VERIFICATION & VALIDATION OF LAUNCHER FLIGHT SOFTWARE

Ada-Europe
International Conference on Reliable Software Technologies
12/06/2019 – Warsaw
Julien GRAND
**ARIANE 5**

- Quite expensive
- Limited versatility

**ARIANE 6**

- 40% cost decrease
- High versatility

- More complex and cheaper flight software

**OBJECTIVE OF HIGH COST EFFICIENCY**

**NOT ADAPTED TO CURRENT MARKET**
1

ARIANE 6 FLIGHT SOFTWARE
ARIANE 6 FLIGHT SOFTWARE

Functions

- Engines control
- Trajectory control
- Manage the sequential events (e.g. stage release)
- Attitude control
- ...
ARIANE 6 FLIGHT SOFTWARE

Specificities

Embedded

- The software is embedded into the launcher computer. This implies limited CPU and memory resources
  - Example: Few megabytes of memory available

Real-Time

- The software is constrained by time. It shall deliver correct results in imposed deadlines
  - Example: Reactivity of a few milliseconds

Critical

- Failure of the software might have huge material damages
  - Example: Failure may result in the launcher destruction
SOFTWARE DESIGN

Functional architecture

Each component is as independent as possible
⇒ Validated independently
⇒ Decrease of cost
DEVELOPMENT METHOD

V-Cycle

System requirements

Functional definition

Code

Qualification phase

Validation phase

Unit tests

REAL HARDWARE
REAL AVIONICS

EMULATED/REAL HARDWARE
EMULATED AVIONICS

EMULATED HARDWARE

IN-HOUSE TOOL
Partial automatic code generation

Go for launch
DESIGN PRINCIPLES

Synchronous approach

Cyclic software

⇒ Functional and Real-Time behaviour independent from WCET

WCET: Worst Case Execution Time
DESIGN PRINCIPLES

Synchronous approach

Extension to multi-task architecture

⇒ Functional and Real-Time behaviour independent from WCET

WCET: Worst Case Execution Time
SOFTWARE DESIGN

Code architecture

Design principle
✓ Synchronous

specification

IN-HOUSE TOOL

Code generator

specification

Ariane 6 SysML model

Software developers

Generic library

Generic code
✓ Mathematical Library
✓ Generic sequential

Code automatically generated

Instantiation of generic sequential

Manual non-algorithmic code

Manual algorithmic code

Manual non-algorithmic code

Manual algorithmic code

Mathematical Library

Generic sequential
VALIDATION METHODS
VALIDATION METHODS

Objectives:

• ensure the correctness of the flight software

• decrease the validation costs
VALIDATION METHODS

Means to perform the validation

• Emulated hardware
  ✓ Faster than real on-board computer
  ✓ Easier debugging
  ⇒ Used for tests preparation

• Real hardware
  ✓ Fully representative of the flight
  ⇒ Used for formal tests

Synchronous principle
  ⇒ Identical behaviour on real and emulated hardware when WCET are met

WCET: Worst Case Execution Time
GENERIC CODE VALIDATION

- Generic code
  ✓ Mathematical Library
  ✓ Generic sequential
- Code automatically generated
- Instantiation of generic sequential
- Manual non-algorithmic code
- Manual algorithmic code

Flight software code architecture
GENERIC CODE VALIDATION

Validation on representative instantiations of the generic code

Example of the mathematical library

- Generic code for matrix operations

⇒ Validation on instantiations for matrix of sizes:
  - 1x1 (specific case)
  - 3x1 (column matrix)
  - 1x3 (row matrix)
  - 5x3
GENERATED CODE VALIDATION

Flight software code architecture

- Generic code
  - ✓ Mathematical Library
  - ✓ Generic sequential

- Code automatically generated

- Instantiation of generic sequential

- Manual non-algorithmic code

- Manual algorithmic code
GENERATED CODE VALIDATION

Flight software criticality: B

In-house code generator criticality: D

⇒ To decrease the cost

⇒ The generated code shall be validated
GENERATED CODE VALIDATION

Code generation principle

- SysML model
  - IBM Rhapsody
  - API Rhapsody
- Domain Specific Language
- In-house tool
- Ada generated code

Review of consistency between SysML model and Domain Specific Language model

Not trusted
GENERATED CODE VALIDATION
Finite State Machines

- Init
  - Off
  - On
- Nominal
- Failed

Flattening

- Init
  - Off_Nominal
  - On_Nominal
  - Off_Failed
  - On_Failed
GENERATED CODE VALIDATION

Finite State Machines

Init

Off
On

Nominal
Failed

Algorithm 1

Flat Finite State Machine

Algorithm 2

Flat Finite State Machine

consistency tests
# GENERATED CODE VALIDATION

<table>
<thead>
<tr>
<th>Generic code</th>
<th>Code automatically generated</th>
<th>Instantiation of generic sequential</th>
<th>Manual non-algorithmic code</th>
<th>Manual algorithmic code</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Mathematical Library</td>
<td>✓ Generic sequential</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Flight software code architecture**
GENERATED CODE VALIDATION

Sequential events validation

Examples of sequential events: stage release, fault management…

- Specification: timed constraints on sequential events
- Validation by simulation
GENERATED CODE VALIDATION

Sequential events validation

Example of sequential specification

```ada
monitor End_Of_Thrust_Event;
when End_Of_Thrust_Event:
  wait 10s;
  Separate_Stage;
```

Implementation

```ada
event Separate_Stage;
event End_Of_Thrust_Event;
Separate_Stage >= End_Of_Thrust_Event + 10s;
```

- **End_Of_Thrust_Event** is raised by a complex algorithm
  - stub the algorithm
**GENERATED CODE VALIDATION**

## Sequential events validation

### Example of sequential specification

```plaintext
DSL
monitor End_Of_Thrust_Event;
when End_Of_Thrust_Event:
    wait 10s;
    Separate_Stage;

Code generator

Algorithms Stubs

Implementation

event Separate_Stage;
event End_Of_Thrust_Event;
Separate_Stage >= End_Of_Thrust_Event + 10s;

Stubs definition

Code generator

Example of sequential specification

at 100s:
    raise End_Of_Thrust_Event;

Sequential Code

Simulation

Traces

Traces verification
```
MANUAL CODE VALIDATION

Non-algorithmic code

Flight software code architecture

- Generic code
  - ✓ Mathematical Library
  - ✓ Generic sequential
- Code automatically generated
- Instantiation of generic sequential
- Manual non-algorithmic code
- Manual algorithmic code
MANUAL CODE VALIDATION

Non-algorithmic code

⇒ Open-Loop approach

Inputs
4.001
(1.0, 2.0, 3.0)

Functionality under test

Outputs
42

Expected Outputs
42

42
MANUAL CODE VALIDATION

Algorithmic code

- Generic code
  - ✓ Mathematical Library
  - ✓ Generic sequential

- Code automatically generated

- Instantiation of generic sequential

- Manual non-algorithmic code

- Manual algorithmic code

Flight software code architecture
MANUAL CODE VALIDATION

Algorithmic code

A classical open-loop approach is not possible for algorithmic code
MANUAL CODE VALIDATION

Algorithmic code

A classical open-loop approach is not possible for algorithmic code

Launcher Attitude → Launcher Control → Thrust Direction

Simulators
Aerodynamics
Engine thrusts
Gravity
...
MANUAL CODE VALIDATION

Algorithmic code

Closed-Loop approach

- Executed on emulated hardware
GLOBAL VALIDATION

Real On-Board computer / Emulated Avionics

- Validation of representative instantiations
- Validation by consistency check
- Validation by simulation
- Open-Loop validation
- Closed-Loop validation

Generic code
- ✓ Mathematical Library
- ✓ Generic sequential

Code automatically generated

Instantiation of generic sequential

Manual non-algorithmic code

Manual algorithmic code
CONCLUSION

• Different kinds of code

• Adaptation of validation methods

• Decrease of costs
THANK YOU FOR YOUR ATTENTION

Julien GRAND
E-mail: julien.grand@ariane.group