Help! I am Drowning in 2 Week Sprints

Please Tell me What NOT to Test!
President of Mary Thorn Consulting, LLC

Chief storyteller of the book The Three Pillars of Agile Testing and Quality, Mary Thorn is owner of Mary Thorn Consulting in Raleigh, NC. During her more than twenty years of experience with healthcare, financial, and HR SaaS-based products, Mary has held director, manager- and contributor-level positions in software development organizations.

A seasoned leader and coach in agile and testing methodologies, Mary has direct experience building and leading teams through large scale agile transformations.

Mary's special expertise is a combination of agile, testing, DevOps, and agile scaling skills that her clients find incredibly valuable.

She is also a frequent speaker, teacher and author. You can connect with Mary via LinkedIn here: https://www.linkedin.com/in/marythorn/
Agenda

1. Introduction
2. 3 Amigos
3. Risked Based Testing
4. Test Ideas
5. Test Case Gaps
6. Pareto
7. All Pairs
8. Wrap Up!
3 Amigos
3-Amigos

- Coined by George Dinwiddie

- Swarming around the User Story by:
  - Developer(s)
  - Tester(s)
  - Product Owner

- Conversation device – reminder for collaboration amongst relevant team members
Are you enabling the bad behavior ……Are you a HERO?????
Risk-Based Testing Background

- It starts with the realization that you can’t test everything – ever!
  
  100% coverage being a long held myth in software development

- There are essentially 5 steps in most of the models
  1. Decompose the application under test into areas of focus
  2. Analyze the risk associated with individual areas – technical, quality, business, schedule
  3. Assign a risk level to each component
  4. Plan test execution, based on your SDLC, to maximize risk coverage
  5. Reassess risk at the end of each testing cycle
Risk–Based Testing Background

● Risk–Based Testing is effectively a risk mitigation technique
  – Not a prevention technique

● It's about trade-offs
  – Human and physical resources
  – Ratio's between Producers (Developers) and Consumers (Testers)
  – Time
  – Rework (retesting & verification)
  – Quality – Coverage vs. Delivery
  – Visibility into the trade-offs
● What are they?

- Risked based test planning technique
- Created by Rob Sabourin
- Replaces traditional waterfall test plan in Agile.
# Test Ideas

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Focus</th>
<th>Test Objective</th>
<th>Business Importance</th>
<th>Technical Risk</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>TID0010</td>
<td>Capabilities</td>
<td>Produce correct box of chocolates based on manifest</td>
<td>HIGH</td>
<td>SIGNIFICANT</td>
<td>5</td>
</tr>
<tr>
<td>TID0100</td>
<td>Failure Modes</td>
<td>What if it runs out of paper</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>5</td>
</tr>
<tr>
<td>TID0170</td>
<td>Usage Scenarios</td>
<td>Can operator stop system</td>
<td>MEDIUM</td>
<td>SIGNIFICANT</td>
<td>5</td>
</tr>
<tr>
<td>TID0260</td>
<td>Outcome</td>
<td>Can we produce correct daily reports</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>5</td>
</tr>
<tr>
<td>TID0020</td>
<td>Failure Modes</td>
<td>Are there gaps in a box</td>
<td>MEDIUM</td>
<td>SIGNIFICANT</td>
<td>4</td>
</tr>
<tr>
<td>TID0040</td>
<td>Capabilities</td>
<td>Can it fill boxes with mixed chocolates</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0110</td>
<td>Failure Modes</td>
<td>What if it runs out of other supplies</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0140</td>
<td>Failure Modes</td>
<td>What if operator enters incorrect data in manifest</td>
<td>MEDIUM</td>
<td>SIGNIFICANT</td>
<td>4</td>
</tr>
<tr>
<td>TID0180</td>
<td>Usage Scenarios</td>
<td>Can emergency repairs be done without stopping production</td>
<td>HIGH</td>
<td>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0200</td>
<td>Usage Scenarios</td>
<td>Can production be resumed after emergency repairs</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0270</td>
<td>Outcome</td>
<td>Can we product correct monthly reports</td>
<td>MEDIUM</td>
<td>SIGNIFICANT</td>
<td>4</td>
</tr>
<tr>
<td>TID0290</td>
<td>Input</td>
<td>Can we vary boxes with different speeds of conveyors</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0030</td>
<td>Capabilities</td>
<td>Can it wrap chocolates with ribbons</td>
<td>LOW</td>
<td>SIGNIFICANT</td>
<td>3</td>
</tr>
<tr>
<td>TID0050</td>
<td>Capabilities</td>
<td>Can it fill boxes with one type of chocolate</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0070</td>
<td>Input</td>
<td>Vary Combinations of Ribbons, Paper, Boxes</td>
<td>MEDIUM</td>
<td>MINIMAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0120</td>
<td>Failure Modes</td>
<td>What if machine drops chocolate but continues to try wrapping (in process)</td>
<td>LOW</td>
<td>SIGNIFICANT</td>
<td>3</td>
</tr>
<tr>
<td>TID0130</td>
<td>Failure Modes</td>
<td>What if operator enters WRONG manifest</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0150</td>
<td>Failure Modes</td>
<td>What if something else in conveyor belt not chocolate</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0160</td>
<td>Quality Factors</td>
<td>Is system easy to stop</td>
<td>LOW</td>
<td>SIGNIFICANT</td>
<td>3</td>
</tr>
<tr>
<td>TID0210</td>
<td>Usage Scenarios</td>
<td>Can loader load supplies</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0230</td>
<td>Usage Scenarios</td>
<td>Can loader add ribbons while production is in progress</td>
<td>MEDIUM</td>
<td>MINIMAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0250</td>
<td>Outcome</td>
<td>Can we produce correct batch report</td>
<td>LOW</td>
<td>SIGNIFICANT</td>
<td>3</td>
</tr>
<tr>
<td>TID0300</td>
<td>Input</td>
<td>Can we have batches with high percentage of one type of chocolate</td>
<td>MEDIUM</td>
<td>NEUTRAL</td>
<td>3</td>
</tr>
<tr>
<td>TID0060</td>
<td>Capabilities</td>
<td>Can we support different sizes of chocolates in the same box</td>
<td>LOW</td>
<td>NEUTRAL</td>
<td>2</td>
</tr>
<tr>
<td>TID0080</td>
<td>Failure Modes</td>
<td>Mechanical failure does it handle it gracefully</td>
<td>MEDIUM</td>
<td>MINIMAL</td>
<td>2</td>
</tr>
<tr>
<td>TID0190</td>
<td>Usage Scenarios</td>
<td>Can emergency repairs be done stopping production</td>
<td>LOW</td>
<td>NEUTRAL</td>
<td>2</td>
</tr>
<tr>
<td>TID0240</td>
<td>Outcome</td>
<td>Produce correct reports</td>
<td>MEDIUM</td>
<td>MINIMAL</td>
<td>2</td>
</tr>
<tr>
<td>TID0280</td>
<td>Failure Modes</td>
<td>Will system ever run hot enough to melt the chocolate</td>
<td>LOW</td>
<td>NEUTRAL</td>
<td>2</td>
</tr>
</tbody>
</table>
Test Ideas - Sources

- Capabilities
- Failure Modes
- Quality Factors
- Usage Scenarios
- Creative Ideas
- States
- Data
- Environments
- White Box
- Taxonomies
Test Ideas

● How to find them?
  – Does system do what it is suppose to do?
  – Does the system do things it is not supposed to?
  – How can the system break?
  – How does the system react to it’s environment?
  – What characteristics must the system have?
  – Why have similar systems failed?
  – How have previous projects failed?
Test Ideas - Process

- Life of a test idea
  - Comes into existence
  - Clarified
  - Prioritized
    - Test Now (before further testing)
    - Test before shipping
    - Nice to have
    - May be of interest in some future release
    - Not of interest in current form
    - Will never be of interest
  - Integrate into a testing objective
Test Ideas – 3 Amigos

Test Triage Meeting

– Review Context
  • Business – with PO
  • Technical – With Developer

– Add or remove tests

– Agree to where the cut line is
Test Case Gap Analysis
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature 1.1</td>
<td>None</td>
<td>None</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Critical</td>
<td>High</td>
<td>1</td>
<td>2</td>
<td>UI</td>
<td>A, M, S</td>
</tr>
<tr>
<td>Feature 1.2</td>
<td>Partial</td>
<td>None</td>
<td>None</td>
<td>Y</td>
<td>N</td>
<td>Med</td>
<td>Low</td>
<td>3</td>
<td>COMPLT</td>
<td>N/A</td>
<td>M</td>
</tr>
<tr>
<td>Functional Area 2</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature 2.1</td>
<td>Full</td>
<td>None</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>High</td>
<td>High</td>
<td>COMPLT</td>
<td>2</td>
<td>UI, INT</td>
<td>A, M, S</td>
</tr>
<tr>
<td>Feature 2.2</td>
<td>None</td>
<td>Partial</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Critical</td>
<td>Med</td>
<td>1</td>
<td>3</td>
<td>INT</td>
<td>A, M, S</td>
</tr>
<tr>
<td>Feature 2.3</td>
<td>None</td>
<td>Partial</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Med</td>
<td>High</td>
<td>3</td>
<td>4</td>
<td>INT</td>
<td>A, M, S</td>
</tr>
<tr>
<td>Functional Area 3</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature 3.1</td>
<td>Full</td>
<td>None</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>High</td>
<td>Med</td>
<td>COMPLT</td>
<td>5</td>
<td>INT</td>
<td>A, M, S</td>
</tr>
<tr>
<td>Feature 3.2</td>
<td>Full</td>
<td>None</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Critical</td>
<td>Low</td>
<td>COMPLT</td>
<td>6</td>
<td>UI</td>
<td>A, M</td>
</tr>
<tr>
<td>Feature 3.3</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Y</td>
<td>N</td>
<td>High</td>
<td>Low</td>
<td>2</td>
<td>COMPLT</td>
<td>N/A</td>
<td>M</td>
</tr>
<tr>
<td>Feature 3.4</td>
<td>None</td>
<td>None</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Low</td>
<td>Low</td>
<td>4</td>
<td>7</td>
<td>UI</td>
<td>A, M</td>
</tr>
</tbody>
</table>

*Note - The Blue represents columns that are calculated.*
Pareto Principle
Italian economist Vilfredo Pareto observed that:

*For many phenomena, 80% of the consequences stem from 20% of the causes*

When analyzing personal wealth distribution in Italy.

- Also known as the **80-20 rule**, the **law of the vital few**, and the **principle of factor sparsity**
- Joseph Duran brought the principle forward as a potential quality management technique
- In probability theory referenced as a Pareto distribution
Pareto Principle “Thinking” Examples

● In a Toyota Prius warehouse –
  - 20% of the component boxes take up 80% of the space
  - 20% of the components make up 80% of the overall vehicle cost

● In software applications –
  - 20% of the application code produces 80% of the defects
  - 20% of the developers produce 80% of the defects
  - 20% of the test cases (ideas) find 80% of the defects
  - 20% of the test cases (ideas) take 80% of your time to design & test
  - 20% of the product will be used by 80% of the customers
  - 20% of the requirements will meet 80% of the need
Pareto Principle “Thinking” Examples

● Leads to the notion of defect clustering. Many have observed that software bugs will cluster in specific modules, classes, components, etc.

● Think in terms of stable or well made components versus error-prone, unstable, and fragile components. Which ones should receive most of your attention? Do the areas remain constant?

● Often, complexity plays a large part in the clustering. Either solution (true) complexity OR gold-plating (favored) complexity.
Open Defects per Functional Area “Rolling” Pareto Chart

Open Defects per Functional Area

# of Defects


Jan 1-15
Jan 16-31
Feb 1-14
Feb 15-28
Mar 1-15
Mar 16-30

Install & Config | Internal files | Dbase | Reporting | R-time analysis | Off-line analysis | GUI | Help & docs

Sample Pareto Chart
Pareto Principal  Step 1 – Application Partitioning

- The first major challenge to Pareto-Based risk analysis is meaningfully partitioning your application. Here are some guidelines –
  - Along architectural boundaries – horizontally and/or vertically
  - Along design boundaries
  - At interface points – (API, SOA points, 3’rd party product integrations, external data acquisition points)

- Always do this in conjunction with the development team
- The partitioned areas need to be balanced – in approximate size & complexity
- Shoot for 5-12 meaningful areas for tracking
Pareto Principal Step 2 – Defect Tracking Setup

- Modify your DTS to support specific application component areas

- During triage, effectively identify and assign defect repairs and enhancements to component areas
  - Early on, testers will need development help to clearly identify root component areas (about 20% of the time)

- If you have historical defect data (w/o partitioning), you can run an application analysis workshop to partition data (post release) for future predictions

*It does require discipline and a little extra effort*…
● Sometimes you don’t have the time to start Pareto tracking before starting a project, so reflectively analyze Pareto for future planning –

- Decompose your application or a sub-component of it if pressed for time
- Gather defects surfaced
- Gather your team (developers, testers)
- Discuss locale for each bug and create distribution
- Off-line create your curves and publish insights for the “next” release
- Can also help fine-tune decomposition areas and train the test team in defect localization
Pareto Principal Step 3 – Observations & Adjustments

● Project trending at a component level
  - Look for migration of risk and make adjustments
  - Look for stabilization or regressions (risk)
  - Identify high risk & low risk component areas at a project level
  - Map component rates to overall project goals
  - Trend open & high priority defects at a component level
  - Track or predict project “done”ness at a component level

● Weekly samples of 20% component focus areas – looking for risk migration
  - Sample weekly, then adjust focus across your testing cycles or iterations
Pareto Principal Tools

- Excel can be used to display Pareto like charts, with the cumulative percent trend needing to be simulated.

- There are other packages available that will properly calculate & display Pareto Charts for you. Keeping in mind that it’s a Six Sigma tool, many are associated with supporting it.
All-Pairs Testing

- All-Pairs testing is a method of handling large scale combinatorial testing problems
  - Also referred to as Pairwise, Orthogonal Arrays, and Combinatorial Method
  - It identifies all pairs of variables that need to be tested in tandem – to achieve reasonably high coverage.

- Three primary references include –
  - Lee Copeland – *A Practitioners Guide to Software Test Design*
  - James Bach – Open Source, AllPairs implementation
  - Bernie Berger – *Efficient Testing with All-Pairs* 2003 StarEast paper
All-Pairs Testing Interoperability Testing

<table>
<thead>
<tr>
<th>Client OS</th>
<th>Browser</th>
<th>App Server</th>
<th>Server OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win NT</td>
<td>IE 7</td>
<td>WebSphere</td>
<td>Win NT</td>
</tr>
<tr>
<td>Win Vista</td>
<td>IE 8</td>
<td>WebLogic</td>
<td>Linux</td>
</tr>
<tr>
<td>Linux</td>
<td>Safari 2</td>
<td>Apache</td>
<td></td>
</tr>
<tr>
<td>MAC</td>
<td>Chrome</td>
<td>IIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FireFox 3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FireFox 3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Opera 9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- One **sweet spot** area for All-Pairs testing is interoperability. Something that faces web application testers every day.

- In this example, we want to examine browser compatibility across this specific set of system software levels – focusing on the browser.

- Considering all combinations, there are \((4 \times 7 \times 4 \times 2)\) or 224 possible test cases for the example.
All-Pairs Testing Example

● In All-Pairs test design we are concerned with
  - Variables of a system
  - Possible values that variables could take

● Then we generate a list of test cases that represent the pairing of variables (all pairs) as the most interesting set of test cases to approach in your test design
Hexawise Testing Example

- Using pair-wise on the previous example, we would identify 28 test cases as an alternative to the 224 for absolute coverage.

- We’d then use this output as guidance when designing our test cases.

Note the * indicates a don’t care for this variable

<table>
<thead>
<tr>
<th>OS</th>
<th>Server OS</th>
<th>Browser</th>
<th>Web servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows xp</td>
<td>Windows XP</td>
<td>IE7</td>
<td>Apache</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Linux</td>
<td>IE7</td>
<td>IIS</td>
</tr>
<tr>
<td>Linux</td>
<td>Windows XP</td>
<td>IE7</td>
<td>Weblogic</td>
</tr>
<tr>
<td>MAC</td>
<td>Linux</td>
<td>IE8</td>
<td>Apache</td>
</tr>
<tr>
<td>Windows xp</td>
<td>Windows XP</td>
<td>IE8</td>
<td>Websphere</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Linux</td>
<td>IE8</td>
<td>Apache</td>
</tr>
<tr>
<td>Linux</td>
<td>Windows XP</td>
<td>IE8</td>
<td>Weblogic</td>
</tr>
<tr>
<td>MAC</td>
<td>Linux</td>
<td>IE8</td>
<td>IIS</td>
</tr>
<tr>
<td>Windows xp</td>
<td>Linux</td>
<td>Firefox 3.0</td>
<td>IIS</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Windows XP</td>
<td>Firefox 3.0</td>
<td>Weblogic</td>
</tr>
<tr>
<td>Linux</td>
<td>Linux</td>
<td>Firefox 3.0</td>
<td>Apache</td>
</tr>
<tr>
<td>MAC</td>
<td>Windows XP</td>
<td>Firefox 3.0</td>
<td>Websphere</td>
</tr>
<tr>
<td>Windows xp</td>
<td>Windows XP</td>
<td>Firefox 3.5</td>
<td>Weblogic</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Linux</td>
<td>Firefox 3.5</td>
<td>IIS</td>
</tr>
<tr>
<td>Linux</td>
<td>*</td>
<td>Firefox 3.5</td>
<td>Weblogic</td>
</tr>
<tr>
<td>MAC</td>
<td>*</td>
<td>Firefox 3.5</td>
<td>Apache</td>
</tr>
<tr>
<td>Windows xp</td>
<td>Windows XP</td>
<td>Safari</td>
<td>Apache</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Linux</td>
<td>Safari</td>
<td>Websphere</td>
</tr>
<tr>
<td>Linux</td>
<td>*</td>
<td>Safari</td>
<td>IIS</td>
</tr>
<tr>
<td>MAC</td>
<td>*</td>
<td>Safari</td>
<td>Weblogic</td>
</tr>
<tr>
<td>Windows xp</td>
<td>Windows XP</td>
<td>Chrome</td>
<td>Apache</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Linux</td>
<td>Chrome</td>
<td>Websphere</td>
</tr>
<tr>
<td>Linux</td>
<td>*</td>
<td>Chrome</td>
<td>IIS</td>
</tr>
<tr>
<td>MAC</td>
<td>*</td>
<td>Chrome</td>
<td>Weblogic</td>
</tr>
<tr>
<td>Windows xp</td>
<td>Windows XP</td>
<td>Opera</td>
<td>Apache</td>
</tr>
<tr>
<td>Windows vista</td>
<td>Linux</td>
<td>Opera</td>
<td>Websphere</td>
</tr>
<tr>
<td>Linux</td>
<td>*</td>
<td>Opera</td>
<td>IIS</td>
</tr>
<tr>
<td>MAC</td>
<td>*</td>
<td>Opera</td>
<td>Weblogic</td>
</tr>
</tbody>
</table>
All-Pairs Testing Intent

● Defects
  - The hope of All-Pairs testing is that by running from 1-20% of your test cases you'll find 70% - 85% of your overall defects

● Coverage
  - By way of example (Cohen) a set of 300 randomly selected test cases provided 67% statement coverage and 58% decision coverage for an application. While 200 All-Pairs derived test cases provided 92% statement and 85% decision coverage.

● Important tests can be missed. Use sound judgment when creating tests and add as required
• All-Pairs is simply a tool in your test design arsenal. Don’t use it alone or blindly!

• You won’t find all of your bugs exclusively using this tool!

• Often the strategy is to use All-Pairs to establish your baseline set of test cases
  – Then analyze other business critical combinations and add risk-based tests as appropriate
All-Pairs Testing Brainstorming Value Proposition

- What are some testing area opportunities for All-Pairs?
  - UI type input / output variation testing (functional)
  - Cross-platform (interoperability) testing
  - Anything with high numbers of variables
  - Scenario based testing, with path (variable) variation

- What are not?
  - Performance testing, and most other non-functional testing
  - Exploration
  - Using it solely to derive your test cases
A few cautions from James Bach & Patrick J. Schroeder in paper – *Pairwise Testing: A Best Practice That Isn’t*

- You don’t select the right values to test with
- When you don’t have a good enough oracle
- When highly probable combinations get too little attention
- When you don’t know how the variables interact
Let’s take a look at www.hexawise.com

- We’ll be “driving”, but we expect you to login in later and try things out…

Review:
- Implementation of our earlier platform table
- Implementation of Bernie Berger’s example
Wrapping up!

● There are a lot of old and new testing techniques that can be used to enhance your agile testing journey.

● Here we discussed just a few…

● Read blogs, go to conferences, read our book😊