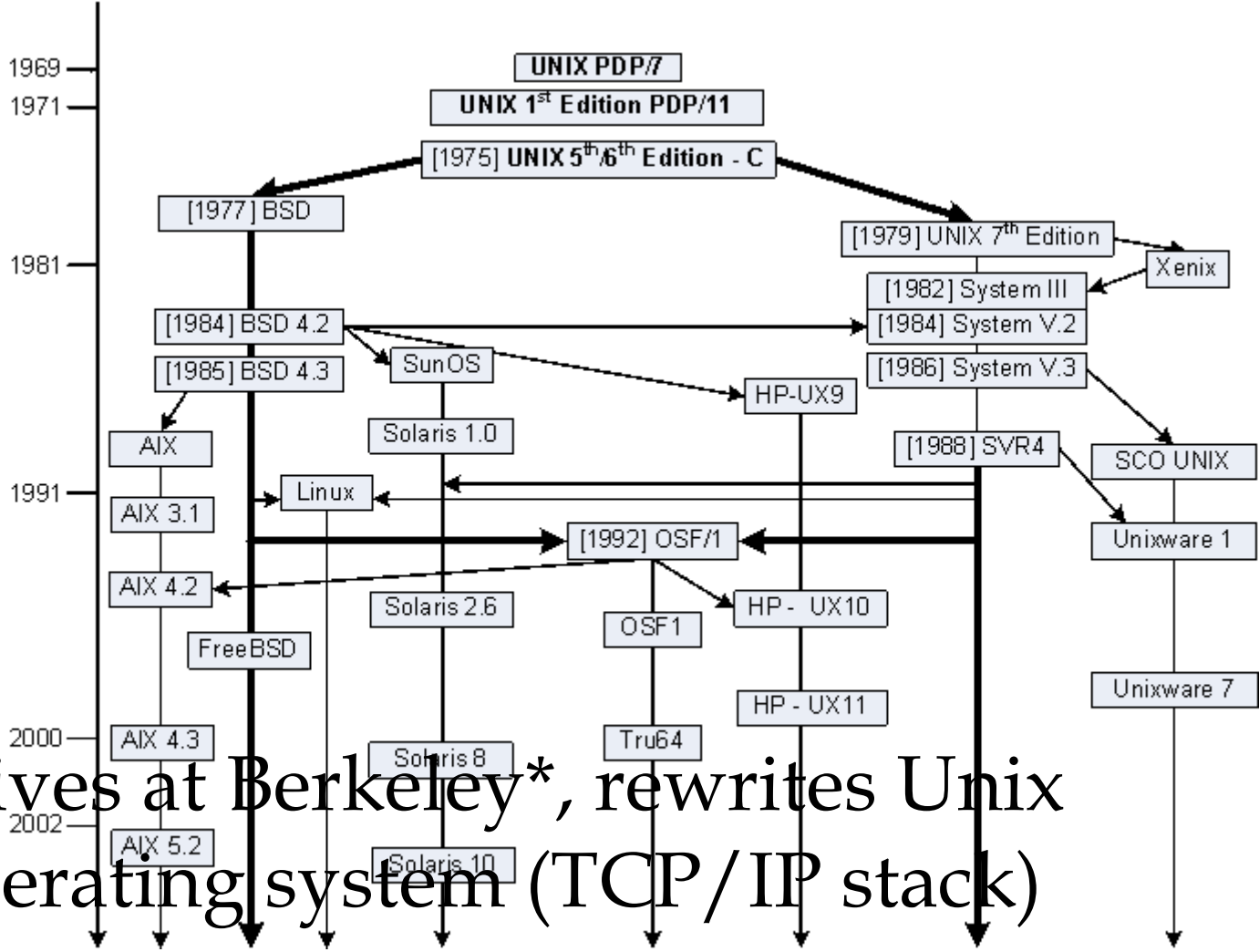
George N. Dafermos
g.n.dafermos@tudelft.nl
4th Oekonux Conference, Manchester University, March 29, 2009

Division of labour in free & open source software development: the FreeBSD project

Definition

The term free software and open source software denotes computer software that is freely distributed and made generally available under a license (eg. GNU GPL) or institutional arrangement (ie. public domain) giving users the right to *use, modify and re-distribute* modified or unmodified versions.

(very) brief timeline



1969 – Unix

1975 – Bill Joy arrives at Berkeley*, rewrites Unix

1980s: DARPA operating system (TCP/IP stack)

1982 – Bill Joy leaves

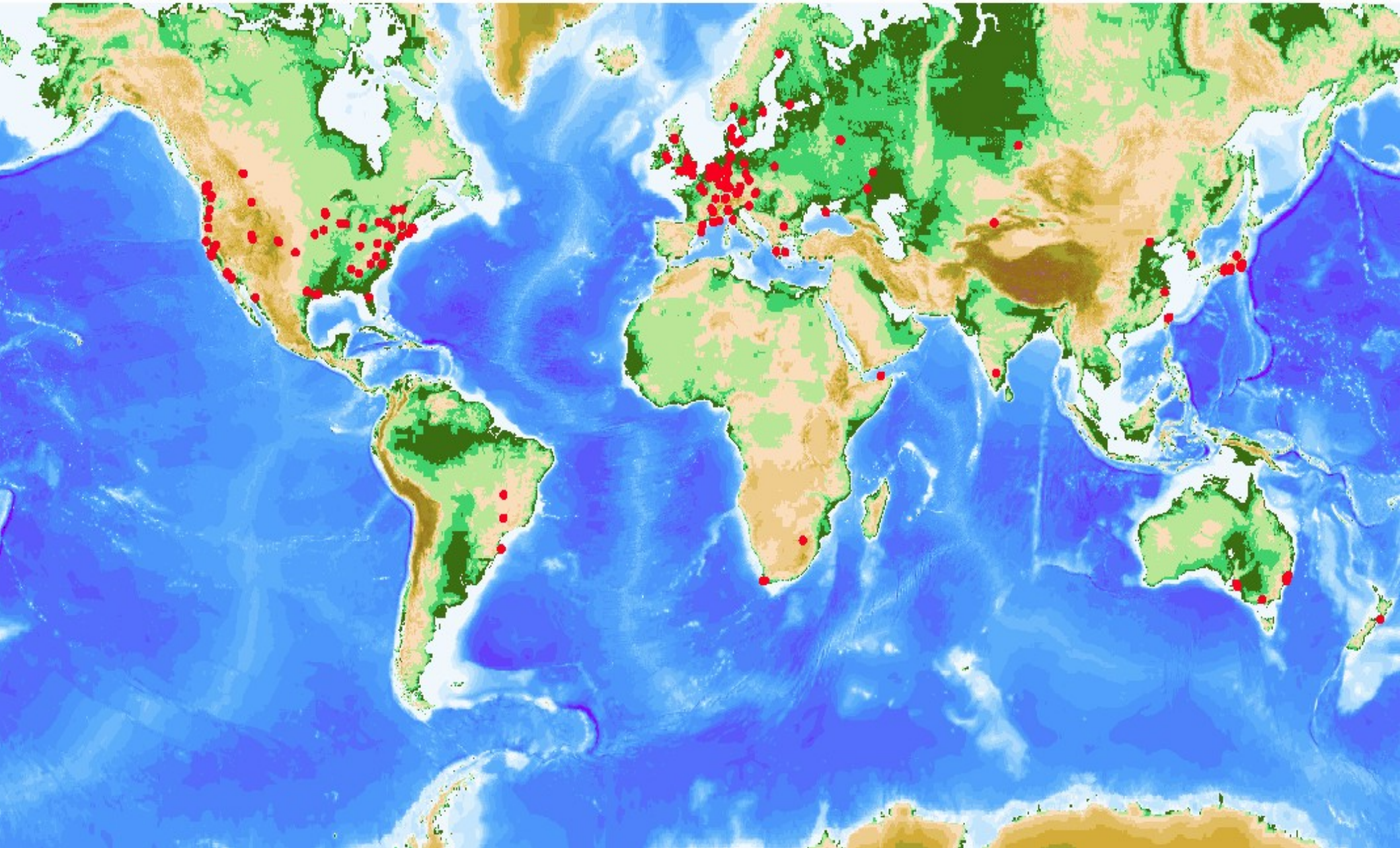
1990s – AT&T lawsuit (1992); 386BSD forks into FreeBSD and NetBSD (1993) ; FreeBSD v.1.0 (Dec. 1993); FreeBSD v.2.0 (Jan. 1995)

2000s – FreeBSD most popular BSD-descendant

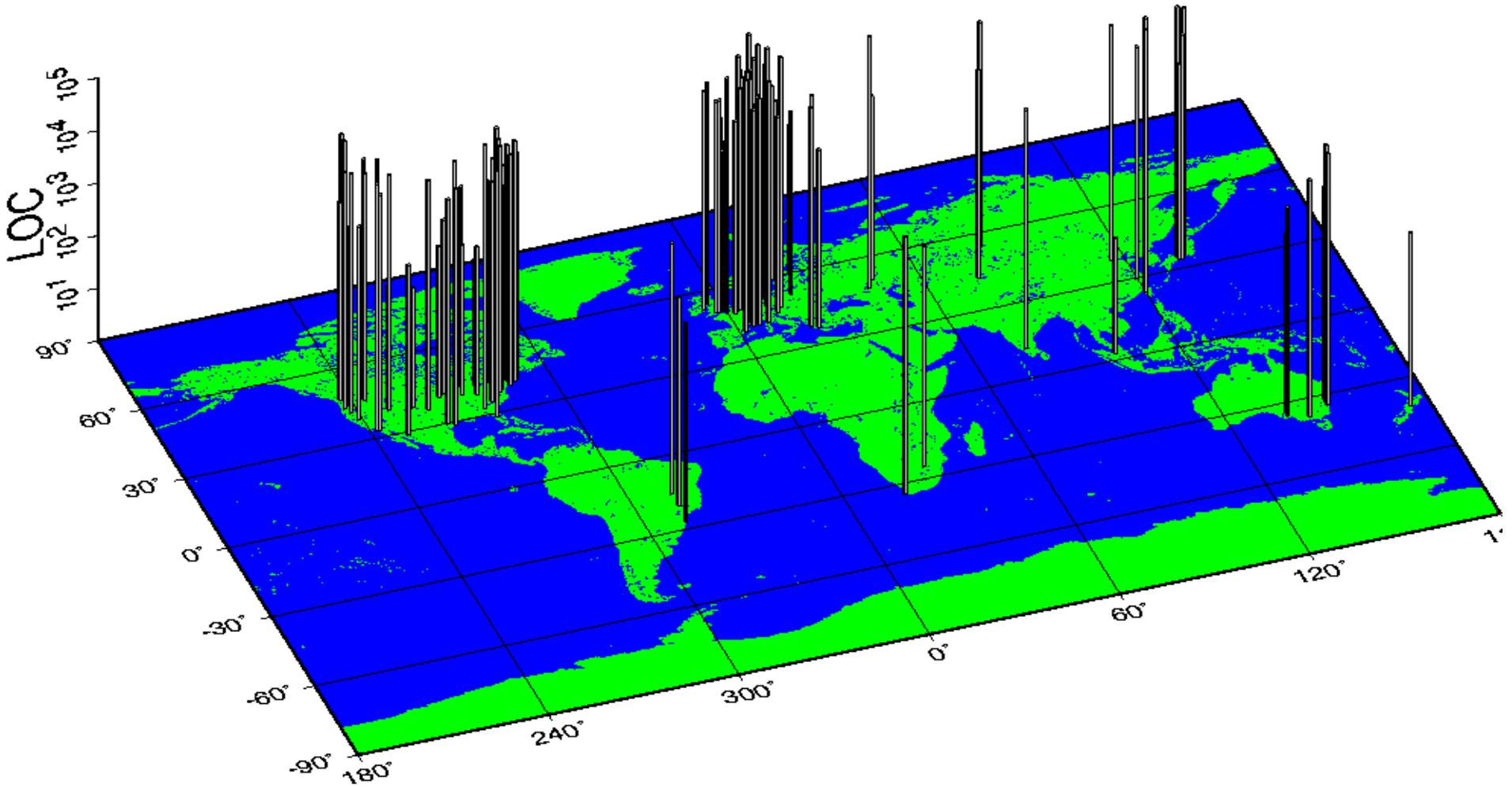
Forking

- Right to fork = Easy access to exit option
- Exit option dampens the emergence of conflicts
- Forking is an extreme example of the exit option
- Conflicts are usually translated into parallel development lines
- Under which conditions does a project fork?

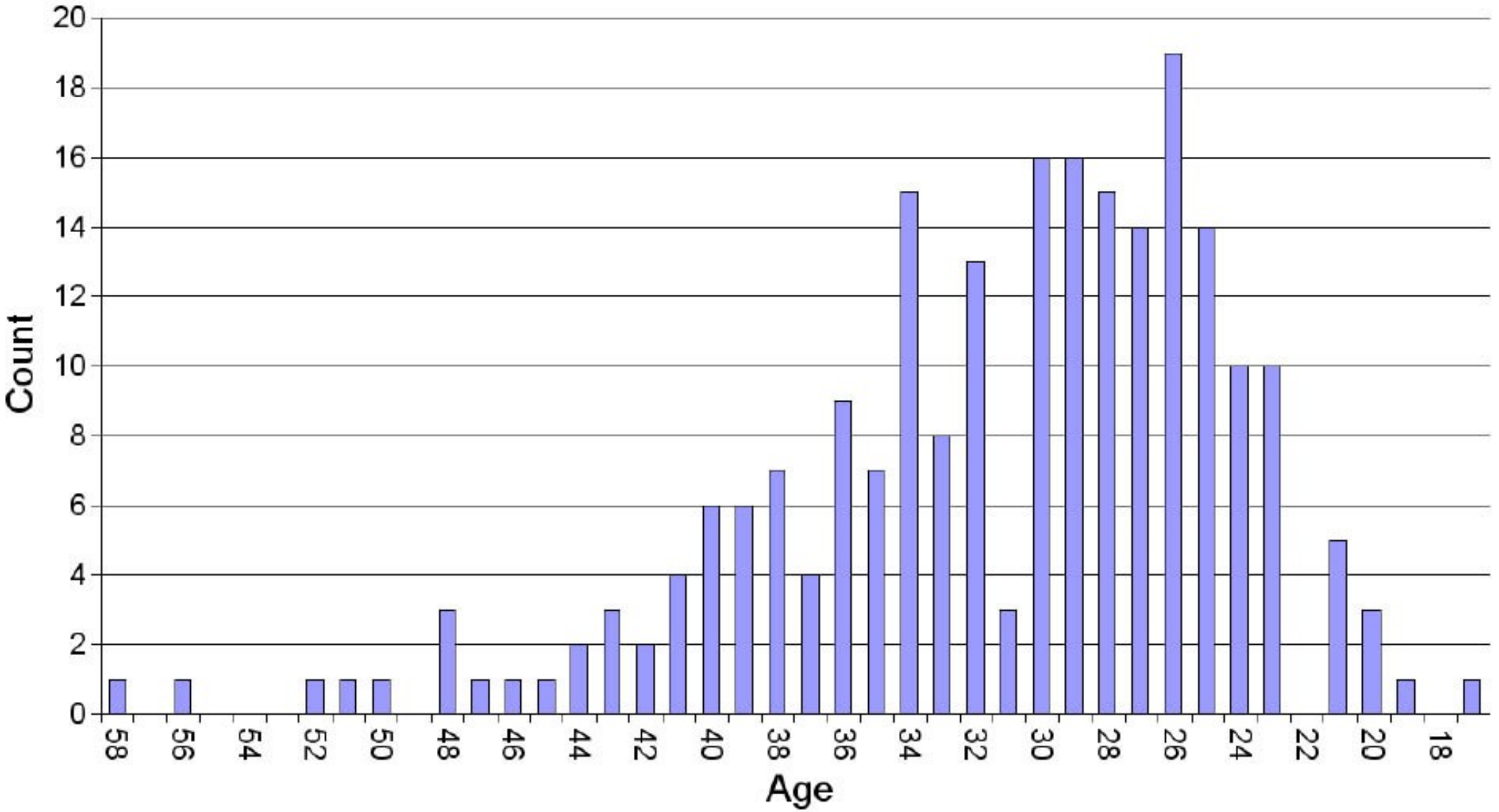
34 countries, 6 continents



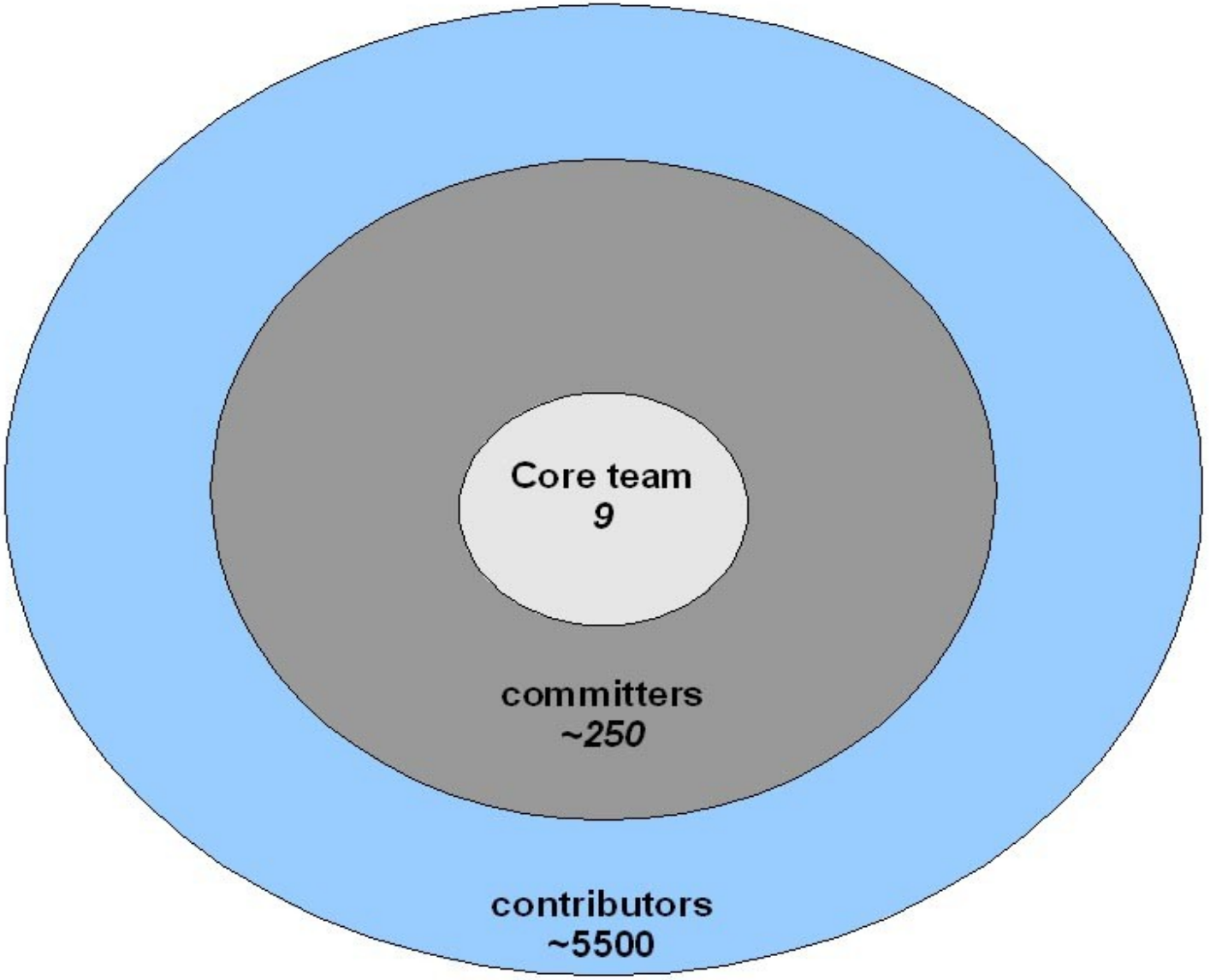
international

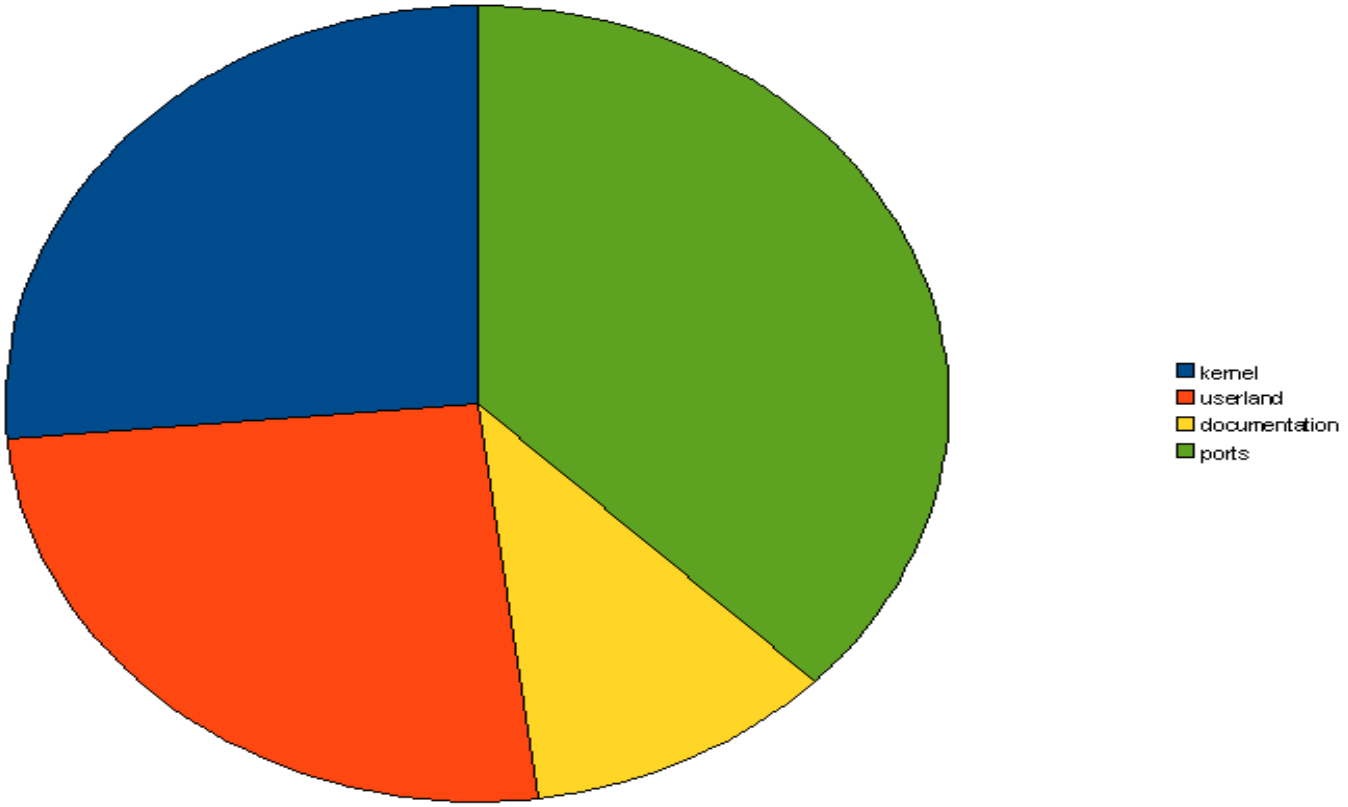


age


Count
20
18
16
14
12
10
8
6
4
2
0
58
56
54
52
50
48
46
44
42
40
38
36
34
32
30
28
26
24
22
20
18
Age

Organisational structure





Of the 275 committers who made commits in 2002

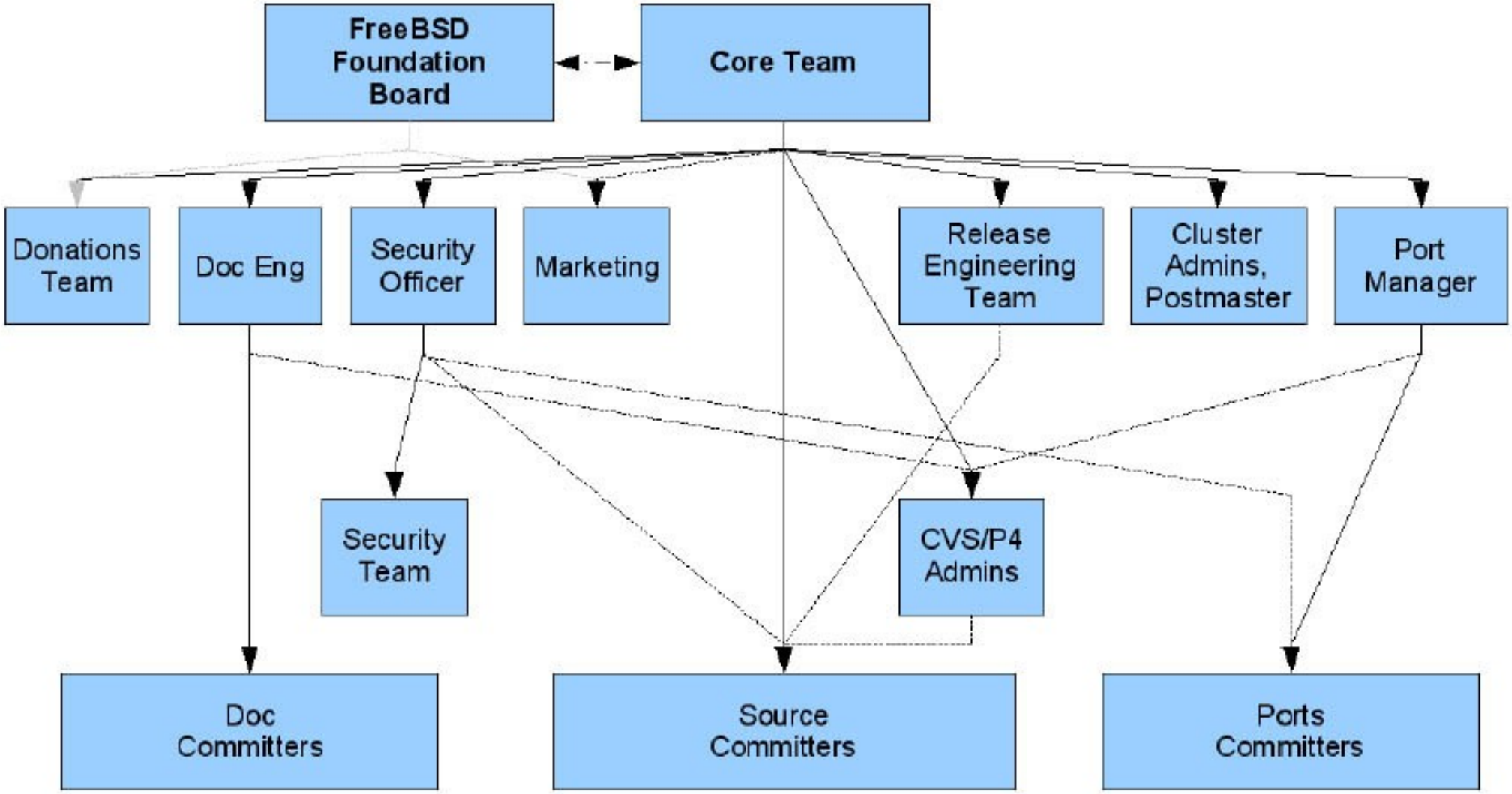
- 102 kernel committers
- 99 userland committers
- 41 documentation committers
- 144 ports committers

How-to become a committer

"If you submit enough useful and correct problem reports (PRs) [or patches] eventually some committer will get sick of taking care of your work and will ask you if you want to be able to commit them yourself"

(M. Lucas, 2002)

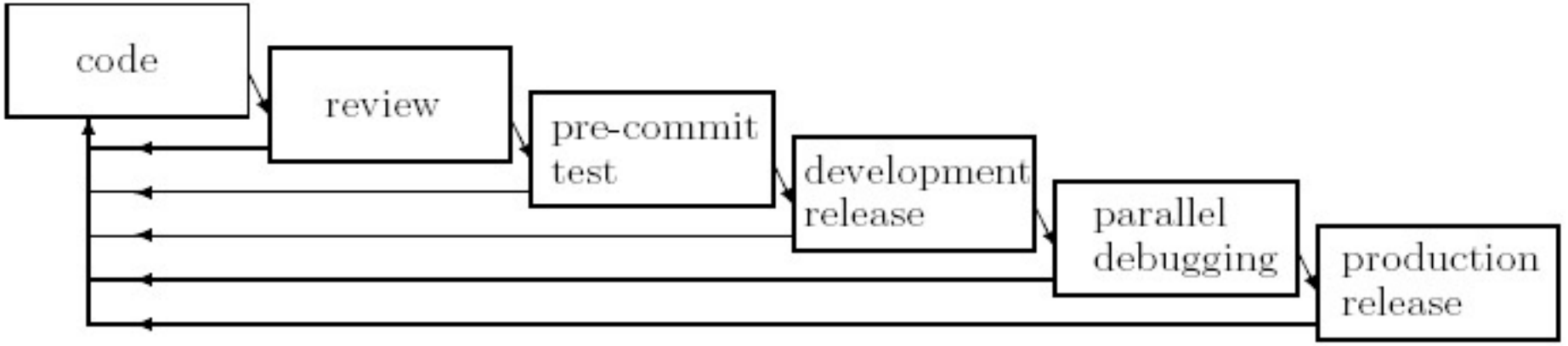
Organisational chart



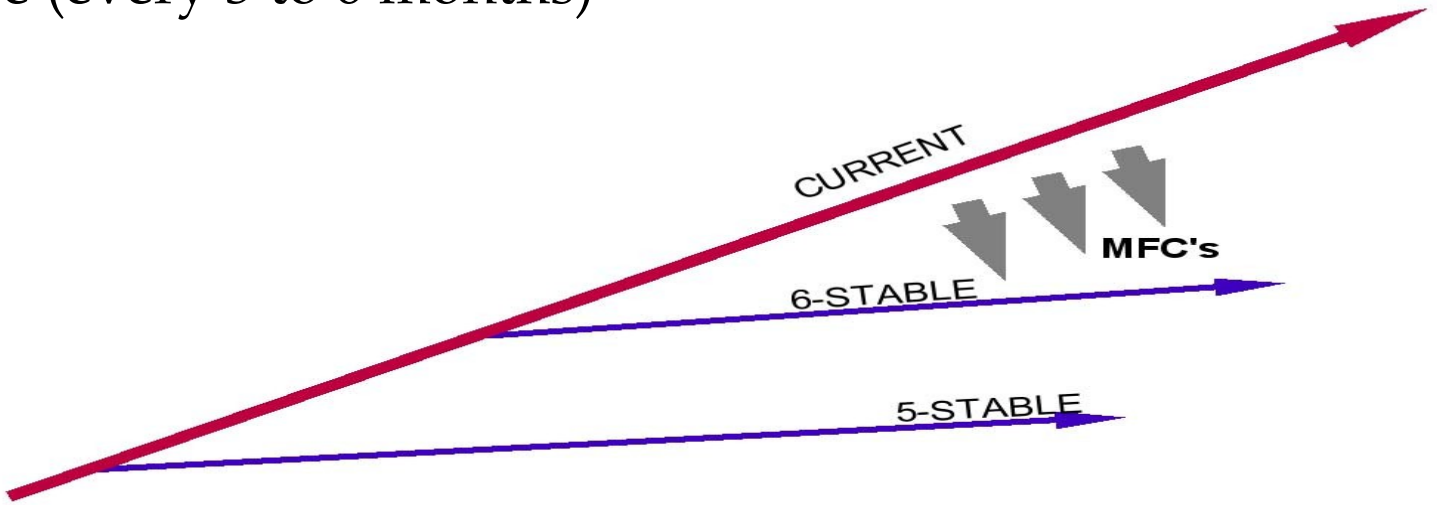
Development tools

- ***CVS***: revision control (primary)
- ***Perforce***: revision control (support for highly-branched development/new kernel development)
- ***GNATS***: database maintenance (ie. problem-reports)
- ***Mailing lists***: main communication channel
  - About 70 public lists
  - Plus some team-specific private lists: i) committers, ii) core team, iii) Release Engineering Team, iv) Port Manager
- ***Tinderboxes***: build process (doing a daily build is the key coordinating mechanism)
- ***PGP***: public-key cryptography
- ***To-do lists***
- ***CVSup***: distribution (164 servers in 50 countries in Feb. 2009)

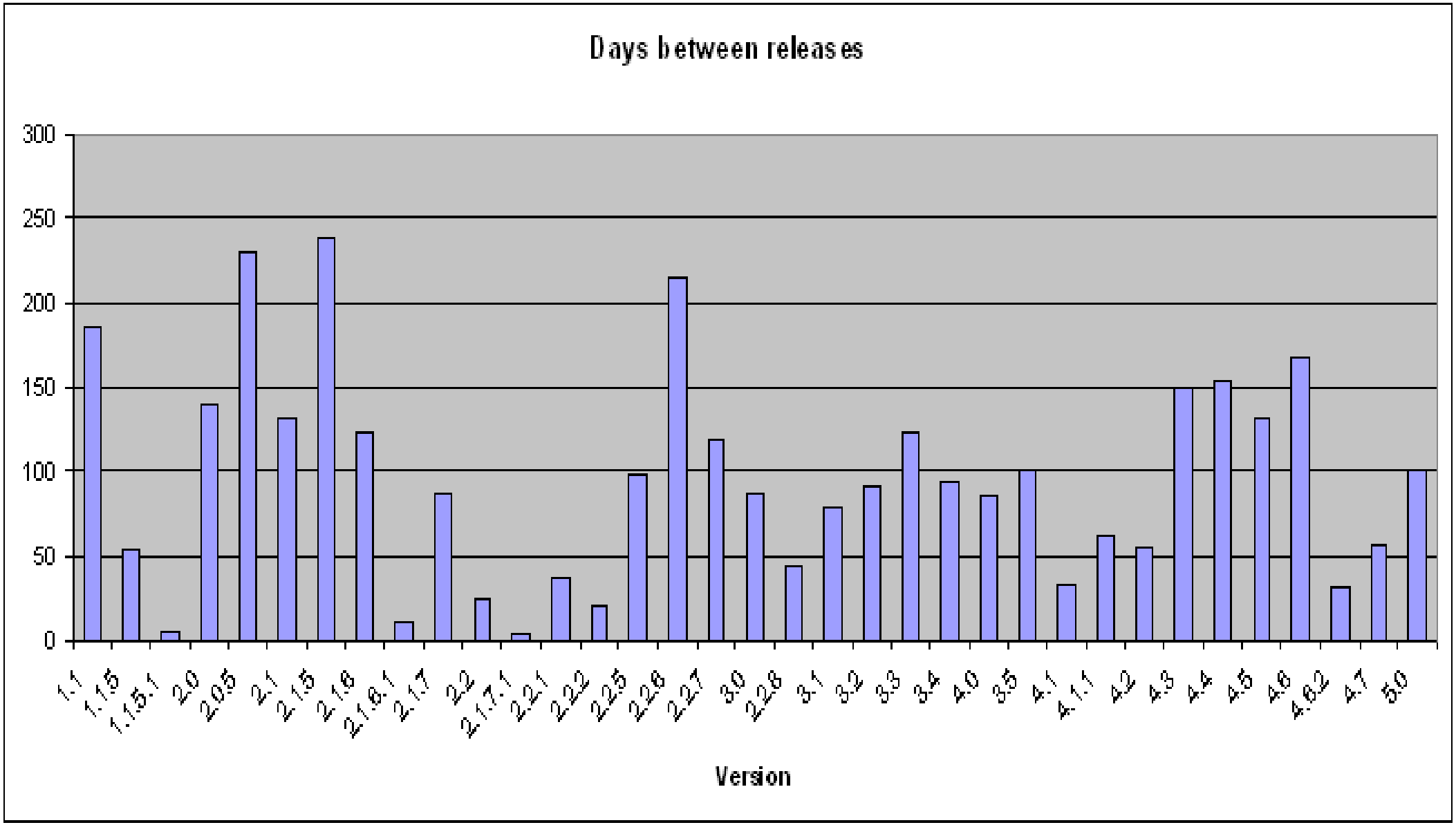
Development process



After coding, ask for community review – test locally – commit to *current* branch --> development release made twice a day available for download – downloaded, tested and debugged – when stable, merged by committer in *stable* (*MFC*) --> 'code slush', 'code freeze' - production release (every 3 to 6 months)



average 96,2 days (1993-2003)



Division of labour

Division of labour is emergent: not dictated by the top but premised on self-selection of tasks.

Is the immediate result of the usual procedure by which one joins a project and advances from peripheral (yet necessary) activities such as defect reporting and fixing to the development of new functionality.

Participation asymmetry is explained by that participants contribute according to their abilities.

1993-2003 (*current branch, src*)

334 individuals committed code

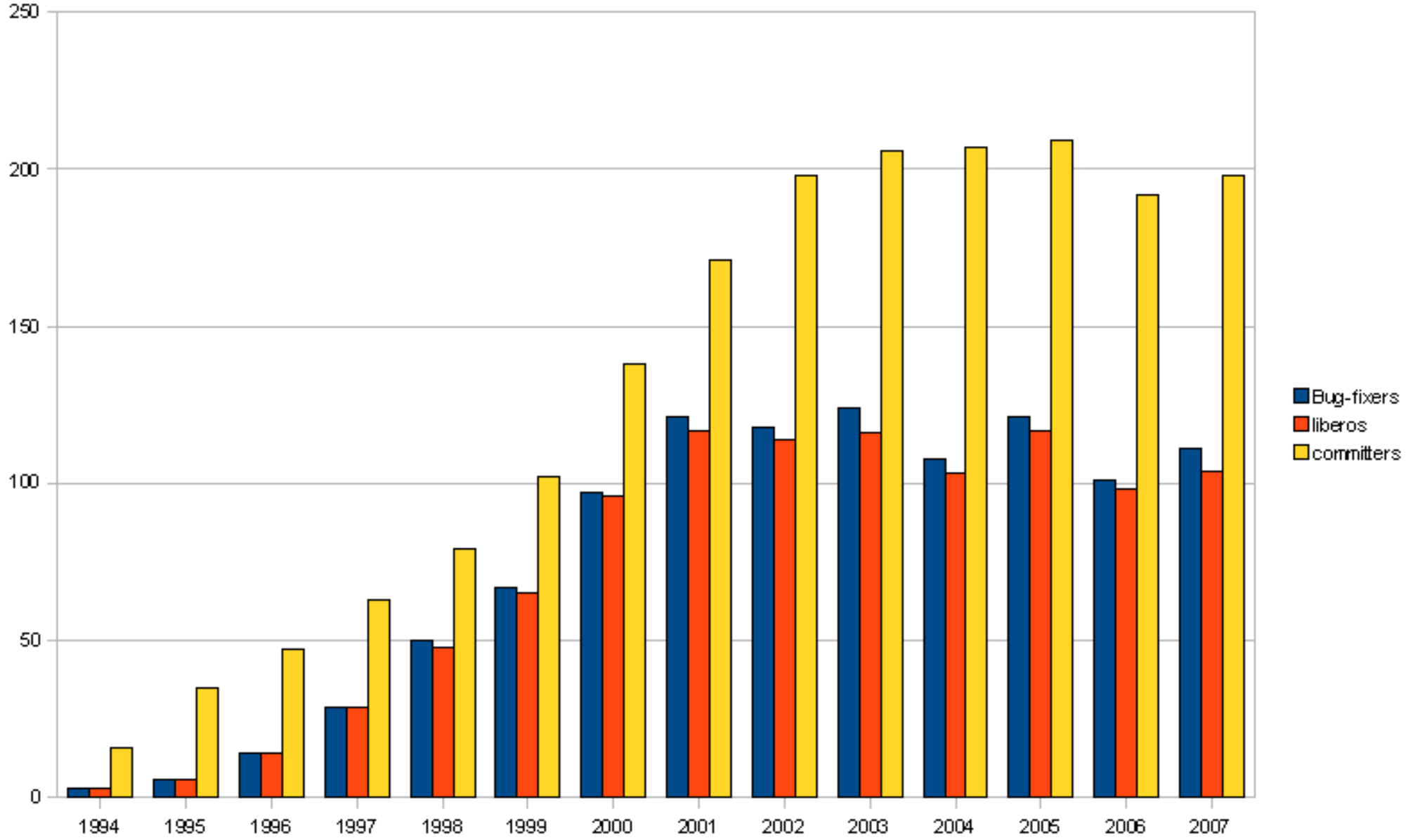
- Of them, 231 contributed 11,406 bug-fixes
- 329 contributed 516,540 changes for new functionality
- 226 checked-in code to both fix bugs and add new features
- Also, 5,645 individuals (of whom 183 are committers) contributed 16,115 bug-reports

Task specialisation falls over time

Year	*Bug-fixers/ all committers*	*Liberos* / all committers*
1993-1994	3/16	3/16
1994-1995	6/35	6/35
1995-1996	14/47	14/47
1996-1997	29/63	29/63
1997-1998	50/79	48/79
1998-1999	67/102	65/102
1999-2000	97/138	96/138
2000-2001	121/171	117/171
2001-2002	118/198	114/198
2002-2003	124/206	116/206
2003-2004	108/207	103/207
2004-2005	121/209	117/209
2005-2006	101/192	98/192
2006-2007	111/198	104/198
total(1993-2003)	*231/334*	*226/334*

FreeBSD current branch

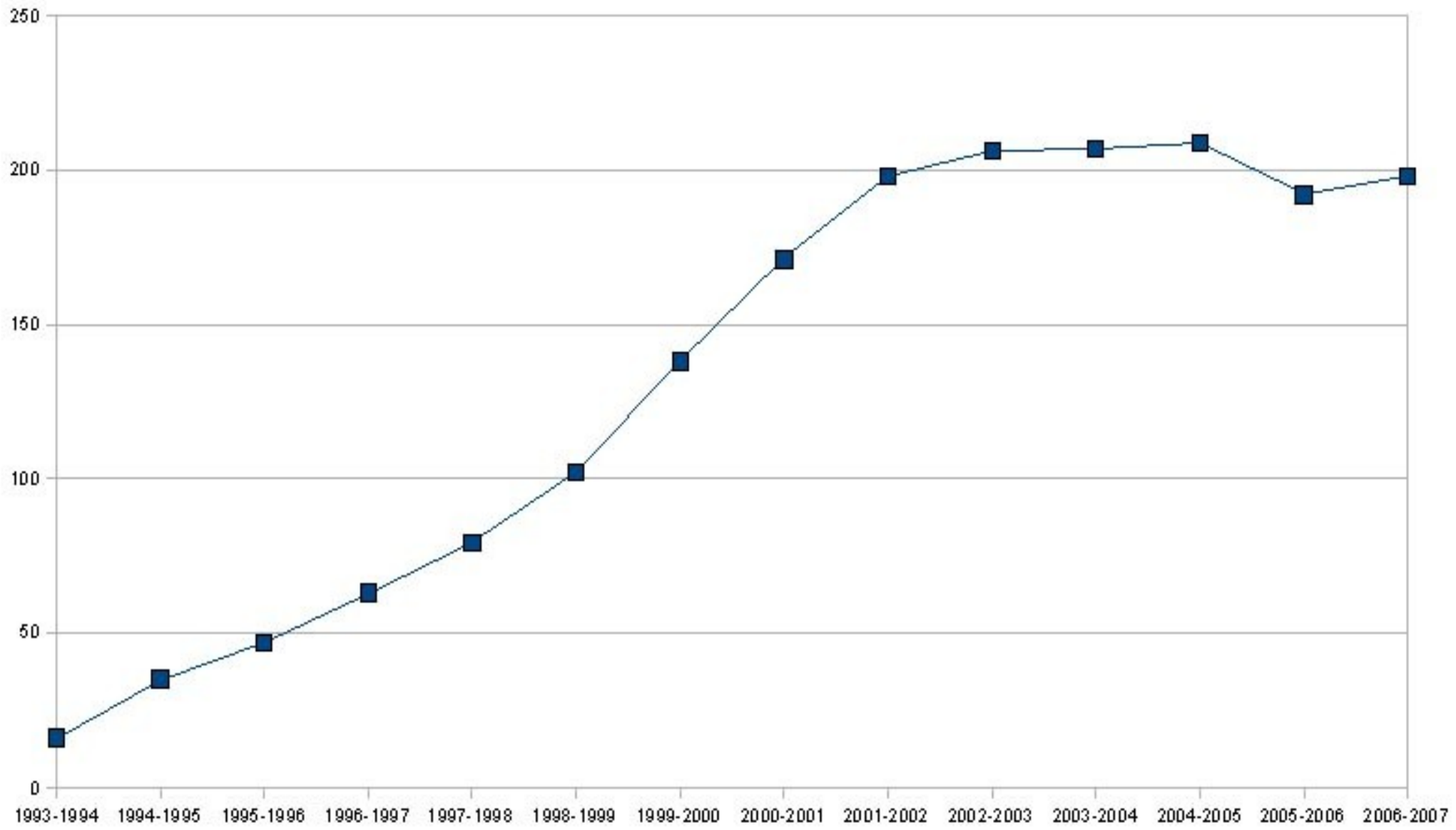
*(*liberos: committers who conribute both new code and bug fixes)*



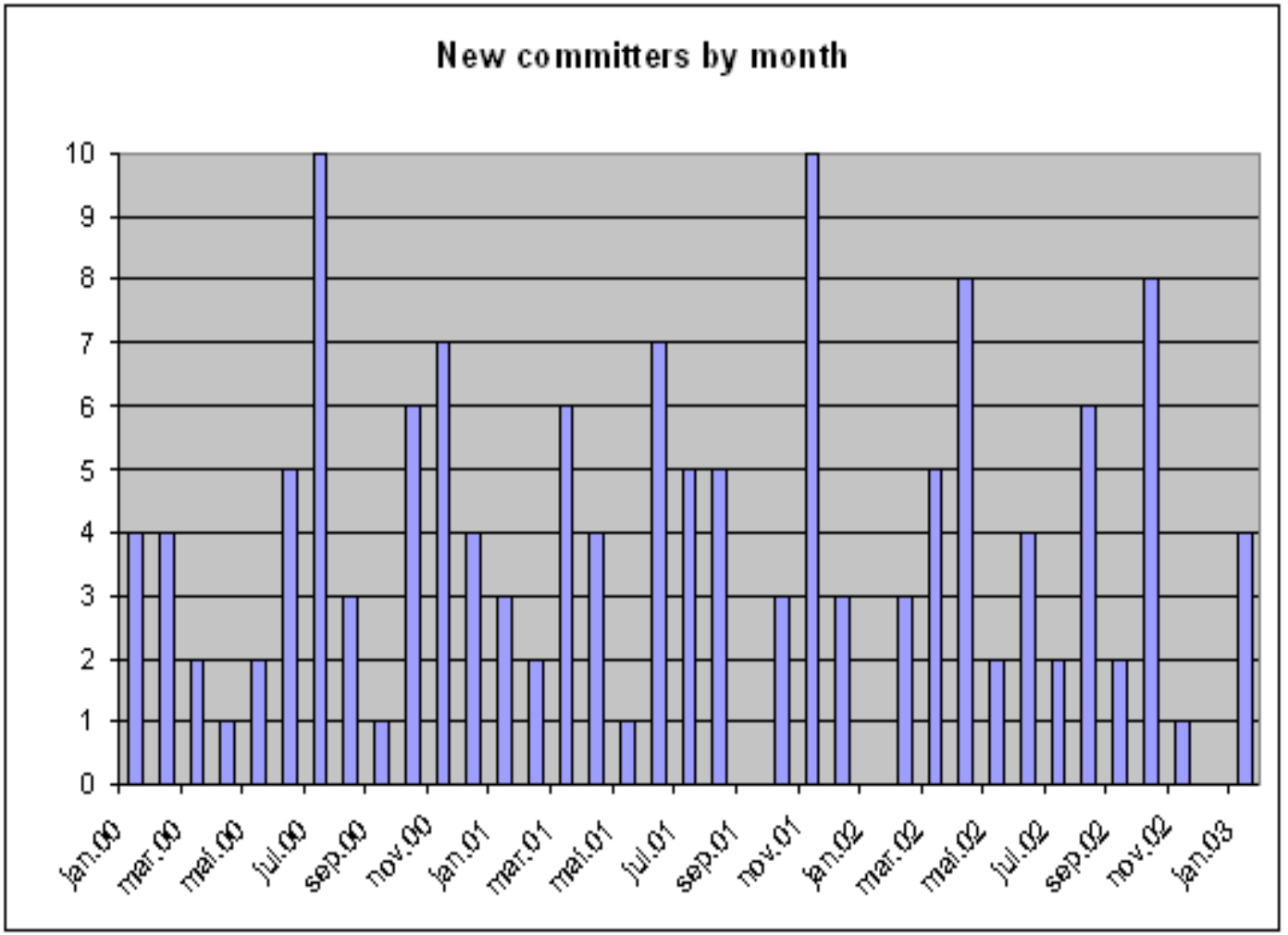
*Liberos: committers who contribute both bug-fixes and new functionality

Is FOSS a clique?

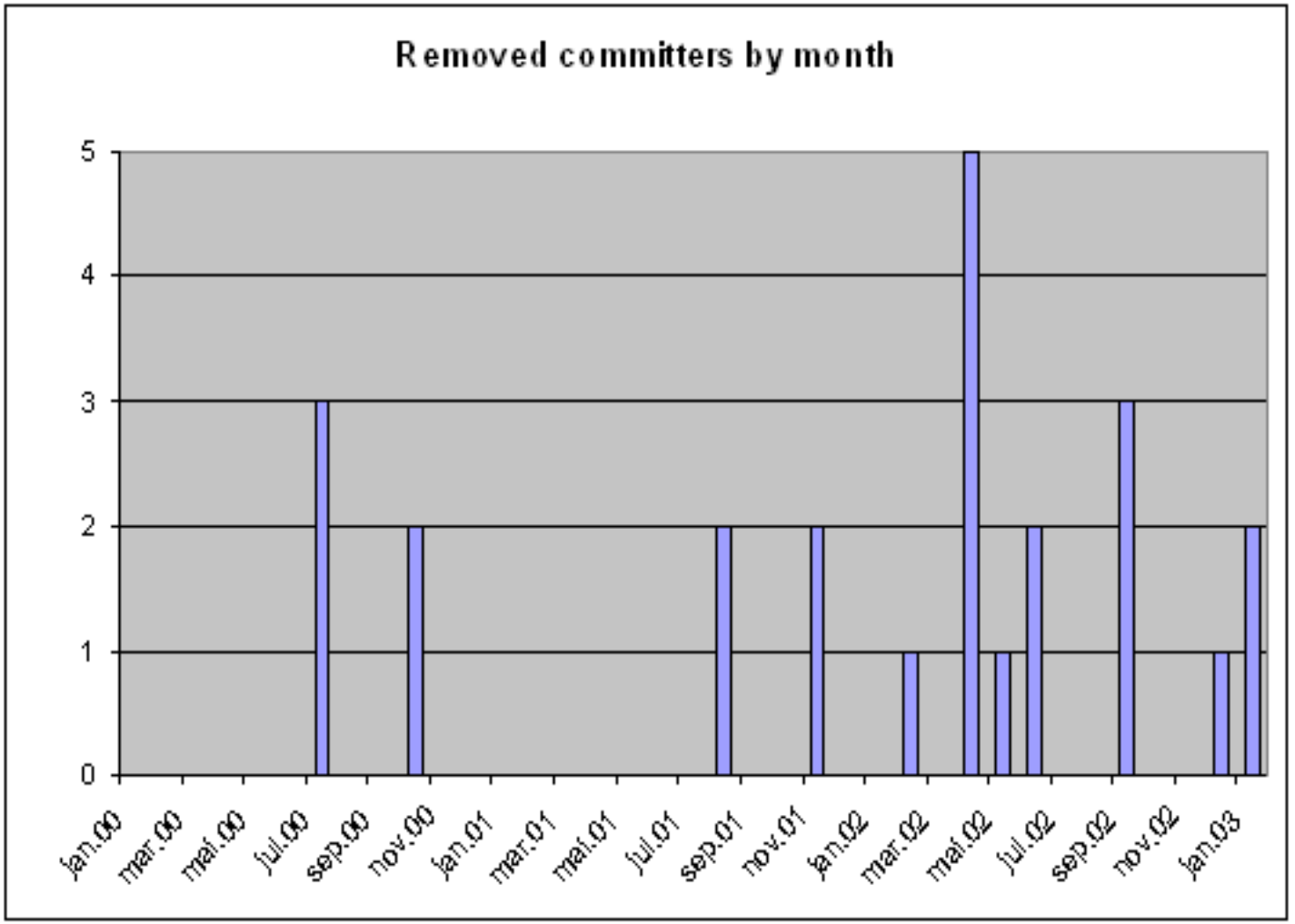
Committers (*current branch, src*)



More and more committers join...



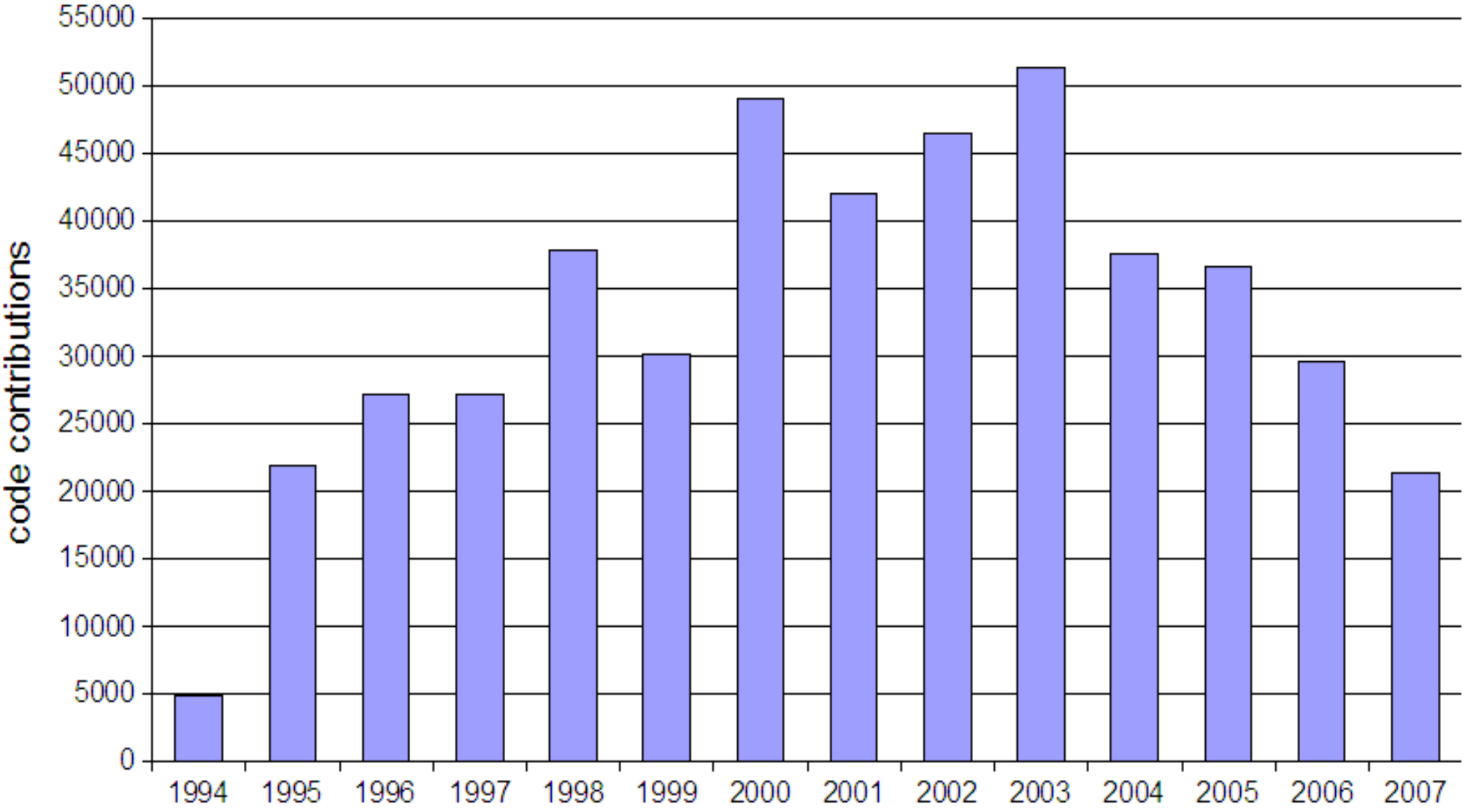
...but they seldom leave



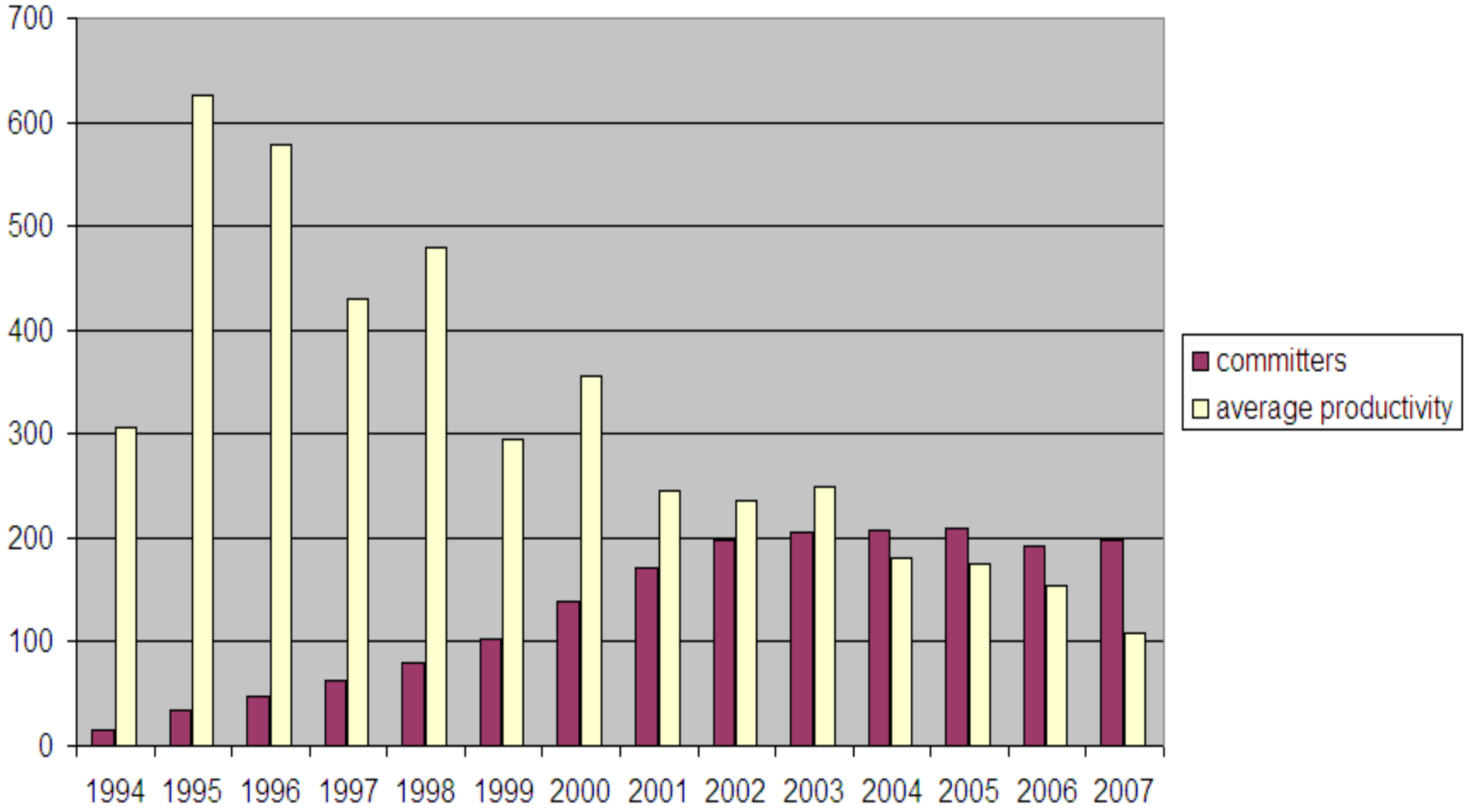
Jan 2000 – Jan 2003
4 new committers per month
142 added, 24 removed

Code contributions

(*current branch, src*)

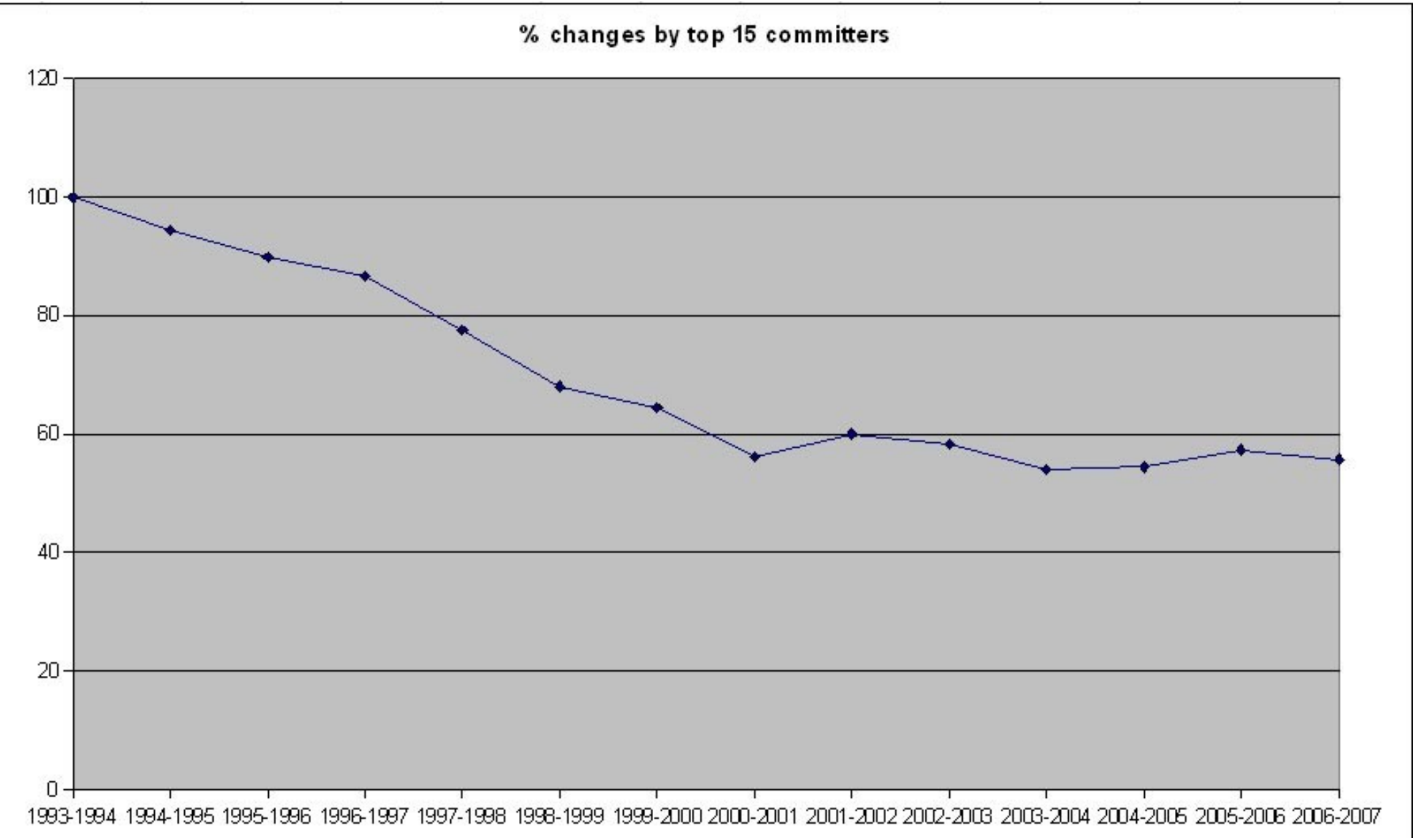


Average productivity
(code contributions per committer)

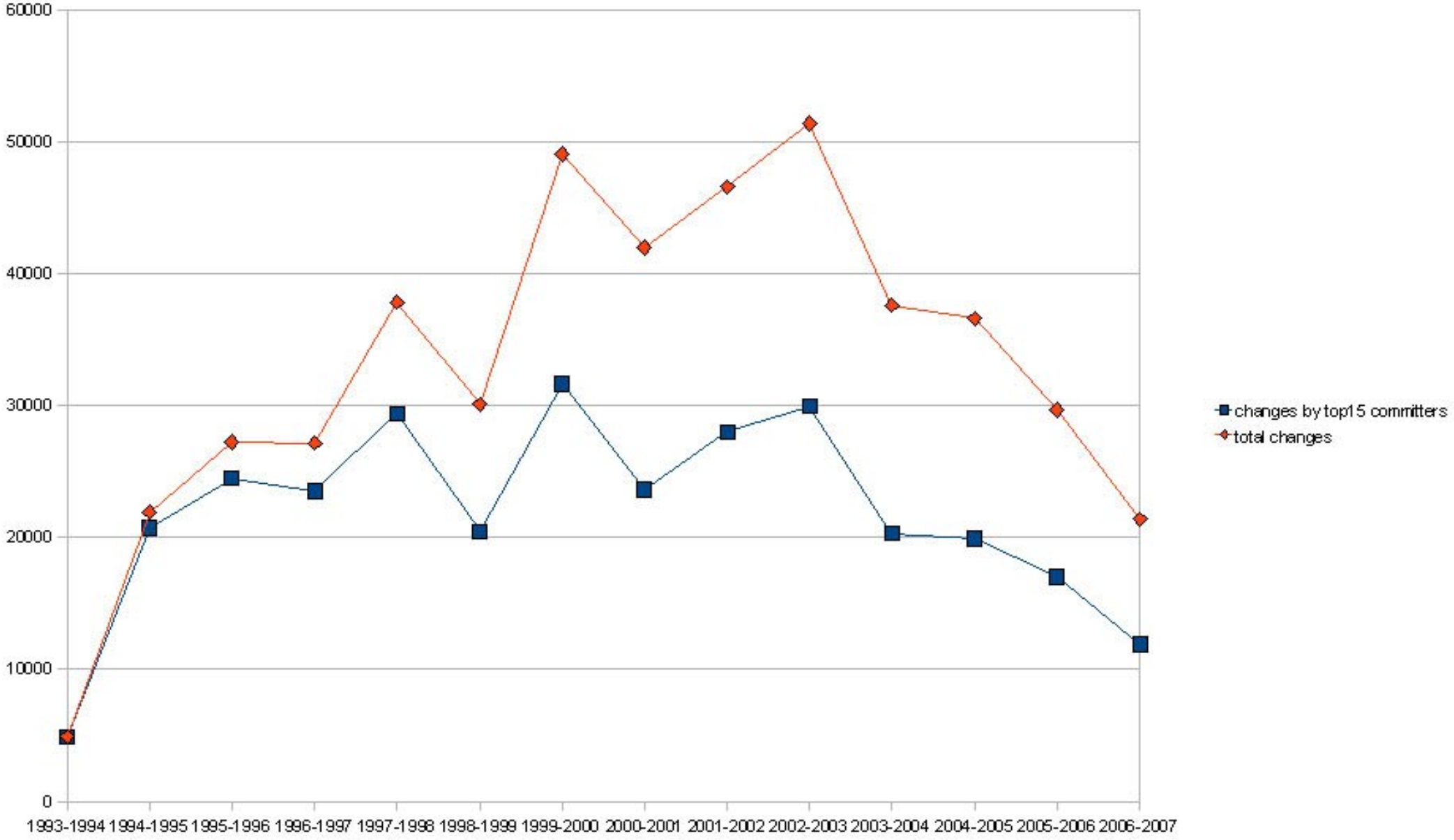


Top 15 committers (*current branch, src*)

Year	*% changes by top 15 committers*	*No of committers required for 80% of changes*
1993-1994	99.9	6
1994-1995	94.4	8
1995-1996	89.8	10
1996-1997	86.6	12
1997-1998	77.6	17
1998-1999	68	23
1999-2000	64.5	29
2000-2001	56.2	37
2001-2002	60	35
2002-2003	58.2	39
2003-2004	54	37
2004-2005	54.4	42
2005-2006	57.3	38
2006-2007	55.7	41
total(1993-2003)	*53.6*	*54*


% changes by top 15 committers
120
100
80
60
40
20
0
1993-1994
1994-1995
1995-1996
1996-1997
1997-1998
1998-1999
1999-2000
2000-2001
2001-2002
2002-2003
2003-2004
2004-2005
2005-2006
2006-2007

Code contributions by top 15 committers *Vs.* all contributions



Comment on Brooks' Law

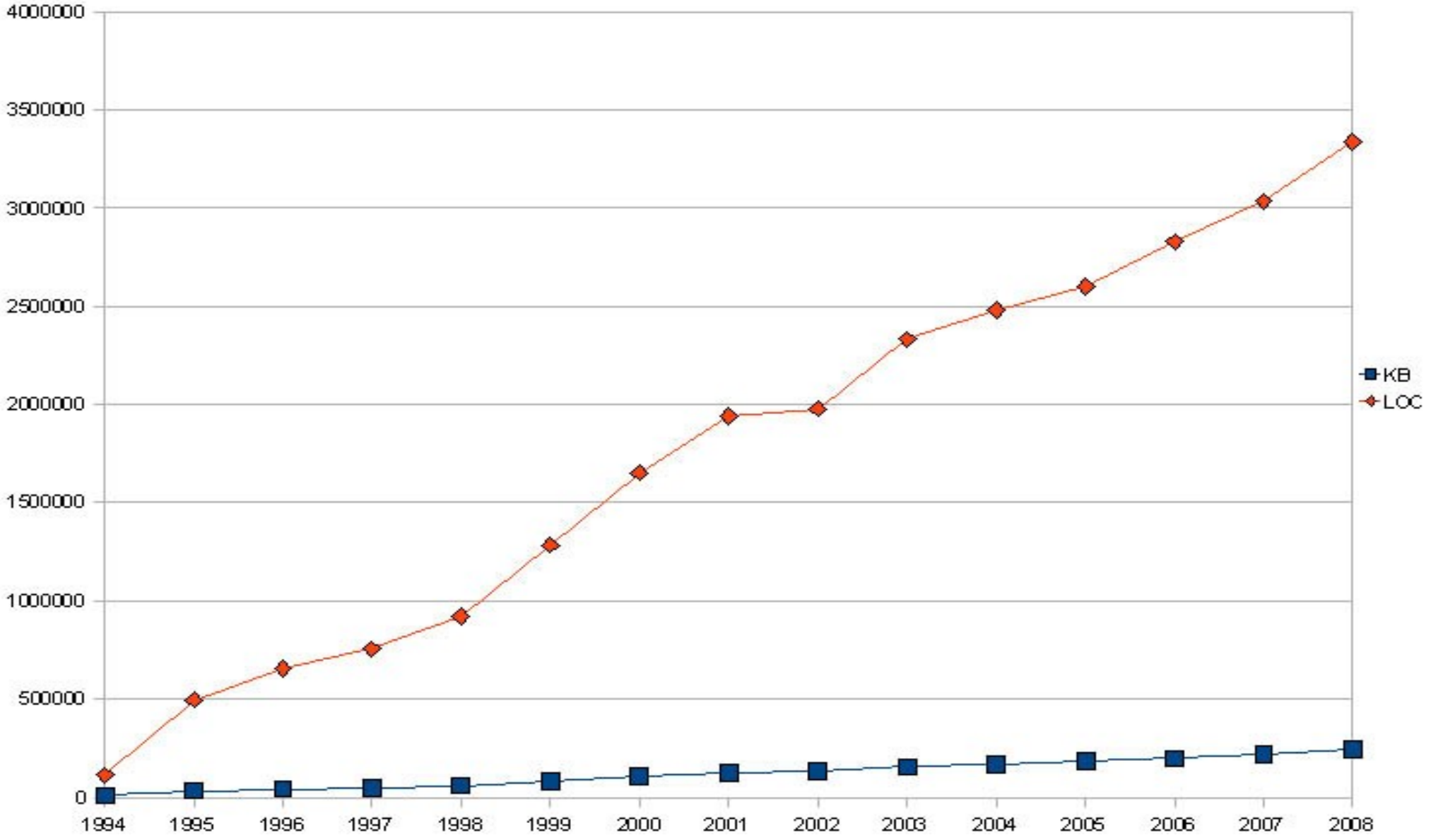
The increase of developers has not affected negatively the productivity of high-contribution participants.

What accounts for this? The two-tier structure and modularity? Not sufficiently (as dependencies among modules rise in line with committers, $\Delta R^2 = .438$, $F(1,278) = 217$, $p<.001$).

Do they spend more time as the project unfolds?

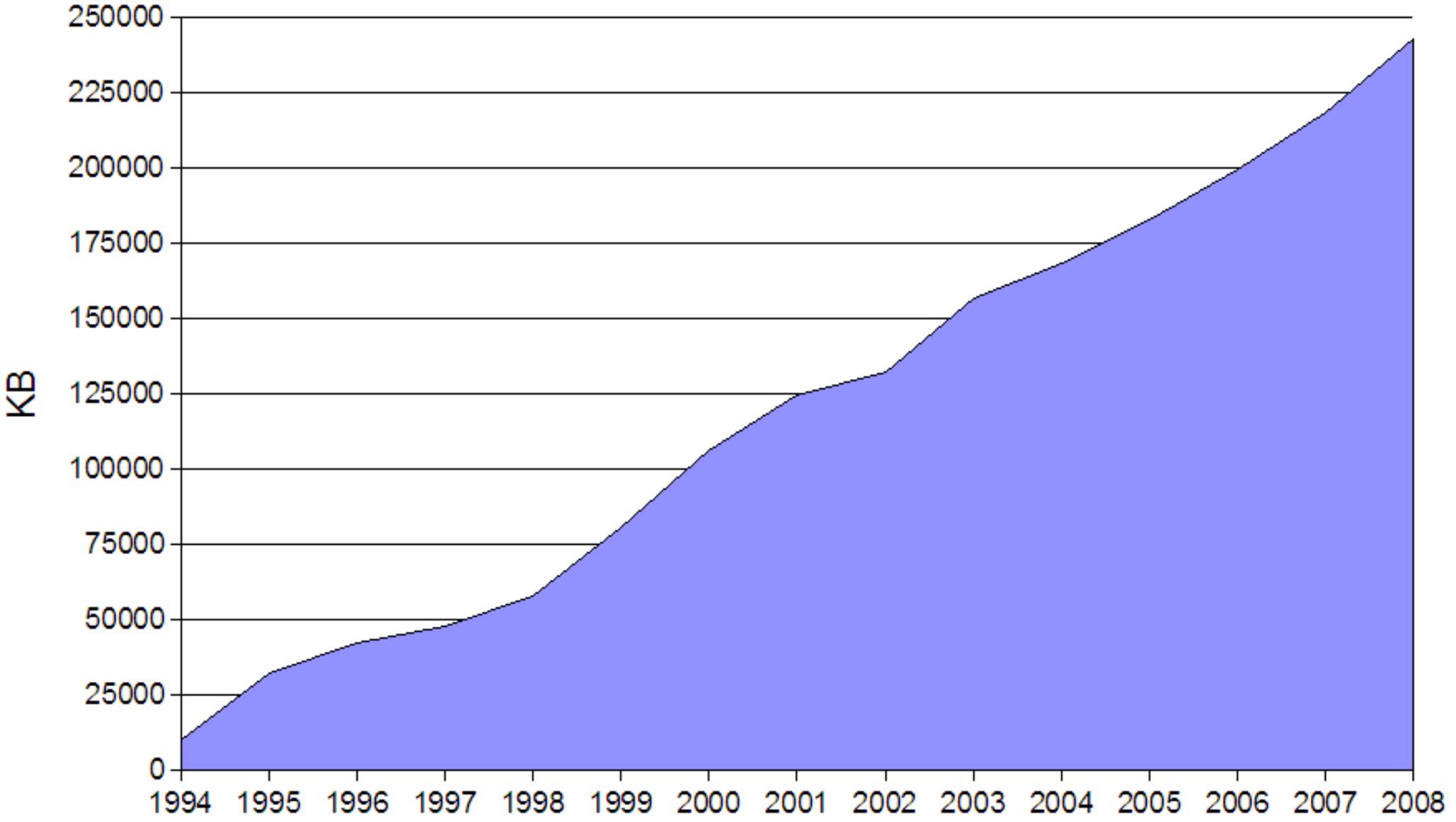
Scale: codebase evolution

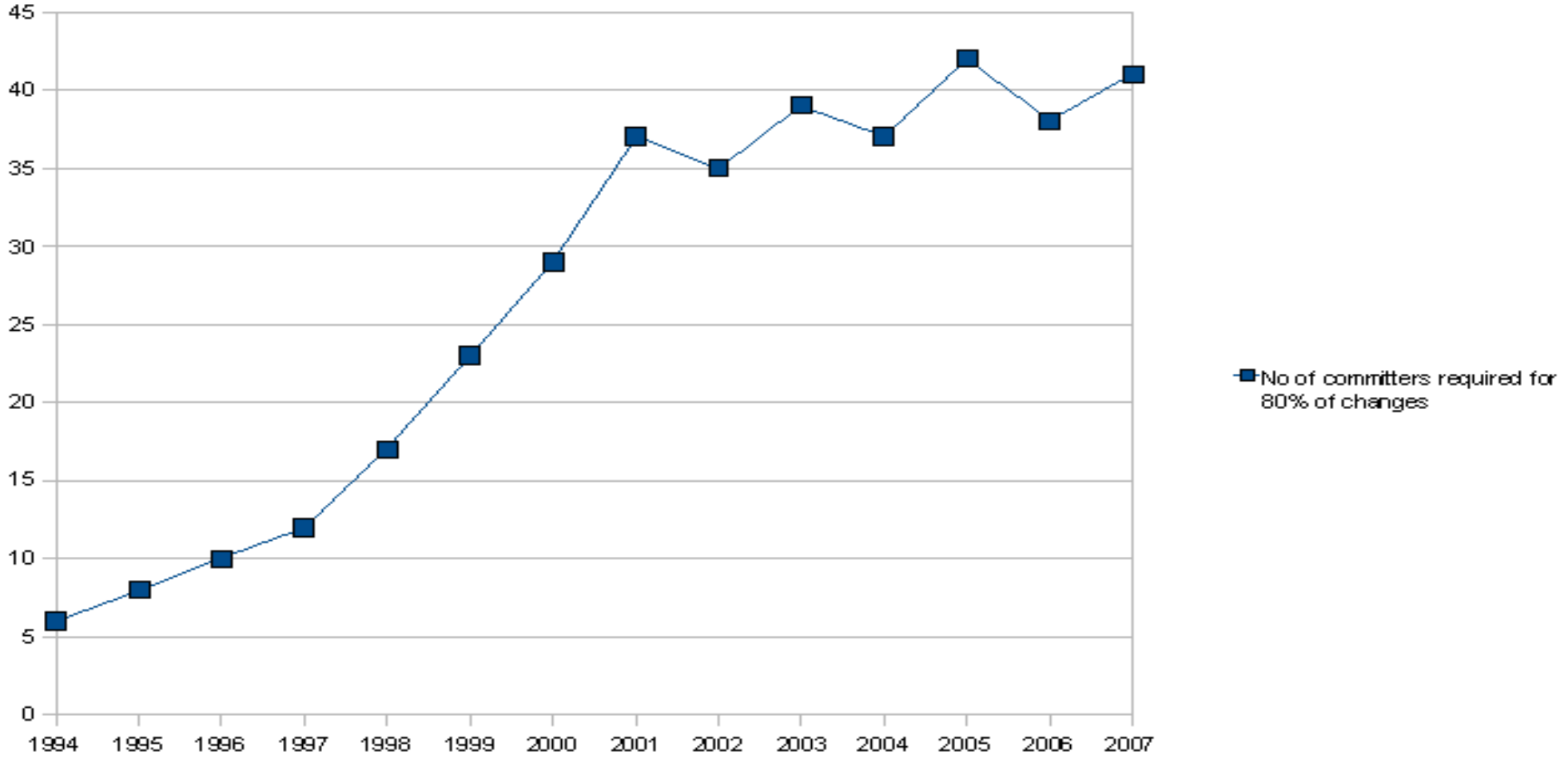
(*current branch, src*)



Scale: codebase evolution

(*current branch, src*)

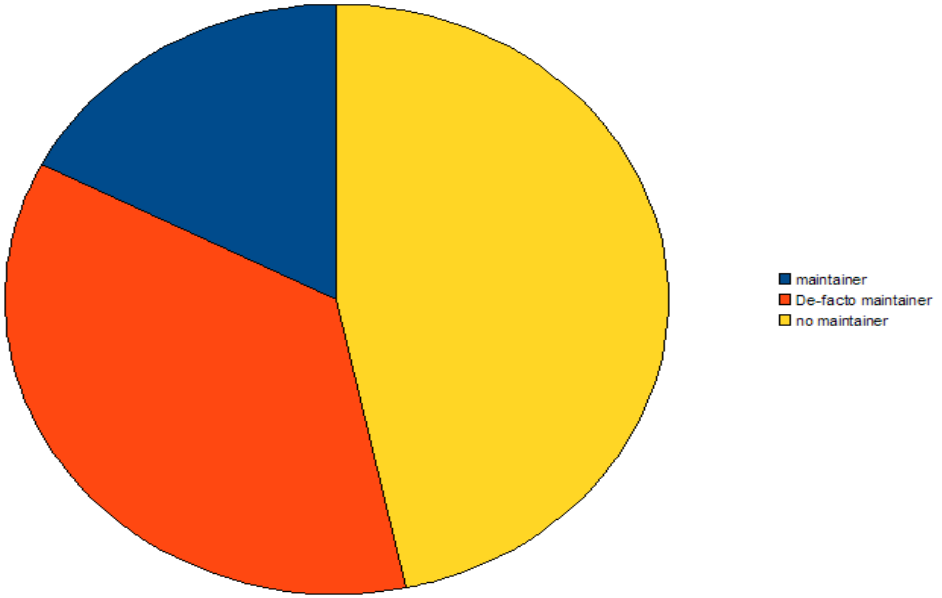




The greater the size of the technology under development the more developers will be required to produce 80% of new functionality

Module maintainership

(Current branch, Feb. 2003)

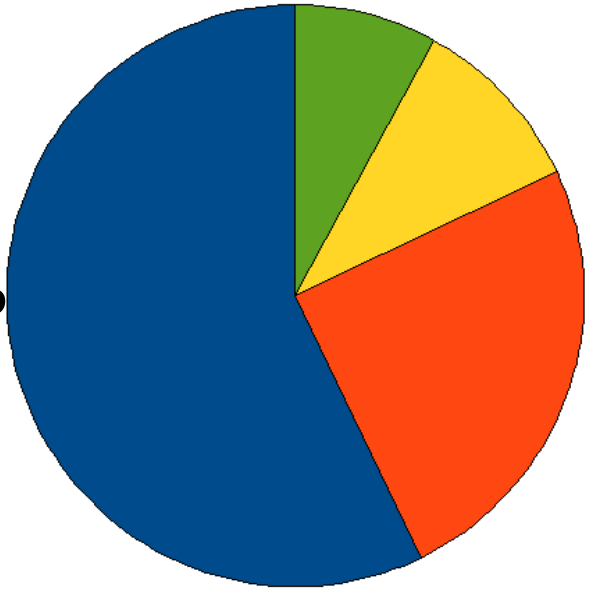


- 18% of modules (125 of 716) have a maintainer
- 82% of modules (591 of 716) are maintainer-less, but 257 of them have a *de-facto** maintainer
- *So,* 53% of modules (382 of 716) have a designated or de-facto maintainer

*de-facto maintainer: >50% commits during last 12 months

Code ownership (Current branch, 2007)

- files with 1 committer: 31,831 = 47,4%
- files with 2 committers: 13,825 = 20,6%
- files with 3 committers: 5,577 = 8,3%
- files with >9 committers: 4,445 = 6,6%



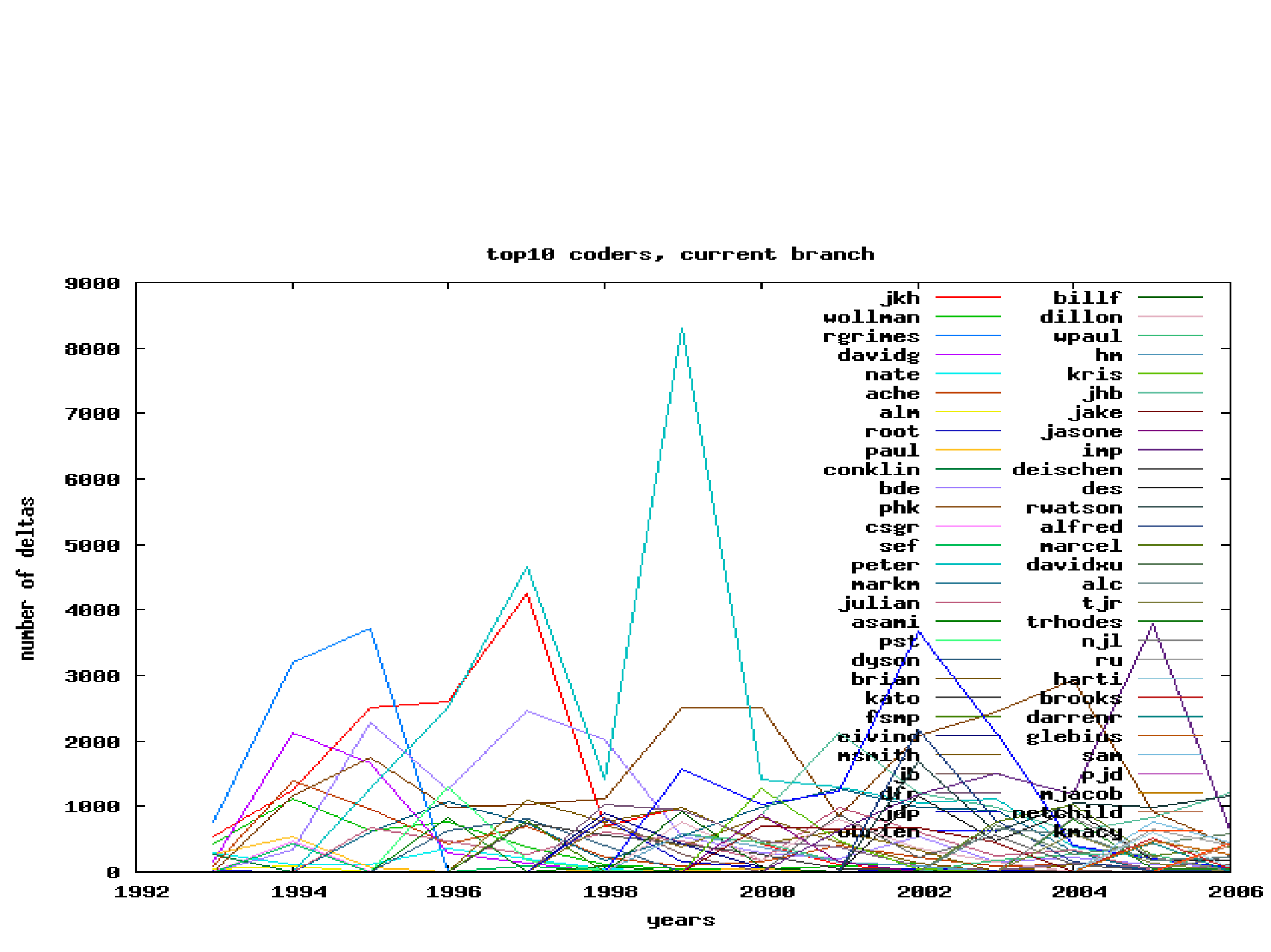
Code checked-in by a committer can be easily modified by others - *Maintainers* are recognised as experts on certain areas of the code, for which they are responsible - Responsibility in this case should not be confused with a mode of ownership configured around the right to exclude.

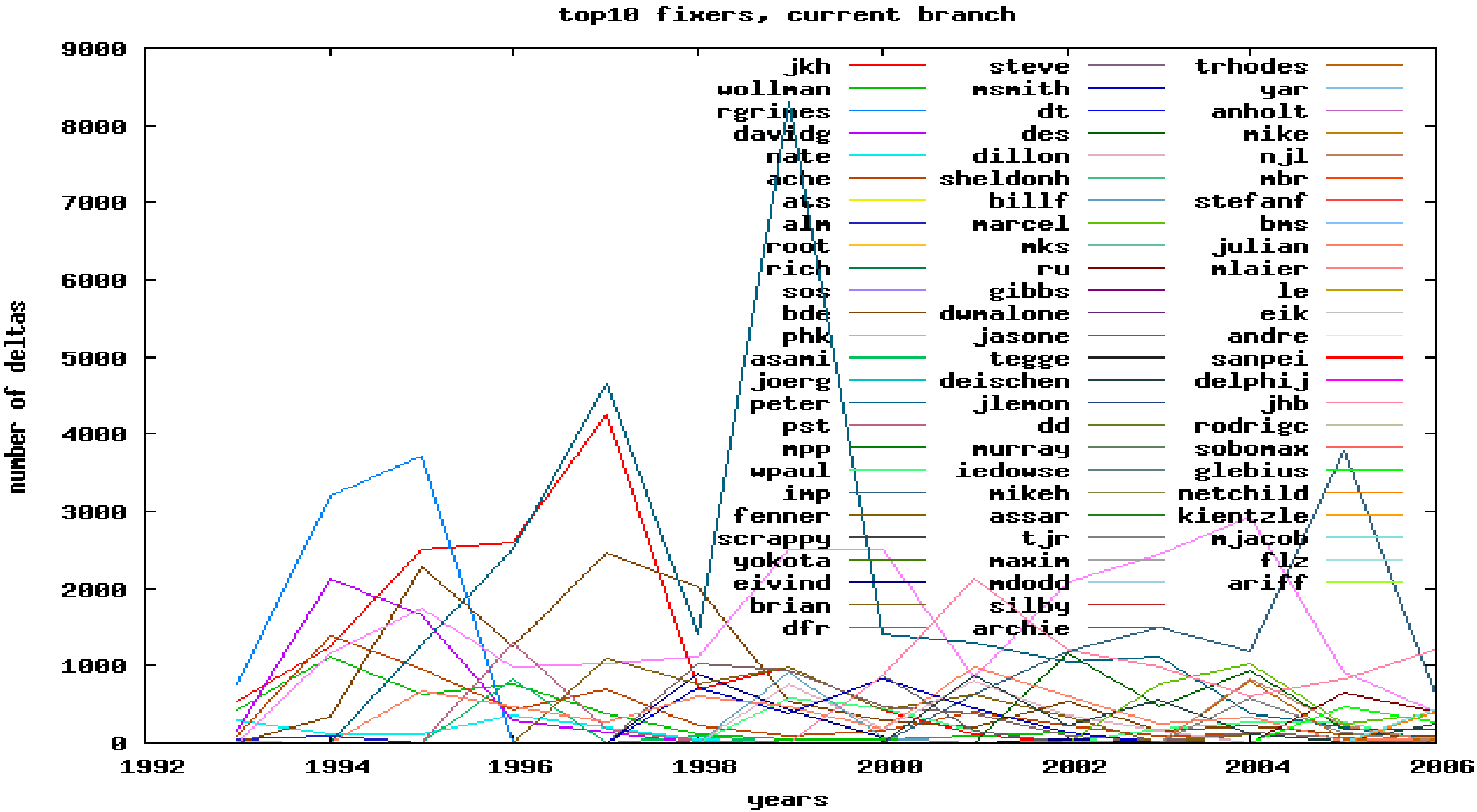
Code leadership

In 13 yrs, of the 58 who populated the ranks of the ten most productive committers:

one with 13 yrs (phk: Poul-Henning Kamp)
one with 11 yrs (peter: Peter Wemm)
one with 10 yrs (markm: Mark Murray)

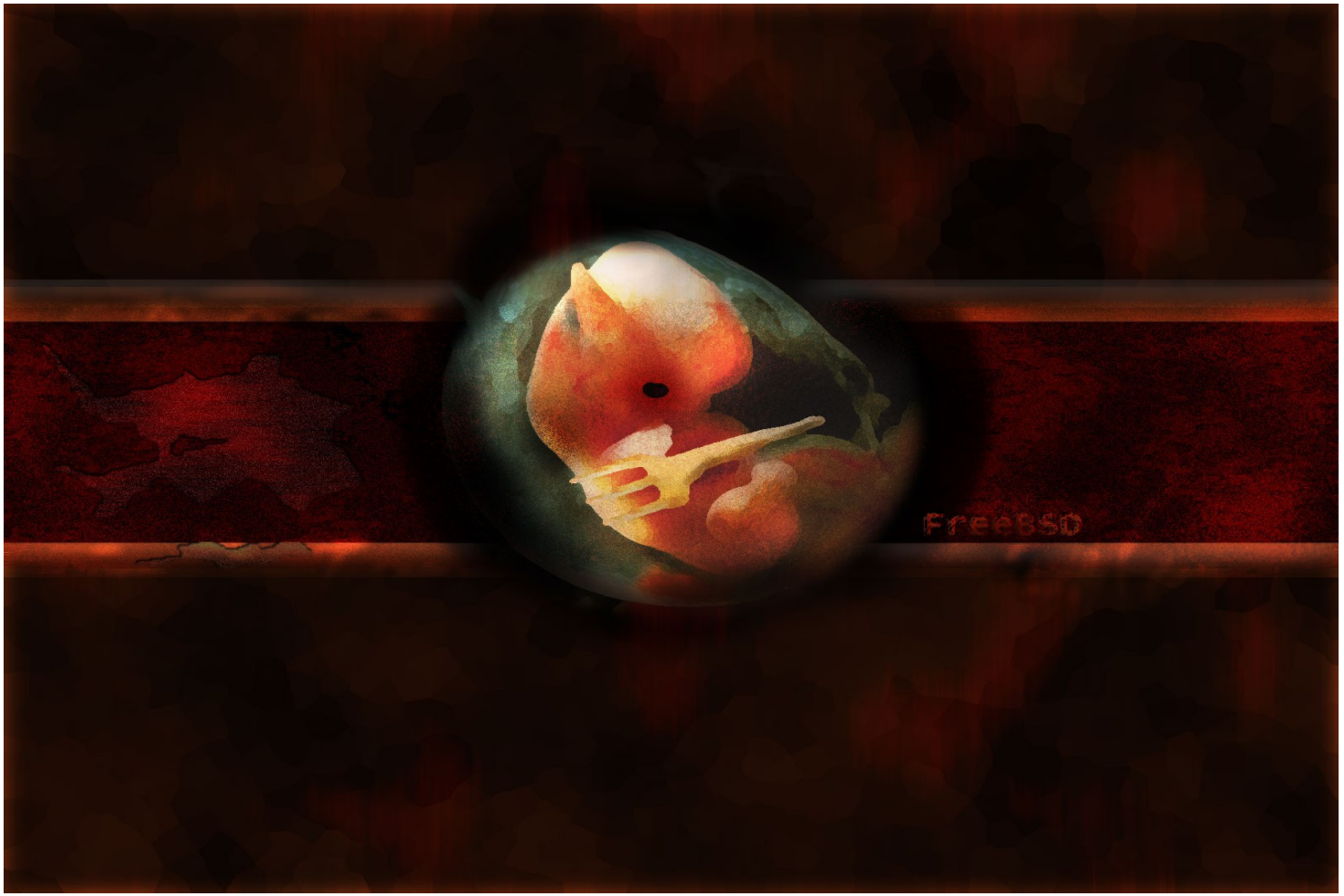
average 3.5 yrs


top10 coders, current branch
number of deltas
years
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
1992
1994
1996
1998
2000
2002
2004
2006
jkh
wollman
rgrimes
davidg
nate
ache
alm
root
paul
conklin
bde
phk
csgr
sef
peter
markm
julian
asani
pst
dyson
brian
kato
fsmp
eivind
msmith
jb
dfr
jdp
obrien
billf
dillon
wpaul
hm
kris
jhb
jake
jasone
imp
deischen
des
rwatson
alfred
marcel
davidxu
alc
tjr
trhodes
njl
ru
barti
brooks
darrenr
glebius
sam
pjd
mjacob
netchild
knacy


top10 fixers, current branch
number of deltas
years
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1992
1994
1996
1998
2000
2002
2004
2006
jkh
wollman
rgrimes
davidg
nate
ache
ats
alm
root
rich
sos
bde
phk
asami
joerg
peter
pst
mpp
wpaul
imp
fenner
scrappy
yokota
eivind
brian
dfr
steve
msmith
dt
des
dillon
sheldonh
billf
marcel
mks
ru
gibbs
dwmalone
jasone
tegge
deischen
jlemon
dd
murray
iedowse
mikeh
assar
tjr
maxim
mdodd
silby
archie
trhodes
yar
anholt
mike
njl
mbr
stefanf
bms
julian
mlaier
le
eik
andre
sanpei
delphij
jhb
rodrigc
sobomax
glebius
netchild
kientzle
mjacob
flz
ariff

Code leadership is distributed across different groups of developers over time.

FreeBSD does not depend on a *code god,* but is driven by different groups of developers over time: the project 'regenerates' itself.



the end

Bibliography

T.T. Dinh-Trong and J.M. Bieman, 2005. The FreeBSD Project: A Replication Case Study of Open Source Development, *IEEE Transactions on Software Engineering*, Vol. 31, No. 6, June, pp. 481-494.

N. Jørgensen, 2001. Putting it All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project, *Information Systems Journal*, Vol. 11, No. 4, pp. 321-336, at http://webhotel.ruc.dk/nielsj//research/publications/freebsd.pdf.

N. Jørgensen, 2005. Incremental and Decentralized Integration in FreeBSD, in J. Feller, B. Fitzgerald, S. Hissam & K. R. Lakhani (Eds.) *Perspectives on Free and Open Source Software*, MIT Press, at http://mitpress.mit.edu/books/chapters/0262062461chap12.pdf

N. Saers, 2005. *A project model for the FreeBSD Project*, FreeBSD Project, at http://niklas.saers.com/thesis/thesis.html.

D. Spinellis, 2006. Global software development in the FreeBSD project, in P. Kruchten et al. (eds) *International Workshop on Global Software Development for the Practitioner*, ACM Press, pp. 73–79, at http://www.spinellis.gr/pubs/conf/2006-GSD-FreeBSD/html/GSD-FreeBSD.html

R. Watson, 2006. How the FreeBSD Project Works, *Proceedings of EuroBSDCon*, at http://www.watson.org/~robert/freebsd/2006eurobsdcon/eurobsdcon2006-howfreebsdworks.pdf.

Credits

Ludo Gorzeman *data-mining*