Tools for Reliable Software
  - Multithreaded Software
  - Trustworthy Web Apps

Cormac Flanagan
Software and Programming Languages Laboratory
"The First Bug", 1945

0800  Anden started
1000  stopped - anterior

13°C (032) MP - MC
(033) PRO 2  2.130476915
  convex  2.130476915

Relays 6-2 in 033 failed speed spin test
In relay
  Relays changed

1100  Started cosine tape (Sine check)
1525  Started Multi Adder Test.

1545  Relay #70 Panel F
      (moth) in relay.

First actual case of bug being found.
1630  unabated started.
1700  closed down.
Maurice Wilkes

As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought...

I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs.
As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought... I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs.
Maurice Wilkes

As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought...

I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs.

other people's
Restart and set the recovery options in the system control panel or the \CRASHDEBUG system start option.
The Blue Screen of Death (BSOD)
The Blue Screen of Death (BSOD)
The Blue Screen of Death (BSOD)
this is the first time
start your computer. If
these steps:

eck for viruses on your
rd drives or hard drive
ake sure it is proper
 CHKDSK /F to check for
start your computer.
USS Yorktown

- Smart Ship
  - 27 Pentium-based PCs
  - Windows NT 4.0

- September 21, 1997:
  - data entry error caused a "Divide-By-0" error
  - entire system failed
  - ship dead in the water for over 2 hours

[Wired 1997]
French Guyana, June 4, 1996
$800 million software failure
Mars Climate Orbiter

**Purpose:** Collect data. Relay signals from Mars Polar Lander ($165M)

**Failure:** Smashed into Mars (1999)

**Bug:** Failed to convert English to metric units
**Mars Climate Orbiter**

**Purpose:** Collect data. Relay signals from Mars Polar Lander ($165M)

**Failure:** Smashed into Mars (1999)

**Bug:** Failed to convert English to metric units

---

**Mars Polar Lander**

**Purpose:** Lander to study the Mars climate ($120M)

**Failure:** Smashed into Mars (2000)

**Bug:** Spurious signals from sensors caused premature engine shutoff
<table>
<thead>
<tr>
<th><strong>Mars Climate Orbiter</strong></th>
<th><strong>Mars Polar Lander</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Collect data. Relay signals from Mars Polar Lander ($165M)</td>
<td><strong>Purpose:</strong> Lander to study the Mars climate ($120M)</td>
</tr>
<tr>
<td><strong>Failure:</strong> Smashed into Mars (1999)</td>
<td><strong>Failure:</strong> Smashed into Mars (2000)</td>
</tr>
<tr>
<td><strong>Bug:</strong> Failed to convert English to metric units</td>
<td><strong>Bug:</strong> Spurious signals from sensors caused premature engine shutoff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Therac25 Radiation Therapy</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Computer-controlled radiation therapy machine</td>
<td></td>
</tr>
<tr>
<td><strong>Failure:</strong> gave fatal radiation doses to 2 cancer patients (1986)</td>
<td></td>
</tr>
<tr>
<td><strong>Bug:</strong> timing bug</td>
<td></td>
</tr>
</tbody>
</table>
### Mars Climate Orbiter

**Purpose:** Collect data.
Relay signals from Mars Polar Lander ($165M)

**Failure:** Smashed into Mars (1999)

**Bug:** Failed to convert English to metric units

### Mars Polar Lander

**Purpose:** Lander to study the Mars climate ($120M)

**Failure:** Smashed into Mars (2000)

**Bug:** Spurious signals from sensors caused premature engine shutoff

### Therac25 Radiation Therapy

**Purpose:** Computer-controlled radiation therapy machine

**Failure:** gave fatal radiation doses to 2 cancer patients (1986)

**Bug:** timing bug

### USS Vincennes

**Failure:** Shot down an Airbus jet that was mistaken for a F-14. 290 people died. (1988)

**Bug:** tracking software displayed cryptic and misleading output
**Mars Climate Orbiter**

**Purpose:** Collect data. Relay signals from Mars Polar Lander ($165M)

**Failure:** Smashed into Mars (1999)

**Bug:** Failed to convert English to metric units

---

**Mars Polar Lander**

**Purpose:** Lander to study the Mars climate ($120M)

**Failure:** Smashed into Mars (2000)

**Bug:** Spurious signals from sensors caused premature engine shutoff

---

**Therac25 Radiation Therapy**

**Purpose:** Computer-controlled radiation therapy machine

**Failure:** Gave fatal radiation doses to 2 cancer patients (1986)

**Bug:** Timing bug

---

**USS Vincennes**

**Failure:** Shot down an Airbus jet that was mistaken for a F-14. 290 people died. (1988)

**Bug:** Tracking software displayed cryptic and misleading output

---

**Software defects cost $60 billion/year**  
[**NIST 03**]
How to Develop More Reliable Software

• “Correct by construction”
  - use better languages to preclude certain errors in new software
    ▪ type systems for multithreaded software
    ▪ information flow for browser security

• “Post hoc validation”
  - better tools for detecting defects in existing software
    ▪ static analysis and/or verification
    ▪ dynamic analysis
Multithreading and Multicore

Multithreaded programming is notoriously difficult, in part due to schedule-dependent behavior

- race conditions, deadlocks, atomicity violations, ...
- difficult to detect, reproduce, or eliminate
Race Conditions

- Two threads access shared variable without synchronization, at least one thread does a write
- Very common

2003 Blackout ($6 Billion)

Mars Rovers

Therac-25
package java.util;
...

class Vector {
    Object elementData[];
    int elementCount;

    synchronized int lastIndexOfOf(Object elem, int n) {
        for (int i = n; i >= 0; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    int lastIndexOfOf(Object elem) {
        return lastIndexOfOf(elem, elementCount - 1);
    }

    synchronized void trimToSize() { ... }
    synchronized boolean remove(int index) { ... }
}
A Type System to Guarantee Race Freedom

package java.util;
...

class Vector {
    Object elementData[] guarded_by this ;
    int elementCount guarded_by this ;

    synchronized int lastIndexOf(Object elem, int n) {
        for (int i = n ; i >= 0 ; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    int lastIndexOf(Object elem) {
        return lastIndexOf(elem, elementCount - 1);
    }

    synchronized void trimToSize() { ... }
    synchronized boolean remove(int index) { ... }
package java.util;

class Vector {
    Object elementData[];  // guarded_by this
    int elementCount;      // guarded_by this

    synchronized int lastIndexOf(Object elem, int n) {
        for (int i = n; i >= 0; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    synchronized void trimToSize() {
        // ...
    }

    synchronized boolean remove(int index) {
        // ...
    }

    int lastIndexOf(Object elem) {
        return lastIndexOf(elem, elementCount - 1);
    }
}
package java.util;
...

class Vector {
    Object elementData[
    int elementCount

    synchronized int lastIndexOf(Object elem, int n) {
        for (int i = n ; i >= 0 ; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    int lastIndexOf(Object elem) {
        return lastIndexOf(elem, elementCount - 1);
    }

    synchronized void trimToSize() { ... }
    synchronized boolean remove(int index) { ... }

    guarded_by this
    guarded_by this

    Type Error: Race on line 73
package java.util;
...

class Vector {
    Object elementData[];  // guarded_by this
    int elementCount;      // guarded_by this

    synchronized int lastIndexOf(Object elem, int n) {
        for (int i = n; i >= 0; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    int lastIndexOf(Object elem) {
        return lastIndexOf(elem, elementCount - 1);
    }

    synchronized void trimToSize() {
        ...}

    synchronized boolean remove(int index) {
        ...}

    Type Error: Race on line 73

    Program Crash:
    Uncaught IndexOutOfBoundsException at line 52
...
Dynamic Race Detection

- Precision
- Performance

Happens Before
[Lamport 78]

Eraser
[SBN+ 97]
Dynamic Race Detection

- Eraser [SBN+ 97]
- Barriers [PS 03]
- Initialization [vPG 01]
- ...
Dynamic Race Detection

Vector Clocks [M 88]
Goldilocks [EQT 07]
DJIT+ [ISZ 99, PS 03]
TRADe [CB 01]
...

Happens Before
[Lamport 78]

Barriers [PS 03]
Initialization [vPG 01]
...

Eraser [SBN+ 97]
Dynamic Race Detection

- Vector Clocks [M 88]
- Goldilocks [EQT 07]
- Vector Clocks [M 88]
- DJIT+ [ISZ 99, PS 03]
- TRaDe [CB 01]
- MultiRace [PS 03]
- Hybrid Race Detector [OC 03]
- RaceTrack [YRC 05]
- Eraser [SBN+ 97]
- Barriers
- Initialization
- Happens Before [Lamport 78]
Dynamic Race Detection

- FastTrack [PLDI’09]
- Vector Clocks [M 88]
- Out-of-order clocks [EQT 07]
- TRaDe [CB 01]
- RaceTrack [YRC 05]
- MultiRace [PS 03]
- DJIT+ [ISZ 99, PS 03]
- Hybrid Race Detector [OC 03]
- Eraser [SBN+ 97]

Performance

- Happens Before [Lamport 78]

Precision
Traditional HB Race Detectors: $O(n)$ overhead

Thread A

Thread B

Thread C

Thread D

$x = 0$

$x = 1$

read $x$

$x = 3$

$O(n)$
No Races Yet: Writes Totally Ordered!
FastTrack: $O(1)$ Time, Space Overhead

Thread A  Thread B  Thread C  Thread D

$x = 0$  $x = 1$  $x = 1$  $x = 3$

read $x$  \( O(1) \)
FastTrack: $O(1)$ Time, Space Overhead

Thread A

Thread B

$\text{read } x$

$O(1)$
Race Freedom is Not Sufficient

public StringBuffer {
    private int count;  // guarded by this
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }

    public synchronized void append(StringBuffer sb){
        int len = sb.length();
        ...
        ...
        sb.getChars(..., len,...);
        ...
    }
}

Race Freedom is Not Sufficient

```java
public StringBuffer {
    private int count;  // guarded_by this
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }
    public synchronized void append(StringBuffer sb){
        int len = sb.length();
        ...
        ...
        sb.getChars(...,len,...);
        ...
    }
}
```
Race Freedom is Not Sufficient

```java
public StringBuffer {
    private int count;  // guarded_by this
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }

    public synchronized void append(StringBuffer sb) {
        int len = sb.length();
        ...  // sb.length() acquires the lock on sb, gets the length, and releases lock
        ...  // other threads can change sb
        sb.getChars(..., len, ...);
        ...
    }
}
```
Race Freedom is Not Sufficient

```java
public StringBuffer {
    private int count;  // guarded_by this
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }

    public synchronized void append(StringBuffer sb)
    {
        int len = sb.length();
        ...
        ...
        sb.getChars(..., len,...);
        ...
    }  
}
```

- `sb.length()` acquires the lock on `sb`, gets the length, and releases lock
- Other threads can change `sb`
- Use of stale `len` may yield `StringIndexOutOfBoundsException` inside `getChars(...)`
Atomicity

public StringBuffer {
    private int count;
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }
    atomic
    public synchronized void append(StringBuffer sb) {

        int len = sb.length();
        ...
        ...
        ... sb.getChars(..., len,...);
        ...
    }
}
Atomicity

public StringBuffer {
    private int count;
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }
    atomic
    public synchronized void append(StringBuffer sb) {
        int len = sb.length();
        ...
        ...
        sb.getChars(..., len,...);
        ...
    }
}
**Atomicity**

```java
public class StringBuffer {
    private int count;
    public synchronized int length() { return count; }
    public synchronized void getChars(...) { ... }

    public synchronized void append(StringBuffer sb) {
        int len = sb.length();
        ...
        ...
        sb.getChars(..., len, ...);
        ...
    }
}
```

**Atomicity Specification**
- No thread interference
- Enables sequential reasoning
- Amenable to automatic verification
- Violations catch defects
- 90% of Java methods atomic

**StringBuffer.append is not atomic:**
- **Start:**
  - at StringBuffer.append(StringBuffer) at Thread1.run(Example.java:17)
- **Commit:** Lock Release
  - at StringBuffer.length(StringBuffer) at StringBuffer.append(StringBuffer) at Thread1.run(Example.java:17)
- **Error:** Lock Acquire
  - at StringBuffer.getChars(StringBuffer) at StringBuffer.append(StringBuffer) at Thread1.run(Example.java:17)
Towards Trustworthy Web Apps
Towards Trustworthy Web Apps

bank.com

confidential information

Mozilla Firefox®
Towards Trustworthy Web Apps

bank.com

confidential information

Mozilla Firefox

gmail.com
Towards Trustworthy Web Apps

confidential information

bank.com

gmail.com

Mozilla Firefox

adserver.com
Towards Trustworthy Web Apps

- bank.com
- gadgets.com
- gmail.com
- adserver.com
- ads-evil.com

confidential information
Towards Trustworthy Web Apps

bank.com

gadgets.com

gmail.com

adserver.com

gadgets-bad.com

ads-evil.com

Mozilla Firefox

confidential information
Towards Trustworthy Web Apps

- bank.com
- gadgets.com
- gmail.com
- adserver.com
- gadgets-bad.com
- ads-evil.com

C++ code

confidential information

Mozilla Firefox®
Towards Trustworthy Web Apps

JavaScript Engine

C++ code

bank.com

gadgets.com

gmail.com

adserver.com

gadgets-bad.com

ads-evil.com

confidential information
Towards Trustworthy Web Apps

Firefox JS code (UI, config, ...)

JavaScript Engine

C++ code
Towards Trustworthy Web Apps

- bank.com
- gadgets.com
- gmail.com
- adserver.com
- gadgets-bad.com
- ads-evil.com

Mozilla Firefox

- Firefox JS code (UI, config, ...)
- Grease-Monkey scripts
- JavaScript Engine
- C++ code
Towards Trustworthy Web Apps

- gadgets.com
- adserver.com
- gadgets-bad.com
- ads-evil.com
- bank.com
- gmail.com
- NYT
- JS code for adverts
- JS mashup, several gadgets
- Banking web app
- Grease-Monkey scripts
- Firefox JS code (UI, config, ...)
- Confidential information
- JavaScript Engine
- C++ code
Towards Trustworthy Web Apps

- **gadgets.com**
- **gmail.com**
- **adserver.com**
- **ads-evil.com**

- **bank.com**
  - JS mashup, several gadgets
  - Banking web app
  - NYT
  - JS code for adverts

- **Firefox JS code (UI, config, ...)**
- **Grease-Monkey scripts**
- **JavaScript Engine**
- **C++ code**

Confidential information
Towards Trustworthy Web Apps

- gadgets.com
- gmail.com
- adserver.com
- gadgets-bad.com
- ads-evil.com
- bank.com
- banking.web app
- NYT
- JS code for adverts
- JS mashup, several gadgets
- FT
- firefox.js code (UI, config, ...)
- Grease-Monkey scripts
- confidential information

Buffer overruns
C, C++ code
Towards Trustworthy Web Apps

- gadgets.com
- adserver.com
- gmail.com
- gadgets-bad.com
- ads-evil.com

- JS mashup, several gadgets
- Banking web app
- NYT JS mashup, several gadgets
- JS code for adverts
- JS code (UI, config, ...)
- Grease-Monkey scripts

- memory safe Javascript
- Firefox JS code

- buffer overruns

- C, C++ code

- confidential information
Same Origin Policy - Which Origin for Ads?
Same Origin Policy—Which Origin for Ads?
Same Origin Policy - Which Origin for Ads?

doubleclick.com

nytimes.com
Same Origin Policy—Which Origin for Ads?

doubleclick.com

linkstorm.net

nytimes.com
Same Origin Policy - Which origin for Mash-up?
Same Origin Policy—Which origin for Mash-up?

Lots of gadgets, lots of domains, lots of flexibility, lots of security worries.
JavaScript for Trustworthy Web Apps

- Collaboration with UCI and Mozilla (CTO Eich)
- Extend Javascript to enforce security policies
  - Add provenance label to all data (including DOM)
    - where did it come from, who influenced it, how sensitive is it, who can see it?
  - JS engine tracks provenance through all JS code
    - minimal changes to existing web apps
  - ... and enforces composable, configurable policies
    - eg data from bank.com never sent to other servers
- Malicious or faulty JS code cannot violate policies
JavaScript for Trustworthy Web Apps

- Collaboration with UCI and Mozilla (CTO Eich)

- Extend Javascript to enforce security policies
  - Add provenance label to all data (including DOM)
    - where did it come from, who influenced it, how sensitive is it, who can see it?
  - JS engine tracks provenance through all JS code
    - minimal changes to existing web apps
    - ... and enforces composable, configurable policies
      - eg data from bank.com never sent to other servers

- Malicious or faulty JS code cannot violate policies
JavaScript for Trustworthy Web Apps

- Collaboration with UCI and Mozilla (CTO Eich)
- Extend JavaScript to enforce security policies
  - Add provenance label to all data (including DOM)
    - where did it come from, who influenced it, how sensitive is it, who can see it?
  - JS engine tracks provenance through all JS code
    - minimal changes to existing web apps
    - ... and enforces composable, configurable policies
      - eg data from bank.com never sent to other servers

- Malicious or faulty JS code cannot violate policies
JavaScript for Trustworthy Web Apps

- Collaboration with UCI and Mozilla (CTO Eich)
- Extend JavaScript to enforce security policies
  - Add provenance label to all data (including DOM)
    - where did it come from, who influenced it, how sensitive is it, who can see it?
    - JS engine tracks provenance through all JS code
  - minimal changes to existing web apps
  - ... and enforces composable, configurable policies
    - eg data from bank.com never sent to other servers

- Malicious or faulty JS code cannot violate policies
Summary

French Guyana, June 4, 1996
$800 million software failure
Summary

French Guyana, June 4, 1996
$800 million software failure
Summary

French Guyana, June 4, 1996
$800 million software failure

$60+ billion/year
Summary

French Guyana, June 4, 1996
$800 million software failure

$60+ billion/year
Summary

French Guyana, June 4, 1996
$800 million software failure

$60+ billion/year

Dynamic Race Detection

A Type System for Race Freedom

```java
package java.util;
...

class Vector {
    Object elementData[]    guarded_by this;
    int elementCount    guarded_by this;

    synchronized int lastIndexOf(Object elem, int n) {
        for (int i = n; i >= 0; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    int lastIndexOf(Object elem) {
        return lastIndexOf(elem, elementCount - 1);
    }

    synchronized void trimToSize() { ... }
    synchronized boolean remove(int index) { ... }
```
Summary

French Guyana, June 4, 1996
$800 million software failure

$60+ billion/year

Dynamic Race Detection

FastTrack [PLDI'09]
Vector Clocks [M 86, YRC 05]
Race Monitors [EQT 07]

A Type System for Race Freedom

package java.util;

... Class Vector {
  Object elementData[] guarded_by this;
  int elementCount guarded_by this;
  synchronized int lastIndexOf(Object elem, int n) {
    for (int i = n; i >= 0; i--)
      if (elem.equals(elementData[i])) return i;
    return -1;
  }
  int lastIndexOf(Object elem) {
    return lastIndexOf(elem, elementCount - 1);
  }
  synchronized void trimToSize() { ... }
  synchronized boolean remove(int index) { ... }
}

Velodrome Atomicity Checker [PLDI'08]

• Builds precise HB graph over all ops
• Looks for cycles
• Which indicate atomicity violations
• Fewer false alarms than earlier tools

Cycle in transactional HB order
⇒ trace is not serializable
⇒ report atomicity violation
Summary

$60+ billion/year

A Type System for Race Freedom

```java
package java.util;

class Vector {
    Object elementData[]
    int elementCount

    synchronized int lastIndexOf(Object elem, int n) {
        for (int i = n; i >= 0; i--)
            if (elem.equals(elementData[i])) return i;
        return -1;
    }

    int lastIndexOf(Object elem) {
        return lastIndexOf(elem, elementCount - 1);
    }

    synchronized void trimToSize() {
    }
    synchronized boolean remove(int index) {
    }
}
```

Velodrome Atomicity Checker [PLDI'08]

- Builds precise HB graph over all ops
- Looks for cycles
- Which indicate atomicity violations
- Fewer false alarms than earlier tools

Cycle in transactional HB order
⇒ trace is not serializable
⇒ report atomicity violation
Summary

French Guyana, June 4, 1996
$800 million software failure

$60+ billion/year

Velodrome Atomicity Checker [PLDI'08]
- Builds precise HB graph over all ops
- Looks for cycles
- Which indicate atomicity violations
- Fewer false alarms than earlier tools

Cycle in transactional HB order
⇒ trace is not serializable
⇒ report atomicity violation