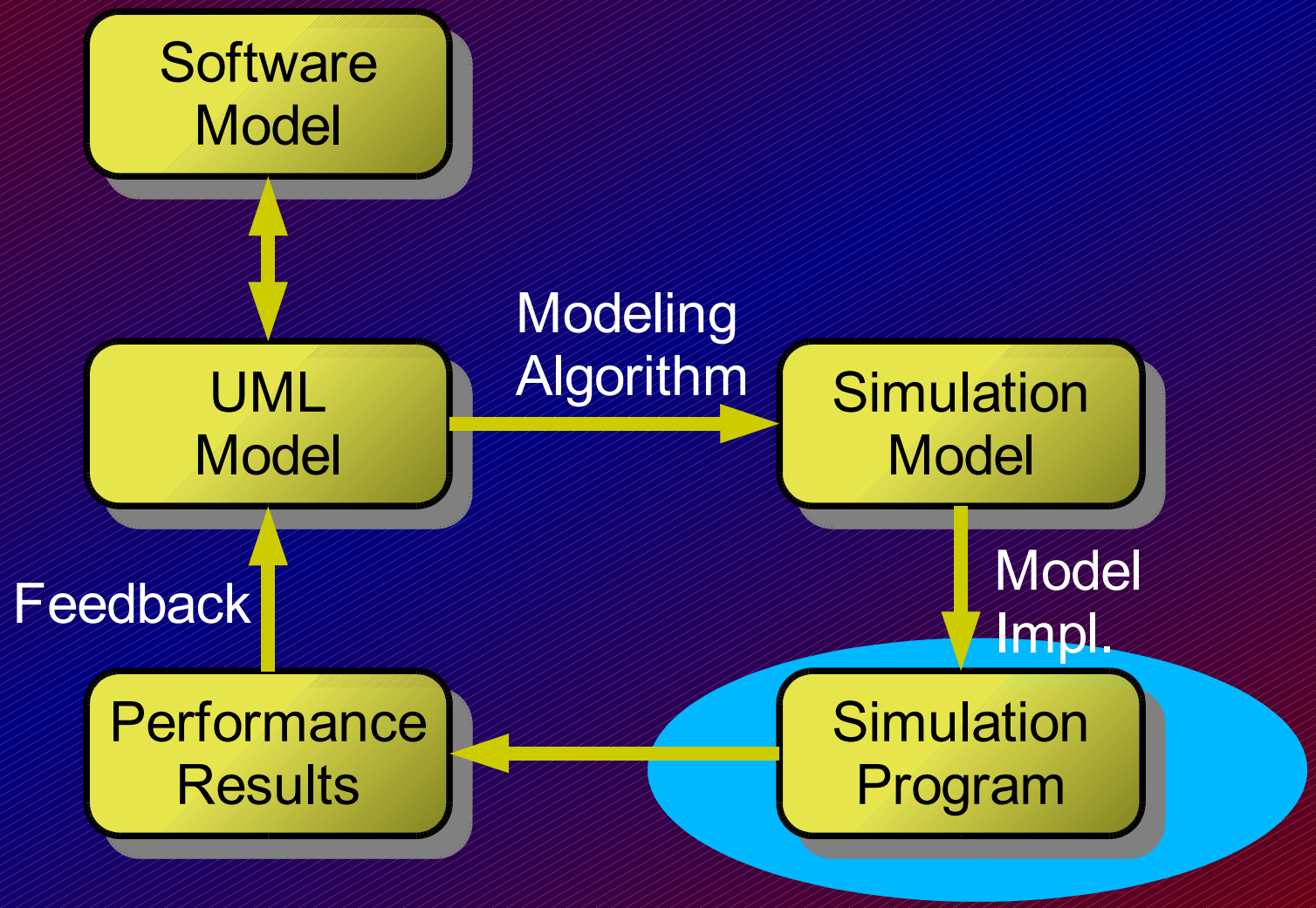

```
process SimpleAction( A: Action )
var
  i    : integer;
  next : ActionBase;
begin
  while ( true ) do begin
    hold( A.PAdelay );
    for i:=1 to A.PArep do begin
      if ( A.PAhost <> nil ) then
        { Request service to A.PAhost }
      else
        hold( A.PAdemand );
      if ( i < A.PArep )
        hold( A.PAinterval );
    end;
    { Activate next step }
    next := choose_next_action( );
    activate( next );
    { Wait for the next execution }
    passivate( );
  end;
end;
```

Translating Active Resources




```
process ActiveResource( N: NodeInstance )
var
   j    : Action;
   jobq : queue of Action; { Queue of jobs }
begin
   while ( true ) do begin
      while ( not jobq.empty( ) ) do begin
         j := jobq.first( );
         jobq.remove_first( );
         hold( j.Pademand / N.PArate );
         activate( j );
      end;
      passivate( );
   end;
end;
```

The approach



The Simulation Program

- Based on a process-oriented discrete-event simulation library in C++
- The library provides basic (simulation) process scheduling facilities (SIMULA-like)
- Basic statistical functions for output data analysis are available
  - Mean, standard deviation, quantile, histogram
  - Initialization bias removal

UML-Ψ

- Written in C++
- Parses the XMI representations of annotated UML models produced by ArgoUML
- Tag values can be written in Perl
- Automatically derives the simulation model
- Creates the simulation program using a C++ process-oriented simulation library
- Executes the simulation and returns the results into the UML model as tagged values

Modeling in action

File Edit View Create Diagram Arrange Generation Critique Help

Diagram-centric

Activity Diagram: Request Video

- untitledModel
  - Deployment Diagram
  - Request Video
    - Confirm
    - Handle Selection
    - Initialize Player
    - Initiate Playout
    - PAhost = "ClientWorkstatio
    - PAhost = "ClientWorkstatio
    - PAhost = "ClientWorkstatio
    - PAhost = "Internet"PAdema
    - PAhost = "VideoServerNode
    - PAhost = "WebServerNode"P
    - Pahost = "ClientWorkstatio
    - Pahost = "VideoServerNode
    - Request Video Start
    - Request Video Stop
    - Select Service
    - Send Terminate
    - Send Video
    - Terminate Player
    - anonymous
    - anonymous

Birdview | Priority

Select Service
PAhost = "ClientWorkstation"
PAdemand = ["assm","dist", ["exponential", 1.0 / 10.0]]

Handle Selection
PAhost = "WebServerNode"
PAdemand = ["est","dist", ["constant", 2.0]]

Confirm
PAhost = "ClientWorkstation"
PAdemand = ["msrd","dist", ["exponential", 1.0 / 0.5]]

Initiate Playout
PAhost = "VideoServerNode"
PAdemand = ["assm", "dist", ["exponential", 1.0 / 5.0]]

Initialize Player
Pahost = "ClientWorkstation"
PAdemand = ["msrd", "dist", ["constant", 0.5]]

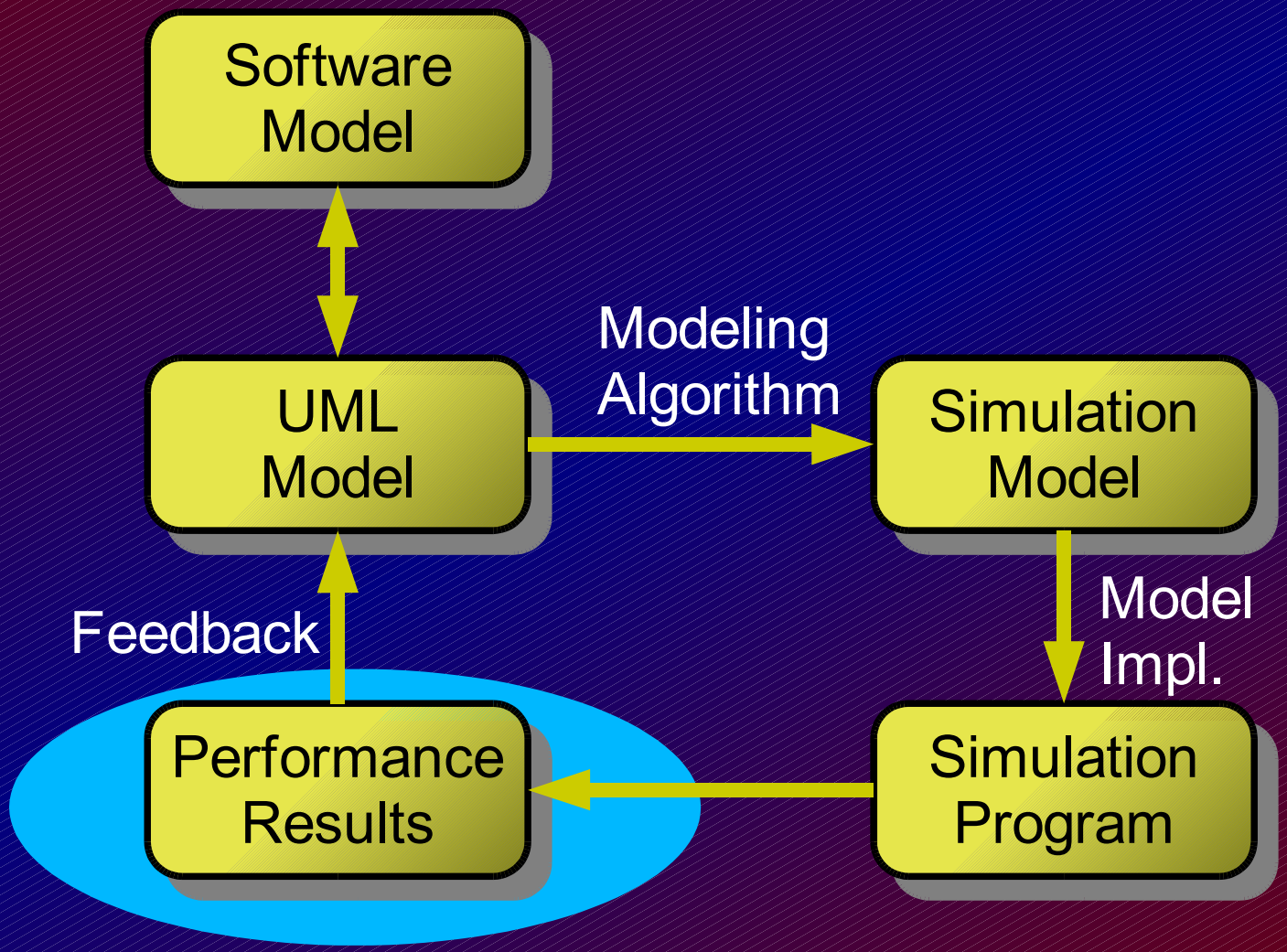
Send Video
PAhost = "Internet"
PAdemand = ["assm", "dist", ["exponential", 1.0 / 0.5]]
PArep = 100

Send Terminate

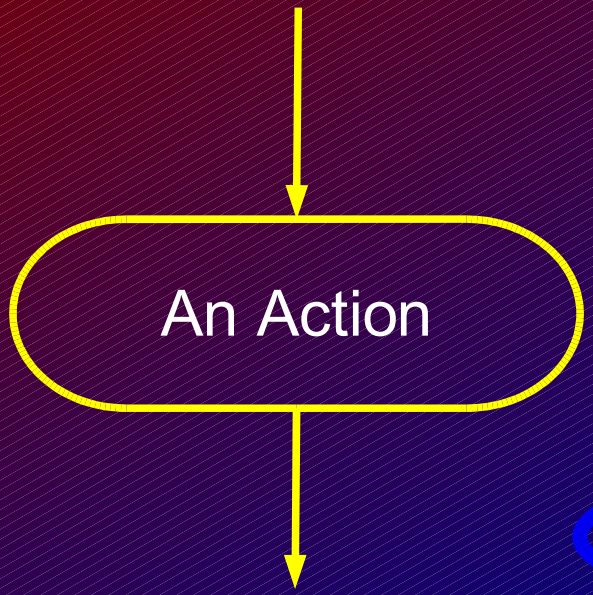
Properties | Style | Documentation | Java Source | Constraints | Tagged Values

Tag	Value
PArespTime	0.0
PAhost	"ClientWorkstation"
PAdemand	["assm","dist",["exponential",1.0 / 10.0]]

The approach



Performance Results



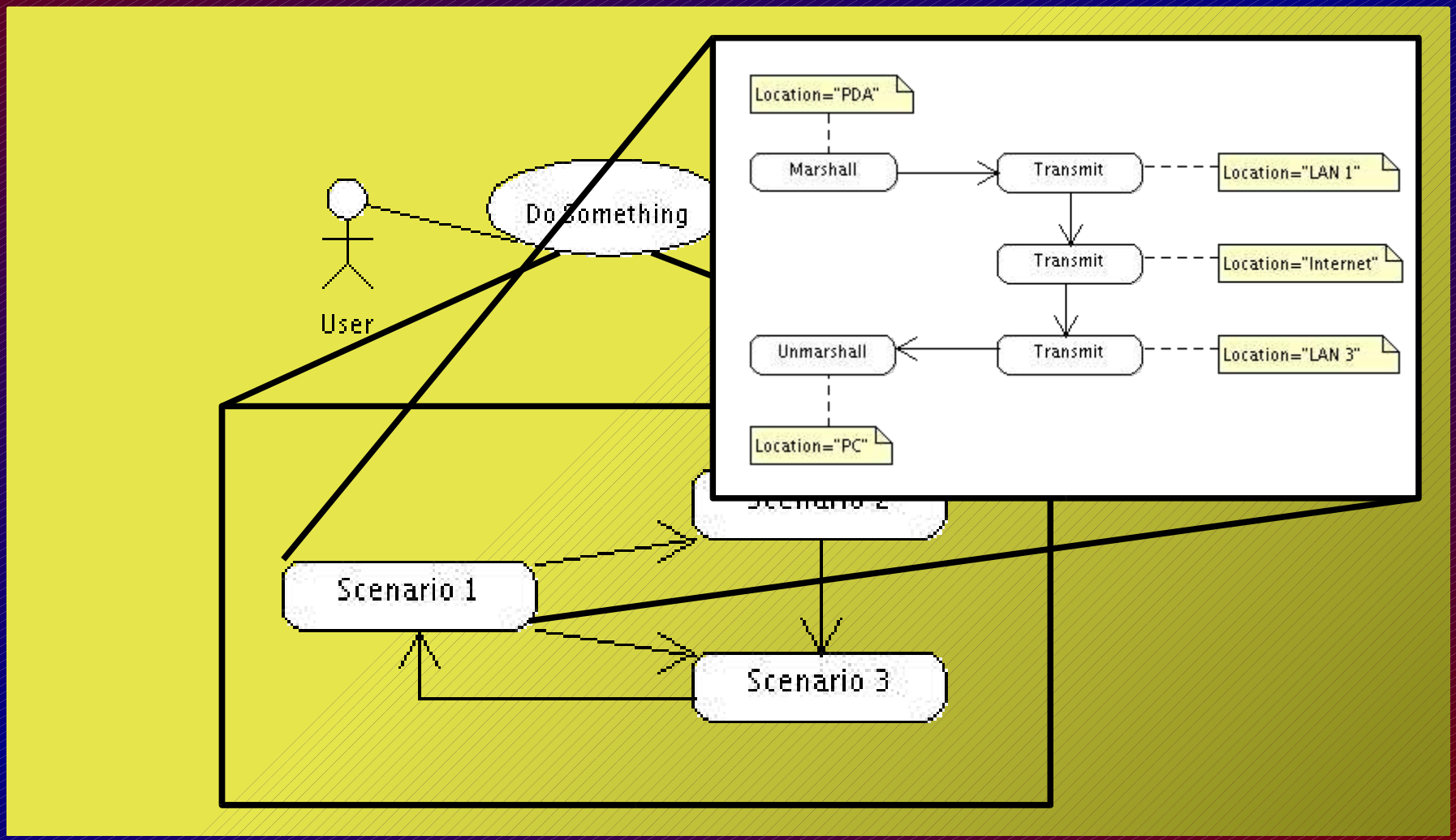
<<PAstep>>

PArep = 5
PAinterval = [“assm”,”dist”, [“exponential”, 0.1]]
PAdemand = [“msrd”,”dist”, [“exponential”, 0.2]]
PAhost = “Workstation”
PAdelay = [“msrd”,“dist”, [“constant”,0.1]]
PArespTime = @xx@

<<PAhost>>

PAutilization = @xx@
PAthroughput = @xx@
PAschedPolicy = “FIFO”
PActxSwT = [“constant”, 0.1]
PArate = 2.0

Mobility and Performance Modeling with UML



Conclusions

- We described how performance analysis can be done at the SA design level
- A UML profile has been defined for adding quantitative annotations to UML models
- We define a process-oriented simulation model of a SA
- A prototype tool (UML-Ψ) has been developed
  - Parses annotated UML diagrams saved in XMI format
  - Generates simulation model
  - Executes simulation and reports feedback

Open Problems

- UML-related improvements
  - Include more types of UML diagrams in the software model, such as State Charts and Sequence diagrams
- Simulation-related improvements
  - Compute more performance indices
- Further improvements
  - Integrate the approach into a general framework including different kinds of functional and non functional analyses

Related publications

- S. Balsamo and M. Marzolla. *Simulation Modeling of UML Software Architectures* Proc. ESM'03, Nottingham - UK, Jun 9-11 2003, pp. 562-567
- S. Balsamo and M. Marzolla. *A Simulation-Based Approach to Software Performance Modeling*. Proc. ESEC/FSE 2003, Helsinki - FI, Sep 1-5 2003, pp. 363-366
- S. Balsamo and M. Marzolla. *Towards Performance Evaluation of Mobile Systems in UML*. Proc. ESMc'03, Napoli, Oct 27-29 2003, pp. 61-68
- S. Balsamo, A. Di Marco, P. Inverardi and M. Marzolla. *Experimenting Different Software Architectures Performance Techniques: A Case Study*. To appear in Proc. WOSP 2004.
- All available on http://www.dsi.unive.it/~marzolla/publications.html