Engineering Software Analytics Studies

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
http://www.spinellis.gr/
@CoolSWEng

curiosity

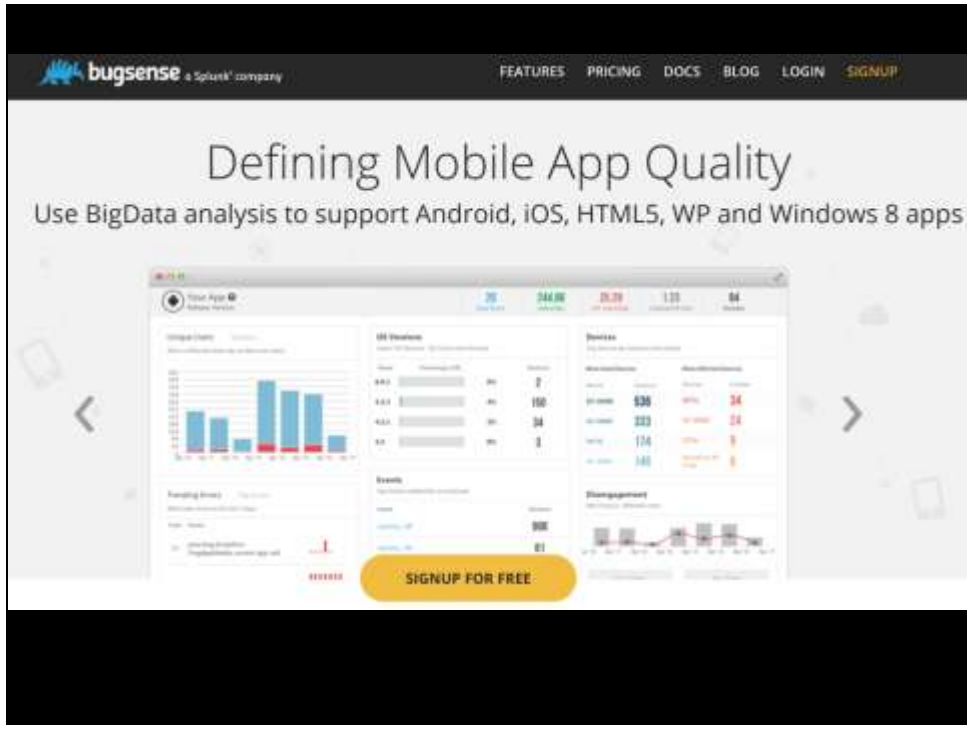
serendipity

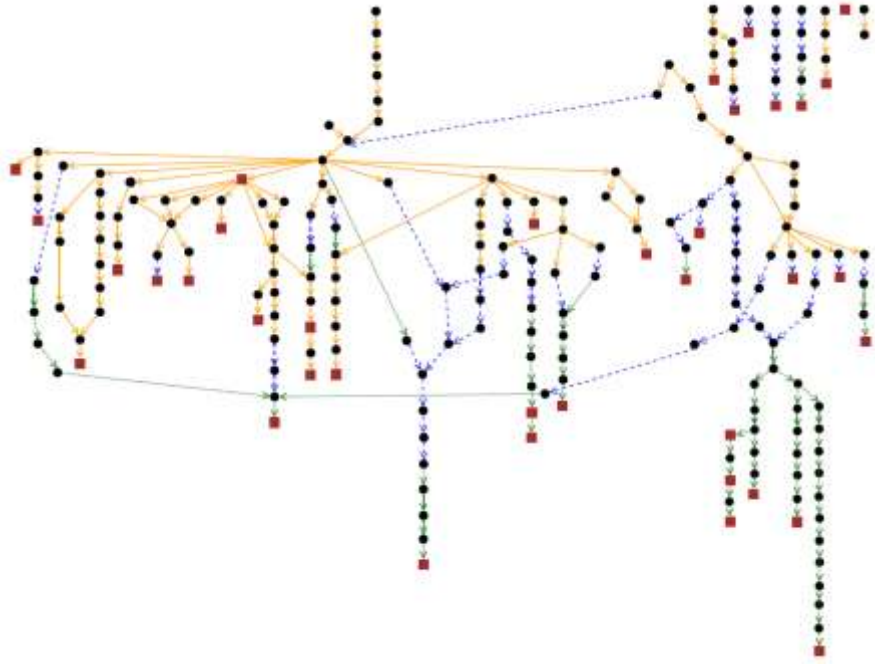
Outline

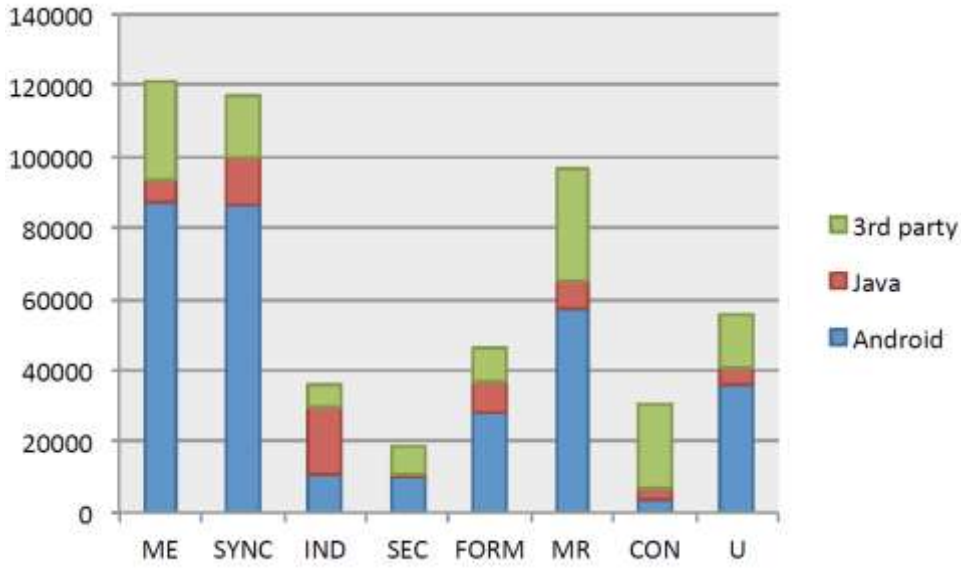
- Data gathering
- Processing
  - Methods
  - Advice
- Performance

Data gathering


Alliances


bugsense a Splunk company
FEATURES
PRICING
DOCS
BLOG
LOGIN
SIGNUP
Defining Mobile App Quality
Use BigData analysis to support Android, iOS, HTML5, WP and Windows 8 apps
SIGNUP FOR FREE




140000
120000
100000
80000
60000
40000
20000
0
ME
SYNC
IND
SEC
FORM
MR
CON
U
3rd party
Java
Android

The Journal of Systems and Software 85 (2012) 666–682

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Organizational adoption of open source software

Diomidis Spinellis[a,*], Vaggelis Giannikas[b]

[a] Department Management Science and Technology, Athens University of Economics and Business, Patission 76, GR-104 34 Athens, Greece
[b] Institute for Manufacturing, University of Cambridge, 17 Charles Babbage Road, Cambridge CB3 0FS, United Kingdom

ARTICLE INFO

Article history:
Received 6 February 2011
Received in revised form 19 August 2011
Accepted 15 September 2011
Available online 22 September 2011

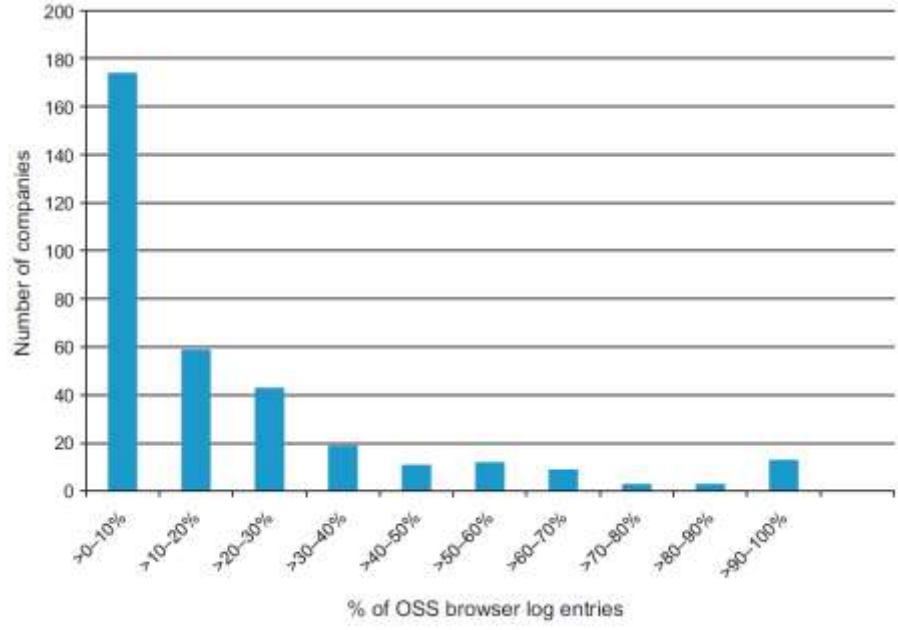
Keywords:
Open source software
Technology adoption
Industrial practice

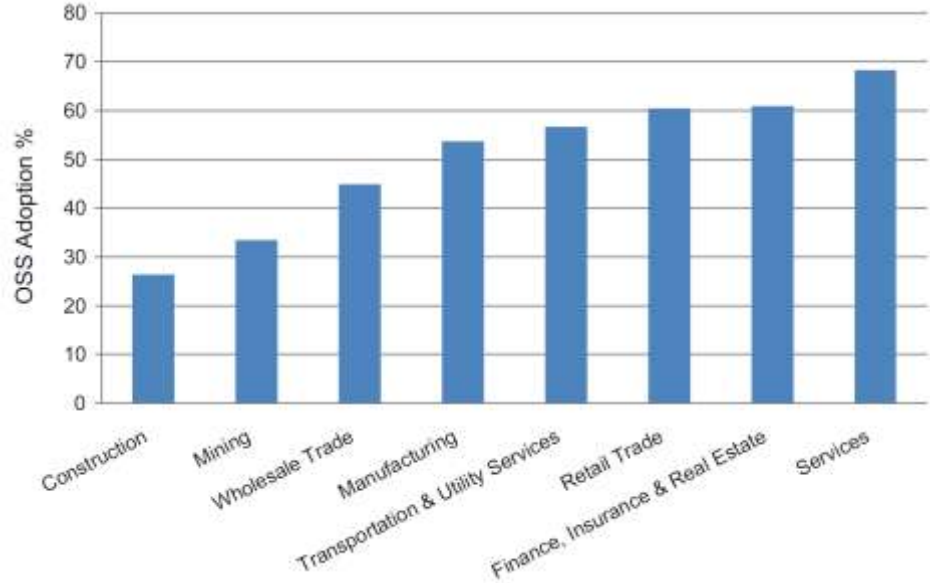
ABSTRACT

Organizations and individuals can use open source software (OSS) for free, they can study its internal workings, and they can even fix it or modify it to make it suit their particular needs. These attributes make OSS an enticing technological choice for a company. Unfortunately, because most enterprises view technology as a proprietary differentiating element of their operation, little is known about the extent of OSS adoption in industry and the key drivers behind adoption decisions. In this article we examine factors and behaviors associated with the adoption of OSS and provide empirical findings through data gathered from the US Fortune-1000 companies. The data come from each company's web browsing and serving activities, gathered by sifting through more than 278 million web server log records and analyzing the results of thousands of network probes. We show that the adoption of OSS in large US companies is significant and is increasing over time through a low-churn transition, advancing from applications to platforms. Its adoption is a pragmatic decision influenced by network effects. It is likelier in larger organizations and those with many less productive employees, and is associated with IT and knowledge-intensive work and operating efficiencies.

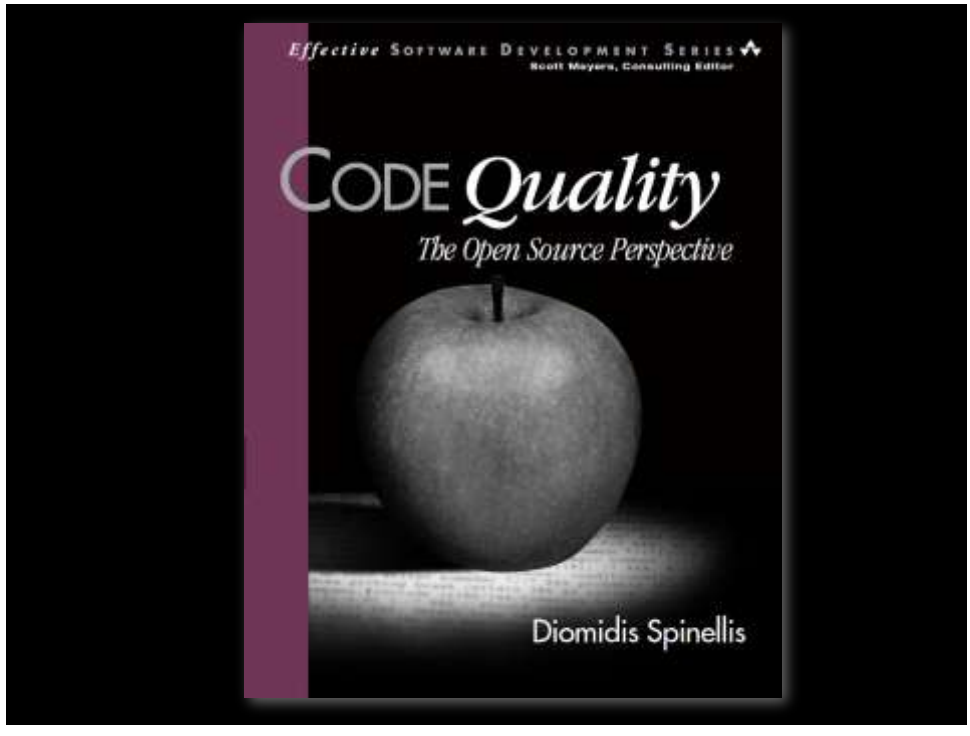
1. Introduction

allow the creation of derived works provided they respect the cre-

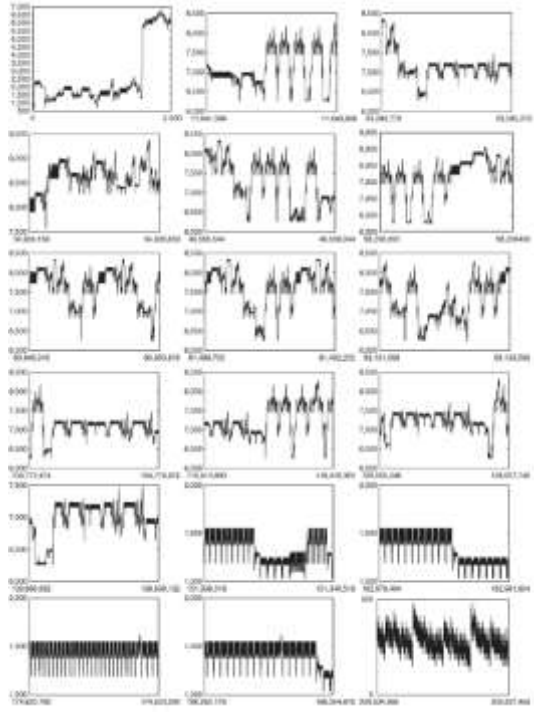

200
180
160
140
120
100
80
60
40
20
0
Number of companies
>0–10%
>10–20%
>20–30%
>30–40%
>40–50%
>50–60%
>60–70%
>70–80%
>80–90%
>90–100%
% of OSS browser log entries

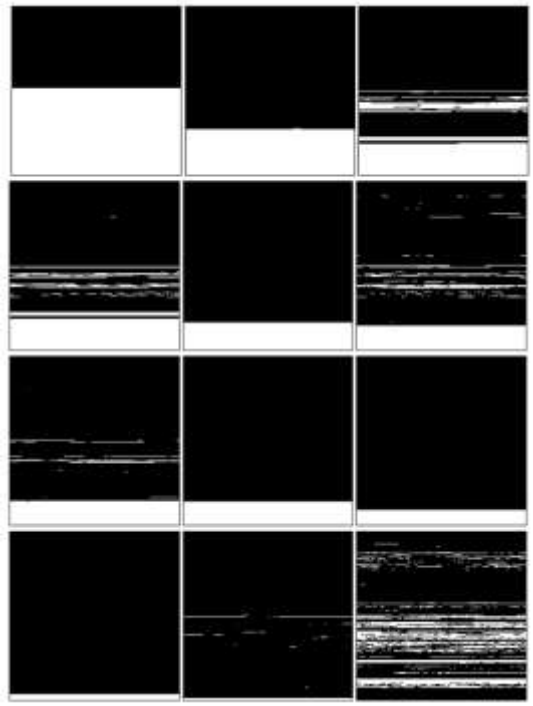

80
70
60
50
40
30
20
10
0
OSS Adoption %
Construction
Mining
Wholesale Trade
Manufacturing
Transportation & Utility Services
Retail Trade
Finance, Insurance & Real Estate
Services


Instrumentation


Effective Software Development Series
Scott Meyers, Consulting Editor
Code Quality
The Open Source Perspective
Diomidis Spinellis

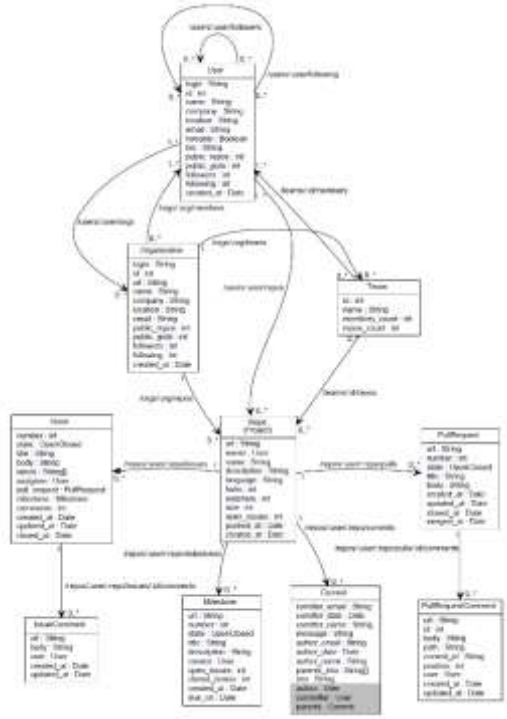
```
_MCOUNT_DECL(frompc, selfpc)   /* _mcount; may be static, inline, etc */
        u_long frompc, selfpc;
{
        struct gmonparam *p;
        void *stack = &frompc;

        p = &_gmonparam;
        /*
         * check that we are profiling
         * and that we aren't recursively invoked.
         */
        if (p->state != GMON_PROF_ON)
                return;
        p->state = GMON_PROF_BUSY;
        /*
         * check that frompcindex is a reasonable pc value.
         * for example: signal catchers get called from the stack,
         *              not from text space.  too bad.
         */
        frompc -= p->lowpc;
        if (frompc <= p->textsize)
                write(fd, &stack, sizeof(stack));
        p->state = GMON_PROF_ON;
        return;
}

_MCOUNT
```

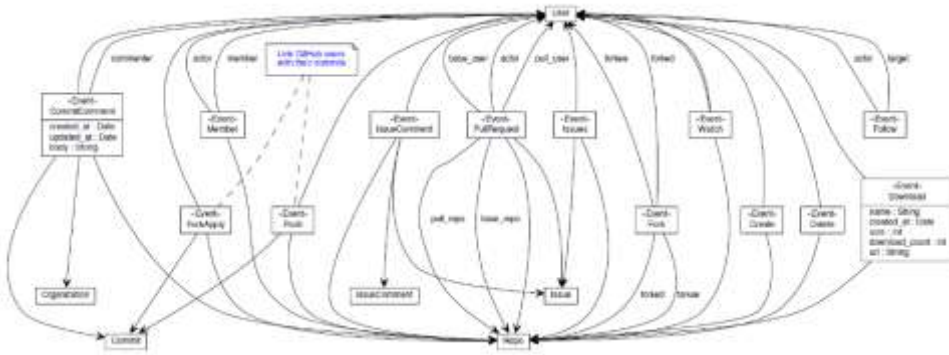




```
4266  utimensat(AT_FDCWD,
"boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", {{1336157319,
67677176}, {1090801932, 0}}, 0) = 0
4266  chown("boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", 0, 0)
= 0
4266  chmod("boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", 0644)
= 0
4266  open("boost_1_44_0/libs/bind/test/bind_stdcall_test.cpp",
O_WRONLY|O_CREAT|O_EXCL, 0600) = 4
4266  write(4, ""..., 2789)             = 2789
4266  close(4)                         = 0
4266  utimensat(AT_FDCWD,
"boost_1_44_0/libs/bind/test/bind_stdcall_test.cpp", {{1336157319,
67677176}, {1090801932, 0}}, 0) = 0
4266  chown("boost_1_44_0/libs/bind/test/bind_stdcall_test.cpp", 0, 0) =
0
4266  chmod("boost_1_44_0/libs/bind/test/bind_stdcall_test.cpp", 0644) =
0
4266  open("boost_1_44_0/libs/bind/test/bind_test.cpp",
O_WRONLY|O_CREAT|O_EXCL, 0600) = 4
4266  read(3,  <unfinished ...>
```


Recursion





GHTorrent: Github's Data from a Firehose

Georgios Gousios and Dionisis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece
{gousiosg,dds}@aueb.gr

***Abstract*—A common requirement of many empirical software engineering studies is the acquisition and curation of data from software repositories. During the last few years, GitHub has emerged as a popular project hosting, mirroring and collaboration platform. GitHub provides an extensive REST API, which enables researchers to retrieve both the commits to the projects' repositories and events generated through user actions on project resources. GHTorrent aims to create a scalable off line mirror of GitHub's event streams and persistent data, and offer it to the research community as a service. In this paper, we present the project's design and initial implementation and demonstrate how the provided datasets can be queried and processed.**

***Keywords*-dataset; repository; GitHub; commits; events**

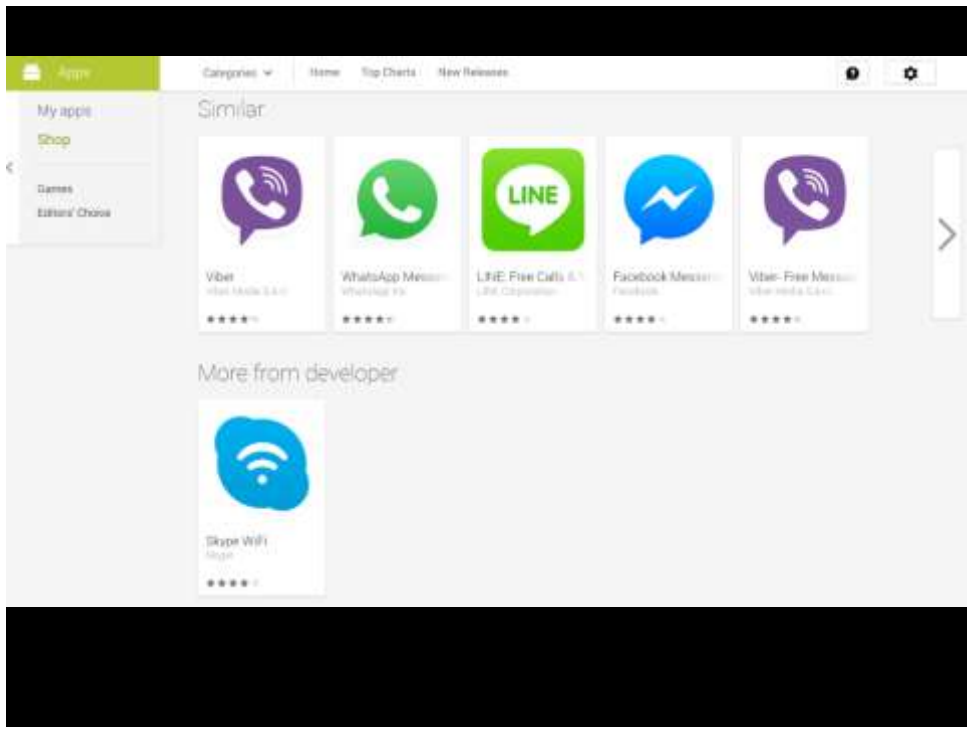
I. Introduction and Motivation

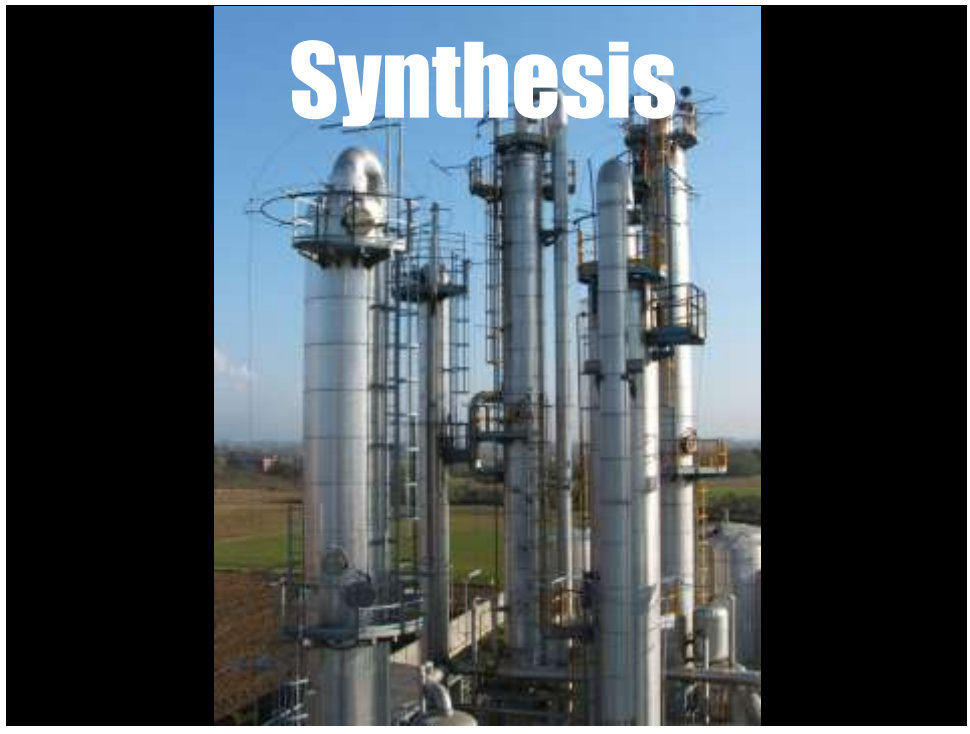
When conducting empirical research with data from software repositories, a typical step involves the downloading of the project data to be analysed. Obtaining data from open source software (OSS) project repositories is a tedious exercise lacking scientific value, while the obtained datasets are often non-homogeneous which makes further analysis also facilitates project contributions from non-team members through features such as forking, which creates a private copy of a *git* repository, and pull requests, which organize a series of commits to a forked repository into a mergeable patch for the source repository. Moreover, GitHub offers the usual gamut of software forge facilities, such as a bug tracker, a wiki, and developer communication tools. What makes GitHub particularly attractive for the repository mining community, is that it provides access to its internal data stores through an extensive REST [2] API,[1] which researchers can use to access a rich collection of unified and versioned process and product data.

Although GitHub's API is well designed and documented, researchers wishing to use it in order to analyze its data will face a number of challenges.

- GitHub's data is huge (on the terabyte scale, according to our estimation) and an API rate limit of five thousand requests per hour makes the complete download of the data impossible.
- The overall schema of GitHub's data is not docu-

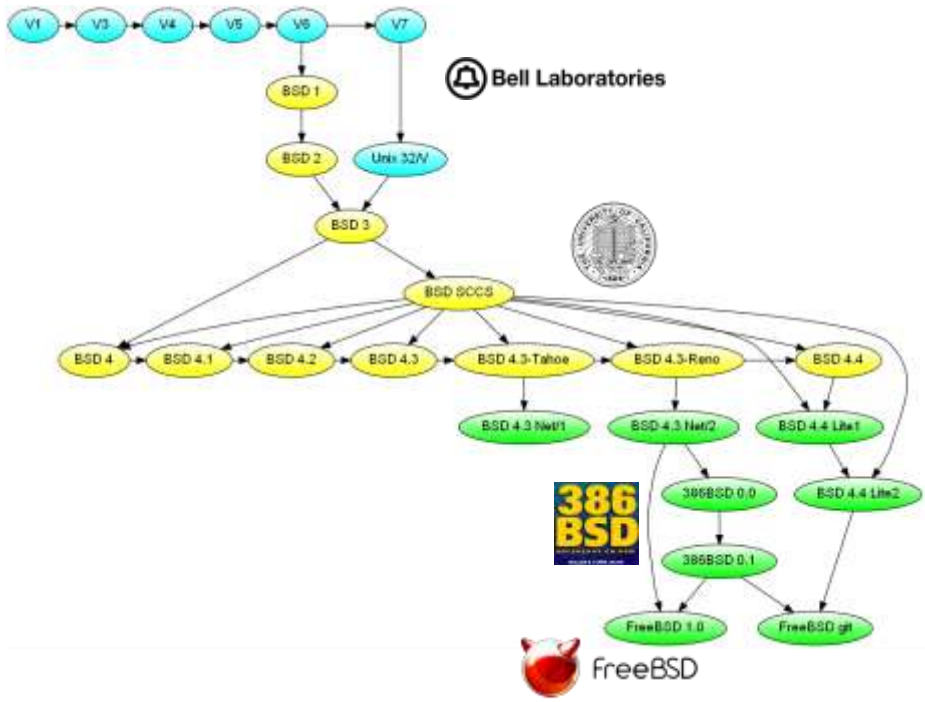

CHABADA
Checking App Behavior Against App Descriptions
About CHABADA
How do we know a program does what it claims to do? After clustering Android apps by their description topics, we identify outliers in each cluster with respect to their API usage. A "weather" app that sends messages thus becomes an anomaly; likewise, a "messaging" app would typically not be expected to access the current location. Applied on a set of 22,500+ Android applications, our CHABADA prototype identified several anomalies; additionally, it flagged 56% of novel malware as such, without requiring any known malware patterns.
Videos
Watch the video of the CHABADA work presented at Microsoft Research.
CHABADA on ARD! Watch the video here (in german).
CHABADA on SR! Watch the video here (in german).
Downloads
Download the preprint of the CHABADA paper here. The final revised version of the paper will appear at the International Conference of Software Engineering (ICSE) 2014.
Download the whole dataset of the CHABADA experiments here.
@inproceedings{GorlaIdaVi CHABADA:ICSE:2014,
author = {Alessandra Gorla and Ilaria Tavecchia and Florian Gross and Andreas Zeller},
title = {Checking App Behavior Against App Descriptions},
booktitle = {ICSE'14: Proceedings of the 36th International Conference on Software Engineering},
location = {Hyderabad (India), 31 May - 7 June},
year = {2014},
Contact:
SE chair at Saarland University
Alessandra Gorla
Ilaria Tavecchia
Florian Gross
Andreas Zeller


Apps
Categories
Home
Top Charts
New Releases
My apps
Shop
Games
Editors' Choice
Similar
Viber
WhatsApp Mes
LINE: Free Calls
Facebook Mes
Viber- Free Mes
More from developer
Skype Wi-Fi



`cvs2svn`

git fast-import



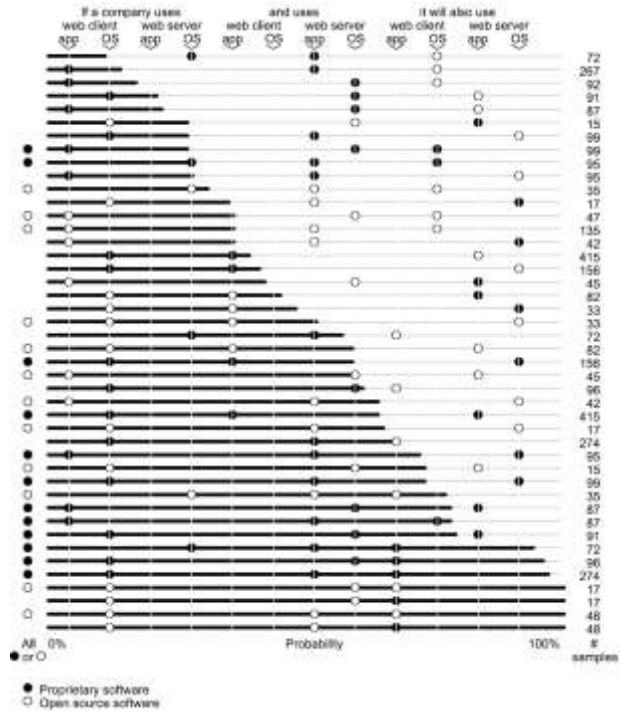
```
$ git log --reverse --date-order
```

```
commit 94a21182365ebb258eeee2aa41c5fbcb1f7fd566
Author: Ken Thompson and Dennis Ritchie <research!{ken,dmr}>
Date:   Tue Jun 20 04:00:00 1972 -0500
    Research V1 development

    Work on file u5.s
commit b7b2640b9e27415d453a8fbe975a87902a01849d
[…]
Author: Ken Thompson <research!ken>
Date:   Tue Nov 26 18:13:21 1974 -0500
    Research V5 development

    Work on file usr/sys/ken/slp.c
commit 3d19667a65d35a411d911282ed8b87e32a56a349
[…]
Author: Dennis Ritchie <research!dmr>
Date:   Mon Dec 2 18:18:02 1974 -0500
    Research V5 development

    Work on file usr/sys/dmr/kl.c
[…]
commit 171931a3f6f28ce4d196c20fdec6a4413a843f89
Author: Brian W. Kernighan <research!bwk>
Date:   Tue May 13 19:43:47 1975 -0500
    Research V6 development

    Work on file rat/r.g
[…]
commit ac4b13bca433a44a97689af10247970118834696
Author: S. R. Bourne <research!srb>
Date:   Fri Jan 12 02:17:45 1979 -0500
    Research V7 development

    Work on file usr/src/cmd/sh/blok.c
[…]
Author: Eric Schmidt <schmidt@ucbvax.Berkeley.EDU>
Date:   Sat Jan 5 22:49:18 1980 -0800
    BSD 3 development

    Work on file usr/src/cmd/net/sub.c
```

```
$ git blame -C -C usr/sys/sys/pipe.c
```

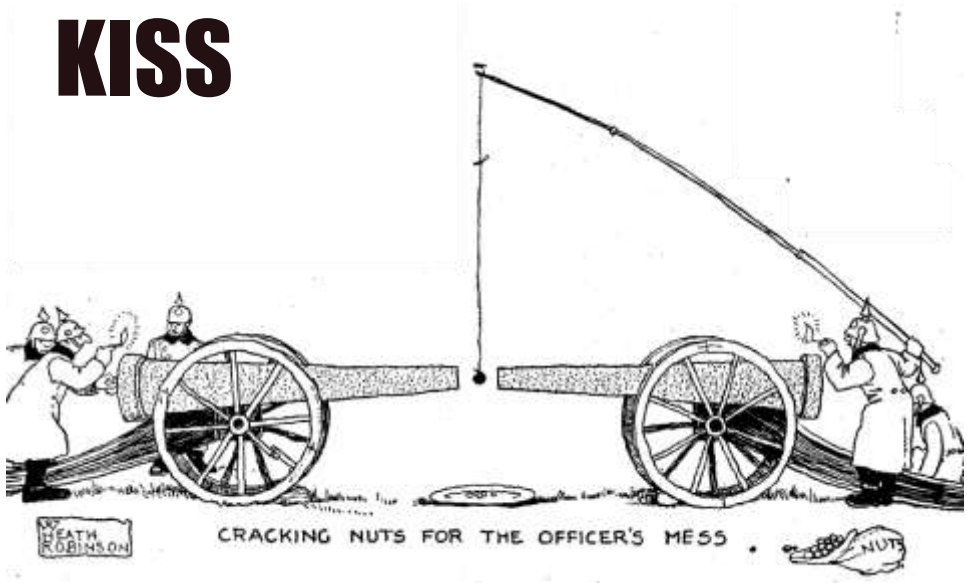
```
3cc1108b usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  53)   rf->f_flag = FREAD|FPIPE;
3cc1108b usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  54)   rf->f_inode = ip;
3cc1108b usr/sys/ken/pipe.c     (Ken Thompson 1974-11-26 18:13:21 -0500  55)   ip->i_count = 2;
[...]
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 122)   register struct inode *ip;
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 123)
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 124)   ip = fp->f_inode;
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 125)   c = u.u_count;
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 126)
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 127) loop:
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 128)
1f183be2 usr/sys/sys/pipe.c     (Ken Thompson 1979-01-10 15:19:35 -0500 129)   /*
9a9f6b22 usr/src/sys/sys/pipe.c (Bill Joy     1980-01-05 05:51:18 -0800 130)    * If error or all done, return.
9a9f6b22 usr/src/sys/sys/pipe.c (Bill Joy     1980-01-05 05:51:18 -0800 131)    */
9a9f6b22 usr/src/sys/sys/pipe.c (Bill Joy     1980-01-05 05:51:18 -0800 132)
9a9f6b22 usr/src/sys/sys/pipe.c (Bill Joy     1980-01-05 05:51:18 -0800 133)   if (u.u_error)
9a9f6b22 usr/src/sys/sys/pipe.c (Bill Joy     1980-01-05 05:51:18 -0800 134)               return;
6d632e85 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500 135)   plock(ip);
6d632e85 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500 136)   if(c == 0) {
6d632e85 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500 137)               prele(ip);
6d632e85 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500 138)               u.u_count = 0;
6d632e85 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500 139)               return;
6d632e85 usr/sys/ken/pipe.c     (Ken Thompson 1975-07-17 10:33:37 -0500 140)   }
```

`nmap`



Processing methods

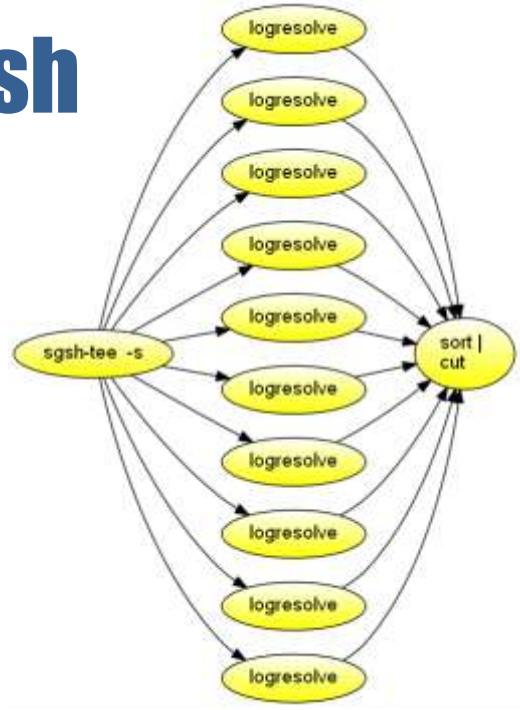
KISS



Count word occurrences

- 20GB in 36,260 files
- Hadoop: 12:25:20

sgsh



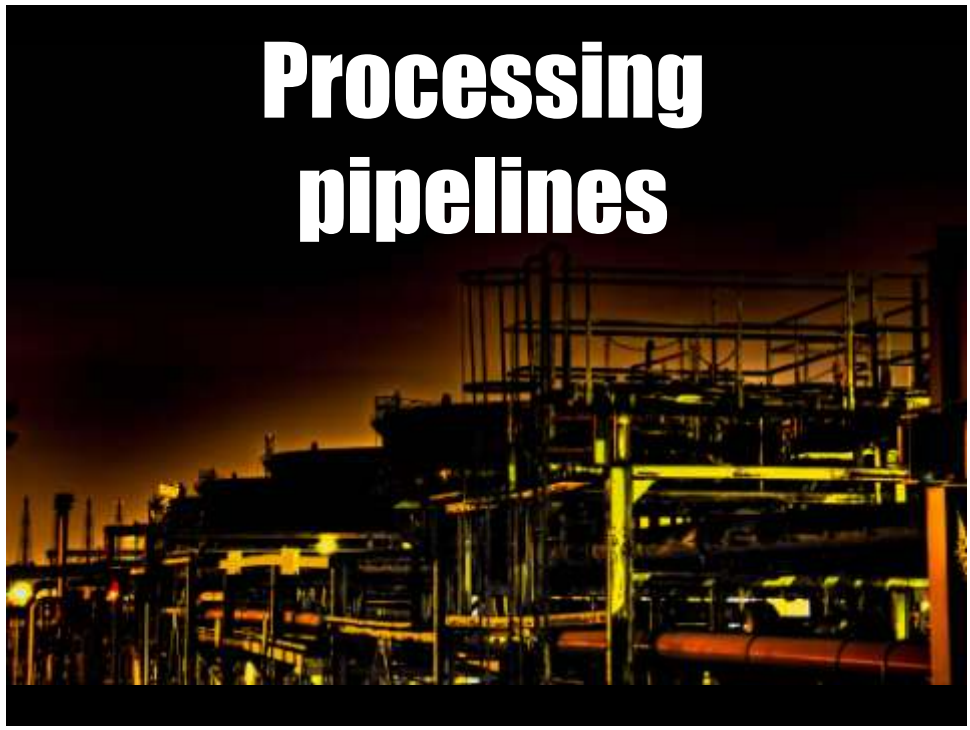
Count word occurrences

- 20GB in 36,260 files
- Hadoop: 12:25:20
- sgsh: 00:34:38

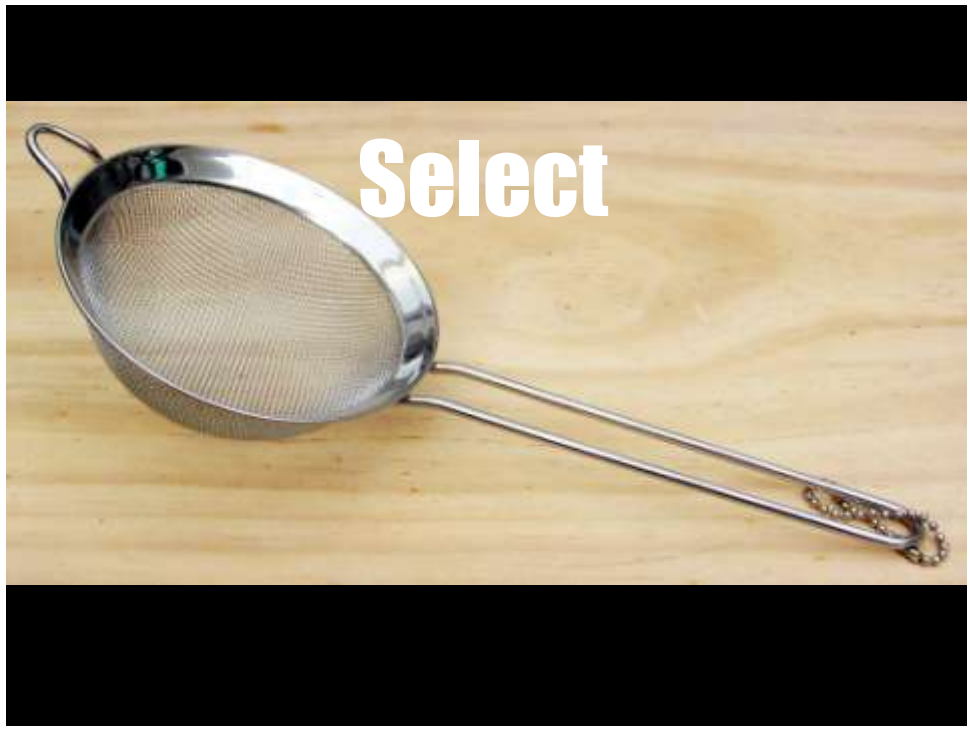
```
find Gutenberg -type f |
xargs cat |
tr -s ' \t\n\r\f' \\n |
sort -S 6G |
uniq -c
```

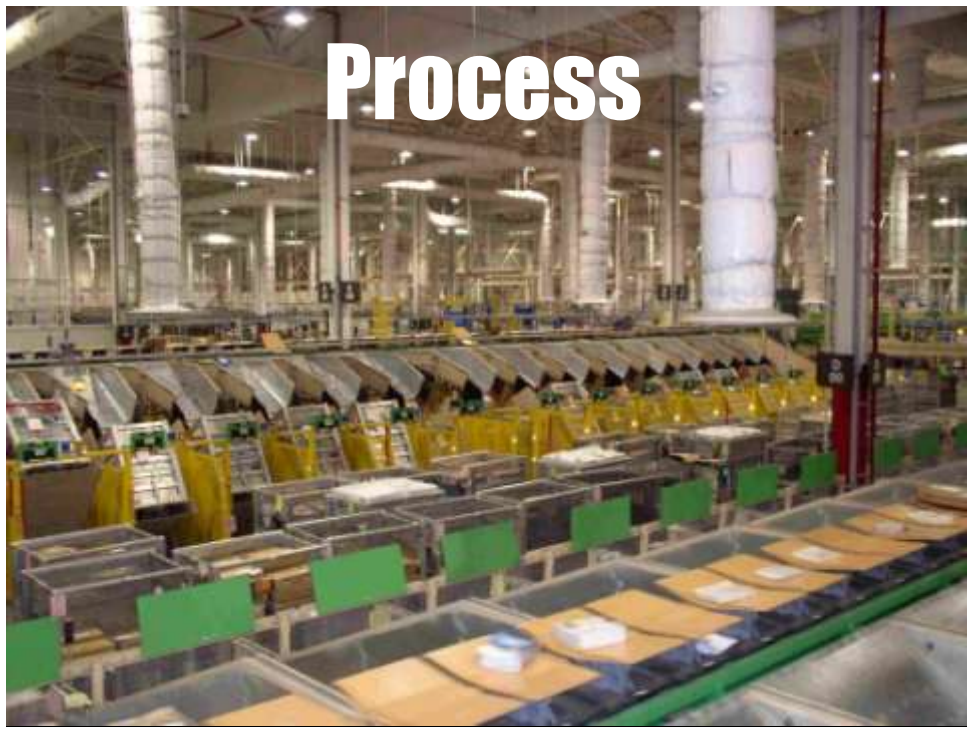
Count word occurrences

- 20GB in 36,260 files
- Hadoop: 12:25:20
- sgsh: 00:34:38
- Unix pipeline: 00:21:37


Processing
pipelines


Get


Select


Process

Summarize

curl
wget

git log
git blame

```
$ # Print author names
git log --format='%an' |

# Order them by author
sort |

# Count number of commits for each author
uniq --count |

# Order them by number of commits
sort --numeric-sort --reverse |

# Print top ten
head -10
```

```
  20131 Linus Torvalds
   8445 David S. Miller
   7692 Andrew Morton
   5156 Greg Kroah-Hartman
   5116 Mark Brown
   4723 Russell King
   4584 Takashi Iwai
   4385 Al Viro
   4220 Ingo Molnar
   3276 Tejun Heo
```

Journal of Universal Computer Science, vol. 18, no. 19 (2012), 2625-2645
submitted: 30/3/12, accepted: 6/9/12, appeared: 1/11/12 © *J.UCS*

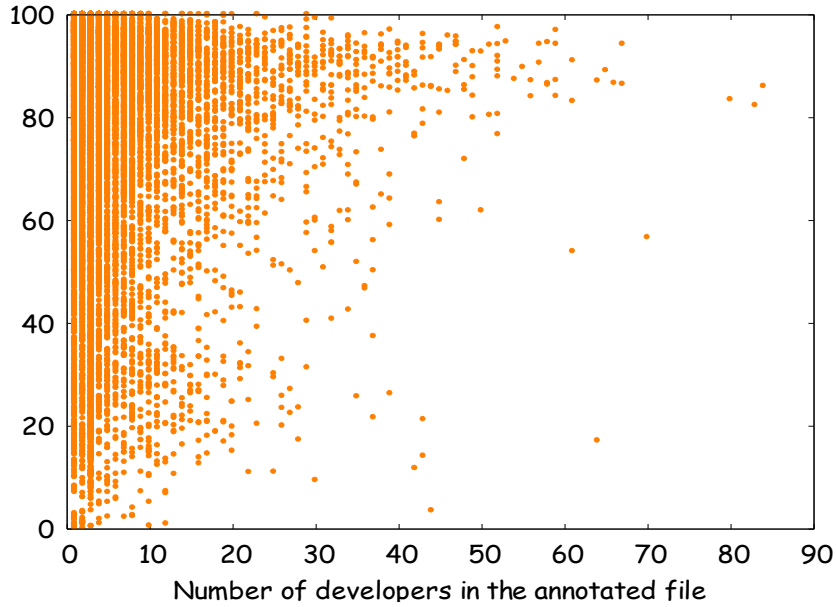
Division of Effort, Productivity, Quality, and Relationships in FLOSS Virtual Teams: Evidence from the FreeBSD Project

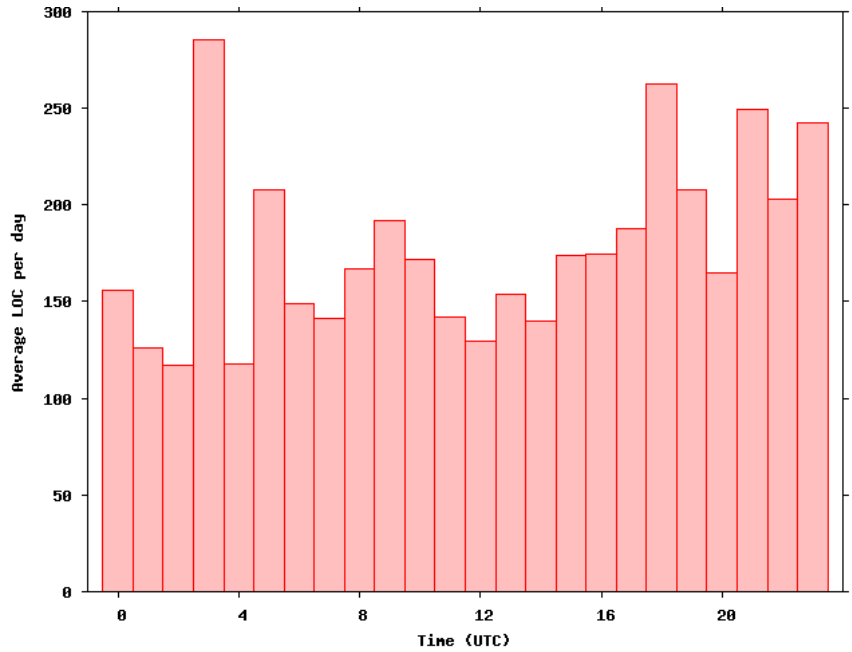
George M. Giaglis
(Department of Management Science and Technology
Athens University of Economics and Business, Greece
giaglis@aueb.gr)

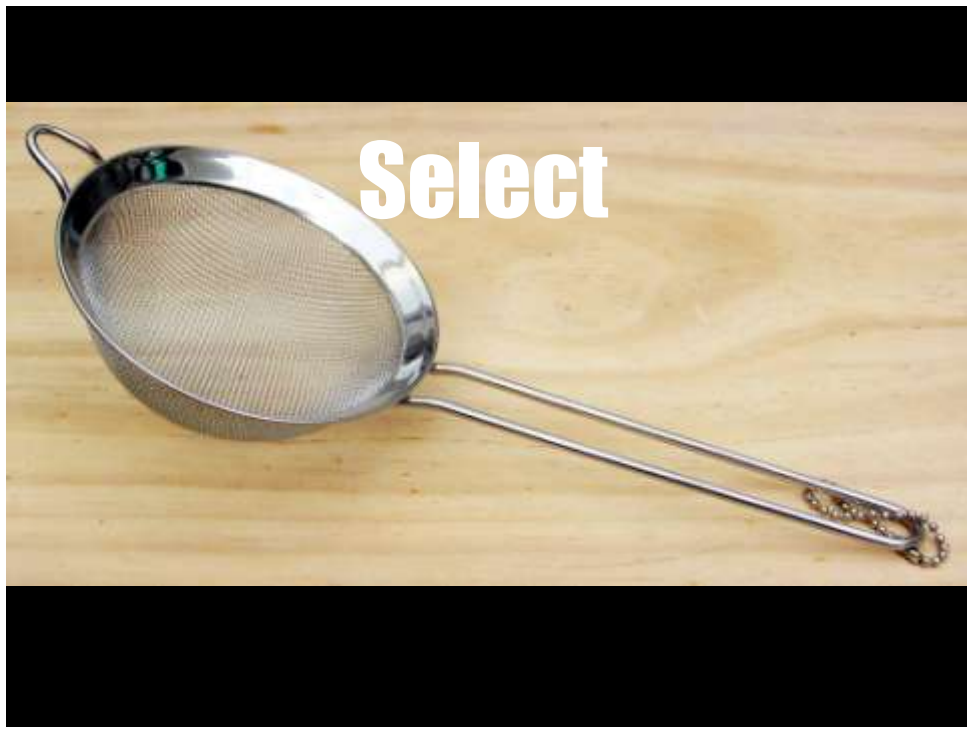
Dionidis Spinellis
(Department of Management Science and Technology
Athens University of Economics and Business, Greece
dds@aueb.gr)

Abstract: Research in virtual teams and distributed work argues that the lack of collocation places an overhead on the performance potential of large, globally distributed teams. In this paper, we revisit this tenet through a case study of Free/Libre Open Source Software (FLOSS) development to demonstrate how globally dispersed FLOSS communities manage to overcome the problem of geographic separation of their members. Our results show that successful FLOSS teams demonstrate a truly global distribution of members, who perform different types of work so as to achieve consistent round-the-clock development, without any apparent ill effects on team productivity and the quality of the resulting outcomes. Cooperation between team members is abundant, especially at more complex work items, and does not seem to be affected by distance; only mentoring relationships appear in some cases to be easier to cultivate between individuals living closer together. These findings challenge the conventional wisdom of research in distributed work, in cases where virtual teams consist of highly skilled and motivated individuals, who leverage the power of communication technologies to overcome problems associated with physical distance.

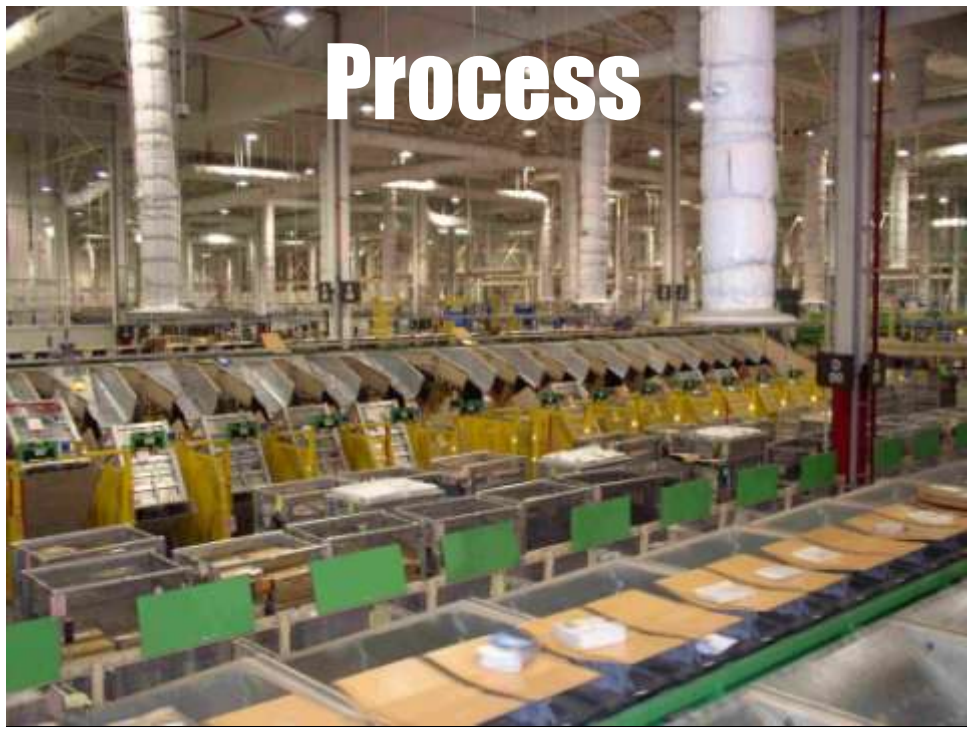
Keywords: FLOSS, open source development, virtual teams, distributed work, FreeBSD
Categories: D.2.9, K.4.3, K.6.1, K.6.3




Average LOC per day
300
250
200
150
100
50
0
0
4
8
12
16
20
Time (UTC)


Select

```
grep
cut
awk
sed
```



sort

- -n
- -r
- +2nr
- -u
- -t:

diff

```
$ diff file1 file2 |
grep '^[<>]' |
wc -l
```

comm

```
$ comm Linux FreeBSD
                [
arch
ash
bash
                                cat
                chflags
chgrp
                chio
                                chmod
chown
                                cp
cpio
                                csh
                                date
                                dd
                                df
dir
dmesg
dnsdomainname
                domainname
                                echo
                                ed
egrep
                expr
false
                getfacl
```


join


gvpr

```
digraph D {
        "archivers/arj" -> "devel/gmake"
        "archivers/arj" -> "devel/autoconf259"
        "archivers/dact" -> "devel/gmake"
        "archivers/dact" -> "security/libmcrypt"
        "archivers/dact" -> "archivers/lzo"
        "archivers/deepforest" -> "x11-toolkits/tkstep80"
        "archivers/deepforest" -> "graphics/libimg"
        "archivers/deepforest" -> "x11/xorg-libraries"
        "archivers/fastjar" -> "devel/gmake"
        "archivers/fastjar" -> "lang/perl5.8"
        "archivers/gtar" -> "converters/libiconv"
        "archivers/gtar" -> "devel/gettext"
```

```
BEG_G {
        $tvtype = TV_fwd;
        $tvroot = node($, ARGV[0]);
}
N [$tvroot = NULL; 1]
END_G {
        induce($T);
        write($T);
        exit(0);
}
```

Open Source Licensing across Package Dependencies

Maria Kechagia, Diomidis Spinellis and Stephanos Androutsellis-Theotokis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece GR-104 34
Email: mkehagia@dmst.aueb.gr, {dds, stheotok}@aueb.gr

***Abstract*—Licensing dependencies among open source software (oss) packages reveal a lot about software compatibility relationships and the practicalities of oss licensing. There is, however, limited information on these in the literature. In this paper, we discuss various aspects of oss licensing, and we present an empirical study on FreeBSD ports collections concerning their licensing dependencies, in an attempt to identify specific patterns. Our results highlight different types of dependencies, that could be used to explain, or even guide the license selection process of oss projects.**

***Keywords*-oss; licenses; dependencies; FreeBSD ports**

I. Introduction

Open source software (OSS), can be freely used, modified, and distributed, provided certain restrictions are observed regarding its copyright and the protection of its status as OSS. These rights and restrictions are expressed through the software's license, i.e. a contract between the software owner(s) (the licensors) and its prospective users (the licensees). OSS

1) Are some license types particularly conductive to reuse?
2) Do the licenses across dependencies follow an order associated with the permissiveness of each license?

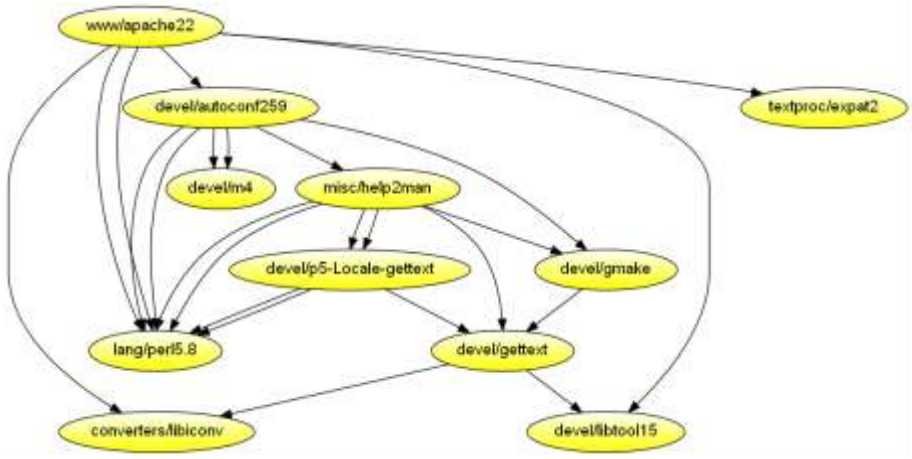
Results could explain or even guide the selection of a particular OSS license.

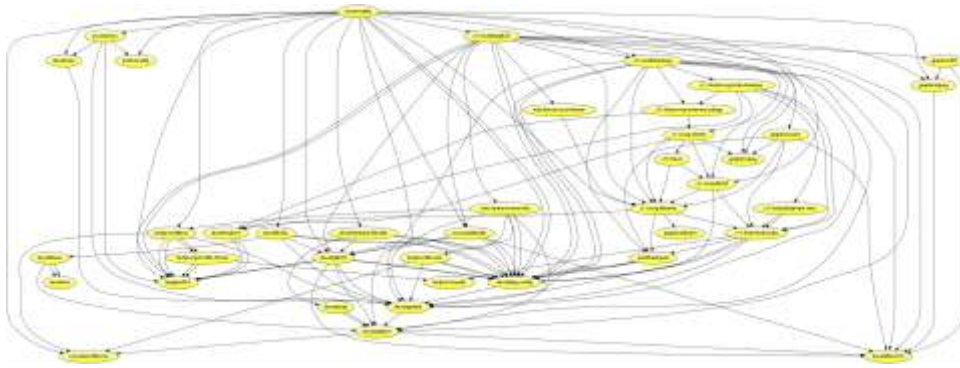
II. Concepts and definitions

Before discussing the different types of OSS licenses, we briefly introduce the background concepts delineating the degrees of freedom available while distributing the product of any intellectual activity, including software.

A. Intellectual property, copyrights and patents

The term intellectual property is used to encompass a wide range of areas of law, including copyrights, patents and even trademarks [2]. These are all means used to encourage private investment in research, technology and innovation, by ensuring that innovators will be able to get private returns for their work.





Summarize

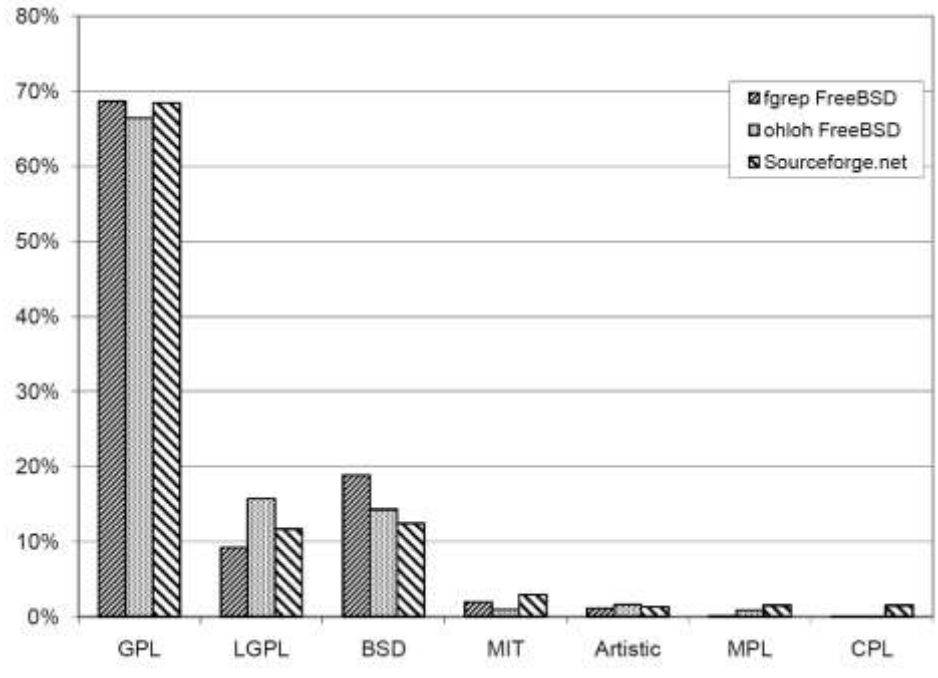
wc
uniq

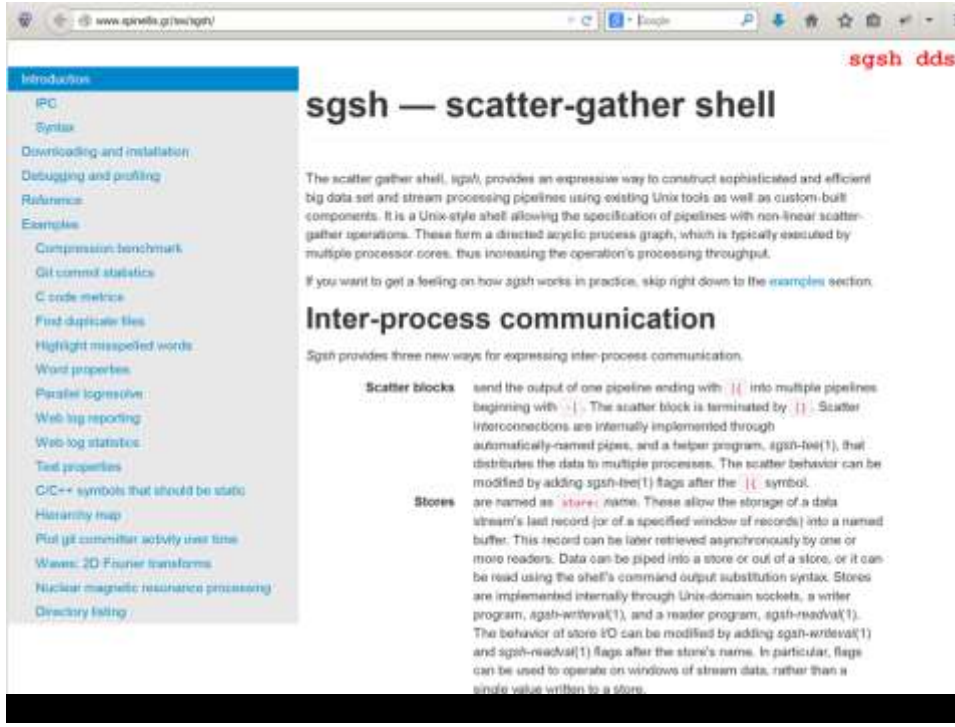
head
tail

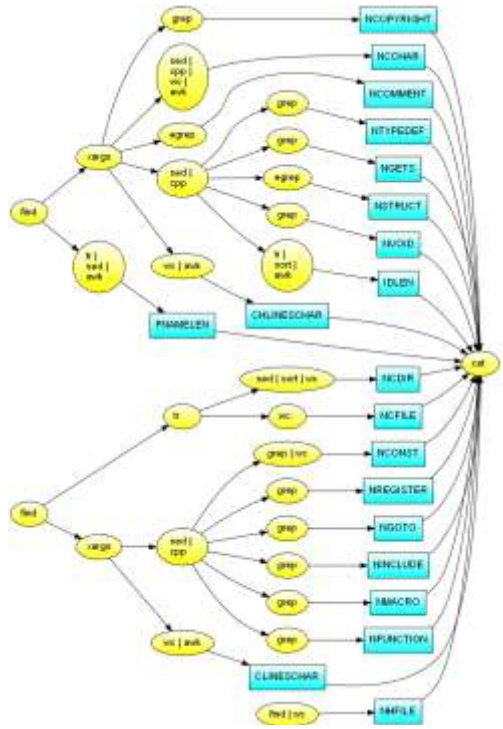
fmt

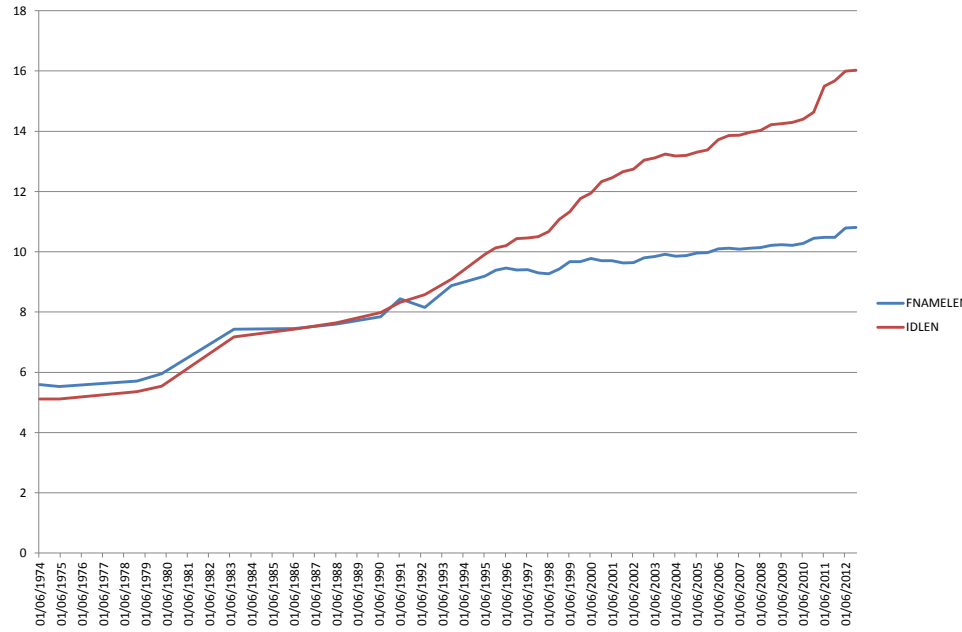
awk

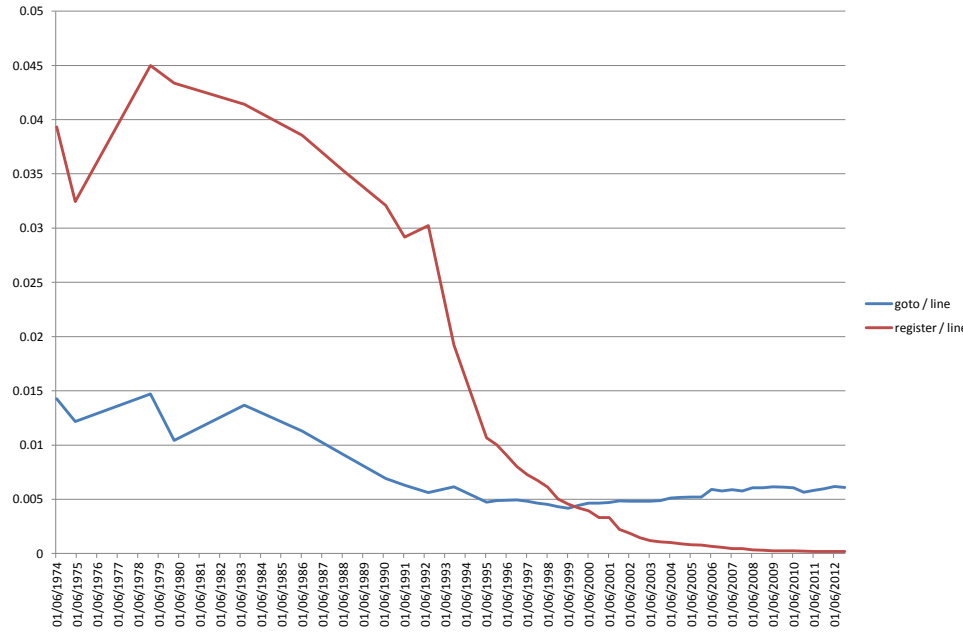
```
awk '{print $2}' licenses |
sort |
uniq -c |
sort -rn
```




sgsh dds
Introduction
IPC
Syntax
Downloading and installation
Debugging and profiling
Reference
Examples
Compression benchmark
Git commit statistics
C code metrics
Find duplicate files
Highlight misspelled words
Word properties
Parallel lexgrosolve
Web log reporting
Web log statistics
Text properties
C/C++ symbols that should be static
Hierarchy map
Plot git committer activity over time
Waves: 2D Fourier transforms
Nuclear magnetic resonance processing
Directory listing
sgsh — scatter-gather shell
The scatter gather shell, sgsh, provides an expressive way to construct sophisticated and efficient big data set and stream processing pipelines using existing Unix tools as well as custom-built components. It is a Unix-style shell allowing the specification of pipelines with non-linear scatter-gather operations. These form a directed acyclic process graph, which is typically executed by multiple processor cores, thus increasing the operation's processing throughput.
If you want to get a feeling on how sgsh works in practice, skip right down to the examples section.
Inter-process communication
Sgsh provides three new ways for expressing inter-process communication.
Scatter blocks send the output of one pipeline ending with |{ into multiple pipelines beginning with -| . The scatter block is terminated by |} . Scatter interconnections are internally implemented through automatically-named pipes, and a helper program, sgsh-tee(1), that distributes the data to multiple processes. The scatter behavior can be modified by adding sgsh-tee(1) flags after the |{ symbol.
Stores are named as store: name. These allow the storage of a data stream's last record (or of a specified window of records) into a named buffer. This record can be later retrieved asynchronously by one or more readers. Data can be piped into a store or out of a store, or it can be read using the shell's command output substitution syntax. Stores are implemented internally through Unix-domain sockets, a writer program, sgsh-writeval(1), and a reader program, sgsh-readval(1). The behavior of store I/O can be modified by adding sgsh-writeval(1) and sgsh-readval(1) flags after the store's name. In particular, flags can be used to operate on windows of stream data, rather than a single value written to a store.


find
xargs
sed | cpp
grep
NCOPYRIGHT
NCCHAR
NCOMMENT
NTYPEDEF
NGETS
NSTRUCT
NVOID
NDEBN
CHLINESCHAR
PNAMELEN
cat
NCIDR
NCFILE
NCONST
NREGISTER
NGOTO
NINCLUDE
NMACRO
NFUNCTION
CLINESCHAR
NNFILE


18
16
14
12
10
8
6
4
2
0
01/06/1974
01/06/1975
01/06/1976
01/06/1977
01/06/1978
01/06/1979
01/06/1980
01/06/1981
01/06/1982
01/06/1983
01/06/1984
01/06/1985
01/06/1986
01/06/1987
01/06/1988
01/06/1989
01/06/1990
01/06/1991
01/06/1992
01/06/1993
01/06/1994
01/06/1995
01/06/1996
01/06/1997
01/06/1998
01/06/1999
01/06/2000
01/06/2001
01/06/2002
01/06/2003
01/06/2004
01/06/2005
01/06/2006
01/06/2007
01/06/2008
01/06/2009
01/06/2010
01/06/2011
01/06/2012
FNAMELEN
IDLEN


0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
01/06/1974
01/06/1975
01/06/1976
01/06/1977
01/06/1978
01/06/1979
01/06/1980
01/06/1981
01/06/1982
01/06/1983
01/06/1984
01/06/1985
01/06/1986
01/06/1987
01/06/1988
01/06/1989
01/06/1990
01/06/1991
01/06/1992
01/06/1993
01/06/1994
01/06/1995
01/06/1996
01/06/1997
01/06/1998
01/06/1999
01/06/2000
01/06/2001
01/06/2002
01/06/2003
01/06/2004
01/06/2005
01/06/2006
01/06/2007
01/06/2008
01/06/2009
01/06/2010
01/06/2011
01/06/2012
goto / line
register / line

Scripting languages

github.com/dspinellis/unix-history-repo

- Research Edition Unix: V1, V3–V7
- Unix 32V
- BSD 1, 2, 3, 4, 4.1, 4.2, 4.3 *, 4.4 *
- 386BSD 0.0, 0.1
- FreeBSD 1.0–10.0

Bell Laboratories

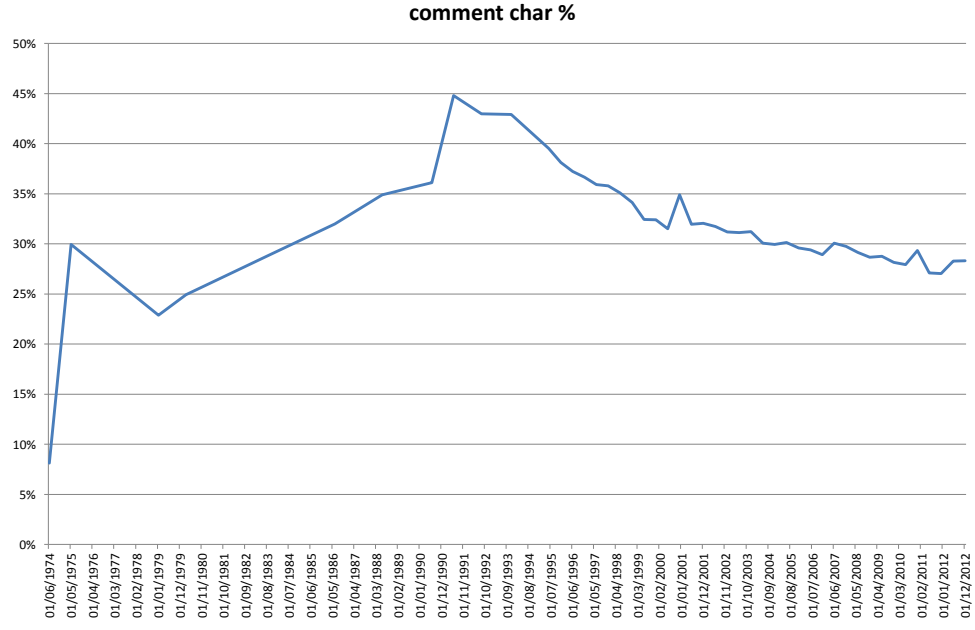
386BSD

- Tags
- Contributors
- Branches and merges

Tool front ends


```
find /usr/ports/ -name Makefile
-maxdepth 3 |
sed 's,/Makefile,,' |
while read dir
do
    cd $dir
    make -V PORTNAME\
-V RUN_DEPENDS -V BUILD_DEPENDS\
-V LIB_DEPENDS -V FETCH_DEPENDS\
-V DEPENDS
done
```

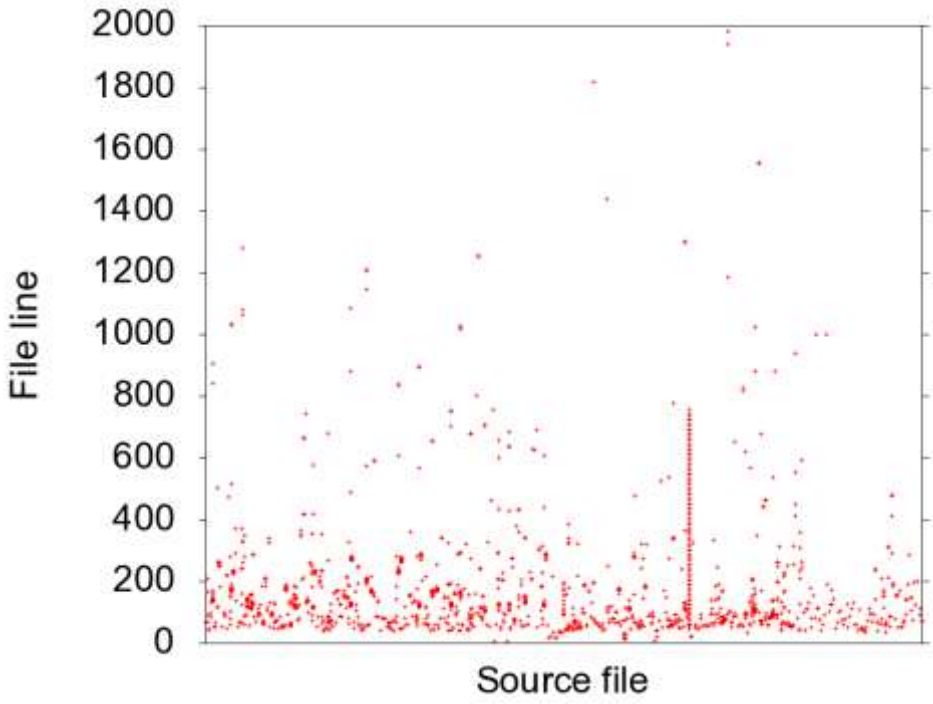

C Preprocessor


comment char %
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
01/06/1974
01/05/1975
01/04/1976
01/03/1977
01/02/1978
01/01/1979
01/12/1979
01/11/1980
01/10/1981
01/09/1982
01/08/1983
01/07/1984
01/06/1985
01/05/1986
01/04/1987
01/03/1988
01/02/1989
01/01/1990
01/12/1990
01/11/1991
01/10/1992
01/09/1993
01/08/1994
01/07/1995
01/06/1996
01/05/1997
01/04/1998
01/03/1999
01/02/2000
01/01/2001
01/12/2001
01/11/2002
01/10/2003
01/09/2004
01/08/2005
01/07/2006
01/06/2007
01/05/2008
01/04/2009
01/03/2010
01/02/2011
01/01/2012
01/12/2012


```
nm,readelf
ar,ldd
dumpbin
javap
```

```
$ cd /usr/bin
$ ldd * 2>/dev/null |
awk '/=>/{print $3}' |
sort |
uniq -c |
sort -rn |
head
```

```
    541 /lib/libc.so.6
    541 /lib/ld-linux.so.2
    104 /lib/libdl.so.2
     92 /lib/libm.so.6
     62 /lib/libncurses.so.5
     59 /lib/libnsl.so.1
     50 /lib/libcrypt.so.1
     34 /usr/lib/libstdc++-
libc6.2-2.so.3
     34 /lib/libpam.so.0
     28 /usr/lib/libz.so.1
```



The Bug Catalog of the Maven Ecosystem

Dimitris Mitropoulos* Vassilios Karakoidas* Panos Louridas* Georgios Gousios†
Diomidis Spinellis*
*Department of Management Science and Technology
*Athens University of Economics and Business
Athens, Greece
{dimitro,bkarak,louridas,dds}@aueb.gr
†Software Engineering Research Group
†Delft University of Technology
Delft, the Netherlands
g.gousios@tudelft.nl

ABSTRACT

Examining software ecosystems can provide the research community with data regarding artifacts, processes, and communities. We present a dataset obtained from the Maven central repository ecosystem (approximately 265GB of data) by statically analyzing the repository to detect potential software bugs. For our analysis we used *FindBugs*, a tool that examines Java bytecode to detect numerous types of bugs. The dataset contains the metrics results that FindBugs reports for every project version (a JAR) included in the ecosystem. For every version we also stored specific metadata such as the JAR's size, its dependencies and others. Our dataset can be used to produce interesting research results, as we show in specific examples.

Categories and Subject Descriptors

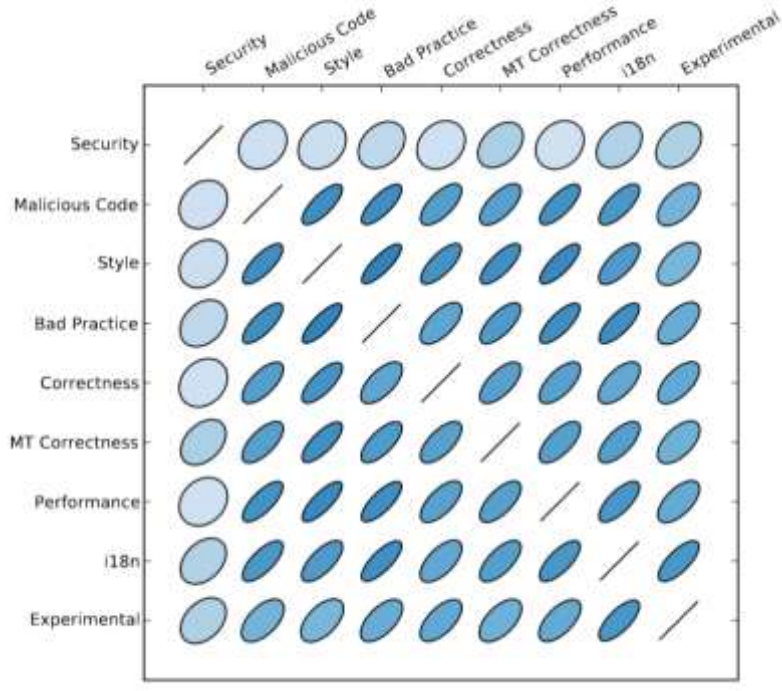
D.2.4 [**Software Engineering**]: Software/Program Veri-

Measurement	Value
Projects	17,505
Versions (total)	115,214
Min (versions per project)	1
Max (versions per project)	338
Mean (versions per project)	6.58
Median (versions per project)	3
Range (over versions)	337
1st Quartile (over versions)	1
3rd Quartile (over versions)	8

Table 1: Descriptive statistic measurements for the Maven repository.

(Comprehensive Perl Archive Network), Ruby's RubyGems[3] and the Maven Central Repository.[4]

Maven is a build automation tool used primarily for Java



Tool Interoperability

Fork me on GitHub

ckjm dds

ckjm — Chidamber and Kemerer Java Metrics

The program ckjm calculates Chidamber and Kemerer object-oriented metrics by processing the bytecode of compiled Java files. The program calculates for each class the following six metrics proposed by Chidamber and Kemerer.

- WMC: Weighted methods per class
- DIT: Depth of Inheritance Tree
- NOC: Number of Children
- CBO: Coupling between object classes
- RFC: Response for a Class
- LCOM: Lack of cohesion in methods

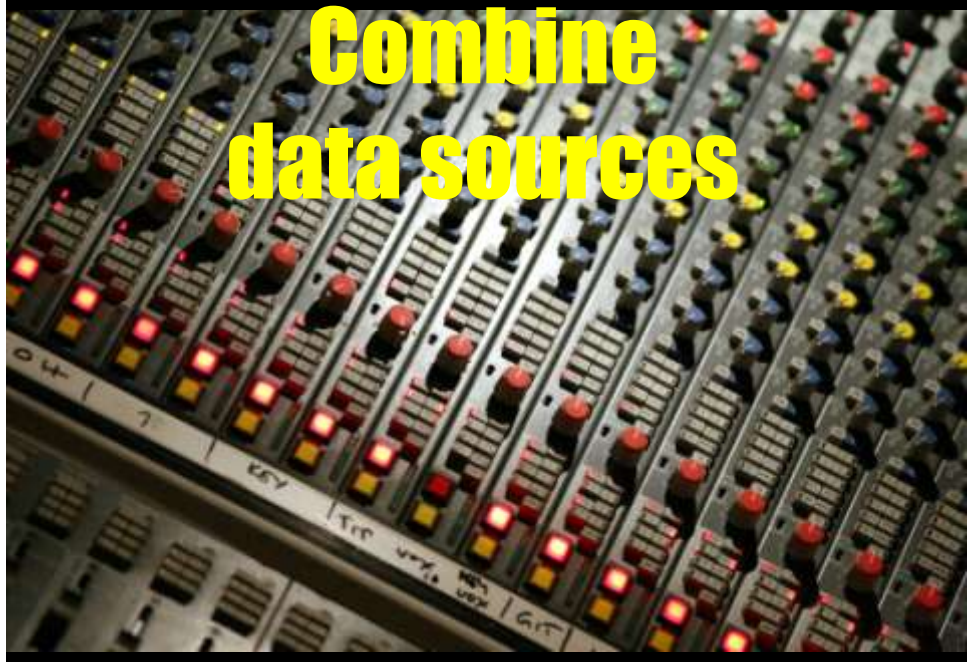
In addition it also calculates for each class

- Ca: Afferent couplings
- NPM: Number of public methods

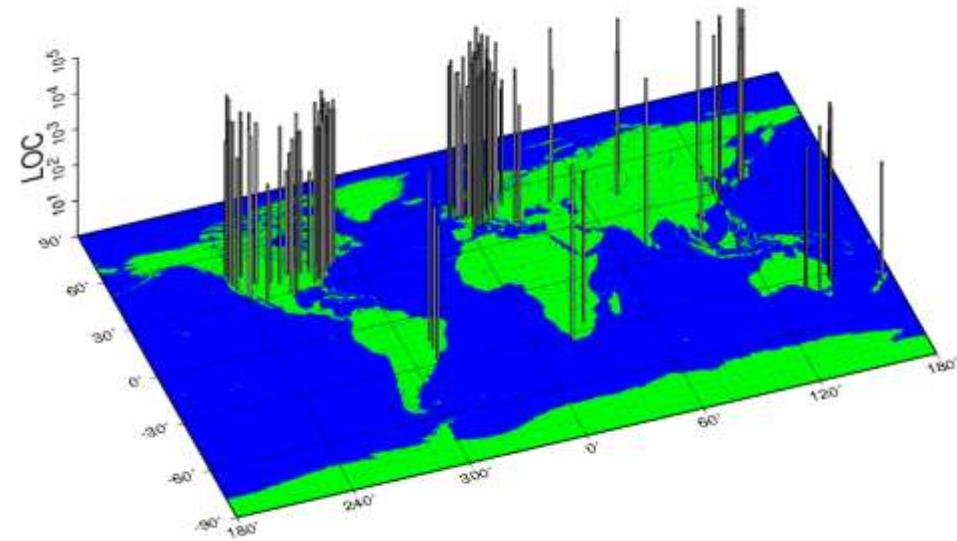
Citation and Background

If you use this tool in your research, please cite it as follows.
Diomidis Spinellis. Tool writing: A forgotten art? *IEEE Software*, 22(4):9–11, July/August 2005. (doi:10.1109/MS.2005.111).

I wrote this program out of frustration over the lack of reliable programs to calculate the Chidamber and Kemerer object-oriented metrics I needed to illustrate some concepts in my book Code Quality: The Open Source Perspective. The programs I found on the web were either incomplete (they calculated only some of the metrics), or unreliable (they calculated results that were obviously wrong), or extremely inefficient (they required GB of RAM and hours of processing). Ckjm is mean and lean, following the Unix tradition of doing one thing well. It will not automatically recurse directories looking for the files you want measured and it does not offer a GUI. However, it does this job thoroughly, and efficiently: on a 1.6GHz Pentium-M machine it will process the 33MB of the Eclipse 3.0 jar files (19717 classes) in 95 seconds.



http://geonames.usgs.gov/
http://earth-info.nga.mil/



OPEN ACCESS Freely available online

PLOS ONE

The Carbon Footprint of Conference Papers

Diomidis Spinellis[1], **Panos Louridas**[1,2]*

1 Department of Management Science and Technology, Athens University of Economics and Business, Athens, Greece, 2 Greek Research and Technology Network, Athens, Greece

Abstract

The action required to stem the environmental and social implications of climate change depends crucially on how humankind shapes technology, economy, lifestyle and policy. With transport CO_2 emissions accounting for about a quarter of the total, we examine the contribution of CO_2 output by scientific travel. Thankfully for the reputation of the scientific community, CO_2 emissions associated with the trips required to present a paper at a scientific conference account for just 0.003% of the yearly total. However, with CO_2 emissions for a single conference trip amounting to 7% of an average individual's total CO_2 emissions, scientists should lead by example by demonstrating leadership in addressing the issue.

Citation: Spinellis D, Louridas P (2013) The Carbon Footprint of Conference Papers. PLoS ONE 8(6): e66508. doi:10.1371/journal.pone.0066508

Editor: Gil Bohrer, The Ohio State University, United States of America

Received February 6, 2013; **Accepted** May 3, 2013; **Published** June 26, 2013

Funding: This research has been co-financed by the European Union (European Social Fund) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework, Research Funding Program: Thalis –Athens University of Economics and Business–Software Engineering Research Platform. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

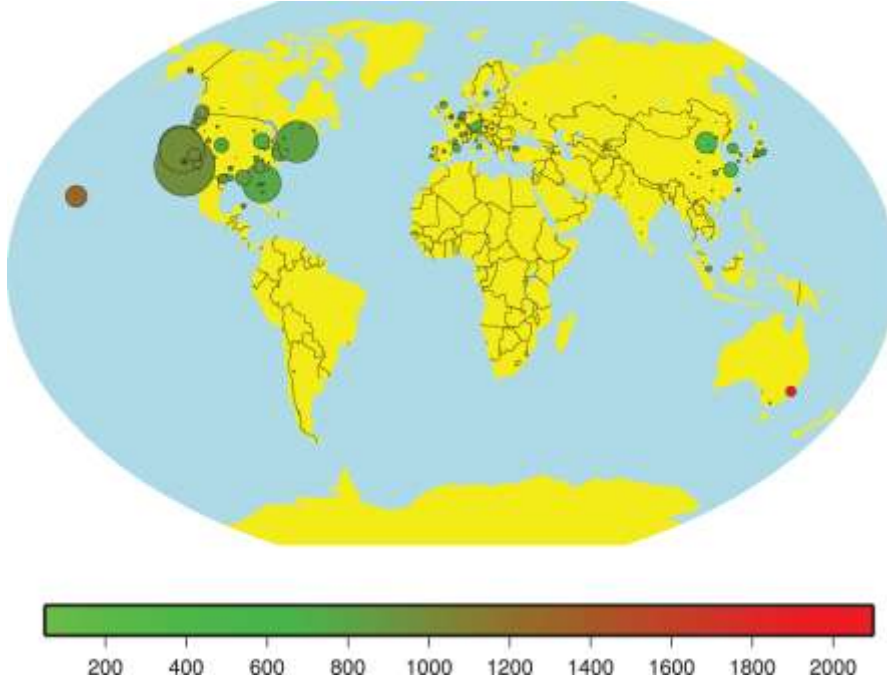
* E-mail: lou@aueb.gr

Introduction

The environmental and social implications of climate change depend not only on Earth's systemic responses, but also on how humankind shapes technology, economy, lifestyle and policy [1]. Action should not be postponed, as it is argued that we have already surpassed a safe threshold in atmospheric carbon dioxide concentration (from a 288 ppm pre-industrial value to 387 ppm today, with a proposed boundary threshold of 350 ppm) [2]. Changes in economy, lifestyle, and policy, entail changes in human behaviour, which will ultimately require decisions involving moral questions. Decisions should not be put off, considering that decisions that delay reduction may have the greatest effect on

communication technologies. Furthermore, the emissions we study are also a subset of the total travel associated with conferences, because some conferences do not publish indexed proceedings, and many scientists attend conferences without presenting a published paper. Extrapolating total conference travel from our data through the use of conference attendance figures is difficult, because, according to our experience, attendance at conferences by scientists who do not have a paper to present tends to be biased toward those living relatively near the conference's location.

We show that CO_2 emissions associated with the trips required to present papers at scientific conferences account for 0.003% of the yearly total travel emissions. This is a bit more than the total



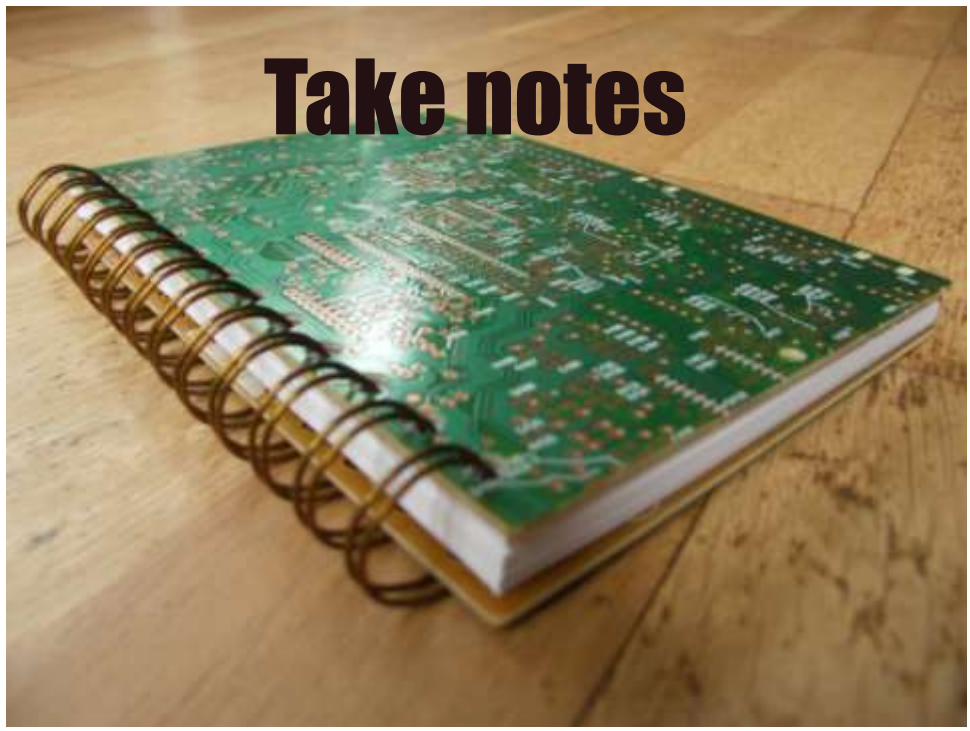
Processing advice

AITMOAF

Automate

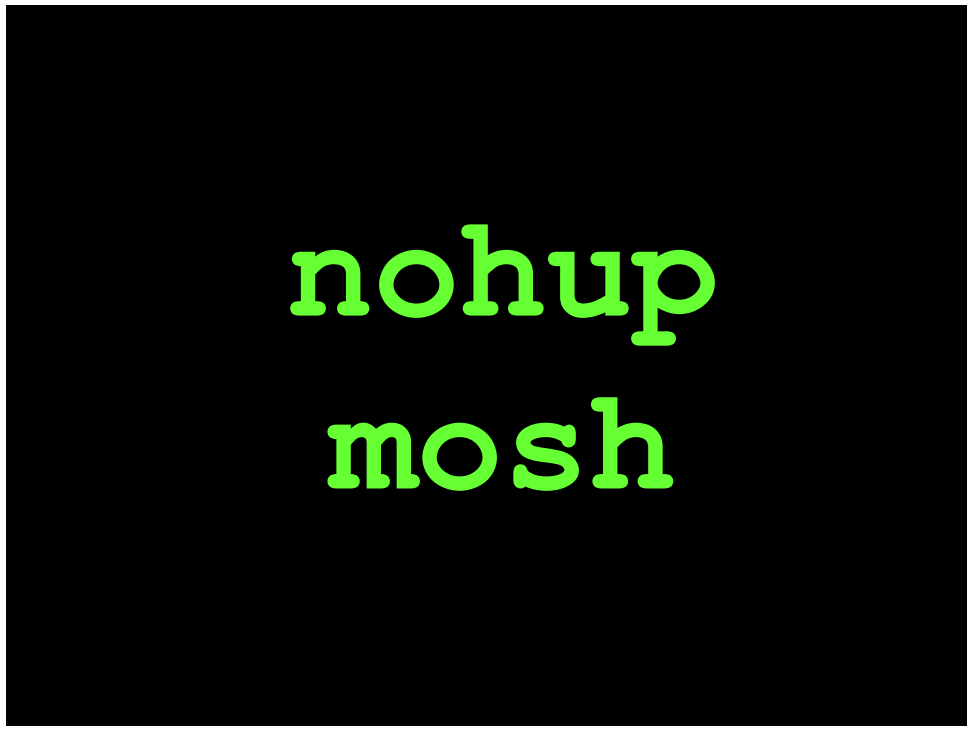
```
make
set -e
a && b && c
```

top/htop
logger
pv
monitor
(www.spinellis.gr/blog/20081027/)



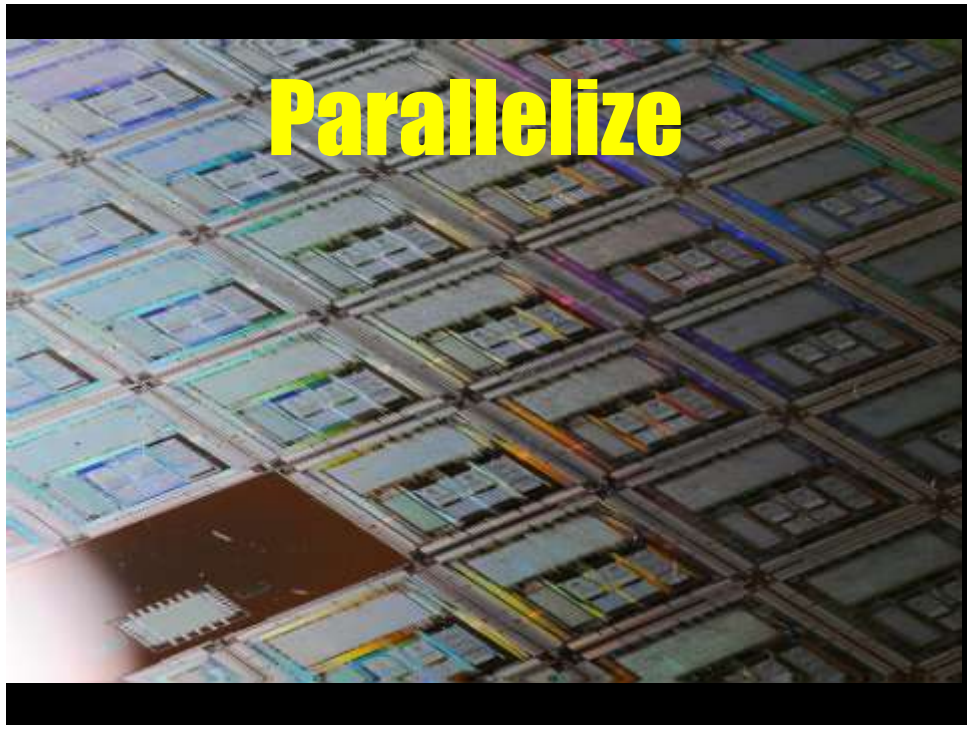

Checkpoint


Handle
broken
connections

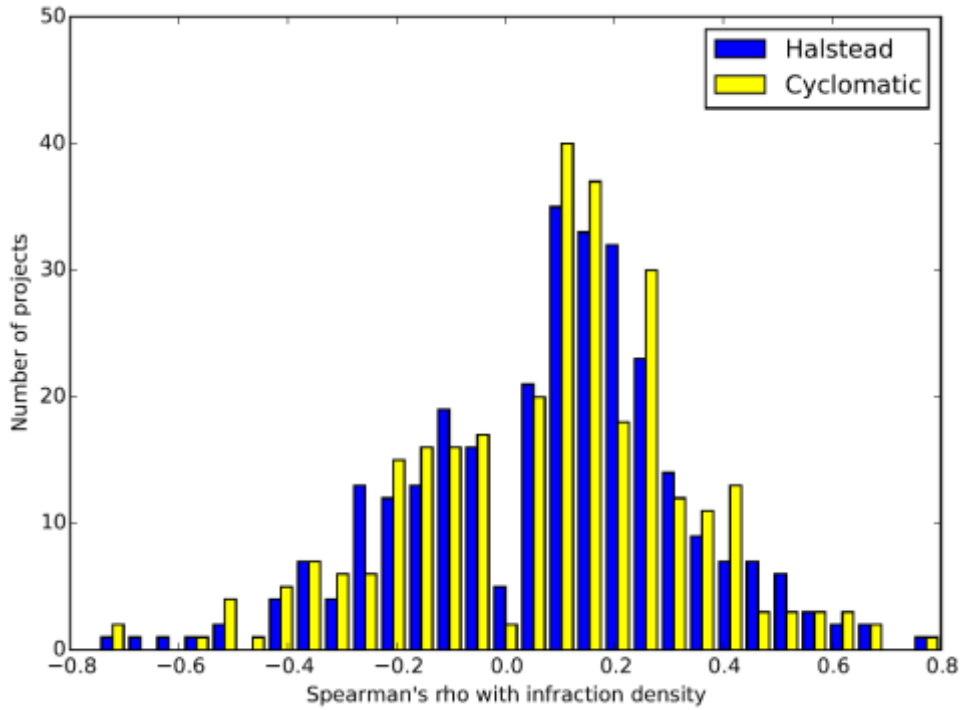


Performance


Optimize

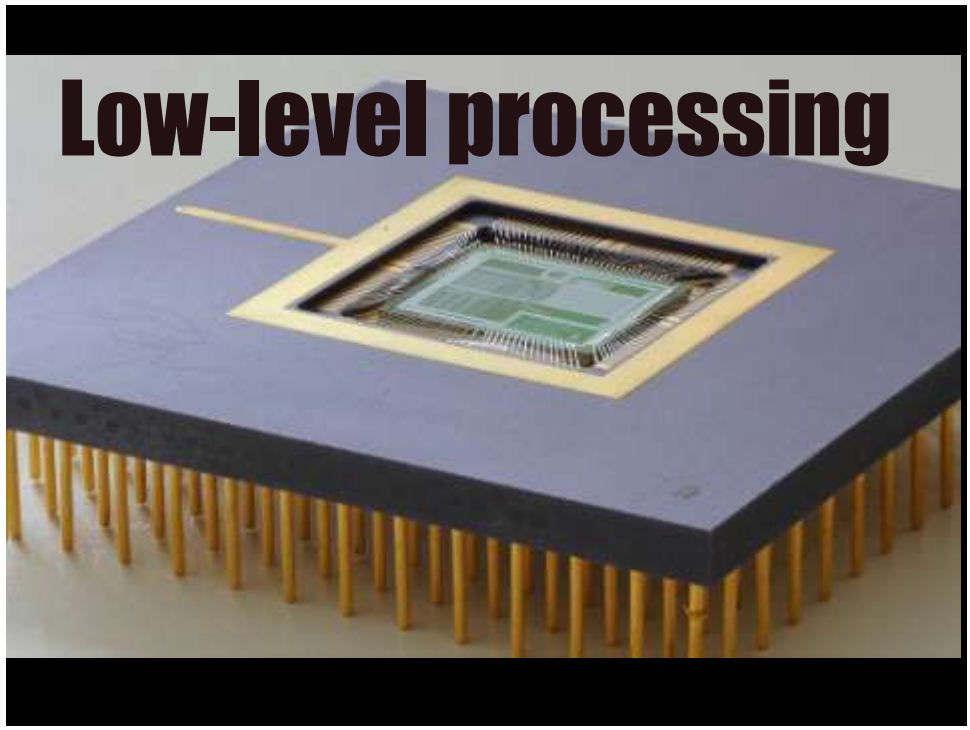

Parallelize

parallel


Halstead
Cyclomatic
Number of projects
Spearman's rho with infraction density
50
40
30
20
10
0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

```
find . -name \*.c -type f -print0 |
xargs -0 -n 1 \
    analyze-file.sh
```

```
find . -name \*.c -type f -print0 |
parallel --keep-order -d '\0' -n 1 -gnu \
    analyze-file.sh
```



DOI:10.1145/1378704.1378720

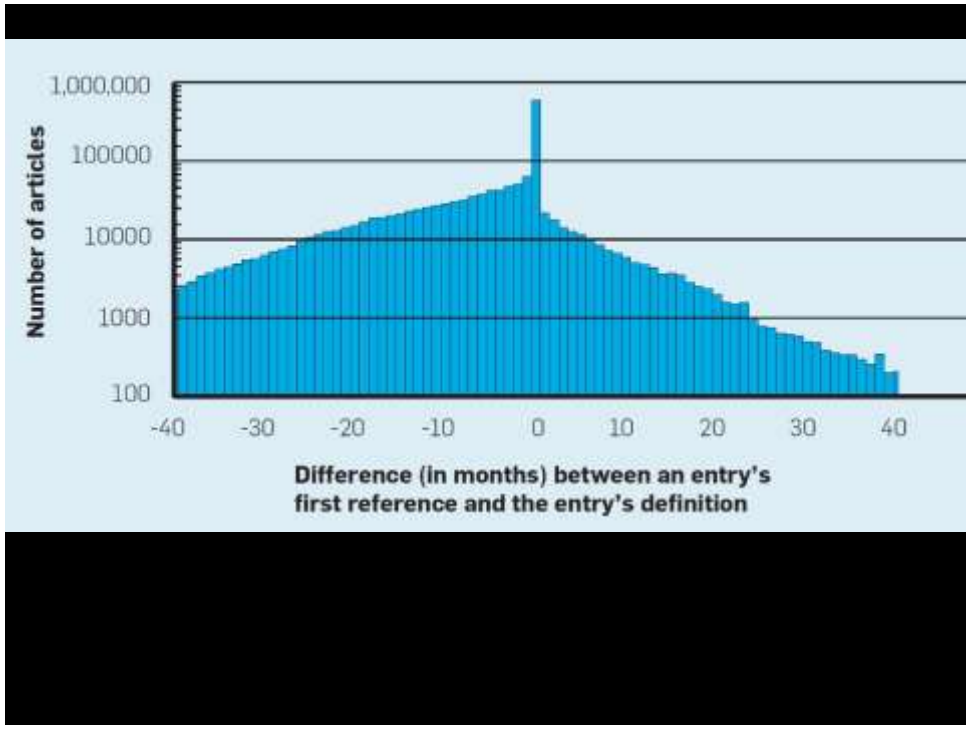
Why Wikipedia's remarkable growth is sustainable.

BY DIOMIDIS SPINELLIS AND PANAGIOTIS LOURIDAS

The Collaborative Organization of Knowledge

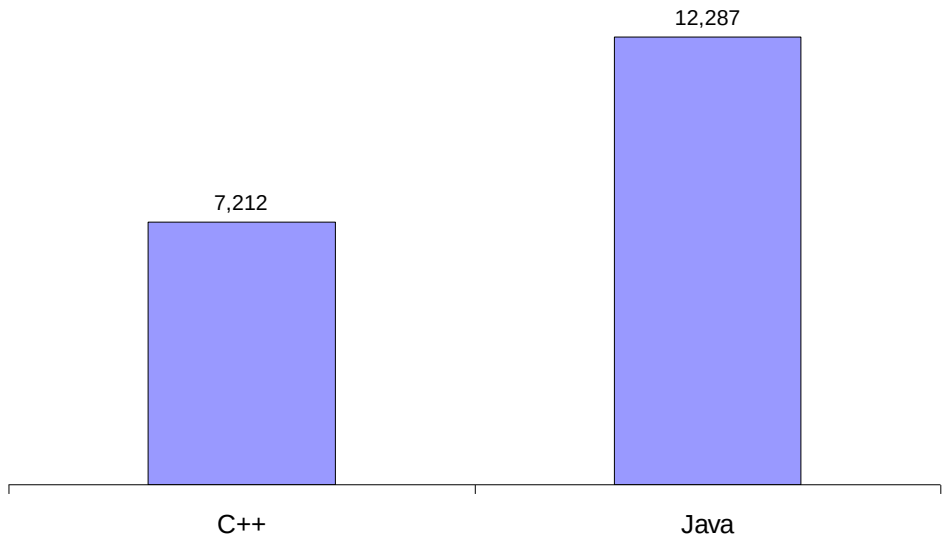
snapshot of all recorded changes and examined how entries are created and linked. Inspecting the timestamps on individual entry definitions and references, we found that links to nonexistent articles often precede creation of new articles. Also, tracking the evolution of article links allowed us to empirically validate Barabási's hypothesis on the formation of scale-free graphs through incremental growth and preferential attachment.[1] Our findings paint a picture of sustainable growth, suggesting that Wikipedia's development process delivers coverage of more and more subjects.

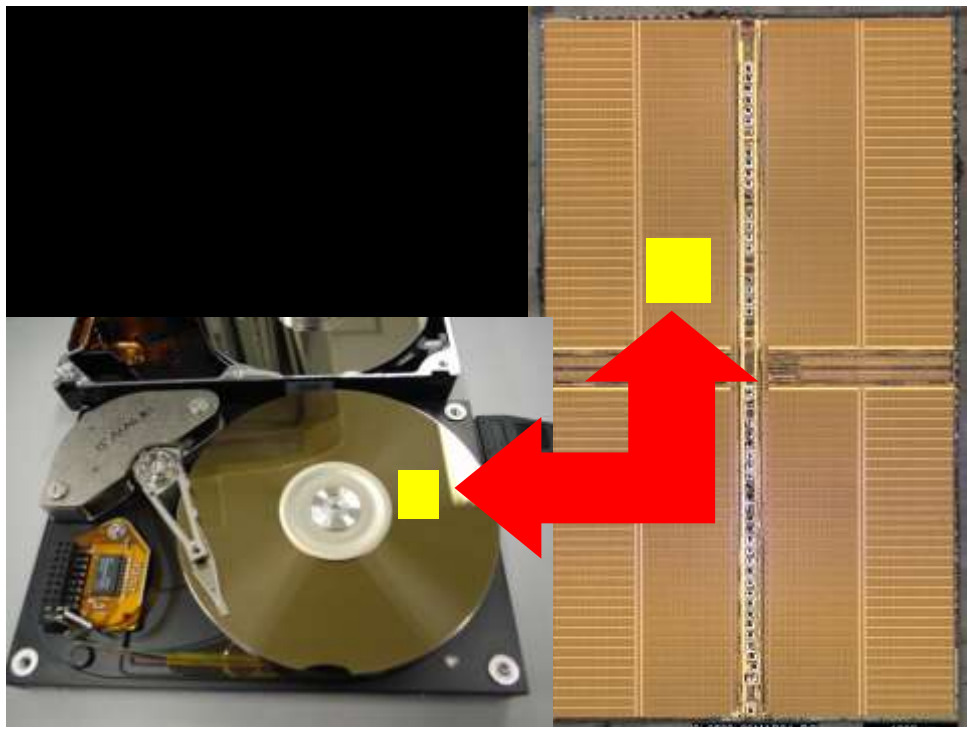
The phenomenal growth of Wikipedia is attributable to a mixture of technologies and a process of open participation. The key technology behind Wikipedia is that of a Wiki—online lightweight Web-based collaboration.[4] Wikipedia content appears online as static HTML pages, though each such page includes an edit button anyone can use to modify its content; editing

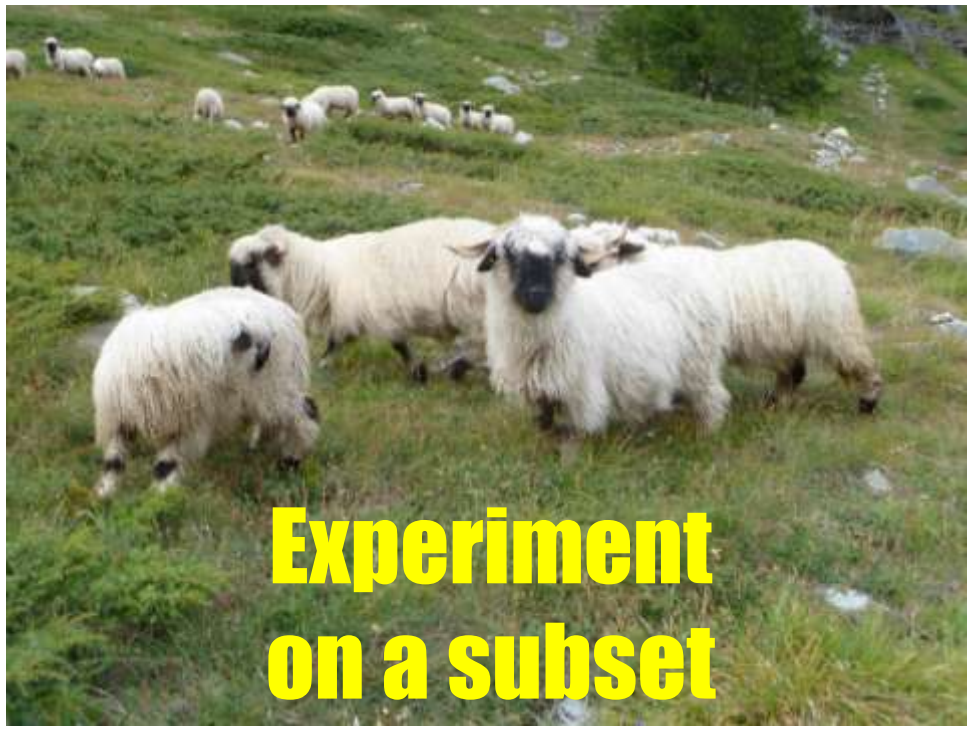


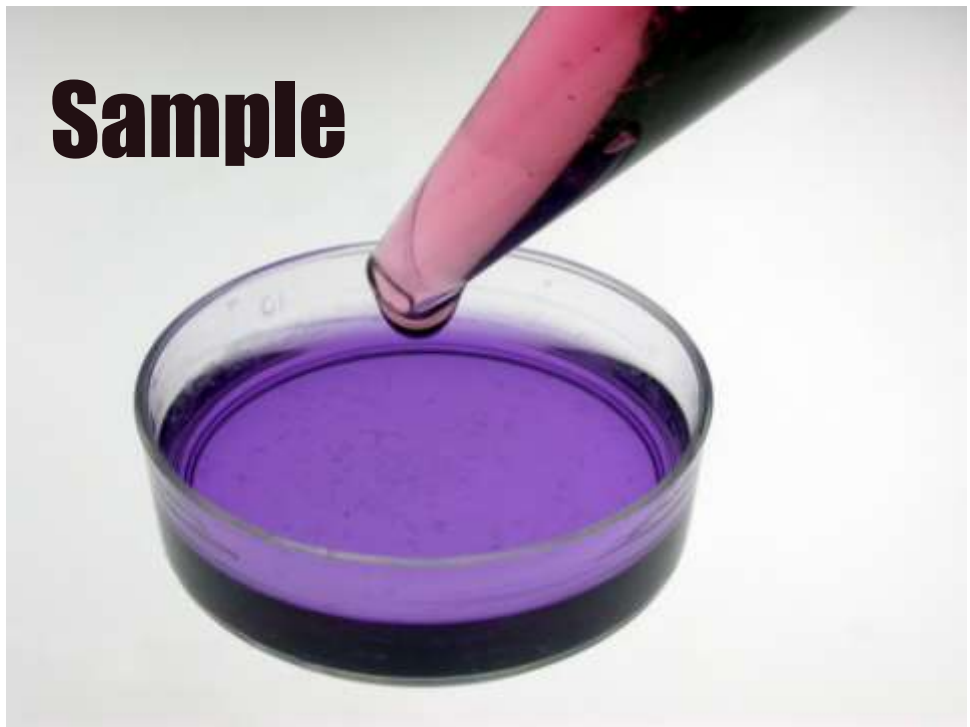

```
class RangeMap {
private:
        vector<bool> active;
public:
        static const int NMONTH = (2009 - 2001) * 12;
        RangeMap() : active(NMONTH, false) {}
};

class EntryDetails {
private:
    RangeMap defined, stub;
    vector <int> nrefs;
    string definer, referrer, firstRef;
    time_t firstDef;
    int numReferences, numContributors;
    int numRevisions, numReverts;
public:
    EntryDetails() : defined(), stub(),
nrefs(RangeMap::NMONTH, 0), firstDef(-1),
numReferences(0), numContributors(0), numRevisions(0),
numReverts(0) {}
};
```

```
class RangeMap {
    private BitSet active;
    public static final int NMONTH = (2009 - 2001) * 12;
    public RangeMap() {
        active = new BitSet(NMONTH);
    }
}

class EntryDetails {
    RangeMap defined, stub;
    ArrayList <Integer> nrefs;
    String definer, referrer, firstRef;
    Date firstDef;
    int numReferences, numContributors;
    int numRevisions, numReverts;
    public EntryDetails() {
        defined = new RangeMap();
        stub = new RangeMap();
        nrefs = new ArrayList<Integer>(RangeMap.NMONTH);
    }
}
```



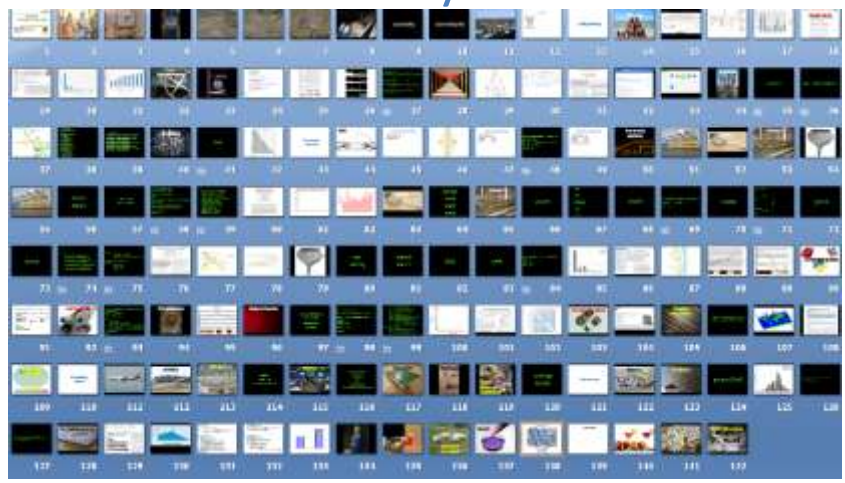



Experiment
on a subset


Sample


YAMAHA

Thank you!



http://www.spinellis.gr

@CoolSWEng

dds@aueb.gr

Image Credits

- Aero Icarus
- Chrysler Group
- Jose Maria Minarro Vivancos
- Kirti Poddar
- Lab Science Career
- Lachlan Donald
- Luigi Chiesa
- Marco40134
- Mark Sze
- Meats Meyer
- NASA
- NTSB
- PSParrot
- Q-Wing
- Rachel Kramer
- Sam Rayner
- Tim Bouwer
- Tudor Barker
- Wikimedia Commons
- William Avery, W. Heath Robinson
- kaktuslampan
- kenmainr
- max guitare
- patrasTimes.gr
- vlc team, ubuntu, Hidro
- zzpza