Component-based Engineering for Embedded Systems

Code Generation and Simulation from Heterogeneous Models
- State of the practice
- State of the art
- Challenges

David LESENS
EADS SPACE Transportation
david.lesens@space.eads.net
EADS SPACE Transportation: Major Products and Capabilities

Launchers:
- Ariane
- Soyuz
- Rockot

Ballistic Missiles
- Missile Defence

Orbital Systems:
- Columbus
- ATV
- Operations
- Atmospheric Reentry Systems

System Design

System Integration & Production

Future Launchers

Real time critical embedded systems
Overview

• Introduction: Why shall we model software components?
• Why shall we consider heterogeneous systems?
• How shall we model software components?
• Scheduling policy
• Conclusion
Why shall we use a model?

System studies

Immediate correction

Validation
- Simulation
- Proof

Refinement

Automatic test generation
- Test replay

Decrease the number of late errors

Model of the software

Real-time Design Development

Functional Validation Integration

Unitary tests

Automatic code generation

Immediate correction

Test replay
Software development with Automatic Code Generation

- Automatic test replay
- Coverage objective
- Automatic test generation

White box testing → Model

Automatic code generation

Black box testing

Functional Validation

Code

Abstract Interpretation

EADS SPACE TRANSPORTATION
Overview

• Introduction: Why shall we model software components?
• Why shall we consider heterogeneous systems?
  – Event driven components
  – Control / command (algorithmic) components
  – Control / command (non algorithmic) components
• How shall we model software components?
• Scheduling policy
• Conclusion
Spacecraft management deals with timed sporadic events

⇒ Use of timed asynchronous semantics
Spacecraft control / command

- **Sensors**
  - Acquisition of measurement

- **Navigation**
  - Where am I?

- **Guidance**
  - Where shall I go?

- **Control**
  - Compute the commands

- **Actuators**
  - Send commands to actuators

Cyclic process
Period = some milli-seconds

**Spacecraft control command**

**Globally Asynchronous / Locally Synchronous (GALS)**
Overview

• Introduction: Why shall we model software components?
• Why shall we consider heterogeneous systems?
• How shall we model software components?
  – State of the practice
  – State of the art
    • OMEGA project 2002 - 2005
  – Challenges
    • ASSERT project 2004 - 2007
    • ACG (Automatic Code Generation) project 2003 - 2006
• Scheduling policy
• Conclusion
Spacecraft management / mission phases

Spacecraft management deals with timed sporadic events

⇒ Use of timed asynchronous semantics
• Compute the commands
  + Control
  - GNC
  - SW

• Acquisition of measurement
  + Sensors

• Where shall I go?
  + Guidance

• Where am I?
  + Navigation

• Send commands to actuators
  + Control
  - Actuators

Spacecraft control / command

Cyclic process
Period = some milli-seconds

Globally Asynchronous / Locally Synchronous (GALS)

SCADE modelling

MATLAB / SIMULINK modelling

SCADE modelling

EADS SPACE TRANSPORTATION
Modelling of control low

Algorithms feasibility

Simulation

GNC SIMULINK model

Environment SIMULINK model

Algorithm development

Rapid prototyping

Processor

Automatic code generation

State of the practice

2006 / 2007

ACG

February 2006
Modelling of a synchronous/asynchronous SW Vehicle management

Software specification

SCADE / UML model

Simulation & proof

Software specification verification

Environment SCADE / UML model

Automatic code generation

Processor

State of the practice

State of the art

February 2006

ACG
Example of proved properties with SCADE

⇒ Simulation + Proof by model checking

- A CAM test can only be triggered by a “red button” signal
  - true_after_false( CAM_TEST_TRIG ) ⇒ RED_BUTTON
  - No assertion is required from the environment to satisfy this property

- When the initialisation of the two MSU chains is correct, they can not triggered both a CAM at the same time
  - #( MSU1_CAM_TRIG , MSU2_CAM_TRIG )
  - It is satisfied only when the initialisation of the 2 MSU is correct
  - cam_arm( SWITCH_ON_MSU1, ARM_MSU1, SWITCH_ON_MSU2, ARM_MSU2, RED_BUTTON )
Example of timed automaton in **OMEGA**

### Method called by an external entity

- **Idle**
  - Reception of an event: `Anomaly() / clock.reset()`
  - Clock reset
  - **Wait**
    - **Clock armed**
      - **Time out**
        - **Guard**
          - **Emission of an event**
            - **Error**
              - **Guard**
                - **Return to the caller**
            - **End**
Complete validation

Interconnection between models / platforms

Software
SCADE / UML
model

GNC
SIMULINK
model

Environment
SCADE / UML
/ SIMULINK
model

Simulation

Simulation

Automatic code
generation

Rapid prototyping

Feasibility

Flight code

ASSERT

2007
Different types of behaviour

- GNC, thermal algorithms
- Cyclic synchronous
- Asynchronous

How to make the glue (static and dynamic)?

⇒ Use of an AADL modelling + proof
Overview

- Introduction: Why shall we model software components?
- Why shall we consider heterogeneous systems?
- How shall we model software components?
  - Scheduling policy
    - Mixing synchronous / asynchronous behaviour
    - Maintainability
- Conclusion
Multitasking for Critical Software
( Current state of the practice )
- Multitasking preemptive with fixed priority
- Rate Monotonic Analysis (RMA)

Well adapted for **Control / Command** (as a **launcher**)
But what about a **more complex system** (as a **spacecraft**)

Deterministic
Critical Real Time Embedded Software

~ 20 states
Flight duration ~ 1 hour

Dynamic creation/deletion of
• objects
• processes
forbidden

> 10000 states
Flight duration > 3 months

✓ Complex asynchronous system
✓ Need of “during the flight” maintenance
RMA to EDF

Guidance, Navigation Control

Cyclical

But what happens in case of missing deadline?
EDF + Criticality Level

- **Cyclic missing deadline**
  - For thermal control ⇔ No impact on the mission → 4
  - For Guidance ⇔ Lost of precision → 3
  - For Control ⇔ Destruction of the spacecraft → 2

- **Acyclic missing deadline**
  - Telecommand ⇔ Lost of precision → 3
  - Post contact boost ⇔ Destruction of the station → 1

⇒ *Take into account the deadline and the criticality*
What can we do when a task does not meet its deadline?

Non critical task

Critical task

Allocated time

Deadline

Missing deadline

Catastrophic missing deadline

⇒ The faulty non critical task has to be killed (or at least suspended)
OASIS approach versus ASSERT approach

Non critical task

Allocated time

Deadline

Critical task

Missing deadline

The task is killed as soon as allocated time is over

OASIS approach

The task is killed as soon as it can avoid another task to meet its deadline

ASSERT approach
Overview

• Introduction: Why shall we model software components?
• Why shall we consider heterogeneous systems?
• How shall we model software components?
• Scheduling policy
• Conclusion
Conclusion

Different types of behaviour

- GNC, thermal algorithms
- Cyclic synchronous
- Asynchronous

- ACG
- SIMULINK
- SCADE
- UML
- C/ADA code
- Building Blocks

- Middleware AADL modelling
- Mathematical BB
- Mission management
- TM / TC

Repository

Processor

Page 26  Component-based Engineering for Embedded Systems – 07/07/2005