CPSC 319
Team 2

2COMMUNICATE

Dr. Pat Mirenda

Test Plan

Version: 1.1 Date: (04/07/2006)

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Michael Tien	Initial Draft	Mar 24th, 2006
1.1	Michael Tien	Final Draft	Apr 7th, 2006

Contents

1 Introduction

This section covers the objectives and extent of the tests.

1.1 Overview

The goal of this test plan is to guide the testers through the test phase so that the testers can execute the necessary tests in an efficient and cost-effective way. It also gives the managers a brief idea of how the tests shall be done, and the time frame each test phase shall require.

1.2 Definitions, Acronyms, and Abbreviations

ACSS – Autistic Conversational Skills Software, the software that this SRS describes, simulating a real life conversation and allowing users to choose when to interrupt the given conversation

SRS – Software Requirements Specification, this document which outlines the requirements that the software must fulfill. Entirely design independent.

InterruptED – The temporary internal developmental name for the ACSS, used for the simplicity of labeling windows/title bars, etc. The completed project will be named according to marketing and research needs.

User – any person who uses the program, with the general case being children ages 6-19 with autism or having Asperger’s disorder who have conversational difficulties.

Administrator – a person who has administrative privilege/access to the system

GUI – graphical user interface

Main Menu Page – the initial menu to allow users to register/login

Game Menu Page – the initial menu for users who have logged in

Registration Page – the page that allows the user to create new accounts

Login Page – the page that allows the user to login if they have a valid login

UML – Unified Modeling Language. Refer to http://www.uml.org/

OS – operating system

SDS – Software Design Specification

RC – Release Candidate

CVS – Concurrent Versioning System

Client – Dr. Pat Mirenda, with possible inclusion of her department and/or research staff.

2 Relationship To Other Documents

This section covers the relationship of this test plan to the other documents created during the design phase such as SRS and SDS.

2.1 Software Requirements Specification Document (SRS)

The SRS identifies the product functions (section 2.2) that are used to categorize the test cases for the system test. The implementation of the software is broken down into smaller classes based on the production functions, and the test cases shall be created accordingly.

Section 3 of the SRS (external interfaces) also provides the procedures to some black box testing for the system test. Each table contains a title, an input, and an expected output. The test cases shall be created according to these tables.

Section 2.4 of the SRS (constraints) specifies the system requirements for the tests such as the version of the OS and the size of the RAM.

2.2 Software Design Specification Document (SDS)

Section 5 for the SDS (lower level design) contains all the diagrams of the design such as class analysis diagrams, state chart, and the sequence diagrams. The class analysis diagrams contain all the methods and attributes of each class, and this shall be used to generate the test cases for the unit test. The state chart and sequence diagram identify the relationships and flow between classes, and these shall be used to design the test cases for the integration test.

2.3 Project Plan

Section 4.2 of the project plan (quality assurance) contains a brief outline of the test plan. It also specifies an approximate time allocation for each test phase. This section shall be used to derive the test schedule.

3 System Overview

This section covers an overview of the system in terms of the components that need to be tested during the unit test.

3.1 Components

These are the four major components of the software packages:

- User Interface Subsystem
- User Accounts Subsystem
- Game Subsystem
- Storage Subsystem

Each component is responsible for a specific aspect of the software.
The four major components also rely on other independent sources such as the XML database and the video files.
The User Interface Component will be tested with screen-state transition tests.
The User Account, Game and Storage Components will be tested congruently to ensure data consistency.
There must also be an individual test for the XML database to check that the database state is consistent with the actions that have been applied.
The game module must be checked so that any errors during processing will result in appropriate runtime messages.

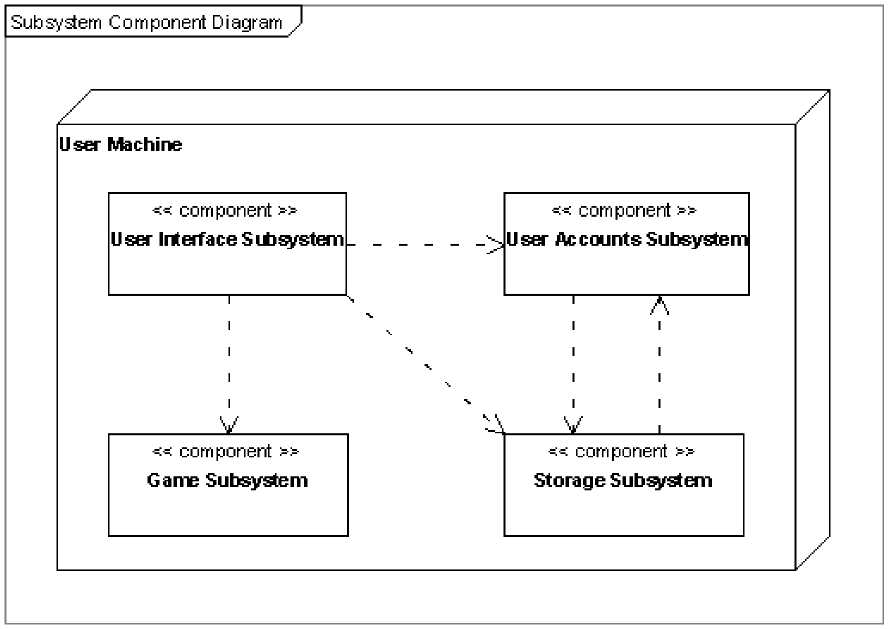

Fig 3.1: Subsystem Component Diagram

3.2 Dependencies

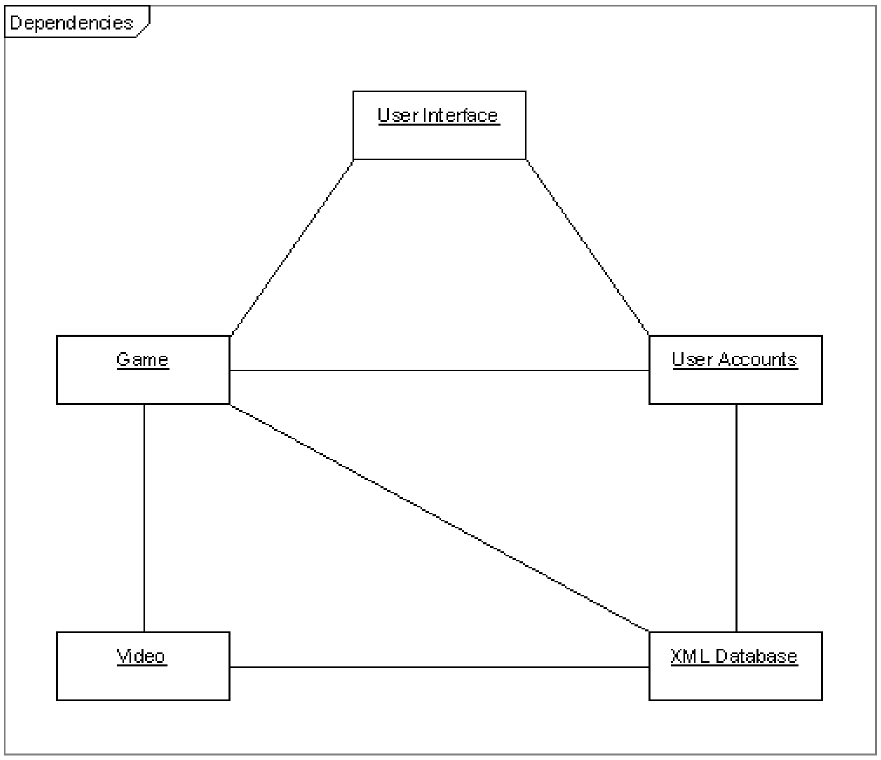

Fig 3.2: Dependencies Diagram

4 Features to be tested / not to be tested

This section covers the functional aspects of testing.

4.1 Features To Be Tested

According to the SRS, the following features of the software shall be tested:
- Register
- Log-in
- Demo
- Tutorial
- Practice
- Recorded score review
- Settings category
- Video category and difficulty
- Keyboard settings
- Video learning session
- Bonus learning session
- Reward
- Score output per current session
- Log-out
- Exit

4.2 Features Not To Be Tested

The following features of the software shall not be tested:
- Resolution of the videos
- Quality of the audio

5 Pass / Fail Criteria

This section covers the pass and fail criteria of a test case.

5.1 Unit Test

For unit testing, all the test cases shall be implemented using the JUnit Test Suite. For each test case, there shall be an expected output and an actual output. If the actual output matches the expected output, the test case shall pass; otherwise, it shall fail.

5.2 Integration Test

For the integration test, all the test cases shall also be implemented using the JUnit Test Suite. Each test case contains a combination of multiple unit test cases. If the actual output matches the expected output, the test case shall pass; otherwise, it shall fail.

5.3 System Test

For the system test, all the test cases shall be created using an Excel sheet. Each test case should contain a title, brief test procedures, and status (this shall be initialized to null before it is tested), which will be either passed, failed or suspended. If a test follows the test procedures correctly, it shall pass; otherwise, it shall fail.

5.4 Performance Test

For the performance test, it is not necessary to create test cases. Each test is compared accordingly to one of the performance requirements specified in the SRS. For each test, there shall be a reasonable range of deviation. (e,g the video clip shall be loaded within 10 seconds, and the deviation is 0.1 seconds). If the outcome of a test is within the range of deviation, the test shall pass; otherwise, it shall fail.

5.5 Installation Test

For installation testing, it is not necessary to create test cases; however, this shall still be noted in the document. The software shall be tested on both Windows and Mac (multiple versions if time allows). If the software can be installed on a test machine without errors and runs smoothly, the test shall pass; otherwise, it shall fail.

6 Approach

This section covers the general approach to the testing process.

6.1 Overview

Testing comprises mainly of two parts: validation and verification. Validation is determined by whether or not the team has built the program according to the client's expectations and requirements.

The majority of validation testing will be covered by the presentation to the client as well as allowing the client or any prospective users to test the product independently.

For the verification aspect, testing will primarily involve system tests done by members of the team, that is whether or not the program conforms to the specifications and requirements outlined in the SRS and the SDS.

The verification testing for the ACSS will be done through utilization of the JUnit module in Eclipse. This method was chosen for consistency and reliability. This method also yields immediate and practical results that will help with identification of bugs and errors.

Specifically for the integration testing, JUnit will be used to test the interactions between the components in the overall system. The team will design test cases using both white and black box methods to make sure that the overall system functions properly and accurately.

For the other testing phases (system, performance, installation) the team will simply generate a variety of pass or fail test cases to assess that aspect of the software. This empirical approach will allow the team to make sure the final product not only functions properly and fulfills the design requirements, but also operates consistently and efficiently enough to be used as a viable commercial product.

6.2 UML Diagrams

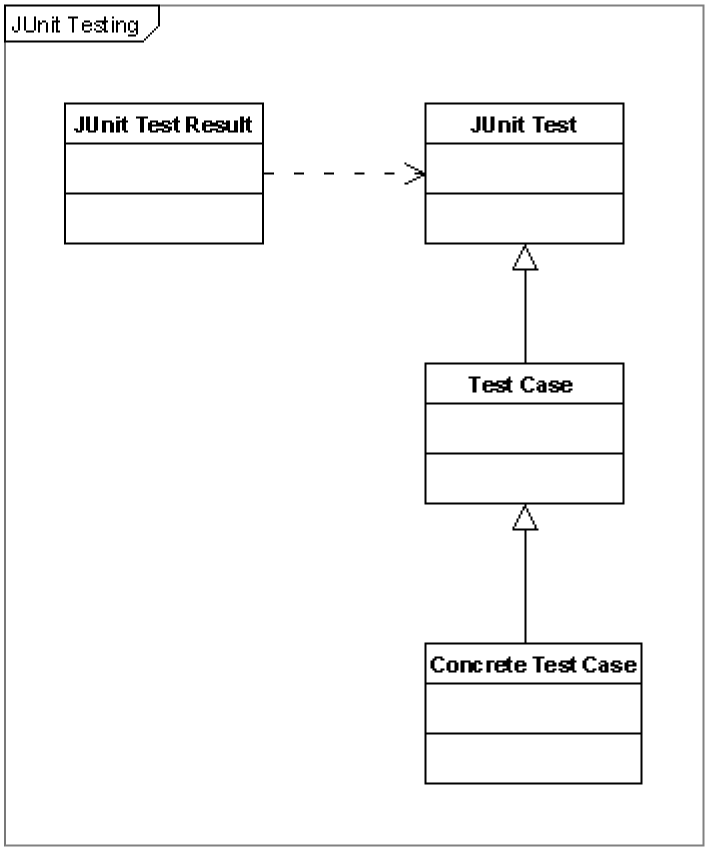

Fig 6.1: JUnit Testing Diagram

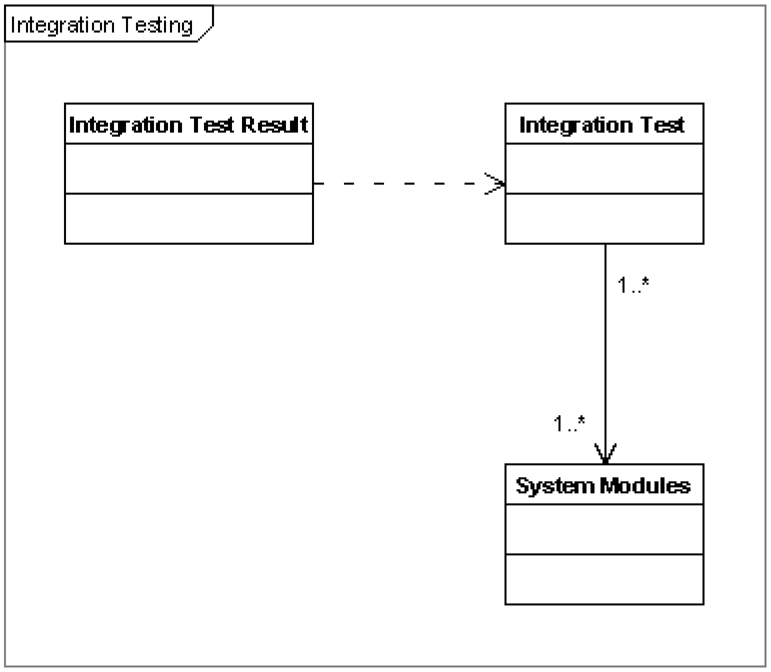

Fig 6.2: Integration Testing Diagram

7 Suspension And Resumption

This section covers the criteria for suspending a test case.

7.1 Suspense criteria

A test case shall be marked as “suspended” only when the precondition test case is failed or suspended. A test case with a failed precondition test case shall not be marked as “failed”.

A test case shall also be marked as “suspended” when the required functionality has not been implemented or finished yet.

7.2 Resumption

A suspended test case shall be executed immediately after the precondition test case has passed or the required functionality has been fixed or finished. If a test case associates with some functionality that will be dropped, this test case shall be deleted from the Test Case Specification document.

8 Test Materials

This section covers the required materials and resources needed during the test phase.

8.1 Hardware

- Macintosh machines with OS 10.1, 10.2, 10.3, 10.4 installed
- Windows machines with Windows 98, 2000, ME, XP installed

8.2 Software

- Eclipse 3.2
- jdk-1.5
- EMF 2.2
- GEF 3.2
- JMF 2.1
- JFreeChart 1.0.1

8.3 Facilities

- UBC computer lab X250
- UBC fine art Mac lab M17-room 8

8.4 Testers

- Team members
- Client
- Sample Users

9 Test Cases

This section covers the test case specification of each test case.

9.1 Unit Test

Please refer to Section 1 of the Test Case Specification Document.

9.2 Integration Test

Please refer to Section 2 of the Test Case Specification Document.

9.3 System Test

Please refer to Section 3 of the Test Case Specification Document.

9.4 Performance Test

Requirement	Result
1. The system shall support only one terminal	**PASS**
2. The system shall support only one simultaneous user on each machine	**PASS**
3. The system shall run on both Mac (OS 10.*) and Windows (Windows 98 and above) machines with at least 128 MB of memory and a CD or DVD ROM.	**PASS** (only on Mac 10.3-10.4 & Windows XP)
4. The video clips attached to the system shall play on both QuickTime and Windows Media Player with the necessary codec(s)	**PASS**
5. The system shall be loaded and functioning within 15 seconds 95% of the time after starting the application	**PASS**
6. Each account shall be stored and activated within 5 seconds after creation	**PASS**
7. Each video clip (less than 5 minutes) shall be loaded completely within 30 seconds 95% of the time	**PASS**
8. Each user input during the session shall be responded to (a simple sign which indicates if the timing of interruption is	**PASS**

correct or incorrect) within 3 seconds 98% of the time	
9. Each sessional grade report shall be generated within 5 seconds at the end of each session 95% of the time	**PASS**

9.5 Installation Test

Operating System	Result
Windows 98	**SUSPENDED**
Windows 2000	**SUSPENDED**
Windows ME	**SUSPENDED**
Windows XP	**PASS**
Mac 10.1	**SUSPENDED**
Mac 10.2	**SUSPENDED**
Mac 10.3	**PASS**
Mac 10.4	**PASS**

10 Test Schedule

This section covers the responsibilities of each member on the team, the risks that might occur during the test phase, and the test schedule that the team is supposed to follow.

10.1 Responsibilities

Task	Member
Bug Fixing	George Firican, Ian Cook
Unit/Integration Test Case Creation	Jeffrey Qua, Dan Liu, Ian Cook
System/Performance Installation Test Case Creation	Michael Tien
Unit/Integration/System Test Execution	All
Performance Installation Test Case	George Firican, Ian Cook, Michael Tien
Test Plan Management	Michael Tien

10.2 Risks

10.2.1 Java

The program was originally implemented for Java compiler 5.0, but this only works with Java plug-in 5.0 that is only supported on Mac 10.4. For Mac OS 10.3 and lower, the program needs to be compiled with Java compiler 1.4 in order to be successfully executed. An update also has to be made for both versions on a regular basis and make sure there are no differences in the functionality.

10.2.2 Time

Every member has heavy loads of work; time will become an issue. Hard deadlines are required. Make sure every member is doing his work.

10.2.3 Videos

Make sure that all videos work on the required systems.

10.2.4 Testing

Make sure that no test cases are missed by mistake.

10.3 Test Schedule

Task	Starting Date	Due Date
Unit Test	Thu Mar 9th, 2006	Sat Mar 11th, 2006
Integration Test	Sun Mar 12th, 2006	Tue Mar 14th, 2006
System Test – Test Case Creation	Wed Mar 15th, 2006	Thu Mar 16th, 2006
System Test – Functional Test	Fri Mar 17th, 2006	Sat Mar 18th, 2006
System Test – Dry Run	Sun Mar 19th, 2006	Mon Mar 20th, 2006
System Test – Second Run	Tue Mar 21st, 2006	Wed Mar 22nd, 2006
System Test – Regression Test	Thu Mar 23rd, 2006	Thu Mar 23rd, 2006
Performance Test	Fri Mar 24th, 2006	Sat Mar 25th, 2006
Installation Test	Sun Mar 26th, 2006	Sun Mar 26th, 2006

10.3.1 Unit test (3 days)

This shall be done with the JUnit Test Suite. Each function in the code shall be tested with an expected output. A JUnit test function would return true if the actual output matches the expected output; otherwise, it will return false.

10.3.2 Integration test (3 days)

The "**Bottom-Up**" testing model shall be used to achieve the integration test.

10.3.3 System test (9 days)

The tests shall be done on both Windows and Mac machines. Since there is only very limited amount of time, tests might only be run on Windows 98 (or 2000) and Mac 10.2 (the oldest edition of OS requested by the client). If time allows, tests shall be run on as many versions of OS as possible. The system test phase is broken down into the follow components:

10.3.3.1 Test cases creation (2 days)

Test cases shall be created with simple description and test procedures. This shall be documented with Microsoft Word and uploaded to the web so that each team member will be able to mark the test result directly in the file.

10.3.3.2 Functional test (1-2 days)

Each team member will be assigned to test certain operations. Please refer to section 4.1 Features To Be Tested.

10.3.3.3 Dry run (1-2 days)

This is the first run of the test. Every test case shall be executed, and the bugs found shall be reported and fixed.

10.3.3.4 Second run (1-2 days)

This is the second run of the test. Every test case shall be executed, and the bugs found shall be reported and fixed.

10.3.3.5 Regression test (1 day)

This is the final run of the test. Only the test cases that failed during the second run shall be executed, the bugs found shall be reported and fixed. If there are any bugs that cannot be fixed after the regression test, they shall be reported in the user document.

10.3.4 Performance test (1-2 days)

This shall be done in order to check if the performance of the system fulfills the performance requirements specified in the SRS document.

10.3.5 Installation test (1 day)

This shall be done in order to determine if the software system can be successfully installed and run on various platforms specified by the client.

The software code will be reviewed once the implementation is completed. The review will consist of evaluation of subsystems and functions code to ensure correct implementation. The code will also be subjected to a requirements check to ensure all requirements are met.

11 Bug Tracking

This section covers the procedures in properly documenting and identifying bugs to help localize the problem and to record progress on fixing these problems.

11.1 Tracking Method

If there is any bug found during the testing process, a “bug track” must be issued with the following parameters: issue number, name of the tester, date, title, a brief description of the bug, test environment (i.e. what kind of OS and version), and status (active, suspended, and closed). All the bug tracks shall be archived on the CVS server that will be available to all the team members. If a bug has been fixed and tested, the bug track must be updated with the new updated test date; name of the tester, and the status of it shall be changed to “closed”.

Bug Track	
Issue Number:	This value should be unique for each bug track
Title:	An appropriate title which reflects the bug well
Tester:	Name of the test who found the bug
Date:	Date of the day when the bug track is issued
Description	
Test Environment:	Version of OS, Java plug-in
Description:	A brief description of the bug
Steps to Reproduce	
Brief steps to reproduce the bug	
Condition	
Status:	(active, suspended, closed) – this is initialized to active
Fixed by:	Name of the developer who fixed the bug
Resolution:	(implemented, function removed, no longer exists …etc)