medidata

THE ARCHITECTURE OF HOPE™

The Testing Profession:

The Good, the Bad and the Ugly

An opinionated tirade of strong opinions and rants

Paul Holland,

Sr. Director, Test Engineering at Dassault Systèmes

Agile Testing Days, November, 2018

27 November 2019

Disclaimer

- The opinions expressed during this workshop are my own and not necessarily those of Dassault Systèmes.

My Background

- Sr. Director Test Engineering at Medidata since 2016
- S/W Testing consultant since 2012-2016
- 20+ years testing telecommunications equipment and reworking test methodologies at Alcatel-Lucent
- 15+ years as a test manager/director
- Frequent conference speaker
- Teacher of S/W testing for the past 7 years
- Teacher of Rapid Software Testing
- Military Helicopter pilot – Canadian Sea Kings

Hot Topic List

Language

Language

- People often use language in non-precise ways
- Some then complain about others pointing out their sloppiness
- They say that we are just arguing about semantics

which is exactly correct

Language

- Definition of Semantics:

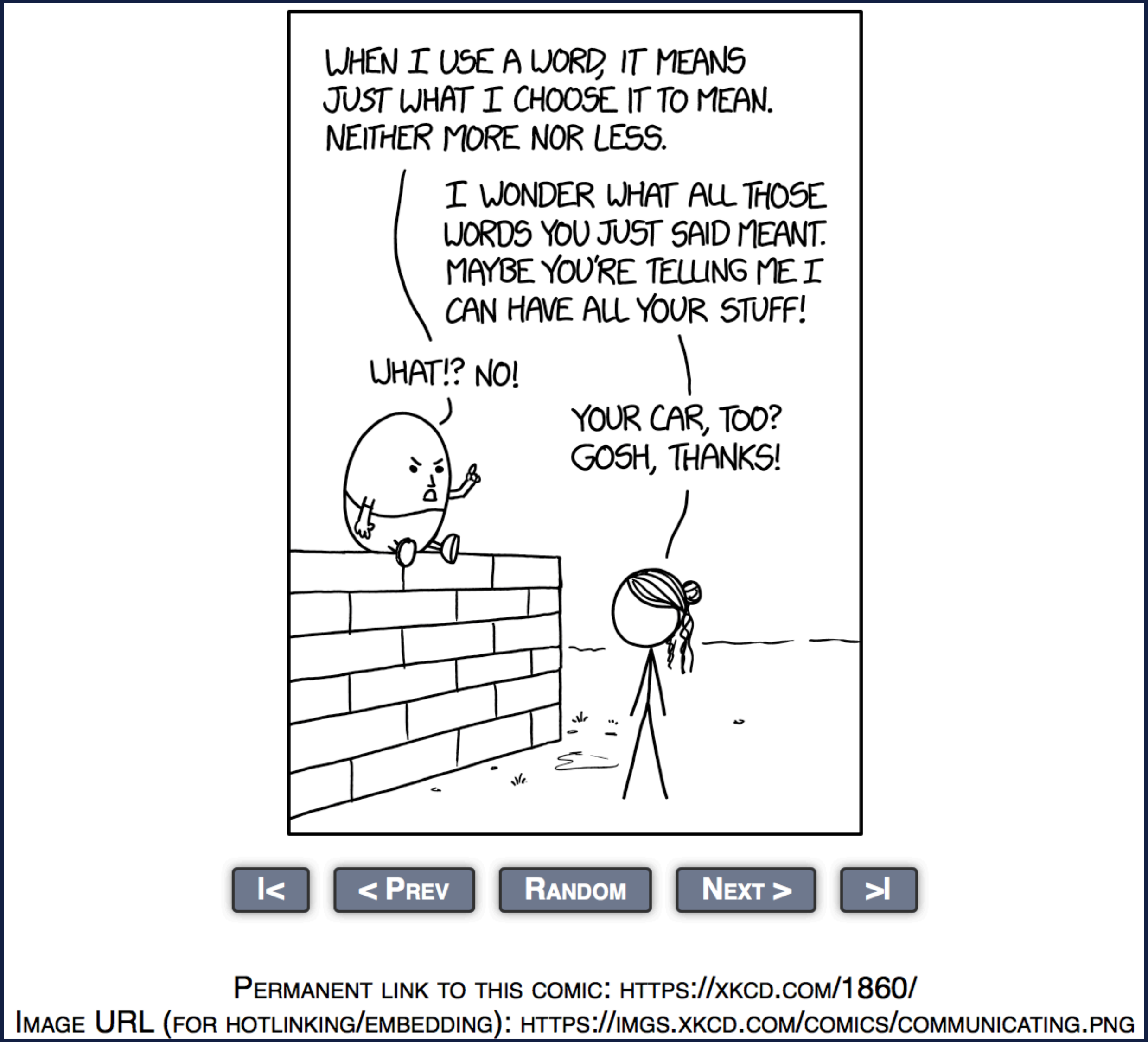
1.1 The meaning of a word, phrase, or text.

Taken from: https://en.oxforddictionaries.com/definition/semantics

Language

- Problem
  - **Poor use of language can lead to confusion and misunderstanding**

Language


WHEN I USE A WORD, IT MEANS JUST WHAT I CHOOSE IT TO MEAN. NEITHER MORE NOR LESS.
I WONDER WHAT ALL THOSE WORDS YOU JUST SAID MEANT. MAYBE YOU'RE TELLING ME I CAN HAVE ALL YOUR STUFF!
WHAT!? NO!
YOUR CAR, TOO? GOSH, THANKS!
|<
< PREV
RANDOM
NEXT >
>|

PERMANENT LINK TO THIS COMIC: HTTPS://XKCD.COM/1860/
IMAGE URL (FOR HOTLINKING/EMBEDDING): HTTPS://IMGS.XKCD.COM/COMICS/COMMUNICATING.PNG

Language

- ➢ Problem
  - Poor use of language can lead to confusion and misunderstanding
- ➢ However
  - There is a time and place to point out sloppy use and errors in language used
- ➢ AND
  - It must be done in a thoughtful, polite and careful manner

Words that drive me nuts

- Quality Assurance (or SQA)
  - **We, as testers, cannot assure quality**
  - **The only thing I can assure is that there are bugs in the software that I have not yet found**
  - **Do we call developers “Bug free coders”?**
  - **I prefer “Tester” over “QA”**

Words that drive me nuts

- Best Practice
  - **If something is the “best” it cannot be improved upon**
  - **Using “Best Practice” will inhibit process improvement and limit innovation**
  - **What works well on one project might be ineffective on another project**
  - **I prefer “Good Practice in this context”**

Words that drive others nuts

- Automated Testing (when referring to automated checking)
- Manual Testing (as opposed to Testing)
- Testing vs. Checking
- Testing Phase (testing should occur throughout the project)
- Ad hoc Testing (Ad hoc means “to this” as in “for a purpose” and not “random” or “chaotic”)

Hot Topic List

Language

Bug Reports

Bad Bug Reports

- Time is wasted attempting to investigate, ask/answer questions
- Bad bug reports can be sent back and forth a number of times before being resolved
- Bad bug reports lower the image of the tester and the entire testing team

Good Bug Reports

“ I write my bug reports as if they were going to be read by a mind-wiped version of myself ” – Hilary Weaver-Robb, Tester at Quicken Loans

PEOPLE WORKing

- Components of a good bug report:
  - Problem – Clearly and concisely describe the issue
  - Example – Recreation steps like you’re new to team
  - Oracle – Why is this a bug?
  - Polite – Avoid emotion and blame
  - Literate – Tell a story. Proper spelling and grammar
  - Extrapolate – What else might be related?
  - WORKaround – Include one, if applicable

Hot Topic List

Language

Bug Reports

Test Escapes

Test Escapes

- Do not (necessarily) overreact to a bug found by your customers
- Creating a check or a test for EVERY bug found by customers is likely a waste of time and effort
- Like a zoo keeper that keeps checking the gates that already let the animals escape instead of looking for other flaws in the cages that still have animals

Test Escapes

- It may be worthwhile to create checks for escapes that are:
  - Critical path
  - Recurring issue
  - Potential revenue impacting
  - Otherwise important enough

Hot Topic List

Language

Metrics

Bug Reports

Test Escapes

Definitions of *METRIC*

(from http://www.merriam-webster.com, April 2012)

- 1 *plural* **:** a part of prosody that deals with metrical structure
- 2 **: a standard of measurement** <no *metric* exists that can be applied directly to happiness — *Scientific Monthly*>
- 3 **:** a mathematical function that associates a real nonnegative number analogous to distance with each pair of elements in a set such that the number is zero only if the two elements are identical, the number is the same regardless of the order in which the two elements are taken, and the number associated with one pair of elements plus that associated with one member of the pair and a third element is equal to or greater than the number associated with the other member of the pair and the third element

Goodhart’s Law

- In 1975, Charles Goodhart, a former advisor to the Bank of England and Emeritus Professor at the London School of Economics stated:

Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes

Goodhart, C.A.E. (1975a) ‘Problems of Monetary Management: The UK Experience’ in *Papers in Monetary Economics*, Volume I, Reserve Bank of Australia, 1975

Goodhart’s Law

- Professor Marilyn Strathern FBA has re-stated Goodhart's Law more succinctly and more generally:

`When a measure becomes a target, it ceases to be a good measure.'

Elements of Bad Metrics

1. Measure and/or compare elements that are inconsistent in size or composition
   - Impossible to effectively use for comparison
   - How many containers do you need for your possessions?
   - Test Cases and Test Steps
     - Greatly vary in time required and complexity
   - Bugs
     - Can be different severity, likelihood - i.e.: risk

Elements of Bad Metrics

2. Create competition between individuals and/or teams
    - They typically do not result in friendly competition
    - Inhibits sharing of information and teamwork
    - Especially damaging if compensation is impacted

    - Number of xxxx per tester
    - Number of xxxx per feature

Elements of Bad Metrics

3. Easy to "game" or circumvent the desired intention
   - Easy to be improved by undesirable behaviour
   - Pass rate (percentage): Execute more simple tests that will pass or break up a long test case into many smaller ones
   - Number of bugs raised: Raising two similar bug reports instead of combining them

Elements of Bad Metrics

4. Contain misleading information or gives a false sense of completeness
   - Summarizing a large amount of information into one or two numbers out of context
   - Coverage (Code, Path)
     - Misleading information based on touching a path in the code once
   - Pass rate and number of test cases

Impact of Using Bad Metrics

- Promotes bad behaviour:
  - Testers may create more smaller test cases instead of creating test cases that make sense
  - Execution of ineffective testing to meet requirements
  - Artificially creating higher numbers instead of doing what makes sense
  - Creation of tools that will mask inefficiencies (e.g.: lab equipment usage)
  - Time wasted improving the "numbers" instead of improving the testing

Hot Topic List

Language

Metrics

Bug Reports

Test Reports

Test Escapes

Bad Test Reports

- Give a (potentially dangerous) summary of the situation with a strong focus on metrics
- Gives Executives a false sense of test coverage
  - All they see is numbers out of context
  - The larger the numbers the better the testing
  - The difficulty of good testing is hidden by large “fake” numbers
- Dangerous message to Executives
  - Our pass rate is at 96% so our product is in good shape
  - Code coverage is at 100% - our code is completely tested
  - Feature specification coverage is at 100% - Ship it!!!
- What could possibly go wrong?

Test Reports

- A good test report should tell a complete story about the product/feature that may include:
  - Executive Summary: Status of the product – in words not numbers
  - History of Important Events [Include scope changes]
  - Testing coverage (especially what wasn't tested)

    Consider using a Release Coverage Outline mind map
  - Testing effort expended vs. planned vs. actual
  - Risks and risk impacts
  - List of questions and assumptions
  - List of "Show stopping" and important bugs - no counts
  - Other bugs – if warranted to be in report
  - Plus tell a ***compelling*** story about your testing…

To test is to compose, edit, narrate, and justify THREE stories.

A story about the status of the PRODUCT…

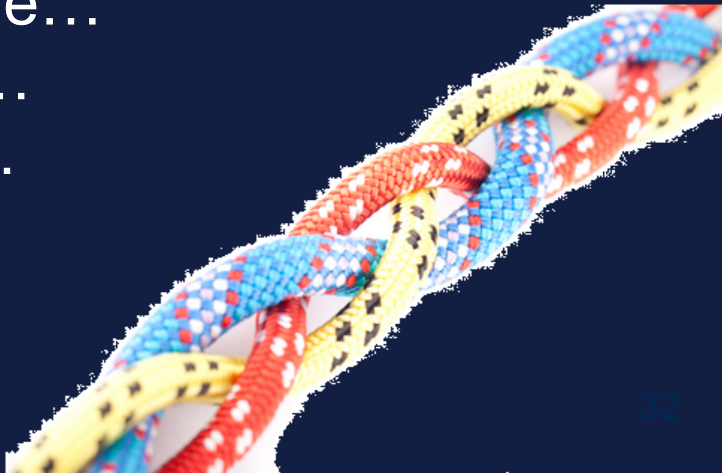
…about how it failed, and how it *might* fail...
…in ways that matter to your various clients

A story about HOW YOU TESTED it…

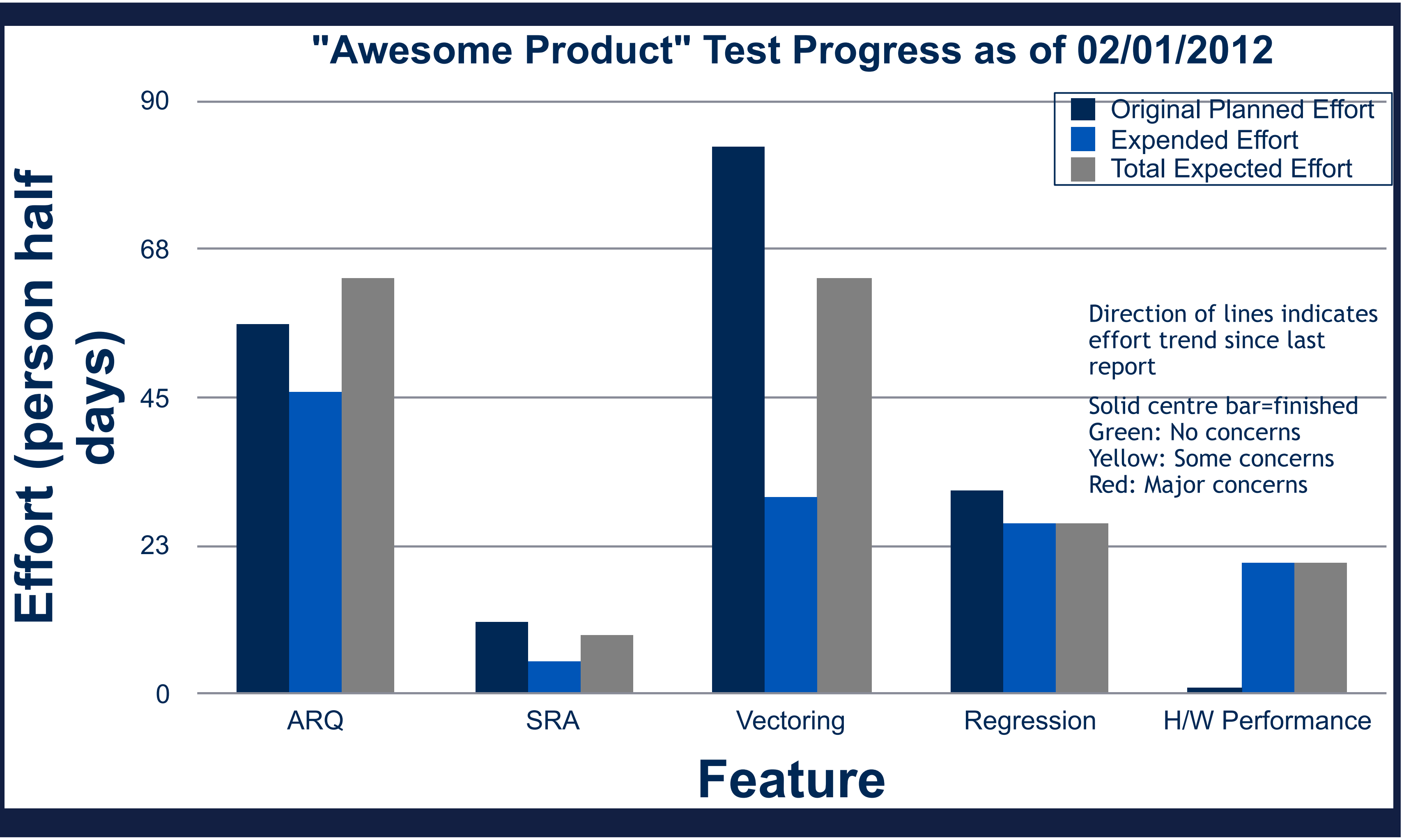
…how you configured, operated and observed it…
…about what you haven’t tested, yet…
…and won’t test, at all, unless our client requests otherwise…

A story about how GOOD that testing was…

…what the risks and costs of testing are…
…what made testing harder or slower…
…how testable (or not) the product is…
…what you need and what
you recommend



Sample Effort Chart

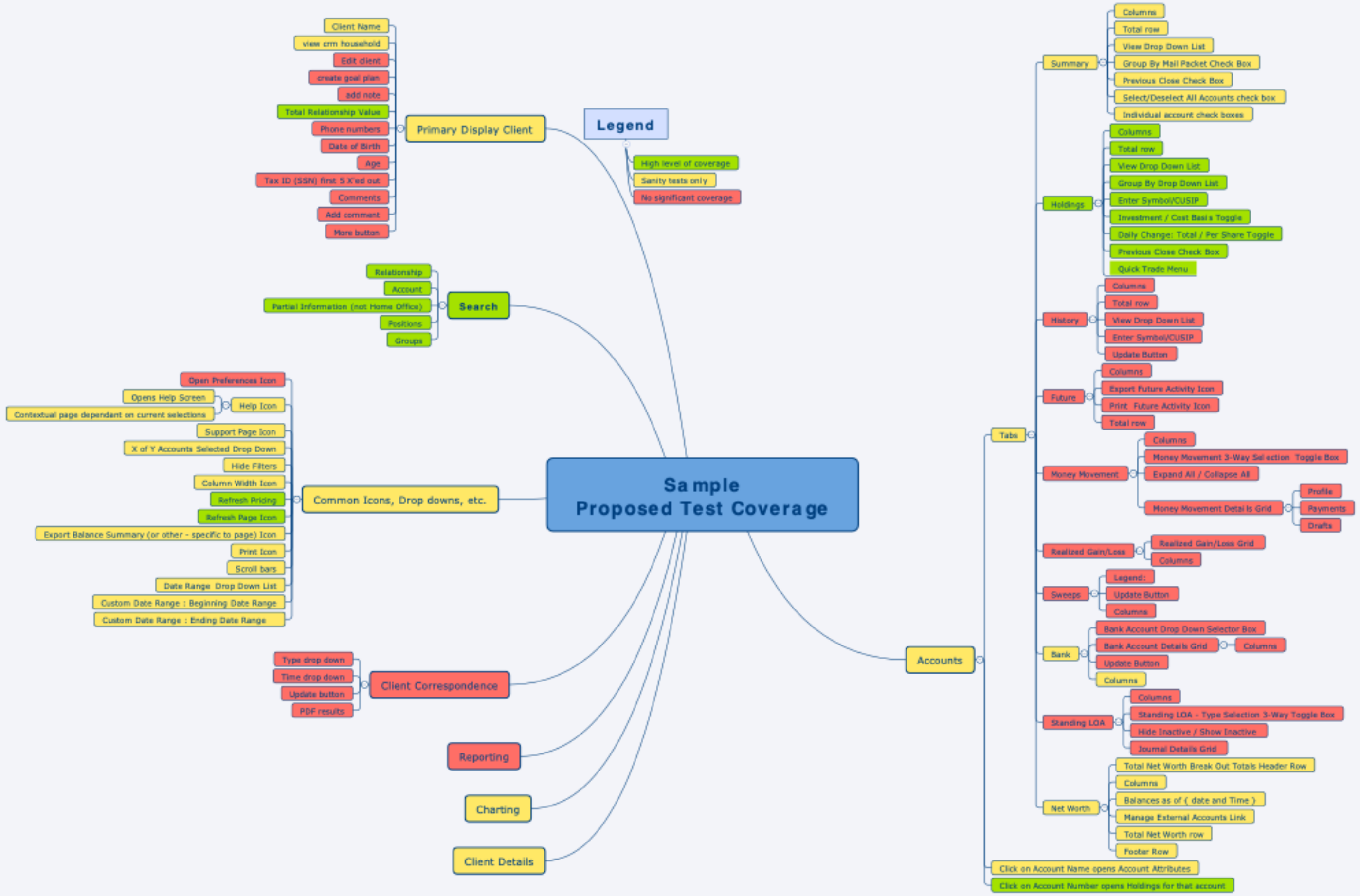


"Awesome Product" Test Progress as of 02/01/2012

Direction of lines indicates effort trend since last report

Solid centre bar=finished
Green: No concerns
Yellow: Some concerns
Red: Major concerns

Sample PCO – Coverage



Sample
Proposed Test Coverage

Legend
High level of coverage
Sanity tests only
No significant coverage

Primary Display Client
Client Name
view crm household
Edit client
create goal plan
add note
Total Relationship Value
Phone numbers
Date of Birth
Age
Tax ID (SSN) first 5 X'ed out
Comments
Add comment
More button

Search
Relationship
Account
Partial Information (not Home Office)
Positions
Groups

Common Icons, Drop downs, etc.
Open Preferences Icon
Help Icon
Opens Help Screen
Contextual page dependant on current selections
Support Page Icon
X of Y Accounts Selected Drop Down
Hide Filters
Column Width Icon
Refresh Pricing
Refresh Page Icon
Export Balance Summary (or other - specific to page) Icon
Print Icon
Scroll bars
Date Range Drop Down List
Custom Date Range : Beginning Date Range
Custom Date Range : Ending Date Range

Client Correspondence
Type drop down
Time drop down
Update button
PDF results

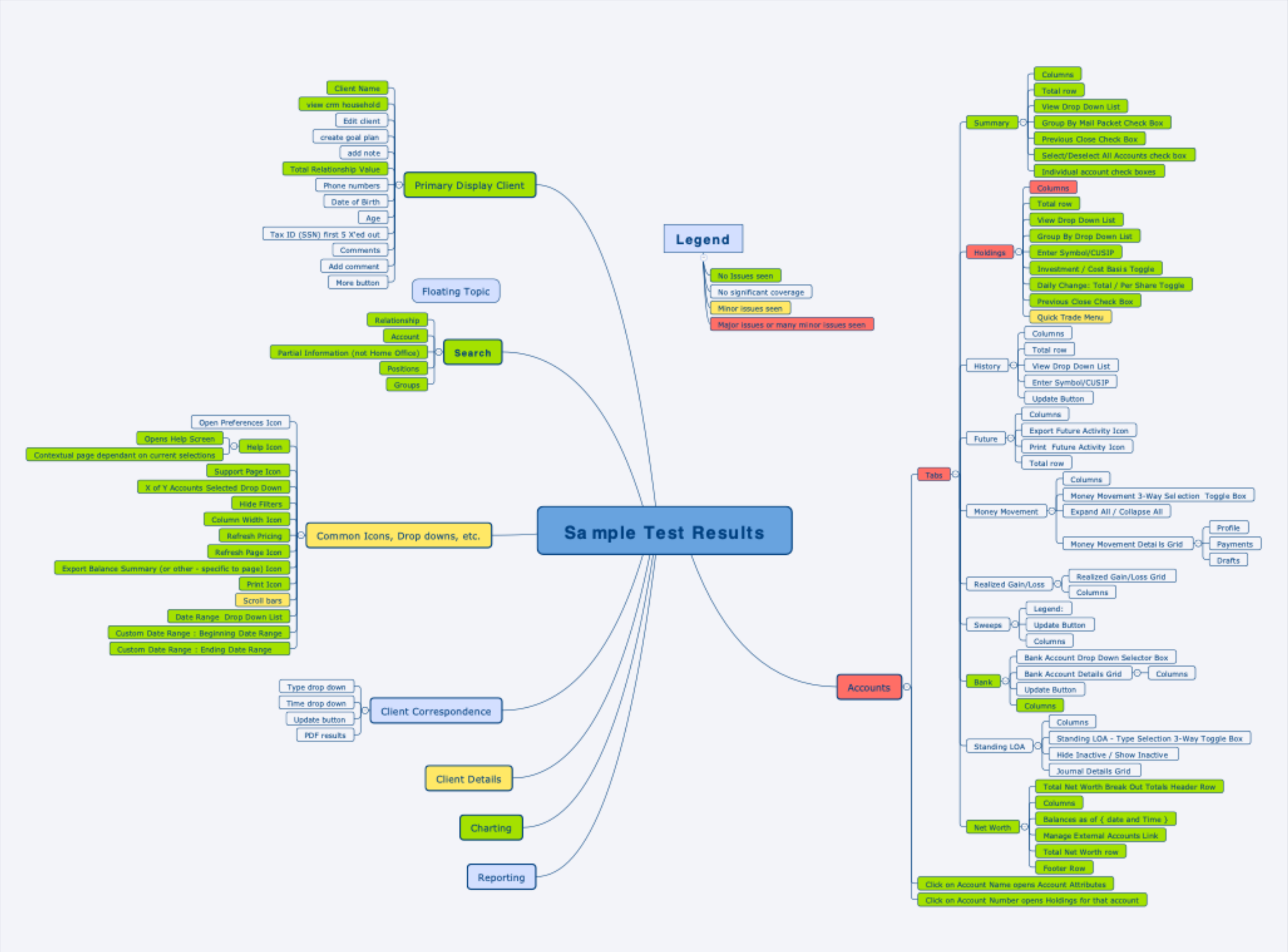
Reporting

Charting

Client Details

Accounts
Tabs
Summary
Columns
Total row
View Drop Down List
Group By Mail Packet Check Box
Previous Close Check Box
Select/Deselect All Accounts check box
Individual account check boxes
Holdings
Columns
Total row
View Drop Down List
Group By Drop Down List
Enter Symbol/CUSIP
Investment / Cost Basis Toggle
Daily Change: Total / Per Share Toggle
Previous Close Check Box
Quick Trade Menu
History
Columns
Total row
View Drop Down List
Enter Symbol/CUSIP
Update Button
Future
Columns
Export Future Activity Icon
Print Future Activity Icon
Total row
Money Movement
Columns
Money Movement 3-Way Selection Toggle Box
Expand All / Collapse All
Money Movement Details Grid
Profile
Payments
Drafts
Realized Gain/Loss
Realized Gain/Loss Grid
Columns
Sweeps
Legend:
Update Button
Columns
Bank
Bank Account Drop Down Selector Box
Bank Account Details Grid
Columns
Update Button
Columns
Standing LOA
Columns
Standing LOA - Type Selection 3-Way Toggle Box
Hide Inactive / Show Inactive
Journal Details Grid
Net Worth
Total Net Worth Break Out Totals Header Row
Columns
Balances as of (date and Time)
Manage External Accounts Link
Total Net Worth row
Footer Row
Click on Account Name opens Account Attributes
Click on Account Number opens Holdings for that account

Sample PCO – Results



Hot Topic List

Language

Metrics

Bug Reports

Test Reports

Test Escapes

UI Automation

UI Automation …

➢ UI Automation

- Often used too much
- Created by non-developers

- When you take a tester away from testing and have them write automation you likely lose a good tester and gain a bad developer

UI Automation …

- Automation will (likely) NOT
  - Increase your coverage
  - Decrease your costs
  - Save you time
  - Allow you to reduce headcount

Unless you also increase Risk

- Automation CAN
  - Give you a decent sanity check of your product
  - Execute in less time than a human performing the same checks

An Experience Report

Nortel Networks (when it was still a thing)

- System Test team had raised 1260 bugs over one year – *attributed to scripts*
- Analysis of whether they were found by the script or by a vigilant tester
- Findings:
  - 70% of the bugs were found by a vigilant tester
  - i.e.: They probably would NOT have been found by automation

Another Experience Report

A Large Insurance Company

- ➢ A decision was made to write an automated test for every requirement
- ➢ Why?
  - To catch more bugs?
  - To increase coverage?
  - To future proof the product?
  - To show thoroughness?
  - Reduce headcount of “manual” testers?
  - To make management feel better?
- ➢ Probably ALL of these reasons (initially)

Another Experience Report (cont)

A Large Insurance Company

➢ When I analyzed their testing situation:

- They had an EXCELLENT automation framework (modular, dereferenced calls, etc)
- They had over 2500 scripts
- Testing all written requirements
- Not ALL requirements, however, just the written ones
- Any bug that did not contradict a requirement was not a bug – until a requirement was created – then the bug could be entered

Another Experience Report (cont)

A Large Insurance Company

- When I analyzed their testing situation:
  - 13 outsourced testers in India took 2 weeks to execute them and investigate failures
  - About 40% of the scripts failed on each execution
  - They would get re-executed and if they passed then “no problem”
  - If it still failed they had to find out why and either update the script or raise a bug
  - On average this team found 6 defects a month
  - I pair tested the product for 20 min and found 5 bugs

Problem 1

There are more kinds of problems than automation can be programmed to recognize

Vigilant testers can observe and evaluate a very large variety of outputs and also vary the inputs to test new code paths – story about web page rendering

Problem 2

There are more checks to automate than can possibly be written

Problem 3

Some things are too difficult to automate effectively

Complex pass/fail algorithms

Perhaps could be done quickly by humans

Problem 4

Investigating reported failures takes a long time

UI level automation tends to be fragile and often breaks when code changes

If investigation does not happen then why run the tests?

Problem 5

Automation is expensive to build and maintain

High cost to value ratio

Sunk cost syndrome

Better Solution

A Strategic Mixed Approach:

- ➢ Automate critical paths
- ➢ Automate paths with the highest use
- ➢ Do NOT write automation for all failures found in the field
- ➢ Consider the cost of automation vs. benefit
  - Difficulty to create
  - Difficulty to maintain (frequency of changes)
  - Difficulty to analyze failures

Better Solution

- Augment automation by performing “testing” (by actual humans)
- Automation is excellent at showing that the code ***CAN*** work
- It takes a human to show how it might not work and how it might fail in important ways
- As Angie Jones (awesome automation developer and conference speaker) has said: She is an automation developer and she will not be able to automate away the job of a tester

THANK
YOU!