New Methodology to Develop Certified Safe and Secure Aeronautical Software - An Embedded Router Case Study

A. Varet    N. Larrieu

Ecole Nationale de l’Aviation Civile (ENAC), Toulouse, France

30th Digital Avionics Systems Conference, October 16-20, 2011
Evolution in aeronautical needs

New communication needs... for aircraft, companies & passengers, for instance:

- New ATC services, such as CPDLC,
- New usages for passengers, such as webmail...

...but future technical challenges

- Heterogeneity of aeronautical communications
- Higher throughputs
- Safe and secure multiplexing on shared links
Safety in aeronautical world

Definition
"The safety is the protection from the event or from exposure to something that causes health or economic losses."

Standards and "levels"

• The DO-178B defines 5 classes of increasing assurance, from the lowest DAL-E to the highest DAL-A
• Soon, the DO-178C will extend the DO-178B with formal method complements
• The safety validation process is called “certification” for a software, “qualification” for a helper tool
Safety in aeronautical world

Example of some associated tool
Security in aeronautical world

Definition
"The security takes into account the actions of people attempting to cause destruction."

Standards and "levels"

- The Common Criteria v3.1 (CC31) defines 7 main classes of increasing assurance, from the lowest EAL-1 to the highest EAL-7
- The security validation process is called “evaluation”

Example of some associated tool
PikeOS(r) is a separation microkernel with secure Inter-partition communication (S-IPC)
How to build a software with safety & security?

Our solution
We propose an adequate methodology to develop aeronautical embeddable code with:

- minimization of the design & development delays
- minimization of the certification and evaluation efforts
- maximization of the safety and security assurance levels of the binary
Outline

Introduction

Methodology
   Principles
   Application on the Router SNG Case-study
   Tool chain synthesis & gains

Conclusion
Methodology principles
Our aeronautical embedded IP-based Router case study

Embedded Secure Next Generation Router
Multiplex streaming from crew and passengers

On-ground SNG Router
Streams demultiplexed for controllers, companies and ISP
(1) Partitioning

**SNG Router Partition classes**
- Functional decomposition of the system into partitions:
  - Pfr4: IPv4 Routing,
  - Piface: Hardware drivers,
  - Pse: security of streams...

**SNG Router Partition objects**
- Runnable instances of the partition classes
  - 3 network interfaces: Piface,
  - 1 common partition: Pfr4,
  - 1 securizing partition: Pse.
(2) Design

Models for the Pfr4 partition class

Simulink

- Matlab toolkit
- Dynamic system modelisation
- Formal Checking features

Stateflow

- Simulink toolkit
- State Machine modelisation
- Formal Checking features too
(3) Transforming

The partition class models are converted into C source code

- In 5 minutes, Gene-Auto generates 21 .c and 24 .h files.

```
$ ls
Common_functions.c
Common_functions.h
CurrentPacketMem.c
CurrentPacketMem.h
Filters.c
Filters.h
Forward_Processing.c
Forward_Processing.h
GACommon.h
GATypes.h
pfr4.c
pfr4.h
pfr4_param.c
pfr4_param.h
pfr4_types.h
Route.c
Route.h
Schedule_4_to_1.c
Schedule_4_to_1.h
Test_One_Filter.c
Test_One_Filter.h
...
```

```c
#include "Test_One_Filter.h"

/* External function of chart Test_One_Filter_init */
void Test_One_Filter_init(t_Test_One_Filter *p_Test_One_Filter)
{
    // Initialisation function...
}

/* Computation function of chart Test_One_Filter */
void Test_One_Filter_compute(t_Test_One_Filter *p_Test_One_Filter)
{
    // Computation function...
}
```

Gene-Auto: qualifiable transformer
(4) Glueing

“Glue” requested to integrate the generated files with the RTOS

- The kernel needs to know the entry-point + when to use it
- The models I/O must be linked to the ones of the kernel
(5) Compiling

A single binary is built for the class
(6) Integrating

A ROM image is generated, containing:

- The Sysgo PikeOS separation kernel
- All binaries for each class
- Set of objects & their scheduling
- Set of authorized Inter-Partition Communications
- Some removable debug materials...
(7) Running

The ROM file may be uploaded to an embedded target or validated with an emulator.
Tool chain synthesis & other available tools
Benefits of our methodology

Development

- Using High-level Models $\Rightarrow$ less complexity, correction early
- Making model-based library source code $\Rightarrow$ allow different platforms and infrastructures to be tested
- Functional decomposition $\Rightarrow$ isolate the problems, ease the instrumentation, reuse the models
- Code production automation $\Rightarrow$ avoid repetitive low-level tasks, reduce copy-and-paste bugs
- Compatible with legacy source code
Benefits of our methodology

Certification (safety)
Task reduction when tools are qualified:

- Gene-Auto $\Rightarrow$ A.2.6 “The source code is developed”
- Gene-Auto + Cc qualification for DAL-A $\Rightarrow$ 16 out of 66 tasks avoided or reduced

Evaluation (security)
PikeOS:

- S-IPC $\Rightarrow$ easier to prove security functionalities “independently” for each partition
- EAL3+ $\Rightarrow$ kernel evaluation done
Conclusions & Perspectives

Conclusions
Our methodology can improve safe and secure software development. The SNG Router case study extends aeronautical network capabilities.

Influenced costs
- time
- money
- human resources

Influenced phases
- development
- safety certification
- security evaluation
Conclusions & Perspectives

Perspectives

- Additional tool chains to complete formal checking toolkit
- Checking of compatibility between methodology and DO178C
- Extension of Network Security Requirements for our Router SNG (into Pse partition)
Thanks

Thanks for your attention!

Do you have any questions?
How Open-Source software improves the gains?

- Appropriation of the code for new programmers in the project
- Deployment
- Adaptability
- Reusability
- Maintainability
- Long term support (no dependence from providers)
- Licenses management, infinite simultaneous users
What are the gains, conforming to the DO-178B DAL-A?

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2.6</td>
<td>The source code is developed</td>
</tr>
<tr>
<td>A.2.7</td>
<td>The Executable Object Code is produced</td>
</tr>
<tr>
<td>A.5.1</td>
<td>The source code complies with low-level req.</td>
</tr>
<tr>
<td>A.5.2</td>
<td>. . . complies with software architecture</td>
</tr>
<tr>
<td>A.5.3</td>
<td>. . . is verifiable</td>
</tr>
<tr>
<td>A.5.4</td>
<td>. . . conforms to standards</td>
</tr>
<tr>
<td>A.5.5</td>
<td>. . . is traceable to low-level req.</td>
</tr>
<tr>
<td>A.5.6</td>
<td>. . . is accurate and consistent</td>
</tr>
<tr>
<td>A.6.3</td>
<td>Executable Object Code complies with low-level req.</td>
</tr>
<tr>
<td>A.6.4</td>
<td>. . . is robust with low-level req.</td>
</tr>
<tr>
<td>A.7.1</td>
<td>Test procedures are correct (partly)</td>
</tr>
<tr>
<td>A.7.2</td>
<td>Test results are correct, discrepancies explained (partly)</td>
</tr>
<tr>
<td>A.7.4</td>
<td>Test coverage of low-level req. is achieved</td>
</tr>
<tr>
<td>A.7.5-7</td>
<td>. . . of software structure is achieved</td>
</tr>
</tbody>
</table>