Program Verification as a Toolbox

A Brief, *Subjective* History

David Cock

January 23, 2015

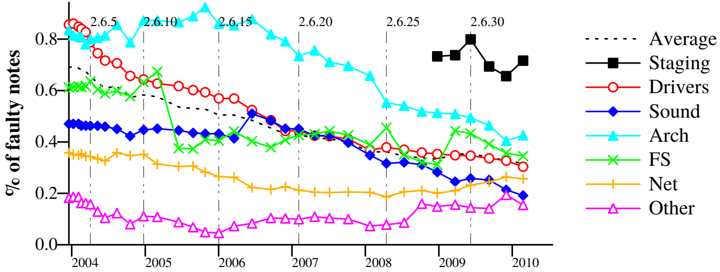
Is Your System Correct?

Short answer — no.

Is Your System Correct?

Short answer — no.

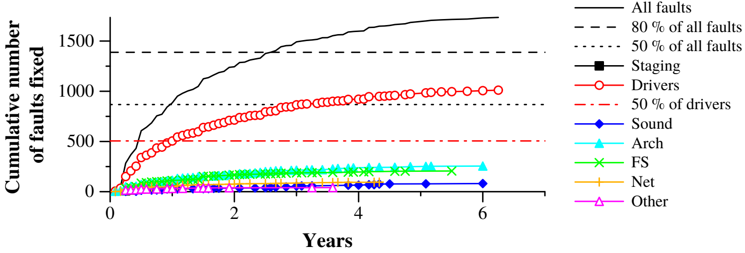
The Bug Rate in Linux [1]



It's dropping, but there's a long way to go.

[1] Source: Palix et. al., Faults in Linux: Ten Years Later, ASPLOS'11

Bug Lifetime in Linux [2]



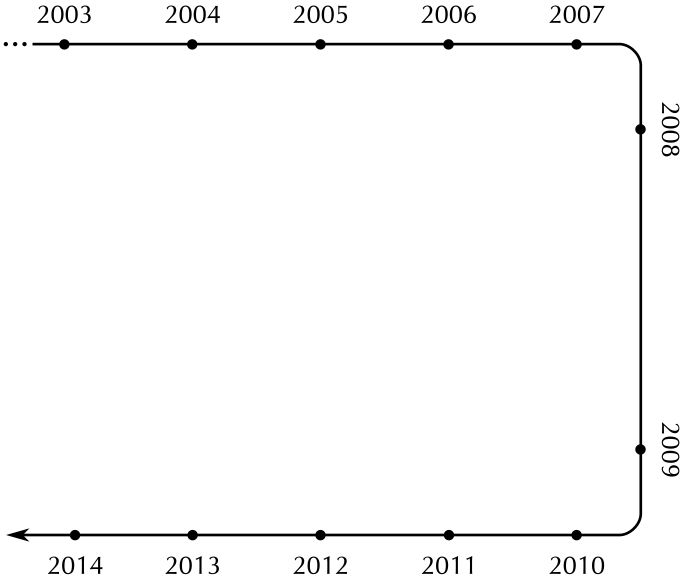
- Only 60% fixed within a year.
- Asymptotic — some bugs live 5 + years!

[2] Source: Palix et. al., Faults in Linux: Ten Years Later, ASPLOS'11

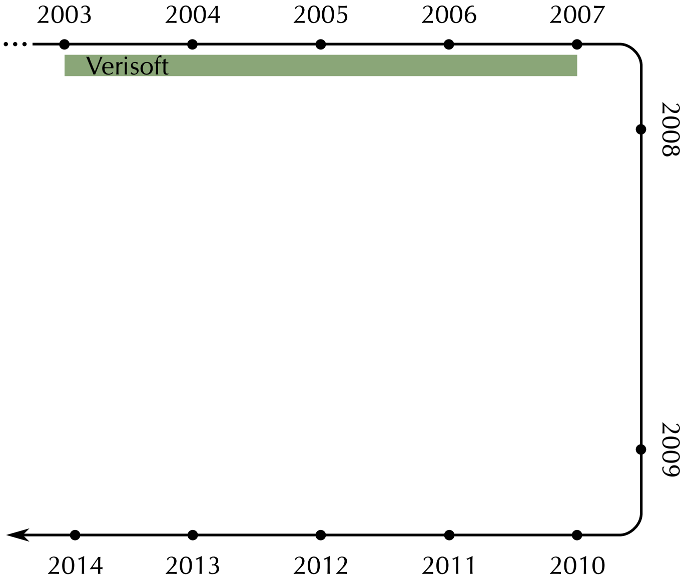
Why Now?

- Less expertise is required than 10 years ago.
- We’ve seen some real milestones:
  - seL4
  - CompCert
- *Tool support* has matured dramatically.

A Timeline

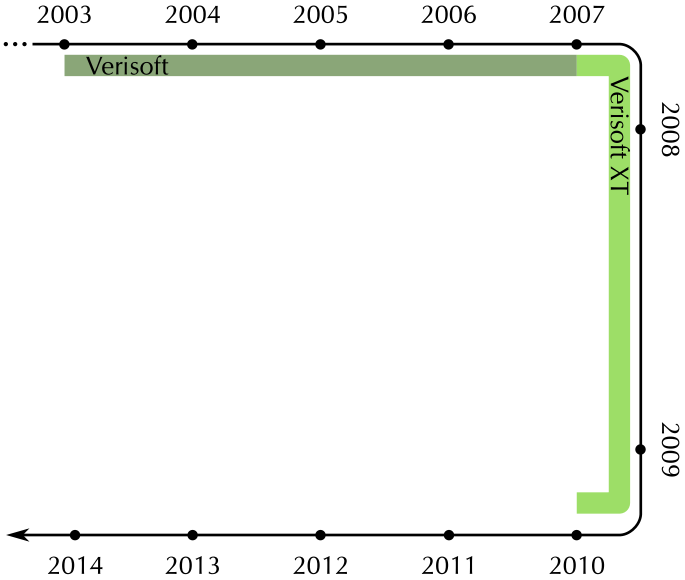


A Timeline

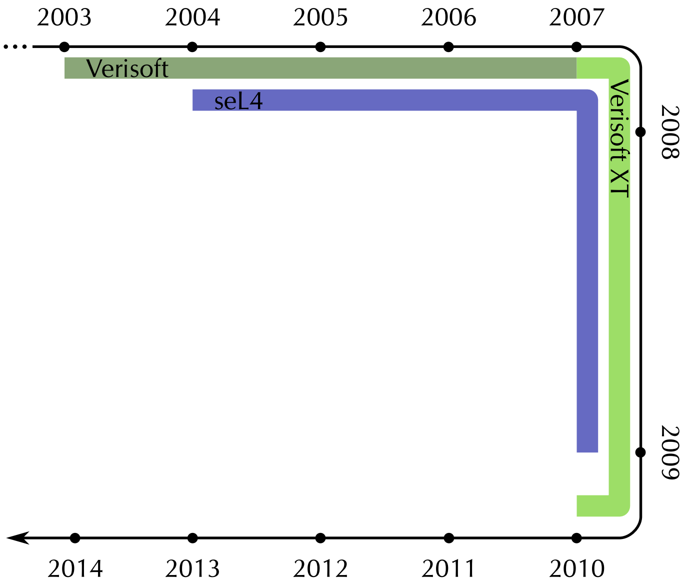


2003
2004
2005
2006
2007
Verisoft
2008
2009
2010
2011
2012
2013
2014

A Timeline

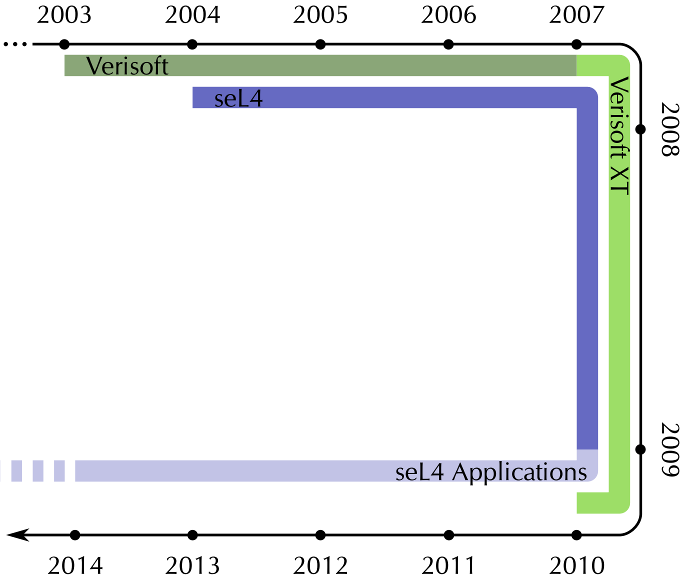

2003
2004
2005
2006
2007
Verisoft
Verisoft XT
2008
2009
2010
2011
2012
2013
2014

A Timeline

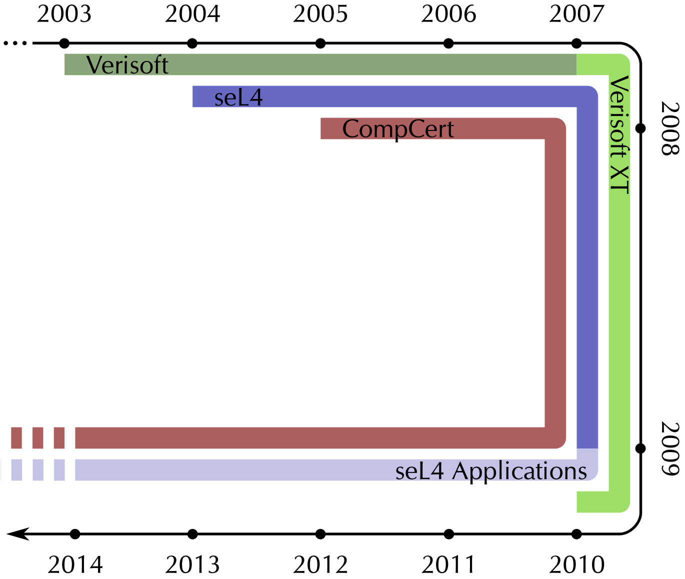


2003
2004
2005
2006
2007
Verisoft
seL4
Verisoft XT
2008
2009
2010
2011
2012
2013
2014

A Timeline

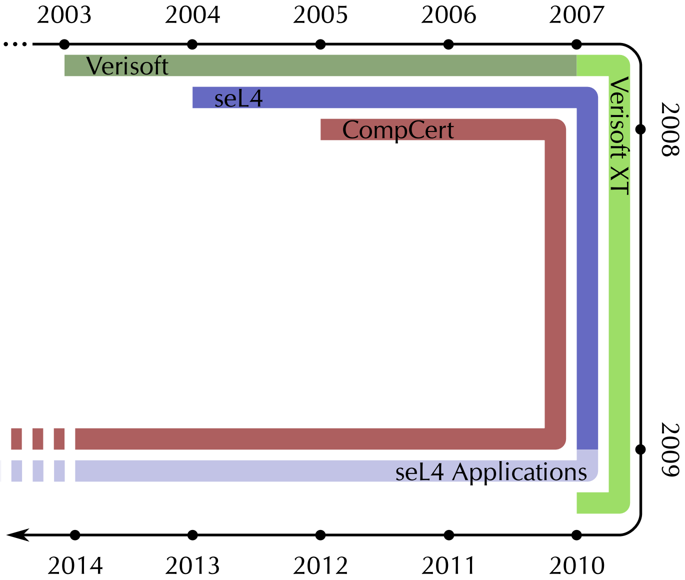

2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
...
Verisoft
seL4
Verisoft XT
seL4 Applications

A Timeline

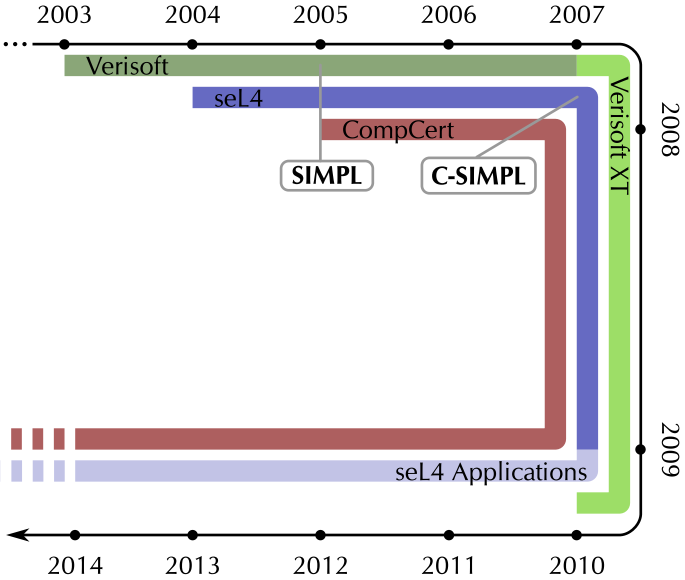


2003
2004
2005
2006
2007
Verisoft
seL4
CompCert
Verisoft XT
2008
2009
seL4 Applications
2014
2013
2012
2011
2010

A Timeline


2003
2004
2005
2006
2007
Verisoft
seL4
CompCert
Verisoft XT
2008
2009
seL4 Applications
2014
2013
2012
2011
2010

A Timeline



2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
Verisoft
seL4
CompCert
SIMPL
C-SIMPL
Verisoft XT
seL4 Applications

SIMPL/C

C is an awful language to reason about…

SIMPL/C

C is an awful language to reason about…
but it's fast and universal.

SIMPL/C

C is an awful language to reason about. . .
but it's fast and universal.

```
*(a++) = ++*a-- + (*(a++))++ * *--a;
```

SIMPL/C

C is an awful language to reason about…
but it’s fast and universal.

```
*(a++) = ++*a-- + (*(a++))++ * *--a;
```

- We’ve now got a formal semantics for C[3].

[3] Winwood et. al., Mind the gap: A verification framework for low-level C, TPHOLS’09

SIMPL/C

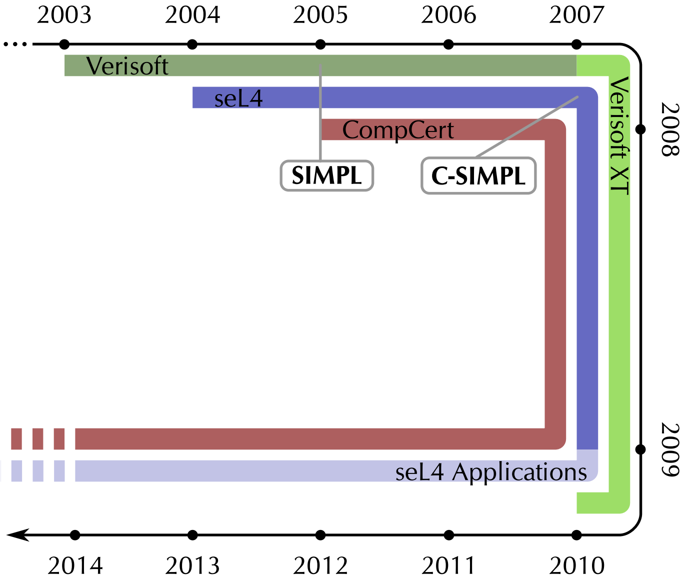
C is an awful language to reason about…
  but it's fast and universal.

```
*(a++) = ++*a-- + (*(a++))++ * *--a;
```

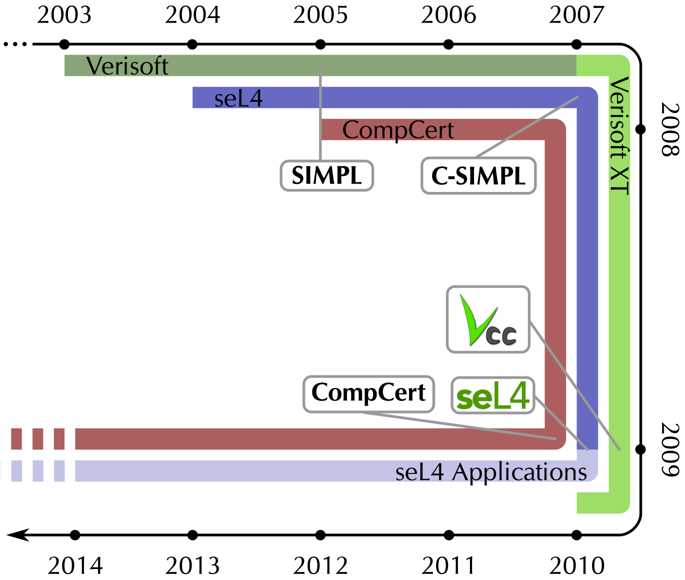
- We've now got a formal semantics for C[3].
- As long as you don't write nonsense like this.

[3] Winwood et. al., Mind the gap: A verification framework for low-level C, TPHOLS'09

A Timeline


2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
Verisoft
seL4
CompCert
SIMPL
C-SIMPL
Verisoft XT
seL4 Applications

A Timeline


2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
Verisoft
seL4
CompCert
Verisoft XT
SIMPL
C-SIMPL
Vcc
CompCert
seL4
seL4 Applications

seL4, VCC & CompCert

As of 2009, we've got:

seL4, VCC & CompCert

As of 2009, we've got:

- A verified kernel: seL4.

seL4, VCC & CompCert

As of 2009, we've got:

- A verified kernel: seL4.
- A *verifying* compiler: CompCert.

seL4, VCC & CompCert

As of 2009, we've got:

- A verified kernel: seL4.
- A *verifying* compiler: CompCert.
- An automatic verifier for concurrent C: VCC.

seL4, VCC & CompCert

Vcc seL4

As of 2009, we've got:

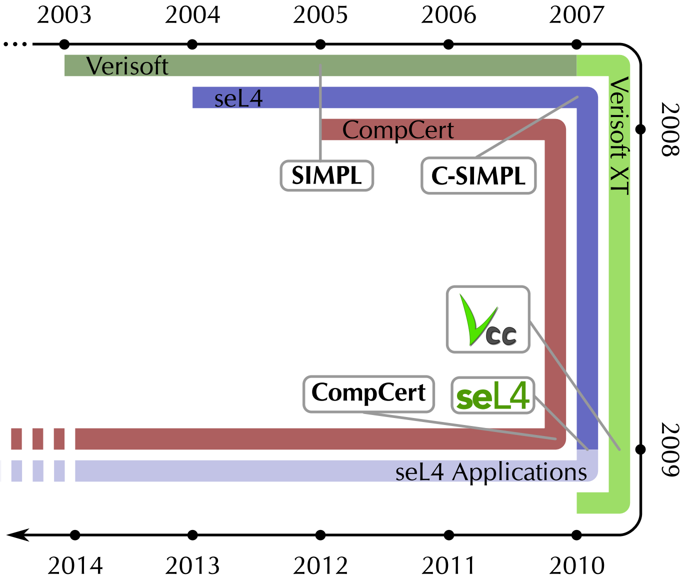
- A verified kernel: seL4.
- A *verifying* compiler: CompCert.
- An automatic verifier for concurrent C: VCC.
- seL4 compiles with CompCert…

seL4, VCC & CompCert

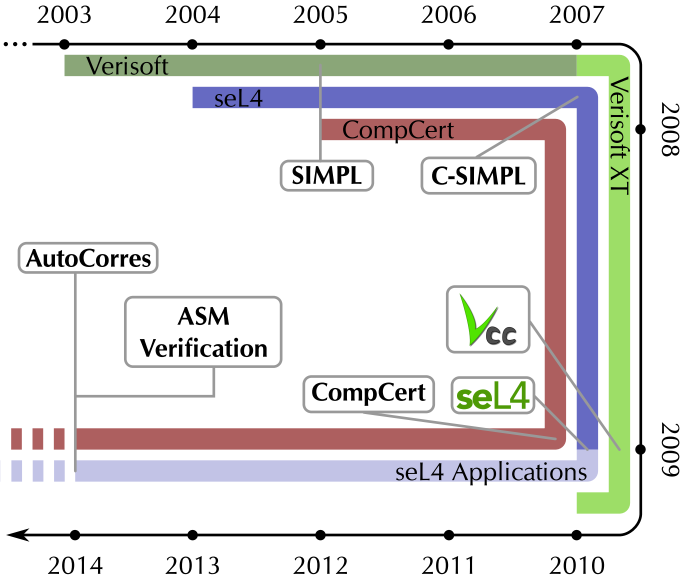
As of 2009, we've got:

- A verified kernel: seL4.
- A *verifying* compiler: CompCert.
- An automatic verifier for concurrent C: VCC.
- seL4 compiles with CompCert. . .
  but VCC can't (yet) verify seL4.

A Timeline

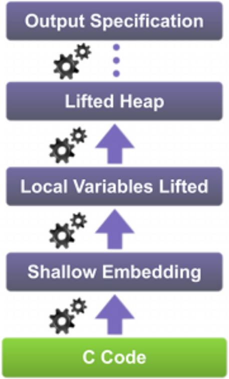

2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
Verisoft
seL4
CompCert
Verisoft XT
SIMPL
C-SIMPL
VCC
CompCert
seL4
seL4 Applications

A Timeline



2003
2004
2005
2006
2007
Verisoft
seL4
CompCert
SIMPL
C-SIMPL
Verisoft XT
2008
AutoCorres
ASM Verification
Vcc
CompCert
seL4
2009
seL4 Applications
2014
2013
2012
2011
2010

AutoCorres[4] & Assembly Verification[5]



- Brand new tools.
- Highly automated.
- Autocorres
  - Abstract from pointers and fixed-length words.
  - Lift to a verification-friendly model.
- ASM Verification
  - Alternative approach to CompCert.
  - Verify the *output* of `gcc -O1` (`-O2` coming).

[4] Greenaway et. al., Don't Sweat the Small Stuff: Formal Verification of C Code Without the Pain, PLDI'14

[5] Sewell et. al., Translation validation for a verified OS kernel, PLDI'13

Putting It Into Practice

- **Tools**
  - Still not seamless.
  - Interoperability and re-use.
  - Formal concerns (different logics).
- **Education**
  - Introduce programmers to the formal mindset.
- **Applications**
  - Trusted partitioning (Virtualisation, SDN, . . .).
  - Trusted computing.
  - Safety-critical systems.

Questions?