Quality Assurance: Test Development & Execution

Ian S. King
Test Development Lead
Windows CE Base OS Team
Microsoft Corporation

Developing Test Strategy

Elements of Test Strategy

- Test specification
- Test plan
- Test harness/architecture
- Test case generation
- Test schedule

Where is your focus?

- The customer
- The customer
- The customer
- The customer
- The customer
- The customer
- The customer
- Schedule and budget

Requirements feed into test design

- What factors are important to the customer?
  - Reliability vs. security
  - Reliability vs. performance
  - Features vs. reliability
  - Cost vs. ?
- What are the customer's expectations?
- How will the customer use the software?

Test Specifications

- What questions do I want to answer about this code? Think of this as experiment design
- In what dimensions will I ask these questions?
  - Functionality
  - Security
  - Reliability
  - Performance
  - Scalability
  - Manageability

Test specification: goals

- Design issues
  - Do you understand the design and goals?
  - Is the design logically consistent?
  - Is the design testable?
- Implementation issues
  - Is the implementation logically consistent?
  - Have you addressed potential defects arising from implementation?

Test specification: example

- CreateFile method
  - Should return valid, unique handle for
    - initial ‘open’ for appropriate resource
    - subsequent calls for shareable resource
    - for files, should create file if it doesn’t exist
  - Should return NULL handle and set error indicator if resource is
    - nonexistent device
    - inappropriate for ‘open’ action
    - in use and not shareable
    - unavailable because of error condition (e.g. no disk space)
  - Must recognize valid forms of resource name
    - Filename, device, ?

Methods of delivering software

- Enterprise/data center
  - Traditional: hardware vendor was software vendor
  - Support usually explicit and structured
- Embedded systems
  - Software is shipped as built-in component
  - Often doesn’t “look like” computing technology
- “Shrink wrap”
  - Software is often installed by end user
  - Goal: minimal involvement post-sale
- Online ‘update’ - subscription
  - Minimal user involvement – goal is transparency

Challenges: Enterprise/Data Center

- Usually requires 24x7 availability
- Full system test may be prohibitively expensive – a second data center?
- Management is a priority
  - Predictive data to avoid failure
  - Diagnostic data to quickly diagnose failure
  - Rollback/restart to recover from failure

Challenges: Embedded Systems

- Software may be “hardwired” (e.g. mask ROM)
- End user is not prepared for upgrade scenarios
  - Field service or product return may be necessary
- End user does not see hardware vs. software
- End user may not see software at all
  - Who wrote your fuel injection software?

Challenges: Shrink Wrap Software

- Software compatibility matrix
  - Operating systems
  - Dependencies (expected and unexpected)
  - Conflicts with other software
- Hardware configuration issues
  - Dependencies (expected and unexpected)
  - Resource conflicts
- Completely unrelated weirdness
- N.B.: there’s no one “on the ground”

Trimming the matrix: risk analysis in test design

- It’s a combinatorial impossibility to test it all
  - Example: eight modules that can be combined
    - One hour per test of each combination
    - Twenty person-years (40 hr weeks, 2 wks vacation)
- Evaluate test areas and prioritize based on:
  - Customer priorities
  - Estimated customer impact
  - Cost of test
  - Cost of potential field service

Test Plans

- How will I ask my questions? Think of this as the “Methods” section
- Understand domain and range
- Establish equivalence classes
- Address domain classes
  - Valid cases
  - Invalid cases
  - Boundary conditions
  - Error conditions
  - Fault tolerance/stress/performance

Test plan: goals

- Enables development of tests
- Proof of testability – if you can’t design it, you can’t do it
- Review: what did you miss?

Test plan: example

- CreateFile method
  - Valid cases
    - execute for each resource supporting ‘open’ action
      - opening existing device
      - opening existing file
      - opening (creating) nonexistent file
    - execute for each such resource that supports sharing
      - multiple method calls in separate threads/processes
      - multiple method calls in single thread/process
  - Invalid cases
    - nonexistent device
    - file path does not exist
    - in use and not shareable
  - Error cases
    - insufficient disk space
    - invalid form of name
    - permissions violation
  - Boundary cases
    - e.g. execute to/past system limit on open device handles
    - device name at/past name length limit (MAXPATH)
  - Fault tolerance
    - execute on failed/corrupted filesystem
    - execute on failed but present device

Performance testing

- Test for performance behavior
  - Does it meet requirements?
    - Customer requirements
    - Definitional requirements (e.g. Ethernet)
- Test for resource utilization
  - Understand resource requirements
- Test performance early
  - Avoid costly redesign to meet performance requirements

Security Testing

- Is data/access safe from those who should not have it?
- Is data/access available to those who should have it?
- How is privilege granted/revoked?
- Is the system safe from unauthorized control?
  - Example: denial of service
- Collateral data that compromises security
  - Example: network topology

Stress testing

- Working stress: sustained operation at or near maximum capability
- Goal: resource leak detection
- Breaking stress: operation beyond expected maximum capability
- Goal: understand failure scenario(s)
  - "Failing safe" vs. unrecoverable failure or data loss

Globalization

- Localization
  - UI in the customer's language
  - German overruns the buffers
  - Japanese tests extended character sets
- Globalization
  - Data in the customer's language
  - Non-US values ($ vs. Euro, ips vs. cgs)
  - Mars Global Surveyor: mixed metric and SAE

Test Cases

- Actual "how to" for individual tests
- Expected results
- One level deeper than the Test Plan
- Automated or manual?
- Environmental/platform variables

Test case: example

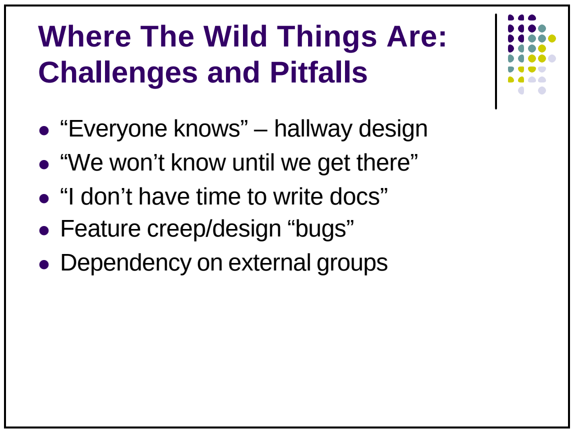
- CreateFile method
  - Valid cases
    - English
      - open existing disk file with arbitrary name and full path, file permissions allowing access
        - create directory 'c:\foo'
        - copy file 'bar' to directory 'c:\foo' from test server; permissions are 'Everyone: full access'
        - execute CreateFile('c:foo\bar', etc.)
        - expected: non-null handle returned

Test Harness/Architecture

- Test automation is nearly <u>always</u> worth the time and expense
- How to automate?
  - Commercial harnesses
  - Roll-your-own (TUX)
  - Record/replay tools
  - Scripted harness
- Logging/Evaluation

Test Schedule

- Phases of testing
  - Unit testing (may be done by developers)
  - Component testing
  - Integration testing
  - System testing
- Dependencies – when are features ready?
  - Use of stubs and harnesses
- When are tests ready?
  - Automation requires lead time
- The long pole – how long does a test pass take?



Where The Wild Things Are: Challenges and Pitfalls

- “Everyone knows” – hallway design
- “We won’t know until we get there”
- “I don’t have time to write docs”
- Feature creep/design “bugs”
- Dependency on external groups