Timing analysis of sensors voting using IF

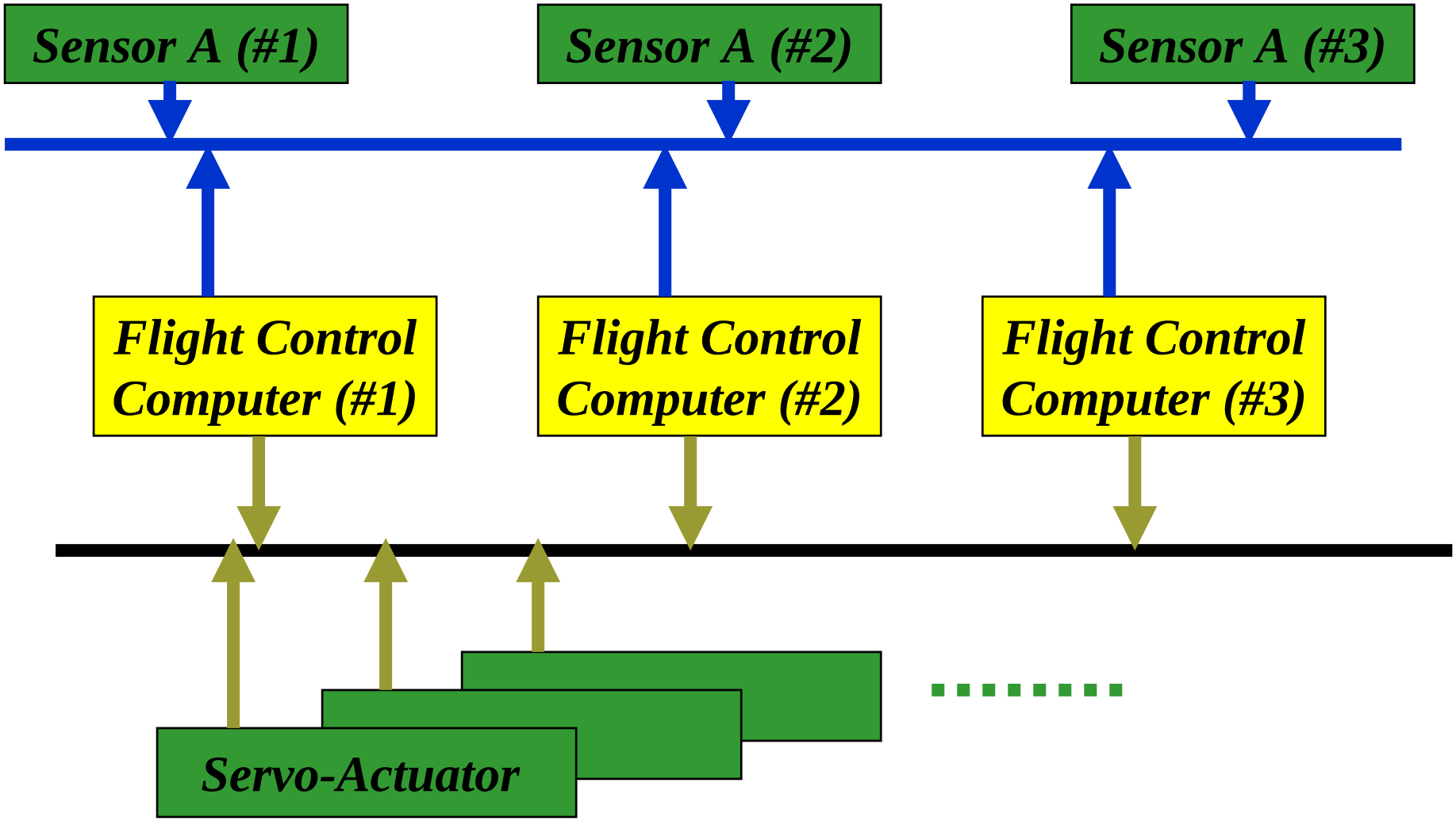
Omega workshop

Grenoble – February 17, 2005

Meir Zenou

ISRAEL AIRCRAFT INDUSTRIES LTD

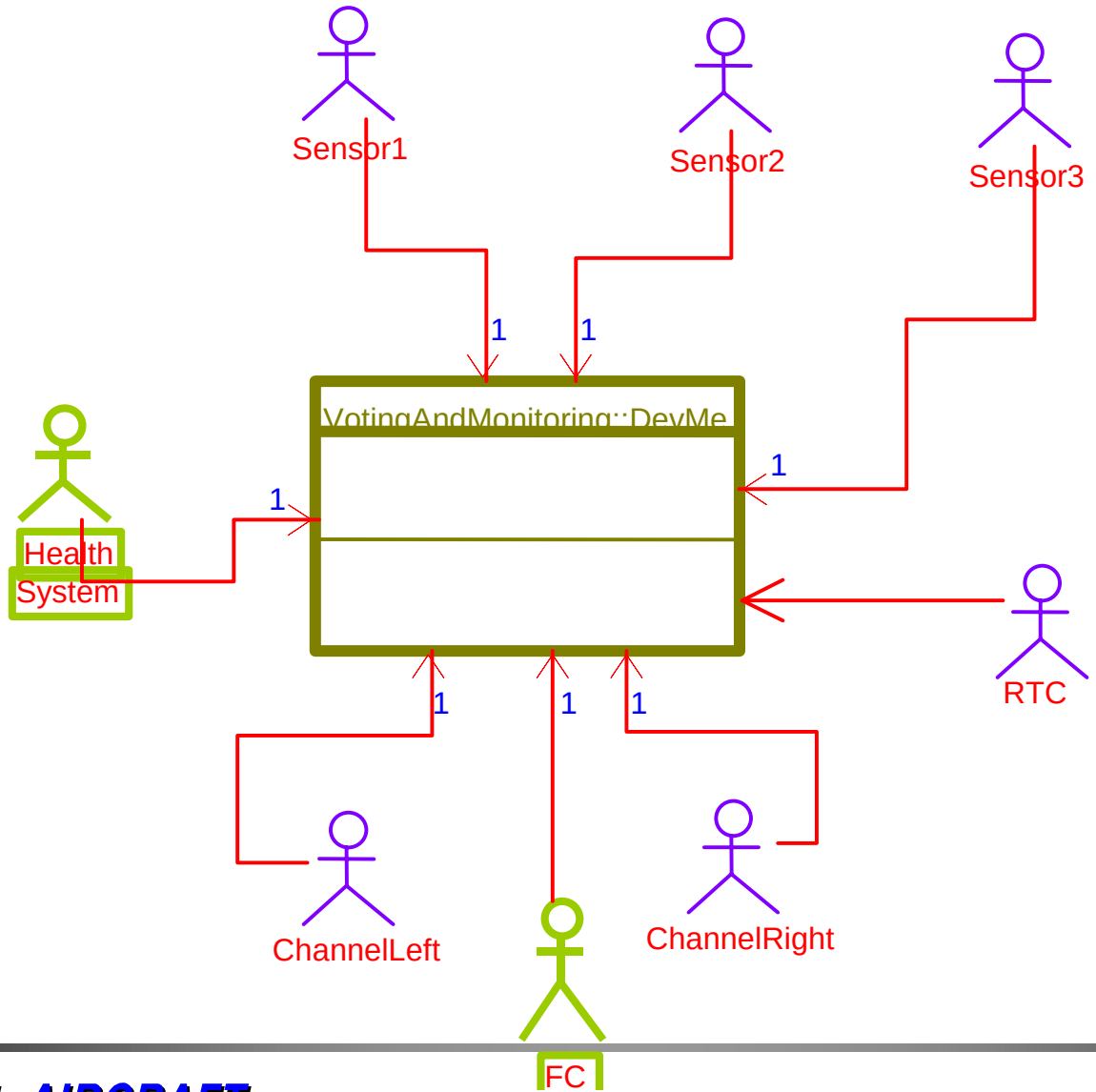
System overview



Voting & Monitoring

- **Voting :**
  - **From the three received Sensor or Command (Channel) values , detect if one of them is "out of range" (e.g : largely different from the others)**
- **Monitoring :**
  - **If a sensor/channel is detected discrepant for more than N successive cycles,this channel is disqualified . Also , if a channel is correct for more than N cycles , it is qualified**
  - **If a sensor/channel is detected discrepant for more than N' cycles (not successive) , a warning is generated**
  - **Results are provided to System Health Manager**

System overview (3)



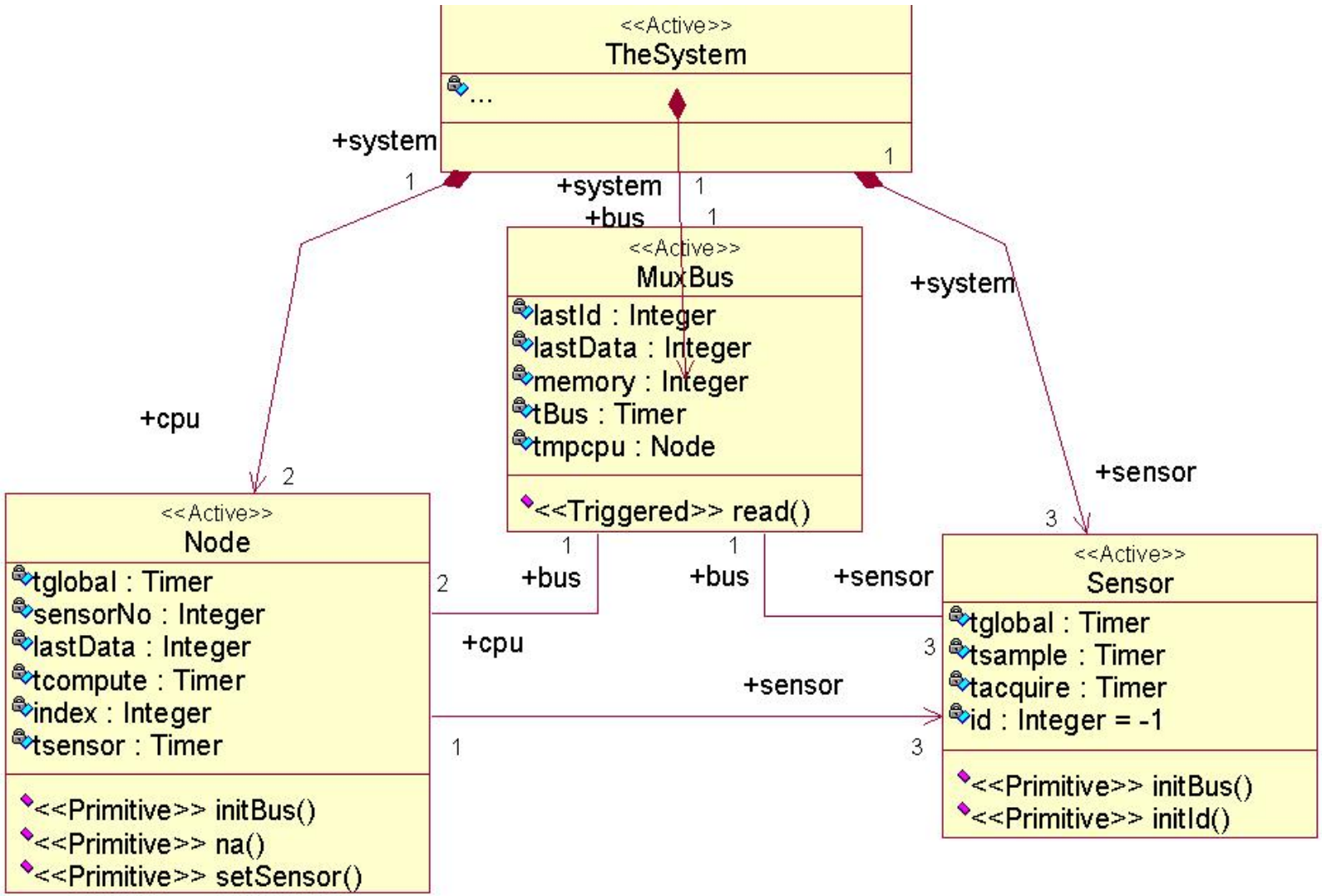
Tools evaluation

Tools	Case study	Activities
Play Engine	One CPU and 3 sensors	GUI , Behavior specification , Behavior verification
RUVE	Focus on non-realtime issues Reduced Model (12 classes , 4 statecharts)	Drive to state & Drive to Property (direct and negative)
IF	No functionality (voting , monitoring , computations..) All objects are active Two CPUs.	Mainly Verification of timed properties

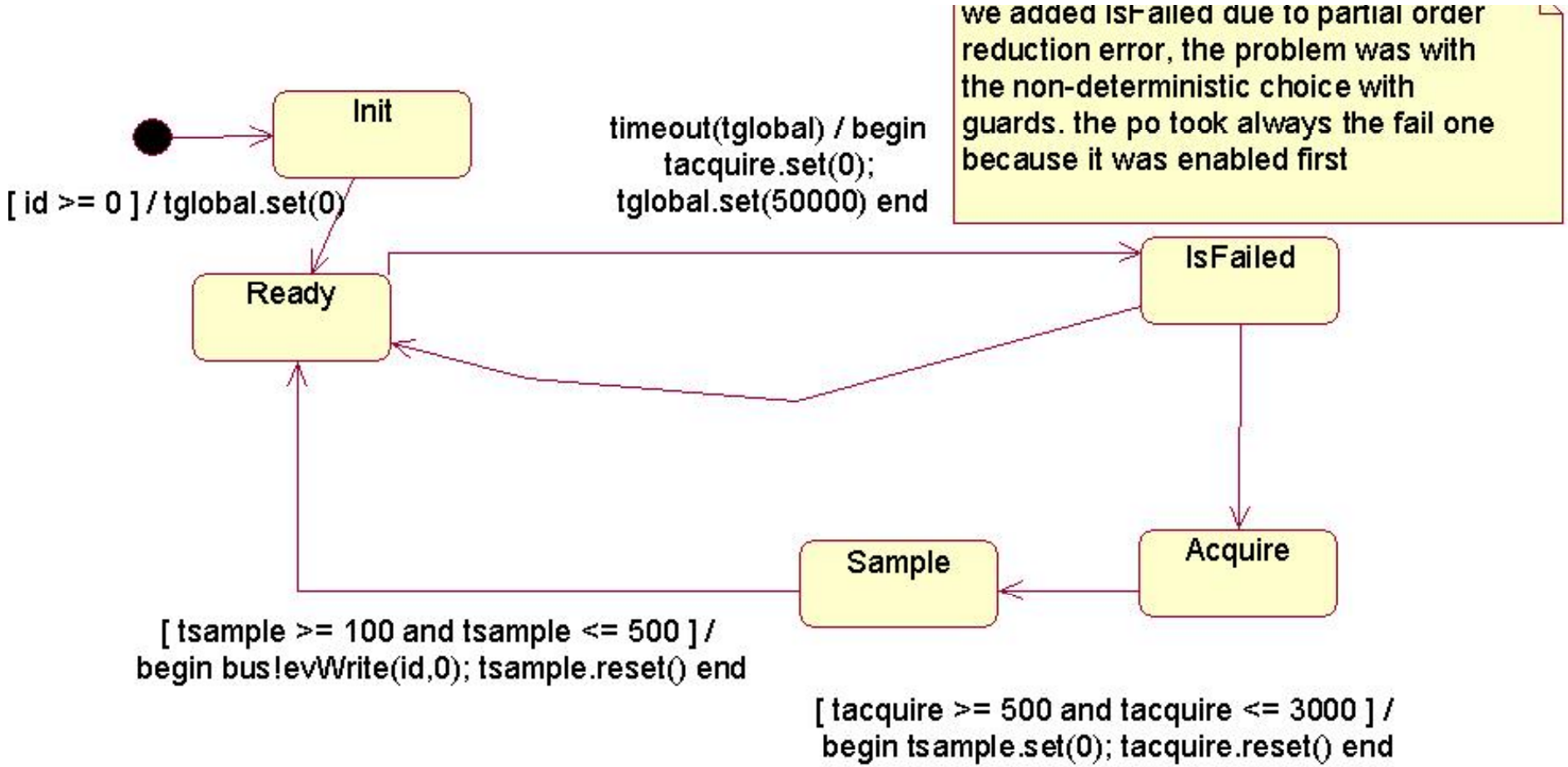
Time requirements

- **Sensor Time specifications**
  - **Acquiring of physical measurement requires 0.5 to 3 msec**
  - **Treatment and transfer to Muxbus requires 0.1 to 0.5 msec**
- **Muxbus Time specifications**
  - **Writing data from Sensor to its memory requires 100 to 200 usec**
  - **Reading data from its memory and provision to CPU requires 50 to 100 usec**

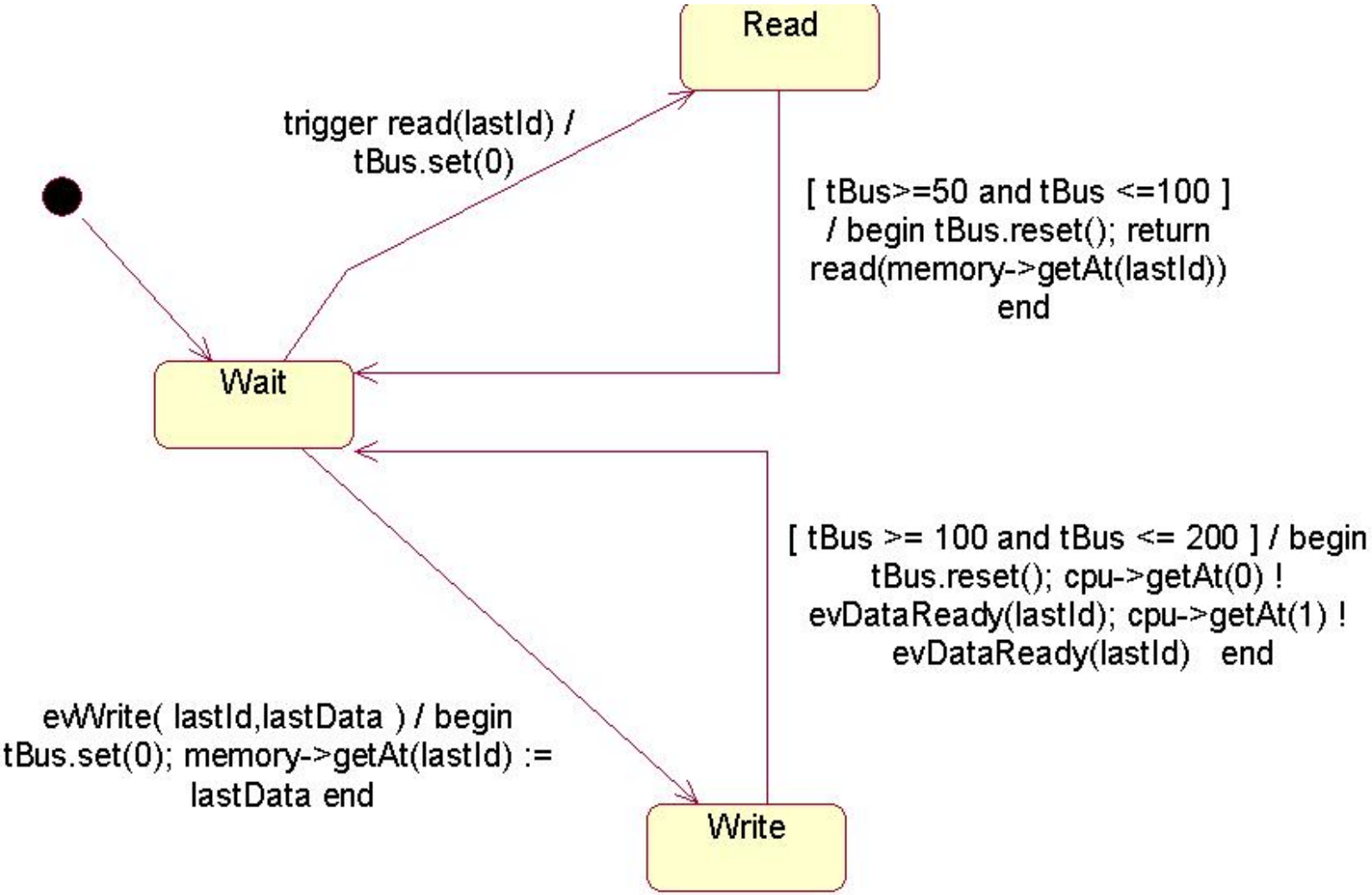
Class diagram



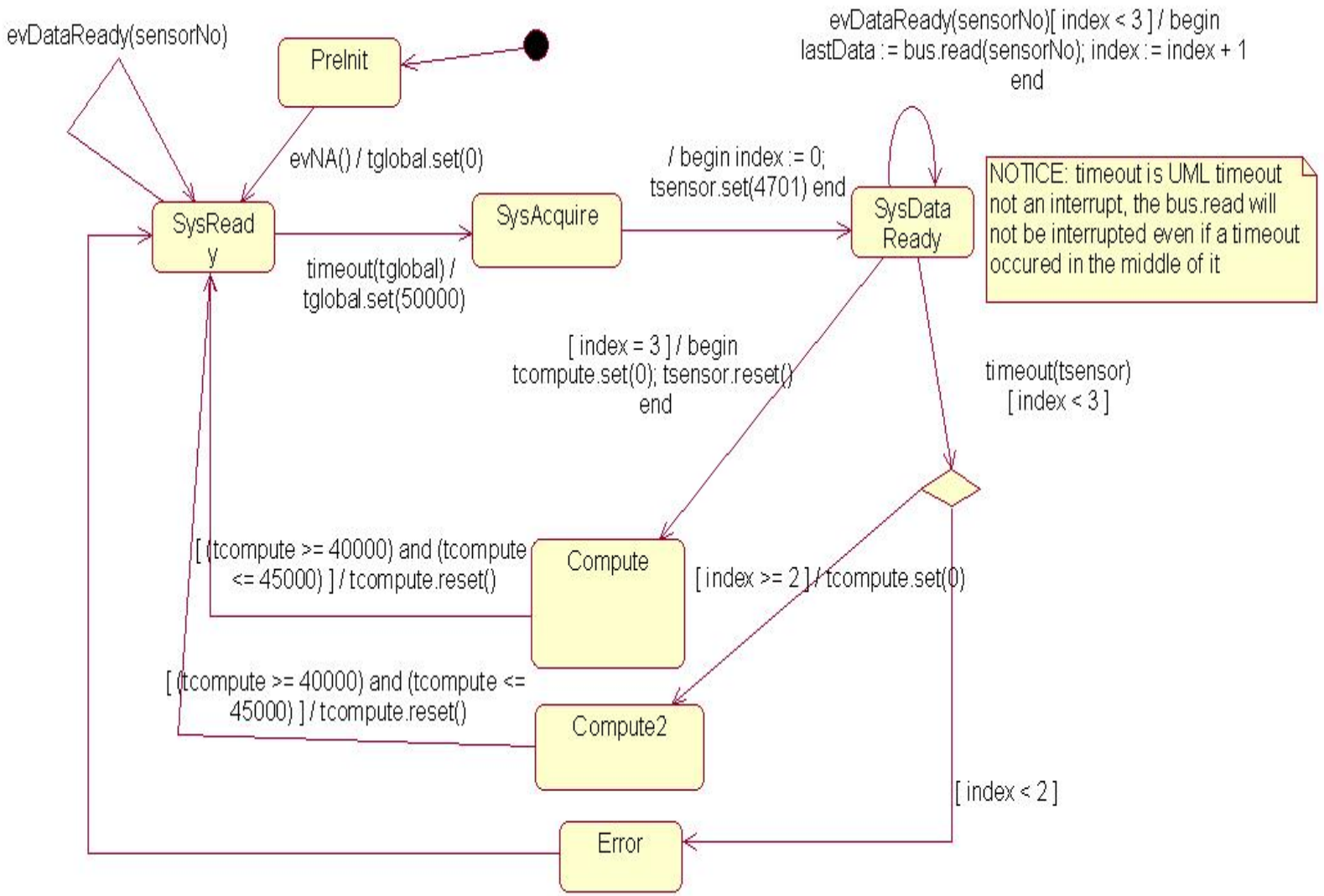
Sensor


we added isFailed due to partial order reduction error, the problem was with the non-deterministic choice with guards. the po took always the fail one because it was enabled first
Init
[id >= 0] / tglobal.set(0)
timeout(tglobal) / begin tacquire.set(0); tglobal.set(50000) end
Ready
IsFailed
Acquire
Sample
[tsample >= 100 and tsample <= 500] / begin bus!evWrite(id,0); tsample.reset() end
[tacquire >= 500 and tacquire <= 3000] / begin tsample.set(0); tacquire.reset() end

Muxbus


Read
trigger read(lastId) /
tBus.set(0)
[tBus>=50 and tBus <=100]
/ begin tBus.reset(); return
read(memory->getAt(lastId))
end
Wait
[tBus >= 100 and tBus <= 200] / begin
tBus.reset(); cpu->getAt(0) !
evDataReady(lastId); cpu->getAt(1) !
evDataReady(lastId) end
evWrite(lastId,lastData) / begin
tBus.set(0); memory->getAt(lastId) :=
lastData end
Write

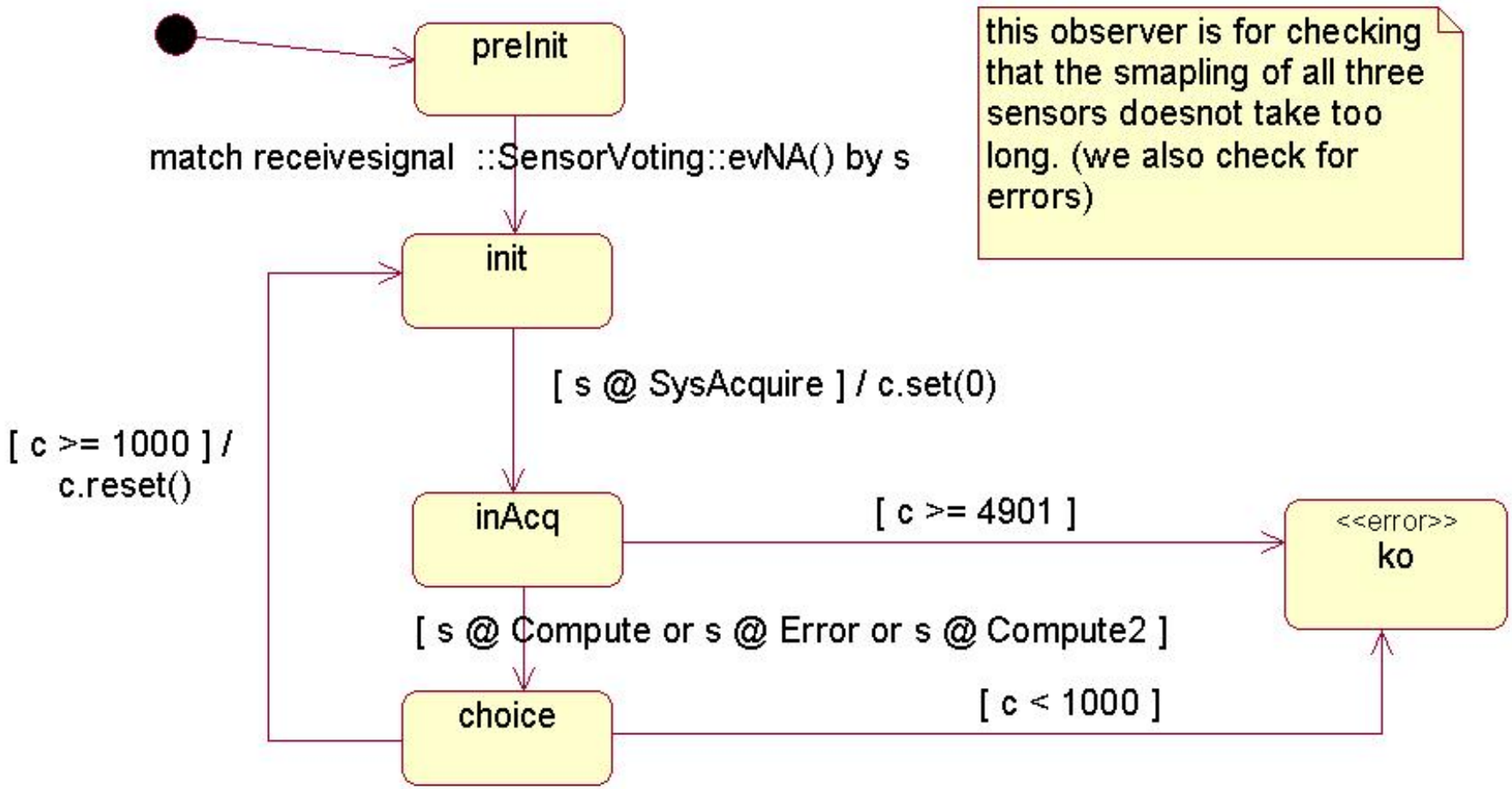
System


evDataReady(sensorNo)
PreInit
evNA() / tglobal.set(0)
SysRead y
SysAcquire
timeout(tglobal) / tglobal.set(50000)
/ begin index := 0; tsensor.set(4701) end
SysData Ready
evDataReady(sensorNo)[index < 3] / begin lastData := bus.read(sensorNo); index := index + 1 end
NOTICE: timeout is UML timeout not an interrupt, the bus.read will not be interrupted even if a timeout occured in the middle of it
[index = 3] / begin tcompute.set(0); tsensor.reset() end
timeout(tsensor) [index < 3]
[(tcompute >= 40000) and (tcompute <= 45000)] / tcompute.reset()
Compute
[index >= 2] / tcompute.set(0)
[(tcompute >= 40000) and (tcompute <= 45000)] / tcompute.reset()
Compute2
[index < 2]
Error

IF observer : Sampling time limits

- **Express the minimal and maximal delays authorized to the System till it enters the *compute* state :**
  - **Minimal delay (msec) : Min(acquiring) + Min(treatment) + 3 X Min(muxbus Write) + 3 X Min (muxbus Read) = 500 + 100 + 3x100 + 3x50 = 1050**
  - **Maximal delay (msec) : Max(acquiring) + Max(treatment) + 3 X Max(muxbus Write) + 3 X Max (muxbus Read) = 3000 + 500 + 3x200 + 3x100 = 4400**

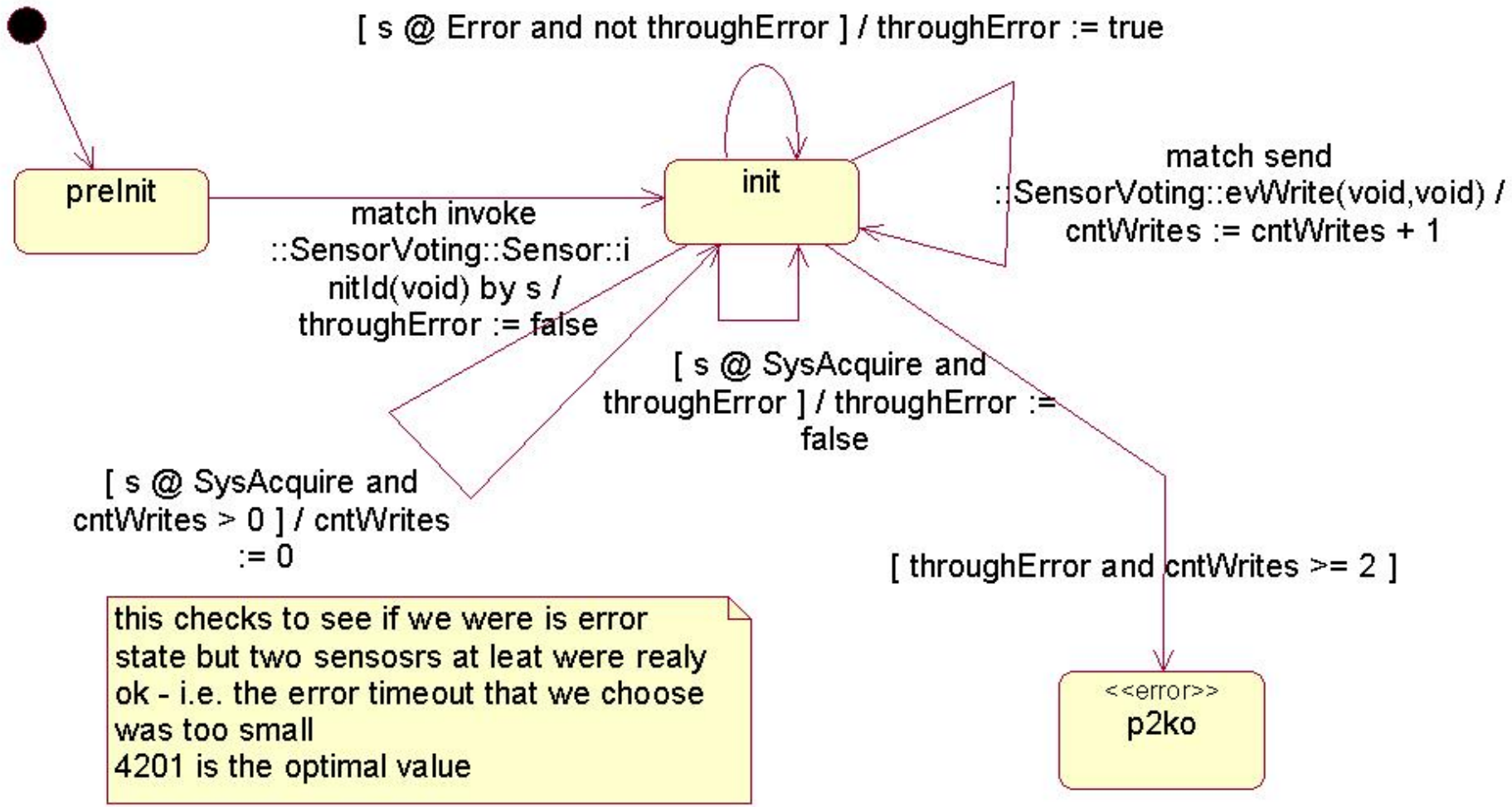
IF observer : Sampling time limits



IF observer : Entering error state

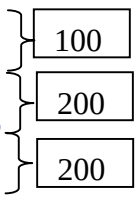
- Express that if the *system* was in *error* state , at most one sensor was OK
- This is obtained by counting the generations of *evWrite* events (expressing that the sensor is OK) and checking the counter value when the system has entered the *error* state

IF observer : Entering error state



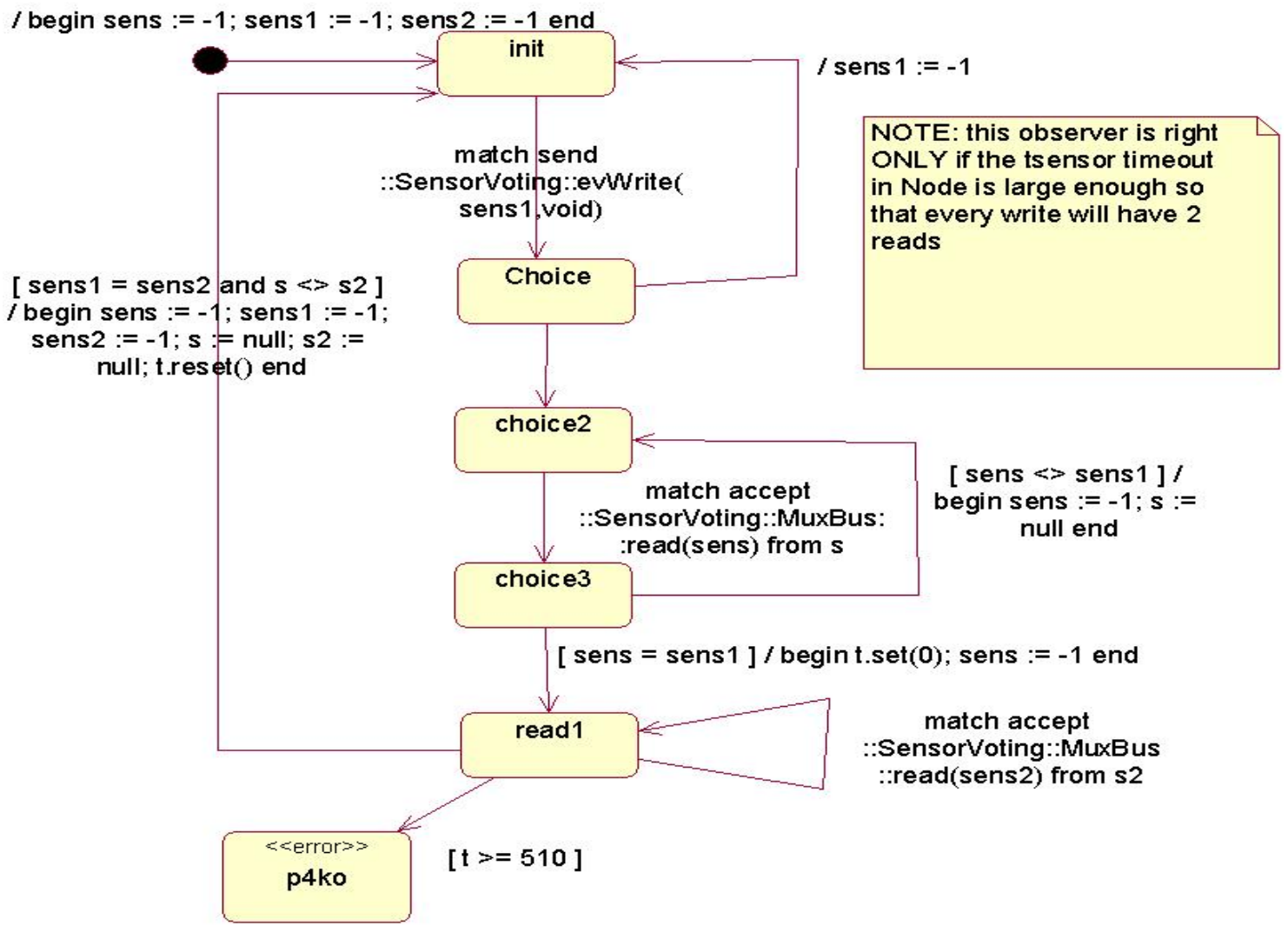
IF observer : Time difference

- **Evaluate (*t* timer) the time delay between the read of the same sensor from Muxbus memory by two different Nodes and check that this delay does not exceed an expected limit .**
- **The time limit corresponds to the following worst sequence**
  - Sensor writes Data 1
  - Node 1 reads Data 1
  - Sensor writes Data 2
  - Sensor writes Data 3
  - Node 2 reads Data 1



- **We checked the model with 2 values for the timeout : With 500 it is OK while with 501 usec we reach error state**

IF Observer : Time difference



Conclusions

Strong capability of time analysis and model checking
Can serve for Model debugging – simulation .
User friendly Observers statecharts

Observers statecharts multiplication can complicate the model.
Cryptic error messages
Scalability problem