Software Engineering Group Projects – Operating Procedures and Configuration Management Standards

Author:	C. J. Price
Config Ref:	SE.QA.08
Date:	23 January 2018
Version:	2.3
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 GitLab on the Department of Computer Science's GitLab server at gitlab.dcs.aber.ac.uk is mandated.

GitLab 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 versions of software, or snapshots of documentation;
3. check in changes to the file, causing the changes to be recorded;
4. inquire about differences between versions, obtain a log summarising the changes checked in for a particular version and produce a history of a 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. Design Specification, including any accompanying diagrams;
3. Test Specification;
4. User Interface document, including any on-screen presentations
5. Maintenance Manual;
6. End-of-Project Report;
7. Source code and tools (e.g., build files);

The QA Manager will be responsible for adherence to the configuration management procedures, and will manage 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 config 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.
- **config**. management documents are stored in this directory. Specifically, there should be a minutes directory (see Section 2.4.3, "Managing Minutes of Meetings " below), and a configuration file.
- **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 sub-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 20180214 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.

A basic structure for the group project will be provided along with the instructions for using GitLab. In order to be able to set it up for the group, all group members need to have logged into GitLab at least once, so that they appear in the list of GitLab users. One member of the group should then set up a Project, and include all of the other group members as *developers*, and the project manager and the client as *reporters*. When setting up the project, the provided project template should be used as a framework for the project.

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 will all have working copies 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. This will often involve informing other group members that they are working on a specific document or piece of code. Major changes may take place through version control branch.

2.4.3. Managing Minutes of Meetings

Minutes of meetings will be stored in the config/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 March 2018 might be saved in a file called 2018-03-06_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

3.1 Issue tracking

The project will use the issue tracking system provided with GitLab for assigning weekly tasks to group members. When the group member commits files related to the issue, they should refer to the issue number in their commit message, so that the group can see what has been done in response to the issue.

3.2 Changes to released documents

Problems relating to *Released* items in the **docs** directory should go through a slightly more formal problem reporting and action process. When an issue is raised with such a document, then the person assigned to make the change to the document should be recorded and the issue moved to the *Doing* board. When they have made the change to the document, they should change its status to *For Review*, and move it to the *Reviewing* board. The QA manager should check that the changes are appropriate (organising a formal review if they judge it necessary), and check out the document and change its status to Release, and mark the issue as Done.

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
2.3	N/A	23/01/18	Changed repository structure, and using GitLab	CJP