

Quality Assurance: Test Development & Execution

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



Introduction: Ian King

- Manager of Test Development for Windows CE Base OS (kernel, drivers, file systems)
- Previous projects at Microsoft:
 - MSN 1.x online service, Site Server 3.0, TransPoint online service, Speech API 5.0
- Previously: business analyst, Pacific Telecom



Implementing Testing



What makes a good tester?

- Analytical
 - Ask the right questions
 - Develop experiments to get answers
- Methodical
 - Follow experimental procedures precisely
 - Document observed behaviors, their precursors and environment
- Brutally honest
 - You can't argue with the data



How do test engineers fail?



- Desire to “make it work”
 - Impartial judge, not “handyman”
- Trust in opinion or expertise
 - Trust no one – the truth (data) is in there
- Failure to follow defined test procedure
 - How did we get here?
- Failure to document the data
- Failure to believe the data

Testability



- Can all of the feature’s code paths be exercised through APIs, events/messages, etc.?
 - Unreachable internal states
- Can the feature’s behavior be programmatically verified?
- Is the feature too complex to test?
 - Consider configurations, locales, etc.
- Can the feature be tested timely with available resources?
 - Long test latency = late discovery of faults

Test Categories



- Functional
 - Does it work?
- Performance
 - How fast/big/high/etc.?
- Security
- Usability
- Stress
 - Working stress
 - Breaking stress – how does it fail?
- Reliability/Availability

Manual Testing



- Definition: test that requires direct human intervention with SUT
- Necessary when:
 - GUI is tested element
 - Behavior is premised on physical activity (e.g. card insertion)
- Advisable when:
 - Automation is more complex than SUT
 - SUT is changing rapidly (early development)

Automated Testing



- Good: replaces manual testing
- Better: performs tests difficult for manual testing (e.g. timing related issues)
- Best: enables other types of testing (regression, perf, stress, lifetime)
- Risks:
 - Time investment to write automated tests
 - Tests may need to change when features change

Types of Automation Tools: Record/Playback



- Record “proper” run through test procedure (inputs and outputs)
- Play back inputs, compare outputs with recorded values
- Advantage: requires little expertise
- Disadvantage: little flexibility - easily invalidated by product change
- Disadvantage: update requires manual involvement

Types of Automation Tools: Scripted Record/Playback



- Fundamentally same as simple record/playback
- Record of inputs/outputs during manual test input is converted to script
- Advantage: existing tests can be maintained as programs
- Disadvantage: requires more expertise
- Disadvantage: fundamental changes can ripple through MANY scripts

Types of Automation Tools: Script Harness



- Tests are programmed as modules, then run by harness
- Harness provides control and reporting
- Advantage: tests can be very flexible
- Advantage: tests can exercise features similar to customers’ code
- Disadvantage: requires considerable expertise and abstract process

Types of Automation Tools: Verb-Based Scripting



- Module is programmed to invoke product behavior at low level – associated with ‘verb’
- Tests are designed using defined set of verbs
- Advantage: great flexibility
- Advantage: changes are usually localized to a given verb
- Disadvantage: requires considerable expertise and high-level abstract process

Test Corpus



- Body of data that generates known results
- Can be obtained from
 - Real world – demonstrates customer experience
 - Test generator – more deterministic
- Caveats
 - Bias in data generation?
 - Don't share test corpus with developers!

Instrumented Code: Test Hooks



- Code that enables non-invasive testing
- Code remains in shipping product
- May be enabled through
 - Special API
 - Special argument or argument value
 - Registry value or environment variable
- Example: Windows CE IOCTLs
- Risk: silly customers....

Instrumented Code: Diagnostic Compilers



- Creates ‘instrumented’ SUT for testing
 - Profiling – where does the time go?
 - Code coverage – what code was touched?
 - Really evaluates testing, NOT code quality
 - Syntax/coding style – discover bad coding
 - lint, the original syntax checker
 - Complexity
 - Very esoteric, often disputed (religiously)
 - Example: function point counting

Advanced Tools: Modeling



- Example: AsmL
 - Model behavior as set of states and transitions
 - Even multithreaded code is inherently serial
 - Stochastic elements can be explicit
- Advantage: test design before code is written
- Advantage: test the *test code*
- Disadvantage: creation and maintenance overhead

Instrumented platforms



- Example: App Verifier
 - Supports ‘shims’ to instrument standard system calls such as memory allocation
 - Tracks all activity, reports errors such as unreclaimed allocations, multiple frees, use of freed memory, etc.
- Win32 includes ‘hooks’ for platform instrumentation

Environment Management Tools



- Predictably simulate real-world situations
 - MemHog
 - DiskHog
 - CPU ‘eater’
 - Data Channel Simulator
- Reliably reproduce environment
 - Source control tools
 - Consistent build environment
 - Disk imaging tools

Test Monkeys



- Generate random input, watch for crash or hang
- Typically, ‘hooks’ UI through message queue
- Primarily catches “local minima” in state space (logic “dead ends”)
- Useless unless state at time of failure is well preserved!

Finding and Managing Bugs



What is a bug?



- Formally, a “software defect”
- SUT fails to perform to spec
- SUT causes something else to fail
- SUT functions, but does not satisfy usability criteria
- If the SUT works to spec and someone wants it changed, that’s a feature request

What do I do once I find one?



- Bug tracking is a valuable tool
 - Ensures the bug isn’t forgotten
 - Highlights recurring issues
 - Supports formal resolution/regression process
 - Provides important product cycle data
 - Can support ‘higher level’ metrics, e.g. root cause analysis
 - Valuable information for field support

What are the contents of a bug report?



- Repro steps – how did you cause the failure?
- Observed result – what did it do?
- Expected result – what should it have done?
- Collateral information: return values/output, debugger, etc.
- Environment
 - Test platforms must be reproducible
 - “It doesn’t do it on my machine”



Tracking Bugs

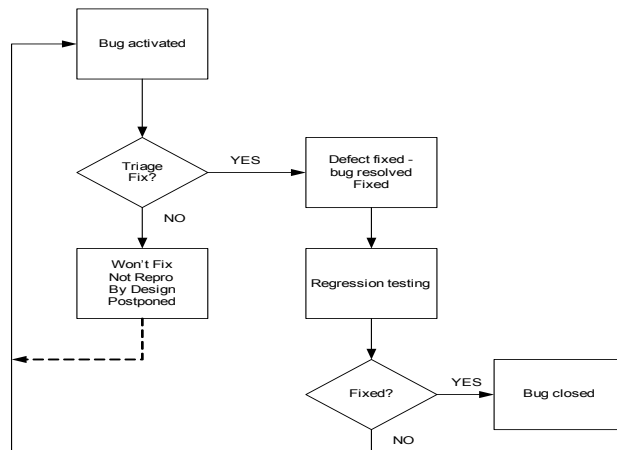
- Raw bug count
 - Slope is useful predictor
- Ratio by ranking
 - How bad are the bugs we're finding?
- Find rate vs. fix rate
 - One step forward, two back?
- Management choices
 - Load balancing
 - Review of development quality



Ranking bugs

- Severity
 - Sev 1: crash, hang, data loss
 - Sev 2: blocks feature, no workaround
 - Sev 3: blocks feature, workaround available
 - Sev 4: trivial (e.g. cosmetic)
- Priority
 - Pri 1: Fix immediately - blocking
 - Pri 2: Fix before next release outside team
 - Pri 3: Fix before ship
 - Pri 4: Fix if nothing better to do 😊

A Bug's Life



Regression Testing

- Good: rerun the test that failed
 - Or write a test for what you missed
- Better: rerun related tests (e.g. component level)
- Best: rerun all product tests
 - Automation can make this feasible!



To beta, or not to beta



- Quality bar for beta release: features mostly work if you use them right
- Pro:
 - Get early customer feedback on design
 - Real-world workflows find many important bugs
- Con:
 - Do you have time to incorporate beta feedback?
 - A beta release takes time and resources

Developer Preview



- Different quality bar than beta
 - Known defects, even crashing bugs
 - Known conflicts with previous version
 - Setup/uninstall not completed
- Goals
 - Review of feature set
 - Review of API set by technical consumers

Dogfood



- “So good, we eat it ourselves”
- Advantage: real world use patterns
- Disadvantage: impact on productivity
- At Microsoft: we model our customers
 - 50K employees
 - Broad range of work assignments, software savvy
 - Wide ranging network (worldwide)

When can I ship?



- Test coverage is “sufficient”
- Bug slope, find vs. fix lead to convergence
- Severity mix is primarily low-sev
- Priority mix is primarily low-pri
 - Caveat: priority is a judgement call