Software Engineering Group Projects – Operating Procedures and Configuration Management Standards

Author: C. J. Price
Config Ref: SE.QA.08
Date: 7 February 2017
Version: 2.2
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1 INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to specify procedures enabling all items produced by a group project to be properly controlled; and to provide a mechanism by which problems and corresponding changes to project items can be recorded.

1.2 Scope

This document describes the procedures, tools and techniques to be used for configuration management, and the procedures to be used for problem reporting and corrective action. It should be read by all members of the project group.

This document assumes the reader is already familiar with the QA Plan [1]. The Project Management Standards [2] should be read in conjunction with this document.

1.3 Objectives

- To describe a practical means of effectively managing all documents, diagrams, source code modules, executable systems and any other significant items that are produced by the project group and which are stored on computer.
- To describe procedures for notifying problems with any of these items,
- To specify the procedures which must be followed when a problem has been discovered: these will cover analyzing the problem; identifying the solution, and causing corrective action to be taken.

2 SOFTWARE CONFIGURATION MANAGEMENT

2.1 Introduction

Configuration management is the management and control of all kinds of changes made to a system so that the state of each component is always known. Ideally, configuration management begins at the start of the project, and continues throughout development and on into the maintenance phase.

It is not concerned just with source code, since all approved documents, such as the Test Specification and the User Interface document, must be kept under control of the configuration management system. The use of Subversion is mandated. Each student has been provided with an individual Subversion account, and there is also a group account that is where your project will store its shared documents and code. The individual account was intended for exploring use of Subversion - you should use the group account for your group repository.

The configuration management system will allow a user to:

1. retrieve copies of any version of a file, enabling recovery of previous or ‘old’ versions;
2. retrieve copies of any version of a directory structure, with its files, enabling recovery of previous version;
3. check in changes to the file, causing the changes to be recorded and the version number to be incremented;
4. inquire about differences between versions, obtain a log summarising the changes checked in for a particular version and produce a history of the file showing all changes and the users responsible.

Where the following items are produced on a project, they *must* be kept under the control of the configuration management system:

1. List of project deliverables;
2. Requirements Specification;
3. Design Specification, including any accompanying diagrams;
4. Test Specification;

5. User Interface document, including any on-screen presentations
6. Maintenance Manual;
7. End-of-Project Report;
8. Source code and tools (e.g., build files);

The QA Manager will be responsible for adherence to the configuration management procedures, and will directly control access to items in the configuration management system.

2.2 Configuration Items, References and Status

Configuration items are project items which are controlled by the configuration management system, and thus include the items listed in section 2.1 above.

The Quality Manager should compile a list of project deliverables that will be kept under configuration management, along with a description and location of the item. It will be in a file called `config_refs` in the configuration directory (see section 2.3).

e.g.

Item	Name	Location
Test Specification	TestSpecGroup8.doc	Folder docs/testspec

The QA Manager is responsible for the allocation of configuration references and for the maintenance of a file called `config_refs` in the configuration directory (see section 2.3).

Any document must have one of the following statuses:
1. Draft - the document is currently under development;
2. For review - the document is ready for formal review;
3. Release - the document has successfully passed its review and is thus complete and correct.

Each item will be under the version control system. An item will progress from Draft, through For review to Release but may well return to an earlier status. It may fail its review or a released version may need to be corrected and reviewed again.

2.3 Directory Structures

One aspect of configuration management is the standardisation of the directory structure which holds the project data. This structure is managed by the version control system. Both files and the directory structure may change (typically grow) and this must be managed.

The *QA manager* will ensure that an initial structure is created. All members will check out a working copy. They will work in this structure and commit changes to the repository as appropriate. Working copies will have any generated items (processed documents, compiled code etc.) locally generated. Working files and directories which are not part of the final product (e.g. temporary output files) may also be present.

The directories described below should be present and additional directories may be created as required.

- **docs**. This contains submitted documents produced by the group members. Each document should be in its own directory (see Section 2.2, “Configuration Items, References and Status ” above). They should be submitted once they are ready to be reviewed.
- **man**. management documents are stored in this directory. Specifically, there should be a minutes directory (see Section 2.4.3, “Managing Minutes of Meetings ” below)
- **src**. This contains the source code for the project, including any module tests, arranged into the relevant directories corresponding to their position in the package hierarchy or in other language defined arrangements. If a project requires more than one program in the system, one directory,

appropriately named, may be required for each program. Source code should be under configuration control once it is part of the designed system. Before that, it should be put in **dev**.

- **dev**. This directory should contain a folder for the date of each tutorial, if there is a tutorial on 14th Feb, a folder named 20170214 should be created. Where a student creates a draft document or some code as part of their duties that week, it should be submitted to the folder for the next tutorial, so that all group members can see it.

IDEs will typically manage file directories at an appropriate level for including your emerging product in **src**. For example, NetBeans will use a directory called nbproject; Eclipse will have a number of "dot" files. These must be under version control. Unit test code will also be in an appropriate directory under **src**. The IDE will normally manage this. Directories generated by the IDE from source must not be under version control. For example, `build` or `dist` directories.

2.4 Managing Documents

2.4.1. Managing Documents/Code During Their Initial Development

When documents are initially created, they should be developed in the appropriate place in the author's working copy directory. The author will repeatedly edit the file, and check it into the repository as convenient and whenever sharing with other developers is necessary.

When the document is ready for review or testing, its status is updated to *For review* and it is checked into the repository. Review or testing is the carried out on an identified revision number to ensure that all work is carried out on the same version. When it has been approved, the status is updated to Release. If it is not approved, its status may be reduced to Draft while changes are made, or a new revision may be created if minor changes require only one edit.

2.4.2. Updating Documents

Team members may all have working copies of any or all of the repository. They are responsible for ensuring that they have the correct version and do not over-write changes that took place after they took a copy of the repository. Major changes may take place through version control branch. This will often involve informing other group members that they are working on a specific document.

2.4.3. Managing Minutes of Meetings

Minutes of meetings will be stored in the man/minutes directory. There must be one file per meeting, and the file name must have the following form:

y*yyy-mm-dd_minutes*

(plus any file extension) where:

- *yyyy* is the year;
- *mm* is a two digit number representing the month, using leading zeros where required (January = 01);
- dd is a two digit number representing the day of the month using leading zeros where required.

e.g., the minutes of the regular weekly meeting of the project group held on 6th January 2017 might be saved in a file called 2016-01-07_minutes.txt. The advantage of using this naming convention is that when the file names are listed, they can be made to appear in order of the date.

See [3] for a description of the format of minutes.

3 PROBLEM REPORTING AND CORRECTIVE ACTION

Problems relating to items in the **docs** directory must go through a formal problem reporting and action process. If we had a stable software product during the project, this would also be true of code, but typically, group projects do not have stable code until the end.

A form is necessary to record the process of changing released documents: the Change Control Form (CCF). An example blank CCF is available in GPDocs. Completed forms are to be kept secured in a ring binder or similar, known as the *Change File* (although see below for on-line option). The binder must contain dividers with the following headings: *Outstanding CCFs*, and *CCFs Dealt With*. The Change File should be maintained by the Project Leader. The QA Manager must check that CCFs are being completed and used correctly. The use of these forms and the binder are described in the following problem reporting and corrective action procedure:

- On discovery of a problem in a configuration item that has been allocated a version number and has a status of *Draft*, or *Release*, the details of the problem must be noted on a Change Control Form (CCF).
- The CCF must then be given to the QA Manager, who will assign a unique CCF number to the form and who will then investigate the problem with the aid of other team members as appropriate and identify what configuration items need to be changed, if any. A copy of the CCF (online copy allowed) must be given to the person authorised to make the change, who will then carry out the change.
- When the change has been completed, the person who made the change will report to the QA manager who will check the amended configuration item. The QA manager may initiate formal testing to confirm the change fixed the problem.
- If the change is accepted by the QA manager, the configuration item must have its version number incremented using the procedures specified in this document. The CCF corresponding to the reported problem will be then be moved from the *Outstanding CCFs* section of the Change File to the *CCFs Dealt With* section by the Project Leader.
- If the change is not accepted, then the QA manager will advise on further action and if necessary complete a new CCF. New forms will only be completed if a new problem is identified, or if the original CCF indicated a change which failed to fix the problem.

3.1 (Optional) Online Reporting and Corrective Action

Project teams can choose to run all reporting and corrective action online. In order to do so, they will have to make the following changes to the procedure above:

- They must create a directory `CCFS` at the top level. A copy of each raised CCF (using the provided Word template for CCFs) should be kept in the `CCFS` directory. The first should be named `CCF001.doc`, etc.
- Two files must be maintained by the Project Leader in the CONFIG directory, called `Outstanding_CCFs` and `CCFs_Dealt_With`. Each file must contain a list of all CCFs in that state. For each CCF in the list, there should be a one-line summary of what it is about.

REFERENCES

[1] QA Document SE.QA.01 - Quality Assurance Plan.

[2] QA Document SE.QA.03 - Project Management Standards.

[3] QA Document SE.QA.02 - General Documentation Standards.

[4] QA Document SE.QA.06 - Test Procedure Standards.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	09/10/01	Document given complete overhaul in Word	CJP
1.5	N/A	12/09/08	Changed document template to be Aber Uni	CJP
2.0	N/A	12/08/12	Simplified process	CJP
2.1	N/A	22/12/16	Updated with redone documentation	CJP
2.2	N/A	07/02/17	Changed it to take account of use of Subversion	CJP