

## Sample Outline Contents: "SW CM Plan" template<sup>1</sup>

### 1. Introduction

#### 1.1 Purpose

In general, software configuration management (SCM) consists of:

- Identifying and defining the configuration items (CIs) in a system
- Controlling the release and change of CIs throughout the system life cycle
- Recording and reporting the status of CIs and change requests
- Verifying the completeness and correctness of CIs

This document defines how these activities are performed in *<Project ID>*. As far as possible, use is made of standard procedures documented in the company's QMS.

It is noted that at least all deliverable SW life cycle data – not just deliverable source & executable code – are subject to CM procedures. *In general, CIs may be single (unitary) entities or may be collections (including hierarchies) of other CIs.*

Each CI defined for *<Project ID>* has a unique identifier, and also has a version number that identifies a stage in its evolution. The CI scheme is such that it can accommodate new CIs without requiring modification of existing CI identifiers.

This plan is re-issued as necessary to reflect more detailed CI data ...

#### 1.2 Scope

The configuration management activities defined in this plan shall apply during the development of *<Project ID>* and, possibly, during its maintenance phase.

#### 1.3 Applicable and reference documents - self-explanatory

#### 1.4 Acronyms - self-explanatory

### 2. Management

The Project Manager, or delegate, acts as the Software Configuration Manager.

### 3. Configuration identification

*Table 3-1 lists the identified configuration items (to be tailored for specific projects):*

<i>Configuration Item (CI)</i>	<i>Identifier</i>
<i>Contract and related documents</i>	
<i>Project plans ...</i>	

#### <sup>1</sup> Template-specific conventions & notes (rough)

1. *< ... >* contain text to be replaced by project-specific data. For emphasis, sometimes in *bold italics*.

2. Guidelines are *italicised* and, in part, may be paraphrased for specific projects.

<b>Configuration Item (CI)</b>	<b>Identifier</b>
<i>Executable object code</i>	<i>See Scode for numbering</i>
<i>Source code</i> <Overall SW name> <Name of first level component> <Name of second level component> ...	X X_1 X_1_1 ...
<i>Requirement, design, User documentation &amp; data</i>	
<i>Applicable standards</i>	
<i>Test designs, cases, procedures, reports</i>	
<i>Other verification documentation</i>	
<i>Problem- and change-related records, etc</i>	

**Table 3-1: Set of configuration items for <Project ID>**

#### 4. Baselines & Traceability

In general, a “baseline” is an assembly of configuration items (e.g. a single document or a complete SW product) that has been formally reviewed and agreed, and is to be a basis for further development. *Formal change control procedures (raising of problem reports, etc) are required to modify a baseline.*

For <Project ID> project baselines are established at the following milestones ●●●

*Integration of the software is coordinated by identification and control of intermediate baselines. It is important to note that the mechanism for incorporating a SW unit level CI (SW unit, for short) into an (intermediate) baseline is by successful execution of the relevant integration test(s). The SW unit’s development is controlled prior to this also, though less formally. Figure 4-1 depicts both the informal and formal configuration control regimes for SW units. It introduces the *logical concepts* of development (or dynamic), master (or controlled) and archive (or static ) libraries, without prejudice regarding their physical realisation.*

All document CIs will have the standard format specified in the company's QMS (including change record, status sheet etc).

All SW code CIs shall have (or there shall be an equivalent record) a header which contains its identifier, author, creation date and change history ...

Storage media used will contain the project name, document configuration item identifier, version date and a contents description.

#### 5. Configuration item, Baseline and Release storage

In general, a software library is a controlled collection of configuration items, it consists of one or more files, and it may exist on various media such as magnetic disk or paper. As illustrated in Figure 4-1 (for SW code CIs) there will be 3 types of library, development, master and archive. *The same library types apply for other CIs too, though the mechanism or criterion for transferring from a development to a master library may differ. For example, a document may be entered in a master library & baselined following a review.*

When a baseline is released, copies of all the relevant master files are taken. These copies are called archive libraries and are **never** modified.

Up to date "security" copies of master & archive libraries will always be available. Procedures for regular back-up of development libraries are per QMS.

## 6. Problem reporting & Change control

In general, software change control is the process of evaluating proposed changes to configuration items and coordinating the implementation of approved changes.

Formal change management of an item does not commence until the item is included in a baseline. Before that, for those CIs that are software code, development is carefully coordinated through source code control (see Figure 4-1) ...

Key mechanism for initiating changes in an item that has come under formal change control is to open a problem report... (*more complex if contractual issues involved*).

*Decision making body for problem reports & for authorising changes, is CCB (change control board). ... composition depends on life-cycle stage etc ...*

The CCB authorises changes and also decides when they should be implemented ...

*Conceptually, all major project reviews include a CCB ...*

## 7. Configuration status accounting

Configuration status accounting is the administrative tracking and reporting of all CIs.

Key instrument for configuration status accounting is the software configuration index (SCI) (or equivalent). The SCI lists all currently identified configuration items (as given in Table 3-1) and specifies their currently applicable date and version.

*Each delivery (or release) of a software baseline must be accompanied by a Software release note, including a list of all CIs included in the release.*

## 8. Tools, techniques and methods for SCM

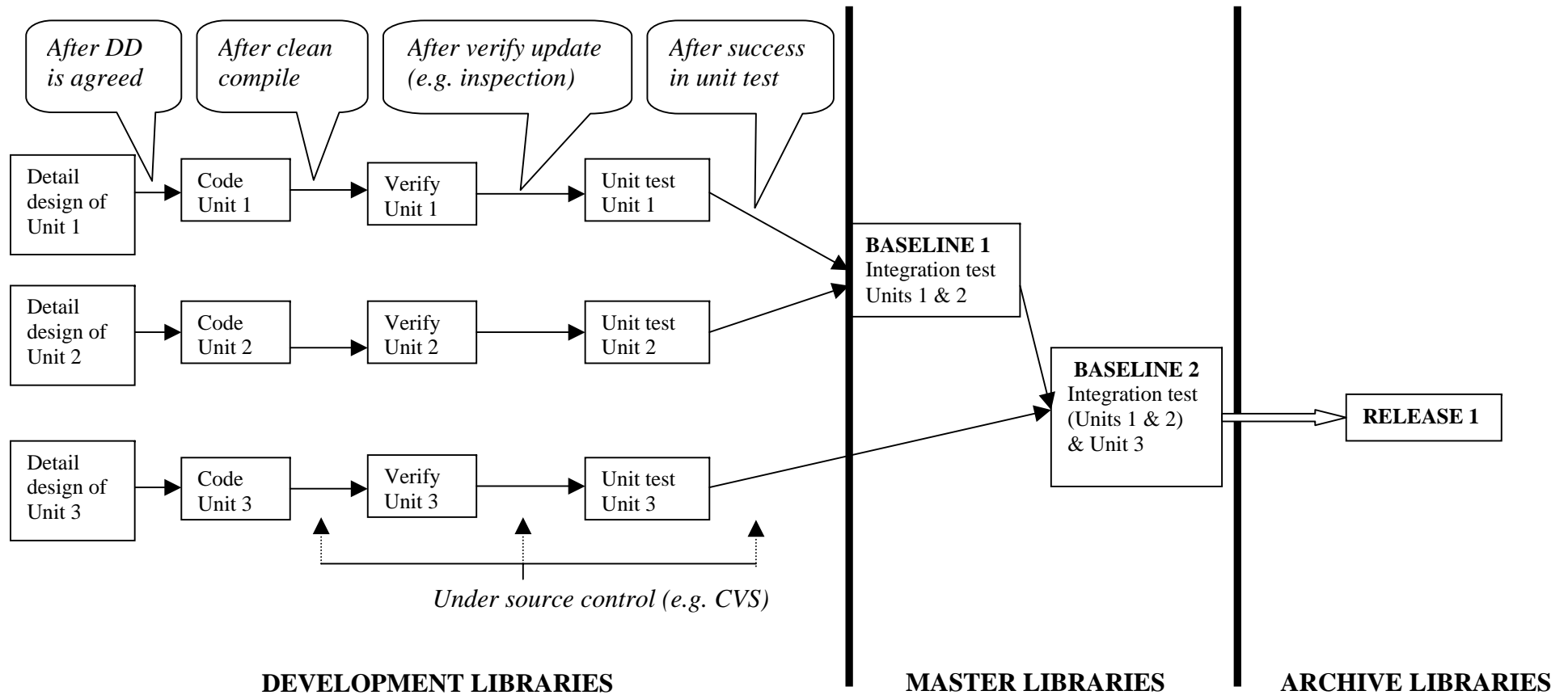
Typically two types of tools are need for software configuration management, a problem reporting system (an example is GNATS) and a version control system (an example is CVS). It may be a necessary to manage documents separately.

## 9. Supplier control

This is the means of applying SCM process requirements to sub-tier suppliers.

## 10. Records collection and retention

In general, SCM records include such items as CI lists, baseline or software library records, change history reports, archive records, and release records ...



**Figure 4-1: Illustration of baselines and releases**

Note: Libraries above are not all, necessarily, in physically different areas.