Software Analysis

Analysis in Practice

Presenter: Jonathan Aldrich

Microsoft and Software Quality

- People
- Organization
- Technical
- Testing
- Static Analysis
- Process

Microsoft People

- Perhaps most important factor in quality
- 10x rule of thumb
  - Most productive developers are 10x better than least productive developers
  - Holds for quality, too!
- Microsoft aims to hire 10xers

Managers

- Technical background
  - Focus: technical knowledge + business
    - Contrast: most companies hire based on people skills
  - Team leads (first-level managers) cut code
    - True all the way up to dev leads of Word/Windows
- Consequences
  - technical tradeoffs made by qualified people
  - middle management weak on people skills

Developers

- Hire 2-3% of applicants
  - Chosen by technical people
  - Question gauntlet
    - e.g. how much water flows in Mississippi?
- Gates: hiring priorities priorities
  - IQ (#1)
  - ambition
  - technical expertise
  - business judgment
    - ability to ship products, no distaste for business

Testers

- Focus
  - Understanding needs of users
  - Technically competent
  - Attitude
    - eternal skeptics
    - focus on breaking things
- Not a way to get a development spot
  - Often look outside CS programs
    - e.g. science majors

Rewards

- Salary historically low
  - $45k entry-level salary in mid-90s
  - $90k for top developer grade
- Generous support & bonuses
  - Personal offices (not cubicles)
  - Good tool resources
  - Large annual bonuses
  - Stock options, purchase plan

Microsoft and Software Quality

- People
- Organization
- Technical
- Testing
- Static Analysis
- Process

Organization: Large Teams Work Like Small Teams

- Culture of independence
  - Individual developers and teams
- Close communication
  - Minimize dependencies across groups
- Culture of openness
  - Honesty rewarded even when the news is bad
  - Or at least, hiding information is penalized!
- Shallow hierarchy
  - Monthly reports go all the way to Gates

Team Structure

- Program Manager
  - specify/schedule/coordinate
- Development lead
  - Developer
  - Developer
  - Developer
  - ...
  - (up to 8)
- Test lead
  - Tester
  - Tester
  - Tester
  - ...
  - (up to 8)

Microsoft and Software Quality

- People
- Organization
- Technical
- Testing
- Process

Architecture

- Focus on independence between teams
  - Divided by feature
- Little up-front specification
  - Instead negotiate between teams
    - Aided by nightly build cycle
  - Gates: “One document. It’s the *source code*.”
- Emphasis on shared code
  - Pressure from Bill Gates for years
- Common language - C
- Shared code ownership
  - Everything understood by more than one person

Microsoft and Software Quality

- People
- Organization
- Technical
- Testing
- Static Analysis
- Process

Culture that Values Testers

- Separate management chain
  - tester → test lead → test manager of product unit → manager of product business unit
- Separate technical promotion path
  - Ladder levels 9-13 w/ pay increase
  - Advance w/o switch to management or development
- Hiring specifically for test positions

Separate Testers, Closely Tied In

- Provides independent check on development
  - Testers have different mindset
- One tester pair with each developer
  - High ratio compared with industry
    - Reduces load on development
    - Increases confidence in correctness
    - Cheap compared with product problems!
- Developers test, too!
  - Test features
  - Run regression tests before check-in
  - Interact with customer in usability lab
  - Keeps developers accountable for quality

Developer/Tester Coordination

- Private release to tester
  - Irons out bugs before release
  - Major tester responsibility: rapid feedback
- Test release document
  - what is ready, what are issues

Test Perspectives

- User
  - how will customers use it, will it make sense
- International
  - will work across languages/nationalities
- Hardware
  - compatible with platforms, peripherals
- Software
  - compatible with other software
- Spec compliance
  - compliance with product concept and program manager spec
- Stability
  - measurable metrics: bug-find rate, bug severity, others
  - gut feel: “are we ready” to ship product

Testing Practices

- Heavy automation of tests
  - Facilitated by in-house testing tools
- Testers responsible for validation
  - Interact with customer, support personnel
  - Study critics, competitors
- Test Plans
  - Focus on high-risk areas
  - Scripts for manual tests
  - Leave room for ad-hoc testing
    - Deviate from scripts, use in different ways to cause product to fail
  - Design based on post-mortem of last version
    - drive test strategy based on experience
- Checklists
  - Product-specific issues for testing focus

Reviewing

- Few formal process
  - Seen as overly heavyweight
- Targeted review use
  - High-level design reviews for new features
  - Peer reviews for new hires
  - Critical code

Microsoft and Software Quality

- People
- Organization
- Technical
- Testing
- Static Analysis
  - see other slide deck
- Process

Analysis Maturity Model

Caveat: not yet enough experience to make strong claims

- Level 1: use typed languages, ad-hoc tool use
- Level 2: run off-the-shelf tools as part of process
  - pick and choose analyses which are most useful
- Level 3: integrate tools into process
  - check in quality gates, milestone quality gates
  - integrate into build process, developer environments
  - use annotations/settings to teach tool about internal libraries
- Level 4: customized analyses for company domain
  - extend analysis tools to catch observed problems
- Level 5: continual optimization of analysis infrastructure
  - mine patterns in bug reports for new analyses
  - gather data on analysis effectiveness
  - tune analysis based on observations

Static Analysis in Engineering Practice

- A tool with different tradeoffs
  - Soundness: can find all errors in a given class
  - Focus: mechanically following design rules
- Major impact at Microsoft and eBay
  - Tuned to address company-specific problems
  - Affects every part of the engineering process

Microsoft and Software Quality

- People
- Organization
- Technical
- Testing
- Static Analysis
- Process

Product Strategy

- Good enough product out quickly
  - Helps achieve adequate level of quality
- Gradually refine and add features

Synchronize and Stabilize

- Work in parallel across teams
- Synchronize with daily build
  - Synchronize with new code every day
  - Check in code by deadline, synch code afterwards
  - Compile and test overnight
  - Bugs that “break the build” fixed immediately
- Stabilize at milestones
  - Typical schedule
    - 8 weeks development
    - 2 weeks integration
    - 3 weeks buffer (allow for running over)
  - Temporarily freeze new development
  - Drive bug count towards zero – releasable!

Zero Bugs Memo

- Memo from Word Dev Manager
  - Context: “infinite defects”
    - Testers found bugs faster than dev could fix
    - Fixes introduced new bugs
- Quotes
  - “your goal should be to have a working, nearly-shippable product every day”
  - “when this [bugs] happens, you must evaluate the problem and fix it immediately”
- What does “zero bugs” mean?
  - Extensive testing with no high-severity bugs
  - All remaining bugs deferred to next cycle

Development Practices

- Work in one location (Redmond)
  - Facilitate communication
- Close interaction with customer
  - beta releases, Watson reports, usability labs
  - call center – weekly reports generated
  - "eat your own dog food"
    - using Microsoft's products within Microsoft
    - e.g. Bill Gates required tool group to support features needed by Microsoft developers

Process and Product Improvement

- Project post-mortem
  - Source of bugs?
  - How could design generate less bugs?
  - How could tools generate less bugs?
- Milestone 0 – cleanup
  - 1-2 months to fix problems before starting next version