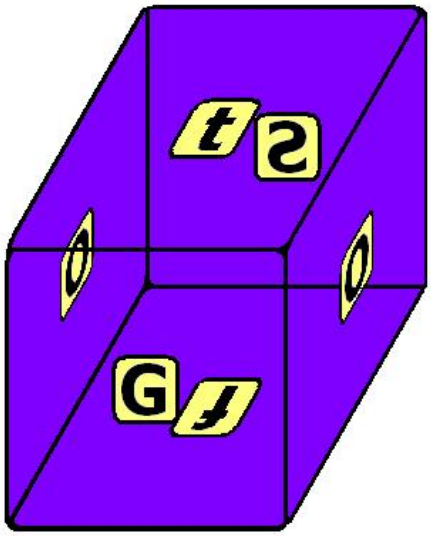
CENG 490
Senior Project

"A 3D – Massively Multiplayer Online Game"
The Ma3e-3D

Configuration Management Plan

by



Ömer Akyüz e1347079
Önder Babur e1347186
Süleyman Cincioğlu e1347277
Güneş Aluç e1462670

Table of Contents

1. Introduction

1.1. Purpose of the Document

The purpose of the document is to set a guideline for all team members of the “Ma3e-3D” project when making any changes/updates on any type of project material and during any stage of project management. Without a doubt, a standard way of handling changes will increase project’s overall productivity and reduce errors due to inconsistent approaches.

1.2. Scope of the Document

The intended audience of this document is mainly members of the GOSOft team, the instructor and the project assistant.

This document is about the Configuration Management(CM) activities of the “Ma3e-3D” project. The SCM activities explained in this document are applicable during the development and maintenance phases of the project.

1.3. Definitions, Acronyms and Abbreviations

CI Configuration Item
CM Configuration Management
CMP Configuration Management Plan
SCM Software Configuration Management
SCMP Software Configuration Management Plan

1.4. Document References

This document is based on the standards/guidelines documented in the following references:

1. “Software Configuration Management”, The presentation prepared in METU Computer Engineering Department for the course CENG492

2. Pressman, Roger S. Software Engineering: A Practitioner's Approach, Sixth edition. New York, NY: McGraw-Hill

3. IEEE Standard for Software Configuration Management Plans (IEEE Std 828-1998)

1.5. Document Overview

The following sections of the document are mainly as follows:

1. The Organization CM Framework: It gives detailed information about the CIs in relation to managerial and technical organization of the project. Furthermore, it matches the responsibilities of the team members with the CIs explained. Finally, the tools and infrastructures that are to be used during the CM are listed.
2. The CM Process
    a. Identification: A guideline on how to identify a particular item as a CI is given. The CIs specific to the project are outlined.
    b. Management and Control: Guidelines are set for the engineering, system management, development, defect tracking, change management, building and deployment tasks that are to be carried out during the project lifecycle.
3. Schedule: Critical dates and milestones are outlined. Project specific approaches concerning the time management are clearly identified.

2. The Organizations CM Framework

2.1. Organization

In our project The Maze3D all the Configuration Management Organization is divided into 2 main categories which are Manegarial Organization and Technical Organization.

2.1.1. Managerial Organization

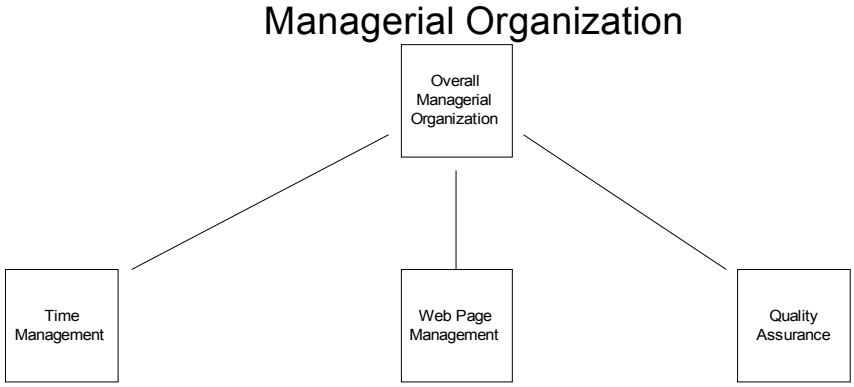


Diagram 1 : Managerial Organization Overview

Managerial Organization [Diagram 1 : Managerial Organization Overview] aims that the project works proper and ensures that all responsibilities will be completed according to the schedule. To accompolish these goals, some categories are created in the managerial organization.

- Overall Project Management: Overall Project Management deals with the overall configuartion and problems of the project. Weekly Meetings are important for this. Every ideas of the team members are discussed in these meetings. More than these Project Leader is also responsible for dealing with the problems of the team members and keeping their motivation allways high.

- Time Management: Time Management ensures that all the work in the project is completed in the given time in the Living Schedule. All the members are responsible for their own time management. To deal with this every week the Living Schedule is updated according to the progress and progress that is left is discussed in the weekly meetings.

- Web Page Management: Web Page is an important aspect in the Configuration Management. All the progress of the project must be seen in the web site. The documents must be updated weekly according to the changes. And whenever a new task is finished, the document or the downloadable hardcode of this task must be uploaded to the site.

- Quality Assurance: Quality Assurance ensures the quality of the project. To accompolish every week the progress of the project will be discussed in the weekly meetings. Moreover, the project will be handled as Release-Based. When a new module is created or a major change is made, a new version of the project will be released and this release have to be documented. So that the project can be monitored better. Also before the release and milestones in the project, the team will gather for the formal technical reviews. The Project Leader can also call for a technical meeting whenever he needs.

2.1.2. Technical Organization

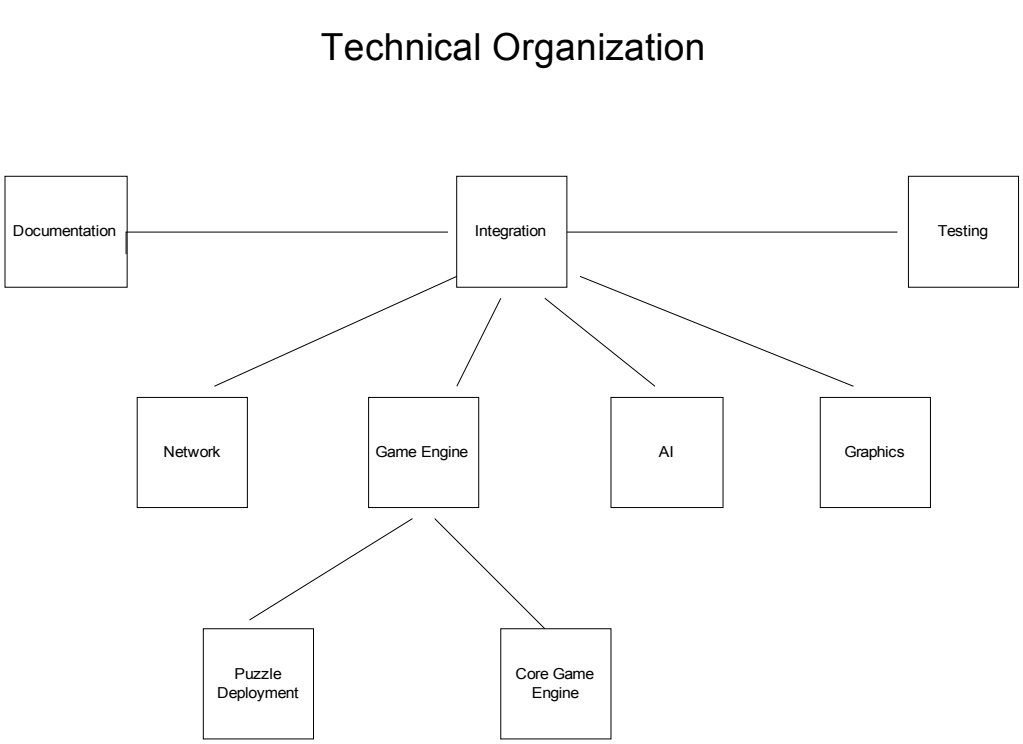


Diagram 2: Technical Organization Overview

Technical Organization [Diagram 2: Technical Organization Overview] deals with the technical parts of the project, the modules taht will be implemented in the project. In order to ensure that any module must not collapse because of one member's failure, for every role at least two team members are assigned. By this, the module could be finished faster and no member will be bored of dealing with only one position. The roles are discussed below:

- Network: Network deals with the network modules in the project, by using appropriate protocols and client-server model, it connests the players who are playing the game in the internet. If one of two members in this role encounters a problem or bug in network, first he will report this problem to his partner and they will try to fix it on their own.

- Puzzle Deployement: deals with the deployement of the puzzles in the game. The puzzle system is the heart of the game. This part makes the game enjoyable. If one of two members in this role encounters a problem or bug in puzzle deployement, first he will report this problem to his partner and they will try to fix it on their own. If needed new puzzles can be designed to make the game more enjoyable.

- Core Game Engine: This part handles the rules of the game. All rules in the game has to identified and implemented accordingly. If one of two members in this role encounters a problem or bug in core game engine, first he will report this problem to his partner and they will try to fix it on their own.

- AI: This part deals with the AI and AI players in the game. To accompolish this some AI models have to be implemented. If one of three members in this role encounters a problem or bug in AI, first he will report this problem to his partner and they will try to fix it on their own.

- Graphics: It is the face of the game, developing rooms, acquiring models and rendering the 3D world in the game is the responsibilities of Graphics. If one of two members in this

role encounters a problem or bug in graphics, first he will report this problem to his partner and they will try to fix it on their own.

- Integration: 5 main roles above have to be integrated well in order to ensure a proper gameplay. This part is the most critical part of the game, so all members of the team are responsible for the integration.

- Testing: In order to ensure a bugfree gameplay, testing palys an important role, the product must be tested several times before the final release. All members are responsible for testing.

- Documentation: As testing documentation is also a very important part of the project in order to develop a player friendly game. As integration and testing it will done by all team members.

2.2. Responsibilities

The people's responsibilities about the given roles above are demonstrated by a chart above. The responsibilities of all members of the team can be seen vividly.

	Süleyaman	Güneş	Önder	Ömer
1.Managerial Organization				
ProjectLeader	X			
Overall ProjectManagement	X	X	X	X
Time Management	X	X	X	X
Web Page Management				X
Quality Assurance	X	X	X	X
2. Technical Organization				
Network		X		X
Puzzle Deployement	X	X		
Core Game Engine	X		X	
AI		X	X	X
Graphics	X		X	
Integration	X	X	X	X
Testing	X	X	X	X
Documentation	X	X	X	X

2.3. Tools & Infrastructure

We have divided the tools and infrastructures into two categories which are the tools used for Project Management and tools used for Technical Coordination.

2.3.1. Tools for Project Management

- **CVS :** CVS stands for Concurrent Versioning System. It allows text and binary files to be versioned in a central repository and allows concurrent development of files. Text files which change a lot in development will be in CVS. Since these files mostly consist of source codes and it is a great opportunity that cvs can let you see any previous version of a file. Every member will check-in and check-out his source code using CVS.

- **Living Schedule:** Living Schedule is very important for time management. The tasks and modules that has to be completed are demonstrated in a spreadsheet and a Gantt Chart. Every member has to complete his part according to the living schedule.

- **Web Page:** All the progress of the project can be viewed via website. The documents(all reports) are uploaded to web site and news about the project can be viewed here.

2.3.1. Tools for Technical Coordination

- **Microsoft Visual Studio 2005:** The project will mainly be developed by this development enviroment. C++ will be used as the main language for the project. Our game will be played in a Windows enviroment and MS Visual Studio is very suitable choice for coding because it is also a Microsoft product as Windows. So we dont have any compatibility.
- **OGRE 3D:** OGRE 3D is our graphics engine in the project. It contains all needed graphics libraries for the project. Its programming language is C++ and it is also compatible with the Visual Studio 2005.

- **XML:** XML is the main language that will be used for puzzle deployement and developing the onthologies.

- **CVS :** CVS is also used for technical coordination . We will use Tortoise as our CVS client. All members will upload their files via Tortoise.

3. The CM Process

3.1. Identification

3.1.1. Identification Criteria

Our configuration items (CIs) are classified according to their physical and functional characteristics and their location within our system. Any component of our project becomes a CI after it meets its baseline (and incrementally changes through our intermediate baselines).

The CIs may be subject to change, which requires formal technical reviews since all the CIs are past their baselines and are dependently interior to the whole system.

3.1.2. Configuration Items

Our product includes the following CIs:

- Managerial Documentation – includes general activities involved with development of the process affecting the system as a whole:
    - Configuration Management Plan: activities planned for CM.
- Technical Documentation – includes technical specifications that describe the system and its functions:
    - Design Report: software design documentation, architectural and functional specifications

- Software Components – includes any program developed by the project team together with all the tools and underlying software that are to be employed by our project:
    - Source Code: all the code output of the project
    - Executables: executable files built
    - CVS Tools: tools used for CVS monitoring ang management (Tortoise, SSH)
    - Support Tools: auxiliary tools (XML & OWL Editor, ShowMesh Viewer, Mesh Editor & Converter)
    - IDEs: environments (Visual Studio 2005)
    - Libraries: imported components (OGRE3D Graphics Engine, OpenAL Audio Library)
- Graphics Resources – includes graphical characters and items:
    - Models: character&item representations (OGRE Mesh models)
- Data and Database Components – includes any source of data that and exists outside is not embedded into the program code (i.e. not hard-coded):
    - Main Database: consists of all the game data
    - Puzzle and Rule Representation Files: XML files of puzzle and rule data
    - Onthology Files: OWL files of action classification information
- Hardware Components – includes the underlying hardware:
    - Server Machine
    - Client Machines
    - Network Connector Device: any means of network connection to allow multiplayer

3.2. Management and Control

3.2.1. Engineering

The following measures are to be applied for engineering management and control:

- Formal Technical Reviews - these reviews, on a weekly basis, involve participation of the whole team for architectural review in order to:
    - revise the system architecture from the perspectives of developers of different modules
    - monitor architecture-implementation consistency
    - propose means of compromise in the case of any incapability

3.2.2. System Management

The following measures are to be applied for system management and control:

- Progress Monitoring – horizontal communication among groups will be employed through informal meetings in order to:
    - monitor the activity and progress of each individual group
    - verify their contribution to the system
- Integration Activities – integration meetings 1-2 weeks before the milestones with different groups in order to:
    - consistently integrate the product
    - deploy the executable release
    - document any incompatibility among different modules
- CVS Utilization – fully utilize CVS in order to:
    - employ code authorization for different groups
    - facilitate version history and change history documentation
    - allow parallel development of the code

3.2.3. Development

The following measures are to be applied for development management and control:

- Coding Conventions – conventions applying to source code & files
  - Document Identification: There will be document headers containing information about the code, programmers of the code, functionality and purpose of the code.
  - Version and Change History: Every change made is documented with date information, so as to facilitate tracking and recovery. A sample source file header is as follows:

```
/**************************************************************************
* Project:      Ma3e-3D
* Filename:     AnimationEngine.cpp
* Date:         02/03/2007 13:40
* Purpose:      Implementation of the class AnimationEngine
* Author:       Önder Babur, Süleyman Cincioğlu
* Version:      1.01
*
**************************************************************************
Changes
01/03/2001 21:00:  Süleyman Cincioğlu : Modified walking animation speed

**************************************************************************/
```

  - Function Comments: Function history, with version, date information and notes are presented as function header. A sample function header is as follows:

```
/**********************************************************************
* Routine:      processAnimation
* Purpose:      Read BackColor property for the ClipList control.
* Arguments:    None
* Returns:      None
* Notes:        As a main loop for initializing and maintaining all the animation
* Version:      1.01
* Since:        1.00

**************************************************************************
Changes
01/03/2001 21:00:  Süleyman Cincioğlu : Set mWalkSpeed 100->150

 **********************************************************************/
```

- Naming and Labeling Conventions – conventions for consistency and understandability of the file and directory structure of the project:
  - Main Repository: The global repository will be in CVS. Everyone will get their own copy of that as a local repository in their own machine. The structure of the main repository will have the following structure:

Directory name	**Content information**
backup	Regular backups of source code with version information and explanation, (see below item for examples)
bin	Executables
conf	Configuration files. e.g. containing IP and port number, player number
docs	Readme's, release notes, user manual, etc.
lib	Library files for OGRE3D and OpenAL
src	Source code files. Has the following structure:

	src ----- client ----- GameEngine | |-------- GraphicsEngine | |-------- AudioModule | |-------- NetworkModule | |------- server ----- GameEngine |--------- NetworkModule |--------- AIEngine

- Version backup: Regular backups will be taken as a recovery mechanism. Some sample backup files may be given as follows:
  TheMa3e_v1.00_02.03.2007_beforeAnimationEngine.rar
  TheMa3e_v3.87_05.09.2007_beforeAIAlgorithm.rar

- IDE Utilization – maximum exploitation of the facilities of Visual Studio 2005
  - CVS Automatization: Using the automatic CVS check-in/out property of VS 2005 to increase efficiency.
  - Database Implementation: Using VS 2005 feature of Database management tools.

Testing & Debugging: Using the high level debugger and automated testing facility of VS 2005 where possible.

3.2.4. Defect Tracking

During the lifetime of the "Ma3e-3D" project conducted by GOSOft, all responsible parties will take the following formal definition of ***defect*** into account:

> *A defect is a problem realized during the software management process that affects the quality of a particular or a group of specifically identified Configuration Items (CIs). A problem will affect the quality of a CI if as a result the CI does not meet the expected requirements previously set for that particular item and/or group of items.*

When a responsible party detects a defect in a particular and/or group of CIs, the following action plan will be followed:

1. The defect will be classified based on the resource(s) it is associated with.
2. An informal priority (extremely urgent, high, medium, low) will be assigned to the defect by the responsible party at the time of detection.
3. The defect will be documented in a standard format defined in the SCMP document.
4. Unless, classified as *extremely urgent*, all documented defects will be discussed on the regular Saturday meeting for initiating change requests. Otherwise, the responsible party will call out an impromptu meeting.
5. In case of an impromptu meeting, the responsible party will determine the attendees based on the scope of the resource(s) that the defect possibly affects (i.e. if the problem needs extremely urgent handling but it only affects the components within the Network architecture, only the responsible staff will be called to the impromptu meeting). The meeting will produce a change request. The team manager will be informed about the progress. Furthermore, the issue will be made public to all team members during the Saturday meeting.

Following is a non-normative guideline as to how the responsible party should classify the defect:

- Extremely urgent:
  The defect indicates that the root cause is a theoretical phenomenon and/or a major design flaw. It is very likely that the design and even the theoretical requirements that led to that design will be reconsidered.
- High:

The defect is most probably associated with a design flaw. However, there exists a documented/undocumented method the responsible party plans to take to overcome the situation (i.e. if performance tests indicate that the performance of Ma3e-3D is low due to network issues, the action to be taken is to split the servers and the responsible party is aware of such a plan). It is probable that the defect will lead to medium-level rearrangements in the schedule of the project.

- Medium:

  The defect is most-probably based on minor design flaws. The responsible party is capable of coming around the problem in a timely manner. However, when evaluated together with other defects, it might be a warning of a larger problem. Hence, it requires all team members' attention.

- Low:

  The responsible party considers the defect to be local (functional, class level) and/or due to programming errors.

Schema 1 : Non-normative Guideline for Defect Priority Assesment

In documenting the defects the following format [Table 1: Formal Defect Documentation] will be utilized:

Field	Description
Date and Time	The date and time when the defect is detected.
Responsible Party	The team member(s) issuing the defect.
Associated Resource(s):	The Configuration Management Items (CMI) due to which the defect is. In case of software components; modules, classes, functions, data items and code lines must clearly be identified together with the versioning information. For documents and other resources, the name of the resource and the names of the sections should be reported.
Description	A brief but self-explanatory verbal description of the defect. A fluent, error-free language should be utilized.
Judgment	The responsible party's possible interpretation on how to resolve the problem. Technical aspects should be in focus.
Priority	The responsible party's individual assessment of the priority of the defect based on the principles in .

Table 1: Formal Defect Documentation

3.2.5. Change Requests

A change request is a formal declaration that indicates a change is to be made on a Configuration Item (CI) due to the existence of associated defects.

Throughout the development of the "Ma3e-3D" project, change requests will be made based upon the documented defects. On every regular Saturday meeting, the documented defects will be briefly discussed among the team members. During the evaluation process, the defects will be categorized based upon their associated resources. First, the responsible parties for the associated resources will come up with the change requests. Then the change requests will be discussed among the team members. Diagram 3 : Association Between Defects and Change Requests depicts a general guideline for reaching a decision on issuing change request based on the documented defects.

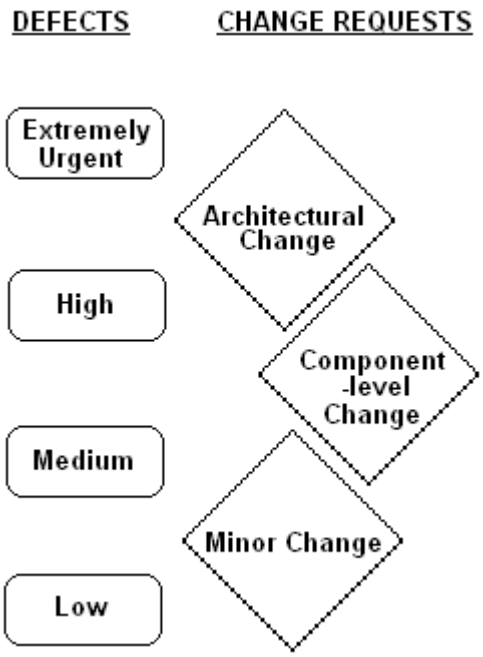


Diagram 3 : Association Between Defects and Change Requests

A change request should be classified as belonging to either of the following categories:

1. Architectural change
2. Component-level change
3. Minor change

Table 2: Change Request Categorization thoroughly describes how to classify a change request and which plan to follow in implementing the change.

Category	Classification Schema	Action Plan
Architectural Change	• Involves critical modifications on the design and even various refinements on the requirements to be made. • The change disrupts the interaction between the functionally classified components of the framework (e.g. Game Engine, Graphics Engine, Network Component, etc.). Hence involves a collaborative planning and management. Every team member's involvement is required. • The change requires refinement on the project schedule.	• The issue is discussed in either an impromptu or a regular meeting. Advisors and teachers are consulted. A decision is reached whether or not to proceed with the change[1]. Modifications on the schedule are made such that the project suffers the least amount of cost. • Workload is divided based on the general responsibilities of the team members. • Integration level test cases are developed. • Modifications are made. • System integrity is verified via the predefined test scenarios.
Component-level change	• The change is on various locally unidentifiable parts of a module or resource that serves a particular function within the framework.	• The responsible team members (usually two) make an informal meeting and devise a plan to apply the change.

[1] For the "Maze-3D" project, the First Release Milestone serves as a point-of-no-return for architectural changes.

	• The change generates side effects on other sections of the same module/component; therefore a holistic approach is necessary. • (e.g. changes made in various classes in the Network Backbone, Game Engine, etc.)	• Simple test scenarios are formed to make sure possible side-effects are dealt with. • Changes are made in the locally unidentifiable parts of the module/resource. • Tests are conducted for the verification. • The team is informed of the progress in the first upcoming regular meeting (Tuesday | Saturday).
Minor change	• The change is local to a particular, identifiable portion of a resource and its effects are not to generate side-effects on other sections and/or components. • (e.g. a change made within a function that does not alter its overall behavior, or a change made within a class that does not interfere with its interaction with other classes)	• The team member (usually one) responsible for the particular, identifiable portion of the resource makes the modification. • The modification is documented in compliance with the Coding Conventions presented in this document.

Table 2: Change Request Categorization

In addition to in-place documentation of the modifications based on the Coding Conventions presented in this document, throughout the "Maze-3D" project lifecycle, change requests will be formally described based on the format specified on [Table 3: Change Request Formal Documentation].

Field	Description
Date and Time of Request	The date and time when the request is approved.
Effective Date Range	The predicted dates on which the modifications are to be made.
Issuer	The team member(s) issuing the change request.
Category	The category to which the change request belongs (as decided by the group based upon the guidelines on Table 2: Change Request Categorization).
Responsible Members	The name of the team members responsible for implementing the modifications. Their specific responsibilities should be clearly outlined.
Associated Resource(s):	The resources identified via the names of the class, module, function, line number and version on which changes are to be made. On a functional level, the changes to be made will be documented.
Description	A brief but self-explanatory verbal description of the overall change request.
Side Effects	A brief but self-explanatory verbal description of the possible side-effects associated with the change. Threatened resources are clearly identified.
Conclusion	Comments on the procedure after the changes are made. Was it successful? Are there any additional problems?

Table 3: Change Request Formal Documentation

3.2.6. Building and Deployment

It is important to keep in mind that during the development of the "Ma3e-3D" project, the CVS will not be utilized in directly building the releases and/or temporary executables. The game to be developed utilized Operating System specific libraries and consequently it should be developed and run under certain conditions. More specifically,

- Microsoft Windows XP Operating System (second edition), and
- OGRE-3D libraries are required.

Furthermore, the team members have decided to use Microsoft Visual Studio 2005 as the development environment. A uniform selection is due to the fact that the team members desire problems due to configuration differences to be as narrow as possible.

As a matter of fact, the building and deployment process is as follows:

1. The responsible party checks-out the whole directory structure from the CVS.
2. There exists the Microsoft Visual Studio solution file. The developer opens it.
3. The developer makes the modifications in the solution. There are two projects: 1) TheMa3eClient 2) TheMa3eServer.
4. The developer uses Microsoft Visual Studio's compiler to build the temporary/release executable. The solution file is set such that the temporary/release executable is stored in the bin directory.
5. The whole directory structure is checked-in using the CVS client.

When a responsible party wishes to run the game, he/she

1. Checks out the whole directory structure from using the CVS client.
2. Runs the server, which is located under bin/server.
3. Runs the client, which is located under bin/client.

Every two or three weeks the team will deploy a new release. In such a case, the directories "src", "bin" and "doc" will be archived and named with the following naming convention:

Ma3e_Release_<Release Count>_<Year>_<Month>_<Day>

On every release, the team members will clearly document the functionalities implemented so far, all the bugs and errors encountered and what to do until the next release.

4. Project Schedules

Below you can find the main milestones of our project. You can find the full living schedule [Diagram 4 : Living Schedule] in the Appendix.

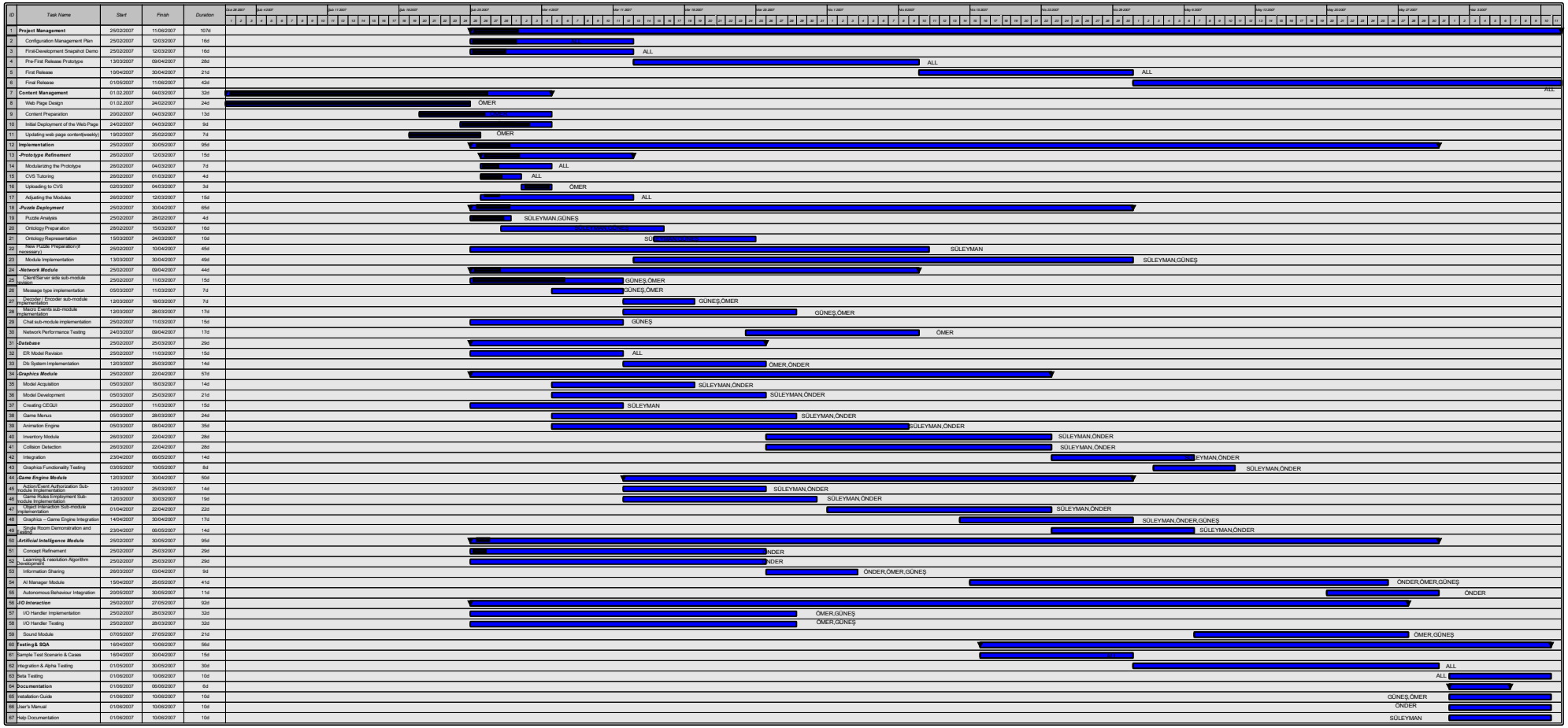
- **First Development Snapshot Demo:** It is the modular version of the first semester's prototype. All the modules will be identified and new modules can be added upon the first semester's prototype. Its deadline is 12.03.2007.

- **Pre-First Release Prototype:** It is the milestone of our project which the skeleton must be finished. By this prototype network and database modules will be complete and some progress in other modules are necessary. Its deadline is 09.04.2007.

- **First Release:** It is an official milestone and after this milestone no architectural change can be made in the project. Puzzle deployement and Core Game Engine will be completed and significant progress wll be done on graphics module. Its deadline is 30.04.2007.

- **Final Release:** It is the end of the project. All modules must be finished till then , and integration, testing and documentation must be completed. Its deadline is 11.06.2007.

5. Appendix



ID	Task Name	Start	Finish	Duration
1	**Project Management**	25/02/2007	11/06/2007	107d
2	Configuration Management Plan	25/02/2007	12/03/2007	16d
3	First-Development Snapshot Demo	25/02/2007	12/03/2007	16d
4	Pre-First Release Prototype	13/03/2007	09/04/2007	28d
5	First Release	10/04/2007	30/04/2007	21d
6	Final Release	01/05/2007	11/06/2007	42d
7	**Content Management**	01.02.2007	04/03/2007	32d
8	Web Page Design	01.02.2007	24/02/2007	24d
9	Content Preparation	20/02/2007	04/03/2007	13d
10	Initial Deployment of the Web Page	24/02/2007	04/03/2007	9d
11	Updating web page content(weekly)	19/02/2007	25/02/2007	7d
12	**Implementation**	25/02/2007	30/05/2007	95d
13	***-Prototype Refinement***	26/02/2007	12/03/2007	15d
14	Modularizing the Prototype	26/02/2007	04/03/2007	7d
15	CVS Tutoring	26/02/2007	01/03/2007	4d
16	Uploading to CVS	02/03/2007	04/03/2007	3d
17	Adjusting the Modules	26/02/2007	12/03/2007	15d
18	***-Puzzle Deployment***	25/02/2007	30/04/2007	65d
19	Puzzle Analysis	25/02/2007	28/02/2007	4d
20	Ontology Preparation	28/02/2007	15/03/2007	16d
21	Ontology Representation	15/03/2007	24/03/2007	10d
22	New Puzzle Preparation(if necessary)	25/02/2007	10/04/2007	45d
23	Module Implementation	13/03/2007	30/04/2007	49d
24	***-Network Module***	25/02/2007	09/04/2007	44d
25	Client/Server side sub-module design	25/02/2007	11/03/2007	15d
26	Message type implementation	05/03/2007	11/03/2007	7d
27	Decoder / Encoder sub-module implementation	12/03/2007	18/03/2007	7d
28	Macro Events sub-module implementation	12/03/2007	28/03/2007	17d
29	Chat sub-module implementation	25/02/2007	11/03/2007	15d
30	Network Performance Testing	24/03/2007	09/04/2007	17d
31	***-Database***	25/02/2007	25/03/2007	29d
32	ER Model Revision	25/02/2007	11/03/2007	15d
33	Db System Implementation	12/03/2007	25/03/2007	14d
34	***-Graphics Module***	25/02/2007	22/04/2007	57d
35	Model Acquisition	05/03/2007	18/03/2007	14d
36	Model Development	05/03/2007	25/03/2007	21d
37	Creating CEGUI	25/02/2007	11/03/2007	15d
38	Game Menus	05/03/2007	28/03/2007	24d
39	Animation Engine	05/03/2007	06/04/2007	35d
40	Inventory Module	26/03/2007	22/04/2007	28d
41	Collision Detection	26/03/2007	22/04/2007	28d
42	Integration	23/04/2007	06/05/2007	14d
43	Graphics Functionality Testing	03/05/2007	10/05/2007	8d
44	***Game Engine Module***	12/03/2007	30/04/2007	50d
45	Action/Event Authorization Sub-module Implementation	12/03/2007	25/03/2007	14d
46	Game Rules Employment Sub-module Implementation	12/03/2007	30/03/2007	19d
47	Object Interaction Sub-module Implementation	01/04/2007	22/04/2007	22d
48	Graphics – Game Engine Integration	14/04/2007	30/04/2007	17d
49	Single Room Demonstration and Testing	23/04/2007	06/05/2007	14d
50	***-Artificial Intelligence Module***	25/02/2007	30/05/2007	95d
51	Concept Refinement	25/02/2007	25/03/2007	29d
52	Learning & resolution Algorithm Development	25/02/2007	25/03/2007	29d
53	Information Sharing	26/03/2007	03/04/2007	9d
54	AI Manager Module	15/04/2007	25/05/2007	41d
55	Autonomous Behaviour Integration	20/05/2007	30/05/2007	11d
56	***-I/O Interaction***	25/02/2007	27/05/2007	92d
57	I/O Handler Implementation	25/02/2007	28/03/2007	32d
58	I/O Handler Testing	25/02/2007	28/03/2007	32d
59	Sound Module	07/05/2007	27/05/2007	21d
60	**Testing& SQA**	16/04/2007	10/06/2007	56d
61	Sample Test Scenario & Cases	16/04/2007	30/04/2007	15d
62	Integration & Alpha Testing	01/05/2007	30/05/2007	30d
63	Beta Testing	01/06/2007	10/06/2007	10d
64	**Documentation**	01/06/2007	06/06/2007	6d
65	Installation Guide	01/06/2007	10/06/2007	10d
66	User's Manual	01/06/2007	10/06/2007	10d
67	Help Documentation	01/06/2007	10/06/2007	10d

Diagram 4 : Living Schedule