Faults in Linux: 10 years later

Nicolas Palix, Gaël Thomas, Suman Saha
Christophe Calvès, Julia Lawall, Gilles Muller

DIKU / INRIA Regal / LIP6

What is a fault?

A fragment of code that may cause a runtime error.

```
static unsigned int tun_chr_poll(struct file *file, poll_table *wait) {
  struct tun_file *tfile = file->private_data;
  struct tun_struct *tun = __tun_get(tfile);
  struct sock *sk = tun->sk;
  unsigned int mask = 0;

  if (!tun)
    return POLLERR;
  ...
}
```

Possible effects in the Linux kernel:

- Kernel crash
- Rootkit

Why study faults in OS code?

Find bugs

- Over 1900 patches based on fault-finding tools in 2005–2010.

Give users confidence

- “Linux creator Linus Torvalds released the much anticipated 2.6.11 Linux kernel declaring, ’so it’s now _officially_ all bug-free.’ ”

Identify research and development priorities

- “drivers have an error rate up to 7 times higher than the rest of the kernel” [Chou *et al.*, SOSP01]

10 years ago

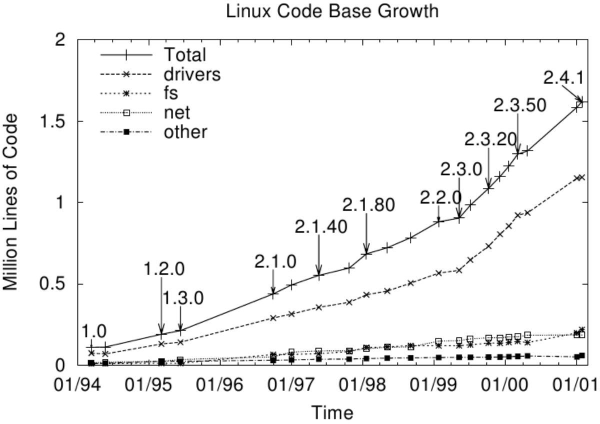
- In SOSP'01, Chou *et al.* studied faults (errors/bugs) in Linux code.
- Faults collected using static analysis.
- Faults collected in Linux 1.0 (1994) to 2.4.1 (2001).
  - Primarily "development" versions.
  - x86 code.

Considered fault types

Checker	
Block	*"To avoid deadlock, do not call blocking functions with interrupts disabled or a spinlock held."*
Null	*"Check potentially NULL pointers returned from routines"*
Var	*"Do not allocate large stack variables (>1K) on the fixed-size kernel stack."*
INull	*"Do not make inconsistent assumptions about whether a pointer is NULL."*
Range	*"Always check bounds of array indices and loop bounds derived from user data."*
Lock	*"Release acquired locks; do not double-acquire locks."*
Intr	*"Restore disabled interrupts."*
Free	*"Do not use freed memory."*
Float	*"Do not use floating point in the kernel."*
Size	*"Allocate enough memory to hold the type for which you are allocating."*

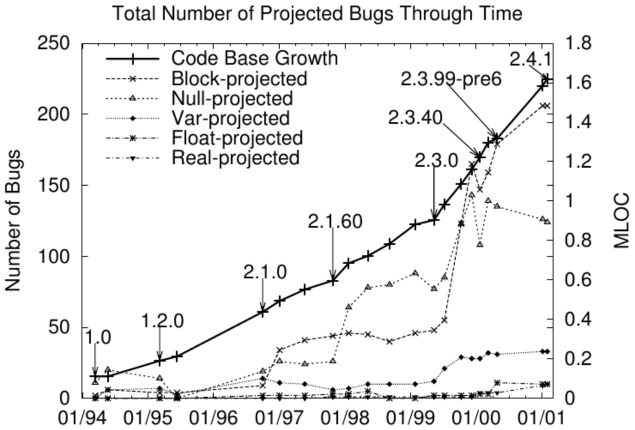
Code size was increasing ...

Up to 70% of code dedicated to drivers



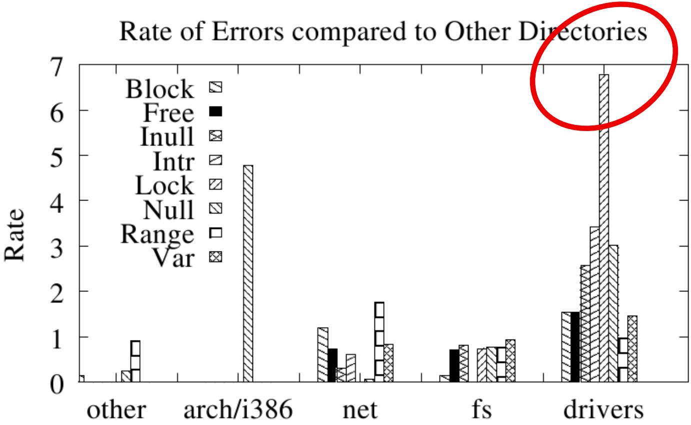
[Chou et al, SOSP01]

... faults were rising ! ...



[Chou et al, SOSP01]

... up to 7x higher fault rate in drivers than in any other directory.



[Chou et al, SOSP01]

What about today?

Lots more code

- up to 8 MLOC

New release model

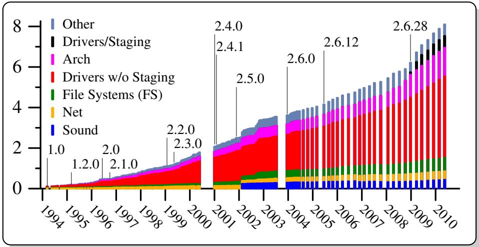
- 2.{4,5} vs 2.6.x

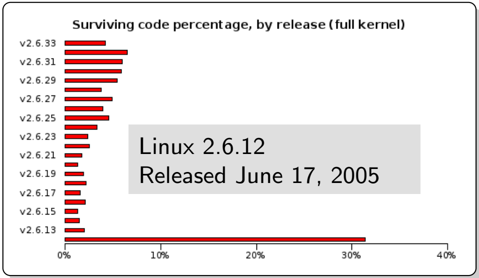
New SCM

- GIT since 2.6.12

Lots of new code

- 69% new code since 2.6.12




Surviving code percentage, by release (full kernel)

Linux 2.6.12
Released June 17, 2005

We Need New Data!

How to update a 10-year old study?

- Static analysis tool of Chou *et al.* not available.
- Checkers only informally described.
- Inter-version correlation strategy not described.
- Results no longer available.
- The Linux code has changed a lot.

Our approach

Open source analysis tools.

- Coccinelle to find faults.
- Herodotos to correlate fault reports across versions.

Results stored in an established open archive.

Multi architecture.

Refinements to some rule types.

- LockIntr, in addition to Lock and Intr

Iteration to collect functions with specific properties.

- Blocking functions, etc.

Allows Linux code quality to be continually reassessed.

A few numbers

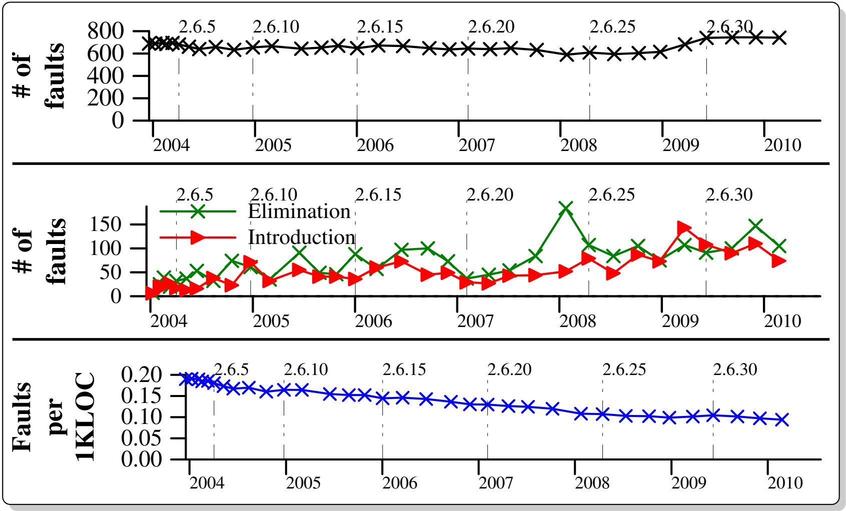
Study of Linux 2.4.1 and 34 versions of Linux 2.6 (2004-2010)

- More than 170 MLOC analyzed
- 697K files
- 6.15M functions

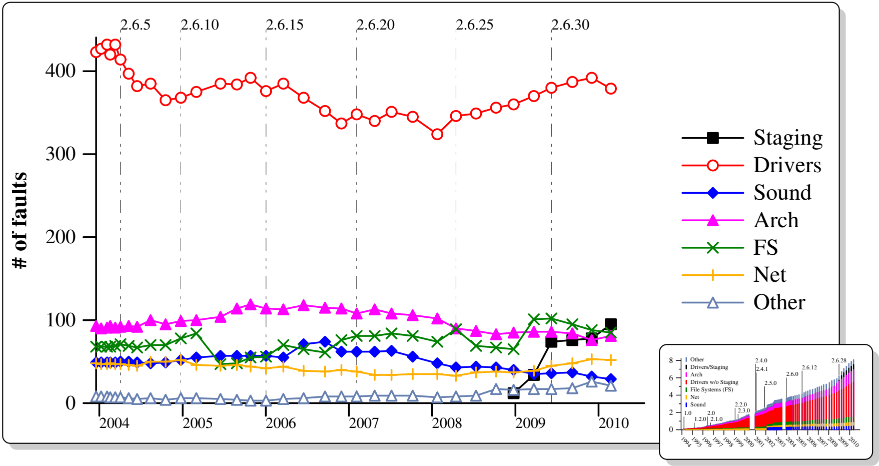
47 Coccinelle patterns for finding faults (30) and notes (17)

- 4.44M notes
- 40,177 fault reports
- 4,815 correlated reports (all verified)
- 3,052 correlated faults

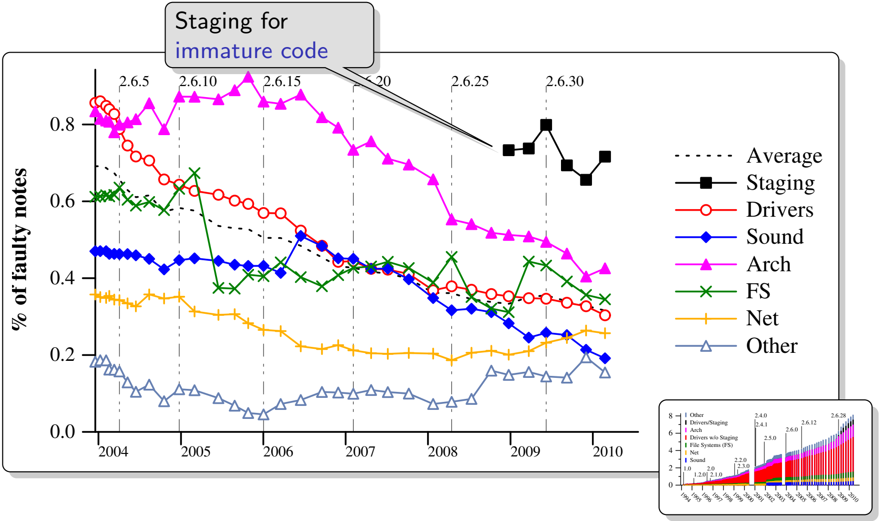
Faults are no longer rising...



Most of the faults are still in drivers

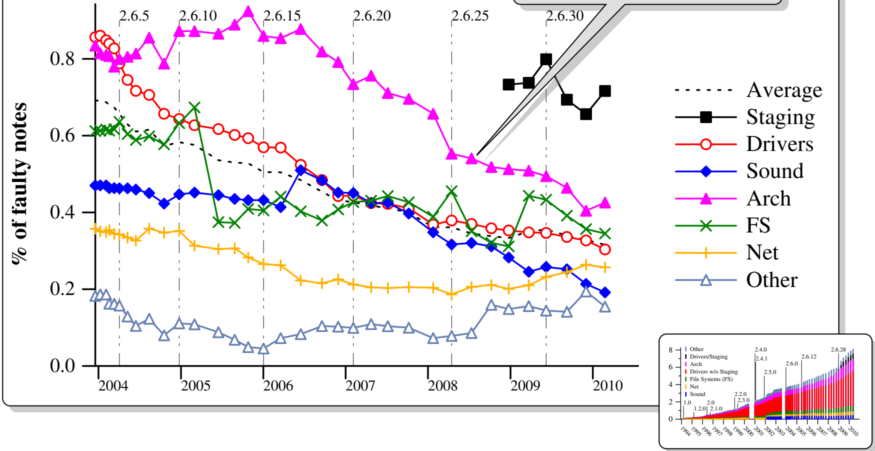


Fault rate

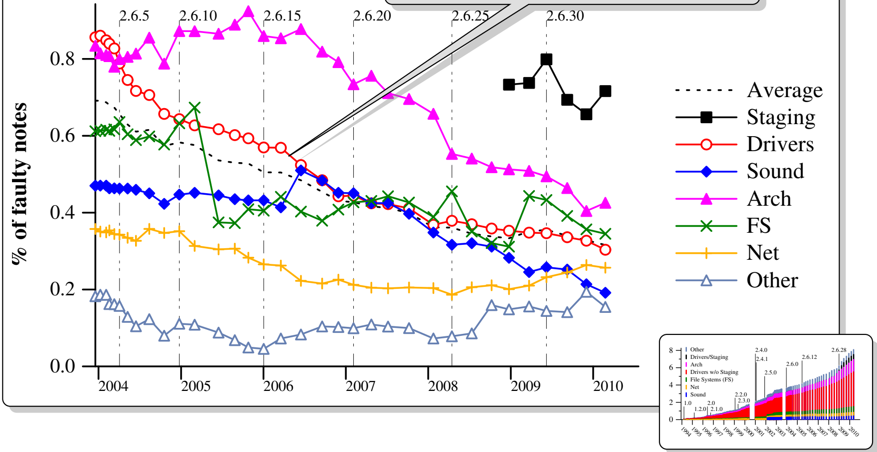

Staging for immature code

Fault rate

Arch has improved but remains a problem

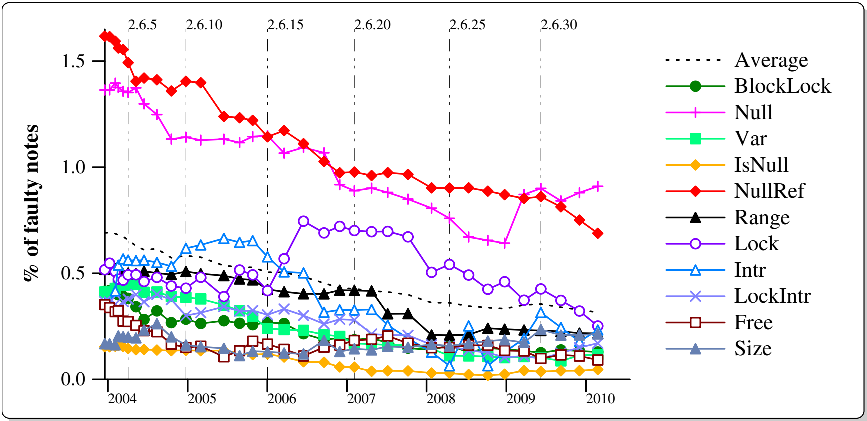


Fault rate

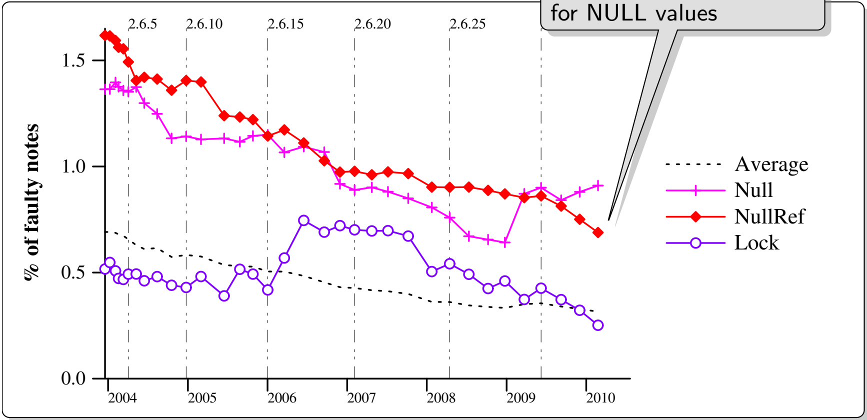


Drivers are constantly improving
From worst to average

Fault kinds

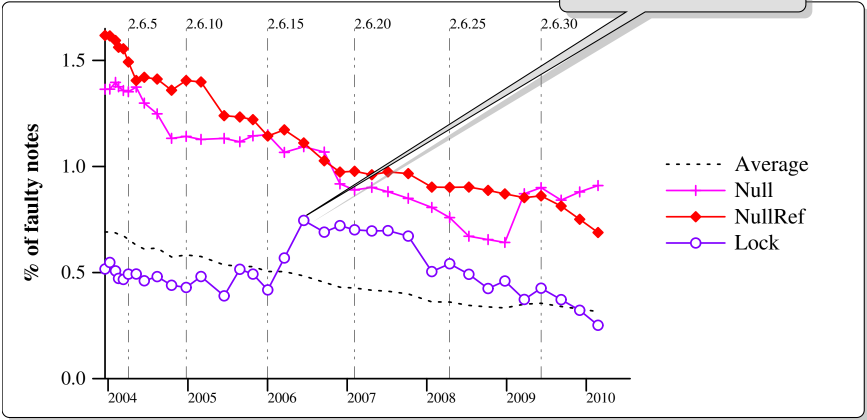

2.6.5
2.6.10
2.6.15
2.6.20
2.6.25
2.6.30
% of faulty notes
1.5
1.0
0.5
0.0
2004
2005
2006
2007
2008
2009
2010
Average
BlockLock
Null
Var
IsNull
NullRef
Range
Lock
Intr
LockIntr
Free
Size

Fault kinds



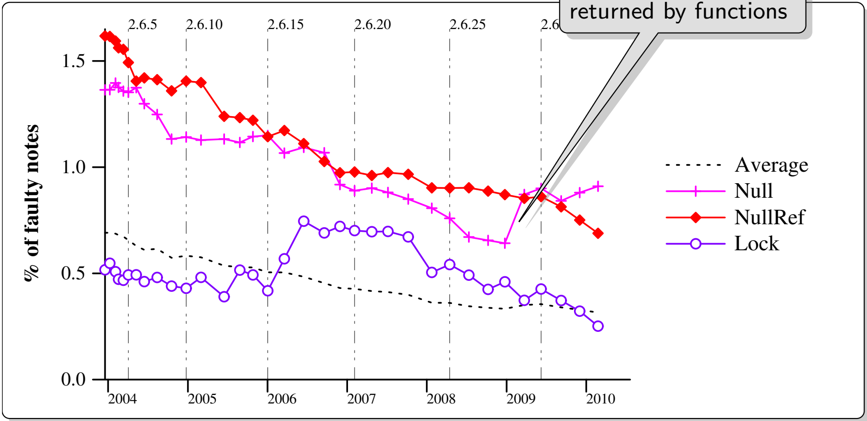
Dereference of pointers
before checking
for NULL values

Fault kinds



Mutex was added

Fault kinds



Btrfs was added
Unchecked pointers
returned by functions

Assessment

Fault trends

- Fault rate in drivers decreasing.
- Fault rate high in arch, but also decreasing.
- Arch has many committers, but few patch authors.

New functionalities often cause a spike in fault rate

- These issues are resolved over time.

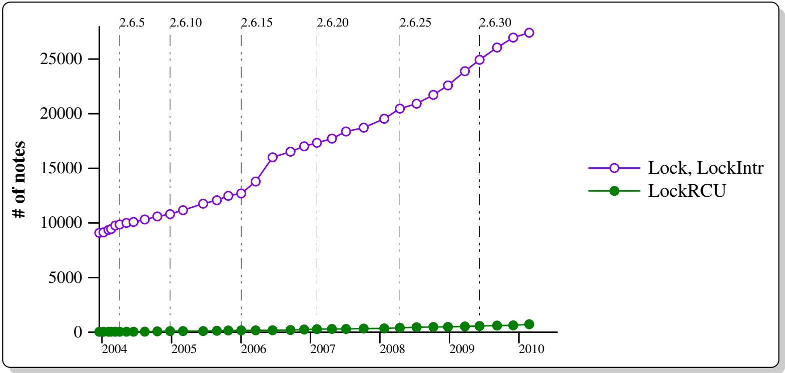
The management of pointers is still a problem

- NULL is often used to compensate for the lack of exceptions.

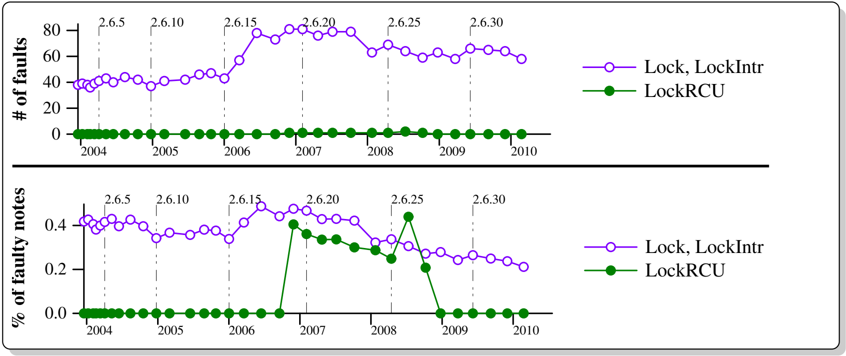
What next?

Consider new fault types

- RCU: Read-copy-update locking
- Increasingly used, but not as much as traditional locks.

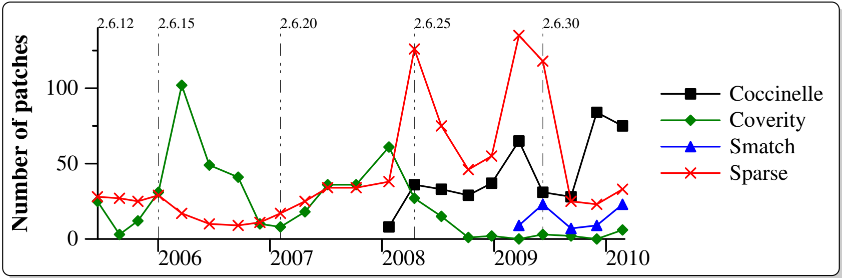


Some faults found



Systematize tool usage

- Since 2001 all of our faults could be found by tools.
- Still, between 600 and 700 faults per version.
- Tools not deeply integrated into the development process.
- Finding a fault can be easier than fixing it.



Conclusion

Methodology

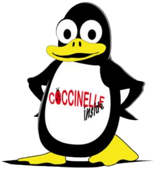
- Based on open-source (FLOSS)
- Fault definitions in Coccinelle

Linux 2.6

- Number of faults is roughly constant while the code size is increasing
- Drivers have improved,
  now at the average fault rate
- Arch now has the worst fault rate
- NULL handling is still a problem

Existing tools are under-exploited

- Integration of Coccinelle rules since v2.6.36
- We have contributed some patches to Linux based on our results.



Future work

Add new checkers

- Reflect new APIs

Integrate tools better into in the Linux development process

- Ease the use of Coccinelle

Does ASPLOS 2012 need a session on
“Enhancing Arch Reliability”?

Availability – Results and tools

Fault definitions and correlation annotations

- `http://faultlinux.lip6.fr/`

Database of reports

- `http://faultlinux.lip6.fr/phppgadmin/` (Browseable)
- `http://hal.inria.fr/inria-00509256/` [RR7357 appendix]

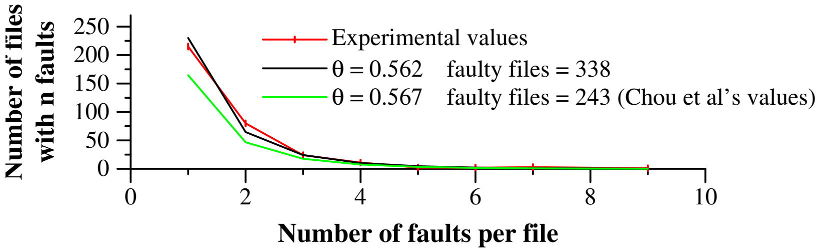
Coccinelle

- `http://coccinelle.lip6.fr/`

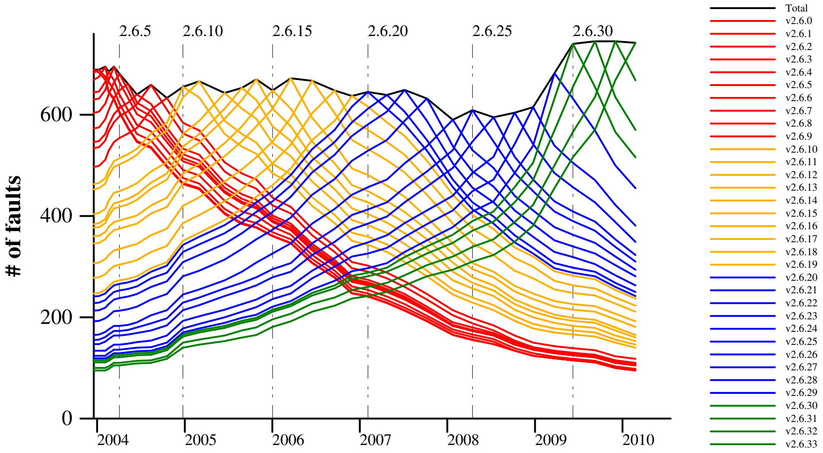
Herodotos

- `http://coccinelle.lip6.fr/herodotos.html`

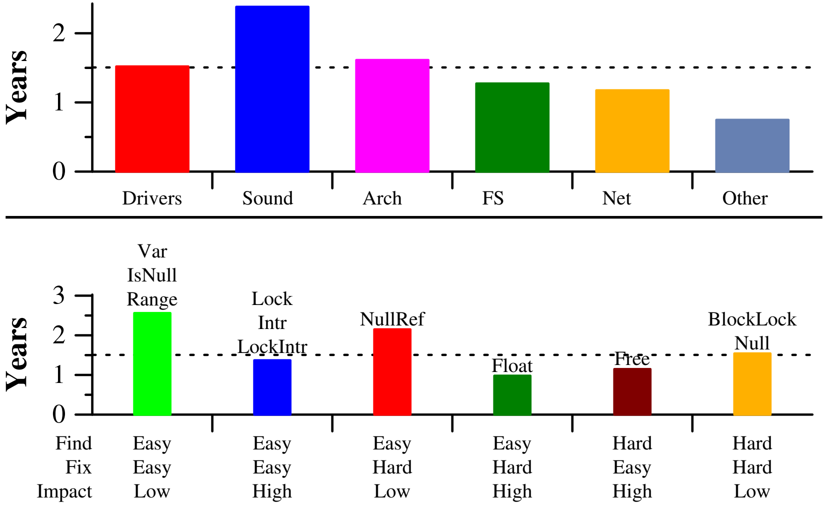
Comparable results (2)



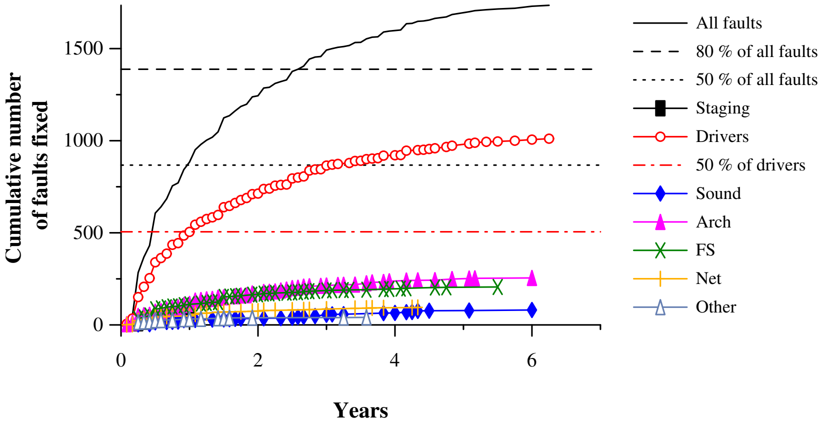
Stable fault lifespans



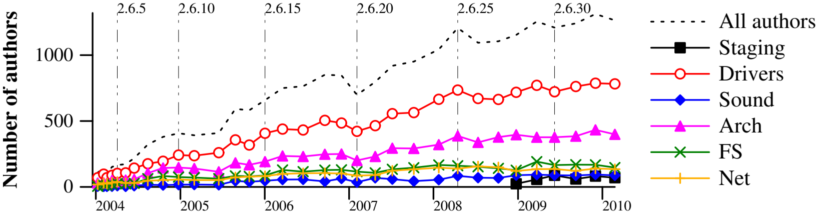
Average lifetime

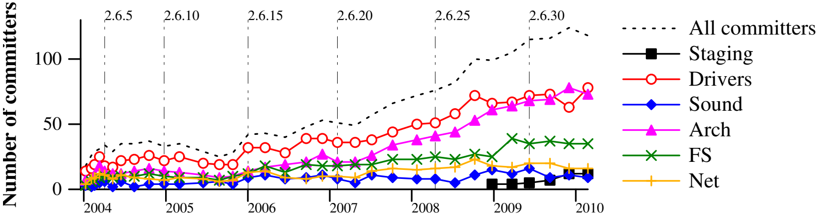


Time to fix faults



Linux workforce


Number of authors
All authors
Staging
Drivers
Sound
Arch
FS
Net


Number of committers
All committers
Staging
Drivers
Sound
Arch
FS
Net