Middle East Technical University

Department of Computer Engineering

-TURKUAZ PROJECT-

RadeX

TIRAN SOFTWARE CONFIGURATION MANAGEMENT PLAN

Tahir Bilal	*1394741*
Onur Deniz	*1391002*
Soner Kara	*1395110*
Mert Karadağlı	*1395128*

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PURPOSE OF CMP

The discipline of Software Engineering tries to develop a systematic approach to analysis, design, implementation and maintenance of software. Configuration Management is the subfield of SE that deals with maintenance of software development and it also demonstrates some heuristic directives in order to cope with *change* in software development.

Since no software development process is immune to *change*, developing a configuration management plan is crucial for every software developer team. In case an unforeseen change occurs during the development, it should be easy to integrate the changing part with the other software configuration items (like code, data). If the methodology to be used in case of a *change* in development is not thoroughly planned and documented beforehand, it will not take much time to throw all the development process into chaos. Nobody wants to develop software in a completely chaotic environment in that nothing can be foreseen; we neither, so we developed this configuration management report in which we briefly explain our strategies to handle the *changes* that we may come across during the development process of our project. We are confident that it will boost our performance, and make us feel much more content about the *change*s that will occur this semester. It feel give us the comfort of having a plan at hand that we can always consult to.

1.2. SCOPE OF THE DOCUMENT

This report documents the configuration management plan of the softare package Radex that is currently being developed by Tiran Software.

This plan defines what will be put under configuration management control, who will be involved in the configuration management process, what procedures are to be used, and what timeline has been planned for these activities.

1.3. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

CI	Configuration Item
CM	Configuration Management
CVS	Concurrent Versions System
SCA	Software Configuration Audit
SE	Software Engineering
CCB	Configuration Control Board
CMG	Configuration Management Group
DT	Developer Team
TT	Testing Team

1.4. DOCUMENT REFERENCES

1. Software Engineering A Practitioner's Approach, 5th Edition , **Roger S. Pressman**
2. IEEE Standard for Software Configuration Management Plans (IEEE Std 828-1998)
3. Schaum's Outlines Software Engineering, **David Gustafson**

2. THE ORGANIZATIONS CM FRAMEWORK

2.1. ORGANIZATION:

Since we are working for an eight month project, we need a properly planned hierarchy between organizational units which handle CM activities.

CM activities are handled by the following groups;

- Configuration Control Board:
  This group is responsible for handling CM activities. It supervises all the other groups.
  Its duties are:
  - Reviewing SCRs offered by customers, developer and testers.
  - Accepting or denying SCRs according to review results.
  - Holding audit.

- Configuration Management Group:
  This group has the following responsibilities;

    - Creates and maintains CMPs.
    - Coordinates and reports CM activities.
    - Updates CM schedule.

- Developer Team:
  This group is responsible for implementing CM activities assigned by CMG.

- Testing Team:
    - This group evaluates the source code according to requirements.
        - According to test result, testing team may generate a SCR.
    - Also, planning test cases is another duty for Testing Team.

2.2. RESPONSIBILITIES:

Since RadeX is developed by four people, all of these four people are in CCB and also CMG. Therefore all of the members are responsible for duties of CCB and CMG.

All members should complete the assigned CM activity on time.

When a member changes-modifies a document or source code, he will add related comments to CVS via Eclipse in predefined format .He will also inform the other members about the change using mail group.

2.3. TOOLS AND INFRASTRUCTURE

Tiran Software uses the following tools throughout the development of RadeX:

2.3.1. CVS

During the project, we will need concurrent accesses to project sources and files. In order to handle disambiguation stemming from concurrent accesses, we need a version control system allowing concurrent access. Because of this reason we will use CVS, The Concurrent Versions System as our version control system. CVS allows different branches of a project. Moreover, it holds log files of any change in file system and takes back-ups of older version files. In case of a mistake/bug we will be able to obtain an older but working version using these backups. CVS will be provided by our department; therefore it does not take our time to set up a version control system. These reasons make us to choose CVS as version control system in our project.

2.3.2. ECLIPSE

Since we implement our project in Java, we choose Eclipse as our development environment. Also, eclipse provides a complete and easy CVS support, which facilities our CVS usage substantially.

3. THE CONFIGURATION MANAGEMENT PROCESS

3.1. CONFIGURATION IDENTIFICATION

3.1.1. SOURCE CODE CI

Source Code CI consists of all the implemented modules of the RadeX project. Organizing the source code in a consistent manner will be established using CVS. CVS will also help storing, protecting and retrieving the code.

Design reports are our main resource in organizing the code. We split the project into packages; packages into classes and each class have their own methods for encapsulation. Class and method names are chosen to be meaningful. An example for a method name is given below:

..\src\arac.Preprocessor.yuklemBul()

Here, "src" is the folder containing all the source code. "arac" stands for some package name and "Preprocessor" is one of the classes in "arac" package. Finally, "yuklemBul()" is one of the methods in "Preprocessor" class.

3.1.2. DATA CI

Data CI consists of all the data items used by the program. This includes our lexicons stored in files or via DB4O, extracted information residing in database tables via PostgreSQL. Moreover we are planning to tag some reports for use in machine learning. This tagged corpus should also be considered as an important data item.

3.1.3. DOCUMENT CI

Today, well-structured, clear and understandable documents are inevitable for high-quality software. We are going to provide user manuals and installation manuals with the released product. Moreover, all the reports that we are preparing during the development are also included as document CIs, namely, project proposal, requirements analysis report, design reports, test specifications and living schedules. All documents can be retrieved from the web site of Tiran Software.

Documentation of the RadeX project will be partitioned between the members. One will be responsible for documenting that part of the code, which he implemented. We are going to follow some naming conventions for documents.

The format for the living schedules will be living_schedule_DD-MM-YY.jpg.

Other documents will be named as RadeX_versionNo_documentName.pdf.

3.2. CONFIGURATION MANAGEMENT AND CONTROL

3.2.1. CHANGE REQUESTS

During the development of the RadeX any member of the team can request any change any time. Change request reports should be prepared only if the change is major and/or it can affect other classes or modules. The change can be functional, algorithmic, optimization or even re-design issue.

Change request should be prepared in a structured format and send to yahoo groups so that all the members see and read it. Below is the general format for change request report:

- Date of request
- File name of the source code
- Line number(s) of the code
- Reason for the change
- Proposed changes

Minor changes should not be reported formally, but the can be explained verbally or can be stated informally via yahoo groups. But, it's more important to add CVS logs about what is changed and why the change has been done.

3.2.2. EVALUATION OF THE REQUEST

At each weekly meeting we discuss the previous weeks change requests, if any. In the evaluation, change reasons and the possible effects of changes are discussed. The member responsible for the part of the code which is subject to change will give his opinions and it will be compared with the proposed change in the request. Evaluation order of the request will be decided by the team leader according to their urgency levels.

3.2.3. IMPLEMENTATION OF THE CHANGE

Before the implementation can start, the change request should be accepted by the majority of the members. If accepted, the one responsible with that code part will start to implement the changes. Also, it's possible that the member proposing the change request to implement his own request. After the implementation has been done, change request report should be moved to "finished requests" folder.

3.3. CONFIGURATION STATUS ACCOUNTING

Since our project, Turkuaz, is relatively small in terms of code and data compared to other commercial products, we are not going to prepare change request, build, defect and release reports. Instead, we will conduct some informal methodology to account configuration status.

Since we are only four people sharing the code, it's enough to use CVS logs and yahoo group for sharing and stating the current status, new changes and bugs. To speed up testing & debugging process, it's important to find out existing bugs effectively. For this purpose we are planning to allocate some place for sending only bugs at yahoo groups in a consistent format.

Finally, our project leader time to time writes unofficial progress and work division reports, so that all the members read and discuss the current state of the project.

3.4. CONFIGURATION AUDITS

In order to release a high-quality final product, we are going to consider the following audit items carefully.

At each development snapshot builds we will check how close or how far we are to the requirements of the projects. Also necessary unit and stress tests will be conducted to achieve a robust and fully-functional product.

Regular meetings at each week help us for maintaining the process alive and we have the chance of discussing necessary changes regarding to design or implementation. The team member proposing the idea will provide documentation and verbal explanations.

4. PROJECT SCHEDULES – CM MİLESTONES

4.1. PROJECT SCHEDULE

We have already prepared a living schedule which consists of tasks and their deadlines. We have divided the development of the whole software package into development of basic components. Next we shared the development of those components among us. The most crucial component is 'template matching' component which is developed by one of our members. The Graphical User Interface component and the Information Retrieval component(previously called searcher) is under development. Another member is still working on the enhancement of the preprocessor. For the first release all of these components will be finished. Our program will be able to find relations by template matching, it will have a nice GUI and improved database searching capabilities. However, in the first build we assume that it will solely do text mining on some limited kind of reports like mamography.

After the first build we will split all the reports int 3 by their captions. All 3 member swill start to write templates for his own captions. Meanwhile another member will start to tag the reports by the sense features that we developed in the final design report. In the second build of our software package RadeX will be able to do text mining in all kinds of reports. After the second build deadline, we will have a tagged corpus with sense features, so will try to integrate ML algorithms for named entity recognition and partial chunking. We hope that this phase will boost the performance of template matching. For the final build we will try to improve all the things we made that far, template matching engine will be made faster. The preprocessor and the database searcher will be finalized. Lastly graphical user interface will be beatifulized.

4.2. CM MILESTONES

The following are the CM milestones:

• first build with template matcher	7.3.2008
• second build	27.3.2008
• third build	24.4.2008
• documentation	27.05.2008
• final version	13.6.2008

5. PROJECT RESOURCES

In order to perform our CMP and CM activities, we will use CVS client in Eclipse. We keep online resources other than source codes such as documents and living schedules in our web-site.

6. PLAN OPTIMIZATION

During the development of our project, we will obey the schedule plan as much as possible. But, since software programs are wicked and always new problems arise, it's quite impossible to follow the current plan strictly.

If any time there is a need to make some change in CMP, team members should get together immediately, discuss the change and put the new plan in action. Tuesday meetings are important for the members to be kept informed about new changes and future plans.

Besides weekly meetings, since all the members are responsible for following the mail groups, it's quite easy to discuss new plans and optimizations any time during the week. This will prevent the development to be delayed and will help using our precious time more efficiently. Examining living schedules is also important for us to see our progress and do necessary changes in case of a problem.

Finally we always support changes in development process and we think these changes and optimizations make the software better.