dbViZ
Master Test Plan

Version 1.1

Revision History

Date	Version	Description	Author
11/Dec/02	1.0	First draft of the Master Test Plan	David Hampshire
03/02/03	1.01	Second Draft of the Master Test Plan	Saurabh Gandotra

Table of Contents

Master Test Plan

1. Introduction

1.1 Purpose

The purpose of this Test Plan is to gather all of the information necessary to plan and control the test effort for testing dbViZ. It describes the approach to testing the software, and will be the top-level plan used by testers to direct the test effort.

This *Test Plan* for dbViZ supports the following objectives:

- Identifies the items that should be targeted by the tests.
- Identifies the motivation for and ideas behind the test areas to be covered.
- Outlines the testing approach that will be used.
- Identifies the required resources and provides an estimate of the test efforts.
- List the deliverable elements of the test activities.

1.2 Scope

This test plan describes the unit, subsystem integration, and system level tests that will be performed on components of dbViZ.

It is assumed that prior to testing each subsystem to be tested will have undergone an informal peer review and only code that has successfully passed a peer review will be tested.

Unit tests will be done through test driver program that perform boundary checking and basic black box testing.

Subsystem tests will verify defined interfaces between the following packages:

1. Schema Package
2. GUI
3. Main Application
4. Schema Import

The external interfaces to be tested:

1. Schema parser.
2. Database interfaces that are supported (Oracle, OgreSQL?)

The following performance measurements will be tested:

1. Response time for creation of an ER diagram table.
2. Response time for adding a column to an SQL query.
3. Response time for importing the baseline schema file.

1.3 Intended Audience

This test plan is being written for two audiences. For students writing dbViZ this test plan will serve as a guide. The professor and TAs will use this test plan as a metric to determine how well dbViZ has been tested.

1.4 References

dbViZ Glossary.

dbViZ Software Development Plan.

2. Evaluation Mission and Test Motivation

2.1 Background

This test plan was developed as a guide to the dbViZ testing effort.

2.2 Evaluation Mission

Testing will be done primarily to verify dbViZ satisfies requirements set forth by its use cases. Secondarily testing will be done to verify alpha product quality.

2.3 Test Motivators

Testing will be motivated by the desire to get a good grade in cs327 and to ensure that functional and non-functional requirements are met. Also, standard documentation and coding procedures are being followed

3. Target Test Items

The following have been identified as targets for testing:

Java Platforms:

- Java 1.4.1

Databases:

- Oracle
- TBD

Operating Systems:

- Windows 98/ME
- Windows NT/2000/XP

4. Outline of Planned Tests

4.1 Outline of Test Inclusions

The following testing will be done on dbViZ:

- Smoke Testing
- Functional testing.
- Platform testing.
- Data integrity testing.
- UI testing.
- Installation testing.

4.2 Outline of Other Candidates for Potential Inclusion

The following tests may be worth pursuing if resources are available during the course of the class:

- Performance profile testing.
- Stress testing.

4.3 Outline of Test Exclusions

The following tests will not be performed on dbViZ as they do not fall under project scope:

- Security testing.
- Business cycle testing.
- Failure testing.
- Load Testing

5. Test Approach

Where possible tests will be automated by using the JUnit test frameworks. Before any code is checked into CVS it must pass the appropriate unit tests. Any bugs found should be posted to the project website with pertinent information, which should include who found it, how, a description of the problem, and eventually who fixed it and when. Also, re-testing of the code will be done to make sure that defect HAS been fixed and there no bugs produced due to change in code.

5.1 Testing Techniques and Types

5.1.1 Data Integrity Testing

Data integrity testing will be done to ensure schema data is not corrupted by the parser or internal data structures. This testing will be done independent of the User Interface in a white box fashion.

Technique Objective:	Verify the data integrity of the imported schema independent of the UI.
Technique:	Verify the parser retrieves correct schema data from the database and from a schema file. Load test schemas into the schema package and verify the data contents.
Oracles:	The sample SQL files located on the dbViZ website.
Required Tools:	Java compiler and debugger., WinRunner
Success Criteria:	All schema elements can be stored and retried with no data loss.

5.1.2 Function Testing

Functional testing will be performed to verify all functional requirements are met successfully. This will be accomplished through black box testing.

Technique Objective:	Verify system functional requirements.
Technique:	Verify the requirements put forth in the use cases are met.
Oracles:	dbViZ use cases.
Required Tools:	x-Unit base configuration imager and restorer (TBD) backup and recovery tools (TBD) installation-monitoring tools (registry, hard disk, CPU, memory, and so forth) (TBD) Data-generation tools (TBD)
Success Criteria:	All of the following are successfully tested: • all key use-case scenarios • all key features

5.1.3 User Interface Testing

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the UI provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Technique Objective:	Exercise the following to observe and log standards conformance and target behavior: Navigation through the target-of-test reflecting business functions and requirements, including window-to-window, field-to- field, and use of access methods (tab keys, mouse movements, accelerator keys). Window objects and characteristics can be exercised–such as menus, size, position, state, and focus.
Technique:	Create or modify tests for each window to verify proper navigation and object states for each application window and object.
Oracles:	The tester will verify proper functionality based on requirements.
Required Tools:	WinRunner, Rational TestFactory, or Rational Visual Test if available.
Success Criteria:	All window objects can be exercised, proper navigation through test target, and test target acts as expected.

5.1.4 Performance Profiling

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviors as a function of conditions such as workload or hardware configurations.

Technique Objective:	Determine test target's behavior under: • normal anticipated workload • anticipated worst-case workload
Technique:	Import schema from a database. Import schema from a file. Add a table to the ER diagram. Remove a table from the ER diagram. Add a column to a SQL query. Remove a column from a SQL query.
Oracles:	Software timers.

Required Tools:	an application performance profiling tool, such as Rational Quantify , Load Runner -- installation-monitoring tools (registry, hard disk, CPU, memory, and so on (TBD) resource-constraining tools (TBD, but possibly Canned Heat)
Success Criteria:	The test completes in the time indicated by the use cases.

5.1.5 *Stress Testing*

Stress testing is a type of performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources, like database locks or network bandwidth, although some of these tests are usually addressed under functional and load testing.

Technique Objective:	Exercise the target-of-test functions under the following stress conditions to observe and log target behavior that identifies and documents the conditions under which the system **fails** to continue functioning properly • multiple users performing the same transactions against the same data or accounts (database locks) • running on a system with below recommended requirements (memory and or CPU)
Technique:	• To test limited resources, tests should be run on a single machine, and RAM and persistent storage space on the server should be reduced or limited. • Make queries against a locked database, or a database under a heavy load.
Oracles:	Tester determination.
Required Tools:	Load Runner, Transaction Load Scheduling and control tool installation-monitoring tools (registry, hard disk, CPU, memory, and so on) resource-constraining tools (for example, Canned Heat)
Success Criteria:	dbViZ gracefully deals with too few resources, or the inability to access the database in a reasonable amount of time.

5.1.6 Volume Testing

Volume testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database, and would check that the software behaved normally and produced the correct report.

Technique Objective:	Exercise dbViZ under the following high volume scenarios to observe and log target behavior: • Maximum database size supported
Technique:	• Read in the schema to an abnormally large database. The schema should be read directly from the database and from a schema file.
Oracles:	Tester.
Success Criteria:	The schema is successfully parsed into dbViZ and displayed correctly on the GUI.

5.1.7 Configuration Testing

Configuration testing verifies the test-target operates correctly under different software configurations and interacting with different software.

Technique Objective:	Verify the test-target functions correctly on different platforms and under different configurations.
Technique:	Exercise unrelated software on the same platform as the test-target to verify there are no side effects.
Oracles:	Test-target behavior.
Required Tools:	base configuration imager and restore (TBD) installation monitoring tools (registry, hard disk, CPU, memory, and so on) (TBD)
Success Criteria:	The test-target behaves as expected and the non test-target software also behaves as expected.

5.1.8 Installation Testing

Installation testing has two purposes. The first is to ensure that the software can be installed under different conditions—such as a new installation, an upgrade, and a complete or custom installation—under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify that, once installed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing.

Technique Objective:	Exercise the installation of the target-of-test onto each required hardware configuration under the following conditions to observe and log installation behavior and configuration state changes: new installation: a new machine, never installed previously with dbViZ update: a machine previously installed dbViZ, same version error case: a new machine, never installed previously with dbViZ without a JVM installed. error case: a new machine, never installed previously with dbViZ with the incorrect JVM version installed.
Technique:	Install to a new machine and verify basic functionality. Install to a previously installed machine and verify basic functionality. Install to a new machine without a JVM. Install to a new machine with an incorrect version of a JVM.
Oracles:	Basic functionality tests pass Correct error handling occurs.
Required Tools:	The technique requires the following tools: base configuration imager and restorer
Success Criteria:	The technique supports the testing of the installation of the developed product in one or more installation configurations.

6. Entry and Exit Criteria

6.1 Test Plan

6.1.1 Test Plan Entry Criteria

Package coding is complete and the code has been informally reviewed by the team.

6.1.2 Test Plan Exit Criteria

Either the second semester ends or all of the use case requirements have been verified.

6.1.3 Suspension and Resumption Criteria

Testing will be suspended on critical design flaws that will require a package, package interface redesigned. Testing will resume when the coding is complete and code is reviewed successfully.

6.2 Test Cycles

6.2.1 Test Cycle Entry Criteria

TBD

6.2.2 Test Cycle Exit Criteria

All tests specified at the start of the iteration have completed successfully, or the end of the second semester occurs.

7. Deliverables

The following artifacts will be testing deliverables, available to the stakeholders.

7.1 Test Evaluation Summaries

These summaries will outline the tests conducted and their results.

7.2 Incident Logs and Change Requests

Incident log entries will be made for all bugs found during testing. The log will be used to track the status of the bugs.

Any modifications to the requirements must be done through change requests, which will ensure the proposed change is fully reviewed before being incorporated into dbViZ.

8. Testing Workflow

See the Test Plan section of the Development Case.

TBD in a later version (For Master Test Plans, we recommend avoiding detailed task planning, which is often an unproductive effort if done as a front-loaded activity at the beginning of the project. A Master Test Plan might usefully describe the phases and the number of iterations, and give an indication of what types of testing are generally planned for each Phase or Iteration.)

9. Environmental Needs

This section presents the non-human resources required for the **Test Plan**.

9.1 Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Test Plan*.

System Resources		
Resource	**Quantity**	**Name and Type**
Database Server	1	TBD
Network or Subnet		TBD
Server Name		TBD
Database Name		TBD
Test Development PCs	TBD	TBD

9.2 Base Software Elements in the Test Environment

The following base software elements are required in the test environment for this *Test Plan*.

[Note: Add or delete items as appropriate.]

Software Element Name	Version	Type and Other Notes
NT Workstation	Service Pack 6	Operating System
Windows 2000		Operating System
Windows XP		Operating System
Java Virtual Machine	1.4.1	JVM

9.3 Productivity and Support Tools

The following tools will be employed to support the test process for this *Test Plan*.

Tool Category or Type	Tool Brand Name	Vendor or In-house	Version
Test Management	TBD		
Defect Tracking	Project Website	In-house	
DBMS tools			

10. Responsibilities, Staffing, and Training Needs

This section outlines the personnel necessary to successfully test dbViZ. Since staff size is fixed these number may change.

10.1 People and Roles

This table shows the staffing assumptions for the test effort.

Human Resources		
Role	**Minimum Resources Recommended** **(number of full-time roles allocated)**	**Specific Responsibilities or Comments**
Test Analyst	1	Identifies and defines the specific tests to be conducted. Responsibilities include: • identify test ideas • define test details • determine test results • document change requests • evaluate product quality
Test Designer	1	Defines the technical approach to the implementation of the test effort. Responsibilities include: • define test approach • define test automation architecture • verify test techniques • define testability elements • structure test implementation

Human Resources		
Role	**Minimum Resources Recommended** **(number of full-time roles allocated)**	**Specific Responsibilities or Comments**
Tester	2	Implements and executes the tests. Responsibilities include: • implement tests and test suites • execute test suites • log results • analyze and recover from test failures • document incidents
Database Administrator, Database Manager	1	Ensures test data (database) environment and assets are managed and maintained. Responsibilities include: • support the administration of test data and test beds (database).
Designer	1	Identifies and defines the operations, attributes, and associations of the test classes. Responsibilities include: • defines the test classes required to support testability requirements as defined by the test team
Implementer	2	Implements and unit tests the test classes and test packages. Responsibilities include: • creates the test components required to support testability requirements as defined by the designer

10.2 Staffing and Training Needs

This section outlines how to approach staffing and training the test roles for the project.

Staffing is fixed for the duration of this project. It is likely most of the staff will assume some testing role.

11. Iteration Milestones

TBD in a later version of this document.

Milestone	Planned Start Date	Actual Start Date	Planned End Date	Actual End Date
Iteration Plan agreed				

Milestone	Planned Start Date	Actual Start Date	Planned End Date	Actual End Date
Iteration starts				
Requirements baselined				
Architecture baselined				
User Interface baselined				
First Build delivered to test				
First Build accepted into test				
First Build test cycle finishes				
[Build Two will not be tested]				
Third Build delivered to test				
Third Build accepted into test				
Third Build test cycle finishes				
Fourth Build delivered to test				
Fourth Build accepted into test				
Iteration Assessment review				
Iteration ends				

12. Risks, Dependencies, Assumptions, and Constraints

[List any risks that may affect the successful execution of this **Test Plan**, and identify mitigation and contingency strategies for each risk. Also indicate a relative ranking for both the likelihood of occurrence and the impact if the risk is realized.]

Risk	Mitigation Strategy	Contingency (Risk is realized)
Inadequate test cases.	The tester(s) will strive to ensure adequate test cases are generated.	• Redefine test data • Review Test Plan and modify • Rewrite test cases and have them reviewed by the team.
Second semester ends before testing is complete.	Work like mad.	• End Game – hope for a good grade.

13. Management Process and Procedures

13.1 Problem Reporting, Escalation, and Issue Resolution

Problems will be reported to the team by e-mail. Bugs will be listed on the project page with the developer responsible for fixing each bug. Each bug will be given a priority, which will determine when it is addressed in the current iteration.

13.2 Approval and Signoff

The team manager must approve the initial test plan. As tests are successfully completed the testers will sign off use case requirements.