# Configuration Management

# 5

**Abstract**

This chapter discusses configuration management and discusses the fundamental concept of a baseline. Configuration management is concerned with identifying those deliverables that must be subject to change control and controlling changes to them.

**Keywords**

Configuration management system · Configuration items · Baseline · File naming conventions · Version control · Change control · Change control board · Configuration management audits

## 5.1 Introduction

Software configuration management (SCM) is concerned with tracking and controlling changes to the software and project deliverables, and it provides full traceability of the changes made during the project. It provides a record of what has been changed, as well as who changed it. SCM involves identifying the configuration items of the system; controlling changes to them; and maintaining integrity and traceability.

The origins of software configuration management go back to the early days of computing when the principles of configuration management used in the hardware design field were applied to software development in the 1950s. It has evolved over time to a set of procedures and tools to manage changes to the software.



G. O'Regan, *Concise Guide to Software Engineering*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-57750-0_5

The configuration items are generally documents in the early part of the software development lifecycle, whereas the focus is on source code control management and software release management in the later parts of development. Software configuration management involves:

- Identifying what needs to be controlled
- Ensuring those items are accurately defined and documented
- Ensuring that changes are made in a controlled manner
- Ensuring that the correct version of a work product is being used
- Knowing the version and status of a configuration item at any time
- Ensuring adherence to standards
- Planning builds and releases.

Software configuration management allows the orderly development of software, and it ensures that only authorized changes to the software are made. It ensures that releases are planned and that the impacts of proposed changes are considered prior to their authorization. The integrity of the system is maintained at all times, and the constituents of the software (including their version numbers) are known at any time.

Effective configuration management allows questions such as the following (Table 5.1) to be easily answered:

The symptoms of poor configuration management include corrected defects that suddenly begin to reappear, difficulty in or failure to locate the latest version of source code or failure to determine the source code that corresponds to a software release.

Therefore, it is important to employ sound configuration management practices to enable high-quality software to be consistently produced. Poor configuration management practices lead to quality problems resulting in a loss of the credibility and reputation of a company. Several symptoms of poor configuration management practices are listed in Table 5.2.

**Table 5.1** Features of good configuration management

| Features of good configuration management |
|---|
| What is the correct version of the software module to be updated? |
| Where can I get a copy of R4.7 of software system X? |
| What versions of the software system X are installed at the various customer sites? |
| What changes have been introduced in the new release of software (version R4.8 from the previous release of R4.7)? |
| What version of the design document corresponds to software system version R3.5? |
| What customers use R3.5 of the software system? |
| Are there undocumented or unapproved changes included in the released version of the software? |

**Table 5.2** Symptoms of poor configuration management

| Symptoms of poor configuration management |
|---|
| Defects corrected suddenly begin to reappear |
| Cannot find the latest version of the source code |
| Unable to match the source code and object code |
| Wrong version of software sent to the customer |
| Wrong code tested |
| Cannot replicate previously released code |
| Simultaneous changes to same source component by multiple developers with some changes lost |

Configuration management involves identifying the configuration items to be controlled and systematically controlling change to them, in order to maintain the integrity and traceability of the configuration throughout the software development lifecycle. There is a need to manage and control changes to documents and source code, including the project plan, the requirements document, design documents, code and test plans.

A key concept in configuration management is that of a "*baseline*," which is *a set of work products that have been formally reviewed and agreed upon and serves as the foundation for future development work.*

A baseline can only be changed through formal change control procedures, which leads to a new baseline. It provides a stable basis for the continuing evolution of the configuration items, and all approved changes move forward from the current baseline leading to the creation of a new baseline. The change control board (CCB) or a similar mechanism authorizes the release of baselines, and the content of each baseline is documented. All configuration items must be approved before they are entered into the released baselines.

Therefore, it is necessary to identify the configuration items that need to be placed under formal change control and to maintain a history of the changes made to the baseline. There are four key parts to software configuration management (Table 5.3).

A typical set of software releases (e.g., in the telecommunications domain) consists of incremental development, where the software to be released consists of a number of releases builds with the early builds consisting of new functionality, and the later builds consisting of fix releases.

Software configuration management is planned for the project, and each project will typically have a configuration management plan which will detail the planned delivery of functionality and fix release for the project (Table 5.4).

Each of the R.1.0.O.*k* baselines are termed release builds, and they consist of new functionality and fixes to the identified problems. The content of each release build is known; i.e., the project team and manager will target specific functionality and fixes for each build, and the actual content of the particular release baseline is documented. Each release build can be replicated, as the version of source code to create the build is known, and the source code is under control management.

**Table 5.3** Software configuration management activities

| Area | Description |
|---|---|
| Configuration identification | This requires identifying the configuration items to be controlled and implementing a sound configuration management system, including a repository where documents and source code are placed under controlled access. It includes a mechanism for releasing documents or code, a file naming convention and a version numbering system for documents and code and baseline/release planning. The version and status of each configuration item should be known |
| Configuration control | This involves tracking and controlling change requests and controlling changes to the configuration items. Any changes to the work products are controlled and authorized by a change control board or similar mechanism. Problems or defects reported by the test groups or customer are analyzed, and any changes made are subject to change control. The version of the work product is known, and the constituents of a particular release are known and controlled. The previous versions of releases can be recreated, as the source code constituents are fully known and available |
| Configuration auditing | This includes audits to verify the integrity of the baseline, and audits of the configuration management system verify that the standards and procedures are followed. The results of the audits are communicated to the affected groups, and corrective action is taken to address the findings |
| Status accounting | This involves data collection and report generation. These reports include the software baseline status, the summary of changes to the software baseline, problem report summaries and change request summaries |

**Table 5.4** Build plan for project

| Release baseline | Contents | Date |
|---|---|---|
| R. 1.0.0.0 | $F_4$, $F_5$, $F_7$ | 31.01.17 |
| R. 1.0.0.1 | $F_1$, $F_2$, $F_6$ + fixes | 15.02.17 |
| R. 1.0.0.2 | $F_3$ + fixes | 28.02.17 |
| R. 1.0.0.3 | $F_8$ + fixes (functionality freeze) | 07.03.17 |
| R. 1.0.0.4 | Fixes | 14.03.17 |
| R. 1.0.0.5 | Fixes | 21.03.17 |
| R. 1.0.0.6 | Official release | 31.03.17 |

There are various tools employed for software configuration management activities, and these include well-known tools such as Clearcase, PVCS and Visual Source Safe (VSS) for source code control management. The PV tracker tool and Clearquest may be used for tracking defects and change requests. A defect-tracking tool will list all of the open defects against the software, and a defect may require several change requests to correct the software (as a problem may affect different parts of the software product as well as different versions of the product, and a change request may be necessary for each part). The tool will generally link the

**Table 5.5** CMMI requirements for configuration management

| Specific goal | Specific practice | Description of specific practice/goal |
|---|---|---|
| SG 1 | | *Establish baselines* |
| | SP 1.1 | Identify configuration items |
| | SP 1.2 | Establish a configuration management system |
| | SP 1.3 | Create or release baselines |
| SG 2 | | *Track and control changes* |
| | SP 2.1 | Track change requests |
| | SP 2.2 | Control configuration items |
| SG 3 | | *Establish integrity* |
| | SP 3.1 | Establish configuration management records |
| | SP 3.2 | Perform configuration audits |

change requests to the problem report. The current status of the problem report can be determined, and the targeted release build for the problem identified.

The CMMI provides guidance on practices to be implemented for sound configuration management (Table 5.5).

The CMMI requirements are concerned with establishing a configuration management system; identifying the work products that need to be subject to change control; controlling changes to these work products over time; controlling releases of work products; creating baselines; maintaining the integrity of baselines; providing accurate configuration data to stakeholders; recording and reporting the status of configuration items and change requests; and verifying the correctness and completeness of configuration items with configuration audits. We shall discuss the key parts of configuration management in the following sections.

## 5.2 Configuration Management System

The configuration management system enables the controlled evolution of the documents and the software modules produced during the project. It includes

- Configuration management planning
- A document repository with check in/check out features
- A source code repository with check in/check out features
- A configuration manager (may be a part-time role)
- File naming convention for documents and source code
- Project directory structure
- Version Numbering System for documents
- Standard templates for documents
- Facility to create a baseline
- A release procedure
- A group (change control board) to approve changes to baseline

– A change control procedure
– Configuration management audits to verify the integrity of baseline.

### 5.2.1 Identify Configuration Items

The configuration items are the work products to be placed under configuration management control, and they include project documents, source code and data files. They may also include compilers as well as any supporting tools employed in the project.

The project documentation will typically include project plans, the user requirements specification, the system requirements specification, the architecture and technical design documents and the test plans.

The items to be placed under configuration management control are identified and documented early in the project lifecycle. Each configuration item needs to be uniquely identified and controlled. This may be done with a naming convention for the project deliverables and source code and applying it consistently. For example, a simple approach is to employ mnemonics labels and version numbers to uniquely identify project deliverables. A user requirements specification for project 005 in the finance business area may be represented simply by:

FIN_005_URS

### 5.2.2 Document Control Management

The project documents are stored in a document repository using a configuration management tool such as PVCS or VSS. For consistency, a standard directory structure is often employed for projects, as this makes it easier to locate particular configuration items. A single repository may be employed for both documents and software code (or a separate repository for each).

Clearly, it is undesirable for two individuals to modify the same document at the same time, and the document repository will include *check in*/*check out* procedures. The document must be checked out prior to its modification, and once it is checked out, another user may not modify it until it has been checked back in. An audit trail of all modifications made to a particular document is maintained, including details of the person who made the change, the date that the change was made and the rationale for the change.

**Version Numbering of Documents**
A simple version numbering system may be employed to record the versions of documents: e.g., v0.1, v0.2 and v0.3 is often used for draft documents, with version v1.0 being the first approved version of the document. Each time a document is modified its version number is incremented, and the document history records the reasons for the modification.

– V0.1 Initial draft of document
– V0.*x* Revised draft ($x > 0$)
– V1.0 Approved baseline version
– V1.*x* Approved minor revision ($x > 0$)
– V*n*.0 Approved major revision ($n > 1$)
– V*n*.*x* Approved minor revision ($x > 0$, $n > 1$).

The document will provide information on whether it is a draft or approved, as well as the date of last modification, the person who made the modification, and the rationale for the modification. The configuration management system will provide records of the configuration management activities, as well as the status of the configuration items and the status of the change requests. The revision history of the configuration items will be maintained.

### 5.2.3 Source Code Control Management

The source code and data files are stored in a source code repository using a tool such as PVCS, VSS or Clearcase, and the repository provides an audit trail of all the changes made to the source code. An item must first be checked out for modification, the changes are made, and it is then checked back into the repository. The source code management system provides security and control of the configuration items, and the procedures include:

– Access controls
– Checking in/out configuration items
– Merging and Branching
– Labels (labelling releases)
– Reporting.

The source code configuration management tool ensures the integrity of the source code and prevents more than one person from altering the software code at the same time.

### 5.2.4 Configuration Management Plan

A software *configuration management plan* (it may be part of the project plan or a separate plan) is prepared early in the project, and it defines the configuration management activities for the project. It will detail the items to be placed under configuration management control, the standards for naming configuration items,

the version numbering system, as well as version control and release management.[1] The CM plan is placed under configuration management control.

The content of each software release is documented as well as installation and rollback instructions. The content includes the requirements and change requests implemented, as well as the defects corrected and the version of the new release. A list is maintained of the customer sites of where the release has been installed. All software releases are tested prior to their approval. The CM plan will include:

- Roles and responsibilities
- Configuration Items
- Naming Conventions
- Version Control
- Filing Structure for the project.

The stakeholders and roles involved are identified and documented in the CM plan. Often, the role of a *software configuration manager* is employed, and this may be a full time or part-time role.[2] The CM manager ensures that the configuration management activities are carried out correctly and will conduct and report the results of the CM audits.

## 5.3 Change Control

A change request (CR) database[3] is set up to record change requests made during the project. The change requests are documented and considered by the change control board (CCB). The CCB may just consist of the project manager and the system owner for small projects, or a management and technical team for larger projects.

The impacts and risks of the proposed change need to be considered, and an informed decision made on whether to reject or approve the CR. The proposed change may have technical impacts, as well as introducing new project risks, and may adversely affect the schedule and budget. It is important to keep change to a minimum at the later stages of the project in order to reduce risks to quality.

Figure 5.1 describes a simple process for raising a change request, performing an impact assessment, deciding on whether to approve or reject the change request and proceeding with implementation (where applicable).

The results of the CCB review of each change request (including the rationale of the decision made) will be recorded. Change requests and problem reports for all configuration items are recorded and analyzed, reviewed, approved (or rejected) and tracked to closure.

[1] These may be defined in a Configuration Management procedure and referenced in the CM plan.

[2] This depends on the size of the organization and projects. The project manager may perform the CM manager role for small projects.

[3] This may just be a simple Excel spread sheet or a sophisticated tool.

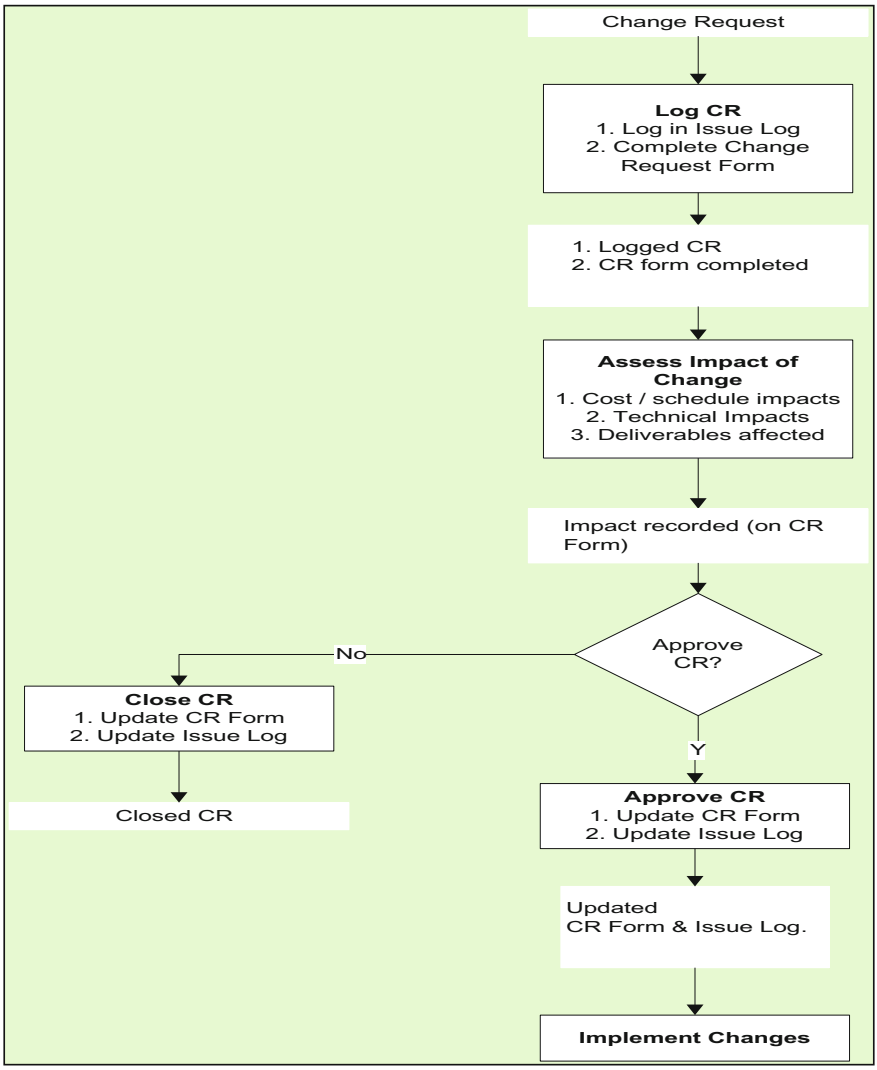


**Fig. 5.1** Simple process map for change requests

A sample configuration management process map is detailed in Fig. 5.2, and it shows the process for updates to configuration information following an approved change request. The deliverable is checked out of the repository; modifications are made and the changes approved; configuration information is updated and the deliverable is checked back into the repository.

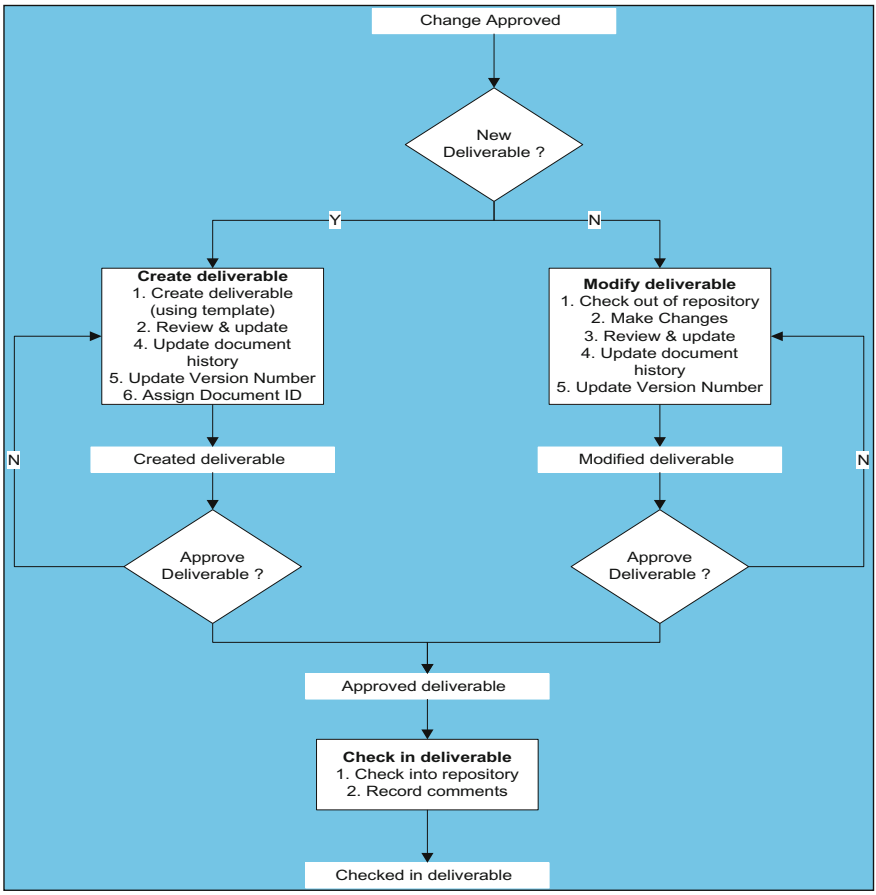


**Fig. 5.2** Simple process map for configuration management

## 5.4 Configuration Management Audits

*Configuration management audits* are conducted during the project to verify that the configuration is consistent and complete. Every project should have at least one configuration audit, and the objective is to verify the completeness and correctness of the configuration system for the project. The audit will check that the records correctly identify the configuration and that the configuration management standards and procedures have been followed. Table 5.6 presents a sample configuration management checklist.

**Table 5.6** Sample configuration management audit checklist

| No. | Item to check |
|---|---|
| 1. | Is the Directory Structure set up for the project? |
| 2. | Are the configuration items identified and listed? |
| 3. | Have the latest versions of the templates been used? |
| 4. | Is a unique document Id employed for each document? |
| 5. | Is the standard version numbering system followed for the project? |
| 6. | Are all versions of documents and software modules in the document/source code repository? |
| 7. | Is the Configuration Management plan up to date? |
| 8. | Are the roles defined in the Configuration Management Plan performing their assigned responsibilities? |
| 9. | Are changes to the approved documents formally controlled? |
| 10. | Is the version number of a document incremented following an agreed change to an approved document? |
| 11. | Is there a change control board set up to approve change requests? |
| 12. | Is there a record of which releases are installed at the various customer sites? |
| 13. | Are all documents/software modules produced by vendors under appropriate configuration management control? |

There may also be a *librarian role* in setting up the filing structure for the project, or the configuration manager may perform this role. The project manager assigns responsibilities for performing configuration management activities. All involved in the process receive appropriate training on the process.

## 5.5 Review Questions

1. What is software configuration management?
2. What is change control?
3. What is a baseline?
4. Explain source code control management.
5. Explain document control management.
6. What is a configuration management audit and explain how it differs from a standard audit?
7. Describe the role of the configuration manager and librarian.
8. Describe the main elements in a software configuration management system.

## 5.6 Summary

Software configuration management is concerned with the orderly development and evolution of the software. It is concerned with tracking and controlling changes to the software and project deliverables, and it provides full traceability of the changes made during the project.

It involves identifying the configuration items that are subject to change control, controlling changes to them, and maintaining integrity and traceability throughout the software development lifecycle. The configuration items are generally documents in the early part of the development lifecycle, whereas the focus is on source code control management and software release management in the later parts of the development lifecycle.

The company standards need to be adhered to, and the correct version of a work product should be known at all time. There is a need for a document and source code repository, which has access controls, checking in and checking out procedures and labelling of releases.

A project will have a configuration management plan, and the configuration manager role is responsible for ensuring that the configuration management activities are carried out correctly.

Configuration management ensures that the impacts of proposed changes are considered prior to authorization. It ensures that releases are planned and that only authorized changes to the software are made. The integrity of the system is maintained, and the constituents of the software system and their version numbers are known at all times. Configuration audits will be conducted to verify that the CM activities have been carried out correctly.