Quality “Deep-Dive”
More Effective Testing Today

As presented at

CSC 2009 CONNECT SEIZE EVERY DAY
SEPTEMBER 13-17 • LAKE BUENA VISTA, FLORIDA

Peter Briggs Jr.
New Instruction, LLC
615 Valley Road
Upper Montclair, NJ 07043
(973) 744-3339
www.newinstruction.com
pbriggs@newinstruction.com

Quality “Deep-Dive” (973) 744-3339 © 2009 New Instruction, LLC

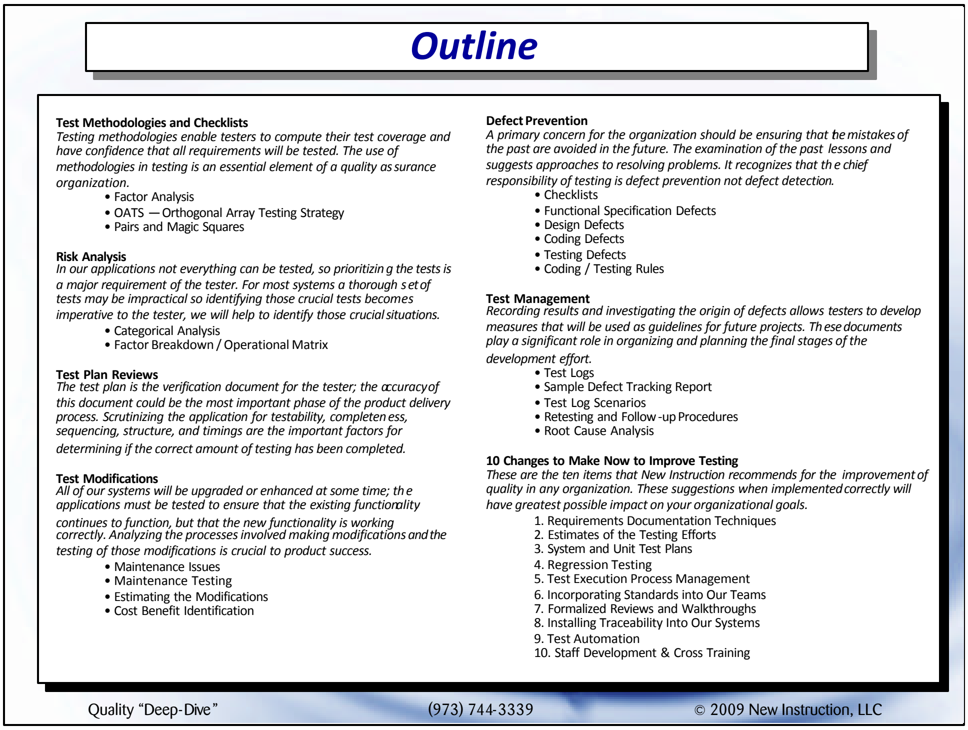
All slide notes are courtesy of …

Ginny A. Eiwen, PMP

CSC Exceed Quality Office Manager

CSC Financial Services Group

Ginny’s knowledge of this presentation was augmented through the previous exposure of bringing New Instruction’s 3-day workshop in-house for her team.



Outline

Test Methodologies and Checklists
Testing methodologies enable testers to compute their test coverage and have confidence that all requirements will be tested. The use of methodologies in testing is an essential element of a quality as surance organization.

- Factor Analysis
- OATS — Orthogonal Array Testing Strategy
- Pairs and Magic Squares

Risk Analysis
In our applications not everything can be tested, so prioritizin g the tests is a major requirement of the tester. For most systems a thorough s et of tests may be impractical so identifying those crucial tests becomes imperative to the tester, we will help to identify those crucial situations.

- Categorical Analysis
- Factor Breakdown / Operational Matrix

Test Plan Reviews
The test plan is the verification document for the tester; the accuracy of this document could be the most important phase of the product delivery process. Scrutinizing the application for testability, completen ess, sequencing, structure, and timings are the important factors for determining if the correct amount of testing has been completed.

Test Modifications
All of our systems will be upgraded or enhanced at some time; th e applications must be tested to ensure that the existing functionality continues to function, but that the new functionality is working correctly. Analyzing the processes involved making modifications and the testing of those modifications is crucial to product success.

- Maintenance Issues
- Maintenance Testing
- Estimating the Modifications
- Cost Benefit Identification

Defect Prevention
A primary concern for the organization should be ensuring that the mistakes of the past are avoided in the future. The examination of the past lessons and suggests approaches to resolving problems. It recognizes that th e chief responsibility of testing is defect prevention not defect detection.

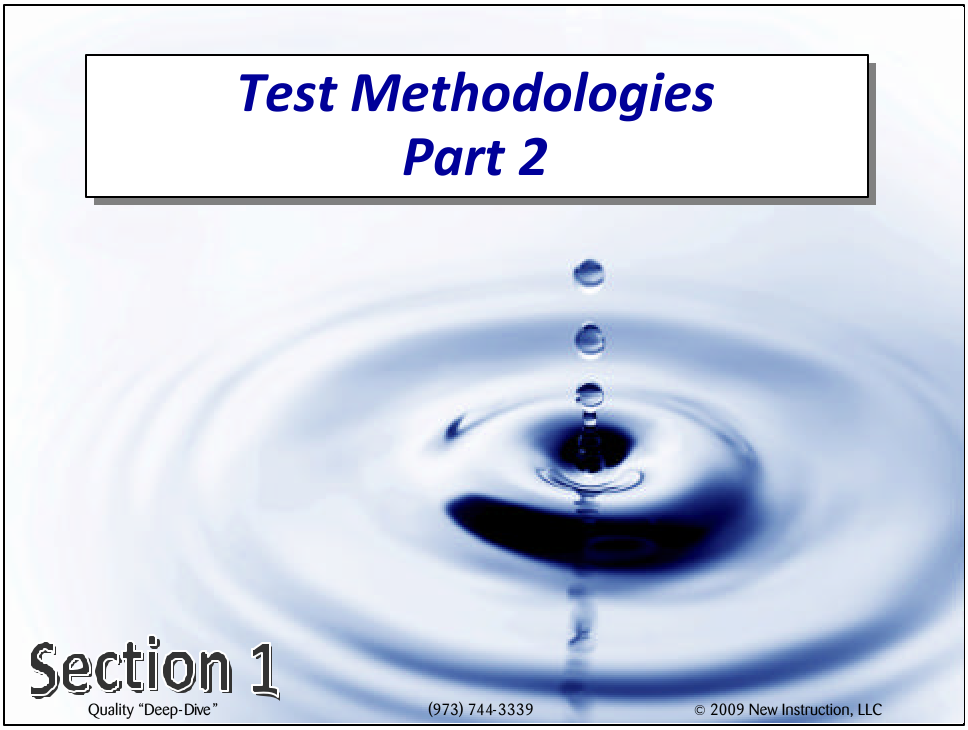
- Checklists
- Functional Specification Defects
- Design Defects
- Coding Defects
- Testing Defects
- Coding / Testing Rules

Test Management
Recording results and investigating the origin of defects allows testers to develop measures that will be used as guidelines for future projects. Th ese documents play a significant role in organizing and planning the final stages of the development effort.

- Test Logs
- Sample Defect Tracking Report
- Test Log Scenarios
- Retesting and Follow -up Procedures
- Root Cause Analysis

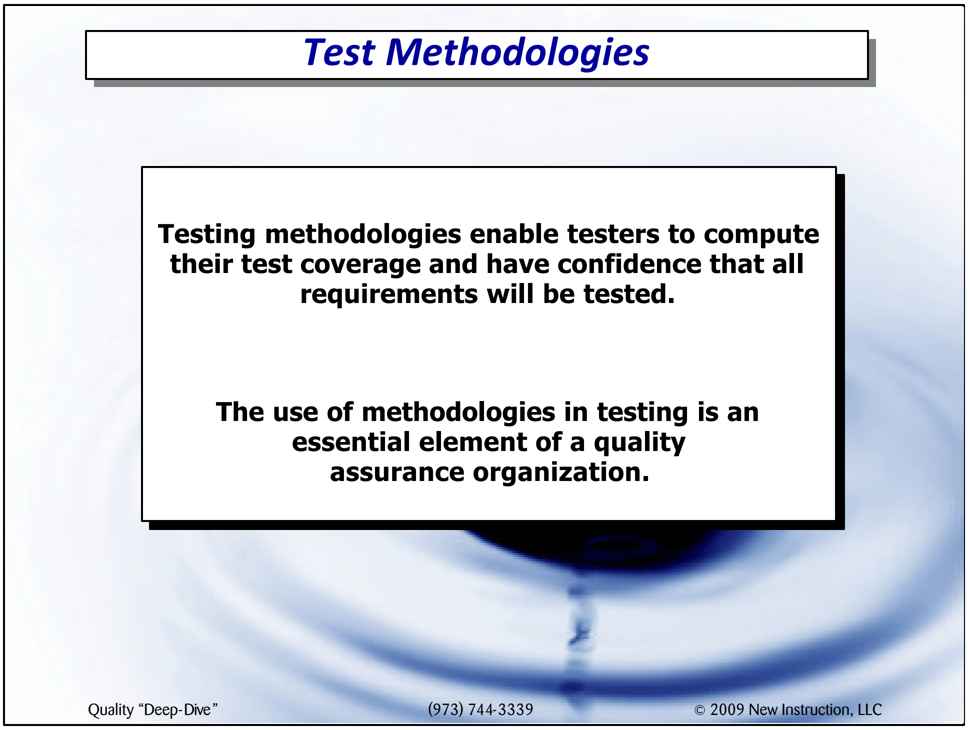
10 Changes to Make Now to Improve Testing
These are the ten items that New Instruction recommends for the improvement of quality in any organization. These suggestions when implemented correctly will have greatest possible impact on your organizational goals.

1. Requirements Documentation Techniques
2. Estimates of the Testing Efforts
3. System and Unit Test Plans
4. Regression Testing
5. Test Execution Process Management
6. Incorporating Standards into Our Teams
7. Formalized Reviews and Walkthroughs
8. Installing Traceability Into Our Systems
9. Test Automation
10. Staff Development & Cross Training



Test Methodologies
Part 2

Section 1



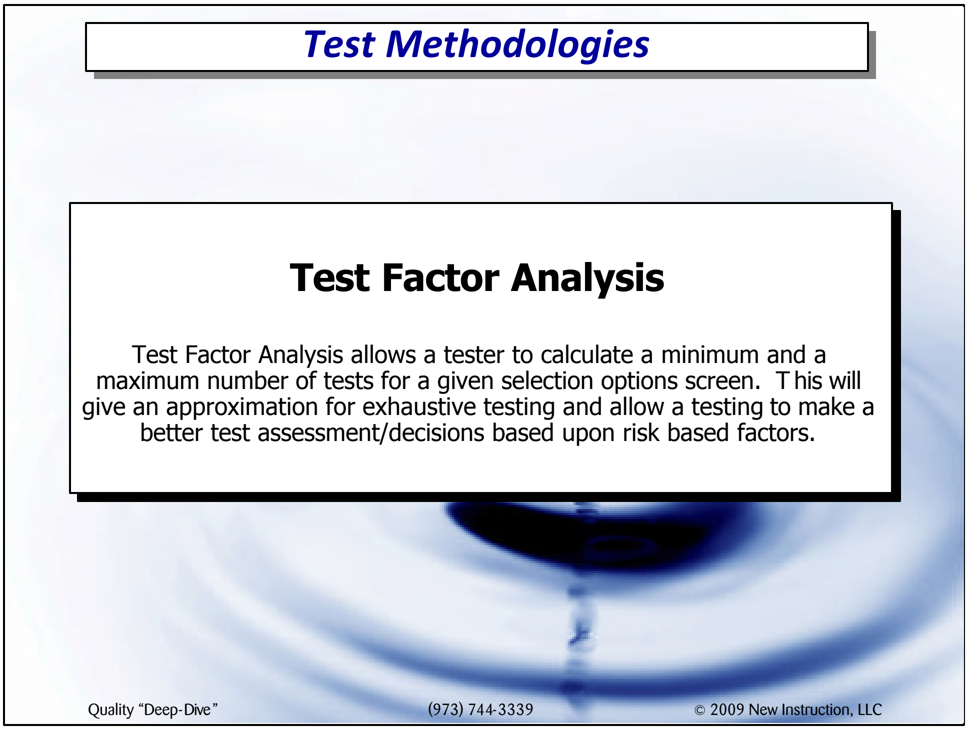
Test Methodologies

Testing methodologies enable testers to compute their test coverage and have confidence that all requirements will be tested.

The use of methodologies in testing is an essential element of a quality assurance organization.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

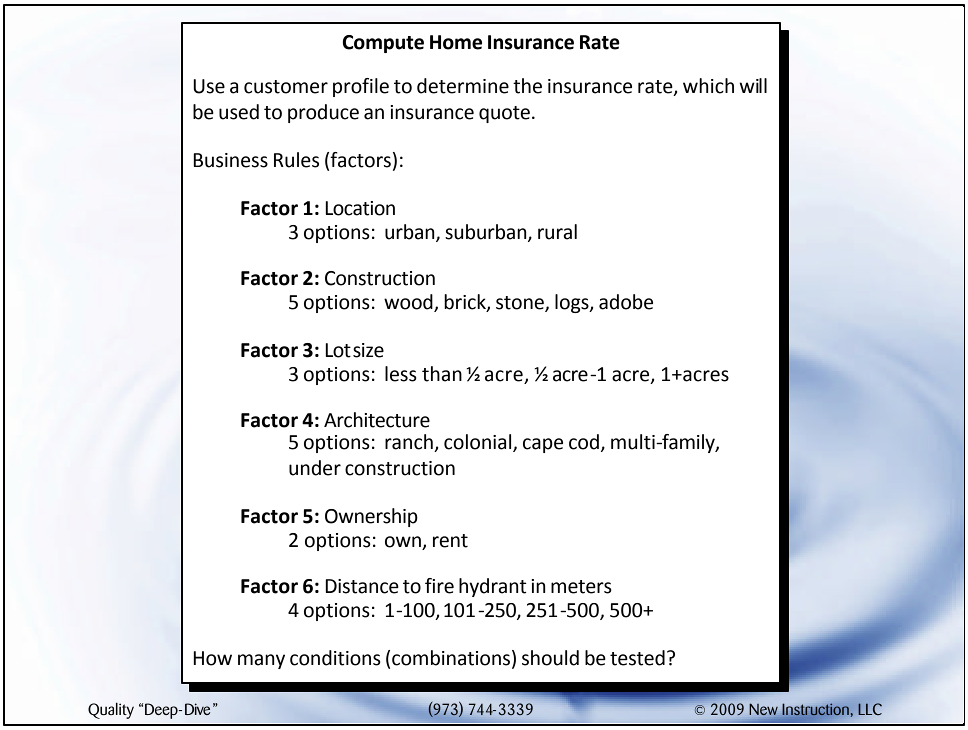
Ways of uncovering tests that might not be obvious from requirements.



Test Methodologies

Test Factor Analysis

Test Factor Analysis allows a tester to calculate a minimum and a maximum number of tests for a given selection options screen. This will give an approximation for exhaustive testing and allow a testing to make a better test assessment/decisions based upon risk based factors.



Compute Home Insurance Rate

Use a customer profile to determine the insurance rate, which will be used to produce an insurance quote.

Business Rules (factors):

Factor 1: Location
3 options: urban, suburban, rural

Factor 2: Construction
5 options: wood, brick, stone, logs, adobe

Factor 3: Lot size
3 options: less than ½ acre, ½ acre-1 acre, 1+acres

Factor 4: Architecture
5 options: ranch, colonial, cape cod, multi-family, under construction

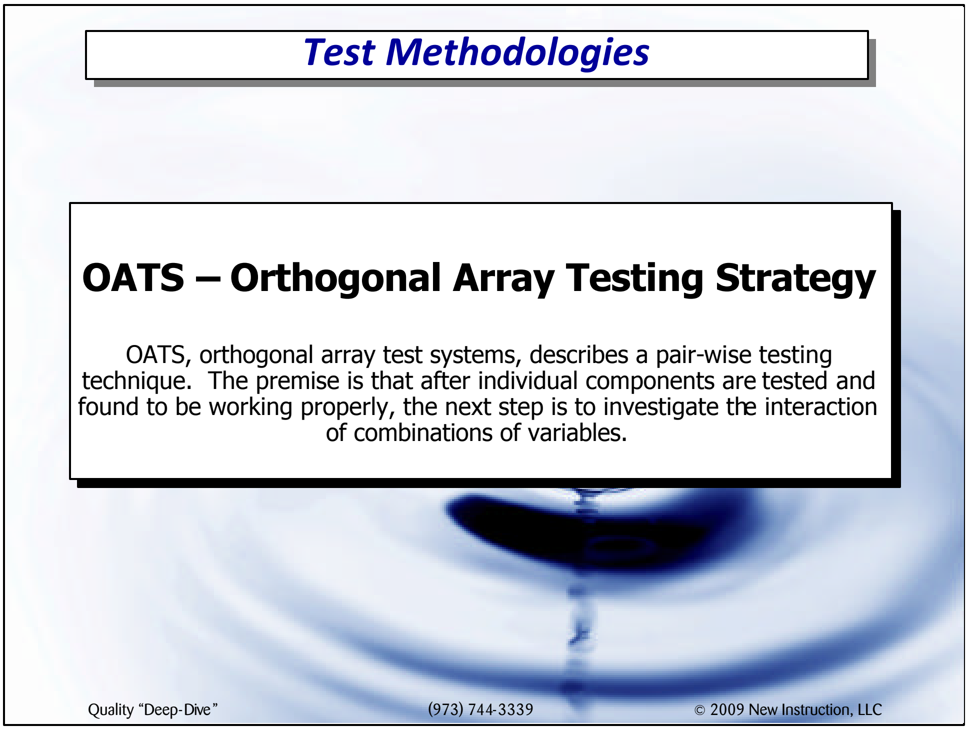
Factor 5: Ownership
2 options: own, rent

Factor 6: Distance to fire hydrant in meters
4 options: 1-100, 101-250, 251-500, 500+

How many conditions (combinations) should be tested?

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

For each test factor, take the number of test options, and add those to identify the minimum number of test cases required. Maximum number of tests determined by multiplying the number of options together – on Compute Home Insurance Rate slide, the range of test cases would be 22 to 1800 test cases.



Test Methodologies

OATS – Orthogonal Array Testing Strategy

OATS, orthogonal array test systems, describes a pair-wise testing technique. The premise is that after individual components are tested and found to be working properly, the next step is to investigate the interaction of combinations of variables.

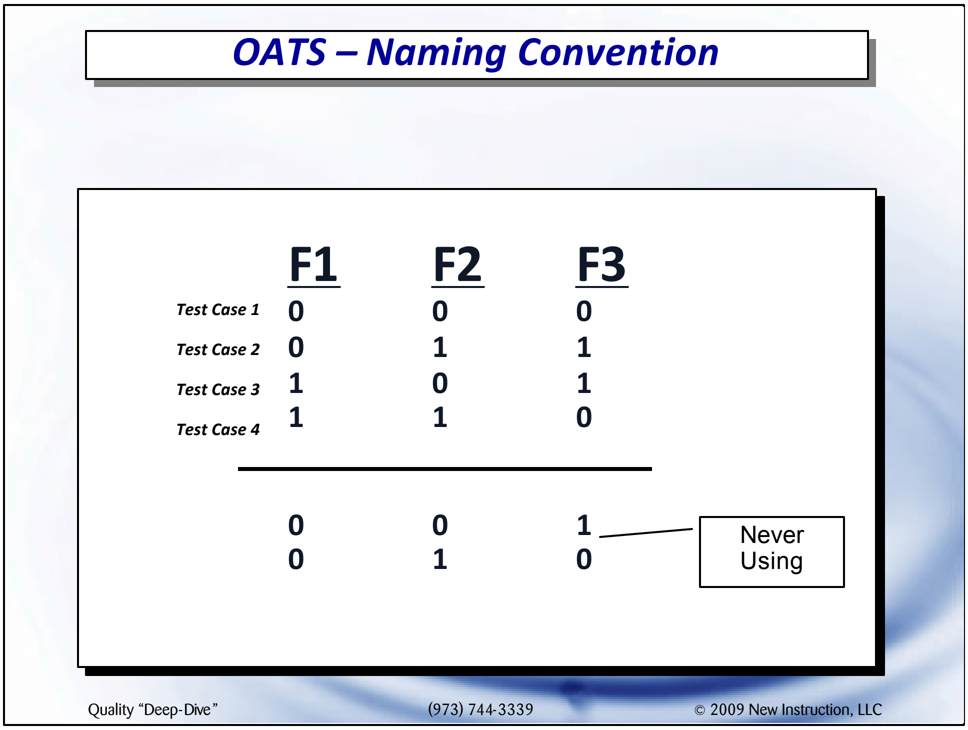
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

...aka 'Pairwise Testing'

Minimize the number of test cases required by looking at factors in combination.

For each possible factor it can be included / excluded in a test.

Assumes that as long as each pair of factors is tested in SOME paired combination, you can eliminate test cases that would contain redundant combinations of factors



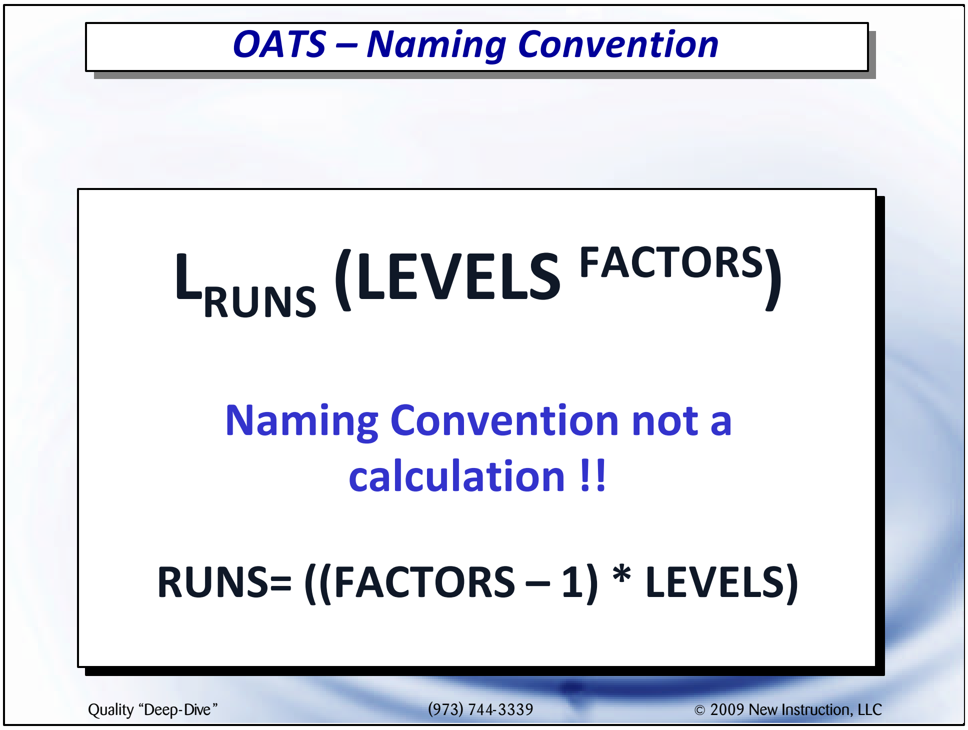
OATS – Naming Convention

	F1	F2	F3
Test Case 1	0	0	0
Test Case 2	0	1	1
Test Case 3	1	0	1
Test Case 4	1	1	0

0	0	1	Never Using
0	1	0	

Quality “Deep-Dive” (973) 744-3339 © 2009 New Instruction, LLC

OATS tables use a NAMING CONVENTION – don’t confuse NAMING CONVENTION with a calculation.



OATS – Naming Convention

$$L_{\text{RUNS}}\ (\text{LEVELS}^{\text{FACTORS}})$$

Naming Convention not a calculation !!

RUNS= ((FACTORS – 1) * LEVELS)

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

OATS (Orthogonal Array Testing Strategy):

L_{RUNS} (LEVELS $^{\text{FACTORS}}$)

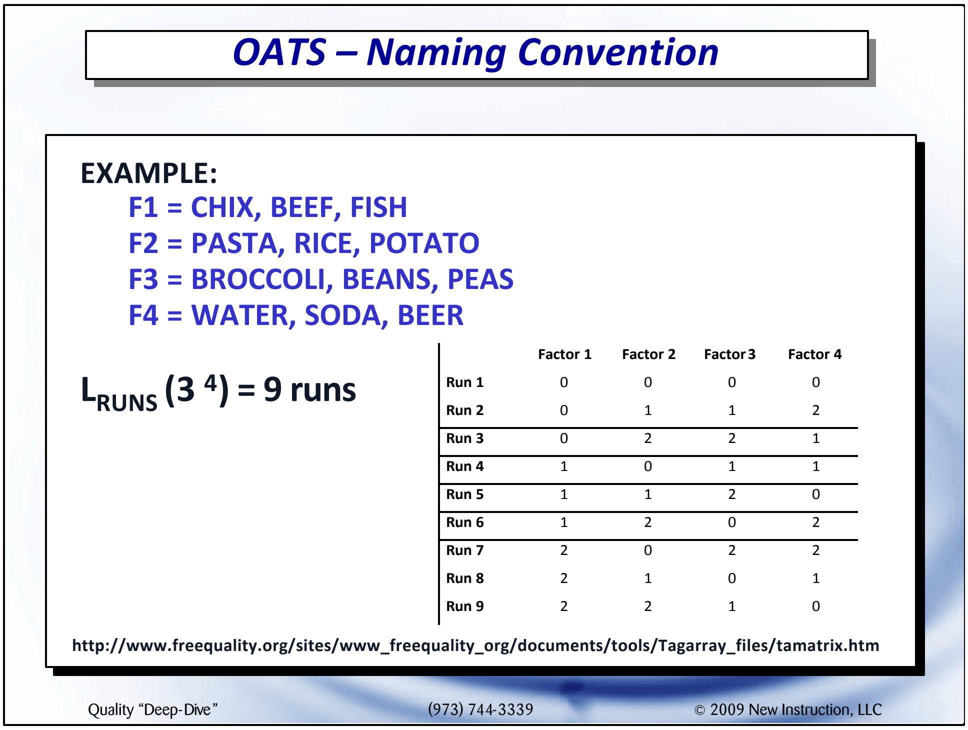
RUNS = ((FACTORS -1) * LEVELS)

Factors = the number of questions

Levels = the maximum number of selection options for any one factor

Runs = number of test cases required – determined by a table look-up

Once you have naming convention, you can look up the actual table



OATS – Naming Convention

EXAMPLE:

F1 = CHIX, BEEF, FISH
F2 = PASTA, RICE, POTATO
F3 = BROCCOLI, BEANS, PEAS
F4 = WATER, SODA, BEER

$L_{RUNS}(3^{4}) = 9$ runs

	Factor 1	Factor 2	Factor 3	Factor 4
Run 1	0	0	0	0
Run 2	0	1	1	2
Run 3	0	2	2	1
Run 4	1	0	1	1
Run 5	1	1	2	0
Run 6	1	2	0	2
Run 7	2	0	2	2
Run 8	2	1	0	1
Run 9	2	2	1	0

http://www.freequality.org/sites/www_freequality_org/documents/tools/Tagarray_files/tamatrix.htm

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

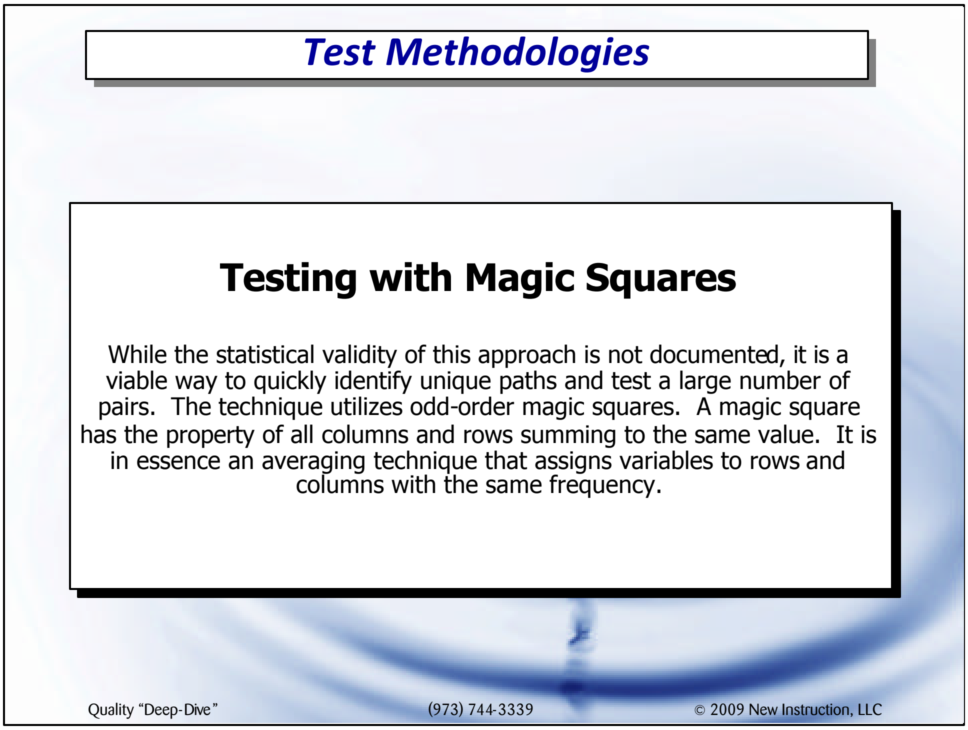
The Example provided (Lunch options on the slide) shows how that 9 runs are required to test all combinations.

Getting the TABLE is the MOST IMPORTANT part of this process!

Factors map to the options available (Factor 0 = the first selection / option for each factor, Factor 1 – the second selection / option for each factor, etc.

If a field is optional, make sure that blank is included as one of the options.

To find the table, see the URL on the slide, or just search Internet for OATS tables



Test Methodologies

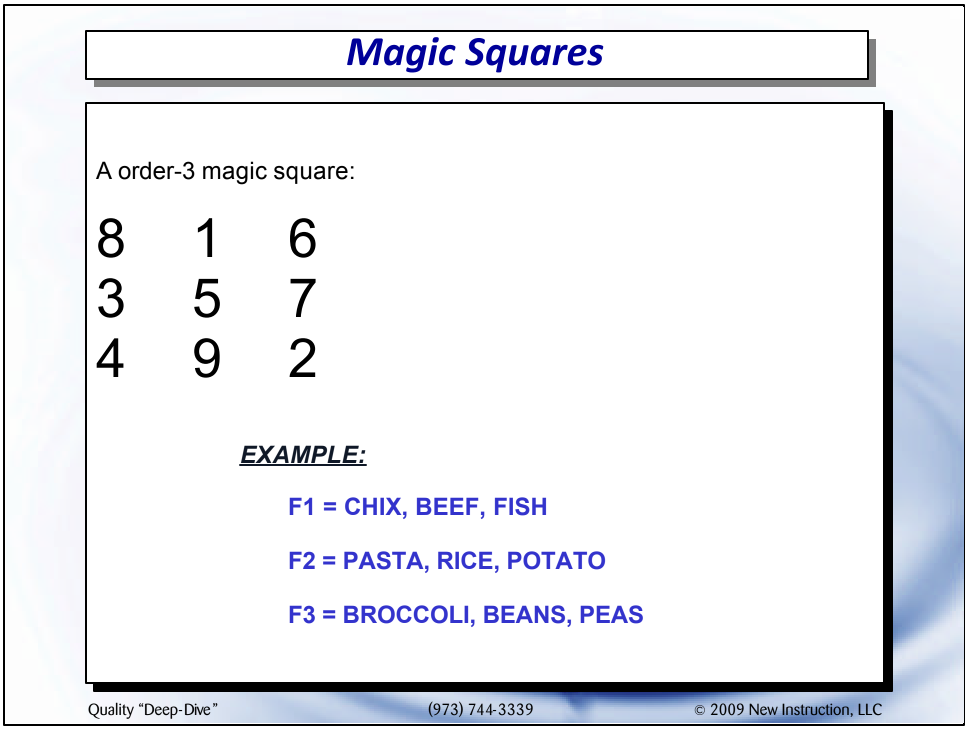
Testing with Magic Squares

While the statistical validity of this approach is not documented, it is a viable way to quickly identify unique paths and test a large number of pairs. The technique utilizes odd-order magic squares. A magic square has the property of all columns and rows summing to the same value. It is in essence an averaging technique that assigns variables to rows and columns with the same frequency.

Order options sequentially starting at 1.

Map numbers to magic square, and build test cases based on horizontal and/or vertical combinations.

Code Toad website has combinations and Magic Squares.



Magic Squares

A order-3 magic square:

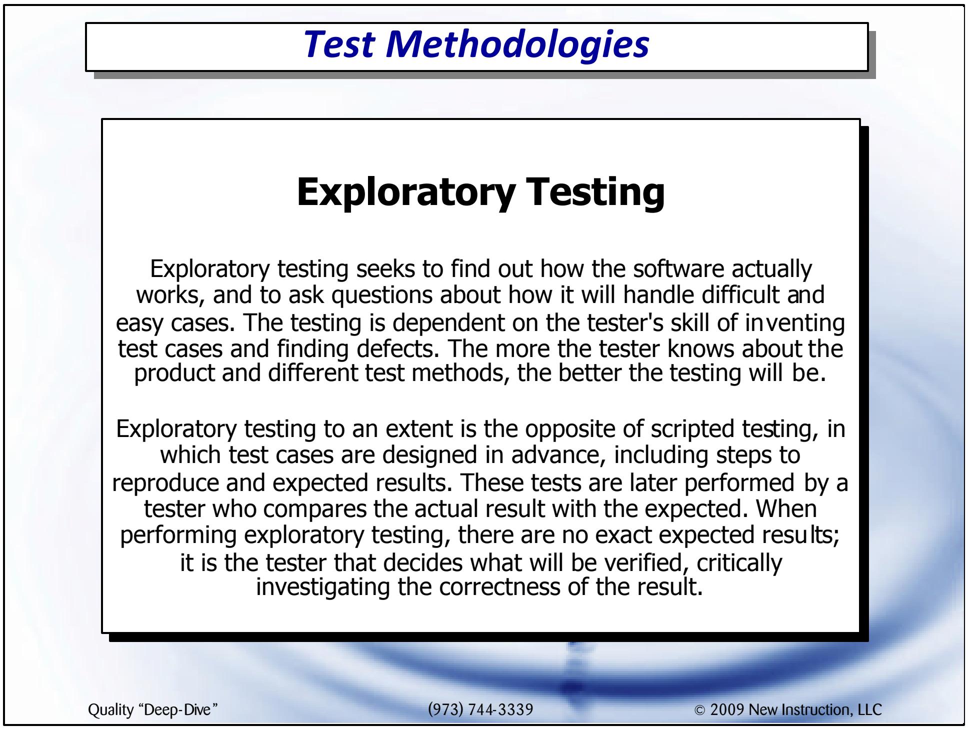
8	1	6
3	5	7
4	9	2

EXAMPLE:

F1 = CHIX, BEEF, FISH

F2 = PASTA, RICE, POTATO

F3 = BROCCOLI, BEANS, PEAS



Test Methodologies

Exploratory Testing

Exploratory testing seeks to find out how the software actually works, and to ask questions about how it will handle difficult and easy cases. The testing is dependent on the tester's skill of inventing test cases and finding defects. The more the tester knows about the product and different test methods, the better the testing will be.

Exploratory testing to an extent is the opposite of scripted testing, in which test cases are designed in advance, including steps to reproduce and expected results. These tests are later performed by a tester who compares the actual result with the expected. When performing exploratory testing, there are no exact expected results; it is the tester that decides what will be verified, critically investigating the correctness of the result.

Starts with basic testing and moves to unscripted testing based on tester knowledge. (I wonder what happens if I . . . . .)

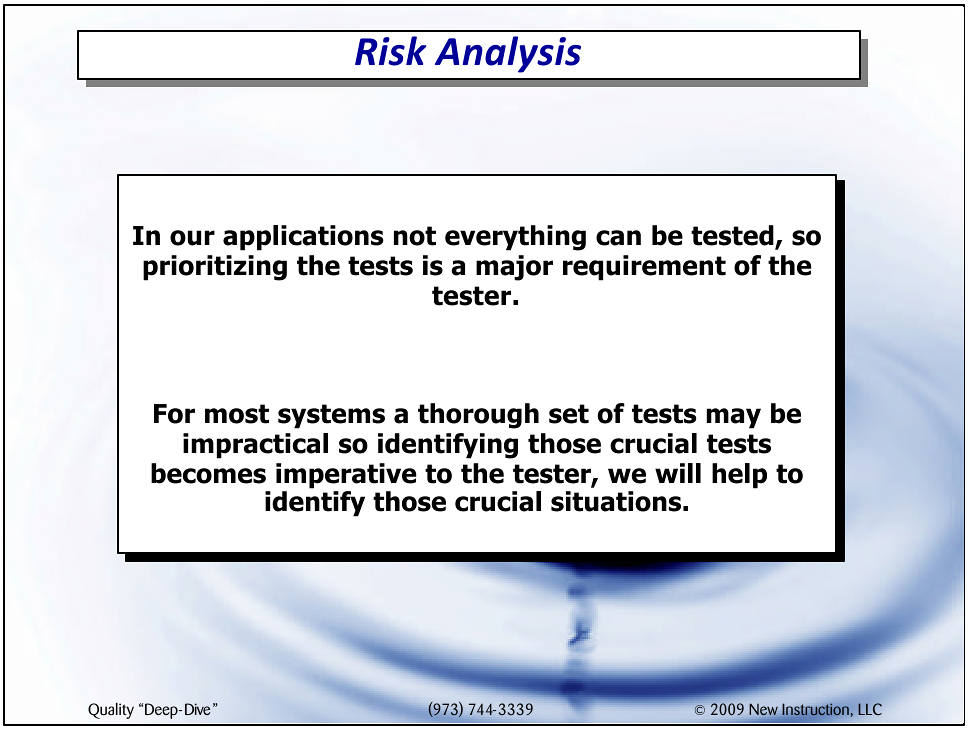
The key to this testing is that it won't do everything, but it allows greater flexibility in our testing. The difficulty with this type of testing is to be able to define exactly what happened to generate the error so the error can be recreated.

First phase – not documented, not scripted.

Second phase – should be documented and scripted to become part of the test suite.

Risk Analysis

Section 2

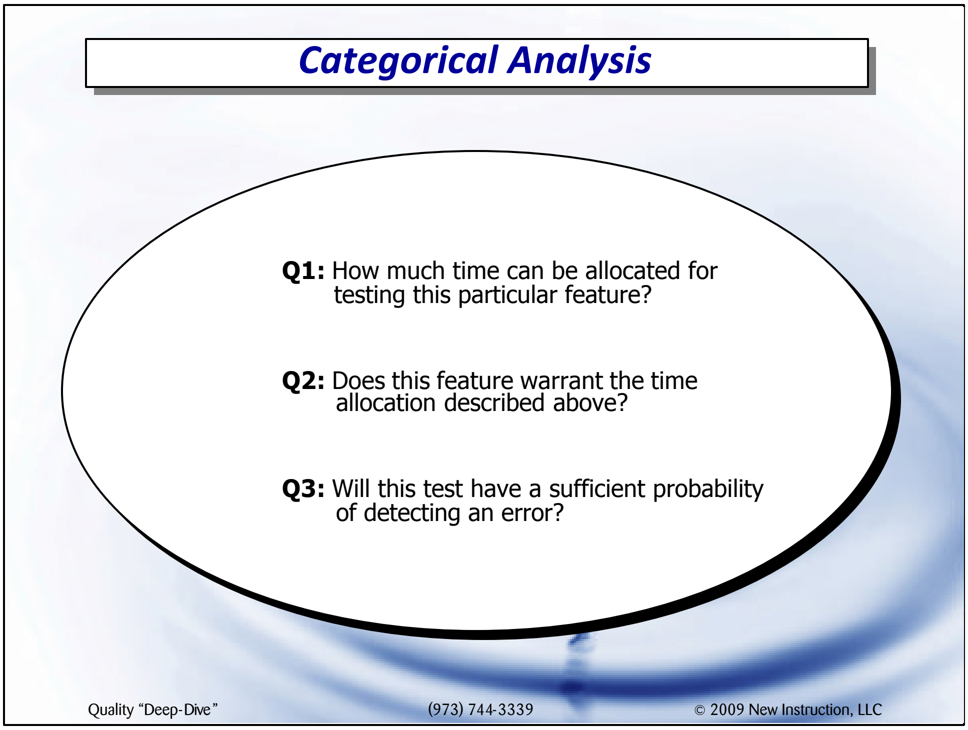


Risk Analysis

In our applications not everything can be tested, so prioritizing the tests is a major requirement of the tester.

For most systems a thorough set of tests may be impractical so identifying those crucial tests becomes imperative to the tester, we will help to identify those crucial situations.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



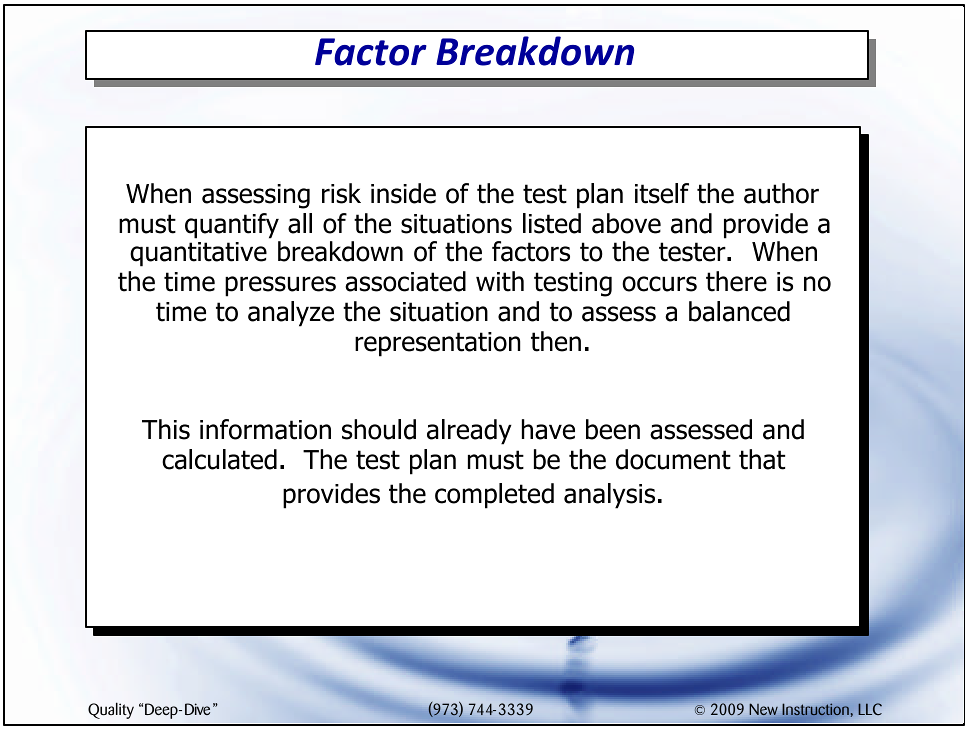
Categorical Analysis

Q1: How much time can be allocated for testing this particular feature?

Q2: Does this feature warrant the time allocation described above?

Q3: Will this test have a sufficient probability of detecting an error?

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

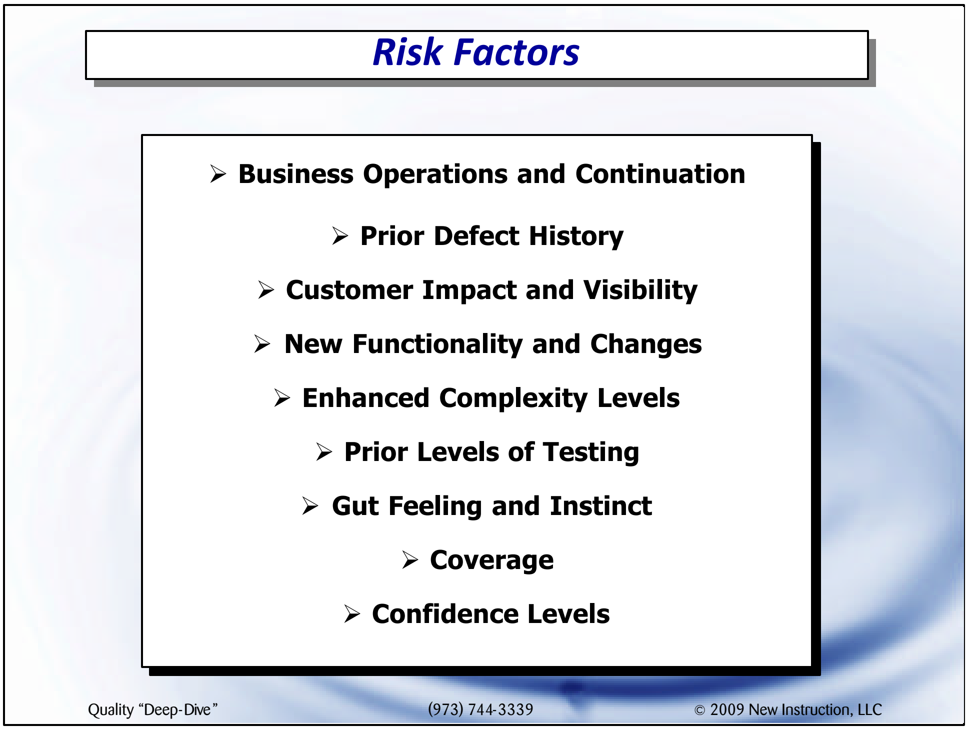


Factor Breakdown

When assessing risk inside of the test plan itself the author must quantify all of the situations listed above and provide a quantitative breakdown of the factors to the tester. When the time pressures associated with testing occurs there is no time to analyze the situation and to assess a balanced representation then.

This information should already have been assessed and calculated. The test plan must be the document that provides the completed analysis.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



Risk Factors

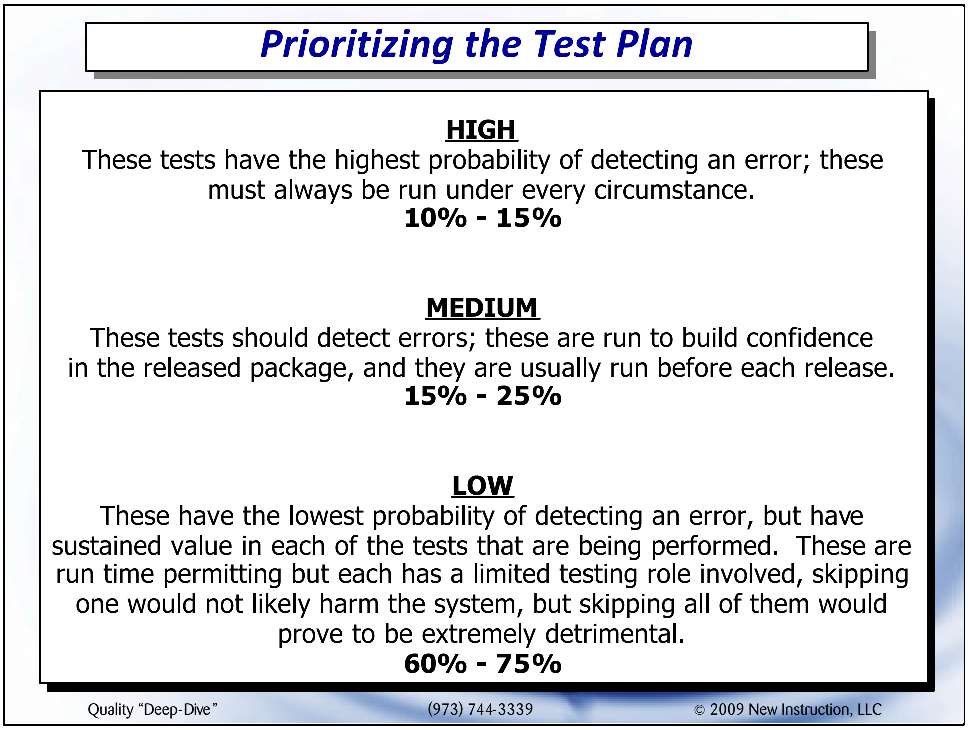
- ➢ **Business Operations and Continuation**
- ➢ **Prior Defect History**
- ➢ **Customer Impact and Visibility**
- ➢ **New Functionality and Changes**
- ➢ **Enhanced Complexity Levels**
- ➢ **Prior Levels of Testing**
- ➢ **Gut Feeling and Instinct**
- ➢ **Coverage**
- ➢ **Confidence Levels**

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Must be determined as part of test planning – too late to try and do this in the actual testing window.

Help determine the number of test cases required in a particular system area.

Confidence Levels – Developers' confidence in the work completed – lower confidence should indicate need for additional testing.



Prioritizing the Test Plan

HIGH

These tests have the highest probability of detecting an error; these must always be run under every circumstance.

10% - 15%

MEDIUM

These tests should detect errors; these are run to build confidence in the released package, and they are usually run before each release.

15% - 25%

LOW

These have the lowest probability of detecting an error, but have sustained value in each of the tests that are being performed. These are run time permitting but each has a limited testing role involved, skipping one would not likely harm the system, but skipping all of them would prove to be extremely detrimental.

60% - 75%

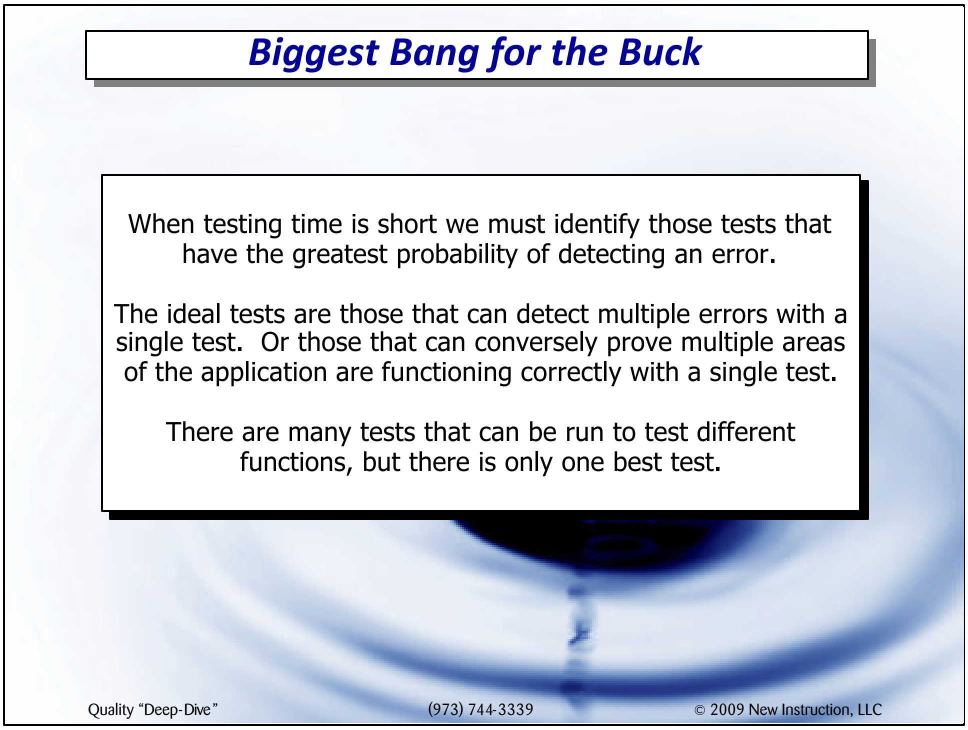
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Identifies the make-up of the DOCUMENTED project test plan.

High - 10-15% - includes critical functions - things that would stop all business continuation.

Medium - 15-25% - includes combinational errors / hiders / no showstoppers.

Low – 60-75% - typically test only one feature / one function.

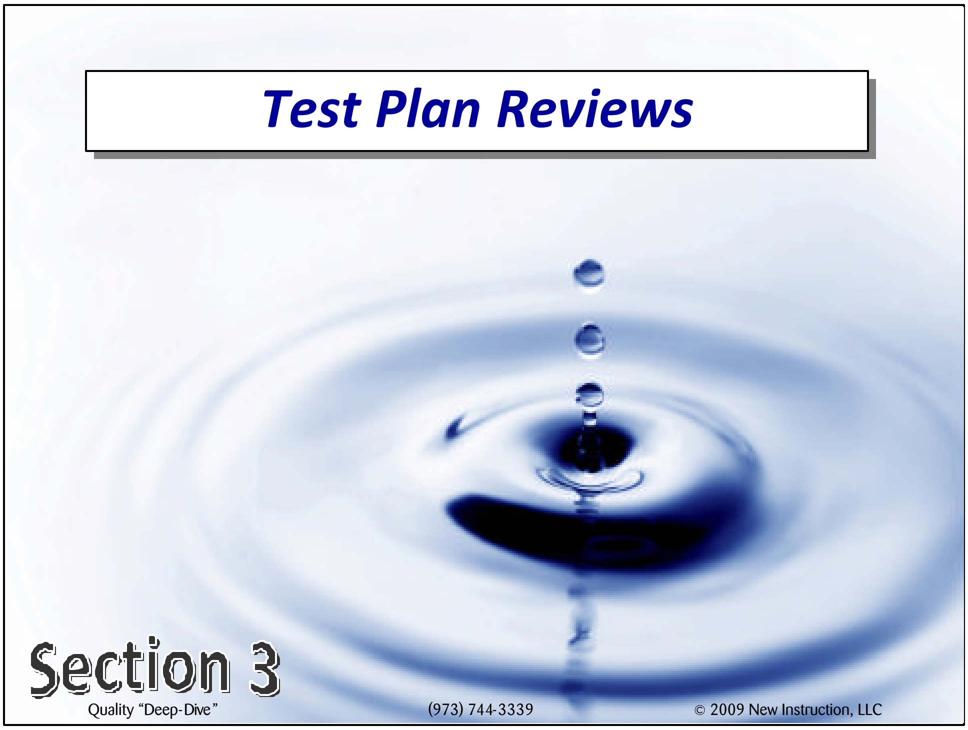


Biggest Bang for the Buck

When testing time is short we must identify those tests that have the greatest probability of detecting an error.

The ideal tests are those that can detect multiple errors with a single test. Or those that can conversely prove multiple areas of the application are functioning correctly with a single test.

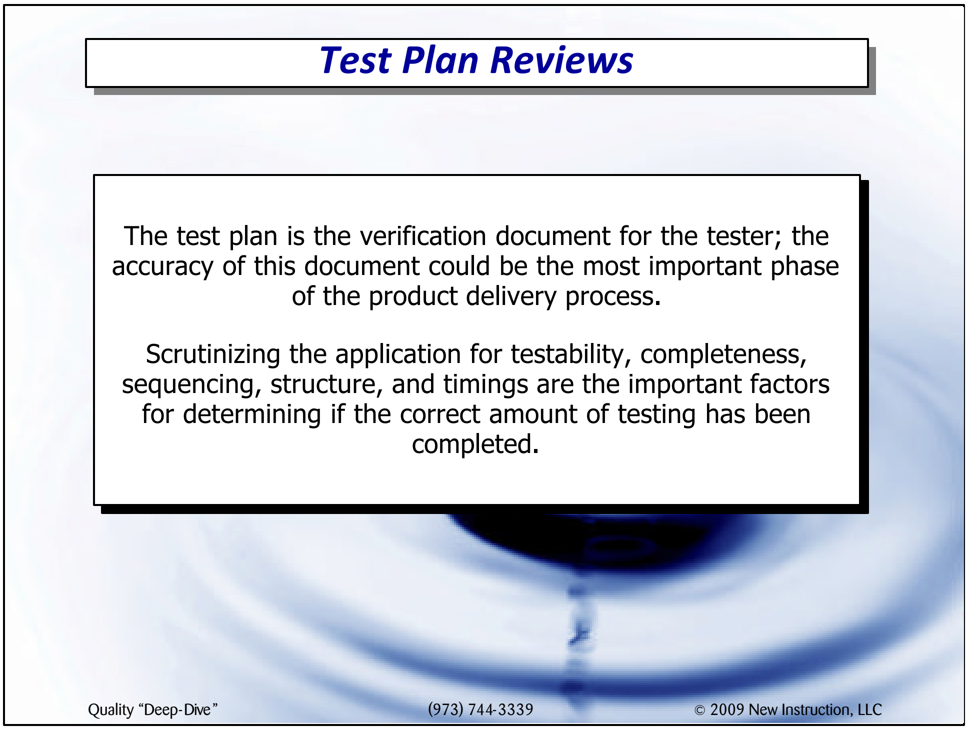
There are many tests that can be run to test different functions, but there is only one best test.



Test Plan Reviews

Section 3

Quality “Deep-Dive” (973) 744-3339 © 2009 New Instruction, LLC

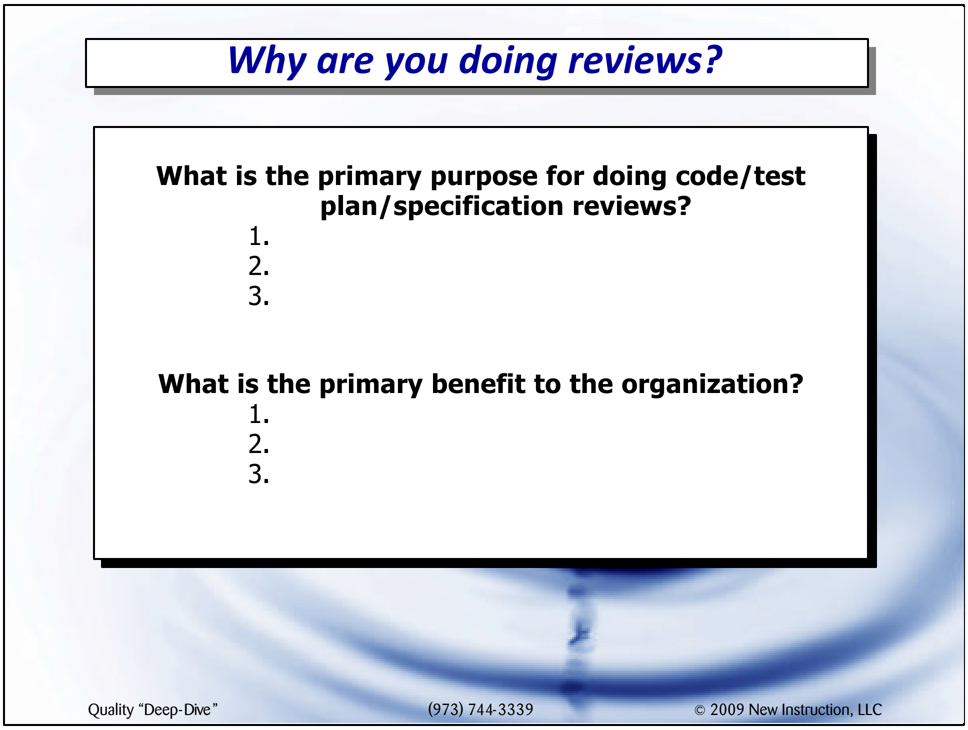


Test Plan Reviews

The test plan is the verification document for the tester; the accuracy of this document could be the most important phase of the product delivery process.

Scrutinizing the application for testability, completeness, sequencing, structure, and timings are the important factors for determining if the correct amount of testing has been completed.

Test plan reviews serve as verification that plans are detailed enough for someone other than the author to be able to execute a particular test.



Why are you doing reviews?

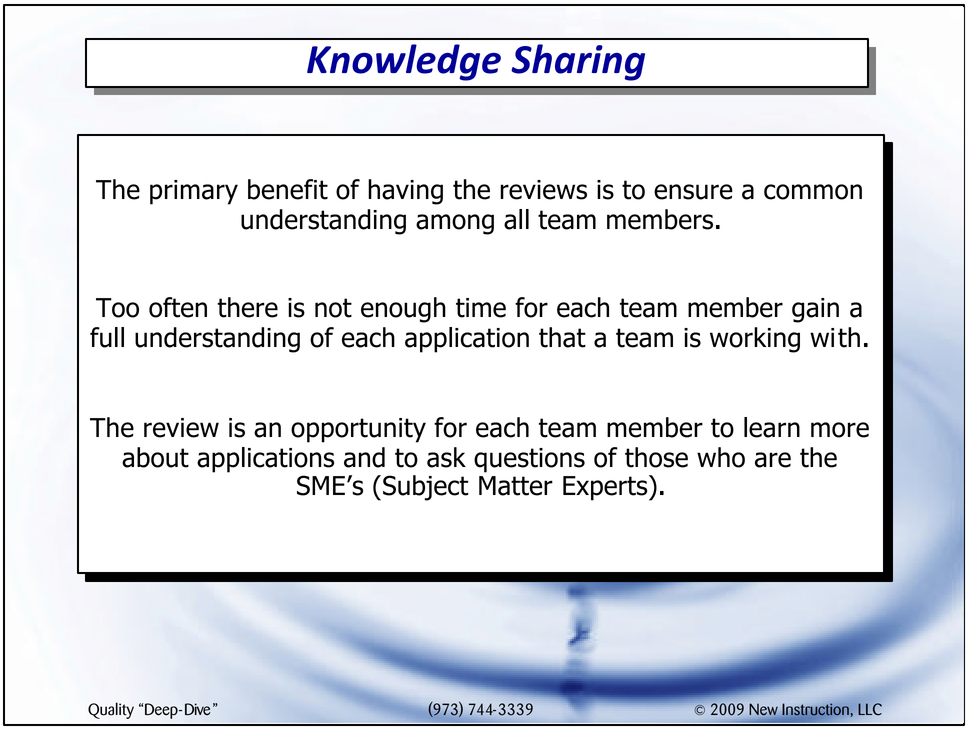
What is the primary purpose for doing code/test plan/specification reviews?

1.
2.
3.

What is the primary benefit to the organization?

1.
2.
3.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



Knowledge Sharing

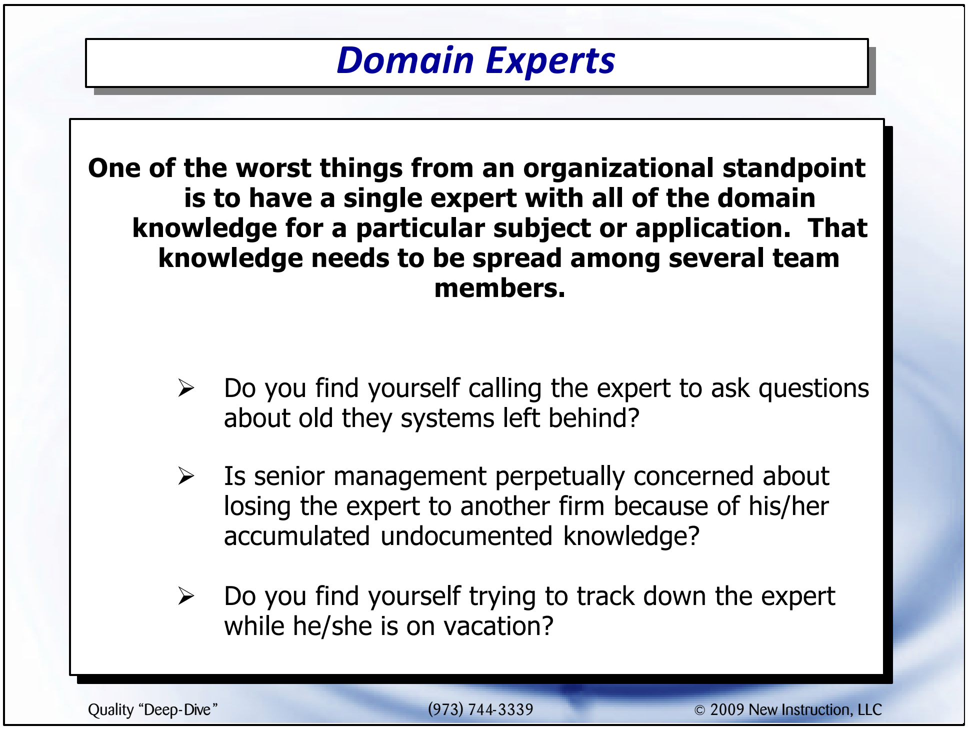
The primary benefit of having the reviews is to ensure a common understanding among all team members.

Too often there is not enough time for each team member gain a full understanding of each application that a team is working with.

The review is an opportunity for each team member to learn more about applications and to ask questions of those who are the SME's (Subject Matter Experts).

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Primary purpose for doing reviews – Knowledge transfer. Reviews are excellent training tools for new employees – they can hear questions being asked and learn from both questions and answers.



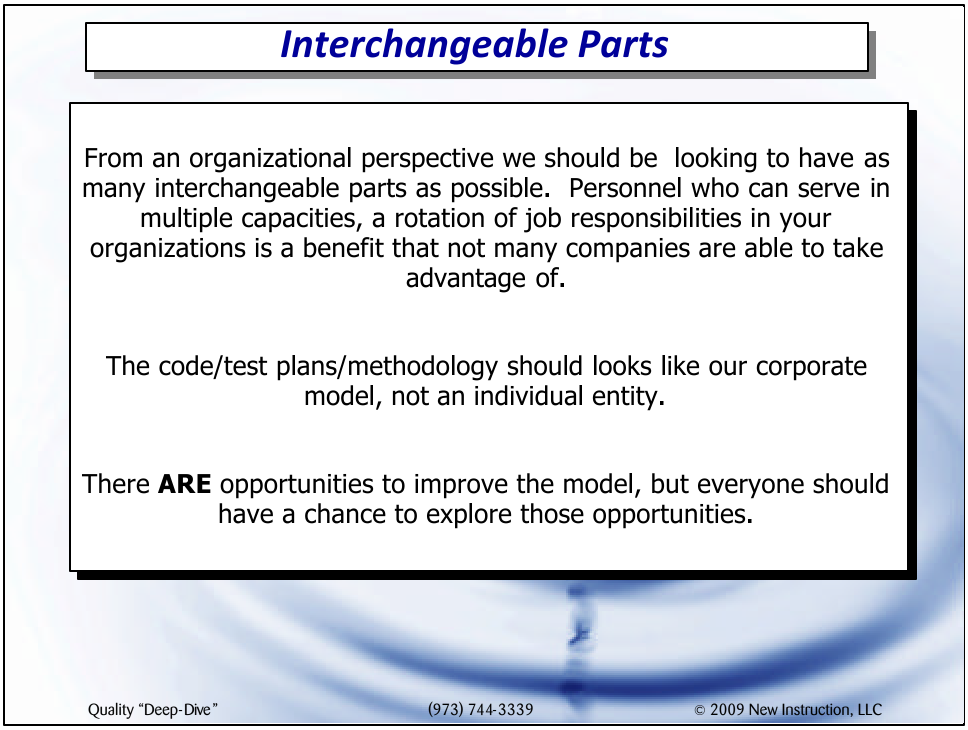
Domain Experts

One of the worst things from an organizational standpoint is to have a single expert with all of the domain knowledge for a particular subject or application. That knowledge needs to be spread among several team members.

- Do you find yourself calling the expert to ask questions about old they systems left behind?
- Is senior management perpetually concerned about losing the expert to another firm because of his/her accumulated undocumented knowledge?
- Do you find yourself trying to track down the expert while he/she is on vacation?

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

ALWAYS have more than one expert on a certain domain area. Encourage 'interchangeable parts' approach. This requires excellent process and product documentation and standard processes and documents.



Interchangeable Parts

From an organizational perspective we should be looking to have as many interchangeable parts as possible. Personnel who can serve in multiple capacities, a rotation of job responsibilities in your organizations is a benefit that not many companies are able to take advantage of.

The code/test plans/methodology should looks like our corporate model, not an individual entity.

There **ARE** opportunities to improve the model, but everyone should have a chance to explore those opportunities.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Author provides a review session document, including test plan, requirements being tested, test data information to all participants.

Participants include 2 – 3 other testers, 1 – 2 programmers (hopefully with application knowledge), 1 - 2 BAs (with application knowledge), 1 – 2 observers (new employees present to learn).

One of the other testers is charged with presenting the test plan – this is important because it gets around pride of ownership issues, provides a check on thoroughness of test plan (because if presenter can't provide information, then test plan is not detailed enough).

Techs are present to understand why tests are being conducted as planned, to ensure that they have complete understanding of requirement.

BAs are present to ensure that customers' needs are being met,

Testers are present to ensure they understand what testing will be done and how testing will take place.

Review meeting should be no longer than 1 – 1.5 hours.

In the actual Review meeting, if anyone hasn't reviewed material prior to the meeting, they are asked to leave – because they will slow down the meeting unacceptably.

Can go line by line through test plan, or better, is to go around the room asking each participant what questions they had.

Author is present to answer any questions that the presenter can't cover – in which situation the test plan should be updated with the new information.

In last 10 minutes of the meeting, next steps / updates required are summarized

A follow up meeting, which should take no longer than 30 minutes, is scheduled to ensure that all updates have been completed.

Testers and BAs should be included in CODE REVIEWS – if non-technicians can understand the logic of the program, then technicians who have to actually do the code should easily understand the work before them.

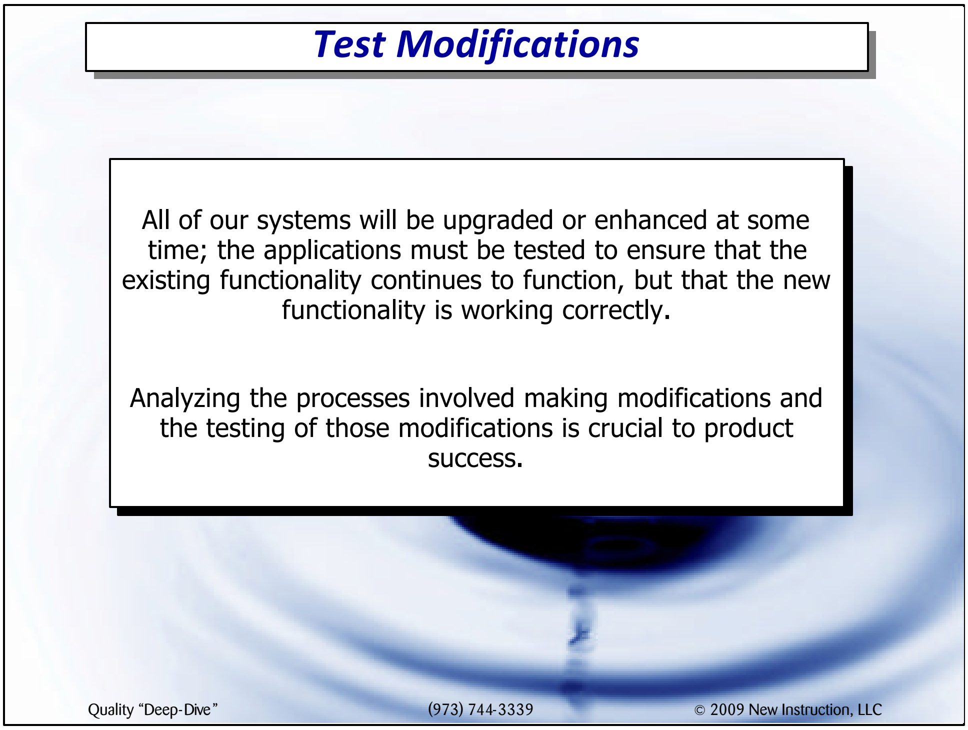
Test Modifications

Section 4

Quality “Deep-Dive”

(973) 744-3339

© 2009 New Instruction, LLC

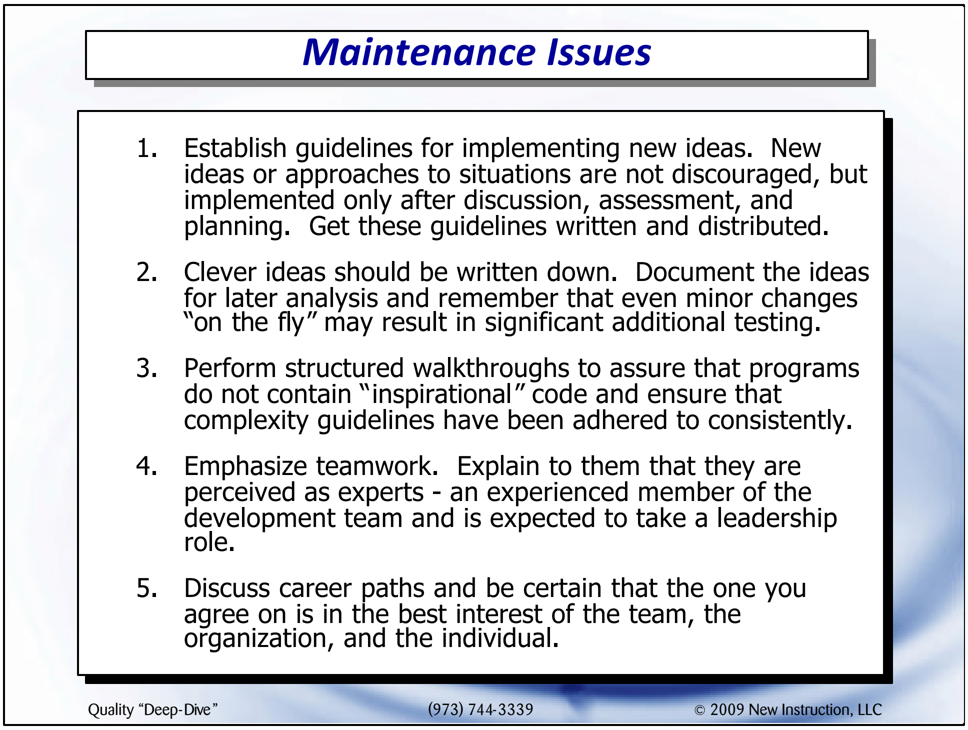


Test Modifications

All of our systems will be upgraded or enhanced at some time; the applications must be tested to ensure that the existing functionality continues to function, but that the new functionality is working correctly.

Analyzing the processes involved making modifications and the testing of those modifications is crucial to product success.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

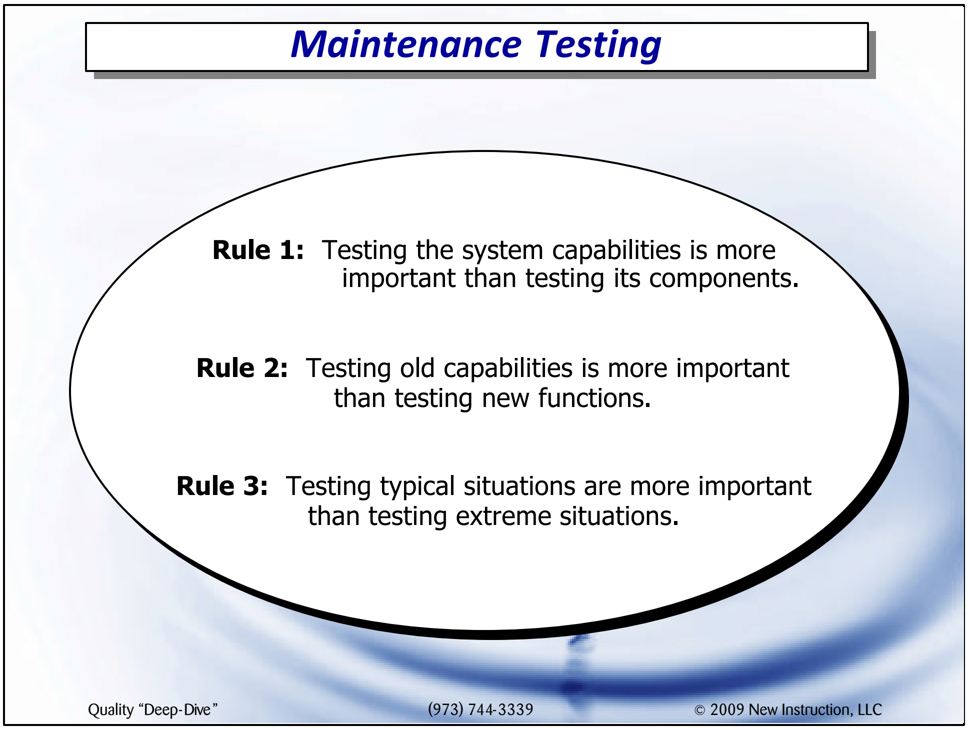


Maintenance Issues

1. Establish guidelines for implementing new ideas. New ideas or approaches to situations are not discouraged, but implemented only after discussion, assessment, and planning. Get these guidelines written and distributed.
2. Clever ideas should be written down. Document the ideas for later analysis and remember that even minor changes "on the fly" may result in significant additional testing.
3. Perform structured walkthroughs to assure that programs do not contain "inspirational" code and ensure that complexity guidelines have been adhered to consistently.
4. Emphasize teamwork. Explain to them that they are perceived as experts - an experienced member of the development team and is expected to take a leadership role.
5. Discuss career paths and be certain that the one you agree on is in the best interest of the team, the organization, and the individual.

Need documented guidelines for implementing new ideas.

No 'on the fly' changes. No 'inspirational code' – that only one person can understand / maintain.

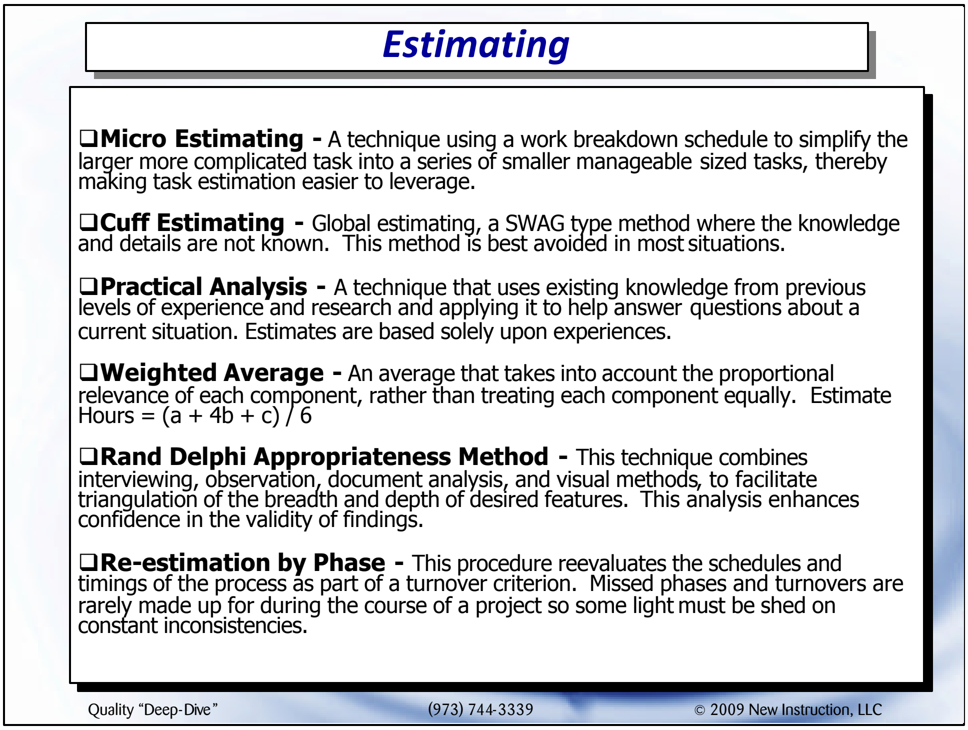


Maintenance Testing

Rule 1: Testing the system capabilities is more important than testing its components.

Rule 2: Testing old capabilities is more important than testing new functions.

Rule 3: Testing typical situations are more important than testing extreme situations.



Estimating

❑**Micro Estimating -** A technique using a work breakdown schedule to simplify the larger more complicated task into a series of smaller manageable sized tasks, thereby making task estimation easier to leverage.

❑**Cuff Estimating -** Global estimating, a SWAG type method where the knowledge and details are not known. This method is best avoided in most situations.

❑**Practical Analysis -** A technique that uses existing knowledge from previous levels of experience and research and applying it to help answer questions about a current situation. Estimates are based solely upon experiences.

❑**Weighted Average -** An average that takes into account the proportional relevance of each component, rather than treating each component equally. Estimate Hours = (a + 4b + c) / 6

❑**Rand Delphi Appropriateness Method -** This technique combines interviewing, observation, document analysis, and visual methods, to facilitate triangulation of the breadth and depth of desired features. This analysis enhances confidence in the validity of findings.

❑**Re-estimation by Phase -** This procedure reevaluates the schedules and timings of the process as part of a turnover criterion. Missed phases and turnovers are rarely made up for during the course of a project so some light must be shed on constant inconsistencies.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Micro-Estimating

Most precise, preferred method of estimating.

Break project into component phases, activities, tasks – estimate at task level and roll up to develop the project level estimate.

Most error-prone method of estimating – if not broken down correctly or if pieces are missed, then estimates are inaccurate.

Requires intricate knowledge of the function.

Top Down / Global

Off the top of the head.

Doesn't include detailed review of content / tasks.

Weighted Average

Good first technique if not using any formal method currently.

(A + 4B + C) / 6, where estimate A = best case, estimate B = most likely, and estimate C = worst case.

Rand – Delphi Appropriateness

Bring 3 – 4 very knowledgeable people together, have them separately document their estimate, including their concerns and rationale in developing the estimate.

They rotate estimates, and update the estimate in front of them, in turn, for all estimates developed.

End result is a set of documents containing estimates and factors / considerations in building the estimate and plans.

Re-Estimation by Phase

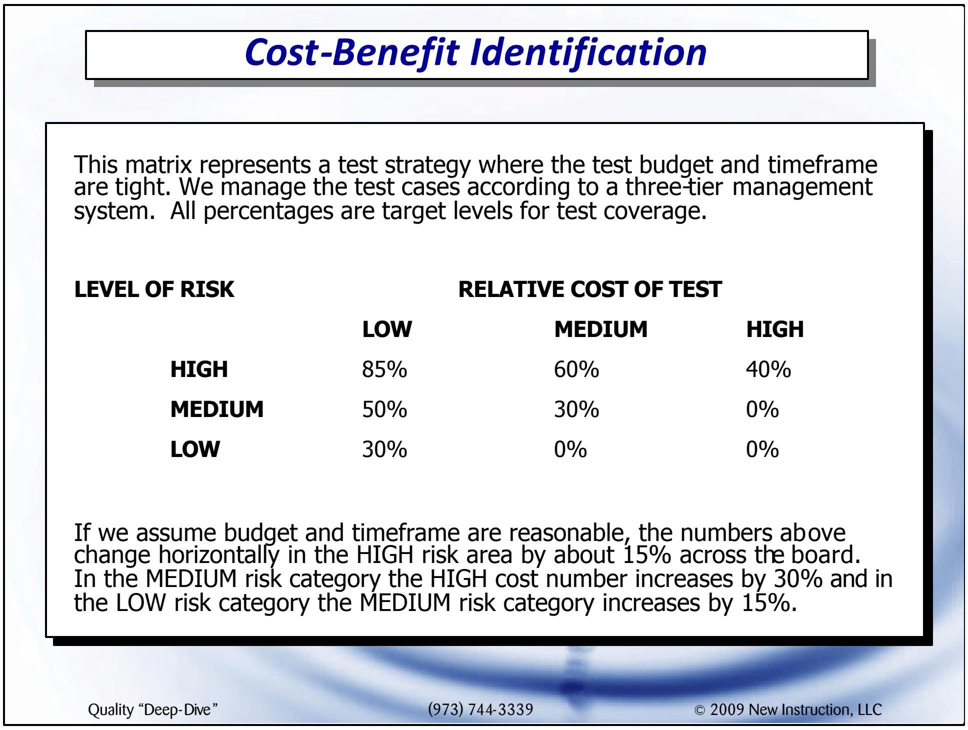
Develop detailed estimate by phase for each phase based on output from prior phase activities.

Historical Estimates – base estimates on actual historical data from prior projects

Compare current project to prior comparable projects.

Must compare to ACTUALS not to prior project estimates.

Estimation is NOT negotiation.



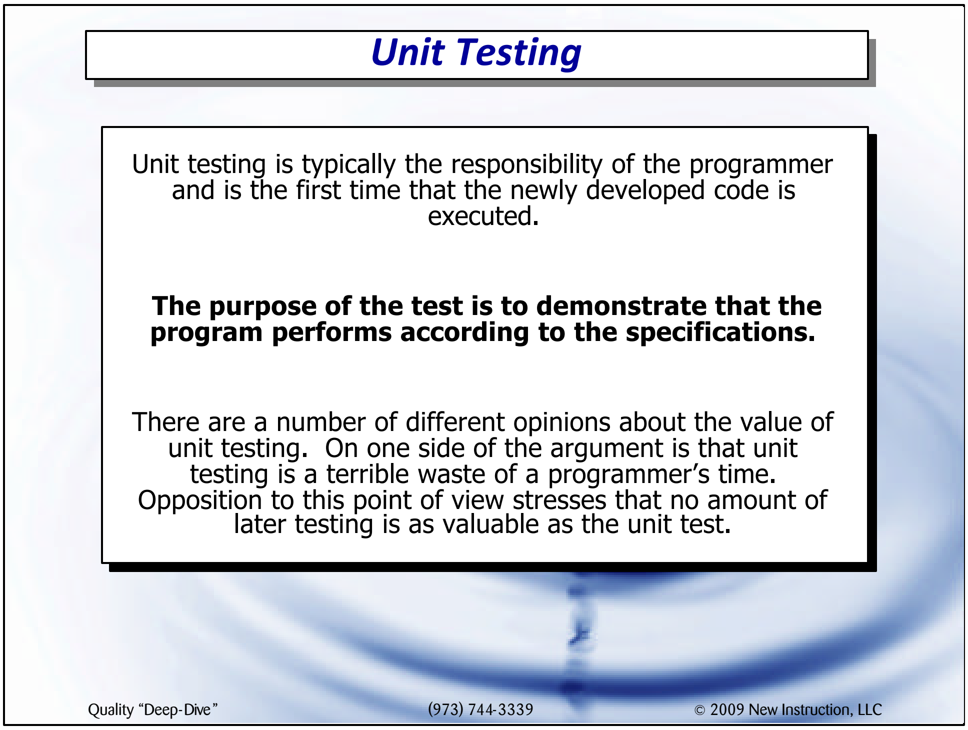
Cost-Benefit Identification

This matrix represents a test strategy where the test budget and timeframe are tight. We manage the test cases according to a three-tier management system. All percentages are target levels for test coverage.

LEVEL OF RISK	**RELATIVE COST OF TEST**		
	LOW	**MEDIUM**	**HIGH**
HIGH	85%	60%	40%
MEDIUM	50%	30%	0%
LOW	30%	0%	0%

If we assume budget and timeframe are reasonable, the numbers above change horizontally in the HIGH risk area by about 15% across the board. In the MEDIUM risk category the HIGH cost number increases by 30% and in the LOW risk category the MEDIUM risk category increases by 15%.

Focuses on ROI for test investment.

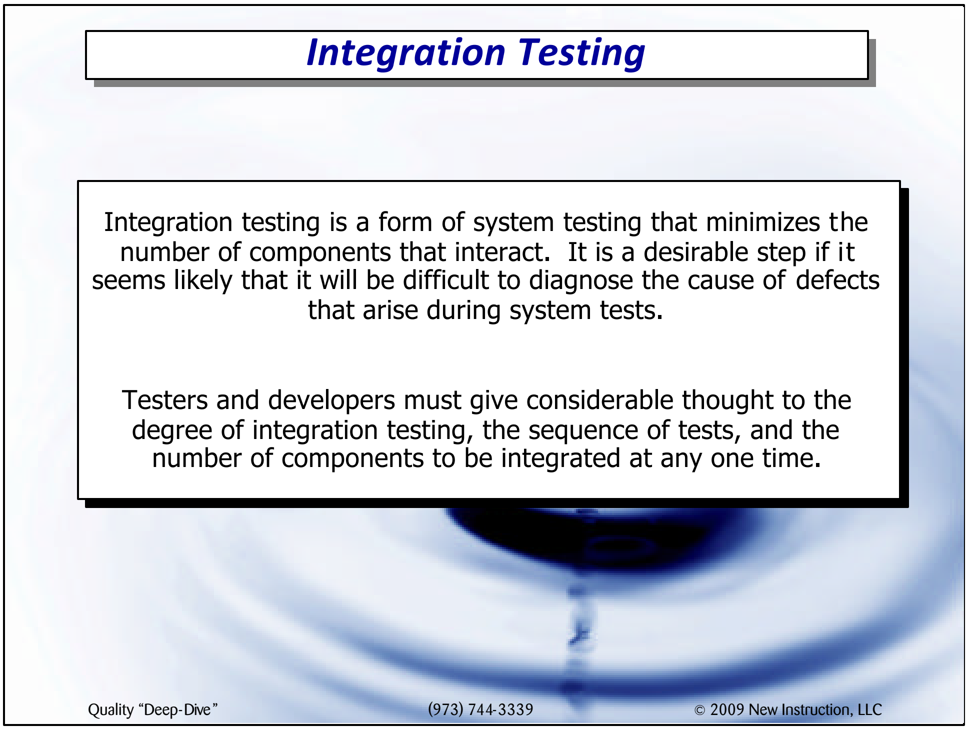


Unit Testing

Unit testing is typically the responsibility of the programmer and is the first time that the newly developed code is executed.

The purpose of the test is to demonstrate that the program performs according to the specifications.

There are a number of different opinions about the value of unit testing. On one side of the argument is that unit testing is a terrible waste of a programmer's time. Opposition to this point of view stresses that no amount of later testing is as valuable as the unit test.



Integration Testing

Integration testing is a form of system testing that minimizes the number of components that interact. It is a desirable step if it seems likely that it will be difficult to diagnose the cause of defects that arise during system tests.

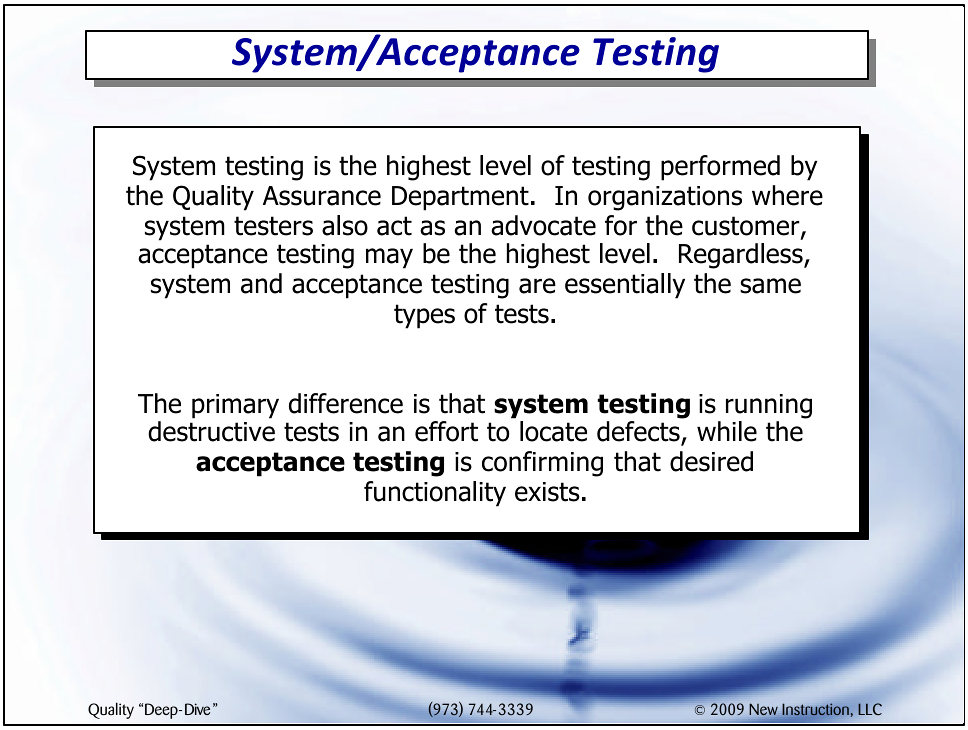
Testers and developers must give considerable thought to the degree of integration testing, the sequence of tests, and the number of components to be integrated at any one time.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Moves from programmer to system tester to execute.

Module / component level.

Continues until entire application is completely coded.



System/Acceptance Testing

System testing is the highest level of testing performed by the Quality Assurance Department. In organizations where system testers also act as an advocate for the customer, acceptance testing may be the highest level. Regardless, system and acceptance testing are essentially the same types of tests.

The primary difference is that **system testing** is running destructive tests in an effort to locate defects, while the **acceptance testing** is confirming that desired functionality exists.

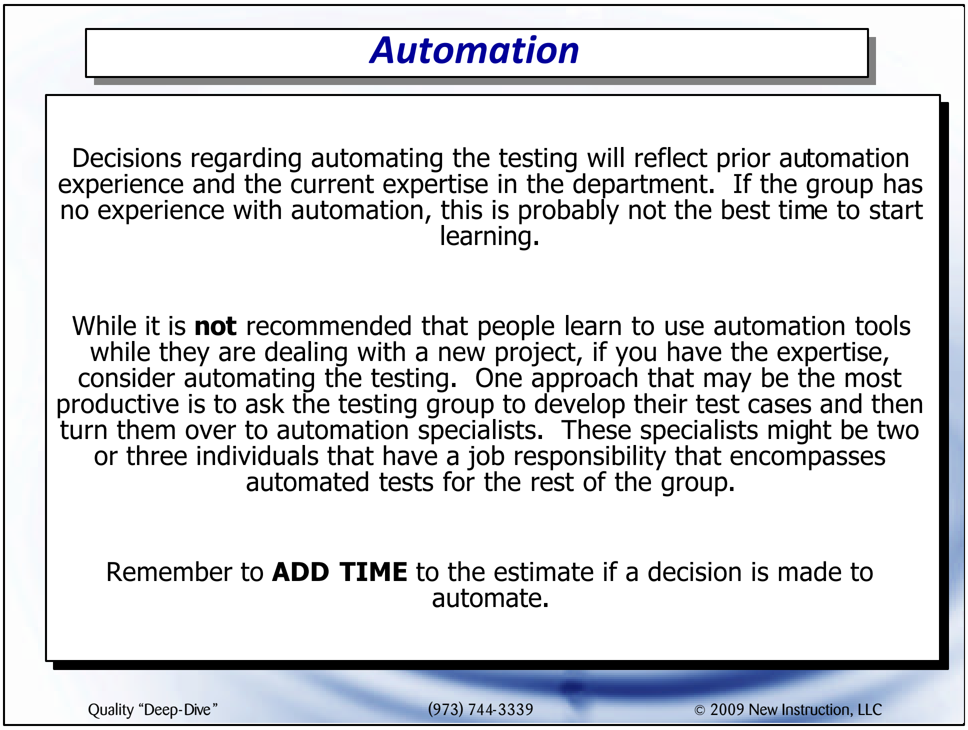
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

(Conducted by malicious QA group!)

System Testing – negative testing.

Acceptance Testing – positive testing – done after System testing is completed at a component level, usually by the same person.

Users don't always do a good job of acceptance, requiring QA or Systems testing to be more thorough.



Automation

Decisions regarding automating the testing will reflect prior automation experience and the current expertise in the department. If the group has no experience with automation, this is probably not the best time to start learning.

While it is **not** recommended that people learn to use automation tools while they are dealing with a new project, if you have the expertise, consider automating the testing. One approach that may be the most productive is to ask the testing group to develop their test cases and then turn them over to automation specialists. These specialists might be two or three individuals that have a job responsibility that encompasses automated tests for the rest of the group.

Remember to **ADD TIME** to the estimate if a decision is made to automate.

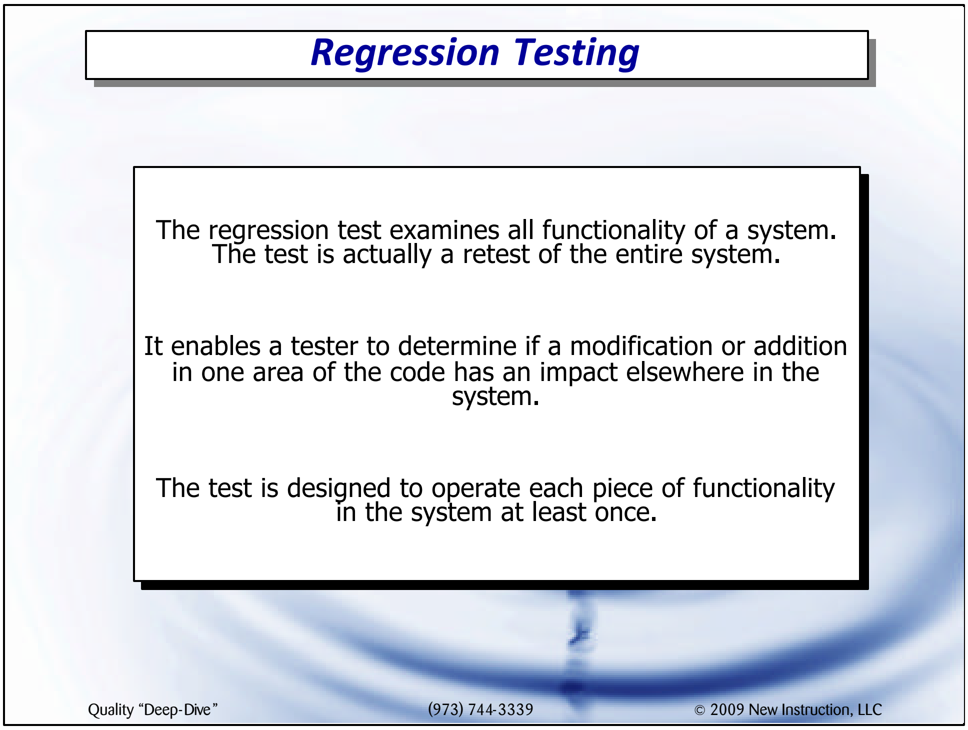
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

"A fool with a tool ... is still a fool."

Must know what to test, how to build a good test plan before attempting to automate.

Standard tests should be automated (60% on average), the remaining portion becomes manual. As systems become more stable, that percentage moves heavier toward manual testing.

Alfa uses Quick TestPro for automation, CSC P&C uses Compuware's Xpediter Code Coverage Tool.



Regression Testing

The regression test examines all functionality of a system. The test is actually a retest of the entire system.

It enables a tester to determine if a modification or addition in one area of the code has an impact elsewhere in the system.

The test is designed to operate each piece of functionality in the system at least once.

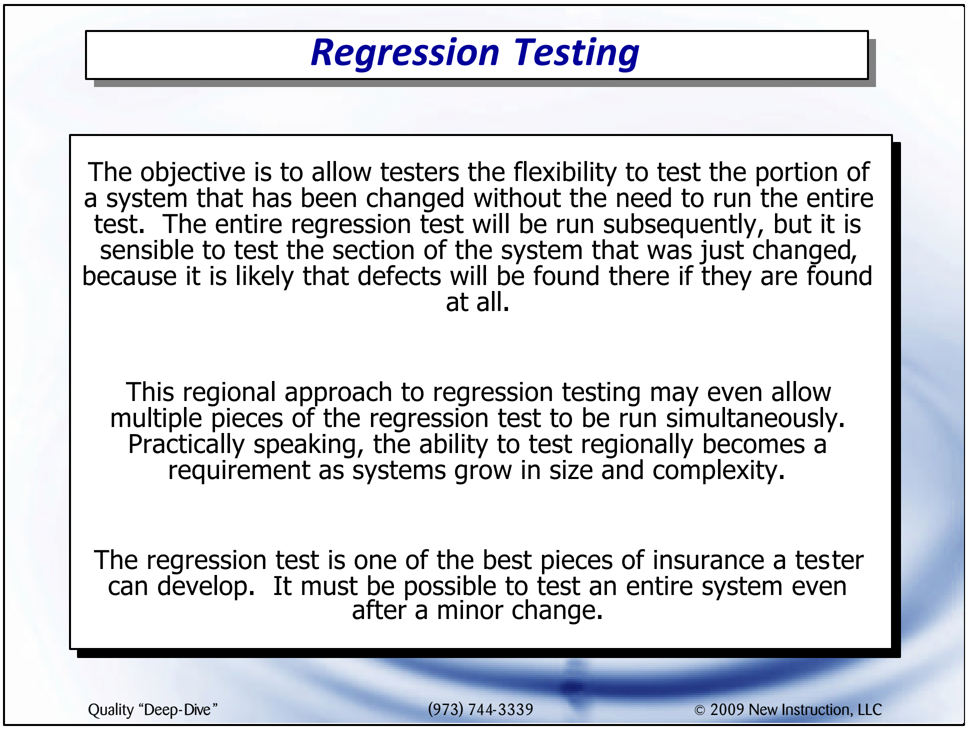
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Comprehensive test of entire system to ensure new code not inadvertently breaking the system.

Especially should focus on Critical Function.

Identify 'regional testing' pieces – sections that don't touch each other, which allow sections to be tested in parallel, reducing elapsed execution time.

> Example: 100 test cases in suite, for new release, run all 100 test cases, identify problem, fix problem, get next release, run entire test again, find new problems, fix problems, get next release, run entire test again, continue until all problems fixed OR re-test only the function that failed in previous execution and then on last pass, re-test everything that wasn't run on last successful test.



Regression Testing

The objective is to allow testers the flexibility to test the portion of a system that has been changed without the need to run the entire test. The entire regression test will be run subsequently, but it is sensible to test the section of the system that was just changed, because it is likely that defects will be found there if they are found at all.

This regional approach to regression testing may even allow multiple pieces of the regression test to be run simultaneously. Practically speaking, the ability to test regionally becomes a requirement as systems grow in size and complexity.

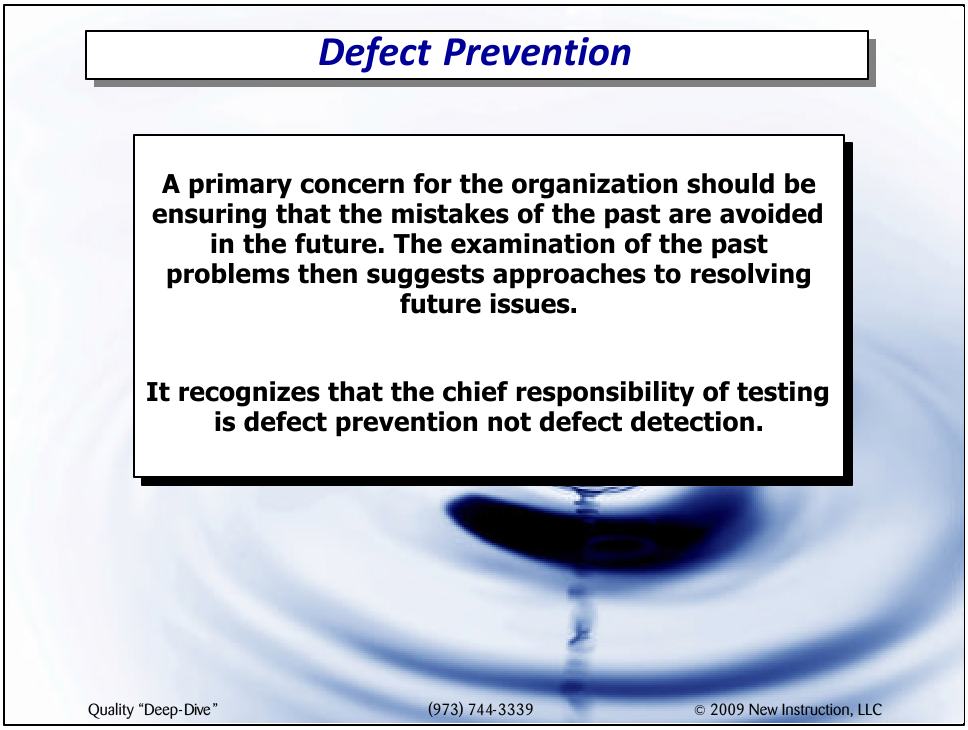
The regression test is one of the best pieces of insurance a tester can develop. It must be possible to test an entire system even after a minor change.

Regression test should NOT encounter a high find rate. Failures in regression test indicate a failure in earlier testing phases.

ANY defect that QC group finds cost justifies the existence of the QC group – because in theory, there should be NO defects available for QC to find, and without the group, those defects would have leaked out to the end users / customers.

Defect Prevention

Section 5



Defect Prevention

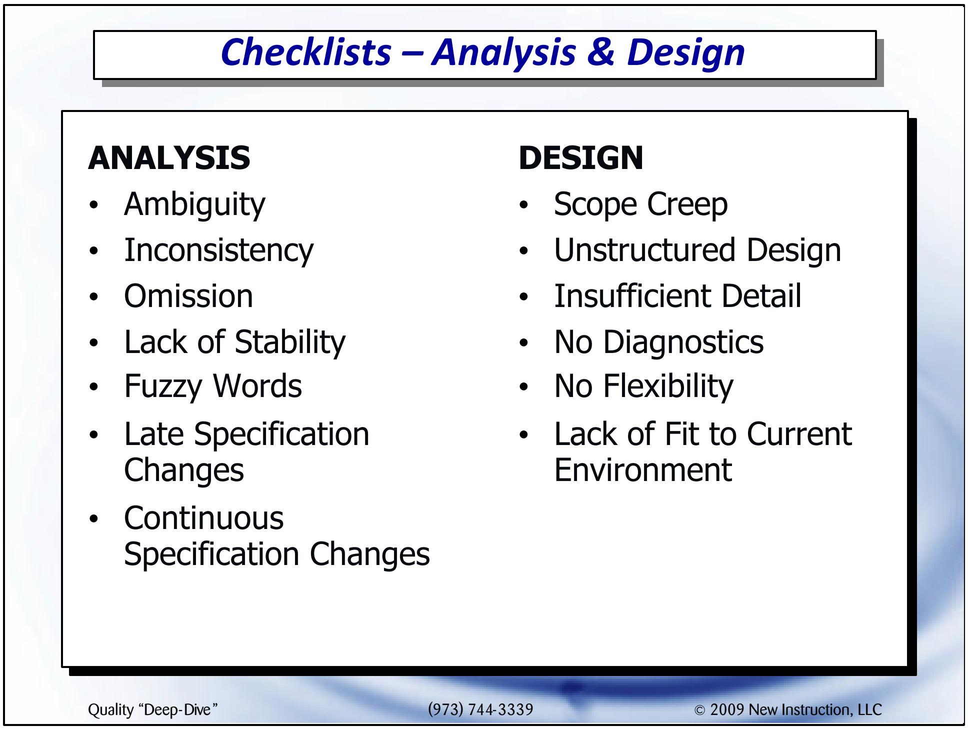
A primary concern for the organization should be ensuring that the mistakes of the past are avoided in the future. The examination of the past problems then suggests approaches to resolving future issues.

It recognizes that the chief responsibility of testing is defect prevention not defect detection.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

The chief responsibility of testing is defect prevention – not defect detection!

Different activities in different phases can help with prevention.



Checklists – Analysis & Design

ANALYSIS

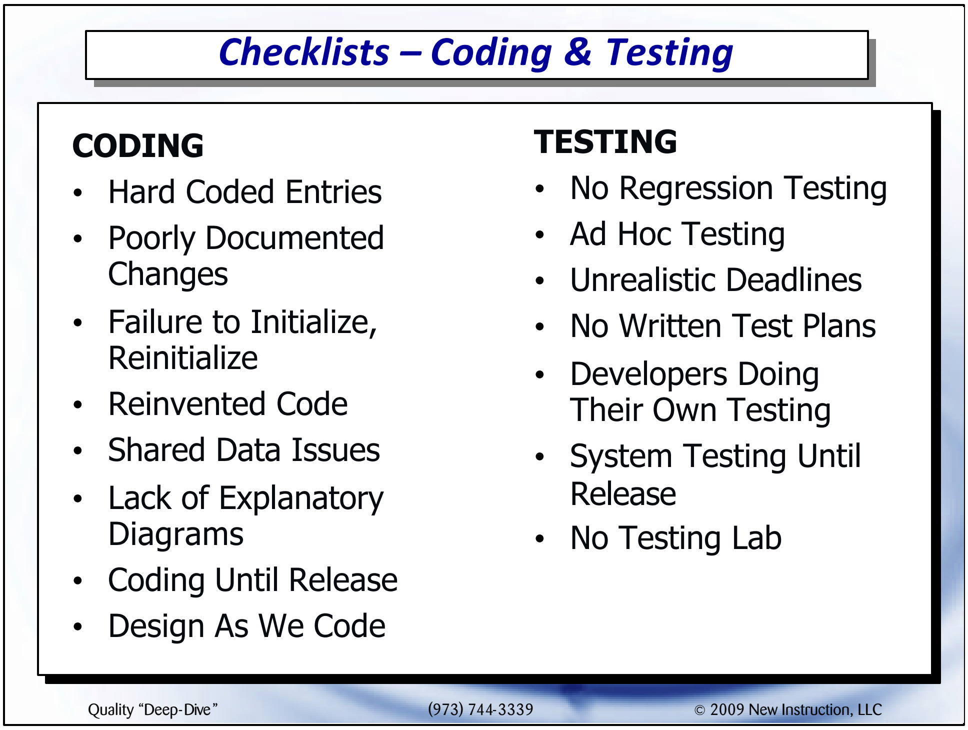
- Ambiguity
- Inconsistency
- Omission
- Lack of Stability
- Fuzzy Words
- Late Specification Changes
- Continuous Specification Changes

DESIGN

- Scope Creep
- Unstructured Design
- Insufficient Detail
- No Diagnostics
- No Flexibility
- Lack of Fit to Current Environment

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Checklists for life cycle phases highlight symptoms that result when defects are not being adequately prevented.



Checklists – Coding & Testing

CODING

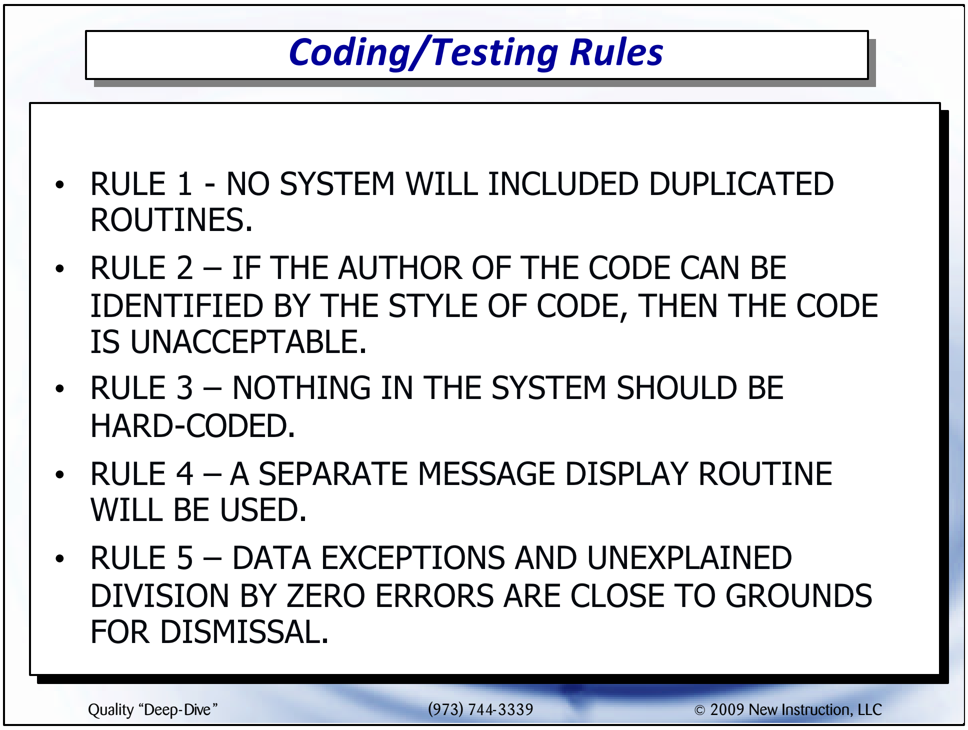
- Hard Coded Entries
- Poorly Documented Changes
- Failure to Initialize, Reinitialize
- Reinvented Code
- Shared Data Issues
- Lack of Explanatory Diagrams
- Coding Until Release
- Design As We Code

TESTING

- No Regression Testing
- Ad Hoc Testing
- Unrealistic Deadlines
- No Written Test Plans
- Developers Doing Their Own Testing
- System Testing Until Release
- No Testing Lab

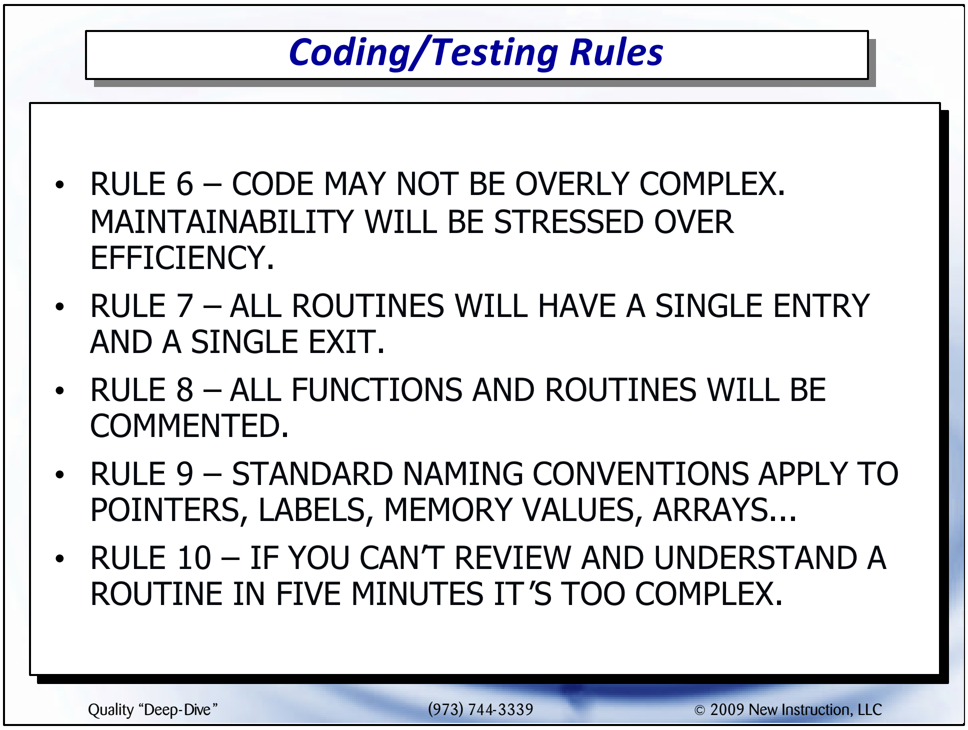
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Coding / Testing Rules should be in place in every organization.



Coding/Testing Rules

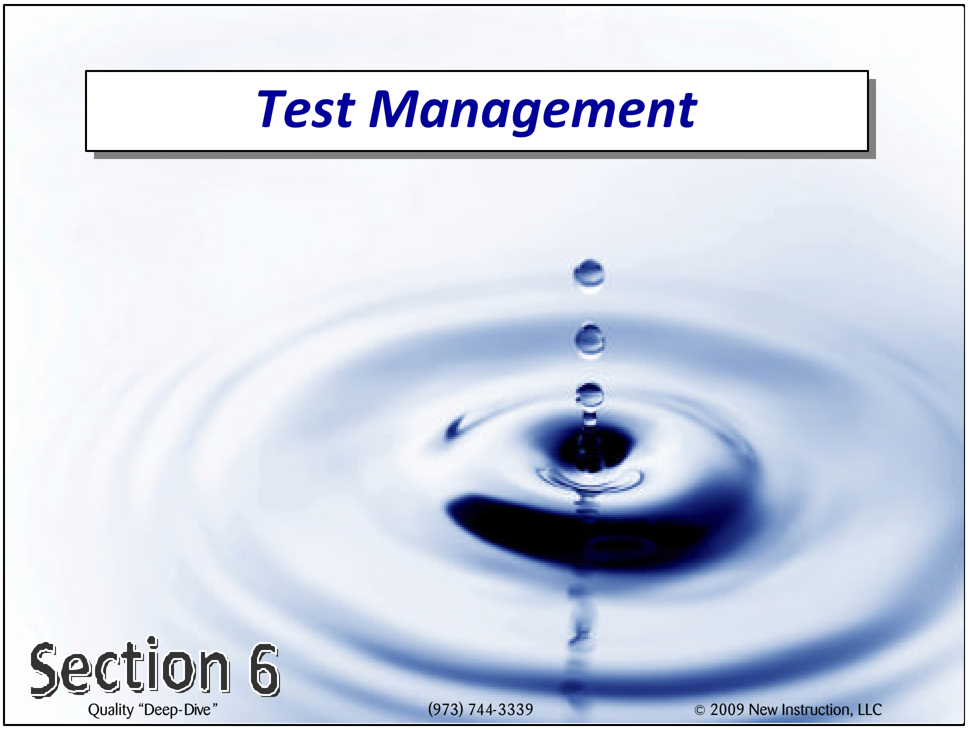
- RULE 1 - NO SYSTEM WILL INCLUDED DUPLICATED ROUTINES.
- RULE 2 – IF THE AUTHOR OF THE CODE CAN BE IDENTIFIED BY THE STYLE OF CODE, THEN THE CODE IS UNACCEPTABLE.
- RULE 3 – NOTHING IN THE SYSTEM SHOULD BE HARD-CODED.
- RULE 4 – A SEPARATE MESSAGE DISPLAY ROUTINE WILL BE USED.
- RULE 5 – DATA EXCEPTIONS AND UNEXPLAINED DIVISION BY ZERO ERRORS ARE CLOSE TO GROUNDS FOR DISMISSAL.



Coding/Testing Rules

- RULE 6 – CODE MAY NOT BE OVERLY COMPLEX. MAINTAINABILITY WILL BE STRESSED OVER EFFICIENCY.
- RULE 7 – ALL ROUTINES WILL HAVE A SINGLE ENTRY AND A SINGLE EXIT.
- RULE 8 – ALL FUNCTIONS AND ROUTINES WILL BE COMMENTED.
- RULE 9 – STANDARD NAMING CONVENTIONS APPLY TO POINTERS, LABELS, MEMORY VALUES, ARRAYS...
- RULE 10 – IF YOU CAN'T REVIEW AND UNDERSTAND A ROUTINE IN FIVE MINUTES IT'S TOO COMPLEX.

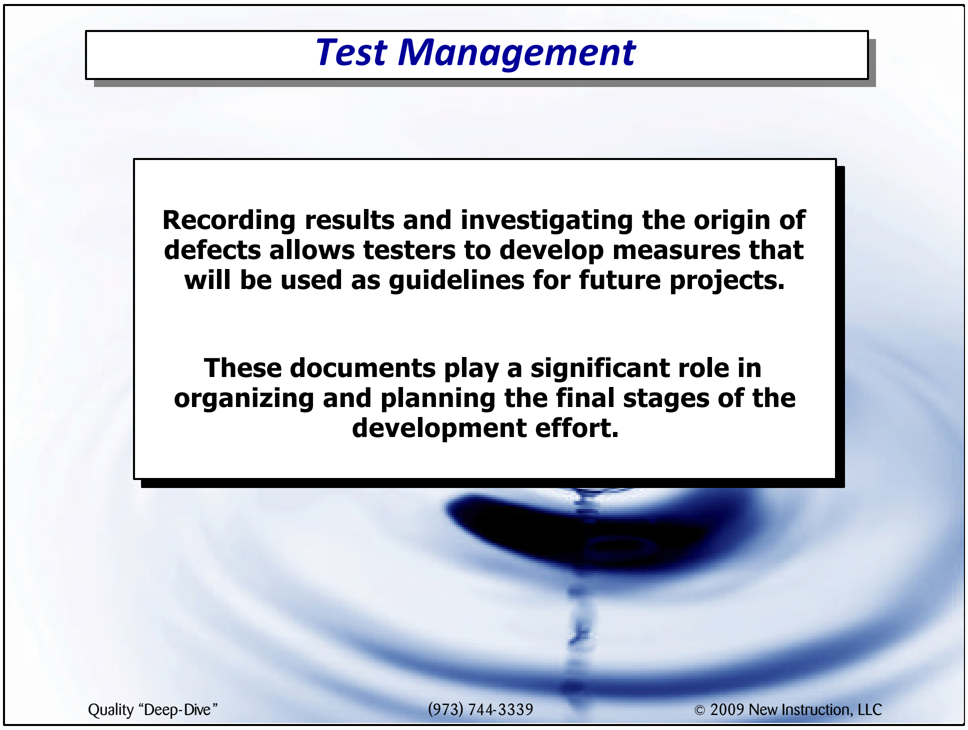
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



Test Management

Section 6

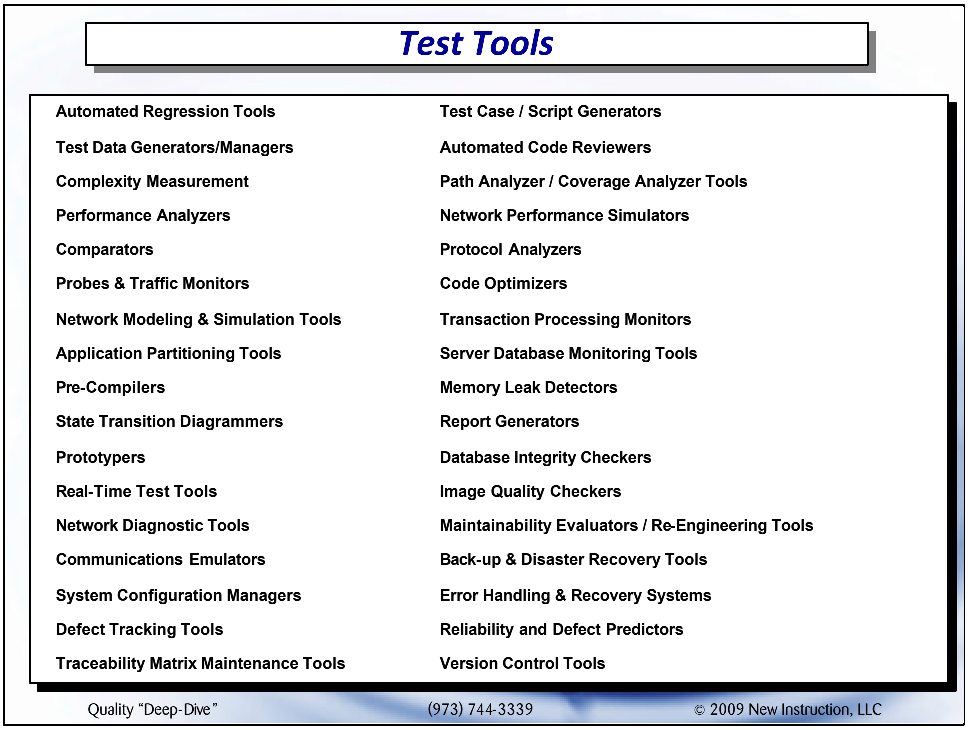
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



Test Management

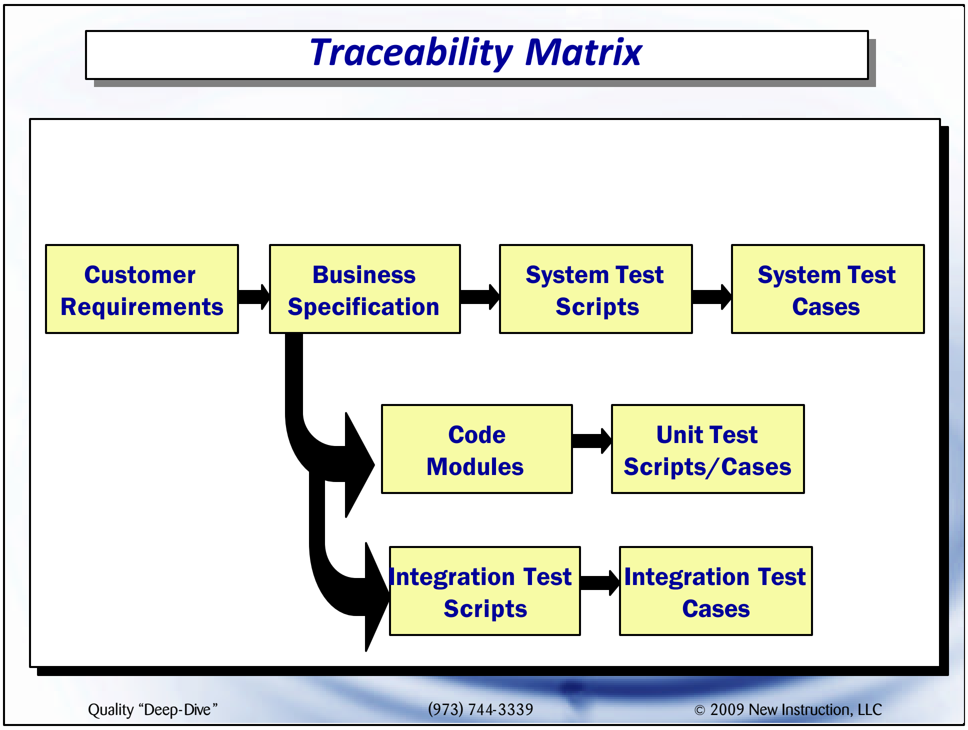
Recording results and investigating the origin of defects allows testers to develop measures that will be used as guidelines for future projects.

These documents play a significant role in organizing and planning the final stages of the development effort.



Test Tools

- Automated Regression Tools
- Test Data Generators/Managers
- Complexity Measurement
- Performance Analyzers
- Comparators
- Probes & Traffic Monitors
- Network Modeling & Simulation Tools
- Application Partitioning Tools
- Pre-Compilers
- State Transition Diagrammers
- Prototypers
- Real-Time Test Tools
- Network Diagnostic Tools
- Communications Emulators
- System Configuration Managers
- Defect Tracking Tools
- Traceability Matrix Maintenance Tools
- Test Case / Script Generators
- Automated Code Reviewers
- Path Analyzer / Coverage Analyzer Tools
- Network Performance Simulators
- Protocol Analyzers
- Code Optimizers
- Transaction Processing Monitors
- Server Database Monitoring Tools
- Memory Leak Detectors
- Report Generators
- Database Integrity Checkers
- Image Quality Checkers
- Maintainability Evaluators / Re-Engineering Tools
- Back-up & Disaster Recovery Tools
- Error Handling & Recovery Systems
- Reliability and Defect Predictors
- Version Control Tools



Traceability Matrix

Customer Requirements

Business Specification

System Test Scripts

System Test Cases

Code Modules

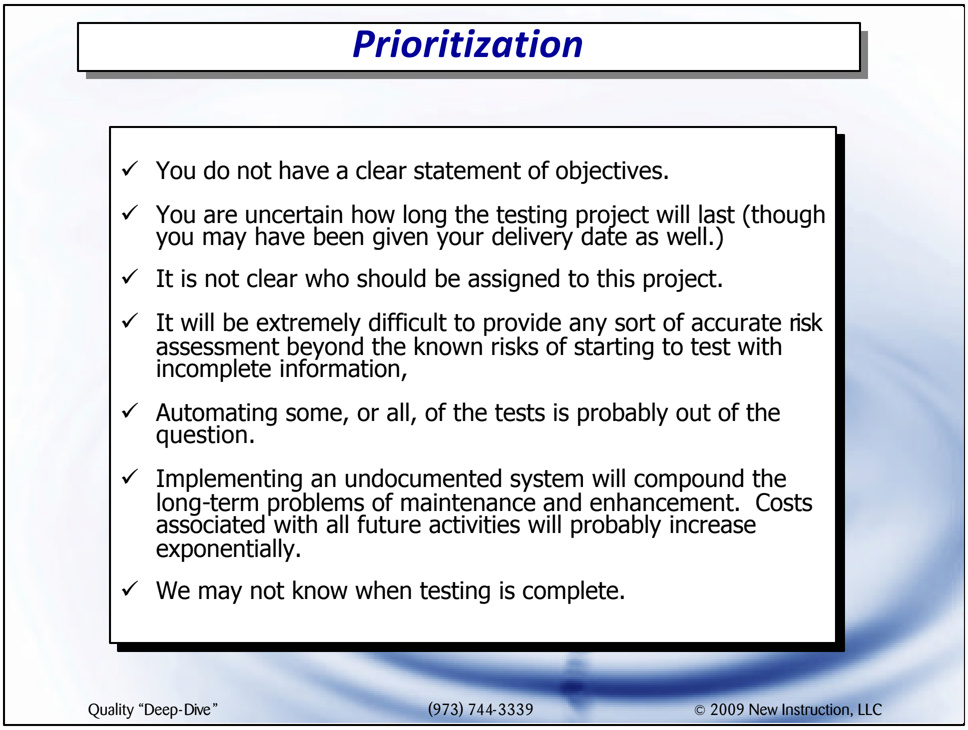
Unit Test Scripts/Cases

Integration Test Scripts

Integration Test Cases

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Allows measurement of test density – the number of tests associated with each requirement.

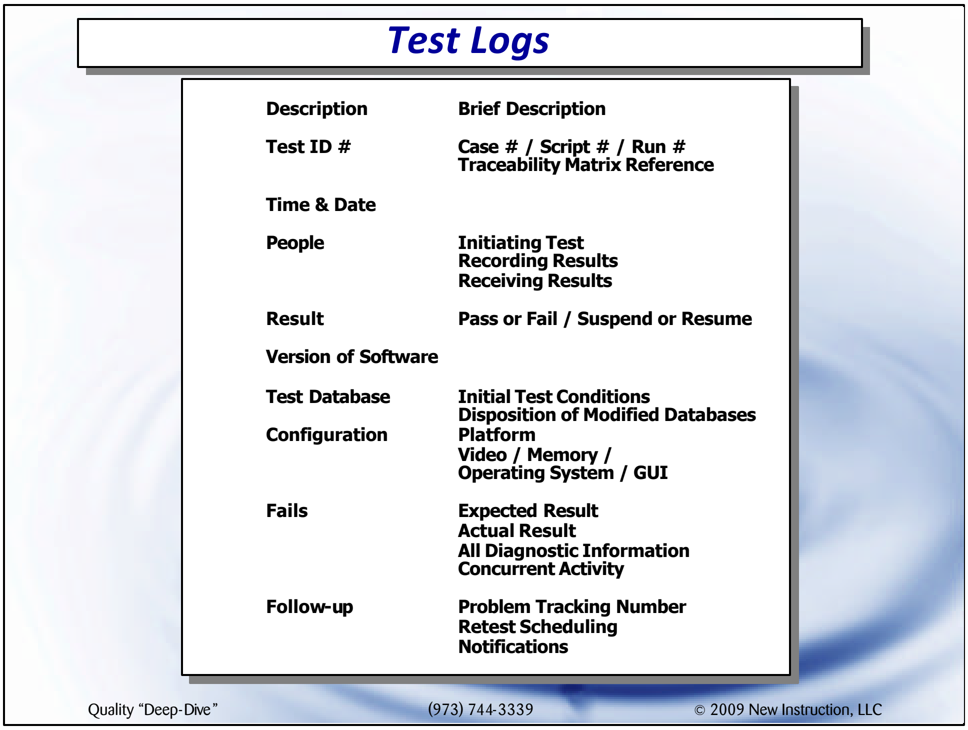


Prioritization

- ✓ You do not have a clear statement of objectives.
- ✓ You are uncertain how long the testing project will last (though you may have been given your delivery date as well.)
- ✓ It is not clear who should be assigned to this project.
- ✓ It will be extremely difficult to provide any sort of accurate risk assessment beyond the known risks of starting to test with incomplete information,
- ✓ Automating some, or all, of the tests is probably out of the question.
- ✓ Implementing an undocumented system will compound the long-term problems of maintenance and enhancement. Costs associated with all future activities will probably increase exponentially.
- ✓ We may not know when testing is complete.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

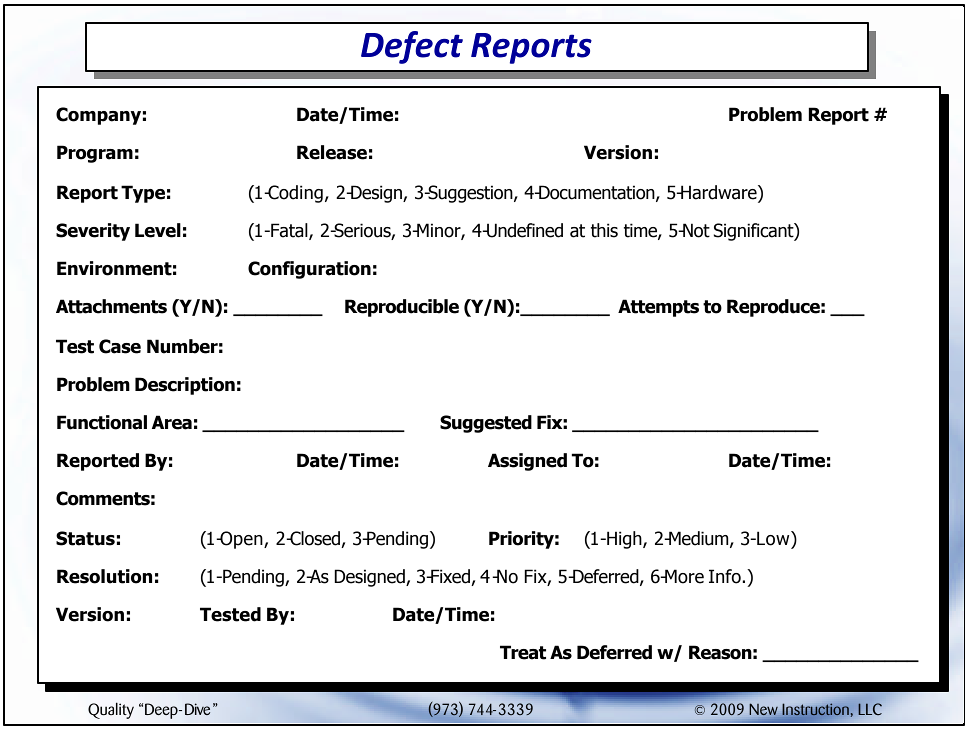
Tests to be executed must be prioritized when conditions exist limiting test coverage.



Test Logs

Description	Brief Description
Test ID #	Case # / Script # / Run # Traceability Matrix Reference
Time & Date	
People	Initiating Test Recording Results Receiving Results
Result	Pass or Fail / Suspend or Resume
Version of Software	
Test Database	Initial Test Conditions Disposition of Modified Databases
Configuration	Platform Video / Memory / Operating System / GUI
Fails	Expected Result Actual Result All Diagnostic Information Concurrent Activity
Follow-up	Problem Tracking Number Retest Scheduling Notifications

Documents test execution. Most organizations capture date / time / result only.



Defect Reports

Company: **Date/Time:** **Problem Report #**

Program: **Release:** **Version:**

Report Type: (1-Coding, 2-Design, 3-Suggestion, 4-Documentation, 5-Hardware)

Severity Level: (1-Fatal, 2-Serious, 3-Minor, 4-Undefined at this time, 5-Not Significant)

Environment: **Configuration:**

Attachments (Y/N): _________ **Reproducible (Y/N):**_________ **Attempts to Reproduce:** ___

Test Case Number:

Problem Description:

Functional Area: ____________________ **Suggested Fix:** ________________________

Reported By: **Date/Time:** **Assigned To:** **Date/Time:**

Comments:

Status: (1-Open, 2-Closed, 3-Pending) **Priority:** (1-High, 2-Medium, 3-Low)

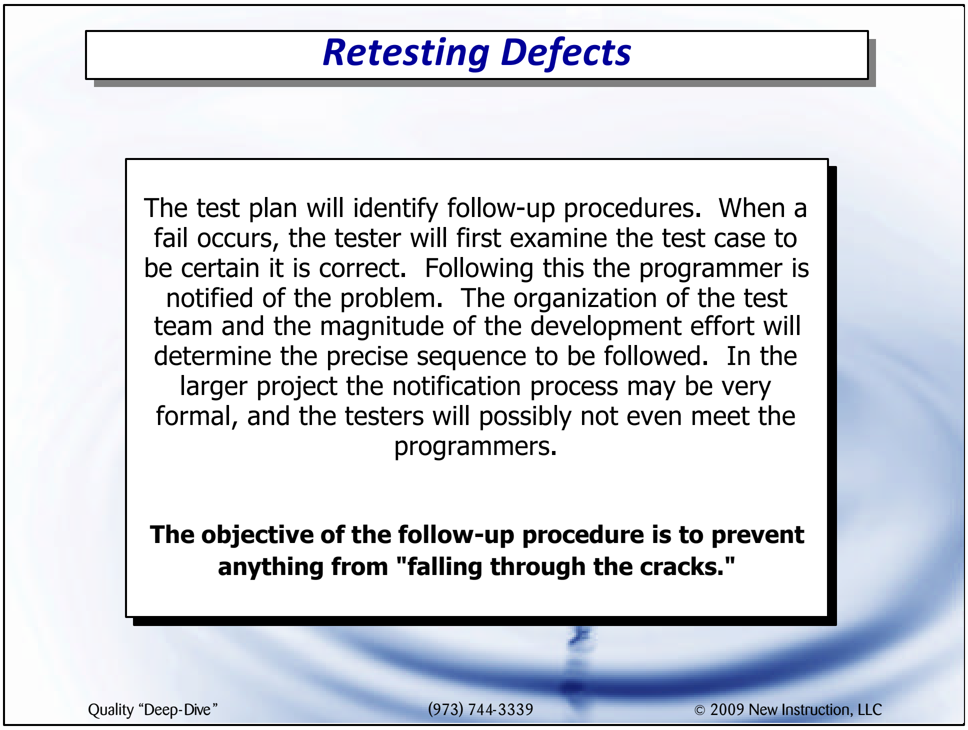
Resolution: (1-Pending, 2-As Designed, 3-Fixed, 4-No Fix, 5-Deferred, 6-More Info.)

Version: **Tested By:** **Date/Time:**

Treat As Deferred w/ Reason: _______________

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

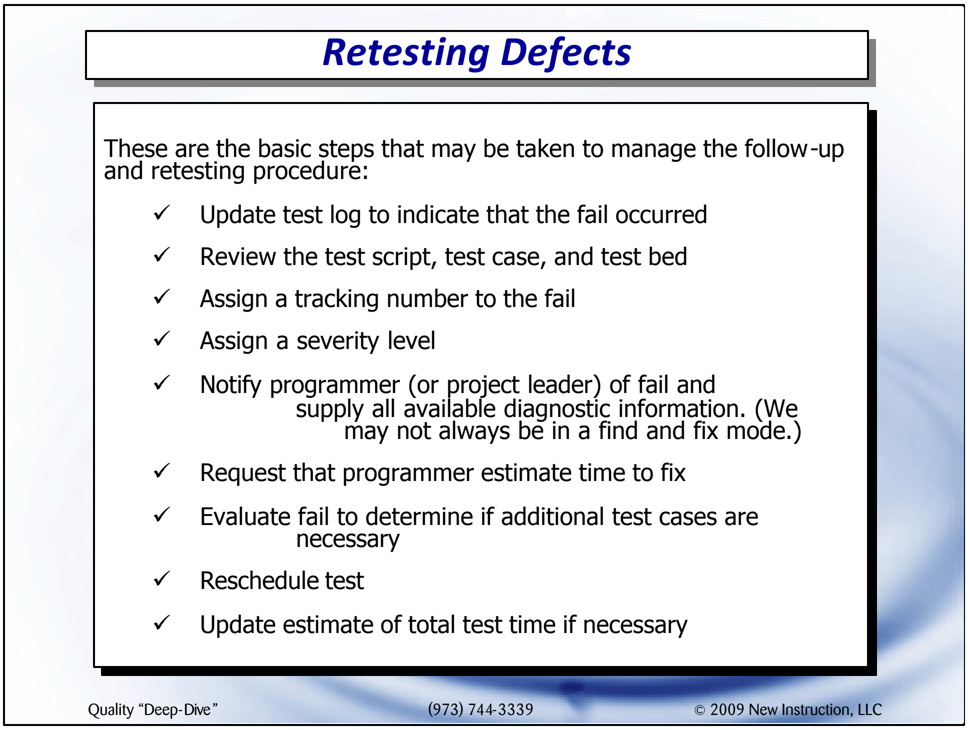
Reviewing report portfolio gives insight into defect sources – process / procedures that need to be improved.



Retesting Defects

The test plan will identify follow-up procedures. When a fail occurs, the tester will first examine the test case to be certain it is correct. Following this the programmer is notified of the problem. The organization of the test team and the magnitude of the development effort will determine the precise sequence to be followed. In the larger project the notification process may be very formal, and the testers will possibly not even meet the programmers.

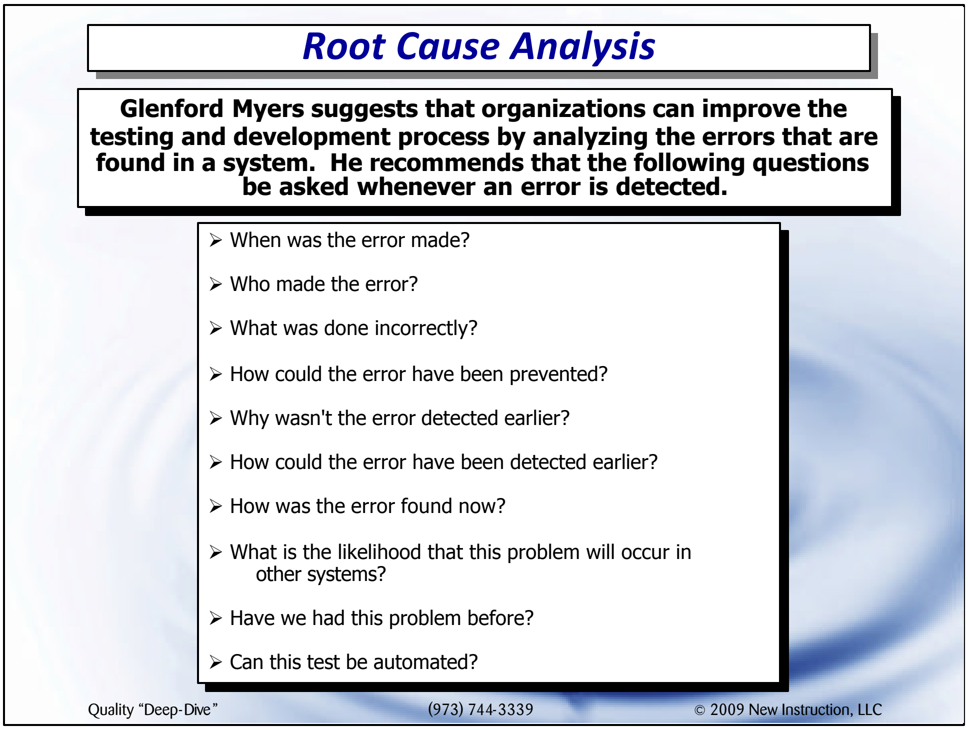
The objective of the follow-up procedure is to prevent anything from "falling through the cracks."



Retesting Defects

These are the basic steps that may be taken to manage the follow-up and retesting procedure:

- ✓ Update test log to indicate that the fail occurred
- ✓ Review the test script, test case, and test bed
- ✓ Assign a tracking number to the fail
- ✓ Assign a severity level
- ✓ Notify programmer (or project leader) of fail and supply all available diagnostic information. (We may not always be in a find and fix mode.)
- ✓ Request that programmer estimate time to fix
- ✓ Evaluate fail to determine if additional test cases are necessary
- ✓ Reschedule test
- ✓ Update estimate of total test time if necessary



Root Cause Analysis

Glenford Myers suggests that organizations can improve the testing and development process by analyzing the errors that are found in a system. He recommends that the following questions be asked whenever an error is detected.

- When was the error made?
- Who made the error?
- What was done incorrectly?
- How could the error have been prevented?
- Why wasn't the error detected earlier?
- How could the error have been detected earlier?
- How was the error found now?
- What is the likelihood that this problem will occur in other systems?
- Have we had this problem before?
- Can this test be automated?

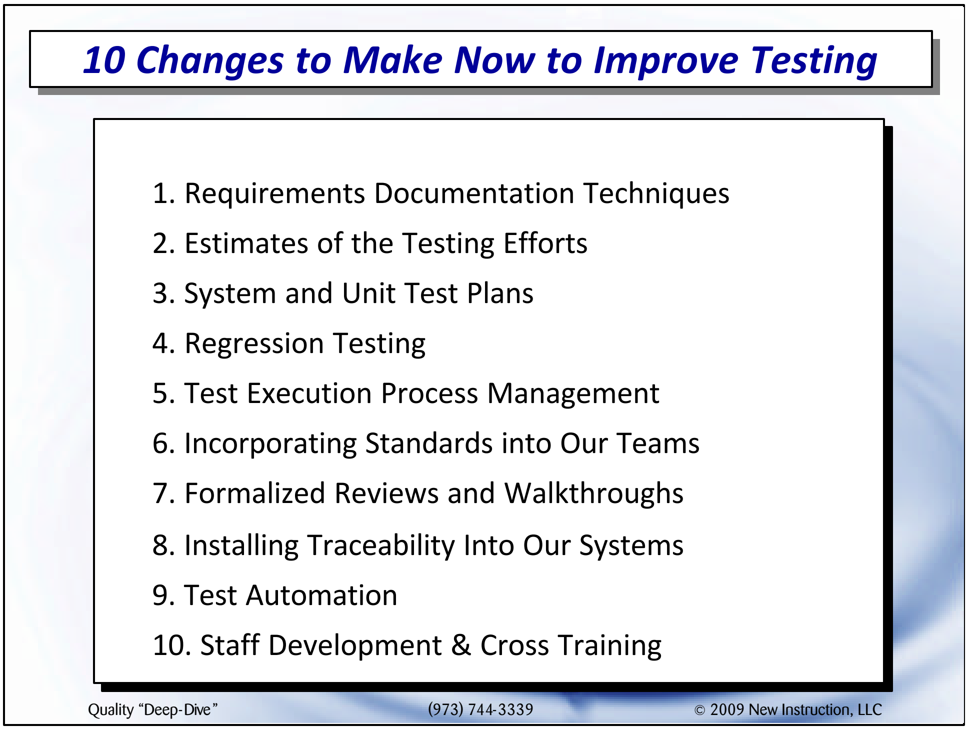
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

Key step in future defect prevention. Analysis is typically owned by the QA group (whether or not part of the ISQA group). This analysis becomes part of their process improvement efforts.

10 Changes to Make Now to Improve Testing

Section 7

Quality “Deep-Dive” (973) 744-3339 © 2009 New Instruction, LLC

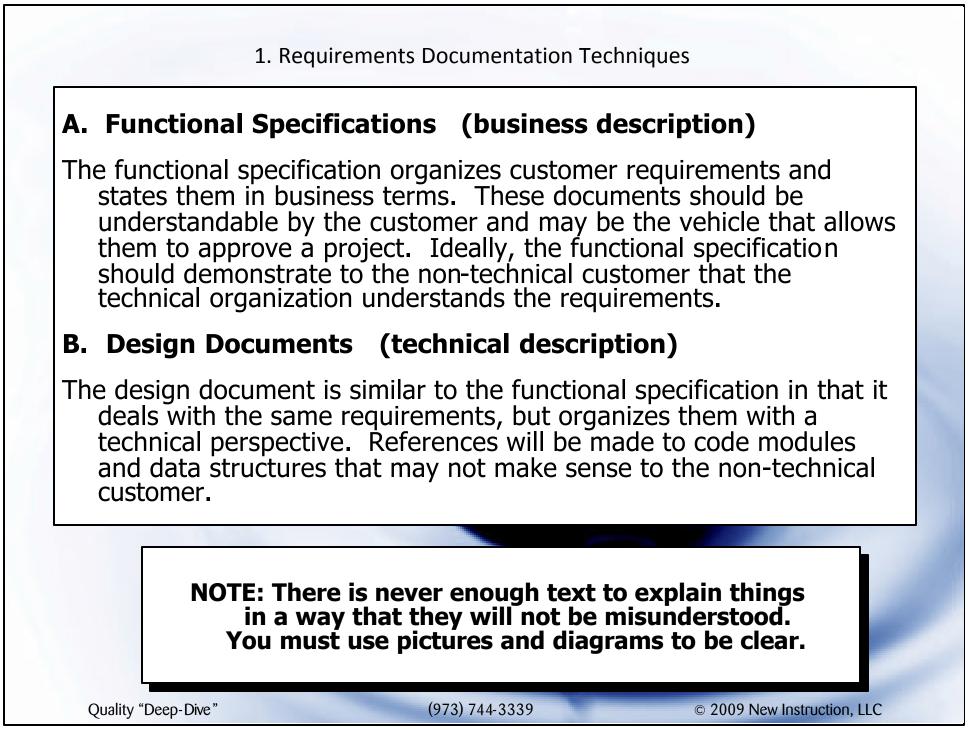


10 Changes to Make Now to Improve Testing

1. Requirements Documentation Techniques
2. Estimates of the Testing Efforts
3. System and Unit Test Plans
4. Regression Testing
5. Test Execution Process Management
6. Incorporating Standards into Our Teams
7. Formalized Reviews and Walkthroughs
8. Installing Traceability Into Our Systems
9. Test Automation
10. Staff Development & Cross Training

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

1. Requirements and Specifications should be documented separately. Requirements capture 'what' is needed / the system should do. Specifications define 'how' the function will be done. Diagrams / pictures / charts are extremely effective in conveying details.
2. Estimates of the testing effort – build estimates based on number of test scripts – based on number of spec pages.
3. Document system and unit test plans.
4. Conduct automated regression testing.
5. Have a Documented Test Execution Process - What will happen when a test fails? What is your process to work through to completion?
6. Incorporate standard processes and procedures in teams.
7. Formalize review and walkthrough processes – just the expectation that work will be reviewed improves quality. Managers are NOT included in the review process – because it could discourage the truth – people don't want to admit in front of a manager that there are problems.
8. Install traceability into systems – helps to keep focus, identifies tests that need to be run for changes to functionality.
9. Automate tests – Development team and test team must be in agreement in the testing tool selected. Total commitment to the tool is required – the tools are expensive, you need to be sure that it's the right / best tool for your situation before you buy.
10. Conduct staff development and cross-training.



1. Requirements Documentation Techniques

A. Functional Specifications (business description)

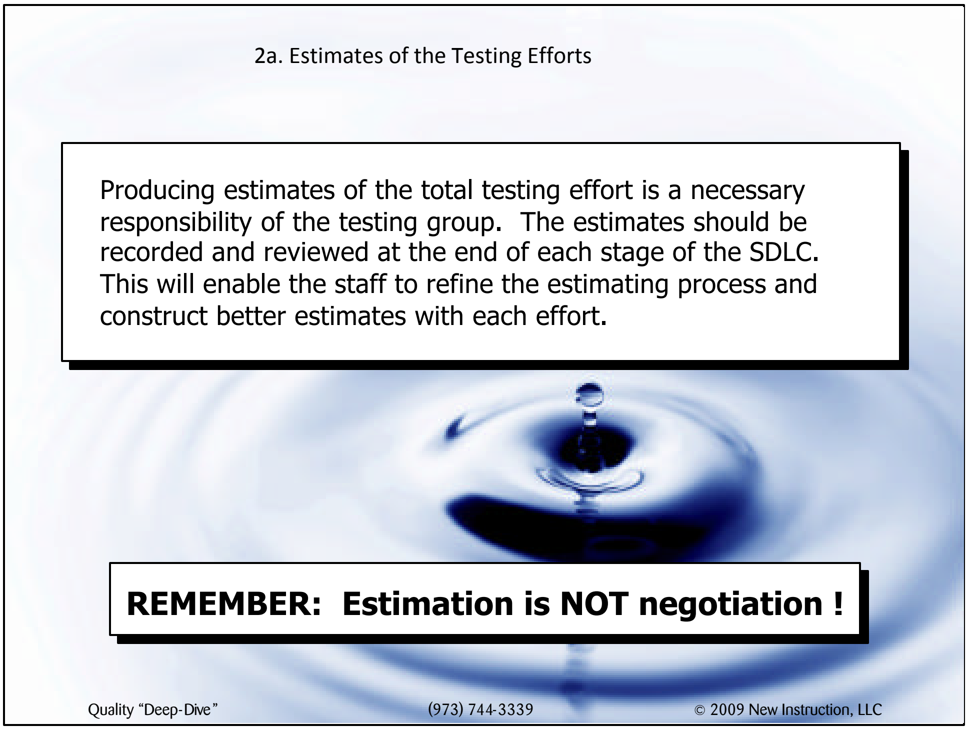
The functional specification organizes customer requirements and states them in business terms. These documents should be understandable by the customer and may be the vehicle that allows them to approve a project. Ideally, the functional specification should demonstrate to the non-technical customer that the technical organization understands the requirements.

B. Design Documents (technical description)

The design document is similar to the functional specification in that it deals with the same requirements, but organizes them with a technical perspective. References will be made to code modules and data structures that may not make sense to the non-technical customer.

NOTE: There is never enough text to explain things in a way that they will not be misunderstood. You must use pictures and diagrams to be clear.

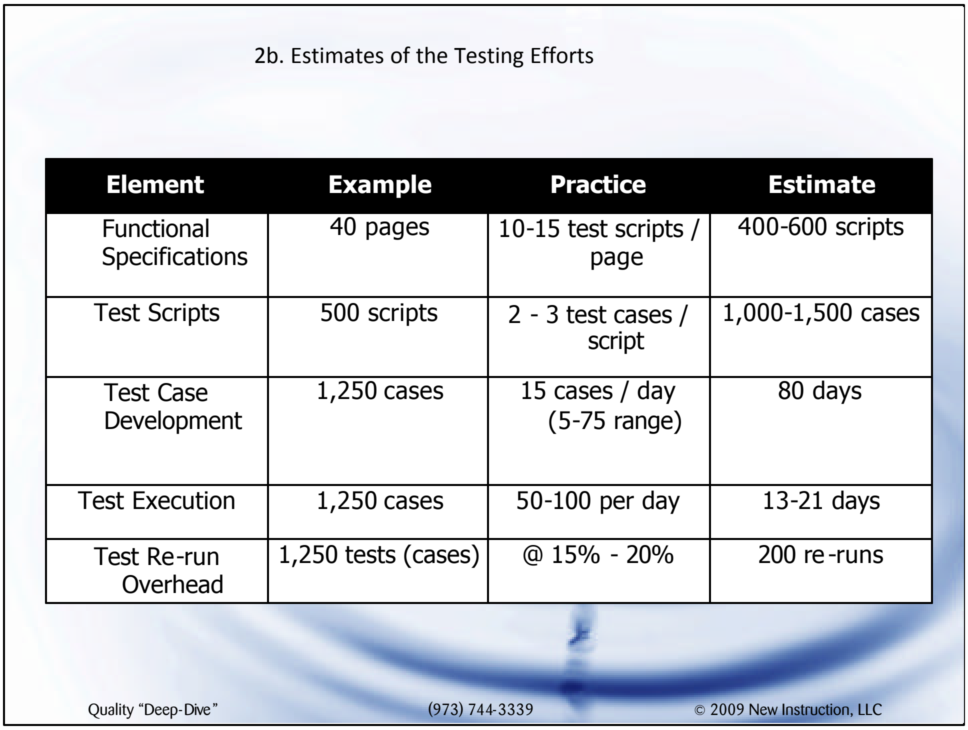
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



2a. Estimates of the Testing Efforts

Producing estimates of the total testing effort is a necessary responsibility of the testing group. The estimates should be recorded and reviewed at the end of each stage of the SDLC. This will enable the staff to refine the estimating process and construct better estimates with each effort.

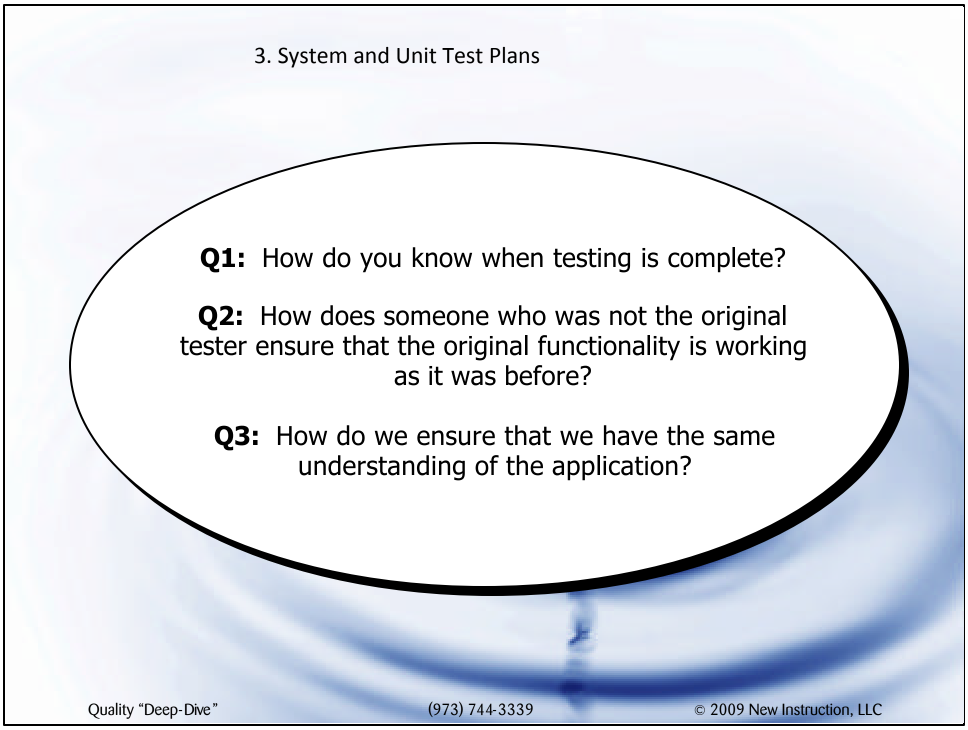
REMEMBER: Estimation is NOT negotiation !



2b. Estimates of the Testing Efforts

Element	Example	Practice	Estimate
Functional Specifications	40 pages	10-15 test scripts / page	400-600 scripts
Test Scripts	500 scripts	2 - 3 test cases / script	1,000-1,500 cases
Test Case Development	1,250 cases	15 cases / day (5-75 range)	80 days
Test Execution	1,250 cases	50-100 per day	13-21 days
Test Re-run Overhead	1,250 tests (cases)	@ 15% - 20%	200 re-runs

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



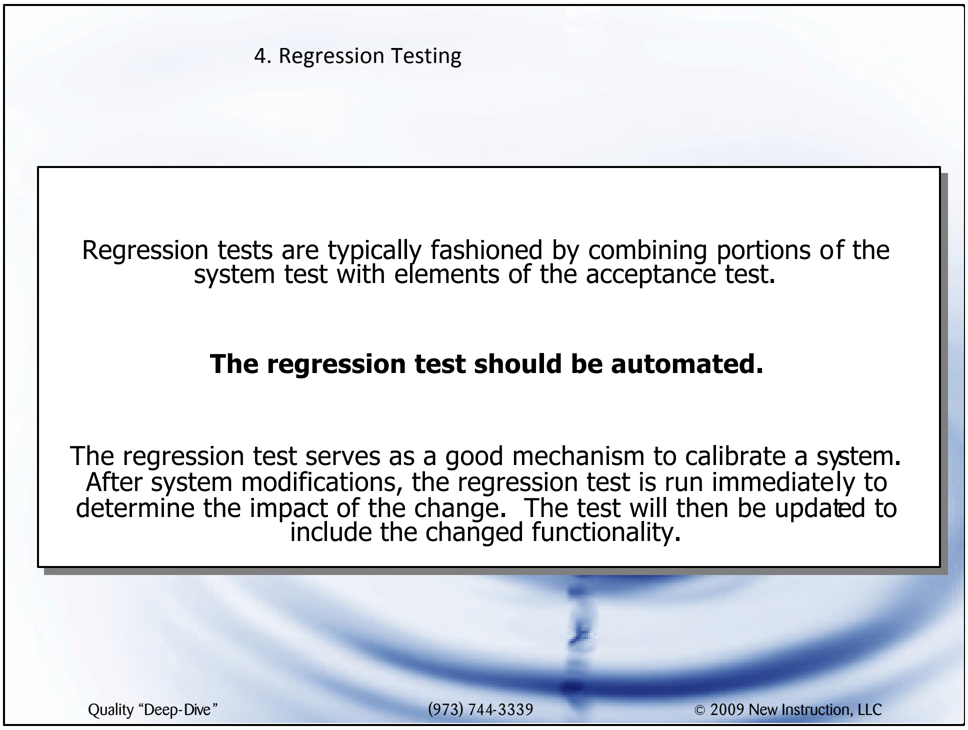
3. System and Unit Test Plans

Q1: How do you know when testing is complete?

Q2: How does someone who was not the original tester ensure that the original functionality is working as it was before?

Q3: How do we ensure that we have the same understanding of the application?

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

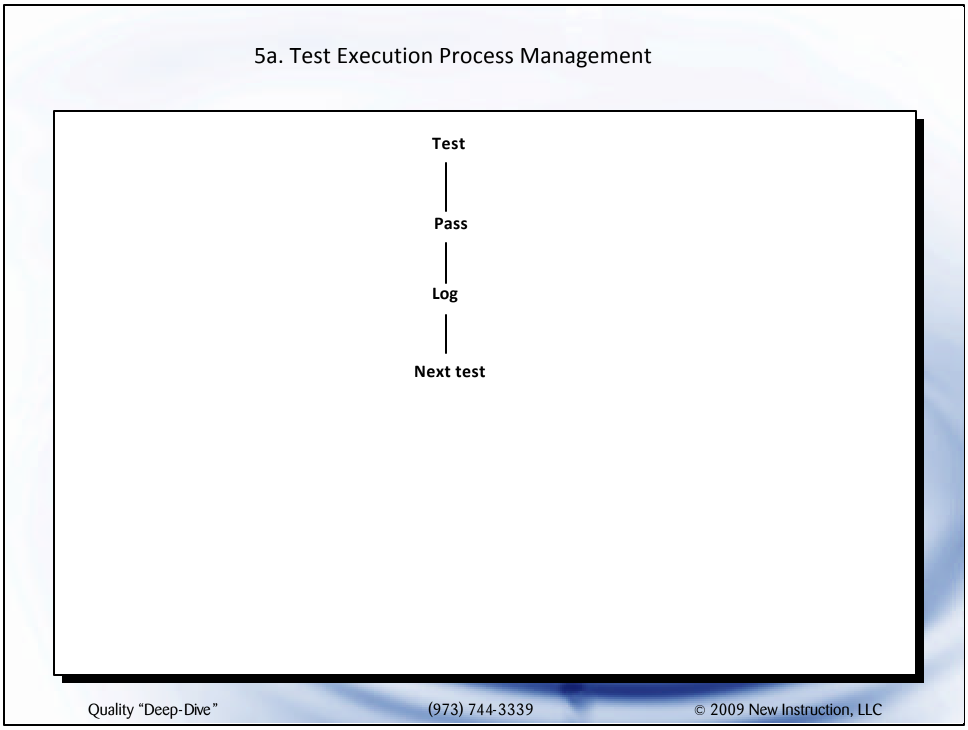


4. Regression Testing

Regression tests are typically fashioned by combining portions of the system test with elements of the acceptance test.

The regression test should be automated.

The regression test serves as a good mechanism to calibrate a system. After system modifications, the regression test is run immediately to determine the impact of the change. The test will then be updated to include the changed functionality.



5a. Test Execution Process Management

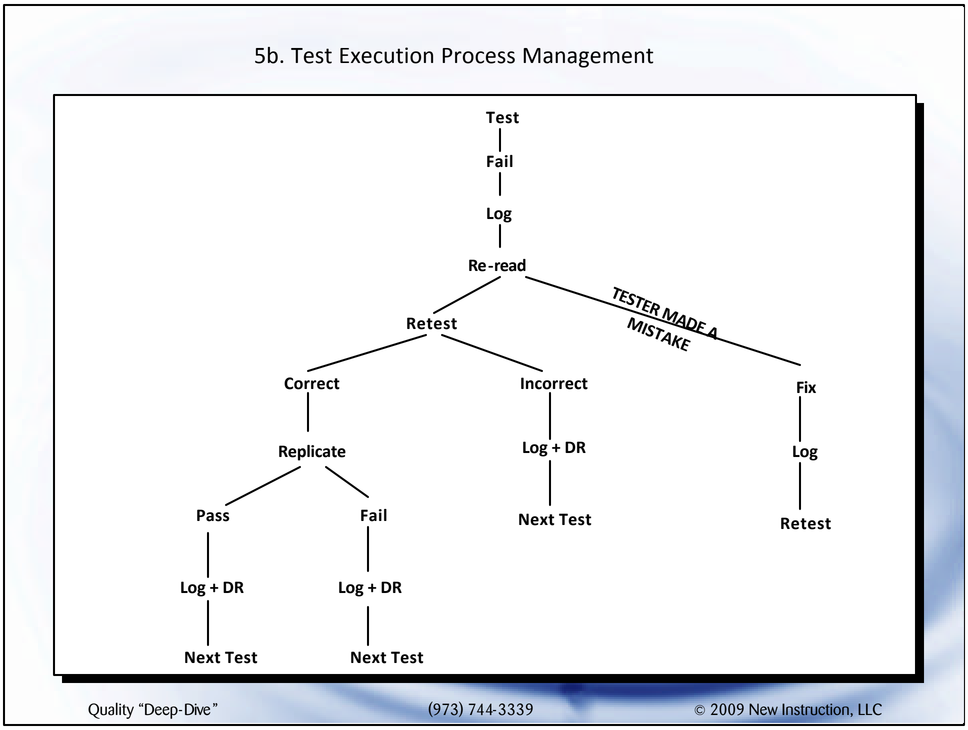
Test

Pass

Log

Next test

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



5b. Test Execution Process Management

Test

Fail

Log

Re-read

Retest

TESTER MADE A MISTAKE

Correct

Incorrect

Fix

Replicate

Log + DR

Log

Pass

Fail

Next Test

Retest

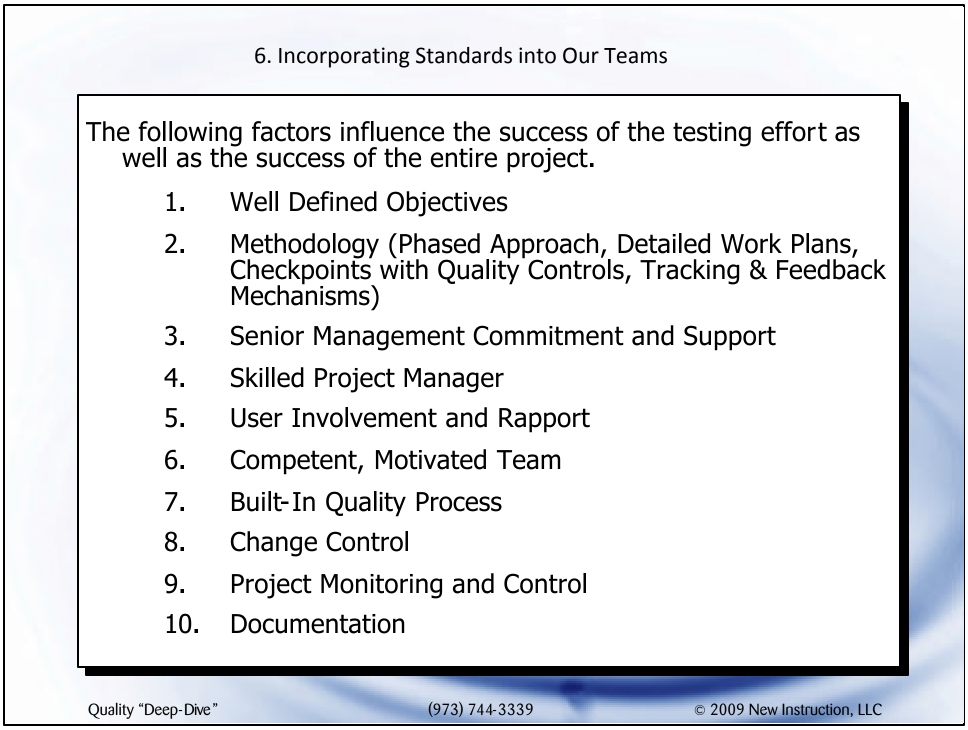
Log + DR

Log + DR

Next Test

Next Test

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

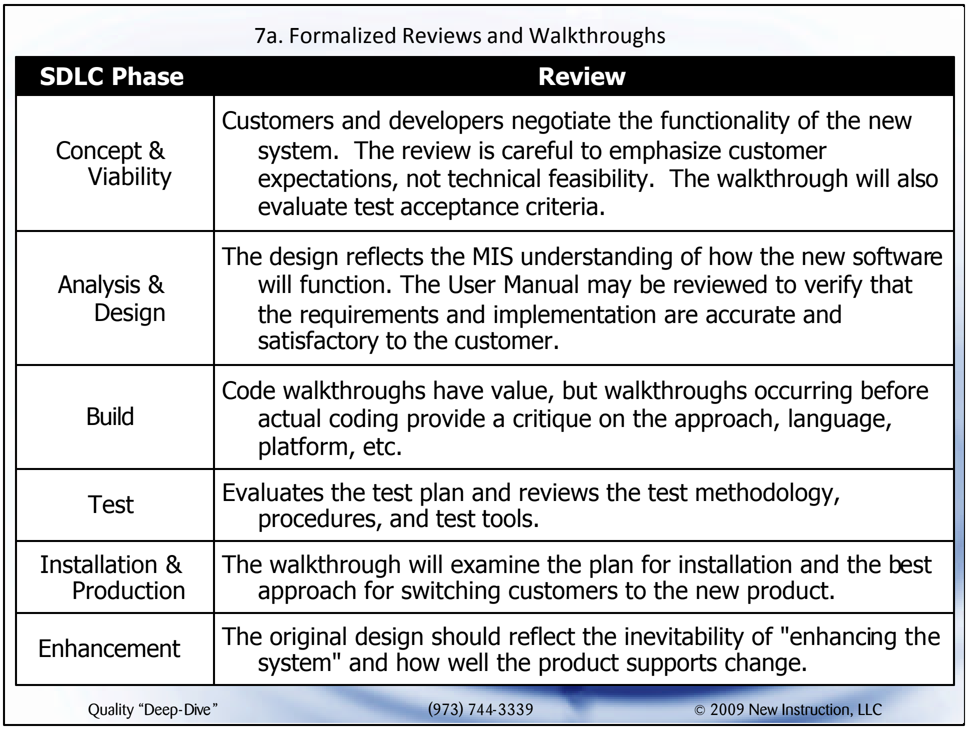


6. Incorporating Standards into Our Teams

The following factors influence the success of the testing effort as well as the success of the entire project.

1. Well Defined Objectives
2. Methodology (Phased Approach, Detailed Work Plans, Checkpoints with Quality Controls, Tracking & Feedback Mechanisms)
3. Senior Management Commitment and Support
4. Skilled Project Manager
5. User Involvement and Rapport
6. Competent, Motivated Team
7. Built-In Quality Process
8. Change Control
9. Project Monitoring and Control
10. Documentation

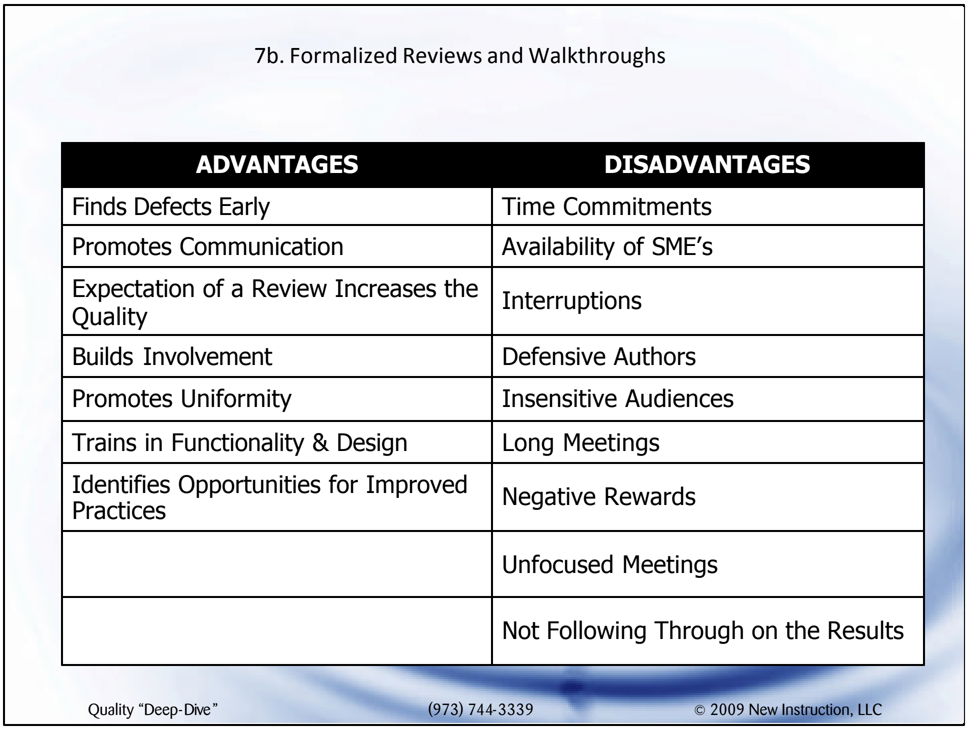
Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



7a. Formalized Reviews and Walkthroughs

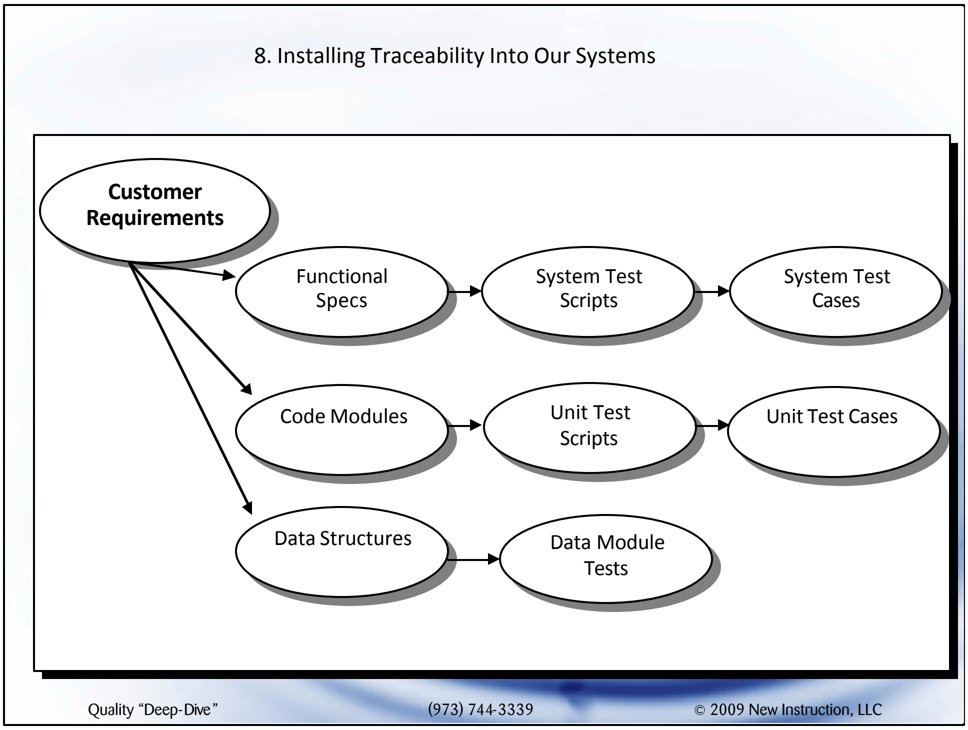
SDLC Phase	Review
Concept & Viability	Customers and developers negotiate the functionality of the new system. The review is careful to emphasize customer expectations, not technical feasibility. The walkthrough will also evaluate test acceptance criteria.
Analysis & Design	The design reflects the MIS understanding of how the new software will function. The User Manual may be reviewed to verify that the requirements and implementation are accurate and satisfactory to the customer.
Build	Code walkthroughs have value, but walkthroughs occurring before actual coding provide a critique on the approach, language, platform, etc.
Test	Evaluates the test plan and reviews the test methodology, procedures, and test tools.
Installation & Production	The walkthrough will examine the plan for installation and the best approach for switching customers to the new product.
Enhancement	The original design should reflect the inevitability of "enhancing the system" and how well the product supports change.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



7b. Formalized Reviews and Walkthroughs

ADVANTAGES	DISADVANTAGES
Finds Defects Early	Time Commitments
Promotes Communication	Availability of SME’s
Expectation of a Review Increases the Quality	Interruptions
Builds Involvement	Defensive Authors
Promotes Uniformity	Insensitive Audiences
Trains in Functionality & Design	Long Meetings
Identifies Opportunities for Improved Practices	Negative Rewards
	Unfocused Meetings
	Not Following Through on the Results



8. Installing Traceability Into Our Systems

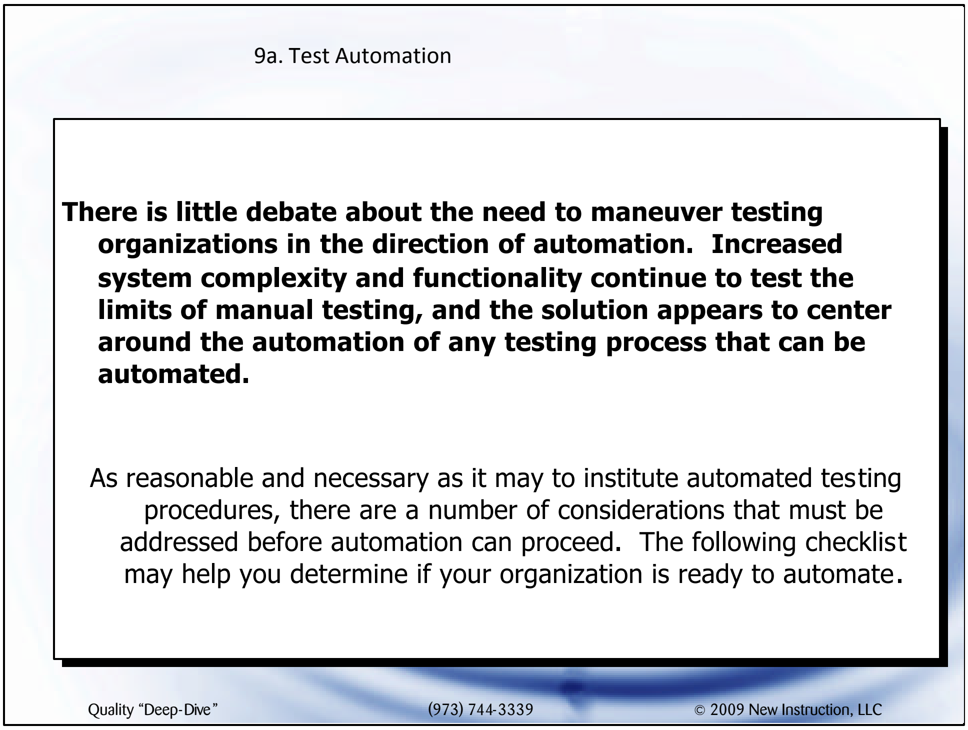
Customer Requirements

Functional Specs → System Test Scripts → System Test Cases

Code Modules → Unit Test Scripts → Unit Test Cases

Data Structures → Data Module Tests

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC

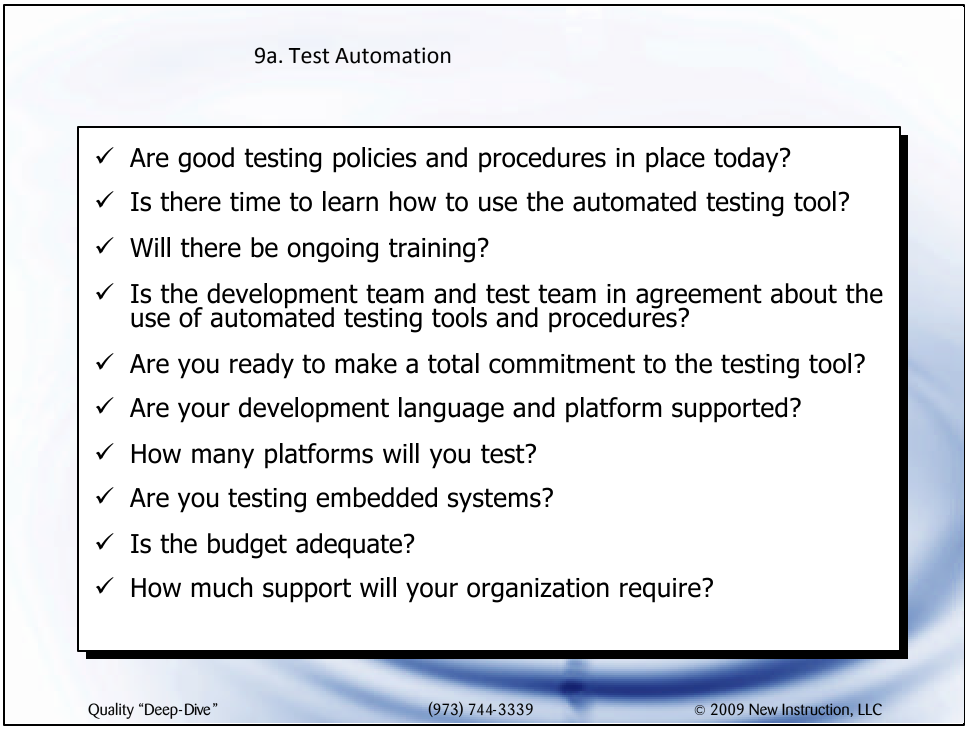


9a. Test Automation

There is little debate about the need to maneuver testing organizations in the direction of automation. Increased system complexity and functionality continue to test the limits of manual testing, and the solution appears to center around the automation of any testing process that can be automated.

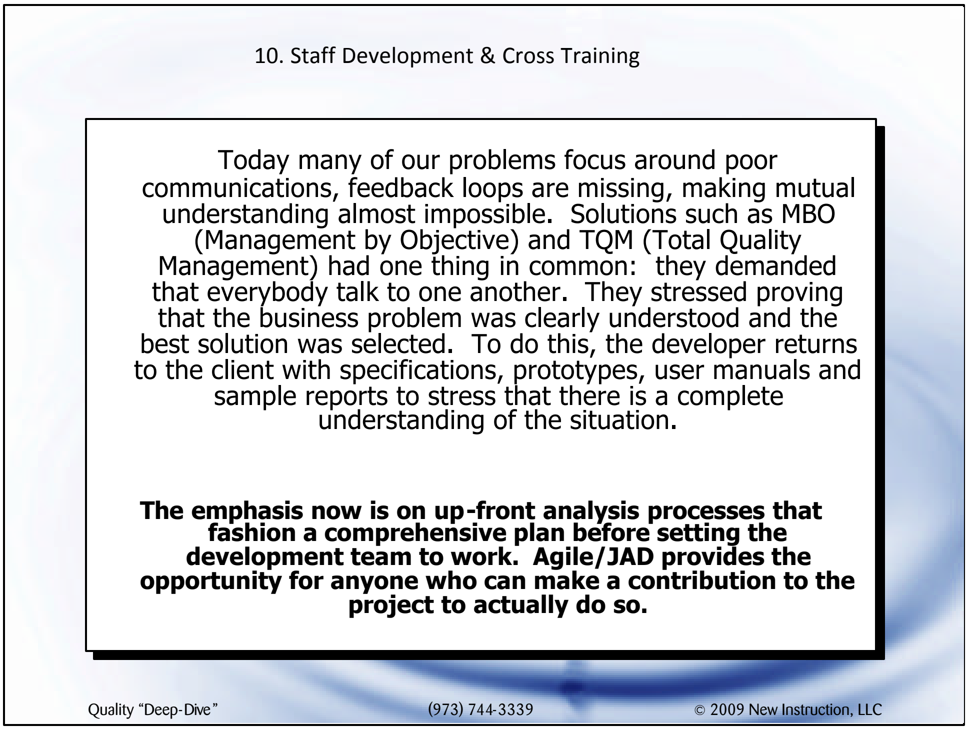
As reasonable and necessary as it may to institute automated testing procedures, there are a number of considerations that must be addressed before automation can proceed. The following checklist may help you determine if your organization is ready to automate.

Quality "Deep-Dive" (973) 744-3339 © 2009 New Instruction, LLC



9a. Test Automation

- ✓ Are good testing policies and procedures in place today?
- ✓ Is there time to learn how to use the automated testing tool?
- ✓ Will there be ongoing training?
- ✓ Is the development team and test team in agreement about the use of automated testing tools and procedures?
- ✓ Are you ready to make a total commitment to the testing tool?
- ✓ Are your development language and platform supported?
- ✓ How many platforms will you test?
- ✓ Are you testing embedded systems?
- ✓ Is the budget adequate?
- ✓ How much support will your organization require?



10. Staff Development & Cross Training

Today many of our problems focus around poor communications, feedback loops are missing, making mutual understanding almost impossible. Solutions such as MBO (Management by Objective) and TQM (Total Quality Management) had one thing in common: they demanded that everybody talk to one another. They stressed proving that the business problem was clearly understood and the best solution was selected. To do this, the developer returns to the client with specifications, prototypes, user manuals and sample reports to stress that there is a complete understanding of the situation.

The emphasis now is on up-front analysis processes that fashion a comprehensive plan before setting the development team to work. Agile/JAD provides the opportunity for anyone who can make a contribution to the project to actually do so.