Dedicated to innovation in aerospace

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**

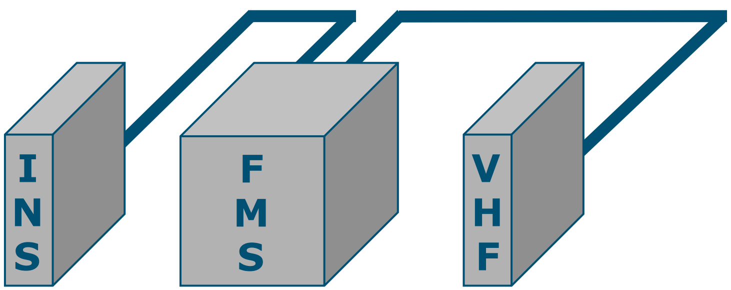
Avionics systems

- 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

Architectural evolution

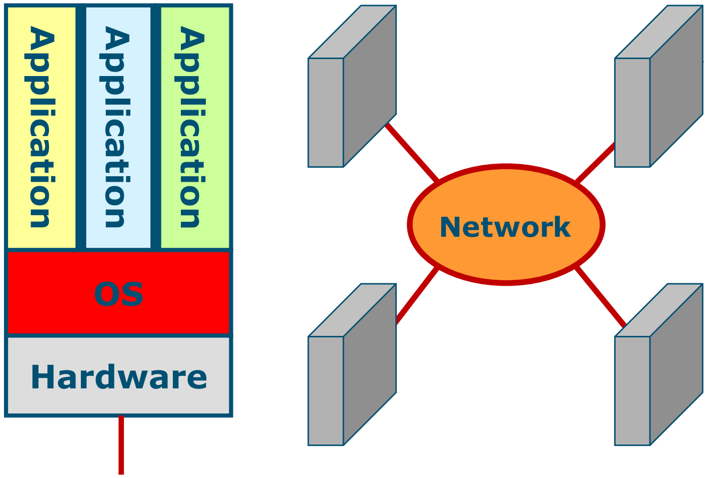
Federated architecture

- **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**

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

CHARTER approach

Critical and High Assurance Requirements Transformed through Engineering Rigour

2009 - 2012

ARTEMIS

aicas realtime

atego

CHALMERS

DUNDALK INSTITUTE of TECHNOLOGY
Institiúid Teicneolaíochta Dhún Dealgan

impronova

luminis
think broad, act bright

NLR

QRTECH
INNOVATIVE ENGINEERING

Radboud University Nijmegen

THE Open GROUP

UNIVERSITY OF TWENTE.

CHARTER project overview

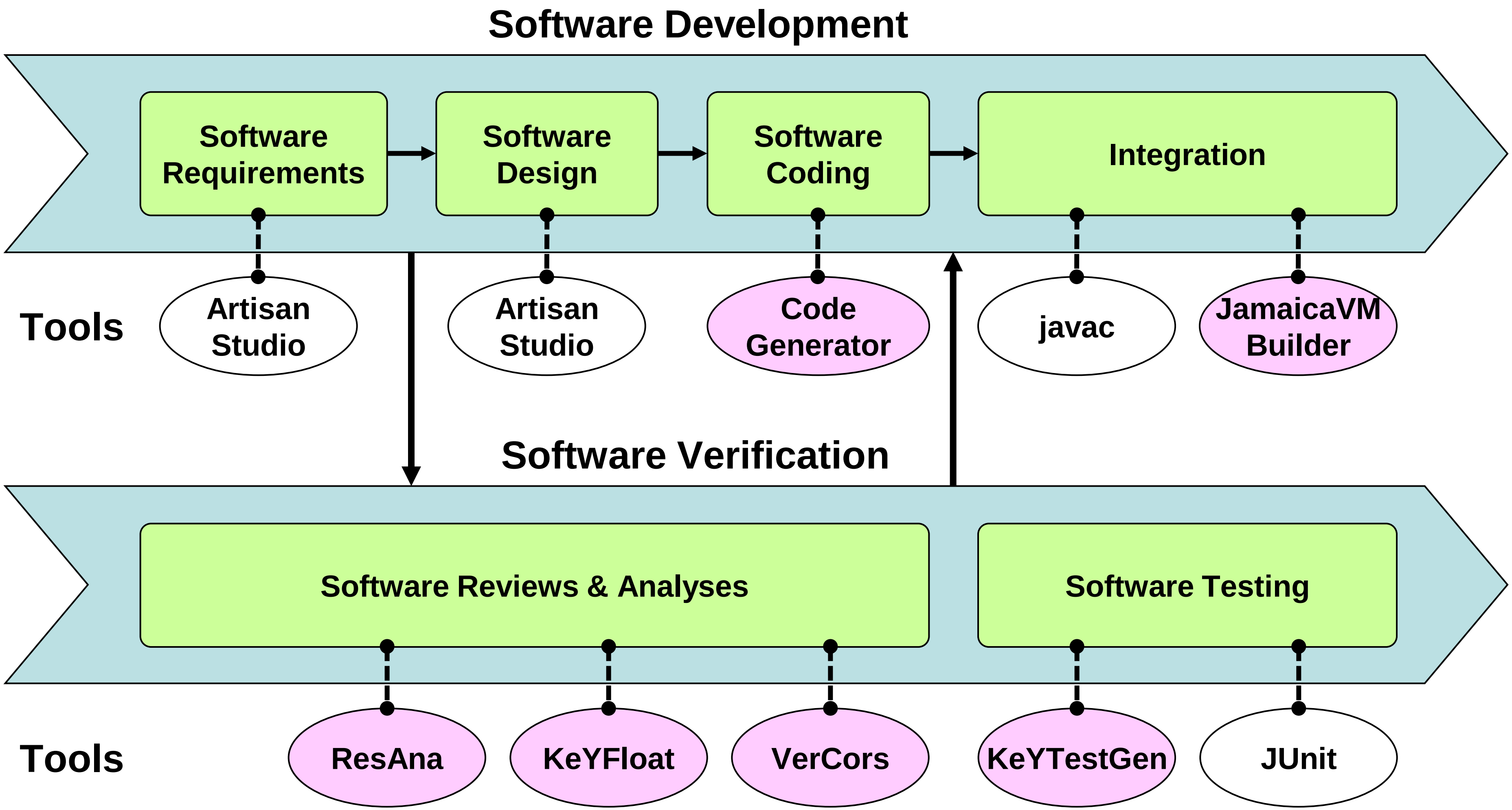
Goal

- 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



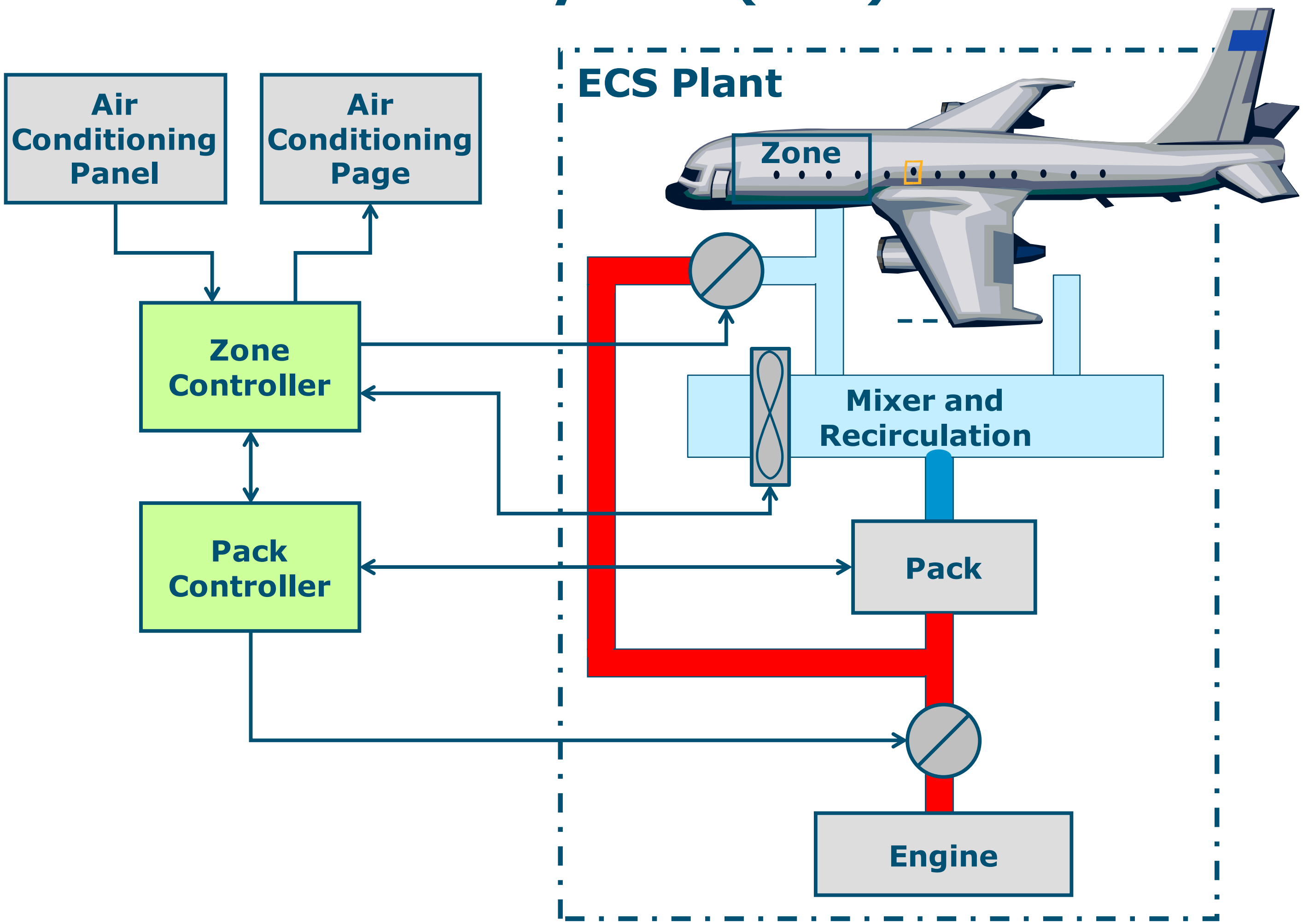
Evaluation of CHARTER approach

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

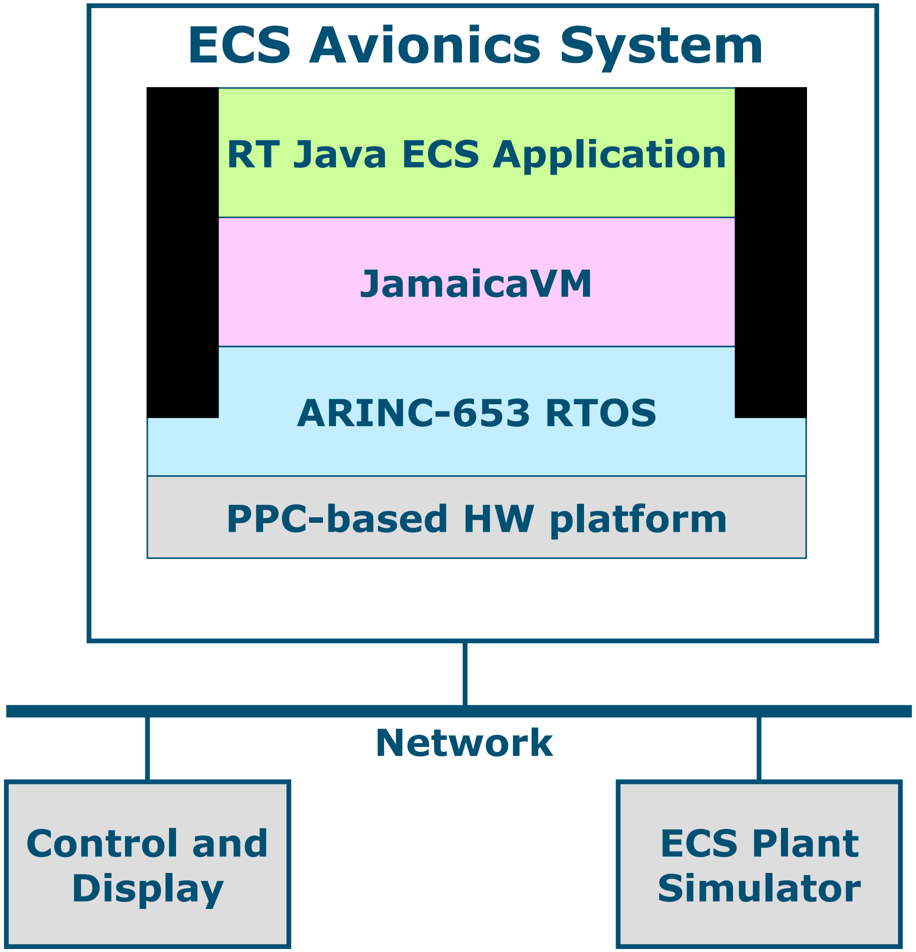
* Machine code generator was implemented for the ARM architecture

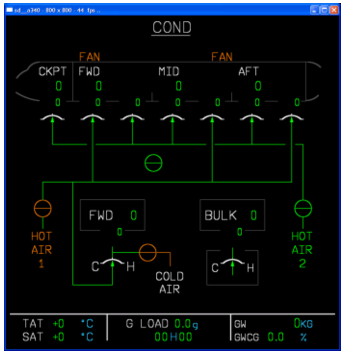
Safety-critical avionics application

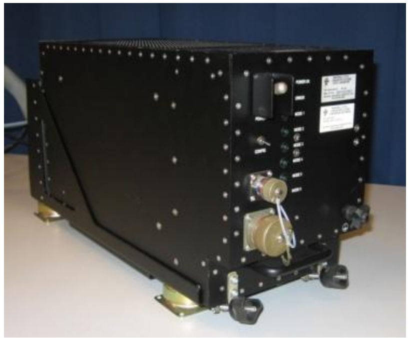
Environmental Control System (ECS)

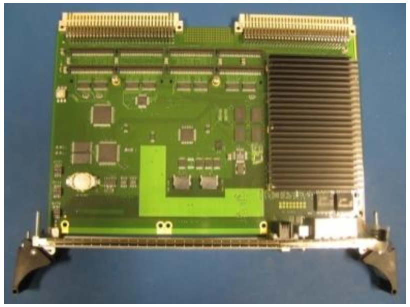


ECS Demonstrator Configuration









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)**

Process	Baseline	CHARTER	% Change
Software Requirements	105.2	112.9	7.3
Software Design	210.4	178.5	-15.2
Software Coding	210.4	176.1	-16.3
Integration	105.2	116.5	10.7
Software Reviews & Analyses	63.1	94.9	50.4
Low-Level Software Testing	252.5	69.5	-72.5
Total	**946.8**	**748.4**	**-21.0**

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/**