Construction and Verification of Software

2019 - 2020

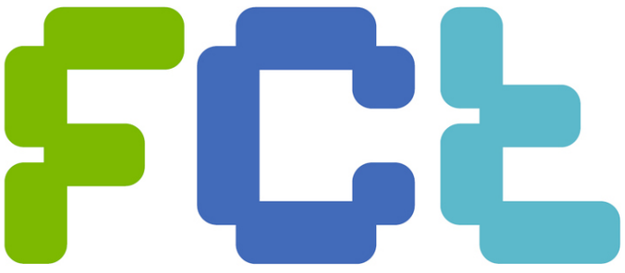
MIEI - Integrated Master in Computer Science and Informatics

Consolidation block

Lecture 1 - Introduction and Motivation

João Costa Seco (joao.seco@fct.unl.pt)

based on previous editions by **Luís Caires** (lcaires@fct.unl.pt)



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Bug? a small exercise…

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = 0;
        for(int i = 0; i < a.length; i++)
            if( a[i] >= m ) m = a[i];
        return m;
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = 0;
        for(int i = 0; i < a.length; i++)
            if( a[i] >= m ) m = a[i];
        return m;
    }
}
```

```
public class MyCollectionTest {

    @Test
    public void test1() {
        assertEquals(1, MyCollection.max(new int[]{1,2,3,4,5,6},1));
        assertEquals(2, MyCollection.max(new int[]{1,2,3,4,5,6},2));
        assertEquals(4, MyCollection.max(new int[]{1,2,3,4,5,6},4));
        assertEquals(6, MyCollection.max(new int[]{1,2,3,4,5,6},6));
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = 0;
        for(int i = 0; i < N; i++)
            if( a[i] >= m ) m = a[i];
        return m;
    }
}
```

```
public class MyCollectionTest {

    @Test
    public void test1() {
        assertEquals(1, MyCollection.max(new int[]{1,2,3,4,5,6},1));
        assertEquals(2, MyCollection.max(new int[]{1,2,3,4,5,6},2));
        assertEquals(4, MyCollection.max(new int[]{1,2,3,4,5,6},4));
        assertEquals(6, MyCollection.max(new int[]{1,2,3,4,5,6},6));
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = 0;
        for(int i = 0; i < N; i++)
            if( a[i] >= m ) m = a[i];
        return m;
    }
}
```

```
public class MyCollectionTest {

    @Test
    public void test2() {
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},1));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},2));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},4));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},6));
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = Integer.MAX_VALUE;
        for(int i = 0; i < N; i++)
            if( a[i] >= m ) m = a[i];
        return m;
    }
}
```

```
public class MyCollectionTest {

    @Test
    public void test2() {
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},1));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},2));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},4));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},6));
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = Integer.MIN_VALUE;
        for(int i = 0; i < N; i++)
            if( a[i] >= m ) m = a[i];
        return m;
    }
}
```

```
public class MyCollectionTest {

    @Test
    public void test2() {
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},1));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},2));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},4));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},6));
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
public class MyCollection {

    static int max(int[] a, int N) {
        int m = a[0];
        for(int i = 1; i < N; i++)
            if( a[i] >= m ) m = a[i];
        return m;
     }
}
```

```
public class MyCollectionTest {

    @Test
    public void test2() {
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},1));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},2));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},4));
        assertEquals(-1, MyCollection.max(new int[]{-1,-2,-3,-4,-5,-6},6));
    }
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
method max(a:array<int>, N:int) returns (M:int)
{
  var m:int := a[0];
  var i:int := 0;
  while i < N {
    if a[i] >= m { m := a[i]; }
    i := i + 1;
  }
  return m;
}
```

Can't compile: Request compile failed with message:
max.dfy(3,16): Error: index out of range
max.dfy(6,8): Error: index out of range

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
method max(a:array<int>, N:int) returns (M:int)
  requires 0 < N <= a.Length
{
  var m:int := a[0];
  var i:int := 0;
  while i < N
    decreases N - i
  {
    if a[i] >= m { m := a[i]; }
    i := i + 1;
  }
  return m;
}
```

Bug?

- Determinar o máximo dos primeiros N elementos de um array.

```
method max(a:array<int>, N:int) returns (M:int)
  requires 0 < N <= a.Length
  ensures forall i :: 0 <= i < N ==> M >= a[i]
{
  var m:int := a[0];
  var i:int := 0;
  while i < N
    decreases N - i
    invariant 0 <= i <= N
    invariant forall j :: 0 <= j < i ==> m >= a[j]
  {
    if a[i] >= m { m := a[i]; }
    i := i + 1;
  }
  return m;
}
```

Construction and Verification of Software

This course covers principles, methods, techniques and tools for the dependable and trustworthy construction and validation of software systems, ensuring as much as possible the absence of programming errors ("bugs"), with a focus on CONCURRENCY and SAFETY.

Project based learning using specialised techniques and tools.

Syllabus

- **Verified Software Construction**
  - Assertion methods and Hoare and Separation Logic; Assertion Inference; Abstract and Behavioural types; Representation Invariants; Abstract interpretation; Model-checking.
  - Hands-on exercises / final project using verification tools (Dafny, Verifast, INFER).
- **Concurrent Programming**
  - Sharing, confinement, ownership. Control of interference. Reasoning about concurrent code with monitors and locks based on resource invariants. Construction of concurrency control code from behavioural specs.
- **Software Testing**
  - Test selection and test generation; Model-based testing; Fault-based testing. Property based testing; Symbolic execution; Automated testing; Tools.

Objectives

- **Static Verification of Software**
  - understand the principles and know how to use assertion methods in practice to specify, reason about, and verify software
- **Dynamic Verification of Software**
  - understand the principles and methods for software testing.
- **ADTs and Concurrent Programming**
  - write correct concurrent programs and ADTs
  - understand ADT programming methodologies
  - understand concurrent programming methodologies

Bibliografia

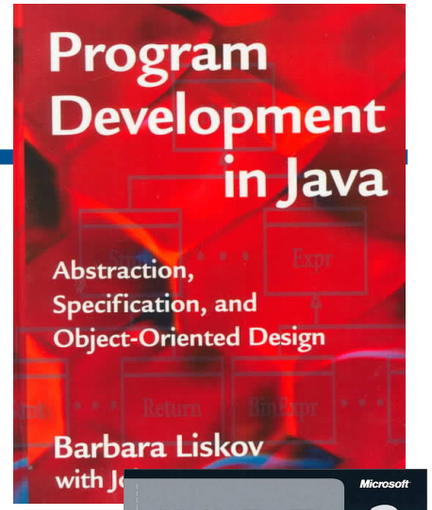
Program Development In Java:
Abstraction, Specification, and Object-Oriented Design.
Barbara Liskov (with John Guttag); MIT Press.

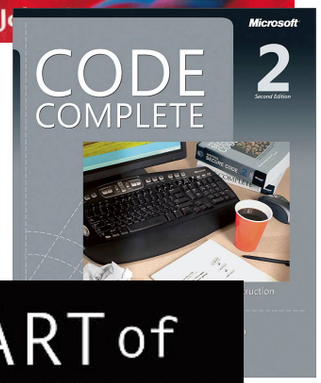
Code Complete:
A Practical Handbook of Software Construction, Second Edition.
Steve McConnell, Microsoft Press.

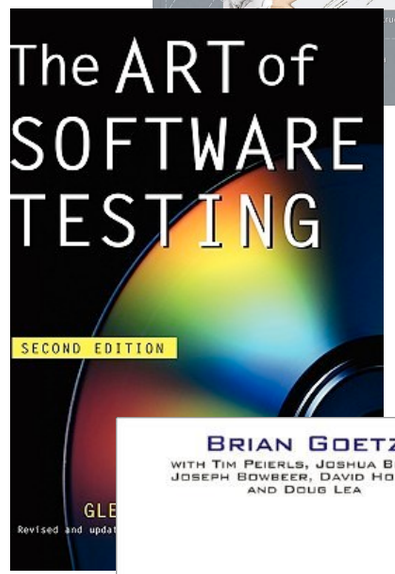
The Art of Software Testing, Second Edition
Glenford Myers, Corey Sandler, Tom Badgett

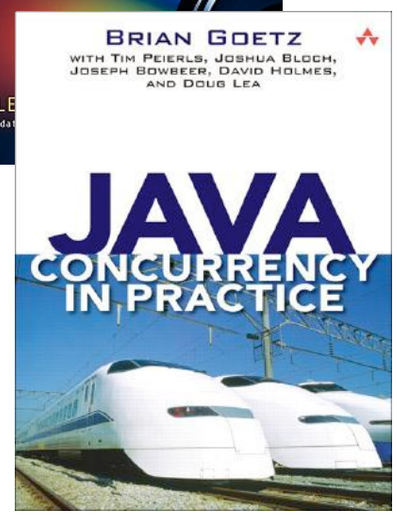
Java Concurrency in Practice,
Goetz et al. Addison-Wesley, 2006.

Tutorials for Dafny and Verifast









Logistics and Evaluation

- ~12 Lectures
  - Midterm (w6) and Final Test (w12) — to be determined
- Lab Sessions (3 Lab classes)
  - Teams of 1-2 students
  - Handouts - Test generation, Dafny exercises
  - Project (two deliveries)
    Development and verification of a concurrent system - Verifast
- Communication channel: — to be determined soon
- Evaluation details not yet final in CLIP, will be updated

What's the True Cost of a Software Bug?

What's the True Cost of a Software Bug?

- A software bug can have direct impact in time and revenue and also indirect costs in user loyalty and reputation of a company.

"the cost to fix an error found after product release was 4 to 5 times higher than if it's uncovered during the design phase, and up to 100 more expensive than if it's identified in the maintenance phase." (IBM)

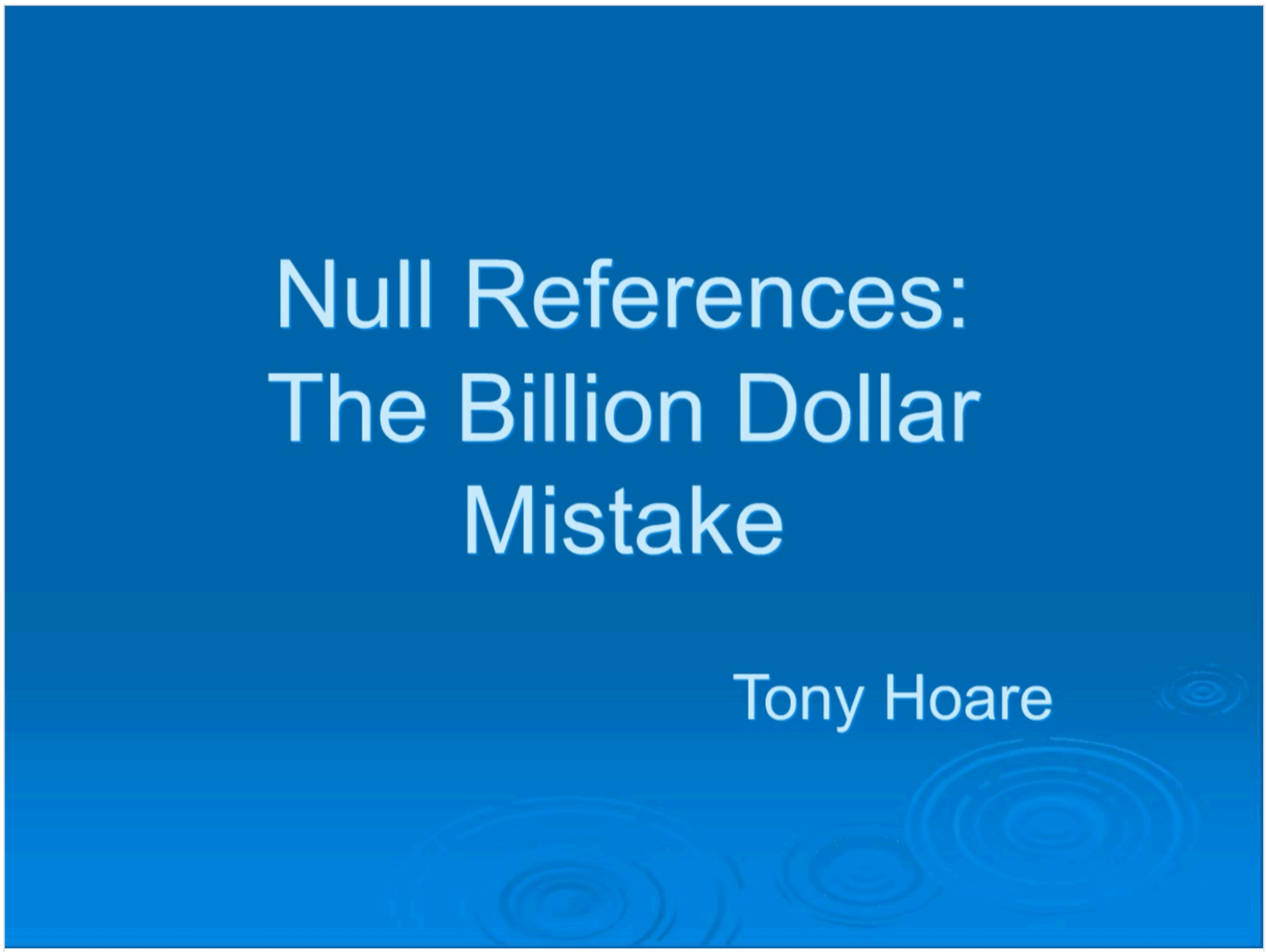
https://crossbrowsertesting.com/blog/development/software-bug-cost/

Null References: The Billion Dollar Mistake

LIKE 6

View Presentation Speed: 1X 1.5X 2X

01:01:58



Summary

Tony Hoare introduced Null references in ALGOL W back in 1965 "simply because it was so easy to implement", says Mr. Hoare. He talks about that decision considering it "my billion-dollar mistake".

REPORT: SOFTWARE FAILURES COST $1.1 TRILLION IN 2016

March 8, 2017 Michael Joseph

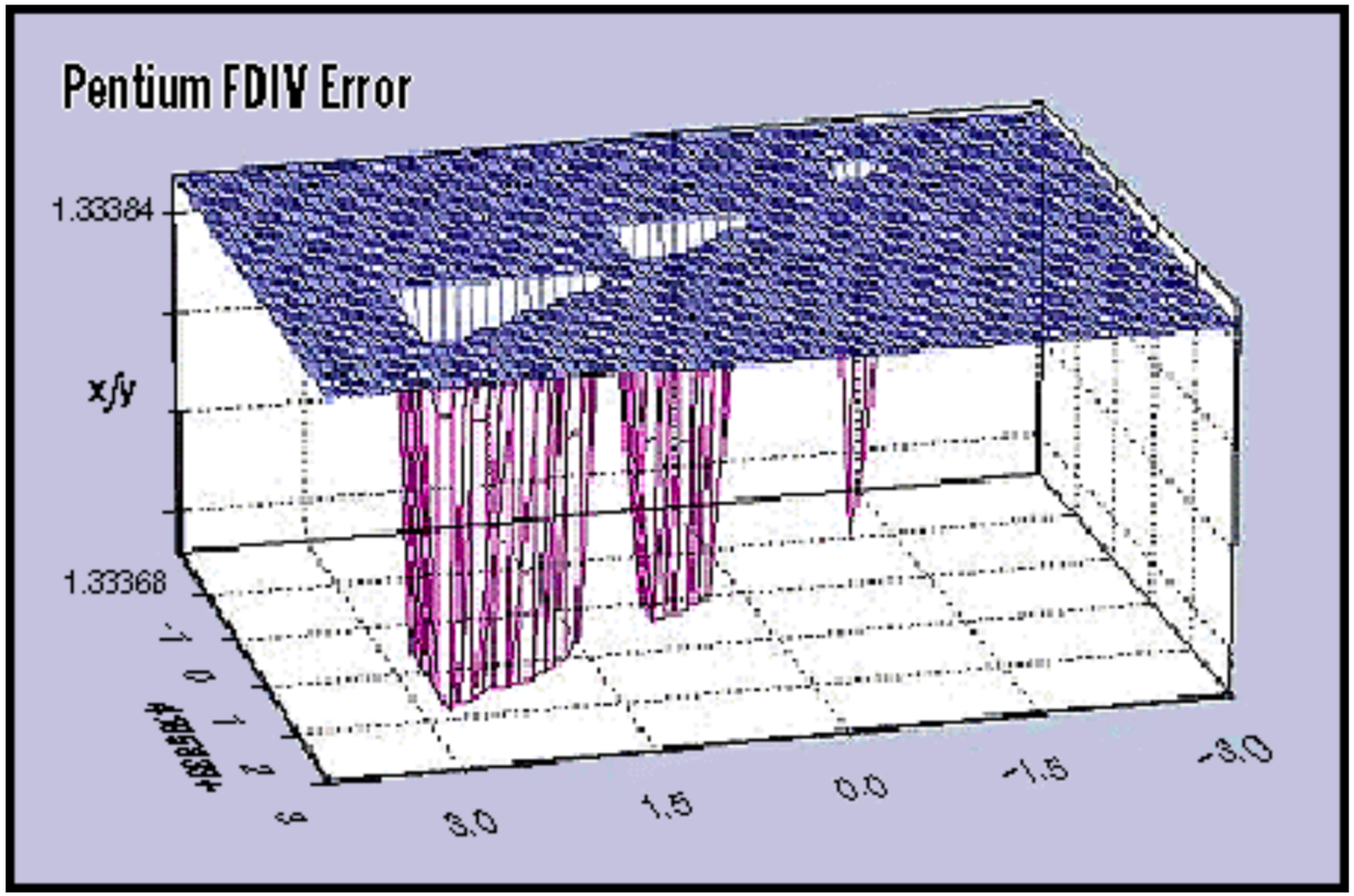
Not really a new thing

- Byte Magazine 1995

Hardware bugs are even worse

- Byte Magazine March 1995

Hardware Bugs are even worse



A 3-D plot of the ratio 4195835/3145727 calculated on a Pentium with FDIV bug. The depressed triangular areas indicate where incorrect values have been computed. The correct values all would round to 1.3338, but the returned values are 1.3337, an error in the fifth significant digit. Byte Magazine, March 1995.

https://www.cs.earlham.edu/~dusko/cs63/fdiv.html

Too easy to make flawed software

United Airlines

A first-class cock up

Feb 16th 2015, 16:55 BY B.R.

Timekeeper Like 3.8k Tweet 68

WHEN Matt and Emil, a couple of expat Americans living in London, were invited to be groomsmen at a friend's wedding in New York, they feared they would not be able to afford to make the transatlantic trip. And then fortune intervened. They heard about a glitch on United Airlines' British website. A computer error meant that the airline was offering trips across the pond for just £52 ($80), as long users selected to pay in Danish kroner. Even more remarkably, the tickets were for the first-class cabin.

SOCIEDADE

A justiça num verdadeiro «estado de Citius»

Repórter TVI verificou com os próprios olhos o caos vivido nos tribunais. Programa informático que suporta a atividade judicial está sem funcionar há mais de 30 dias

Por: Redação / Cláudia Rosenbuch | 29 de Setembro de 2014 às 22:59

Topic: *Security* Follow via:

Microsoft reveals Windows vulnerable to FREAK SSL flaw

Summary: *Redmond has said that the FREAK security flaw is found in versions of its Windows operating system from Windows Server 2003, Windows Vista, and higher.*

By Chris Duckett | March 6, 2015 -- 03:12 GMT (03:12 GMT)

Follow @dobes 2,273 followers Get the ZDNet Announce UK newsletter now

Comments 74 Share on Facebook 171 Tweet 247 Share 89 more +

The FREAK security bug that allows attackers to conduct man-in-the-middle attacks on Secure Sockets Layer (SSL) and Transport Layer Security (TLS) connections encrypted using an outmoded cipher has claimed another victim. This time, it is Microsoft's Secure Channel stack.

"Microsoft is aware of a security feature bypass vulnerability in Secure Channel (Schannel) that affects all supported releases of Microsoft Windows," the company said in a security advisory. "The vulnerability facilitates exploitation of the publicly disclosed FREAK technique, which is an industry-wide issue that is not specific to Windows operating systems."

Although Microsoft Research was part of the team to uncover FREAK alongside European cryptographers, Redmond chose not to reveal Windows as vulnerable until today.

"When this security advisory was originally released, Microsoft had not received any information to indicate that this issue had been publicly used to attack customers," the company said.

What's Hot on ZDNet

→ Windows 10: Will your PC run it?

Bug Report from Apple (2013)

iOS 7.0.2

- **Passcode Lock**

  Available for: iPhone 4 and later

  Impact: A person with physical access to the device may be able to make calls to any number

  Description: A NULL dereference existed in the lock screen which would cause it to restart if the emergency call button was tapped repeatedly. While the lock screen was restarting, the call dialer could not get the lock screen state and assumed the device was unlocked, and so allowed non-emergency numbers to be dialed. This issue was addressed by avoiding the NULL dereference.

  CVE-ID

  CVE-2013-5160 : Karam Daoud of PART - Marketing & Business Development, Andrew Chung, Mariusz Rysz

- **Passcode Lock**

  Available for: iPhone 4 and later, iPod touch (5th generation) and later, iPad 2 and later

  Impact: A person with physical access to the device may be able to see recently used apps, see, edit, and share photos

  Description: The list of apps you opened could be accessed during some transitions while the device was locked, and the Camera app could be opened while the device was locked.

  CVE-ID

  CVE-2013-5161 : videosdebarraquito

http://news.cnet.com/8301-1009_3-57603787-83/apple-promises-to-fix-ios-7-lock-screen-hack/

This is really bad!! (all over the news)

Twitter, Inc. (US) https://twitter.com/lemiorhan/status/935578694541770752

Home Notifications Messages Search Twitter Tweet

Lemi Orhan Ergin
@lemiorhan

Follow

Dear @AppleSupport, we noticed a *HUGE* security issue at MacOS High Sierra. Anyone can login as "root" with empty password after clicking on login button several times. Are you aware of it @Apple?

6:38 PM - 28 Nov 2017

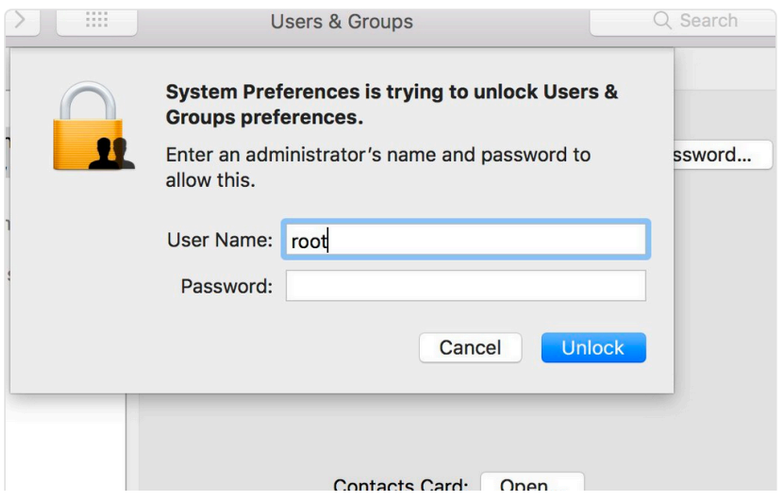
12,665 Retweets **15,539** Likes

1.2K 13K 16K

Tweet your reply

Lemi Orhan Ergin @lemiorhan · 28 Nov 2017

You can access it via System Preferences>Users & Groups>Click the lock to make changes. Then use "root" with no password. And try it for several times. Result is unbelievable!



Lemi Orhan Ergin
@lemiorhan

Agile Software Craftsman, iyzicoder @ iyzico.com, Founder of Software Craftsmanship Turkey @scturkey, The community guy bit.ly/lemiorhan

Istanbul, Turkey
lemiorhanergin.com
Joined January 2009

“Weird Facebook glitch breaks News Feed for some users”

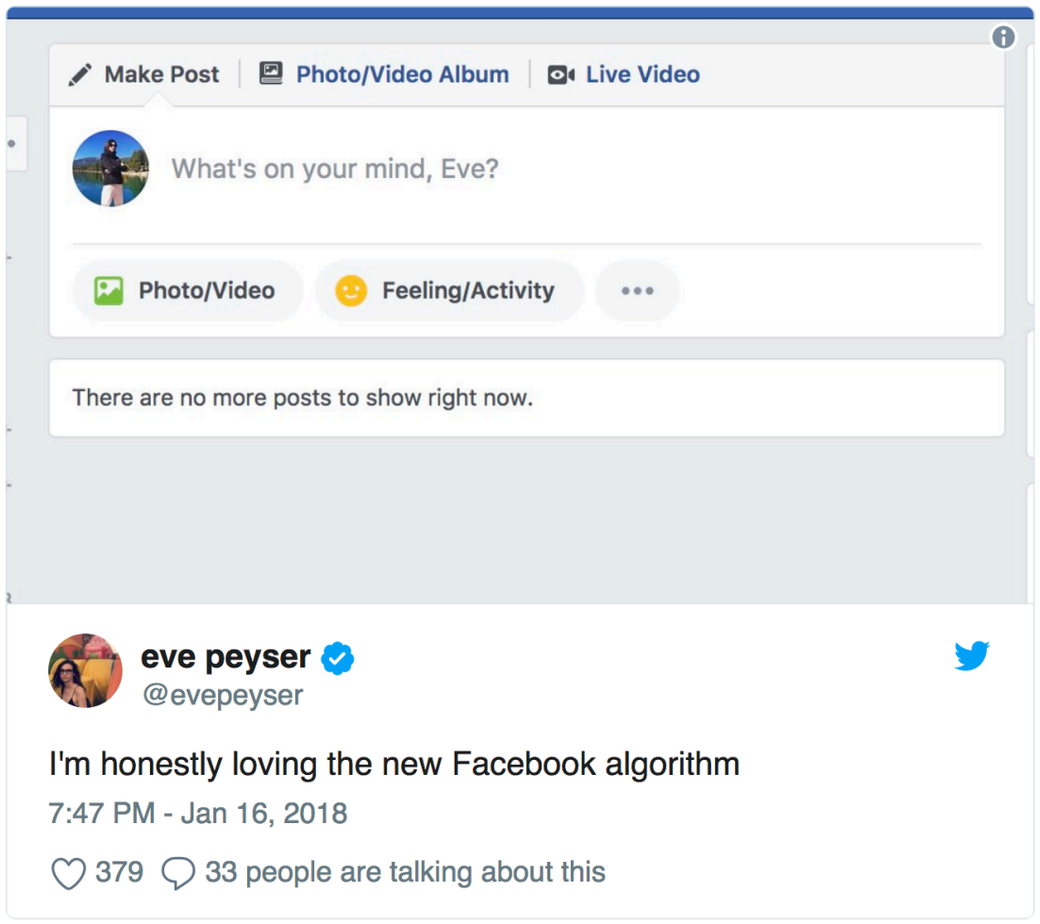
BY KARISSA BELL

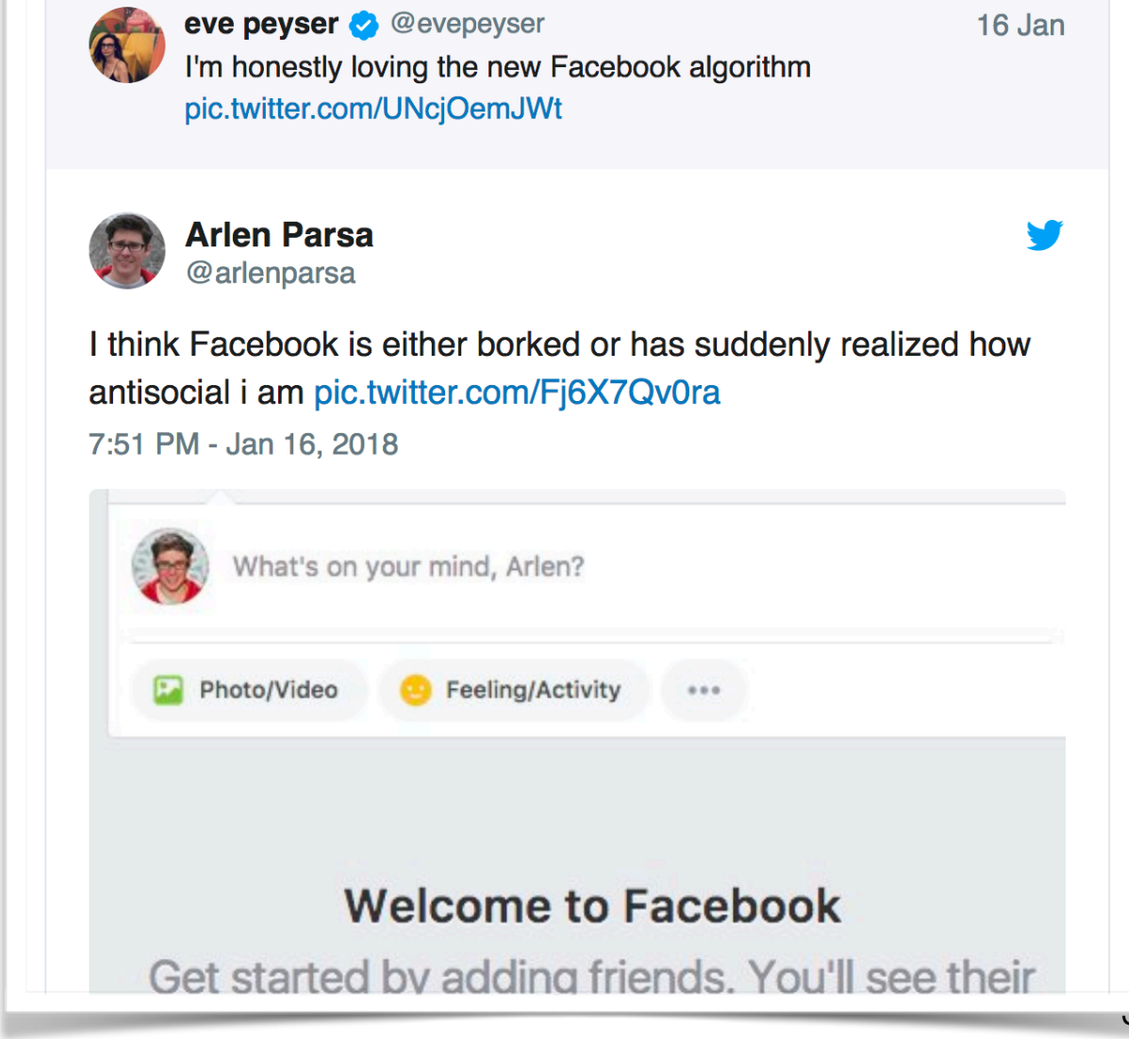
JAN 16, 2018

Facebook's News Feed is broken. No, that is't a comment on the current state of social media or Mark Zuckerberg's pledge to fix what's broken about Facebook.

I mean, it's literally broken. Many users are reporting that they're opening the Facebook app and website only to see a big, blank space that says "there are no more posts to show right now."

SEE ALSO: I deleted Facebook off my phone and you should too



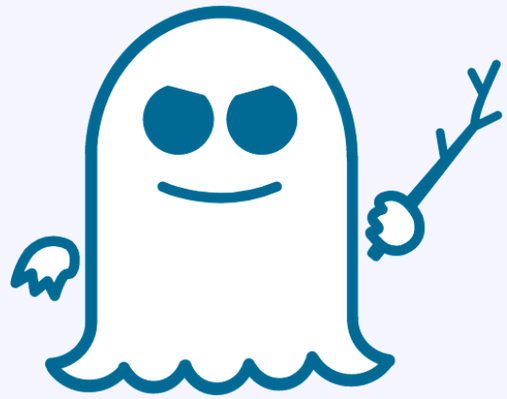


https://meltdownattack.com/

Meltdown

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

If your computer has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information without the chance of leaking the information. This applies both to personal computers as well as cloud infrastructure. Luckily, there are software patches against Meltdown.



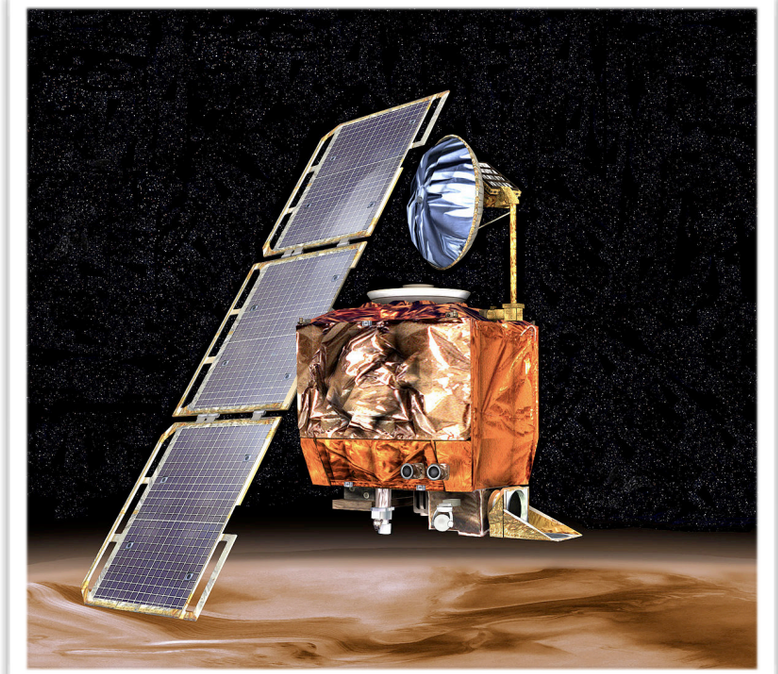
Spectre

Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate. However, it is possible to prevent specific known exploits based on Spectre through software patches.

Making Sure Software Really Works

- Software failures:
  - system crashes
  - unresponsive services
  - data losses
  - incorrect behaviours
  - security flaws



- can have huge impacts:
  - economic: NASA's Mars Climate Orbiter - $125M+; Ariane5, $8B+;
  - user hassle: FB - 2.2B; Gmail - 1B+; Instagram - 500M; Twitter - 330M; Netflix - 120M
  - data and systems security: Vulnerabilities reported in 10y (Microsoft:3000, Oracle:3100, Apple:2600, …)
  - military: Stuxnet (USA->Iran); F22 Crash; Patriot Missiles missed targets;

Home
Browse :
Vendors
Products
Vulnerabilities By Date
Vulnerabilities By Type
Reports :
CVSS Score Report
CVSS Score Distribution
Search :
Vendor Search
Product Search
Version Search
Vulnerability Search
By Microsoft References
Top 50 :
Vendors
Vendor Cvss Scores
Products
Product Cvss Scores
Versions
Other :
Microsoft Bulletins
Bugtraq Entries
CWE Definitions

Top 50 Vendors By Total Number Of "Distinct" Vulnerabilities

Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 20
2019 2020 All Time Leaders

	Vendor Name	Number of Products	Number of Vulnerabilities	#Vulnerabilities/#Products
1	Microsoft	529	6814	13
2	Oracle	644	6115	9
3	IBM	1064	4679	4
4	Google	84	4572	54
5	Apple	119	4512	38
6	Cisco	3676	4167	1
7	Adobe	132	3314	25
8	Debian	97	3197	33
9	Redhat	301	2805	9
10	Linux	17	2370	139
11	Mozilla	24	2199	92
12	Canonical	30	2025	68
13	HP	3579	1794	1
14	SUN	204	1628	8
15	Opensuse	25	1315	53
16	Apache	198	1218	6

Pressure to update software fast

- Software development is increasingly competitive
- Any mistake can be extremely expensive
- Pressure is on to deliver fast and change even faster
- Companies deploy software at an astonishing pace:
  - Amazon: “every 11.7 seconds”
  - Netflix: “thousands of times per day”
  - Facebook: “bi-weekly app updates”

What's the proper way of doing it?

To Type or Not to Type: Quantifying Detectable Bugs in JavaScript

Zheng Gao
University College London
London, UK
z.gao.12@ucl.ac.uk

Christian Bird
Microsoft Research
Redmond, USA
cbird@microsoft.com

Earl T. Barr
University College London
London, UK
e.barr@ucl.ac.uk

***Abstract*—JavaScript is growing explosively and is now used in large mature projects even outside the web domain. JavaScript is also a dynamically typed language for which static type systems, notably Facebook's Flow and Microsoft's TypeScript, have been written. What benefits do these static type systems provide?**

Leveraging JavaScript project histories, we select a fixed bug and check out the code just prior to the fix. We manually add type annotations to the buggy code and test whether Flow and TypeScript report an error on the buggy code, thereby possibly prompting a developer to fix the bug before its public release. We then report the proportion of bugs on which these type systems reported an error.

Evaluating static type systems against public bugs, which have survived testing and review, is conservative: it understates their effectiveness at detecting bugs during private development, not to mention their other benefits such as facilitating code search/completion and serving as documentation. Despite this uneven playing field, our central finding is that both static type systems find an important percentage of public bugs: both Flow 0.30 and TypeScript 2.0 successfully detect 15%!

***Keywords*-JavaScript; static type systems; Flow; TypeScript; mining software repositories;**

to invest in static type systems for JavaScript: first Google released Closure[1], then Microsoft published TypeScript[2], and most recently Facebook announced Flow[3]. What impact do these static type systems have on code quality? More concretely, how many bugs could they have reported to developers?

The fact that long-running JavaScript projects have extensive version histories, coupled with the existence of static type systems that support gradual typing and can be applied to JavaScript programs with few modifications, enables us to under-approximately quantify the beneficial impact of static type systems on code quality. We measure the benefit in terms of the proportion of bugs that were checked into a source code repository that might not have been if the committer were using a static type system that reported an error on the bug.

In this experiment, we sample public software projects, check out a historical version of the codebase known to contain a bug, and add type annotations. We then run a static type checker on the altered, annotated version to determine if the type checker errors on the bug, possibly triggering a developer to fix the bug.

r/programming

Posts FAQ

1.9k

Posted by u/phantomfive 1 year ago

Strongly Typed Languages Reduce Bugs by 15%

blog.acolyer.org/2017/0...

780 Comments Share Save 94% Upvoted

This thread is archived
New comments cannot be posted and votes cannot be cast

SORT BY BEST

tdammers 853 points · 1 year ago

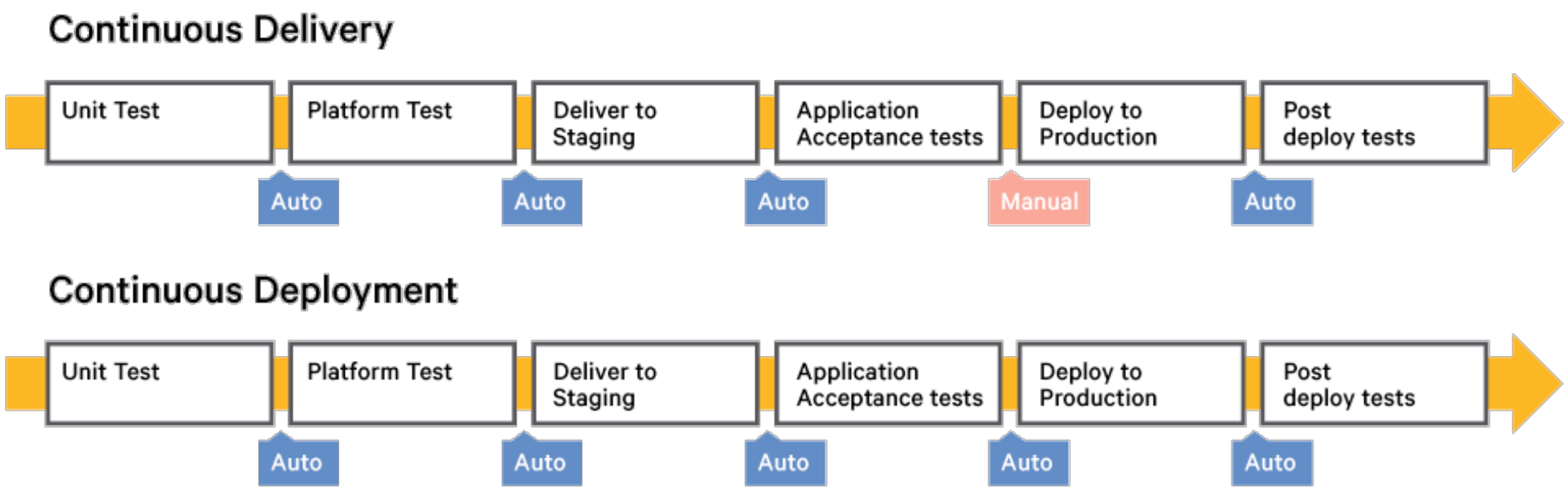
And that's not even the main benefit of a type system. Type systems really shine when it comes to making assumptions, conventions, patterns, and invariants explicit. Types are enforced and unambiguous documentation; "catching bugs" is a nice extra that you get for free, but the real benefit IMO is freeing up brain capacity because you can offload so much information to the toolchain.

Share Report Save

What's the proper way of doing it?

Processes and Tools

- Processes and Methods for software construction and software deployment (DevOps)
- Specification and development methods
- Testing tools and toolchains
- Validation and Verification techniques



Multi-Cloud Continuous Delivery with Spinnaker report now available.

Netflix Technology Blog Follow
May 15, 2018 · 2 min read

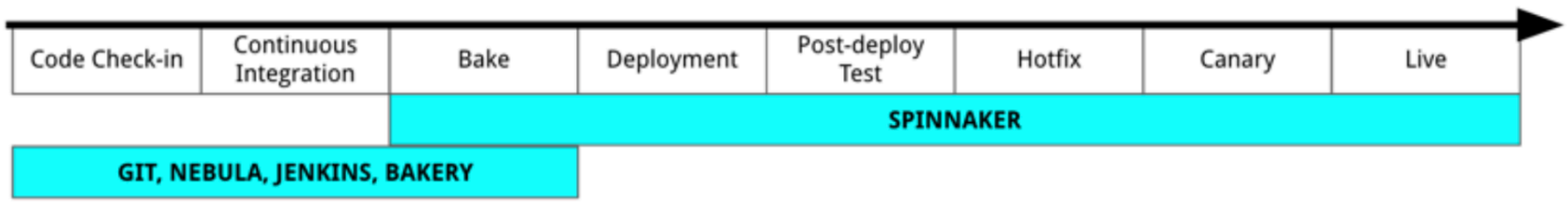
by Emily Burns, Asher Feldman, Rob Fletcher, Tomas Lin, Justin Reynolds, Chris Sanden and Rob Zienert

We're pleased to announce the release of our O'Reilly report, *Continuous Delivery with Spinnaker*. The report is available to download for free on the Spinnaker website. (Pdf | Epub | Mobi)

How We Build Code at Netflix

Netflix Technology Blog Follow
Mar 9, 2016 · 8 min read

How does Netflix build code before it's deployed to the cloud? While pieces of this story have been told in the past, we decided it was time we shared more details. In this post, we describe the tools and techniques used to go from source code to a deployed service serving movies and TV shows to more than 75 million global Netflix members.



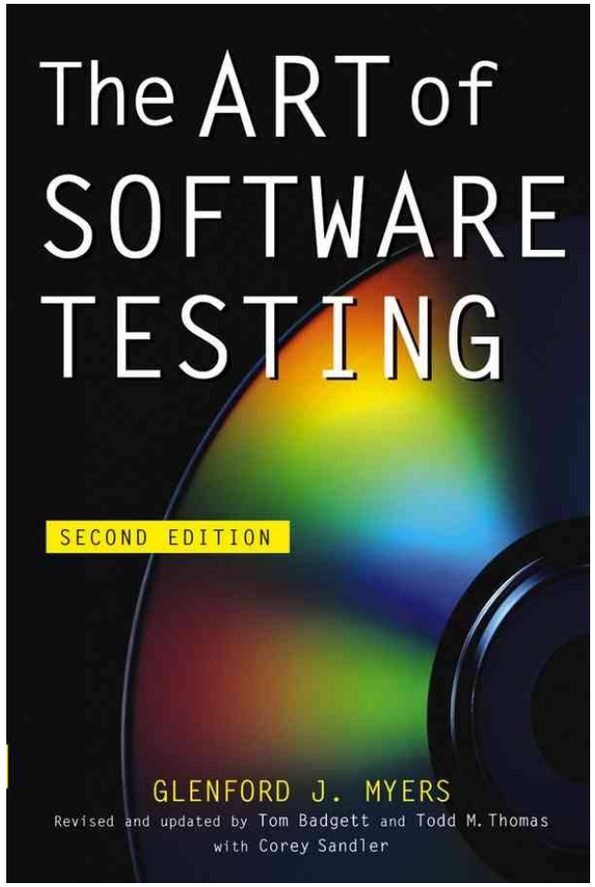
The above diagram expands on a previous post announcing Spinnaker, our global continuous delivery platform. There are a number of steps that need to happen before a line of code makes it way into Spinnaker:

- Code is built and tested locally using Nebula
- Changes are committed to a central git repository
- A Jenkins job executes Nebula, which builds, tests, and packages the application for deployment

Software Testing

"Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended"

The Art of Software Testing, Second Edition.
Glenford Myers.



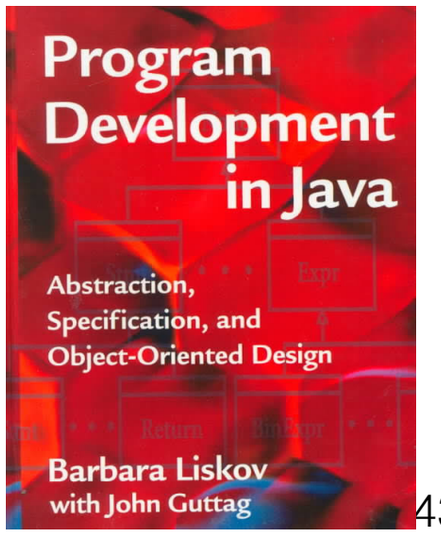
Software Testing - Validation

“Validation is the process designed to increase our confidence that a program works as intended. It can be done through verification or testing.”

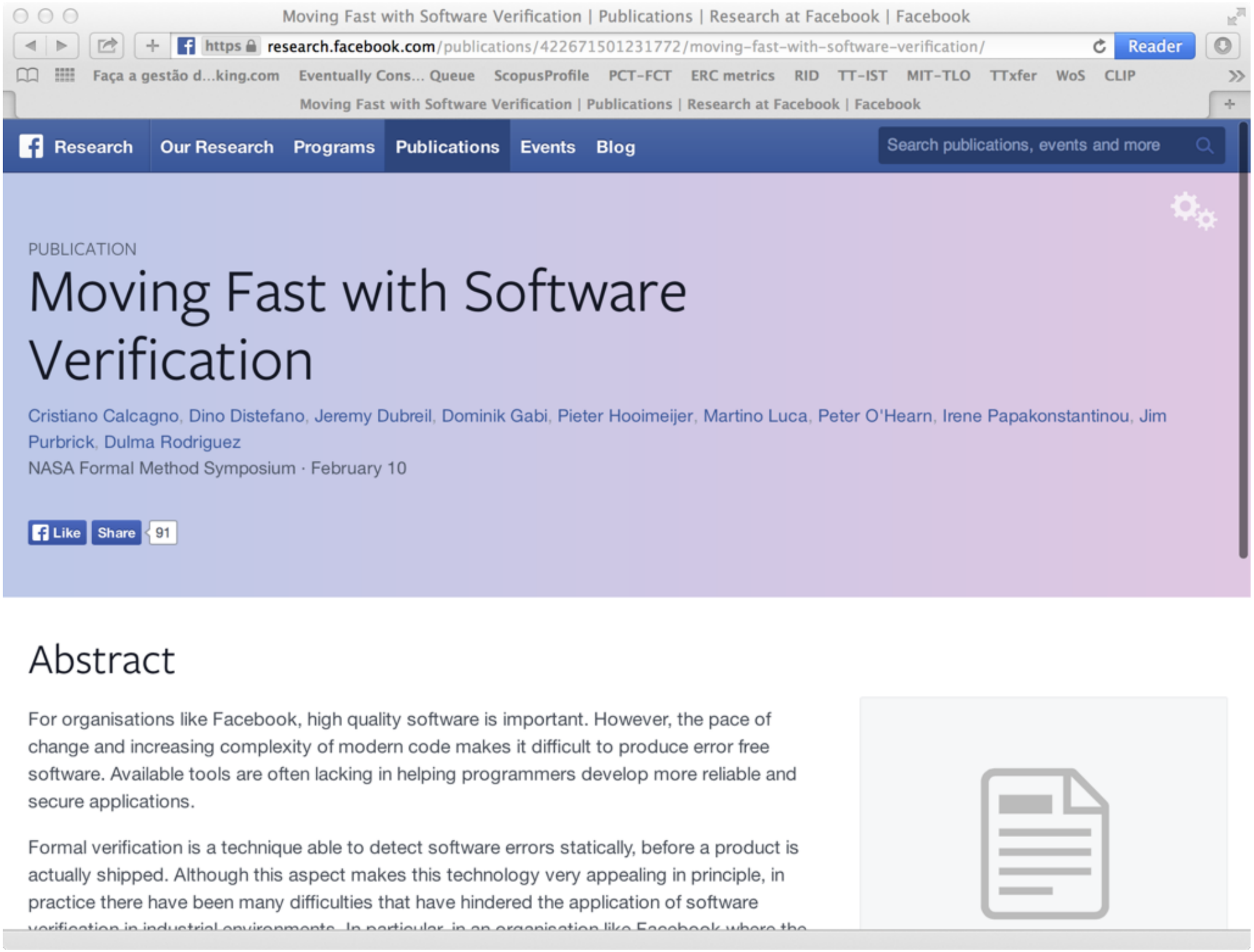
“Verification is a formal or informal argument that a program works on all possible inputs”.

“Testing is the process of running a program on a set of test cases and comparing the actual results with expected results”

in *Program Development in Java* (p222)



Software Verification at Facebook



Software Verification at Facebook

Open-sourcing Facebook Infer: Identify bugs before you ship

 Cristiano Calcagno  Dino Distefano  Peter O'Hearn

Today, we're open-sourcing **Facebook Infer**, a static program analyzer that Facebook uses to identify bugs before mobile code is shipped. Static analyzers are automated tools that spot bugs in source code by scanning programs without running them. They complement traditional dynamic testing: Where testing allows individual runs through a piece of software to be checked for correctness, static analysis allows multiple and sometimes even all flows to be checked at once. Facebook Infer uses mathematical logic to do symbolic reasoning about program execution, approximating some of the reasoning a human might do when looking at a program. We use Facebook Infer internally to analyze the main Facebook apps for Android and iOS (used by more than a billion people), Facebook Messenger, and Instagram, among others. At present, the analyzer reports problems caused by null pointer access and resource and memory leaks, which cause a large percentage of app crashes.

Software Verification at Facebook

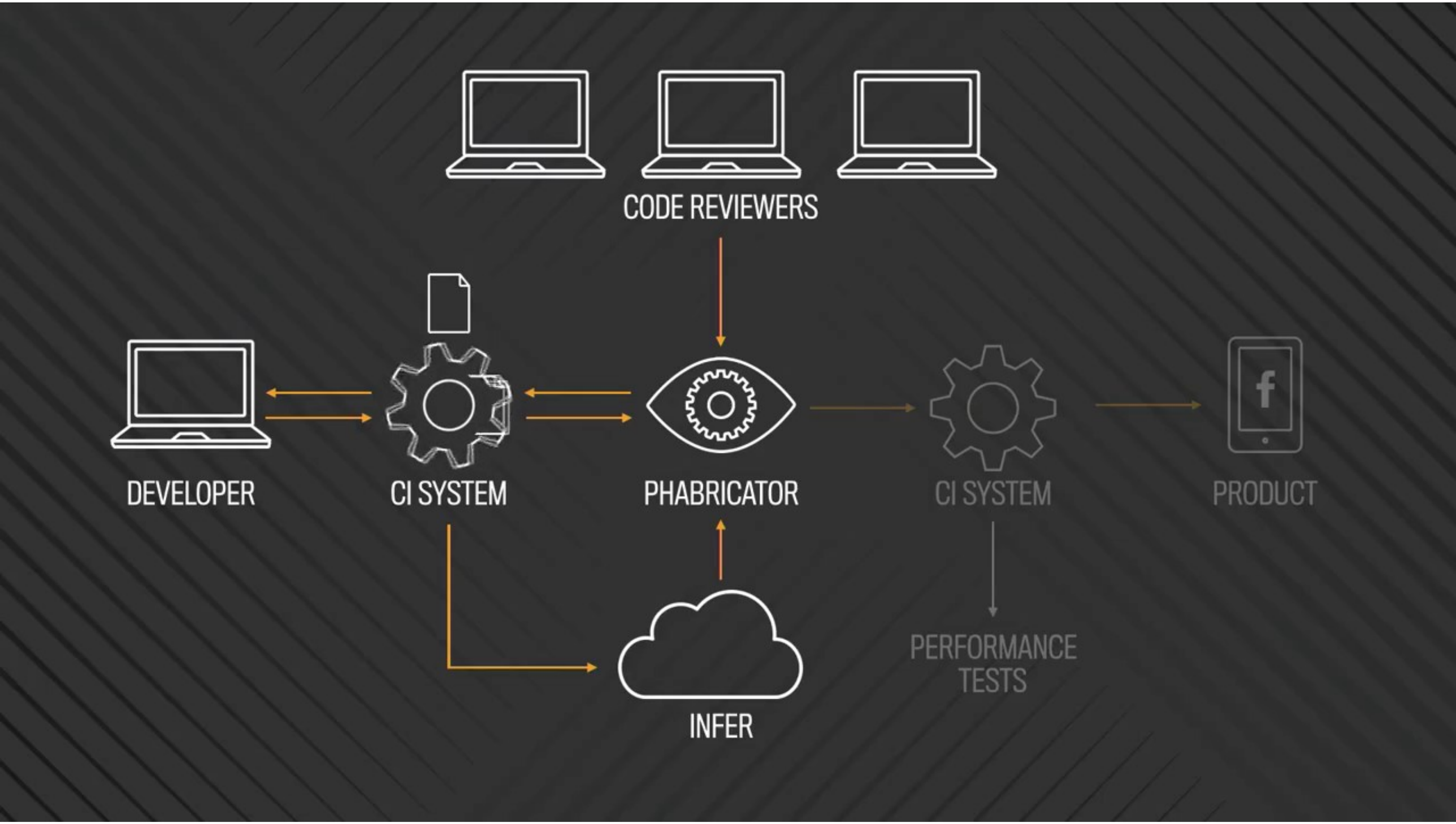
Open-sourcing Facebook Infer: Identify bugs before you ship

 Cristiano Calcagno  Dino Distefano  Peter O'Hearn

To
ide
so
tes
co
Fa
ap
Fa
tha
rep
lar

Each month, hundreds of potential bugs identified by Facebook Infer are fixed by our developers before they are committed to our codebases and deployed to people's phones. This saves our developers many hours finding and fixing bugs, and results in better products for people.

Bug detection tool chain



EWD303

On the reliability of programs.

All speakers of the lecture series have received very strict instructions as how to arrange their speech; as a result I expect all speeches to be similar to each other. Mine will not differ, I adhere to the instructions. They told us: first tell what you are going to say, then say it and finally summarize what you have said.

My story consists of four points.

1. I shall argue that our programs should be correct
2. I shall argue that debugging is an inadequate means for achieving that goal and that we must prove the correctness of programs
3. I shall argue that we must tailor our programs to the proof requirements
4. I shall argue that programming will become more and more an activity of mathematical nature.

The starting point of my considerations is to be found at the "software failure". A few years ago the wide-spread existence of this regrettable phenomenon was established beyond doubt; as far as my information tells me, the software failure is still there as vigorous as ever and its effects are sufficiently alarming to justify our concern and attention. What, however, is it?

https://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD303.html

Testes de Software

"*Testing shows the presence, not the absence of bugs*"

Edsger W. Dijkstra, 1969

Software Correctness

Relevance of Software Correctness

- Quality procedures must be enforced at all levels, in particular at the construction phase, where most of the issues are introduced and difficult to circumvent.
- **Questions for you now**:
  - What methods do you currently use to make sure your code is “bullet-proof” ?
  - How can you prove to yourself (and others) that your code is “bullet-proof” ?
  - What arguments do you use to convince yourself and others that your code works as expected and not goes wrong, with respect to functional correctness, security, or concurrency errors?

Relevance of Software Correctness

- Quality procedures must be enforced at all levels, in particular at the construction phase, where most of the issues are introduced and difficult to circumvent.
- **Questions for you now**:
  - What methods do you currently use to make sure your code is “bullet-proof” ?
  - How can you prove to yourself (and others) that your code is “bullet-proof” ?
  - What arguments do you use to convince yourself and others that your code works as expected and not goes wrong, with respect to functional correctness, security, or concurrency errors?
- You will **know better answers** at the end of this course.

Software Correctness: What and How

- **Key engineering concern**:
  Make sure that the software developed and constructed is “correct”.
- What does this mean?
  - Is it crash-free? (“runtime safety”)
  - Gives the right results? (“functional correctness”)
  - Does it operate effectively? (“resource conformance”)
  - Does it violate user privacy? (“security conformance”)
  - …
- several process and methodological approaches to ensure and validate correctness exist (software engineering course)
- In this course, we cover some techniques to rigorously ensure and validate correctness **during software construction**

Software Correctness: What and How

- "runtime safety" (no crashes, etc.) is a bit easier to define
  - programming language type systems help a bit …
- other kinds of correctness are not so easy to define
- usually relative to special assumptions ...
  - what the system is supposed to do: play chess, manage bank accounts, …
  - the available resources: bandwidth, memory, processing speed, …
  - the security policies: only my friends can see my pics, …
- To precisely define such assumptions, we need
  - 1: precise specifications
  - 2: ways of validating that your system meets the spec

Correctness is against a specification

- Then what does “correct software” mean?
  - Always relative to some given (**our**) specs
- Correct means that software meets **our** specs
  - There is no such thing as the “right specification”
  - In practice, the spec is usually incomplete ...
  - But **the spec must not be wrong** !
  - It should be very easy to check what the spec states
  - The spec **must be simple, much simpler than code**
  - The spec should be **focused** (pick relevant cases)
    - e.g., buffers are not being overrun
    - e.g., never transfer money without logging the source

Checking Specs: Dynamic Verification

- By “dynamic verification” we mean that verification is **done at runtime**, while the program executes
- Some successful approaches:
  - **unit testing**
  - **coverage testing**
  - **regression testing**
  - **test generation**
  - **runtime monitoring**
- use runtime monitors to (continuously) check that code do not violate correctness properties
- violations causes exceptional behaviour or halt, so errors are detected after something wrong already occurred (think of a car crash, or a securiy leak)

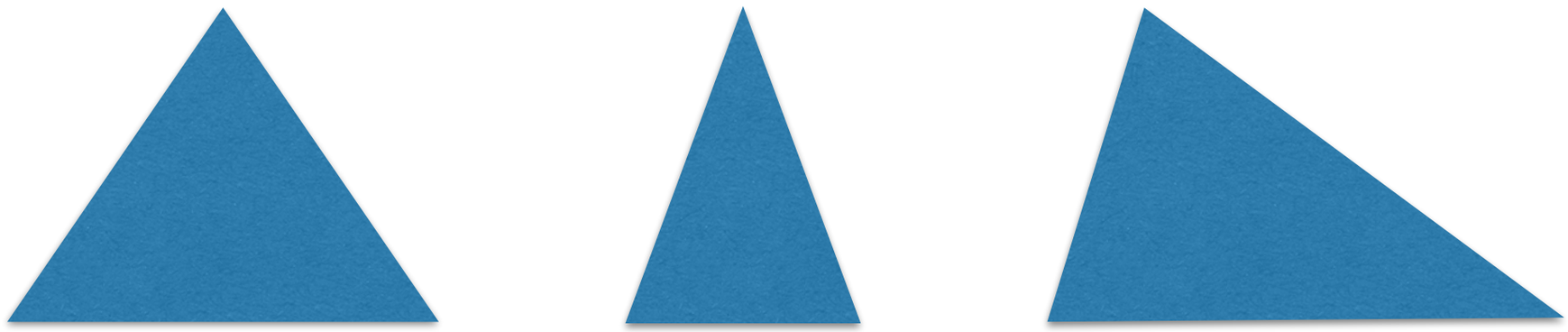
Checking Specs: Dynamic Verification

- Some shortcomings of dynamic verification
  - always introduces a level of performance overhead
  - may show the existence of some errors, but does not ensure absence of errors (the code passed a test suite today, but may fail with some other clever test)
  - **Challenge**: how do you make sure that you are defining the “right” tests and “enough” tests
  - Will talk again about testing methods later on in the course

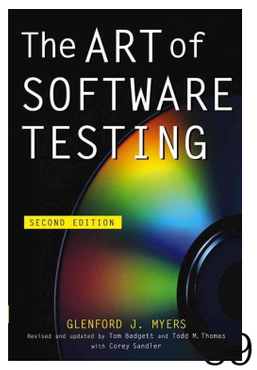
Quiz

Vamos ver quem sabe testar…

"The program reads three integer values from an input dialog. The three values represent the lengths of the sides of a triangle. The program displays a message that states whether the triangle is scalene, isosceles, or equilateral."



Create specific tests (10 minutes)



Quiz

1. Do you have a test case that represents a valid scalene triangle? (Cases such as 1, 2, 3 and 2, 5, 10 are not valid triangles)
2. Do you have a test case that represents a valid equilateral triangle?
3. Do you have a test case that represents a valid isosceles triangle? (Cases such as 2,2,4 are not valid triangles.)
4. Do you have at least three test cases that represent valid isosceles triangles such that you have tried all three permutations of two equal sides (e.g. 3,3,4; 3,4,3; and 4,3,3)?
5. Do you have a test case in which one side has a zero value?
6. Do you have a test case in which one side has a negative value?

Quiz

7. Do you have a test case with three integers greater than zero such that the sum of two of the numbers is equal to the third? (If 1,2,3 is a scalene triangle, it's a bug.)

8. Do you have at least three test cases in category 7 such that you have tried all three permutations where the length of one side is equal to the sum of the lengths of the other two sides (for example, 1,2,3; 1,3,2; and 3,1,2)?

9. Do you have a test case with three integers greater than zero such that the sum of two of the numbers is less than the third (such as 1,2,4 or 12,15,30)?

10. Do you have at least three test cases in category 9 such that you have tried all three permutations (for example, 1,2,4; 1,4,2; and 4,1,2)?

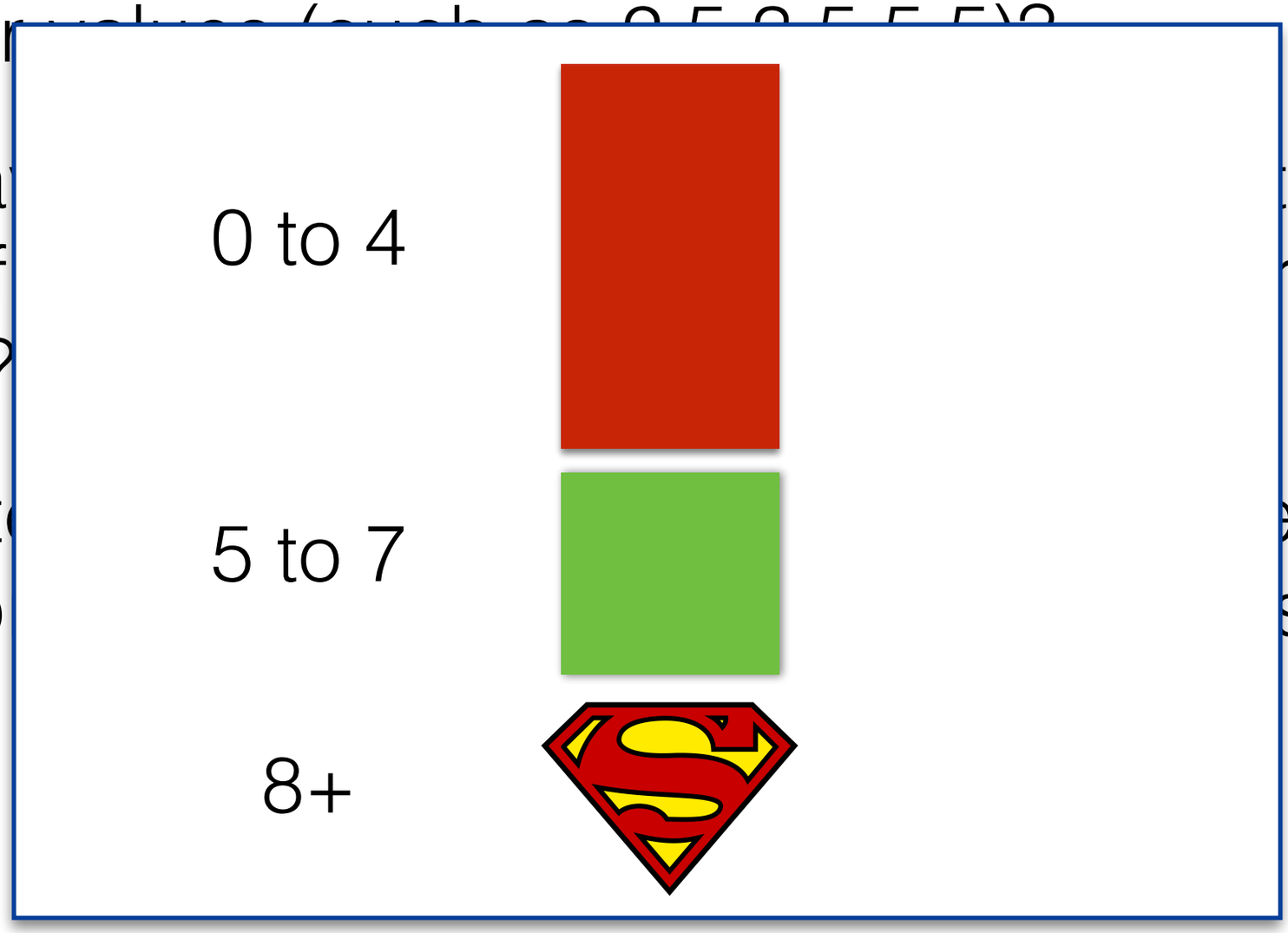
Quiz

11. Do you have a test case in which all sides are zero (0,0,0)?
12. Do you have at least one test case specifying noninteger values (such as 2.5,3.5,5.5)?
13. Do you have at least one test case specifying the wrong number of values (two rather than three integers, for example)?
14. For each test case did you specify the expected output from the program in addition to the input values?

result = ?

Quiz

11. Do you have a test case in which all sides are zero (0,0,0)?
12. Do you have at least one test case specifying noninteger
13. Do you ha the wrong number of s, for example)?
14. For each t d output from the p s?



Readings

- Cost of Bugs
  https://crossbrowsertesting.com/blog/development/software-bug-cost/

- Pentium Bug 1990s
  https://www.cs.earlham.edu/~dusko/cs63/fdiv.html

- Meltdown and Spectre
  https://meltdownattack.com/

- EWD303
  https://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD303.html

- EWD268 Structured Programming
  https://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD268.html

- Program Development in Java, Liskov/Guttag (ch1 and ch10).

- “Dafny: An Automatic Program Verifier for Functional Correctness”, Leino.