Why Does Software Quality (Still) Suck?

The Ongoing Frustration with Lousy Software

RBCS

TIME TESTED.
TESTING IMPROVED.

www.RBCS-US.com

Introduction

- Software quality is abysmal, and has been ever since software was created
- Software is everywhere, and so are software bugs
- These bugs costs millions of dollars and sometimes kill people
- Why do we put up with this situation?
- Manufacturing achieves six sigma levels of quality
- Shouldn’t software quality be better?
- Let’s look at some of the barriers to quality software, and also acknowledge some of our mistakes

Six Sigma Levels of Quality

- Manufacturing can achieve six sigma quality
  - 1 sigma = 31% yield
  - 3 sigma = 93%
  - 4 sigma 99%
  - 6 sigma 99.9997%
- Roughly, 3 defects per million items

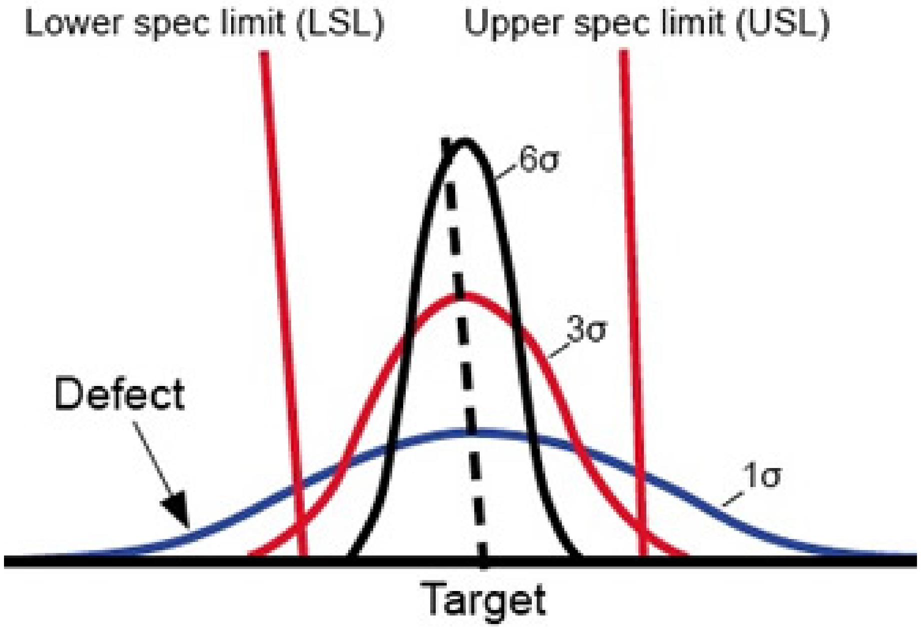


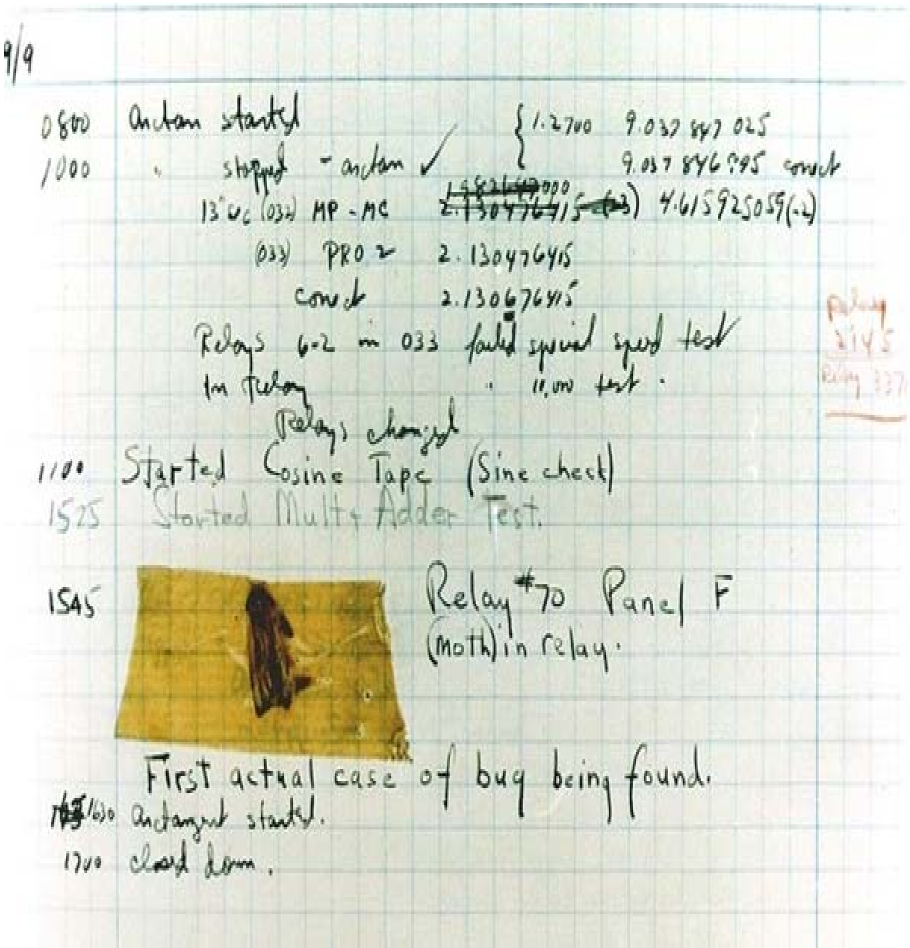
Figure from APO-Tokyo.org

How About Software?

- In software, according to Capers Jones' studies, we release over 13,000 defects in a typical million-line C++ program following typical development practices
- Usually, only 85% of defects are detected (DDE) and removed
- Even RBCS clients that achieve 99% DDE will still release close to 1,000 defects in similar-sized software
- It's even worse: every copy of a software app contains all the bugs, so in fact we have 0% yield!
- Ultimately, software quality must increase dramatically

Bugs, With Us Since the Start

- Sept 9, 1947, Grace Hopper files first bug report to include a real bug!
- Ada Lovelace discussed possibility of bugs in 1843, regarding Babbage's proposed analytical engine
- Assuming about 10MM apps in use, there are trillions of bugs in the wild!
- What can we do?



9/9

0800 Antan started
1000 " stopped - antan ✓ {1.2700 9.037 847 025
9.037 846 995 correct
13" UC (032) MP-MC ~~1.982147000~~ ~~2.130476415~~ (-3) 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 11,000 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 Antangent started.
1700 closed down.

Relay 2145
Relay 3370

What To Do

- Main problem: software engineering has not matured to true engineering yet
- Think of how bridges or aircraft are designed and built
- Now compare how we design and build software
- As Frank Lloyd Wright said, “You can use an eraser on the drawing board or a sledgehammer on the construction site”
- This saying applies to software, too
- While software bugs can often be easily fixed, the consequences of software failures cannot
- So, given the current state of the practice, what mistakes do we make that exacerbate the situation?

Failure to Follow Best Practices

- As Jones' figures show, following best practices leads to an order of magnitude fewer bugs
- The state of development practice lags best practices by at least 25 years
- Software testing is even worse, with few testers using techniques known for 40+ years
- If we applied best practices of software development and testing, software would suck a lot less
- So why don't we?

Bowing to Schedule Pressure

- One major cause of bug-breeding sloppiness is schedule pressure
- Unrealistic schedules (or excessive content in agile) are common
- Long-standing problem: Fred Brooks wrote of "20 pounds in a 10 pound bucket"
- Management imposes unrealistic schedules and then punishes failure to achieve them
- Since testing and other software quality activities are infinitely compressible, they often suffer disproportionately

Underfunding Quality Activities

- This problem compounds the previous one
- Not only are schedules tight, but there’s usually not enough money
- Test professionals don’t always make a good business case for testing and quality
- The need for development is clear, but testing and other quality activities seem less so
- Most software professionals are “vaguely religious” about software testing and quality activities
- Most quality professionals are not very effective proselytizers

Poor Measures of Quality and Its Impact

- Further, we don’t have good tools to measure quality and the impact of not having it
- Defect density (which I used earlier) is actually a weak indicator of quality
- Quality is fitness for use, which is not necessarily directly correlated with defect density
- What measures of quality we do have, we aren’t always able to present realistically and meaningfully
- Sometimes we are encouraged to or rewarded for toning down our findings when they are inconvenient
- Similarly, productivity measures are weak
- These measurement problems combine to result in dysfunctional choices about staffing
- We need to develop and standardize good ways to measure and categorize defects, failures, productivity, and quality costs

Underqualified Software Professionals

- To build a bridge, licensed professional engineers are required
  - To *develop* software?
  - To *test* software?
- Widespread failure to follow best practices makes it hard to say what "qualified" means
- Lack of good quality and productivity measures makes it hard to measure the qualifications of a team
- This has contributed to the race to the bottom with outsourcing
- Certifications are a start, especially if we can make them omnipresent
- It's important, though, that these certifications be credible and valuable, and be seen as credible and valuable

Transfer of Costs of External Failure

- Perhaps most toxic, software companies can transfer the costs of bugs *onto their users and customers*
- Most enterprise software companies make more money from support than from new software
- Thus, they are financially incented to release software that meets minimal quality standards
- Software as a service is often a way that the industry has (very successfully) put lipstick on this pig
- "Fast failure" is a Silicon Valley motto: easy to say when your company doesn't bear the costs
- We need an implied warranty of fitness, as with almost all other products
- We also need better quality measures to those warranties that are given can be enforced

Conclusions

- Software quality problems have plagued our industry from the start
- In the long-term, we need to make software a true form of engineering
- In the short-term, we are making many avoidable mistakes
- Some of these mistakes are the result of bad incentives
- We can and should fix these problems, given the omnipresence (and increasing criticality) of software

To Contact RBCS

For over twenty years, RBCS has delivered consulting, outsourcing, and training services to clients, helping them with software and hardware testing. Employing the industry’s most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address:	RBCS, Inc. 31520 Beck Road Bulverde, TX 78163-3911 USA
Phone:	+1 (830) 438-4830
E-mail:	info@rbcs-us.com
Web:	www.rbcs-us.com
Twitter:	@RBCS, @LaikaTestDog
Facebook:	RBCS-Inc.