Requirements Based Functional Testing

A Practical Proven, and Repeatable Process For Testing

Presented By Bill Ufheil
Manager of Quality Assurance
CDW
One CDW Way
Vernon Hills, IL 60061

Email billufh@cdw.com

Requirements Based Functional Testing - Agenda

- Introduction
- Why Are You Here?
- The QA Requirements Based Functional Testing Process:
  - Initial Analysis and Test Planning
  - Detailed Requirements Analysis
  - Requirements Decomposition
  - Test Case Design and Construction
  - Test Execution
  - Post Implementation
- A Word About This Process
- Questions
- Sources Of Information
- In Closing

"Essentially, all models are wrong, but some are useful"

George E. P. Box

Introduction

- No Time?
- No Tools?
- No Staff?
- No Budget?
- No Knowledge?
- No Problem – I can help!
- This presentation will provide you with practical, proven, and repeatable techniques to quickly start up a QA testing process with no tools other than Microsoft Office
- No sales, no theories, no biases; just real world techniques that will allow you to successfully start software testing
- The intent of this presentation is to give you tools to make you think about all of the things that will be needed to govern your testing project
- A blend of QA and QC practices

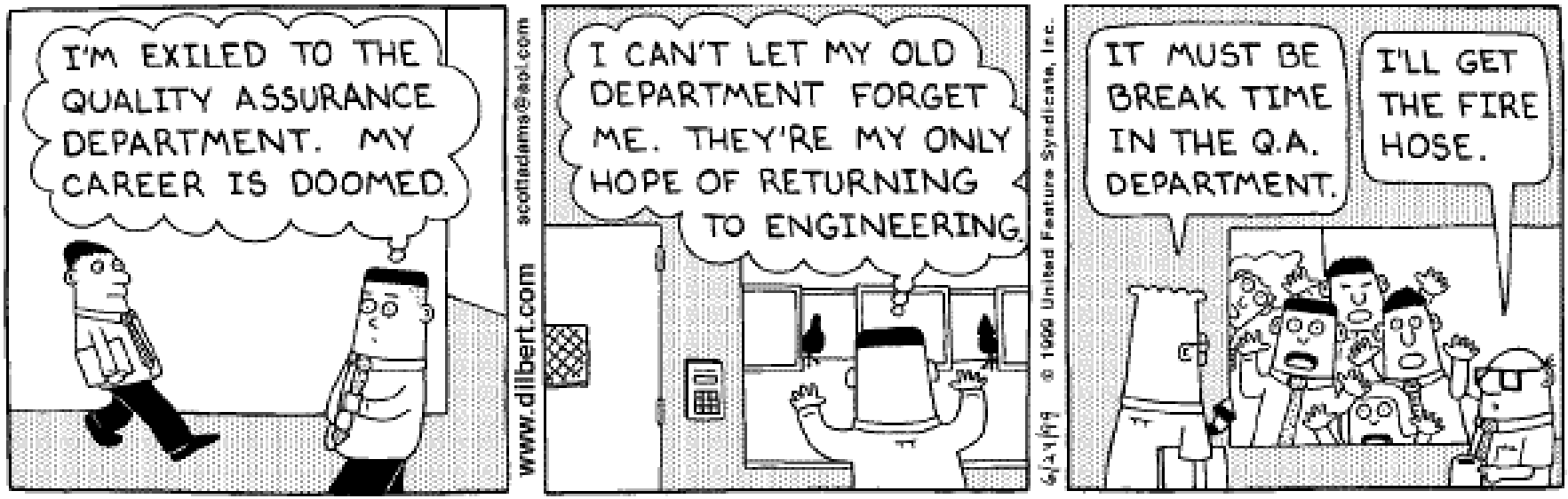
Why Are You Here?

- You may be in the wrong place if:
  - You are an experienced QA tester
  - You are an experienced QA test lead
  - You are an experienced QA manager
  - You are experienced in testing tools
  - You are looking to learn about testing tools
  - You are looking to learn about test automation
  - You thought you were going to hear about Managing Global Test Teams
- You're probably in the right place if:
  - You're new to QA
  - You're a new QA Lead
  - You're a new QA Manager
  - You've been tasked with a testing assignment
  - You have no knowledge of commercial tools

Why Are You Here?

- You're might be in the right place if:
  - You're looking to brush up on old techniques
  - You're looking to confirm what you already know
  - You're looking to confirm what you suspect to be true
  - You need to kill an hour and a half between presentations

Requirements Based Functional Testing



The QA Requirements Based Functional Testing Process

- Initial Analysis and Test Planning
- Detailed Requirements Analysis
- Requirements Decomposition
- Test Case Design and Construction
- Test Execution
- Post Implementation

QA Process Part 1 - Initial Analysis and Test Planning

- Requirements may be in many forms
  - Project Definitions
  - Business Requirements
  - Functional Requirement Specifications
  - Architectural Designs
  - Programmer Designs
  - Others
- You may/will need them all – or as many as you can get

Initial Analysis and Test Planning – How Its Done

- Read the Requirements To:
  - Identify large, obvious errors (unlikely)
  - Educate yourself
  - Understand the project
  - Understand the scope of your efforts

- In this phase do not:
  - Get bogged down in details
  - Over analyze
  - Ask too many detailed questions (there will be time for this later)
  - Rewrite everything in the project definition

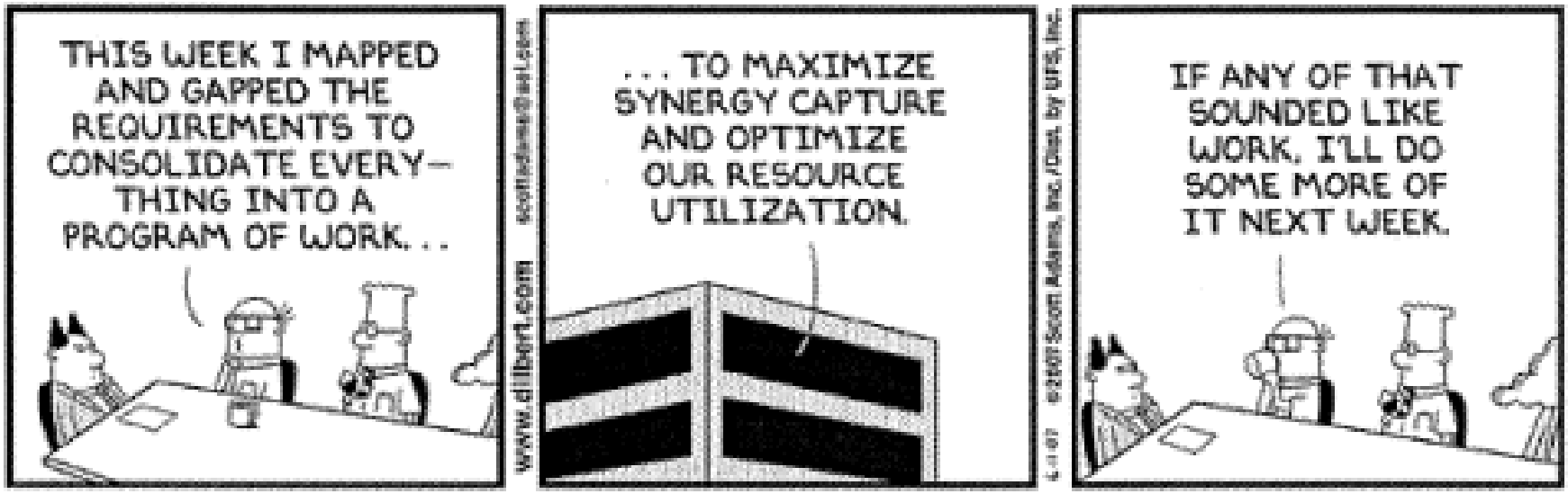
Initial Analysis and Test Planning – How Its Done

- When you have this all figured out
  - Create a Master Test Plan Document
    (use the table of contents or the major section headings and your current knowledge)
    - Scope – what is being tested and thus what is not being tested
    - Approach/Strategy – how it will be tested
    - Risks and Mitigation – what are the key risks, should they appear how will they be resolved
    - Assumptions
    - Entry/Exit Criteria – what do you need to start, what will you produce when you are complete
    - Establish roles and responsibilities
    - Make an initial estimate
  - Walkthrough the master Test Plan Document with project stakeholders
  - Achieve Buy In
    - Doesn't need to be formal
    - It's going to change

Initial Analysis and Test Planning

- Depending on the size of the project this should only take a couple of days
- Why Do This?
  - Confirm QA understands the project
  - Confirm the scope of the project
  - Introduce stakeholders that aren't familiar with QA processes to what you are doing
  - And how you are doing it
  - Have a fallback position if the project should start to experience delays
- Requirement Document Example
  CQAA ATM Requirements.docx
- Master Test Plan Example
  CQAA ATM Master Test Plan.docx

Requirements Based Functional Testing



QA Process Part 2 - Detailed Requirements Analysis

- Detailed Requirements analysis is the first key QA function
- It is the foundation upon which everything else will be built
- The purpose of this step is to assure the requirements are testable
- If the requirements aren't testable then they're probably not code-able
- The Problems With Requirements
  - Can be ambiguous
  - Can be vague
  - Can be contradictory
  - Can be incomplete (missing requirements)
  - Can be Inaccurate
  - Usually document only half the state and this is usually the positive condition

Detailed Requirements Analysis – How Its Done

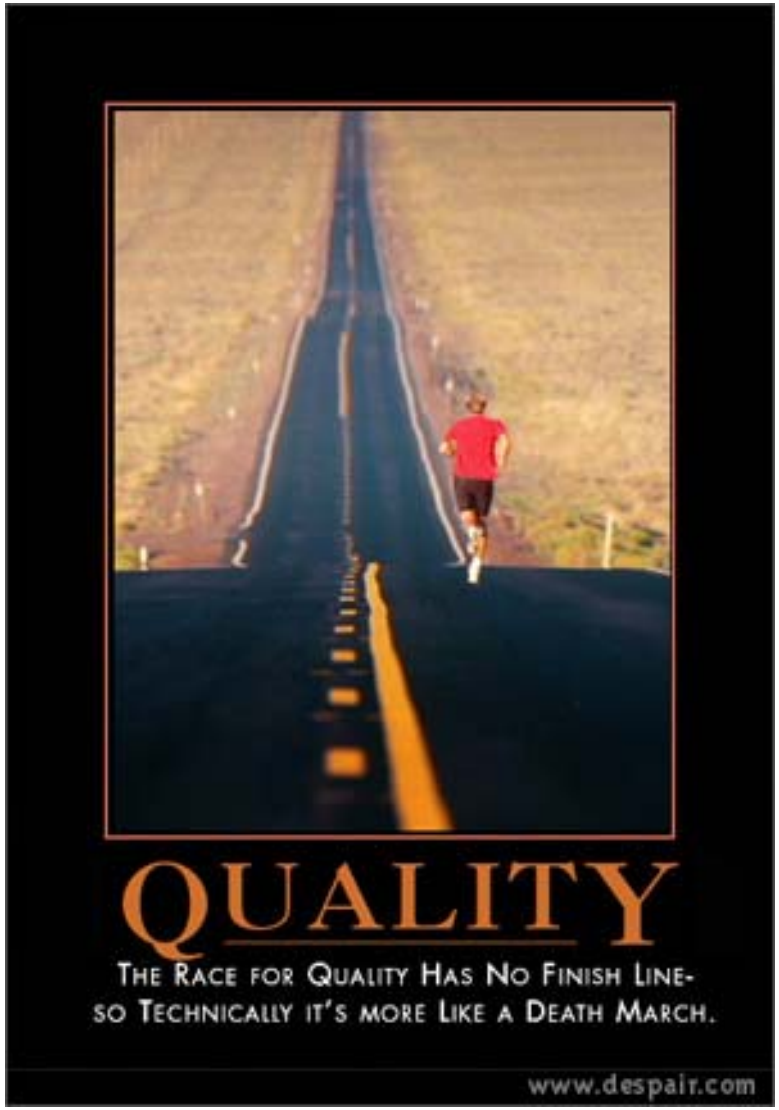
- To Begin
  - Read, Read, Read
  - Question, Question, Question
  - Until you have gained a complete understanding
- Resolve the typical problems, and thus un-testable parts of requirements
- Resolve Ambiguity
  - Look for works like: Abnormal, Accordingly, All, Almost, Also, Approximate, Common, Custom, Depending, Etc, Like, Normal, Similar, Standard, Typical, Usual, User Friendly
  - Note: Use the shortest version of the word, usual will find unusual, usually, etc.
- Resolve Vagueness
  - Look for words like would, should, could, etc
  - Look for adverbs (words that end in ly)
  - Look for Time and Frequency (daily, monthly, holiday, always, sometimes, occasionally)
  - Example: “The application should generate a summary report daily”

Detailed Requirements Analysis – How Its Done

- Resolve Contradictions
  - Do requirements in one section contradict those in another
- Resolve Incompleteness
  - Do sections connect smoothly
  - Be aware of
    - If statements
    - Do While Statements
    - Perform Until Statements
- Resolve Inaccuracies
  - What do you know about the application to be true that conflicts with what is written
- Look for positive or negative conditions
  - Requirements usually focus on positive conditions
  - Where one is stated the corresponding opposite condition exits

Detailed Requirements Analysis – How Its Done

- Look for Conjunctions
  - And, Or, Either, Neither
  - Where one is stated then the opposite condition exists
- Look for Mathematical Operators
  - EQ, NE, GT, LT, GTE, LTE
  - Where one is stated then the opposite condition exists
- Detailed Requirements Analysis is the first value add service that QA provides
- This is where QA begins to earn its money



QA Process Part 3 – Requirements Decomposition

- This is the second key QA function that differentiates QA from the rest of IT
- Decompose the requirements into all of their lowest-level testable conditions
- *Determine what is really meant by what is written*
- This is difficult because:
  - You don't know what you don't know
  - You need to be (or become) a Subject Matter Expert (SME)
  - Requirements documents are often poorly written
- What's Wrong With Requirements
  - Usually identify one version or state of data
  - Usually identify one class of data
  - You need to identify the missing states or classes
  - Usually identify the "positive" conditions
  - Can be vague, ambiguous, contradictory

Requirements Decomposition – How Its Done

- Base your analysis on the inputs (developers usually focus on results)
- This is the start of the "try to break it" (the QA process) vs. the "prove it works" (what development does) mentality
- Identify all of the Input Conditions
  - Look for
    - Equivalence Classes
    - Boundaries
    - Decision Tables
    - Others?
  - Look for Logical Connectors and Realtors
    - And, Or
    - If, Then, Else
  - Look for Binaries – True/False
  - Look for Relational Operators
    - Greater Than, Less Than, Equal, Greater Than or Equal, Less Than or Equal

Requirements Decomposition

- Basically – you need to:
  - "Challenge" everything you read
  - Learn to think in terms of opposites
  - Here's what its says – what else does or could it mean?
- Organize the Groups of Input Conditions
  - Place the input conditions in the chronological order in which they would appear to the user
  - Then place the input conditions in the hierarchical order in which they would relate
  - Usually any group of conditions is a mutually exclusive set – only one item can be true
- Identify all of the Output Conditions (if needed)

Requirements Decomposition Using Validate/Verify

- On a separate document list all of the application inputs
  - Validate Statements
    - Groups of related input conditions
    - There may be several hierarchical layers of Validate statements
  - Verify Statements
    - Lowest level, unique input conditions
    - Unwritten unique input conditions
      - » Unwritten???
      - » Yes - you will need to use your brain – this is where you translate what is written into what is meant!
- Think of this as high-medium-low requirements
  - When low level requirements are satisfied, usually in a test step, a testable condition is satisfied
  - When all medium level requirements are satisfied, usually in test cases, testable functions are satisfied
  - When all of the high level requirements are satisfied the project is complete

Requirements Decomposition Using Validate/Verify

- The goal is to attempt to identify all possible test conditions
  - You'll never get them all
  - But you will create a list of what you do know or have discovered
- Decomposition document should be in a numbered outline format
  - If the requirements document contained a numbering scheme – follow it
  - Else – invent your own

Requirements Decomposition Using Validate/Verify

- Validate/Verify Format
  - Validate/Verify
  - Subject
  - Action (optional – implied)
    - Is - Is Not
    - Can - Cannot
  - Examples:
    - Verify User can logon
    - Verify Total cannot exceed 100
    - Verify button enabled
    - Verify button not enabled

- Validate/Verify Method of Requirements Decomposition Example
  CQAA ATM Decomposition.docx

Requirements Decomposition Using Validate/Verify

- Why Use Validate/Verify
  - Condenses written requirements into manageable format
  - Groups related testable conditions together
  - Condenses written results into manageable format
- Why Not Use Validate/Verify
  - Can be too wordy
  - Not effective at showing relationships
  - Not effective at showing exclusionary relationships
- As you become proficient you may begin to skip this step and go straight to design


I CUT THE QUALITY CONTROL BUDGET TO FREE UP FUNDS TO INCREASE OUR SALES FORCE.
SO YOUR STRATEGY IS TO SELL A LARGER VOLUME OF DEFECTIVE PRODUCTS?
THE QUALITY WILL BE FINE. THE TESTS WILL JUST TAKE LONGER.
SO. . . IT'S AN ANTIQUE THING?
www.dilbert.com scottadams@aol.com
© 2008 Scott Adams, Inc./Dist. by UFS, Inc.

QA Process Part 4 – Test Case Design and Construction

- Requirements have been analyzed for testability and decomposed to identify all testable conditions – now what?
- Design Test Cases that are:
  - Unique – each test case exists for a unique and specific purpose
  - Efficient – requirements, or unique combinations are only tested once
  - Traceable – proving all requirements are covered
  - Clear – steps and results
- Construct Test Cases:
  - Setup
  - Data
  - Steps
  - Results
  - Shut Down

Test Case Design – How Its Done

- How to perform test case design
  - Ad Hoc - SME Best Guess
  - Test Case Design Matrix
  - All Pairs
  - Others?
- Ad Hoc - SME Best Guess
  - Pros
    - Quick
    - Reasonable if SME is truly E
  - Cons
    - Incomplete
    - Results in more of a checklist
    - Will probably cover 60% - 70%

Test Case Design – How Its Done

- The Test Case Design Matrix
  - An Excel spreadsheet
  - That seeks to:
    - Efficiently organize the testable conditions identified during requirements decomposition
    - Prove coverage
    - Guarantee combinations are covered
    - Provide traceability
- A repeatable, predictable, scalable process

Test Case Design – How Its Done

- The Matrix can be a one size fits all tool that is used for
  - Test Design
  - Requirements Traceability
  - Requirements Coverage
  - Prioritization
  - Test Execution
  - Test Metrics
  - Regression Analysis

Test Case Design – How Its Done

- But be careful
  - You can get carried away as this process will begin to force you to do all combinations
  - For example
    - –Requirement 1 = 5 testable conditions
    - –Requirement 2 = 8 testable conditions
    - –Requirement 3 = 2 testable conditions
    - –Requirement 4 = 2 testable conditions
    - –Requirement 5 = 3 testable conditions
    - –All possible combinations would = 480 test cases
  - You will need to know when to stop
    - –Don't repeat negative tests
    - –Don't repeat field level edits
    - –Don't repeat all for things like multi-browser testing

Test Case Design – How Its Done

- Let's look at an example
  - Sample Test Case Design Matrix
    Test Case Design Matrix - Basic.xlsx
  - An Excel Spreadsheet where:
    - Columns
      - Each is a test case
      - Select a single testable condition from a group
      - Combine groups for efficiencies
      - Columns with the same Result are "or" test scenarios
      - Testable Condition Inputs are "and" conditions
    - Rows
      - Provide traceability
      - Prove requirements coverage
      - Each testable condition in a group is mutually exclusive, select only one
      - Each testable condition is traced only once – thus tested only once

Test Case Design – How Its Done

- How To Create A Test Case Design Matrix
  - Set up the test conditions
    - Copy the requirements decomposition into the left hand columns
    - Groups should be in chronological order
    - Groups should be in hierarchical order
    - Place negative conditions before positive conditions
  - Pass One – Design Test Cases
    - Select the test conditions for the test case being defined
    - Do negative tests first
    - Matrix Nomenclature
      - » Traceable Test Conditions
      - » P = Positive Test Result – processing should continue
      - » N = Negative Test Result – an error should be generated and processing should stop
      - » Not Traceable Test Conditions – used for navigation
      - » X = Navigational – test condition has been traced elsewhere, in this usage it is required to get to the next testable condition

Test Case Design – How Its Done

- How to Create A Test Design Matrix – Continued
  - Pass 2 - Group negative test conditions into one test
  - Pass 3 – look for logical relationships and group them. For example does t make sense to test ID and Password fields together or separately
  - Pass 4 – Scale back the testing, you won't be able to execute all possible combinations
    - Examples
      - » Web Testing – do all tests need to run on all browsers?
      - » Access/Permissions testing – do all tests need to be repeated for each permissions class?
- Don't be discouraged – this will take several iterations or projects to master
- The initial matrix usually is rewritten several times as it is being built
- Example Test Case Design Matrix Based On Our ATM Project
  CQAA ATM Test Case Design.xlsx

Test Case Design – How Its Done

- Why Use A Design Matrix
  - Assures coverage
  - Provides traceability
  - Increases efficiency
  - Facilitates making choices
  - Creates a map of where you've been
  - And where your going
- Why Not Use A Design Matrix
  - Can be difficult to comprehend
  - Can become too large
  - Will be difficult to start or get buy-in

Test Case Design – Matrix Is More Than Just Design

- The Design Matrix is complete and we're ready to build test cases

- Or Are We?
  - We can add
    - Risk Assessment
    - Test Priority
    - Regression Tests Identification
    - Test Status
    - All of the Above

Test Case Design – Matrix Is More Than Just Design

- Risk Assessment – How Its Done
  - Establish the likelihood of a test occurring
  - Assign a value of 1 to 3
  - Establish the severity of a failure
  - Assign a value of 1 to 3
  - Multiply (or add) the values
  - Assign a Priority
- Allows you to make conscious decisions on what to test or what not to test
  - Better to omit a test on purpose
  - Than to not perform a test you should have it you knew about it
- Example of a Test Design Matrix with Risk Assessment
  CQAA ATM Test Case Design With Risk.xlsx

Test Case Design – More Than Just Design

- Establish Priority
  - Function of Risk
  - Function of Core Business Processes
  - Function of Complex Code
  - Function of Known Clustering
  - Function of Financial Impact
  - Function of Legal Impact
  - Function of the 80/20 rule

- Example of a Test Design Matrix with Priority
  CQAA ATM Test Case Design With Priority.xlsx

Test Case Design – More Than Just Design

- Identify Regression Tests
  - Function of Core Business Processes
  - Function of Financial Impact
  - Function of Legal Impact
  - Function of the 80/20 rule
- Regression tests are usually end-to-end business scenarios
- They will probably come from your higher priority test cases
- Example of a Test Design Matrix with Regression Identification
  CQAA ATM Test Case Design With Regression.xlsx

Test Case Design – More Than Just Design

- What if you put them all together?
- A complete design
  - With risk established
  - With priority set
  - With regression test identified
- Providing Key Metrics On
  - Coverage
  - Progress
  - Status
  - Defect Density
  - Quality Trends
- Example of a Test Design Matrix with Risk-Priority-Regression Identification
  CQAA ATM Test Case Design With Risk-Priority-Regression.xlsx

Test Case Design Using All Pairs

- What is All Pairs
  - Based on a mathematical concept called Orthogonal Arrays
  - All testable conditions are paired with all other conditions at least once
- Why Use All Pairs
  - Once inputs are established – test cases are quickly established
  - Should quickly find serious defects in core functionality
  - Should satisfy the 80/20 rule
  - OK - this is a tool, but its free!
- Why Not Use All Pairs
  - Not all possible combinations are accounted for
  - Better for a regression test on mature systems
  - You could miss 30% of the test cases (multiple combinations)

Test Case Design Using All Pairs

- More On Orthogonal Array's
- Assume the following
  - Test condition A has 3 values
  - Test condition B has 3 values
  - Test condition C has 3 values
  - Test condition D has 3 values
- The total possible number of test cases would be 3*3*3*3=81
- Applying an Orthogonal Array will reduce the number of test cases to 9
- Each value of A is paired with every value of B which is paired with every value of C which is paired with every value of D
- All possible A-B-C-D combinations are not paired (that would be 81)

Test Case Construction – How Its Done

- Identify Set Up Conditions
  - Equipment or Hardware
  - Software Version
  - Security/Access
  - Test Harness
  - Data backup
- Identify Test Data
  - What data do you need
  - To meet the conditions identified in design
- Document the Test Execution Steps
- And the Expect Results
- Identify Shut Down Conditions
  - Data reset
  - Test Harness Removal

QA Process Part 5 -Test Execution

- What If ...
  - You're on a fast track project
  - You don't have time to create test cases
- You can use the Design Matrix to perform execution
- Just add a status row
- Possible Status Values
  - P = Pass
  - F = Fail
  - B = Blocked
  - NR or blank = Not Run
- Example of a Test Design Matrix with Status
  CQAA ATM Test Case Design With Status.xlsx

Test Execution – Measuring Your Progress

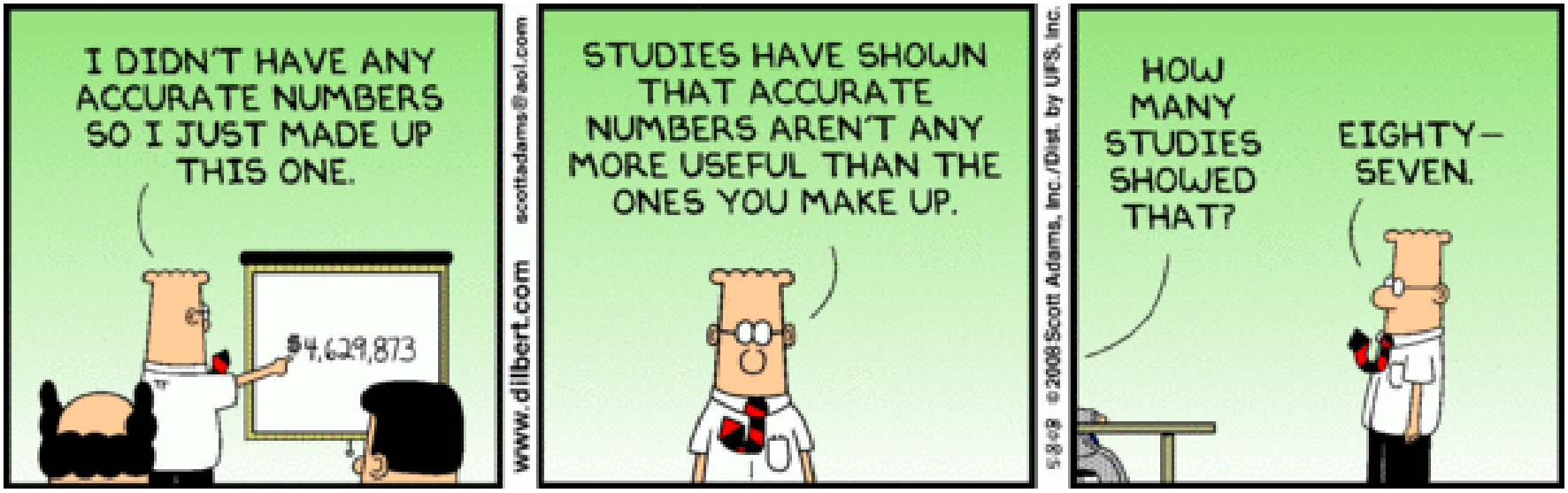
- What about testing progress – how do you know if you are on schedule?
- For Example:
  - You have 100 test cases
  - You have 5 days to do test execution
  - You have to average 20 test cases passing a day to be on schedule
- Let's Add A Run Rate To Our Test Design Matrix
- Example of a Test Design Matrix with Status
  CQAA ATM Test Case Design With Run Rate.xlsx

Test Execution – What If You Fall Behind

- Add more testing resources
- Move the production date
- Fall back to the priority decisions you made during design
- Decision Time
  - Breadth
    - Do a little bit of everything
    - Use the high priority test cases
  - Depth
    - Skip some functionality
    - Probe deeply into riskier or buggier functions
  - Combine the two
- This is why you did the risk analysis and prioritization during design

Test Execution – How Do You Know When You're Done

- Exit criteria are met
- The cost of continued testing is GT than the cost of the defects you will find
- Some Rules of Thumb
  - 90% of test cases have passed
  - All high priority test cases have passed
  - There are no Open Urgent–Show Stopper–High Priority Defects



QA Process Part 6 - Post Implementation

- Open Defects
  - Communicate to Help Desk
  - Log in the Production Issues Log
  - Communicate to 2'nd Level Support
- Lightly tested areas - same as above
- Transfer test cases to regression testing
- Review the QA Process

A Word About This Process

- The included documents are for example purpose only. They are not meant to be complete, all inclusive, or final. You will see flaws….
- The topic of this presentation is Requirements Based Functional Testing so:
  - We assume that requirements are reasonable
  - We will be doing black box functional testing
- If you're not careful, you can get carried away …
- It takes a couple of tries (usually at least 3) to get comfortable
- Start with a smaller scale project
- Learn the process first – add tools later
- Any process or tool is only as good as its inputs

Questions

Why are we doing this?
What problem are we solving?
Is this actually useful?
Are we adding value?
Will this change behavior?
Is there an easier way?
What's the opportunity cost?
Is it really worth it?

Sources of Information

- *Managing The Testing Process*

  Rex Black

  Microsoft Press
- *Systematic Software Testing*

  Rick D. Craig and Stefan P. Jaskiel

  Artech House Publishers
- All Pairs – James Bach
  http://www.satisfice.com/tools.shtml
- Sticky Minds
  http://www.stickyminds.com/

In Closing

- I would love to here from you.
- Is this process working for you?
- Would you like to discuss it further?
- Have you found ways to improve it?
- Contact Me
  - Bill Ufheil
  - billufh@cdw.com