Software Configuration Management Audits

By Linda Westfall

theWestfallTeam
Partnering for Software Excellence

www.westfallteam.com

An audit is a planned and independent evaluation of one or more products or processes to determine conformance or compliance to a set of agreed to requirements. Auditing is an "objective assurance and consulting activity designed to add value and improve an organization's operations." [Hutchins-03] Audits provide assurance by validating that the products and/or processes are implemented in accordance with their requirements and objectives. Audits are consulting activities because they provide on-going analysis of the degree to which those implementations are effective and efficient and they identify opportunities for continuous improvement. Audits also visibly demonstrate management's support for the quality program.

In the case of Software Configuration Management (SCM) audits, three types of audits are typically performed:

- Functional Configuration Audit (FCA), which is an evaluation of the completed software products to determine their conformance, in terms of completeness, performance and functional characteristics, to their requirements specification(s).
- Physical Configuration Audit (PCA), which is an evaluation of each configuration item to determine its conformance to the technical documentation that defines it.
- In-Process SCM Audits, which are ongoing evaluations conducted throughout the life cycle to provide management with information about compliance to SCM policies, plans, processes and systems, and about the conformance of software product to their requirements and workmanship standards.

This paper discusses the purpose of each of these three types of SCM audits. It also provides examples of checklist items that could be used during audit evaluations and suggested evidence gathering techniques for each of the items in those checklists.

Functional Configuration Audit (FCA)

According to the IEEE, a 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

A FCA is performed to provide an independent evaluation that the as-built, as-tested system/software and its deliverable documentation meets its specified functional, performance and other quality attribute requirements. Typically the FCA is conducted just before the final Ready to Beta Test or Ready to Ship review and provides input information into those reviews. A FCA is essentially a review of the system/software's verification and validation data to ensure that the deliverables are sufficiently mature for transition into either beta testing or production, depending on where in the life cycle the FCA is conducted.

Table 1 illustrates an example of a FCA checklist and lists possible objective evidence gathering techniques for each item. 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

Copyright © 2006 The Westfall Team. All Rights Reserved.

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.

Checklist Item	Suggestions for Evidence Gathering Techniques
1. Does the code implement all and only the documented software/system requirements?	• Evaluate requirements-to-source code forward and backward traceability information (e.g., traceability matrix or trace tags) for completeness and to ensure that no unauthorized functionality has been implemented. • Sample a set of requirements and using the traceability information, review the associated code 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 code for implementation completeness and consistency).
2. Can each system/software requirement be traced forward into tests (test cases, procedures, scripts) that verify that requirement?	• Evaluate requirements-to-tests traceability information (e.g., traceability matrix or trace tags) for completeness. • Sample a set of requirements and using the traceability information, review the associated test documentation (e.g., test plans, defined tests) for adequacy of verification by ensuring the appropriate level of test coverage for each requirement).
3. 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 verification and validation reports for accuracy and completeness. • Evaluate approved test documentation (e.g., test plans, defined tests) against test results data (e.g., test logs, test case 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 accuracy of test results.
4. Are all the anomalies reported during testing adequately resolved (or the appropriate waivers/deviations were obtained and known defects with work-arounds are documented in the release notes)?	• Review a sample set of approved test anomaly report records for evidence of adequate resolution. • Sample a set of test anomaly report records and review their resolution status (or if approved for change, evaluate their associated code for implementation completeness and consistency). • Review regression test results data (e.g., test logs, test case status, test metrics) to ensure adequate test coverage after defect correction.

Copyright © 2006 The Westfall Team. All Rights Reserved.

5. Is the deliverable documentation consistent with the requirements and as-built system/software?	• Review minutes from peer reviews and defect resolution information from deliverable documentation reviews for evidence of consistency. • Evaluate formal test documentation (e.g., test plans, defined tests) against test results data (e.g., test logs, test case status, test metrics) to ensure adequate test coverage of the deliverable during test execution. • Review sample set of updates to previously delivered documents to ensure consistency with requirements and as built system/software?
6. Are the findings from peer reviews incorporated into the software deliverables (system/software and/or documentation)?	• Review records from major milestone/phase gate reviews that verified the resolution of peer review defects • Review a sample set of peer review records for evidence of defect resolution • Review a sample set of minutes from peer review and evaluate the defect lists against the associated work products to ensure that the defects were adequately resolved
7. Have approved corrective actions been implemented for all findings from In-Process Software Configuration Management Product Audits?	• Evaluate findings from audit reports against their associated corrective action status. • Re-audit against findings to verify implementation of corrective actions.

Table 1 – Example FCA Checklist and Evidence Gathering Techniques

Physical Configuration Audit (PCA)

According to the IEEE, a PCA is an audit conducted to verify that each configuration item, as built, conforms to the technical documentation that defines it. [IEEE-610] A PCA verifies that:

- All items identified as being part of the configuration are present in the product baseline
- The correct version and revision of each part are included in the product baseline
- They correspond to information contained in the baseline's configuration status report

A PCA is performed to provide an independent evaluation that the coded software has been described adequately in the documentation that will be delivered with it and that the software and its documentation have been captured in the configuration management database and are ready for delivery. Finally, the PCA may also be used to evaluate adherence to legal obligations including licensing and export compliance requirements.

The PCA is typically held either in conjunction with the FCA or soon after the FCA (once any issues identified during the FCA are resolved). A PCA is essentially a review of the software configuration status accounting data to ensure that the software products and their deliverable documentation are appropriately baselined and properly built prior to release to beta testing or production, depending on where in the life cycle the PCA is conducted.

Table 2 illustrates an example of a PCA checklist and lists possible objective evidence gathering techniques for each item.

Copyright © 2006 The Westfall Team. All Rights Reserved.

Checklist Item	Suggestions for Evidence Gathering Techniques
1. Has each nonconformance or noncompliance from the FCA been appropriately resolved?	• Review findings from the FCA audit report, associated corrective actions, follow-up and verification records to evaluate adequacy of actions taken (or appropriate approved waivers/deviations exist).
2. Have all of the identified Configuration Items (e.g., source code, documentation, etc.) been baselined?	• Sample a set of Configuration Items and evaluate them against configuration status accounting records.
3. Do all of the Configuration Items meet workmanship standards?	• Sample a set of source code modules and evaluate them against the coding standards. • Sample a set of deliverable documents (or sections/pages of those documents) and evaluate them against documentation standards.
4. Has the software been built from the correct components and in accordance with the specification?	• Evaluate the build records against the configuration status accounting information to ensure that the correct version and revision of each module was included in the build. • Evaluate any patches/temporary fixes made to the software to ensure their completeness and correctness. • Sample a set of design elements from the architectural design and trace them to their associated detailed design elements and source code. Compare those elements with the build records to evaluate for completeness and consistency with the as built software.
5. Is the deliverable documentation set complete?	• Evaluate the master copy of each document against the configuration status accounting information to ensure that the correct version and revision of each document sub-component (e.g., chapter, section, figure) is included in the document. • Sample the set of copied documents ready for shipment and review them for completeness and quality against the master copy. • Evaluate the version description document against the build records for completeness and consistency. • Compare the current build records to the build records from the last release to identify changed components. Evaluate this list of changed components against the version description document to evaluate the version description document's completeness and consistency.

Copyright © 2006 The Westfall Team. All Rights Reserved.

6. Does the actual system delivery media conform to specification? Has the delivery media been appropriately marked/labeled?	• Evaluate the items on the master media against the required software deliverables (executables, help files, data) to ensure the correct versions and revisions were included. • Sample a set of copied media ready for shipment and review them for completeness and quality against the master media. • Sample a set of copied media ready for shipment and review their marking/labeling against specification.
7. Do the deliverables for shipment match the list of required deliverables?	• Evaluate the packing list against the list of documented deliverables to ensure completeness. • Sample a set of ready-to-ship packages and evaluate them against the packing list to ensure that media (i.e., CD, disks, tape), documentation and other deliverables are included in each package.
8. Have 3rd party licensing requirements been met?	• Evaluate the build records against configuration status accounting information to identify 3rd party components and license information to confirm adequate numbers of licenses exist.
9. Have export compliance requirements been met?	• Evaluate the build records against configuration status accounting information to identify components with export restrictions and confirmed export compliance.

Table 2 – Example PCA Checklist and Evidence Gathering Techniques

In-Process Software Configuration Management (SCM) Audits

In-process SCM audits are performed throughout the software life cycle to provide management with an ongoing independent evaluation of the:

- Adequacy of the organization's SCM policies, plans, processes and systems to meet the organization's objectives
- Ongoing compliance to those documented SCM policies, plans, processes and systems
- Ongoing conformance of the configuration items to their requirements and workmanship standards
- Effectiveness of the SCM plans, processes and systems, and their implementation (e.g., SCM training of personnel and SCM tool capabilities)
- Efficiency of resource utilization
- Identification of areas for continuous improvement to SCM plans, processes, systems and products.

In-process SCM audits are typically focused on either SCM processes or SCM baselines. Table 3 illustrates an example of a checklist for a process-focused in-process SCM audit and lists possible objective evidence gathering techniques for each item. Table 4 illustrates an example of a checklist for a baseline-focused in-process SCM audit and lists possible objective evidence gathering techniques for each item.

Copyright © 2006 The Westfall Team. All Rights Reserved.

Checklist Item	Suggestions for Evidence Gathering Techniques
1. Are there defined SCM policies and/or standards associated with this process and are they adequate to meet the organization's defined objectives?	• Perform a document review of the SCM policies and/or standards associated with the process being audited against the organization's defined objectives • Interviews with key personnel to evaluate their knowledge of the connection between SCM policies and/or standards and organizational objectives.
2. Are there defined SCM project plans associated with this process and are they adequate to meet defined policies and/or standards?	• Perform a document review of the SCM plans associated with the process being audited to evaluate adequacy against SCM policies and/or standards • Interviews with key personnel to evaluate their knowledge of the connection between SCM plans and SCM policies and/or standards.
3. Are the procedures and/or work instructions for the processes adequate to implement defined policies, standards and/or plans?	• Perform a document review of the SCM plans associated with the process being audited to evaluate adequacy against SCM policies, standards and/or plans.
4. Does each person performing SCM tasks associated with the process have access to applicable procedures or work instructions?	• Interview a sample of personnel performing tasks to evaluate their knowledge of the existence, availability and content of the applicable procedures or work instructions.
5. Are the procedures or work instructions up-to-date (latest revision)?	• Check revision numbers of the copies of procedures and work instructions in use by personnel and compare those against current baseline revisions, as interviews are conducted for checklist item 4.
6. Were the entry criteria to the SCM process verified before that process began?	• Interview a sample of personnel performing tasks to determine what entry criteria were used and how they determined that those entry criteria were met before initiation the process and evaluate their answers against process requirements. • Examine a sample quality records (e.g., completed entry criteria checklists) if applicable.
7. Does each person performing SCM tasks have the appropriate education, training, skills & experience?	• Interview a sample of personnel performing tasks to determine their knowledge/skill level or to ask about training received and evaluate their answers against process requirements. • Observe tasks being performed to ensure that they are being performed as specified. • Examine a sample quality records (e.g., completed checklists, data records, minutes, reports) for compliance to specification.

Copyright © 2006 The Westfall Team. All Rights Reserved.

8. Does everyone performing SCM tasks comply with the policies, standards, plans, procedures and work instructions?	• Interview a sample of personnel performing tasks to determine how they think activities are being performed and evaluate their answers against process requirements. • Observe tasks being performed to ensure that they are being performed as specified. • Examine a sample quality records (e.g., completed checklists, data records, minutes, reports) for compliance to specification.
9. Are the environment, infrastructure and tools utilized during the SCM task adequate to achieve conformity with the policies, standards, plans, procedures and work instructions	• Interview a sample of personnel performing tasks to determine adequacy of environment, infrastructure and tools. • Observe tasks being performed to ensure that the environment, infrastructure and tools are adequate.
10. Were the exit criteria to the SCM process verified before that process was considered complete?	• Interview a sample of personnel performing tasks to determine what exit criteria were used and how they determined that those exit criteria were met before completing the process and evaluate their answers against process requirements. • Examine a sample quality records (e.g., completed exit criteria checklists, minutes, reports) if applicable.
11. Are nonconformities/defects appropriately reported and tracked to closure?	• Interview a sample of personnel performing tasks to determine how nonconformities/defects are reported and tracked to closure and evaluate their answers against process requirements. • Examine a sample of quality records (e.g., nonconformance reports, corrective action reports, defect reports) if applicable.
12. Are the appropriate records being kept?	• Examination of the existence of required quality records and their storage and retention.

Table 3 – Example Process Focused In-Process Audit Checklist and Evidence Gathering Techniques

Checklist Item	Suggestions for Evidence Gathering Techniques
1. Does each configuration item in the baseline implement all and only its allocated requirements?	• Evaluate requirements-to-configuration item forward and backward traceability information (e.g., traceability matrix or trace tags) for completeness and to ensure that no unauthorized functionality has been implemented. • Sample a set of requirements and using the traceability information, review the associated configuration item 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 code for implementation completeness and consistency).

Copyright © 2006 The Westfall Team. All Rights Reserved.

2. Has each Configuration Item in the baseline passed the appropriate verification or validation gate required for acquisition?	• Review approved verification and validation quality records (e.g., peer review minutes, test reports) for accuracy and completeness.
3. Are all the defects/anomalies reported during those verification and validation activities adequately resolved (or the appropriate waivers/deviations obtained)?	• Review a sample set of approved defects/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 configuration items for implementation completeness and consistency).
4. Has each Configuration Item in the baseline been properly placed under configuration control?	• Examine a sample of items in the configuration management database to ensure that each item has been entered (typically this consists of ensuring that each item has been appropriately checked into a SCM tool or stored in a controlled SCM library).
5. Do all of the Configuration Items meet workmanship standards?	• Sample a set of source code modules and evaluate them against the coding standards. • Sample a set of documents (or sections/pages of those documents) and evaluate them against documentation standards.

Table 4 – Example Baseline Focused In-Process Audit Checklist and Evidence Gathering Techniques

Conclusion

Conducting SCM audits provides management with independent verification that the SCM processes are being complied with and that the software products are being built as required and at production, they are ready to be released. SCM plans for each project/program should include plans for conducting these SCM audits, including schedules and resource allocations.

Standardized checklists, like the example checklists in this paper, can be created for SCM audits. The advantage of using standardized checklists include:

- Reduction of effort in recreating checklists for each audit
- Lessons learned from previous audits can be incorporated into the standardized checklists to help improve future audits
- Consistency and continuity of implementation from one audit to the next as well as complete coverage

Prior to each audit, these standardized checklists should be reviewed to ensure that they reflect any changes made in the SCM standards, policies or plans since the last audit was conducted. These generic checklists should also be supplemented and tailored to the exact circumstances of each individual audit. For example, if the corrective actions against prior audit findings are being verified with the current audit, specific checklist items for those actions may be added to the checklist. Another example might be the auditing of small projects where certain optional processes do not apply and the corresponding items should be removed from the checklist.

References

IEEE-610	IEEE Standards Software Engineering, *IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 610-1990*, The Institute of Electrical and Electronics Engineers, 1999.

Copyright © 2006 The Westfall Team. All Rights Reserved.

Kasse-00	Tim Kasse and Patricia A. McQuaid, *Software Configuration Management for Project Leaders*, Software Quality Professional, Volume 2, Issue 4, September 2000.
Keyes-04	Jessica Keyes, Software Configuration Management, Auerbach Publications, Boca Raton, 2004.
Russell-00	ASQ Audit Division, J. P. Russell editing director, *The Quality Audit Handbook, 2nd Edition*, ASQ Quality Press, Milwaukee, WI, 2000.
Westfall-06	Linda Westfall, *Bidirectional Requirements Traceability*, *http://www.westfallteam.com/Papers/Bidirectional_Requirements_Traceability.pdf*.
Westfall-07	Linda Westfall, *Risk-Based Configuration Control – Balancing Flexibility with Stability*, *http://www.westfallteam.com/Papers/Risk-Based_Configuration_Control.pdf*.

Glossary

Term	Definition
acquisition	The point at which each baseline or each configuration item, component and unit is initially brought under formal configuration control. [Westfall-07]
build record	The record or documentation that define the contents of a build including the version/revision of each component (including library components) used to build the software product and any switches or options used.
product baseline	The initial approved technical documentation (including for software, the source code listing) defining a configuration item during the production, operation, maintenance, and logistic support of its life cycle. [IEEE-610]
quality record	Documentation, which provides objective evidence of the extent of the fulfillment of the requirements for quality or the effectiveness of the operations of the quality system. [Russell-00]
revision	Making changes to an entity (configuration item, component or unit) that corrects only errors (does not affect its functionality).
traceability	Each requirement can be traced back to a need of the user and forward through the implementation (design elements, code modules and tests) of the software. [Westfall-06] The degree to which a relationship can be established between two or more products of development process, especially products having a predecessor-successor or master-subordinate relationship to one another. [IEEE-610]
trace tags	Unique identifiers used in the subsequent work products to identify backwards traceability to the predecessor document. [Westfall-06]
version	A software configuration item with a defined set of functionality.
version description document	A deliverable document that describe the released version of a software product including an inventory of system or component parts, new/changed features/functionality, know defects and their work arounds, and other information about that version. (also called release notes)

Copyright © 2006 The Westfall Team. All Rights Reserved.