Using CHARTER tools to develop a Safety-Critical Avionics Application in Java

JTRES 2012, Copenhagen, Denmark, 24-26 October 2012

Gosse Wedzinga
Klaas Wiegmink
Outline

• Avionics systems & challenges
  ● Increasing role of software
  ● Architectural evolution
  ● Certification aspects of avionics software

• CHARTER approach
  ● Overview
  ● CHARTER software life-cycle

• Evaluation of CHARTER approach
  ● Tools evaluated
  ● Safety-critical avionics application
  ● Assessment

• Concluding remarks
Avionics systems

- Avionics literally means “aviation electronics”
- Comprises all electronic systems designed for use on an aircraft, artificial satellites, and spacecraft
- An avionics system is safety-critical when its failure could result in loss of life or significant damage
- Present day avionics systems are increasingly based on computers and many functions are realized in software
### Federated architecture
- **One computer system for each unique function**
  - Line Replaceable Units (LRU’s)
  - Unique combination of hardware and software
- **Dedicated interconnections**
  - Point to (multi)point
- **Intrinsic functional isolation**

### Integrated Modular Avionics
- **One computer system for multiple distinct functions**
  - Generic processing modules
  - Independence between application and execution platform
- **Packet-switched network**
  - Virtual links
- **Functional isolation provided by time & memory partitioning**
Architectural evolution

Impact of IMA

● Advantages
  ● Reduced space, weight, and power (SWaP)
  ● Application portability
    – Independent component development (applications, modules)
    – Reduced obsolescence issues
  ● Reduced spares inventory
  ● …

● Challenges
  ● Integration responsibility
  ● IPR issues
    – Multiple suppliers on one platform
  ● Complexity of configuration
    – Tables define resource allocation to applications
Certification aspects of avionics software

- **EUROCAE document ED-12: Software Considerations in Airborne Systems and Equipment Certification**
  - Guidance for production of software for airborne systems
  - Objectives of software life-cycle processes
  - Activities for satisfying the objectives
  - Descriptions of the compliance evidence
- Emphasis on development assurance
  - Requirements-based development
  - Verification (incl. testing)
- Increasing effort with increasing software level
  - Software level is input from system safety assessment

- **Revision C (January 2012)**
  - New supplements, e.g., object-oriented technologies, model-based development, formal verification
Certification aspects of avionics software

- **ED-12 Software levels**

<table>
<thead>
<tr>
<th>Level</th>
<th>Aircraft failure condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Catastrophic</td>
<td>Loss of airplane, multiple fatalities</td>
</tr>
<tr>
<td>B</td>
<td>Hazardous</td>
<td>Damage to airplane, excessive workload, some passengers injured (incl. fatal)</td>
</tr>
<tr>
<td>C</td>
<td>Major</td>
<td>Reduction in airplane capabilities, increased workload, passengers distressed/injured</td>
</tr>
<tr>
<td>D</td>
<td>Minor</td>
<td>Little effect on operation of airplane and crew workload, some physical discomfort</td>
</tr>
<tr>
<td>E</td>
<td>No effect</td>
<td>No effect on operation of airplane or crew workload</td>
</tr>
</tbody>
</table>
CHARTER approach

Critical and High Assurance Requirements Transformed through Engineering Rigour

2009 - 2012
CHARTER project overview

Goal

- Improve software development process for safety-critical embedded systems: reducing cost & increasing quality

Approach

- Apply model-based development
- Use as programming language Real-Time Java augmented with Java Modeling Language (JML) specifications
- Apply Rule-Driven Transformation (RDT) technique
  - Transform UML model elements into Java source code
  - Transform bytecode into machine code
  - Potentially certifiable
- Provide tools for formal verification and automated test case generation
CHARTER software life-cycle

Software Development

- Software Requirements
  - Artisan Studio
- Software Design
  - Artisan Studio
- Software Coding
  - Code Generator
- Integration
  - javac
  - JamaicaVM Builder

Software Verification

- Software Reviews & Analyses
  - ResAna
  - KeYFloat
  - VerCors
- Software Testing
  - KeYTestGen
  - JUnit

Tools

NLR - Dedicated to innovation in aerospace
## Evaluation of CHARTER approach

<table>
<thead>
<tr>
<th>Tool</th>
<th>Activity</th>
<th>Evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artisan Studio Code Generator Add-in</td>
<td>Coding</td>
<td>✓</td>
</tr>
<tr>
<td>JamaicaVM Builder</td>
<td>Building</td>
<td>*</td>
</tr>
<tr>
<td>ResAna</td>
<td>Loop bound analysis</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Heap consumption analysis</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Stack size analysis</td>
<td>-</td>
</tr>
<tr>
<td>VerCors</td>
<td>Verification of concurrent data structures</td>
<td>-</td>
</tr>
<tr>
<td>KeYFloat</td>
<td>Analysis of floating point computations</td>
<td>-</td>
</tr>
<tr>
<td>KeYTestGen</td>
<td>Test case generation</td>
<td>✓</td>
</tr>
</tbody>
</table>

* Machine code generator was implemented for the ARM architecture
Safety-critical avionics application

Environmental Control System (ECS)
Safety-critical avionics application

ECS Demonstrator Configuration

ECS Avionics System
- RT Java ECS Application
- JamaicaVM
- ARINC-653 RTOS
- PPC-based HW platform

Network
- Control and Display
- ECS Plant Simulator
Assessment

- **Attribute: Productivity**
  - Metric: Effort in person-hours to complete each life-cycle process

- **Baseline**
  - Total effort for conventional development
    - Reference data from three similar projects coded in C
    - Establish average productivity for C
    - Similar number of Lines-of-Code in C and Java
  - Effort for each life-cycle process
    - Estimated percentage of total development effort

- **CHARTER**
  - Obtained from NLR administrative accounting system
  - Made corrections for
    - Omitted activities from actual ED-12 processes (+)
    - Unexpected activities (-)
### Assessment

- **Comparison of efforts (person-hours)**

<table>
<thead>
<tr>
<th>Process</th>
<th>Baseline</th>
<th>CHARTER</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Requirements</td>
<td>105.2</td>
<td>112.9</td>
<td>7.3</td>
</tr>
<tr>
<td>Software Design</td>
<td>210.4</td>
<td>178.5</td>
<td>-15.2</td>
</tr>
<tr>
<td>Software Coding</td>
<td>210.4</td>
<td>176.1</td>
<td>-16.3</td>
</tr>
<tr>
<td>Integration</td>
<td>105.2</td>
<td>116.5</td>
<td>10.7</td>
</tr>
<tr>
<td>Software Reviews &amp; Analyses</td>
<td>63.1</td>
<td>94.9</td>
<td>50.4</td>
</tr>
<tr>
<td>Low-Level Software Testing</td>
<td>252.5</td>
<td>69.5</td>
<td>-72.5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>946.8</strong></td>
<td><strong>748.4</strong></td>
<td><strong>-21.0</strong></td>
</tr>
</tbody>
</table>
Assessment

- **Software design** (-15%)
  - Unexpected: JML specification more effort (+)

- **Software coding** (-15%)
  - Code generation (-)
  - Use of Java (-)
  - Inelegant editing (+)
  - May include design effort (+)

- **Software reviews & analyses** (+50%)
  - Application of formal verification (ResAna)
  - Expected to earn (partially) back in other processes

- **Low-level software testing** (-70%)
  - Not all test cases could be generated by KeYTestGen

- **Total** (-20%)
  - Accounts only for processes supported by CHARTER tools
Assessment

Cautions

- **Estimated baseline figures**
  - NLR develops a wide variety of systems
    - Difficult to compare
    - Significant deviation in baseline metrics
  - Effort for each life-cycle process estimated using %

- **Measured CHARTER figures**
  - Errors in recording hours spent
  - Demonstrator is on a single sample

- **Absolute value of figures is limited but figures do indicate productivity improvement using CHARTER tools**
  - Demonstrations for other domains show similar tendency
Concluding remarks

- **CHARTER approach**
  - Model-based development
  - Real-Time Java with Java Modeling Language annotations
  - Rule Driven Transformation
    - model to source code
    - bytecode to machine code
  - Tool support for formal verification and low-level testing

- **Maturity of development tools at high level**
  - Based on existing commercial products

- **Maturity of verification tools need further improvement**
  - But potential to reduce effort is acknowledged

- **JML as a specification language requires getting used to**

- **Reduced effort, lower cost, increased quality**

- **For more info see: http://charterproject.ning.com/**