Open-DO & OSEE

Agile methods for producing High-Integrity software

Nicolas Setton, AdaCore
Stuttgart, May 24-25 2009
A bit of context

• This talk comes from the world of certified software in civilian avionics
  • also relevant for the military and aerospace industries
  • and others.

• Solving a family of problems in software certification
  • by introducing open source and Agile processes
  • and an Eclipse-based tool to implement them
I. Certification

II. Open-DO

III. OSEE
Certification in civilian avionics (1)

How to certify water:
- take a sample of the final product
- inspect the sample and check for harmful content
- the making process is not relevant

How to certify software for civilian avionics:
- we cannot judge software by a sample, so we have to consider the entire software
- we cannot prove the absence of bugs, so we need to test the software
- in order to gain confidence we need to look at how the software is made
Certification in civilian avionics (2)

- Certification is delivered by a Certification Authority
- For airborne software, all aspects of certification are described in DO178B/ED12B

\[
\text{Certification} = \text{DEMONSTRABLE DEPENDABILITY}
\]
DO-178B / ED-12B concepts

“Global” activities

Plan for software aspects of certification (PSAC)
Software Development Plan (SDP)
Software Verification Plan (SVP)
Software configuration management plan (SCMP)
Software quality assurance plan (SQAP)
Software requirements Specifications (SRS)
Software design standard (SDS)
Software code standard (SCS)

(etc)

“Local” activities

Development
Requirements management
Software analysis
Software verification
Code coverage
(etc)

Activities depend on the targeted assurance level

Level A: failure results in catastrophe (crash/multiple deaths)
to
Level E: software has no impact on the mission
DO-178B / ED-12B concepts

“Local” activities

Development
- Requirements management
- Software analysis
- Software verification
- Code coverage
- (etc)

“certification artifacts”
- Sources
- Objects
- Requirements
- Analysis reports
- Test cases
- Coverage reports

Need to guarantee traceability
Some problems with current practices

- Barrier of Entry
- Longevity and Availability
- The Big Freeze problem
  - the RTEMS anecdote
I. Certification

II. Open-DO

III. OSEE
The meeting of 3 worlds

Libre Open Source

Agile Lean

Critical Certification
Open-DO components

- Specialized workflows
- Tools supporting them

Activities Workflows

Document Templates

Qualifiable Tools

Certifiable Components

Education Material

- Open verification tools
- Open Development tools

- DO-178C examples
  - model based
  - formal methods
  - OOP

- Coding standards
  - C, Ada, ...
- DO-178:
  - PSAC, SDP, SVP, SCMP ...
- Other standards

Examples:
- 653 OS
- Light Database
- IP stack
- middleware
- Standard Classes
- ...

- Toy certifiable projects
- Training material

Friday, 26 June 2009
The Big Freeze problem

begin

- Requirements
- Design
- Implementation
- Verification
- Maintenance

certify
deliver

Friday, 26 June 2009
Continuous Integration

Continuous Certification

- Maintain a code repository
- Automate the build
- Automate the testing
- Automate the local certification activities (code coverage, traceability verification, etc)
- Every commit generates a rebuild and a test and the certification activities

☑ Early detection of defects
☑ The system is always release-ready
☑ The system is always certification-ready
Contributing

Open-DO contribution

Agile contribution

Standard contribution

source patch
+ regression test
+ certification elements

source patch
+ regression test

source patch

Greater confidence

Friday, 26 June 2009
The Certification Machine

- Maintain a code repository
- What certification activities can be automated?
- How to implement the *machine* that does this automatically?
- Automate the build
- Automate the testing
- Automate the local certification activities (code coverage, traceability verification, etc)
- Every commit generates a rebuild and a test
I. Certification

II. Open-DO

III. OSEE
OSEE

Open System Engineering Environment

- Eclipse project contributed by Boeing
- Apache Team (Phoenix, AZ)
- 5 years in development, 12 people full-time
- Not specific to DO-178
OSEE

“One of Eclipse’s best-kept secrets”
- Ralph Müller

• “OSEE is a tightly integrated environment designed to support lean engineering principles across a product’s full life-cycle in the context of overall systems engineering approach.”
OSEE

- An integrated tool set
- End-to-end traceability
- Variant configuration management
- Integrated workflows and processes
- A Comprehensive issue tracking system
- Deliverable document generation
- Real-time project tracking and reporting
- Validation and verification of mission software

CERTIFICATION

Friday, 26 June 2009
OSEE Data Model

Data object
- typed
- with inheritance

Key/Value pair
- base types

Artifacts

Attributes

Relations

Relates between artifact types
- typed
- multiplicity
OSEE Services

- Object-Oriented Persistence
- Session Mgmt & Authentication
- Version Control
- Access Control
- Data Store Adapter
- Multi-Level Branching
- Multi-Level Transactions
- Dynamic Artifact Model
- Dynamic Searching API
- Indexing & Tagging
- Remote Event Service
- Extensible Rendering
- Plugin Dev Utilities
OSEE Applications

- Workflow management
- Requirements management
- Task scheduling
- Coverage
- Metrics
- Reporting
- etc
The goal is to create a unified assistive environment for surgery that integrates robotic devices, fused information environments combining preoperative images & models, intraoperative images & other sensors, surgical task modeling, and human-machine cooperative manipulation, as shown in Figure 1 (from Reference 2.1).
Conclusions

Agile is relevant for developing safety-critical software

Consider the OSEE approach
Further readings

- www.open-do.org
- www.eclipse.org/osee/