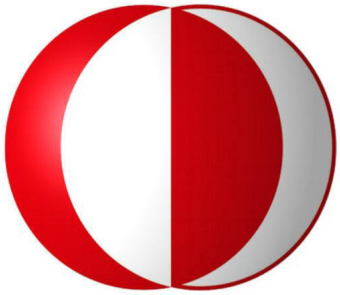
Middle East Technical University



Ceng 492
Test Specifications Report Report

Massively Multiplayer Online Role Playing Game Project
Virtual Turkey

MECAC
Cinar Kilcioglu
Mert Degirmenci
Umit Cavus Buyuksahin

April 24, 2011

Contents

List of Figures

List of Tables

1 Introduction

This document contains the test specifications of "Virtual Turkey" which is a massively multiplayer online role playing game (MMORPG). The approach used in this specification is adapted from IEEE standards [1]. MECAC, the project group assumes full responsibility of the requirements presented in this document.

1.1 Goals and Objectives

All MMORPG projects have to be scalable, secure, and efficient in data transfer as much as possible. The main goals of the testing process for "Virtual Turkey" are determined based on these criteria, and set as following:

- The project has to be scalable, that is, it should be massively multiplayer.
- The project may not have any loss in important data between server and client, such as account info, character attributes.
- The data transfer between server and client should be fast and secure.
- The project must be bug-free.

1.2 Statement of Scope

Scope of software testing process of "Virtual Turkey" includes unit testing, integration testing, stress testing, security testing, alpha testing, and beta testing. Each of these parts is explained below:

Unit Testing Server side and client side are tested separately. In server side, functionality of server network component and management component are tested. The most critical part in server network component is to test whether server holds the required information. Management component is tested whether it is fully functional in terms of data management, non-playing characters, and game loop.

In client side, functionality of client network component and physics component are tested. Client network component is tested whether client is connected to server and available to connect. In physics component, collision detection and all other sorts of physical interactions of the player are tested.

Integration Testing The interaction between the two main parts of the game, server and client are tested with white-box method. Correctness, speed, data loss of the data transmitted between server and client are the main concerns in integration testing.

Stress Testing Server is tested under maximum number of clients concurrently connected. Correctness, speed, data loss of the data transmitted between server and client are evaluated.

Security Testing Whether a client can manipulate other clients' account information is tested.

Alpha Testing Project as a whole is tested by the group members after parts are tested separately.

Beta Testing People outside of the project group are included to perform testing of the project in customer environment. This will be the sanity check of the project.

1.3 Major Constraints

1.3.1 Time

As the project is still being developed, there will not be enough time for fully and effective testing. That is why testing will start while project is being implemented. It is natural to confront with some bugs during testing phase, and correcting them is time consuming.

1.3.2 Hardware

The server computer should have enough computation power and capacity to house multiple clients at the same time. Although a testing client, which works on server computer, has been implemented with minimum memory usage, it is predicted that it will not be enough to estimate the capacity of the server.

1.3.3 Platform Dependency

The project is developed in Microsoft Visual Studio, and it uses XNA. Because of this reason, the project requires an operating system that belongs to Windows NT family.

1.3.4 Lack of Volunteer Beta Tester

Project code has to be downloaded by adequate number of users. However, users may not be volunteer to download the game to their personal computers due to security reasons.

1.4 Definitions, Acronyms, Abbreviations

MMORPG	Massively Multiplayer Online Role Playing Game
NPC	Non-player Characters
SDD	Software Design Descriptions
SRS	Software Requirements Specifications
SVN	Subversion

1.5 References

[1] IEEE, IEEE Std 829-1998, *IEEE Standard for Software Test Documentation*, IEEE Computer Society.
[2] IEEE, IEEE Std 610.12-1990 *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Computer Society.

2 Test Plan

Testing of the project will be implemented in order of testing phases explained below. Firstly, unit testing, integration testing, stress testing, and security testing will be done sequentially, with full support of project group members. Then, alpha testing is done again by group members. Finally, beta testing will be implemented by people outside of the group.

2.1 Software to be Tested

The software being tested has two main components as explained above. First component is server. It has two subcomponents, server network component, and management component. Second main component is client and it has two subcomponents, client network component, and physics component. Interaction between subcomponents and components is the most crucial process of the software to be tested.

2.2 Testing Strategy

As explained above, testing phases will be implemented sequentially. Moreover, due to time limitation of the project, some of these testing phases will be done during implementation of the project.

2.2.1 Unit Testing

Unit testing will be done for each component of the project.

Server component will be tested in terms of server network component and management component. Server network component is the most important part of the project as this is the component that communicates with all connected clients. The project group must ensure that this component can respond all of the requests coming from clients. Units to be tested in this component are:

- Character states
- Client server connection setup
- Client server messaging and event matrix
- NPC states

Management subcomponent inside server component is tested whether account information of the clients are stored correctly. Units to be tested in this component are:

- Server account manager
- Server database connection

Client component has two main subcomponents, client network component and physics component. Client network component can be considered as the client side of the server network component, and it has to be tested whether client has a proper connection with server. Units to be tested in this component are:

- Client server connection setup
- Client server messaging and arrangement of event matrix

Physics component is tested whether the game complies with the real world environment. Units to be tested in this component are:

- Physics helping to combine the physics with the graphics
- Objects drawing necessary objects in the game, such as car, height map, triangle mesh objects.

2.2.2 Integration Testing

Integration of server and client is tested after unit testing is completed. Firstly, it will be ensured that server component is working correctly. Then, client component will be integrated with server component. Finally, the interaction between these two components will be monitored and tested under different conditions.

2.2.3 Validation Testing

Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements [2].

Firstly, server component is tested whether the requirements mentioned in Software Requirements Specifications (SRS) Report have been met with the design explained in Software Design Document (SDD). Any inconsistency is eliminated by either updating the requirements report or changing the design according to design document. After server component is validated, client component is validated.

2.2.4 High-order Testing

High level tests can be divided into four sections, namely, performance testing, stress testing, security testing, and alpha beta testing.

Performance Testing Performance of the server is tested under different client loads and evaluated using test metrics determined by the project group.

Stress Testing Server is tested under maximum number of clients concurrently connected. Correctness, speed, data loss of the data transmitted between server and client are evaluated, as mentioned in section 1.

Security Testing Whether a client can manipulate other clients account information is tested, and any potential security holes will be determined.

Alpha & Beta Testing First, the project will be tested by developers, i.e. the group members, then beta testing will be implemented with the support of volunteer tester, who are from outside the project group. It is planned to increase number of users connected under the control of project group. It will provide to analyze the system with different number of clients. If any important problem has been identified during beta testing, it will be reported as ticket by the help of bug tracker, and the project group will deal with it.

2.3 Test Metrics

Test metrics are decided by the group members as following:

- Maximum number of clients concurrently connected to the server
- Response time of the server to a client
- Size of a random packet transferred between server and client
- CPU and memory usage of a client
- CPU and memory usage of the server

2.4 Testing Tools and Environment

As mentioned above, the project is developed in Microsoft Visual Studio, and it uses XNA. Because of this reason, the project requires an operating system that belongs to Windows NT family. All types of testing are implemented in Microsoft environment.

One of the testing tools being used is WireShark. WireShark is a network sniffing tool, and it is used for monitoring the data transfer between server and clients in the project. Another testing tool to be used is NUnit. NUnit is a unit-testing framework for .NET languages. Finally, TRAC is used for tracing the bugs of the project, which helps to fix them.

3 Test Procedure

3.1 Unit Test Cases

"Virtual Turkey" has two individual units of source code. These are namely server unit, and client unit. It will be determine if they are fit for use by a combination of white box and black box testing procedures.

Server Unit Testing The server will be tested by connecting number of clients distributed around a network topography to it. It will be mainly tested for its capacity to handle the connections without noticeable delays to the end users. White box testing procedures will also be applied to determine if the server component is ready to deploy. This also involves writing test cases that check the consistency of the code optimization for a specific code block. The server unit testing will be done automatically by a tester program that run on

the client side, and a tracker program that monitors the server side. robots which wander around "Virtual Turkey" to test the scalability of server have already been implemented.

Client Unit Testing To test the client unit, a client will manually be used to connect to the server and play. The executable will be downloaded to group members' personal computers and they will play the game. During the game play, the main goal will be to identify any game delays and inconsistencies in the game logic. This unit will be tested using black box testing techniques. There are three testing techniques that are going to be used for this unit. These are as follows:

Control Flow Testing The client unit will be tested if it obeys the certain control logic that has been designed for "Virtual Turkey". This testing technique also involves checking consistency of the game logic.

Data flow Testing The data that flows within the client and through the server will be tested by WireShark. Each and every packet delivered to the server will be examined. To achieve this, the messages will be decoded at first, as they will be compressed.

Penetration Testing Penetration to client unit and changing its control flow to cheat the game and gain obvious advantages over opponents will be deliberately tried. It will be done without peeking through the source codes of the client component. However, for exceptional cases, it may be required to look at the source code.

3.2 Integration Testing

The main aim of integration testing is to expose faults in interaction between integrated components. After the components have been unit tested, they are taken as inputs in integration testing. The project have two main components that are server and client components. These components also have some subcomponents, namely server network component, management component, client network component, and physics component.During unit testing of the main components, their subcomponents have been grouped. In other words, on a small scale integration testing have occurred in unit testing. However, the major integration testing occurs on server and client components.

Although Top-down and Bottom-up integration testing approaches are preferable in general, for this project Big Bang testing is better, since there is not enough components for iterative approaches and differences between level of components are not explicit. Based on Big Bang approach, server and client components are aggregated together to the system.

In this testing, three main requirements are verified that are functionality, performance and reliability.

In functional analysis, functionalities of communication between server and client will be tested with the following items:

- Check if connection between server and client is established.
- Check the request from client to server.
- Check the response from server to client.
- Check if the connection is closed correctly.

In performance analysis, overhead of established connection between server and client will be tested with the following performance requirements:

- Packet size of a request from clients,
- Packet size of a response from server,
- Response time in a given request,
- Total delay for packet transferring.

In reliability analysis, dependability of established connection between server and client will be tested as following:

- Check the response message for given request.
- Checking correctness of responded client.

3.3 Validation Testing

In validation testing procedure, consistency between the final design and proposed design is tested, and whether the requirements specified in SRS is met are checked. Validation testing will begin with server component. After that, client component will be validated.

Data Design Validation Testing

- Have all data objects been created with sufficient attributes as mentioned in SDD?
- Are the attribute types determined correctly?
- Have all relationships between entities designed before being constructed?

In data design, it is vital that relationship between data objects, especially NPC and characters has to be constructed correctly.

Architectural Design Validation Testing

- In client component, does the messaging system from client to server conform the requirements determined before?
- Are all message types (e.g. register, login, update) sent as in SDD?
- Is physics subcomponent active for cheating checking and graphics engine as planned?
- In server component, does the messaging system from server to client conform the requirements determined before?
- Are the prioritization of the messages implemented correctly?
- Is NxN event matrix consistent with SDD?
- Is management subcomponent effective as proposed?

To exemplify an architectural design requirements, client network component must be in contact with client user interface and server network interface with register, login, logout, and chat functions, each of which takes different arguments.

User Interface Validation Testing

- Check if all interfaces mentioned in SDD satisfy the minimum requirements.
- Check if client-server interface is working as expected.
- Check if the screen objects flowchart is implemented.

14 different screens have been mentioned in SDD with each having different properties. These properties were the minimum requirements and have to be satisfied in the final product. For example, "Create Account Screen" has to have username, password, and email cells, minimum. Moreover, transition between user screens has to be implemented correctly, as it directly affects the playability of the game.

3.4 High-order Testing

High-order testing is divided into four sections, namely, performance testing, stress testing, security testing, and alpha and beta testing. Although some parts of these tests have been done in unit testing and integration testing, the product will be tested as a whole in this step.

Performance Testing The project is going to be tested in different level of loads. Load of the game is going to be the number of clients concurrently connected to the server. Results of test metrics will be monitored under different number of clients. First, performance of the game is tested when all clients are connected from METU campus and server is also at METU campus. Then, it will be tested when clients are from off-campus. If the results of test metrics are acceptable, the load of the game will be increased gradually by connecting additional clients to the game as a final step.

Stress Testing As maximum number of clients connected to the server is one of the most important performance criteria of the game, stress testing has to be implemented carefully, and the results of the tests must be anticipated correctly. When the server reaches its maximum capacity, which is determined through the former test phases, playability of the game will be checked by investigating the movements and actions of the robots. Performance of the clients will be tested by investigating test metric results. The response time of the server to client, CPU and memory usage of the server computer are just two example metrics that are going to be monitored in stress testing. If these values are not sufficient, even a client can login to server, it cannot be counted in calculating maximum capacity, as it does not meet the requirements set by MECAC.

When the server reaches maximum number of clients, in order to maintain clients' reliable connections, any additional requests to log in to the game has to refused. This situation will also be tested in stress testing.

Security Testing Security testing determines whether the system meets specified security requirements that are necessary for security of the system. In massively multiplayer game, since there exist quite many users, the game has to deal with their accounts. As users connect each other via server in this project, it must provide security. Server must supply the following items:

- Each user's confidential information has to be preserved.
- Each user's confidential information can only be accessed by authorized clients.

Alpha & Beta Testing Virtual Turkey is currently being alpha tested by MECAC team. The game has been installed and fully functional at two MODSIMMER computers. As the deadline for beta testing is approaching, MECAC is concurrently doing alpha testing with unit testing, and started to setup necessary software for beta testing. The client binaries has been uploaded to the team website, and is currently downloadable. MECAC needs mutual

trust of voluntary beta testers, and assures that the downloadable client package contains no malware of any kind. Advertisement may be needed to find players (a.k.a beta testers). MECAC cooperates with MODSIMMER to determine appropriate amount of resources such as advertisement budget and server hosting.

4 Testing Resources and Staffing

For unit test, each group member will be responsible for a specific component of the project and implementing the unit test of that component. In integration and validation testing, whole group will be testing at the same time. After these tests are been completed, group members will be implementing alpha testing. However, for beta testing, the group will be in need of both volunteer beta testers and additional computers.

In short, until beta testing, the group does not need any additional resource or staff, all it is needed is fully committed group members and a computer for each group member. For beta testing, end-users outside from the group and sufficient number of computers for those end-users are required.

5 Test Work Products

The most important test work product of the project is test robot, which is an automated client who logs in to the game and makes random movements in the map. Number of test robots concurrently connected to the server can be decided by the project group. By playing with this number, it will be available to monitor the data transfer between server and test clients. Since the main aim of the robot is to handling the data transfer, graphics and sound effects of the robot are turned off. This enables to increase the number of robots connected to the server by decreasing the CPU usage and memory usage of the project.

The connection between test robots are monitored by "Visualizer", which has already been implemented. In this program, which robot client makes a connection to which robot client can be monitored via their IP addresses and port numbers. Moreover, the program provides tester to change the maximum connection number as bandwidth. Screenshot of the program when three users are online is given in Figure 1.

6 Test Record Keeping and Test Log

Any kind of test being implemented are reported in weekly progress reports. For test logging, Subversion (SVN) and TRAC are used. SVN provides all versions of the project codes, which

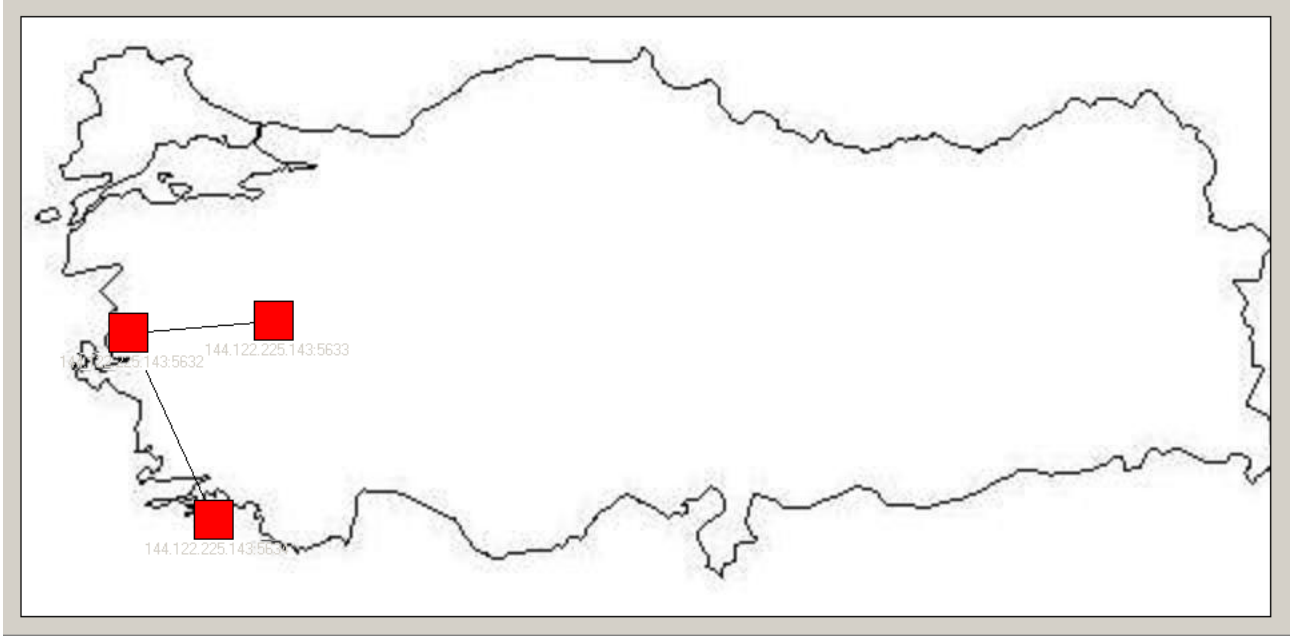


Figure 1: Visualizer

are updated according to test results, and TRAC helps to identify the bugs in the project. They are stored as test log.

7 Organization and Responsibilities

Components of the project mentioned above are divided into the group members. Each group member will be responsible for reporting test results for his own components. After whole group is informed about the unit testing results, integration testing will be done by all group members. As the project team was structured with no team leader, all peers will be monitoring the others' works. Validation and high-order testings are done again as a group. Additional responsibility will be added to each member in beta testing phase, which is finding adequate number of volunteer beta testers.

Each group member will be responsible for validating testing phases with this document. Umit is going to be responsible for whether all unit testing steps mentioned in this document are implemented. Mert will check if all integration testing cases are tested, and Cinar is going to validate that the project passes all high-order testing phases.

8 Test Schedule

Schedule for testing is determined as in Table 1.

Table 1: Test Schedule

Task	**Date**
Unit Testing	April 26 - May 1
Integration Test	May 1 - May 5
Validation Test	May 5 - May 6
Security Test	May 6 - May 9
Stress Test	May 9 - May 14
Alpha Test	May 14 - May 16
Beta Test	May 16 - May 20
Finalization	May 20 - May 25