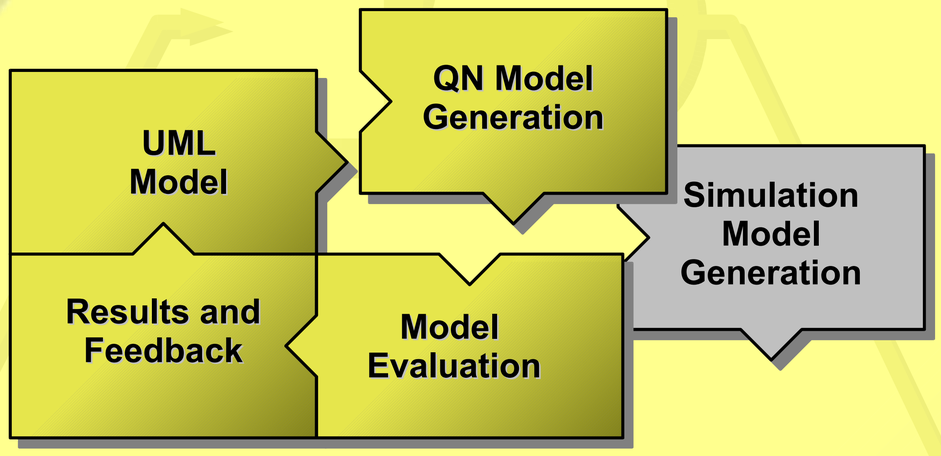
Performance Evaluation of Software Architectures with Multiclass Queueing Network Models

Simonetta Balsamo
balsamo@dsi.unive.it
http://www.dsi.unive.it/~balsamo

Moreno Marzolla
marzolla@dsi.unive.it
http://www.dsi.unive.it/~marzolla

Dipartimento di Informatica Università *Ca' Foscari* di Venezia
via Torino, 155 30172 Mestre (Venezia) ITALY

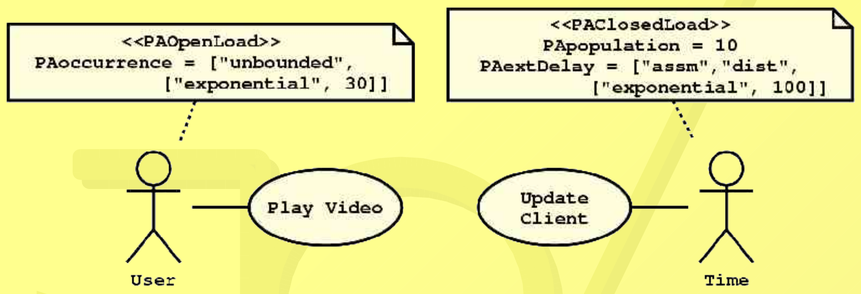
Software performance based on performance models can be applied at early phases of the software development cycle to characterize the quantitative behavior of software systems. We propose an approach based on queueing networks models for performance prediction of software systems at the software architecture level, specified by UML.
Starting from annotated UML **Use Case, Activity and Deployment diagrams** we derive a performance models based on multichain and multiclass Queueing Networks (QN). The UML model is annotated according to the UML Profile for Schedulability, Performance and Time Specification.



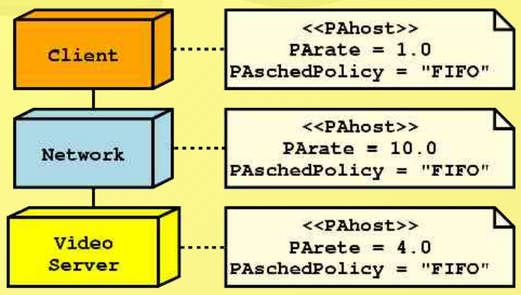
We translate the annotated UML specification into QN performance models, which can then be analyzed using standard solution techniques. Performance results are reported back at the software architecture level in the UML diagrams. As our approach can be fully automated and uses standard UML annotations, it can be integrated with other performance modeling approaches. Specifically, we discuss how this QN-based approach can be integrated with an existing simulation-based software performance modeling tool.

- Use Case Diagrams describe workloads—either open or closed—applied to the system. Annotations specify the type of workload and its parameters (population or interarrival time).
- Activity Diagrams describe the computations performed by the system in term of service requests to resources (processors). Annotations specify the service demand and the resource name on which the action executes.
- Deployment Diagrams describe the resources (processors) available in the system. Annotations specify the processing rate

Use Case Diagram



Deployment Diagram



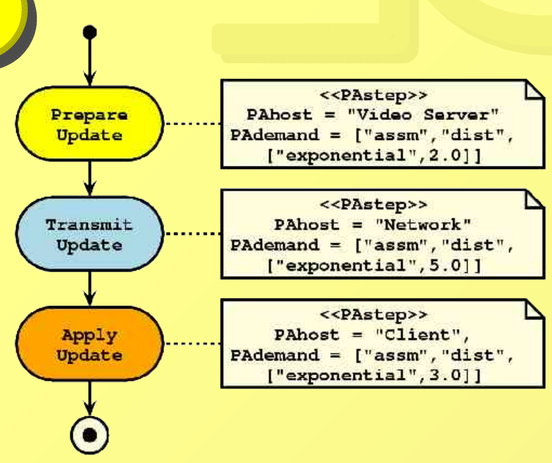
Algoritm 1: Generation of a closed QN

Requires:
Activity diagram G = (A, T),
Deployment diagram R,
Actor W stereotyped as «PAclosedLoad»
for all r ∈ R **do Let** count [r] := 0
for all a ∈ A **do**
count[res[a]] := count[res[a]] + 1
index[a] := count[res[a]]
C := $\max_{r \in R}$ {count [r]} *{Number of Classes}*
K := |R| + 1 *{Number of Service Centers}*
P = P[i, r, j, s] := **new vector** [K × C × K × C]
λ = λ[i, r] := **new vector** [K × C]
μ = μ[i, r] := **new vector** [K × C]
for all t = (x, y) ∈ T **do** *{Define the routing matrix P}*
i := id[res[x]]
r := index[x]
j := id[res[y]]
s := index[y]
P[i, r, j, s] := p[t]
for all a ∈ A - {a_0} **do** *{Define Service Rates μ}*
i := id[res[x]]
r := index[a]
μ[i, r] := rate[r]/demand[a]
Let service center 0 be a Delay Center
μ[0, ·] := 1/extdelay[W] for all job classes
N := population[W]

Algoritm 2: Generation of an open QN

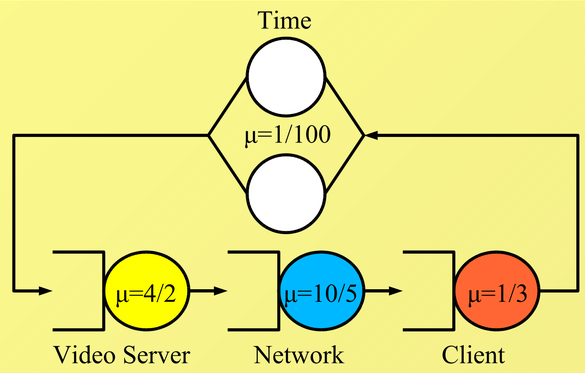
Requires:
Activity diagram G = (A, T),
Deployment diagram R,
Actor W stereotyped as «PAopenLoad»
for all r ∈ R **do Let** count [r] := 0
for all a ∈ A **do**
count[res[a]] := count[res[a]] + 1
index[a] := count[res[a]]
C := $\max_{r \in R}$ {count[r]} *{Number of Classes}*
K := |R| *{Number of Service Centers}*
P = P[i, r, j, s] := **new vector** [K × C × K × C]
λ= λ[r] := **new vector** [C]
q = q[i, r] := **new vector** [K × C]
μ = μ[i, r] := **new vector** [K × C]
{Define the routing matrix P}
for all t = (x, y) ∈ T such that x ≠ a_0 and y ≠ a_0 **do**
i := id[res[x]]
r := index[x]
j := id[res[y]]
s := index[y]
P[i, r, j, s] := p[t]
for all a ∈ A - {a_0} **do** *{Define Service Rates μ}*
i := id[res[x]]
r := index [a]
μ[i, r] := rate[r]/demand[a]
{Define Arrival Rate λ}
for all t = (x, y) such that x = a_0 **do**
i := id[res[y]]
r := index[y]
λ[r] := arrivalrate[W]
q[i, r] := p[t]

1 "Update Client" Activity Diagram



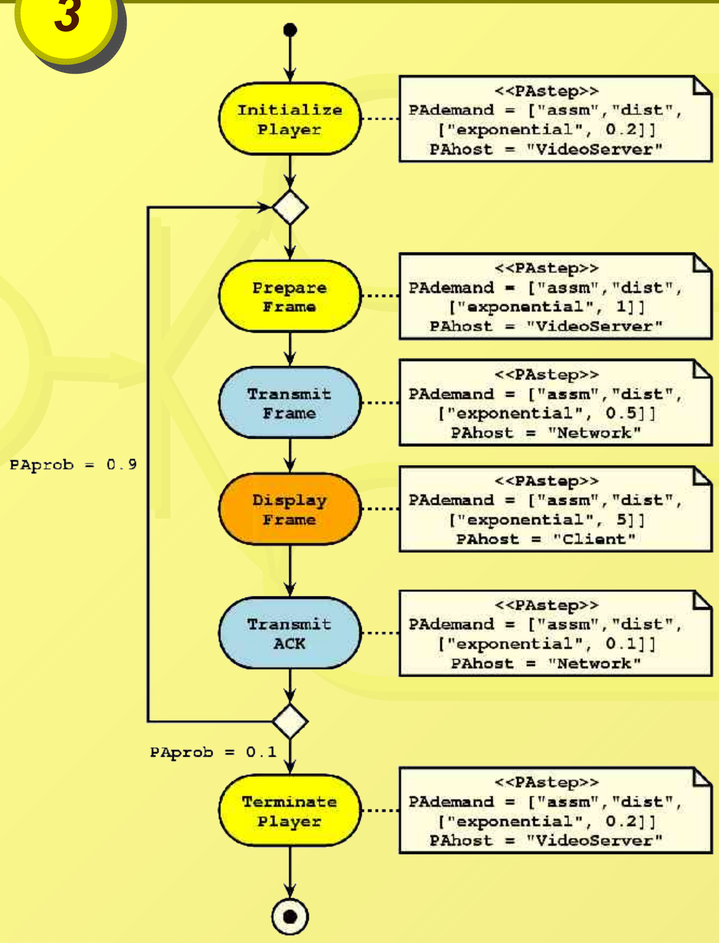
Algorithm 1

2 Closed QN for the "Update Client" Activity Diagram



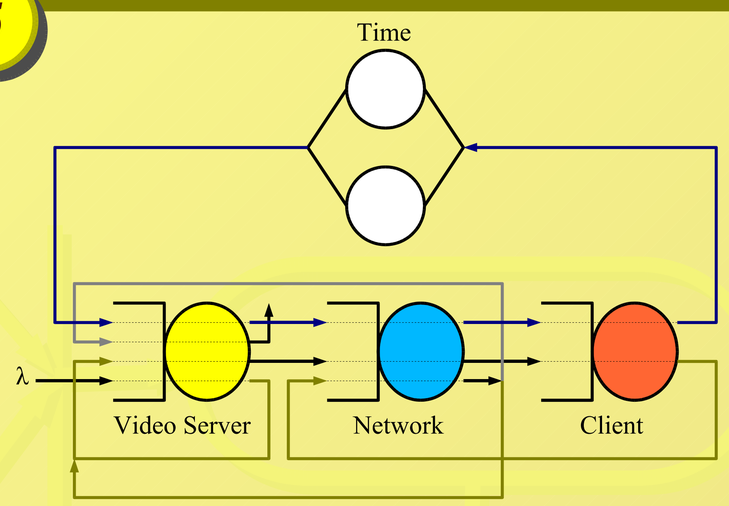
Merge

3 "Play Video" Activity Diagram



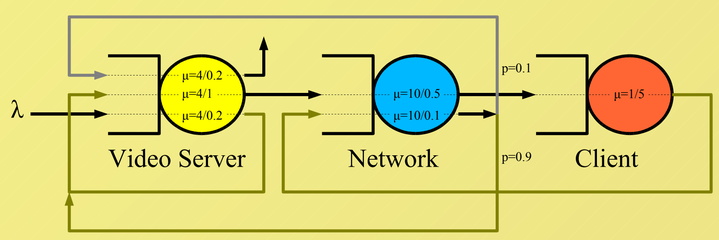
Algorithm 2

5 The complete Queueing Network



Merge

4 Open QN for the "Play Video" Activity Diagram



Conclusions and future work

We propose an algorithm for automatic translation of annotated UML specifications into multiclass QN performance models. We consider UML specifications in terms of Use Case, Activity and Deployment diagrams annotated with quantitative, performance-oriented annotations according to the UML SPT Profile. The resulting multiclass QN model can be analyzed with well-known exact or approximate solution algorithms. performance results are reported back into the software model as tagged values associated to the corresponding UML elements. We plan to extend the approach to UML 2.0 with the Profile for QoS.

WOSP'05, Palma de Mallorca, Spain, July 11—14, 2005