Th3 - Open Source Tools for Test Management

Narayanan C. V., Vice President, Sonata Software Limited
Agenda

- Introduction
- Methodology
- Architectural View
- Test Management Best Practices
- Take Away
Test Management - Introduction

Test Management is a method of organizing Test Assets – Test Requirements, Test Plans, Test Cases, Test Scripts, Test Data, Requirements Traceability and Test Results to enable optimized Test Delivery.
Introduction – Purpose

Challenges

- Prioritization of requirements
- Extent of Test coverage
- Cost Vs Quality
- Aligning to Release Plan
- Environment control

Solution

- Streamlining the process
- Integrated tracking across life cycle
- Optimized Testing based on constraints
- Adoption of Open Source tools

Benefits:

- Helps in managing changing requirements
- Repeatable process
- Effective release cycles and bug tracking
Methodology – [Waterfall / Iterative]
Test Tools Integration Architecture

- Requirement Management Framework
- Automation Testing Framework
- Defect Tracking
- Unit and Build Testing Framework
- Performance Testing Framework
- Security Testing Framework
- SOA Testing Framework

- SONNET Test Data Repository
- Enterprise Service Bus (ESB)
- SONNET Transformation Engine
- SONNET Test Database Engine
- SONNET Reporting Engine
  - Dashboard view of results
  - Result Analysis
- SONNET Risk Based Assessment Framework
Deployment View

- Use case
- Scenarios

- Test Conditions
- Performance

- Code Coverage
  - Method Signature
  - Build & Deployment
  - Smoke Test

- Functional Automation
  - Business Process Testing
  - Risk Based Testing

- Thread Modeling
  - Security Scanning
  - Report

Test Integration Framework (TIF)

Transformation Engine

Reporting Engine

ENTERPRISE SERVICE BUS

- Test case
- Test Scenarios
- Test Conditions
- RTM

- Reviews
  - Functional / Non functional

- Service Identification
  - Request construction
  - Response validation

- Critical Business Scenario / Transactions
- Load Pattern

SONNET REPORTING ENGINE
  - Dashboard view of results
  - Result Analysis
Requirements and Test Case Management Framework

### Requirements and Test Case Management Framework

#### Requirements and Test Management Process

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>TestLink [Open Source]</td>
</tr>
<tr>
<td>Expected Output</td>
<td>• Centralized repository for Requirements and Test management</td>
</tr>
<tr>
<td></td>
<td>• Requirement Hierarchy</td>
</tr>
<tr>
<td></td>
<td>• Requirements mapping to Test Cases for better coverage</td>
</tr>
<tr>
<td>Integration Process</td>
<td>• Test Management for traceability</td>
</tr>
<tr>
<td></td>
<td>• Defect Management for stability</td>
</tr>
<tr>
<td></td>
<td>• Dashboard for monitoring</td>
</tr>
<tr>
<td></td>
<td>[Requirement ID is the primary key for linking other test assets]</td>
</tr>
<tr>
<td>Benefits</td>
<td>• Collect and organize your requirements</td>
</tr>
<tr>
<td></td>
<td>• Track specific information about individual tests</td>
</tr>
<tr>
<td></td>
<td>• Customize Test Link to fit your requirements and processes.</td>
</tr>
<tr>
<td></td>
<td>• Improved user management (editable)</td>
</tr>
<tr>
<td></td>
<td>• Attachments</td>
</tr>
<tr>
<td></td>
<td>• Enhanced reports</td>
</tr>
<tr>
<td></td>
<td>• Import / Export through XML</td>
</tr>
</tbody>
</table>
Risk-Based Testing

- Defect Management System
- Release Notes Management
- Requirement Management System
- Transformation Engine
- Dashboard
- Algorithm
- Reports
# Test Data Management Framework

**Approach:**
- Data Profiling for all environments
- Masking of confidential information
- Data sampling technique used to slice the production data
- Standard template for data requests including for change requests

## Strategy

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator Data [Open Source] Excel [Customized]</td>
<td></td>
</tr>
</tbody>
</table>

## Expected Output

- Centralized Test Data repository for the following
  - Unit Testing, Functional / Regression Testing, Performance Testing etc.
  - Test Environments like Development, Testing, UAT, Production etc.
  - Coverage includes new requirements, change requests / enhancements

## Integration Process

- Requirements Management for traceability
- Test management for mapping
- Defect Management for defect simulation
- Dashboard for Reporting
  - [Requirement ID is the primary key for linking other test assets]

## Benefits

- Test Data is available on demand
- Versioned enabling to roll back whenever required
- Scripts available to refresh / create / update on need basis
Unit Testing & Build Setup Framework

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Suite of Open Source tools like HttpUnit, Jwebunit, Junit, JunitPerf</td>
</tr>
<tr>
<td>Expected Output</td>
<td>Unit Testing for different types of Testing like Functional, Security and Performance.</td>
</tr>
<tr>
<td>Integration Process</td>
<td>The Test Results of the various Unit Testing frameworks are sent to the Sonnet Transformation Engine which in turn gets consolidated in the Sonnet Reporting Engine and displayed in Dash Board view.</td>
</tr>
<tr>
<td>Benefits</td>
<td>Reusable and efficient</td>
</tr>
</tbody>
</table>
Unit Testing & Build Setup - Benefits

Security Testing
- Reusable Unit Test Case for Security, Covers security threats from OWASP

Dash Board (reSonate)
- Dash Board view providing information on Code Quality, Unit Testing coverage on-going basis, alert notification

End-to-End Build Framework
- Unit Testing framework is extended to provide complete end-to-end Build Framework providing, code analysis, Unit Testing & Deployment

Performance Unit Testing
- Support to Test Performance as part of Unit Testing, provides time taken at Method level and helps to identify bottlenecks early in phase of Development

Functional Unit Testing
- Framework provides feature to verify functionality at Unit level

Best Practices
- Framework provides various standards, Guideline, Checklists to implement Unit Testing in an optimum way.
SOA Testing Framework

Sonnet Automation Test Framework Based on (QTP, SOAPUI)

Keyword-driven Functional & Regression Testing Framework
Process / Integration Testing Framework
Component Level Testing Framework
System Level Testing Framework

SOA Governance
QoS

Presentation Layer / Application Level
Browser
POS
Web2.0 UI
Thick Client

Business Layer

Service Layer / Component Layer

Database/ External Application / Internal Application
SOA Testing Framework

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>SOAPUI</td>
</tr>
</tbody>
</table>
| Expected Output        | • Regression, Unit, Interoperability, Security, Governance and Integration Testing of Web services  
                          • Ensuring the quality of service of Web Services                               |
| Integration Process    | The results of the SOA Testing will be exported to the Sonnet Transformation engine which in turn will be displayed in the dashboard of the Sonnet reporting engine |
| Benefits               | Stabilize the Functionality Testing early                                    |
**Functional Automation Framework**

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
<th>Tool</th>
</tr>
</thead>
</table>
| Regression Test case selection | - Identifying the Regression Test Cases to be automated  
- If Test case is not detailed enough, create the Detailed Granular Test case |        |
| Additional Keyword Development | - Identify if new Keyword need to be written  
- Write script for new Keyword if required  
- Publish the newly created Keyword |        |
| Automation Test Suite execution & Analysis | - Create/Update the Test Selector Suite  
- Execute the Cycle Driver for the specific Release/Cycle.  
- View the Test Execution Summary  
- Analyze the Test Results/Error Log | Selenium |
| Automation Test run | - Verify Automation Test suite in the local machine  
- Modify/Correct the Test Cases as needed  
- Baseline Regression suite |        |
| Integration Process | - Scripts / component mapping to Test Cases for coverage  
- Defect Management for re-executing Test scripts  
- Test Results exported to Sonnet Reporting Engine for Risk-based Assessment / Testing |        |
| **Expected Output** | - Customized keywords for Functional Automation  
- Regression Test Cases selection  
- Test scripts mapping to Test Cases and in turn to Requirements |        |
| **Benefits** | - Reusability of Test Scripts for Smoke & Sanity Test  
- Execution of Test Scripts across browsers and version includes executing scripts on different staging environment  
- Auto Configuration settings makes it drive independent. Hence easy to carry and maintain  
- Test Data file facilitates execution of same Test Case with different sets of Data (multiple iteration) |        |

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tool</strong></td>
<td>Selenium</td>
</tr>
</tbody>
</table>
| **Expected Output**       | - Customized keywords for Functional Automation  
- Regression Test Cases selection  
- Test scripts mapping to Test Cases and in turn to Requirements |
| **Integration Process**   | - Scripts / component mapping to Test Cases for coverage  
- Defect Management for re-executing Test scripts  
- Test Results exported to Sonnet Reporting Engine for Risk-based Assessment / Testing |
| **Benefits**              | - Reusability of Test Scripts for Smoke & Sanity Test  
- Execution of Test Scripts across browsers and version includes executing scripts on different staging environment  
- Auto Configuration settings makes it drive independent. Hence easy to carry and maintain  
- Test Data file facilitates execution of same Test Case with different sets of Data (multiple iteration) |
Security Testing Framework

Secure SDLC

- Product Requirements
- Functional Design
- Technical Design
- Development / Implementation
- Testing
- Release / Deployment

Security Gates

- Security Requirements Document
- Architectural Risk Analysis Guidelines
- Secure Coding Checklists / Guidelines
- OWASP & PCI Threat Modeling Manual and Tool-based testing Test Report

Sonata Accelerator

- 500+ Pre-built Test Cases
- Covers OWASP Top 10 Threats
Security Testing Framework

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Paros Proxy</td>
</tr>
</tbody>
</table>
| Expected Output   | • Threat Modeling of Application  
                    • Secure Code review  
                    • Security assessment of the application                                  |
| Integration Process | • Requirement ID mapping to security requirements  
                       • Vulnerabilities identified get consolidated in Sonnet Reporting Engine and a Dash Board view of the threats is displayed |
| Benefits          | • Security Testing is in compliance with well known standards like OWASP,PCI/DSS,WASC and CWE/SANS  
                       • Security Testing across SDLC  
                       • Hack resilient application                                               |
# Performance Testing

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tool</strong></td>
<td>Jmeter</td>
</tr>
</tbody>
</table>
| **Expected Output** | • Set Goals and System boundaries  
• Define Services/Components and possible outcomes  
• Select Metrics  
• List Parameters  
• Select Factors to Study  
• Select Evaluation Technique  
• Select Workload  
• Design Tests  
• Run, Analyze and Interpret data  
• Present Results |
| **Integration Process** | • The results of the Performance Testing will be exported to the Sonnet Transformation engine which in turn will be displayed in the Dash Board of the Sonnet reporting engine. |
| **Benefits**      | Help to mix and match the load                                             |
Sonnet Reporting Engine

Centralized report for the following

• Single view of the overall QA
• Requirements Traceability to Test Cases and in turn to automation Test Scripts
• Dash Board with functional and non-functional Testing details
• Day / Build-wise Test Design and execution count
• Weekly comparison / progress of Test Design and execution
• Categorization of defects based on build, severity / priority, requirements
• Defect Trends across multiple releases
Test Management – Take Away

- Tools Integration Architecture
- Option for Open Source and branded tools
- Functional and NFR Testing Integration
- Risk-based Testing to optimise the COQ
- Integrated view supporting Unit and Integrated testing
- TCO Reduction
Thank you for attending this session.

Please fill out an evaluation form and place it in the box located at the back of the room.