How to Optimize Your Web Testing Strategy

Software Test & Performance Conference 2007

The Marriott, San Mateo, CA

Hung Q. Nguyen

LogiGear® Corporation

April 17 - 19, 2007

The Theme

The key challenge in the practice of Web testing lies in adoption of a Web testing strategy and the allocation of resources behind it to support execution.

There are several different possible objectives for testing, such as identifying threats to customer satisfaction, certifying the product's behavior against a standard, minimizing technical support costs, or helping the project manager understand the risks of putting the software under test into production.

A strategy-focused company chooses a compatible set of processes (the ways the group gets things done), practices (test techniques and tools) and people who can use the techniques within the processes in order to achieve the objective.

The Story

A division of company A

- Produces telecommunication servers and Web-based applications.
- 75 test engineers supporting several product lines, 40 are in the US in three different locations, and 35 are in offshore location.
- US-based test engineers are very knowledgeable with their product lines.
- The business has been around for over a decade.
- Several tools including homegrown are in place to facilitate test automation, do performance and scalability testing, to perform code coverage, API testing and other product specific customized tests.
- Close to 50% of the test cases are automated.
- Dedicated performance and capability engineering team.
- Requirement-based testing is the main practice.
- Unit testing is done sporadically.
- External beta testing is unrealistic.
- Problem tracking and management, and configuration management systems are solid.
- While not strictly enforced, there are several good Software Development Life Cycle (SDLC) processes in place with a lot of documentation produced throughout the development phases.

The Story

The problems

- Often, within 90 days of the release, the customers report an average of 15% of the total (valid) bug counts.
- A good number of bugs are user-scenario and configuration specific, but many others are functional and exception handling specific.
- It costs the company significantly to release patches and support the customers in resolving the bugs.

What's wrong?

The Story

A division of company B

- Delivers various products catering to the financial industry.
- 50% of the products are on a mainframe platform with client/server supports, and the other 50% are on a Web platform. The goal is to deliver 80% of the products on a Web platform within the next two years.
- 60 test engineers supporting several product lines in two different US locations.
- Test engineers are very knowledgeable about their product lines.
- The business has been around for over a decade.
- Several tools are in place to facilitate test automation, and do performance and scalability testing.
- Requirement-based testing is the main test practice.
- Unit and API testing are done sporadically.
- User Acceptance Test (UAT) is done both internally and externally.
- Problem tracking and management, and configuration management systems are solid.
- CMM Level II certified, and is about get Level III certified.

The Story

The problem

- Often, within 90 days of the release, the customers report up to 30% of the total (valid) bug counts.
- A good number of bugs are user-scenario and configuration specific, but many others are functional, configuration and exception handling specific.
- It costs the company to resolve the bugs.
- Performance of Web applications fails to meet customer expectations.
- Losing business to competitors because it takes the company too long to deliver a release (6 months versus a month by the competitors).

What's wrong?

The Assignment

- To conduct an independent assessment on the test engineering's state-of-the-practice with the intention of reporting findings, recommending an action plan for improving performance and mitigating process and quality related problems.
- The objectives
  - Company A: To deliver a better testing program that leads to better quality releases.
  - Company B: To deliver a better testing program that leads to better quality and faster releases.

Testing Strategy in Context

A test strategy is a holistic plan that starts with a clear understanding of the core objective of testing, from which we derive a structure for testing by selecting from many testing styles and approaches available to help us meet our objectives.

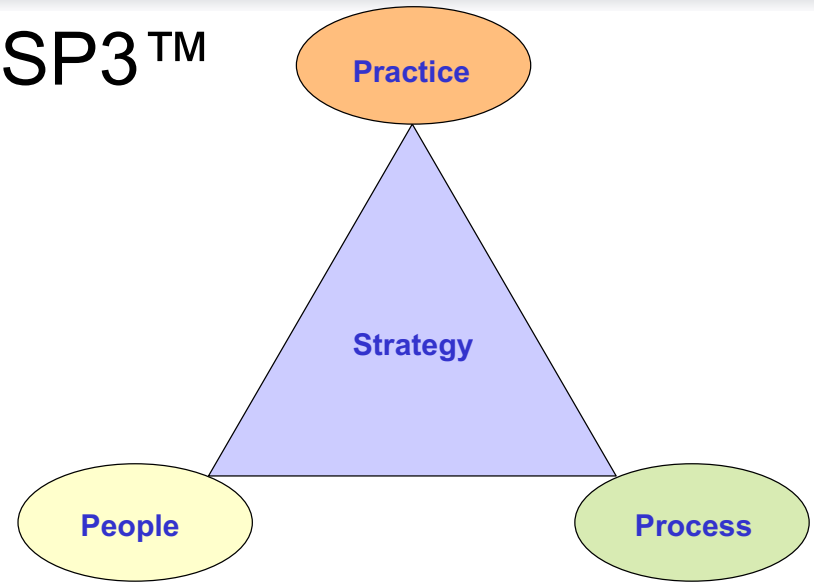
There are three focal points (SP3) that one can adjust for best results—People, process, and practice (methods and tools). "Process" means the ways of doing things that people have agreed to. "Practice" means a successful combination of testing methods and tools that help meet the agreed objectives. "People" means the human resources that are capable of applying the practice.

Combined with Strategy we call this SP3:

Strategy = People + Process + Practice

Testing Strategy in Context

SP3™



Source: I developed this model by learning from Mitchell Levy, Value Framework® Institute

Examples of Possible Improvements

☑: Yes
▫: Maybe
•: No

SP3: Strategy that consists of People, Process and Practice
Process: Life cycle
People: Skill sets, communication and morale
Practice: **Methods and tools**

Plausible Corrective Actions	Faster	Better	Cheaper	SP3
Automate test design	☑	▫	▫	Practice
Automate test documentation	☑	▫	▫	Practice
Automate test execution	☑	▫	▫	Practice
Deploy test automation strategy	☑	▫	▫	Practice
Focus testing on bug-finding rather than documentation	☑	▫	•	Process
Improve meaningful metrics with well defined correlation/corrective action	☑	☑	▫	Practice
Improve visibility of testing/QA activities	☑	☑	☑	Practice
Lessen number of cycles	☑	•	▫	Process
Reduce automation test script maintenance	☑	•	☑	Practice
Reduce test case maintenance	☑	•	☑	Practice
Remove redundancies	☑	•	☑	Practice
Shorten the test cycle	☑	•	▫	Process
Test earlier	☑	▫	•	Process
Upgrade talent through training and/or churning	☑	☑	▫	People
Improve test selection	☑	▫	▫	Practice
Leverage outsourcing including global resources	▫	•	☑	Practice
Broaden test coverage	•	☑	•	Practice
Improve test design	•	☑	•	Practice

Typical Assessment Activities

- **Identify an executive sponsor—Often, this is the CTO, VP of Engineering or a GM of a division.**
- Conduct half a dozen pre-assessment information gathering meetings.
- Conduct a dozen 2 - 3 hour interview sessions with staff across the organization, including executive and middle management staff across functions, project management and programming staff and ultimately, software testing and QA staff.
- Conduct software testing survey consisting of about a hundred questions designed to gather intelligence about testing practices, and groups' as well as individuals' capability.
- Review about a thousand pages of documents.

The Results—Company A's Top Issues

- Ineffective test design methods and techniques.
- Testing is requirement-focused.
- An ineffective test automation strategy and framework diminishes the overall effectiveness (and causes resource depletion in some cases).
- Poor resource allocation.
- Testing staff has low morale.
- Lack of visibility into QA and test activities.
- Testing staff has strong domain expertise but its industry QA and skill-based testing expertise is light.
- Metrics program does not lead to actionable improvement activities
- Poor process definitions and executions (e.g., no approvals, no checkpoints, no traceability).
- Poor integration and management of offshore resources.

The Results—Company B's Top Issues

- The only focus of testing is requirement-based.
- Test process is documentation heavy, more than necessary which slows everything down.
- Testing lacks depth—More checklist rather bug-finding oriented.
- Poor configuration and scenario coverage.
- Lack of visibility to help evaluate testing effectiveness and efficiency.
- Test automation is ad-hoc. Much of the automation is done by translating manual test scripts.
- Lack of visibility to help understand test automation effectiveness and efficiency.
- The practice of system testing is redundant to other testing.
- The objectives of external testing are not aligned with business objectives.
- Poor external beta test practice and management.

Conclusions in Both Cases

- Key issues to be addressed before focusing on tactics:
  - Lack of a corporate-wide software testing strategy that aligns testing/QA objectives with business objectives.
  - Unclear testing/QA objectives and direction from management at the organizational level (across-functions).
  - Lack of hug-hunt testing and methods and techniques to support that level of testing
  - Lack of a test automation strategy, framework and tool that deliver long-term cost-effective, manageable and scalable results.

Common Findings

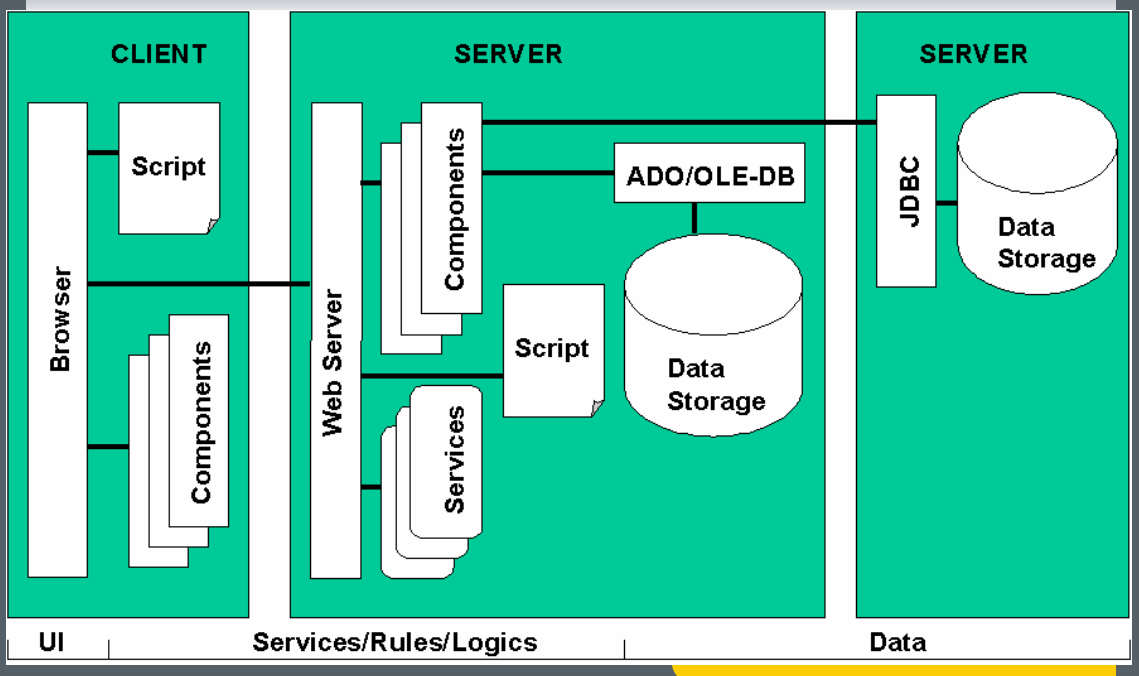
Based on several dozens of assessments:

- The key issue is not a lack of testing tools, processes or competent staff.
- It is the lack of a corporate-wide test strategy that leads to other problems.

"What's the use of running if you are not on the right road."

German Proverb

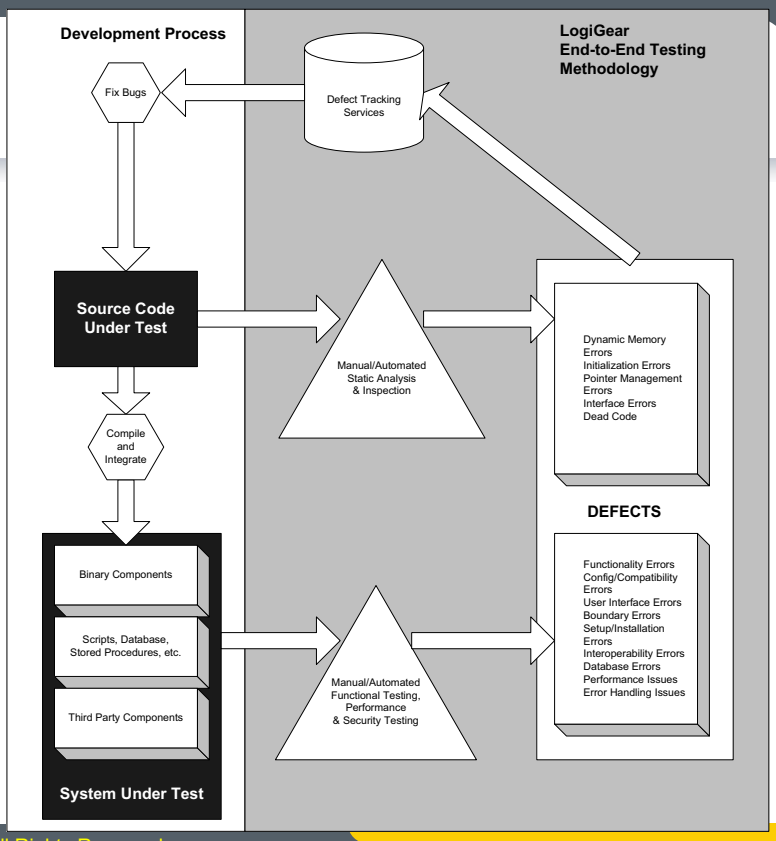
The Technical Context



Software Components

Application Server Components	3rd Party Components
Server Side	Java Components
Web Server	ActiveX Controls
Scripting	Standard EXE's
Java Virtual Machine	Standard DLL's
Database Server	CGI's
Data Access Service	API's
Transaction Service	Web Services
etc.	etc.
Client Side	
Web Browser	
Scripting	
Java Virtual Machine	
etc.	
Integrated Application Components	
HTML, DHTML, XHTML, JavaScript, VBScript, Jscript, PERL Script, SOAP, AJAX, UDDI, WSDL, etc.	
Standard EXE's	
CGI's	
API-based components	
Java Components	
ActiveX Controls	
Standard DLL's	
etc.	

A Test Planning Strategy Example



Test Planning Strategies

- Generate a list of physical deliverables. The list should includes binary-based components as well as source-based components (such as stored procedures and scripts).
- For each identified component, write a short description of what it does and its dependencies.
- Determine testing strategies at the unit/component level as well as at the integration level.
- Determine various levels of change control management for various development phases (green, yellow and red).
- Determine how defect reports are tracked and managed.
- Identify opportunities to use testing tools to compliment manual testing.

A Formulated Strategy Should Also Include

1. Identifying different product development styles from inception through maintenance, so that we can eventually map the appropriate test strategy to each.
2. Mapping out phases, milestones and relevant activities on a typical timeline.
3. Identifying the equivalent type of test strategies for each development method.
4. Prescribing what's involved in each test strategy.

LogiGear® The Strategy Framework

Test Type	Test Type Attributes			Tes Phases							
				SE			TE			C	O
	Bug Type (Objective)	Interface Level	App Maturity	Pre-coding	Coding	Pre-integration	System	System Integration	Final	Beta/User Acceptance	Post Production
Requirement Review		Static									
Design Review		Static									
Code Review		Static									
Code Walkthrough		Static									
Design Walkthrough		Static									
Unit		Function and class									
Module		API									
API		API									
External Functional		UI and system non-UI									
Useability		UI									
Accessibility		UI									
Configuration		UI and system non-UI									
Compatibility		UI and system non-UI									
I18N		UI and system non-UI									
L10N		UI and system non-UI									
Regression		UI and system non-UI									
Performance		UI and system non-UI									
Load		UI and system non-UI									
Stress		UI and system non-UI									
Soak		UI and system non-UI									
Failover/Recovery		UI and system non-UI									
Installation		UI and system non-UI									
Security		UI and system non-UI									
Data verification testing		UI and system non-UI									
Compliance		UI and system non-UI									

LogiGear® The Strategy Framework

Test Approach							Quality Objective						Test Documentation
Requirement-based	Scenario-based	Model-based	Attack-based	Risk-based	Fault Injection	DAST	Functionality	Usability	Performance	Security	Compatibility	Scalability	etc.	
Test creation relies primarily on requirement documentation.	Building test cases based upon a story or use of the application not requirements	Using a model of the application under test to generate test cases	Aggressive attacks against the APIs of the application	Prioritizing test cases based upon perceived risk.	Purposefully putting the application into error states.	Diagnostic Approach- simply, a variety of 20 Questions.							

The Strategy Framework

Terminology	Definition
Test Strategy	A test strategy is a holistic plan that starts with a clear understanding of the core objective of testing from which, we derive a structure for testing by selecting from many testing styles and approaches available to help us meet our objectives. There are three focal points (SP3) that one can make adjustment to—People, process, and practice (methods and tools) to get the best results. “Process” means the ways of doing things that people have agreed to. “Practice” means a successful combination of
Test Type	A test type is a category of test activities with the objective of exposing specific types of bugs at certain interface and product marturity levels. The selection of the test types must satisfy the test or quality objectives.
Test Phase	A test phase is a time a block often derived from the SDLC phases. However, within each SDLC phase, there might be more than one test phase. Another property that defines the test phase is the objective of the testing during that phase.
Test Approach	A test approach is a testing philosophy or style that drives the selection of certain test design methods. The test approach often, may imply an embedded method to carry out that testing philosophy. In addition, certain development methods may require a certain test approach.
Test Design Method/Technique	A test design method is a systematic procedure in which you create tests to be executed. A test type or test approach may use one or more test design methods.

Strategy Formulation Process

1. What are your quality objectives or characteristics?
2. What are the requirements for each characteristic?
3. What are the types of bugs that affect each quality characteristic?
4. What are the test types or activities needed to support finding problems described in #3?
5. What are the most effective approaches to finding specific types of bugs as early as possible?
6. What is the required application maturity to support #4?
7. How would #5 and #6 be mapped to the various phases in the SDLC?
8. How would you qualify the maturity of the software to determine that it has reached its milestone?
9. How do you quantify and measure your work?
10. What tools can help you improve your work and which framework is needed to implement the tool successfully?

Quality Objective Examples

- Functionality
- Usability
- Performance
- Security
- Compatibility
- Scalability
- Recovery

Requirements for Quality Objective Examples

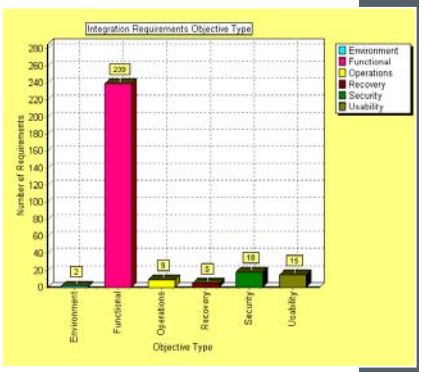
SEIU R-I

Description: Technical System Design Supplemental Integration Requirements are included in this report. The Integration Requirements will detail the outline for performing exploratory testing using Test Director. The Integration Requirements, themselves, are the Test Specifications.

Document Version: 0.1

Document Date: 19-Aug-2003 04:18 PM

Related Documents: SEIU Technical Design Specification



Chapter 1 Requirements

Requirement Phase	Req ID	Reference	Requirement	Objective Type
Integration-Risk Based Exploratory Assessment	2974	Software	SEIU Database Roles give basic privileges to the system	Security
Integration-Risk Based Exploratory Assessment	3136	Software	The SEIU Move Study Groups are functional	Functional
Integration-Risk Based Exploratory Assessment	3137	Software	The SEIU Move Study Group Sets are functional	Functional
Integration-Risk Based Exploratory Assessment	3138	Software	The SEIU Move Extract is functional	Functional

Bug Type Examples

1. Design errors
2. Coding standard, other coding-related error
3. Design and coding-related errors
4. Design and logic errors
5. Function/Class specific errors
6. Exported function specific errors
7. Functional errors at run-time
8. UI design errors
9. Errors caused by environments
    1. Configuration: Composed of certain mixes of software and/or hardware configurations
    2. Compatibility: Composed of certain mixes of software and/or hardware types and versions
10. Errors that prevent the system from meeting satisfactory response-time
11. Functional errors caused by heavy traffic or usage of the system
12. Exception-handling errors at the system level
13. Errors caused by resource depletion due to memory leaks or clean-up routines
14. Errors in functional and data recovery after a system failure
15. Installation errors
16. Vulnerabilities that expose security risks to the operator as well as the user of the system
17. Source data errors
18. Errors in meeting certain standards or regulations

Test Type Examples

1. Design Review
2. Code Inspection/Review
3. Code Walkthrough
4. Design Walkthrough
5. Unit
6. API
7. External Functional
8. Usability
9. Accessibility
10. Configuration
11. Compatibility
12. Regression
13. Performance
14. Load
15. Stress
16. Failover/Recovery
17. Installation
18. Security
19. Compliance

Test Approach Examples

- Requirement-based
- Scenario-based
- Soap Opera
- Model-based
- Attack-based
- Risk-based
- Fault Injection
- DAST
- Exploratory

Application Maturity Examples

- Pre-coding
- Coding stage
- Code and unit test complete
- Pre-integration
- Run-time testable
- Functionality complete (assuming that much of the developer centric work has been done earlier)
- Functionality complete and tested
- Functionally stable for multiple users
- Install-ready

Test Phase Examples

Test Phases							
Done by software engineer			Done by test engineer			Done by customer	Done by operation staff
Pre-coding	Coding	Pre-integration	Integration	System	Final	Beta/User Acceptance	Post Production

Strategy Formulation Process

1. What are your quality objectives or characteristics?
2. What are the requirements for each characteristic?
3. What are the types of bugs that affect each quality characteristic?
4. What are the test types or activities needed to support finding problems described in #3?
5. What are the most effective approaches to finding specific types of bugs as early as possible?
6. What is the required application maturity to support #4?
7. How would #5 and #6 be mapped to the various phases in the SDLC?
8. How would you qualify the maturity of the software to determine that it has reached its milestone?
9. How do you quantify and measure your work?
10. What tools can help you improve your work and which framework is needed to implement the tool successfully?

Results After Implementation

- In both cases, the missed bug rates have been reduced to less than 10% at the same cost structure.
- More importantly, none of the bugs missed are considered critical bugs that require immediate patches.
- “Faster” is a trickier attribute to measure because it depends on many variables, many of which are not testing-specific. However, both organizations have moved to a release-train model.
- Test automation frameworks are deployed for better visibility, maintainability and productivity.
- While much progress has been made, obtaining metrics remains the largest problem. This is due to the fact that it requires some infrastructure to ensure data integrity.

About LogiGear Corporation

LogiGear provides global solutions for software testing including the world-leading test automation solution with Action Based Testing. For over a decade, we've worked with hundreds of companies, from Fortune 500 to startups, delivering unique testing solutions that meet their unique needs. We double their test coverage, cut test time in half, improve quality and reduce cost.

LogiGear has built a reputation for offering the widest range of services in the software testing industry. Be it turn-key test automation, consulting, training, outsourced testing, or products, we partner with software organizations to create approaches that precisely meet their demands.

www.LOGIGEAR.com

Enjoy STP Conference 2006!

"Strategy without tactics is the slowest route to victory. Tactics without strategy is the noise before defeat."

Sun Tzu

www.LOGIGEAR.com