Limbo Lower Now: An Agile Approach to Defect Management

Agile Testing Days 2010

Lisa Crispin

With Material from Janet Gregory
Introduction

My defect tracking background:
- As programmer – no DTS
- Tech support – DTS developed in-house
- Traditional QA – vendor DTS, customized
- Agile teams – variety of approaches
Your Defect Tracking?

How about you?

- How many use a DTS?
- How many don’t use one?
- How low is your “defect limbo” bar?
Takeaways

- Defects and technical debt
- When to use a Defect Tracking System
- Alternatives
- Defect metrics
- Setting the bar lower
Traditional View

- Bugs must be documented
  - Especially those found in production
- Provide information for root cause analysis
- Knowledge base – did similar bug occur earlier?
- DTS provides useful metrics
Lean Development / Agile View

- Defect queues are queues of rework – waste (Poppendieck)
- Find, fix test-first, and forget
- Better yet, prevent
  - zero bug tolerance
- Bugs => Technical Debt
Choices

An agile approach:

- Understand the problem
- Do what works for your team
- Focus on goals: bug prevention
- Start simple, add as needed
- Explore alternatives
Why Log Bugs – or not

What are the advantages of using a DTS?

What are the downsides to using a DTS?
Some Advantages of Using DTS

- Knowledge base
  - Place to keep supplemental docs
- Prioritizing
- Traceability
- Distributed/large teams
- Customer ease, visibility, convenience
- Metrics
  - Redundant with test-first, frequent releases
Some Disadvantages of using DTS

- Overhead
- Duplicates test
- Impedes communication
- Waste
  - Inaccurate or inadequate reports
  - Caught during development
  - Won’t get fixed
- Doesn't work towards goal
  - zero defects, bug prevention
Defect Metrics

- Traditionally, tell us state of quality, release readiness
- Using tests to drive incremental, iterative development tells us this
- Trends more important than numbers
- Who cares how many bugs found in development?
- May be useful for team goals
- Careful how you use metrics
Some Other Perspectives

- What we can learn from bugs
- Alternatives to logging bugs
- When tracking defects makes sense
Bugs as the “Hidden Backlog”

- Acceptance tests = desired behavior
- Defect reports = misbehavior (Antony Marcano)
- Ignored defects = missed features
  - Misbehavior not addressed
- Defects prioritized over new features
  - Misbehavior over desired features
- Information can get lost in DTS
Choosing to Not Log a Bug

- **Unit-level bugs**
  - Write a test, not a bug
- **Higher-level regression failures**
  - Test usually enough
- **Bugs found during development**
  - Fix immediately
  - Log on task card
Choosing to Log a Bug

- Found in later iteration
  - Log if not fixed immediately
- Legacy bugs
  - Write new story/feature
  - Don't log if it won't be fixed
- Found in production
  - Fix immediately or
  - Prioritize, estimate, plan
- Non-bug production issues
  - “production support requests”
Alternative Ways to Track Bugs

- Cards
- DTS
- Tests
Logging Bugs on Cards

Story, task or bug cards – physical or online

- Tangible
- Visible – color coding
- Attach screen prints
- Use when:
  - Disciplined team
  - Bugs need high visibility
Logging Bugs in DTS

Use a DTS when
- Team is distributed
- Audit requirements
- Release notes
- Not all bugs will be fixed immediately
- Legacy system with lots of defects
Logging Bugs with Tests

Use the test written to reproduce the bug and verify the fix when:

- Everything learned from fixing bug is captured in test
- Test can be automated
- Team is disciplined and writes tests for every bug found
When to Fix Bugs

- **Fix now**
  - way cheaper
  - prevents technical debt

- **Estimate, prioritize and fix later**
  - complex, high effort
  - legacy code, may destabilize
  - customer decision

- **Never fix**
  - about to be rewritten
  - low risk
  - don't pretend you'll fix if you won't!
Lowering the Bar

- Exploring alternatives
- Preventing bugs in the first place
Experiment!

- Start a new project with no DTS, and see what you need.
- Set rules - “no more than 10 bugs at once”
  - Analyze and address bad trends
- Fix all bugs
  - Low priority usually quick to fix
  - Unfixed bugs build technical debt quickly
- Treat bugs as stories
  - Estimate and prioritize
- Use color-coded cards and stickers
  - Visible, promotes communication
Make It a Team Problem

- Understand your needs
- Choose solution by team consensus
  - Cards and stickers (real or online)
  - Story board
  - Online DTS
  - Combo: DTS and cards
- Ask other teams
Prevent Them In the First Place!

- Make testing integrated part of development
- Practice TDD and ATDD
- Gain domain expertise
- Exploratory testing by testing professionals
- Consider ripple effects of code changes
- Look at problem areas, experiment with solutions
Action!

- Think of one thing your team can do to prevent defects, starting next week.
How Low Can You Go?

- Zero defects is achievable
  - But go for baby steps
- Focus on bug prevention
- Do what works for *your* team
- Test/code together
- Use retrospectives
Questions?
Agile Testing

Agile Testing: A Practical Guide for Testers and Agile Teams

By Lisa Crispin and Janet Gregory

www.agiletester.ca
All Proceeds to Charity!

*Beautiful Testing: Leading Professionals Reveal How They Improve Software*

Edited by Tim Riley, Adam Goucher

Includes chapter by yours truly
Recommended

Bridging the Communication Gap: Specification By Example and Acceptance Testing

Gojko Adzic
Lean Development

Implementing Lean Development: From Concept to Cash

Mary and Tom Poppendieck
Some Agile Testing Resources

- lisacrispin.com
- janetgregory.ca
- exampler.com
- agile-testing@yahoogroups.com
- testobsessed.com
- testingreflections.com
- dhemery.com