HEURISTIC
RISK-BASED
TEST
AUTOMATION
A HYBRID APPROACH
Problem
Current Pressures of Functional/Feature Testing

- For any complex system there are a vast number of tests
- There is infinite cloud of possible tests
- Being in Agile World, shorter release cycles are need of the hour
- Selective and efficient test coverage is inherent requirement
- Continuous Integration - automating the whole process

Solution
Heuristic Risk Based Automation

Risk-based testing helps us meet all the above by prioritizing and reducing scope which helps tester to find bugs early in the cycle and also manage testing in a smaller duration of time. Automating the whole process of prioritizing optimizes it further.
Risk-based testing (RBT) is a type of **software testing** that prioritizes the tests of features/functions/test cases based on the risk of their failure, it is

- A testing strategy to determine what to test, when, and how much, based on risk
- An opportunity to negotiate what “quality” means, and the acceptable level of quality for a project
- A way to confirm that risk remaining after testing is visible and acceptable to the business

## TYPES OF RISKS

### BUSINESS OR OPERATIONAL

- High use of a subsystem, function or feature
- Criticality of a subsystem, function or feature, including the cost of failure

### TECHNICAL

- Geographic distribution of development team
- Complexity of a subsystem or function

### EXTERNAL

- Sponsor or executive preference
- Regulatory requirements

### NON-FUNCTIONAL WEB RELATED

- Static content defects
- Web page integration defects
- Service (Availability and Performance) related failure
- Usability and Accessibility-related failure
- Security vulnerability
- Large Scale Integration failure
HOW TO CALCULATE RISK?

- Identify heuristics for exploring software risk.
- Risk analysis using inside-out approach.
- Inside-out is a direct form of risk analysis. It asks “What risks are associated with this thing?”
- Identify cost and probability factors based on this approach.
- Use heuristics to assign weights [cost (weight) and probability (weight)] to the factors.
- Calculate risk using these factors using a generic formula.

Use Case Risk
= Impact of Occurrence * Likelihood of Occurrence
= Average cost (Σ (cost (weight) * cost (value)) / Number of cost factors) * Probability (Σ (defect generator (weight) * defect generator (value)) / Σ (defect generator (weight) * defect generator (max value)))

Test Case Risk = Use Case Risk * Test case Importance

- Risk Calculation can be tedious and time consuming.
- Also with continuous Integration being a mandate process in this agile world, regular runs are need of the time.
- This brings the need to automate the Risk calculation process.

IS RISK-BASED TESTING SUFFICIENT?
NEED FOR AUTOMATION

EFFECTIVE RISK BASED CI AUTOMATION RUNS

<table>
<thead>
<tr>
<th>Priority</th>
<th>Execution Frequency</th>
<th>Criteria</th>
<th>Automation Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>With every deployment</td>
<td>Acceptance criteria for the build</td>
<td>Smoke</td>
</tr>
<tr>
<td>P1</td>
<td>With every deployment</td>
<td>Only when P0 cases pass</td>
<td>High Sanity</td>
</tr>
<tr>
<td>P2</td>
<td>Alternate deployment</td>
<td>Deployment with major changes</td>
<td>Sanity</td>
</tr>
<tr>
<td>P3</td>
<td>Start and end of sprint</td>
<td>Sign off criteria for the Testing Cycle</td>
<td>Regression</td>
</tr>
</tbody>
</table>
### CASE STUDY
(MOBILE – WEB APPLICATION FEATURE)

**Steps to calculate Risk Exposure for Use Case:**

- Identify cost factors and assign weights to each of them. (Range 1-3) For e.g. Visibility - 2
- Add weights for each use case aligning to each cost factor.
- Calculate Average Cost for each Use case.

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Cost factors</th>
<th>Maintenance</th>
<th>Visibility</th>
<th>Reputation</th>
<th>Total Cost</th>
<th>Average Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weight</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web_T5_01</td>
<td>Validate introduction menu on promotion top page</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>24 (2<em>3+2</em>3+3*4)</td>
<td>8 (24/3)</td>
</tr>
<tr>
<td>Web_T5_02</td>
<td>Validate downloading of application from promotion top page</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>31 (2<em>3+2</em>5+3*5)</td>
<td>10.33 (31/3)</td>
</tr>
</tbody>
</table>

- Identify probability factors and assign weights to each of them. (Range 1-3) For e.g. Complexity - 3
- Add weights for each use case aligning to each probability factor.
- Calculate Probability for each Use case.

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Defect Generators</th>
<th>Complexity</th>
<th>Changes</th>
<th>New Technology</th>
<th>Total Sum</th>
<th>Average Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weight</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web_T5_01</td>
<td>Validate introduction menu on promotion top page</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>20 (3<em>3+1</em>3+2*4)</td>
<td>0.83 (20/24)</td>
</tr>
<tr>
<td>Web_T5_02</td>
<td>Validate downloading of application from promotion top page</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>24 (3<em>4+1</em>4+2*4)</td>
<td>1 (24/24)</td>
</tr>
</tbody>
</table>

- Calculate Risk exposure

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Use Case</th>
<th>Average Cost</th>
<th>Probability</th>
<th>Risk Exposure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web_T5_01</td>
<td>Validate introduction menu on promotion top page</td>
<td>8 (24/3)</td>
<td>0.83 (20/24)</td>
<td>6.64</td>
</tr>
<tr>
<td>Web_T5_02</td>
<td>Validate downloading of application from promotion top page</td>
<td>10.33 (31/3)</td>
<td>1 (24/24)</td>
<td>10.33</td>
</tr>
</tbody>
</table>
**Steps to calculate Risk Exposure for Test Case:**

To calculate test case risk:

- Define Importance and assign weight factor to each of them.
  - High – 1.2
  - Medium – 1.0
  - Low – 0.8
- Assign Importance (High, Medium, and Low) to each test case.
- Calculate Test Case Risk.

<table>
<thead>
<tr>
<th>Test case id (Use case id)</th>
<th>Risk Exposure</th>
<th>Importance</th>
<th>Test Case Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1 (Web_TS_01)</td>
<td>6.64</td>
<td>High (1.2)</td>
<td>7.97</td>
</tr>
<tr>
<td>TC2 (Web_TS_01)</td>
<td>6.64</td>
<td>Medium (1.0)</td>
<td>6.64</td>
</tr>
<tr>
<td>TC3 (Web_TS_02)</td>
<td>10.33</td>
<td>Medium (1.0)</td>
<td>10.33</td>
</tr>
<tr>
<td>TC4 (Web_TS_02)</td>
<td>10.33</td>
<td>Low (0.8)</td>
<td>8.26</td>
</tr>
</tbody>
</table>

**Observation**

Hence TC1 has high risk than TC4 even if the use case it belongs to has lesser risk exposure.

Note: The values given for the importance depend on what we want to prioritize; the use case risk or the importance for the test case. Currently low is given 0.8, medium is given 1.0 and high is given 1.2, this will adjust the risk exposure for the use case with 20 %

**Automate the Test Case**

<table>
<thead>
<tr>
<th>Use Case Id (Total Test cases)</th>
<th>Smoke (5%)*</th>
<th>High-Sanity (15%)*</th>
<th>Sanity (30%)*</th>
<th>Regression (100%)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1 (Web_TS_01)</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>TC2 (Web_TS_01)</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

*rounded % of automatable test cases
A properly implemented RBT methodology helps assure that high-risk areas are tested first, then medium-risk and, finally, low-risk areas.

Dividing the test cycle to 4 quarters, the % of bugs found in each priority is as follows.

- In a sprint, the % of bugs found in each priority is as follows
  - 100% coverage of high risk features in the 1st sprint, and daily testing of the same further using automation with minimal effort, to keep an eye on the breakage.
  - In case of patch release as there is no time for exhaustive testing, Prioritize Execution of Test Scenarios based on Risk like Smoke, High-Sanity, Sanity and Regression if time permits. This diminishes the probability of immediate production issues by 80% (as per Pareto’s 80-20 rule, 80% of usage on 20% of the app)

As Risk-based testing is

- A Systematic approach
- Accommodates time constraints

Hereby ROI is

- High priority bugs can be found early in the cycle.
- Reduced effort, cost, duration
- Enables sizing remaining risk at any point

To be noted: It may leave undetected defects; hence acceptable risk related criteria need to be agreed upon from stake holders.
About the Author

Shabnam Choudhury
Test Architect
Product and Engineering Services
Happiest Minds

She is an expert in Test Automation of GUI and Web Services, along with Performance and Security Testing with robust Test Management skills. She has Extensive Exposure in domains such as Retailing and Auction, SPM, ERP, Healthcare, Logistics and Technology, Small Business and Advertisement Platform domains.

She started her career in software testing in the year 2000. She believes, “to be a tester is easy, but to be a good tester can be very challenging”. Hence in her strive to be a good tester she keeps exploring different ways to make testing efficient while delivering quality products. This has led her to discover the risk-based testing strategy which she has studied in-depth and has implemented a complete solution around it.

She has a variant industry experience working for multinational service providers like Mphasis-HP and Mindtree, and fortune 100 MNC like Yahoo! Currently while pursuing her EGMP program at IIM-B, realizes that the macro organizational behavior of IT services industry has turned the corner and needs to focus primarily on customer business problems and holistic, end-to-end IT solutions addressing them rather than products, feature sets, individual technology or expertise based offerings.

Summary

There are several ways to do Risk-Based Testing, from using risk to find the test, to using the risks as a way to prioritize what areas to focus during testing. The choice was a method to prioritize use cases and eventually in a risk analysis. Test cases are designed from the most risky use cases and resources for testing are estimated. This method was chosen because it seemed to be a good way to prioritize the tests.

References

http://en.wikipedia.org/wiki/Talk:Risk-based_testing
www.stickyminds.com/sitewide.asp?Function=edetail&ObjectName
http://www.methodsandtools.com/archive/archive.php?id=31

www.happiestminds.com

About Happiest Minds Technologies

Happiest Minds enables Digital Transformation for enterprises and technology providers by delivering seamless customer experience, business efficiency and actionable insights through an integrated set of disruptive technologies: big data analytics, internet of things, mobility, cloud, security, unified communications, etc. Happiest Minds offers domain centric solutions, IPs in IT Services, Product Engineering, Infrastructure Management and Security. These services have applicability across industry sectors such as retail, CPG, e-commerce, banking, insurance, hi-tech, engineering R&D, manufacturing, automotive and travel/transportation/hospitality. As a Mindful IT Company, the focus is on ‘Being Mindful’ and ‘Doing Mindful’ which involves perceiving immersively, processing non-judgmentally and performing empathetically. 60 minutes in a week is committed towards inculcating a mindful approach within the organization, using a select set of tools and techniques. Headquartered in Bangalore, India, Happiest Minds has operations in the US, UK, The Netherlands, Australia, Middle East and Turkey.