Concurrent Object Oriented Languages

Assignment 3

https://wiki.cse.yorku.ca/course/6490A

Assignment 3

Section on Performance Measurements

- Discuss why throughput (number of push and pop operations per second) is appropriate for measuring the performance of the concurrent implementations of a stack.
- Discuss the relevant parameters:
  - the number of threads that perform push and pop operations, and
  - the percentage of pop operations.

Assignment 3

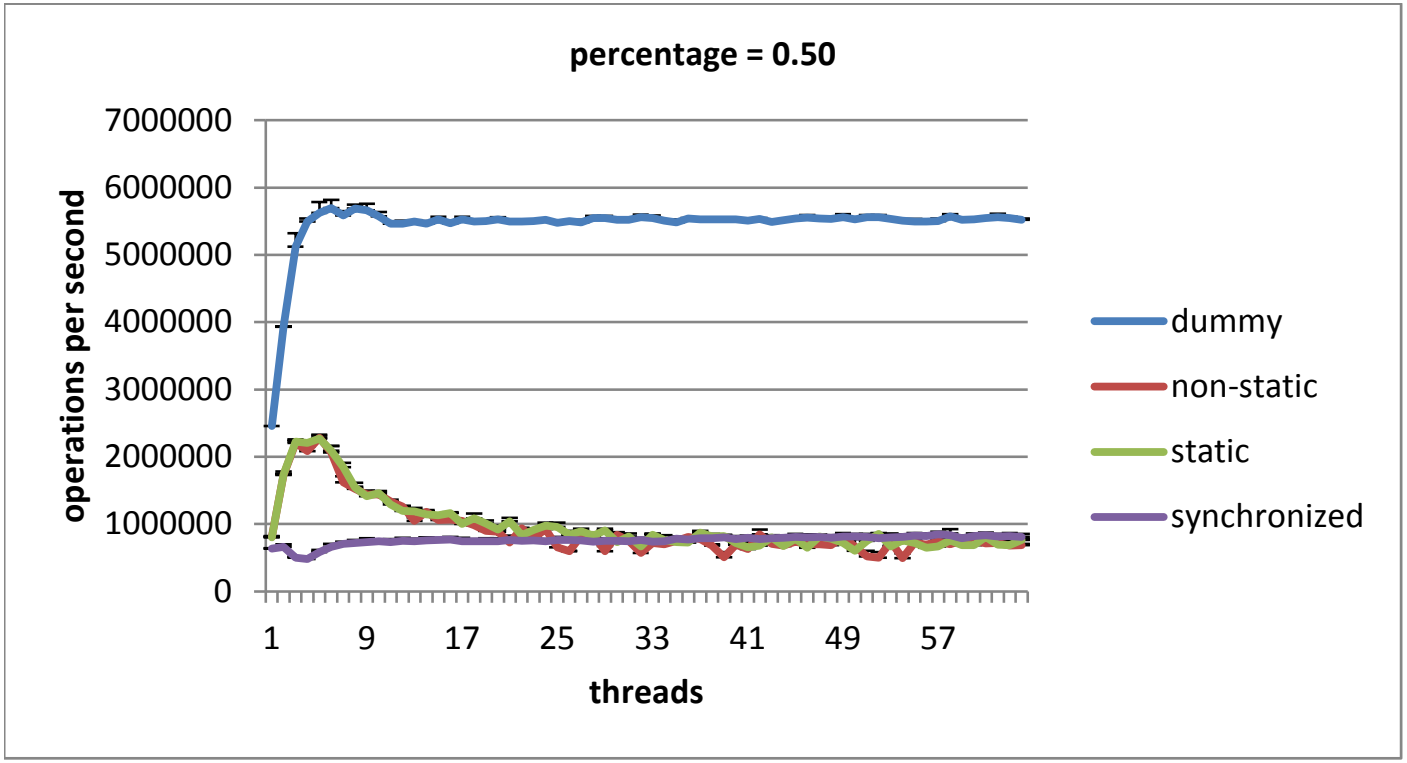
Section on Experimental Set-up

- Give a high level description of the `Main` and `Task` classes.
- Highlight some details.
- Describe how `Main` was run.
- Describe the machine on which the experiments were run, the operating system, the version of the Java virtual machine, etc. (Research results should be reproducible.)

Section on Results

- Present the results graphically.
- Interpret the results.

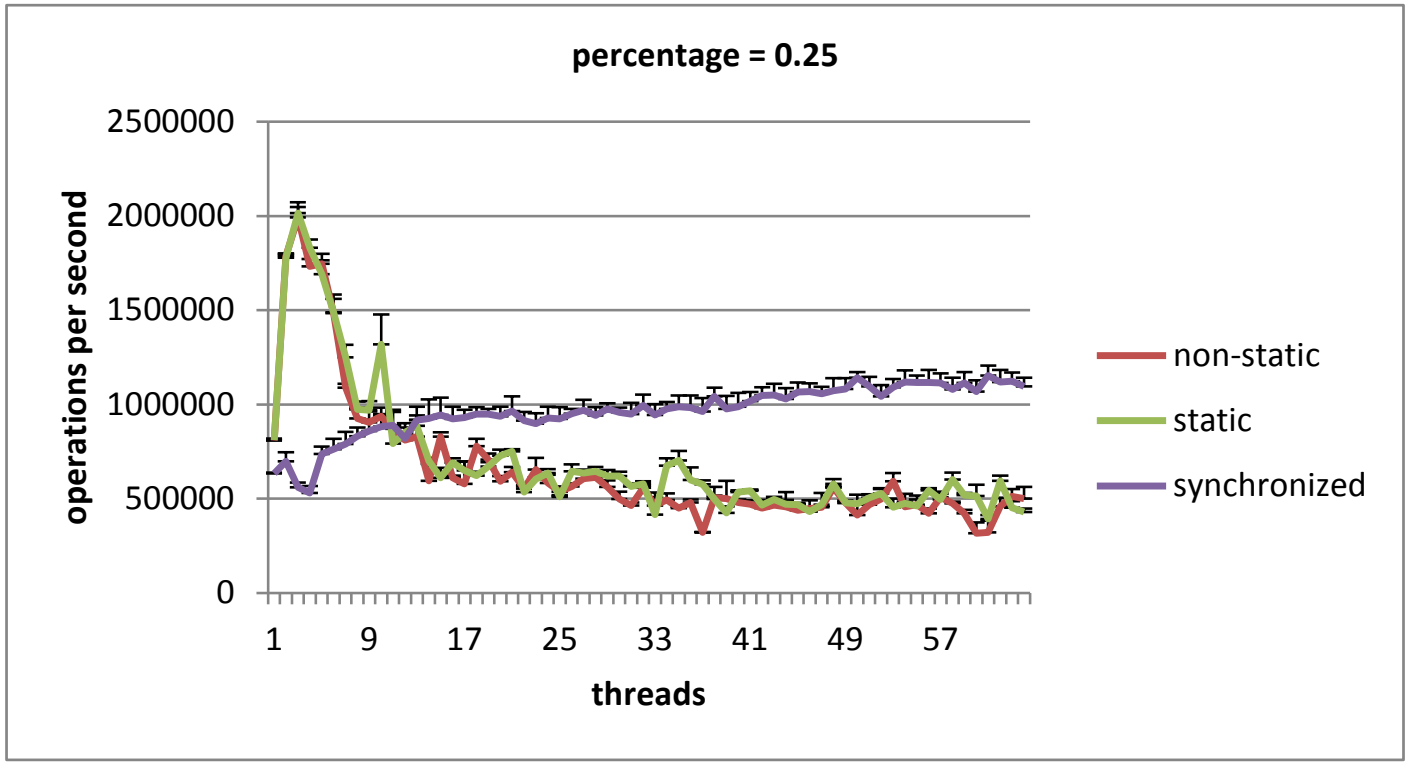
Overhead


percentage = 0.50
operations per second
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
1
9
17
25
33
41
49
57
threads
dummy
non-static
static
synchronized

Overhead

The overhead of the framework to measure the performance is negligible.

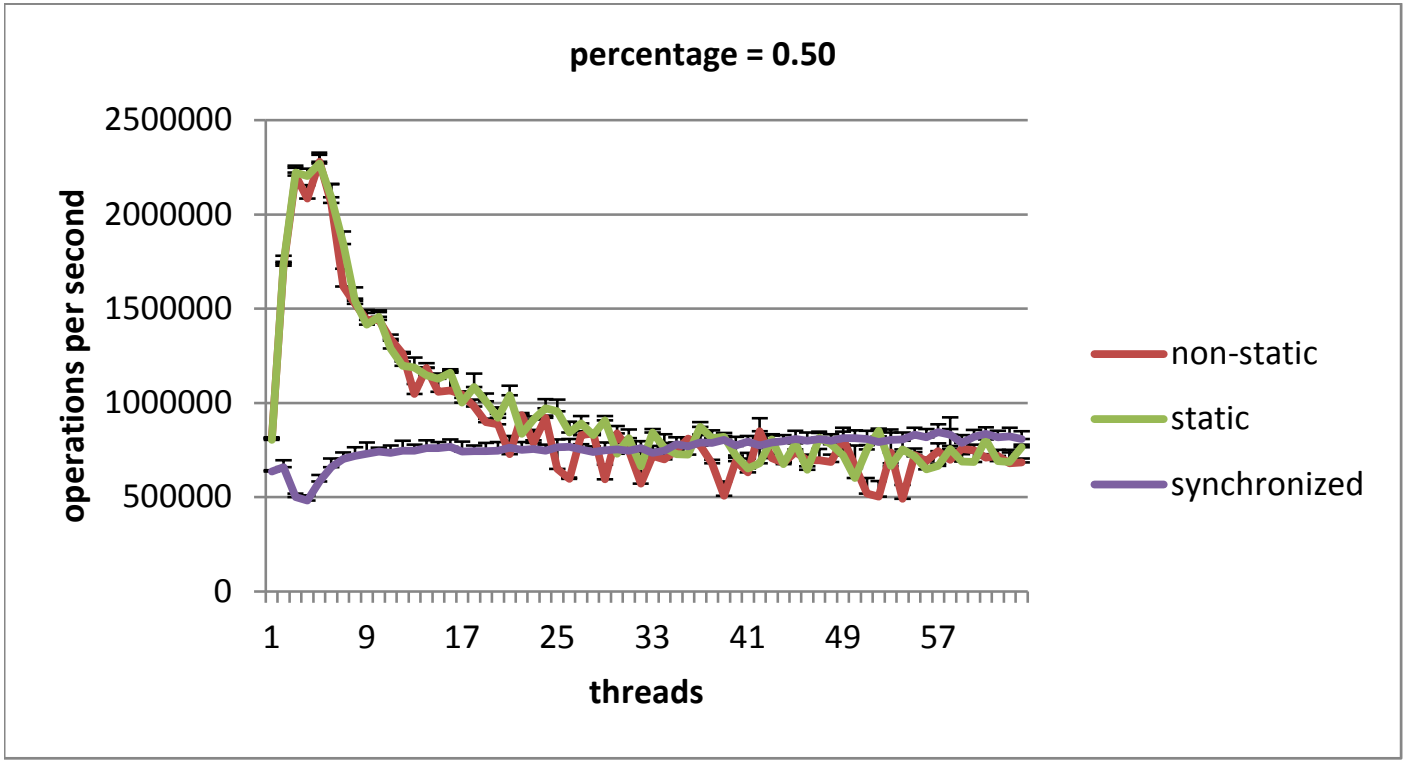
Percentage = 0.25



Percentage = 0.25

Static and non-static outperform synchronized for 1–8 threads.

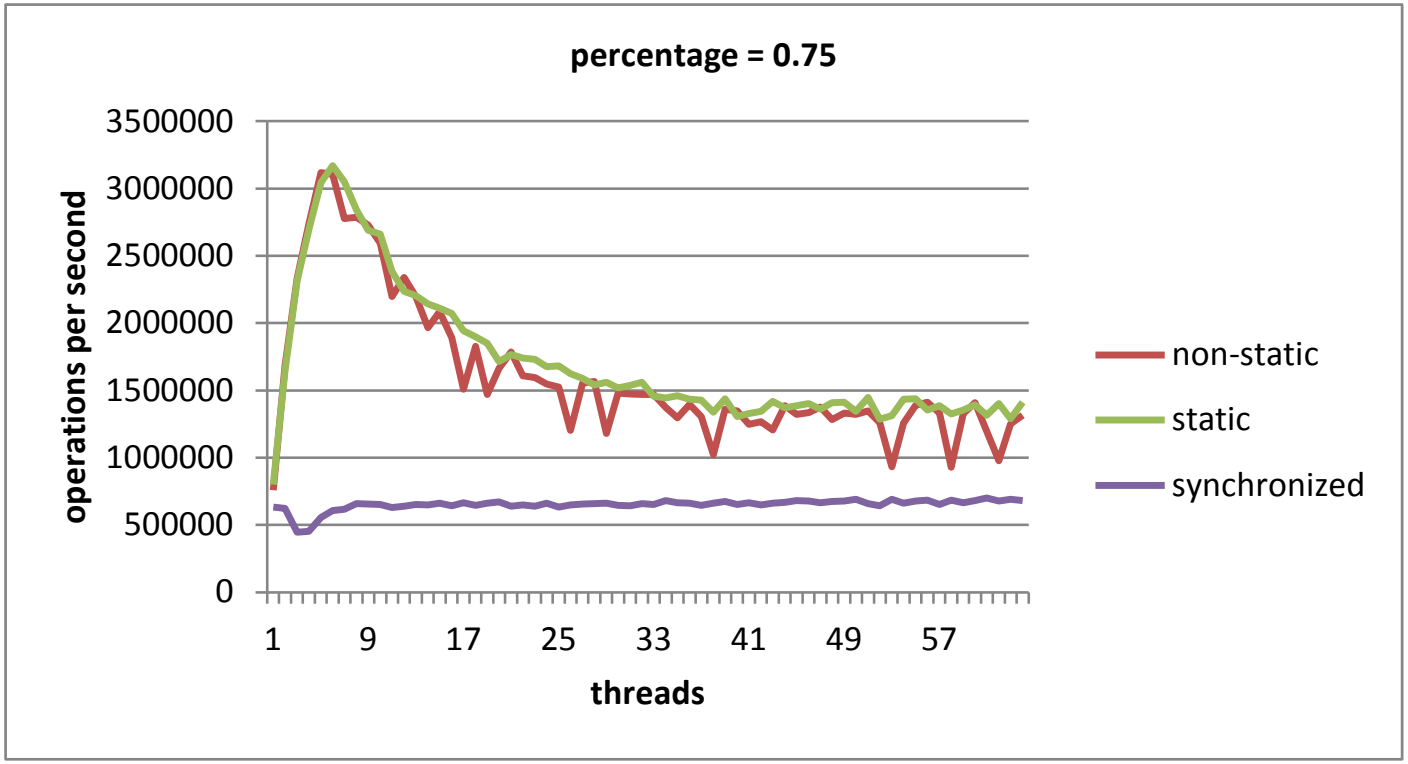
Percentage = 0.50



Percentage = 0.50

Static and non-static outperform synchronized for 1–20 threads.

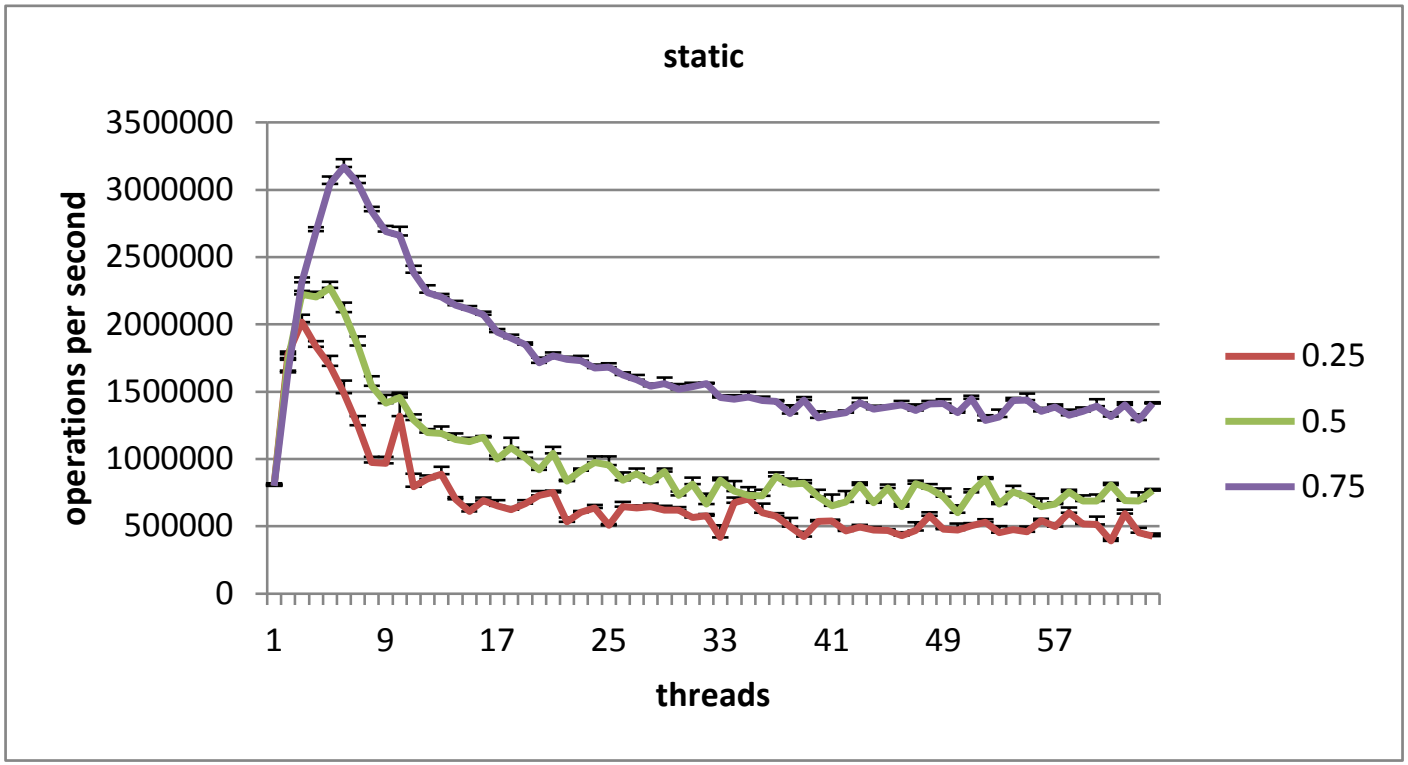
Percentage = 0.75



Percentage = 0.75

Static and non-static outperform synchronized.

Static


static
operations per second
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
1
9
17
25
33
41
49
57
threads
0.25
0.5
0.75

The more pop operations, the higher the throughput.

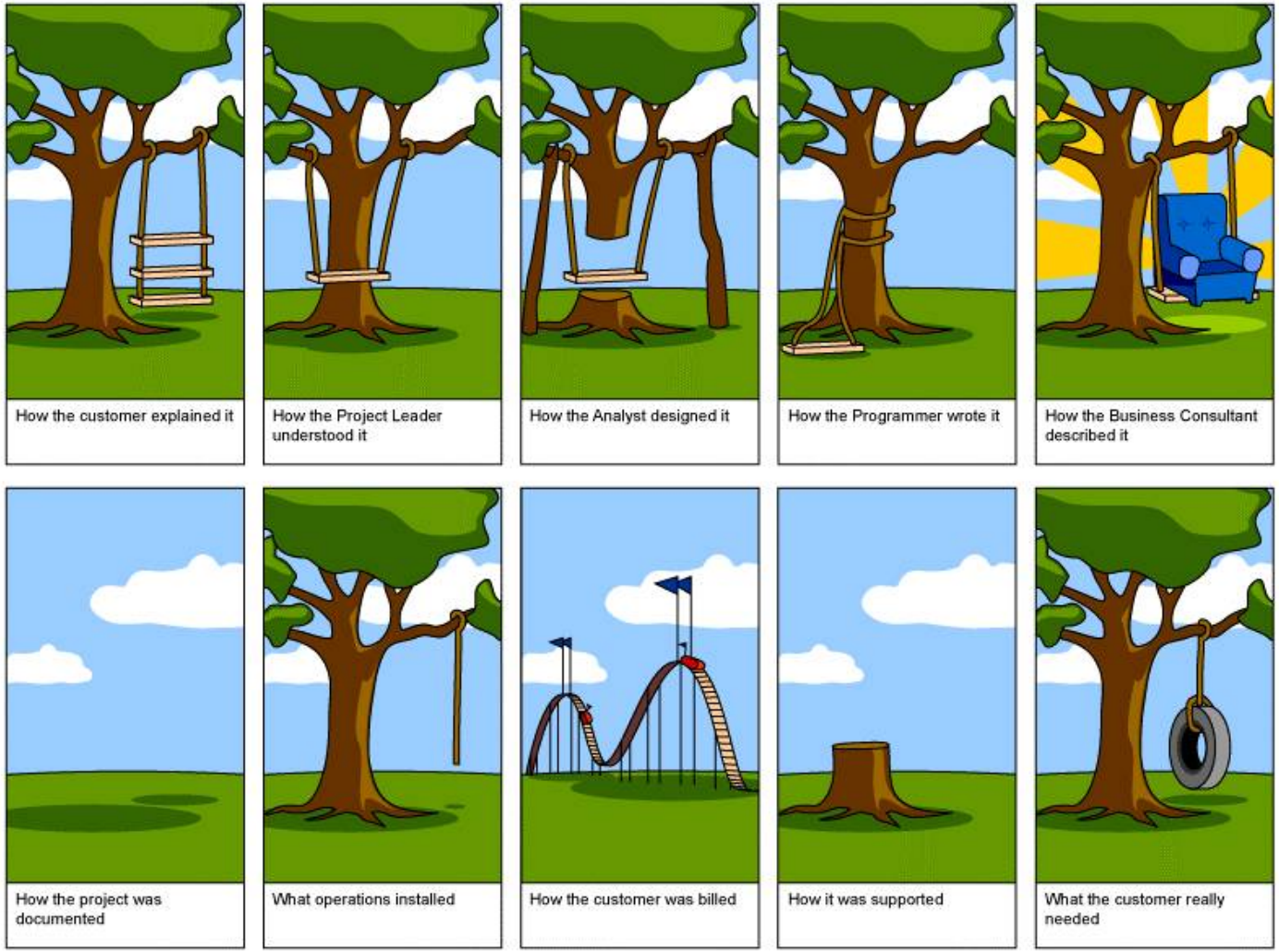
Concurrent Object Oriented Languages

Formal Verification

https://wiki.cse.yorku.ca/course/6490A

What is verification?

“Have you made what you were trying to make?”



Source: Paragon Innovations

What is verification?

"Have you made what you were trying to make?"

Does the hardware/software system satisfy all the properties obtained from its specification?

In contrast to . . .

"Have you made the right thing?"

Is the specification of the system correct?

which is also known as *validation*.

Why do we verify?

Bugs are everywhere.

Source: Bruce Campbell

Classic bug

1968 Brazilian Beetle

Source: Dan Palatnik

Classic bug

"A clear example of the risks of poor programming and verification techniques is the tragic story of the Therac-25—one in a series of radiation therapy machines developed and sold over a number of years by Atomic Energy Canada Limited (AECL). As a direct result of inadequate programming techniques and verification techniques, at least six patients received massive radiation overdoses which caused great pain and suffering and from which three died."

Source: unknown

Peter H. Roosen-Runge. Software Verification Tools. 2000.

Classic bug

"A computer malfunction at Bank of New York brought the Treasury bond market's deliveries and payments systems to a near standstill for almost 28 hours . . . it seems that the primary error occurred in a messaging system which buffered messages going in and out of the bank. The actual error was an overflow in a counter which was only 16 bits wide, instead of the usual 32. This caused a message database to become corrupted. The programmers and operators, working under tremendous pressure to solve the problem quickly, accidentally copied the corrupt copy of the database over the backup, instead of the other way around."

Wall Street Journal, November 25, 1985

Source: unknown

Classic bug

"To correct an anomaly that caused inaccurate results on some high-precision calculations, Intel Corp. last week confirmed that it had updated the floating-point unit (FPU) in the Pentium microprocessor. The company said that the glitch was discovered midyear and was fixed with a mask change in recent silicon. "This was a very rare condition that happened once every 9 to 10 billion operand pairs," said Steve Smith, a Pentium engineering manager at Intel."

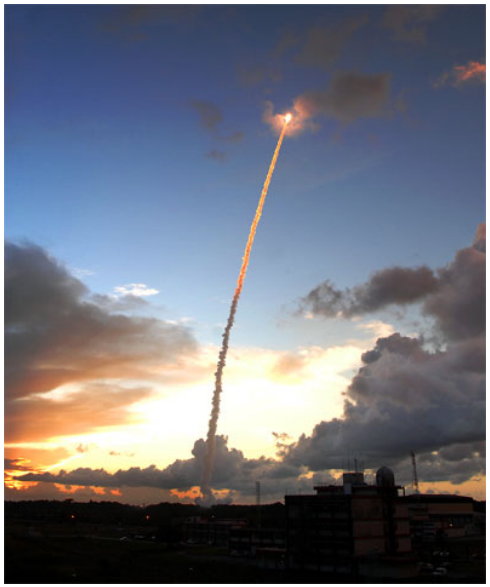
EE Times, November 7, 1994

Source: Konstantin Lanzet

Classic bug

"On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 meters, the launcher veered off its flight path, broke up and exploded. . . . The reason why the active SRI 2 did not send correct attitude data was that the unit had declared a failure due to a software exception. . . . The data conversion instructions (in Ada code) were not protected from causing an operand error, although other conversions of comparable variables in the same place in the code were protected."

Report of the Ariane Inquiry Board



Source: unknown

Bug of the 21st century

2012 Beetle

Source: unknown

Bug of the 21st century

The Toronto Skyline

Source: unknown

Bug of the 21st century

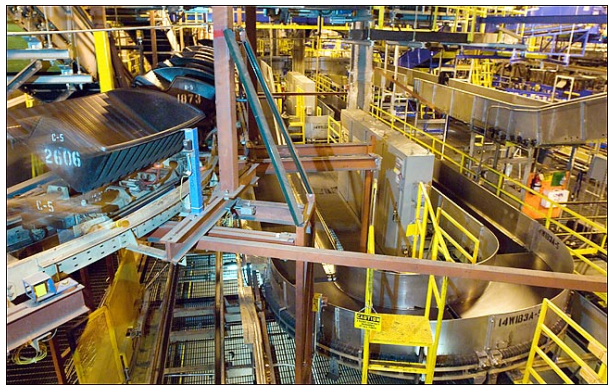
The Toronto Skyline on August 14, 2003

Source: unknown

Bug of the 21st century

"Ten years ago, the new Denver International Airport marched boldly into the future with a computerized baggage-handling system that immediately became famous for its ability to mangle or misplace a good portion of everything that wandered into its path. Now the book is closing on the brilliant machine that couldn't sort straight. Sometime over the next few weeks, in an anticlimactic moment marked and mourned by just about nobody, the only airline that ever used any part of the system will pull the plug."

New York Times, August 27, 2005



Source: Kevin Moloney

Bug of the 21st century

"When it comes to lethal bugs, the computer glitch that set fire to $440 million of Knight Capital Group's funds last Wednesday ranks right up there with the tsetse fly.

In less than an hour, Knight Capital's computers executed a series of automatic orders that were supposed to be spread out over a period of days. Millions of shares changed hands. The resulting loss, which was nearly four times the company's 2011 profit, crippled the firm and brought it to the edge of bankruptcy."

Brian Patrick Eha, CNN, August 9, 2012

Source: CNN

Why do we verify?

Don’t ask Jessie J!

Source: unknown

It’s all about the money money money.

What's the price tag?

Bank of New York bug: $ 5 million
Pentium bug: $ 475 million
Ariane bug: $ 500 million
Blackout bug: $ 6 billion
Denver bug: $ 300 million
Knight bug: $ 440 million

"The cost of software bugs to the U.S. economy is estimated at $ 60 billion per year."

National Institute of Standards and Technology, 2002

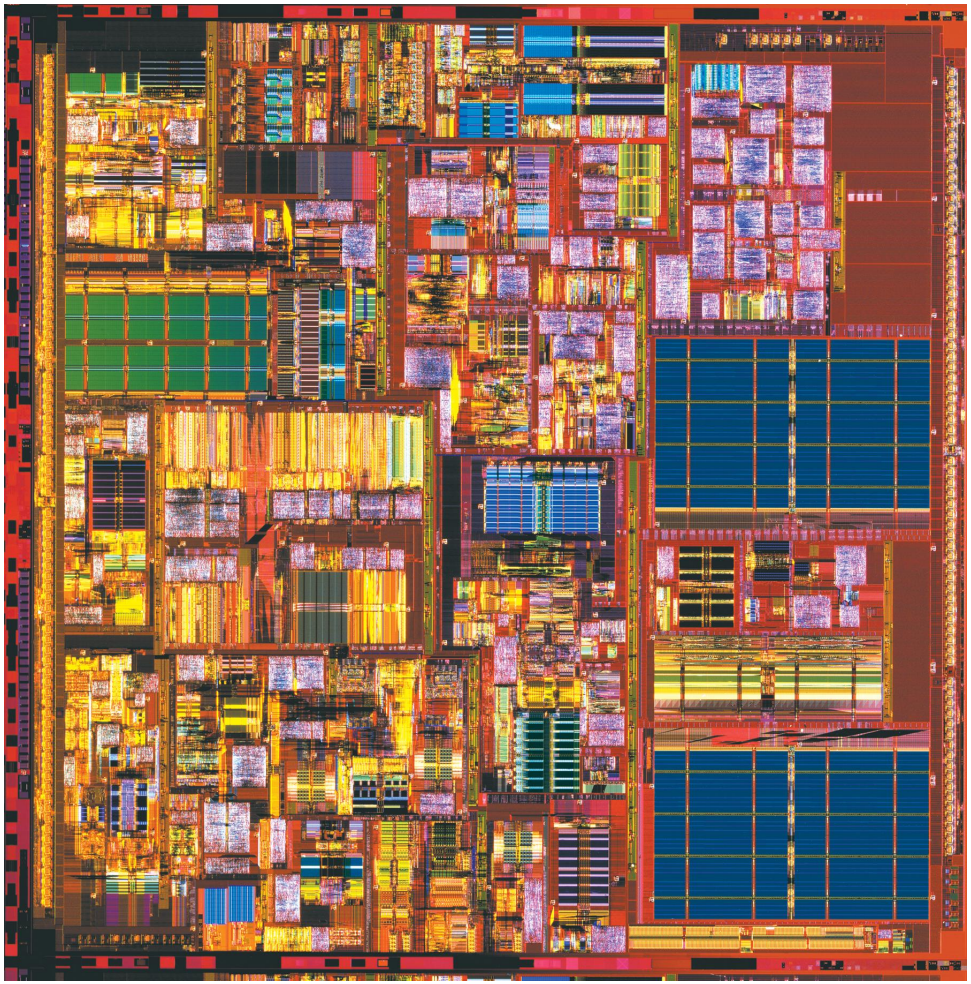
"Wages-only estimated cost of debugging: US $312 Billion per year."

Tom Britton, Lisa Jeng, Graham Carver, Paul Cheak and Tomer Katzenellenbogen, 2013

Why are bugs introduced?

Hardware and software systems are among the most complex artifacts ever produced by humans.

Pentium 4 microprocessor



Source: unknown

- transistors:
  55 million
- area:
  146 mm^2
- clock rate:
  3.2×10^9 Hz

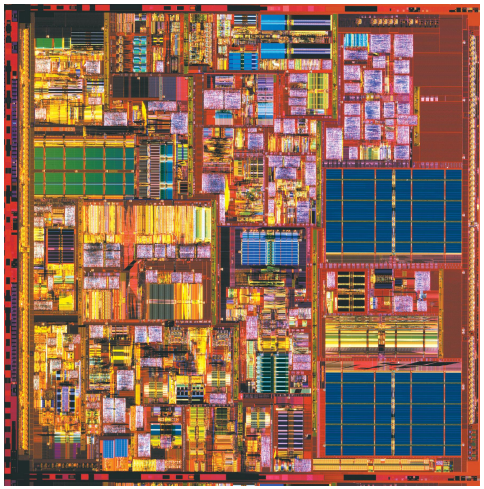
Based on slide of Prof. Tom Melham

If . . .

. . . the connections on a microprocessor were roads in Scotland, . . .

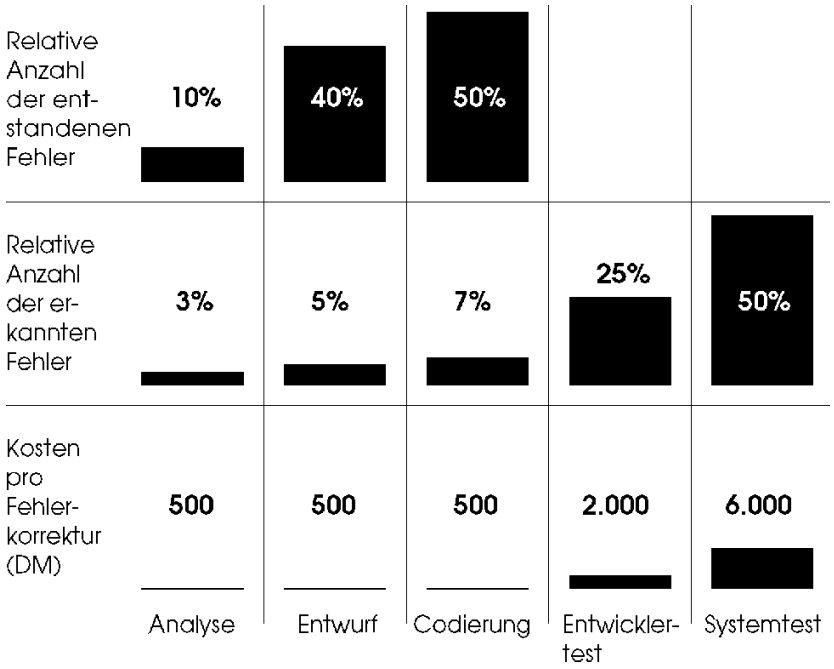
Area of microprocessor: 146 mm^2
Area of Scotland: 80,234 km^2
Scale: 12 mm / 283 km $\approx$ 1 / 14,150,000

. . . then, since each connection is 0.13μm wide, the roads in Scotland would be 1.84 m wide, 1.84 m apart and eight layers deep!



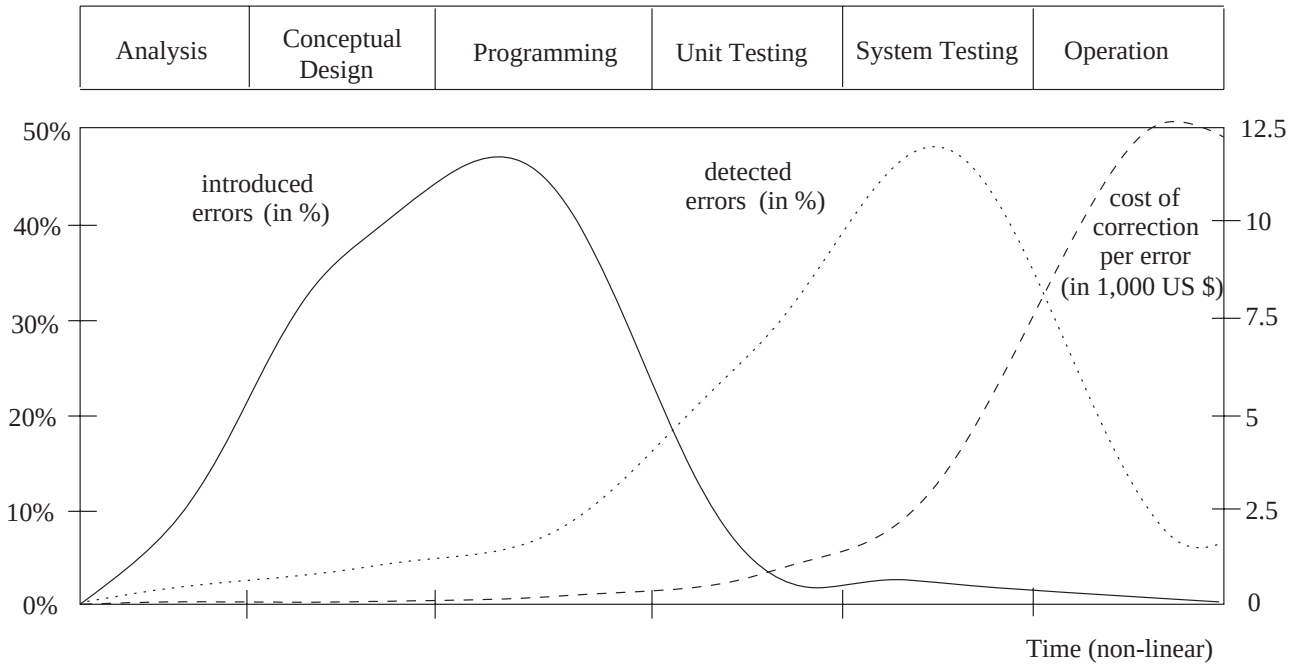
Based on slide of Prof. Tom Melham

When are bugs introduced and detected?



Peter Liggesmeyer, Martin Rothfelder, Michael Rettelbach, and Thomas Ackermann. Qualitätssicherung Software-basierter technischer Systeme – Problembereiche und Lösungsansätze. *Informatik-Spektrum*, 21(5):249–258, October 1998.

When are bugs introduced and detected?



Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press. Cambridge, MA, USA. 2008.

How are bugs detected?

- Manual inspection (very costly and error prone)
- Simulation
- Testing
- Verification

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

How many can we check in one second?

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

How many can we check in one second?
3×10^{9}

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

How many can we check in one second?
3×10^{9}

How many seconds does it take?

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

How many can we check in one second?
3×10^{9}

How many seconds does it take?
$1.2 \times 10^{77} / 3 \times 10^{9} = 4 \times 10^{67}$

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

How many can we check in one second?
3×10^{9}

How many seconds does it take?
$1.2 \times 10^{77} / 3 \times 10^{9} = 4 \times 10^{67}$

How many years is that?

Limitations of simulation

How long does it take to simulate a 128-bit multiplier on a 3 GHz machine?

How many cases need to be checked?
$2^{2\times 128} \approx 1.2 \times 10^{77}$

How many can we check in one second?
3×10^{9}

How many seconds does it take?
$1.2 \times 10^{77} / 3 \times 10^{9} = 4 \times 10^{67}$

How many years is that?
2×10^{59}

Based on slide of Prof. Tom Melham

Limitations of testing

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

Edsger Wybe Dijkstra, October 1969

How does verification detect bugs?

- Property checking: Does system S satisfy property P?
- Equivalence checking: Are systems S_1 and S_2 equivalent?

What is formal about formal verification?

- The system is *formalized* (as some mathematical entity).
- The property is *formalized* in some logic.
- Whether the system satisfies the property is *formalized* as well.

Model of a system

A *model* of a system is an *abstraction* of the system.

There are many levels of abstraction and, hence, a system can be modelled in many different ways.

Transition system

Definition

A *transition system* is a tuple $\langle S, \rightarrow, L, \ell \rangle$ consisting of

- a set S of *states*,
- a *transition relation* $\rightarrow \subseteq S \times S$,
- a set L of *labels*, and
- a *labelling function* $\ell : S \rightarrow 2^L$.

Transition relation

Instead of $(s, s') \in \rightarrow$ we write $s \rightarrow s'$.

We restrict our attention to transition relations such that each state has an outgoing transition, that is

$$\forall s \in S : \exists s' \in S : s \rightarrow s'.$$

Model of a traffic light

A traffic light can be red, green, orange or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

How many different states do we need to model the traffic light?

Model of a traffic light

A traffic light can be red, green, orange or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$S = \{1, 2, 3, 4\}$$

1 red
2 green
3 orange
4 black

What are the transitions?

Model of a traffic light

A traffic light can be red, green, orange or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$
\begin{aligned}
S &= \{1, 2, 3, 4\} \\
\rightarrow &= \{(1,2), (2,3), (3,1), (1,4), (2,4), (3,4), (4,1)\}
\end{aligned}
$$

How is the labelling function defined?

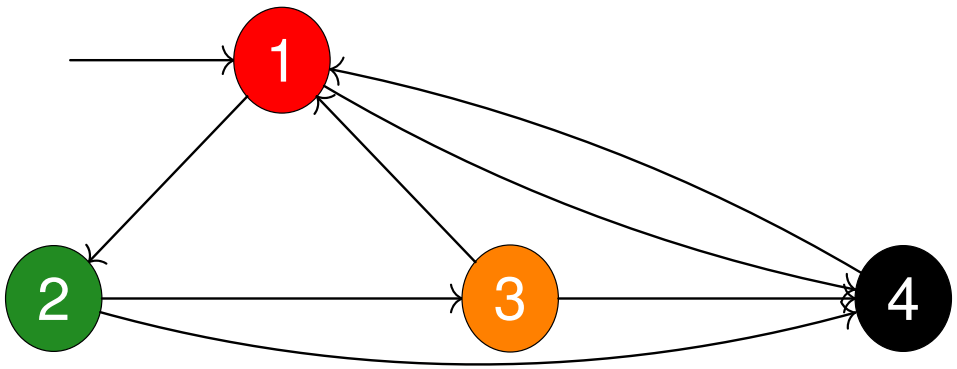
Model of a traffic light

A traffic light can be red, green, orange or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$
\begin{aligned}
S &= \{1,2,3,4\} \\
\rightarrow &= \{(1,2),(2,3),(3,1),(1,4),(2,4),(3,4),(4,1)\} \\
L &= \{r,o,g\} \\
\ell &= \{1 \mapsto \{r\}, 2 \mapsto \{g\}, 3 \mapsto \{o\}, 4 \mapsto \emptyset\}
\end{aligned}
$$

Model of a traffic light

A traffic light can be red, green, orange or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.



Java bytecode

The verification tool Java PathFinder (JPF) is a Java virtual machine and manipulates Java bytecode. The states of JPF capture the stack frames, the heap, etc and the transitions correspond to sequences of bytecode instructions.