Comparing two UML Profiles for Non-functional Requirement Annotations: the SPT and QoS Profiles

Simona Bernardi
Università di Torino, Torino (Italy)
and
Dorina C. Petriu
Carleton University, Ottawa (Canada)

SVERTS Lisbon (Portugal) Oct.11th, 2004

Outline

- Introduction
- Embedded automation system example
- Application of the QoS and of the SPT annotation approaches
- Comparison between the two Profiles
- Conclusion

Introduction

- Motivation
  - Provide an useful comparison between the two Profiles at an "appropriate moment in time"
- Contribution
  - Show "how to use" the new QoS Profile
  - Emphasize the main advantages/drawbacks of the two annotation approaches
  - Identification of new concepts that should be included in the two Profiles

Embedded automation system example (I)

- Distributed cyclic application activating two concurrent processes
- Process behavior:
  - Reading of a sample input from plants
  - Elaboration of the future state
  - Saving of the new state in memory and production of output for the controlled plants
- Synchronization at the end of each cycle

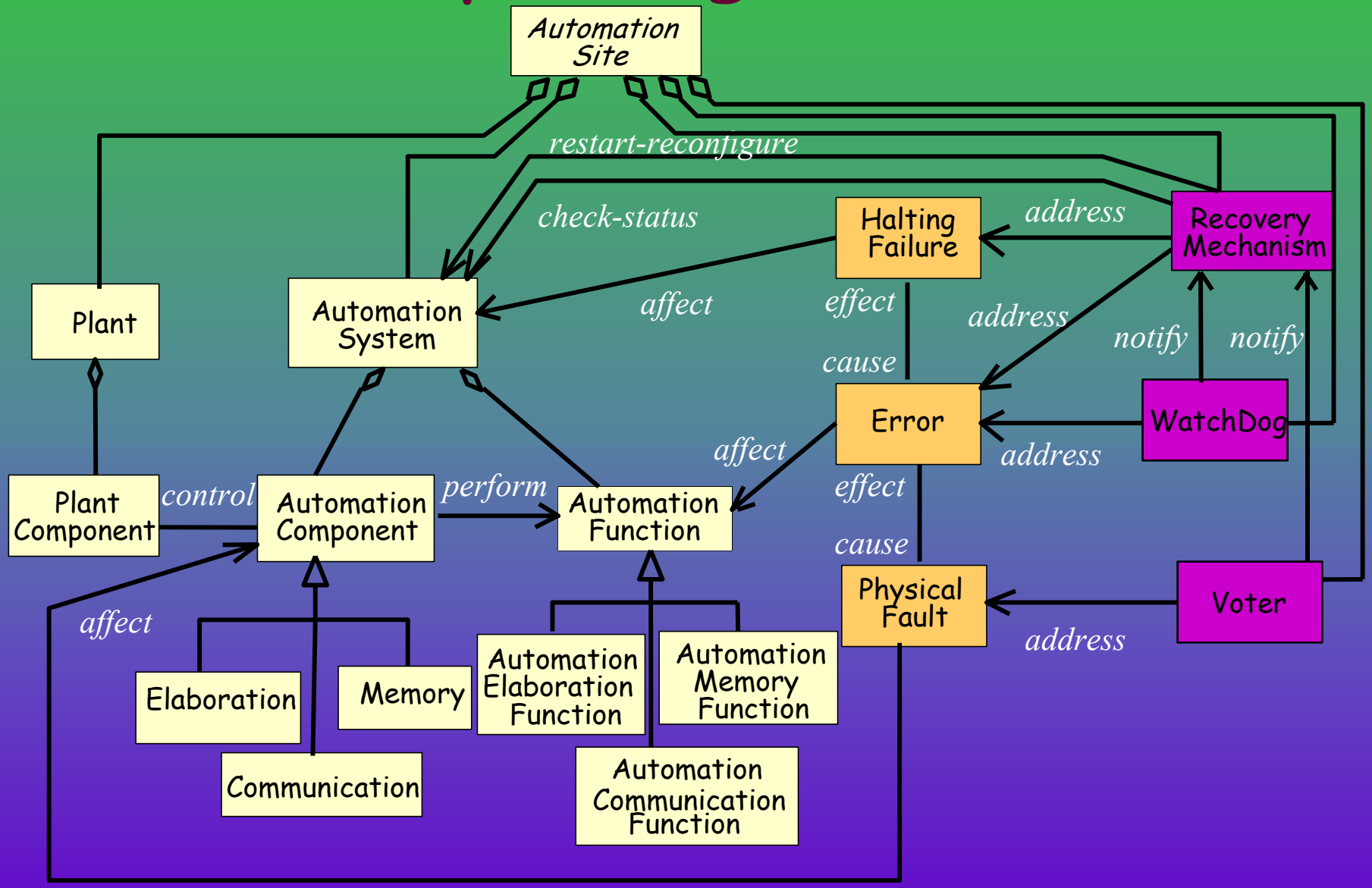
Embedded automation system example (II)

- A fault tolerance strategy is adopted including:
  - Fault masking during elaboration of the future state (hw/sw replication and voting)
  - Error detection during read/save operations (sw watchdog)
  - Error diagnosis
  - Error backward recovery or reconfiguration from failure

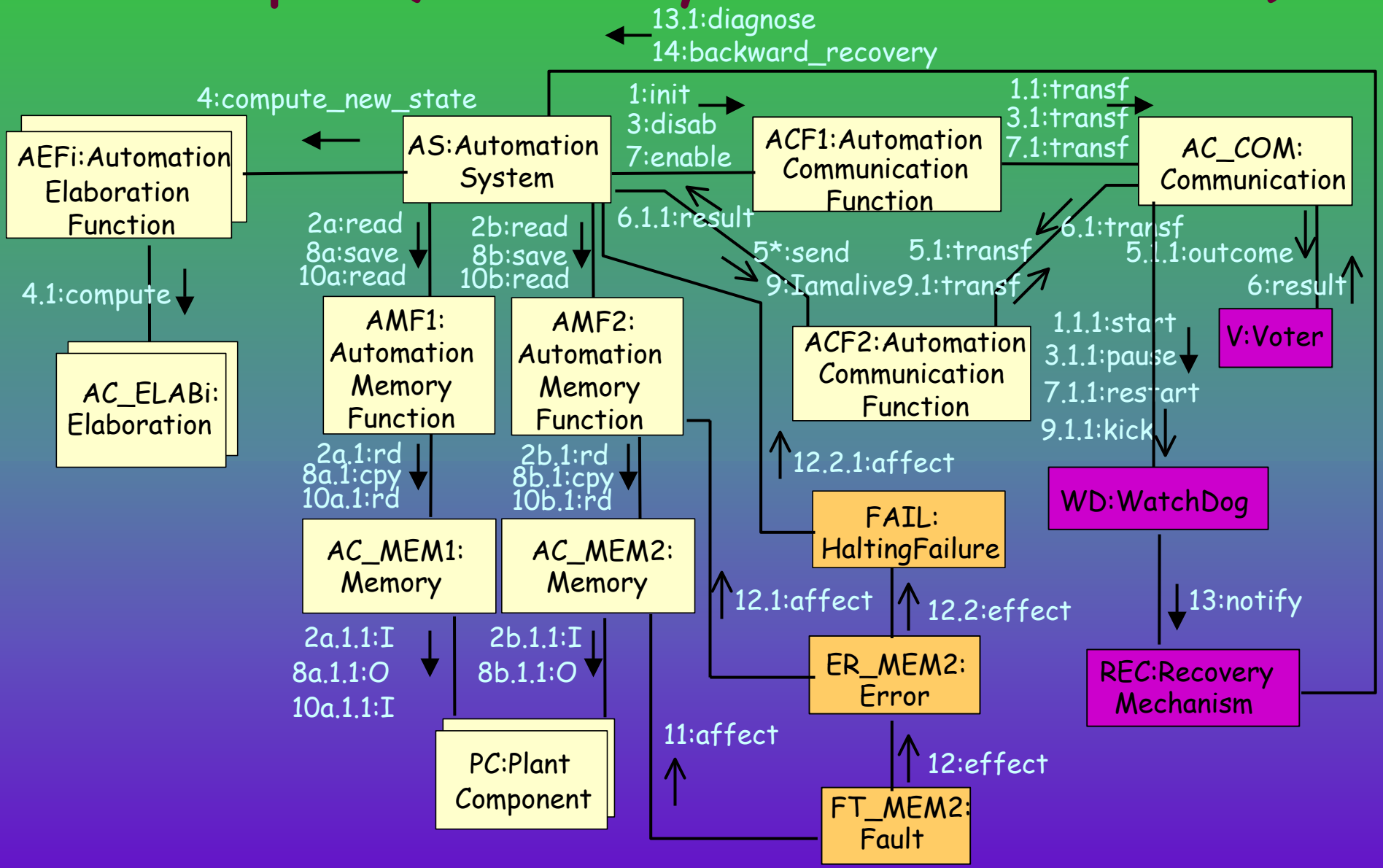
Embedded automation system example (III)

- Different non-functional reqs:
  - Timing:
    - "cycle-time to be at most of 15 ms."
  - Dependability:
    - "mean availability to be at least 98%"
  - Performability:
    - "optimal values for the timer duration of the watchdog and for the number of replicas to have the minimum *overhead* for the adopted fault tolerance strategy"

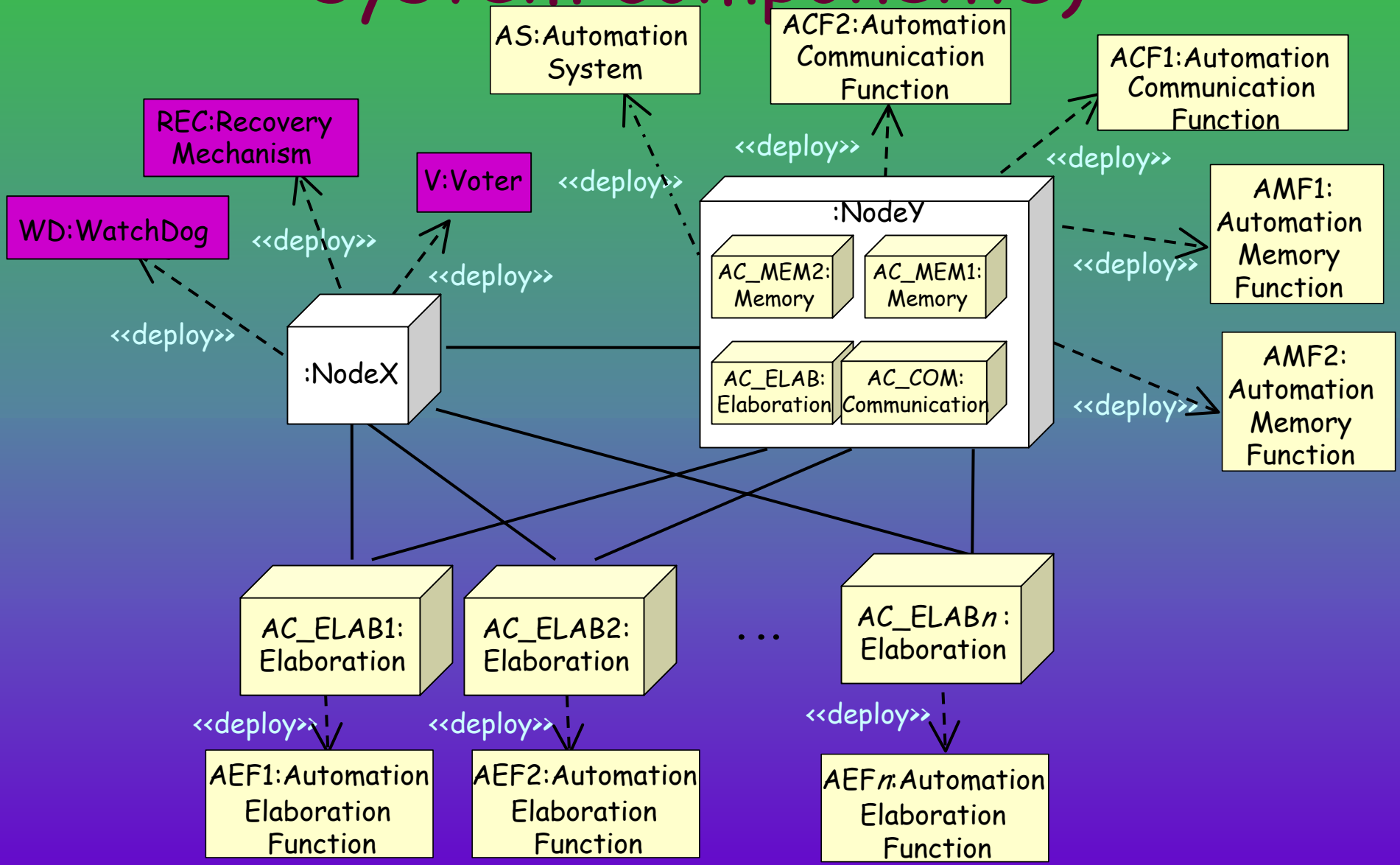
Example (logical view)



Example (Memory Fault Scenario)



Example (Physical allocation of the system components)



QoS annotation: overview

- Recently proposed with a wider scope than the SPT Profile, allowing for *user defined* QoS and Fault-Tolerance concepts
- It contains:
  - QoS subprofile - extends the General Resource Model (GRM) from the SPT profile
  - QoS model library
  - Risk Assessment subprofile
  - Fault Tolerance subprofile
- Current status
  - Adopted by OMG in June 2004 (OMG document ptc/2004-06-01)
  - Under improvements by the Finalization Task Force, preceding the formal adoption.
- No complete examples of its application

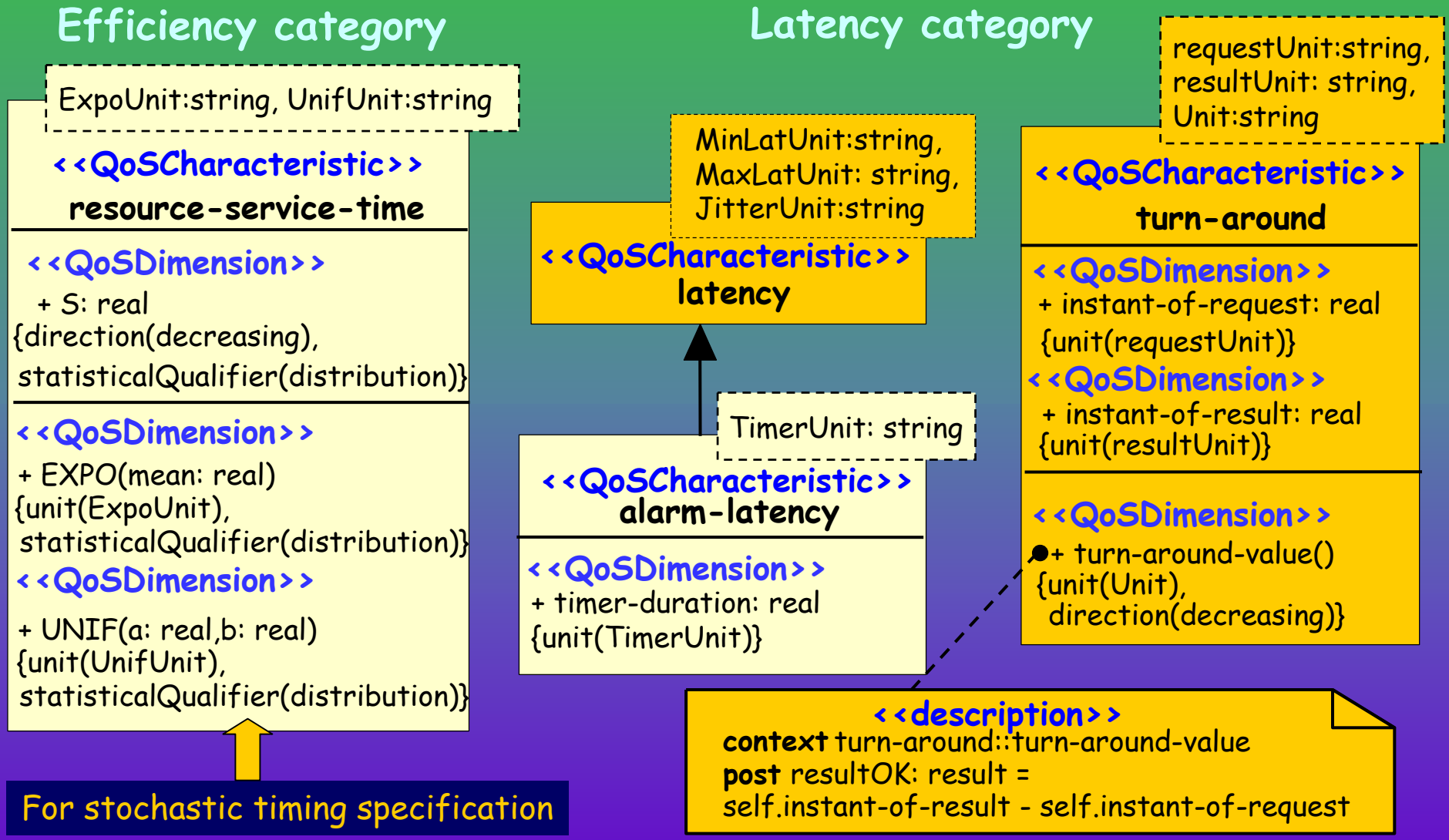
QoS annotation: approach

Three-step:

1. Define the *QoS characteristics* of interest for the analysis to be carried out in a given system domain
   → definition of template classes
2. Define the "Quality Model" for the system under study
   → the parameters of the *QoS characteristic* template classes specified in the first step are all resolved by bindings
3. Annotation of the system models through *QoS constraints*

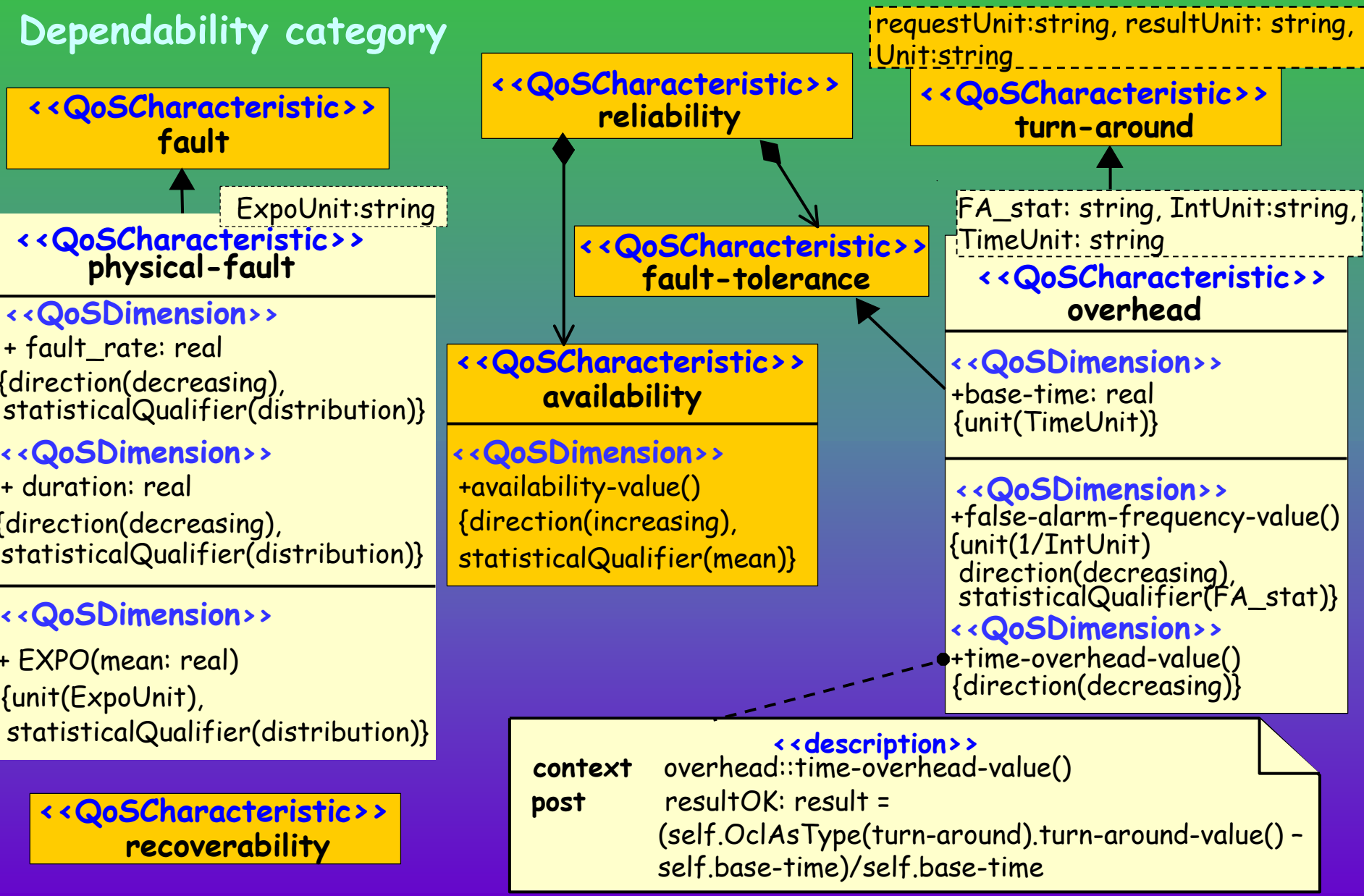
☞ Let us apply the QoS approach to the running example for performability analysis purpose …

QoS annotation: step 1 (I)

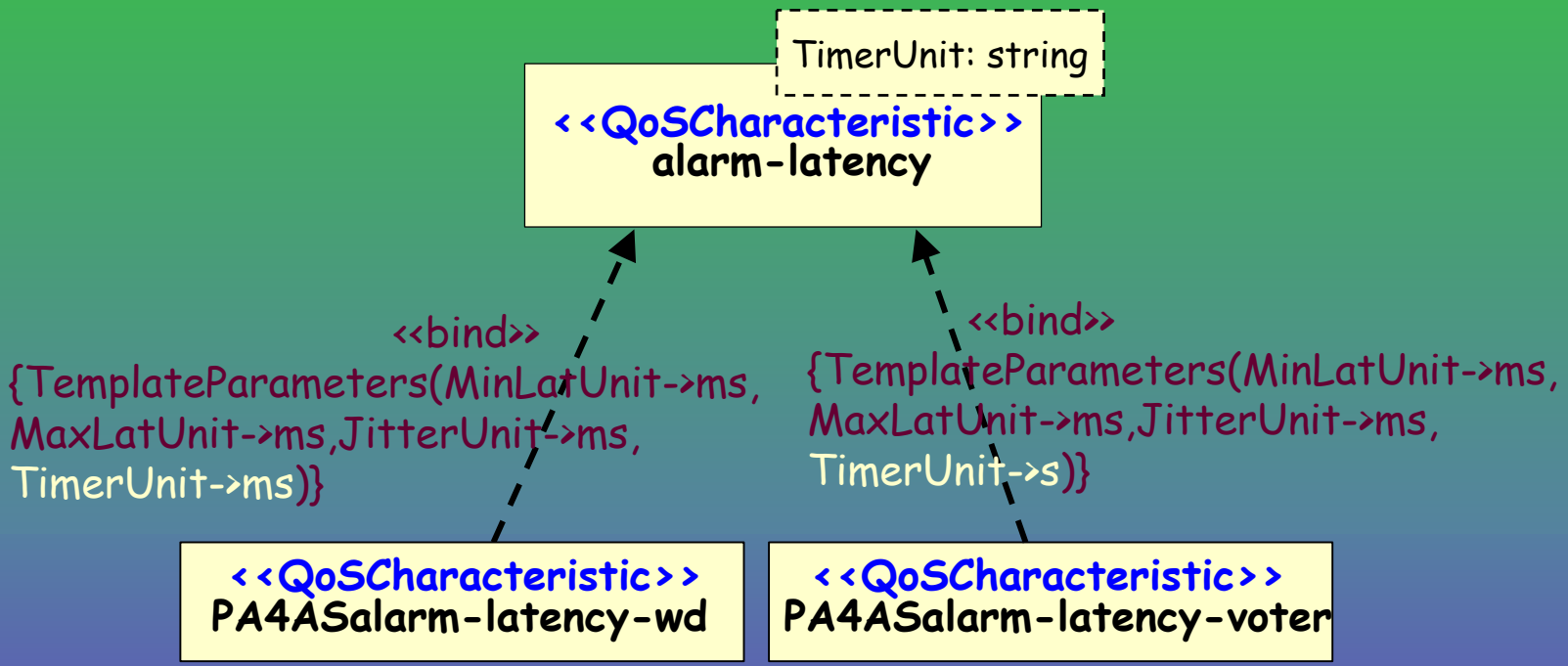


QoS annotation: step 1 (II)

Dependability category



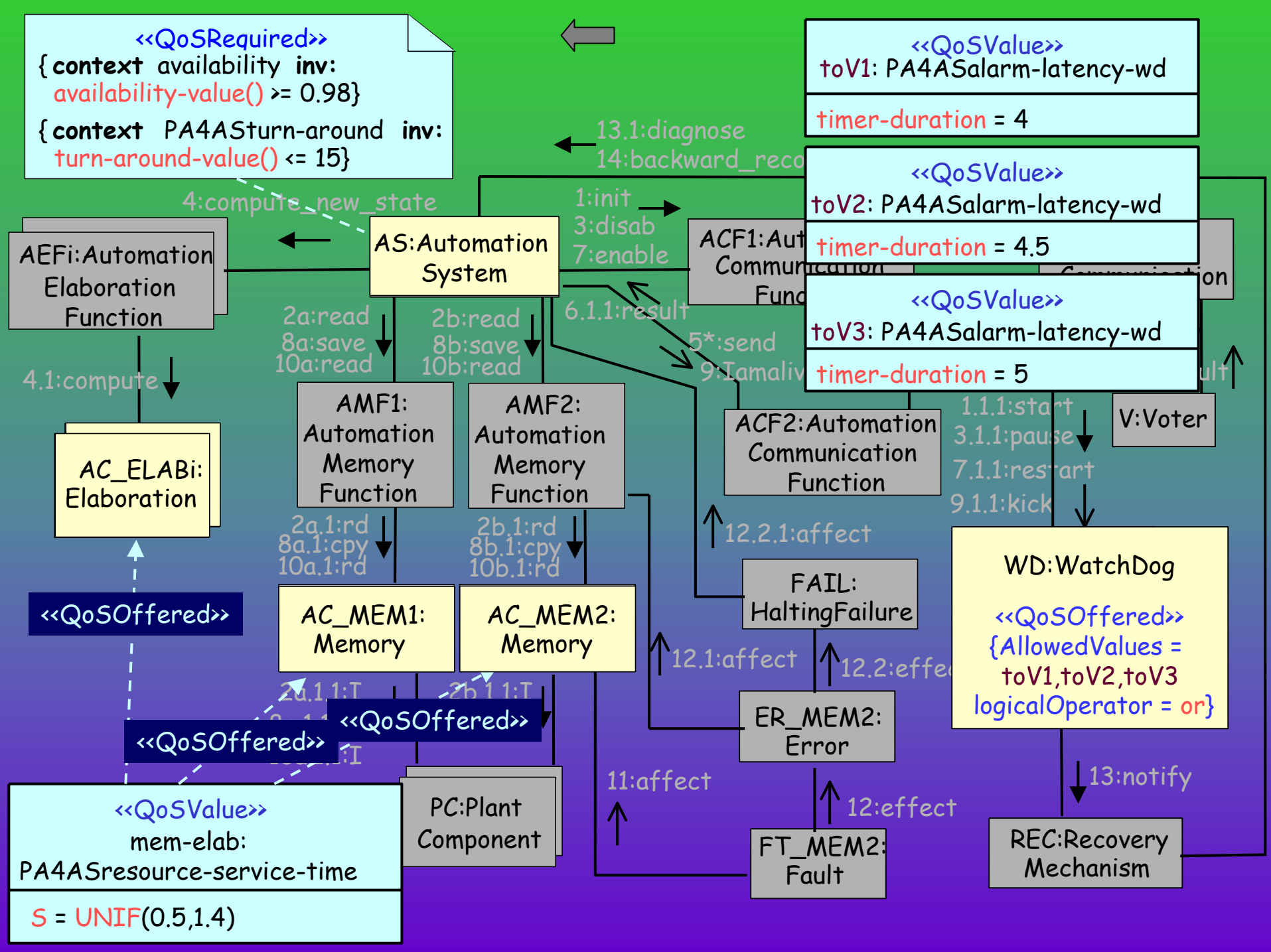
QoS annotation: step 2



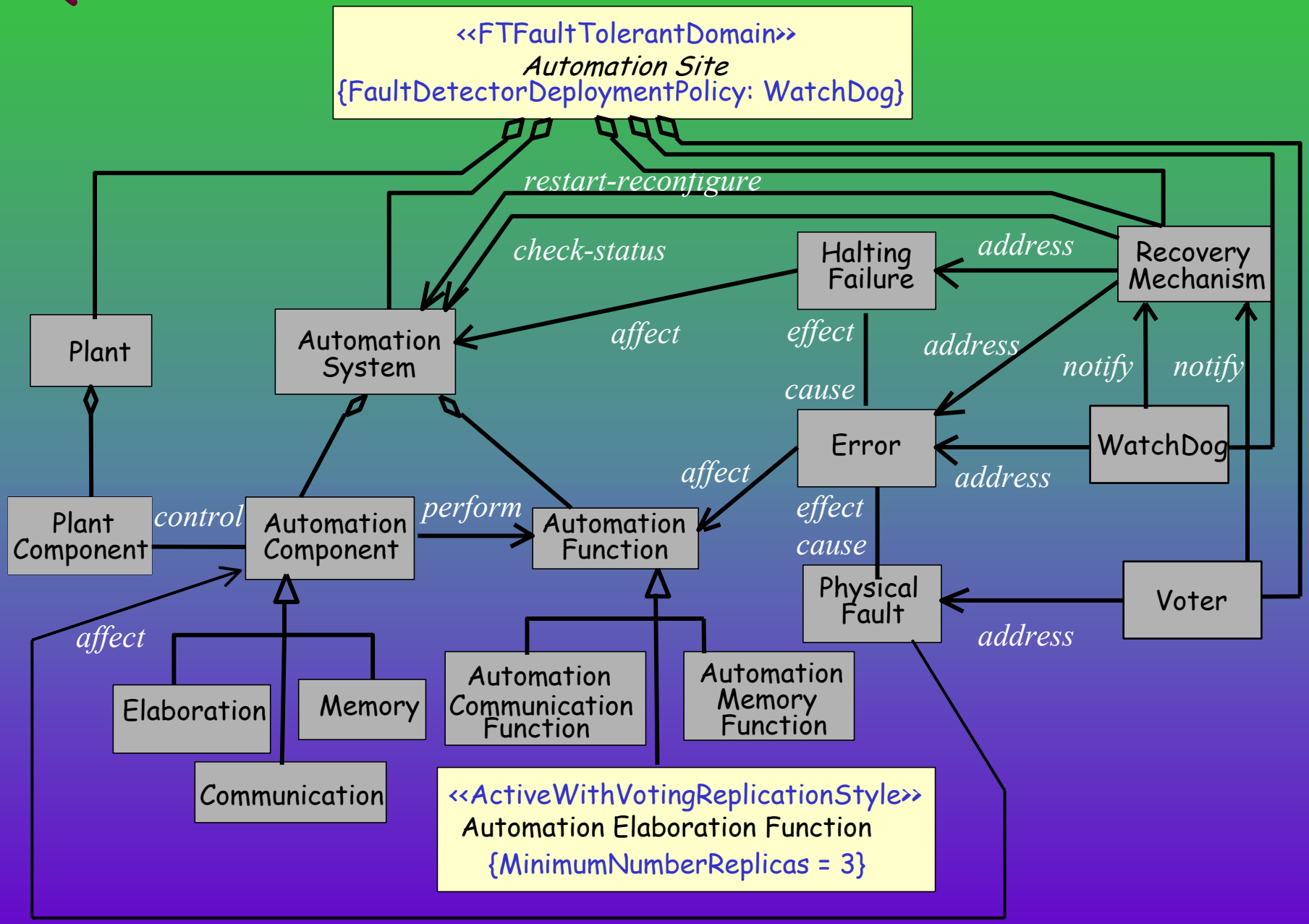
All the template class parameter must be resolved !!
→ Parametric performability models cannot be derived from UML models annotated according to the QoS Profile

QoS annotation: step 3

- Carried out by means of *QoSConstraints:*
  - *QoSRequired, QoSOffered, QoSContract*
- Three possible ways:
  1. Attach a note with a *QoSConstraint* written in OCL to a model elements (e.g)
  2. Connect the constrained element with an instance of a class stereotyped as *QoSValue* by a *QoSConstraint* dependency (e.g.)
  3. Stereotype the constrained model element with a *QoSConstraint* and use *AllowedValue* and *LogicalOperator* properties to reference a set of *QoSValues* and their relationships (e.g.)


«QoSRequired»
{context availability inv:
availability-value() >= 0.98}
{context PA4ASturn-around inv:
turn-around-value() <= 15}
«QoSValue»
toV1: PA4ASalarm-latency-wd
timer-duration = 4
«QoSValue»
toV2: PA4ASalarm-latency-wd
timer-duration = 4.5
«QoSValue»
toV3: PA4ASalarm-latency-wd
timer-duration = 5
13.1:diagnose
14:backward_reco
4:compute_new_state
1:init
3:disab
7:enable
AEFi:Automation Elaboration Function
AS:Automation System
ACF1:Aut Commun Func
2a:read
8a:save
10a:read
2b:read
8b:save
10b:read
6.1.1:result
5*:send
9:Iamaliv
4.1:compute
AMF1: Automation Memory Function
AMF2: Automation Memory Function
ACF2:Automation Communication Function
V:Voter
1.1.1:start
3.1.1:pause
7.1.1:restart
9.1.1:kick
AC_ELABi: Elaboration
2a.1:rd
8a.1:cpy
10a.1:rd
2b.1:rd
8b.1:cpy
10b.1:rd
12.2.1:affect
FAIL: HaltingFailure
WD:WatchDog
«QoSOffered»
{AllowedValues = toV1,toV2,toV3
logicalOperator = or}
«QoSOffered»
AC_MEM1: Memory
AC_MEM2: Memory
12.1:affect
12.2:effe
«QoSOffered»
«QoSOffered»
ER_MEM2: Error
13:notify
11:affect
12:effect
«QoSValue»
mem-elab:
PA4ASresource-service-time
S = UNIF(0.5,1.4)
PC:Plant Component
FT_MEM2: Fault
REC:Recovery Mechanism

QoS annotation: FT architectures



SPT annotation: overview

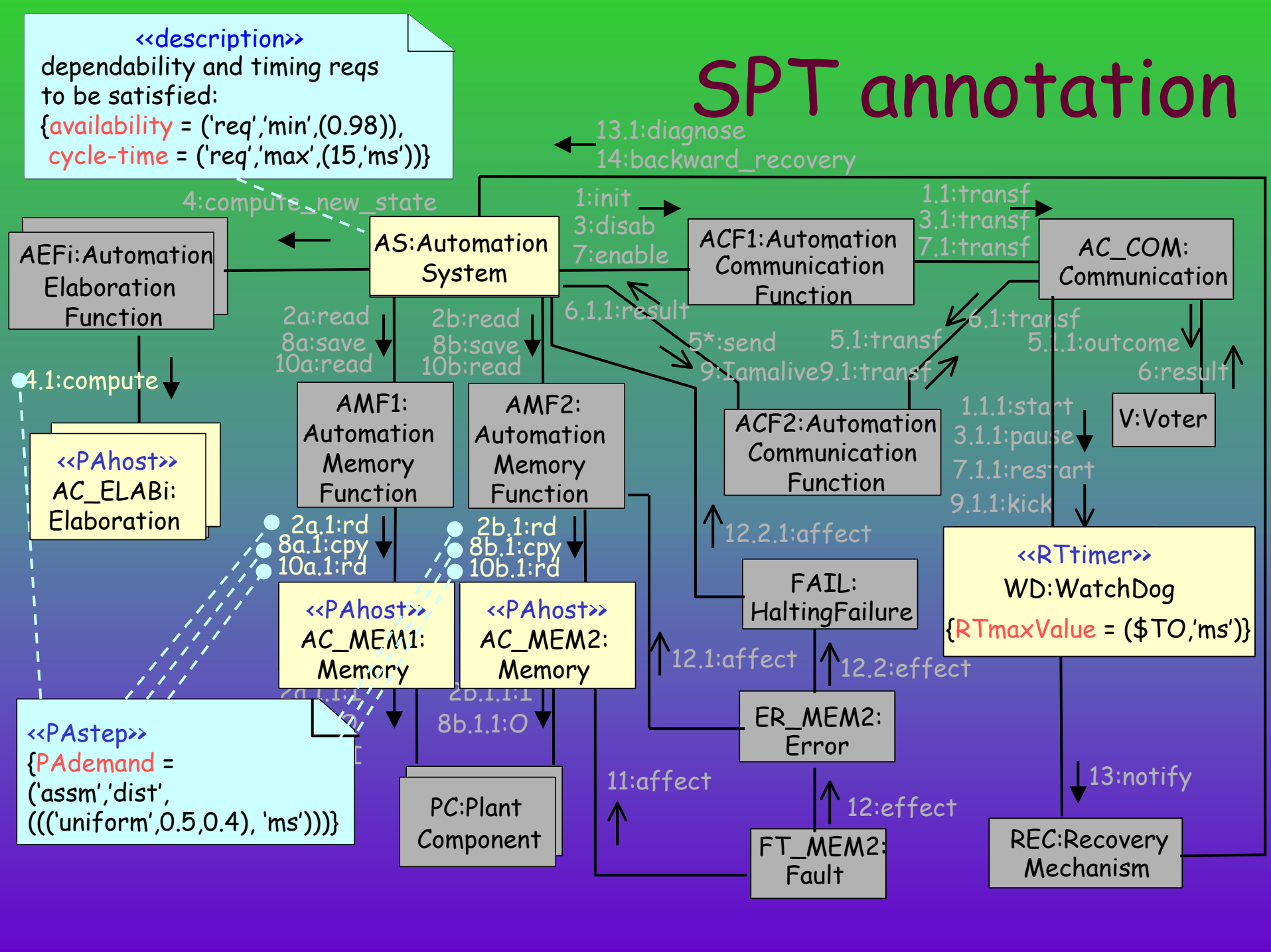
- First attempt to extend UML with basic timing and concurrency concepts for expressing non-functional reqs/properties
- It contains:
  - General Resource Modeling Framework (GRM)
  - Analysis Models (schedulability and performance sub-profiles)
  - Infrastructure Models (Real-Time CORBA)
- Current status:
  - Formal OMG adoption in Sept. 2003
  - On-going process of issuing a new RFP for V2 to bring it in line with UML2.0
  - Applied on several case studies/examples
  - Integrated in several UML commercial tools (e.g., Artisan Real-time Studio, Rhapsody)

SPT annotation: approach

- More straightforward to apply than QoS but less flexible
- Provides a set of stereotypes and related attributes that can be used directly for the annotation of the system models
- No customization possibilities

☞ Let us apply the SPT approach to the running example for performability analysis purpose …

SPT annotation



QoS vs/ SPT Profile (I)

- ✓ Type of analysis supported
  - ➢ STP limited to schedulability and performance
- ✓ Specification level
  - ➢ Both "class" and "instance" levels in principle, "instance" level in practise
- ✓ Annotation approach
  - ➢ three-step based for QoS, more straightforward for SPT
- ✓ Annotation in the system models
  - ➢ Which UML models can be annotated ?
  - ➢ How many different ways ?

QoS vs/ SPT Profile (II)

- ✓ Parameterization capabilities
  - In SPT, by convention parameters are expressed by symbolic variables prefixed by "$"
- ✓ Discrimination of the type of specification
  - QoS: *QoSRequired, QoSOffered, QoSContract*
  - SPT: modifying field for assumptions,requirements, metrics, properties ('assm', 'req', 'msr', 'pred')
- ✓ Specification of stochastic timings and related issues
  - SPT: general format for time value expressions including PdFs – "open list" of PdFs
  - QoS: no support for PdF specification
- ✓ Basic common concepts: resources

Conclusion

- ✓ Summary of the work
  - Application of SPT and QoS Profiles to embedded automation system domain for performability analysis purpose
  - Comparison between the two annotation approaches
- ✓ From the comparative analysis comes up…
  - New concepts are needed in both the Profiles to express time interval between two arbitrary events
  - A common agreement on the specification of complex timing values (especially on stochastic timing) should to be reached
  - In the QoS approach the parameterization of models is still an open problem

END

Stochastic timings and related issues