

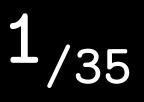
Software Testing Pitfalls

Yegor JPo Mosco 5 Ar



- Yegor Bugayenko
 - JPoint 2019
- Moscow, Russia
 - 5 April, 2019

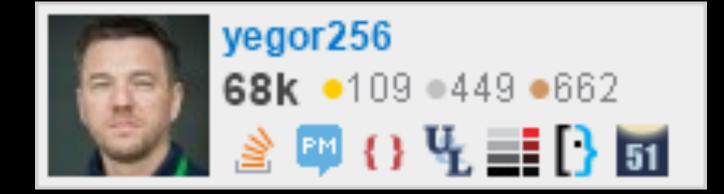
Obloghacks

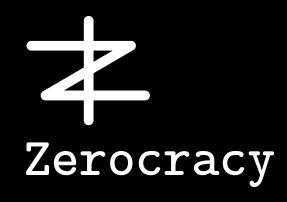














Elegant Objects



1.3 September 12, 2016



TL;DR There a object-orient completely as books. For getters, set evil.



Code Ahead

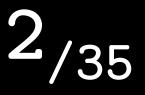
TL;DR It's an autobiographical fiction book about a software architect who is involved in programming, DevDps, testing, quality assuranc and control, corporative management, project management, and...love.

by Yegor Bugayenko

ORACLE®

Certified Master

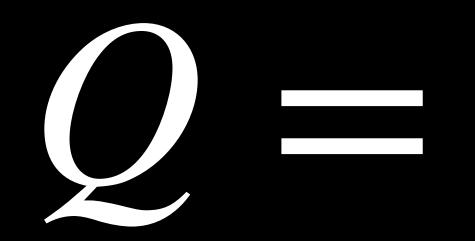
Java EE 5 Enterprise Architect





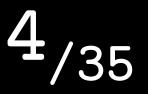


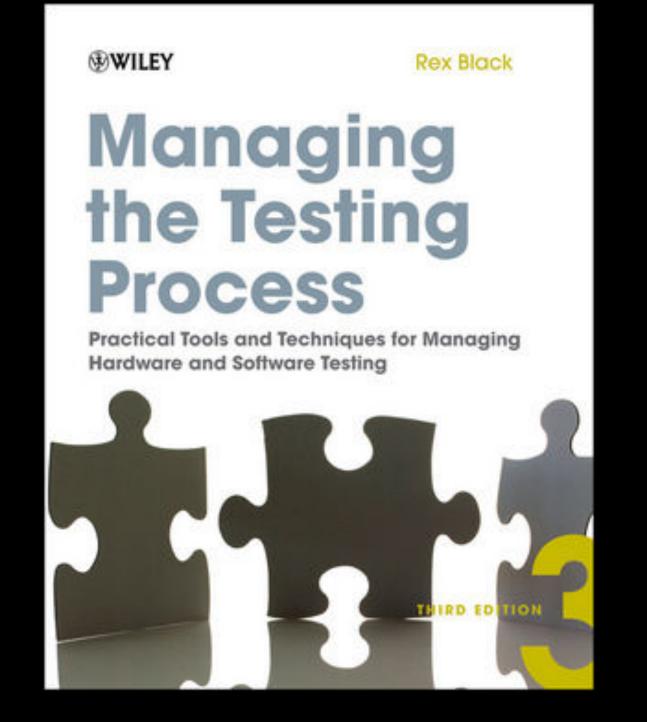




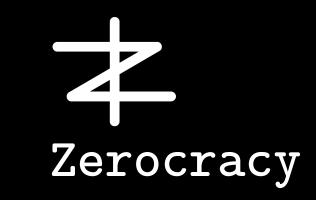








Rex Black, 2009 Defect Detection Effectiveness







Software Defect Removal Efficiency

Ву

Capers Jones, President Capers Jones & Associates LLC Email: CJonesiii@cs.com

Abstract

The most important contributor to the quality of software-intensive systems is the quality of the software components. The most important single metric for software quality is that of defect removal efficiency (DRE). The DRE metric measures the percentage of bugs or defects found and removed prior to delivery of the software. The current U.S. average in 2011 is only about 85% of total defects removed. However, best in class projects can top 99% in defect removal efficiency. High levels of DRE cannot be achieved using testing alone. Pre-test inspections and static analysis are necessary to top 95% in defect removal efficiency.

Copyright© 2011 by Capers Jones & Associates LLC. All rights reserved.

Introduction

In the 1970's the author worked for IBM. Software applications were growing larger and more complex so quality was becoming a serious issue. IBM began a careful analysis of software quality. Measurements were taken of defects found in software requirements, design documents, source code, user manuals, and also "bad fixes" or secondary defects accidentally included in defect repairs.

At the same time IBM developed the function point metric, because it was necessary to analyze non-coding defects and non-coding development activities as well. After several years of data collection, it was possible to determine the relative contribution of various defect origins on total software defects. The total number of defects from all five sources was termed the "defect potential" of a software application.

Table 1 shows approximate U.S. averages from more than 13,000 projects. Table 1 shows the average volumes of defects found on software projects, and the average percentage of defects removed prior to delivery to customers:

Table 1: Defect Removal Efficiency by Origin of Defects Circa 2011 (Data Expressed in Terms of Defects per Function Point)

Defect Origins	Defect	Removal	Delivered
	Potentials	Efficiency	Defects
Requirements	1.00	77%	0.23



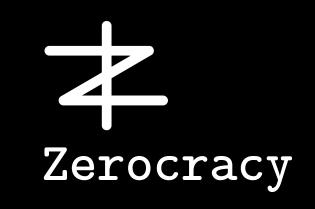


Capers Jones, 1996 Defect Removal Efficiency





Programmers





Testers

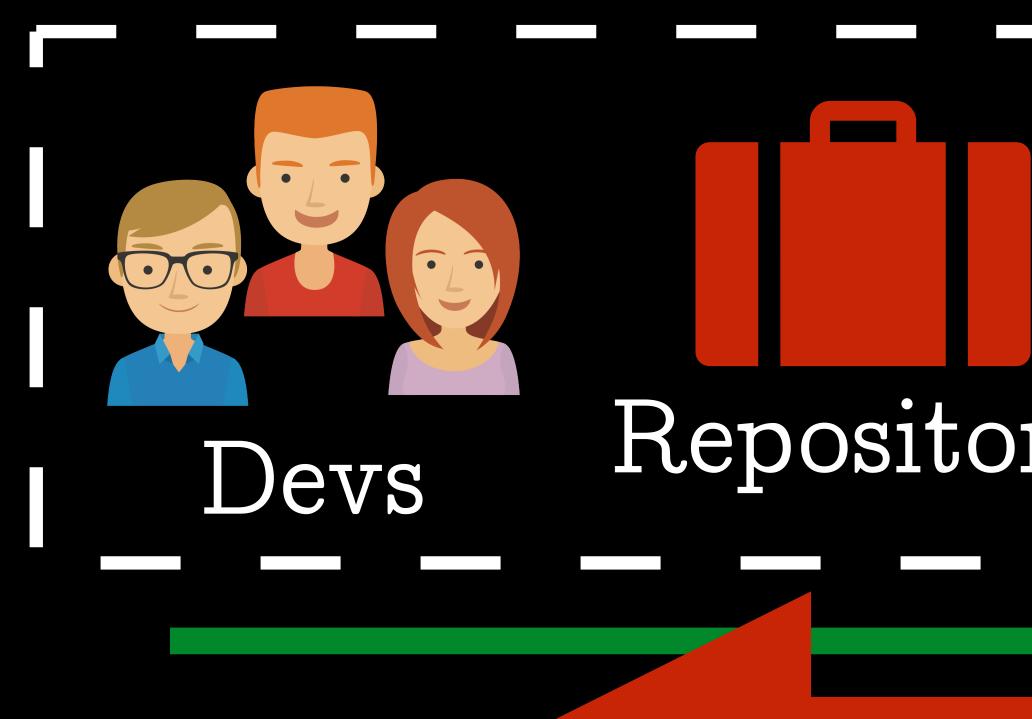






"A good programmer will e code, ess Bugs "







Repository Staging Production

Pipeline



ractice tutorial

What Is Software **Testing?** And Why Is It So Hard?

Software testing is James A. Whittaker, Florida Institute of Technology

arguably the least understood part of the development process. Through a four-phase approach, the author shows why eliminating bugs constant trade-off.

irtually all developers know the frustration of having software bugs reported by users. When this happens, developers inevitably ask: How did those bugs escape testing? Countless hours doubtless went into the careful testing of hundreds or thousands of variables and code statements, so how could a bug have eluded such vigiis tricky and lance? The answer requires, first, a closer look at software testing within the why testing is a context of development. Second, it requires an understanding of the role

decisions.

never tested. We might have

the environment but had no it. Perhaps we did not (or o

replicate the user's comb

hardware, peripherals, oper

tem, and applications in our te

software testers and developers-two very different functions-play. Assuming that the bugs users report I The user's operating envir occur in a software product that really is in error, the answer could be any of these:

- The user executed untested code. Because of time constraints, it's not uncommon for developers to release untested code-code in which users can stumble across bugs.
- The order in which statements were executed in actual use differed from that during testing. This order can determine whether software works or fails.
- The user applied a combination of untested input values. The possible input combinations that thousands of users testing problem and process, this article can make across a given software inter- investigates the problems that testers face face are simply too numerous for testers and identifies the technical issues that any to apply them all. Testers must make solution must address. I also survey existing tough decisions about which inputs to classes of solutions used in practice.

70 IEEE SOFTWARE January/February 2000

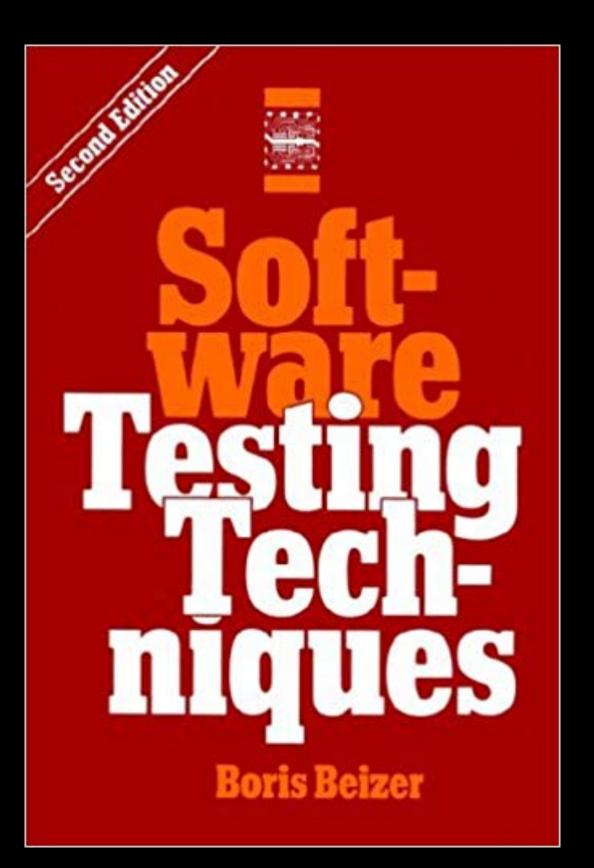
For example, although comp write networking software are up to create a thousand-node networ their testing lab, users can-and create such networks. Through an overview of the softwa

0740-7459/00/\$10.00 @ 2000 IEEI

James A. Whittaker: "Software testing is + cess of re system to mether it matches specification and executes in its intended environment."



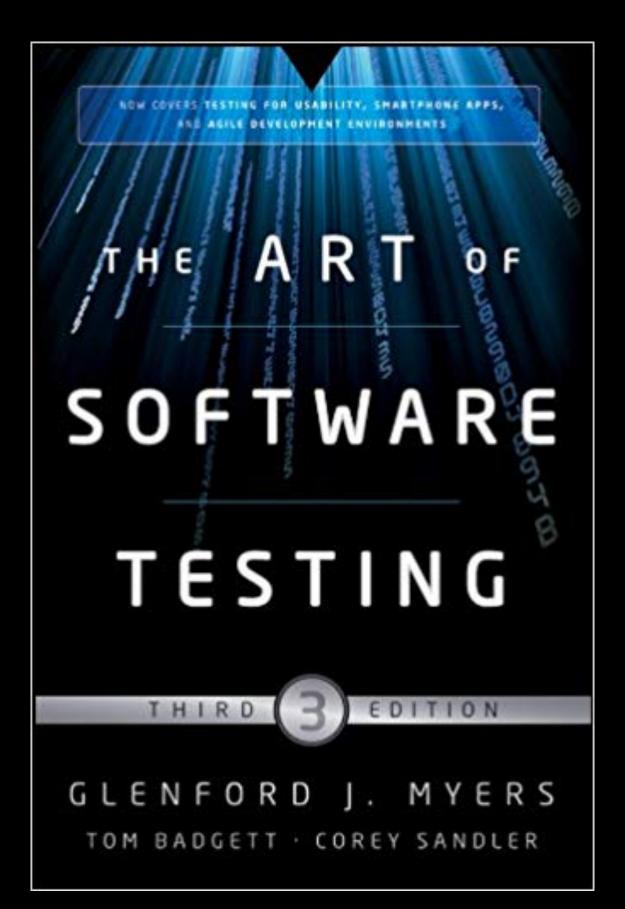






Boris Beizer: "If your objective is to demonstrate a high probability of working, that objective is best achieved by not testing at all!"





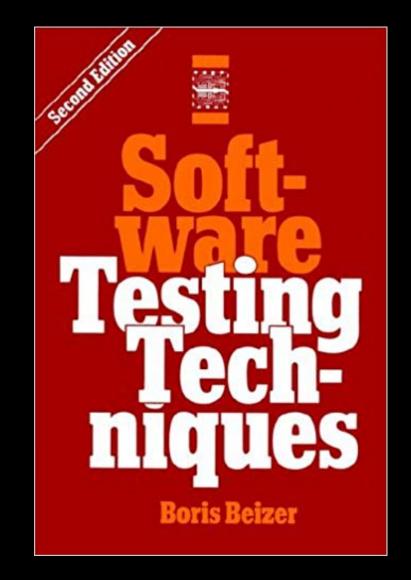
errors."

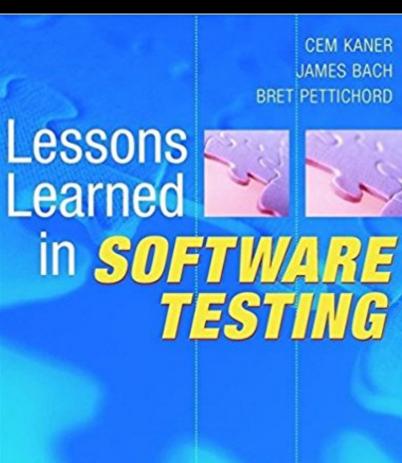




- Glenford Myers: "despite the plethora of software testing tomes available on the market today, many developers seem to have an attitude that is counter to extensive testing. Testing is the process of executing a
- program with the intent of finding

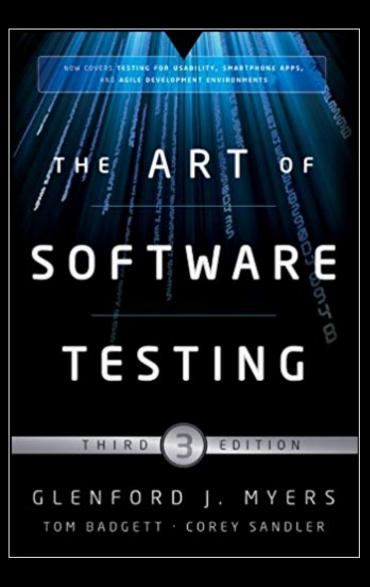


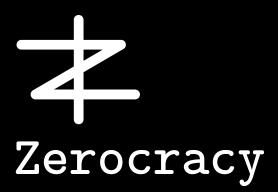




A Context-Driven Approach

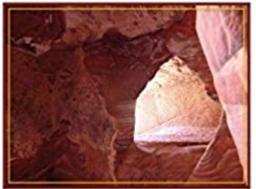
WILEY Copyrighted Material Testing Computer Software The bestselling software testing book of all time! **Second Edition** Cem Kaner Jack Falk Hung Quoc Nguyen





Copyrighted Material Auerbach Publications

Manage Software Testing



Copyrighted Materia

Peter Farrell-Vinay



Code Ahead by Yegor Bugayenko

TL;DR It's an autobiographical fiction book IL; UK IT'S AN AUTODIOGRAPHICAL FICTION DOOK about a software architect who is involved in about a software architect who is involved in programming, DevOps, testing, quality assurance and control, corporative management, project management, and...love.



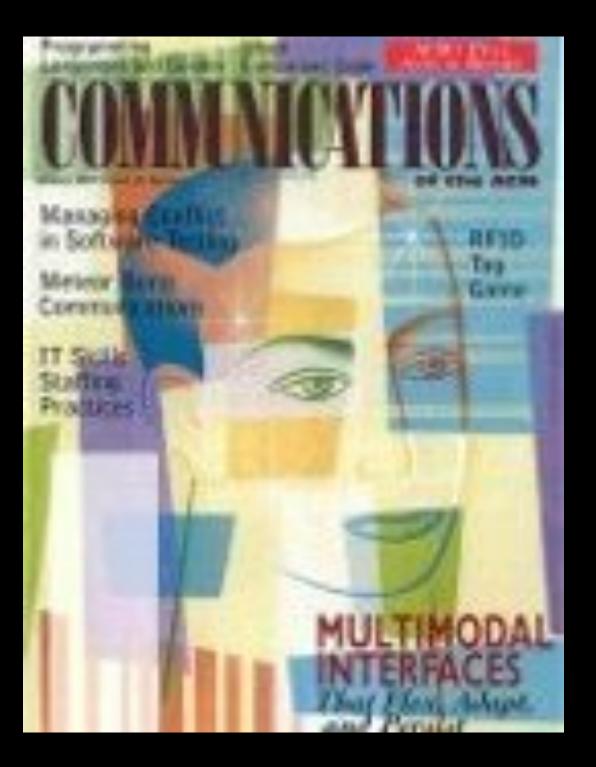


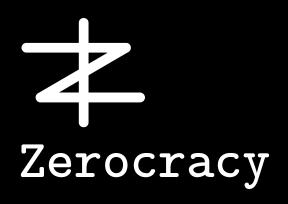


Testers are not second-class citizens.









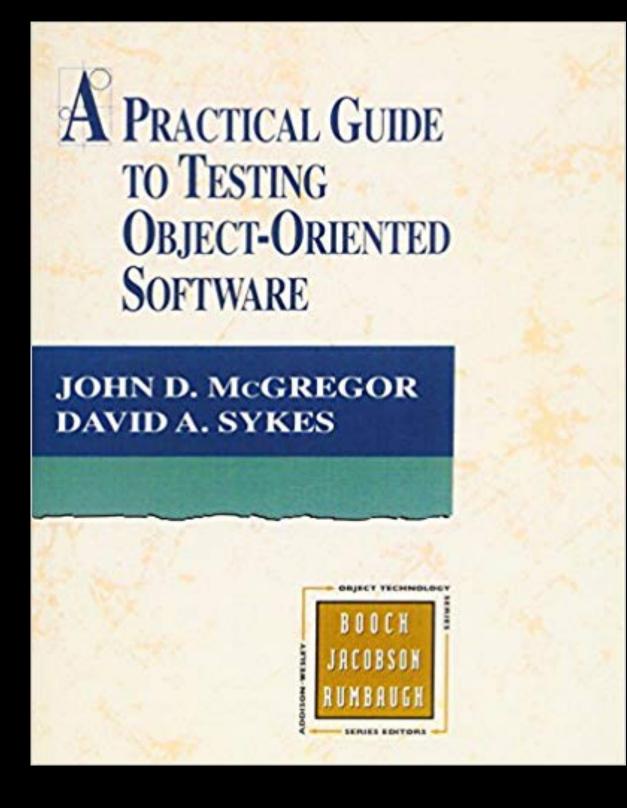
Cynthia Cohen: "The lack of

- status and support makes the
- tester's job more difficult and
- time consuming, as the struggle
- for recognition becomes part of

the job itself"

Managing Conflict in Software Testing, Communications of the ACM, Volume 47, Issue 1, 2004





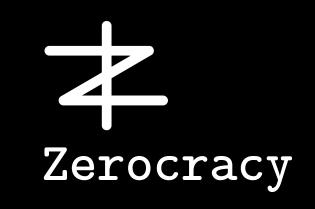


- John D. McGregor: "Being a good tester is <u>harder</u> than being a good developer
- because testing requires not only a very good understanding of the development
- process and its products, but it also demands an ability to <u>anticipate</u> likely faults and errors."











Seniority

More!





Middle

Junior



Tester

Senior



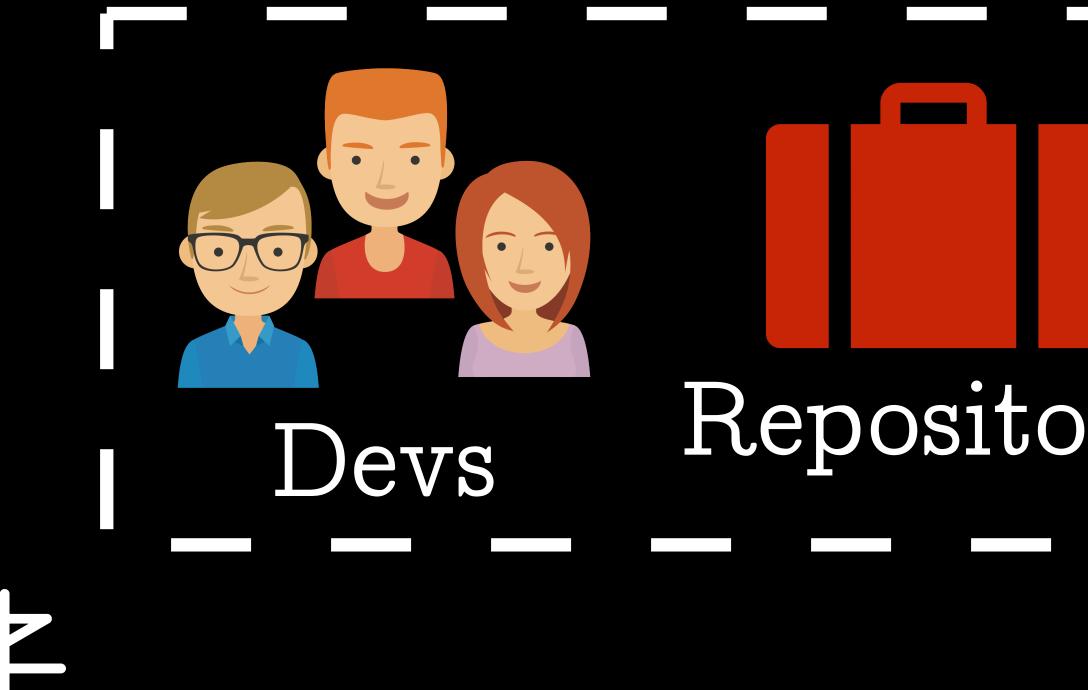




Testers don't tell us when to release.







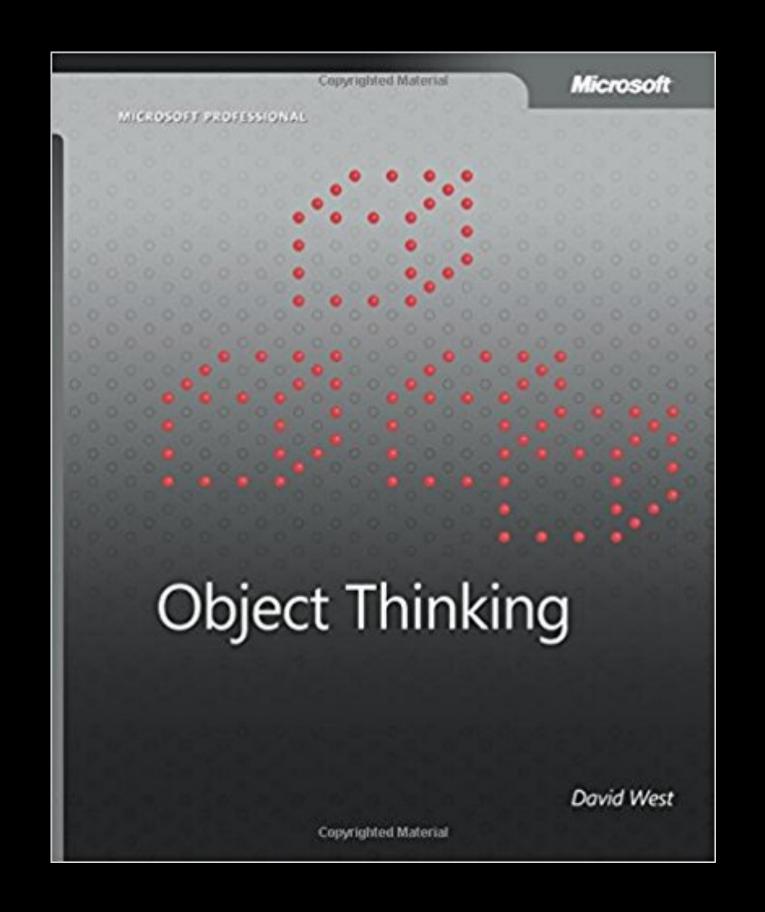




Testers

Repository Staging Production

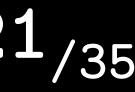




acceptable."



David West: "Software is released for use, not when it is known to be correct, but when the <u>rate</u> of discovering errors slows down to one that management considers









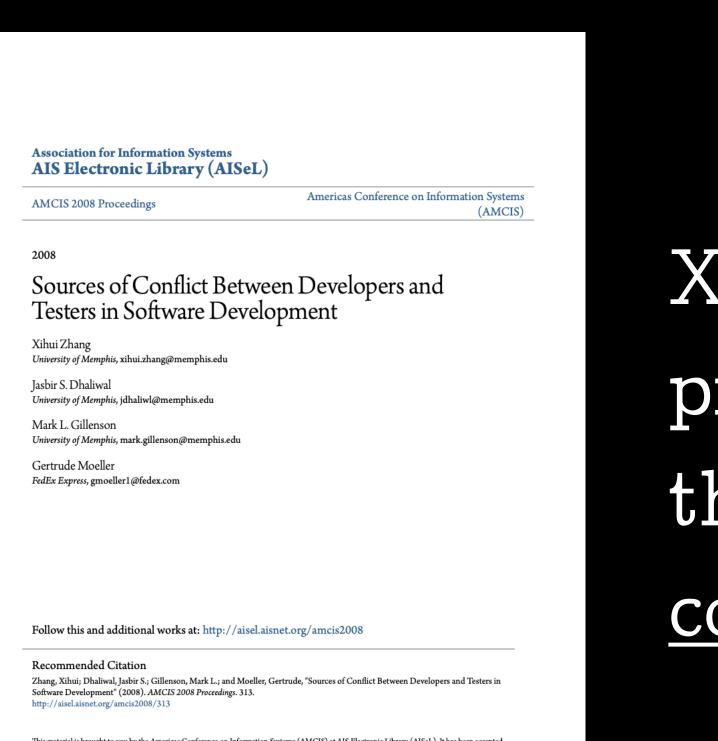


PM



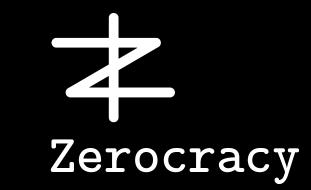
Testers





This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2008 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

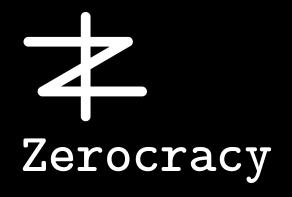
conflict."



- Xihui Zhang: "The software testing
- process is inherently adversarial, setting the stage for inevitable <u>developer-tester</u>

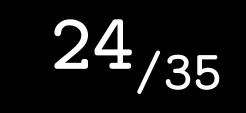


GitFlow by Vincent Driessen

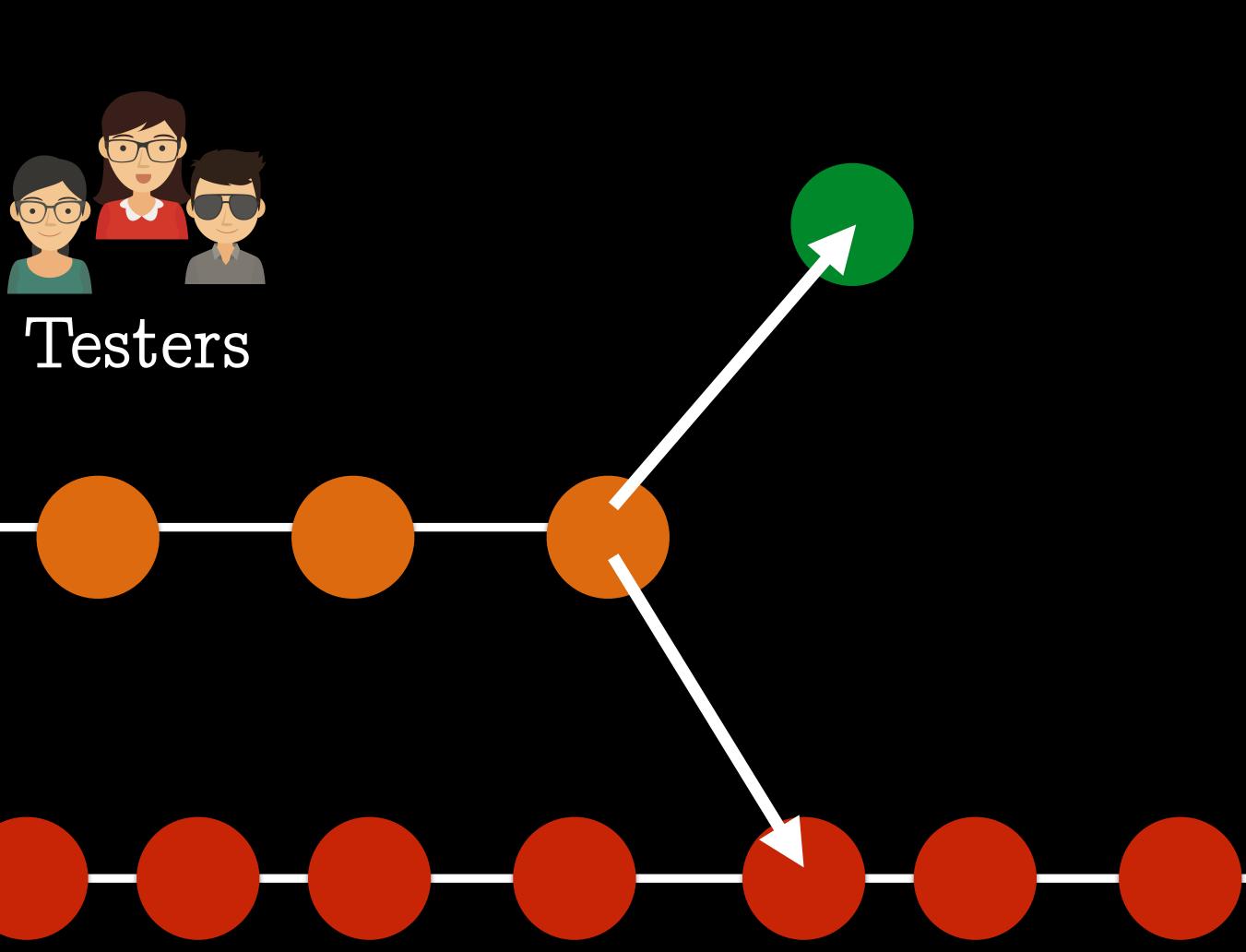




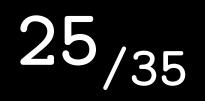
https://nvie.com/posts/a-successful-git-branching-model/









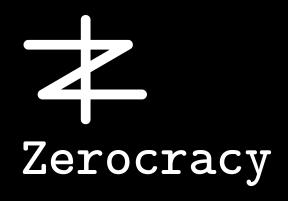


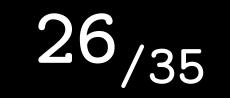


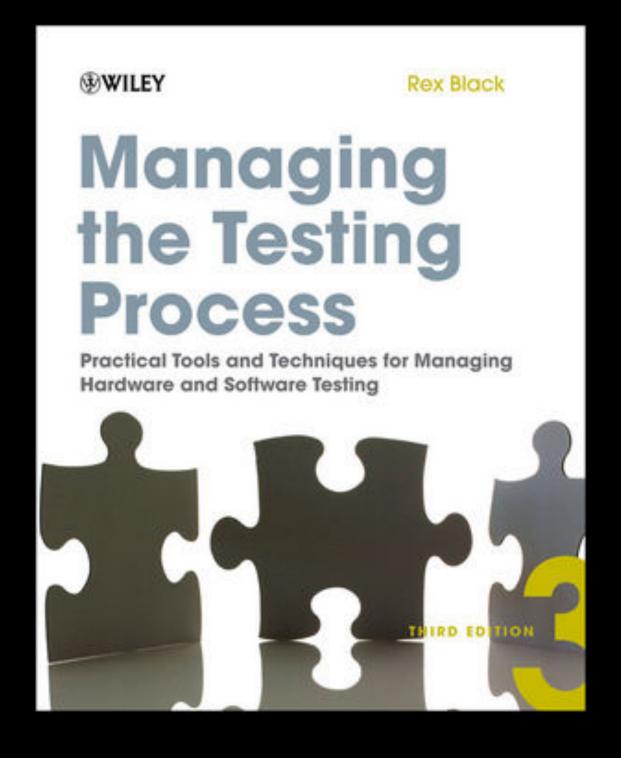


Testing is finished when enough bugs are found.









problems..."



- Rex Black: "Suppose, though, that you could estimate the total number of bugs
- in the system under test. Or, perhaps you could measure the bug-finding
- effectiveness of your test system. There are three techniques for solving these





Time

Scripts









Guess





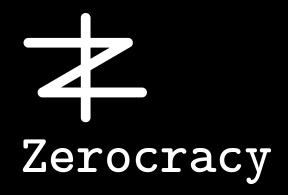
Experience







bugs they find.



Testers must be rewarded for the







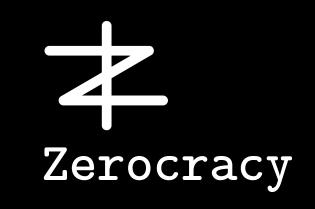
- Yegor Bugayenko: "The best way to
- motivate testers to find more and better
- bugs is to pay them for each one."





Price

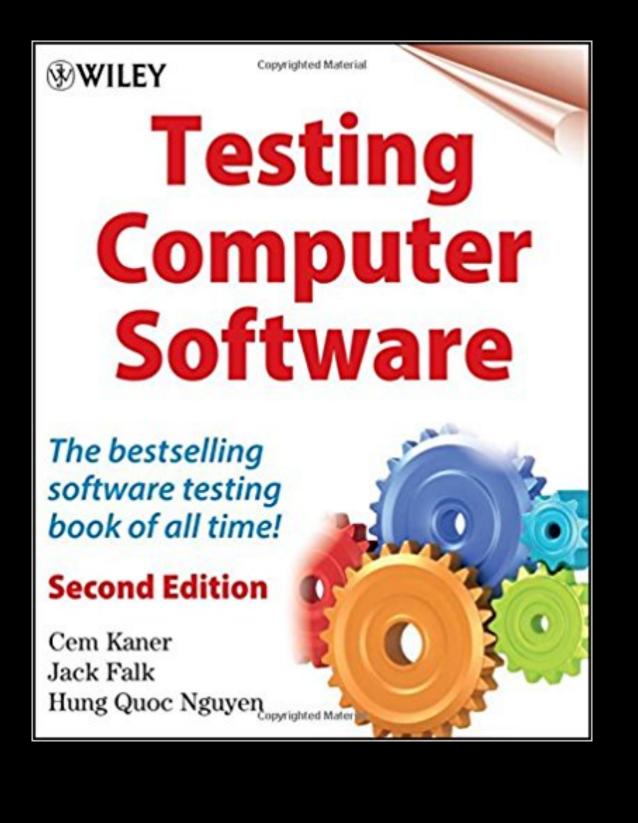




Quality

Time





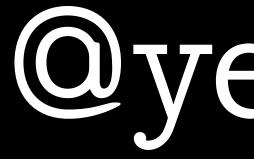
bugs fixed."



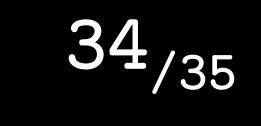
- Cem Kaner: "The best tester isn't the one who finds the most bugs or who embarrasses the most programmers. The
- best tester is the one who gets the most







Oyegor256



@yegor256news

