Construction and Verification of Software
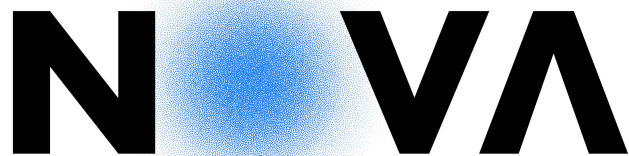
2021 - 2022

MIEI - Integrated Master in Computer Science and Informatics
Consolidation block

Lecture 1 - Introduction and Motivation
Bernardo Toninho (btoninho@fct.unl.pt)
based on previous editions by **João Seco** and **Luís Caires**

NOVA
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

A small exercise…

- To determine the maximum of the first N elements of an array.

Bug?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an array.

```java
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?

- To determine the maximum of the first N elements of an 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?

- To determine the maximum of the first N elements of an 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?

- To determine the maximum of the first N elements of an 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;
}
```

Another small exercise...

- Binary search in an array:

```java
int binarySearch(int key, int[] arr) {
    int low = 0;
    int high = arr.length;

    while (low < high) {
        int mid = (low+high)/2;

        if (arr[mid] == key)
            return mid; //found
        else if (arr[mid] < key) low = mid + 1;
        else high = mid;

    }
    return -1; //not found
}
```

Binary Search in an array — Bug?

```java
int binarySearch(int key, int[] arr) {
    int low = 0;
    int high = arr.length;

    while (low < high) {
        int mid = (low+high)/2;

        if (arr[mid] == key)
            return mid; //found
        else if (arr[mid] < key) low = mid + 1;
        else high = mid;

    }
    return -1; //not found
}

@Test
public void testBS() {
    assertEquals(-1, MyCollection.binarySearch(-6, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(-1, MyCollection.binarySearch(0, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(2, MyCollection.binarySearch(-3, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(0, MyCollection.binarySearch(-5, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(4, MyCollection.binarySearch(-1, new int[]{-5,-4,-3,-2,-1}));
    //...
}
```

Binary Search in an array — Bug?

```java
int binarySearch(int key, int[] arr) {
    int low = 0;
    int high = arr.length;

    while (low < high) {
        int mid = (low+high)/2;

        if (arr[mid] == key)
            return mid; //found
        else if (arr[mid] < key) low = mid + 1;
        else high = mid;

    }
    return -1; //not found
}

@Test
public void testBS() {
    assertEquals(-1, MyCollection.binarySearch(-6, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(-1, MyCollection.binarySearch(0, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(2, MyCollection.binarySearch(-3, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(0, MyCollection.binarySearch(-5, new int[]{-5,-4,-3,-2,-1}));
    assertEquals(4, MyCollection.binarySearch(-1, new int[]{-5,-4,-3,-2,-1}));
    //...
}
```

Binary Search in an array — Bug?

- The algorithm is correct….

- But what if `low+high` > 2^{31} - 1 ?

- `mid = (low+high) / 2` becomes negative!

  ‣ Best case: `ArrayIndexOutOfBoundsException`

  ‣ Worst case: undefined (i.e., arbitrary) behavior!

- We run **code**, not **algorithms.**

```java
int binarySearch(int key, int[] arr) {
    int low = 0;
    int high = arr.length;

    while (low < high) {
        int mid = (low+high)/2;

        if (arr[mid] == key)
            return mid; //found
        else if (arr[mid] < key) low = mid + 1;
        else high = mid;

    }
    return -1; //not found
}
```

Binary Search in an array — Fixing the bug

- Need to make sure we do not overflow at any point.

  - `mid = (low + high) / 2` ❌

  - `mid = low + (high - low)/2` ✅

```java
int binarySearch(int key, int[] arr) {
    int low = 0;
    int high = arr.length;

    while (low < high) {
        int mid = low+(high-low)/2;

        if (arr[mid] == key)
            return mid; //found
        else if (arr[mid] < key) low = mid + 1;
        else high = mid;

    }
    return -1; //not found
}
```

Binary Search in an array — Fixing the bug

```
newtype int32 = x | -0x8000_0000 <= x < 0x8000_0000

method BinarySearchInt32(a: array<int>, key: int) returns (r: int32)
  requires a.Length < 0x8000_0000
{
  var lo, hi := 0, a.Length as int32;
  while lo < hi
    invariant 0 <= lo <= hi <= a.Length as int32
  {
    var mid := (lo + hi) / 2;  // error: possible overflow
    if key < a[mid] {
      hi := mid;
    } else if a[mid] < key {
      lo := mid + 1;
    } else {
      return mid;
    }
  }
  return -1;
}
```

ⓧ result of operation might violate newtype constraint for 'int32'

Binary Search in an array — Fixing the bug

```
newtype int32 = x | -0x8000_0000 <= x < 0x8000_0000

method BinarySearchInt32(a: array<int>, key: int) returns (r: int32)
  requires a.Length < 0x8000_0000
{
  var lo, hi := 0, a.Length as int32;
  while lo < hi
    invariant 0 <= lo <= hi <= a.Length as int32
  {
    var mid := lo + (hi - lo) / 2;  // fixed overflow
    if key < a[mid] {
      hi := mid;
    } else if a[mid] < key {
      lo := mid + 1;
    } else {
      return mid;
    }
  }
  return -1;
}
```

Binary Search in an array (32-bit integers)

```
newtype int32 = x | -0x8000_0000 <= x < 0x8000_0000

method BinarySearchInt32(a: array<int>, key: int) returns (r: int32)
  requires a.Length < 0x8000_0000
  requires forall i,j :: 0 <= i < j < a.Length ==> a[i] <= a[j]
  ensures 0 <= r ==> r < a.Length as int32 && a[r] == key
  ensures r < 0 ==> key !in a[..]
{
  var lo, hi := 0, a.Length as int32;
  while lo < hi
    invariant 0 <= lo <= hi <= a.Length as int32
    invariant key !in a[..lo] && key !in a[hi..]
  {
    var mid := lo + (hi - lo) / 2;  // fixed overflow
    if key < a[mid] {
      hi := mid;
    } else if a[mid] < key {
      lo := mid + 1;
    } else {
      return mid;
    }
  }
  return -1;
}
```

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 **provable correctness** 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 (e.g. Dafny, Verifast).

- **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.

Learning Objectives

- **Static Verification of Software**

  - Understand the principles and know how to use assertion methods in practice to specify, reason about, and verify software.

- **ADTs and Concurrent Programming**

  - Write correct concurrent programs and ADTs

  - Understand ADT programming methodologies

  - Understand concurrent programming methodologies

- **Dynamic Verification of Software**

 - Understand principles and methods for software testing.

Bibliography

*Program Development In Java:
Abstraction, Specification, and Object-Oriented Design.*
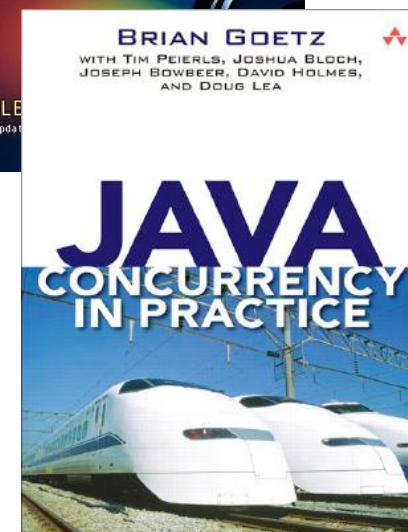Barbara Liskov (with John Guttag); MIT Press.
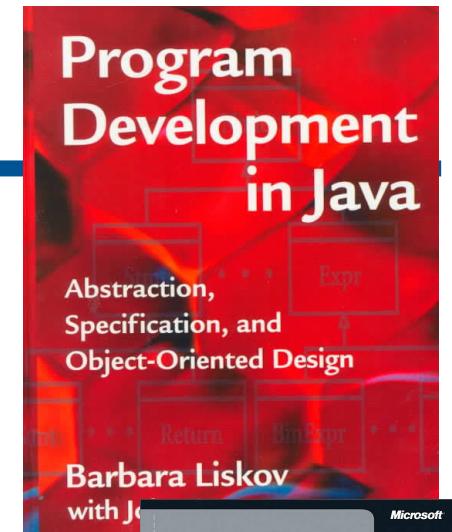
*Code Complete:
A Practical Handbook of Software Construction*, Second Edition.
Steve McConnell, Microsoft Press.
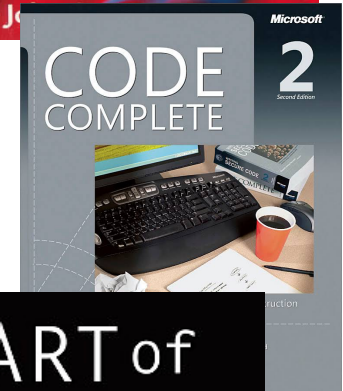
The Art of Software Testing, Second Edition
Glenford Myers, Corey Sandler, Tom Badgett
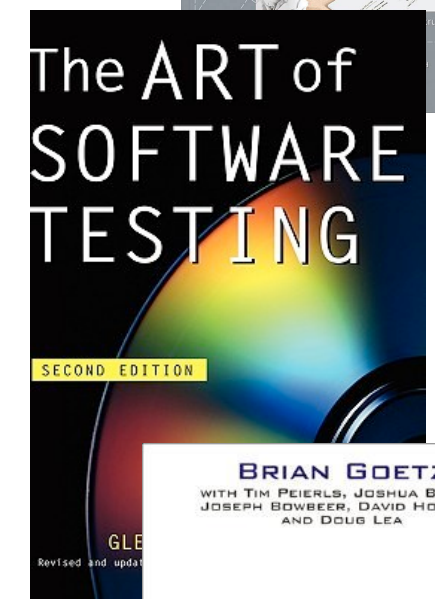
Java Concurrency in Practice,
Goetz et al. Addison-Wesley, 2006.

Tutorials for Dafny and Verifast

Administrivia

- ~12-13 Lectures

  - Midterm (W6) and Final Test (W12) — Dates TBD

- Lab Sessions (3 Eval. Components)

  - Teams of 2 students

  - 2 Handouts/Mini-projects - Dafny code

  - 1 Project
    Development and verification of a concurrent system in Java
    (using verification tools — e.g. Verifast)

- Communication channel: https://discord.gg/HvMJTv37ue

What's the True Cost of a Software Bug?
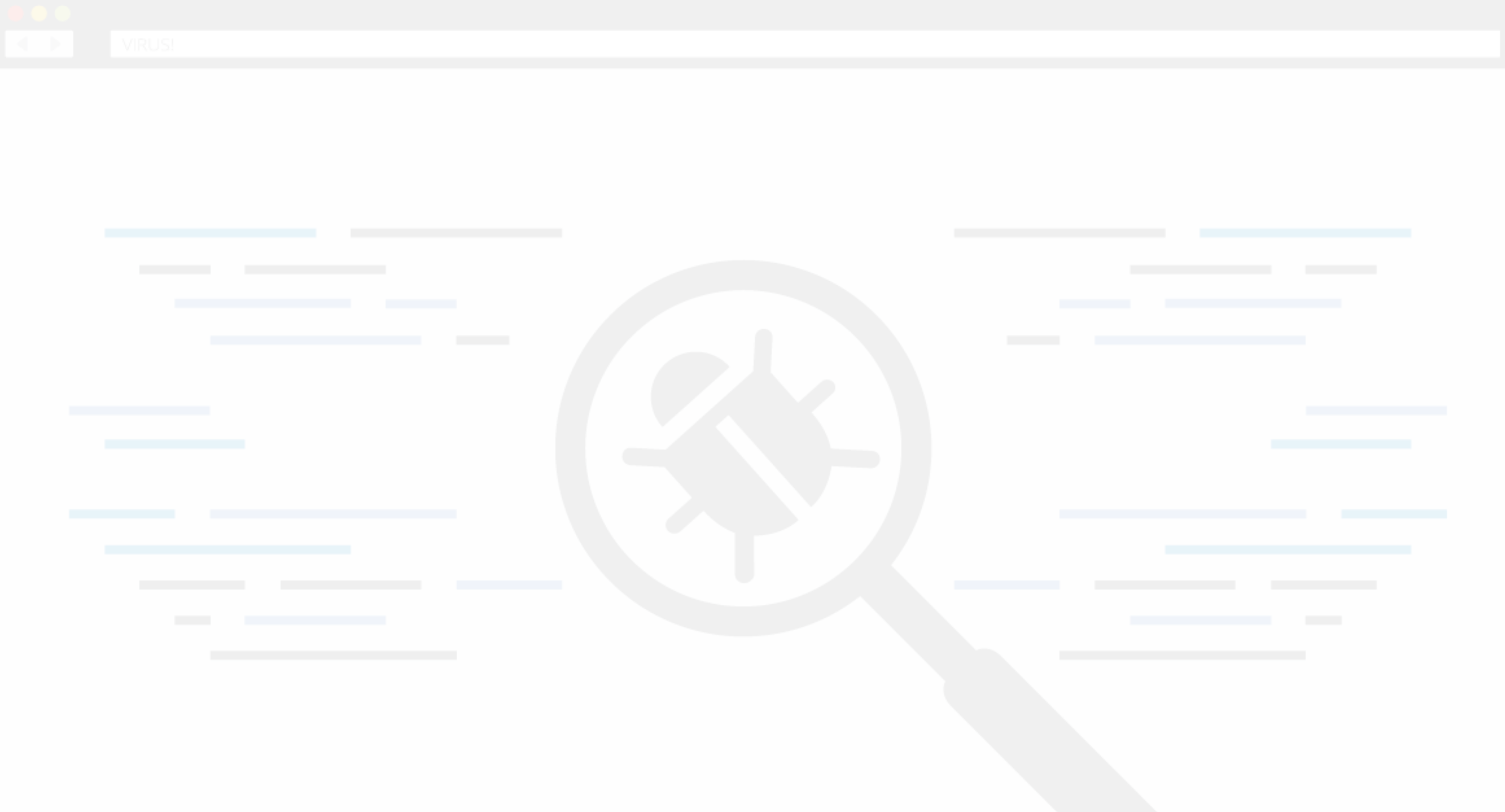
What's the True Cost of a Software Bug?

- 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
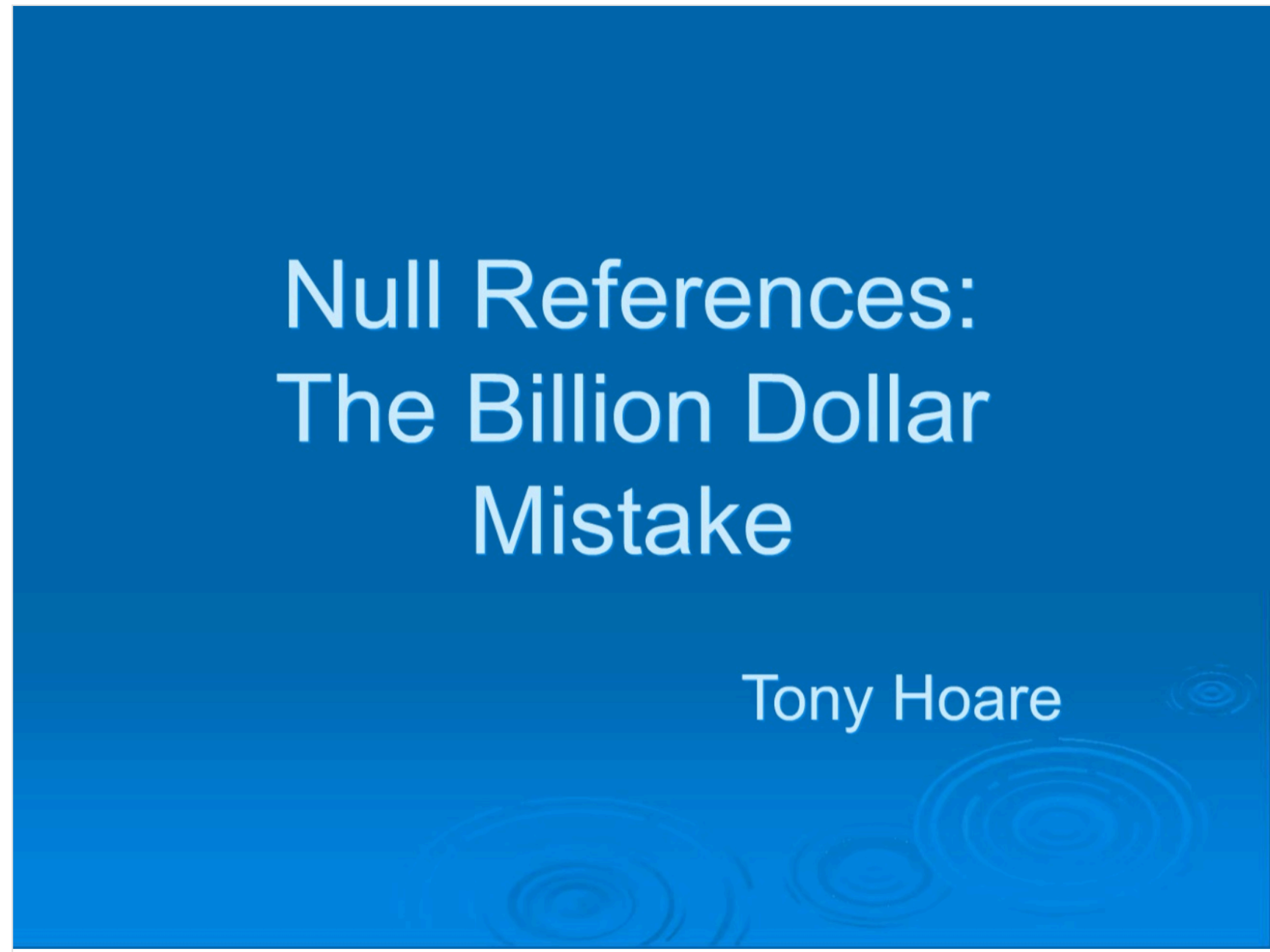
View Presentation ▯▭▭   Speed: 1X 1.5X 2X

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

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

Reuters

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.

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)

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

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!

Boeing 787 software bug can shut down planes' generators IN FLIGHT

Have you turned it off and on again? That's the way to stop the plane becoming a brick

Simon Sharwood   Fri 1 May 2015 // 06:30 UTC

160 💬

The US Federal Aviation Administration (FAA) has issued a new airworthiness directive (PDF) for Boeing's 787 because a software bug shuts down the plane's electricity generators every 248 days.

"We have been advised by Boeing of an issue identified during laboratory testing," the directive says. That issue sees "The software counter internal to the generator control units (GCUs) will overflow after 248 days of continuous power, causing that GCU to go into failsafe mode."

When the GCU is in failsafe mode it isn't making any power. That'll be bad news if all four of the GCUs aboard a 787 were powered up at the same time, because all will then shut down, "resulting in a loss of all AC electrical power regardless of flight phase."

And presumably also turning the 787 into a brick with no power for its fly-by-wire systems, lighting, climate control or in-flight movies.

FEDERAL TRADE COMMISSION
PROTECTING AMERICA'S CONSUMERS

ABOUT THE FTC | NEWS & EVENTS | ENFORCEMENT | POLICY | TIPS & ADVICE

Home » News & Events » Blogs » Tech@FTC » FTC warns companies to remediate Log4j security vulnerability

FTC warns companies to remediate Log4j security vulnerability

By: This blog is a collaboration between CTO and DPIP staff and the AI Str

TAGS: Accountability | Data security | Patches

Log4j is a ubiquitous piece of software used to record activities in a wide ra
products and services. Recently, a serious vulnerability in the popular Java
(CVE-2021-44228) was disclosed, posing a severe risk to millions of consu
web applications. This vulnerability is being widely exploited by a growing s

When vulnerabilities are discovered and exploited, it risks a loss or breach
and other irreversible harms. The duty to take reasonable steps to mitigate
implicates laws including, among others, the Federal Trade Commission Ac
critical that companies and their vendors relying on Log4j act now, in order
consumers, and to avoid FTC legal action. According to the complaint in Ec
vulnerability irreversibly exposed the personal information of 147 million co
million to settle actions by the Federal Trade Commission, the Consumer Fir
states. The FTC intends to use its full legal authority to pursue companies th
consumer data from exposure as a result of Log4j, or similar known vulnera

Check if you use the Log4j software library by consulting the Cybersecurity
(CISA) guidance: https://www.cisa.gov/uscert/apache-log4j-vulnerability-gui

DIVE BRIEF

Apache tells US Senate committee the Log4j vulnerability could take years to resolve

Published Feb. 9, 2022

By David Jones
Reporter

Making Sure Software Really Works

- Software failures:

  - System crashes

  - Unresponsive services

  - Data losses

  - Incorrect behaviours

  - Security flaws

- Dramatic 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;
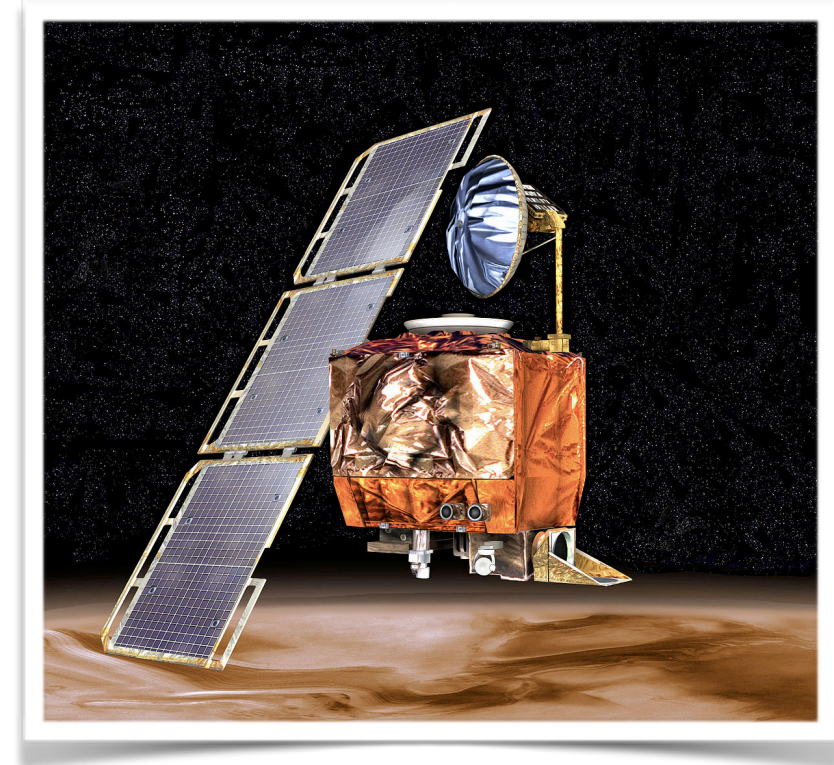
Pressure to update software fast

- 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"

Tech trailblazers

10 companies killing it at DevOps

Top 50 Vendors By Total Number Of "Distinct" Vulnerabilities

	Vendor Name	Number of Products	Number of Vulnerabilities	#Vulnerabilities/#Products
1	Microsoft	665	8394	13
2	Oracle	969	8243	9
3	Google	128	6911	54
4	Debian	108	5989	55
5	Apple	140	5467	39
6	IBM	1335	5365	4
7	Cisco	5607	4159	1
8	Redhat	426	4032	9
9	Canonical	49	3165	65
10	Fedoraproject	21	2850	136
11	Linux	23	2791	121
12	Opensuse	51	2499	49
13	Mozilla	32	2347	73
14	HP	8268	1786	0
15	Apache	280	1663	6
16	SUN	206	1535	7

Top 50 Products By Total Number Of "Distinct" Vulnerabilities

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Debian Linux	Debian	OS	5862
2	Android	Google	OS	4073
3	Ubuntu Linux	Canonical	OS	3126
4	Mac Os X	Apple	OS	2965
5	Fedora	Fedoraproject	OS	2788
6	Linux Kernel	Linux	OS	2762
7	Windows 10	Microsoft	OS	2590
8	Iphone Os	Apple	OS	2573
9	Chrome	Google	Application	2346
10	Windows Server 2016	Microsoft	OS	2334
11	Windows Server 2008	Microsoft	OS	2154
12	Windows 7	Microsoft	OS	2019
13	Firefox	Mozilla	Application	1993
14	Windows Server 2012	Microsoft	OS	1954
15	Windows 8.1	Microsoft	OS	1841
16	Windows Server 2019	Microsoft	OS	1792

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 processes 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 some 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 photos

- To precisely define such assumptions, we need

  - **Precise** specifications

  - Ways of validating that the system **satisfies** the specification

Correctness against a specification

- Then what does "correct software" mean?

  - Always relative to some given (**our**) specification

- Correct means that software meets **our** specification

  - There is no such thing as the absolute "right specification"

  - But **the spec must not be wrong** !

  - Crafting good / checkable specifications can be challenging.

- It should be "easy" to check what the specification states

  - The spec **must be simple, much simpler than the code**

  - The spec should be **focused**

    - e.g., buffers are not being overrun

    - e.g., never transfer money without logging the source

Dynamic Verification

- Verification that is **done at runtime**, during program execution.

- 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 cause exceptional behaviour or halt, so errors are detected **after** something wrong already occurred (think of a car crash, or a security leak).

Dynamic Verification

- Shortcomings of dynamic verification:

    - Can introduce a level of performance overhead.

    - Inadequate in production / critical settings.

    - Can show the existence of some errors, but does not ensure absence of errors.

- **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

Testing your tests

"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."
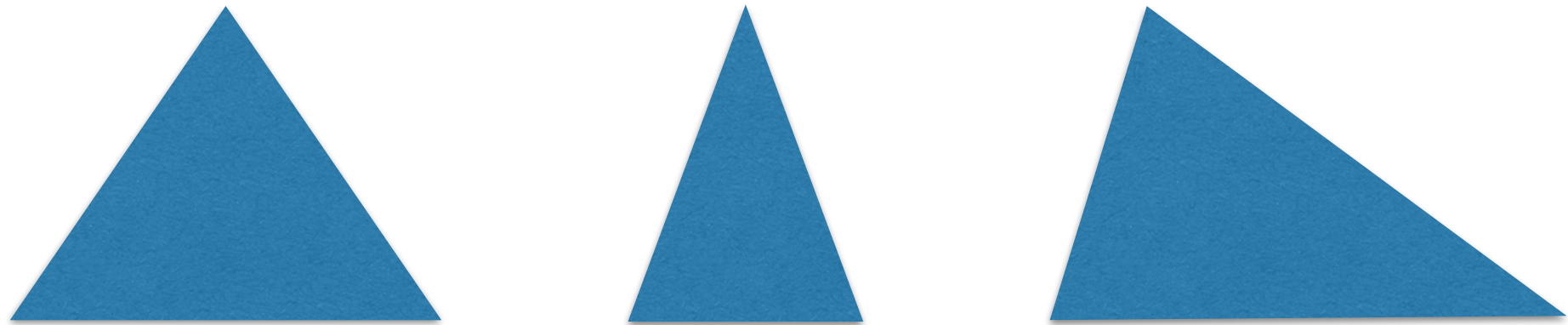
Create specific tests (10 minutes)

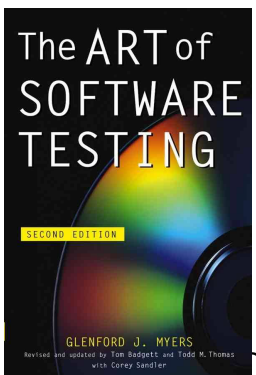# Quiz

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?
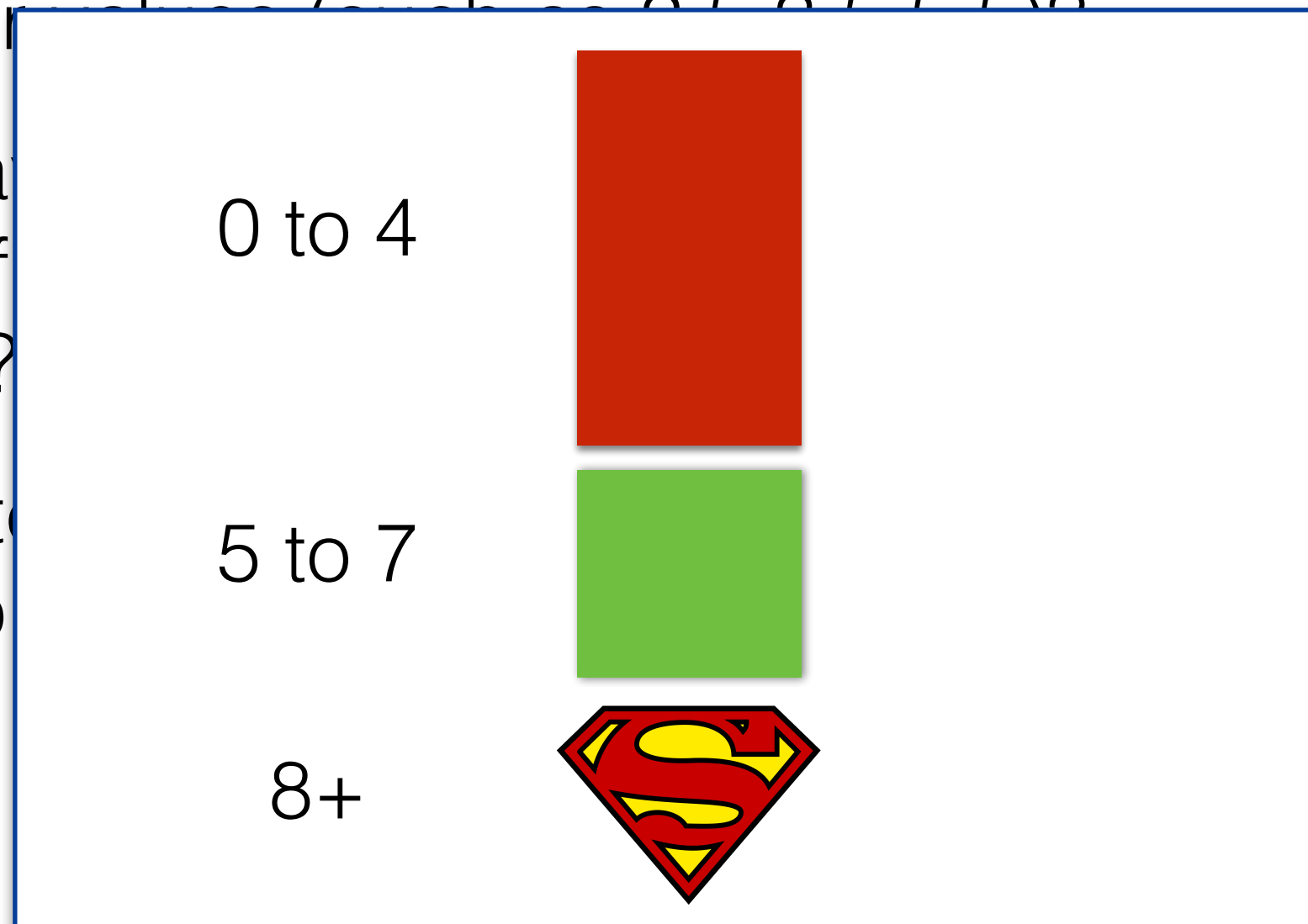
result = ?

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 values (such as 2.5, 3.5, 5.5)?

13. Do you ha[...]the wrong number of [...]s, for example)?

14. For each te[...]ed output from the p[...]s?

0 to 4

5 to 7

8+

How do we know if software is correct?

How to know if software is correct?

- One approach: Testing (i.e., **Dynamic** verification)

  - Probably incomplete, but still very useful.

  - Exhaustive testing is not feasible

- Another: Code review

  - Main concern is not correctness (although important).

  - Humans are fallible and bugs can be subtle.

  - Specification often unclear

- Better: **prove** correctness (i.e., **Static** verification)

  - Specification must be precise.

  - Meaning of code must be well-defined.

  - Reasoning must be sound.

Approaches and Techniques

- Functional Programming: Dependent Types

  - Proofs are expressed in programs (e.g. Agda).

  - Proof tactics are expressed as programs (e.g. Coq, Lean).

- Imperative Programming: Logical contracts

  - Properties are expressed in contracts.

  - Reduce correctness to logical propositions (verification conditions).

  - Use automated provers to discharge VCs.

- This course: A bit of both!

  - Functional and imperative code in Dafny.

  - Automated provers for VCs (which sometimes need "help").

Automated / Algorithmic Verification

- Formal proofs are tedious

- Automatic methods can help:

    - Fill-in low level "tedious" details.

    - Give diagnostic information (e.g. counter-examples).

    - Verify "everything".

- In this course:

    - Make use of these methods.

    - Understand **when**/how they work.

Automated / Algorithmic Verification

Systems that prove that programs match their specifications

- Problem is **undecidable**!

  - Requires annotations

  - Relieves manual burden by inferring some annotations.

- Verifiers are complex systems

  - Why trust the verifier?

  - This course provides a "big picture" understanding

Success Stories

Astrée Static Analyzer (Abstract Interpretation) [2003-]

The Astrée Static Analyzer

CNRS — Centre National de la Recherche Scientifique		École Normale Supérieure	Inria — INRIA (since Sep. 2007)

Participants:

Patrick Cousot (project leader), Radhia Cousot, Jérôme Feret, Antoine Miné, Xavier Rival

Former participants:

Industrial Applications of Astrée

The main applications of Astrée appeared two years after starting the project. Since then, Astrée has achieved the following unprecedented results on the static analysis of synchronous, time-triggered, real-time, safety critical, embedded software written or automatically generated in the C programming language:

- In Nov. 2003, Astrée was able to prove completely automatically the absence of any RTE in the primary flight control software of the Airbus A340 fly-by-wire system, a program of 132,000 lines of C analyzed in 1^h20 on a 2.8 GHz 32-bit PC using 300 Mb of memory (and 50mn on a 64-bit AMD Athlon™ 64 using 580 Mb of memory).

- From Jan. 2004 on, Astrée was extended to analyze the electric flight control codes then in development and test for the A380 series. The operational application by Airbus France at the end of 2004 was just in time before the A380 maiden flight on Wednesday, 27 April, 2005.

- In April 2008, Astrée was able to prove completely automatically the absence of any RTE in a C version of the automatic docking software of the Jules Vernes Automated Transfer Vehicle (ATV) enabling ESA to transport payloads to the International Space Station [32].

VCC (Contract-based verification) [2008]

VCC: A Practical System for Verifying Conc...

Michal Moskal, Thomas Santen, Wolfram Schulte
Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009 | January 2009
Published by Springer

VCC is an industrial-strength verification environment for low-level concurrent system code written i...
(annotated with function contracts, state assertions, and type invariants) and attempts to prove the c...
It includes tools for monitoring proof attempts and constructing partial counterexample executions f...
motivates VCC, describes our verification methodology, describes the architecture of VCC, and repor...
to verify the Microsoft Hyper-V hypervisor.

© Springer Verlag Berlin Heidelberg

Verifying the Microsoft Hyper-V Hypervisor with VCC

Authors   **Authors and affiliations**

Dirk Leinenbach, Thomas Santen

Conference paper

PikeOS RTOS & Hypervisor

PikeOS is a real-time operating system that offers a separation kernel-based hypervisor with multiple partitions for many other operating systems and applications. It enables you to build devices for environments with strong demands for Safety and Security. PikeOS is compliant to the highest Safety standards for Avionics & Space, Railway, Automotive, Medical and Industrial Automation markets.
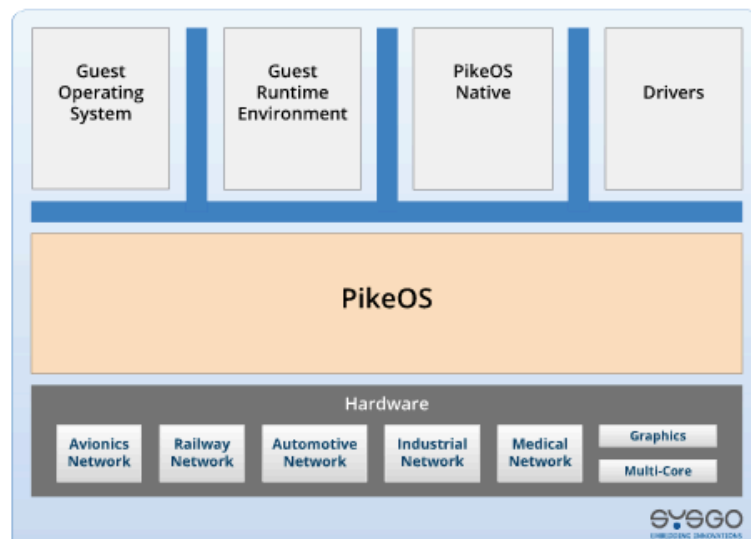
Due to its separation kernel approach, PikeOS is the first choice for systems which demand protection against Cyber-Security attacks. In addition to the broad usage within millions of IoT and edge systems, it has also been deployed within various high critical communication infrastructures.

PikeOS brings together virtualization and real-time by means of unique and never seen before technologies. It allows you to migrate numerous complex embedded circuit boards in to a single hardware. It does not stop when it comes to new hardware concepts such as Big SoCs (Systems-on-a-Chip) with multiple heterogeneous processor cores. Finally, when it comes to certification, SYSGO offers you the right certification kit in order to help you facing the certification authorities.

PikeOS runs on several architectures - supporting processors that come with a Memory Management Unit (MMU) as well as less complex SOCs that contain a → Memory Protection Unit (MPU) only. Please refer to the BSP list for more details.
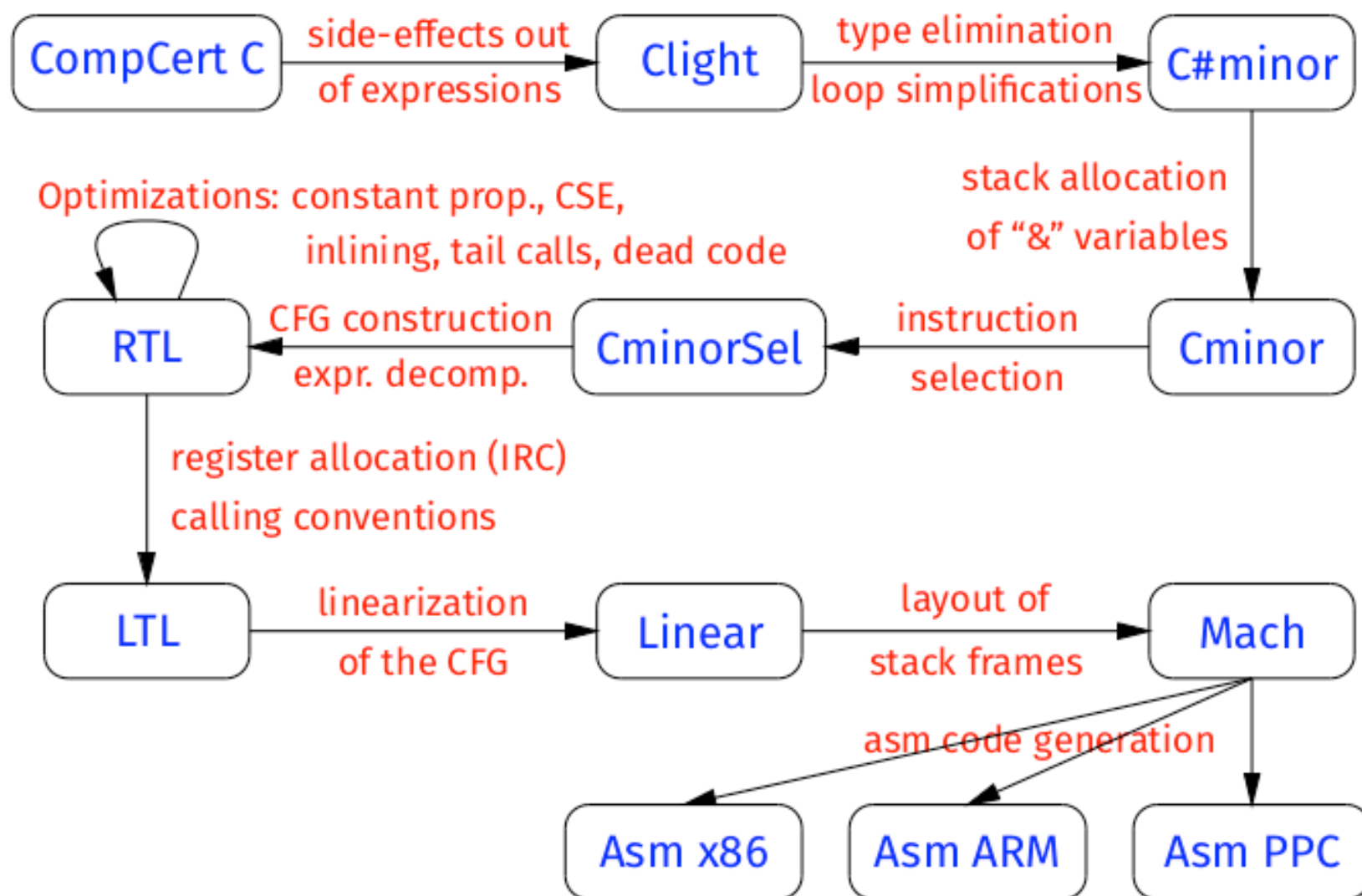
Compcert (Coq) [2009]

THE COMPCERT C COMPILER

[Download CompCert C] [Read the manual]

CompCert C is a compiler for the C programming language. Its in
compilation of life-critical and mission-critical software written
levels of assurance. It accepts most of the ISO C 99 language, wi
few extensions. It produces machine code for the PowerPC, ARM
bits) architectures. Performance of the generated code is decen
PowerPC, about 90% of the performance of GCC version 4 at opt

What sets CompCert C apart from any other production compiler
verified, using machine-assisted mathematical proofs, to be exe
issues. In other words, the executable code it produces is prove
specified by the semantics of the source C program. This level o
correctness of the compilation process is unprecedented and co
highest levels of software assurance. In particular, using the Con
natural complement to applying formal verification techniques (
proof, model checking) at the source code level: the correctnes
guarantees that all safety properties verified on the source code

CompCert C — side-effects out of expressions → Clight — type elimination / loop simplifications → C#minor

Optimizations: constant prop., CSE, inlining, tail calls, dead code

RTL ← CFG construction / expr. decomp. — CminorSel ← instruction selection — Cminor

stack allocation of "&" variables

register allocation (IRC) / calling conventions

LTL — linearization of the CFG → Linear — layout of stack frames → Mach

asm code generation

Asm x86   Asm ARM   Asm PPC

Part 2: Compilation of CompCert C AST to assembly AST. This part is the bulk of the compiler and the one that is proved correct in Coq. It is structured in 16 passes and uses 10 intermediate language, as depicted on the following diagram.
All intermediate languages are given formal semantics, and each of the transformation passes is proved to preserve semantics.

seL4 (Isabelle/HOL) [2010]

The seL4® Microkernel

Security is no excuse for bad performance

The benchmark for p
The world's most hig
Open source & comr

There are two broad approached to formal verification: fully automated methods such as model checking that work on limited systems and properties, and interactive mathematical proof which requires manual effort.

The seL4 verification uses formal mathematical proof in the theorem prover Isabelle/HOL ☑. This theorem prover is interactive, but offers a comparatively high degree of automation. It also offers a very high degree of assurance that the resulting proof is correct.

What does seL4's formal verification mean?

Unique about seL4 is its unprecedented degree of assurance, achieved through formal verification. Specifically, the ARM, ARM_HYP (ARM with virtualisation extensions), and X64 versions of seL4 comprise the first (and still only) general-purpose OS kernel with a full code-level functional correctness proof, meaning a mathematical proof that the implementation (written in C) adheres to its specification. In short, the implementation is proved to be bug-free (see below). This also implies a number of other properties, such as freedom from buffer overflows, null pointer exceptions, use-after-free, etc.

Infer (Abs Int.+) [2010-]

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 source testing correct Faceb approx Faceb than a reports large p

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.

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.

CVS, TLDR:

- **Part I: Verified Software Construction (Dafny)**

  - Functional Correctness, Contract-based approaches (Hoare Logic)

    - Pre- and post-conditions, loop invariants, assertions.

    - Termination metrics.

- **Part II: Concurrent Programming (Java + Verifast)**

  - 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.

- **Part III: Software Testing**

  - Test selection and test generation; Model-based testing; Fault-based testing. Property based testing; Symbolic execution; Automated testing; Tools.