HELP!!!
I am Drowning in 2 Week Sprints

How do I determine what not to test?
During her more than 20 years of experience with financial, healthcare, and SaaS-based products, Mary has held VP, Director, and Manager level positions in various 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 expertise is a combination of agile scaling, agile testing, and DevOps that her clients find incredibly valuable.

She is also Chief storyteller of the book *The Three Pillars of Agile Testing and Quality*, and avid keynote and conference speaker on all things agile and agile testing.
1. Introduction
2. Words of Wisdom
3. 3 Amigos
4. Risked Based Testing
5. Test Ideas
6. Test Case Gaps
7. Pareto
8. All Pairs
9. Wrap Up!
Words of **Wisdom**
• Don’t Enable the Bad Behavior
• Risk Mitigation Techniques, NOT Prevention Techniques
• Entire Team is Trained
• Test Design or Scrummerfall Issue?
3 Amigos
Coined by George Dinwiddie

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

Conversation Device
- Reminder for collaboration amongst relevant team members
Risk Based Testing
5 Step Model:

1. **Decompose** the application into areas of focus
2. **Analyze the risk** associated with individual areas
3. **Assign a risk level** to each component
4. **Plan test execution** to maximize risk coverage
5. **Reassess risk** at the end of each testing cycle

It starts with the realization that you can’t test everything – ever! *100% coverage being a long held myth in software development*
Risk-Based Testing is a Risk Mitigation Technique

- Not a prevention technique like test automation

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
<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>HIGH</td>
<td>SIGNIFICANT</td>
<td>5</td>
</tr>
<tr>
<td>TID0170</td>
<td>Usage Scenarios</td>
<td>Can operator stop system</td>
<td>HIGH</td>
<td>SIGNIFICANT</td>
<td>5</td>
</tr>
<tr>
<td>TID0260</td>
<td>Outcome</td>
<td>Can we produce correct daily reports</td>
<td>LOW</td>
<td>SIGNIFICANT</td>
<td>5</td>
</tr>
<tr>
<td>TID0220</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>HIGH</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>SIGNIFICANT</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>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0180</td>
<td>Usage Scenarios</td>
<td>Can emergency repairs be done without stopping production</td>
<td>MEDIUM</td>
<td>SIGNIFICANT</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>LOW</td>
<td>NEUTRAL</td>
<td>4</td>
</tr>
<tr>
<td>TID0050</td>
<td>Capabilities</td>
<td>Can it wrap chocolates with ribbons</td>
<td>LOW</td>
<td>SIGNIFICANT</td>
<td>3</td>
</tr>
<tr>
<td>TID0070</td>
<td>Input</td>
<td>Vary Combinations of Ribbons, Paper, Boxes</td>
<td>HIGH</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 manifests?</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>NEUTRAL</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>HIGH</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>TID0050</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>Mechnical 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>TID0200</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>
• Capabilities
• Failure Modes
• Quality Factors
• Usage Scenarios
• Creative Ideas
• States
• Data
• Environments
• White Box
• Taxonomies
How to Find Them

- Does the system Do...
  - What it’s Supposed to?
  - What it’s 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?
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 a Future Release
  - Not of Interest in Current Form
  - Will never be of Interest
- Integrate into a Testing Objective
Test Triage Meeting

- Review Context
  - Business – with PO
  - Technical – With Developer
- Add or Remove Tests
- Agree to Where the Cut Line is
WHEN TO USE TEST IDEAS

✓ Bring to Planning

✓ Bring to the Daily Scrum

✓ When you have Multiple Scrum Teams

✓ To Understand Where the Test Risk is

✓ When You Need to Prioritize Hardening
Test Case Gap Analysis
Are you **drowning** in Automation debt?

<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>3</td>
<td>Functional Area 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>
</tr>
<tr>
<td>4</td>
<td>Feature 1.1</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>CMPLT</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>Feature 1.2</td>
<td>Full</td>
<td>None</td>
<td>N</td>
<td>Y</td>
<td>High</td>
<td>High</td>
<td>CMPLT</td>
<td>2</td>
<td>N/A</td>
<td>M</td>
</tr>
<tr>
<td>6</td>
<td>Functional Area 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>
</tr>
<tr>
<td>7</td>
<td>Feature 2.1</td>
<td>None</td>
<td>None</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>High</td>
<td>CMPLT</td>
<td>5</td>
<td>INT</td>
<td>A, M, S</td>
</tr>
<tr>
<td>8</td>
<td>Feature 2.2</td>
<td>Full</td>
<td>None</td>
<td>N</td>
<td>Y</td>
<td>Med</td>
<td>Low</td>
<td>CMPLT</td>
<td>6</td>
<td>UI</td>
<td>A, M</td>
</tr>
<tr>
<td>9</td>
<td>Feature 2.3</td>
<td>None</td>
<td>Partial</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>High</td>
<td>Low</td>
<td>2</td>
<td>CMPLT</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>Functional Area 3</td>
<td>None</td>
<td>Partial</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>
</tr>
</tbody>
</table>

*Note - The Blue represents columns that are calculated.*
Pareto Principle
When analyzing personal wealth distribution in Italy, Italian economist Vilfredo Pareto observed that...

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

Also known as the:

• 80-20 rule
• Law of the Vital Few
• Principle of Factor Sparsity
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
• Leads to the notion of **defect clustering**
• **Stable components** versus error-prone, unstable, and fragile components
• **Complexity** plays a large part in the clustering
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
• Modify your Defect Tracking System 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*...
• 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
All Pairs
• Pairwise Testing is a test design technique that delivers 100% test coverage.

• Designed to execute all possible discrete combinations of each pair of input parameters.

• Method of handling large scale combinatorial testing problems
  • Aka: Pairwise, Orthogonal Arrays, and Combinatorial Method
  • it identifies all pairs of variables to be tested in tandem – to achieve reasonably high coverage.
• 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.
- 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
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, 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

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, I often use Test ideas to do this.
### 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
ALL-PAIRS TESTING FAILS WHEN...

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
• **PICT** – ‘Pairwise Independent Combinatorial Testing’, provided by Microsoft Corp.
• **IBM FoCuS** – ‘Functional Coverage Unified Solution’, provided by IBM.
• **ACTS** – ‘Advanced Combinatorial Testing System’, provided by NIST, an agency of the US Government.
• **Hexawise**
• **Jenny**
• **Pairwise** by Inductive AS
• **VPTag** free All-Pair Testing Tool
Scrummerfall / Wagile
• Did you plan it in?
• Are your stories too big? Are they vertical sliced?
• Are your team members multi-skilled and team players so they can pick up any task?
• Have you tried a code freeze?
• Do you have WIP limits? (Silver bullet)
• There are a lot of old and new testing techniques that can used to enhance your agile testing journey.

• Here we discussed just a few...

• Read blogs, go to conferences, read Bob Galen and I’s book😊