

The suckless.org universe

anselm@garbe.us

suckless.org

- Founded ~2002 (wmi, wmii)
- Approx. 5000 users world wide
- ~1000 dev@suckless.org subscriptions
- ~110 people in #suckless @ irc.oftc.net

Disclaimer: Expect **harsh** feedback/criticism

simple, minimal, usable,
advanced user focus

Philosophy

**KISS, less is more,
worse is better**

Philosophy

*Write programs that do one
thing and do it well.*

Doug McIlroy

Metrics of sucking

1. (S)LOC

(Source) lines of code

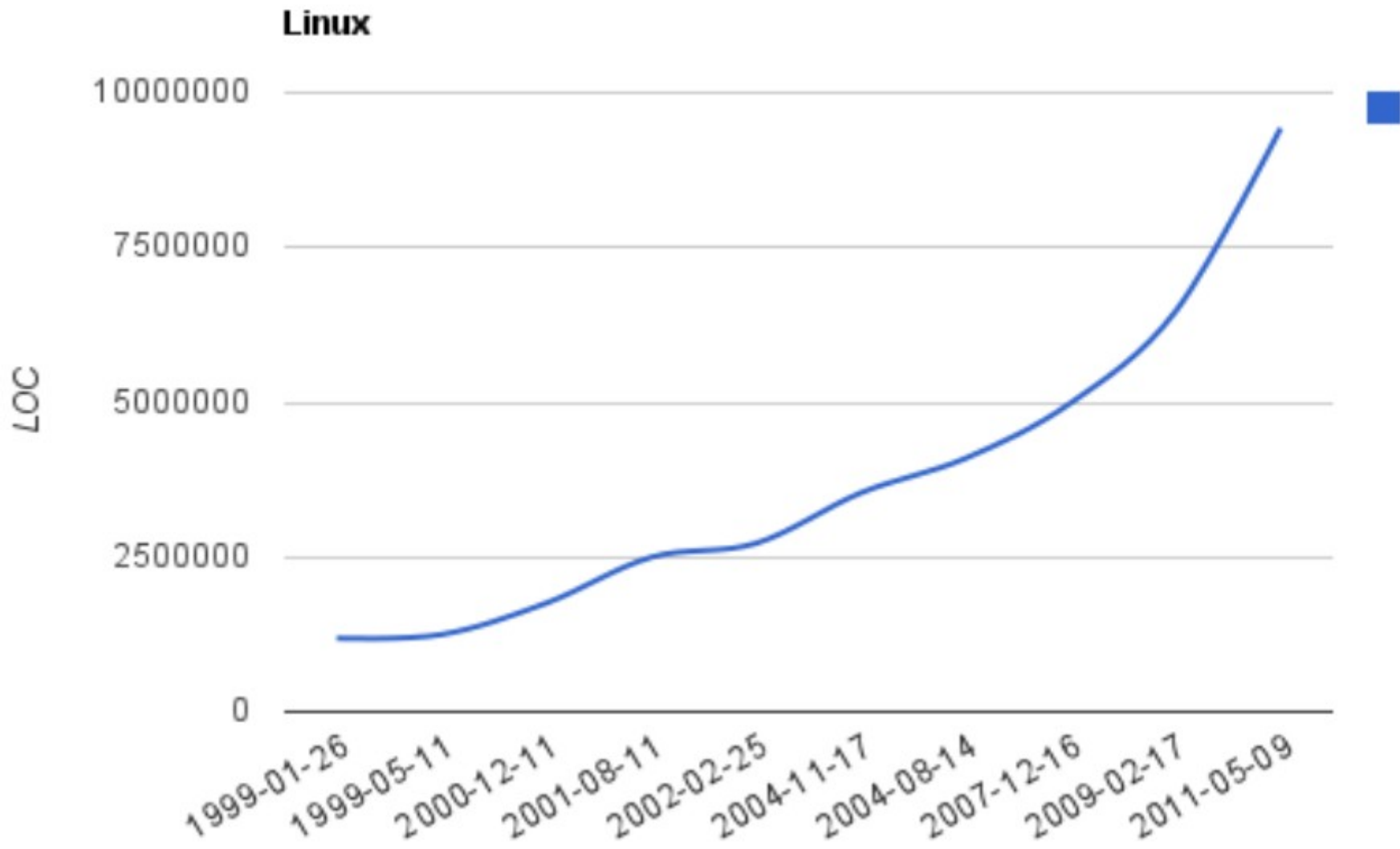
1. (S)LLOC

High SLOCs may indicate:

- Code complexity
- Bad software design
- Wrong philosophy
- Bugs
- High maintenance costs
- Soon end

→ SLOCCount, David A Wheeler or `wc -l`

1. (S)LOC – Linux is extreme



Linux source grows
1M SLOC/year

Do they **never** remove code?

1. (S)LOC – Linux ext2 vs btrfs

Linux 2.2.0 ext2 implementation 1999:

→ 4,260 SLOC

Linux 2.6.38.6 btrfs implementation 2011:

→ 46,619 SLOC

2021:

→ 100,000 SLOC?

2. Programming language

Indirect metric

2. Programming language

A bad choice may lead to:

- Code complexity
- Bad software design principles
- Many bugs
- The 'need' of “Design Patterns”
- High maintenance costs
- A rewrite sooner than later

*Being really good at C++ is like being
really good at using rocks to sharpen
sticks.*

Thant Tessman

*Java: the elegant simplicity of C++ and
the blazing speed of Smalltalk.*

Jan Steinman

Ruby is Scheme mated with Perl in such a way that the best genes of both failed to exert a phenotype.

frumious in reddit

*C++: an octopus made by nailing extra
legs onto a dog.*

Steve Taylor

3. Harmful

harmful.cat-v.org

3. Harmful

Harmful

- SGML, XML, YAML
- NFS, SMB, WebDAV
- Perl, ruby
- bash, tcsh, zsh
- glibc
- GNU autohell, gmake
- UTF-16, UTF-32, latin1
- Jabber, XMPP
- SQL databases
- head

Less harmful

- JSON, CSV, plain text
- 9P
- rc, awk
- OpenBSD pdksh, rc
- uclibc, musl, dietlibc
- mk, portable Makefiles
- UTF-8
- IRC
- filesystem
- sed 11q

High SLOC + Wrong lang +
more harmful stuff

= YOU SUCK

Dedicated to software that
sucks *less*

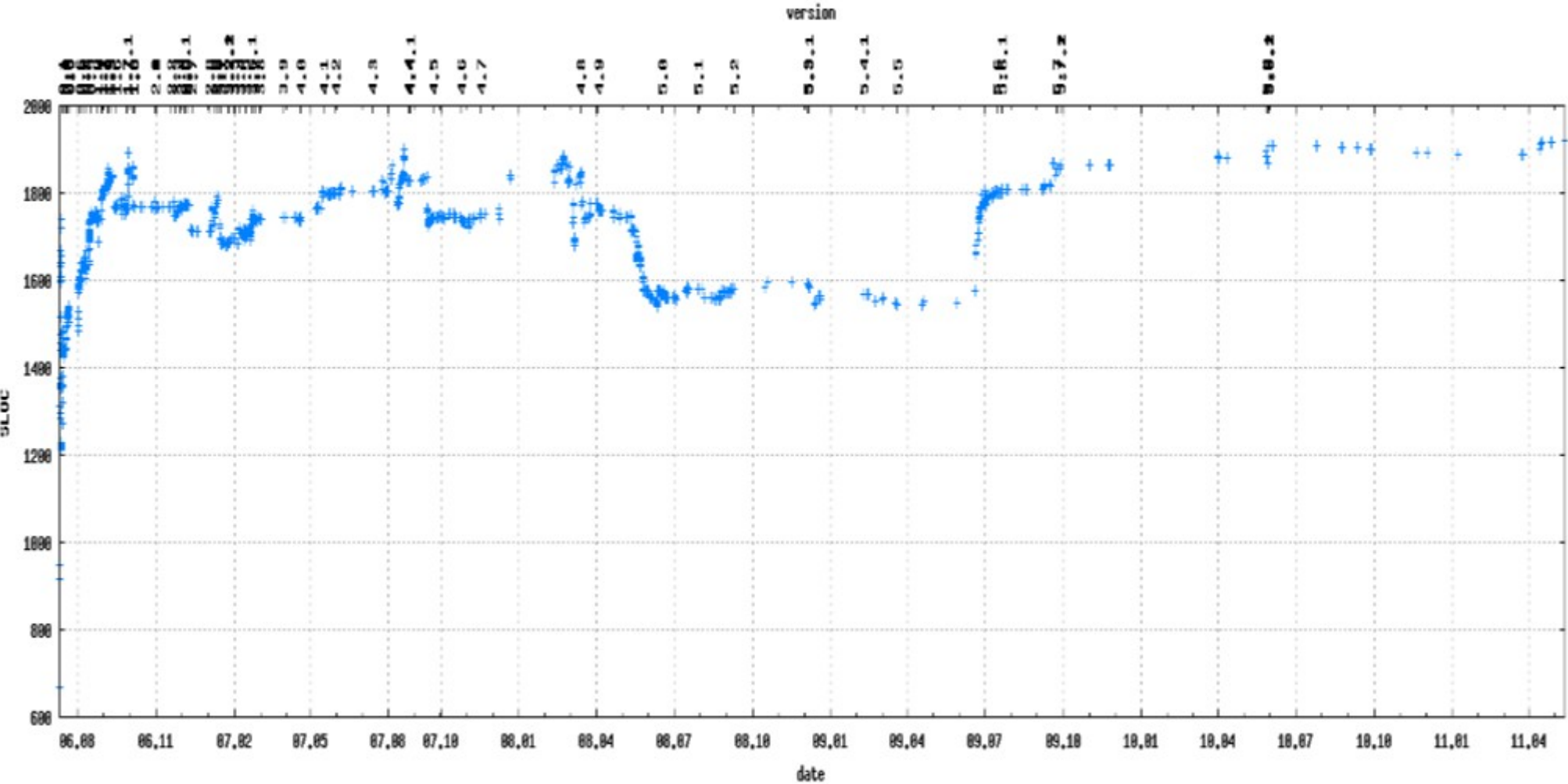
dwm

dynamic window manager

dwm

- Started 2006
- No 9P
- No configuration files
- tagging
- dynamic tiling
- ~2k SLOC
- MIT License

dwm - SLOC evolution



dwm demo

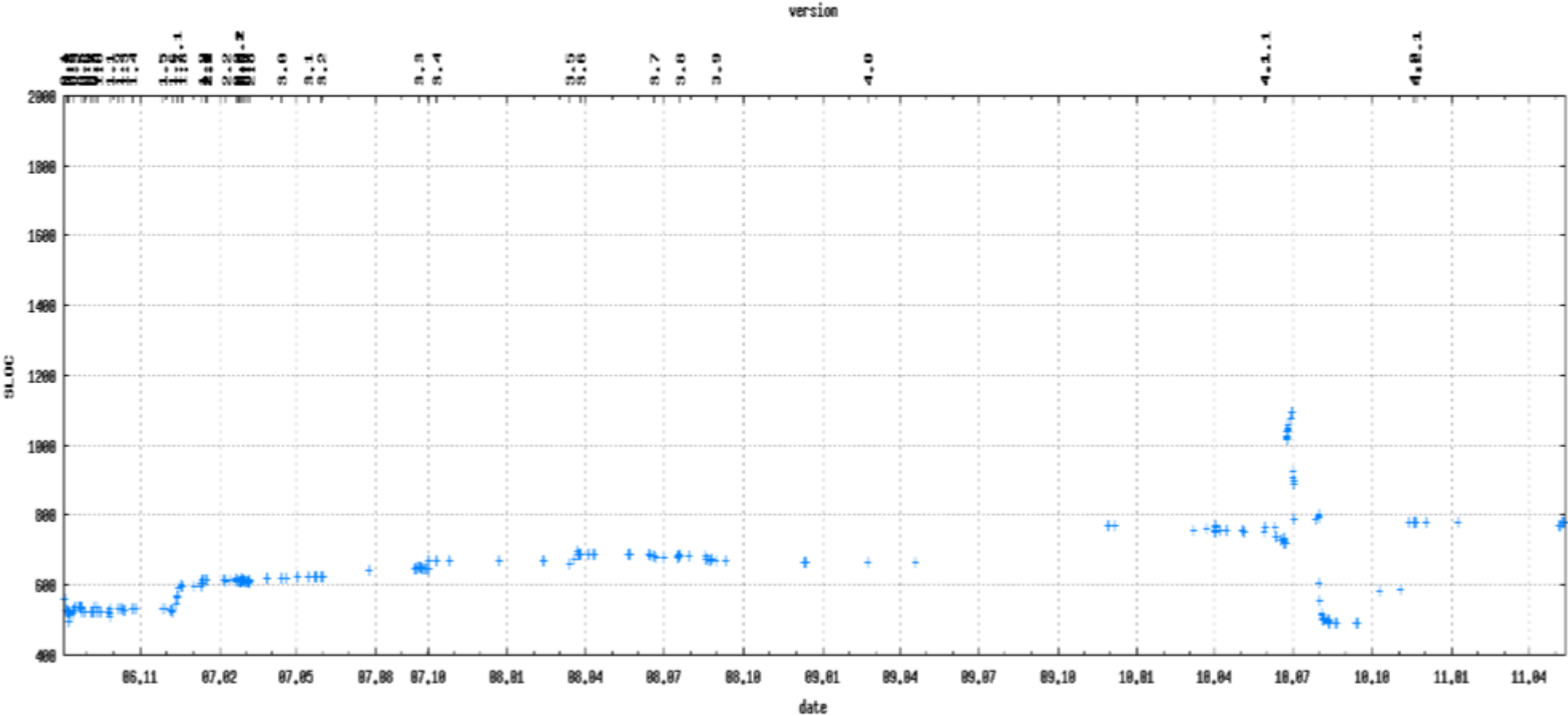
dmenu

dynamic menu

dmenu

- Started 2006
- Simple I/O program
- 10000+ menu items
- <1k SLOC
- MIT License

dmenu - SLOC evolution



dmenu demo

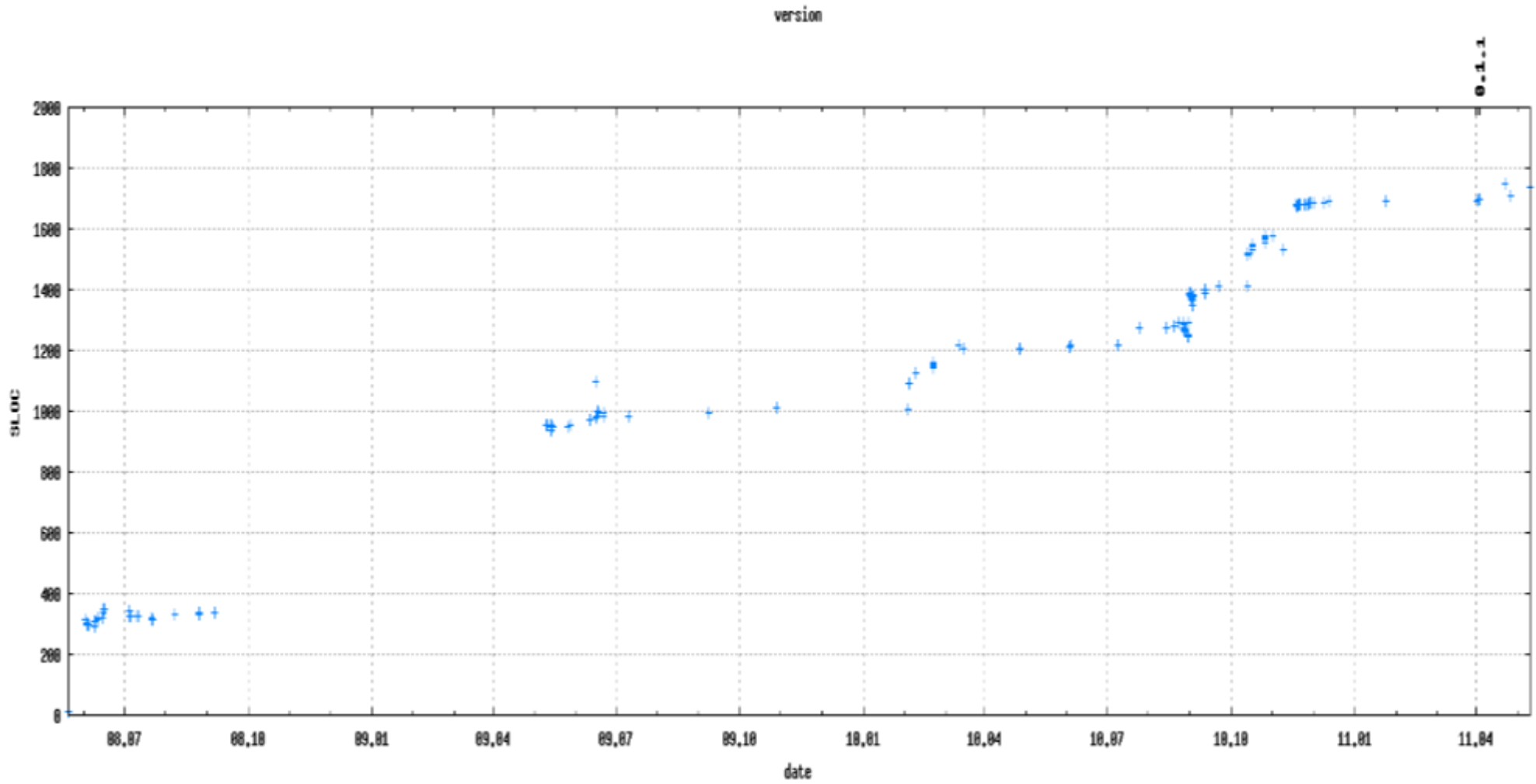
st

simple terminal

st

- Started 2008
- Aurelien Aptel, Matthias Christian Ott, et.al.
- Early stage
- <2k SLOC
- MIT License

st - SLOC evolution



st demo

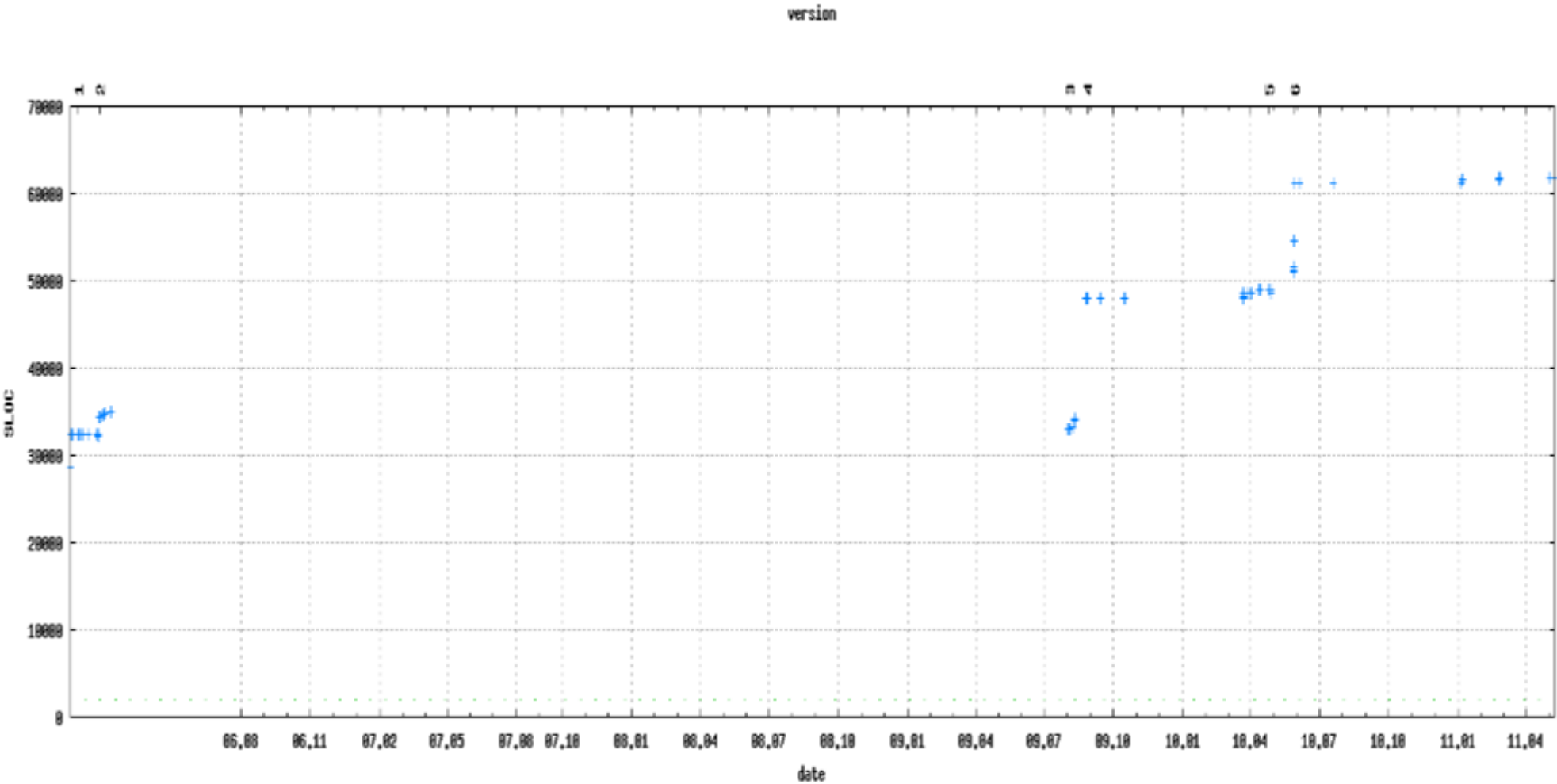
9base

Static Plan 9 userland selection
for Unix/Linux

9base

- Started 2005
- Plan 9 libc, libbio, libregex, libfmt and libutf
- ascii, awk, basename, bc, cal, cat, cleanname, cmp, date, dc, du, dd, diff, echo, ed, factor, fortune, fmt, freq, getflags, grep, hoc, join, look, ls, mk, mkdir, mtime, pbd, primes, **rc**, read, sam, sha1sum, sed, seq, sleep, sort, split, strings, tail, tee, test, touch, tr, troff, unicode, uniq, unutf
- ~60k SLOC

9base - SLOC evolution



9base

→ bash 4.2 **alone** has 120k SLOC

*A **single shell** is twice the size as
a **complete userland**?*

Why 9base?

- Does sed/awk/grep/... support this option?
- GNU make or BSD make or nmake or something else?
- UTF-8 capable grep?
- ...

- **werc** can be run with 9base
- My network monitor uses 9base

9base demo

rc – the Plan 9 shell

sta.li

static linux

sta.li

- Started 2008
- **Still almost** vaporware
- Focus on static executables
- Small core system
- No configuration
- No multi-user scope
- init system: just `/etc/rc.start`

sta.li demo?

Not today

How can **you** suck less?

Some conclusions

1. Apply the Unix philosophy

2. Use the right tool for the task

3. KISS, less is more, worse is better

4. Removed code is debugged code.

5. UI: don't confuse users with
achieving the same in different
ways

6. Never perform big changes.

7. Rewrite from scratch instead of trying to fix.

8. If your software isn't usable from day 1, your approach sucks.

9. Data structures are more important than actual code.

10. Use the simplest data structure.

Don't make up performance issues.

11. *If in doubt, use brute force.*

Ken Thompson

12. Never design "future extensions".



Good luck

Thanks