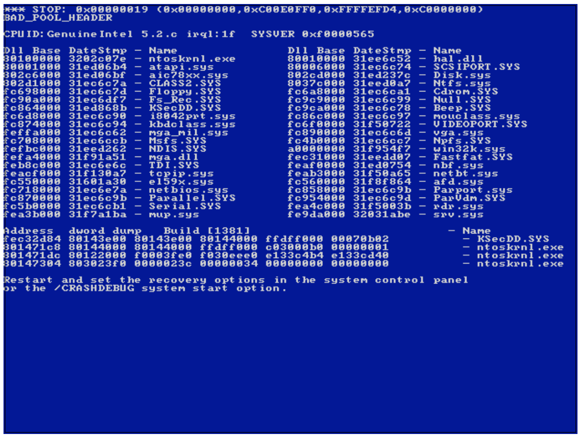
Squashing the Bugs: Tools for Building Better Software

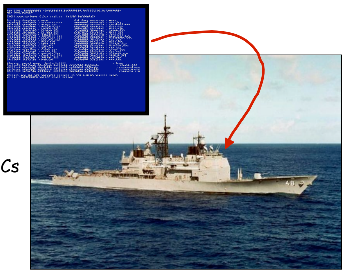
Stephen Freund
Williams College



```
*** STOP: 0x00000019 (0x00000000,0xC00E0FF0,0xFFFFEFD4,0xC0000000)
BAD_POOL_HEADER

CPUID:GenuineIntel 5.2.c irql:1f   SYSVER 0xf0000565

Dll Base DateStmp - Name                   Dll Base DateStmp - Name
80100000 3202c07e - ntoskrnl.exe           80010000 31ee6c52 - hal.dll
80001000 31ed06b4 - atapi.sys              80006000 31ec6c74 - SCSIPORT.SYS
802c6000 31ed06bf - aic78xx.sys            802cd000 31ed237c - Disk.sys
802d1000 31ec6c7a - CLASS2.SYS             8037c000 31eed0a7 - Ntfs.sys
fc698000 31ec6c7d - Floppy.SYS             fc6a8000 31ec6ca1 - Cdrom.SYS
fc90a000 31ec6df7 - Fs_Rec.SYS             fc9c9000 31ec6c99 - Null.SYS
fc864000 31ed868b - KSecDD.SYS             fc9ca000 31ec6c78 - Beep.SYS
fc6d8000 31ec6c90 - i8042prt.sys           fc86c000 31ec6c97 - mouclass.sys
fc874000 31ec6c94 - kbdclass.sys           fc6f0000 31f50722 - VIDEOPORT.SYS
feffa000 31ec6c62 - mga_mil.sys            fc890000 31ec6c6d - vga.sys
fc708000 31ec6ccb - Msfs.SYS               fc4b0000 31ec6cc7 - Npfs.SYS
fefbc000 31eed262 - NDIS.SYS               a0000000 31f954f7 - win32k.sys
fefa4000 31f91a51 - mga.dll                fec31000 31eedd07 - Fastfat.SYS
feb8c000 31ec6e6c - TDI.SYS                feaf0000 31ed0754 - nbf.sys
feacf000 31f130a7 - tcpip.sys              feab3000 31f50a65 - netbt.sys
fc550000 31601a30 - el59x.sys              fc560000 31f8f864 - afd.sys
fc718000 31ec6e7a - netbios.sys            fc858000 31ec6c9b - Parport.sys
fc870000 31ec6c9b - Parallel.SYS           fc954000 31ec6c9d - ParVdm.SYS
fc5b0000 31ec6cb1 - Serial.SYS             fea4c000 31f5003b - rdr.sys
fea3b000 31f7a1ba - mup.sys                fe9da000 32031abe - srv.sys

Address  dword dump   Build [1381]                            - Name
fec32d84 80143e00 80143e00 80144000 ffdff000 00070b02          - KSecDD.SYS
801471c8 80144000 80144000 ffdff000 c03000b0 00000001          - ntoskrnl.exe
801471dc 80122000 f0003fe0 f030eee0 e133c4b4 e133cd40          - ntoskrnl.exe
80147304 803023f0 0000023c 00000034 00000000 00000000          - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option.
```

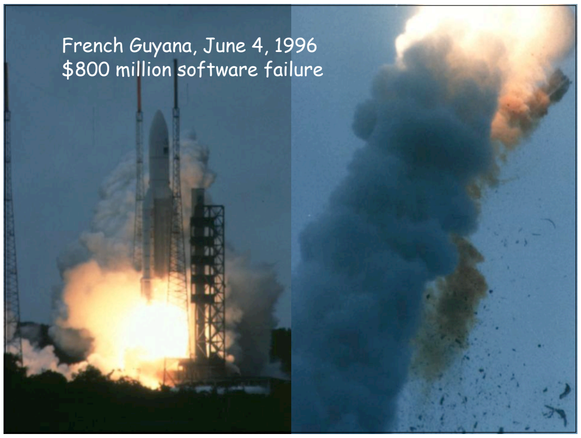
The Blue Screen of Death (BSOD)

USS Yorktown

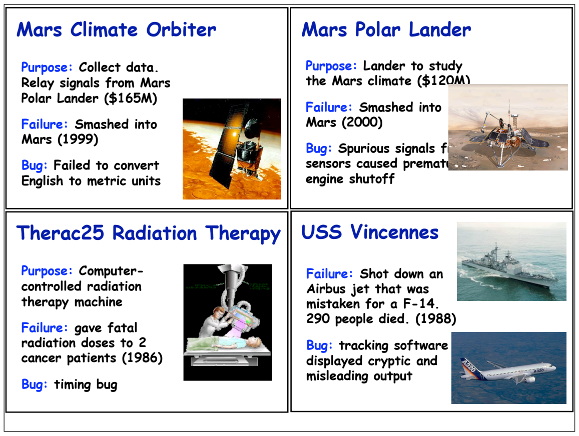


- Smart Ship
  - 27 Pentium-based PCs
  - Windows NT 4.0
- September 21, 1997:
  - data entry error caused a "Divide-By-0" error
  - entire system failed
  - ship dead in the water for over 2 hours

[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 Polar Lander

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

Failure: Smashed into Mars (2000)

Bug: Spurious signals f sensors caused premat engine shutoff

Therac25 Radiation Therapy

Purpose: Computer-controlled radiation therapy machine

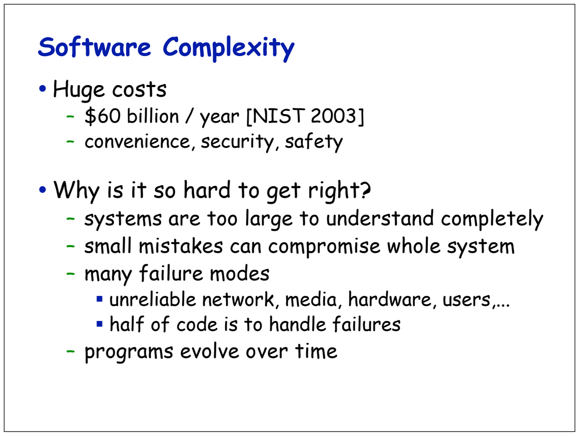
Failure: gave fatal radiation doses to 2 cancer patients (1986)

Bug: timing bug

USS Vincennes

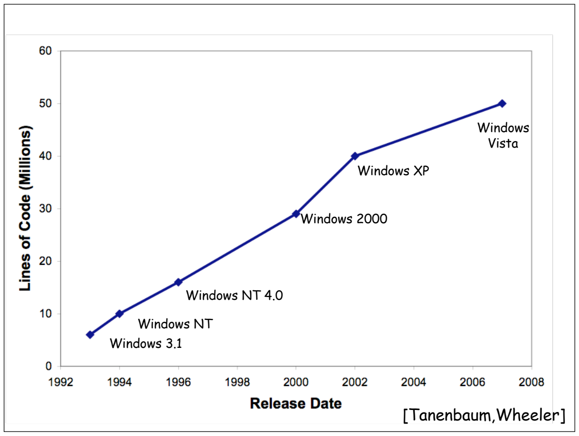
Failure: Shot down an Airbus jet that was mistaken for a F-14. 290 people died. (1988)

Bug: tracking software displayed cryptic and misleading output

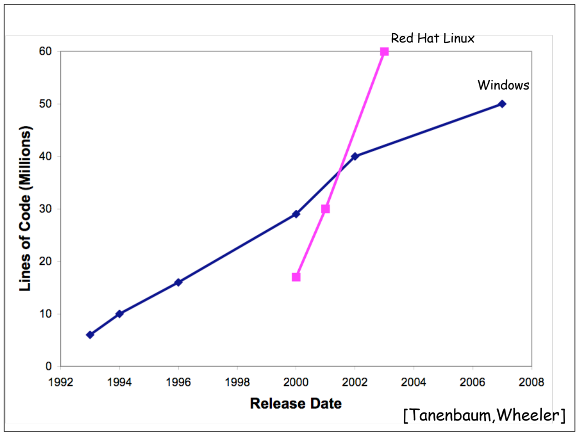


Software Complexity

- Huge costs
  - $60 billion / year [NIST 2003]
  - convenience, security, safety
- Why is it so hard to get right?
  - systems are too large to understand completely
  - small mistakes can compromise whole system
  - many failure modes
    - unreliable network, media, hardware, users,...
    - half of code is to handle failures
  - programs evolve over time

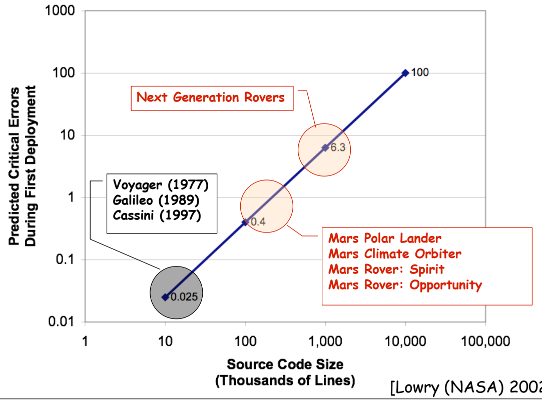


[Tanenbaum,Wheeler]



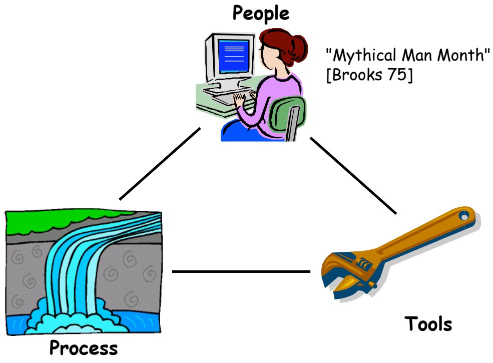
[Tanenbaum,Wheeler]

400 horses
100 microprocessors



[Lowry (NASA) 2002]

Managing Software Complexity

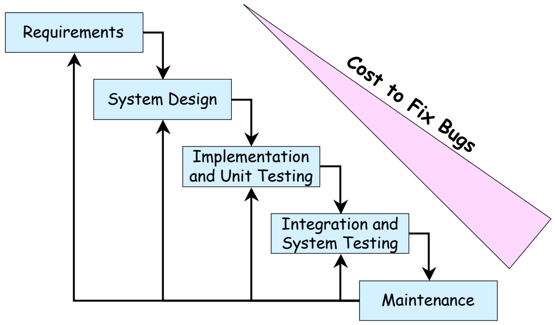


"Mythical Man Month"
[Brooks 75]

Software Processes

- Organization of large software project
  - other examples: army, company, college admin
  - planning, responsibilities, interactions
- Benefits
  - avoid miscommunication, misunderstandings
  - predict time and cost
  - recognize problems early
- Process failures do happen (frequently...)
  - Air Traffic Control system, $2.6Billion, 1983-1996

Waterfall Model



Requirements

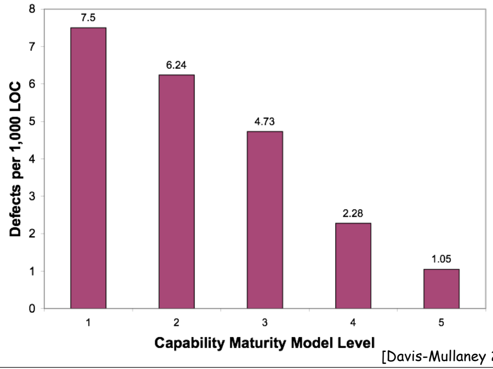
System Design

Implementation and Unit Testing

Integration and System Testing

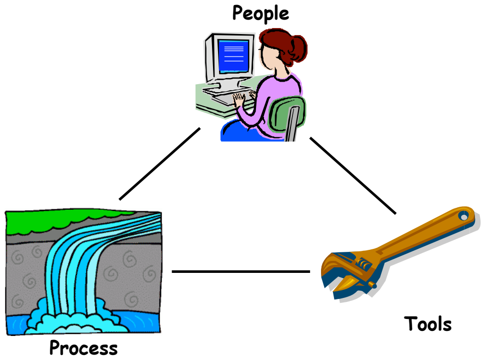
Maintenance

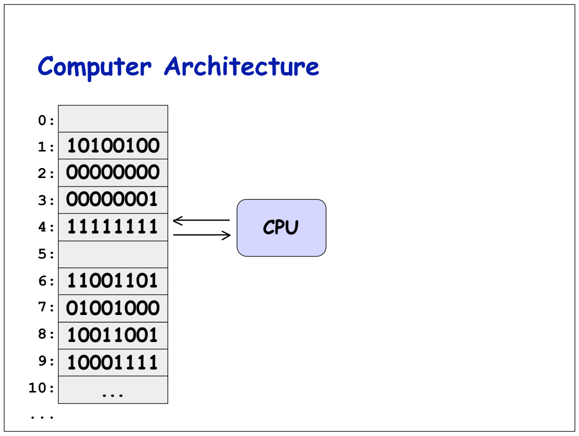
Defect Density in Delivered Software



[Davis-Mullaney 2003]

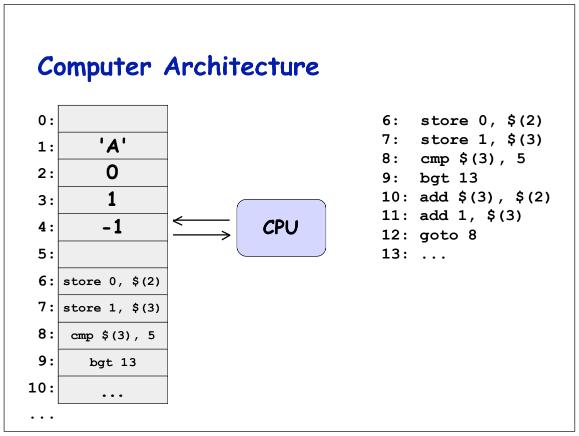
Managing Software Complexity





Computer Architecture

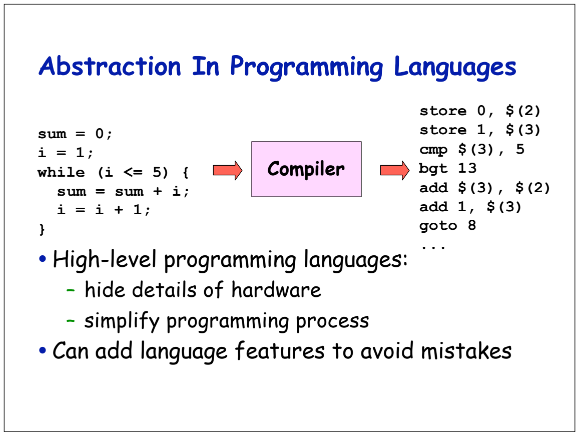
0:	
1:	10100100
2:	00000000
3:	00000001
4:	11111111
5:	
6:	11001101
7:	01001000
8:	10011001
9:	10001111
10:	...
...	



Computer Architecture

0:	
1:	'A'
2:	0
3:	1
4:	-1
5:	
6:	store 0, $(2)
7:	store 1, $(3)
8:	cmp $(3), 5
9:	bgt 13
10:	...
...	

```
6:  store 0, $(2)
7:  store 1, $(3)
8:  cmp $(3), 5
9:  bgt 13
10: add $(3), $(2)
11: add 1, $(3)
12: goto 8
13: ...
```

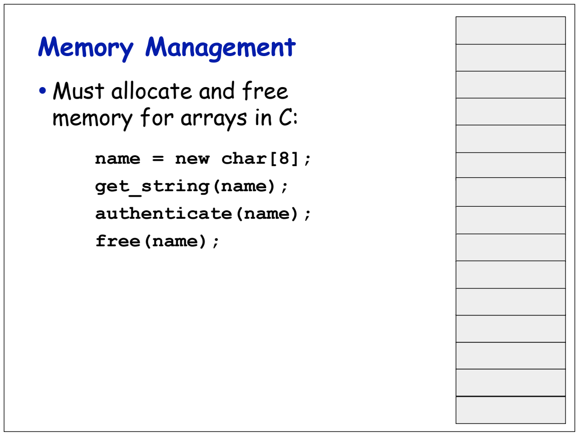


Abstraction In Programming Languages

```
sum = 0;
i = 1;
while (i <= 5) {
  sum = sum + i;
  i = i + 1;
}
```

```
store 0, $(2)
store 1, $(3)
cmp $(3), 5
bgt 13
add $(3), $(2)
add 1, $(3)
goto 8
...
```

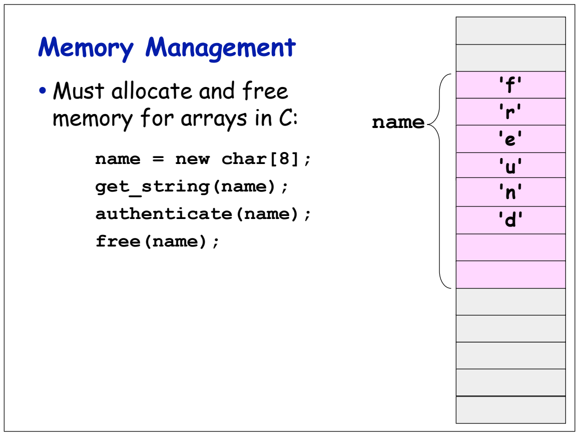
- High-level programming languages:
  - hide details of hardware
  - simplify programming process
- Can add language features to avoid mistakes



Memory Management

- Must allocate and free memory for arrays in C:

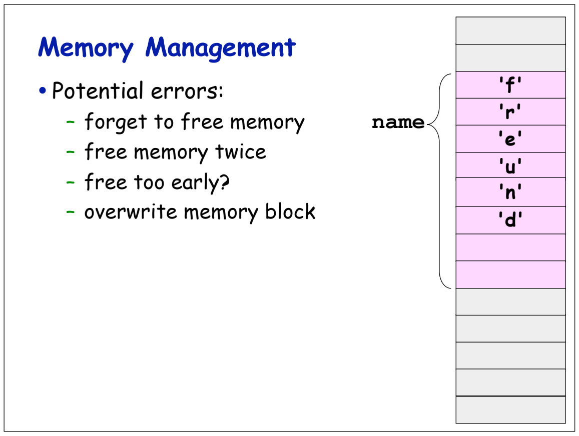
```
name = new char[8];
get_string(name);
authenticate(name);
free(name);
```



Memory Management

- Must allocate and free memory for arrays in C:

```
name = new char[8];
get_string(name);
authenticate(name);
free(name);
```



Memory Management

- Potential errors:
  - forget to free memory
  - free memory twice
  - free too early?
  - overwrite memory block

Garbage Collection

- Never explicitly free memory

```
name = new char[8];
get_string(name);
authenticate(name);
```

- Program periodically pauses to find and reclaim unused memory
- Used in Java, C#, Python, ...

Tradeoffs in Design

- No free lunch
  - features impact performance / expressiveness
  - can be prohibitively expensive (or impossible!)
- Other features
  - object-oriented languages / modules
  - exception handling
  - threads

Program Checking

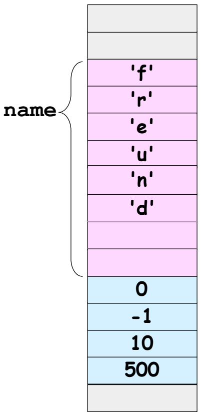
- Design algorithm to automatically identify errors
- Example 1: type errors

```
WebPage w = new WebPage("http://...");
int x = w - 3;  ← BAD

String s = "hello";
if (s < 100) ...  ← BAD
```

- Example 2: buffer overruns in C programs

Buffer Overruns in C

```
name = new char[8];
get_string(name);
authenticate(name);
free(name);
```



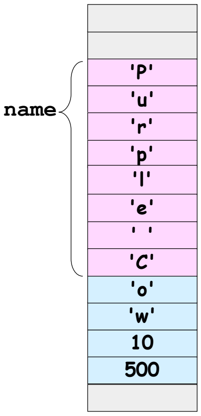
name

'f'
'r'
'e'
'u'
'n'
'd'

0
-1
10
500

Buffer Overruns in C

```
name = new char[8];
get_string(name);
authenticate(name);
free(name);
```

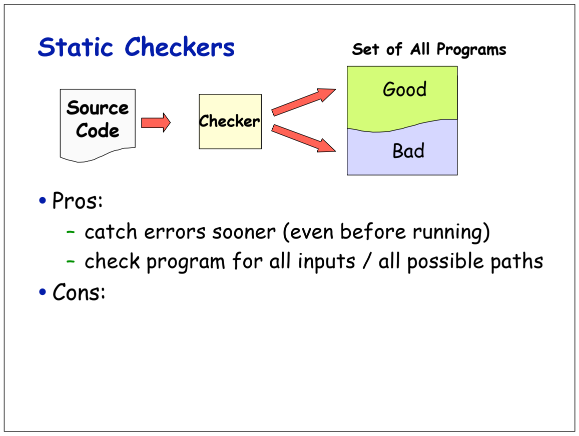


name

'P'
'u'
'r'
'p'
'l'
'e'
' '
'C'
'o'
'w'
10
500

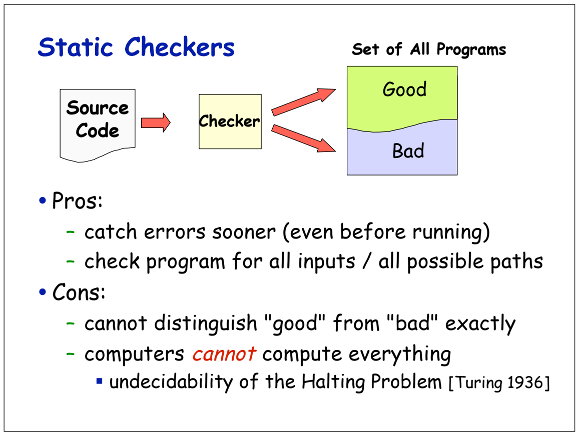
Dynamic Monitors

- Check program as it executes
  - example: buffer overruns
- Pros:
  - identify cause of bug faster than testing
  - easy to add to development process
- Cons:
  - coverage
  - performance
  - must run whole program



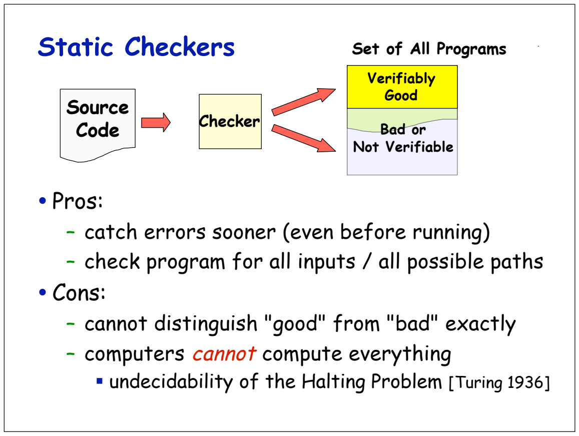
Static Checkers

- Pros:
  - catch errors sooner (even before running)
  - check program for all inputs / all possible paths
- Cons:



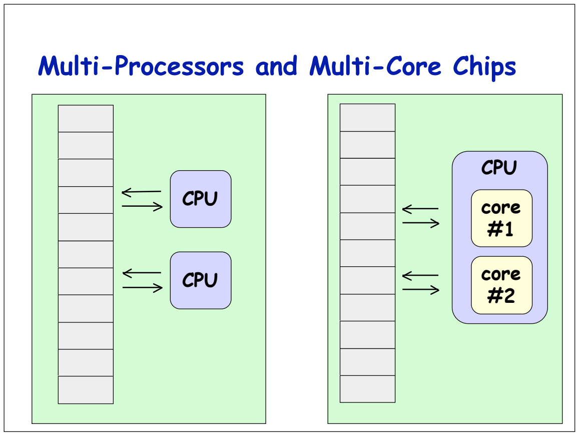
Static Checkers

- Pros:
  - catch errors sooner (even before running)
  - check program for all inputs / all possible paths
- Cons:
  - cannot distinguish "good" from "bad" exactly
  - computers *cannot* compute everything
    - undecidability of the Halting Problem [Turing 1936]

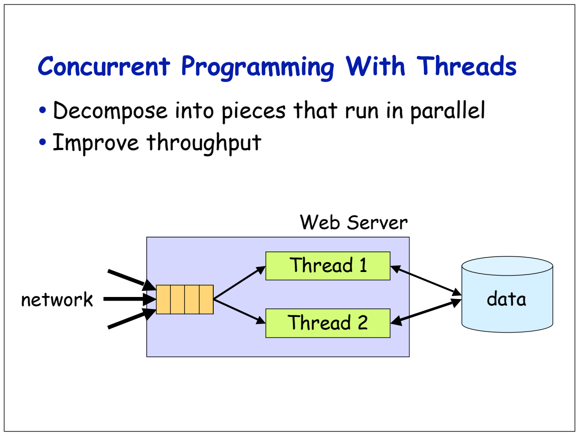


Static Checkers

- Pros:
  - catch errors sooner (even before running)
  - check program for all inputs / all possible paths
- Cons:
  - cannot distinguish "good" from "bad" exactly
  - computers *cannot* compute everything
    - undecidability of the Halting Problem [Turing 1936]



Multi-Processors and Multi-Core Chips



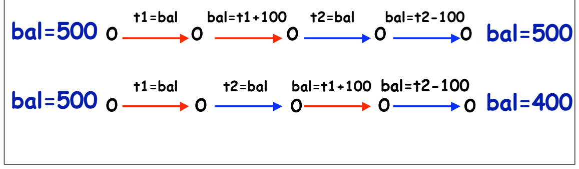
Concurrent Programming With Threads

- Decompose into pieces that run in parallel
- Improve throughput

Demo

Multithreaded Program Execution

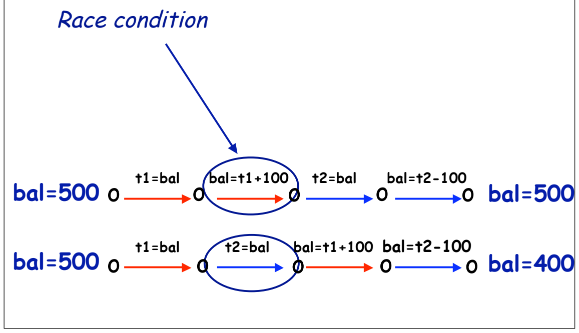
```
Thread 1              Thread 2

  ...                   ...
  t1 = bal;             t2 = bal;
  bal = t1 + 100;       bal = t2 - 100;
  ...                   ...
```



bal=500 t1=bal bal=t1+100 t2=bal bal=t2-100 bal=500

bal=500 t1=bal t2=bal bal=t1+100 bal=t2-100 bal=400

Multithreaded Program Execution



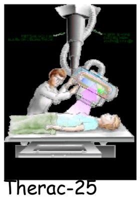
Race condition

bal=500 t1=bal bal=t1+100 t2=bal bal=t2-100 bal=500

bal=500 t1=bal t2=bal bal=t1+100 bal=t2-100 bal=400

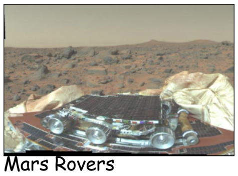
Race Conditions

- Common
- Hard to find via testing
  - scheduler dependent
- Memory, files, printers, ...



Therac-25

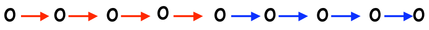
2003 Blackout ($6 Billion)



Mars Rovers

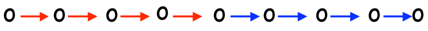
Preventing Race Conditions Using Locks

```
Thread 1              Thread 2
  acquire(m);           acquire(m);
  t1 = bal;             t2 = bal;
  bal = t1 + 100;       bal = t2 - 100;
  release(m);           release(m);
```

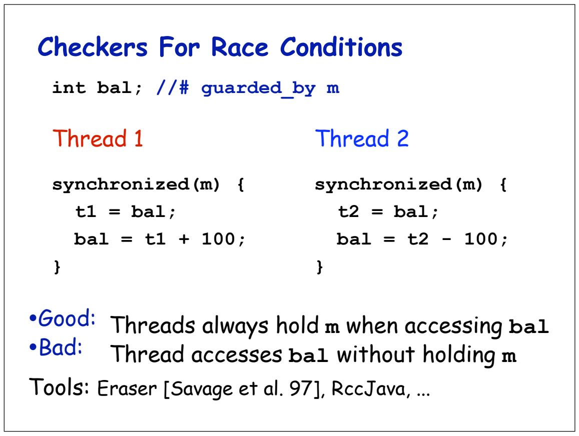


Preventing Race Conditions Using Locks

```
Thread 1              Thread 2
synchronized(m) {     synchronized(m) {
  t1 = bal;             t2 = bal;
  bal = t1 + 100;       bal = t2 - 100;
}                     }
```



Demo



Checkers For Race Conditions

```
int bal; //# guarded_by m
```

Thread 1

```
synchronized(m) {
  t1 = bal;
  bal = t1 + 100;
}
```

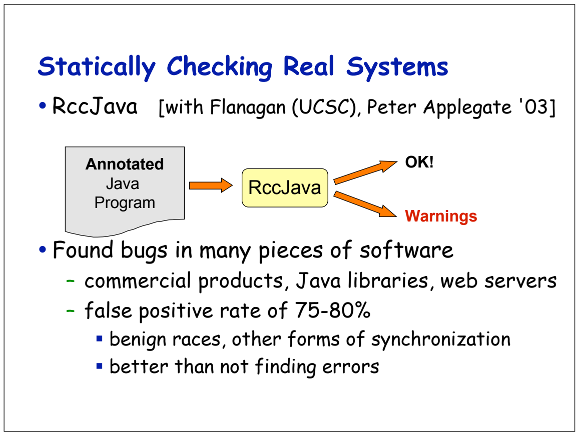
Thread 2

```
synchronized(m) {
  t2 = bal;
  bal = t2 - 100;
}
```

•Good: Threads always hold `m` when accessing `bal`

•Bad: Thread accesses `bal` without holding `m`

Tools: Eraser [Savage et al. 97], RccJava, ...

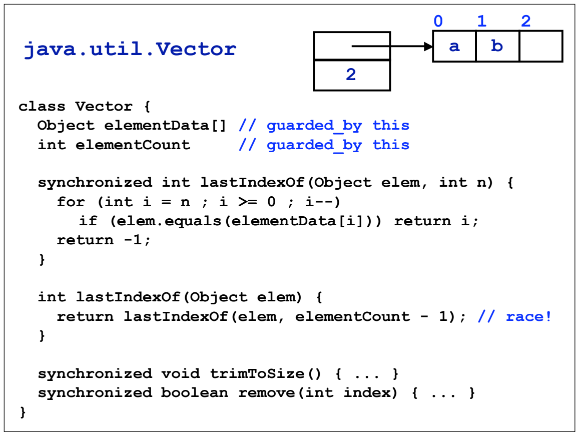


Statically Checking Real Systems

- RccJava [with Flanagan (UCSC), Peter Applegate '03]

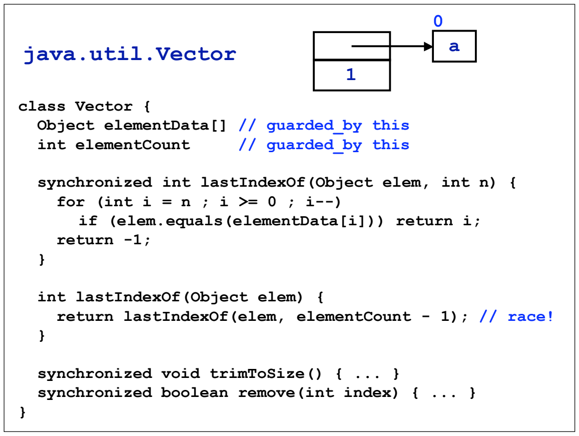
Annotated
Java
Program
RccJava
OK!
Warnings

- Found bugs in many pieces of software
  - commercial products, Java libraries, web servers
  - false positive rate of 75-80%
    - benign races, other forms of synchronization
    - better than not finding errors



java.util.Vector

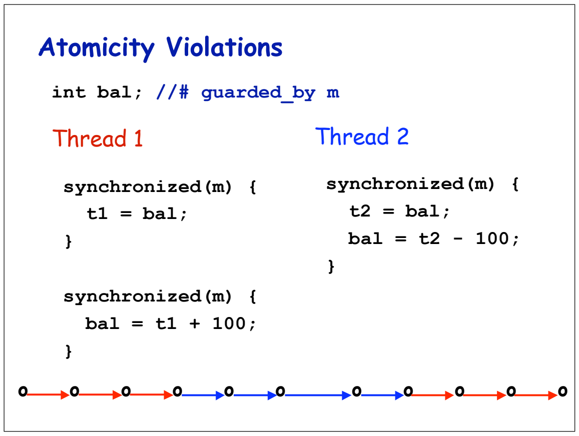
```
class Vector {
  Object elementData[] // guarded_by this
  int elementCount     // guarded_by this

  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); // race!
  }

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



java.util.Vector

```
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); // race!
  }

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

Demo



Atomicity Violations

```
int bal; //# guarded_by m
```

Thread 1

```
synchronized(m) {
  t1 = bal;
}

synchronized(m) {
  bal = t1 + 100;
}
```

Thread 2

```
synchronized(m) {
  t2 = bal;
  bal = t2 - 100;
}
```

Atomicity Violations

```
int bal; //# guarded_by m
```

Thread 1

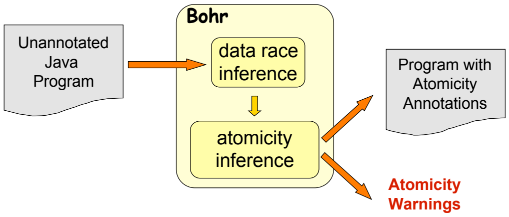
```
atomic {
  synchronized(m) {
    t1 = bal;
  }

  synchronized(m) {
    bal = t1 + 100;
  }
}
```

Thread 2

```
atomic {
  synchronized(m) {
    t2 = bal;
    bal = t2 - 100;
  }
}
```

Bohr

- Compute `atomic` annotations automatically
- Identify methods that may suffer interference
- With Masha Lifshin '05



The (Long) Road to Reliable Software

- Bugs are a real problem
- Checking tools will improve life for everyone
- Industry starting to adopt checkers

- Lots of problems (and fun) left
  - tools often hard to use, imprecise
  - simple tools pave way for more sophisticated
  - need teachable design methodologies

Thanks

- Pete Applegate '03
- Masha Lifshin '05
- Cormac Flanagan (UCSC)
- Martín Abadi (UCSC and Microsoft)
- Shaz Qadeer (Microsoft)

- NSF/NASA HDCCSR Program