Integrating Fault-Tolerant Techniques into the Design of Critical Systems

Ricardo J. Rodríguez and José Merseguer
{rjrodriguez, jmerse}@unizar.es

Universidad de Zaragoza
Zaragoza, Spain

23rd June 2010

ISARCS'10: 1st International Symposium on Architecting Critical Systems
Prague, Czech Republic

Introduction (I)

Software failures

- Environmental damage (*safety-critical system*, e.g. nuclear station)
- Non-achieved goal (*mission-critical system*, e.g. Mars missions)
- Financial losses (*business-critical system*, e.g. Amazon website)

Means to attain dependability

- Fault prevention and fault tolerance
- Considered by designers of critical systems
  - e.g., quality control techniques, replication...

We want to...

- Evaluate a given FT technique for a concrete software design
- UML + well-defined interfaces $\rightarrow$ analysable model

Introduction (II)

UML profiles

- Free the software engineer from manual generation of formal model
- Capture dependability, performance and security properties in UML designs
  - Security Analysis and Modelling (SecAM) profile

Main idea summary

- UML model elements annotated with above mentioned profiles
  - Dependability, performance and security requirements captured
- Analysable model (e.g., Petri net) to report results

Proactive and reactive techniques (I)

FT techniques

- Robust system against faults (fault prevention and fault tolerance)
- Proactive and reactive fault recovery techniques
- Proactive:
  - long-term protection against break-ins
  - Periodic refreshments and distribution
- Reactive:
  - concurrent error detection

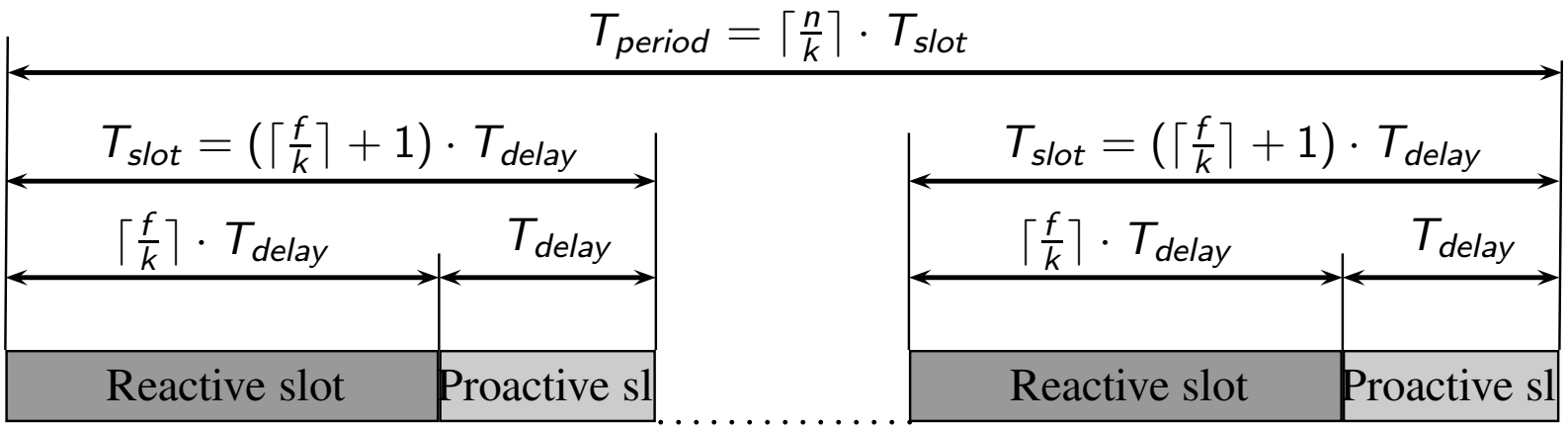
Proactive and reactive techniques (II)

Complementary techniques (not mutually exclusive)

- Proactive → fault prevention (passive part)
- Reactive → fault removal (active part)
- n devices; up to f failure devices; recovering in parallel up to k devices

$$n \geq 2 \cdot f + k + 1$$

- Sousa, P. et al. Resilient Intrusion Tolerance through Proactive and Reactive Recovery. *Procs. of the 13th IEEE Pacific Rim Dependable Computing Conf.*, pp. 373–380, 2007

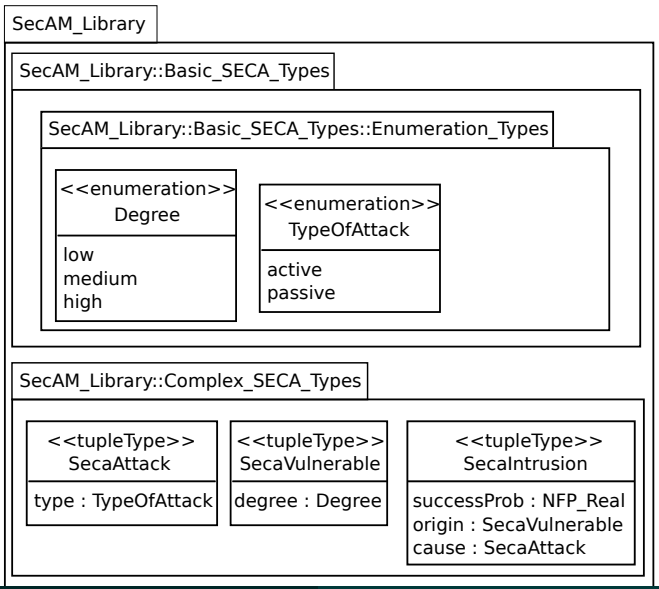


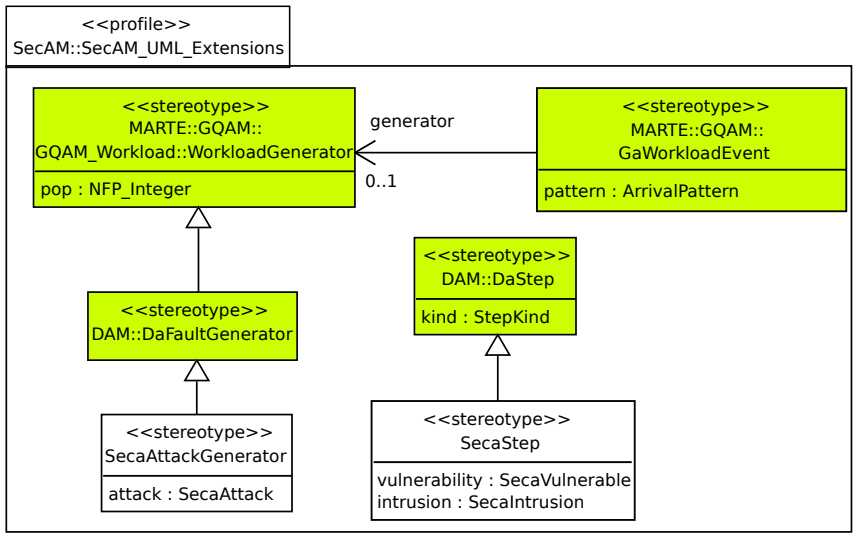
The SecAM profile (I): UML profiles

- UML tailored for specific purpouses: *profiles*
- Set of stereotypes and tagged values for annotating design model elements
- Modelling and Analysis of RT and Embedded systems (MARTE) profile
  - Schedulability and performance analysis
- Dependability Analysis and Modelling (DAM) profile
  - MARTE specialisation
  - Dependability as NFP in UML models

The SecAM profile (II): Security Analysis and Modelling

- Integrated in MARTE-DAM framework
- Model and analyse security properties
- Currently just addresses resilience topic
- Rodríguez, R.J. et al. Modelling and Analysing Resilience as a Security Issue within UML. *Procs. of the 2nd Int. Workshop on Soft. Eng. for Resilient Systems* (SERENE), 2010.



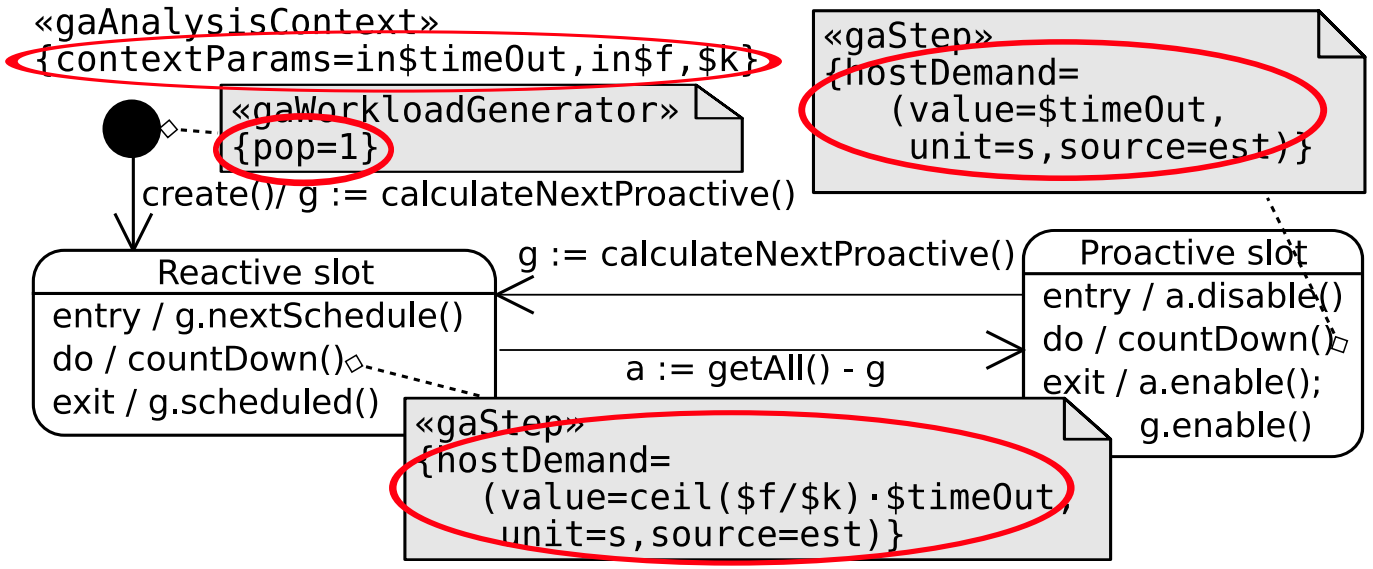


UML modelling (I): from UML models to GSPNs

- UML state-machine diagrams annotated with described profiles
- GSPN obtained by model transformation (ArgoSPE tool)
- ArgoSPE tool: annotated UML-SC as input data
- Working out the PN to incorporate DAM and SecAM annotations
- Coloured Petri Nets (CPN) $\rightarrow$ hierarchy and symmetries in the problem

UML modelling (II): Proactive and reactive techniques

Scheduler UML-SC diagram

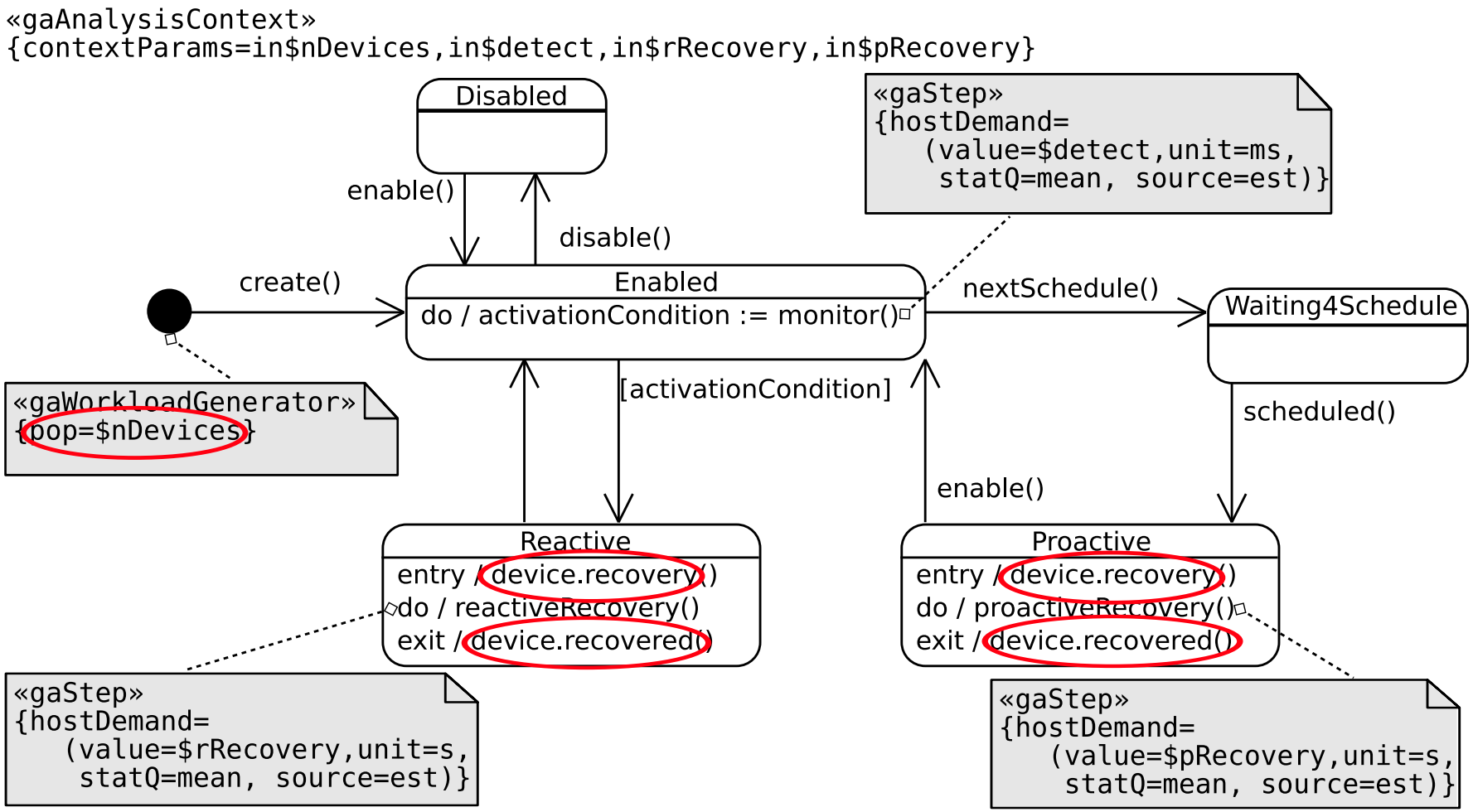


Stereotypes from MARTE

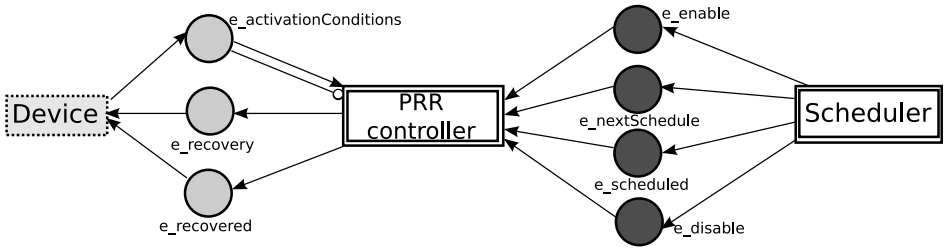
- Initial analysis variables: `gaAnalysisContext` stereotype
  - Duration of recoveries (`timeOut`), faulty devices (`f`), recoveries in parallel (`k`) (recall: $n \geq 2 \cdot f + k + 1$)
- Population: tag `pop` of `gaWorkloadGenerator` stereotype
- Duration of activities: tag `hostDemand` of `gaStep` stereotype

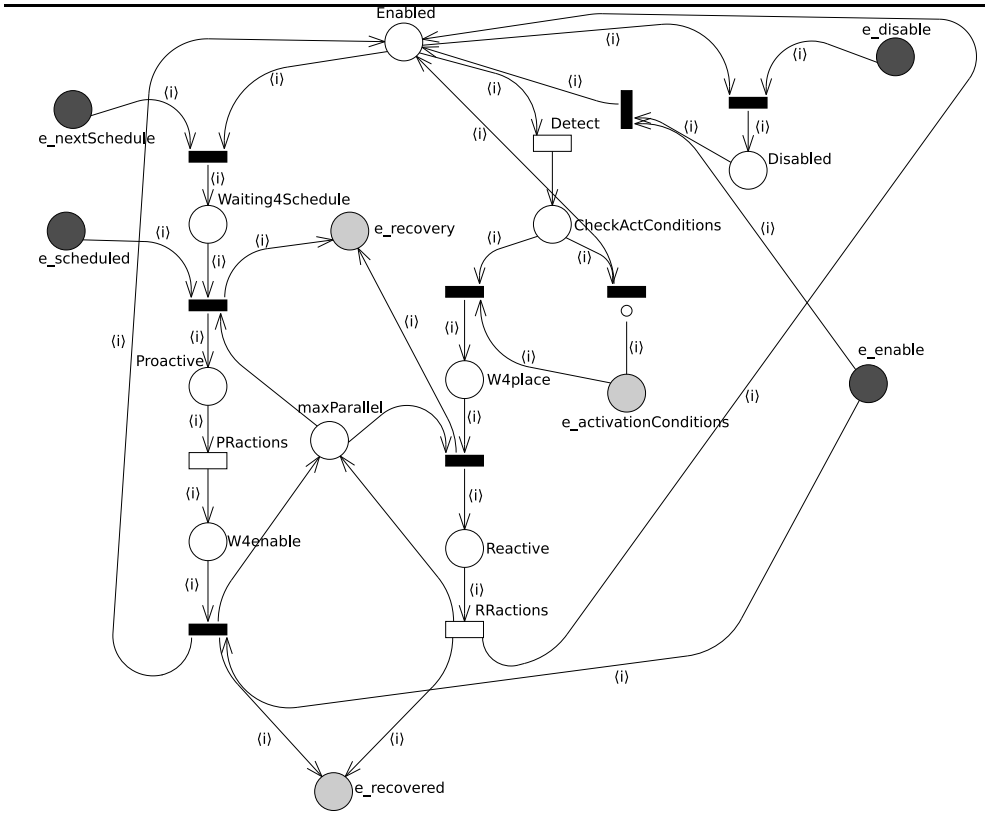
UML modelling (II): Proactive and reactive techniques

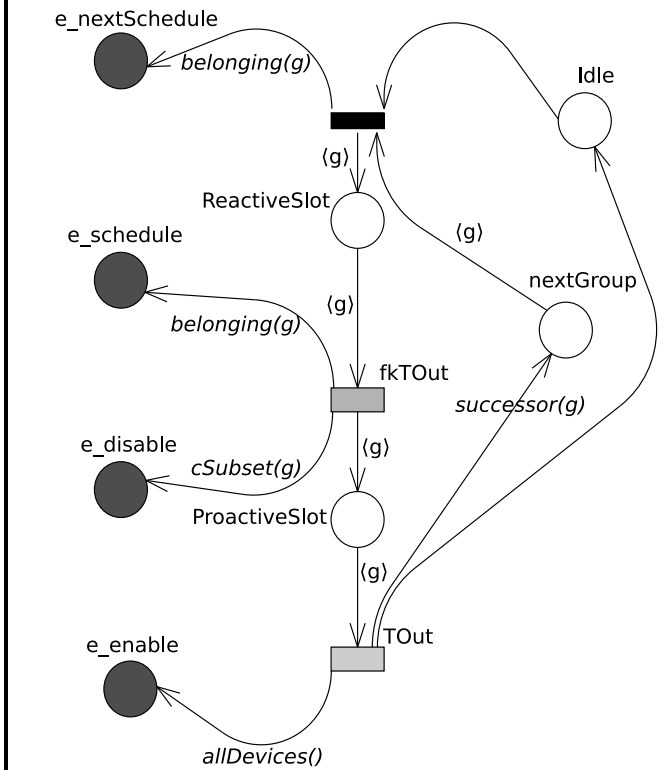
Proactive and reactive recovery controller UML-SC diagram



Formal modelling through Petri Nets (I)



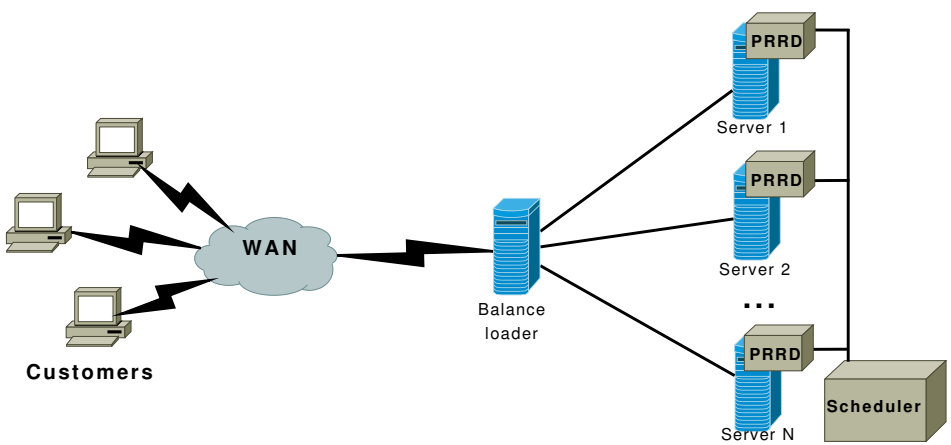




Example: definition

Blueprint about how to

- Add proactive and reactive techniques → improving fault tolerance
- Get an analysable formal model
- Obtain results from such model



Business-critical system

- On-line shopping website
- Balance loader
- Servers attending customers
  - Up to k in parallel
- Scheduler + external PRRDs

Example: UML modelling (I)

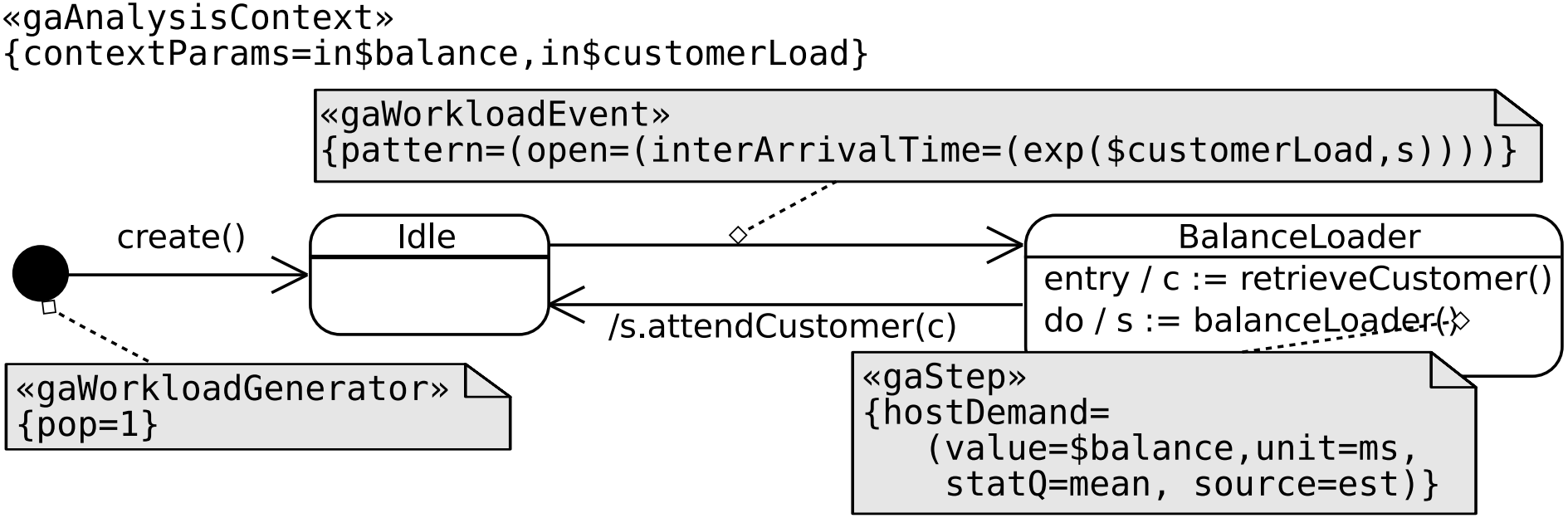


Figure: Balance loader UML state-machine diagram.

Example: UML modelling (II)

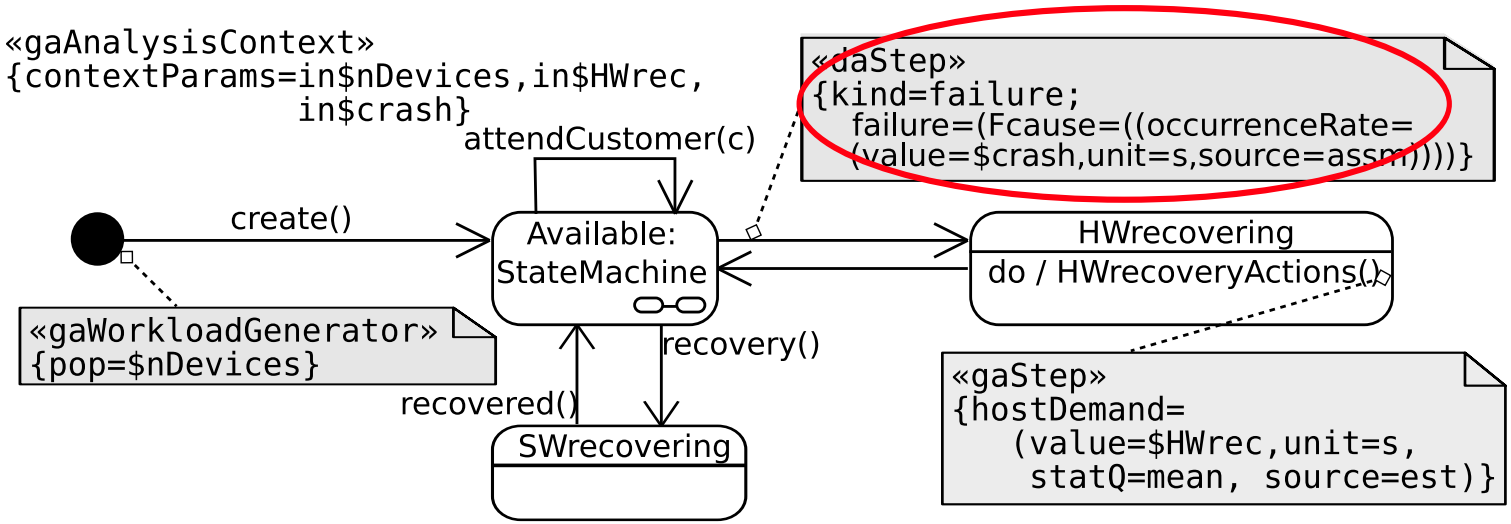


Figure: Server UML state-machine diagram.

Example: UML modelling (III)

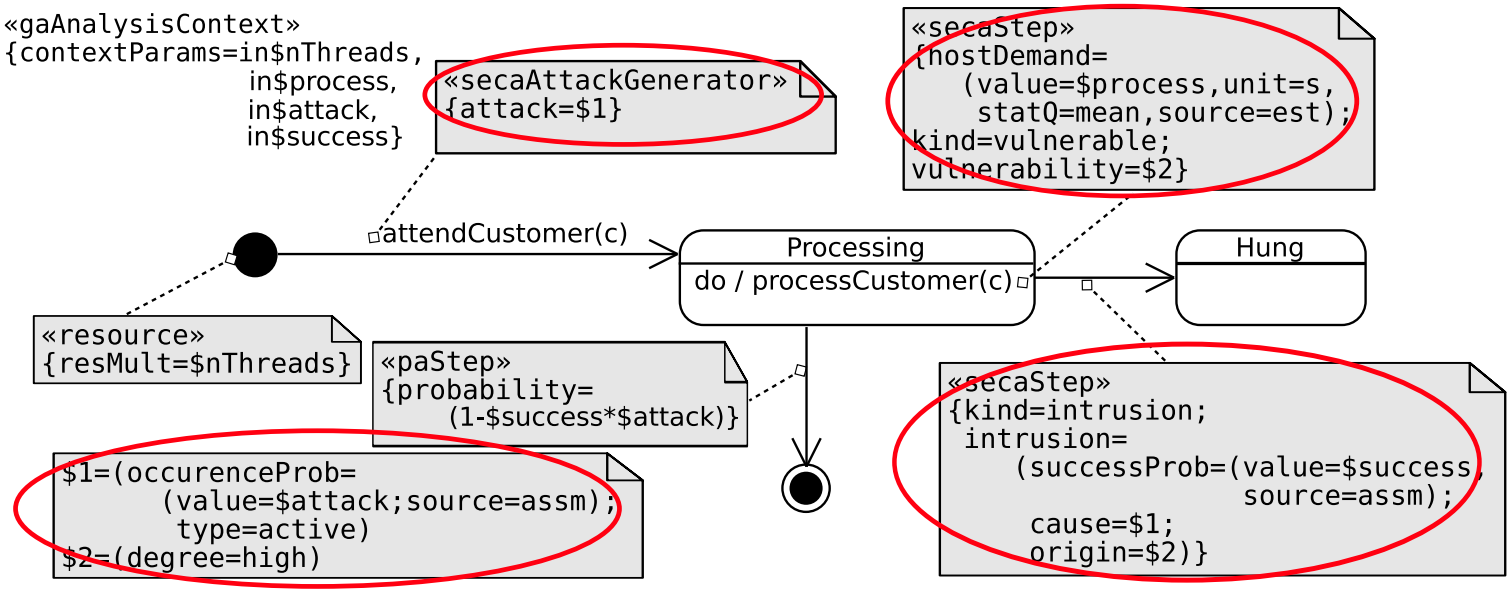
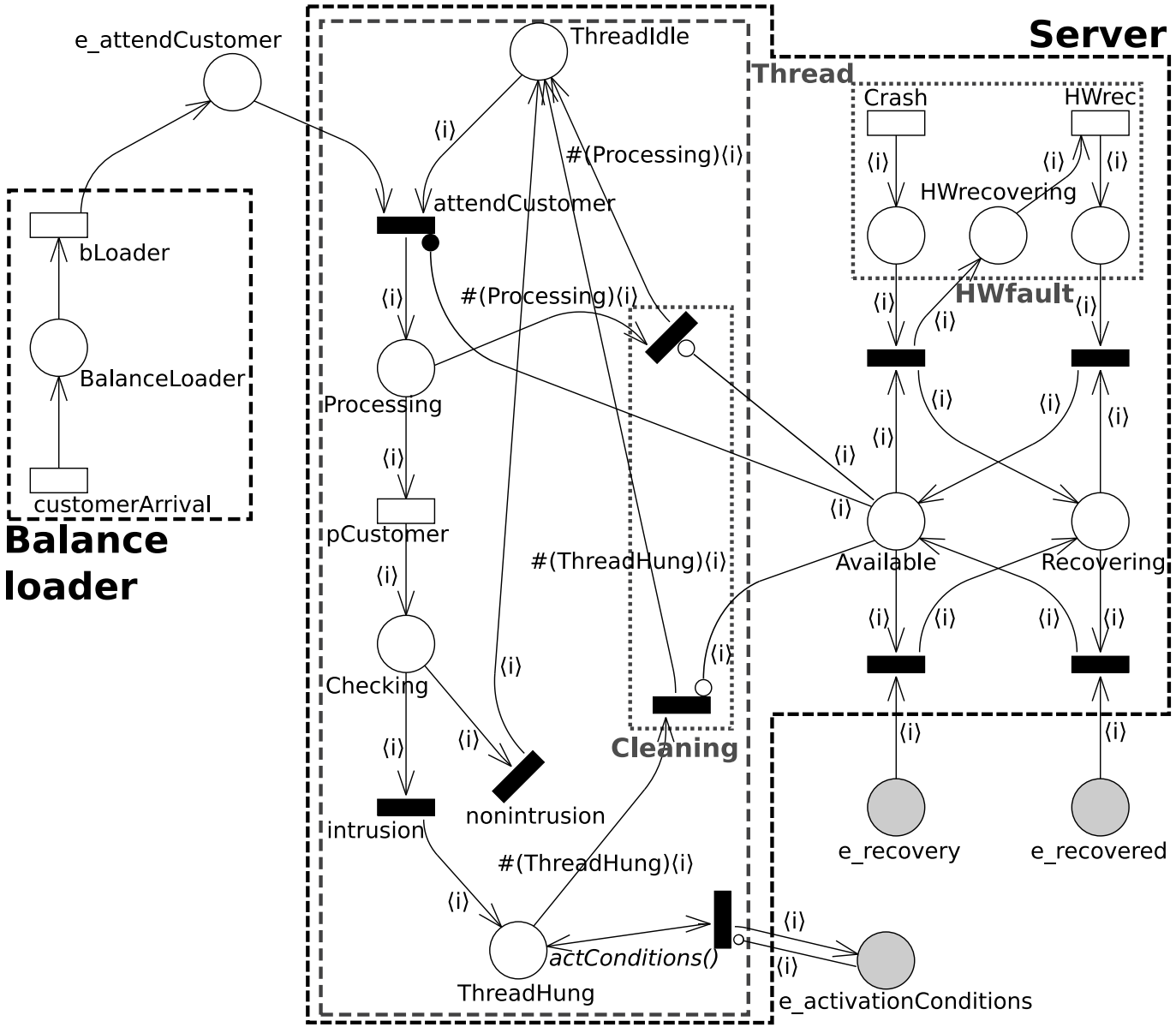


Figure: Available UML sub-state machine diagram.

Example: formal model (I)



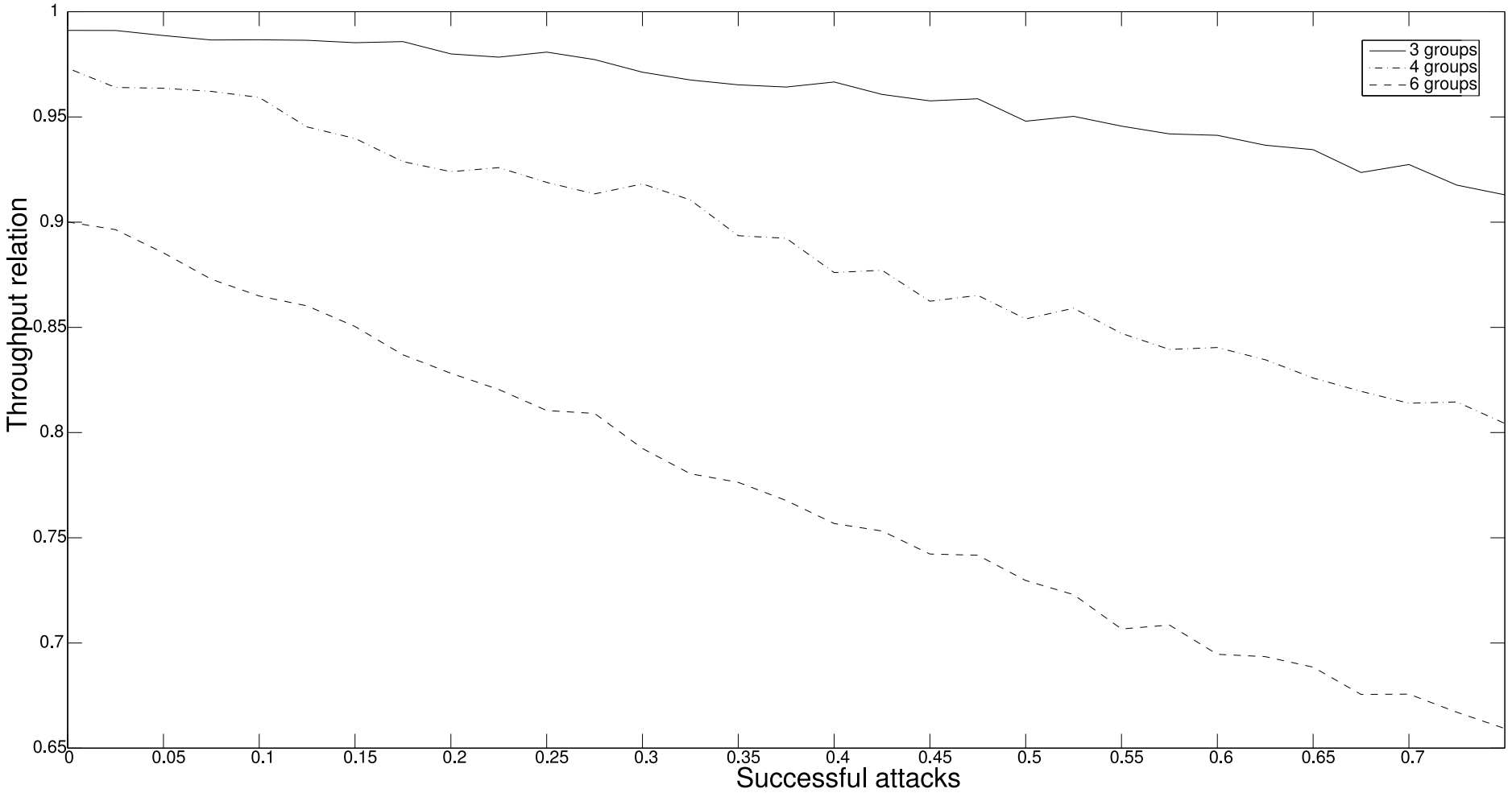
Example: experiments (I)

Parameters	Value
nDevices	12
k	2, 3, 4
f	1
timeOut	120, 180*s*
detect	100 ms
pRecovery	120 s
rRecovery	120 s
nThreads	10
crash	432000 s
HWrec	43200 s
balance	200 ms
customerLoad	0.5 customers/s
process	300 s
attack	30%
success	0% $\cdots$ 75%

Simulation parameters

- CPN simulated with GreatSPN tool
- Confidence level 99%, accurancy 1%
- Evolution phase 604800 t.u., initialisation phase 86400 t.u.

Example: results



Related work and conclusion (I)

Related work

- Petri Nets + design of critical systems
  - Specifying data (Z) and validation (Petri Nets), *Heiner et al.*
  - TB nets to specify control, function and timing issues, *Ghezzi et al.*
  - CPNs to quantify operational system integrity, *Houmb et al.*
- FT techniques applied at software architectural level
  - Architectural patterns, satisfaying quality attributes, *Harrison and Avgeriou*
  - Reflective model for making failure-resistant apps, *Nguyen-Tuong and Grimshaw*
  - System dependability modelling using AADL, *Rugina et al.*
  - UML diagrams to dependability models *Bondavalli et al.*
  - UML *ad-hoc* library as a framework for analysis of software performance degradation due to security solutions (*Cortellesa et al.*)

Related work and conclusion (II)

Conclusions

- Combining FT techniques + software behavioural designs
  - $\rightarrow$ useful for dependability assessment
- Contributions:
  - Proactive and reactive techniques + sw design $\rightarrow$ analysis
  - Easy integration of FT techniques into software design

Future work

- Main goal: build FT UML libraries
  - Reuse of the approach with other FT techniques
- Extend to other UML diagrams (sequence diagrams?)

Being critical...

- Combination should be made at UML level (not at PN models)

Integrating Fault-Tolerant Techniques into the Design of Critical Systems

Ricardo J. Rodríguez and José Merseguer
{rjrodriguez, jmerse}@unizar.es

Universidad de Zaragoza
Zaragoza, Spain

23rd June 2010

ISARCS'10: 1st International Symposium on Architecting Critical Systems
Prague, Czech Republic