Exploratory Testing of non-functional requirements

by

Kalyan Manikonda, Specialist Master, and
Giridhar Manenga, Specialist Master, Deloitte Consulting
Risk Based Exploratory Testing of Non-Functional requirements

June 2014

Introduction of speakers

Kalyan Manikonda
kmanikonda@deloitte.com

Giridhar Mangena
gmangena@deloitte.com

Background and Experience

Kalyan is specialized in managing and delivering testing projects across multiple industries involving on-site and offshore delivery. He has global implementation experience for over 14 years in Health Care Life Sciences, Telecommunications, and Insurance industries. He has strong test program management skills with vast experience in test advisory, planning, and implementing Quality Assurance (QA) processes; played key roles in project management office.

He has deep understanding of full software development life cycle and led multiple engagements involving Waterfall, Agile, and Iterative development methodologies.

He is a certified scrum master

Giridhar has more than 10 years of experience in testing industry helping clients in test planning, project management, test advisory, and as an established SME in performance testing and capacity planning areas.

He has extensive experience in requirement engineering, automation testing, managing, and delivering testing projects across multiple industries in onshore–offshore delivery model. He has vast experience in streamlining QA processes, tools, methodologies to the clients. He has sound knowledge on industry standard processes and implementation experience using various development models in testing projects.
Exploratory Testing of non-functional requirements

Agenda

- Introduction to Non-functional requirements
- Introduction to Performance testing
- How Performance testing carried out in regular and agile models
- Performance Testing challenges
- Exploratory Testing and Risk Based Testing
- Proposed approach - Risk based exploratory(RBE) non-functional testing
- Risk based exploratory non-functional testing workflow
- Risk based exploratory non-functional testing framework
- Case study
- Reporting risk based exploratory non-functional testing results
- Benefits
- Conclusion
- References

Video

Sky Sports - Formula 1® Broadcaster of the Year
Introduction to Non-Functional requirements

The non-functional requirements define how a system or product is supposed to behave. They also describe the characteristics in various dimensions that are important either to users or to developers and maintainers.

Nonfunctional aspects of a system

- Reliability
- Scalability
- Security
- Availability
- Performance
- Usability
- Disaster Recovery

Introduction to performance testing

Performance testing helps to

- Determine the current system capacity of the multi-user critical applications by simulating the real world use.
- It is a technical investigation done to determine or validate the responsiveness, throughput, reliability, and/or scalability of a system under a given workload.

Performance testing is commonly conducted to accomplish the following

- Adequacy of current capacity
- Infrastructure Capacity
- Compare current & desired quality characteristics
- Speed, Scalability & Stability
- Application Bottlenecks
- Application behaviour at various loads
- "Go-Live" readiness
- Prevent Revenues loss & Brand Credibility
- Resource utilization
- Response time
- Through put
How performance testing carried out in traditional and agile models

In traditional SDLC life cycle, the scope of performance testing is for the entire system, whereas the scope is different in Agile implementations as the system is built in sprints and in continuous basis.

Performance testing challenges

Conducting performance testing is always challenging irrespective of the SDLC life cycle adopted, if appropriate precautions are not taken.

<table>
<thead>
<tr>
<th>Common challenges in Waterfall model</th>
<th>Common challenges in Agile model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clearly defined non-functional and performance requirements are not provided to evaluate the system behavior</td>
<td>Changes to business process in various sprints may invalidate the results captured for that business process in previous sprints</td>
</tr>
<tr>
<td>Executing all the scripts without errors won’t give the assurance that the system is working correctly.</td>
<td>Can not learn the complete system behavior as multiple dimensions of the system under test is unknown to perform the planned tests.</td>
</tr>
<tr>
<td>Scaled down environment may or may not yield the appropriate results that the customers are looking for</td>
<td>Since most of the functional testing happens in the dev environment which is not of same capacity as production environment, the result may not be comparable</td>
</tr>
<tr>
<td>Most of the tests are not simulated with real world scenarios</td>
<td>Focus of sprint delivery is to deliver “Working Software” and may not meet the performance SLA’s</td>
</tr>
<tr>
<td>Inadequate tuning and performance trouble shooting activities don’t find actual issues in the system</td>
<td>Shorter development sprints may result in more performance cycles</td>
</tr>
<tr>
<td>Consistency of the response times/results is questionable if the workload varies.</td>
<td></td>
</tr>
</tbody>
</table>
What is Exploratory testing?

Cem Kaner, who gave the term in 1983, defines exploratory testing as “a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.”

![Exploratory Testing Diagram]

**Approach**
- Simultaneous learning
- Test design
- Test execution
- Analysis

**Key success factors**
- Trust in testers
- Dependence of individuals skills
- Focus on finding problems

**Benefits**
- Less preparation is needed for testing
- Important bugs are found quickly
- Scripting is not required.
- Testers can use deductive reasoning based on previous results
- After initial testing, most bugs are discovered by some sort of exploratory testing

What is Risk based testing, why should it be implemented?

Risk based testing (RBT) is “a proactive way to manage risks in the software development lifecycle to organize testing efforts so that we appropriately balance risk with schedule and cost, with the consensus from key stakeholders - business, technology, and testing”. Its key objective is to improve time to market with quality is still maintained.

<table>
<thead>
<tr>
<th>Identification</th>
<th>Analysis</th>
<th>Risk Assessment</th>
<th>Test planning/execution</th>
<th>RBT governance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder objectives</td>
<td>Stakeholder objectives</td>
<td>Stakeholder objectives</td>
<td>Stakeholder objectives</td>
<td>Stakeholder objectives</td>
</tr>
<tr>
<td>Risk factors</td>
<td>Risk factors</td>
<td>Risk factors</td>
<td>Risk factors</td>
<td>Risk factors</td>
</tr>
<tr>
<td>Functionality breakdown by complexity</td>
<td>Functionality breakdown by complexity</td>
<td>Functionality breakdown by complexity</td>
<td>Functionality breakdown by complexity</td>
<td>Functionality breakdown by complexity</td>
</tr>
<tr>
<td>Probability of failure</td>
<td>Probability of failure</td>
<td>Probability of failure</td>
<td>Probability of failure</td>
<td>Probability of failure</td>
</tr>
<tr>
<td>Business impact</td>
<td>Business impact</td>
<td>Business impact</td>
<td>Business impact</td>
<td>Business impact</td>
</tr>
<tr>
<td>Frequency of usage</td>
<td>Frequency of usage</td>
<td>Frequency of usage</td>
<td>Frequency of usage</td>
<td>Frequency of usage</td>
</tr>
<tr>
<td>System complexity</td>
<td>System complexity</td>
<td>System complexity</td>
<td>System complexity</td>
<td>System complexity</td>
</tr>
<tr>
<td>Risk categorization</td>
<td>Risk categorization</td>
<td>Risk categorization</td>
<td>Risk categorization</td>
<td>Risk categorization</td>
</tr>
<tr>
<td>Weighted averages</td>
<td>Weighted averages</td>
<td>Weighted averages</td>
<td>Weighted averages</td>
<td>Weighted averages</td>
</tr>
<tr>
<td>Test scope</td>
<td>Test scope</td>
<td>Test scope</td>
<td>Test scope</td>
<td>Test scope</td>
</tr>
<tr>
<td>Business process and test case prioritization</td>
<td>Business process and test case prioritization</td>
<td>Business process and test case prioritization</td>
<td>Business process and test case prioritization</td>
<td>Business process and test case prioritization</td>
</tr>
<tr>
<td>Monitoring and control</td>
<td>Monitoring and control</td>
<td>Monitoring and control</td>
<td>Monitoring and control</td>
<td>Monitoring and control</td>
</tr>
<tr>
<td>Reporting</td>
<td>Reporting</td>
<td>Reporting</td>
<td>Reporting</td>
<td>Reporting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risk Level</th>
<th>Risk Assessment (Likelihood/Impact)</th>
<th>Actions</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must Have</td>
<td>H/H, M/H, H/M</td>
<td>Elevated level of rigor to confirm appropriate test coverage</td>
<td>Elevated level of rigor to confirm appropriate test coverage</td>
</tr>
<tr>
<td></td>
<td>H/L, M/L, L/H</td>
<td>Prioritized to be executed first</td>
<td>Prioritized to be executed after critical cases</td>
</tr>
<tr>
<td>Important</td>
<td>H/L, M/L, L/H</td>
<td>Normal level of rigor (requirements covered, reasonable data permutations)</td>
<td>Normal level of rigor (requirements covered, reasonable data permutations)</td>
</tr>
<tr>
<td>Nice to Have</td>
<td>L/M, M/L, M/L</td>
<td>Relatively low level of rigor (focus on breadth over depth)</td>
<td>Relatively low level of rigor (focus on breadth over depth)</td>
</tr>
<tr>
<td></td>
<td>L/L</td>
<td>Executed depending on project constraints (cost, schedule)</td>
<td>Executed depending on project constraints (cost, schedule)</td>
</tr>
</tbody>
</table>

Copyright © 2014 Deloitte Development LLC. All rights reserved.
Risk-based exploratory (RBE) non-functional testing approach

This is an idea of combining exploratory non-functional testing techniques and tactics in conjunction with regular non-functional testing in a risk-based approach.

- Subjecting to a system, application, or tier/component to a set of unusual parameters or scenarios that are not represented from the business context when defining the NFRs or SLAs.
- Exploratory non-functional tests can be included in both waterfall and agile models.
- Uses an iterative approach where an exploratory selection criterion evolves as the testers learn about the system behavior or how the system has been tuned for a particular circumstance.

Risk based exploratory non-functional testing workflow

The following workflow depicts the step by step activities to identify the priorities for business scenarios:

1. Select Test Type
2. Select Quality Characteristic
3. Select Sub System
4. Determine Relative Importance of sub systems
5. Identify Sub System Priority
6. Select Business Scenarios
7. Identify Business Risk
8. Identify Test Priorities for Business Scenarios

The determination of relative importance should preferably be done with inputs from all stakeholders or at a minimum test strategist should determine the relative importance and get it reviewed by stakeholders. This will ensure that a general image of the importance of the sub-system is obtained from all concerned parties.
Components of RBE non-functional testing

The following elements define and drive the RBE framework.

- **Test types**
  - The common non-functional test types are:
    - Component performance testing
    - Soak/Endurance testing
    - Load testing
    - Stress testing
    - Volume testing
    - Security testing
    - User experience testing

- **Quality characteristics (QC)**
  - The common quality characteristics are:
    - Speed – User experience; responsiveness
    - Scalability – Capacity; Volume; Future growth
    - Stability – Consistency; Reliability; Robustness
    - Response – Synchronous; Asynchronous
    - Efficiency – Resource consumption
    - Availability – Failure mode; Recovery
    - Data accuracy and security

- **Subsystems**
  - Evaluating the system components/tiers includes, but is not limited to, the following activities:
    - Understand and capture the physical and logical architecture
    - Identify any other process/systems using the architecture.
    - When testing a Web farm, consider the usage of the Internet protocol switching techniques.

Risk based exploratory non-functional testing framework

Once the core elements of RBE framework are identified, they'll be used in identifying the business and test priorities.

- **Relative importance (RI)**
- **Sub-system priority**
- **Business priority**
- **Test Priority**

- The steps to determine the sub-system priority is to:
  - Select RI of every quality characteristic for every sub-system
  - Once RI of quality characteristics is determined for all the quality characteristics for the sub-system, the scalar product of RI of quality characteristics and RI of sub-system determines the overall “Sub-system priority”

- The steps to determine the business risks are:
  - Identify business scenarios for each sub-system
  - The business risks can be determined by the scalar product of “probability of failure” and their impact based on “frequency of usage”.

- The “Test Priority” associated with business scenarios will be scalar product of sub-system priority and business risk (priority). The determination of test priority at business scenario level helps identify risks of the product/system/solution at the lowest possible level thus projecting a realistic picture of associated risks.
Risk based exploratory non-functional testing

Below is an example of using RBE framework to arrive at test priorities for various business scenarios related to a sub system. The parameters may change every cycle as we learn, design and execute the scripts.

<table>
<thead>
<tr>
<th>QC vs. Test Type</th>
<th>Component</th>
<th>Endurance</th>
<th>Load</th>
<th>Stress</th>
</tr>
</thead>
<tbody>
<tr>
<td>User satisfaction</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Synchronicity</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SLA violation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Response time</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Capacity</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Volume</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Future growth</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Identify QC Vs. test types**

**Identify sub-system**

<table>
<thead>
<tr>
<th>Sub-System</th>
<th>Priority</th>
<th>Impact</th>
<th>Business Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-system 1</td>
<td>High</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Sub-system 1</td>
<td>High</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Sub-system 2</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Sub-system 2</td>
<td>M</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Sub-system 3</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Sub-system 3</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Sub-system 4</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Sub-system 4</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Sub-system 5</td>
<td>M</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Sub-system 5</td>
<td>M</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

**Find relative importance**

<table>
<thead>
<tr>
<th>Sub-System</th>
<th>QC</th>
<th>RI</th>
<th>SC</th>
<th>OCC</th>
<th>FTU</th>
<th>Sub-system priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-system 1</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Sub-system 2</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-system 3</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-system 4</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-system 5</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Identify test priority**

<table>
<thead>
<tr>
<th>Business Sectors</th>
<th>Business Priority</th>
<th>Product Selection</th>
<th>Test Priority</th>
<th>Probabilty of Failure</th>
<th>Impact</th>
<th>Business Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Creation</td>
<td>H</td>
<td>L</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account Creation</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Selection</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment Method</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finance</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Identify business priority**

<table>
<thead>
<tr>
<th>Business Sectors</th>
<th>Business Priority</th>
<th>Product Selection</th>
<th>Test Priority</th>
<th>Probabilty of Failure</th>
<th>Impact</th>
<th>Business Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Creation</td>
<td>H</td>
<td>L</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account Creation</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Selection</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment Method</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finance</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Identify sub-system priority**

<table>
<thead>
<tr>
<th>Sub-System</th>
<th>QC</th>
<th>RI</th>
<th>SC</th>
<th>OCC</th>
<th>FTU</th>
<th>Sub-system priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-system 1</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Sub-system 2</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-system 3</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-system 4</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-system 5</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Case study**

RBE testing approach has been adopted for one of the e-commerce implementation to achieve various objectives.

**Objective**
- Very limited NFRs /SLAs
- Enable e-commerce functionality for new regions and launch new products
- Implement Search Engine Optimization (SEO) for better ranking of products in search engine bots in Internet
- Improved Customer satisfaction

**Outcome**
- Benchmarking of solution with right set of characteristics.
- Achieved optimized system performance
- Uncovered bottlenecks in key business risk areas.
- Reduced cost of quality.
- Verified all the URLs in the system to conform to the search engine standards.

**Planning and execution in cycles**

**Cycle 1**
- Test Type: Load
- Sub System: Portal Business Process: Place Orders, Search
- OC: Speed & Scalability
- Test priority: Critical

**Cycle 2**
- Test Type: Volume
- Sub System: URL’s Business Process: SEO
- OC: Accuracy of URLs
- Test priority: High

**Cycle 3**
- Test Type: Batch processing
- Sub System: ERP
- Business Process: Place orders while cron jobs run
- OC: Speed Efficiency
- Test priority: Medium

**Cycle 4**
- Test Type: Load + Batch processing
- Sub System: Portal, ERP
- Business Process: Place orders while cron jobs run
- OC: Speed Efficiency
- Test priority: Medium

Learn and repeat the cycles
Sample Report of RBE non-functional testing

Sample status report with risk based exploratory non-functional testing approach. If NFR’s/SLA’s are given or not, business processes and quality characteristics needed to be validated, can be selected as the performance team learns the application, executes the scripts and publish the results.

<table>
<thead>
<tr>
<th>Sub-system</th>
<th>Business process</th>
<th>Quality characteristic</th>
<th>Test priority</th>
<th>Work load (Trans per sec)</th>
<th>Baseline Respons e time</th>
<th>Actual respons e time</th>
<th>Throughput</th>
<th>Status</th>
<th>Issues identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portal</td>
<td>Place Order</td>
<td>Respons e time</td>
<td>Critical</td>
<td>50</td>
<td>4 secs</td>
<td>6 secs</td>
<td>1150</td>
<td>Yellow</td>
<td>Test data used between baseline and actual runs ended up different</td>
</tr>
<tr>
<td>Portal</td>
<td>General search</td>
<td>Future growth</td>
<td>Critical</td>
<td>100</td>
<td>2 secs</td>
<td>2.5 secs</td>
<td>200</td>
<td>Green</td>
<td>None</td>
</tr>
<tr>
<td>Portal</td>
<td>Cron jobs</td>
<td>Synchro nicty</td>
<td>Medium</td>
<td>20</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Red</td>
<td>Only 7 transactions are being processed per hour</td>
</tr>
</tbody>
</table>

Benefits

- Focus on the high priority business scenarios
- No NFRs and SLAs have to be defined for this approach, which opens to unlimited opportunities to find more performance issues upfront, instead of confining to the defined NFR’s/SLA’s
- Provide the capability to right-size the testing effort based on risk tolerance
- Enable stakeholders to identify risks and criteria that is most useful and meaningful to them
- Provide a repeatable framework for other projects
- Cover all the aspects of non-functional testing and is more effective in finding the potential issues
- Develop non-functional testing strategy/test plan and the scenarios focused on these areas
- Help in developing a better exit criteria for testing thus helping management to focus more on critical or high scenarios of the project
- Map the non-functional test scenarios back to risk/quality characteristic (traceability)
- Plan test execution based on the priorities
- Track test execution by risk/quality characteristic
- Report based on quality characteristics and test priorities and take timely measures and decisions based on residual risk
Summary and Conclusion

Exploratory testing is a powerful approach to improve testing outcomes without scripted testing. Most of us practice some form of exploratory testing day-to-day on the projects, however it is currently being limited to functional testing. Exploratory testing will improve the non-functional aspects of the application, and identify the bottlenecks ahead of time in the lifecycle.

This new proposed approach may take an extra effort apart from the regular planning, however it surely reduces a lot of risks by performing additional risk based exploratory cycles. Since each cycle has a variation in the test types, quality characteristics, the sub systems, and business processes, the cost of quality will be reduced and more non-functional issues can be uncovered effectively.

References

- Performance testing guidance for web applications by J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, Dennis Rea
- www.perftestplus.com
- Exploring Exploratory Testing by Andy Tinkham (FIT) and Cem Kaner (FIT)
- The Nature of Exploratory Testing by Cem Kaner and James Bach
Q&A