Software Configuration Management Audits
Part 2 – Functional Configuration Audits (FCA)

By Linda Westfall
lwestfall@westfallteam.com

In the first part of this article, we introduced the three different types of Software Configuration Management Audit:

- Functional Configuration Audit (FCA)
- Physical Configuration Audit (PCA)
- In-Process SCM Audits

We also talked about when these audits occur in the software development life cycle

This second part of the article talks about Functional Configuration Audits and their purpose. It will also provide examples of checklists that could be used during FCA evaluations and suggests evidence-gathering techniques for each item in those checklists.

Purpose of a Functional Configuration Audit (FCA)

According to the IEEE, an FCA is an audit conducted to verify that: [IEEE-610]

- The development of a configuration item has been completed satisfactorily
- The item has achieved the performance and functional characteristics specified
- Its operational and support documents are complete and satisfactory

An FCA is performed to provide an independent evaluation that the as-built, as-tested system/software and its deliverable documentation meet the specified functional, performance, and other quality attribute requirements.

An FCA is essentially a review of the system/software's verification and validation (V&V) data to ensure that the deliverables are sufficiently mature for transition into either beta testing or production at the end of the development cycle. If FCAs are conducted at intermediate milestones, they review V&V data to ensure that the deliverables of each milestone are mature enough to transition to the next development phase depending on where in the life cycle the FCA is conducted.

Checklist Item Suggestions for Evidence-Gathering Techniques

Table 1 illustrates an example of a checklist and lists possible objective evidence-gathering techniques for each checklist item that would be used for an FCA conducted at any baseline or major milestone.

While several suggested evidence-gathering techniques are listed for each checklist item, the level of rigor chosen for the audit will dictate which of these techniques (or other techniques) will actually be used. For example, when evaluating whether the code implements all and only the documented requirements, a less rigorous approach would be to evaluate the traceability matrix, while a more rigorous audit might examine actual code samples and review the code against the allocated requirements.

Table 1 – Example Checklist and Evidence-Gathering Techniques Used During Any FCA

Checklist Item	Suggestions for Evidence-Gathering Techniques
1. Does each baselined configuration item (CI) implement all and only the documented software/system requirements?	• Evaluate requirements-to-CI forward and backward traceability information for completeness and to ensure that no unauthorized functionality has been implemented. • Sample a set of requirements and using the traceability information, review each associated, baselined CI for implementation completeness and consistency. • Sample a set of approved enhancement requests and review their resolution status (or if approved for change, evaluate their associated, baselined CIs for implementation completeness and consistency). • Sample a set of baselined CIs and compare with the previous versions to identify changes. Ensure that each change corresponds to a requirement or approved change request.
2. Are all the defects/anomalies reported during verification & validation (V&V) activities adequately resolved (or the appropriate waivers/deviations obtained and known defects with work-arounds are documented in the release notes)?	• Review a sample set of approved defect/ anomaly report records for evidence of adequate resolution. • Sample a set of defect/anomaly report records and review their resolution status (or if approved for change, evaluate their associated CIs for implementation completeness and consistency). • Review V&V iteration results data (e.g., re-peer review records, re-test/regression test logs, test case status, and/or metrics) to ensure adequate V&V iteration coverage after defect correction.

Table 2 illustrates an example of a checklist and lists possible objective evidence-gathering techniques for each checklist item that would be used for an FCA conducted at the product/release baseline.

Table 2 – Example of Additional Checklist Item and Evidence-Gathering Techniques Used for FCA at Product/Release Baseline

Checklist Item	Suggestions for Evidence-Gathering Techniques
3. Can each system/software requirement be traced forward into tests cases/procedures that V&V that requirement?	• Evaluate requirements-to-tests traceability information for completeness. • Sample a set of requirements and using the traceability information, review the associated test documentation (e.g., test plans, defined test cases/procedures) for adequacy of V&V by ensuring the appropriate level of test coverage for each requirement.
4. Is comprehensive system/software testing complete, including functional testing, interface testing and the testing of required quality attributes (performance, usability, safety, security, etc.)?	• Review approved V&V reports for accuracy and completeness. • Evaluate approved test documentation (e.g., test plans, defined test cases/procedures) against test results data (e.g., test logs, test case/procedure status, test metrics) to ensure adequate test coverage of the requirements and system/software during test execution. • Execute a sample set of test cases to evaluate the accuracy of test results.
5. Is the operational & support documentation consistent with the requirements and as-built system/software?	• Review minutes from peer reviews and defect resolution information from operational & support documentation reviews for evidence of consistency. • Evaluate formal test documentation (e.g., test plans, defined test cases/procedures) against test results data (e.g., test logs, test case/procedure status, test metrics) to ensure adequate test coverage of the operational & support documentation during test execution. • Review sample set of updates to previously delivered documents to ensure consistency with requirements and as built system/ software?