

CONFIGURATION MANAGEMENT

Adapted from notes from the NCC Training Software Engineering Training Library

Contents

1	Introduction	1
1.1	A Definition of Configuration Management	1
2	Software Configuration Development Stages	1
2.1	Requirements Specification	1
2.2	Functional Design	1
2.3	Detailed Design and Implementation	2
2.4	Acceptance	2
2.5	Maintenance	3
3	Configuration Control	3
3.1	Identification Control	3
3.2	Project Library	4
3.3	Change Control	6
3.4	Defect Report Control	8
4	Change Control	9
4.1	What is “change”?	9
4.2	Stages	9
4.3	Procedures	11
4.4	Control After Full Program Testing	12
4.5	Support Software	12
5	Status Accounting	12
6	Configuration	13
7	Management of Recycling	13
7.1	Requirements	13
7.2	Design	14
7.3	Module Design, Code and Test	14
8	Overall Effects of Change at each Life Cycle Stage	14
8.1	Requirements Specification	14
8.2	Design	15
8.3	Implementation	15
8.4	Service	16

1 Introduction

During the course of design work involving the production of hardware or software, there is a need to control changes so as to ensure that the design does not deviate from the requirement definition (design specification) and to ensure that, where more than one person is involved, changes effected by one person do not adversely affect the others.

All the Design Management activities aimed at controlling design change from the earliest conception are referred to as CONFIGURATION MANAGEMENT activities.

1.1 A Definition of Configuration Management

The totality of all those management activities by means of which it is ensured that:

All the “parts” or elements of a design, when connected or related to form the whole, will meet the requirement definition.

The parts and the whole of the end product of the design activity are adequately documented and authorised.

2 Software Configuration Development Stages

2.1 Requirements Specification

The first milestone in the software development life cycle is reached with the issuing of the software requirements specification. This is the first document to be placed under configuration control and is the stage of the life cycle where software configuration management begins. As soon as possible thereafter a test specification should be produced and controlled.

2.2 Functional Design

During this stage the software requirements are mapped into software terms, and the software structure is formulated as identifiable modules.

This stage must commence, therefore, with the establishment of the key to all subsequent control activities - a component referencing and identification method which ensures uniqueness of identifiers within the system.

The documentation standard to be applied must be specified in a documentation manual. A documentation items list, stating what information will be provided for each component, at each stage of design, must be issued.

At the end of this phase a high-level functional design specification document is produced and placed under configuration control (Milestone 2).

The documentation items list can now be turned into an index by entering the reference

number and location of the documents as they are written. Thus we have a good idea at an early stage in the software life cycle of how much documentation is to be provided, a check list to ensure that nothing that we wish to describe is left out, and a dynamic documentation build state.

2.3 Detailed Design and Implementation

At each lower level of design expansion the same documentation, planning and monitoring activity should be carried out.

A hierarchical design will be reflected in a hierarchical documentation system. Design will be linked to documentation by a common, hierarchical, component referencing system.

2.3.1 Module Design, Code and Test

Module designs and test specifications form the first stage of the lowest level of component design and documentation. The working code and the test results are the final stage. The lowest level module documentation should be complete at the end of this stage.

When modules are working to the satisfaction of the responsible programmer they should be registered with the project library. A list of modules and version numbers together with the documentation index forms a complete, dynamic software build stage, helping to provide visibility at a stage of development notoriously difficult to assess.

2.3.2 Integration and Commissioning

This is the stage where many bugs are found. Until the end of integration, therefore, very careful change controls procedures should be used so as not to impose too great a burden on the software team.

At the end of this stage it may well be feasible to “bond” the software before moving into formal acceptance testing against the test specification. By this time, program documentation is complete.

2.4 Acceptance

This is undertaken with the customer (usually after some dress rehearsals or with run-up tests) in order to demonstrate to him that the software meets his requirements.

In systems where the software is merely part of a “black box” so far as the customer is concerned, he may wish to “accept” only the complete system and require no separate software validation exercise. In these cases the software will normally be “accepted” by the group responsible for conducting the system test.

Bonded software must be subject to rigorous defect investigation and change control procedures.

2.5 Maintenance

If the software developer is to maintain the software in service, the same defect reporting and change control procedures as for bonded software must be applied. When the maintenance task is assumed by the customer, the configuration control documentation, with particular reference to software queries, defects and changes, should be delivered with the software.

3 Configuration Control

To illustrate the need for the control processes described later, here is a realistic example.

A prototype radar system incorporating software has been developed. Although work has now begun on the production model which incorporates a number of changes, the prototype will continue to be used in trials for some years and the software will need to be developed further and maintained. At the same time the production model software is being produced.

Many modules will be common to both configurations, some will contain minor differences and some will be new.

To ensure that software changes have no adverse effects upon either program suite, it is essential to be able to identify which software versions are for which hardware configuration. It is also essential to apply identification and change controls sufficient that after all the modifications have been carried out it is possible to line together those versions that will meet a particular requirement definition, supported by a complete and accurate set of documentation.

To achieve the necessary degree of control, the following activities should be covered by configuration control:

- Identification Control;

- Project Library;

- Change Control;

- Defect Report Control.

3.1 Identification Control

Identification Control should be responsible for performing the following functions:

- Establishing methods of identification for software and documentation, so that each item of software or documentation within a project can be uniquely identified;

- Controlling the issue of the agreed form of item reference numbers to the programmers;

Ensuring that programmers follow the established methods of identification.

3.1.1 Establishment of Methods of Identification

The method of identification to be used on a project should be determined when either:

The design requirement specification has been written and approved;

A software project programming team has been set up.

When the method of identification has been determined, the following activities should be carried out:

Agree the form of identification to be used on each unit of software and documentation;

Inform those interested parties concerned with the software of the agreed form of identification and invite any comments that they may have;

Correlate any comments received and modify, where necessary, the form of identification;

Convey the finalised method of identification to all members of the Project programming team, the Project Library, Change Control, and any other interested parties.

3.1.2 Control of Item Reference Numbers

Having established the method of identification to be used on a project, it should be the responsibility of Identification Control to keep a master list of item reference numbers in use by the programming team. This will avoid the possibility of any duplication of numbers taking place and ensure that uniformity of reference is maintained throughout the project's life.

3.2 Project Library

The Project Library should be responsible for the custody of the project software and documentation through the design, coding, testing and approval stages to customer acceptance and completion of the project. It should also be responsible for the custody of other items directly concerned with the project.

The project library's functions should be:

To ensure the identity of the software and documentation items is in accordance with the agreed standard for identification;

To ensure the items are cross-referenced in accordance with the agreed standard for

identification;

Registration and bonding of software and documentation and to ensure that all software and documentation is registered before bonding and that bonding continues through all stages of development until handover to the user;

To store under the correct environmental and security conditions all media, ie magnetic tapes, discs, paper tape etc, directly concerned with the project;

To store the correct bonded issues of support software and documentation required for the production and testing of project software;

To produce and store master and working copies of ~ny source or binary software brought under configuration control and regenerate working copies as required;

To file all documentation related to the project software including source listings, specifications, flow diagrams etc and to produce working copies of that documentation;

To file all documentation related to tests or trials including test plans, test reports, defect reports etc;

To store any test data required for tests or trials;

To ensure that all software items to be used for a test of software are registered issues and are true copies of the masters held in the Project Library;

To produce and maintain a master list of all software and issues of documentation;

To control the distribution of software and documentation to project personnel;

To give information notices to interested parties regarding any items of change;

To keep copies of all standards and procedures that have been, or are being used during a project's life;

To keep and maintain all Quality Assurance records of the project supporting documents.

3.2.1 Establishment of the Project Library

The project Library should be established and the procedures for its operation defined before the project commences. This will enable the software team to be aware of:

When to transfer software and documentation to the library;

What records the Project Library will keep and how they are maintained;

How to make changes to registered and bonded software and documentation.

Transfer should take place when:

The programmer is satisfied that the software meets the specification and is ready for formal test or the next level of integration;

The identity of the software items is in accordance with the agreed standard for identification.

3.2.2 Registration

Software and associated documentation initially transferred to and held by the project library should be termed registered.

The procedures for registration should be laid down prior to the start of coding.

Software and documentation should be bonded when the stage in development has been reached where the Project Management are confident that the software and documentation will require only a minimum number of changes during further development.

A definition by the Project Management should be made prior to the start of coding as to when bonding will be effected and what formal change procedures will be adopted after bonding.

3.2.4 Storage

Magnetic media should be stored in files which are situated in a "clean room" environment. By definition, the storage area should be dust free and environmental conditions shall exist in the temperature range of 60 - 80 degrees F and relative humidity range of 40 % - 60 %.

At least one copy of all the Project Software and documentation should, where possible, be filed in suitable storage environments in an entirely separate building to that which houses the main Project Library.

Adequate archiving procedures should be adopted to avoid considerable loss of work effort in the event of magnetic peripheral failure.

3.3 Change Control

3.3.1 Software

Modification of registered or bonded software falls into two distinct categories:

Software that is modified as a direct result of test or trials defects;

Software that is modified following changes of the system, ie improvements not

attributed to test or trials defects.

It is, the purpose of Change Control to control and monitor any software modifications that are made.

Change Control should:

Check the level of authority for the change;

Assess and check all the possible effects of the modification on the software concerned, the software directly or indirectly associated with the modified software and the documentation associated with the modified software

Ensure that software to be changed is correctly identified;

Ensure that the change is unambiguously specified together with the reasons for the change;

Ensure that all other software which is affected by the change is listed and cross-referenced to other changes or document changes;

Ensure that parameters affected by the change are identified, eg memory, run time;

Ensure that the required retesting is specified and is adequate to prove the modification;

Ensure that any associated software which has been affected by the modification is adequately retested.

3.3.2 Documentation

Modification of registered or bonded documentation falls into the same categories as those for software modifications. Like software, Change Control should:

Assess and check possible effects of the modifications on the documentation concerned and any other documentation that is associated with the modified documentation;

Ensure that the documentation to be changed is correctly identified;

Ensure that the change is unambiguously specified together with the reasons for the change and if there is an associated software change, adequate cross-referencing is given;

Ensure that all other documentation affected by the change is listed and cross-referenced to other changes or software;

Ensure that the required change has been carried out correctly, as specified.

3.3.3 Establishment of Change Control Procedures

Change control procedures should be established the commencement of a project to enable the software team to be fully aware of:

Which items of software and documentation will be subject to change control procedures;

Which stages of software development informal and formal change procedures will apply;

The procedure(s) to be used for requesting a change;

The change request forms that are to be used for requesting changes;

The approval bodies that will approve or reject changes;

The procedure to be used for implementing an approved change.

3.3.4 When Change Controls should be Implemented

An informal method of change control should be implemented as soon as the project starts, and should be the responsibility of each programmer in the software team.

When each module of software is registered each change thereafter should be recorded after application.

When each module of software is bonded, each change must be formally requested using suitable change request forms and approved by members of the project team set up as the approval authority when the change procedures are established. Each change request must be cleared by Quality Assurance.

3.4 Defect Report Control

The purpose of Defect Report Control is to ensure that any defects or queries about either bonded software or bonded documentation during design and development are recognised and progressed to an appropriate solution in an efficient manner.

It should also provide the means by which the user can feed back information about apparent defects to those responsible for the design and development of the software and documentation.

The functions of Defect Report Control are:

To ensure that all software (including documentation) queries or defect reports are recorded and distributed to those who may be affected by them;

To ensure that remedial actions are implemented;

To progress software defect reports or queries to a satisfactory conclusion To prepare reports listing all outstanding defects and queries for Design Reviews and progress meetings when necessary.

3.4.1 Establishment and Implementation of the Defect Reporting Procedure

The defect reporting procedure for project software and documentation should be established at the commencement of the project and be implemented at the stage in development when the formal change procedures are implemented.

A defect reporting procedure should, however, be implemented at the commencement of coding project software for the purpose of reporting defects in the compiler and utility software.

4 Change Control

4.1 What is “change”?

Hardware changes or modifications are accomplished by a physical alteration of the unit concerned. If it were the only example of its type the “unit-before-change” will have disappeared to be replaced by the “unit-after-change”. Should subsequent events show the change to have been undesirable, rework is inevitable.

Given sensible archiving and library procedures, software changes do not result in the loss of the original unit. Software changes are not really changes to software. They are changes in function brought about by the creation of new versions of modules and linking those modules to form a function ally different piece of software that may take its place on the shelf alongside the “old” software.

The user needs to know the functional differences between the versions.

The software team (and especially those maintaining the software) need to know how and why the changes were introduced.

The software manager needs to know that his team's efforts are not being used on changes or enhancements that he may consider undesirable or of low priority.

A change control procedure must be established to satisfy all these requirements.

4.2 Stages

Most software changes occur early in program development, during module testing and early

in integration.

It is desirable to interfere with production as little as possible at this stage.

Two levels of control are therefore exercised - an “undemanding” programmer-oriented system during early module life and a strict change proposal and approval system when the software has been given “frozen status”.

4.2.1 Level 1 - Registered Software

When a module (a module is the smallest identified unit, say 100 lines of source code) is working to the satisfaction of the programmer it is REGISTERED, with its documentation, in the Project Library.

This software supplied will become the masters from which all future official copies will be generated.

A module control form, test record and amendment record are created and the build-state updated.

Once a module is registered authorisation, at the appropriate level, is required before any changes can be implemented.

Changes to the module or its documentation found necessary during subsequent development and integration are implemented by the software team. Details of the change and why it was made, together with the old and new version numbers of that module are entered on the module amendment record. Date of retest is entered on the test record.

The new version and amended documentation is registered with the library.

4.2.2 Level 2 - Bonded Software

This level of control should be instituted when project management are confident that the software and documentation will require only a minimum number of changes during further development.

It should precede the formal testing and acceptance activity. (For formal test results to be meaningful, all the software used during the test, as well as the software under test, should be subject to configuration control procedures).

Changes to bonded software must be the subject of a formal change, request, approval, implementation procedure followed where applicable by re-issue of programs and documentation.

All queries relating to defects in bonded software or documentation should be raised on a form designed for that purpose.

After hand-over to a customer who plans to assume fully responsibility for all future queries, changes or system enhancements involving software maintenance, it is envisaged that the configuration control records would be delivered with the software. (This is particularly important in the case of software queries still uncleared.)

4.3 Procedures

4.3.1 Software Query Procedures

Queries may arise from a number of sources external to the software team. All software queries should be addressed to a designated Defect Report Control Authority.

The query will be checked by defect report control to ensure that it has been correctly completed, but in particular to see if it is already under investigation or if a solution has already been found (records are essential).

When accepted, the query should be acknowledged and submitted to the relevant team for investigation.

The team may not be able to reproduce the fault or may need more evidence, but dependent upon the result of the investigation the query will be either rejected or accepted and a solution proposed.

Defect Report Control will monitor the progress of the query, inform the originator of its rejection or the proposed solution and record the satisfactory implementation of the solution.

4.3.2 Change Request Procedure

Requests for software changes may come from software query investigations or from the software team itself.

Change requests must be raised for changes to “bonded” software or its supporting documentation.

Effects upon store space and run time must be specified as well as the resources necessary to implement the change, test it and document it.

The proposed change must be examined for possible unwanted effects so far as other users are concerned and whether the change will have effects upon software, documentation or even hardware.

It should be noted that in some cases changes that will render a particular version of a program unusable to some recipients may still be implemented for special reasons. (eg a flying program may have to be modified when it is run on a ground rig). In such cases, the configuration control problems involved (ie identification, storage and documentation) should be identified in advance and procedures designed to cope with them.

All change requests should be registered with a central change request authority responsible for circulating the request to interested parties for comment, informing users of the impending change and progressing the change to a satisfactory conclusion.

4.4 Control After Full Program Testing

After integration and full program testing has been completed satisfactorily, before either off-site testing, acceptance or trials are carried out, the Project Library will ensure that it has in its custody:

Master and copies of relevant module sources and documentation referenced by their associated control forms;

The master, sub-master and two working copies of the object program;

All copies of cleared, rejected and out-standing Software Change Requests (SCR), Support Software Change Requests (SSCR) and any other software-related change request;

All copies of cleared, rejected and outstanding Software Query (SQ);

All copies of software change notification forms and software information notices;

The program's test record, specification and report;

Complete and approved documentation at its latest issues.

4.5 Support Software

Problems can occur when the support software configuration changes during the project's software development phase.

For example, a change of operating system software may render binary files created under an earlier version of the system incompatible with those created later.

It is important therefore to be able to relate particular project software items with the support software configuration under which they were created.

Each series of support software will be allocated a "compile number" and the Project Librarian will keep a record of the issue/version of each item of support software, the respective series to which the item belongs and the series compile number.

Whenever an item within a series is updated and is either reissued as a new issue or a new version then the series compile number will be incremented by one.

This compile number should be included in a binary file label.

5 Status Accounting

Status accounting is the recording (and reporting on) status and change with the passage of time.

The content of the first approved issue of software, consisting of a list of the exact versions of all the component parts, is recorded.

Thereafter, all change proposals and their rejection or implementation are logged, together with any new module version numbers. These form the link with the next approved issue of software at which stage a new list of the versions of all the component parts is created. Information on documentation can also be recorded which will link special document changes (user and operator instructions, for example) to particular software configurations.

6 Configuration

Configuration audits have two aims. To ensure that the deliverable configuration meets the functional requirements and to ensure that the deliverable documentation accurately describes the deliverable configuration.

This is achieved by ensuring that changes to the requirement are progressed to a satisfactory conclusion, that changes to software do not result in departures from the requirement, and that all changes are satisfactorily described and reflected in the technical documentation.

Procedures that ensure regular progress reports on changes in the pipeline and their implementation should be specified. Before formal tests and certainly before any formal approval of deliverable software is given, all relevant changes must be seen to have been satisfactorily cleared.

7 Management of Recycling

The loops in development may be executed many times before system delivery takes place.

The danger of unmonitored recycling lies in “invisible” work, ie work involving changes to areas already recorded as “complete”. This can result in extended timescales, over use of resources, and failure to produce a program that meets the requirement specification on the planned date.

Care must be taken, therefore, to formalise the recycling procedure sufficiently that proposed changes are examined not only for their effect on the system's function but also for their effect on the overall plan.

It may well be preferable to start trials on time with a program containing known defects than to wait indefinitely for “perfect” software.

7.1 Requirements

Review of the requirement specification and subsequent design, implementation and testing will involve rework of the requirement.

All changes to the requirement must be subject to formal control procedures.

Requirement should be frozen as soon as possible in the life of the project. The main aims are to ensure that proposed changes are essential (where they can be left to be incorporated in subsequent issues of software, this should be done); that the consequences of the change are rigorously examined; that the cost of the change is known; that the change is incorporated at the right time with sufficient information to those concerned to avoid people working from different specifications.

Adequate issue identification and change procedures are essential.

7.2 Design

The high level software design will be subject to change arising from both requirement specification changes and during lower level design, coding, testing, and system integration.

The stage at which formal change controls will be applied must be specified.

A high-level design document should be registered early in the software life-cycle. Subsequent lower level design documentation should also be registered as early as possible. Before the introduction of formal change request procedures, the modification of such documentation is the only record of design change. Identification methods that clearly identify and cross-relate various software versions and documentation issues are essential.

7.3 Module Design, Code and Test

Modules will be subject to alteration due to change in the high level design, testing and integration phases.

8 Overall Effects of Change at each Life Cycle Stage

The effects of change at each life-cycle stage can be represented by the curve of the costs of errors at each live cycle stage. This is self-evident because the costs of errors are the costs of changing the software. The costs remain the same whatever the cause of the change. The later in the software life cycle that change takes place, the wider the effects of that change and the more expensive (in time to examine and approve, time to implement and time to verify) it is likely to be.

8.1 Requirements Specification

It is not sensible to describe Requirements Specification as a “stage” with respect to software changes. Before design work has commenced, changes to the requirement merely have the effect of delaying the issue of the specification. After the document has been issued, changes to the requirement will effect changes to the software dependent upon the stage the software

has reached.

Ideally, the requirement specification should be “frozen” at a predefined stage. In practice, however, the need to change the requirement is weighed against the cost of implementing the change. (Here, extended delivery dates may well be more important than financial implications). If the change is important enough the “frozen” specification is often thawed. Changes to specifications must always be formally controlled and all users of the specification must be notified at the same time to avoid the possibility of different requirements being in force simultaneously. Informal change agreement between software team members and specifying personnel can lead to extended time scales and increased costs “invisible” to management. The following sections discuss the effects of change without distinction between change caused by amendment of the requirement or change necessary to bring the software up to the requirement.

8.2 Design

Change during the design stage is easier to accommodate than at any other time. If implementation has begun on some well-defined areas while design continues on other areas there is always the risk that change will cause some rework. If implementation does not start until design is complete all that is affected is paper and time. The more detailed the design, the more time must be spent examining the system for possible side effects.

8.3 Implementation

8.3.1 Module Code and Test

If the change is as a result of higher level specification or design change module communications and data areas must be examined to ascertain what modules and data areas are affected. Any change at this stage may involve amending design documentation, source code, and perhaps the test program. Even if the test program is not affected, the modules should be re-tested after modification.

Records should be kept of all amendments performed including details of the change and why it was necessary.

8.3.2 Integration

Most changes at this stage will be to correct errors not discovered during module testing (in which case the module's test program and test data should be critically examined, even rewritten, and the module requested) and to cater for module to module and hardware interface problems. Care must again be taken to identify all data and all modules affected.

Documentation must be updated where necessary. It may well be necessary in the case of major changes to start the integration exercise again. Management must ensure changes to requirements are regulated so as not to stretch integration beyond the planned duration.

8.3.3 Commissioning and Acceptance

During commissioning, the software will be exercised as part of a system introducing operating conditions that would have been difficult, if not impossible, to simulate effectively in the earlier stages. It is inevitable therefore, that more errors will be discovered at this stage.

It is legitimate to use “patches” both as a means of checking that a proposed correction will not have the adverse effects elsewhere in the system and to maximise the use of expensive proving rigs which will probably have to be booked in advance. Control of patches is of paramount importance. The identification system should consist of the name and issue/version number of the software for which it was created and a sequence number. Information recorded for each patch should include the purpose of the patch, the old contents of the address(es) changed and the new contents.

The proposed change (with details of the tested patch) should be incorporated into the source as soon as possible and re-compiled, loaded and tested. Whether patches are used or not it is important that any changes effected to the software at this stage are carefully analysed for possible repercussions elsewhere in the system. To this end communication matrices and module/data area access charts are a help. A list of modules activated by various test schedules should be created at the time the test schedules are written. This gives a good feel for the amount of retesting necessary. Automation of such analysis is becoming available and should prove a valuable tool in this area.

Documentation must be updated to reflect any changes.

8.4 Service

In service the whole of the foregoing cycle (specification, design, implementation) may be repeated as system enhancements or change of operating conditions are catered for. The same controls are called for - analysis of the proposed change with particular regard to its effects on the rest of the system, approval of the proposed change, implementation and careful documentation of the change.

A formal reporting system needs to be established by the time the software has reached the commissioning/acceptance stage. The system should include procedures for acknowledging the defect report, passing it to a software team for investigation, and progressing it either to rejection or acceptance followed by implementation, retesting and re-issue.

The process becomes more expensive when personnel other than those who developed the system are required to maintain it. They are less familiar with the design and thus need more time to assess proposed changes, and they are also more prone to introduce unwanted side-effects with consequent work.

The importance of meticulous examination of proposed changes cannot be overemphasised. The later in development that change is called for, the more pressure is usually applied to hurry the change through. Yet the later in development, the greater the possibility of subtle impact else where in the system. Hurriedly implemented changes can result in a worse program than the original, resulting in recycling and wasted resources.