PaCo @

Performability Modeling & Analysis in UML

March 2-3, 2010: PaCo second mid-term meeting
(L'Aquila, Italy)

Luca Berardinelli | luca.berardinelli@univaq.it|
Dipartimento di Informatica
Università dell’Aquila (ITALY)

PACO: project macro objectives

1. Studyng of logic, models and languages for:
   a) Modeling performability-aware systems;
   b) Specification of performability metrics;
2. Definition of model transformation functions
   a) From design to (multiple) analysis models (direct);
   b) Among different analysis models (direct/inverse);
   c) From analysis to design models (inverse);

L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, (**NFPinDSML 2009 @ MODELS 2009**)
"UML Profiles for Non-Functional Properties at Work: Analyzing Reliability, Availability and Performance",
Proc. of the 2nd International Workshop on Non-functional System Properties in Domain Specific Modeling Languages

L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, (**QoSA 2010**)
" The Fault-error-failure Chain: a Challenge for Modeling and Analyzing Performability in UML-based Software Architectures ", **Submitted** to the Sixth International Conference on the Quality of Software Architectures (QoSA 2010)

UML Profiles for NFPs at Work

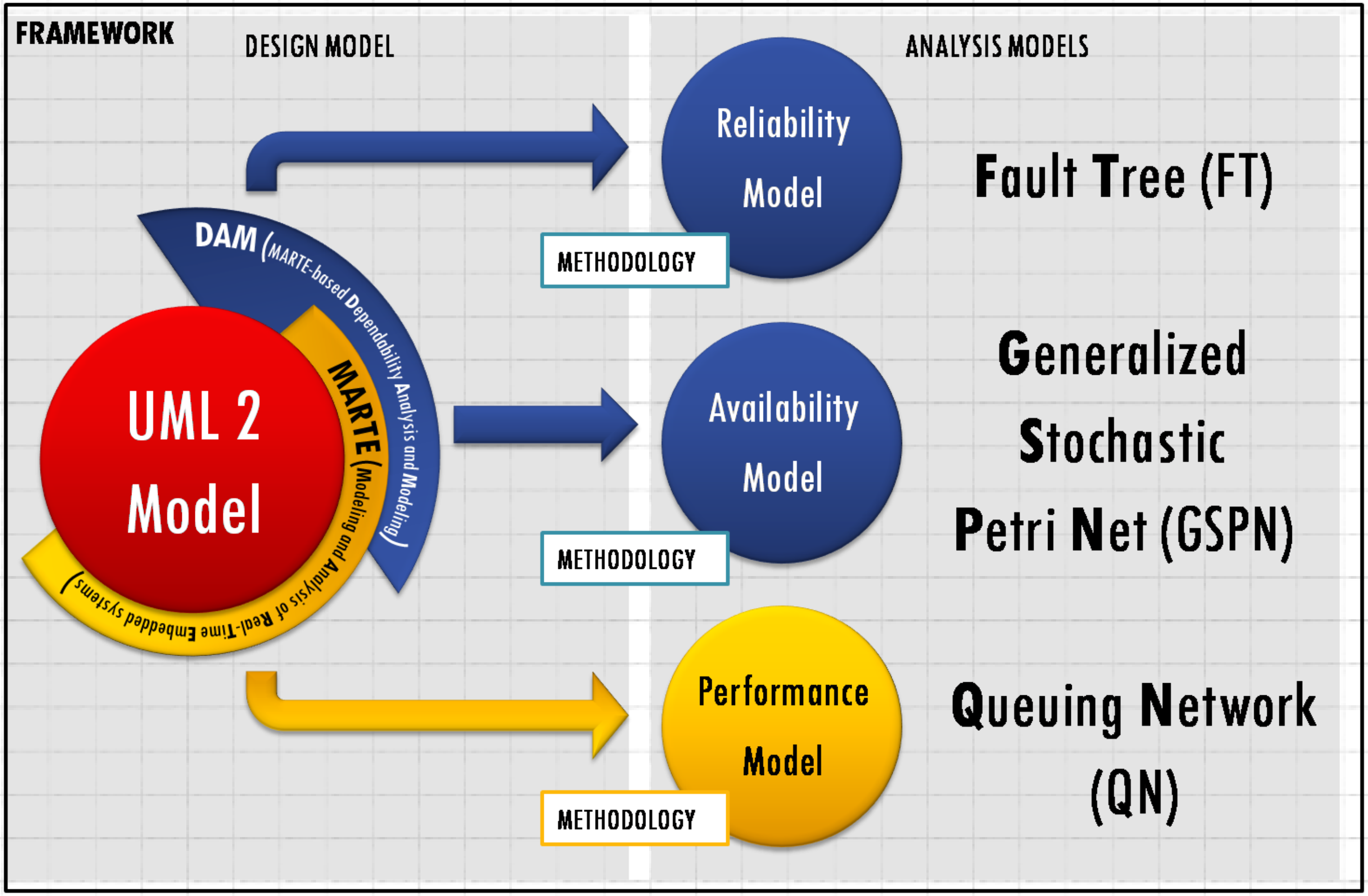
1. Studyng of logic, models and languages for:
   a) Modeling performability-aware systems:
      - **Input**: UML Design Model. **Output:** Petri Net, Queuing Network, Fault Tree
      - Integration of different NFPs by I/O paramter sharing: Dependability (Reliability, Availability), Performance

   b) Specification of performability metrics: Not considered
2. Definition of model transformation functions
   a) From design to analysis models (direct):
      - **Unique Source** Design Model (UML), **Multiple Target** Analysis Model
      - **Methodologies** and **Tools**

   b) Among different analysis models (direct/inverse);
   c) From analysis to design models (inverse);

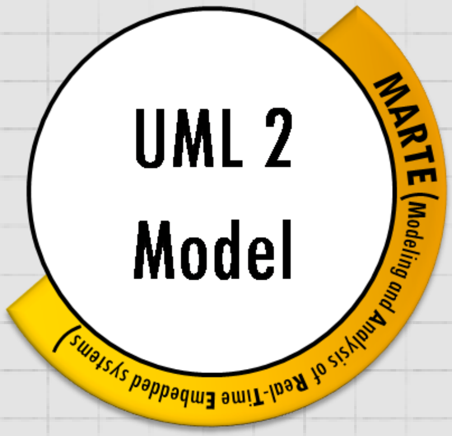
L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, **(NFPinDSML 2009 @ MODELS 2009)**
"UML Profiles for Non-Functional Properties at Work: Analyzing Reliability, Availability and Performance",
Proc. of the 2nd International Workshop on Non-functional System Properties in Domain Specific Modeling Languages

UML Profiles for NFPs at Work:the Framework



UML Profiles for NFPs at Work: MARTE

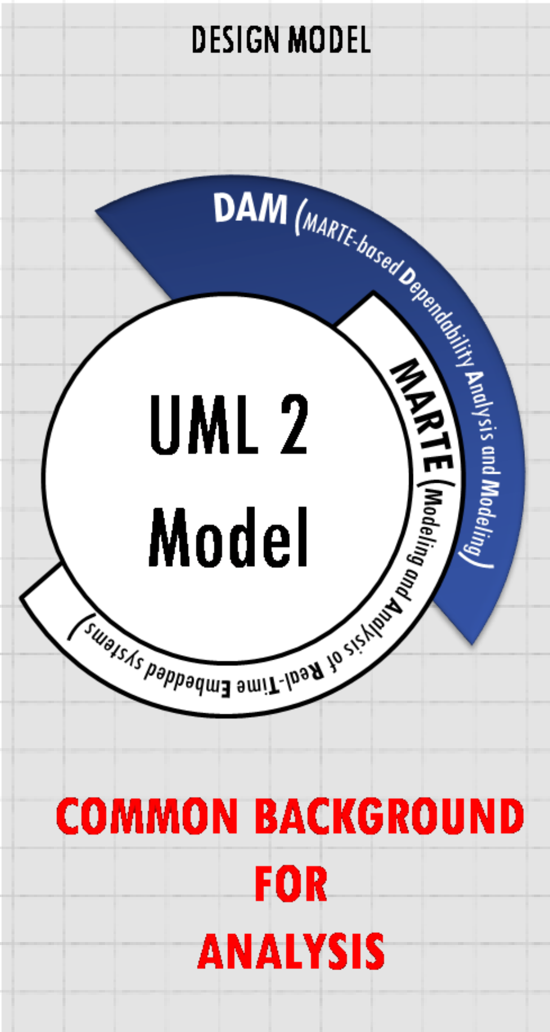
DESIGN MODEL



COMMON BACKGROUND
FOR
ANALYSIS

- UML **lightweight extension (profile)** for **M**odeling and **A**nalysis of **R**eal-**T**ime **E**mbedded systems
- Allows the **Design** of **Software** and **Hardware Resources**
- Allows the **specification of NFPs of Sw/Hw Resources** using the Value Specification Language (VSL) for
  - Generic Quantitative Analysis (GQAM)
  - Schedulability Analysis (SAM)
  - PERFORMANCE ANALYSIS (PAM)

UML Profiles for NFPs at Work: DAM



- UML **lightweight extension (profile)** for **D**ependability **A**nalysis **M**odeling.
- S. Bernardi, J. Merseguer, and D.C. Petriu.
  A Dependability Profile within MARTE.
  Journal of Software and Systems Modeling, 2009.
- **Built upon MARTE**: it reuses **MARTE Foundation** and Generic Quantitative Analysis Model (**GQAM**)
- It support the specification of dependability properties and requirements such as **<u>Reliability</u>**, **<u>Availability</u>**, Maintainability and Safety

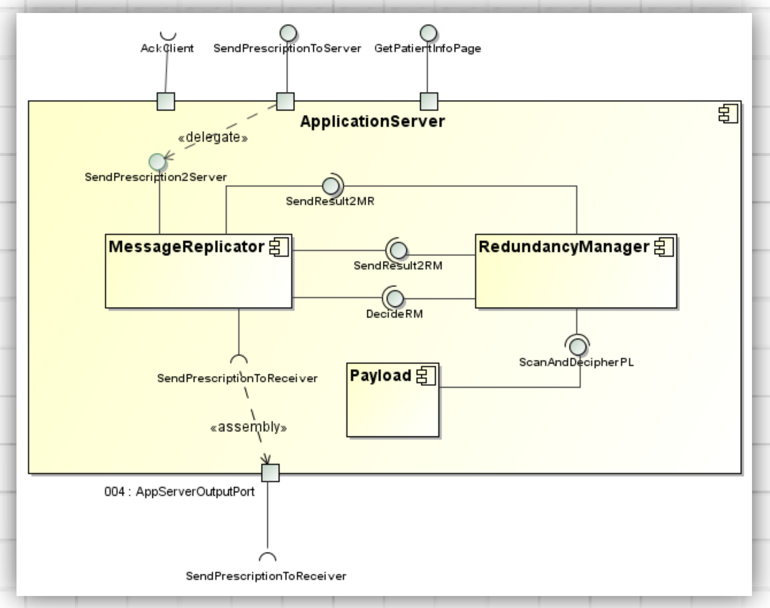
UML Profiles for NFPs at Work: eHealth

A doctor provided with his/her generic computing device (e.g. PDA, laptop), through **medical distributed services**, is able to:

Make Prescription : after visiting the patient, the doctor can make a prescription to be sent to the hospital where eventually the patient will take the medicines.

This service requires some particular **Availability**, **Reliability** and **Performance** requirements

The eHealth System is equipped with Message Redundancy Systems:



Redundancy Manager:

(i) create a Message Replicator for each prescription sent to the Application Server

Message Replicator:

(i) replicates n-times the prescription message,
(ii) create and destroy Payloads
(iii) assign each replica to a different Payload
(iv) calculate the voting result.

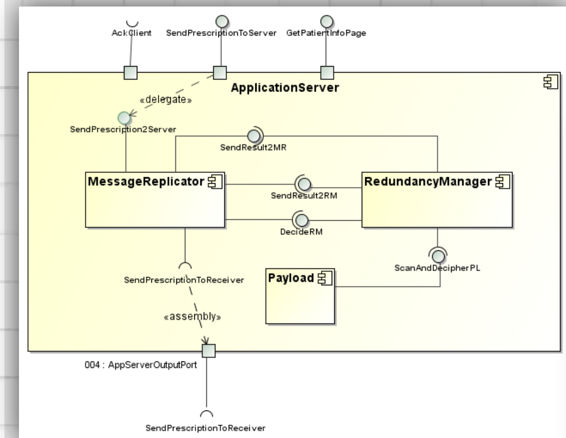
Payload:

(i) scan & decipher message replicas and
(ii) votes for replica integrity.

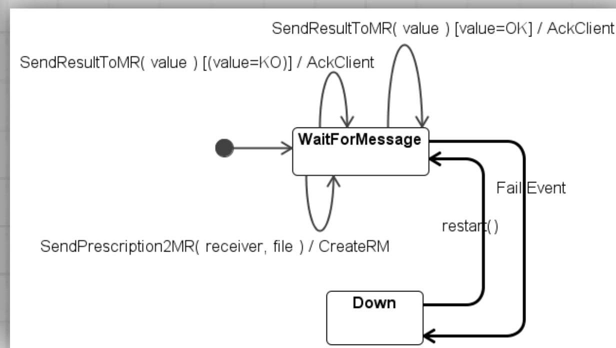
UML Profiles for NFPs at Work: UML Model

UML DESIGN MODEL (without annotation)

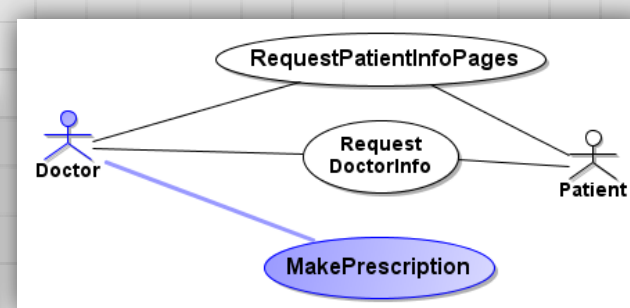
Component Diagram



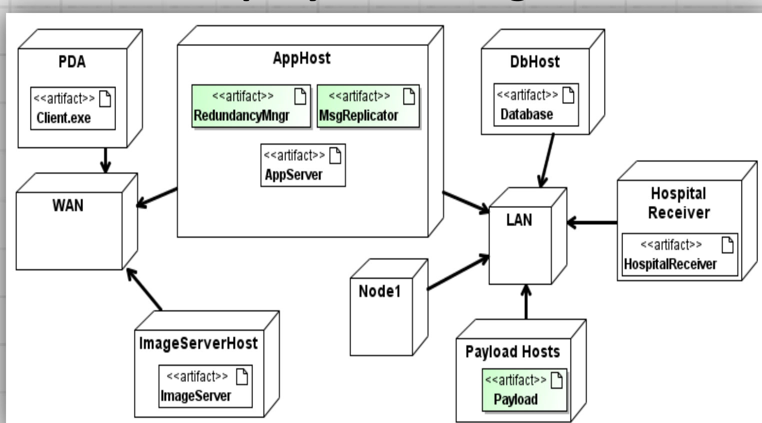
State Machine



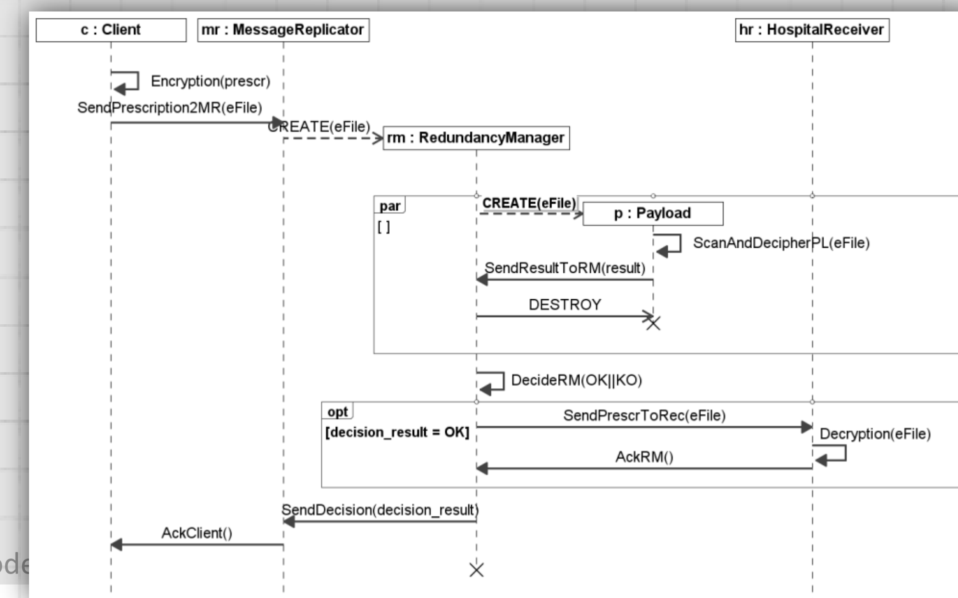
Use Case Diagram



Deployment Diagram



Sequence Diagram



UML Profiles for NFPs at Work: Reliability

1.Annotate→2.Transform→ 3.Solve

UML Design Model → **Reliability Target Formalism (Mathematical Expression)**

Model-Driven Anallysis Methodology

RELIABILITY ON DEMAND :

The probability of a system working within specifications for a certain number of invocations without system-level repair.

Vittorio Cortellessa, Harshinder Singh, and Bojan Cukic.

Early reliability assessment of UML based software models.

In *Workshop on Software and Performance, pages 302–309,* 2002.

Probability of System Failure =

$$\theta_S = 1 - \sum_{k=1}^{K} p_k \Big(\prod_{i=1}^{N} (1-\theta_i)^{bp_{ik}} \cdot \prod_{(i,j)} (1-\psi_{ij})^{interact(i,j,k)} \Big)$$

NFPs (i.e. UML annotations)

- θ_i = failure prob of **component** i
- bp_{ik} = busy periods (i.e. **activation**) of component i in scenario k
- $\psi_{i,j}$= failure prob of **hardware connector** among remote hosting nodes of components i and j
- $interact_{i,j,k}$ = **interactions** (# of invocations) between components i and j in scenario k
- p_k= execution probability of **scenario** k

It can be represented using a FAULT TREE

UML Profiles for NFPs at Work: Reliability

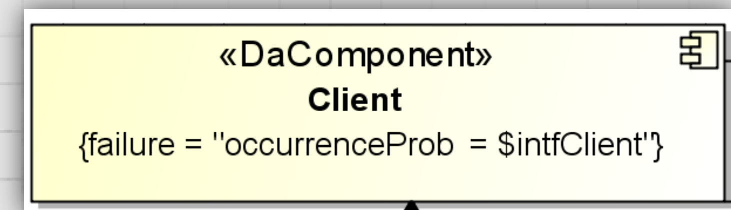
1.Annotate (MARTE+DAM)→2.Transform→ 3.Solve

UML DESIGN MODEL (with **Reliabiliy Annotation**)

DAM Profile

Component Diagram

θ_i = failure prob of component i

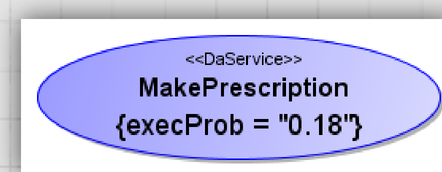


State Machine

DAM Profile

Use Case Diagram

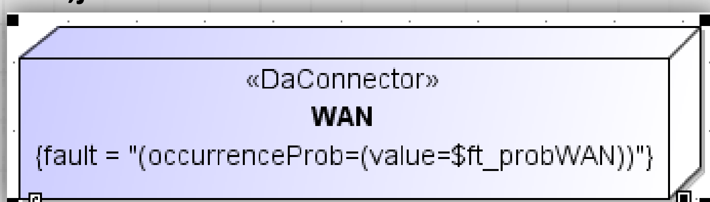
$\mathbf{p}_k$= execution probability of scenario k

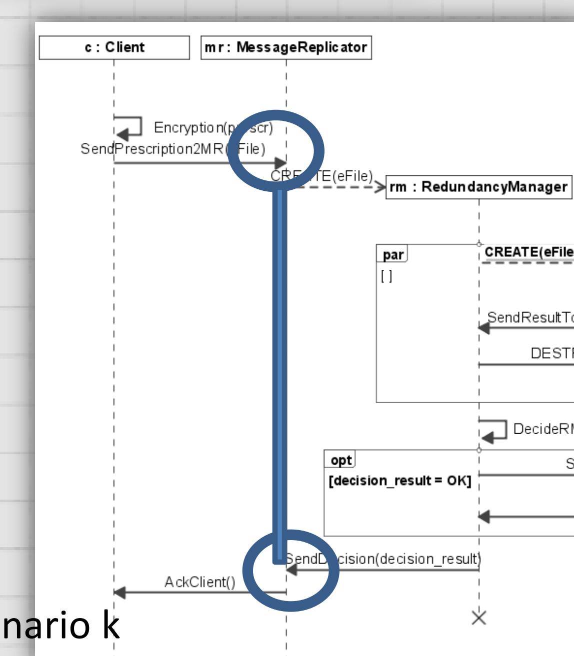


DAM Profile

Deployment Diagram

$\psi_{i,j}$= failure prob of hardware connector





UML Plain
Sequence Diagram

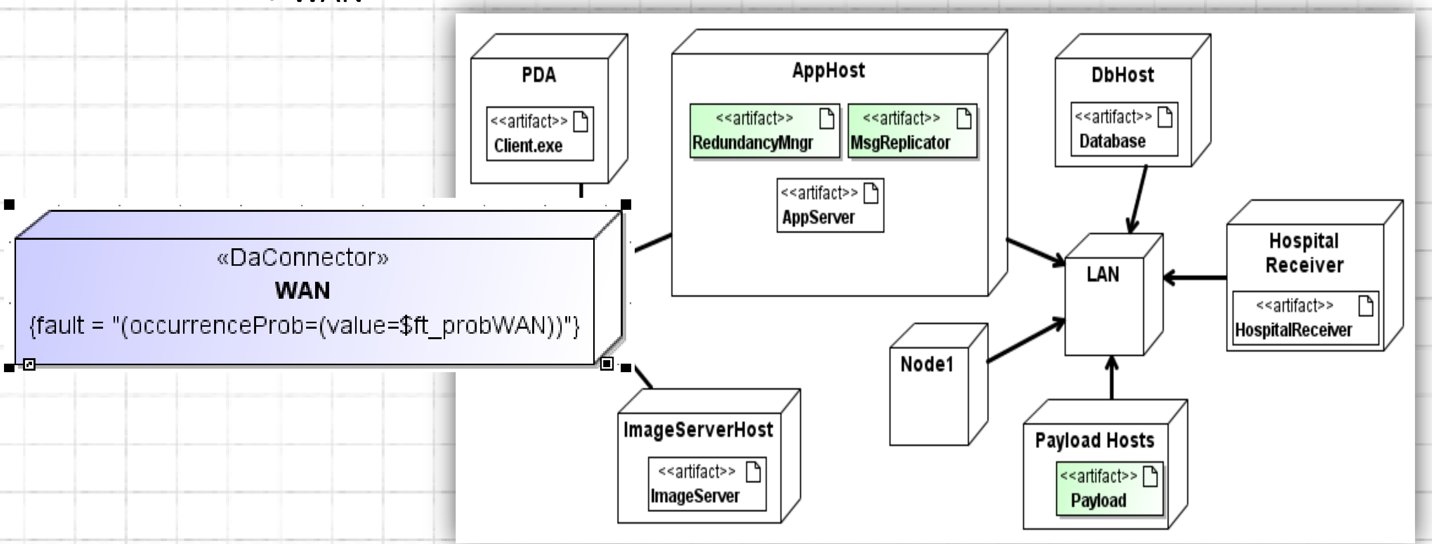
interact $_{i,j,k}$ = message exchanged between components i and j in scenario k

bp $_{i,k}$ = # of busy periods (i.e. activation) of component i in scenario k

UML Profiles for NFPs at Work: Reliability

UML Design Model — 1.Annotate→**2.Transform**→ **3.Solve** → **Reliability Target Formalism** (**Mathematical Expression**)

Sensitivity w.r.t. ψ_{WAN}= failure prob of **WAN connector**



$$(1-\theta_{Client})^2 \cdot (1-\theta_{MsgReplicator})^4 \cdot (1-\theta_{RedundancyMgr})^{f(N)} \cdot (1-\psi_{Client-MsgReplicator})^2$$

FIXED INPUT PARAMETER	$ft_probWAN	Make Prescription Reliability on Demand
$ft_prob**PDA** =0.001 $ft_prob**AppHost** =0.000001 $N = 3 (i.e multiplicity ofPayload Comps)	**0.01**	97.80%
	0.1	81.00%
	1	0.00%

UML Profiles for NFPs at Work: Availability

1.Annotate→2.Transform→ 3.Solve

UML Design Model → **Availability Target Formalism (Generalized Stochastic Petri-Net)**

Model-Driven Anallysis Methodology

AVAILABILITY:

The **probability** that the system is up and running to deliver its service to users when they request them

J. Merseguer, S. Bernardi, J. Campos, and S. Donatelli.

A compositional semantics for UML State Machines aimed at performance evaluation.

In Silva M., Giua A. and Colom J.M., editor, *WODES02: 6th International Workshop on Discrete Event Systems, pages 295–302,* Zaragoza, Spain, October 2002. IEEE Computer Society.

NFPs (i.e. UML annotations)

- Service availability requirement ;
- Software Failure (no fault masking mechanism) and Recovery rates
- Workload specification for service behaviors;
- Hardware Fault (propagates to sw failure) and Recovery rates of hardware nodes

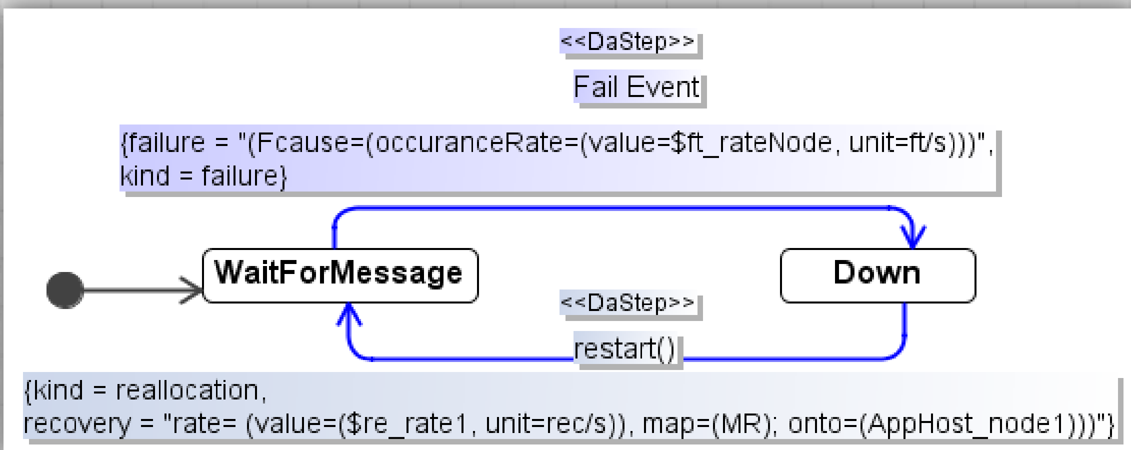
UML Profiles for NFPs at Work: Availability

1.Annotate (MARTE+DAM)→2.Transform→ 3.Solve

DESIGN MODEL (with **Availability Annotation**)

Component Diagram

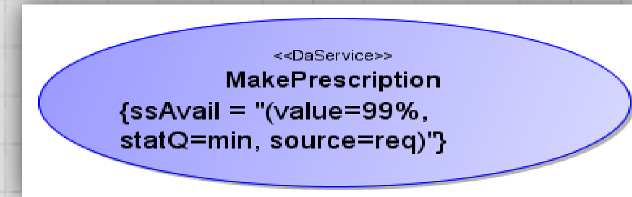
Message Replicator Component



DAM Profile

Use Case Diagram

Service availability requirement



DAM Profile

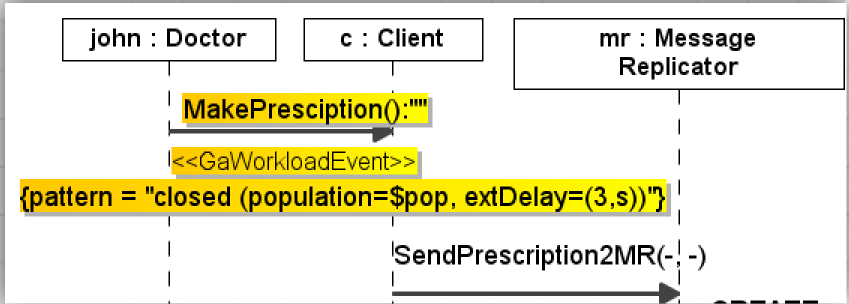
State Machine

Software Failure and Recovery rates of components

MARTE Profile

Sequence Diagram

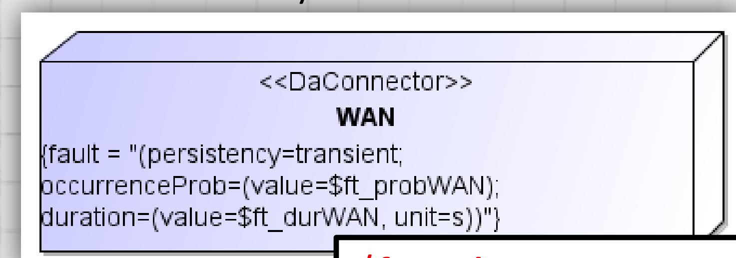
Workload specification of system services



DAM Profile

Deployment Diagram

Hw Fault and Recovery rates of execution and comm. hosts



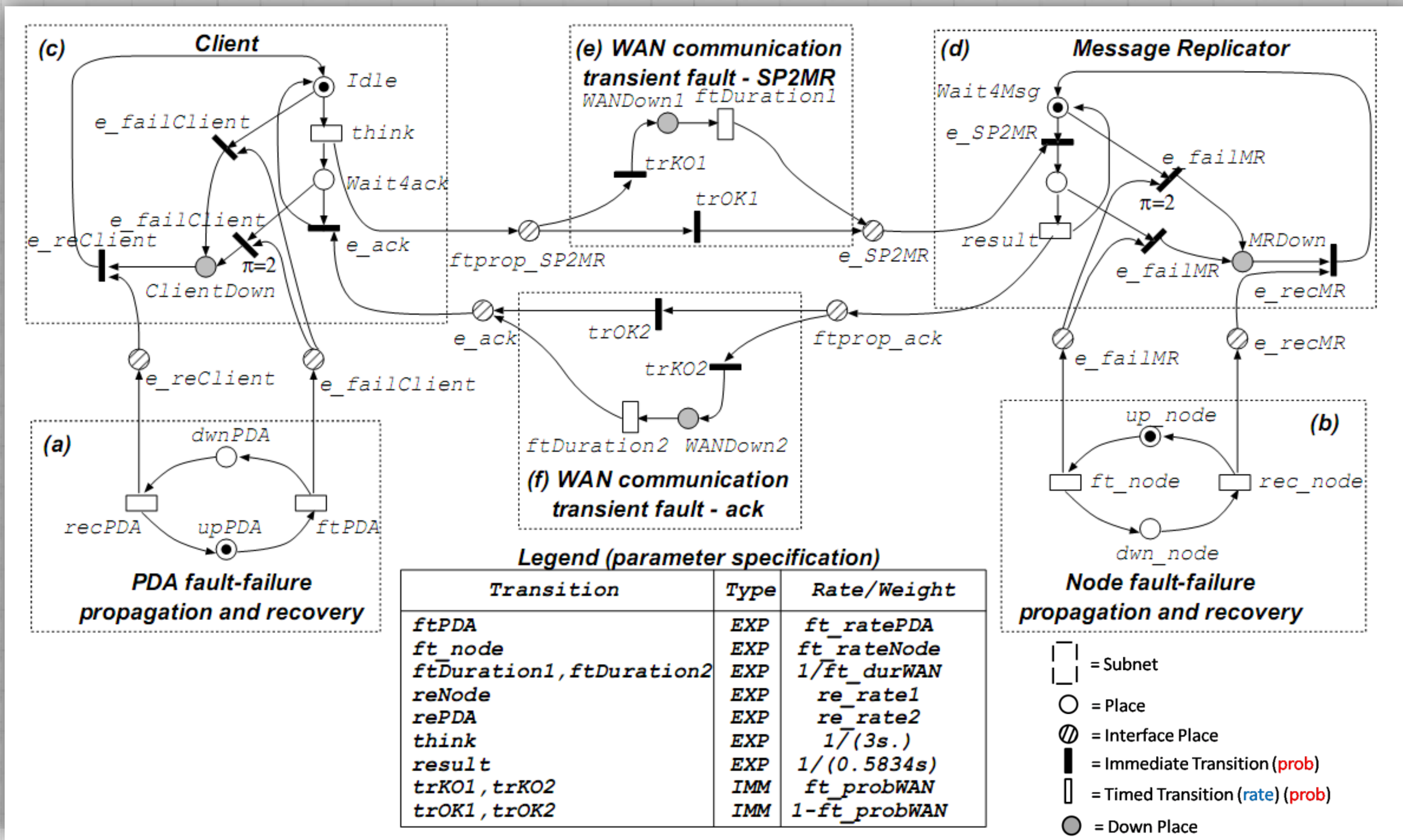
***$ft_probWAN* - INPUT Parameter sharing with *Reliability* Analysis**

UML Profiles for NFPs at Work: Availability

1.Annotate→**2.Transform**→ 3.Solve

ANALYSIS MODEL (gspn)

- Execution Host Failures (subnet) cause failure of deployed Component (subnet)



Legend (parameter specification)

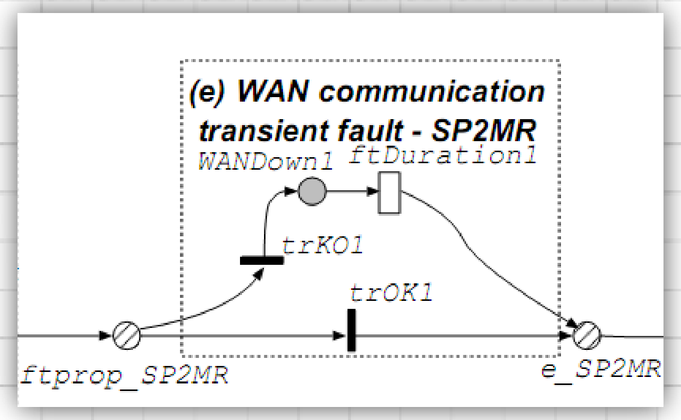
Transition	Type	Rate/Weight
ftPDA	EXP	ft_ratePDA
ft_node	EXP	ft_rateNode
ftDuration1,ftDuration2	EXP	1/ft_durWAN
reNode	EXP	re_rate1
rePDA	EXP	re_rate2
think	EXP	1/(3s.)
result	EXP	1/(0.5834s)
trKO1,trKO2	IMM	ft_probWAN
trOK1,trOK2	IMM	1-ft_probWAN

UML Profiles for NFPs at Work: Availability

UML Design Model — 1.Annotate→2.Transform→ **3.Solve** → **Reliability Target Formalism (GSPN Model)**

Sensitivity analysis w.r.t. fault rate probability of **WAN connector**

(no masking mechanism then hw fault→hw failure)



The **Service Availability** is defined as the

PROBABILITY THAT **ALL DOWN PLACES** MODELING THE DOWN STATES ARE **ALL EMPTY**

FIXED INPUT PARAMETER		$ft_probWAN	AVAIL
$ft_rate**PDA**=$ft_rate**Node**=	30 CRASH FAULT/ YEAR FOR **HW COMPS**	0.01	99.98%
$rec_**PDA**=$rec_**Node**=	60 SEC RECOVERY MEAN DURATION FOR PDA	0.1	99.88%
$ft_dur**WAN**=	1 HOUR OF WAN COMMUNICATION MEAN DOWN TIME	1	99.38%

UML Profiles for NFPs at Work: Performance

1.Annotate→2.Transform→ 3.Solve

UML Design Model → **Availability Target Formalism (Queuing Netowrk)**

Model-Driven Anallysis Methodology

PERFORMANCE:

The **probability** that the system is up and running to deliver its service to users when they request them

Antinisca Di Marco and Paola Inverardi.
Compositional Generation of Software Architecture Performance QN Models.
In *WICSA, pages 37–46, 2004.*

NFPs (i.e. UML annotations)

- Scheduling Policies of Software Components and Resource Demand (Time) for their Operations,
- Workload specification of service behaviors;

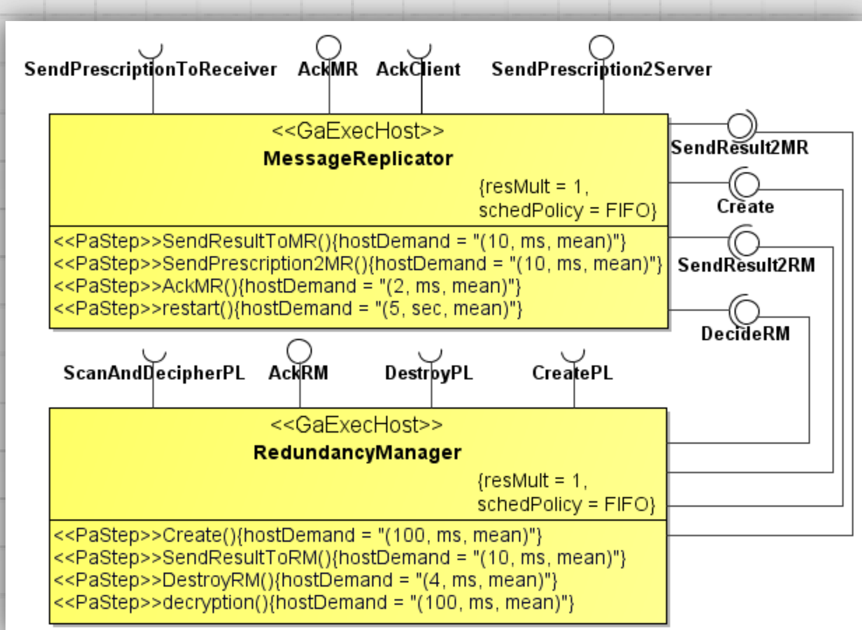
UML Profiles for NFPs at Work: Performance

1.Annotate (MARTE+DAM)→2.Transform→ 3.Solve

DESIGN MODEL (with **Availability Annotation**)

MARTE Profile

Component Diagram

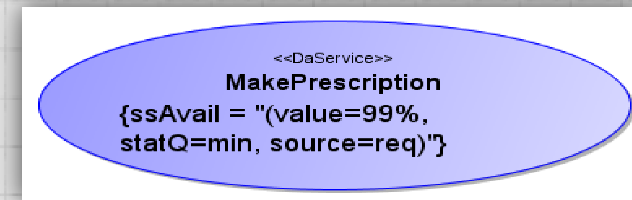


State Machine

DAM Profile

Use Case Diagram

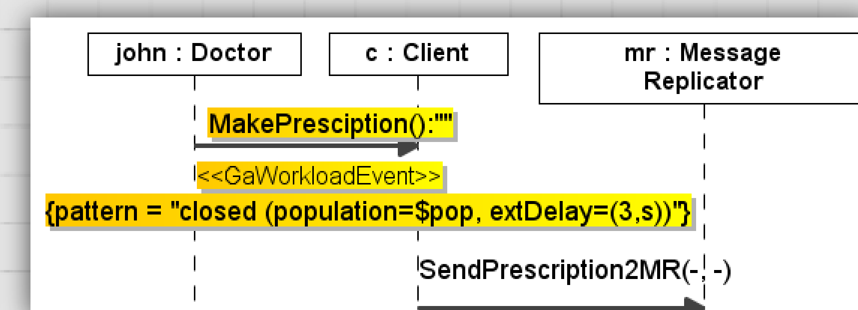
Service availability requirement



MARTE Profile

Sequence Diagram

Workload specification of system services



Deployment Diagram

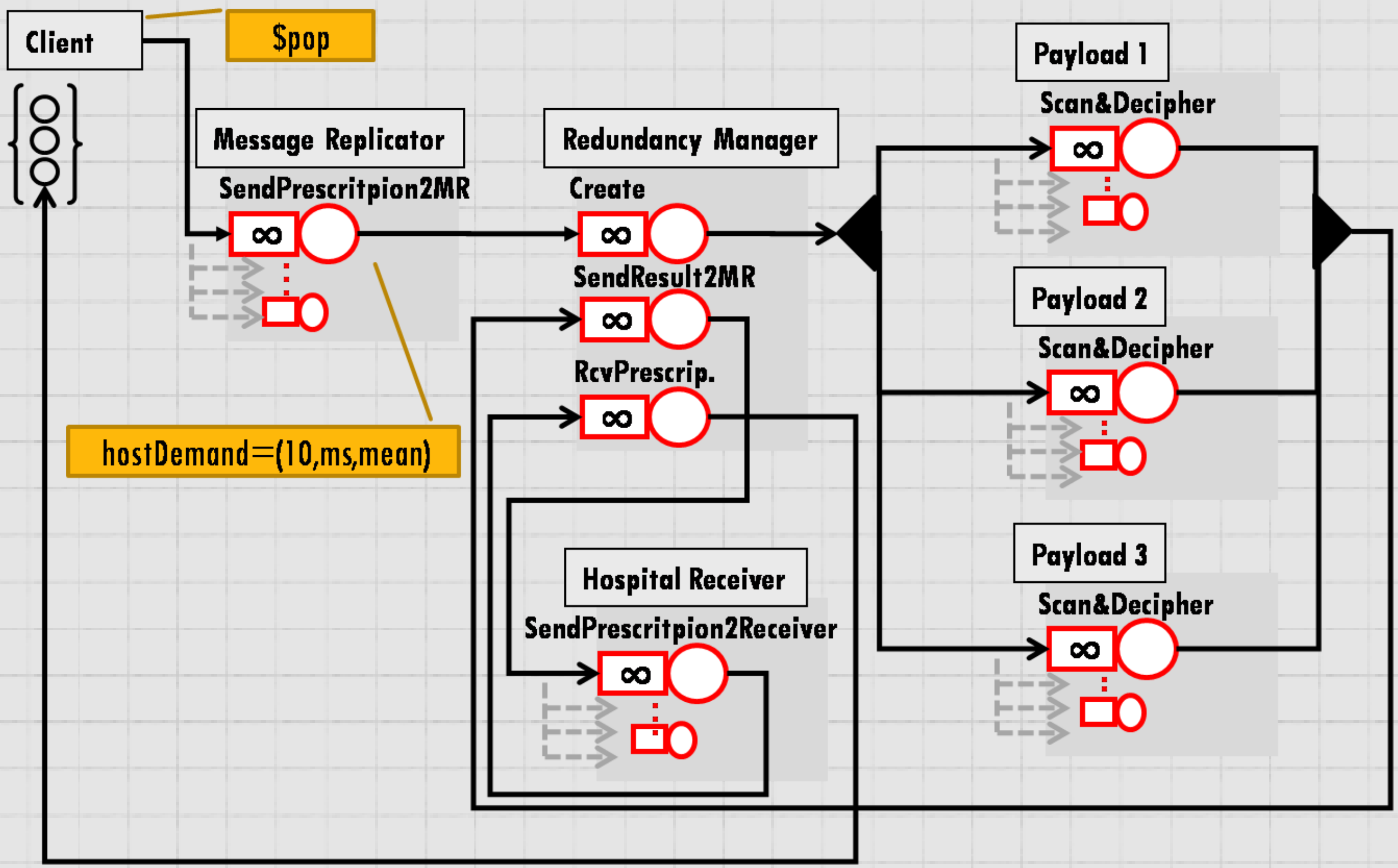
GaWorkloadEvent* - *INPUT Parameter sharing with Availability.Analysis

UML Profiles for NFPs at Work: Performance

1.Annotate→**2.Transform**→ 3.Solve

ANALYSIS MODEL (QN)

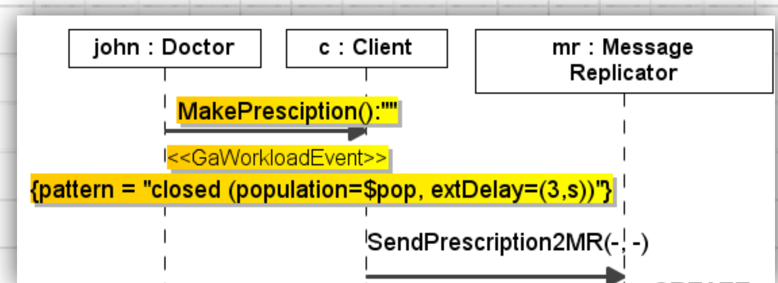
- Component → Multiqueue Service Center ; Component Operation → Queue

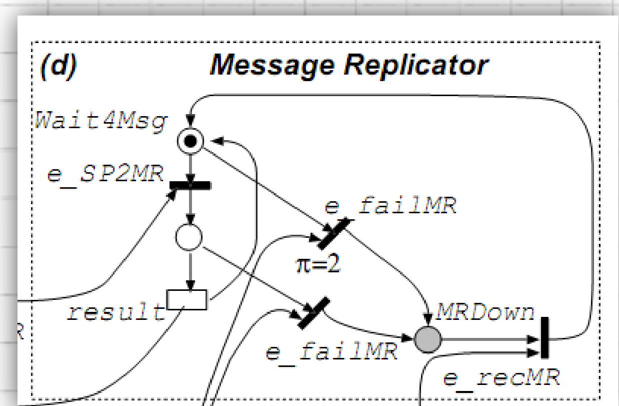


UML Profiles for NFPs at Work: Performance

UML Design Model → 1.Annotate→2.Transform→ **3.Solve** → **Performance Target Formalism (QN Model)**

Sensitivity analysis w.r.t. workload characteristics (e.g. population)





result	EXP	1/(0.5834s)

***INPUT/OUTPUT* Parameter sharing with *Availability*.Analysis**

Simulation (MVA Analysis not possible due to Fork&Join)

$pop	out$RT
1	**0.58333333**
2	0.61200189
3	0.64261899
4	0.67536272
5	0.71042945

$pop	out $RT
6	0.748034489
7	0.789521871
8	0.832765210
9	0.879266880
10	0.930278346

UML Profiles for NFPs at Work: conclusion

1. Studyng of logic, models and languages for:
   a) Modeling performability-aware systems:
      - **Input**: UML Design Model+Profiles. **Output:** GSPN, QN, FT
      - Integration of different NFPs by I/O paramter sharing: Dependability (Reliability, Availability), Performance

   b) Specification of performability metrics: Not considered
2. Definition of model transformation functions
   a) From design to analysis models (direct):
      - **Unique Source** Design Model (UML), **Multiple Target** Analysis Model
      - **Methodologies** and **Tools**

   b) Among different analysis models (direct/inverse);

   c) From analysis to design models (inverse);

L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, **(NFPinDSML 2009 @ MODELS 2009)**
"UML Profiles for Non-Functional Properties at Work: Analyzing Reliability, Availability and Performance",
Proc. of the 2nd International Workshop on Non-functional System Properties in Domain Specific Modeling Languages

The FEF Chain: M&A Performability in UML SA

1. Studyng of logic, models and languages for:
   a) Modeling performability-aware systems:
      - **Input**: UML Design Model+Profiles+FEF PAttern. **Output:** GSPN, ~~QN, FT~~
      - Integration of different NFPs by I/O paramter sharing: Dependability (Reliability, ~~Availability~~), Performance

   b) Specification of performability metrics: **Mean Service Time** of System subject to working/repairing/working cycles
2. Definition of model transformation functions
   a) From design to analysis models (direct):
      - **Unique Source** Design Model (UML), **Unique Target** Analysis Model
      - **Methodologies** and **Tools**

   b) Among different analysis models (direct/inverse);
   c) From analysis to design models (inverse);

L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, (**QoSA 2010**)
" The Fault-error-failure Chain: a Challenge for Modeling and Analyzing Performability in UML-based Software Architectures ", Submitted to the Sixth International Conference on the Quality of Software Architectures (QoSA 2010)

The FEF Chain: M&A Performability in UML SA

General Modeling & Analysis Process for quantitative analysis of SA :

1. Annotate a software model with appropriate performability parameters (UML Model + stereotypes)
2. Transform the software model in a performability model (UML2GSPN)
3. Solve the performability model (GSPN) and get the result

0. Define a Failure Model to be applied (reliability)

The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→2.Transform→ 3.Solve

UML Design Model → **Performability Target Formalism (Generalilzed Stochastic Petri-Net)**

Model-Driven Anallysis Methodology

PERFORMABILITY:

The **probability** that the system is up and running to deliver its service to users when they request them

L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer,

" The Fault-error-failure Chain: a Challenge for Modeling and Analyzing Performability in UML-based Software Architectures ",

Submitted to the Sixth International Conference on the Quality of Software Architectures (QoSA 2010)

NFPs (i.e. UML annotations)

- Failure Model
- Workload specification of service behaviors;
- Software Fault, Error, Failure probabilities and Error Detection/Isolation duration

The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→2.Transform→ 3.Solve

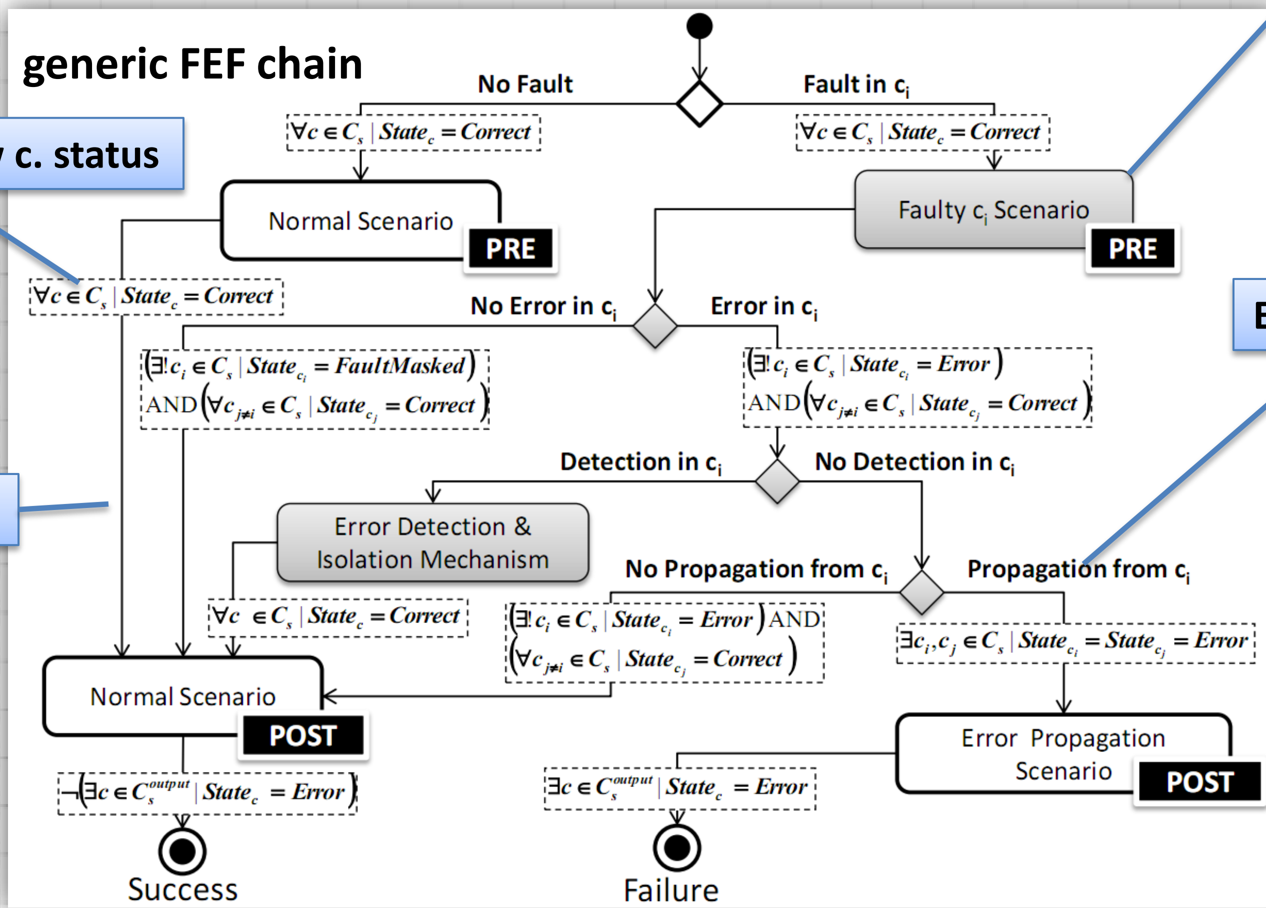
- Type of fault: Single Failure Mode (info content failure)
- Modeling System Behavior: generic FEF chain

Execution fragments

Predicates on sw c. status

Events

Control flow

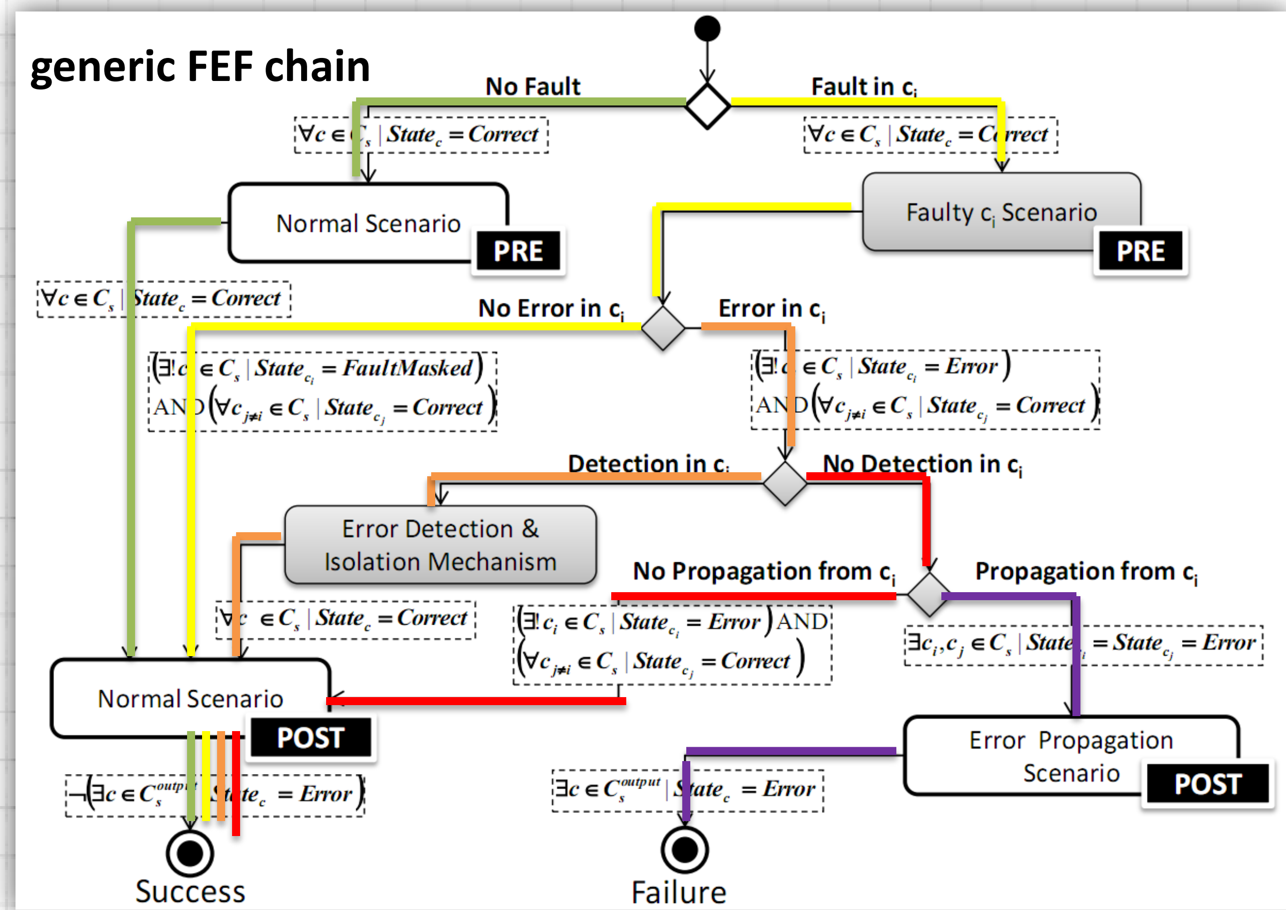


The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→2.Transform→ 3.Solve

Correct path | Error detected and isolated | **Failure**

Fault M asked | Error NOT detected BUT NOT Propagated



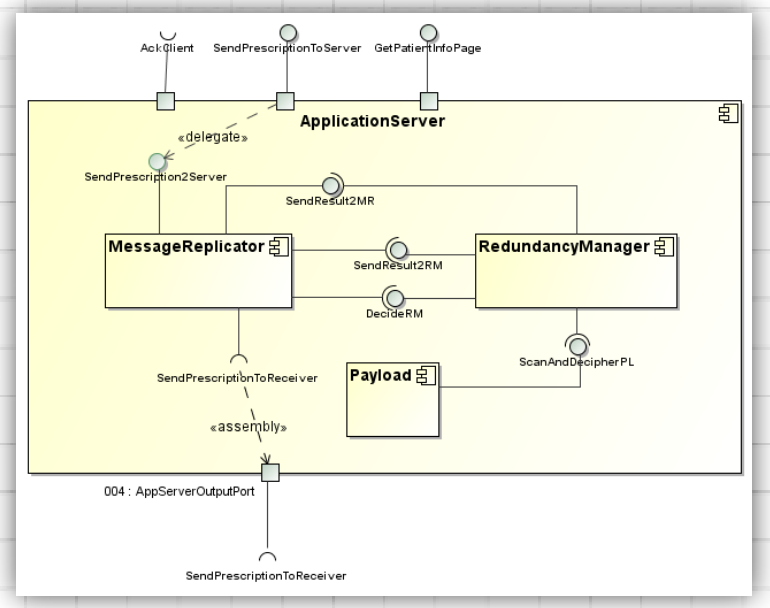
The FEF Chain: M&A Performability in UML SA

A doctor provided with his/her generic computing device (e.g. PDA, laptop), through **medical distributed services**, is able to:

Make Prescription : after visiting the patient, the doctor can make a prescription to be sent to the hospital where eventually the patient will take the medicines.

This service requires some particular **Availability**, **Reliability** and **Performance** requirements

The eHealth System is equipped with Message Redundancy Systems:



Redundancy Manager:

(i) create a Message Replicator for each prescription sent to the Application Server

Message Replicator:

(i) replicates n-times the prescription message,

(ii) create and destroy Payloads

(iii) assign each replica to a different Payload

(iv) calculate the voting result.

Payload:

(i) scan & decipher message replicas and

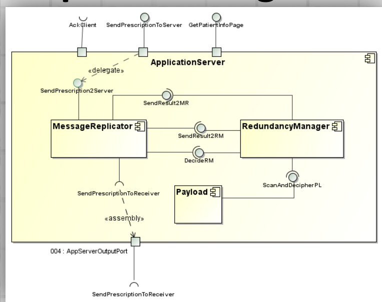
(ii) votes for replica integrity.

The FEF Chain: M&A Performability in UML SA

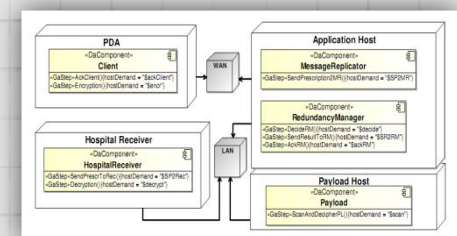
0.Failure Model→**1.Annotate**→2.Transform→ 3.Solve

UML DESIGN MODEL

Component Diagram



Deployment Diagram



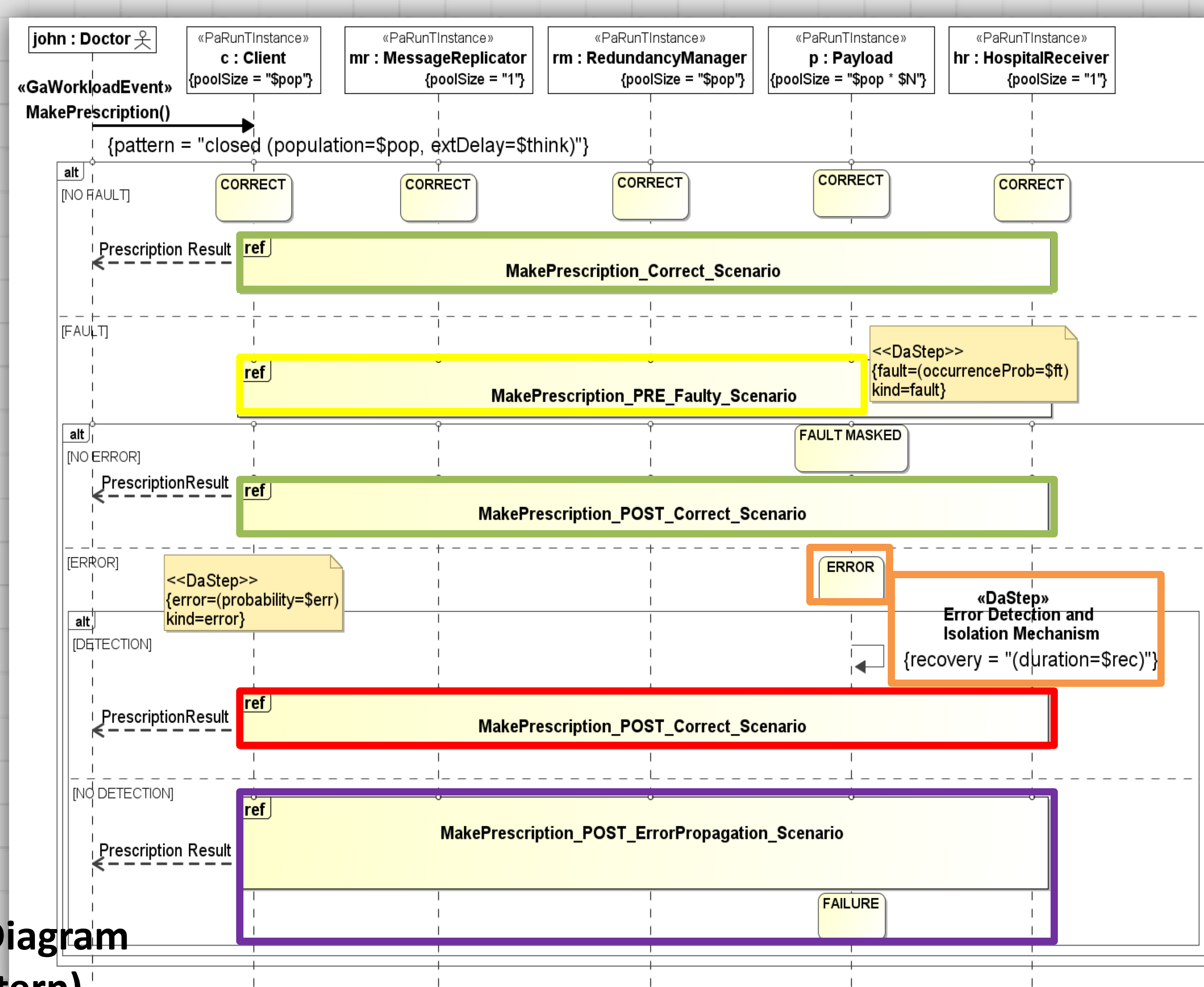
Correct path

Fault M asked

Error detected and isolated

Error NOT detected BUT NOT Propagated

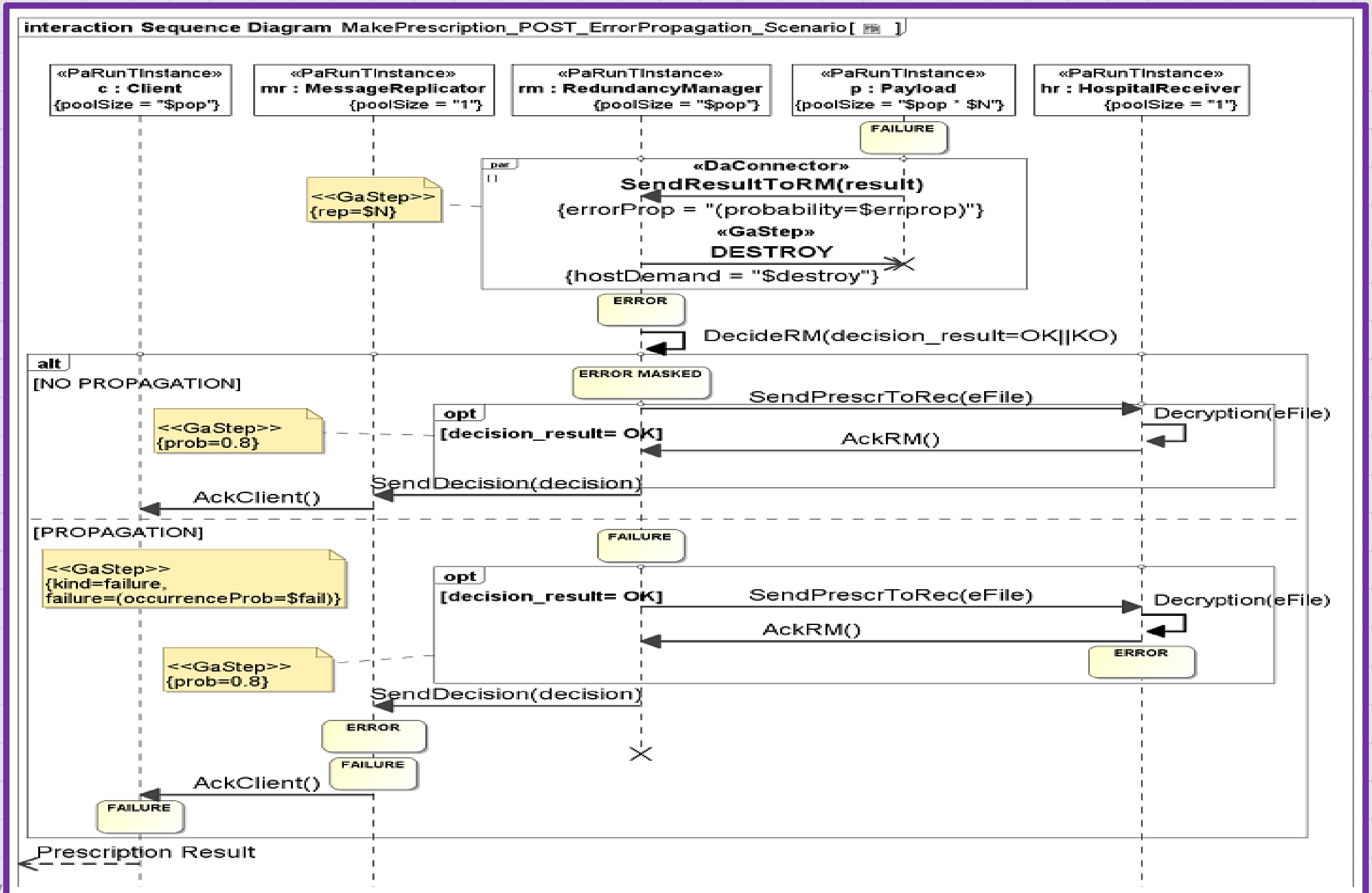
Failure



Sequence Diagram (FEF pattern)

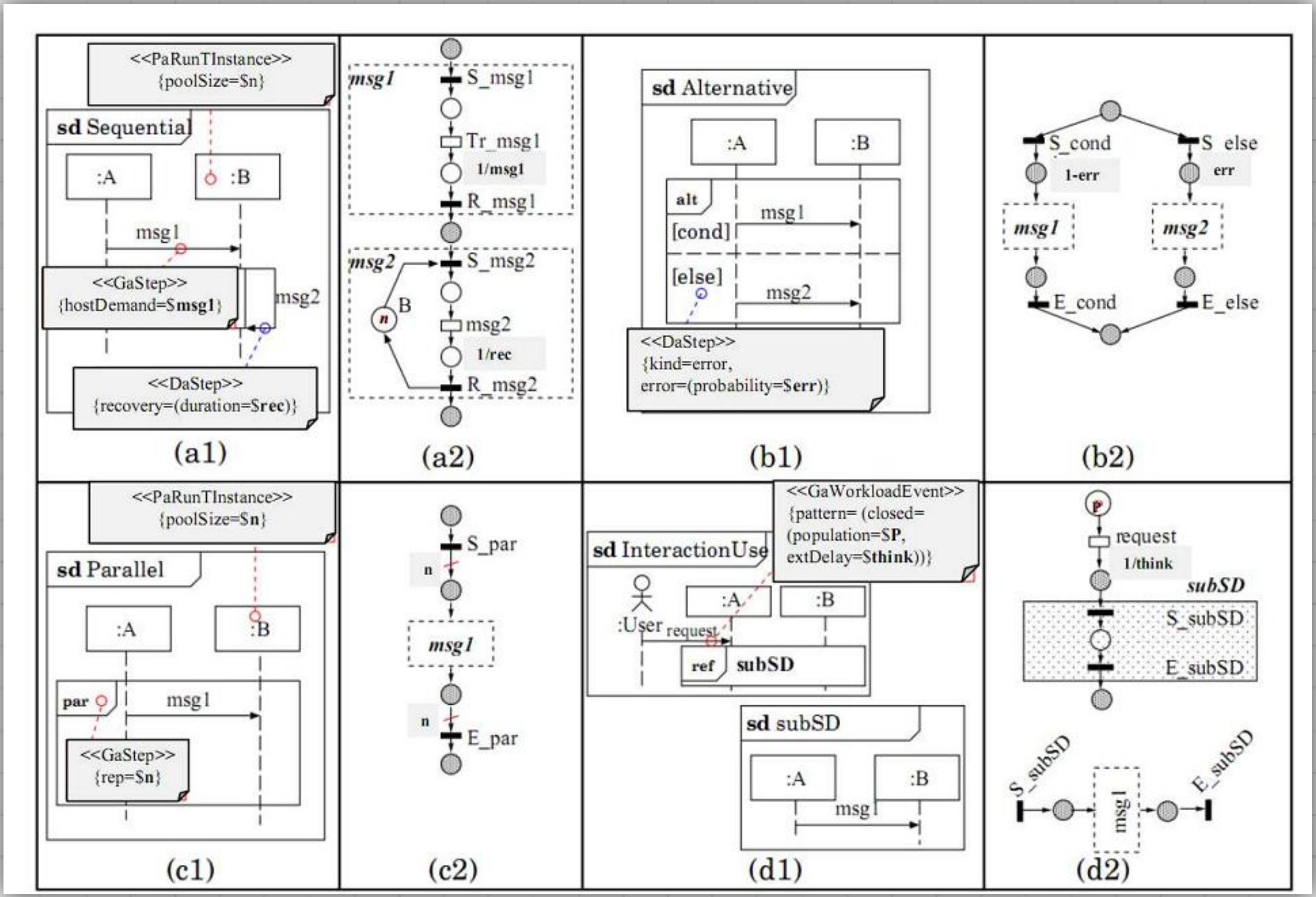
The FEF Chain: M&A Performability in UML SA

0.Failure Model→**1.Annotate**→2.Transform→ 3.Solve



The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→**2.Transform**→ 3.Solve



Workload specification of service behaviors;
Software Fault, Error, Failure probabilities and Error Detection/Isolation duration

The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→**2.Transform**→ 3.Solve

Correct path

Fault M asked

Error detected and isolated

Error NOT detected BUT NOT Propagated

Failure

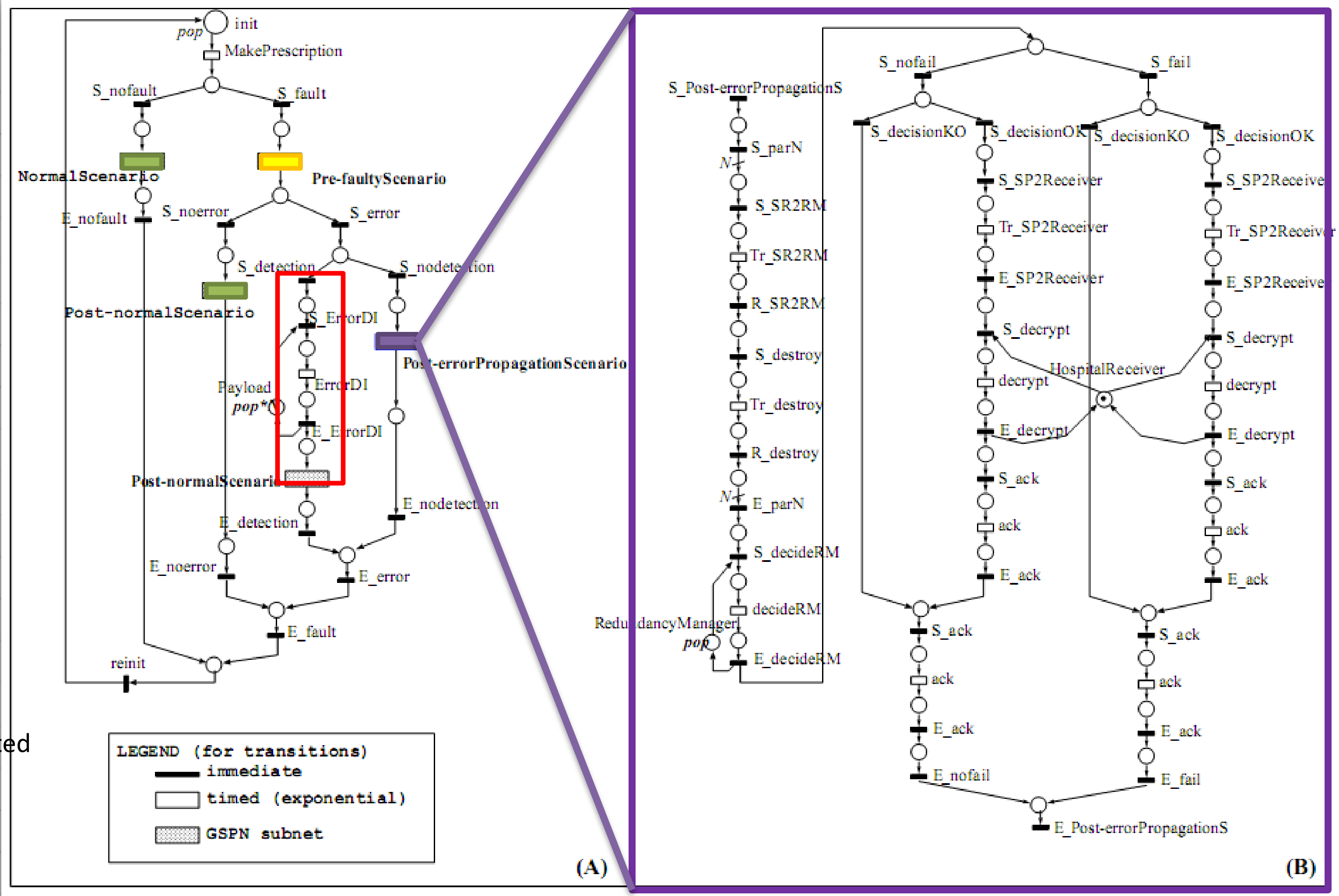


Fig. 7. GSPN model: main scenario (A), post-error propagation scenario (B)

The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→2.Transform→ **3.Solve**

UML Design Model → **Performability Target Formalism (Generalilzed Stochastic Petri-Net)**

Model-Driven Anallysis Methodology

$$MST = \frac{pop}{(X\_reinit - X\_S\_fail)} - think$$

Mean time to process correctly the makeprescription request (MST) under payload fault assumption (single/content failure model)

- **pop** = population (i.e. # of doctor that concurrently requires prescriptions)
- **X_reinit** = **Throughput** of **transition re_init** (termination of main scenario, see GSPN)
- **X_S_fail** = **Throughput** of **transition S_fail** (error propagation from RM to MR, see ErrorProp SD):
- **think** = external delay

Parameter Name	UML Notation	Dia	Var	Value/[Range]	Type
fault probability	DAM::DaStep.fault	SD	$ft	[0.1-90]	%
error probability	DAM::DaStep.error	SD	$err	[0.1-90]	%
error prop. P RM	DAM::DaConnector.errorProp	SD	$errprop	[0.1-90]	%
error prop. RM-MR (binomial)	DAM::DaStep.failure	SD	$fail	[3E-18; 8.19E-1]	%

The FEF Chain: M&A Performability in UML SA

0.Failure Model→1.Annotate→2.Transform→ **3.Solve**

UML Design Model → **Performability Target Formalism (Generalilzed Stochastic Petri-Net)**

Model-Driven Anallysis Methodology

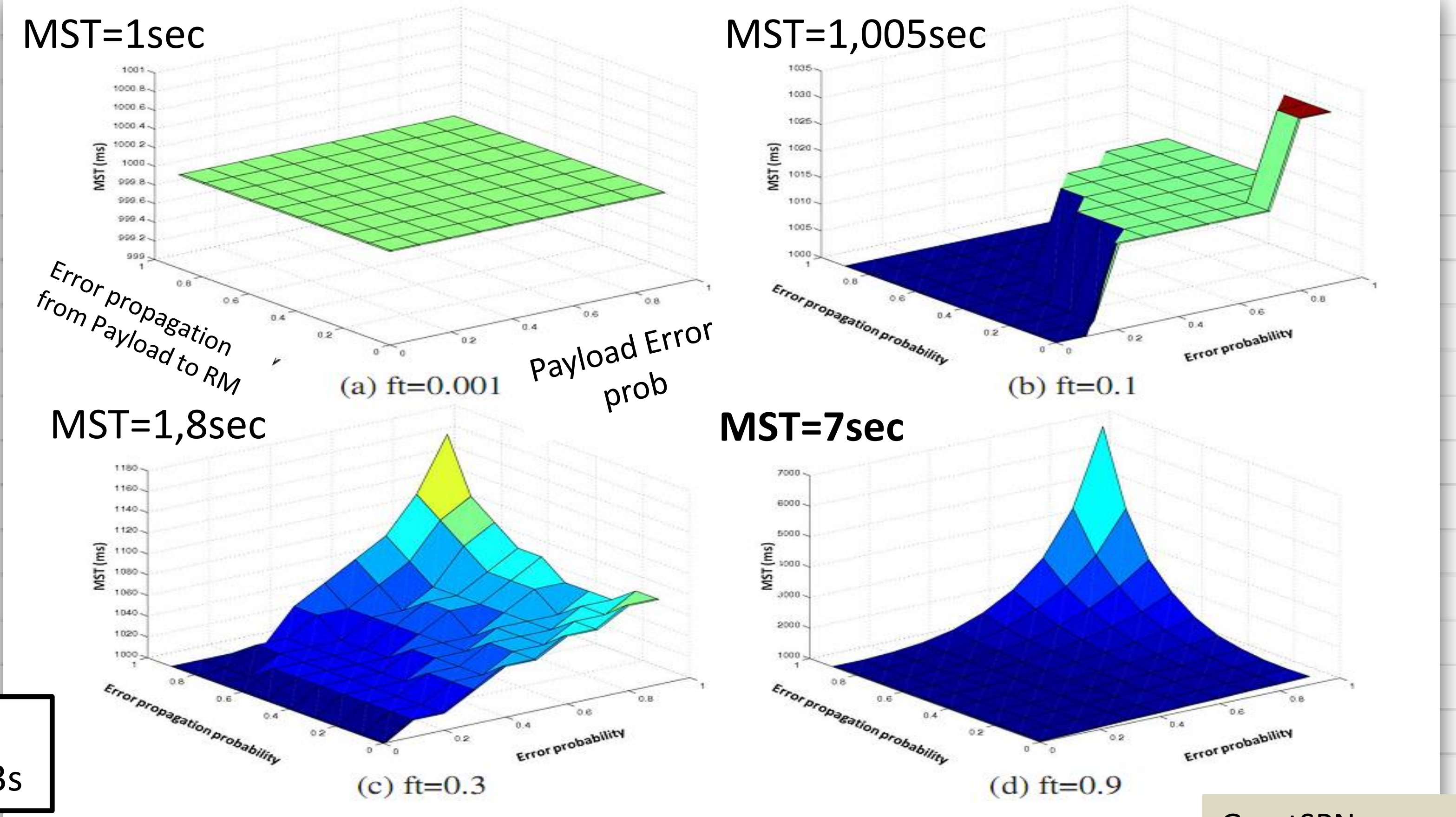


Fig. 8. Mean service time under different fault assumptions.

pop = 1
think = 3s

GreatSPN
(Markovian solver)

The FEF Chain: M&A Performability in UML SA

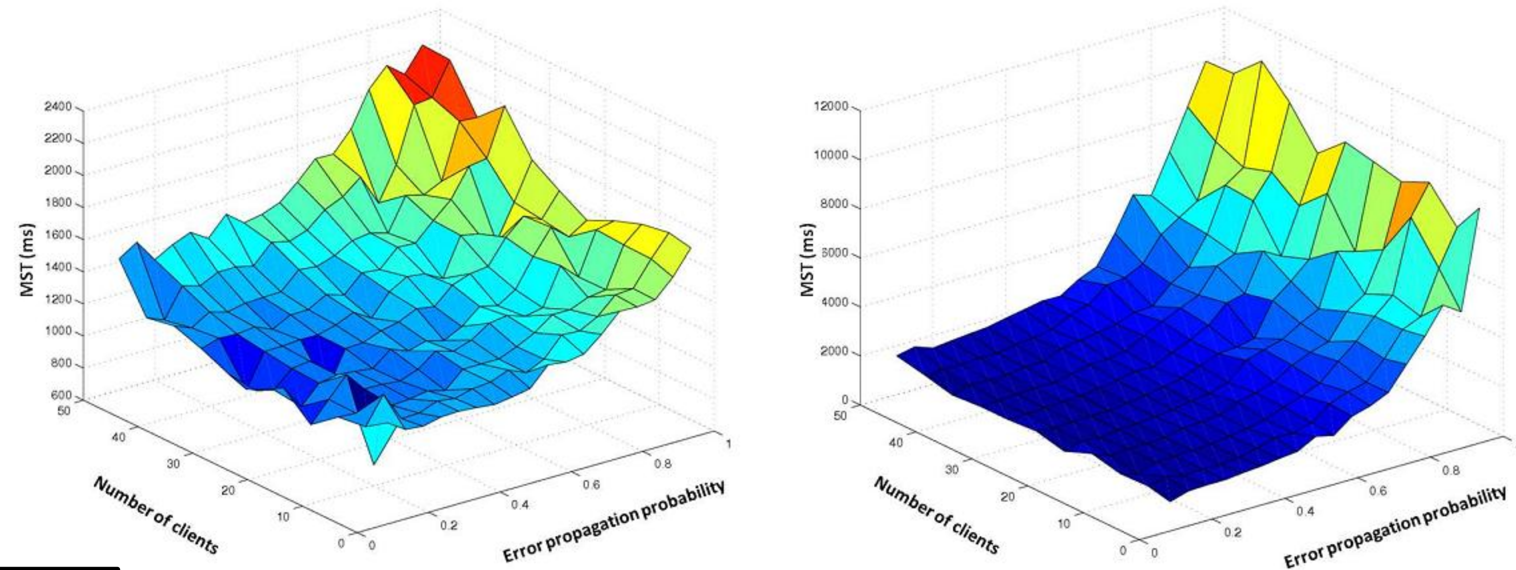
0.Failure Model→1.Annotate→2.Transform→ **3.Solve**

UML Design Model → **Performability Target Formalism (Generalilzed Stochastic Petri-Net)**

Model-Driven Anallysis Methodology

MST=2,19sec

MST=10.19sec



(e) ft=0.5, err=0.9

(f) ft=0.9, err=0.9

Fig. 9. Mean service time under different fault and workload assumptions.

pop = [1,50]
think = 3s

GreatSPN
(Simulator: accuracy= 15% ,confidence interval 99%)

The FEF Chain: Conclusion

Future works

(i) Experiment the approach on different mechanisms for error masking that take very different time to be executed.

(ii) Extend the Failure Model to multiple failures and/or different type of failures. Consequently the UML pattern should change.

L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, (**QoSA 2010**)
" The Fault-error-failure Chain: a Challenge for Modeling and Analyzing Performability in UML-based Software Architectures ", Submitted to the Sixth International Conference on the Quality of Software Architectures (QoSA 2010)