

Software Configuration Management Plan (SCMP)

Intelligent Lifestyle

Team Daedelus (`s440gf`)

Ho-Jung Simon Youn - (`hjsy`)

Jian Alan Huang - (`jhua`)

Nathaporn Eiamvittayakorn - (`neiam`)

Saw Ai Soon - (`sasoon`)

Wendy Wai-Tak Tsang - (`wwttsang`)

Revision: 2.0.0
29 October 2004

Maintained by: Saw Ai Soon - (`sasoon`)

Abstract

This document details the activities and procedures for the Software Configuration Management throughout the development of the Intelligent Lifestyle, with the aim of ensuring simple maintainability of the entire software product.

Contents

- 1 Introduction 6**
 - 1.1 Purpose 6
 - 1.2 Scope 6
 - 1.3 Intended Audience 6
 - 1.4 Project Overview 6
 - 1.5 Personnel 7
 - 1.5.1 Development Team 7
 - 1.5.2 Supervisor 7
 - 1.5.3 Clients 7
 - 1.6 Definitions and Acronyms 7
 - 1.6.1 Definitions 7
 - 1.6.2 Acronyms 9
 - 1.7 Reference Documents 9
 - 1.8 References 10

- 2 Software Configuration Organisation 11**
 - 2.1 Organisation 11
 - 2.1.1 Configuration Control Board 11
 - 2.2 Responsibilities 11
 - 2.2.1 CCB Member 11
 - 2.2.2 Configuration Manager 11
 - 2.2.3 Project Manager 12
 - 2.2.4 Risk Manager 12
 - 2.2.5 QA Sub-team Leader 12
 - 2.2.6 Configuration Item Manager 13

- 3 Software Configuration Management Activities 14**
 - 3.1 Identifying Configuration Items 14
 - 3.1.1 Document Configuration Items 14
 - 3.1.2 Code Configuration Items 15
 - 3.1.3 Other Configuration Items 15
 - 3.2 Naming Configuration Items 15
 - 3.2.1 Document Naming, Revisioning and Tagging 16
 - 3.2.1.1 Naming Document 16
 - 3.2.1.2 Revisioning Document 16
 - 3.2.1.3 Tagging Document 16
 - 3.2.2 Code Naming, Revisioning and Tagging 17
 - 3.2.2.1 Naming Code 17
 - 3.2.2.2 Revisioning Code 17
 - 3.2.2.3 Tagging Code 17
 - 3.3 Acquiring Configuration Items 17
 - 3.3.1 Storing Configuration Items 17
 - 3.3.1.1 Document CIs 17
 - 3.3.1.2 Code CIs 18
 - 3.3.1.3 Other Configuration Items 19
 - 3.4 Baselining Procedure 19

3.4.1	Entry Criteria for Baseline	19
3.4.2	Baselining	20
3.5	Configuration Control	21
3.5.1	Requesting Changes	21
3.5.2	Evaluating Changes	22
3.5.2.1	Evaluation Procedure	23
3.5.3	Approving or Disapproving Changes	23
3.5.4	Implementing Changes	24
4	Software Configuration Resources	25
4.1	Concurrent Version System	25
4.1.1	Setting Up CVSROOT	25
4.1.2	Checking out Files Stored under CVS	25
4.1.3	Adding Files or Directories into CVS	25
4.1.4	Deleting Files from CVS	26
4.1.5	Deleting Directory from CVS	26
4.1.6	Updating Files	26
4.1.7	Committing Files into CVS	27
4.1.8	Symbolic Naming	27
4.1.9	Renaming File in CVS	28
4.1.10	Renaming Directory in CVS	28
4.2	Backup and Recovery Process	28
4.2.1	Backup Process	28
4.2.1.1	Recovery Process	29
5	Appendices	30
5.1	Change Log	30
5.2	Appendix A: Baseline Request Form	30
5.3	Appendix B: Change Request and Evaluation Form	31

List of Figures

List of Tables

1	Development Team	7
2	CCB members and their functional roles	11
3	Other Configuration Items	15
4	Tag labels for different evolutionary stages of documents	16
5	Storage of Document CIs	17
6	Storage of Document CIs	18
7	Storage of Other Configuration CIs	19
8	CIMs that involve in CCB	21
9	Change Log	30

1 Introduction

1.1 Purpose

The purpose of the SCMP is to provide Team Daedalus with a plan for configuration management. The document will set forth the policy is processes and procedures to be followed regarding configuration management.

1.2 Scope

This document is intended for use throughout the development of the Intelligent Home project. This document may be updated at any stage of the software life cycle, to reflect changes in the configuration management of software items used. The document will assist the development of the proposed system by:

1. Identifying the proposed or implemented configuration of the Intelligent Lifestyle project.
2. Systematically recording and tracing changes to all system components throughout the life-cycle.
3. Providing tools and processes for controlling changes.
4. Detailing the procedures to be followed for baselining of code and documents.

1.3 Intended Audience

The intended audiences for this document are:

1. Team Daedalus, as identified in Section [1.5.1](#)
2. Supervisor for Team Daedalus, as identified in Section [1.5.2](#)

1.4 Project Overview

The aim of the Intelligent Lifestyle project is

- **To design and build a system via the ROADMAP methodology comprising of some intelligent agents, for the explicit purpose of providing demonstrations of Intelligent Agents.**

This aim is an attempt to balance the two requirements from the Clients. This is necessary as meeting both fully would be impossible with current time constraints. The individual aims of the Clients are shown below.

1. To provide a demonstration of intelligent agents.
The Clients wish to have something that can demonstrate agents, agent behaviour and intelligence. Scenarios will be used to demonstrate these features.
2. To implement the ROADMAP methodology(section [12.3.2](#)) and create an intelligent agent system.
This Clients wish to test out ROADMAP and produce an example of an intelligent agent system. They believe this methodology will be useful in implementing the project.

1.5 Personel

1.5.1 Development Team

Name	Login	Phone no.
Carol Poon	cyspoon	0401-959-660
Dominic Mendonca	dxm	0411-093-253
Glenn Fry	gmfr	0418-372-176
Simon Youn	hjsy	0403-438-830
Jian Alan Huang	jhua	0402-001-910
Kieran Simpson	kieranjs	0412-821-128
Masyuri Tjhandana	masyurit	0413-150-311
Mei Ling Leong	mlleong	0413-689-314
Nathaporn Eiamvittayakorn	neiam	0407-565-824
Quyen Quach	qlq	0412-122-031
Shirley Soon	sasoon	0407-552-338
Wendy Tsang	wwttsang	0412-049-823
Ivan Wong	ywong	0411-863-261

Table 1: Development Team

Email addresses of team members can be derived from the user's login name by appending "@students.cs.mu.oz.au".

1.5.2 Supervisor

Kendall Lister
Department of Computer Science and Software Engineering
University of Melbourne
krl@cs.mu.oz.au

1.5.3 Clients

Leon Sterling
Department of Computer Science and Software Engineering
University of Melbourne
leon@cs.mu.oz.au

Thomas Juan
Department of Computer Science and Software Engineering
University of Melbourne
tlj@cs.mu.oz.au

1.6 Definitions and Acronyms

1.6.1 Definitions

Baseline

State of a CI after it has undergone validation as outlined in the SVVP and the CCB has decided

to baseline the CI.

Convention

A technique, practice, or procedure that is established by usage and widely accepted.

Guideline

Non-strict processes created for individual benefit and the benefit of the team.

Intelligent Agent

Acts on instruction of the user and can use his knowledge of the user's interest and wishes to do his work.

Procedure

A set of instructions that members must follow to achieve a specific goal contributed towards the final outcome of the project.

Process

A sequence of steps, tasks or activities that converts inputs from suppliers to an output.

`$Group`

`/home/se440/s440gf`

`$GROUPCVS`

`/home/se440/s440gf/Repository`

1.6.2 Acronyms

<i>AREP</i>	Active Risk Elimination Plan
<i>BP</i>	Build Plan
<i>BR</i>	Baseline Request
<i>CCB</i>	Configuration Control Board
<i>CI</i>	Configuration Item
<i>CIM</i>	Configuration Item Manager
<i>CR</i>	Change Request
<i>CRE</i>	Change Request and Evaluation
<i>CVS</i>	Concurrent Versions System
<i>DOS</i>	Disk Operating System
<i>EPS</i>	Encapsulated Postscript
<i>ID</i>	Identification
<i>PC</i>	Personal Computer
<i>PDF</i>	Portable Document Format
<i>PIP</i>	Process Improvement Proposal
<i>QA</i>	Quality Assurance
<i>RAP</i>	Review and Audit Plan
<i>RMP</i>	Risk Management Plan
<i>SADD</i>	Software Architecture Design Document
<i>SCM</i>	Software Configuration Management
<i>SCMP</i>	Software Configuration Management Plan
<i>SDD</i>	Software Design Document
<i>SPMP</i>	Software Process Management Plan
<i>SQAP</i>	Software Quality Assurance Plan
<i>SRS</i>	Software Requirements Specifications
<i>SS</i>	Scenario Specification
<i>SVVP</i>	Software Verification Validation Plan
<i>TM</i>	Tracibility Matrix
<i>TP</i>	Test Plan
<i>TR</i>	Technology Report
<i>V&V</i>	Verification and Validation
<i>UD</i>	User Documentation

1.7 Reference Documents

- BP
- RAP
- RMP
- SCMP
- SPMP
- SQAP
- SVVP

1.8 References

1. Software Engineering: Principles and Practice 2nd Edition, Hans Van Vliet, Wiley and Sons Ltd 2000
2. IEEE Standard 828-1990 Software Configuration Management Plans
3. Software Configuration Management Plan, Team Atomic, 2003

2 Software Configuration Organisation

This section describes the responsibilities of the organisations and individuals within Team Daedalus, as well as the authorities they have to carry out SCM activities.

2.1 Organisation

The organisational units who are responsible to carry out SCM activities are:

1. All members of Team Daedalus; and
2. Configuration Control Board (CCB)

2.1.1 Configuration Control Board

The CCB is an authoritative body that manages changes to project components that have been baselined. The personnel and functional roles of the CCB members are outlined in the below table:

CCB Member	Personnel	Functional Role
CM	sasoon	To oversee team Daedalus's adherence of SCM activities as outlined in the SCMP.
PM	jhua	To schedule and allocate resources to implementation activities as a result of CRs.
RM	mleong	To identify, manage, mitigate, and monitor risks associated with making CRs to CIs.
QA Manager	wwttsang	To assess the impact CRs can have on V&V activities previously performed and to indentify necessary V&V activities to maintain the desired quality already achieved.

Table 2: CCB members and their functional roles

2.2 Responsibilities

2.2.1 CCB Member

The responsibilities of the CCB members are:

1. Evaluating schedule, quality and risk considerations when approving and disapproving BRs and CRs to CIs.
2. Making informed decisions regarding the approval and disapproval of BRs and CRs to CI.
3. Helping the CIM to identify the exact changes to be made to a CI
4. Attending CCB meetings if necessary.

2.2.2 Configuration Manager

The responsibilities of the CM in the CCB are:

1. Managing the baselining activities for document and code CIs.

2. Approving baselining and making CRs to CIs.
3. Tracing the interdependencies of CIs.
4. Evaluating the impact CRs have on interdependent CIs.
5. Evaluating the feasibility of CRs and validating CRs.
6. Ensuring CRs are acted upon and resolved by the CCB.
7. Initiating evaluation of CRs at CCB meetings or via email.
8. Scheduling CCB meetings if necessary.
9. Preparing agendas for CCB meetings.
10. Chairing CCB meetings.

2.2.3 Project Manager

The responsibilities of the PM in the CCB are:

1. Resolving CR deadlocks.
2. Ensuring the implementation of changes can be scheduled into the project plan.
3. Identifying the most suitable resources available for implementing different changes.

2.2.4 Risk Manager

The responsibilities of the RM in the CCB are:

1. Informing the CCB of any possible risks a CR may create.
2. Suggesting to the CCB how risks resulting from CRs can be managed, mitigated, and monitored.

2.2.5 QA Sub-team Leader

The responsibilities of the QA Sub-team Leader in the CCB are:

1. Informing the CCB which V&V activities will be required once a change has been implemented.
2. Informing the CCB if the implementation of a change may introduce regression faults and how such faults can be tested.

2.2.6 Configuration Item Manager

The responsibilities of the CIM in the CCB are:

1. Using the BR form to notify the CCB when a particular CI is ready to be baselined.
2. When requesting a change to a CI, explains the necessity of the change to the CCB during evaluation of change.
3. Explaining the feasibility and the possible impact on interdependent CIs when evaluating the CR at CCB meetings or via email.
4. Identifying tasks to implement changes and ensuring the tasks are carried out.

3 Software Configuration Management Activities

This section identifies all functions and tasks required to manage the configuration of the Intelligent Lifestyle project. Both technical and managerial activities must be identified. These activities are Identifying Configuration Items, Naming Configuration Items, Acquiring Configuration Items, Baselining Processes and Configuration Control.

3.1 Identifying Configuration Items

There are three types of CIs; Document CIs, Code CIs and other CIs. One or more files can collectively be identified as a CI, where each of those files is also considered a CI. Files that are generated from CIs, through compilation or some other repeatable process, are not considered as CIs (e.g. .pdf file created from compiling source code, the source code is considered as a CI whereas the .pdf file is not).

3.1.1 Document Configuration Items

The following are the Document CIs of Team Daedalus:

1. AREP
2. BP
3. RAP
4. RMP
5. SADD
6. SCMP
7. SDD
8. SPMP
9. SQAP
10. SRS
11. SS
12. SVVP
13. SVVR
14. TM
15. TP
16. TR
17. UD

3.1.2 Code Configuration Items

The following are the Code CIs of Team Daedalus:

1. Application
2. Common
3. Context
4. Hardware
5. Prototypes
6. TR

3.1.3 Other Configuration Items

Other configuration items are mainly formed by managerial and research items of Team Daedalus. CVS will assign appropriate revisions to these items as they mature, thus providing version control of the items. The following are the other CIs for Team Daedalus:

CI Name	CI Description
<i>Audit</i>	All files relating to Audit.
<i>Design_notebook</i>	All files relating to design notebook.
<i>Log</i>	All log files relating to baseline, change log, change request, decision, info sheet, logistic, meeting metric, mini audit, PIP, risk, Roadmap, technical request, timesheet, web update and workshop.
<i>Management</i>	All files relating to management.
<i>Meeting</i>	Meeting agendas and minutes.
<i>Misc</i>	Other miscellaneous files such as contact detail of team members, skill assessment and etc.
<i>Research</i>	All files relating to research.
<i>Review</i>	All files relating to review.
<i>Script</i>	Shell scripts.
<i>Social</i>	All files that relating to social outings.
<i>Template</i>	Template files.
<i>Test</i>	All files that relating to testing.
<i>www_public</i>	All files regarding to the team's website.

Table 3: Other Configuration Items

3.2 Naming Configuration Items

Each CI is identified by a unique name and its evolutionary stage is indicated by a revision number (refer to table 4 for evolutionary stages). Tagging is used for easier identification at certain stages during a CI's evolution. This section will explain how the Document and Code CIs are named, revisioned and tagged.

3.2.1 Document Naming, Revisioning and Tagging

3.2.1.1 Naming Document

Refer to SQAP for directory and files naming conventions.

3.2.1.2 Revisioning Document

1. Document revisioning is managed by document CIMs. CIMs will be in charge of manually entering the correct revision number throughout the development of the document.
2. Revision numbers are represented in the following format:

X.Y.Z

- *X* represents the number of times the document has been baselined.
 - *Y* represents the number of times the document has been external reviewed or reviewed by supervisor or reviewed by client.
 - *Z* represents the number of times the document has been internal reviewed or went through workshop.
 - For example, the revision number: 1.2.3 means that the document went through two external reviews, three internal reviews and one CR after the first baseline.
3. If the revision number begins with 0.0.0, the revision number is incremented when:
 - (a) An internal review or workshop is conducted and modification after the internal review or workshop is complete. The revision number *Z* is incremented by 1.
 - (b) An external review or supervisor review or client review is conducted and modification after the supervisor review is complete. The revision number *Y* is incremented by 1.
 - (c) A baselining of the document is approved by CCB. The revision number *X* is incremented by 1 and the revision number of *Y* and *Z* are reset to 0.

3.2.1.3 Tagging Document

For the ease of traceability, document tagging is performed to mark an evolutionary stage of the document's development. It allows document CI revisions to be compiled together at one time, even though they may be of different revision numbers. The following table identifies the labelling standards for assigning tags to the evolutionary stages of documents.

Evolutionary Stage	Tag Label
Internal Review	Document_Internal_Review_X
External Review	Document_External_Review_X
Supervisor Review	Document_Supervisor_Review_X
Client Review	Document_Client_Review_X
Document Baselining	Document_Baseline_X

Table 4: Tag labels for different evolutionary stages of documents

X represents the number of V&V activities (refer to SVVP).

Document represents the name of the document.

Example: SQAP_Internal_Review_1.

3.2.2 Code Naming, Revisioning and Tagging

3.2.2.1 Naming Code

Refer to SQAP for directory and files naming conventions.

3.2.2.2 Revisioning Code

Refer to 5.6 of BP for revisioning code CIs.

3.2.2.3 Tagging Code

Refer to 5.5 of BP for tagging code CIs.

3.3 Acquiring Configuration Items

This section describes the procedures for acquiring CIs from the CVS repository.

3.3.1 Storing Configuration Items

3.3.1.1 Document CIs

The location of Document CIs are described in the following table.

CI Name	Module Location
AREP	\$GROUPCVS/Document/AREP
BP	\$GROUPCVS/Document/BP
RAP	\$GROUPCVS/Document/RAP
RMP	\$GROUPCVS/Document/RMP
SADD	\$GROUPCVS/Document/SADD
SCMP	\$GROUPCVS/Document/SCMP
SDD	\$GROUPCVS/Document/SDD
SPMP	\$GROUPCVS/Document/SPMP
SQAP	\$GROUPCVS/Document/SQAP
SRS	\$GROUPCVS/Document/SRS
SS	\$GROUPCVS/Document/SS
SVVP	\$GROUPCVS/Document/SVVP
SVVR	\$GROUPCVS/Document/SVVR
TM	\$GROUPCVS/Document/TM
TP	\$GROUPCVS/Document/TP
TR	\$GROUPCVS/Document/TR
UD	\$GROUPCVS/Document/UD

Table 5: Storage of Document CIs

3.3.1.2 Code CIs

The location of Code CIs are described in the following table.

CI Name	Module Location
Application	\$GROUPECVS/Code/Application
Common	\$GROUPECVS/Code/Common
Context	\$GROUPECVS/Code/Context
Hardware	\$GROUPECVS/Code/Hardware
Prototypes	\$GROUPECVS/Code/Prototypes
TR	\$GROUPECVS/Code/TR

Table 6: Storage of Document CIs

3.3.1.3 Other Configuration Items

The location of Other CIs are described in the following table:

CI Name	Module Location
Audit	\$GROUPOCVS/Audit
Design_notebook	\$GROUPOCVS/Design_notebook
Log	\$GROUPOCVS/Log
Management	\$GROUPOCVS/Management
Meeting	\$GROUPOCVS/Meeting
Misc	\$GROUPOCVS/Misc
Research	\$GROUPOCVS/Research
Review	\$GROUPOCVS/Review
Script	\$GROUPOCVS/Script
Social	\$GROUPOCVS/Social
Template	\$GROUPOCVS/Template
Test Related Files	\$GROUPOCVS/Test
www_public	\$GROUPOCVS/www_public

Table 7: Storage of Other Configuration CIs

3.4 Baselining Procedure

3.4.1 Entry Criteria for Baseline

Before a BR is made, the following software configuration activities are administered:

1. The CIM is in charge of creating the CI by ensuring that tasks are assigned to sub-team members to produce the CI to be ready for the verification activity (Refer to SVVP) of the CI. However, it is the responsible sub-team leaders responsibility to assign tasks to the sub-team members.
2. When the document CI is sufficiently complete, CIM manages the evolution of the document CI through the verification activities (Refer to SVVP) by ensuring all necessary changes are implemented.
3. CIM is in charge of setting the document revision and tagging according to the standards described in section 3.2.1 and section 3.2.2 for each evolutionary stages.
4. Responsible sub-team leaders are responsible for deciding when CI should be baselined.
5. Changes to the CI can be done freely and need not go through the CCB if the CI is not baselined.
6. At least one internal and one external review must be done before CIM can decide to baseline a document. However, internal review is not required for document that is produced by one team member.
7. The SRS must pass a client review in order to be baselined.
8. A proof read must be done for each CI document to check spelling and grammar of the whole document.

3.4.2 Baselineing

To baseline a CI, the following process must be followed:

1. Document Manager is allowed to request for baselineing a CI's section. The baseline procedures for baselineing a CI's section are the same as baselineing CI.
2. To request for baselineing a CI, CIM must fill in a BR form (refer to section 5.2 Appendix A) and email it to CCB using [440 Config] email tag to make a formal request.
3. The BR template is named `template_baseline_request.txt` and located at

`$GROUPOCVS/Document/Template.`

4. Completed BR forms are to be saved as

`ciname.br.number.txt`

- *ciname* is the name of the CI to be baselined (refer to section 3.1.1, 3.1.2 and 3.1.3 for CI's name).
- *number* is the BR's ID.
- Example of the file name: `sqap_br_01.txt`.

5. The completed BR form is to be stored in

`$GROUPOCVS/Document/Log/Baseline/`

6. To evaluate the request,
 - (a) CM should check the approval criteria for the baselined CI from the history section in the BR form (refer to section 5.2 Appendix A).
 - (b) CM must email the CIM within 24 hours after the BR is being made if more information is needed for the evaluation.
7. To approve the request for the first baselineing of a document CI, the document CI must pass the following criteria:
 - (a) At least one external review and one internal review.
 - (b) The SRS must pass a client review in order to be baselined.
 - (c) Modification corresponding to external review and internal review is being done.
 - (d) A proof read to check the spelling and grammar of the whole document.
8. To approve the request for the baselineing after first baseline of a document CI, the document CI must pass the following criteria:
 - (a) Modification corresponding to the V&V activities that being done after the previous baseline has been done.
 - (b) A proof read to check the spelling and grammar of the whole document.

9. The result of the request must be returned to the CIM within three days after the BR is being requested. CM is will be responsible for approving BR. However, PM will take over CM's responsibility if CM is also CIM at the same time.
10. If the request of baselining the CI is disapproved,
 - (a) CM or PM will set the status of BR as **Disapproved** and states the reason why the BR is not approved from the result of the evaluation.
 - (b) CIM can modify and re-submit the same BR again after modifications have been made by submitting another BR form.
11. If the request of baselining the CI is approved,
 - (a) CM or PM will set the status of BR as **Approved** and states the rationale for why the BR is approved.
 - (b) CIM will baseline the document within 48 hours after the baseline approval is notify via email by the CM. CIM will ensure the CI revisioning and tagging follow the standards described in sections [3.2.2](#) and [3.2.1](#).
 - (c) For document CI, CIM is responsible to make ensure the baselined version of the document is on the web.
 - (d) After baselining the CI, CIM should email the team and the subteam that produce the CI about the baseline using

[440 General][440 sub team] *ciname* Baseline.

- *ciname* is the name of the CI to be baselined(refer to section [3.1.1](#), [3.1.2](#) and [3.1.3](#) for CI's name).
- *subteam* is the tag for the specific sub team(Refer to SQAP for email tag).
- Example of the email's subject: [440 General][440 QA] SCMP Baseline.

3.5 Configuration Control

This section describes the process for each change on baselined CIs. The process states the sequence of steps for requesting, approving or disapproving and implementing the changes.

Below is the table for the CIMs that involve in this process for CR on different CI:

CI	CIM
SRS	SRS Manager, SADD Manager, Coding Leader and Testing Leader.
SADD	SADD Manager, Coding Leader and Testing Leader.
SDD CIs	SDD Manager, Coding CIM and Testing Leader.

Table 8: CIMs that involve in CCB

3.5.1 Requesting Changes

Changes can only be requested to baselined CIs. However, CRs are not required for QA sub-team Documents as changes will be evaluated during Process Improvement Procedure (PIP) and during the Review or Audit workshop. For further information of PIP and Review and Audit Process

please refer to SQAP (Section 5.5) and RAP respectively. CRs are not required for if changes made don't influence the content of the CIs. To request a change to a baselined CI, the procedures are as follows:

1. Any CIM can request for the changes to CI by filling a CRE form (refer to section 5.3 Appendix B). Email CCB the CRE form using [440 Config] tag to make a formal request for the change to the CI.

2. The CRE template is named `template_change_request.txt` and located at

`$GROUPCVS/Document/Template.`

3. After the CR part of the form is completed, CRE form is saved as

`ciname_cre_number.txt`

- *ciname* is the name of the CI(refer to section 3.1.1, 3.1.2 and 3.1.3 for CI's name).
- *number* is the CR id.
- Example of a file name: `sqap_cre_01.txt`.

4. The CRE form is to be stored in

`$GROUPCVS/Document/Log/Change_request/.`

5. Upon receiving a CRE form, CM is in charge of checking the completeness of the CR part of the form by making sure sufficient data is given for following sections:

- Change Description
- Reason for Change
- Change Benefits
- Change Impact

6. If the CR section of the form is incomplete, CM will set the status of the form to **Disapproved**. An email will be sent to the CIM that made the CR with an explanation of why the CR is disapproved. Modifications are allow to be made before the CR is reissued.

7. If the CR part of the form is complete, CM will set the status of the form to **Evaluation**. Then, CM is in charge of contacting other CIMs that will involve in this process if necessary and conducting the activities for evaluating changes and approving or disapproving changes within two days after the CRE has been sent to CCB.

3.5.2 Evaluating Changes

All CRs are to be evaluated by considering:

1. Change Description
2. Reason for Change
3. Change Benefits
4. Change Impact, especially regarding the impact on interdependent CIs, impact on the schedule of the project and availability of the sufficient resources to implement the change within the schedule.

3.5.2.1 Evaluation Procedure

The primary method of evaluating CRs are via email discussions. For more substantial and CRs that may have a greater impact on the project, CM can organise CCB workshop to resolve the CR. CRs can also be evaluate in team meetings if deemed necessary.

1. Evaluation via Email

To evaluate the CR via email discussion, the procedures are as follows:

- (a) The CM instigates an email discussion amongst CCB members by eliciting the change considerations that are stated in section 3.5.2. The closing date of the discussion is stated in the email according to the urgency of the change and the proposed resolution date will be given in the CRE form.
- (b) On the closing date, the CM will summarise all issues raised by the CCB members during the email discussions. This email will be treated as a solid basis to approve or disapprove a CR.
- (c) If a CCB member disapproves the CR and can't be resolution can't be reached via email, the CM will organise a CCB workshop to resolve the CR evaluation to approve or disapprove the CR.

2. Evaluation via Workshop

To evaluate the CR during a workshop, the procedures are as follow:

- (a) CM will organise a workshop and send CCB members a notification via email. Workshop initiating procedures according to the SQAP section 4.1 will be followed.
- (b) CM will bring the CRE form to the workshop.
- (c) Each CCB member will assess the CR on their respective areas and share these in the workshop.
- (d) CM is responsible to summarise the result of the evaluation.
- (e) Decision of approving or disapproving the change will be made by voting (refer to Voting Procedures in SQAP) after the summary of the evaluation of the change has been made.
- (f) The evaluation of the change and its outcome will be recorded by the CM during the workshop.
- (g) After the meeting, CM will update the CRE form in the repository.

3.5.3 Approving or Disapproving Changes

Once the CR has been evaluated (refer to section 3.5.2.1) and voting has been conducted (refer to Voting Procedures in SQAP) , the CR is resolved by completing the following actions:

1. CM needs to set the status of CRE form as **Approved** or **Disapproved**.
2. CM needs to fill in "CCB Change Evaluation and Resolution" part of the same CRE form (Refer to 5.3) with the summary of the evaluating changes and voting activities.

3.5.4 Implementing Changes

To implement the changes, the procedures are as follows:

1. CIM is in charge of making sure the resulting actions from the CRE form are implemented.
2. CIM prepares and schedules work package to make the changes as stated in CRE form.
3. CIM will ensure the CI revisioning and tagging follow the standards described in Sections [3.2.2](#) and [3.2.1](#).

4 Software Configuration Resources

This section identifies the software tools, techniques, equipment, personnel, and training necessary for the implementation of software configuration management activities.

4.1 Concurrent Version System

CVS is used to archive and maintain all CIs. CVS will provide version control for all the files in the repository. This section provides the CVS commands to assist in configuration management. There is also a help file on CVS commands(`cvs_help.txt`) in

`$GROUPCVS/Document/Misc.`

4.1.1 Setting Up CVSROOT

For the ease of using CVS, team members are advised to add following line to their `.bashrc` file:

```
export CVSROOT=/home/se440/s440gf/Repository
```

4.1.2 Checking out Files Stored under CVS

To work on files stored under CVS:

1. Change the working directory to the local workspace. Local workspace is an individual workspace where a team member can work on his/her personal files.
2. Check out the module that contains the files into the current workspace by using the command:

```
cvs co module
```

3. The module will then be existed in the local workspace.

4.1.3 Adding Files or Directories into CVS

If a team member wants to add directory to the same layer as the `$GROUPCVS/Document` directory (refer to group directory structure in SQAP), he/she must go through the PIP procedures (refer to SQAP). Otherwise, to add files or directories:

1. Change the working directory to where the file(s) or directory exists.
2. Add text file(s) by using the command:[3.2.1](#)

```
cvs add file(s)
```

3. Add binary file(s) by using the command:

```
cvs add -kb file(s)
```

4. The file(s) or directory must be committed to the repository (refer to [4.1.7](#) for committing files) to perform the actual adding of the file(s) to the repository.

Directories are added in the same ways as adding text files by replacing `file(s)` with name of the directory.

4.1.4 Deleting Files from CVS

To delete file(s) from CVS:

1. Change the working directory to where the file(s) exists.
2. Delete the copy of the file(s) from the local directory.
3. Remove the file(s) from the CVS repository by using the command:

```
cvs remove file(s)
```

4. The file(s) must be committed to the repository (refer to [4.1.7](#) for committing file) to perform the actual removal of the file(s) from the repository.

4.1.5 Deleting Directory from CVS

To delete directory from CVS:

1. Team members do not have privileges to delete an existing directory in CVS as CVS does not support deletion of directory.
2. However, team members can remove the empty directory in their working directory by using the command everytime when they update their working directory:

```
cvs update -P
```

4.1.6 Updating Files

To bring files in the working directory up-to-date with the copy in the CVS repository:

1. Change the working directory to where the file(s) exists.
2. Update the entire module with the CVS repository by using the command:

```
cvs update
```

3. Update the file(s) within the CVS repository by adding the name of the file(s) after the command shown above.
4. Once the update command is executed, CVS will output the status of each file. The status could be one of the following:
 - (a) *U* : The file has been successfully updated.
 - (b) *M* : The local copy of the file has been modified but not yet committed into the repository.
 - (c) *C* : There are conflicts between the local copy and the copy in the repository. These need to be manually resolved.

4.1.7 Committing Files into CVS

To commit files:

1. Change the working directory to where the file(s) exists.
2. Before committing files, it is a guideline for members to:
 - (a) Ensure the files can be compiled with no errors using the command `make` as specified in the makefile located in each document directory.
 - (b) Get rid of M characters in the PC format documents by using the shell script, `convert`, to convert the format of files from DOS to UNIX. The file, `convert` is located at:

```
$GROUPCVS/Script/convert
```

- (c) The command to convert the format of files from dos to unix is

```
dos2unix <filename> <filename> command.
```

- (d) Commit changes done to every file in the directory in CVS by using the command:

```
cvcs commit
```

- (e) Commit the file(s) into the CVS repository by adding the name of file(s) after the command shown above.
- (f) Once the commit command is executed, a CVS log template will appear automatically. Changes made to each file must be clearly stated in the template.

4.1.8 Symbolic Naming

For the ease of traceability, a symbolic name is assigned to all files in a CVS directory using the CVS tag feature at each evolutionary stage (refer to section 3.2.1 and 3.2.2). To do symbolic naming:

1. Change the working directory to where the module exists.
2. Assign a symbolic name that stated which evolutionary stage it is corresponding to by using the command:

```
cvcs rtag "symbolic name" module
```

- Example of "*symbolic name*": "SQAP_Internal_Review_1".
- *module* is the path of the directory in the repository where the CI is located (e.g Document/SQAP).

3. The existing version of the CI can be checked out with the command:

```
cvcs co -r "symbolic name" module
```

4.1.9 Renaming File in CVS

To rename the file in CVS:

1. Change the working directory to where the file exists.
2. Team member who wish to rename the file need to copy the existing to new file by using the command:

```
mv <file> <new_file_name>
```

3. Then, remove the file from CVS by using the command:

```
cvs remove <file>
```

4. At last, add back the file with new file name by using the command:

```
cvs add <new_file_name>
```

4.1.10 Renaming Directory in CVS

To rename the Directory in CVS:

1. Team members do not have the privilege to rename an existing directory in CVS.
2. Team member who wish to rename the directory need to send an email to CM stated the reason for renaming the directory.
3. CCB is required to rename the directory.
4. CCB must then send an email to the team to inform that the renaming of the directory has been done.

4.2 Backup and Recovery Process

There is a risk that problems may occur with the Team Daedalus repository that could result in the loss of data. This section describes the procedures to be followed to back up and recover the Team Daedalus repository.

4.2.1 Backup Process

1. The backup of \$GROUPOCVS is to be performed at 3am on Monday.
2. The backup of Code and Test directory in the \$GROUPOCVS is performed every two days.
3. The backup of \$GROUPOCVS is to be named as [mmdd]_backup.tar.gz. The backup of Code directory is to be named as [mmdd]_code.backup.tar.gz and the backup of Test directory is to be named as [mmdd]_test_backup.tar.gz. For details on naming convention please refer to section 6.1.
4. Upon completion, the Librarian will be responsible to retrieve the data from the backup to check the integrity of the backup. Then, Librarian will also transfer the backup to SAG server for storing.

4.2.1.1 Recovery Process

1. If team member discovers lost or damaged files, an email should be sent to the Librarian identifying the specific files.
2. Librarian will access the most recent backup of the directory to recover the lost or damaged files.
3. Librarian will then email the team to notify the recovery of the files.

5 Appendices

5.1 Change Log

Date	Section	Descriptions
14/08/2004	3.3.1.2	Code configuration items being added.
14/08/2004	3.2.2	Add referencing for code naming, revisioning and tagging.
16/08/2004	4.2.1	Modify backup procedures according to PIP 59. Backup is done for Code and Test directory every two days.

Table 9: Change Log

5.2 Appendix A: Baseline Request Form

```
=====
Baseline Request form
-----
Baseline Request ID:
Date: date request is made
Requested by: (normally is document manager)

Baseline item: Item to be baselined such as the whole SQAP or build 1,...

Version # before request:

Proposed Baseline Date:

History:
-----
-state all review activities, V&V activities that have passed
-state all test activities that have been completed
-previous baseline request that has been rejected.

=====
Status : Approved/Disapproved
Approved by: Configuration Control Board
Rationale: state the criteria that request is approved or disapproved by.
Version # before approval:

=====
```

5.3 Appendix B: Change Request and Evaluation Form

=====
Change Request

Status: Evaluation/Approved/Disapproved/Completed

Change Request ID: #number of change request

Title of Change:

Request Date:

Requested by:

CI Name:

CI Component: Components/sections for the CI that needed to be changed.

CI Version:

Type of change: Addition/Modification/Omission

Urgency of change: Low/Medium/High

Proposed Resolve Date:

Change Description:

Reason for Change: Description of the Problem

Change Benefits: Recommended solution

=====
CCB Change Evaluation and Resolution

Client: (only for sign-up SRS)

CCB members:

Date resolved:

Change Evaluation

Pros:

Cons:

Evaluation Outcome

Client Decision: (only for sign-up SRS) Approved/Disapproved

CCB Decision: Approved/Disapproved

Rationale: state the criteria that request is approved or disapproved by.

Version after approval:

Resulting Actions: Prepare and schedule work package to make the changes.

=====