Model of an ADIRU in AADL
Revisiting Boeing 777-2H6ER bug
AADL Winter’14 meeting

Jérôme HUGUES, ISAE
Initial work by GONG Huawei (MS/EMS’13)
with support from Grant Award W911NF-13-1-0459
Outline

1. Introduction
2. Boeing 777-2H6ER ADIRU
3. The Model of ARINC 653 Architecture
4. The Implementation of the Model
5. Conclusion
1. Introduction
2. Boeing 777-2H6ER ADIRU
3. The Model of ARINC 653 Architecture
4. The Implementation of the Model
5. Conclusion
Introduction

■ Objectives

1. The ADIRU (Air Data Inertial Reference Unit) of the Boeing 777-2H6ER will be modeled in AADL. Its fault caused an hazardous accident to Malaysian Air flight 124 in 2005,
2. The failing process and how to avoid the accident will be demonstrated with SHM (Software Health Management) technique used in the model.

■ Why SHM technique is necessary?

1. The complexity of the software in modern embedded system is greatly increasing.
2. The runtime fault in software, which appears only on very special conditions, has more chance to occur.
3. SHM technique is effective to deal with such kind of errors.
1. Introduction
2. Boeing 777-2H6ER ADIRU
3. The Model of ARINC 653 Architecture
4. The Implementation of the Model
5. Conclusion
About Boeing 777-2H6ER ADIRU

> (from ATSB report 200503722)

- Multiple levels of redundancy.
- Work without maintenance with one fault in each FCA.
The ADIRU OPS versions up to and including version -07 contained a **latent software error** in the algorithm to manage the sensor set used for computing flight control outputs which, after the unit went through a power cycle, did not recognise that accelerometer number-5 was unserviceable. The status of the failed unit was recorded in the on-board maintenance computer memory, but that memory was not checked by the ADIRU software during the start-up initialisation sequence. The software error had not been detected during the original certification of the ADIRU and was present in all versions of the software. **The effect of the error was suppressed by other software functions in OPS version -03.** When the OPS version - 04 was released in December 1998, the software functions that suppressed the error were further revised to improve shop repair capability, **re-exposing the undiscovered latent problem.**
See ISIS-11-101 TR from Vanderbilt University

- Four modules
- Two types of ports
Outline

1. Introduction
2. Boeing 777-2H6ER ADIRU
3. The Model of ARINC 653 Architecture
   1. AADL models
   2. Model analysis
4. The Implementation of the Model
5. Conclusion
Follow ARINC653 annex document +AADLv2.1
  » Using OSATE2.<rand ()> version from this summer
  » Semantic analysis + analysis plug-ins

ARINC 653 verification plugs-ins
  » Check connections
  » Check Configuration - Major Frame Correctness.
  » Check Configuration - Properties of Memory Components
  » Check Configuration - Dimensioning of Memory Components.
  » Check Configuration – Partitions Bindings
  » Check Configuration - Partitions Executions
  » Check Configuration - Separation of Memory into Segments.
Accelerometer Module

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Thread acc</th>
<th>Thread acc_HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatch_Protocol</td>
<td>Period</td>
<td>Period</td>
</tr>
<tr>
<td>Priority</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Period</td>
<td>50 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>Compute_Execution_Time</td>
<td>20 ms..30 ms</td>
<td>20 ms..30 ms</td>
</tr>
<tr>
<td>Deadline</td>
<td>30 ms</td>
<td>30 ms</td>
</tr>
</tbody>
</table>

Boeing 777-2H6ER ADIRU Case Study
ADIRU Processor Module

### Properties

<table>
<thead>
<tr>
<th></th>
<th>Thread ADIRUp_solv</th>
<th>Thread ADIRUp_HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatch Protocol</td>
<td>Period</td>
<td>Period</td>
</tr>
<tr>
<td>Priority</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Period</td>
<td>50 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>Compute Execution Time</td>
<td>20 ms..50 ms</td>
<td>20 ms..50 ms</td>
</tr>
<tr>
<td>Deadline</td>
<td>50 ms</td>
<td>50 ms</td>
</tr>
</tbody>
</table>

Boeing 777-2H6ER ADIRU Case Study
Voter/Display Module

Boeing 777-2H6ER ADIRU Case Study
<table>
<thead>
<tr>
<th>Properties</th>
<th>Thread voter</th>
<th>Thread voter_HM</th>
<th>Thread display</th>
<th>Thread display_HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatch Protocol</td>
<td>Period</td>
<td>Period</td>
<td>Period</td>
<td>Period</td>
</tr>
<tr>
<td>Priority</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Period</td>
<td>50 ms</td>
<td>50 ms</td>
<td>50 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>Compute Execution Time</td>
<td>20 ms..50 ms</td>
<td>20 ms..50 ms</td>
<td>20 ms..50 ms</td>
<td>20 ms..50 ms</td>
</tr>
<tr>
<td>Deadline</td>
<td>50 ms</td>
<td>50ms</td>
<td>50ms</td>
<td>50ms</td>
</tr>
</tbody>
</table>
SHM Module

**Properties**

<table>
<thead>
<tr>
<th>Thread</th>
<th>DISPATCH_PROTOCOL</th>
<th>PRIORITY</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarmAggregate</td>
<td>1 Period</td>
<td>1</td>
<td>100 ms</td>
</tr>
<tr>
<td>diagnosisEngine</td>
<td>2 Period</td>
<td>2</td>
<td>100 ms</td>
</tr>
<tr>
<td>mitigationActor</td>
<td>3 Period</td>
<td>3</td>
<td>100 ms</td>
</tr>
</tbody>
</table>

**Compute Execution Time**

<table>
<thead>
<tr>
<th>Thread</th>
<th>DISPATCH_PROTOCOL</th>
<th>PRIORITY</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>systemHM_acc1</td>
<td>50 ms</td>
<td>1</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_acc2</td>
<td>50 ms</td>
<td>2</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_acc3</td>
<td>50 ms</td>
<td>3</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_acc4</td>
<td>50 ms</td>
<td>1</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_acc5</td>
<td>50 ms</td>
<td>2</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_acc6</td>
<td>50 ms</td>
<td>3</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_ADIRUp1</td>
<td>50 ms</td>
<td>1</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_ADIRUp2</td>
<td>50 ms</td>
<td>2</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_ADIRUp3</td>
<td>50 ms</td>
<td>3</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_ADIRUp4</td>
<td>50 ms</td>
<td>1</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_VDI</td>
<td>50 ms</td>
<td>2</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_VDC</td>
<td>50 ms</td>
<td>3</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_VDr</td>
<td>50 ms</td>
<td>1</td>
<td>100 ms</td>
</tr>
<tr>
<td>systemHM_action</td>
<td></td>
<td></td>
<td>100 ms</td>
</tr>
</tbody>
</table>

**Deadline**

- 100 ms
Status of the models

> Valid with OSATE 2.<random()}, to be updated to latest OSATE2, minor errors reported on latest release

> Model exploitation
  » All OSATE2 ARINC653 analysis plug-ins check passed
  » Code generated from Ocarina + POK targeting Qemu x86

> Miss a fully-featured graphical editor 😞

> Model available on AADLib github project,
  » See https://github.com/yoogx/AADLib/tree/master/examples/adiru

> Technical report available on demand
  » Describes the full model + current analysis status
  » Will be made public Q42014
1. Introduction
2. Boeing 777-2H6ER ADIRU
3. The Model of ARINC 653 Architecture
   1. AADL models
   2. Model analysis
4. The Implementation of the Model
5. Conclusion
Adding Health Monitoring concerns

» Needs to extend it with Health Monitoring capabilities
  » Note that HM is pure software!
  » Actual bug is: information on a failure not properly reused when restarting system, leading to misconfiguration

» Faulty state not part of the acc. but part of config. database

» How to model this in AADLv2 + EMV2 (+BA?) ?
  » EMV2 models failure propagation to a logging event
  » Must ensure this information is read when configuring system during reboot => behavioral contract on system
> As defined in ARINC653 annex document

Supported_Error_Code: type enumeration
(...,
    Hardware_Fault, -- process level

Supported_Process_Recovery_Action: type enumeration
(Ignore, Confirm, Partition_Stop, Process_Stop,
    Process_Stop_And_Start_Another, Process_Restart,
    Nothing, Cold_Restart, Warm_Restart);
-- XXX NEED a store/log/acknowledge/report fault option

> Annex document states: “The list of possible values is implementation dependent and can be modified by the system designer.”

» But this means modifying a standard property set ...

» Need an errata to address this part
Need to consider propagation of faults/errors using EMV2
   » Through careful definition of required model elements
   » Error types, propagation, occurrences, …

Here, interested in a situation where an accelerometer sends bogus values
   » Error types reduced to valid/invalid data
   » Occurrence rate deduced from product data sheet
   » Then propagation/handling deduced from architecture
Devil’s advocate: let’s bypass EMV2

> Can we avoid EMV2?

> Solution: whenever an accelerometer device is marked as FAILED, issue an event to corresponding logging entity
  » Rationale: in case of a failure of an accelerometer, a BITE will report the device as faulty, thus using ports is enough
  » Or this would be caught by some voting block somewhere
  » Software solution seems sufficient, as it is a logical issue
    • No hardware faults propagated.
    • “What happens in the accelerometer stays in the accelerometer.”
We need to check the following elements

» “In all situations, if an accelerometer is defined as faulty, it cannot be part of a running configuration”

» Running configuration ⇔ operational mode

Needs to make configuration algorithm visible

» Part of the BA, in conjunction with mode change policy (?)

» Transaction to update data state in configuration database

Note: configuration database is central in avionics

» How to model the link between modes and config. data ?

» Shall we build on the data mini-annex to support this?

» To be considered for ARINC653/664/IMA annexes?
Proof objective is split in two parts
- Step#1: all faults occurrences are logged
- Step#2: actual status is being used
- Transactional support ensures #2 and #1 are not concurrent

Step#1 is part of architectural validation, part of monitoring policy at system-level
- Hence new ARINC653 enumerators for logging all faults
- + checks on system topology towards database
  - And the need to make it visible as first-class citizen in system
- Could be a PSL observer thanks to ACL

Step#2 is part of system behavior .. to be continued …
Outline

1. Introduction
2. Boeing 777 ADIRU
3. The Model of ARINC 653 Architecture
4. The Implementation of the Model
5. Conclusion
> Using Ocarina + POK as indicators model is “sound”

> `pok-toolchain.pl` performs analysis, validation, code generation, compilation and execution
Adapting the Model for POK

- Initial model is distributed: update to run on one module
  - For this purpose each module has to be converted into process so that they can be bound to one processor.
<table>
<thead>
<tr>
<th>Source files</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc_code.c</td>
<td>void accxdataoutput (int* p, int event_in)</td>
</tr>
<tr>
<td></td>
<td>void achm_monitor (int in_data1, int in_data2, int in_data3, int in_data4, int in_data5, int in_data6, int* p1, int* p4, int* p6, int* p7, int* p8, int* p9, int recovery_action, int* p2, int* p3, int* p5, int* p10, int* p11, int* p12, int* p13)</td>
</tr>
<tr>
<td>adirup_code.c</td>
<td>void adirup_hm (int in_data1, int in_data2, int in_data3, int in_data4, int in_data5, int in_data6, int* output1, int* output2, int* output3, int* output4, int* output5, int* output6, int recovery_action, int* error_msg)</td>
</tr>
<tr>
<td></td>
<td>void solvex (int in_data1, int in_data2, int in_data3, int in_data4, int in_data5, int in_data6, int* p)</td>
</tr>
<tr>
<td>voter_code.c</td>
<td>void voter_hm (int in_data1, int in_data2, int in_data3, int in_data4, int* p)</td>
</tr>
<tr>
<td></td>
<td>void middle_value_calcx (int in_data1, int in_data2, int in_data3, int in_data4, int* p)</td>
</tr>
<tr>
<td>shm_code.c</td>
<td>void alarm_aggregator (int acc_error_msg, int adirup_error_msg, int voter_error_msg, int* errors)</td>
</tr>
<tr>
<td></td>
<td>void diagnosis_engine (int error_msg, int* diagnosed_error)</td>
</tr>
<tr>
<td></td>
<td>void mitigation_actor (char error_msg, int* recovery_action)</td>
</tr>
</tbody>
</table>
The functions are called in corresponding subprograms;

Execution Sequence of the threads

acc1 → acc2 → acc3 → acc4 → acc5 → acc6 → acc_hm → adirup_hm → solver1 → solver2 → solver3 → solver4 → vd_hm → vd1 → vd2 → vd3 → alarmAggregator → diagnosisEngine → mitigationActor → acc1...

Demo to be integrated in AADLib

» Need to update build scripts accordingly
1. Introduction
2. Boeing 777-2H6ER ADIRU
3. The Model of ARINC 653 Architecture
4. The Implementation of the Model
5. Conclusion
Conclusion

Achieved

1. The AADL model for the ADIRU done;
2. The model is successfully executed on top of POK;
3. The demonstration shows the effectiveness of the SHM technique developed for the model.

Work To Be Done

1. The source code for voter and display module are not finished yet.
2. Time stamp of the data from accelerometers are not used in the model. And it is preferred for more reliable fault detection.
3. TFPG (Timed-failure Propagation Graph) is not developed yet. This technique is necessary for more reliable diagnosis result.