Help! I am Drowning in 2 Week Sprints

Please Tell me What NOT to Test!


About me

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/

2
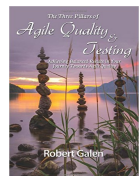
Agenda

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

Agile Testing

3

---

3 Amigos

4

3-Amigos

- Coined by George Dinwiddie
  - http://rgalen.com/agile-training-news/2014/4/13/3-amigos-in-agile-teams
- Swarming around the User Story by:
  - Developer(s)
  - Tester(s)
  - Product Owner
- Conversation device – reminder for collaboration amongst relevant team members

The Three Amigos!

5

Are you enabling the bad behavior ……Are you a HERO?????

Risked Based Testing

6

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

7

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

8

Test Ideas

MaryThorn Consulting LLC

●What are they?

– Risked based test planning technique

– Created by Rob Sabourin

– Replaces traditional waterfall test plan in Agile.

9

Test Ideas

MaryThorn Consulting LLC

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

10

Test Ideas - Sources

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

11

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?

12

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
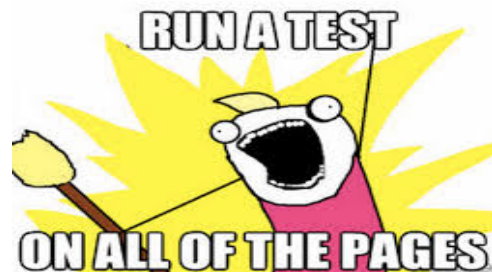 – Integrate into a testing objective

Test Your Big Idea

13

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

RUN A TEST ON ALL OF THE PAGES

14

Test Case Gap Analysis

Test Case Gap Analysis

|	A	B	C	D	E	F	G	H	I	J	K	L
		Current State - Test Cases (None, Partial, Full)		Future State - Automated Test Cases (None, Yes)	Manual Gap Exists	Automation Gap Exists	Severity (Critical, High, Med, Low)	Priority (High, Med, Low)	Manual Gap Ranking (1 - 4)	Gap Ranking (1 - 7)	Automation Type (User Interface, Integration (i.e. Services))	Regression Type (Automate/Manual/Smoke)
1												
2	Functional Areas/Area Path	Manual	Automated									
3	Functional Area 1											
4	Feature 1.1	None	None	Y	Y	Y	Critical	High	1	1	UI	A, M, S
5	Feature 1.2	Partial	None	None	Y	N	Med	Low	3	CMPLT	N/A	M
6	Functional Area 2									N/A		
7	Feature 2.1	Full	None	Y	N	Y	High	High	CMPLT	2	UI, INT	A, M, S
8	Feature 2.2	None	Partial	Y	Y	Y	Critical	Med	1	3	INT	A, M, S
9	Feature 2.3	None	Partial	Y	Y	Y	Med	High	3	4	INT	A, M, S
10	Functional Area 3									N/A		
11	Feature 3.1	Full	None	Y	N	Y	High	Med	CMPLT	5	INT	A, M, S
12	Feature 3.2	Full	None	Y	N	Y	Critical	Low	CMPLT	6	UI	A, M
13	Feature 3.3	None	None	None	Y	N	High	Low	2	CMPLT	N/A	M
14	Feature 3.4	None	None	Y	Y	Y	Low	Low	4	7	UI	A, M
15												
16												
17	*Note - The Blue represents columns that are calculated.											

Pareto Principle

17

Pareto Principle

Italian economist Vilfredo Pareto observed that -
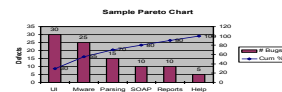
For many phenomena, 80% of the consequences stem from 20% of the causes

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

18

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
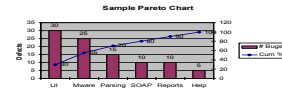
19

Pareto Principle "Thinking" Examples
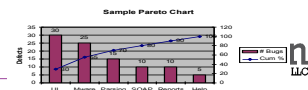
- Leads to the notion of defect clustering. Many have observed that software bugs will cluster in specific modules, classes, components, etc.

- 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.
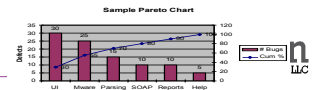
20

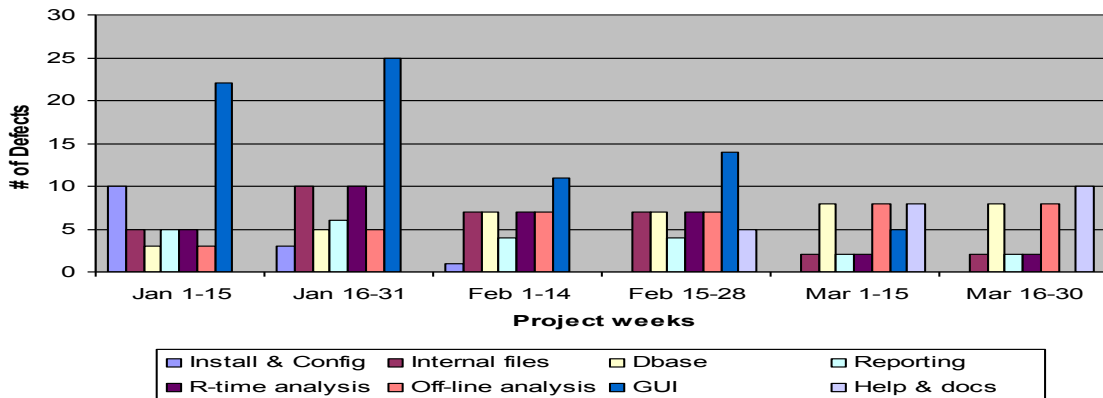**Open Defects per Functional Area Trending – Pareto (80:20 Rule) Chart**
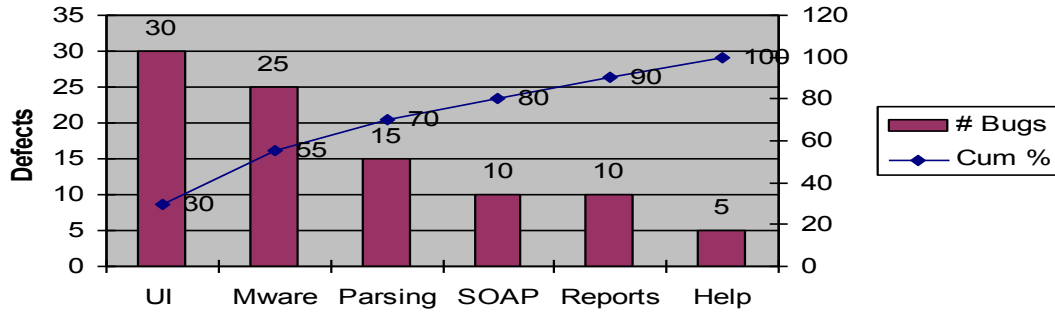
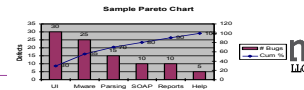**Open Defects per Functional Area "Rolling" Pareto Chart**

**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

23

**Pareto Principal Step 2 – Defect Tracking Setup**
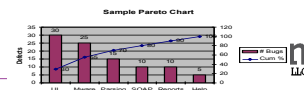
- Modify your DTS to support specific application component areas

- 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…

24

**Pareto Principal Application Analysis Workshop**
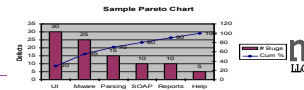
- Sometimes you don't have the time to start Pareto tracking before starting a project, so reflectively analyze Pareto for future planning –

  - 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

25

Pareto Principal Step 3 – Observations & Adjustments
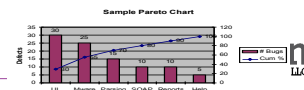
- Project trending at a component level
  - Look for migration of risk and make adjustments
  - Look for stabilization or regressions (risk)
  - Identify high risk & low risk component areas at a project level
  - Map component rates to overall project goals
  - Trend open & high priority defects at 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

26

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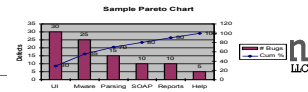
- Excel can be used to display Pareto like charts, with the cumulative percent trend needing to be simulated

- 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.

27

All Pairs

28

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

29

All-Pairs Testing Interoperability Testing

Client OS	Browser	App Server	Server OS
Win NT	IE 7	WebSphere	Win NT
Win Vista	IE 8	WebLogic	Linux
Linux	Safari 2	Apache	
MAC	Chrome	IIS	
	FireFox 3.0		
	FireFox 3.5		
	Opera 9		

- 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 x 7 x 4 x 2) or 224 possible test cases for the example.

30

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

31

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*

OS	Server OS	Browser	Web servers
Windows xp	Windows XP	IE7	Apache
Windows vista	Linux	IE7	Websphere
Linux	Windows XP	IE7	IIS
MAC	Linux	IE7	Weblogic
Windows xp	Windows XP	IE8	Websphere
Windows vista	Linux	IE8	Apache
Linux	Windows XP	IE8	Weblogic
MAC	Linux	IE8	IIS
Windows xp	Linux	Firefox 3.0	IIS
Windows vista	Windows XP	Firefox 3.0	Weblogic
Linux	Linux	Firefox 3.0	Apache
MAC	Windows XP	Firefox 3.0	Websphere
Windows xp	Windows XP	Firefox 3.5	Weblogic
Windows vista	Linux	Firefox 3.5	IIS
Linux	*	Firefox 3.5	Websphere
MAC	*	Firefox 3.5	Apache
Windows xp	Windows XP	Safari	Apache
Windows vista	Linux	Safari	Websphere
Linux	*	Safari	IIS
MAC	*	Safari	Weblogic
Windows xp	Windows XP	Chrome	Apache
Windows vista	Linux	Chrome	Websphere
Linux	*	Chrome	IIS
MAC	*	Chrome	Weblogic
Windows xp	Windows XP	Opera	Apache
Windows vista	Linux	Opera	Websphere
Linux	*	Opera	IIS
MAC	*	Opera	Weblogic

32

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

33

All-Pairs Testing Intent

- 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

34

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

  - Performance testing, and most other non-functional testing
  - Exploration
- What are not?
  - Using it solely to derive your test cases

35

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

36

All-Pairs Tools

- 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
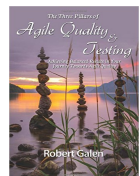 – Implementation of Bernie Berger's example

37

Wrapping up!

- 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 our book☺

38