Middle East Technical University
Department of Computer Engineering

CONFIGURATION MANAGEMENT PLAN

ÖZGÜR YAZILIM

Özgür Özgür
Fırat Erdoğan
Onur Demircan
Abdulkerim Mızrak
Mehmet Emin Ulusoy

1. Introduction

1.1 Purpose of CMP

The CM plan is a vital component of a software project in terms of management and development issues. This applies our project too.

The development process is not a straightforward issue. Throughout the development, the project may expand or shrink, and certainly there shall be some problems about different issues. At this point it becomes inevitable to have a good plan and organization to manage the changing conditions and to fix the problems.

1.2 Scope of Document

This document is prepared by the members of Özgür Yazılım project group and introduces the Software Configuration Management Plan for the game Treasure Hunt.

This document involves the key issues about the standards of the Treasure Hunt project, and also introduces how to act when problems occur or some modifications are needed.

Every member of the team shall behave in a way described in this plan and responsible from the overall mechanism of the project.

1.3 Definitions, Acronyms and Abbreviations

ACRONYMS	DEFINITION
CM	Configuration Management
CMP	Configuration Management Plan
SCMP	Software Configuration Management Plan
CVS	Concurrent Versions System
MMOG	Massively Multiplayer Online Game
AI	Artificial Intelligence
NPC	Non-personal Character

1.4 Document References

While preparing this plan, we made use of:

- Our Requirement Analysis Report
- Our Final Design Report

- SCMP Guideline: IEEE Standard for Software Configuration Management Plans

1.5 Document Overview

Being a 3D MMOG, our project consists of multiple modules involving network and graphics components. This results in problems of integrating different modules throughout the development process and prior to the release of new versions.

During the development phase of our game, modules should be working consistently, and there should not be any conflict between them. To achieve this, in addition to a good plan and organization, there should be some criteria about the design and implementation issues. Those criteria involve key issues like the tools to be used, the standards in coding, and the action plan when problems or changes occur.

This document introduces those criteria and will act as a guide throughout our progress.

2. The Organizations CM Framework

2.1 Organization

Our team consists of five people.

Having a democratic-decentralized organization, there is not a permanent leader in our group, but coordinators. The coordinators are also changeable according to the type of task to be done or the change in the task. A good communication between group members is essential in this model, since every individual must have a complete understanding of how things are to be handled. The main and most important thing about this type of organization is making decisions by a group consensus. The negative aspect of this organization is the difficulty of reaching a consensus in some situations.

2.2 Responsibilities

Having different modules to implement, division of labor is inevitable in our group. This is achieved by dividing the job to be done mainly into two parts, which are network and graphics. However, there is not a clear-cut about the responsibilities and team members may sometimes work on the same issue to meet the deadlines. This is a possible situation and it is impossible to overcome this problem without having standards by means of implementation and tools to be used.

Each group member should be active at the time of determination of the deadlines. Then each member is responsible from his task and should do his best to meet the deadlines.

The group, as a whole, is also responsible from the overall architecture and organization of the project, and also motivation of the team.

2.3 Tools & Infrastructure

- Our game will run on Microsoft Windows XP Operating System.
- We use C++ to develop our project. The development environment we use is Microsoft Visual Studio.NET 2003.
- In terms of graphics we make use of the OpenGL libraries.
- To create 3D objects, textures, and the game environment, we make use of 3D Studio Max and Adobe Photoshop.
- The Concurrent Version System (aka CVS) is another tool we use. Since we are developing our project as a group, it becomes inevitable to use a tool like CVS in order to make the sources accessible by every member and to keep different versions of our program.
- For documentation purposes we make use of MS Word, Visio, and Smart Draw.

3. The CM Process

3.1 Identification

We have divided our project into units of configuration that can be individually managed and versioned. These configuration items can be examined mainly in 3 parts: Engines & modules, game data, and documents.

3.1.1 Engines & Modules:

- *GameEngine:* Core of the game engine.
  Server Side Game Engine
  Client Side Game Engine

- *Network Engine:* The interaction between the clients will be done by this module by a server.

- *Input Module:* Input handling will be done.

- *Artificial Intelligence Engine Module:* NPC control and AI engine.

- *Menu Module:* Graphical User Interface management

- *Graphics Engine:* handle all of the rendering operations during the game according to user input.
  Characters
  Map
  Models
  Physics
  Camera

- *Audio Module:* The Audio Module will hold the path of the selected sound file and load the file to memory when desired
  Sounds

- *Chat Module:* Chat module will enable players to communicate with others in real time.

3.1.2 Game Data:

- Map
- Model
- Character Models
- Character Animations
- Texture
- Music
- Puzzles
- SoundTracks

3.1.3 Documents:

- Design Report
- Configuration Management Plan
- Installation Manual
- User Manual

In order to assign unique identifiers to each item to be controlled, we have specified an identification system. The identification system will work in a way like this: the prefix of each sub-module name is the module that it belongs to. For example, Server Side Game Engine is a sub-module of Game Engine and the initialization of the game is a sub-module in the Server Side Game Engine. We name initialization as: *gameengine.serversidegameengine.initilization*. As can be seen; all letters are small.

Since we have a modular system, we have the flexibility of working independently. For the development and implementation phase we will define baselines that will help us for the milestones. CVS server will be used to physically define these baselines.

3.2. Management and Control

In this section we will introduce the process of the possible update and change operations during the semester. This update process can be analyzed in four phases: Change Request, Approving or Disapproving Requested Changes, Implementing Changes, and Version Control.

3.2.1. Change Request

Our group consists of two sub-groups. One is network sub-group, the other is graphics sub-group. Any change request can be made by anybody in the group, but firstly the sub-group will evaluate this request, if they accept the request, the other sub-group will be informed too.

3.2.2. Evaluation

Firstly the sub-group will evaluate the request. If they accepted the request they will inform the other group also. The sub-group will have to implement a small demo that shows the request in practice. After that the other group will explain their opinion about the request.

3.2.3. Approving or Disapproving Requested Changes

The request that has been negotiated by the group will be announced on the web page of our group. If rejected, the reason will be announced also. If the change is accepted, the one who proposed the modification will start implementation.

3.2.4. Implementing Changes

First, the baseline configuration item of project is checked out and changes are applied. Next, the updated item is tested for quality assurance. Then the configuration auditing is performed and finally the new version of the software is saved and every group member is informed about the modification.

3.2.5. Version Control

Version numbers must be used in order to follow our project development stages. As we develop our project we give the most final version a number to identify it. Our version numbering system will be like this;

<major change>.<minor change>

If there is a major change then we increase the number in the major change field by one. If the change is minor then we increase the number in the minor change field by one.

Our project's version control will be done by CVS. Every update of the project will be saved in CVS and its version control functionality will maintain a history of all updates.

Every group member has his own CVS account and a copy of the source tree. After making some modifications the member can add or remove the files and by using CVS commit he merges his changes back into the repository. So that the modified version can be seen by other group members using CVS. If the group member wants to merge updates committed by other group members into is working directory then he simply uses CVS update. If there are uncommitted changes CVS attempts to merge the changes from the repository with the changes in the working directory. If the merge fails then CVS indicates a conflict and tries to correct it manuall

3.3. Configuration Status Accounting

In the CVS we will have four main directories: Graphics directory, Audio directory, Puzzle directory, Codes directory.

In the graphics directory there will be the models, meshes, 3d objects, textures and maps.

In the audio directory there will be music and sound effects that will be used in the game.

In the puzzle directory there will be puzzle database.

In the codes directory there will be all the files: cpp files, header files.

In each of these directories there will be a Change_X_.txt file that will contain this information:

- Detailed description of the change: The change made will be explained. Why this change is needed and how it is made will be recorded.
- Analysis of the change: After the change, what changed in the system and how th e system works after the change.
- Names of people making the change: The names of the group members who made the changes.
- Title, type, release and version, with date: The title, type of the changes and the re lease and version numbers of the system after the changes.

The Change_X_.txt will be saved as Change_(X+1)_.txt to avoid loosing the previous change information.

3.4 Configuration Auditing

Auditing is the act of evaluating an organization, system, process, project or product. Because of being a democratic-decentralized team we don't have a CM manager. In our project the auditing is held in two ways: By weekly meeting, and by requesting.

There are meetings each week where the members explain any changes or any problems by informing the other members about his weekly progress. During audits, the baseline configuration and current configuration is compared. According to that the team either verifies that any item in the plan is achieved or not. By the way, if there are any questions, the other members try to answer that question of that member. At the end of each audit decisions are taken about the project plan.

There may be audit requests from any member of the team. Auditing sessions will be held after release of each version and every important progress. During auditing sessions, it will be checked whether the correct changes are made, the changes are correctly implemented, any additional changes are needed, and whether the final release is in compliance with the baseline and requirements. Since the modules are well-defined, every team member will test his modules independently. Also there will be continually testing and according to these results, deficiencies will be noted in the audit report, and coder(s) of the module are expected to fix the deficiency. Auditing of the data can be done separately by any member of the group by informing the code owner and the other members via e-mail.

4. Project Schedule

All the tasks have been divided while we were four people but this semester one more person joins to us. Now we will show all the milestones
that we have designed previously. The new-comer friend will be ready in a few weeks, after that we will reschedule the milestones and
give some task to him. The project milestones are available in the appendix part.

Our schedule will be put on the web and each week this schedule will be updated.

5. Project Resources

Özgür Yazilim was consisting of four members. By this term a new member is added to the. Team members need some tools to carry out the CM operations, changes, development, communication and documentation. The main tool for CM activities is CVS supplied by the department. When a modification is finished; the latest version will be updated by CVS. Also any team member will access to test version of the project. Project members can work on same files simultaneously by using CVS. CVS notify when any conflicts occur.

For documentation Özgür Yazilim team members will use MS Word, Visio and Smart Draw.

Özgür Yazılım members communicate with each other by e-mail, newsgroup and forum on our web site. (http://senior.ceng.metu.edu.tr/2007/ozguryazilim/)

We will use FTP and Web Server for location and updating project web site.

6. Plan Optimization

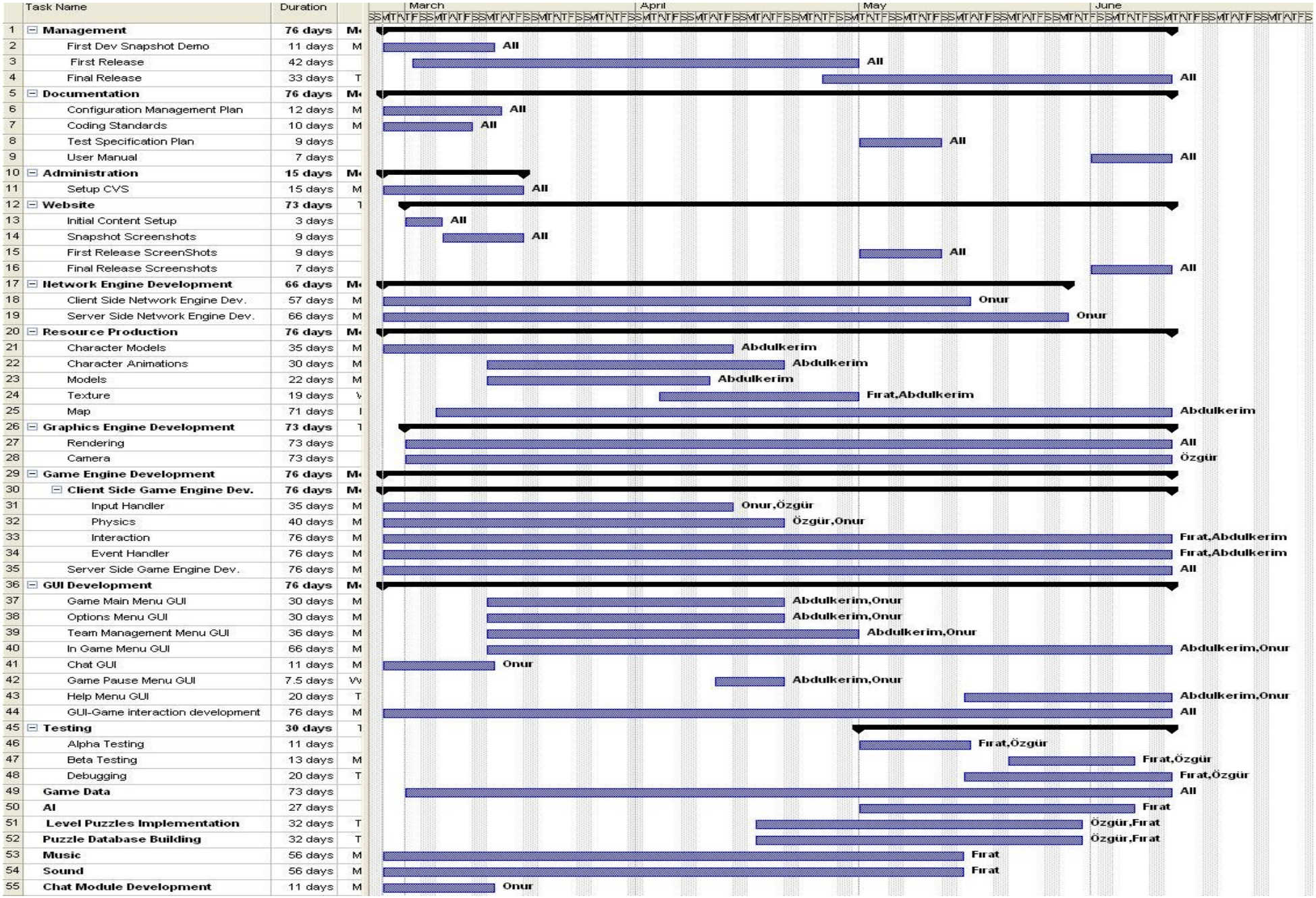
Özgür Yazilim members consider the CMP as a mainframe for the project development process. Our progress will be compatible with CM. Nevertheless we are aware of possible changes and optimizations in our CMP.

Throughout the project development, we will need optimization of CMP. Considering our weekly meetings and continuous communication with various tools, we will update our plan continuously. After every milestone, we must prevent our CMP becoming out-of-date.

All project members are responsible all the project process cycle and also take care of maintaining of CMP. When we need changes we will discuss the necessary optimizations and we will keep our plan contemporary. Using CVS and communicating with other group members continually is very important to keep our plan up-to-date. By using CVS in the development process any changes in the project can be informed to other group members, and if necessary, new changes can be applied to the CMP.

Changes will also be made in case of dropping behind our schedule.

APPENDIX


March
April
May
June
Task Name
Duration
1 Management 76 days
2 First Dev Snapshot Demo 11 days
3 First Release 42 days
4 Final Release 33 days
5 Documentation 76 days
6 Configuration Management Plan 12 days
7 Coding Standards 10 days
8 Test Specification Plan 9 days
9 User Manual 7 days
10 Administration 15 days
11 Setup CVS 15 days
12 Website 73 days
13 Initial Content Setup 3 days
14 Snapshot Screenshots 9 days
15 First Release ScreenShots 9 days
16 Final Release Screenshots 7 days
17 Network Engine Development 66 days
18 Client Side Network Engine Dev. 57 days
19 Server Side Network Engine Dev. 66 days
20 Resource Production 76 days
21 Character Models 35 days
22 Character Animations 30 days
23 Models 22 days
24 Texture 19 days
25 Map 71 days
26 Graphics Engine Development 73 days
27 Rendering 73 days
28 Camera 73 days
29 Game Engine Development 76 days
30 Client Side Game Engine Dev. 76 days
31 Input Handler 35 days
32 Physics 40 days
33 Interaction 76 days
34 Event Handler 76 days
35 Server Side Game Engine Dev. 76 days
36 GUI Development 76 days
37 Game Main Menu GUI 30 days
38 Options Menu GUI 30 days
39 Team Management Menu GUI 36 days
40 In Game Menu GUI 66 days
41 Chat GUI 11 days
42 Game Pause Menu GUI 7.5 days
43 Help Menu GUI 20 days
44 GUI-Game interaction development 76 days
45 Testing 30 days
46 Alpha Testing 11 days
47 Beta Testing 13 days
48 Debugging 20 days
49 Game Data 73 days
50 AI 27 days
51 Level Puzzles Implementation 32 days
52 Puzzle Database Building 32 days
53 Music 56 days
54 Sound 56 days
55 Chat Module Development 11 days
All
Onur
Abdulkerim
Fırat,Abdulkerim
Özgür
Onur,Özgür
Özgür,Onur
Abdulkerim,Onur
Fırat,Özgür
Özgür,Fırat
Fırat