EADS SPACE Transportation Case Study
Ariane 5 Flight Software

David LESENS
Email: david.lesens@space.eads.net
Overview

- Description of the Ariane 5 case study
  - Merge asynchronous and cyclical behaviors
  - Environment
- Asynchronous behavior
- Cyclic behavior
- Tools
- Evaluation & Conclusion
Spacecraft management deals with **timed sporadic events**

⇒ **Use of timed asynchronous semantics**
Spacecraft control / command

- Acquisition of measurement
- Where am I?
- Where shall I go?
- Compute the commands
- Send commands to actuators

Globally Asynchronous / Locally Synchronous (GALS)
Construction of the UML model

- 1) Development of the spacecraft behavior model
- 2) Addition of complement for the SW ctrl / cmd design
- At each stage: environment and properties

We use
- Rationale Rose
- IF toolbox
Description of the Ariane 5 case study

Asynchronous behavior
- Model
- Property description

Cyclic behavior

Tools

Evaluation & Conclusion
Mission timing
Vehicle mode change command

Flight phases

Only the phases GROUND, EAP, and GDP are described. Phases EPC and BAL are not described.
**Environment description**

**Valve description**

- **Close**
  - Property: The Close signal shall not be received in this state.
  - Property: A valve shall remain at least 50ms in this state.
  - trigger Close() / return Close(true)
  - trigger Close() / return Close(false)

- **Open**
  - Property: The Open signal shall not be received in this state.
  - Property: A valve shall remain at least 50ms in this state.
  - [true] / return Open(true)

**Non deterministic choice**

- **Failed_Open**
  - trigger Open() / return Open(false)

- **Failed_Close**
  - trigger Close() / return Close(false)

**HW failure**

Property: The delay between two commands open and close shall be at least 50 ms.
Property example (timed)

- **Informal description**
  - If the liftoff is performed, the boosters shall be released at due time.

- **Formal description**
  - Using an observer
  - Liftoff = pyro1.ignition
  - Boosters release = pyro2.ignition

match send ::EADS::Signals::Start(void) by g / begin mc := g.Acyclic.MissionConstants; tc := g.Acyclic.TimeConstants end

wait_start

wait_ignition_p1

[ g.Acyclic.EAP.Pyro1 @ Ignition_done ]

Too late

p1_ignited

[ now >= (tc.MN_5 * 2 + mc.Tpstar_prep) ]

<<error>>

ko

Too early

choice

[ now < (tc.MN_5*2 + mc.Tpstat_prep) ]

[ g.Acyclic.EAP.Pyro2 @ Ignition_done ]

ok

enole, 17 February 2005
Overview

- Description of the Ariane 5 case study
- Asynchronous behavior
  - Cyclic behavior
    - Bus model
    - Multitasking model
    - CPU consumption model
- Tools
- Evaluation & Conclusion
Real time software design

Design of the Ariane 5 Flight Software (also used for ATV, Vega, …)

- **Use of a 1553 MIL BUS**
  - Reservation of predefined timed slot for each type of transfer
    - Bus access forbidden during physical transfer
  - Definition of a bus frame with respect to the required reactivity

- **Multitasking**
  - One thread by frequency
    - 1Hz, 10Hz, acyclic, …
  - Preemptive with fix priority
    - The higher frequency has the higher priority

A real time scheduler runs the different processes, taking into account the multitasking and the bus frame
Real time design: low reactivity

- Bus frame construction (depending on the required reactivity)
- Real time scheduler
- Measurement available at cycle start
- Commands sent at cycle end

Synchronous hypothesis
Use of SCADE
Correct “a priori”

Reactivity > basic cycle duration

60 to 90% of the SW
Real time design: high reactivity

- Bus frame construction (depending of the required reactivity)
- Real time scheduler
- Measurement available during the cycle
- Commands sent during the cycle

Synchronous hypothesis violated
Use of Ω UML
Verification "a posteriori"

10 to 40% of the SW
⇔ Hard point analysis

Reactivity < basic cycle duration
Model of the bus behaviour

Cyclic activation

Measurement on bus

Resource busy

Access forbidden

Reactivity < cycle

Commands on bus

- Grenoble, 17 February 2005
Multitasking preemptive with fix priority
Model of multitasking in Ω UML

- **Definition of task priority**
  
  ```
  begin
  theTask := new::CPU::Task::Task(1, Acyclic.Ground.CPU)
  end
  ```

  This task has the first priority

- **Definition of CPU consumption for each function**

  ```
  begin
  Cyclicas.theTask.exec(5)
  end
  ```

  This action consumes 5 units of time
Overview

- Description of the Ariane 5 case study
- Asynchronous behavior
- Cyclic behavior
  - Tools: IF Toolbox
    - Problem of time scale
    - Simulator
    - Proof tool
- Evaluation & Conclusion
Time scale problem

- Basic cycle of the cyclic behavior
  - 100 ms
  - About 100 steps

- 1 hour mission
  - 3 600 000 steps

- 6 months mission
  - 15 000 000 000 steps

- 15 years mission
  - 300 000 000 000 steps

- Explosion of the number of states
  - Several hours/days/… of simulation => not usable
  - Limit of the proof tools reached
Time scale problem: first solution

- **Abstraction of the cyclic parts**
  - Proof of the asynchronous part without the cyclic part

- **Abstraction of the asynchronous parts**
  - Proof of the cyclic part without the asynchronous part
Time scale problem: second solution

- Reduction of the mission duration
  - 30 seconds mission instead of 1 hour
  - 30 000 steps
  - Whole system validated (cyclic + acyclic)

<table>
<thead>
<tr>
<th>Mission duration</th>
<th>Number of states</th>
<th>Number of transitions</th>
<th>Proof duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 7 000 ms</td>
<td>51 324</td>
<td>54 697</td>
<td>00:03:30</td>
</tr>
<tr>
<td>2 15 000 ms</td>
<td>161 956</td>
<td>171 734</td>
<td>00:12:06</td>
</tr>
<tr>
<td>3 22 000 ms</td>
<td>303 496</td>
<td>321 206</td>
<td>00:11:33</td>
</tr>
<tr>
<td>4 30 000 ms</td>
<td>463 932</td>
<td>490 901</td>
<td>00:22:58</td>
</tr>
<tr>
<td>5 37 000 ms</td>
<td>658 981</td>
<td>696 031</td>
<td>00:34:53</td>
</tr>
</tbody>
</table>
trans no=1
  - event kind=INPUT value=2i_call_EADS_Environment_Values Open(p1=<nil>,p2=<EADS_States_EPC>,p3=<EADS_States_EPC>)
  - event kind=INFORMAL value=start transition from Close to 2i_choice___af_1411 to Open
  - event kind=INFORMAL value=return
  - event kind=OUTPUT value=2i__return_EADS_Environment_Values_Open(p1=<EADS_States_EPC>,p2=<EADS_States_EPC>)

trans no=2
  - event kind=INPUT value=2i_call_EADS_Environment_Values Open(p1=<nil>,p2=<EADS_States_EPC>,p3=<EADS_States_EPC>)
  - event kind=INFORMAL value=start transition from Close to 2i_choice___af_1411
  - event kind=INFORMAL value=start transition from 2i_choice___af_1411 to Failed_Open
  - event kind=INFORMAL value=return
  - event kind=OUTPUT value=2i__return_EADS_Environment_Values_Open(p1=<EADS_States_EPC>,p2=<EADS_States_EPC>)
A counter example is generated
## Metrics on the proof tool

<table>
<thead>
<tr>
<th>Property</th>
<th>Number of states</th>
<th>Number of transitions</th>
<th>Proof duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>liftoff_aborted_right</td>
<td>36037</td>
<td>38149</td>
<td>00:00:36</td>
</tr>
<tr>
<td>pyro_not_ignited_twice</td>
<td>35988</td>
<td>38092</td>
<td>00:00:42</td>
</tr>
<tr>
<td>valve_not_abused</td>
<td>36082</td>
<td>38210</td>
<td>00:00:37</td>
</tr>
<tr>
<td>valve_not_close_in_close</td>
<td>36010</td>
<td>38114</td>
<td>00:00:44</td>
</tr>
<tr>
<td>valve_not_open_in_open</td>
<td>35998</td>
<td>38102</td>
<td>00:00:38</td>
</tr>
<tr>
<td>liftoff_performed_right1</td>
<td>46075</td>
<td>48713</td>
<td>00:00:49</td>
</tr>
<tr>
<td>liftoff_performed_right2</td>
<td>37897</td>
<td>40550</td>
<td>00:00:55</td>
</tr>
<tr>
<td>liftoff_performed_right3</td>
<td>37961</td>
<td>40632</td>
<td>00:01:12</td>
</tr>
<tr>
<td>liftoff_performed_right3</td>
<td>1048028</td>
<td>1347402</td>
<td>00:42:50 abort</td>
</tr>
<tr>
<td>no_clock_reset</td>
<td>abort</td>
<td>abort</td>
<td></td>
</tr>
<tr>
<td>liftoff_performed_right4</td>
<td>35986</td>
<td>38090</td>
<td>00:00:38</td>
</tr>
<tr>
<td>CPU_not_in_error</td>
<td>35980</td>
<td>38084</td>
<td>00:00:53</td>
</tr>
<tr>
<td>G_cycle_is_schedulable</td>
<td>36012</td>
<td>38116</td>
<td>00:00:48</td>
</tr>
<tr>
<td>NC_cycle_is_schedulable</td>
<td>36380</td>
<td>38484</td>
<td>00:00:39</td>
</tr>
<tr>
<td>read_write_coherence</td>
<td>36618</td>
<td>38722</td>
<td>00:00:47</td>
</tr>
</tbody>
</table>
A bad written property

Match send Start(H0) by g / clock.set(0)

Lift-off performed

[ Pyro1 @ Ignition_done ]

Clock >= 2 min

<<error>>

EAP released

[ Pyro2 @ Ignition_done ]

States > 1 048 028
Transitions > 1 347 402
Duration > 00:42:50

Proof stopped
The same well written property

Proof succeed

Liftoff performed

Wait_start

Match send Start(H0) by g / clock.set(0)

Wait_ignition_pyro1

clock >= 2 min / clock.reset()

Liftoff_aborted

[ Pyro1 @ Ignition_done ]

Pyro1_ignited

Clock >= 2 min

<<error>>

ko

[ Pyro1 @ Ignition_done ]

EAP released

[ Pyro2 @ Ignition_done ]

clock.reset()

Pyro2_ignited

States 37 961
Transitions 40632
Duration 00:01:12
Overview

- Description of the Ariane 5 case study
- Asynchronous behavior
- Cyclic behavior
- Tools
  - Evaluation & Conclusion
Summary of work

- **Model of**
  - The spacecraft behavior (mission management, **asynchronous**)
  - The ctrl / cmd SW (control / command, **cyclic behavior**)
    - Multitasking
    - CPU consumption
  - The **environment**
    - Avionics (valve, pyrotechnic commands, ground control center, …)
    - Communication bus

- **Validation of all the specified properties**
  - By simulation
  - By proof]

Proof is a complement of test but does not replace test
  - Detection of intentional bugs of the model
Cyclical behavior

- For “low” reactivity needs
  - Use of the synchronous hypothesis
  - Control command described using SCADE
  - 60% to 90% of the software

- For “high” reactivity needs
  - When the synchronous hypothesis is violated
    (Required reactivity < basic cycle duration)
  - Asynchronous semantics
  - Important effort of modeling (several thousands of bus transfers)

⇒ OMEGA UML for hard point analysis
Respect the OMEGA syntax/semantics

Powerful debugging facilities

Precise interpretation of results requires some knowledge of tool internals
- No automated feedback from the VERIMAG tool towards the UML tool
- Objects of the IF model are visible, even if not defined by the user
- Slow for big scenario (>30Mb, >30000 transitions)
  ⇒ Hard to use in practice for “cyclical” debugging (several hours)
Property description

Use of observers

- Powerful
  - Untimed and timed properties
  - Intrusive properties ("integration tests")
  - Non intrusive properties ("validation tests")

- Property description formalism very easy to understand
  - Finite state machine
  - Defined in OMEGA UML syntax and semantics
  - Intuitive concept of real time
IF proof tool

- **Answers all the user needs**
  - Very quick result
  - In case of non satisfied property, computation of a failed scenario
  - All properties proved

- **Same defaults as the simulator**
  - No feedback from the VERIMAG tool toward the UML tool (for the computed failed scenario)
  - No usable failed scenario for big models
  => Debugging not easy
General conclusion

- OMEGA UML allows to model the real time behaviors of a spacecraft
- Validation early in the development cycle
  - Improve the software quality
  - Decrease the software development costs
- No link with SCADE

We need industrial tools to use OMEGA UML
Questions ?