TECHNICAL REPORT

ISO/IEC TR 18018

First edition
2010-02-15

Information technology — Systems and software engineering — Guide for configuration management tool capabilities

Technologies de l'information — Ingénierie des systèmes et du logiciel — Guide pour les capacités d'outil de gestion de configuration

Reference number
ISO/IEC TR 18018:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 18018, which is a Technical Report of type 2, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

Introduction

Configuration management (CM) is a process central to the software engineering life cycle. CM has been established as an ISO/IEC standard life cycle process in ISO/IEC 12207:2008, *Systems and software engineering — Software life cycle processes* and ISO/IEC 15288:2008, *Systems and software engineering — System life cycle processes*.

ISO/IEC 12207 and ISO/IEC 15288 describe a comprehensive set of processes, activities and tasks to be performed when acquiring or developing software. However, these documents do not address the capabilities that a CM tool user can expect from a tool in order to support the CM process and other software engineering life cycle activities. There is a gap between CM process descriptions and corresponding CM process automation which affects both tool users and tool suppliers.

This Technical Report provides guidance in the evaluation and selection for CM tools during acquisition. CM tool evaluation by prospective users can be complex, time consuming, and expensive. This Technical Report helps to characterize what a CM tool can and cannot do in the CM process.

This Technical Report provides guidance for tool manufacturers in implementing a minimum set of capabilities. The capabilities defined in this Technical Report are linked to ISO/IEC 12207 and ISO/IEC 15288, and will provide tool manufacturers with guidance on the characteristics their tools should support to meet these International Standards.

Information technology — Systems and software engineering — Guide for configuration management tool capabilities

1 Scope

This Technical Report provides guidance for configuration management tool capabilities from which systems and software development life cycle activities can be supported.

ISO/IEC 14102:2008, *Information technology — Guideline for the evaluation and selection of CASE tools*, details a set of evaluation criteria for CASE tools without referencing a specific activity or task which the tool supports. This lack of consideration on a specific activity or task causes users confusion and difficulty in evaluating and selecting the right tools.

This Technical Report supplements ISO/IEC 14102:2008 by providing a set of minimum tool capabilities for configuration management. It can be used as the set of criteria by a potential user during an acquisition process, or by a configuration management tool supplier to help identify desirable tool capabilities.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 12207:2008, *Systems and software engineering — Software life cycle processes*

ISO/IEC 15288:2008, *Systems and software engineering — System life cycle processes*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
attribute
property associated with a set of real or abstract things that is some characteristic of interest

3.2
baseline
version of a configuration, specification, or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures

[ISO/IEC 12207:2008 and ISO/IEC 15288:2008]

3.3
branch
deviation from the main development line for a configuration item, which allows different persons to work on the same item at the same time

3.4
build
process of generating (archiving) an executable and testable system from source versions or baselines

NOTE The build needs to compile and link the various versions in the correct order. The build tools can be integrated into a configuration management tool.

3.5
change request
CR
formal procedure for submitting a request for an adjustment of a configuration item

3.6
configuration item
entity within a configuration that satisfies an end use function and that can be uniquely identified at a given reference point

[ISO/IEC 12207:2008 and ISO/IEC 15288:2008]

3.7
configuration management
CM
coordinated activities to direct and control configuration

3.8
CM services
abstract description of work done by CM tools

NOTE A service is self contained, coherent, discrete, and can be composed of other services.

3.9
CM tool
software product that can assist software engineers by providing automated support for configuration management activities

3.10
configuration status accounting
CSA
element of configuration management that consists of the recording and reporting of information needed to manage a configuration effectively

[ISO/IEC 24765]

3.11
delta
difference between two versions

3.12
software/system element
element that defines and prescribes what a software or system is composed of (for example, requirements, design, code, test cases, and version number)

NOTE An element can contain sub elements or other software/system elements that are dependent on the top level element.

3.13
release
particular version of a configuration item that is made available for a specific purpose (for example, test release)

3.14
traceability
degree to which each element in a software development product establishes its reason for existing

3.15
version
identified instance of a configuration item

NOTE Modification to a version of a software product, resulting in a new version, requires configuration management action.

3.16
version identifier
supplementary information used to distinguish a version of a configuration item from other versions

NOTE Version numbers are used to compare the version of the software product against another version.

4 Application of this Technical Report

4.1 Overview

This clause presents the benefits to groups of people that acquire, supply, develop, operate, and maintain a CM tool. The objective is to provide a road map for the users of this Technical Report so that they can orient themselves in it and apply it judiciously.

4.2 CM personnel

Personnel involved in the performance of one or more CM activities will benefit from this Technical Report as follows:

— obtain a better understanding of the relationship between the activities in which they are involved and CM tool capabilities

— identify processes or activities that can be improved through better support by a CM tool

— have an objective basis for a better comparison, evaluation, and assessment of CM tools

4.3 Tool suppliers

Suppliers of software engineering tools will benefit from this Technical Report as follows:

— develop CM tools consistent with the International Standards ISO/IEC 12207:2008, ISO/IEC 15288:2008, and ISO/IEC 14102:2008

— provide CM tools that can be shown to support an internationally accepted set of capabilities

4.4 Acquirers

People involved in the purchase of CM tools will benefit from this Technical Report as follows:

— review CM services that can contribute to CM process improvement

— identify criteria for selecting CM tools

— compare competing CM tools based upon this Technical Report

5 Capabilities of configuration management tools

5.1 Overview of configuration management tool capabilities

Throughout the systems and software life cycle, different artifacts (e.g., hardware, software, documents) exist in different versions at different times and changes arise constantly. Configuration Management (CM) verifies and assures that a product performs as intended by providing visibility and control of product's functional and physical characteristics. A CM tool can provide automated assistance for CM activities: configuration identification, change management, reports, status accounting, and auditing. This Technical Report provides the tool capabilities for the automation of CM activities to support software and systems lifecycle processes.

NOTE The following documents have been reviewed for identifying the CM tool capabilities: ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 15846, ISO/IEC TR 19759 SWEBOK, ISO/IEC 14102:2008, ISO 10007:2003, ISO/IEC 15940:2006, IEEE 828:1990, ANSI/EIA 649:1998, and commercial tool information. The life cycle processes and activities from these references have been used as the basis for the CM tool capability categorization found in this document.

5.2 Configuration management tool capabilities

5.2.1 Overview

The CM activities described in ISO/IEC 12207:2008 and ISO/IEC 15288:2008 includes configuration identification, configuration baselining, configuration control, configuration status accounting, configuration auditing, and release management and delivery. Besides of these CM activities, integrating CM tools into the software development environment is also a key for automated CM support. The CM tool should provide not only the capabilities supporting the CM activities but also the tool integration capability into other tools and platforms.

5.2.2 Configuration identification

Configuration identification should capture the attributes of the software to be controlled: the software contents, the different versions of contents, the operation data, and any other essential elements that constitute the configuration item. The tools should provide the following capabilities for the configuration items:

— identifying configuration items

— specifying configuration item relationship

— defining states and specifying status of configuration items

NOTE Refer to ISO/IEC 12207:2008 and ISO/IEC 15288:2008 for the output of software lifecycle process, activity, and task.

5.2.3 Configuration baselining

For each configuration item and its versions, the tool should provide the following capabilities for baselines:

— creating baseline

— storing the baseline with its version references and identification details

— tracing baselines

— reporting baseline status

5.2.4 Configuration control

The tool should provide the following capabilities for configuration control:

— controlling changes

— controlling access

— controlling versions

5.2.5 Configuration status accounting

The tool should provide the following capabilities for status accounting:

— recording status information

— tracing the status

— reporting status of configuration management items

5.2.6 Configuration auditing

The tool should provide the following capabilities for configuration audits:

— planning audit

— auditing the functional completeness of configuration items against requirements

— auditing the physical completeness of the configuration items (whether their design and code reflect an up-to-date technical description)

— reporting audit results

— analyzing anomaly

5.2.7 Release management and delivery

The tool should provide the following capabilities for release management and delivery:

— building software products

— storing the master copies of the code and documentation over the life of the software product

— replicating, packaging, and delivering of code and documentation in accordance with the policies of the organizations involved

— tracking build and release identification

5.2.8 Other configuration management tool capabilities

The tool should provide the following general capabilities:

— integrating with other tools and platforms

5.3 Configuration identification

5.3.1 Overview

Configuration identification is the basis for the subsequent control of the software configuration. This activity includes identifying items to be controlled, establishing an identification scheme for the items and their versions, and specifying the relationship and status of the configuration items.

5.3.2 Identifying configuration items

Configuration item identification should capture at a minimum the following characteristics of the software/system elements to be controlled:

— software/system element contents (e.g., requirements, design, analysis, code, test cases, and traceability between the elements)

— different versions of configuration item and its constituent components

— software/system element operation data (e.g., history of change, version number, number of change requests)

— name of configuration item.

The tool should provide the following capabilities for configuration item identification:

— generating a unique identifier to distinguish each item (e.g., alphabetic, alphanumeric)

— providing a means of documentation (e.g., templates or forms) of the configuration information to be used to describe each configuration item (e.g., configuration identification number, link to other configuration item, link to software structure, link to baseline, link to version hierarchy, link to storage, owner, creation date, version number, and configuration status)

— linking to change requests associated with the configuration item.

5.3.3 Specifying configuration item relationship

The structural relationships among the configuration items affect other CM activities or tasks, such as software building or analyzing the impact of proposed changes. The tool should provide the following capabilities for configuration item relationships:

— the tracing to and from the configuration items

— the tracing the configuration item relationship links both upward and downward in the software hierarchy

— the mapping of the identified configuration items to the elements of software structure (e.g., system, sub-system, component, library, or unit code).

5.3.4 Defining states and specifying status of configuration items

A configuration item may have a lifecycle that goes through different states (e.g., checked-in, checked-out, change-requested, change-approved, change-rejected, change-reviewed, change-tested, and change-completed). Changes in state are triggered by conditions or events (e.g., submit, retrieve, change-request-submit, change-approve, change-review, change-reject, testing, and change-complete).

The status of a configuration item is defined by the state in which it exists at any given time.

The tool should provide the following capabilities for configuration item status:

— defining the set of states of configuration items

— defining the conditions or events that enable or enact the change of configuration item status

— specifying the status of configuration items.

5.4 Configuration baselining

5.4.1 Overview

A baseline is a set of configuration items designated at a specific time during the software life cycle. It represents the approved status of configuration items and provides the point of departure for further evolution. A configuration item belongs to more than one baseline. Baselining activities include the creating, storing, tracing, access controlling, and reporting of baseline.

5.4.2 Creating a baseline

Creating a baseline consists of identifying which configuration items are part of it. The tool should provide the following capabilities for initiating baselines:

— naming the configuration items and versions specified in the configuration item hierarchy that comprise the baseline

5.4.3 Storing baseline

The baseline shall be stored with the detailed attributes associated with acceptance, change, support, and disposal of baseline. The tool should provide the following capabilities for storing baselines:

— placing the baseline attributes into a repository, (e.g., creation date, owner, acceptance criteria, and change history)

— the placing of associated tool information for versioning, build, and release

— searching the attribute information of baseline from the repository

— changing the attribute information from the repository

5.4.4 Tracing baseline

Traceability is a key capability that helps investigate the history of every change in a baseline chain. The tool should provide the following capabilities for tracing baselines:

— the tracing across baselines (e.g., the system requirements baseline, software requirements baseline, design baseline, and delivery baseline)

— tracing the systems and software artifacts including derived ones (e.g., the project plan, requirement definition document, standards, system analysis document, system design document, prototype, system test plan and specification, program source code, object code and executable, unit test plan and specification, test and project data, and user manuals)

5.4.5 Reporting baseline status

The status report on baseline should include a range of baseline information. The tool should provide the following capabilities for reporting baseline status:

— searching for the baseline information at a minimum by configuration item, change history, status, release identifier, and links to other baselines

— generating a status report on a baseline at a minimum by configuration item, change history, status, release identifier, and links to other baselines

— reporting change requests associated with a baseline

5.5 Configuration control

5.5.1 Overview

Software configuration control is concerned with managing changes during the software life cycle. It covers the activities for controlling changes, controlling access to the configuration item, and controlling versions resulting from the modification of configuration items.

5.5.2 Change control

5.5.2.1 Overview

Change control covers the activities for determining what changes to make, approving changes, and implementing changes as follows:

— proposing the change request

— analyzing the changes

— approving or disapproving the request

— implementing and releasing the changes

— communicating the disposition

— tracking the changes

5.5.2.2 Proposing changes

Change request needs formal procedures for submitting and recording the anomaly or enhancement from the baselined configuration item. The tool should provide the following capabilities for proposing changes:

— providing a template or a form for change request which includes at a minimum, configuration and version identification number, requester, date of request, reason for change, priority of the proposed changes

— storing the change request information into a repository and notifying the request to the authorized person by using email, groupware, or any other electronic communication means

— querying and tracing the change request at a minimum, by requester, date, configuration identification number, and reason for change

5.5.2.3 Analyzing changes

Impact analysis is required to determine the extent of modifications that the change request. The tool should provide the following capabilities for analyzing changes:

— tracing configuration items and related baselines affected by the proposed change

— tracing any approved modifications affecting the identified configuration items or baselines

5.5.2.4 Approving/disapproving change requests

The authority for accepting or rejecting proposed changes depends on a Configuration Control Board (CCB). The board reviews each proposed change, approve or disapprove it, and if approved, coordinate the change with the affected group. The tool should provide the following capabilities to support the CCB:

— providing a template or a form for describing the scope and reason of the change (e.g., accept, modify, reject, or defer)

— enabling to specify the associated information (e.g., change priority, change effort, approver, responsible persons for the change, due date for the change implementation)

— providing a means of recording the decision of the CCB

5.5.2.5 Implementing/releasing the changes

After the change request is approved, modification should be performed on only the approved modification. To implement and release each approved change, the tool should provide the following capabilities:

— tracking of which change requests are incorporated into particular software versions and baselines

— allowing only the versions and baselines which are associated to the approved modifications be changed in a workspace

— monitoring that the changes are completed on time and are implemented by the authorized person

— releasing the changed versions to the designated medium and destination

— the roll back of a configuration item so that at any point, a configuration item can be brought back to its previous state

5.5.2.6 Communicating disposition of change requests

The designated changes should be distributed by notifying everyone impacted by the disposition. The tool should provide the following capabilities for communication dispositions:

— notifying those who use the affected configuration items if the change is approved

— notifying those who proposed change if the change is rejected or deferred

5.5.2.7 Tracking changes

Managing the audit trail for each configuration item is important in controlling the changes. The tool should provide the following capabilities for tracking changes:

— tracking the configuration items and versions by change attributes (e.g., the reasons of change, change request date, change requester, change completion date, configuration identification number, or change status)

— tracking the status of the modification (e.g., implemented, verified, released)

— representing the status of changes in a reporting form (e.g., table, graph, or chart)

5.5.3 Access control

5.5.3.1 Overview

Access control ensures the protection of information and data so that unauthorized persons or systems cannot read or modify the information and, at the same time, authorized persons or systems are not denied access to the information and data.

5.5.3.2 Specifying and maintaining access

Controlling access to configuration items starts with specifying the access permission and security levels. The tool should provide the following capabilities for specifying and maintaining access:

— specifying different access permissions to configuration items with different status (e.g., login, check in, check out for read, check out for modification, change request, change approved/rejected, change implemented, release, audit)

— specifying different access permissions by user, group, or role

— specifying different access permissions to different granularities (e.g., line, code, function, module)

— maintaining the log for all the access to the configuration item by permission and granularity

— providing an encryption method to ensure secure usage and transmission of data

5.5.3.3 Concurrency control

Different users should be able to gain access concurrently to the configuration information. The tool should provide the following capabilities for concurrent control:

— producing the latest version of a branch when a version is retrieved for modification

— ensuring that only one person at a time can create a new version using a locking mechanism, forcing serialized changes to any given version

— enabling parallel concurrent development by allowing multiple users access to different versions at the same time

5.5.4 Version control

5.5.4.1 Overview

A version consists of a file or a set of files, each with a particular version identification number. Version control maintains multiple versions or a set of related versions of an application. The fundamental operations associated with version control are version creation, version modification, version merge, and version comparison.

5.5.4.2 Version creation

A new version should be created if changes are anticipated which should not affect the existing version.

The tool should provide the following capabilities for version creation:

— identifying the version of the configuration item to be changed

— placing a new version into the version branches (e.g., sequential, multiple)

5.5.4.3 Version modification

Version modification allows an existing version to evolve by creating a new version.

The tool should provide the following capabilities:

— a means of modifying a version by creating a new version.

— a means of preventing simultaneous conflicting modifications by different users to the same version of a configuration item.

NOTE The challenge here is in a multi-user environment in which several users may wish to make simultaneous changes to the same version. A common solution to this challenge is to use a check-in/check-out mechanism, which prevents a configuration item from being edited by multiple users. Other mechanisms allow multiple changes that are later merged.

5.5.4.4 Merging versions

Merging is the act of reconciling multiple changes made to different version of the same file. The changes are merged, resulting in a single new version that contains both sets of changes. In some cases, the merge can be performed automatically, e.g., the changes don't conflict. In other cases, a person shall decide exactly what the resulting version should contain. The tool should provide the following capabilities for merging versions:

— merging the content of designated versions

— propagating the merge result to the subsequent versions along the version tree

— recording the merge history in the repository (e.g., date and time, owner, types of merge, location of changes, new version identification number)

— placing a new merged version in the version tree or graph

5.5.4.5 Comparing versions

Two versions of a software/system element may be statically compared, providing a list of the differences, called deltas.

The tool should provide the following capabilities for managing deltas:

— comparing two versions of a software/system element and identifying the deltas

— undoing selected deltas

— applying selected deltas to other versions of the same software/system element

5.5.4.6 Change history

A record may be kept of the sequence of changes made to a version of a software/system element, This information may then be used to roll back a sequence of changes within the version.

The tool should provide the following capabilities for managing change history:

— recording the sequence of changes made to a software/system element

— reporting the sequence of changes made to a software/system element

— undoing changes

5.5.4.7 Granularity

Configuration items will vary in size and granularity (e.g., line, functions modules or files). The appropriate granularity should be specified and document for version control. The tool should provide the following capabilities for configuration item granularity:

— specifying the granularity of version, access permission, delta, and merge (e.g., line, functions, modules, files)

— specifying the granularity changes on version, access permission, delta, and merge

5.6 Configuration status accounting

5.6.1 Overview

Configuration status accounting ensures that configuration status is recorded, monitored, and reported. Configuration status can be identified and managed by tracing the changes to a configuration item in the chain.

5.6.2 Recording status information

To enable the configuration status accounting, the configuration item status information should be identified and maintained. The tool should provide the following capabilities for recording status information:

— recording the identification and status of each new and modified configuration item and of baseline

— recording the version and status of each modification (e.g., version/release identifiers, change requested, and change approved)

5.6.3 Tracing the status

To capture and search the status and history of configuration items, the tool should provide the following capabilities:

— tracing the configuration and version identification number (e.g., number of changes to the latest versions, comparisons of versions, the number of releases)

— tracing the other documents affected by each change and update

— tracing the implementation of approved modifications

5.6.4 Configuration status reporting

Reported information can be used by various organizations and project elements. Reporting can take the form of ad hoc queries or the periodic production of pre-designed reports. The tool should provide the following capabilities for reporting and status accounting:

— change/problem tracking and reporting (e.g., who made the change, when initiated the change, change history, and change request (CR) status)

— allowing users to tailor the information in various formats (e.g., all unassigned CRs, all pending CRs, all CRs assigned to a particular person, CRs sorted by date, severity, priority, classification, completion date, status, and other criteria)

— generating difference reports containing the differences between two versions of an item

— generating release reports containing the release identification and status

— providing event or transaction log with the history of CM activities performed (e.g., transaction number date, originator, nature of the entry, affected items, activity, description, participants, impacted items, and remarks)

— allowing users to write queries to get the configuration information in their own format

— tracking among change requests to source problems (e.g. a failed test may generate multiple change requests)

5.7 Configuration auditing

5.7.1 Overview

Configuration auditing verifies that the software system matches the configuration item specification and that the product package contains all of the components it is supposed to contain and performs as promised. Effective auditing should support the verification and validation processes to ensure completeness and integrity of software products (functional, physical, baseline auditing). Configuration auditing consists of activities including audit planning, executing auditing, reporting auditing results, and resolving anomalies.

5.7.2 Audit planning and procedures

Audit plan describes all CM aspects (e.g., resources and procedures) that the auditing shall include. The tool should provide the following capabilities for the audit planning:

— specifying unique identifiers of configuration items and their current status

— specifying configuration baselines, releases, and their status

— identifying the latest configuration item versions and reporting their status

— specifying the person responsible for each CM activity (e.g., configuration identification, baselining, change control, status accounting, and auditing)

— specifying build instruction and tools used to develop, produce, test, and verify the product

— tracing change history and generating an audit trail

5.7.3 Executing auditing

5.7.3.1 Overview

Auditing activity determines the extent to which an item satisfies the required functional and physical characteristics. Audits check the completeness of the software, system, or product.

5.7.3.2 Functional configuration auditing

Functional auditing verifies that a configuration item's actual performance agrees with the requirements specified in the requirements definition and system design documents. The tool should provide the following capabilities for functional audits:

— tracing and maintaining input documents for testing that include test plans and test data

— test methods to trace that all functional parameters that were tested and test results

5.7.3.3 Physical configuration auditing

Physical auditing is to verify that a configuration item conforms to the technical documentation that defines it. The tool should provide the following capabilities for physical audits:

— tracing across configuration items, versions, links to software structure, and release structure

— specifying the product baseline

— checking that the software product specification and version identification are consistent with the real software product (e.g., source code, user documents, or any other configuration items)

5.7.4 Reporting auditing results

After completing an audit, the audit results should be documented and reported. The audit results contain any problems found in the audit and related problem resolutions planned. The tool should provide the following capabilities for reporting audit results:

— searching and tracing the configuration items, build, or release by configuration information (e.g., identifiers, date, change requester)

— searching and tracing the latest configuration item versions for a system build and application

— reporting the metric statistics (e.g., number of (approved) change requests for a system, the number of baselines and releases, the usage of configuration items in build and release, average time taken for the resolution of change request, time spent on development, testing, number of defects found after each release, time taken to identify the source of a defect, time taken to resolve a defect, amount of reused code, and the number of configuration items impacted by a change request)

— comparing baselines and releases, and reporting the differences

— report anomalies

— customizing report generation and exporting the reports to other applications such as word processors, spread sheets

5.7.5 Analyzing anomaly

After the execution of audit, any anomaly reported need to be examined and analyzed for a future resolution. The tool should provide the following capabilities for anomaly/problem resolution:

— tracking the anomaly/problem from its origin and capture the details of the problem, (e.g., as originator, date of problem identification, cause of the problem, how and when it was fixed, how much time was taken, and what kind of skill were needed)

— reporting the identified anomaly to the person responsible for coordinating the anomaly/problem resolution

— creating automatic alerts based on the discovery of the anomaly/problem

— tracing all the change requests that are associated with particular configuration item, reasons for change, change request status, change result, and the configuration items that the change affects

— providing a checklist or a form to ensure that all discovered anomalies are recorded, analyzed, and resolved

— storing problem resolution results into a repository so that users can look up the knowledge base to determine if the same problem has occurred

5.8 Release management and delivery

5.8.1 Overview

Release management refers to the distribution of a software configuration item outside the development activity. The release and delivery of software products and associated documentation requires that master copies of source code and associated documentation be maintained for the life of the software product.

5.8.2 Building

Capturing the details of every build for a system to be tested or released is an important activity. The tool should provide the following capabilities for building configuration items:

— tracing configuration items to find out which configuration items are used in the build and what versions of the configuration items are used

— grouping configuration items by marking the appropriate version of each configuration item with a symbolic name or identification number that is specific to the release

— scanning automatic source code

— maintaining the traceability of software artifacts so that the software products and development environment can be changed and reproduced in the future

— retaining the baseline for the software library and environment

— programming and running build scripts

— creating build audit trails

5.8.3 Storing

Before versions of the software products are released and delivered, the released versions and their associated documentation should be stored and maintained in a repository. The tool should provide the following capabilities for storing:

— maintaining the history of the previous builds and releases in the repository

— selecting storage medium

— marking the identification of the release in terms of physical medium (e.g., CD-ROM or magnetic tape) and electronic releases (e.g., download into operational libraries)

5.8.4 Replicating

Replication is a manufacturing stage for software by making copies. The tool should provide the following capabilities for replication:

— copying the configuration items, versions, and other related documents into a designated place

— checking that the release contains no extraneous items (e.g., software viruses or test data)

— preserving the integrity of the contents by freezing or locking the version during replication to prevent modification during or after delivery

5.8.5 Packaging and delivery

Packaging and delivery should be done by approved procedures and clearly marked with the release identification. The tool should provide the following capabilities for packaging and delivery:

— archiving a copy with the configuration items including the product information (e.g. license agreement and copyright statement).

5.8.6 Tracking the distribution of products

After the delivery of the product, the bug reports and change requests can be submitted from customers and systems. The tool should provide the following capabilities for tracking the distribution of the product:

— tracking the distribution of products to customers

— connecting the release management process to the change request process in order to map release contents to the change requests

5.9 Other configuration management tool capabilities

Integrating CM tools into the software development environment is a key capability to providing automated CM support in software development environments. The tool should allow the CM tool to integrate with the software development environment by:

— providing integration with CASE tools, code generators, test data generators, automatic testing tools, and code analyzers

— supporting the interface with a variety of control and data integration

— separating the interface from the service mechanisms

— enabling the use of multiple implementations of environment services

— sharing the workspace with CASE tools so that users can perform relevant CM functions within the shared workspace

Annex A
(informative)

Focus areas of each reference

A.1 Configuration management focus area

A.1.1 General

The features and processes of configuration management are found in many standards. Each of the following references describes some sets of CM capabilities appeared in this Technical Report.

A.1.2 ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes

- management of CM process
- configuration control
- configuration status accounting
- configuration evaluation (auditing)
- release management

A.1.3 ISO/IEC 15288:2008 Systems and software engineering - System life cycle processes

- configuration management strategy
- identifying items that are subject to configuration control.
- maintaining information on configurations with an appropriate level of integrity and security
- ensuring that changes to configuration baselines are properly identified, recorded, evaluated, approved, incorporated, and verified.

A.1.4 ISO/IEC 14102:2008 Information technology- Guideline for the evaluation and selection of CASE tools

- access control
- tracking of modifications
- definitions and management of multiple versions
- configuration status accounting
- release generation
- archival capability

A.1.5 ISO/IEC TR 19759 Software engineering – Guide to the Software Engineering Body of Knowledge (SWEBOK)

- management of CM process
- configuration identification
- configuration control
- configuration status accounting
- configuration auditing
- release management and delivery

A.1.6 ANSI/EIA 649 National Consensus Standard for Configuration Management

- configuration management planning and management
  - identifying context and environment
  - configuration management plan
  - implementation procedures
  - training
  - performance measurement
  - supplier configuration management

- configuration identification
  - product information
  - product structure
  - product identifiers
  - documents identification
  - baselines
  - product identification recovery
  - interface control

- configuration control management
  - change identification
  - change evaluation and coordination
  - change implementation and verification
  - change management process applied to variances

- configuration status accounting
  - CSA information
  - CSA system

- configuration verification and audit
  - design and document verification
  - configuration audit
  - continuing performance audits and surveillance

- configuration management of digital data
  - digital data identification
  - data status level management
  - maintenance of data and product configuration relationships
  - data version control and management of review, comment, annotation and disposition
  - digital data transmittal
  - data access control

A.1.7 Mapping of references

The following table summarizes and maps the related references to the key CM capabilities described in this Technical Report:

Table A.1 – CM capabilities and related references

CM capabilities	Related references
Process management (instantiating and defining, scope, plan, controlling execution, reviewing and evaluation, closing)	ISO/IEC TR 19759, ANSI/EIA 649, ISO/IEC 15288:2008
Configuration identification	ISO/IEC 14102:2008, ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759, ANSI/EIA 649
Configuration baselining	ISO/IEC 14102:2008, ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759, ANSI/EIA 649
CM capabilities	**Related references**
Configuration control (version control, access control, change control, release control)	ISO/IEC 14102:2008, ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759, ANSI/EIA 649
Configuration status accounting	ISO/IEC 14102:2008, ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759, ANSI/EIA 649
Configuration auditing	ISO/IEC 14102:2008, ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759, ANSI/EIA 649
Release management and delivery	ISO/IEC 14102:2008, ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759, ANSI/EIA 649
Other requirements (e.g., integration with other tools, non-functional characteristics)	ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model IEEE 828:1990 IEEE Standard for Software Configuration Management Plans

Annex B
(informative)

Configuration management services

B.1 Overview of configuration management service

B.1.1 Introduction to CM as a service

CM practitioners may need to understand what services CM or CM tool can provide for their software lifecycle activities and tasks. Better understanding and implementation of CM services supports users in attaining a higher quality of product, with more time for being productive on creative tasks and better forecasting of software costs. The CM services can provide a mean for evaluating and choosing sets of compliant CM capabilities in supporting software life cycle processes.

B.1.2 Rationale of CM services categorization

The CM services described in this annex reflect broad functional activities of a typical software engineering organization found in ISO/IEC 12207:2008, ISO/IEC 15288:2008, ISO/IEC TR 19759 SWEBOK, ISO/IEC 14102:2008, and ISO/IEC 15940:2006 (Check Annex A for the focus areas of each reference). The lifecycle processes from these references are analyzed and regrouped in a view point of services as following:

a) project management support services
   1) requirement management service
   2) software project planning service
   3) software subcontract management service
   4) risk management service

b) process improvement support services
   1) process establishment service
   2) process assessment service
   3) continuous process improvement service

c) quality assurance support services
   1) product assurance service
   2) process assurance service

d) reuse support services

B.1.3 Structure of service description

Each service is defined under three headings:

- Service concept, to provide a description of the service in terms that are not related to a specific implementation.
- Basic service operations, to list those operations to be included in a service. This list of operations represents, in most cases, only primary services. List is not intended to be complete.
- Automated service operations, provide a list of operations to be automated. This list is a subset of the operations listed under 'Basic service operations'.

B.2 Project management support services

B.2.1 Requirements management service

Service concepts

Requirements management is the process of establishing the common understanding on the project between the customer and the developer. Since the customer's requirement frequently changes throughout the software life cycle, maintaining, controlling, and tracing of the changes are important for the success of a project.

Basic requirements management service operations are as follows:

— identifying requirements baseline as configuration item

— modifying requirements baseline

— browsing requirements baseline

— reporting and displaying requirements baseline

— managing traceability between the requirement baseline and other documents

— auditing validation and verification of requirements

Automated requirements management service operations are as follows:

— baselined (customer) requirements are maintained

— changes to (customer) requirements are traced and change status is reported

— baselined requirements specifications are maintained

— changes to requirements specifications are traced, and change status is reported

— the dependencies between requirement documents and other documents is stored and traced