MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT of COMPUTER ENGINEERING

SOFTWARE CONFIGURATION MANAGEMENT PLAN for DEVEMB Project

By
ResolveSOFT

Mehmet Fatih DOĞU
Hayri Çağlayan ERDENER
Adem HALİMOĞLU
Ulaş TUTAK

11.03.2007

Ankara

TABLE OF CONTENTS

1. INTRODUCTION

1.1 The Purpose of the Configuration Management Plan

Organization is crucial part of the large scaled projects. One of the most important parts of this organization is a well designed Configuration Management Plan (CMP) because it is very useful for the programmers during the development of the project. Normally, CMP should

- ➢ identify the components and structures,
- ➢ control the releases and changes,
- ➢ record and report status,
- ➢ validate completeness.

In the course of development, it can be very dangerous to change or update any past code because that code may affect more parts of the software than expected. In order to avoid these risky situations and confusions in the project, it is very important to analyze the changes before they are made. By the software configuration management strategies, it is possible to improve the quality of the product, reduce the bugs in the product and spend less time for the project.

CMP contributes to the development of the projects in the following manners:

- ➢ Many developers can work on the same project together by reporting.
- ➢ In order to concurrent changes do not conflict with each other, coordination of access to the configuration items is managed.
- ➢ Developers can access to the stable and all previous versions of the project.
- ➢ Changes are controlled and managed effectively to realize the only necessary and sufficient changes.

1.2 The Scope of the Configuration Management Plan

This document describes the Configuration Management strategies of the ResolveSOFT that is going to be followed during the development of the DEVEMB project. This document mainly focuses on configuration identification, change control, status accounting, organization and responsibilities. Thus, this document determines the responsibilities and authorities for accomplishing the planned activities, details of the items under configuration management process and necessary coordination of SCM activities with the other activities in the project.

1.3 Definitions and Acronyms

- **CVS (Concurrent Version System):** CVS is one of the version control system in which changes done on the project can be kept properly.
- **CM (Configuration Management):** CM is used to identify, control and handle the versions of all related parts of the project. It is required to guarantee that all changes are recorded and handled correctly.
- **CMP (Configuration Management Plan):** CMP is prepared to form a written and accessible form of determined rules for CM processes.
- **SCM (Software Configuration Management):** The way of identifying system configurations at specified points. It is also a way for controlling the changes done on the system during the development process.
- **SCR (Software Change Request):** The way of reporting and handling change suggestions to the system.

1.4 References

- IEEE Standard for Software Configuration Management Plans (IEEE Std 828-1998)
- http://www.rspa.com/docs/

2. SCM TASKS

Development of software is a long and continuous process and it passes lots of phases until it finishes. In order to maintain the development of the project, a well designed procedure is needed. In this section, the strategies to hold control of the project as it is exposed to changes are going to be explained.

2.1 Identifying Configuration Items

2.1.1 Identification

Configuration items that are determined for our project are as follows:

- **Specifications and Design Data:** Previous semester, we have designed our project and we determined specifications of our project as well as we can do. However, after starting to the implementation of the project some failures can be realized from a project member or a new function is wanted to be added to our product. If this possibility is come to true, after group decision about this topic a new version is released (if it is a big problem).
- **Source Code:** For this part of configuration process, we have determined the classes of our project that we defined before. Source Code CIs, that are determined for our project, are as follows:
    - Simulator Class
    - StimuliPipe Class
    - Breakpoint Class
    - PICMemory Class
    - SimulationConfiguration Class
    - StimuliRecorder Class
    - StimuliFile Class
    - StimuliGenerator Class
    - CompilerWrapper Class
    - FileManager Class
    - Project Class
    - Controller Class
    - WatchWindow Class
    - Loader Class
    - Thread Class
    - HexFileCoDec Class
    - SimulatorManager Class
    - EditorComponent Class
    - SimulatorGUI Class
    - MemoryContetntDisplay Class
    - ProjectWindow Class

These are our all classes that constitute our project according to our design. As I said above, we determined these classes as our Source Code CIs. Each class must have a source (cpp) and header (hpp) files. However, some large classes, such as Simulator Class, can be divided into more than these two files to facilitate editing and compiling. For example, there may be more than one cpp, include or header files.

- **Software and Tool:** This includes the development environment, compiler, libraries, tools, etc. The followings are the Software and Tool CIs:
  - Dev – C++
  - Win32++
  - FLTK (Fast Light Toolkit)

We are going to make use of the latest and stable versions of these. Changes about these software and tools can only be achieved after revision and acceptance of all the group members especially Fatih DOĞU.

- **Documentation:** For this part of identification process, Documentation CIs that are determined for our project are as follows:
  - Internal Documentation
  - Developer Manual
  - User Manual

While we are implementing our project, we are trying to adding comments in our codes to provide ease of understanding our codes by other project members or other developers who want to see our project code. Then at the end of the term we are going to prepare a developer manual. For this purpose, we are going to use Doxygen which is a documentation generator for C, C++, etc. Doxygen uses comments in the source code to extracts documentation. It can generate output in HTML, RTF, PDF, LaTeX or PostScript file formats.

2.1.2 Baselines

Baselines are composed of CIs at a specific time in our project development. The baselines are used to control the changes that made on the CIs throughout the project process. As a group, we work on the determination of baselines. These created baselines are going to be tracked, audited, retained and version controlled in the CVS. There are three different baselines that are determined by ResolveSOFT. They are the followings:

2.1.2.1 First Development Snapshot Base Line

Implementations of SimulatorGUI Class, StimuliPipe Class, PICMemory Class, SimulationConfiguration Class, StimuliRecorder Class, StimuliFile Class, StimuliGenerator Class, Thread Class, SimulatorManager Class, EditorComponent Class and MemoryContetntDisplay Class.

2.1.2.2 First Release Base Line

Implementations of Simulator Class, Breakpoint Class, CompilerWrapper Class, FileManager Class, Project Class, HexFileCoDec Class, Loader Class, Controler Class, WatchWindow Class and ProjectWindow Class.

2.1.2.3 Final Package Base Line

Completion of the Internal Documentation, Developer Manual, User Manual and all classes

2.1.3 Naming Configuration Items

Software versioning is the process of giving unique version numbers to the unique states of software. In a version number group, numbers are assigned increasing order and corresponds a new progression in the software.

Although there is a variety of version numbering schemes, we are going to use the most common software version format in which different major releases of the software each is given a single numerical identifier. This software version format is typically expressed with three numbers:

major.minor[.revision[.build]]

The first released version of our product has version 1.0. Numbers that are below 1 means *alpha* or *beta* versions, thus versions that are not stable enough for general or practical deployment.

2.1.4 Acquiring Configuration Items

CVS is going to be used for the configuration control and maintaining the changes of the project configuration items. For each of our Source Code CIs, that are our classes mentioned above, a folder is going to be opened and this folder contains at least a source code file (cpp file) and a header file (hpp file); they sometimes contain more than these files especially for large classes. In order to provide maintaining the changes and configuration control, we are going to keep our documentations in CVS. Like Source Code CIs, for each of our Developer Manual and User Manual we are going to open a new folder. Their formats are going to be html to provide compatibility.

Changes are automatically assigned a change number. This number is stored in an associated archive along with the change. A CI catalog shall be created and placed under CVS server. It is going to be updated whenever a new CI or version is established.

Moreover, all the documents and drafts are going to be kept in all our project members' computers and project group mail account that can be accessed by all of the group members. After the stabilization of the document, it is going to be published on our group's web site. If any changes are required, it is going to be followed the configuration control process.

2.2 Configuration Control

2.2.1 Change

The primary method of communication between the members of the team is e-mail. Any change requests will be first proposed to the team using an email. The properties of the email will be:

- ➢ The email subject should start with "Change Request" and continue with the module (source file) name the change is proposed to.
- ➢ The mail content should clearly include the change description, along with the reason and exact place in the code repository the change is requested.
- ➢ The main responsibility of the change is attributed to the main handler of that module (class), which is defined in the living schedule.
- ➢ The changes will be discussed in the mail group first. If the change request is a major one it will be discussed in the following meeting. Since we live in the same dormitory, we will meet at least twice at weekends and one during the weekly meeting with the assistant. Also, it is possible to communicate using our room phones.

The change proposals will be discussed, voted and implemented if accepted. If inequality happens in the voting the final decision will be made by the CMB director.

The changes are going to be recorded in the source file header. The change number and the author name will be included, along with the date the change was performed.

2.2.2 Bugs

In the case of finding a bug, a different kind of methodology will be followed. The reporting e-mail will be similar to the change request with the following modifications:

- The email subject should start with "Bug" and continue with the module name and if known the class name.
- The mail content should include the inputs that have caused the malfunction or the environment (test-bed) and include the stack trace if an exception is thrown.
- The main responsibility of the bug is assigned to the main handler of that module, which is defined in the living schedule. In case the bug is a trivial one the person who has found the bug may wish to fix it and send an acknowledgment to the team.

2.3 Version Control

2.3.1 Importance of Using Version Control

Using a version control system is an absolute must for developers of a project above a few hundred lines of code, and even more so for projects involving the collaboration of several developers. Using a good version control system is certainly better than the ad-hoc methods some developers use to maintain various revisions of their code.

2.3.2 How Different Versions of the Software Will Be Controlled

Version control is a special kind of software used to track and manage changes. In our case, (like the other groups) CVS version control is used to track any sort of change made to our project, that a series of adjustments to our project where several files, folders, and modules are added to (or removed from) the software.

In our DEVEMB project there are several modules that we have to implement under the guidance of progress schedule. Every member has its own responsibility in different classes.

In an uncontrolled project where multiple authors have access to edit and contribute, the potential for conflict and problems arises so when these authors work from different places at different times of day and night. You may spend the day improving the simulation module for example.

After one has made his changes, another developer who works different place after hours, may spend the night uploading their own newly revised version of the simulator manager class , completely overwriting the others work with no way to get it back. With the module under CVS version control, the late–night author will be alerted to a conflict with simulator module, presented with the exact parts of the file that are causing a problem, and asked to adjust their work to incorporate anything you added and committed to the project while working on it earlier in the day.

CVS allows several people to alter the same module, even the same file, at once. We are going to update our working copy of our project regularly so that members will never miss a major change.

2.3.3 Possible Problems and Solutions

Some problems may arise while working concurrently with the other members, so these kinds of problems may slow down the process. To avoid such problems during version updates we will use some methods.

The simplest method that we can use of preventing concurrent access problems is to lock files so that only one developer at a time has write access to the central "repository" copies of those files. Once one developer "checks out" a file, others can read that file, but no one else is allowed to change that file until that developer "checks in" the updated version (or cancels the checkout).

The other problem may arise from the insufficiency of the implementation methods that developer use. The methods and the progress policy are defined specifically in design report but when come to implementation, some modules may lose their functionality. To prevent this, although every group member is responsible for his own job, the entire executable versions must be tested occasionally. If the members encounter any problems with the working system of modules, they have to report the problem and the developer who is responsible for that problem takes control of the situation.

Working under guidance of a progress schedule may arise some problems. Although the implementation process is being described briefly, members' individual problems can cause trouble. As it is known we have to present our work to the assistant who is responsible, designing our own schedule with respect to progress schedule will be the best solution.

2.4 Configuration Status Accounting

- What data elements are to be tracked and reported for baselines and changes;
- What types of status accounting reports are to be generated and their frequency;
- How information is to be collected, stored, processed, and reported;
- How access to the status data is to be controlled.

In order to fulfill configuration status accounting following data elements shall be tracked:

For each class of the project

- source files
    - file name
    - location in the source tree
    - dependencies
    - names of the methods and functions declared or defined
- class documentation
    - class name
    - properties
        - name
        - attributes (public private etc.)
        - semantics (what do they stand for)

    - methods
        - name & signature
        - attributes (public virtual static inline etc.)
        - semantics (what they accomplish, preconditions postconditions, exceptions raised)

- documentation deliverables
    - User Manual

    - Developer Manual
    - Living Schedule

For each data element mentioned above following reports shall be generated in parallel to the implementation process:

- current version
    - features implemented
    - features awaiting implementation
- revision history
- review status of change requests
- implementation status of approved changes

During the implementation each member updates the implemented features and missing features sections of the **Current Version** document as they implement the sections assigned to them (e.g. in living schedule).

After each revision, revision history is updated with the addition of summary descriptions of the all changes from the last version except the most trivial ones.

A **review status of change requests** entry is created in the **change requests document** by the member responsible for CSA management upon receiving the change request and alerts the members through the mailing list. He also decides how the change decision will be made. The decision processes are not limited to but may include the following methods:

- The member responsible for CSA management decides on his own accord that the request has already been decided upon during the design process.
- He contacts all the members that will be affected by the requested change should it be implemented.
- He may ask the requester to obtain approval of all the members that will be affected by the requested change should it be implemented.
- May summon a group meeting (possibly online) should the change request be toot substantial and fundamental to be handled by other means or it is not clear what change will address the issues raised by the member who made the change request.

An **implementation status of approved changes** entry is created in the **approved changes** document by the member responsible for CSA management if a consensus is reached by the group pertaining to a change request and alerts the members through the mailing list. He informs the group who is responsible for the implementation of the change and when the change is due.

The implementation status is updated *ad hoc* as:

- The member responsible for the implementation informs the member responsible for CSA management as the implementation progresses or is completed.
- The member responsible for CSA management inquires the status of the implementation from the party responsible, possibly upon request by other members.

2.5 Audits and Reviews

We have three control points for the program configuration items: namely development snapshot, first release, and final package release. Final package release is also a control point for documentation. We will have a review and an audit for each control point. The review roughly half way between the control points and the audit a week before the release control point.

All members shall participate in reviews and audits.

2.5.1 Description

In each first review we will establish what will be delivered when we reach the control point. The capabilities and features to be delivered will be put on paper, and assignment of responsibilities for each configuration item will be finalized. How much of the configuration items to be delivered have been finished will also be determined and recorded.

In each audit shall be conducted by the group members, as we were not able to procure any outside help to compensate for the absence of customer/stakeholder feedback. In these audits we will discuss, determine and record how much of the goals set in the previous review have been achieved.

The configuration items to be included in these reviews shall be determined by the member responsible for identification as described in the respective section of this document.

1st audit: Date: 13 March

Agenda: Pertaining to the establishment of a baseline for 1st Control Point (Development Snapshot).

The configuration items to be included:

Simulator/Debugger Module:

- SimulatorManager Class
- Simulator Class
- PICMemory Class
- SimulatorGUI Class
- MemoryContentDisplay Class

Compiler Module:

- CompilerWrapper Class

File/Project Module:

- FileManager Class
- EditorComponent Class

Utility Classes:

- Thread Class
- Controller Class
- HexFileCoDec Class

1st review: Date: April 9

Agenda: Pertaining to the determination of the requirements for the baseline for the 2nd Control Point (First Release) and their current status of completion.

The configuration Items to be included:

Simulator/Debugger Module:

- SimulatorManager Class
- Simulator Class
- PICMemory Class
- SimulatorGUI Class
- SimulationConfiguration Class
- StimuliFile Class
- StimuliPipe Class
- StimuliRecorder Class
- StimuliGenerator Class
- MemoryContentDisplay Class
- WatchWindow Class
- Breakpoint Class

Compiler Module:

- CompilerWrapper Class

File/Project Module:

- FileManager Class
- EditorComponent Class
- ProjectWindow Class
- Project Class

Utility Classes:

- Thread Class
- Controller Class
- HexFileCoDec Class

2nd audit: Date: April 30

Agenda: Pertaining to the establishment of a baseline for the 2nd Control Point (First Release).

The configuration Items to be included:

Same as 1st review

2nd review: Date: 4 June

Agenda: Pertaining to the determination of the requirements for the baseline for the 3rd Control Point (Final Release) and their current status of completion.

The configuration Items to be included:

- All Classes
- User Manual
- Developer's Manual

3rd audit: Date: 4 June

Agenda: Pertaining to the establishment of a baseline for the 3rd Control Point (Final Release).

The configuration items to be included:

- All Classes
- User Manual
- Developer's Manual

2.6 Data Collection and Evaluation

As was discussed in the section 2.4, three kinds of change data are collected in order to use for evaluation.

- Change requests
  - mainly received through mailing list.
  - will be stored in an HTML file.
  - the HTML file will be updated by the member responsible for the software configuration accounting
  - the HTML file will be located in a directory in the CVS repository
  - it consists of a **numbered list of** entries
  - each entry in the file will have fields for the following:

    - who created the entry
    - when was the entry made
    - who made the change request
    - when the change request was made
    - to which module files does the change request pertain
    - short description of the change request
    - who has been assigned the task
    - current status

  - fields for which the necessary information is not available will be left empty
  - the fields will be filled as the information necessary becomes available.
  - the member responsible from the web page checks out the page from the CVS repository and makes it available for the members to view
- Approved requests
  - mainly received through mailing list.
  - will be stored in an HTML file.

    - the HTML file will be updated by the member responsible for the implementation of the approved change or the member responsible for SCA
    - the HTML file will be located in a directory in the CVS repository
    - it consists of a **numbered list of** entries
    - each entry in the file will have fields for the following:
    - who created the entry
      - when was the entry made
      - what is the number of the request in the change request document(see above)
      - who was assigned the implementation of the change
      - when was the assignment made
      - a list of entries for each update in the status of the change.

    - fields for which the necessary information is not available will be left empty
    - the fields will be filled as the information necessary becomes available.

    - the member responsible from the web page checks out the page from the CVS repository and makes it available for the members to view

- change information stored in the source

3. SCM Resources

ResolveSOFT software development team has 4 members. Each member is responsible for CM. Obviously, it is crucial to store the old versions of source codes for the team. In order to keep previous versions CVS is a suitable system and is provided by our department. All kind of team information, documents about the projects and new versions of the project will be reachable from the web site of ResolveSOFT. The living schedule that is shown in our web page will be updated accordingly by the group members as the new tasks are accomplished.

4. Quality Assurance Overview

Resolvesoft has put together a comprehensive portfolio of QA and testing services that are designed to ensure the reliability of Devemb. These services effectively address the functionality, usability and performance testing aspects of the product. We can assist in the entire product development life cycle – including requirements phase, architecting the test strategy, tools selection, code reviews, building the test suites, test execution to final builds and product releases.