Testing Process Case Study

Amber Hardy

CSE 7314 Course Project

Testing Process

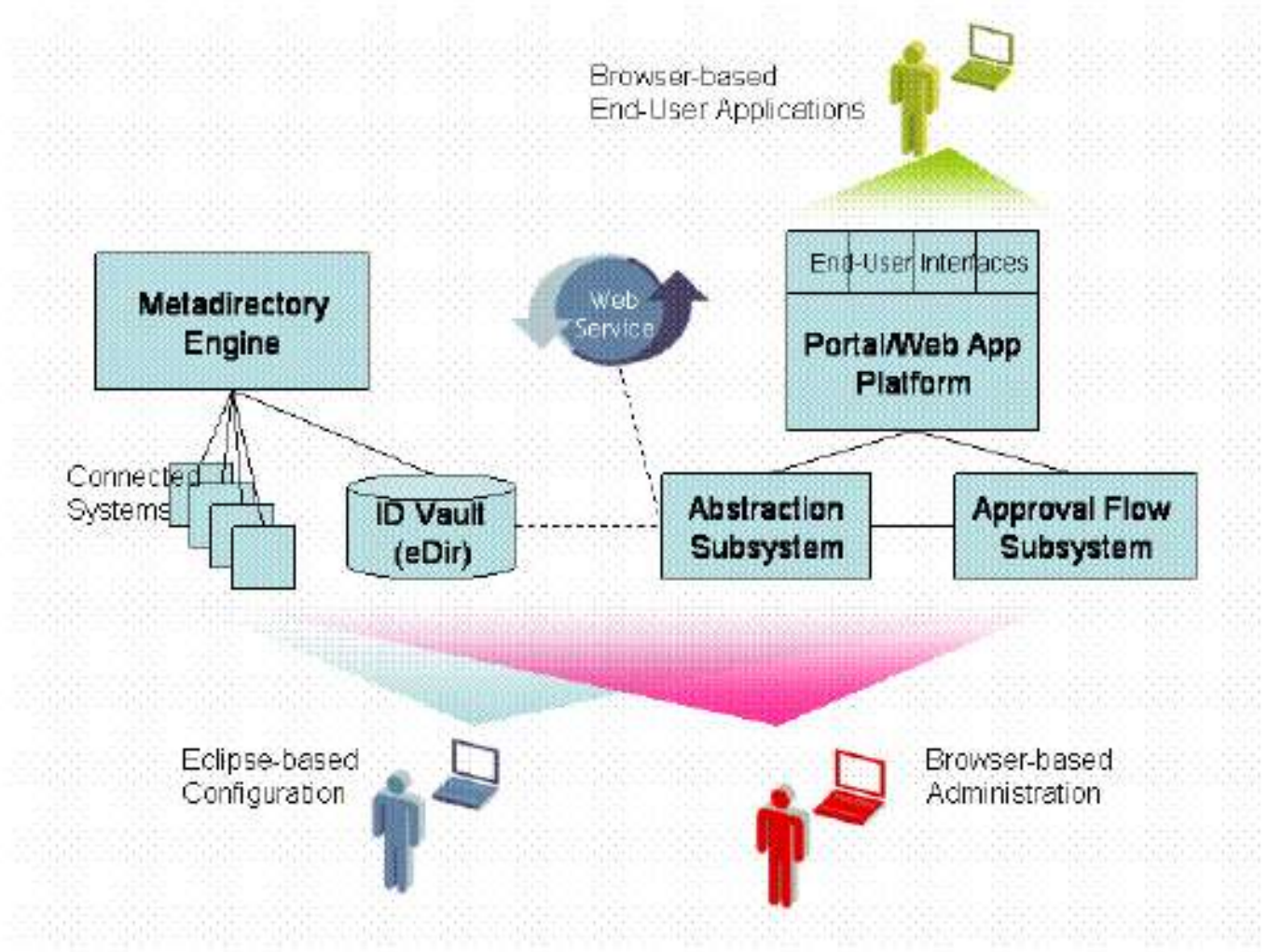
This slideshow is a case study for the testing of Novell Audit logging with Novell Identity Manager Provisioning, a new component being tested for the first time. The following steps are covered. The focus will be on testing techniques used.

- Planning and Preparation
- Execution
- Analysis and Follow-up

Planning and Preparation

Product Overview

Novell Identity Manager System Overview



Product Overview

The Novell Identity Manager Provisioning component contains the following functionality accessible through a user application web portal:

- Approval Flow for Provisioning
- Enhanced White Pages
- Enhanced Organization Chart
- User Search
- Password Management
- Lightweight User Administration

Product Overview

The following events for Novell Identity Manager Provisioning should be logged to Novell Audit:

Delete_Entity
Update_Entity

Change_Password_Failure
Change_Password_Success

Forgot_Password_Change_Failure
Forgot_Password_Change_Success

Search_Request
Search_Saved

Create\Create_Entity

User_Message

Workflow_Error
Workflow_Started
Workflow_Forwarded
Workflow_Reassigned
Workflow_Approved
Workflow_Refused
Workflow_Ended
Workflow_Claimed
Workflow_Unclaimed
Workflow_Denied
Workflow_Completed
Workflow_Timedout

Provision_Error
Provision_Submitted
Provision_Success
Provision_Failure
Provision_Granted
Provision_Revoked
Provision_Retracted

Create_Proxy_Definition_Success
Create_Proxy_Definition_Failure
Update_Proxy_Definition_Success
Update_Proxy_Definition_Failure
Delete_Proxy_Definition_Success
Delete_Proxy_Definition_Failure

Create_Delegatee_Definition_Success
Create_Delegatee_Definition_Failure
Update_Delegatee_Definition_Success
Update_Delegatee_Definition_Failure
Delete_Delegatee_Definition_Success
Delete_Delegatee_Definition_Failure

Create_Availability_Success
Create_Availability_Failure
Delete_Availability_Success
Delete_Availability_Failure

Product Overview

Novell Audit should log the following data for each event:

- User Application server IP
- Timestamp
- Initiator ID
- Recipient
- Event
- Activity
- Process ID
- Secondary User

Testing Goals

- Verify all Identity Manager Provisioning events are properly logged to Novell Audit.
- Accuracy must be as close to 100% as possible, because the data being logged will be used for auditing purposes (such as for Sarbanes-Oxley).
- Coverage-based testing should be used, since for auditing purposes the rarely used events might be the most critical to have logged. Users can't always guess what may be audited.
- Reports must be generated for easy interpretation of tests.
- Tests must be easily re-run for regression purposes.

Testing Issues

- Logging for the events will not occur until the events themselves occur. Time is too short to learn how to create each event and create tests for each event.
- To verify an event is logged, the Audit database has to be searched for the exact matching entry. This can be time consuming as well.
- For easy regression, some portion of the tests will need to be automated.

Testing Solution

- Use automation tests that other testers have created to exercise the various events.
- Create automation to check the database to see if the events were logged properly.
- A black-box testing approach, of sorts. Run the event test script without worrying about the details of the test script.

Finite State Machine

The following common test environment is used as a Finite State Machine for running all test scripts.

Identity Vault	Audit Server	User App Portal	Client
			
IDM : eDirectory IDM iManager **AUDIT:** Audit plug-in for iManager **IDM LOGGING:** Import logging schema file w/ iManager	eDirectory MySQL Audit Engine MySQL DB for Audit	JBoss User App Logging Platform Enable logging in User App	Browser

Test cases

- List-based testing is used. Testcases are generated so each event in the list is logged at least once.
- Partition-based testing is used. Testcases are generated so at least one scenario that will trigger a particular event will occur for each partition (partition = event in the list).
- Coverage-based testing is used. Testcases are generated so each event is covered equally, although some will be covered more than once for convenience in running the tests.

Test cases

- Determine which test scripts created by other testers are needed to trigger which logging events.
- Determine how many times the event will be logged in the course of the test script.
- Create chart....

Event	Script Called	# Times Logged
Delete_Entity	CreateUser	1
Create_Entity	CreateUser	1
Update_Entity	DetailEditVerify	1
Change_Password_Failure	ChangePassword	1
Change_Password_Success	ChangePassword	1
Forgot_Password_Change_Failure	ForgotPassword	1
Forgot_Password_Change_Success	ForgotPassword	1
Search_Request	Search	4
Search_Saved	Search	1

Test cases

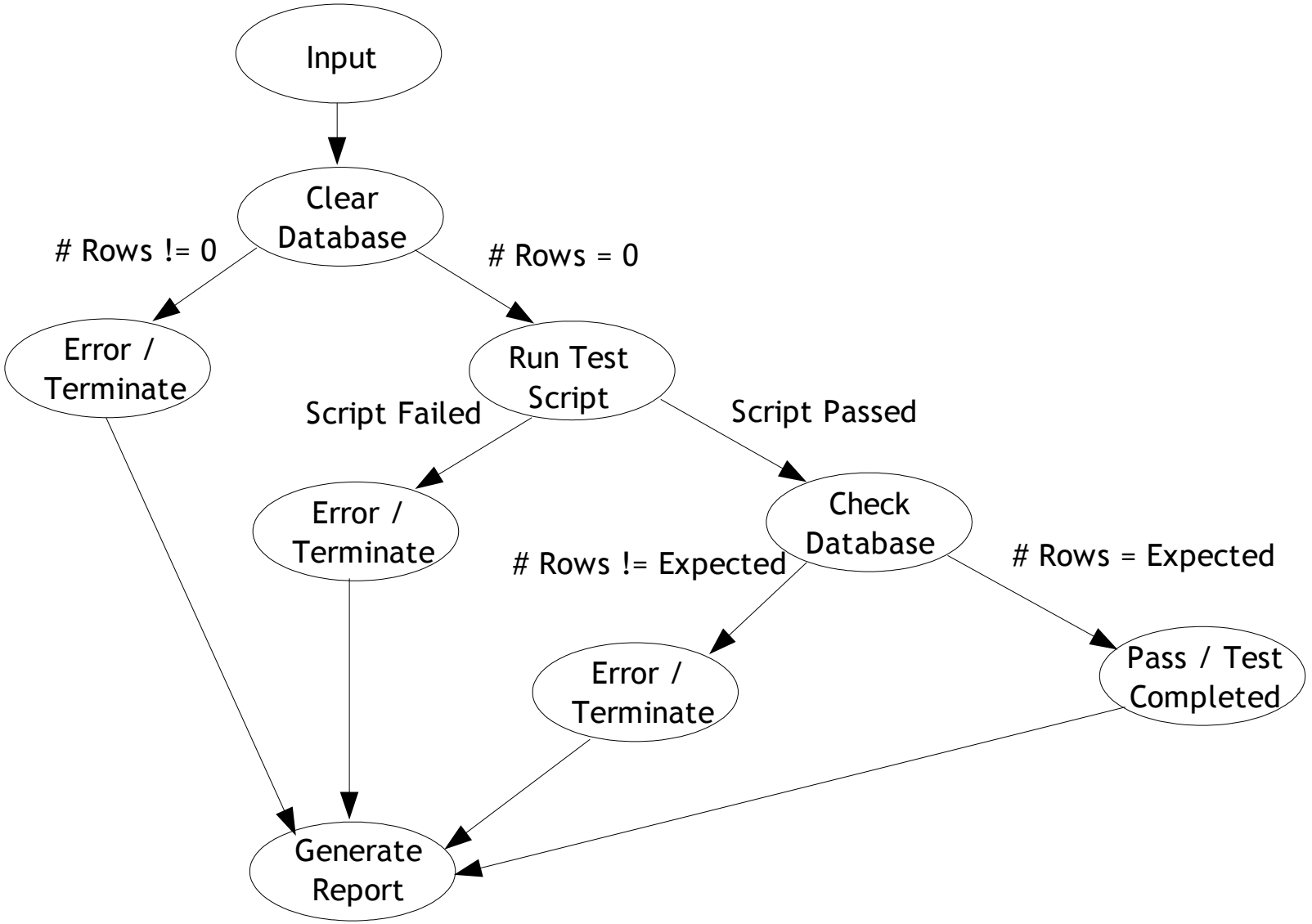
- More of chart...

Event	Script Called	# Times Logged
Workflow_Error	MyWorkflow	1
Workflow_Started	MyWorkflow	2
Workflow_Forwarded	MyWorkflow	6
Workflow_Reassigned	MyWorkflow	1
Workflow_Approved	MyWorkflow	2
Workflow_Refused	MyWorkflow	1
Workflow_Ended	MyWorkflow	2
Workflow_Claimed	MyWorkflow	2
Workflow_Unclaimed	MyWorkflow	1
Workflow_Denied	MyWorkflow	1
Workflow_Completed	MyWorkflow	1
Workflow_Timedout	MyWorkflow	1
User_Message	MyWorkflow	1
Provision_Error	Provisioning	1
Provision_Submitted	Provisioning	4
Provision_Success	Provisioning	1
Provision_Failure	Provisioning	1
Provision_Granted	Provisioning	2
Provision_Revoked	Provisioning	1
Workflow_Retracted	Provisioning	1
Create_Proxy_Definition_Success	MyProxyAssignments	1
Create_Proxy_Definition_Failure	MyProxyAssignments	1
Update_Proxy_Definition_Success	MyProxyAssignments	1
Update_Proxy_Definition_Failure	MyProxyAssignments	1
Delete_Proxy_Definition_Success	MyProxyAssignments	1
Delete_Proxy_Definition_Failure	MyProxyAssignments	1
Create_Delegatee_Definition_Success	MyDelegateAssign	1
Create_Delegatee_Definition_Failure	MyDelegateAssign	1
Update_Delegatee_Definition_Success	MyDelegateAssign	1
Update_Delegatee_Definition_Failure	MyDelegateAssign	1
Delete_Delegatee_Definition_Success	MyDelegateAssign	1
Delete_Delegatee_Definition_Failure	MyDelegateAssign	1
Create_Availability_Success	EditAvailability	1
Create_Availability_Failure	EditAvailability	1
Delete_Availability_Success	EditAvailability	1
Delete_Availability_Failure	EditAvailability	1

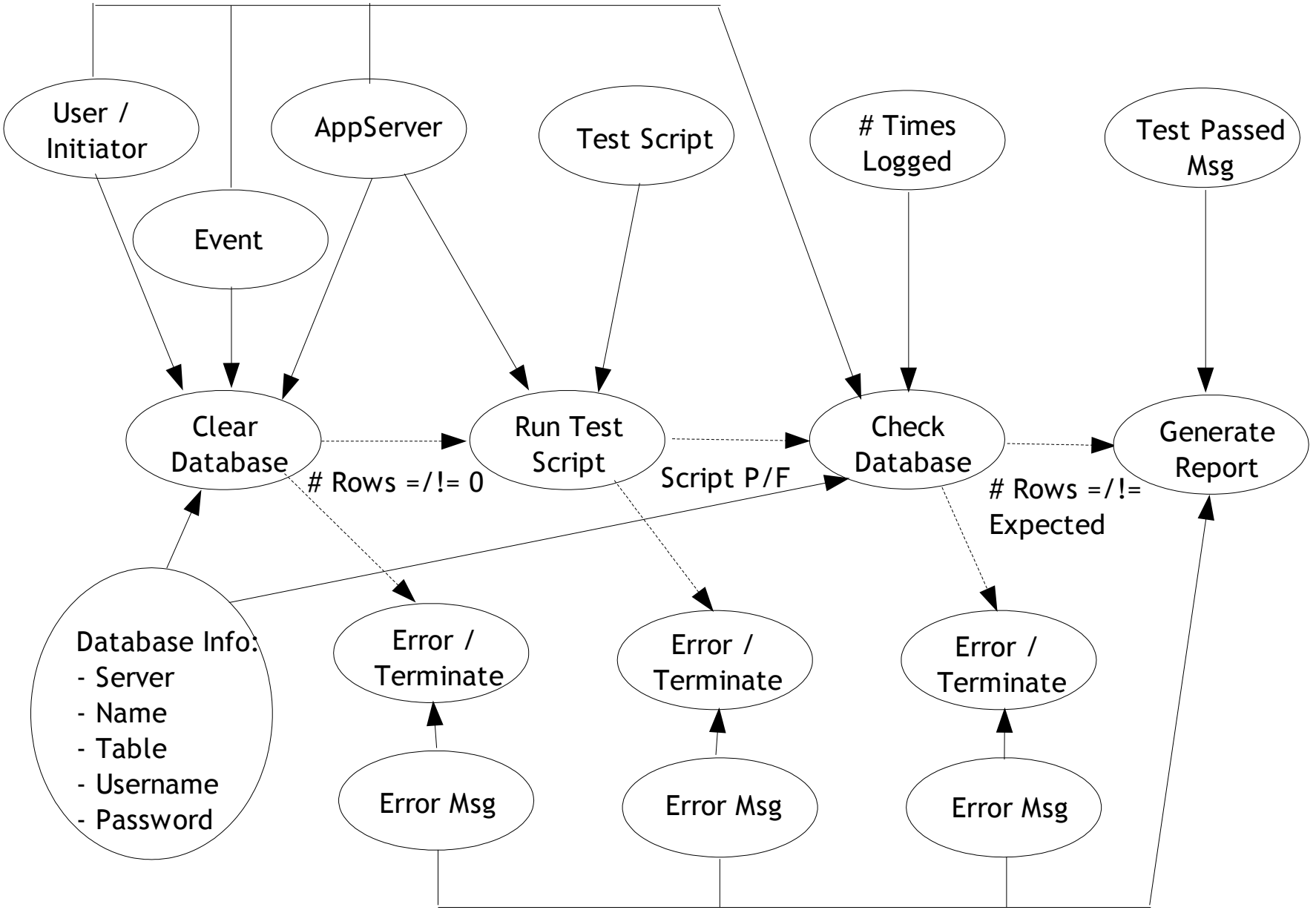
Testing Model

- To check the Audit database, we will check that the User Application server IP, Initiator ID, and Event columns match when checking entries. For simplicity, we will not check that other data columns match. Manual spot checking will verify other data columns.
- To ensure the entries did not exist in the database before the test is run, we will first clear the Audit database of all entries where the Application server IP, Initiator ID, and Event columns match those for the specific test.
- Because the nature of our testing views the application itself as a black box, and we are not concerned with what goes on inside, the test model diagrams model the test rather than the application being tested. Following are the CFG and DDG for our tests.

Control Flow Diagram



Data Flow Diagram



Test Execution

Testing Procedure

- New build is created once per week.
- Smoke tests are then run to validate build.
- Audit logging tests are then run.
- Generated reports are posted on the web.
- Defects are entered for any new bugs.
- Old bugs that are now fixed are marked Verified and Closed.
- Test status is updated.

Testing Tools

- Test Requirements: Test Director
- Test Status: Test Director and Wiki
- Test Scripts: Rational Functional Tester (RFT) using Java
- Test Script Repository: Clearcase
- Testing Reports: RFT, JUnit, and Ant
- Bug Tracking: Bugzilla

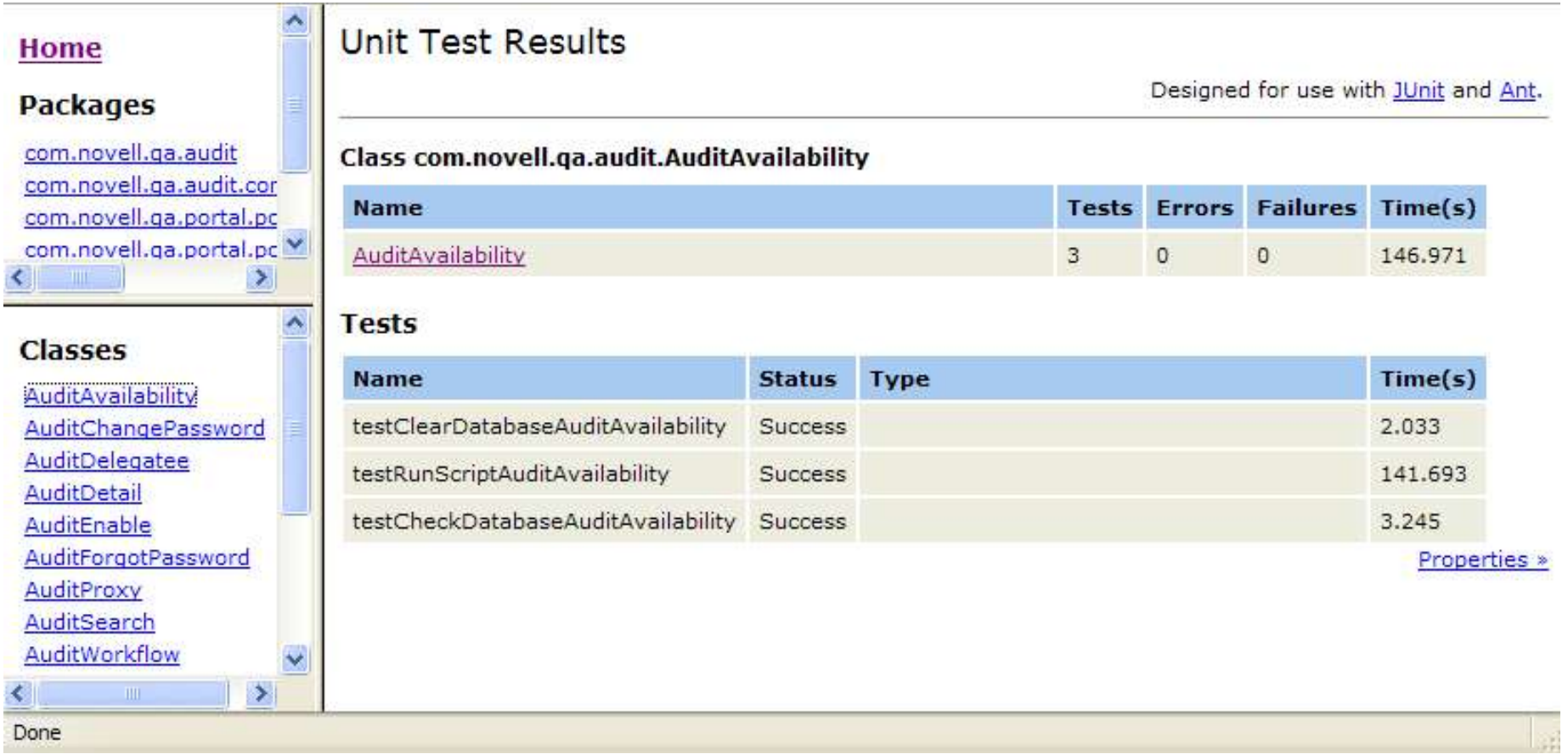
Analysis and Follow-up

When to Stop

- All tests must pass at 100%. Because logged data is used for auditing purposes, we do not have room for any degree of error.
- Errors missed in the code would be due to ill-formed tests, but any failures detected by the current tests must be resolved.

Testing Report

Sample Test Report showing tests pass at 100%



Post-Analysis

- All tests passed.
- Accuracy depends on quality of test. May still be errors that exist that the tests were not designed to catch.
- Suggestions for improvement.
  - Revise tests to check the database using more than just three columns for more detailed checking.