Timing analysis of sensors voting using IF

Omega workshop
Grenoble – February 17, 2005

Meir Zenou
System overview

Sensor A (#1) → Flight Control Computer (#1) → Servo-Actuator
Sensor A (#2) → Flight Control Computer (#2)
Sensor A (#3) → Flight Control Computer (#3)
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
### Tools evaluation

<table>
<thead>
<tr>
<th>Tools</th>
<th>Case study</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play Engine</td>
<td>One CPU and 3 sensors</td>
<td>GUI, Behavior specification, Behavior verification</td>
</tr>
</tbody>
</table>
| RUVE      | Focus on non-realtime issues            | Drive to state 
& Drive to Property (direct and negative) |
|           | Reduced Model (12 classes, 4 statecharts)|                                                 |
| IF        | No functionality (voting, monitoring, computations..) | Mainly Verification of timed properties        |
|           | All objects are active                  |                                                 |
|           | Two CPUs.                               |                                                 |
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
we added IsFailed due to partial order reduction error, the problem was with the non-deterministic choice with guards. The Po took the fail one because it was enabled first.
OMEGA Workshop - Grenoble, February 17, 2005

Muxbus

```
trigger read(lastId) /
  tBus.set(0)

[ tBus >= 50 and tBus <= 100 ]
  begin tBus.reset(); return
  read(memory->getAt(lastId))
  end

[ 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
```
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

This observer is for checking that the sampling of all three sensors does not take too long. (We also check for errors.)

preInit

match receiveSignal ::SensorVoting::evNA() by s

init

[ c >= 1000 ] / c.reset()

inAcq

[ s @ SysAcquire ] / c.set(0)

[ c >= 4901 ]

choice

[ s @ Compute or s @ Error or s @ Compute2 ]

[ c < 1000 ]

<<error>> ko
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

[ s @ Error and not throughError ] / throughError := true

match invoke ::SensorVoting::Sensor::initId(void) by s /
throughError := false

[ s @ SysAcquire and throughError ] / throughError := false

[ s @ SysAcquire and cntWrites > 0 ] / cntWrites := 0

[ throughError and cntWrites >= 2 ]

this checks to see if we were in error state but two sensors at least were really ok - i.e. the error timeout that we choose was too small 4201 is the optimal value
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

```
/ begin sens := -1; sens1 := -1; sens2 := -1 end

init

match send
::SensorVoting::evWrite(sens1, void)

-choice

[ sens1 = sens2 and s <> s2 ]
/ begin sens := -1; sens1 := -1;
 sensit := 1; s := null; s2 :=
 null; t reset() end

choice2

match accept
::SensorVoting::MuxBus::read(sens) from s

-choice2

[ sens = sens1 ] / begin s := -1; s :=
 null end

-choice3

[ sens = sens1 ] / begin t.set(0); sens := -1 end

read1

<<<error>>>

p4ko

[t >= 510 ]
```

NOTE: this observer is right ONLY if tsensor timeout in Node is large enough so that every write will have 2 reads.
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