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

OSEE

I. Certification

II. Open-DO

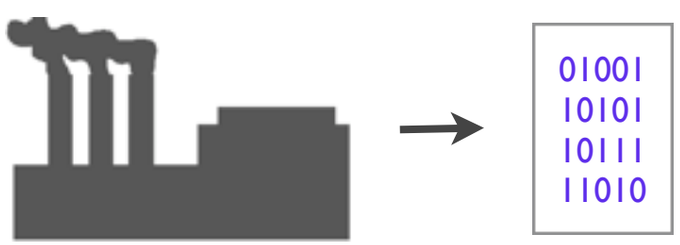
III. OSEE

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 = DEMONSTRABLE DEPENDABILITY

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

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)

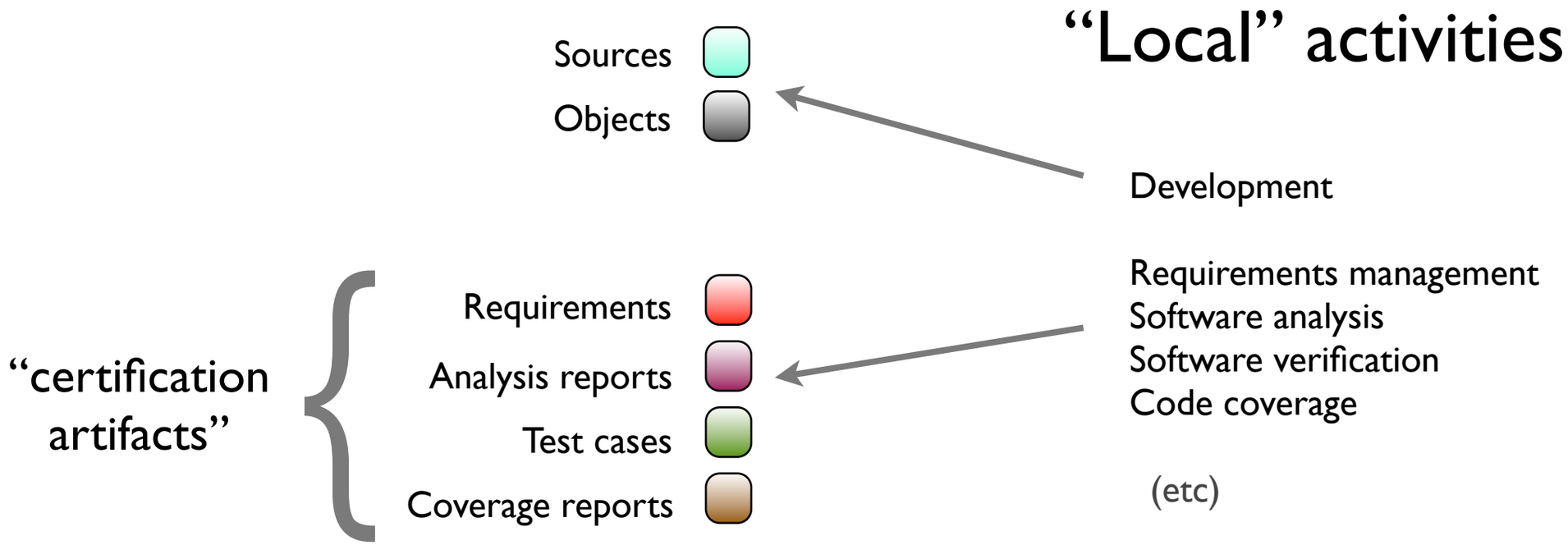
Activities depend on the targeted assurance level

Level A: failure results in catastrophe (crash/multiple deaths)

to

Level E: software has no impact on the mission

DO-178B / ED-12B concepts



Need to guarantee <u>traceability</u>

Some problems with current practices

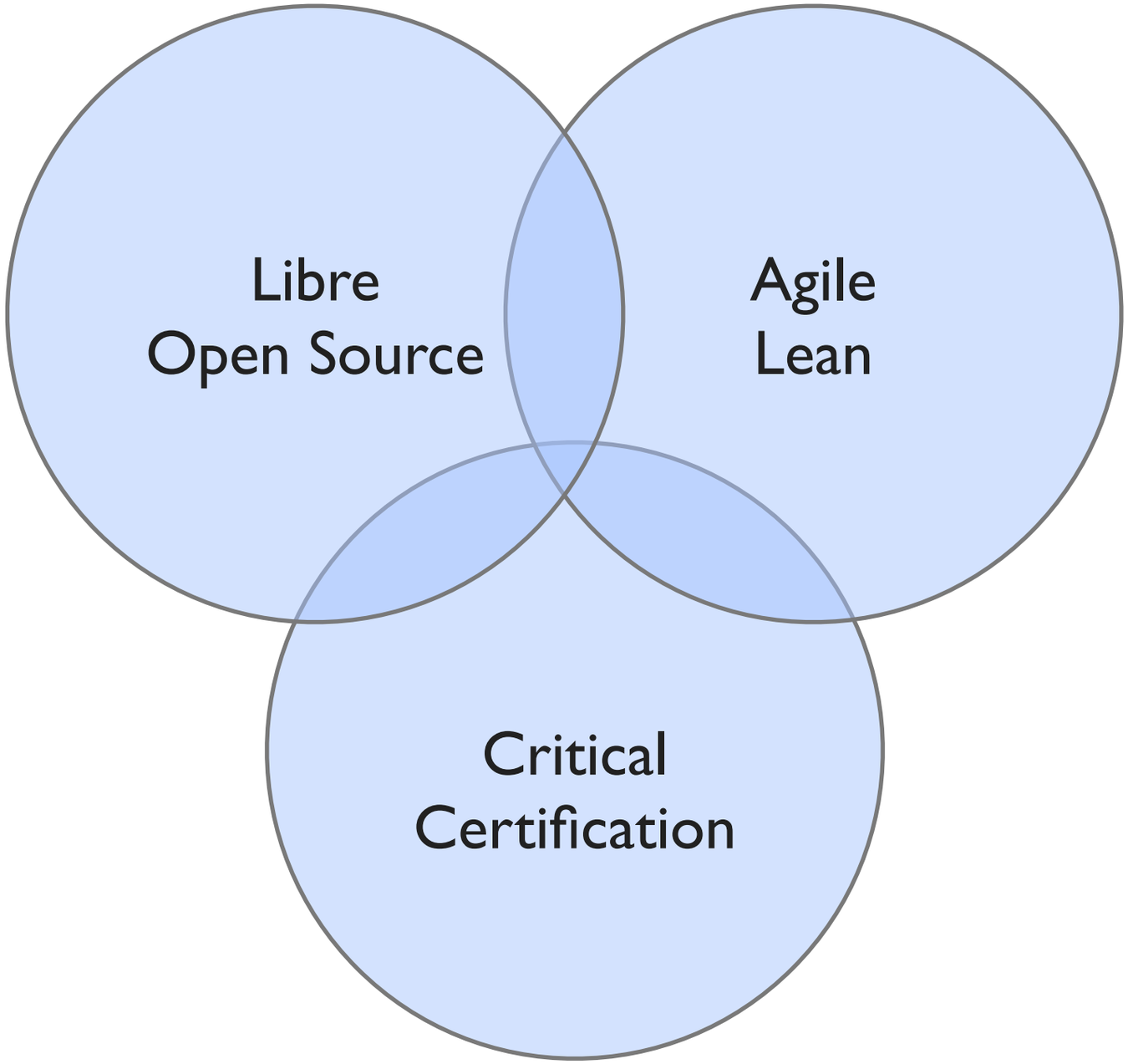
- Barrier of Entry
- Longevity and Availability
- The Big Freeze problem
  - *the RTEMS anecdote*

I. Certification

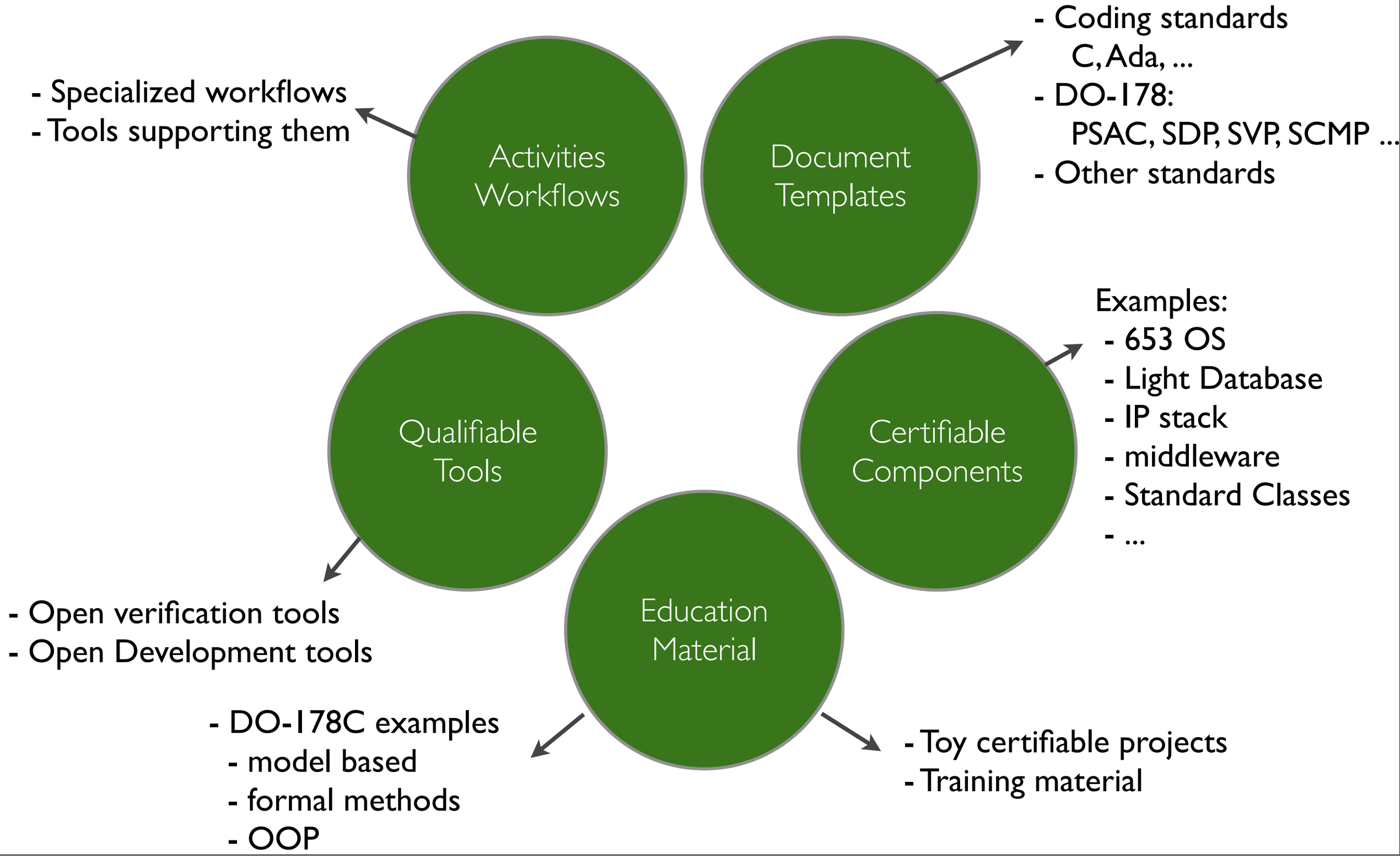
II. Open-DO

III. OSEE

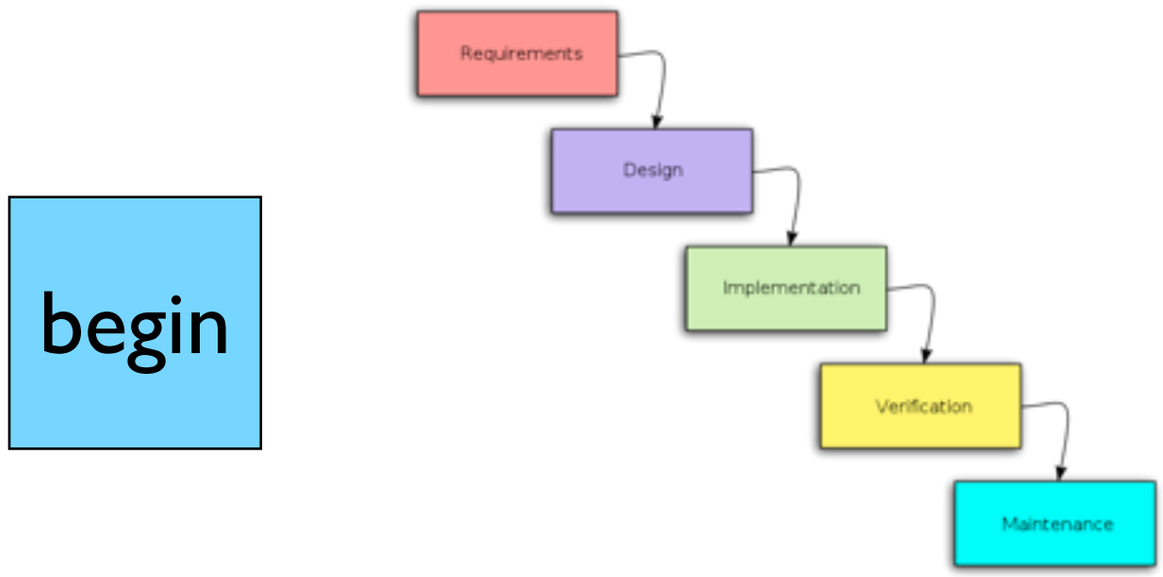
The meeting of 3 worlds



Open-DO components



The Big Freeze problem



begin

certify

deliver

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

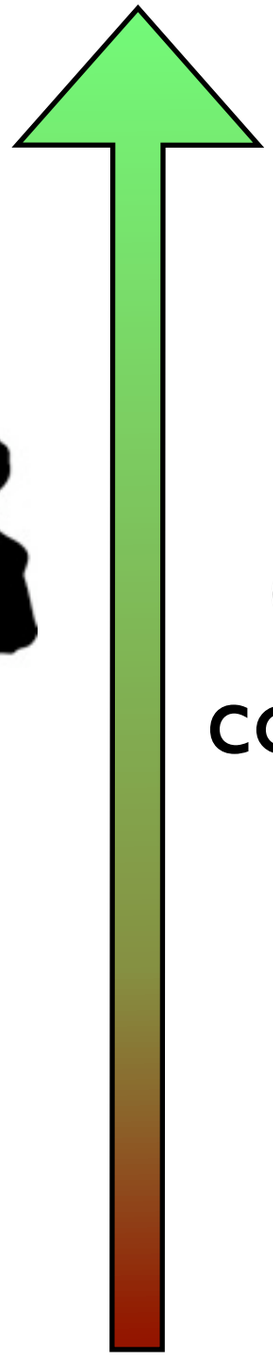
source patch
+ regression test
+ certification elements

Agile contribution

source patch
+ regression test

Standard contribution

source patch



Greater confidence

The Certification Machine

- 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

- What certification activities can be automated?
- How to implement the *machine* that does this automatically?

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

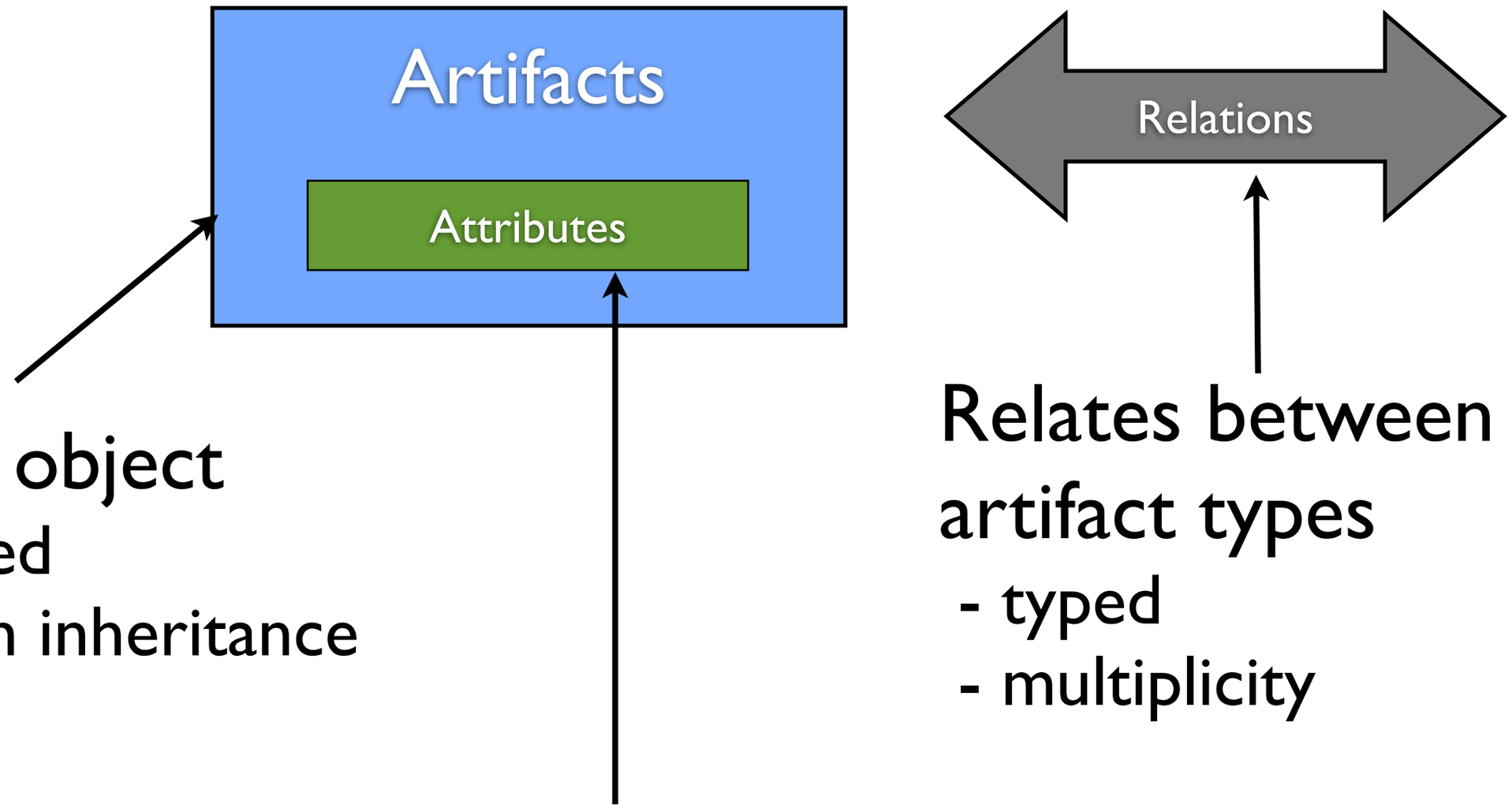
- "OSEE is a tightly integrated environment designed to support lean engineering principles across a product's full life-cycle in the context of overall systems engineering approach."

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

OSEE Data Model

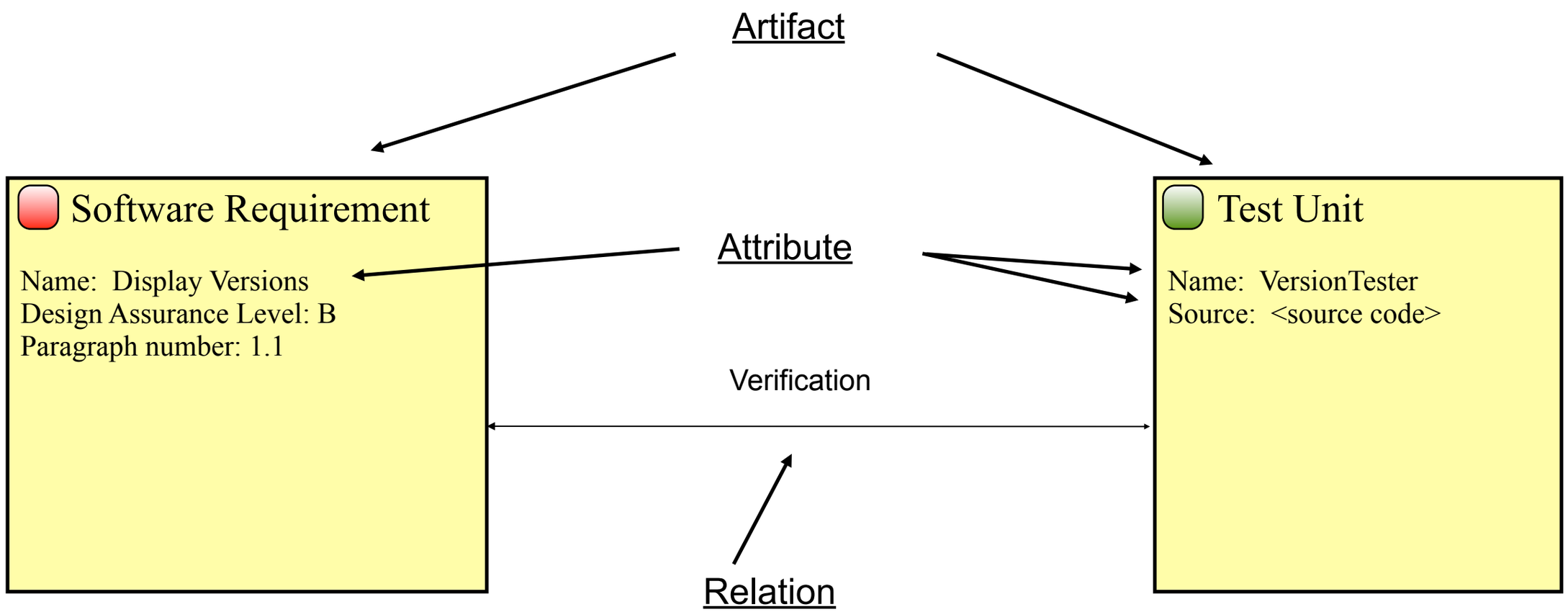


Data object
- typed
- with inheritance

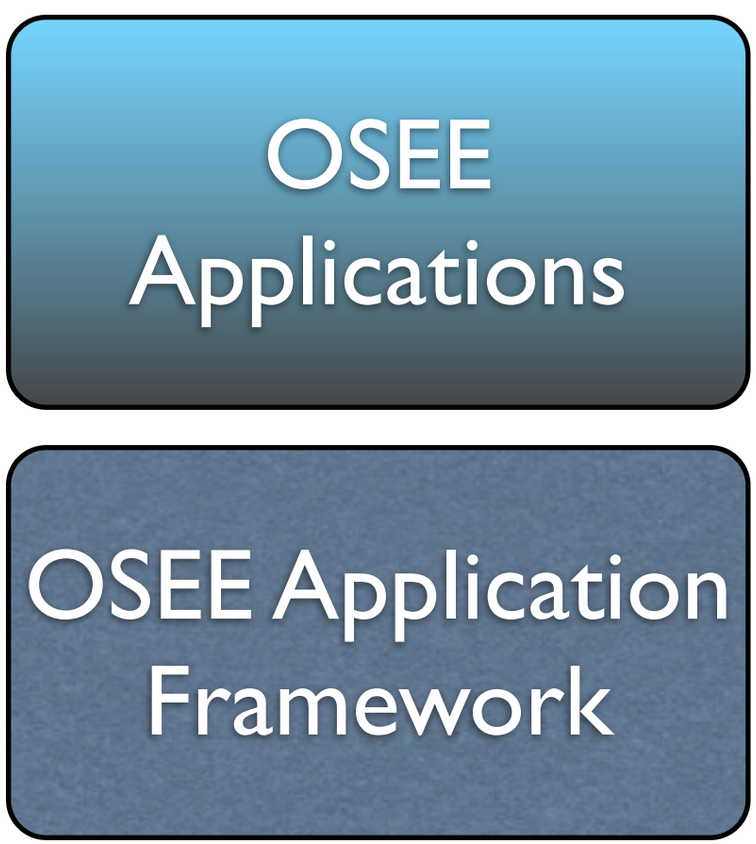
Relates between artifact types
- typed
- multiplicity

Key/Value pair
- base types

OSEE Data Model

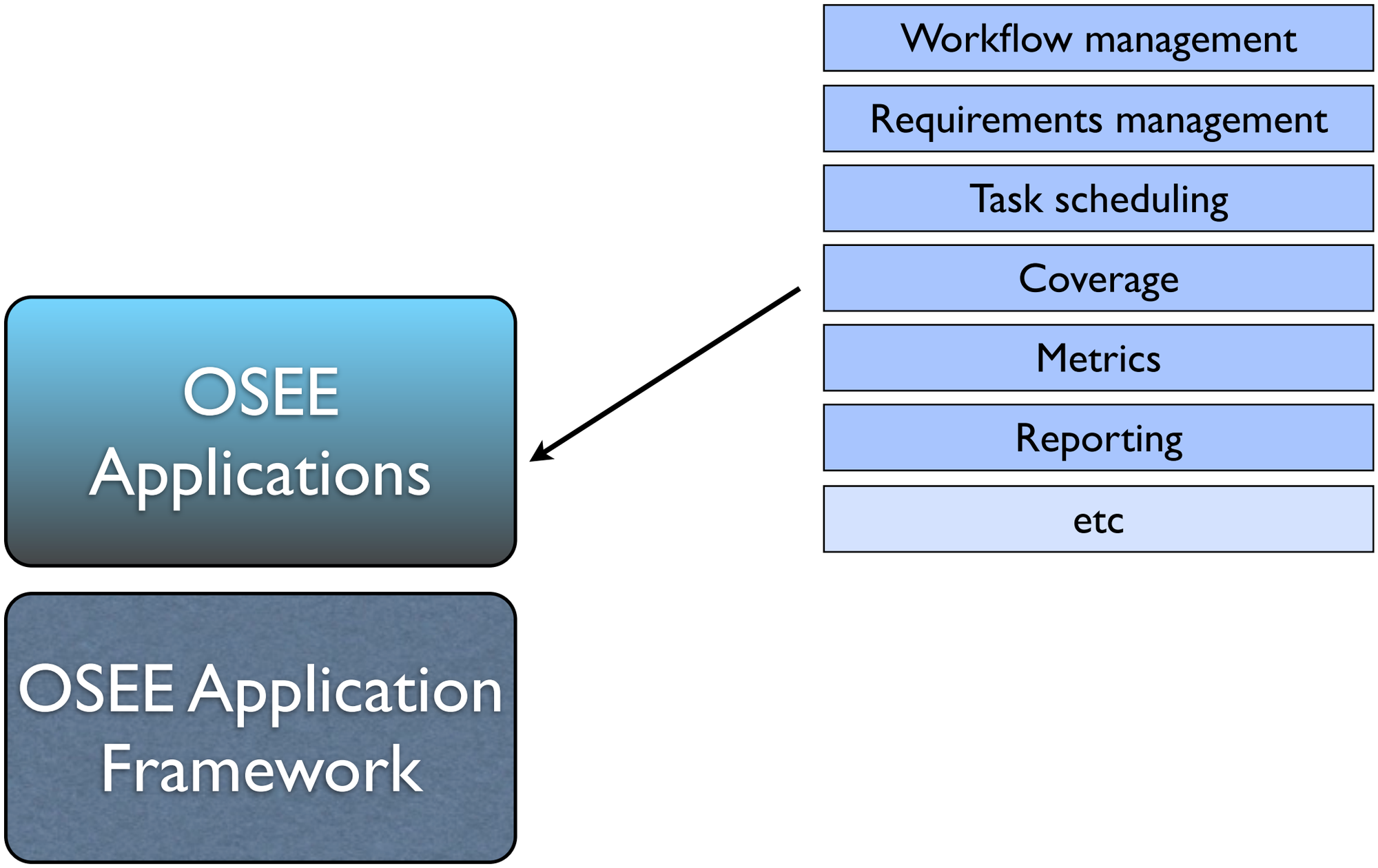


OSEE Services



- 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



ATS - SAW Even More Requirement (no-branch) Changes for Diagram View - Open System Engineering Environment

File Edit Navigate Search Project Run Window Help

Default Hierarchy Root

- Document Templates
- SAW Product Decomposition
- System Requirements
  - Objective
  - References
  - Robot System Overview
  - Performance Requirements
  - Safety Requirements
  - Design Constraints
- Subsystem Requirements
  - Subsystem Requirements
    - Robot API
    - Video processing
    - Other device interfaces
    - Calibration and registration
    - Tool tracking
    - User Interface (Visualization)
    - Telesurgery application framework
    - Volume viewer
- Software Requirements
  - Robot API
    - Robot Interfaces
    - Interface Initialization
    - Robot collaboration
    - Read-only Robots
    - CISST fundamental data types
    - Functional Specification
    - Events
    - Virtual fixtures
- Hardware Requirements
- Verification Tests
  - Verification Test A
  - Verification Test B
  - Verification Test C
- Validation Tests
  - Validation Test 1
  - Validation Test 2
  - Validation Test 3
- Integration Tests
  - integration Test X
  - integration Test Y
  - integration Test Z

[SAW Requirements] - SAW Even More Requirement (no-branch) Changes for Diagram View

SAW Even More Requirement (no-branch) Changes for Diagram View

Current State: Implement Team: SAW Requirements Assignee(s): Joe Smith Originator: Joe Smith Action Id: C

Action Actionable Items: SAW Requirements, SAW Test, SAW Code, SAW SW Design

Team Actionable Items: SAW Requirements

- Endorse - State Completed 03/15/2008 09:51 AM by Joe Smith
- Analyze - State Completed 03/15/2008 09:51 AM by Joe Smith
- Authorize - State Completed 03/15/2008 09:51 AM by Joe Smith
- Implement - Current State assigned to Joe Smith

Statistics

AtsAdmin

Total Percent: 0

Total Hours Spent: 0.60

Target Version: SAW_Bld_2

State Percent: 0

State Hours Spent: 0.00

"Implement" state assigned to Assignee(s): Joe Smith

Resolution Override:

Resolution:

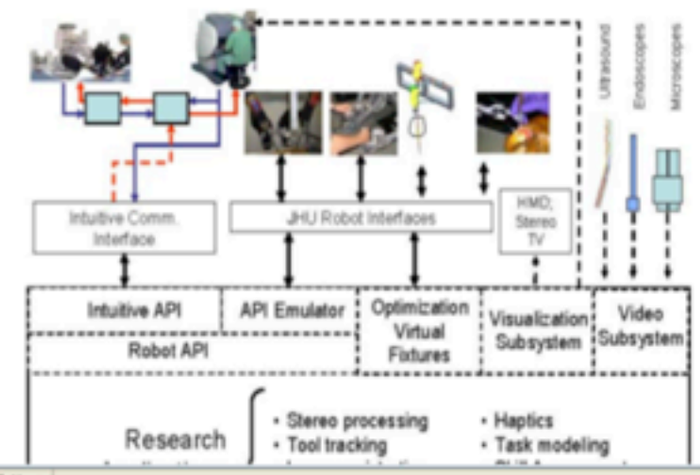
Workflow Tasks

Robot System Overview

Type: "System Requirement" Guid: AAABGLNZotsASsonPBXf7A HRID: EZ0QU Art Id: 293

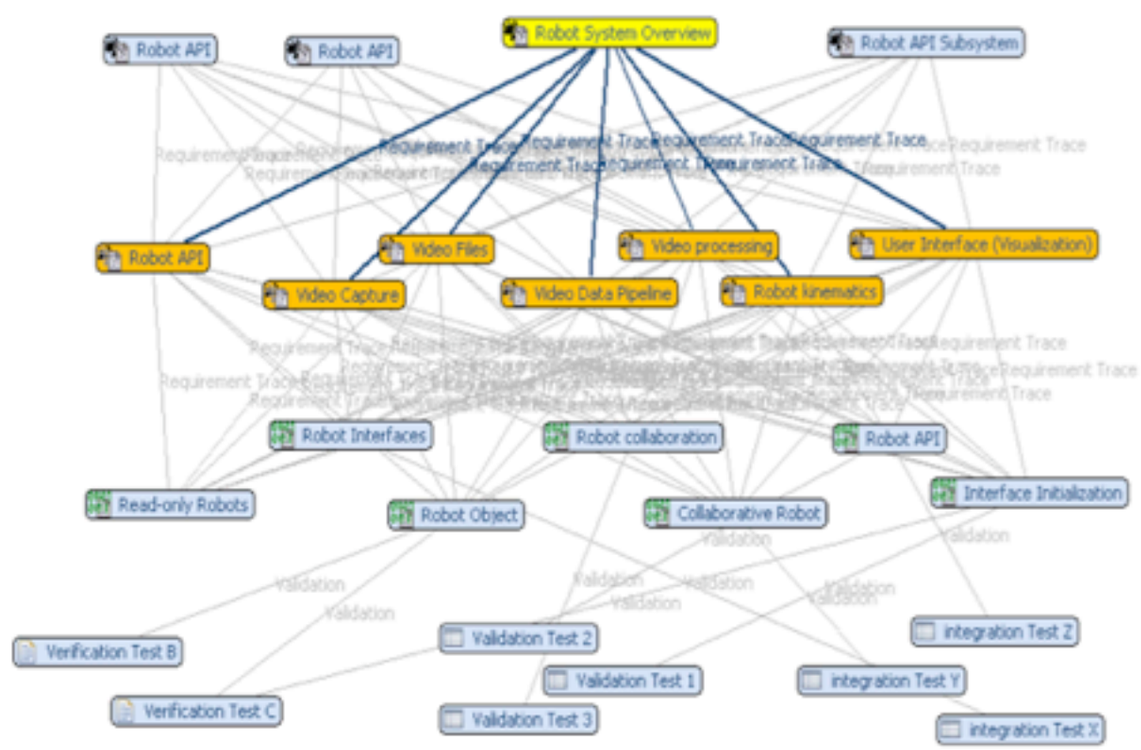
Robot System Overview

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 2.1.1).



Preview Attributes Relations

Sky Walker (Robot System Overview)



Layout: Vertical, Tree, Radial - Down, Grid, Radial - Right (defaul, Spring, Radial - Full

Level: 3

Link Naming: None, Link_Name, Full_Link_Name, Phrasing_A_to_B, Phrasing_B_to_A, Other_Side_Name

Action View

ATS World

My World - Joe Smith

Admin - OSEE, osee - Joe Smith

Type	State	P	C	Assignees
Action	Implement	1		Kay Jones, J...
Action	Implement	1		Jason Michae...
Action	Analyze	1		Kay Jones, J...
Action	Implement	1		Kay Jones, J...
SAW Test Workflo	Implement	1		Kay Jones
SAW Code Workflo	Implement	1		Joe Smith
SAW SW Design W	Implement	1		Kay Jones
SAW Requirements	Implement	1		Joe Smith
Action	Implement	1		Kay Jones, J...
Action	Implement	1		Jason Michae...

6 Loaded - 10 Shown - 1 Selected

Filter:

SAW_Bld_2

Friday, 26 June 2009

Conclusions

Agile is relevant for developing safety-critical software

Consider the OSEE approach

Further readings

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