

# CS 5154: Software Testing

## Introduction

Owolabi Legunsen

# On the state of software quality

## The New York Times *Airline Blames Bad Software in San Francisco Crash*



## GOOGLE SELF-DRIVING CAR CAUSED FREEWAY CRASH AFTER ENGINEER MODIFIED ITS SOFTWARE

BY JASON MURDOCK ON 10/17/18 AT 11:34 AM

Newsweek



~9% of 2017  
US GDP

## Report: Software failure caused \$1.7 trillion in financial losses in 2017



Software testing company Tricentis found that retail and consumer technology were the areas most affected, while software failures in public service and healthcare were down from the previous year.

By Scott Matteson | January 26, 2018, 7:54 AM PST

## Hard Questions Raised When A Software 'Glitch' Takes Down An Airliner **Forbes**

# Why learn software testing?

- Testing is usually the last line of defense against bugs
- Testing is widely used by developers for finding bugs
- **It will be your job to test software!**

# What this course is about

- Systematic, organized approaches to testing
- You will learn
  - how to design and write high quality tests
  - cutting-edge testing techniques and tools
  - how to apply techniques and tools to (real) code

# Prerequisites

- You will likely struggle if you didn't satisfy the prereqs
- We will use knowledge from
  - Discrete math
  - Java
  - Version control with Git, GitHub
  - Prog. languages, software engineering, or compilers
  - ...

# CS 5154: theory meets practice

- We'll use some basic theory to answer practical questions
  - What inputs should we choose?
  - Have we tested software sufficiently?
  - How good are our tests?
  - ...

# What this course is *not* about

5150

- How to develop software for clients
- Basic software engineering knowledge and skills
- A tutorial on Company X's latest testing tools

# Some praise for this course

I work with a large enterprise software company. One of my main projects is to write tests. By applying the concepts that I learned in this class, I was able to write a strong test suite which received high-praise from my manager and team. This class had a direct effect on my efficacy in writing that test suite. So, I want to thank you for teaching me these concepts and always mentioning the practical applications of the concepts.



# Meet the course staff

- **Instructor:** Owolabi Legunsen
  - Office Hours: Tue, Thu 2:30–3:30pm, Gates 442A
- **TAs**
  - Ayaka Yorihiro. OH: TBD, TBD
  - Richard Palmer. OH: TBD, TBD
  - Christian Donovan. OH: TBD, TBD
  - Michael Ye. OH: TBD, TBD

# Course webpage

- <https://www.cs.cornell.edu/courses/cs5154/2022fa>
  - Syllabus
  - Announcements
  - All lecture materials
  - Handy links
  - Where to get help @ Cornell

# Course Communication

- Course email: [cs5154-staff@cornell.edu](mailto:cs5154-staff@cornell.edu)
  - Send all questions, comments, complaints, etc.
  - We plan to respond within 24 hours
  - Plan ahead, don't wait till due dates!
- We will make announcements on Canvas or CMS

# Your grade will be based on...

<b>Homework assignments</b>	<b>30%</b>
<b>Readings</b>	<b>5%</b>
<b>In-class participation activities</b>	<b>5%</b>
<b>Pre-announced in-class quizzes</b>	<b>15%</b>
<b>Prelim 1 (10/6)</b>	<b>20%</b>
<b>Prelim 2 (11/15) or Final exam (TBD)</b>	<b>25%</b>

# Working in a team

- Software engineering (including testing) is a team sport
- Team sizes: ~3
- Form your own teams

Questions on course logistics?

?

# Now it's your turn



- Form groups of two or three
- Introduce yourselves
- Brainstorm:

**If testing is so widely used, why are there still so many software failures?**

# What did your group discuss?

- Tests do not cover enough code
- Developers may have "tunnel vision"
- Lack of test-driven development
- Treating testing as afterthought
- Hard to predict all situations and inputs for running code
- nondeterministic environment
- differences in test vs. production env



# Is findQuotient correct?

```
// Computes remainder and quotient of  $x \div y$   
// where  $x$  and  $y$  are integers
```

```
public Pair findQuotient(int x, int y){  
    int remainder = x;  
    int quotient = 0;  
    while (y <= remainder) {  
        remainder = remainder - y;  
        quotient = 1 + quotient;  
    }  
    return new Pair(remainder, quotient);  
}
```

# Is findQuotient correct?

```

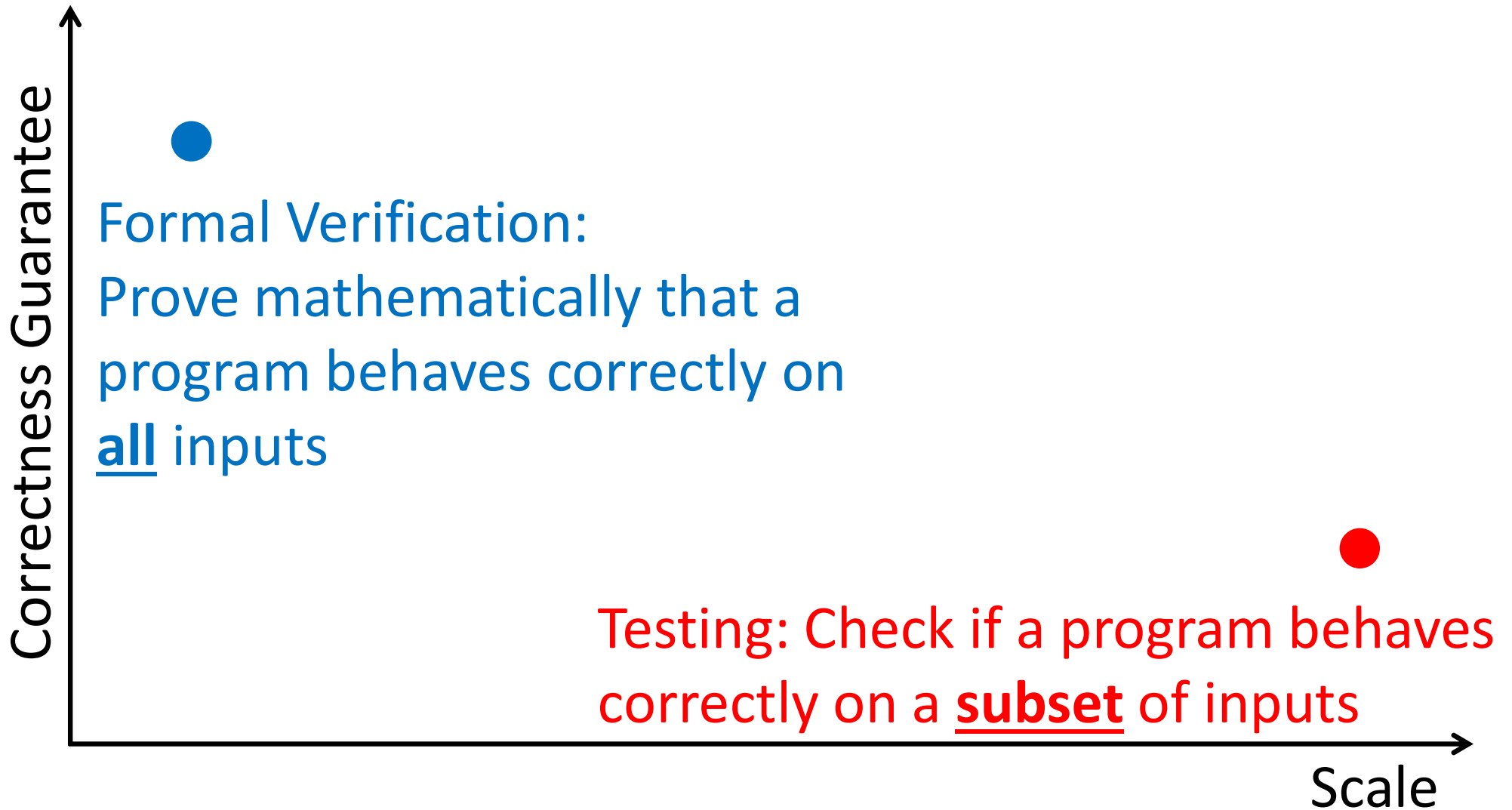
Pair findQuotient(int x,int y){
  int r = x;
  int q = 0;
  while (y <= r) {
    r = r - y;
    q = 1 + q;
  }
  return new Pair(r, q);
}

```

TABLE III

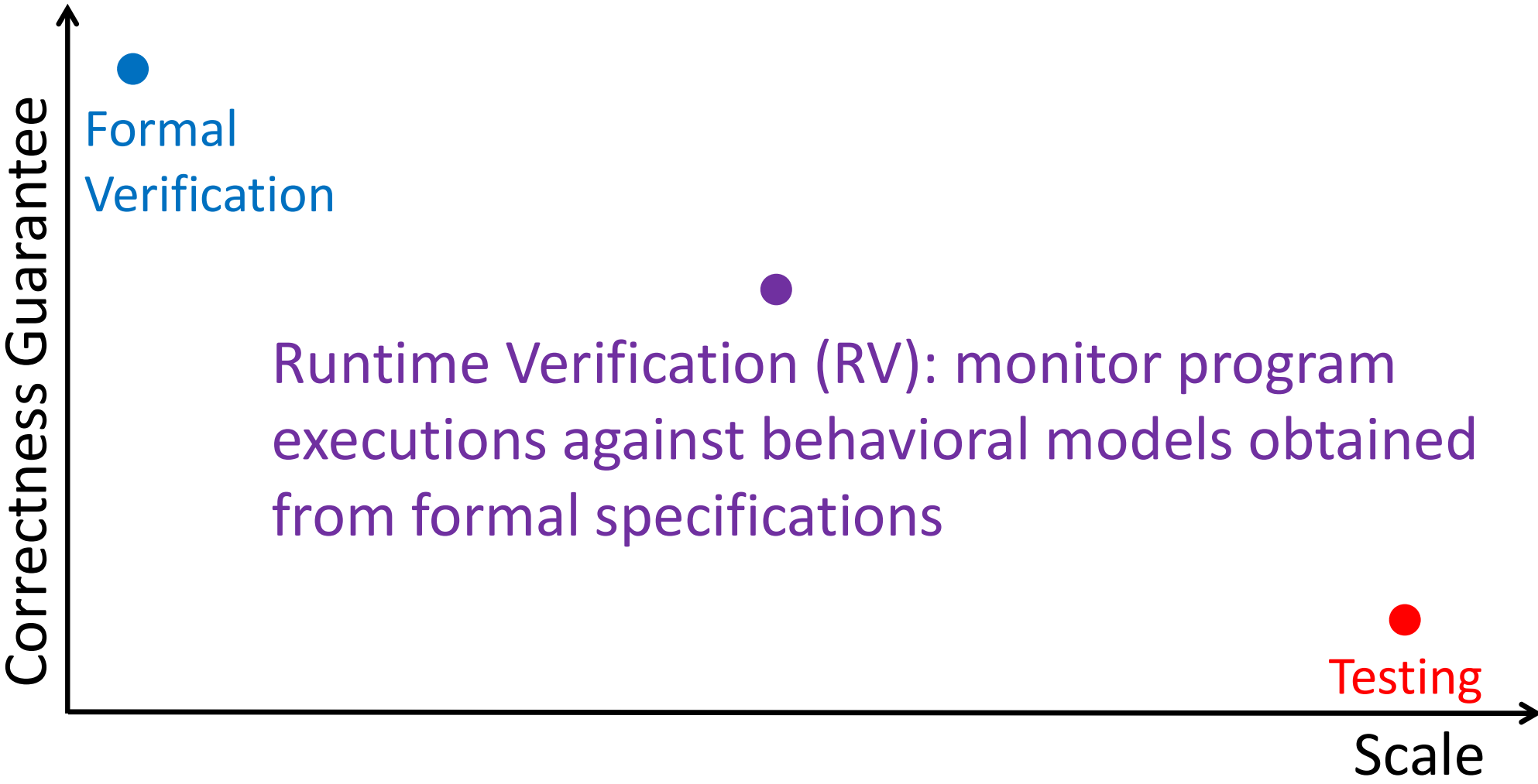
Line number	Formal proof	Justification
1	$\text{true} \supset x = x + y \times 0$	Lemma 1
2	$x = x + y \times 0 \{r := x\} x = r + y \times 0$	D0
3	$x = r + y \times 0 \{q := 0\} x = r + y \times q$	D0
4	$\text{true} \{r := x\} x = r + y \times 0$	D1 (1, 2)
5	$\text{true} \{r := x; q := 0\} x = r + y \times q$	D2 (4, 3)
6	$x = r + y \times q \wedge y \leq r \supset x = (r-y) + y \times (1+q)$	Lemma 2
7	$x = (r-y) + y \times (1+q) \{r := r-y\} x = r + y \times (1+q)$	D0
8	$x = r + y \times (1+q) \{q := 1+q\} x = r + y \times q$	D0
9	$x = (r-y) + y \times (1+q) \{r := r-y; q := 1+q\} x = r + y \times q$	D2 (7, 8)
10	$x = r + y \times q \wedge y \leq r \{r := r-y; q := 1+q\} x = r + y \times q$	D1 (6, 9)
11	$x = r + y \times q \{ \text{while } y \leq r \text{ do } (r := r-y; q := 1+q) \} \neg y \leq r \wedge x = r + y \times q$	D3 (10)
12	$\text{true} \{ ((r := x; q := 0); \text{ while } y \leq r \text{ do } (r := r-y; q := 1+q)) \} \neg y \leq r \wedge x = r + y \times q$	D2 (5, 11)

# The two approaches in a picture



This graph uses notional units

# My lab: enhance testing \w formal methods



Runtime Verification (RV): monitor program executions against behavioral models obtained from formal specifications

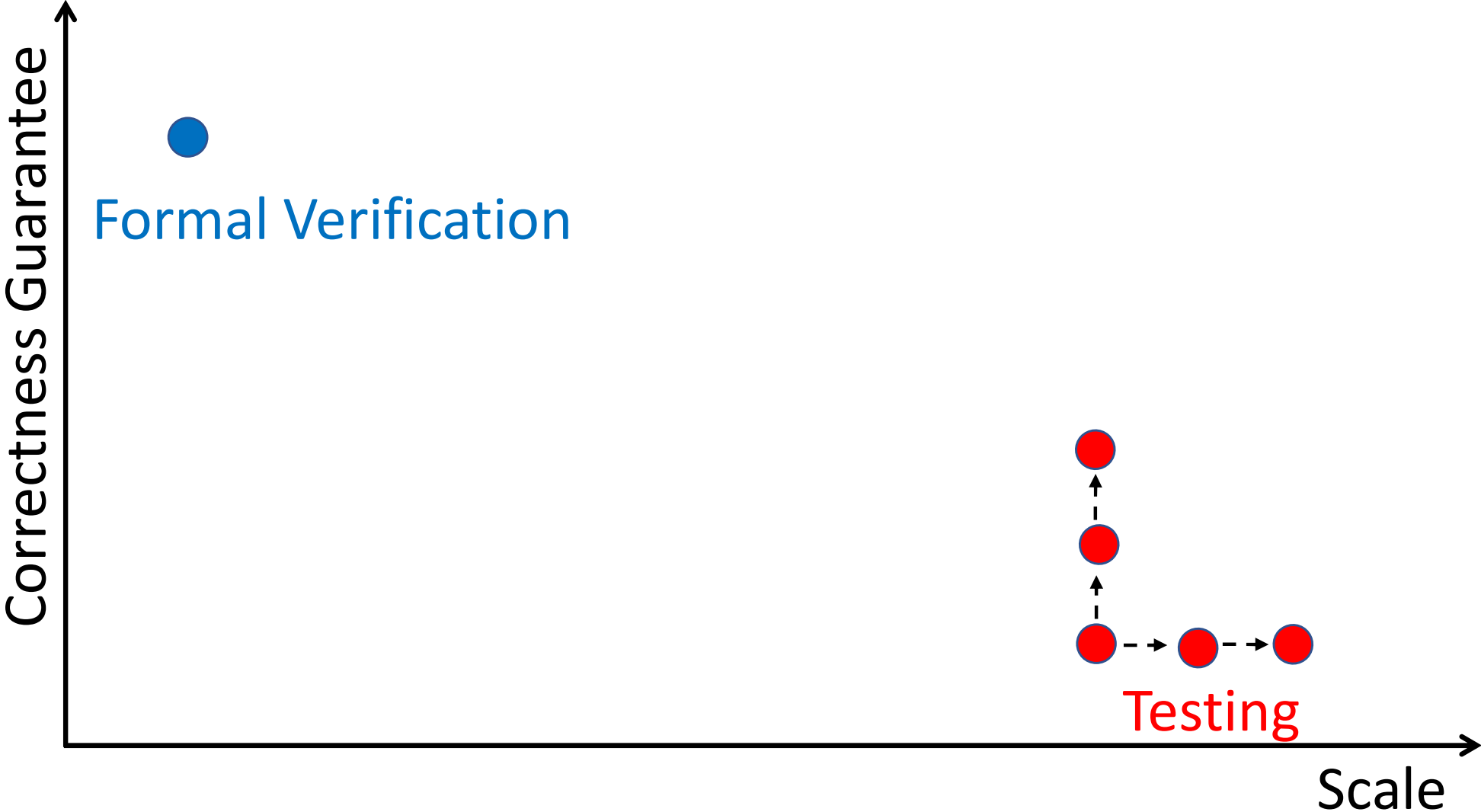
# Software testing from 3000 miles



Pros	Cons
Easier for most developers	Can miss bugs
Scales well in practice	Oracle generation is hard
Uses developer insights	High maintenance costs

This graph uses notional units

# The essence of this course



How to improve the guarantees that testing provides?

How to make testing scale even better?

# CS5154 is organized into six themes

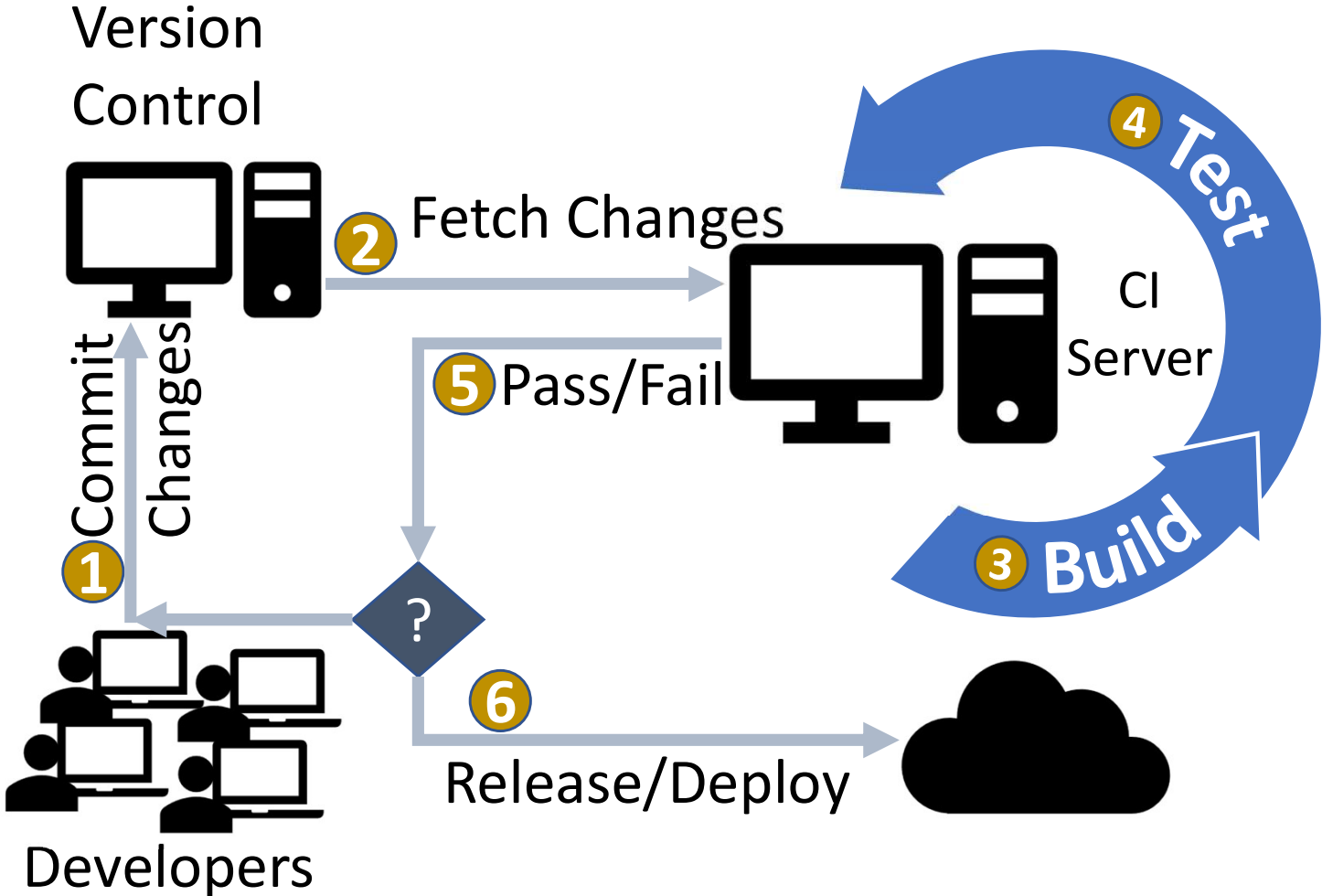
1. How to automate the execution of tests?
2. How to design and write high-quality tests?
3. How to measure the quality of tests?
4. How to automate the generation of tests?
5. How to reduce the costs of running existing tests?
6. How to deal with bugs that tests reveal?

# Theme 1: test automation

- The xUnit paradigm and the JUnit framework
- Parameterized Unit Tests
- The Maven build system
- Continuous Integration (??)



# Automation: Continuous Integration (CI)



# Theme 2: test design

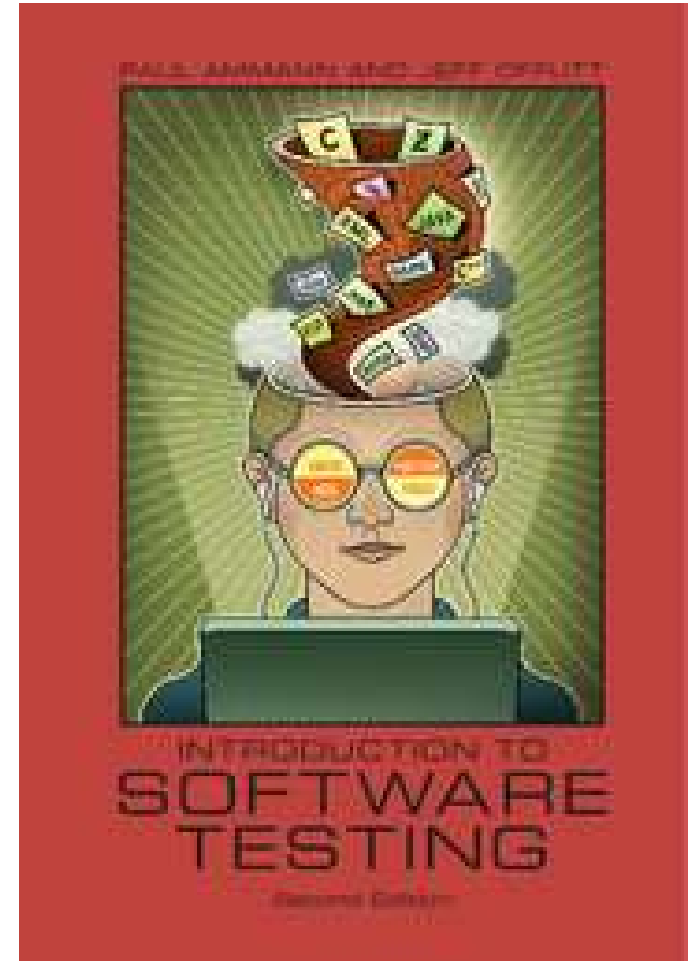
- Goal: systematically derive tests that increase the chance to reveal bugs
- We will use only four models of software
  - Input space
  - Graph structure
  - Logic conditions
  - Syntax

# Theme 3: checking test quality

- Question: How do you test your tests?
- We'll learn about mutation testing

# Required Text for Themes 1 – 3

- Readings will be assigned from book
  - Starting soon
- Copies on reserve @ Uris library
- Copies available via the Bookstore



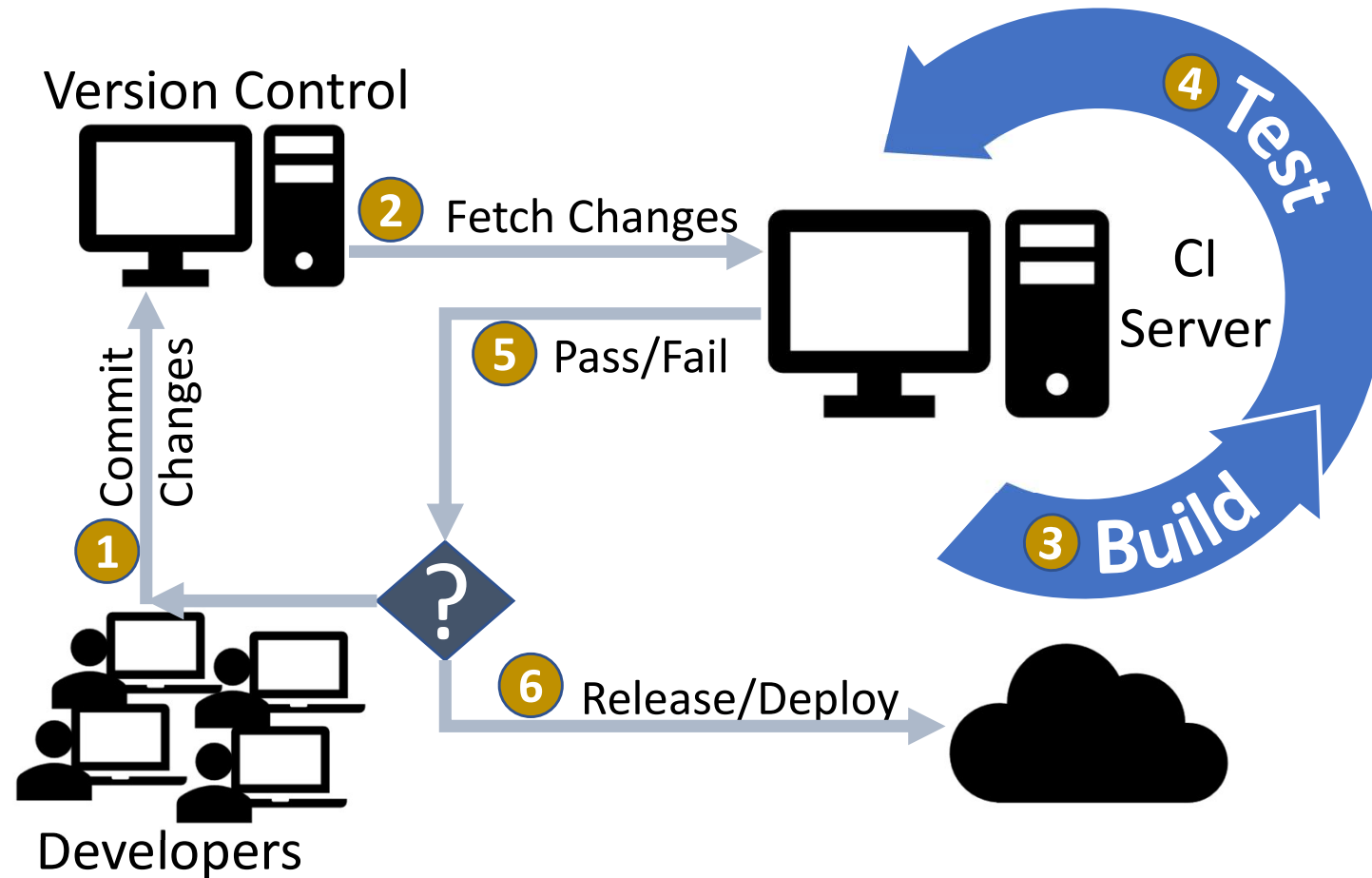
# Theme 4: automatic test generation

- Problem: writing tests is very expensive
  - 75% of development effort at Microsoft (??)
  - “customers pay for features, not tests” 😞
- Goal: learn about automated test generation

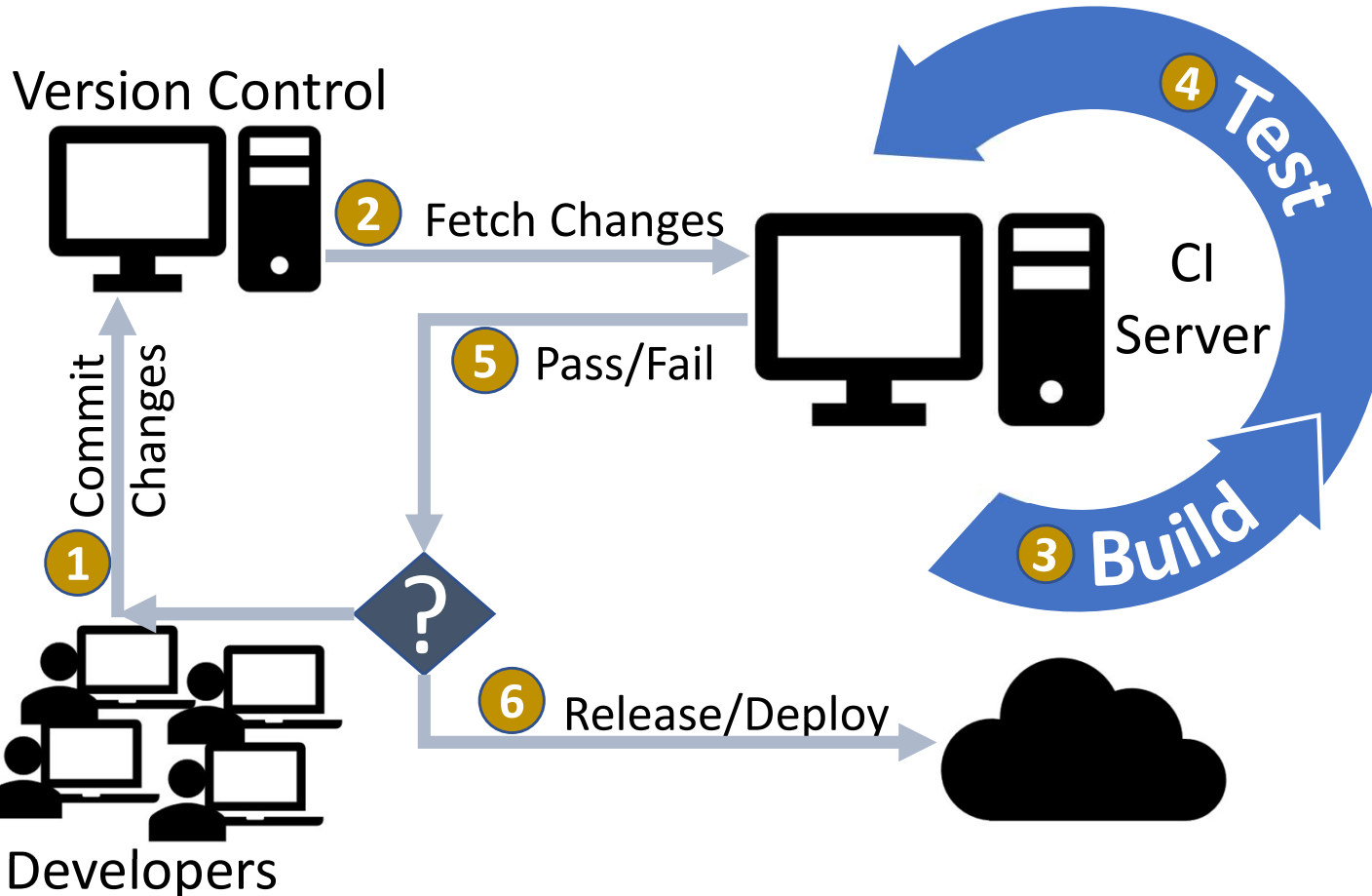
# Quiz: How many CI cycles per day?

At companies like Microsoft, Facebook, Google?

By individual developers?



# Tests are re-run very frequently



Cycles per day:

- Facebook: 60K\*
- Google: 17K
- HERE: 100K
- Microsoft: 30K
- Single OSS: up to 80

Releases per day

- Etsy: 50

\* Android only; Facebook: <https://bit.ly/2CAPvN9> ; Google: <https://bit.ly/2SY4rR> ;  
HERE: <https://oreil.ly/2T0EyeK> ; Microsoft: <https://bit.ly/2HgjUpw> ; Etsy: <https://bit.ly/2liSOJP> ;

# Re-running tests is very costly



↑ no. of changes per day \* ↑ no. of tests

⇒ quadratic increase in test execution time

- 75+ million tests run per day
- 20+ revisions per minute



# Re-running tests is very costly (2)

test execution time      number of tests



~5min

1667



~10min

641534



~45min

1296



~45min

361



~45min

631



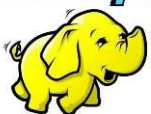
~4h

4975



~17h

8663



Run many times each day

# Did you deal with long-running tests?

- Q: what's the longest your tests took to run?
  
  
  
  
  
  
  
  
  
  
- Q: what do you do while waiting for long-running tests?

# Theme 5: regression testing

- Re-running tests after every code change is expensive
- Goal: learn cutting-edge techniques and tools for making regression testing more efficient and effective

# Theme 6: dealing with bugs

- Context: your tests revealed a bug. Now what?
- If time permits, we'll learn about
  - Debugging
  - Bug Advocacy

Questions on course content?

?

# A review of today's class

- Learning outcomes, course content, and logistics
- High-level introduction to Software Testing
- Comparison of testing with another QA approach
- Whirlwind tour of how this course is organized

# Next class...

- Testing Foundations