



Configuration Management, Quality Assurance and Software Tools

Systems Certification and Integrity
Directorate of Aviation Engineering
Directorate General Technical Airworthiness

1

Directorate General Technical Airworthiness (DGTA-ADF)



Overview

- Software Configuration Management
- Software Quality Assurance
- Software Tool Qualification

2

Directorate General Technical Airworthiness (DGTA-ADF)





Software Configuration Management



Configuration Management

- Purpose: Establish and maintain the integrity of software development life cycle data through the software life cycle.
 - Establish traceability and consistency between the software and the software work products that document the development and verification of the software.
 - How useful is a software test report if you can't trace it to the version of software that was tested?
- Goals:
 - Planned and repeatable process
 - Software work products are identified and controlled
 - Changes to identified software work products are controlled
 - Development and verification is only performed using relevant software baselines
- Integral Process – can't recover Software CM at the end.





Configuration Management Key Tasks

- A process exists (software configuration management plan)
- A library is established as a repository for software baselines.
- Work products to be placed under CM are identified.
- Change requests and problem reports are initiated, recorded, reviewed, approved and tracked in accordance with a documented and repeatable procedure.
- Changes to baselines are controlled in accordance with a documented and repeatable procedure.



Configuration Management Key Tasks

- Release of software products is controlled in accordance with a documented procedure.
 - Software Load Control
- Status of CIs is recorded IAW a documented procedure.
- Reports documenting SCM activities and baselines are available.
- Software baseline audits are conducted according to a documented procedure.
- Software life cycle environment configuration is controlled.





Control Category

- Outputs of the software assurance life cycle require different degrees of configuration control, depending on their role in satisfying software assurance objectives.
- There are two control processes under which data can be classified:
 - Control Category 1 (CC1)
 - Control Category 2 (CC2)

| SCM Process Objective | CC1 | CC2 |
|---|-----|-----|
| Configuration Identification | ■ | ■ |
| Baselines | ■ | |
| Traceability | ■ | ■ |
| Problem Reporting | ■ | |
| Change Control – Integrity and Identification | ■ | ■ |
| Change Control – Tracking | ■ | |
| Change Review | ■ | |
| Configuration Status Accounting | ■ | |
| Retrieval | ■ | ■ |
| Protection Against Unauthorised Changes | ■ | ■ |
| Media Selection, Refreshing, Duplication | ■ | |
| Release | ■ | |
| Data Retention | ■ | ■ |



Challenges for CM

- Large development teams
- Parallel or concurrent development
- Geographically or time dispersed teams
- Multiple versions for different targets/platforms
- Mixture of COTS and bespoke software
- Level of integration between software components
- When should SCM begin?
- Level of control
- Level of automation





Summary of Configuration Management

- Software Configuration Management provides the following benefits:
 - Links between object code and the development and verification evidence that proves its acceptability.
 - Assures that the software that is loaded to the target environment is the same as the software that was verified.
 - Assures that developers will work from the correct version of data.
 - Prevents unintended changes to software or data.
 - Tracks problems through to resolution.
- Software Configuration Management must be incorporated into the development process from the start.



Software Quality Assurance





Software Quality Assurance

- The process for providing adequate assurance that the software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans.
 - Approved plans and standards were followed – did they do what they agreed to do?
 - Transition criteria for life cycle processes are satisfied – were relevant issues resolved before progressing onto later life cycle phases?
 - Software Conformity Review
- Integral Process – can't build quality in at the end
- Requires Independence – dedicated SQA staff
 - Different to other DO-178B independence requirements



Software Conformity Review

- Done at the completion of development
- Similar to FCA/PCA.
- Obtain assurance that:
 - life cycle processes are complete
 - life cycle data is complete
 - executable object code is controlled and can be regenerated
- Must be able to regenerate code using only the instructions provided.
 - Start from a clean PC, follow instructions exactly, can the code be regenerated?
 - An FAA requirement: some code outlasts the company which may impact accident investigations.





Summary – Software Quality Assurance

- An independent check that the developer has done what they said they would do.
- Types of Involvement
 - Check that particular product complies with requirements.
 - Check that transition criteria have been satisfied.
 - Software Conformity Review.
 - Periodic Audits.
- SQA Personnel must be independent enough to be effective (e.g. answer to management).



Software Tool Qualification





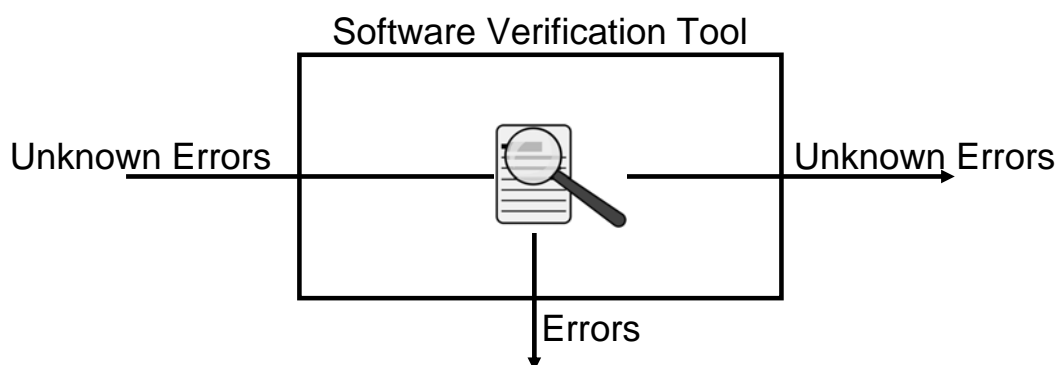
Software Tools

- What is a software tool?
 - A computer program used to develop, test, analyse, produce or modify another program or its documentation.
 - Distinguish from software used to make other engineering decisions (e.g. CAMM2, HUMS, W&B, etc)
- Why use software tools?
 - Improvements in productivity, lower numbers of errors.
 - But tools may also introduce errors that could go undetected.
- Why qualify tools?
 - Ensure the tool provides confidence at least equivalent to that of the processes being eliminated, reduced or automated.
- Two Types of Tools
 - Development Tools and Verification Tools



Verification Tools

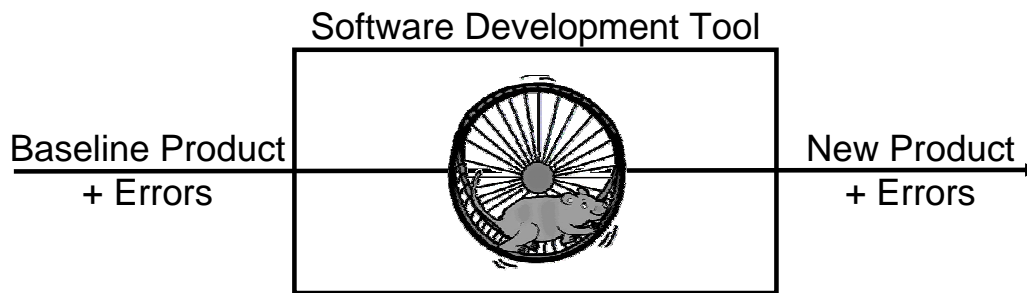
- Unable to create or modify development products
 - Requirements, Design or Code
- May be able to create or modify verification products
 - Test Cases, Test Results, Code Inspections, etc
- Cannot introduce errors into the object code, but can fail to detect them.





Development Tools

- Tools whose output is part of the airborne software
- Modifies or creates requirements, design or code
- Can inject errors into development products



Combined Tools

- Many modern development environments provide the features of both development and verification tools in an integrated suite of tools.
- How to handle combined tools:
 - Categorise functions, rather than the tool.
 - Tool functions may be qualified separately if partitioning between functions can be demonstrated.





Examples of Tool Classifications

- Development
 - Autocode Generators
 - Compilers
 - Software Libraries
- Verification
 - Simulators
 - Emulators
 - Test Case Generators
 - Test Tools
 - Coverage Analysers
 - Static Analysers
 - Proof Checkers
 - Model Checkers



What is tool qualification?

- Process to ensure that a tool provides confidence at least equivalent to the processes that are eliminated, reduced or automated.
- See DO-178B Section 12.2.
- Alternative: Verification of Tool Outputs as per DO-178B Section 6
- Order 8110.49 Chapter 9 provides guidelines





Tool Qualification Criteria

- Not all tools require qualification
- Only need to qualify a software tool if the answers to the following three questions are all yes:
 - Can the tool insert an error or allow an existing error to remain undetected?
 - Are software assurance processes eliminated, reduced or automated?
 - Will the tool's output not be verified per section 6 of DO-178B?



Tool Qualification

- For all tools
 - Tool operational requirements should be defined
 - (as per DO-178B section 12.2.3.2)
- Verification Tools
 - Verification that tool meets operational requirements
 - PSAC, SAS
- Development Tools
 - Software level assigned to tool commensurate with software level assigned to software being developed.
 - Complete DO-178B data package for assigned software level
 - Tool Qualification Plan, Tool Accomplishment Summary





Tool Operational Requirements

- Development Tool
 - Functionality
 - Operational Environment
 - Installation or Operational Information
 - Development Process Performed
 - Expected Response under Normal and Abnormal Conditions
- Verification Tool
 - Functionality
 - Operational Environment
 - Installation or Operational Information



Special Tool Topics

- Compilers
 - Development Tool? – Yes
 - Qualification Required? – No, as other software assurance objectives assure verification of processes handled by compilers.
- Configuration Management Tools
 - Development Tool? – Yes (but...)
 - Most certification authorities treat CM tools as Verification Tools for the purposes of qualification
 - Pragmatic Decision





Example

- Database selects thread libraries to be used at compile time.
 - Libraries assumed to be appropriately assured.
 - Only considering the database software for now.
- What type of tool? Development
- Qualification Case 1
 - If compiled output is reviewed to verify that correct libraries have been included, then qualification is not required.
- Qualification Case 2
 - If selected libraries compiled and linked without review then qualification is required.



Summary – Software Tool Qualification

- **If software tools are to be relied upon, they should be qualified.**
 - Should have the same level of confidence in the tool as the process being reduced, eliminated or automated.
- **Three questions to determine qualification:**
 - Software assurance process reduced, eliminated or automated?
 - Can introduce errors or fail to detect them?
 - Output not independently verified?
- **Verification Tools**
 - Can't introduce an error, but can fail to detect
 - Relatively easy to qualify, in common use
- **Development Tools**
 - Can introduce an error
 - Difficult to qualify, not recommended





Questions

