Test Execution

- Developing Test Cases from Requirements and Use Cases
- Running Dynamic Tests and Documenting Results
- Evaluating Results – Defect Identification What's a problem? What's not?
- Reporting Results - Working with Development on Fixes
- Debugging and Re-Testing
- Test Status Report

Dynamic Testing

- **Exercising the software code on a computer system to see if it performs according to the specifications and requirements**
  - **Will the software perform accurately, reliably and consistently when released for business use?**
- **Discover and repair as many defects as possible prior to release**

> **"A successful test is one that finds an error."**
>
> - Glenford Myers

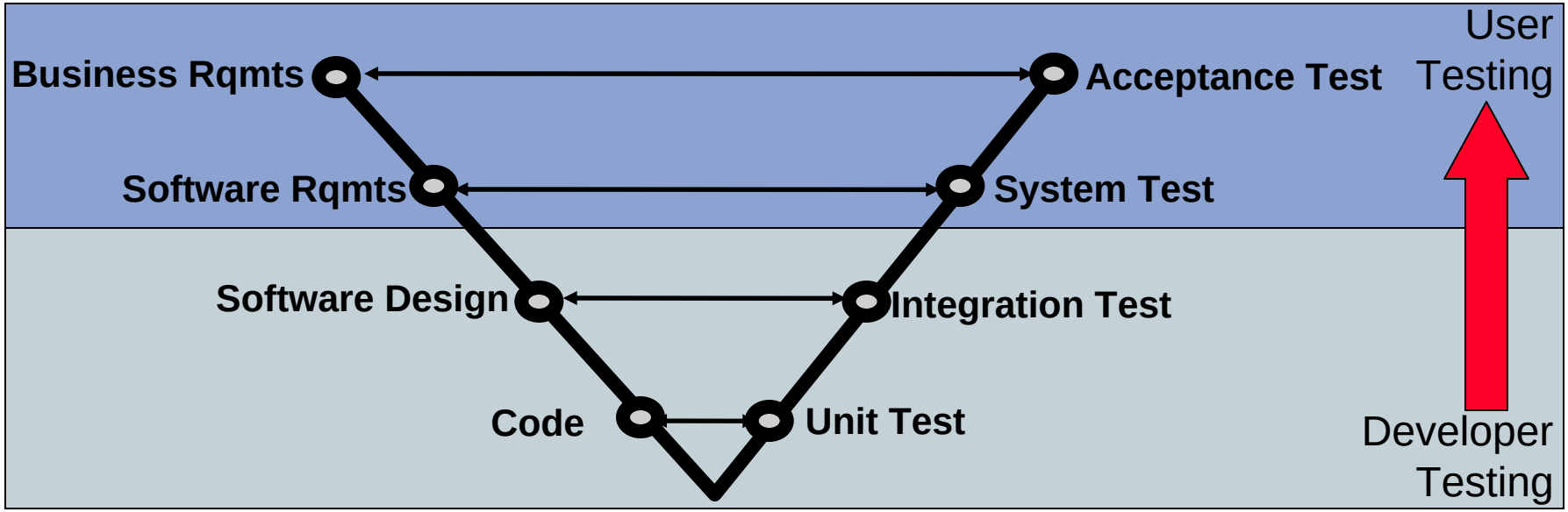
Dynamic Testing

- Quality cannot be tested-in
- No amount of testing can prove that a defect does not exist
- Testing is expensive – balance the cost with the returns
- Test 'smart' – consider:
  - Complexity
  - Risk
  - Expected Usage

> "Testing cannot show us the absence of defects…
>
> it can only show us that defects are present."
>
> - Robert Pressman

The V-Shaped Model for V&V

- **Emphasis on verification and validation 'as you go'**



- **Developers and other IT staff will perform many tests as the software is developed**
  - **Unit testing is done during code development**
  - **Integration testing is done as components are assembled according to the design**

Developer Tests

- White box testing based on knowledge of the internal logic of the code
- Unit testing is the smallest scale of testing - done on the module level before the user interface is developed
- Tests that analyze the code statements, branches, paths and conditions for:
  - Technical accuracy
  - Standards
  - Data integrity and structure
  - Logic
  - Compatibility
  - Internal communications and connections
- Users may want to be informed about the results of previous testing efforts in terms of:
  - Complexity
  - Previous 'hot' spots

System Tests

- Black-box testing is done through the user interface without regard to the internal working of the software
- End-to-end testing once all components are assembled and placed in a executable environment (configuration and compatibility tests)
- Performance or stress testing – tests the performance at peak loads
- Installation testing – tests the process for installing, upgrading or uninstalling the software
- Recovery testing – tests software behavior during an unexpected interruption and it's recovery for such an event
- Security testing – how well the software protects against unauthorized use and threats

System Test Cases

- Are designed to find conditions that could cause the software to fail or produce unexpected results
- Test both functional and non-functional requirements
- Some system test cases should be completed before involving users – to shake out all the obvious defects - sometimes known as sanity or smoke testing.
- Includes both positive and negative test cases
  - Positive – does the software do what it's supposed to do?
  - Negative – does the software not do what it's not supposed to do?

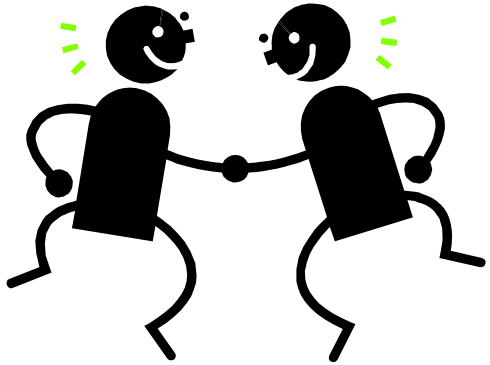
User Testing

- System Testing – generally done by IT or QA testers but users may be involved
- Usability Testing – includes users performing normal tasks using the software and usability specialists recording observations

  Note: Users should review the results of system and usability testing
- Acceptance Testing – is done by the users but may also be performed by an independent test group. May include:
  - Alpha testing – in a test environment
  - Beta – in the user's environment
- Parallel Testing – running the new software simultaneously with the existing software

Combined System/Acceptance Testing

- To achieve schedule compression
- May improve communication between development staff and users
- Beneficial when users have little or no experience testing
- Users should be treated as 'partners' in the process. Participate in and approve test plan, test cases and test results



- User's should avoid 'rush to judgment'. Defects are expected during testing, the full-picture will not be known until later in the process

Usability Tests

- Subjective tests designed to measure user friendliness from the user's point of view
  - Understandability
  - Learn-ability
  - Communicativeness
- Techniques
  - Interviews
  - Surveys
  - Observations
- May test any aspect of use including
  - Ergonomics/comfort
  - Screen design
  - ADA accommodation

Acceptance Test Cases

- **Designed to test conditions by which users can determine if the software meets the requirements**
- **At least one test case per user requirement**
- **One use case = multiple test cases**
- **More positive test cases or 'Happy Path' tests than negative test cases**
- **Based on user interactions with the system as specified in use cases, user scenarios, and user requirements**

> *"There is only one rule in designing test cases:*
>
> *cover all features, but do not make too many test cases"*
>
> -Tsuneo Yamaura

Acceptance Test Cases – how many?

- Enough to confirm the software can be implemented
- Prioritized requirements should drive testing
  - Test the riskiest areas of the software fully
  - Test the features or functions that are the key elements of the business process
  - Idenfity the most complex requirements
  - Find out where the problem spots are
- Ensuring coverage using traceability matrix
  - Run tests on all functions (requirement sets or use cases)

What does a test case look like?

- Test Case Identifier
- Test Case Name
- Test Objective
- Test Item - what's being tested
- Set up instructions are the preconditions as specified in the use case
- Data input requirements – the values and corresponding fields for entry, tolerances, value ranges, etc.
- Procedural requirements the exact sequence of steps to be followed
- Expected results are the post conditions as specified in the use case

Test Case Example

Test Case 4.3.3 **V. 1.0** **Tester:**

Display Maternity Patient Summary **Test Date:**

Objective: Test the function that allows the user to search, select and display maternity patient summaries using patient name. Summary includes patient name, ID, address/phone, age, number of previous pregnancies, live births, pregnancy status, due date or delivery date, applicable diagnosis codes and Rx's. Verify that the data is displayed as expected.

Test Description: Test will use data extracted from the current database. The results of searches will be displayed on the new user interface and will include up to 10 diagnosis codes and up to 20 Rx summaries.

Test Conditions: The database will be loaded with all patient records as of 12/31/2005. All database records have been validated and are accurate. Records must have several patients with same last name.

Test Case Example

Steps:

1)Enter a valid last and first name from the list, select search

1.1 Validate information displayed against report

2)Enter a name not on the list, select search

2.1 Validate message "error – patient not found"

3)Enter a valid last name from the list that appears multiple times, select search

3.1 Validate patient's listed are all the patients with the last name

3.2 Validate that no patients are included that should not be

4)Select patients from the list

4.1 Validate that information displayed to printed report

…

Post Conditions: No data will be changed or modified by this test.

Expected Results: A paper report from the current system as of 12/31/2005 is attached for verification.

Test Case Result: Pass _____ Fail_____

Finding Additional Test Cases

- **Create additional test cases for alternate courses**
- **Plan some guerrilla test cases based on the nastiest, most unusual, business situations that can be thought up. The test may include:**
  - **extreme boundary tests,**
  - **strange error conditions,**
  - **unusual values in complex algorithms and computations,**
  - **non-standard process flow**
- **Exploratory testing: Allow different testers to test the use case, generating tests on the fly**

Traceability Matrix

Use Case		Test Case ID				
ID	Req ID	UTC-1	UTC-2	UTC-4	UTC-5	UTC-6
UC-1	R1	P				
UC-1	R2	P				
UC-1	R3	P				
UC-1	ER-1		P			
UC-2	R1			F		
UC-2	R2-1				I	
UC-2	R2-2				I	
UC-3	R1					X
UC-3	R2					X

P=Pass	I=Incomplete
F=Fail	X=Not tested

Before Testing

- Have all requirements been reviewed and approved including changes?
- Have test criteria for requirements/use cases been developed?
- Has the test plan been developed, reviewed and approved including acceptance criteria?
- Are test cases written, traceability matrix complete? Have they been reviewed and approved?
- Are resources available and appropriately trained?
- Has the test environment been established including test data?
- Have supporting test tools been established for recording and reporting test results?

Documenting Test Case Results

- Evaluate test results
  - Pass – the test results match the expected results
  - Fail – the test results do not match the expected results
- When a test fails:
  - Be sure it's the software and not the test design at fault
  - Review the data inputs are correct
  - Rerun the test, if necessary
- Document results by recording
  - When the test took place
  - Pass/Fail

Managing Test Documentation

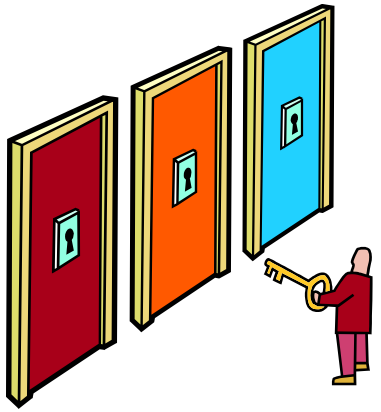
- Careful documentation standards including version control may be necessary on test cases to avoid repeating mistakes
- Establish a problem reporting system and process that records and tracks any event that requires investigation
- Keep a test log to record the status of all test cases and test cycles (hint: you can use the traceability matrix)
- Test status reports – regular summaries to management on testing activities, issues and metrics

Problem Reporting System

- Identify the test case, tester and test date
- Document actual results including screen prints and error messages
- Steps to reproduce the defect
- Tester's assessment of defect severity
  - Severe - Data loss, safety risk, or loss of major business functionality without workaround
  - Moderate – Some loss of business functionality with workaround
  - Minor - Cosmetic error
- Tester's assessment of fix priority
  1. Must fix before release
  2. May be okay to release before fixed
  3. Fix when possible

Confirm Testing Results

- **Review and analyze defect reports to confirm tester's assessment**
  - **Preliminary review by Acceptance Test Leader or Team**
  - **Finalize the assessment**
  - **Defects logged**
  - **High priority items move forward for fix**



Prioritizing Fixes and Changes

- High priority fixes are reviewed by the developer
  - Debug
  - Assess work needed to correct
  - Update defect report
- May be done by a team – sometimes called Control Board that includes:
  - Acceptance Test Leader
  - Project Managers
  - Project Sponsor
  - Lead Developer
- Control Board reviews and approves fixes based on severity, cost, impact on release date

Quality Test Reporting System

- **Benefits of well-documented test results**
  - **Reduces the time spent on re-writing and re-testing**
  - **Improves tester and developer relationship and satisfaction**
  - **Maintains team / tester credibility**
  - **Allows testers and developers to concentrate on testing and fixing**
  - **Expedites / improves status reporting**

Guide for Test Reporting

- Structure: test carefully
- Reproduce: test it again
- Isolate: test it differently
- Generalize: test it elsewhere
- Compare: review similar test results
- Summarize: relate test to intended business use
- Condense: trim unnecessary information
- Disambiguate: use clear words
- Neutralize: be fair and impartial
- Review: make sure the report is complete

Working with Developers on Defects

- **Testing is a destructive process that provides critical feedback to developers**
- **Be hard on the product but considerate of the developer**
- **Report the findings, provide as much supporting information as possible**
- **Be open to working directly with the developer to improve the developers understanding of the problem and the fix**

> ***"Given enough eyeballs, all bugs are shallow (e.g., given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone)."***
>
> -E. Raymond

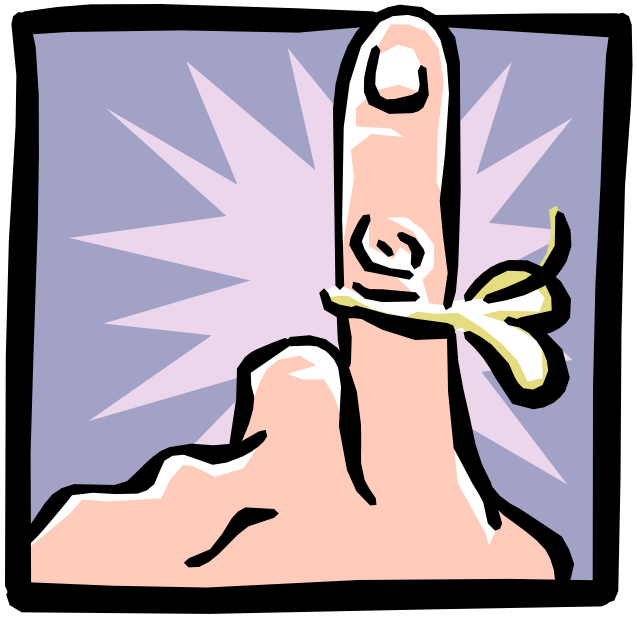
Debugging

- Process initiated when a defect is found
  - To discover the source of the problem
  - To repair the problem
- The problem may be in the software code or in the test case
- Disagreements may occur between users and developers during debugging

Debugging

- Common causes of disagreements:
  - Confusion over unclear or poorly specified requirements
  - Requirements volatility - lots of customer changes
  - Complex environments – distributed environments and changes in configurations, operating systems, communications, platforms, etc.
  - Human error in code or test case
  - Gold plating or 'feature-itis' overcomplicating the software with a fancy solution when a simple solution would do just as well.

Don't Forget Regression Tests



- Re-execution of a subset of tests that have already passed
- Ensures recent fixes to the software have not created unintended problems elsewhere
- Increasing issues with regression tests may indicate a stability problem

Retesting

- Once the debugging is done and the problem is repaired
  - Developers retest the code
  - New code is moved into the test environment – sometimes called a build
  - Testers re-run test case or cases where the defect was found
  - May re-run test cases with related functionality
- Update test documentation to reflect results

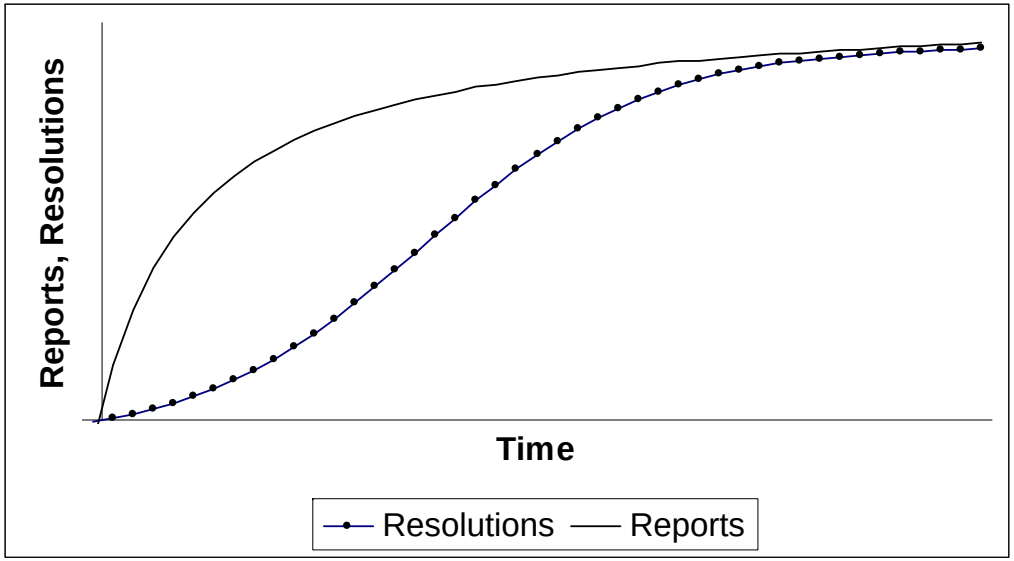
Good Enough Testing
Knowing when to stop testing…

- When all of the most likely uses of the system have been successfully tested
- When the tests on highest priority requirements, features, or functions pass
- When regression tests are completing without serious or unknown defects
- When the risks of not releasing outweigh the risks of not resolving the remaining defects
- Consider visibility, safety impact, financial impact, user impact

Test smart, not hard!

Knowing when your done testing

- **The Pareto Principle – 80% of the errors come from 20% of the software components**
- **Law of diminishing returns – when it costs more to find the remaining bugs than it's worth**
- **When the defect discovery rate and the defect resolution rate converge**



Note: Graph the cumulative reports and resolutions

Next...

Software Acceptance...

And Maintenance!