Quality Assurance

Pragmatic Programmer Tip
Think about Your Work

Turn off the autopilot and take control.
Constantly critique and appraise your work.

Readings

Required:

- *"Rapid Development"*, McConnell, Chapter 4, Section 4.3 – QA Fundamentals
- "*Pragmatic Programmer*", Hunt and Thomas, Chapter 8, p 237-247 – Ruthless Testing

Other good stuff:

- "If you didn't test it, it doesn't work", Coldwell (link on class Resource list)

Outline

- QA basics – getting off to a good start
- Test
  - What makes a good tester?
  - What types of testing can we do?
  - Can tools help?
- Bugs!

Test material adapted from several talks by Ian King, tester at Microsoft for many years.

What does QA mean to you?

Your thoughts?

Things we can do to ensure we produce a high quality (low defect) product

Two main approaches to QA:

- Process: Build in quality from the start
- Test: Add quality through removing bugs

What are ways to foster quality from the get go?

Over to you again – 3 ideas…

Some practices:

- Good design and planning
- Coding style guides
- Code reviews/walkthroughs
- Pair programming

Tools can help!

- Configuration Management
  - Document changes, enforce good practices

- Memory behavior
  - Detects corruption and leaks

- Performance
  - Time of routines

Gnu gprof

Interesting fact

Up to 4x the normal number of defects are reported for released products that were developed under excessive schedule pressure (Jones, 94)

Why?

Test – the most common QA practice

Verify: “Did we build the system right?”

Validate: “Did we build the right system?

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

What types of testing can we do?

Back to you to generate some ideas!

- Functional (include boundaries)
- Performance
- Security
- Stress
- Resource exhaustion, errors, and recovery
- Reliability/availability
- Usability (is it obvious)

Some testing jargon

Black box testing

Treats the system as atomic

Best simulates the customer experience

White box testing

Examines the system internals

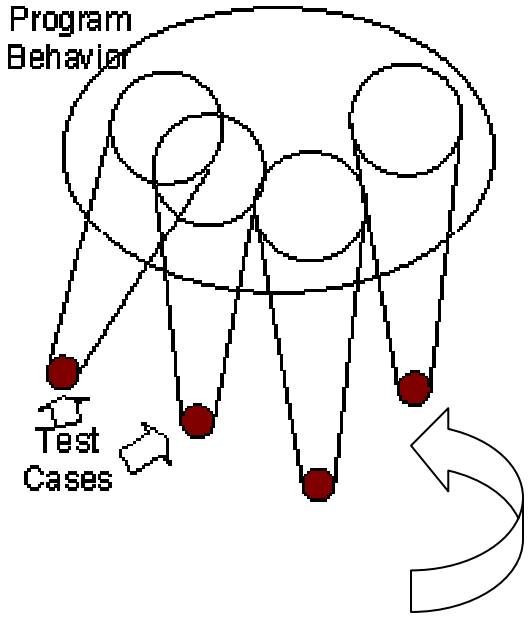
Trace data flow directly (ie, in the debugger

Bug report contains more detail on source of defect

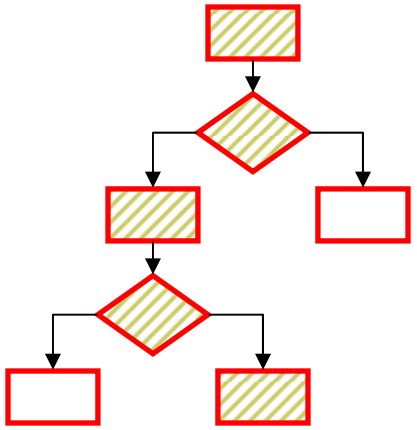
May obscure timing problems (race conditions)

It's black and white, right?!

- In black box, the tests are usually intended to cover the space of behavior
- In white box, the tests are usually intended to cover the space of parts of the program



Often tester driven



Often developer driven

How do we design good tests?

- Well-defined inputs and outputs
  - Consider environment as inputs
  - Consider 'side effects' as outputs

- Clearly defined initial conditions
  Clearly described expected behavior

- Specific – small granularity provides greater precision in analysis

- Test must be at least as verifiable as feature

… and good test cases?

- Valid cases - What should work?
- Invalid cases – What shouldn't?
- Boundary conditions
- Error situations – ie, resource exhaustion

What's the trick to running tests?

Manual Runs

Tests that require direct human intervention with the system

Necessary when:

- GUI is present
- Behavior is dependent on physical activity

Advisable when:

- Automation is more complex system being tested!
- System is changing rapidly (early development)

Automated Testing

Tests that can be executed independent of human interaction

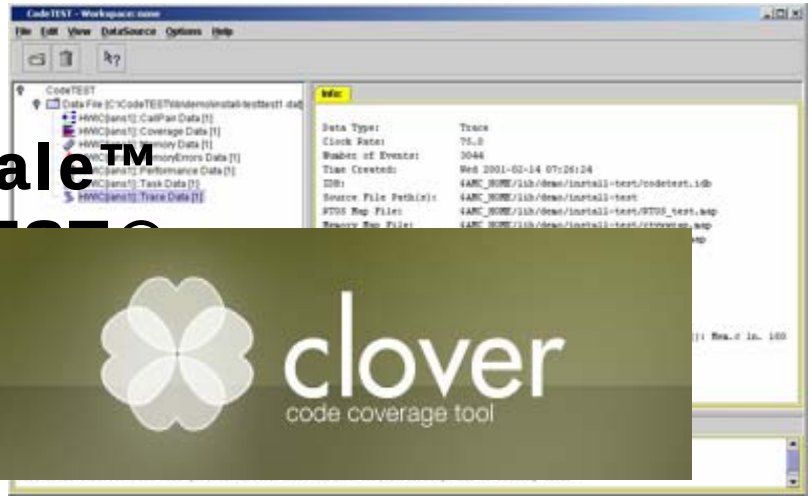
- Good: replaces manual testing
- Better: performs tests difficult for manual testing (e.g. timing related issues)
- Best: enables other types of testing (regression, perf, stress, lifetime)
- Cost: Time investment to link tests into harness

Tools rock!

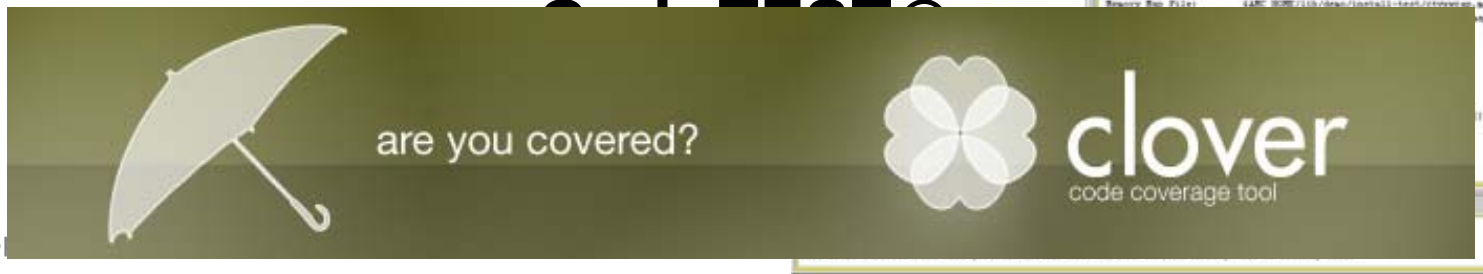
Example – Test Harness

- Tests are programmed as modules, then run by harness
- Harness provides control and reporting

Example – Code Coverage



Freescale™



Pragmatic Programmer Tips

Test early, test often, test automatically

Coding ain't done 'til all the tests run

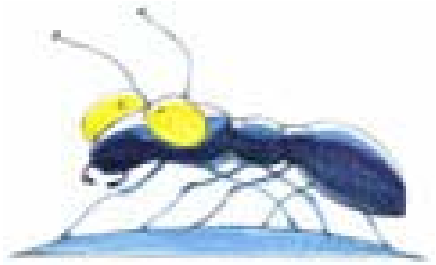
Find bugs once

BUGS!

Managing Bugs

What is a bug?

- Formally, a “software defect”
- System fails to perform to spec
- System causes something else to fail
- System functions, but does not satisfy usability criteria



If the system works to spec and someone wants it changed, that’s a feature request

What makes a good bug report?

Include:

- Reproducible steps – how did you cause the failure?
- Observed result – what did it do?
- Expected result – what should it have done?
- Any collateral information: return values/output,…
- *Environment*
  - OS version , env variables, compiler flags, …
  - Test platforms must be reproducible
  - "It doesn't do it on my machine"

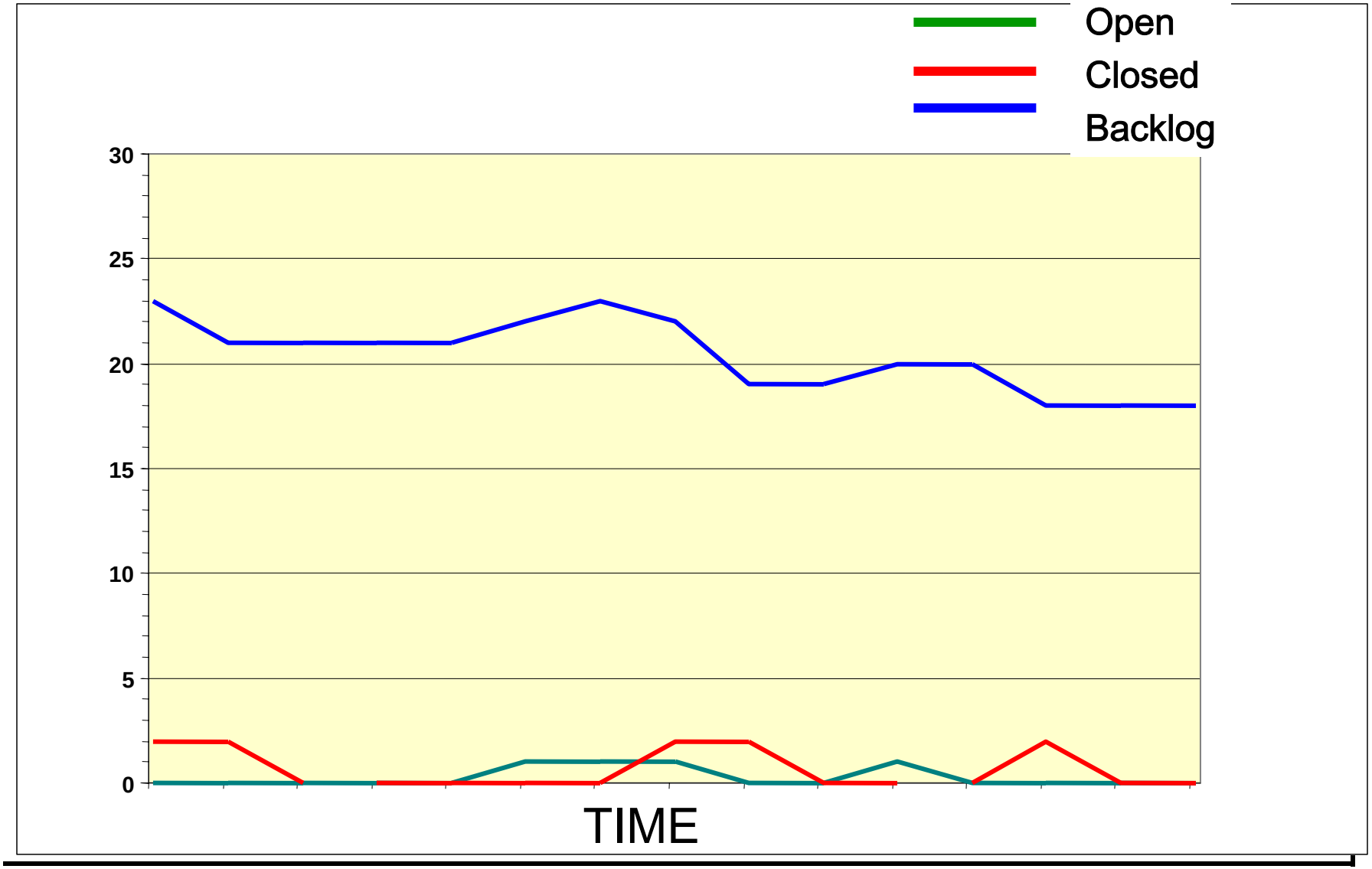
Generally have a form to help

And a database to file it in

Why is a bug db useful?

- Don't lose bugs
- Can pass them around easily
- Justify resources
- Justify your work
- Help indicate when the product ready for release
  - Based on the number and type of bugs
  - Based on a graph of the rate of bugs occurring
- Enables all sorts of cool metrics that management (and customers!) likes!

SPR rate



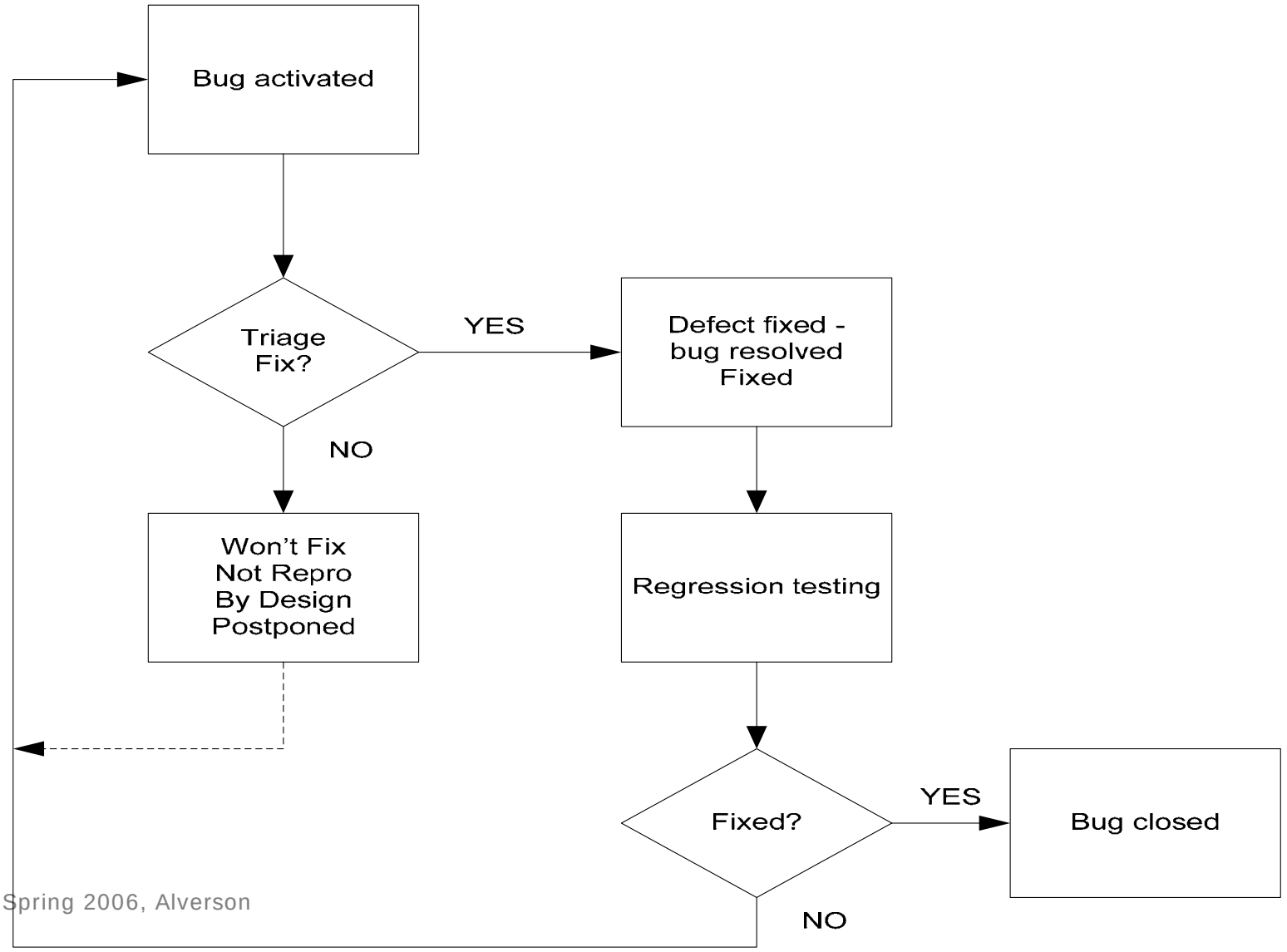
Classifying bugs

Cray links severity with priority. Some keep it separate.

Another classification

- 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
  - 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 (idealistic)



Regression Testing

1. Ensure that the bug you just fixed, doesn't reappear later though other mods
   - Add a test case to your suite for it!
2. Ensure that the fix you just added doesn't break things working previously
   - Rerun the test suite before checkin

When can I ship?

Tune in on Monday!