Middle East Technical University
Department of Computer Engineering

Configuration Management Plan
for
“CL@SS++”

profIT

1250125 - İsmail ÇETİN
1250349 - Yavuz GÜRCAN
1250356 - Recep GÜRLEK
1250661 - Hakan ÖZTÜRK

Spring 2004-2005

1 Introduction

Change is unavoidable when computer software is evolving. Moreover change increases the amount of confusion among software engineers working on a project. This is true for our project too. So we are going to make a lot of changes on our project as our design considerations may change. Also we can add new features and add new capabilities during the implementation process. But we have to handle those changes some way. This handling can be accomplished by Software Configuration Management Plan which defines the implementation of configuration management of a particular software project. It describes the concepts, processes, and procedures for an IT development project. In other words by this plan we can identify the change, control the change, make sure the plan is implemented correctly and to make sure that we report the change to others.

Although our modules do not depend very tightly on each other any change will be reported to other members as they have some common features for example networking and data transfer modules. Because of this fact, changes in any of these modules have to be implemented correctly and everybody involved in the project should be aware of the changes. Otherwise it may lead to a confusion in the project development. Software Configuration Management Plan prevent us from those confusion.

1.1 PURPOSE OF THIS DOCUMENT

This document includes the general the Configuration Management Plan of our project Cl@ss++. The Configuration Management Plan provides a framework within which the four primary Configuration Management functions, which are configuration identification, configuration control, status accounting, and audits and reviews, are managed. The CMP documents the plans for performing configuration management on our project.

1.2 SCOPE OF THIS DOCUMENT

The scope of this document is identification of Software Configuration Items (SCI), management of change control, auditing the changes and reporting the changes in order to inform the involved

people into the project. By this way the changes in the project will be made without any confliction.

We divided our plan into six parts according to topics explained. First part will be the introduction part. Purpose of this document, its scope, its structure and abbreviations used in this document are explained in this part. We have included the reference part at he end of this part.

In the second part we will explain the configuration management organization of our group. The responsibilities of each group member in configuration management are stated at this part. In order to have an effective and successful configuration management in projects that involve more than one developer, using tools for configuration management is inevitable.**** So we specify the tools that we will use in our change management issues and stated them in this part our report.

In the third part we will explain the process of the configuration management process that is how we will handle configuration management. Detailed documentation and reporting of changes is very important for configuration management. Why change is requested, how the change is handled should be carefully documented in order to track changes. So we designed our reporting strategy and explained this reporting strategy in the part.

In the fourth part configuration management report we will be explaining the time management of our configuration management process. The tasks expected finish times will be stated in this part.

In the last two part of our configuration management plan we will explain the resources. And also some optimization issues are stated in this final part.

1.3 TERMS, ACRONYMS AND ABBREVIATIONS

This part of our configuration management plan explains some abbreviations in our project. It also includes some technological terms that we will use through our development phase. Some of the abbreviations and technical abbreviations are explained below.

Abbreviation	Definition
LS	Living Schedule
QM	Quality Management
QMP	Quality Management Plan
SCF	System Changes Folder
TF	Test Folder
ECR	Emergency Change Request
CRF	Change Request Folder
CVS	Concurrent Version System
XML	Extensible Markup Language
UT	Unit Testing
FT	Functional Testing

1.4 DOCUMENT AND WEBSITE REFERENCES

Our configuration management plan refers many documents that we have prepared before. . In this detailed explanation we refer to some documents and web sites these documents are listed below.

- Requirement Analysis Report
- Initial Design Report
- Detailed Design Report
- http://www.cmcrossroads.com/bradapp/acme/scm-defs.html#IEEE_Std_610

2. CONFIGURATION MANAGEMENT RESPONSIBILITIES

In this section we will mention about the structure of our team and give explanation of each member's responsibilities and roles in the CM Plan.

2.1 ORGANIZATION

Since our project is a small project we will not have different teams for CM tasks. Instead every member of our project will be responsible for the CM issues. Although our project consists of different modules that are implemented by different members, we all have to keep track of the changes and we all have to take part in the CM tasks for both individual's own tasks and the other's tasks as a whole. The roles in the CM can be categorized as the following categories;

- Program Director
- QM Specialist
- CM Specialist
- CM Team
- Development Contractor
- Testing Team

2.2 RESPONSIBILITIES

First we will choose a CM leader to control all the control issues. Other than this all the members will do all the responsibilities as we all have different modules. The leader will be responsible for controlling the changes common to modules and he will inform other members about this. Also when integrating the changes with other modules, all the group members will be ready for controlling the changes. Also all members will inform the leader about the change of their own module, by this way the coordination between the group members will be provided.

According to the organizational roles, required responsibilities of each role in is given below.

Program Director

- is responsible for overall CM activities within the lifecycle of the project.
- provides the resources necessary for CM.
- makes the final decision about the changes for the part of the code he/she has written.

QM Specialist

- is responsible for controlling all QM activities and informs the other members of the team about those activities.
- audits CM program and implementation of the CM program.
- provides that CM audits are conducted on all baselines.
- should collect metrics and creates reports for our project management to support the quality control.

CM Specialist

Each member in our project acts as a CM Specialist for the part of the software that he/she will implement and for the other parts of the software that will be implemented by the other members. So, a CM specialist:

- is responsible for informing the other members of the team about all CM activities that he/she is doing individually.
- creates and maintains the CMP.
- should be in coordination with the other members in the team about all CM activities.
- provides that the software baselines are documented, maintained and controlled.
- should be in contact with the other members of the team for presenting and coordinating change requests.
- identifies risks about changes that will be made in any part of the implementation and related to this, implements appropriate risk management measures.

CM Team

- should take role in all CM activities.
- accept the roles that they have taken over in CMP of PROFIT and perform duties according to their roles.

Development Contractor

- is responsible for delivering the resources needed to create the baselines.
- provides CM tools that are appropriate for the CMP.
- is responsible for creation and implementation of a working CMP.

Testing Team

- should do unit testing (each member should test the part of the software that he/she will implement after he/she will complete implementation).
- should do the testing of the integrity of the part of the software that he/she will implement with the other parts of the software that will be implemented.

2.3 TOOLS AND INFRASTRUCTURE

As we stated in our design report we will be implementing our modules concurrently. All the members will be making changes on different files most of the time but sometimes there will be changes in some common files. Also we will integrate some of the modules. In these situations we have to merge files. As a result all the group members should be aware of some changes that are done by the others. In order to follow those changes we will have to use some mechanism.

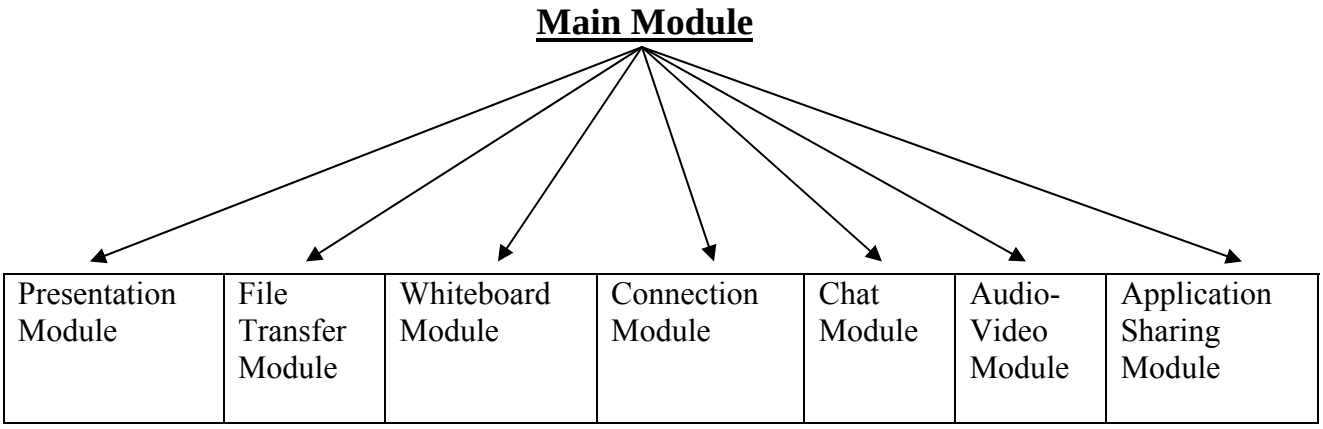
We will use Concurrent Versioning System (CVS) to accomplish the above task and control versioning. Each group member will use CVS and commit their changes when they make changes on their own modules. These regular committing operations will provide us less merge operations in source files and will make other groups more up to date in other modules. Before making changes the members will test their changes carefully on their own machines in order to prevent later conflicts in changes. By applying this testing strategy there will reasonable number of history files in CVS and managing them will be much more efficient.

2.3 VERSION CONTROL

In order to make the development phase of PROFIT easier, a tree structure is decided to be used for the implementation. The tree structure is explained below:

Implementation Tree

In order to use CVS effectively, the directory structures in CVS should be designed in the same way that we divided our modules, because each module is done by a different member. Directories should agree with our modules in our project design. And in our CVS structure for each module there will be a corresponding documentation directory. We will use these documentation directories in configuration management issues. Also the general API directories of .NET environment that will be used in the classes will be in these directories.



All of the directories under these folders will be our source codes and the related files of those modules. Since we are going to implement the modules separately, any change in one of the folders has no effect in the other directories.

3. CONFIGURATION MANAGEMENT PROCESS

CM activities of our project include all functions and tasks required in managing the configuration items of the as specified in the scope of this plan. All CM activities in this CMP are identified and controlled through the following CM processes.

- Configuration Identification
- Configuration Management and Control
- Configuration Status Accounting
- Configuration Auditing

3.1 CONFIGURATION IDENTIFICATION

In this process configuration items for each product released in the life cycle of our project are selected, and their functional and physical characteristics are recorded. These characteristics are called Configuration Items (CIs). These CIs will be used in identifying the current status of each product released during the life cycle of Cl@ss++.

3.1.1 Identifying Configuration Items

Five types of CIs will be used in our CM Plan. They are:

- . Documents,
- Baselines,
- Hardware,
- Software.

Each of the CIs is explained below.

Documents:

All deliverable documents must follow the naming conventions specified later in this section. All documents are placed into a document library after initial formal approval done by all members of our group. Any change to the document following initial formal approval must follow the configuration control procedures. The list of deliverable documents and their status can be reached through our website.

Baselines:

Baselines are composed of CIs at a specific time in our progress. The baselines are for controlling changes to the CIs in our process. We will create new baselines and approve any changes to a baseline. Baselines will be tracked, audited, retained and version controlled in the Concurrent Versions System (CVS) version control tool. We have four different types of baselines: functional, allocated, product and production.
The baselines and their contents are as follows:

Functional Baseline (FBL): A functional baseline is established following the Requirements Analysis (RA) and delivery of required documentation. Every member of PROFIT will be responsible of verifying all required documents and configuration items, related to their developments, are established properly.

Allocated Baseline (ABL): An allocated baseline is established at the end of the design phase. It is verified that all required documents and configuration items are established properly. The CIs in the allocated baseline include all initial system designs, detailed system designs, and associated test plans. The documentation about ABL are located on the web site of SolutionSDC as Initial Design Report and Detailed Design Report.

Product Baseline (PBL): A product baseline is established for each release of the system at the end of the acceptance testing phase. The developer verifies that the tested product is exactly described in the product baseline documents. It includes all CIs, CM data, system test plan reports, manuals, and other plan documents.

Production Baseline (PL): A production baseline is operational when it is established at the end of each release or when a major change to the baseline takes place. It is established at the end of acceptance testing of that release. It includes all CIs, CM data, system test plan reports, acceptance test plan reports, manuals, and other plan documents as the product baseline. The production baseline is the actual system release in use.

Hardware:

Hardware includes the physical equipment required for designing, building, testing and running the system. We have already listed the components of the hardware configuration items in the various reports of us. The documents about hardware are explicitly updated each time a piece of equipment is added, changed or removed from use.

Software:

Software includes all non-document files needed to create a running system. The documents about software are explicitly updated each time a piece of software is added, changed or removed from use in our CVS directory. This explicit updates can be seen at the end of the reports under a separate title.

3.2 CONFIGURATION MANAGEMENT AND CONTROL

Configuration management and control is the process of development, change request, defect tracking and system management. This document is only concerned with formal CM, where all changes are controlled by all of our members.

3.2.1 Development

Development phase of configuration management and control can be done by each member of our project. This process will go in parallel to the living schedule prepared before. This schedule can be accessible from the website. In the implementation phase of a module or a class in CL@S++, unit testing will be used. So, possible conflicts that may occur can be easily detected

by us, on time. Moreover, when a member of project completes the implementation of a task, the source code is added to the related folder in his account. The files in the account must be added to the project tree stored on the CVS server by each member. The CVS server checks whether the added files makes any conflicts with the previous ones or not. If there is not any conflict, living schedule will be update by us by marking the task as it is completed. For a complete test, the whole program will be loaded into a system. The tester will use the software with different data. The result will be documented in the tests folder.

3.2.2 Change Request

A change request is done when a conflict is detected between the newly added code and the previous code. The change request starts with documenting the request on the module, which is entered into the change request folder. All team members can submit a change request.

Approving or Disapproving Requested Changes

All requests are stored under the change request folder but not all requests are approved. If request is approved, it is explicitly explained to the other group members in the meetings. Also the change is stored in the log file of the CVS server implicitly. Furthermore, the changes are reported in the change done folder.

Emergency Change Request

For an emergency change request that needs an immediate fix, the developer makes the change. Next, the developer must inform the other members and add the change request form to the change request folder.

Implementing Changes

When the change request has been approved, the change is implemented by the developer who has initially implemented the changed part. Then the change is reported into the change done folder by adding the related information into the change request form.

3.2.3 Defect Tracking

Defect tracking is done in every addition, subtraction or change done to the source code. This can be easily done by using CVS. The reporting about the defects are exactly the same as the change requests.

3.2.4 System Management

System management will be done in the light of tests done on the system. All possible changes will be done by the members in the meetings. If any change is done on the main structure of the system, this is reported in system change folder.

4. PROJECT SCHEDULE - CM MILESTONES

4.1 CM SCHEDULES

We have designed our CM Schedule so that every member in the team can get information about the progress of tasks and coordination of the Configuration Management activities. This way it will be possible for us to manage the activities in CM.

4.2 SCHEDULE FOR REGULAR CM ACTIVITIES

There are three different types of meetings that we will carry out. These meetings are Software Engineering Meetings to design the plan when a change is approved, emergency meetings that will be held when we encounter unexpected problems, and regular meetings. All these meetings and their content will be further discussed in the following sections.

4.2.1 Software Engineering Meetings

Software Engineering Meetings are to be held upon the approval of a change request. During our software engineering meetings the methodology to handle the approved change request and its effects on our overall system will be discussed. Also the new workload caused by the change will be shared among the members of the team. Results of these meetings, offered solutions, accepted

ones and effects of this approved method on our project will be noted on the Weekly reports and our living schedule will be updated accordingly.

4.2.2 Emergency Meetings

Emergency Meetings are to be held when unexpected problems occur regarding the implementation phase or the logic of the program. These meetings will be arranged and conducted in a short time so that we do not fall back of our schedule and so that every member can keep producing. Within the scope of these meetings solutions to the new problems will offered and their effects to our project will be discussed and again division of workload will be done for the implementation of these solutions.

4.2.3 Regular Meetings

Regular Meetings will be held every week to update our living schedule and to exchange ideas for different implementation methodologies. Within the scope of regular meetings we will try to check the status of the implementation and see if there are any problems. Also, the change requests will be discussed. If an update needs to be made regarding division of workload due to a member's weekly load it will also be discussed in regular meetings.
At the end of the Regular Meetings, the decisions taken will be recorded to the Weekly Progress reports. Also, the updated Living Schedule and the Weekly Progress reports will be published on the web site of our project.

4.3 MANAGEMENT REVIEW OF CM ACTIVITIES

Our CM plan will be a two phase plan. While we carry out the activities discussed in our Software CM Schedule we will also review them and our Software CM Schedule for effectiveness and do the necessary updates. The sequence and dependencies in between CM activities like creation of CBLs, change control procedures, and dates for configuration audits to program milestones or events will be scheduled in our Software CM Schedule.

5. PROJECT RESOURCES

We have identified some online and printed references that will be useful while developing our CM Plan.

1-) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsintro7/html/vxconCommentingCode.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vcoriXMLDocumentation.asp
http://msdn.microsoft.com/msdnmag/issues/02/06/xmlc/

2-) Joseph Phillips, 2002, IT project management: on track from start to finish, McGraw-Hill, Osborne

3-) Struts User Guide at http://struts.apache.org/userGuide/index.html

4-) Kentaro Nobeoka, 1995, Reorganizing for Multi-Project Management, Paper presented at the IMVP Research Briefing Meeting

5-) National Competency Standards For Project Management, Draft-4, 2003, http://www.aipm.com.au

6-) Fincher, Anita, and Levin, 1997, Project Management Maturity Model, Project Management Institute 28th Annual Seminars/Symposium, Chicago, Illinois, September 29 - October 1.

6 PLAN OPTIMIZATION

As we progress through our project the CM Plan and the related tasks, activities and approaches will also be reviewed and improved. While guiding our CM activities and establishing the baseline for change methodologies, our CM Plan will be updated regularly. Hence, we hope to attain maximum level of flexibility and optimization in our plan.