Quality Assurance: Early Work Items

Introduction: Ian King

- Software Test Lead, Microsoft Corporation
- 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

Introduction to QA concepts

- QA, QC and testing
  - Quality Assurance: making it right the first time
  - Quality Control: making it right every time (i.e. in production)
  - QA and QC both include testing as an activity

What is the 'value add' of QA?

- Classic research: fixing a defect is cheaper the earlier you catch it
  - $: found in spec process
  - $$: found during implementation
  - $$$: found during post-implementation testing
  - $$$$: found in the field
    - QFE, service packs, product recalls, lawsuits
    - Customer confidence

'Make It Didn't Happen'

- The best bug is the one that was never born!
- QA is about process
  - Design review
  - Implementation review
  - Structured testing and evaluation
  - Instrumentation/testability
  - Best practices
- QA does not mean bodies: Pacific Telecom
- Lesson of History: good process leads to fewer defects

The deliverable of QA

- QA delivers information
  - What is known about the quality of the code?
  - What are the risks of known defects?
  - What is not known, i.e. untested?
  - What risks may arise from unknown defects?
    - E.g. "We didn't test for malicious use"
- "Bearer of bad news"
- "Validation of the vision"

Scaling to the project

- Large project
  - Individual design/programming and QA teams
  - Another team to coordinate and administer
- Medium-sized project
  - QA often assumes coordination role
- Small/solo project
  - Develop 'functional schizophrenia'
  - Write it down

QA is everyone's job!

…testing is only one part.

What does QA do early in the development cycle?

- Publish (and promote) QA requirements
- Review design work
- Develop testing strategy

Establish QA Requirements

- Statement of requirements
- Feature specifications
- Implementation specifications
- Design change process
- Development schedule
- Build process
- Developer practice
- Defect process
- Release criteria

Statement of Requirements: Why are we here?

- Who is the customer?
- What problem are we solving for the customer?
- Should NOT include:
  - Feature details
  - Implementation details

Feature Specifications

- What will we make to solve the customer's problem?
- Does not prescribe implementation
- Descriptive:
  - Workflow
  - Actor
  - Interface

Implementation specifications

- Typically for large projects, but always beneficial
- How is a feature implemented?
  - Details of resource usage, exception handling, use of published standards, etc.
- Dependency on other feature implementation
- Dependency on external factors
  - Development environment (SDKs)
  - Other products' modules (e.g. MSXML)

Design Change Process

- How are design changes documented?
  - DCR vs. "bug"
- How are change decisions made?
- When has "the ship sailed"? Design Freeze

Development Schedule

- When will specs be complete?
- When will code be available?
- When will features be complete?
- When will code be stable?
- Beta releases?
- Leave enough time for the endgame:
  - Complete test pass on Release Candidate
  - Test of final installation media (may include digital signing, release notes)

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

Build Process

- Source control
  - Undo the 'oops
- Centralized build
  - Be sure everyone is testing the same bits
  - Avoid platform dependencies (msvcrtd)
- How often are new builds generated?
  - Periodic
  - Event-Driven
- Configuration management

Developer Practice

- Private builds
- Buddy builds
- Code review
- Code analysis tools
- Unit testing

Defect Process

- Why are defects tracked?
- How are defects tracked?
- What is the lifecycle of a bug?
- How are defects prioritized?
- Controlled check-ins/triage process
- Defect analysis:
  - Defect source analysis
  - Root cause analysis

Release Criteria

- When are we done?
- Indicators of completeness:
  - Quantity of defects being found
  - Severity of defects being found
  - Completeness of testing

Review Design Work

- Are these documents sufficient to scope the project?
- Are they logically consistent?
- Is the project testable?
  - Test hooks, registry entries, compiler directives
  - Instrumentation
- Does the project address the stated requirements?

Review Design Work (con't.)

- Evaluate use scenarios
  - Sensible control flows?
  - Features appropriate to use? (E.g. quiesce server)
- Evaluate failure scenarios
  - Meaningful error feedback
  - Single points of failure
  - Cascading failures
- Understand dependencies

Developing Test Strategy

Elements of Test Strategy

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

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 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 Harness/Architecture

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

Test Cases

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

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