Large-scale source code archival, publishing, and indexing with Debsources

Stefano Zacchiroli
zack@upsilon.cc

Debian / IRILL / Université Paris Diderot

16 December 2015
Séminaire Codes Sources
Paris, France

Outline

Free Software

A program is free software if the program's users have the four essential freedoms:

- Freedom #0, to run the program, for any purpose
- Freedom #1, to study how the program works, and change it
- Freedom #2, to redistribute copies
- Freedom #3, to improve the program, and release improvements

Free Software also comes with obligations, which vary according to the license: BSD, GPL, Apache, AGPL, ...

Why bother?

Why, as scientists and teachers, should we bother about FOSS?

FOSS is *radically* changing the way software is:

- developed
- tested
- proven
- conceived
- marketed
- sold
- maintained
- taught
- deployed
- ...

The commons and FOSS

Definition (Commons)

The commons is the cultural and natural resources accessible to all members of a society, including natural materials such as air, water, and a habitable earth. These resources are held in common, not owned privately.

https://en.wikipedia.org/wiki/Commons

Definition (Software commons)

The software commons consists of all computer software which is available at little or no cost and which can be altered and reused with few restrictions. Thus all open source software and all free software are part of the commons. [...]

https://en.wikipedia.org/wiki/Software_Commons

Debian

- popular Free and Open Source Software (FOSS) distribution
- 20+ years of history
- one of the largest curated software collections

- good proxy of popular/ relevant FOSS projects
- popular subject for the Empirical Software Engineering / Mining Software Repositories scientific communities

- root of a huge derivatives ecosystem
- ≈50% of active FOSS distributions based on it (`distrowatch`)

Outline

Debsources in a nutshell

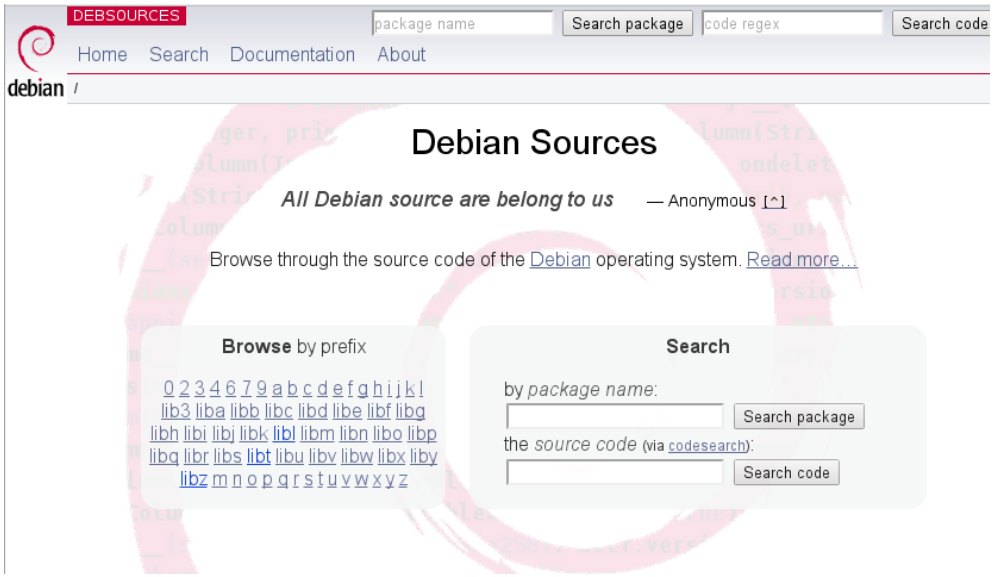
1. an infrastructure to publish Debian source code on the Web
2. a notable instance indexing *all* Debian source code to date: `http://sources.debian.net`

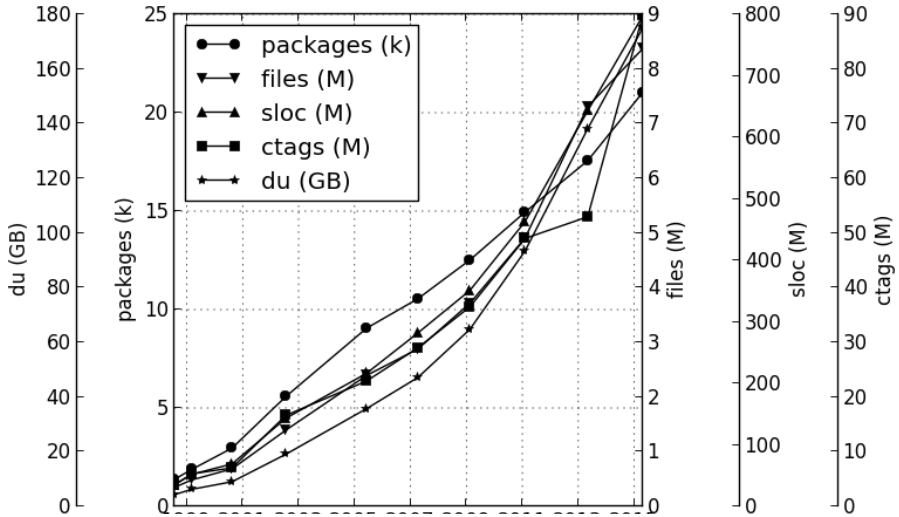
For developers:

- browse/search source code
- syntax highlighting
- pinpoint code lines, annotate

For data miners:

- Debian evolution over time
- 20+ years of FOSS history
- live change monitoring

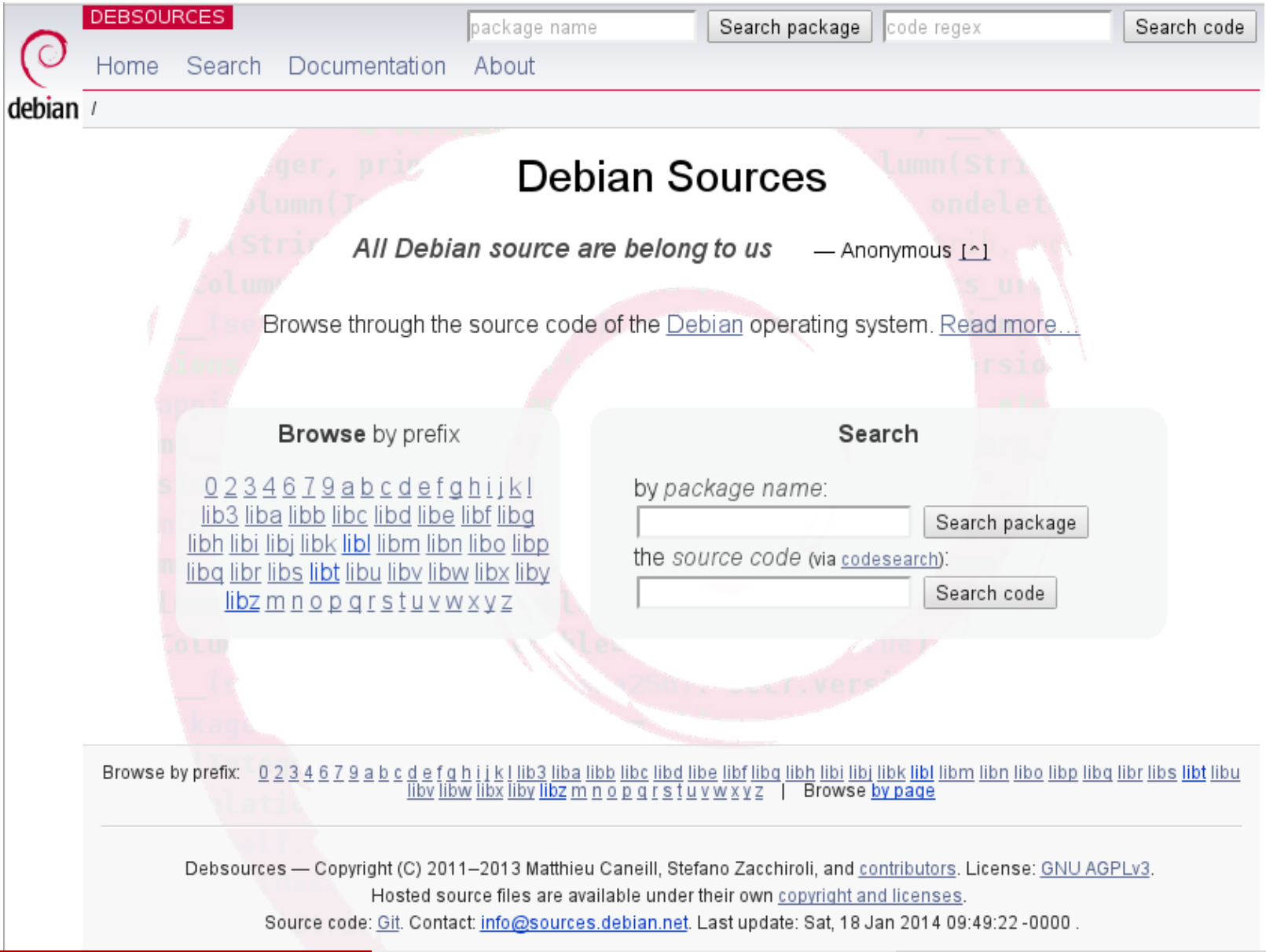




Outline

Debsources for developers

http://sources.debian.net



Features — code browsing

Package browsing: the usual suspects

- by prefix
- . . . then version selection

Code browsing:

- usual file/directory navigation
  - on the source tree obtained with `dpkg-source -x`
- HTML syntax highlighting
  - *client-side* — Javascript, but does graceful degradation
  - *file type detection* — extension + shebang, following Geany

Features — code searching

In house:

- package name search, with substring matching
- file matching given SHA256
  - also used for duplicate detection
- file defining given symbol, AKA ctags

Integrated:

Debian Code Search

Regular expression search on Debian (sid/main) source code, by Michael Stapelberg. See: `http://codesearch.debian.net/`

- search form on `sources.d.n`, which query `codesearch.d.n`
- `codesearch.d.n` result pages link back to `sources.d.n`

Features — external references

- predictable URLs
  e.g., `http://sources.debian.net/src/cowsay/3.03+dfsg1-4/cowsay`
- point to a specific line
  `http://sources.debian.net/src/cowsay/3.03+dfsg1-4/cowsay#L37`
- highlight line ranges
  `http://sources.debian.net/src/cowsay/3.03+dfsg1-4/cowsay?hl=37,39,41,43#L37`
- pop-up messages
  `http://sources.debian.net/src/cowsay/3.03+dfsg1-4/cowsay?hl=22:28&msg=22:Cowsay:Cowsay%20globals#L22`
- `<iframe>` embedding

Doc at `http://sources.debian.net/doc/url/`

Features — API

JSON-based API exposing all of the features available via the Web UI

Doc at `http://sources.debian.net/doc/api/`

Outline

Software evolution [in the large]

In software engineering (more specifically: in software maintenance), software evolution refers to the process of repeatedly updating software, for various reasons, *after* the initial development.

- active area of SWE research since the 70s
- seminal works: the mythical man month, Lehman's laws

FOSS, and distribution specifically, allows for a new scale of software evolution studies:

> *"Software evolution in the large"*
>
> — *Gonzalez-Barahona et. al, 2009*

The study of software evolution, at the scale of software collections, at the granularity they allow (e.g., releases of individual software components).

On studying software collections

Pros

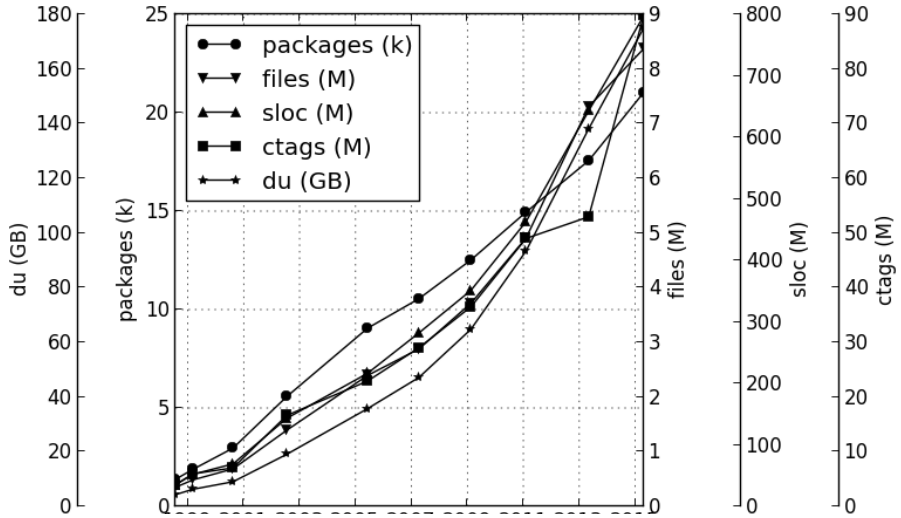
- relevant/popular software distribution model
- long lives (e.g., decades)
- uniform access to the history of contained software
- help with (researcher) selection bias

Cons

- *ad hoc* software ecosystems
- homegrown tools, conventions, social norms

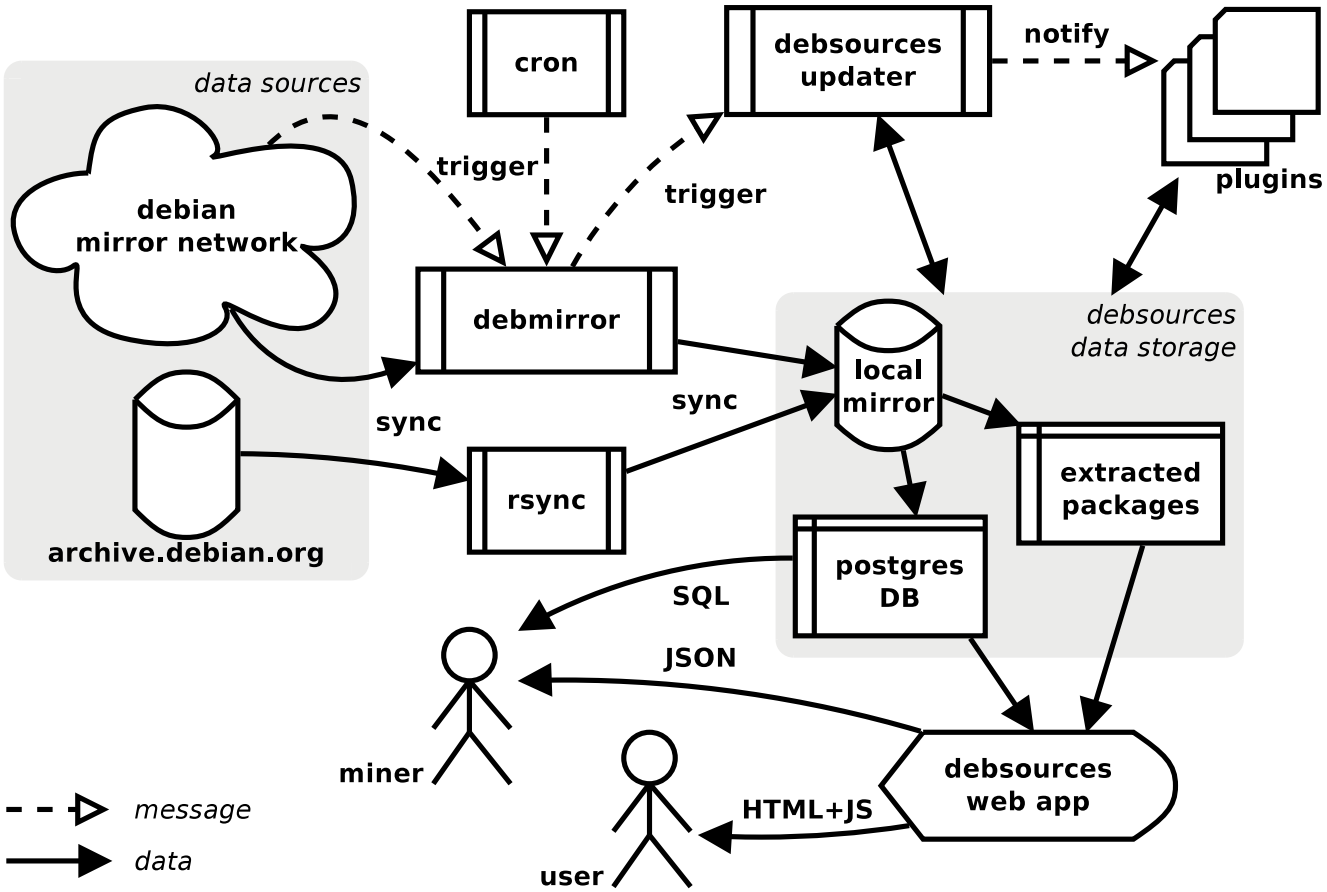
Debsources for researchers / data miners

- obvservation point on Debian macro-level evolution
- 20+ years of history
- both live and perennial monitoring



Debsources eases macro-level software evolution studies on FOSS as a whole, using Debian as a proxy.

Architecture



Debsources does the heavy lifting of maintaining a general purpose, always up to date storage for Debian source code, enabling plugin authors to focus on data extraction.

Available plugins

- disk usage
- `sloccount`
- ctags (functions, classes, types, etc.)
- checksums (SHA256)
- file count (implicit)

Self-assessment: very little effort needed to write plugins for popular source code metrics.

Most complex plugin (ctags): ≈100 SLOCs

Plugin — example (sloccount)

```
def add_package(session, pkg, pkgdir, file_table):          # plugin excerpt
    if 'hooks.fs' in conf['backends']:
        if not os.path.exists(slocfile):  # run sloccount only if needed
            try:
                cmd = ['sloccount'] + SLOCCOUNT_FLAGS + [pkgdir]
                with open(slocfile_tmp, 'w') as out:
                    subprocess.check_call(cmd, stdout=out,
                                          stderr=subprocess.STDOUT)
            except subprocess.CalledProcessError:
                if not grep(['^SLOC total is zero,', slocfile_tmp]):
                        # rationale: sloccount fails
                    raise  # when it can't find source code
            finally:
                os.rename(slocfile_tmp, slocfile)
    if 'hooks.db' in conf['backends']:
        slocs = parse_sloccount(slocfile)
        db_package = dbutils.lookup_package(session, pkg['package'],
                                            pkg['version'])
        if not session.query(SlocCount).filter_by(package_id=db_package.id)\
                                       .first():
            # ASSUMPTION: if *a* loc count of this package has already been
            # added to the db in the past, then *all* of them have
            for (lang, locs) in slocs.iteritems():
                sloccount = SlocCount(db_package, lang, locs)
                session.add(sloccount)
```

sources.debian.net — coverage

Covered releases:

- all stable releases from Debian Hamm (1997) to Jessie (2015)
- LTS security updates
- development releases: testing, unstable, experimental, ...

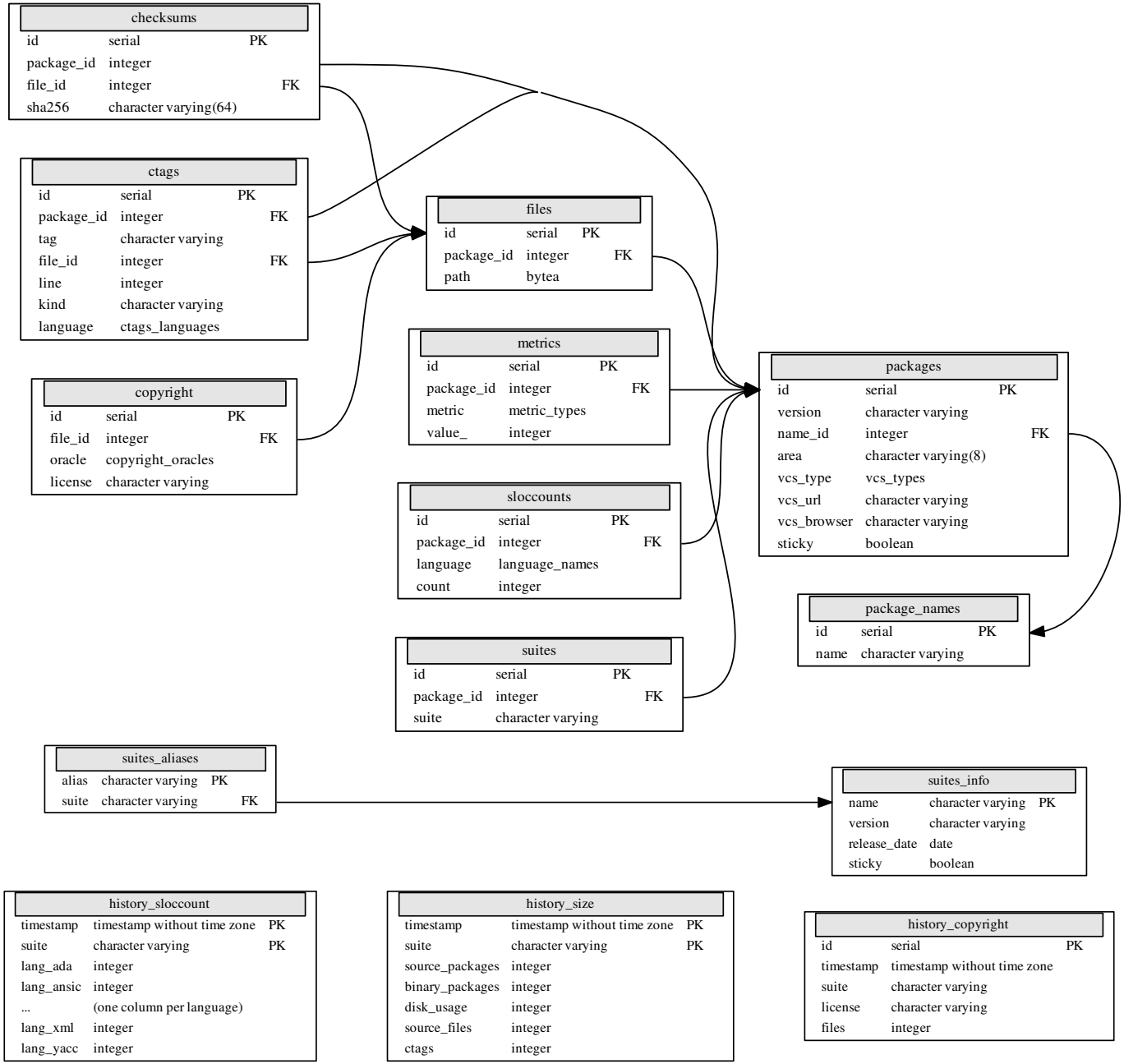
Update frequency: 4 times a day (at each Debian archive change)

Overall content: (Oct 2015)

- 790 GB of source code
- 45 M source code files
  - 18 M *distinct* SHA256
- 4.3 B lines of code
- 485 M developer-defined symbols (ctags)

more stats at
http://sources.debian.net/stats/

Debsources — DB schema



sources.debian.net — dataset

Table: Debsources dataset table sizes[1]

table	tuples	disk size
checksums*	44,050,664	4,589 MB
copyright*	48,854,056	2,601 MB
ctags*	466,225,638	31 GB
files	44,187,278	3,312 MB
history_size	25,547	1,672 KB
history_sloccount	24,624	4,488 KB
metrics*	96,994	4,200 KB
package_names	31,880	1,544 KB
packages	96,994	11 MB
sloccounts*	344,465	15 MB
suites	147,321	6,416 KB
suites_aliases	11	16 KB
suites_info	23	16 KB

[1] data snapshot, 17 September 2015

sources.debian.net — dataset (cont.)

Stefano Zacchiroli
The Debsources Dataset
Zenodo
http://dx.doi.org/10.5281/zenodo.16106

License: Creative Commons Attribution Share-Alike 4.0

Debsources dataset — reproducibility

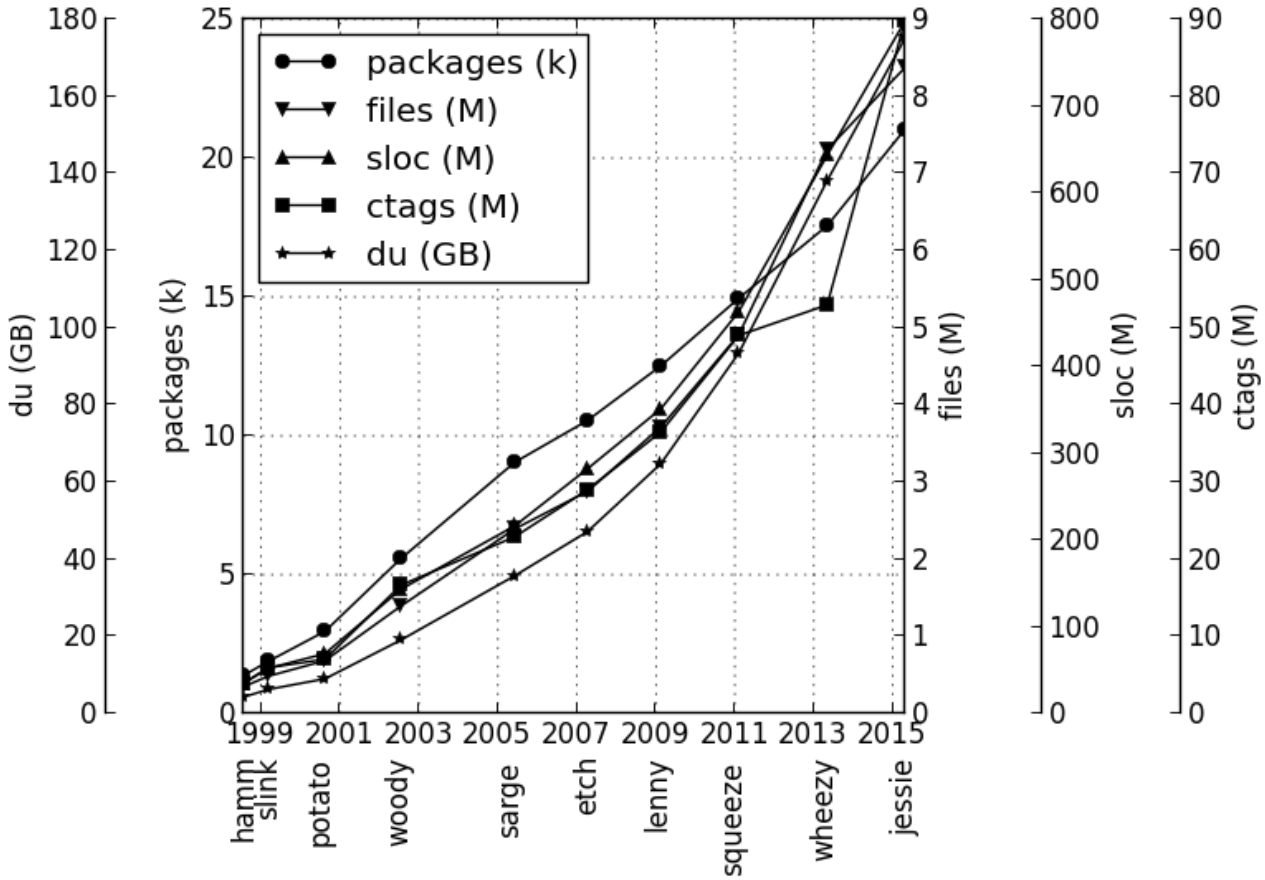
1. deploy the Debsources software
2. point it to a nearby Debian mirror
   - optional: request push update notifications
3. trigger 1st update run `$ bin/debsources-update`

4. mirror `archive.debian.org` `$ rsync`
5. inject all archived suites `$ bin/debsources-suite-archive add`

- processing time (I/O bound):[2] ≈14 days (8 for live suites)
- disk usage: ≈1.1 TB
  - 189 GB (mirror) + 788 GB (extracted packages) + 101 GB (DB)

[2] data snapshot, 17 September 2015

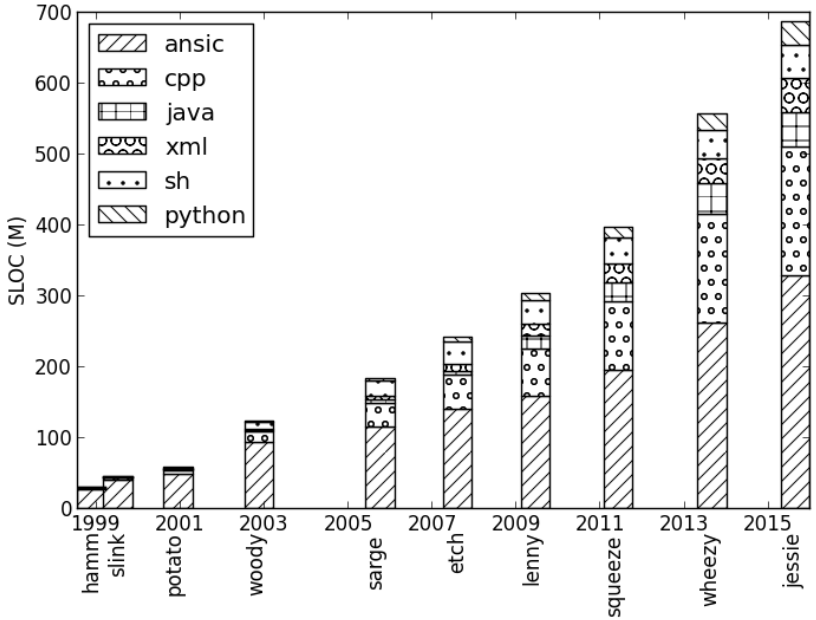
Highlight #1: total size

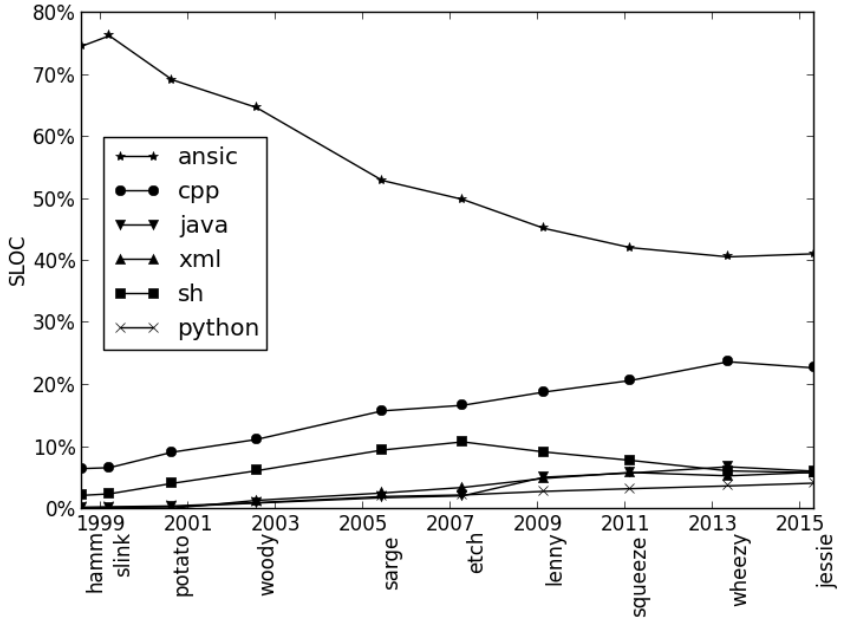


- correlation confirms Herraiz et. al, 2006 & 2007
- exception: package count (distro-level refactoring?)
- pre-*etch* (2007): growth rate slows down (allegedly, due to complexity ceiling)
- post-*etch*: growth rate increases

Highlight #2: programming languages

top-5 most popular programming languages in Debian over time





Recent trends (post-*etch*, 2007):

- C still leads, steady (absolute) growth
- C stops losing (relative) ground to C++
- decrease of Perl/Shell popularity
- Python rises (more maintainable glue code?)
- Lisp halves its popularity
- Java no longer under-represented

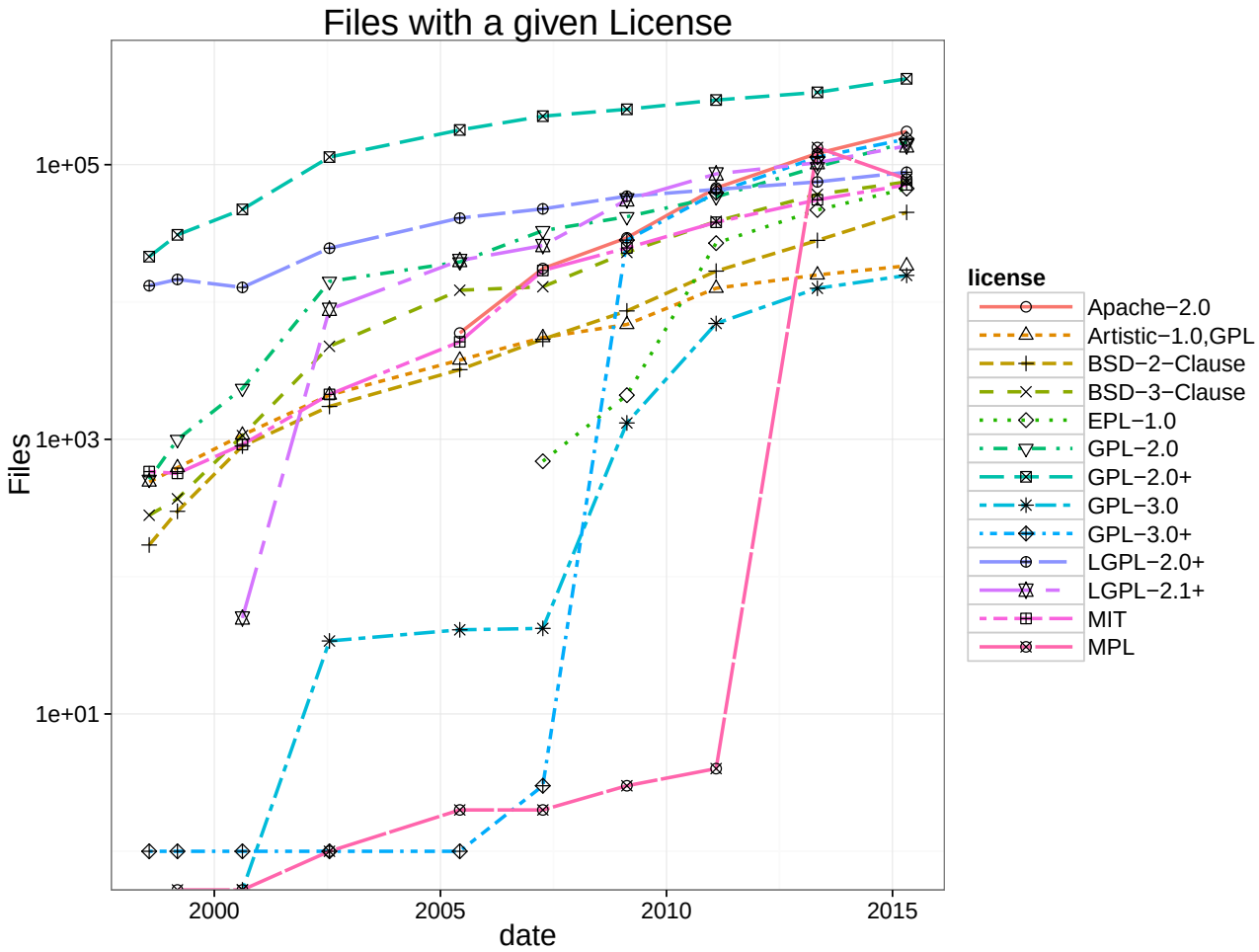
Highlight #3: package maintenance

Changes between Debian releases: ‘c’ for common, ‘u’ for unchanged (upstream), and ’m’ for modified packages (common \ unchanged):

from \ *to*	slink	potato	woody	sarge	etch	lenny	squeeze	wheezy	jessie
hamm	1324c 842u	1198c 463u	1079c 270u	958c 175u	864c 148u	782c 124u	719c 100u	670c 81u	649c 73u
slink		1657c 742u	1455c 384u	1281c 252u	1155c 210u	1037c 172u	941c 136u	881c 113u	852c 101u
potato			2456c 935u	2118c 551u	1881c 436u	1683c 352u	1497c 271u	1399c 220u	1348c 201u
woody				4588c 1688u	3953c 1156u	3497c 908u	3018c 633u	2786c 520u	2648c 458u
sarge					7671c 3832u	6828c 2597u	5896c 1717u	5349c 1367u	5042c 1164u
etch						9230c 4578u	8033c 2906u	7212c 2203u	6778c 1813u
lenny							10823c 5271u	9624c 3673u	8999c 2928u
squeeze								13098c 6802u	12201c 4890u
wheezy									16160c 8427u

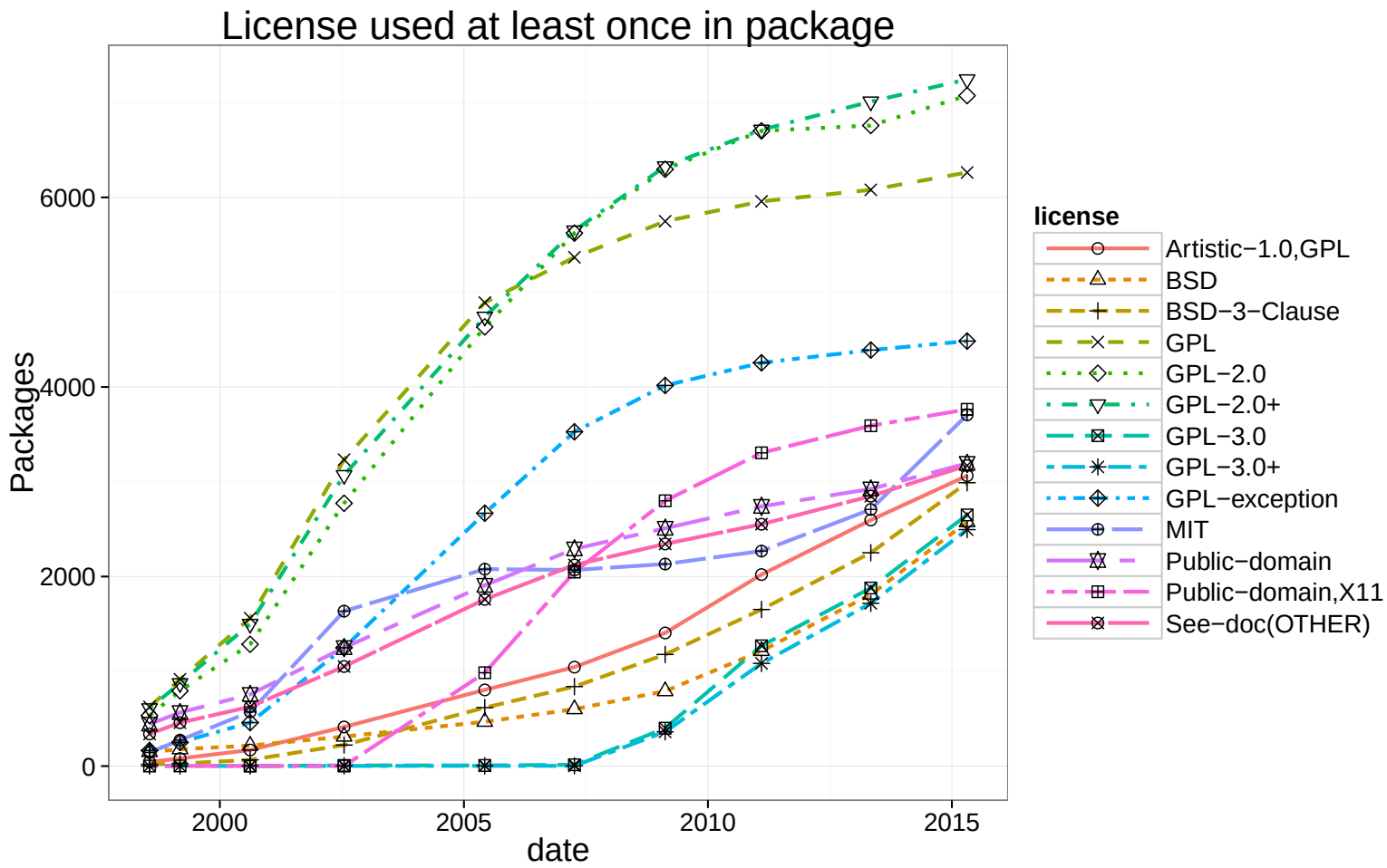
	from previous suite to potato	woody	sarge	etch	lenny	squeeze	wheezy	jessie
modified pkgs	1305m	3127m	4462m	2879m	3287m	4128m	4466m	4881m
changed files per pkg	64.4%	65.3%	67.5%	58.9%	59.8%	60.4%	57.3%	54.7%

Highlight #4: license usage



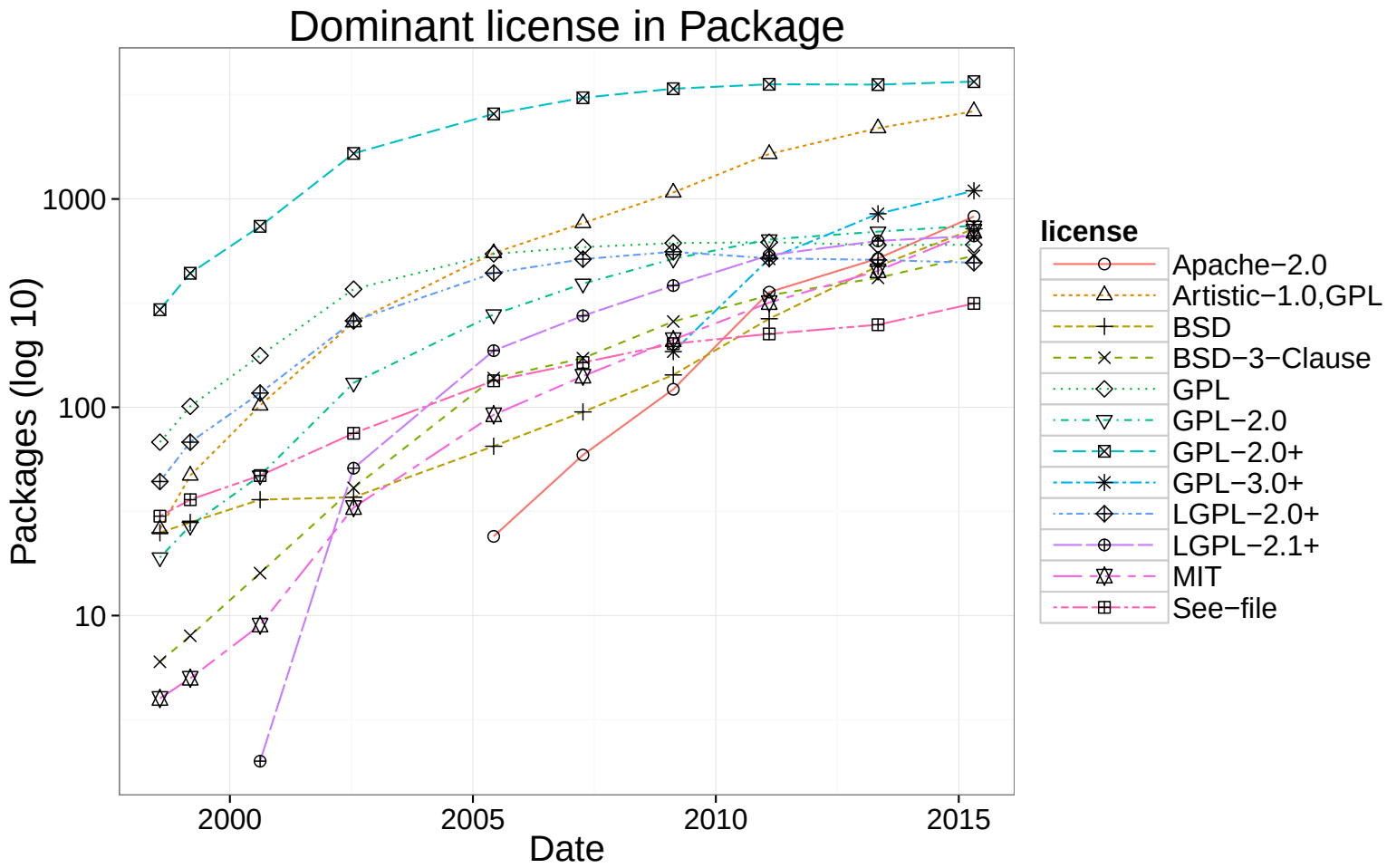
- the licenses census problem is difficult to define
- data obtained using FOSSology on the full Debsources dataset
- the alleged decline of copyleft licensing is *not* evident here

Highlight #4: license usage (cont.)



- the licenses census problem is difficult to define
- data obtained using FOSSology on the full Debsources dataset
- the alleged decline of copyleft licensing is *not* evident here

Highlight #4: license usage (cont.)



- the licenses census problem is difficult to define
- data obtained using FOSSology on the full Debsources dataset
- the alleged decline of copyleft licensing is *not* evident here

Outline

What made Debsources possible?

Source code:

- availability

  "The commons is the cultural and natural resources accessible to all members of a society [. . .]"

- licensing terms

  "The software commons consists of all computer software which is available at little or no cost and which can be altered and reused with few restriction"

- organization
  - package & version namespaces
  - intrinsic identifiers (e.g., SHA256)

What would it take to do the same for the entire software commons? And what can we do with it once we have it?

What made Debsources possible?

Source code:

- availability

  "The commons is the cultural and natural resources accessible to all members of a society [. . .]"

- licensing terms

  "The software commons consists of all computer software which is available at little or no cost and which can be altered and reused with few restriction"

- organization
  - package & version namespaces
  - intrinsic identifiers (e.g., SHA256)

What would it take to do the same for the entire software commons? And what can we do with it once we have it?

What made Debsources possible?

Source code:

- availability

  "The commons is the cultural and natural resources accessible to all members of a society [. . .]"

- licensing terms

  "The software commons consists of all computer software which is available at little or no cost and which can be altered and reused with few restriction"

- organization
  - package & version namespaces
  - intrinsic identifiers (e.g., SHA256)

What would it take to do the same for the entire software commons? And what can we do with it once we have it?

What made Debsources possible?

Source code:

- availability

  "The commons is the cultural and natural resources accessible to all members of a society [. . .]"

- licensing terms

  "The software commons consists of all computer software which is available at little or no cost and which can be altered and reused with few restriction"

- organization
  - package & version namespaces
  - intrinsic identifiers (e.g., SHA256)

What would it take to do the same for the entire software commons? And what can we do with it once we have it?

Like all digital information, software is fragile

An example is worth a thousand words...

let's see a few

Inconsiderate or malicious loss of code

The Year 2000 Bug . . . uncovered an inconvenient truth

in 1999, an estimated 40% of companies had either *lost*, or thrown away the original source code for their systems!

CodeSpaces: source code hosting, 2007-2014

Murder in the Amazon cloud

The demise of Code Spaces at the hands of an attacker shows that, in the cloud, off-site backups and separation of services could be key to survival

InfoWorld | Jun 23, 2014

Yes, for *seven years* all seemed ok.

No, they did not recover the data.

Inconsiderate or malicious loss of code

The Year 2000 Bug . . . uncovered an inconvenient truth

in 1999, an estimated 40% of companies had either *lost*, or thrown away the original source code for their systems!

CodeSpaces: source code hosting, 2007-2014

Murder in the Amazon cloud

The demise of Code Spaces at the hands of an attacker shows that, in the cloud, off-site backups and separation of services could be key to survival

InfoWorld | Jun 23, 2014

Yes, for *seven years* all seemed ok.
No, they did not recover the data.

Business-driven loss of code support: Google

Posted: Thursday, March 12, 2015

When we started the Google Code project hosting service in 2006, the world of project hosting was limited. We were worried about reliability and stagnation, so we took action by giving the open source community another option to choose from. Since then, we've seen a wide variety of better project hosting services such as GitHub and Bitbucket bloom. Many projects moved away from Google Code to those other systems. To meet developers where they are, we ourselves migrated nearly a thousand of our own open source projects from Google Code to GitHub.

As developers migrated away from Google Code, a growing share of the remaining projects were spam or abuse. Lately, the administrative load has consisted almost exclusively of abuse management. After profiling non-abusive activity on Google Code, it has become clear to us that the service simply isn't needed anymore.

Beginning today, we have disabled new project creation on Google Code. We will be shutting down the service about 10 months from now on January 25th, 2016. Below, we provide links to migration tools designed to help you move your projects off of Google Code. We will also make ourselves available over the next three months to those projects that need help migrating from Google Code to other hosts.

- March 12, 2015 - New project creation disabled.
- August 24, 2015 - The site goes read-only. You can still checkout/view project source, issues, and wikis.
- January 25, 2016 - The project hosting service is closed. You will be able to download a tarball of project source, issues, and wikis. These tarballs will be available throughout the rest of 2016.

Business-driven loss of code support: Gitorious

From: Rolf Bjaanes <rolf@gitorious.org>
To: zack@upsilon.cc
Subject: Gitorious.org is dead, long live Gitorious.org
Message-Id:
<30589491.20150416155909.552fdc4d164758.16579271@mail141.wdc04

Hi zacchiro,

I'm Rolf Bjaanes, CEO of Gitorious, and you are receiving this email because you have a user on gitorious.org. As you may know, Gitorious was acquired by GitLab [1] about a month ago *(NDLR: 3/3/2015)*, and we announced that Gitorious.org would be shutting down at the end of May, 2015.

[...]

... Rolf

Disruption of the *web of reference*

Web links *are not* permanent (even *permalinks*)}

there is no general guarantee that a URL. . . which at one time points to a given object continues to do so
T. Berners-Lee et al. Uniform Resource Locators. RFC 1738.

404

URLs used in articles *decay*!

Analysis of *IEEE Computer* (Computer), and the *Communications of the ACM* (CACM): 1995-1999

- the *half-life* of a referenced URL *is approximately 4 years* from its publication date D. Spinellis. The Decay and Failures of URL References.
  Communications of the ACM, 46(1):71-77, January 2003.

The Software Heritage Project

Software Heritage

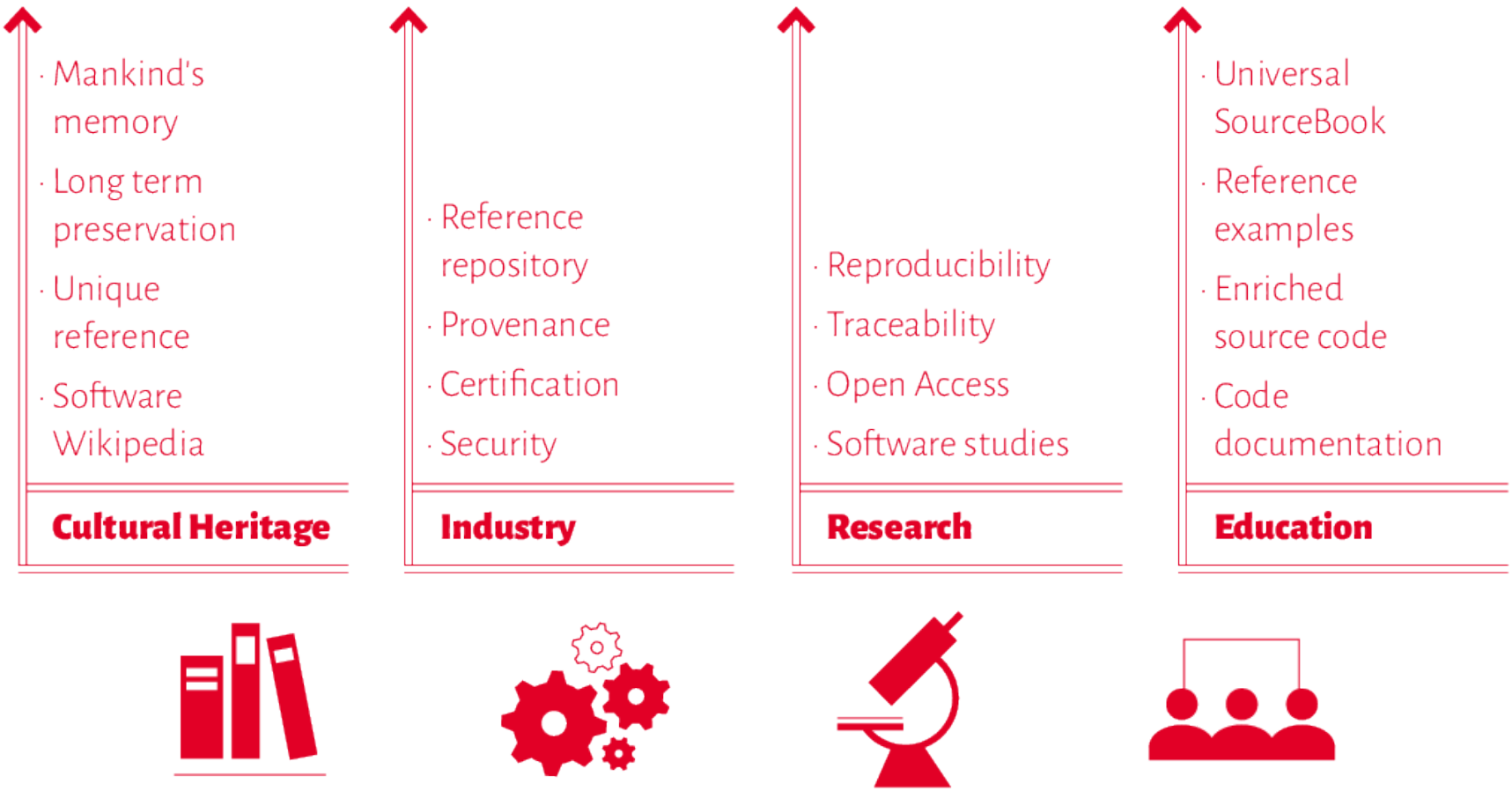
PRESERVING TECHNICAL KNOWLEDGE

Our mission

Collect, *organise*, *preserve* and *share all the software* that lies at the heart of our culture and our society.

Joint work with Roberto Di Cosmo

We are working on the foundations

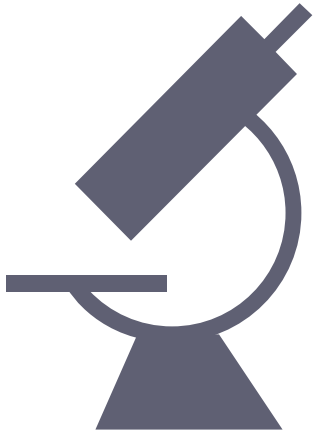


Preserving the world's Software heritage

A structured archive of all of the world's software

- preserve humanity's technological and scientific *knowledge*
- enable continued *access* to all digital documents and information
- building block for *thematic* portals and collections

Supporting more accessible and reproducible Science

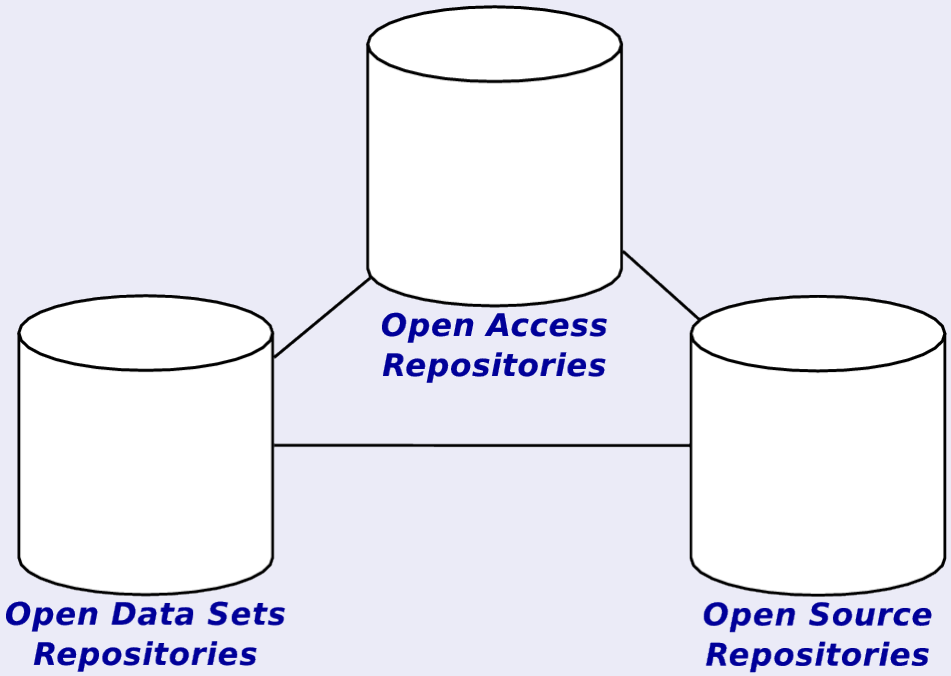


A global library referencing all software used in all research fields

- completes the infrastructure for Open Access in Science
- provides intrinsic persistent identifiers needed for scientific reproducibility
- enables large scale, verifiable Software Studies

The Knowledge Conservancy Magic Triangle

The Knowledge Conservancy Magic Triangle

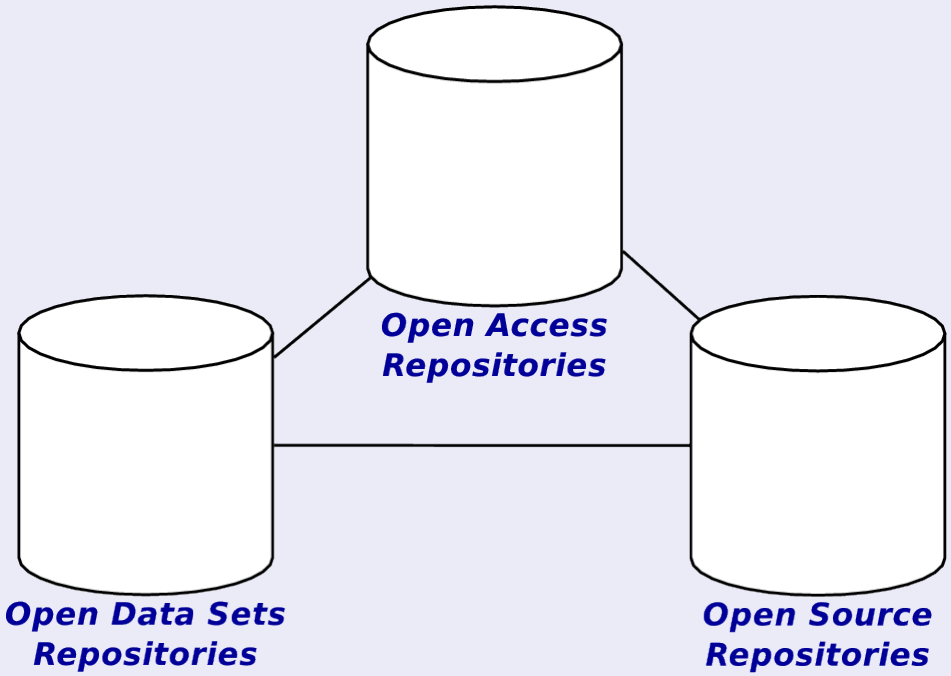


Legenda (links are important!)

- articles: ArXiv, HAL, ...
- data: Zenodo, ...
- software: *Software Heritage* to the rescue

The Knowledge Conservancy Magic Triangle

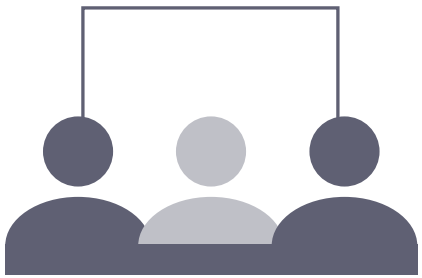
The Knowledge Conservancy Magic Triangle



Legenda (links are important!)

- articles: ArXiv, HAL, ...
- data: Zenodo, ...
- software: *Software Heritage* to the rescue

Fostering wider education to computing



A global source referencing all software

- a *SourceBook* for technological *education*
- intrinsic persistent identifiers for stable course materials
- extensive access to real-world documentation

We need your help

make it easy to integrate

- in development workflow
- in publishing workflow

make it ok to integrate, from the legal point of view

- make licences explicit
- make licences of dependencies explicit

make it useful for research

- contribute to the API

make it sustainable

- support/sponsorship
- open process
- collaboration

We need your help

make it easy to integrate

- in development workflow
- in publishing workflow

make it ok to integrate, from the legal point of view

- make licences explicit
- make licences of dependencies explicit

make it useful for research

- contribute to the API

make it sustainable

- support/sponsorship
- open process
- collaboration

We need your help

make it easy to integrate

- in development workflow
- in publishing workflow

make it ok to integrate, from the legal point of view

- make licences explicit
- make licences of dependencies explicit

make it useful for research

- contribute to the API

make it sustainable

- support/sponsorship
- open process
- collaboration

We need your help

make it easy to integrate

- in development workflow
- in publishing workflow

make it ok to integrate, from the legal point of view

- make licences explicit
- make licences of dependencies explicit

make it useful for research

- contribute to the API

make it sustainable

- support/sponsorship
- open process
- collaboration

Debsources

- http://sources.debian.net

Matthieu Caneill, Stefano Zacchiroli
Debsources: Live and Historical Views on Macro-Level Software Evolution
ESEM 2014: 8th International Symposium on Empirical Software Engineering and Measurement

Stefano Zacchiroli
The Debsources Dataset: Two Decades of Debian Source Code Metadata.
MSR 2015: The 12th Working Conference on Mining Software Repositories

Matthieu Caneill, Daniel M. Germán, Stefano Zacchiroli
The Debsources Dataset: Two Decades of Free and Open Source Software Metadata
Empirical Software Engineering
Springer *(to appear)*

Software Heritage

- mailing list: swh-science@inria.fr
  https://sympa.inria.fr/sympa/info/swh-science

About me

Stefano Zacchiroli <zack@upsilon.cc> http://upsilon.cc/zack

Outline

4. Software Heritage design choices

Free and Open Source Software is crucial

D. Rosenthal, EUDAT, 9/2014

you have to do [digital preservation] with open-source software; closed-source preservation has the same fatal "just trust me" aspect that closed-source encryption (and cloud storage) suffer from.

recommendation

our preferred platform(s) should:

- provide full details on their architecture
- make available all the source code used
- use open standards
- encourage a collaborative development process

Free and Open Source Software is crucial

D. Rosenthal, EUDAT, 9/2014

you have to do [digital preservation] with open-source software; closed-source preservation has the same fatal "just trust me" aspect that closed-source encryption (and cloud storage) suffer from.

recommendation

our preferred platform(s) should:

- provide full details on their architecture
- make available all the source code used
- use open standards
- encourage a collaborative development process

Web links *are not* permanent (even *permalinks*)

T. Berners-Lee et al. Uniform Resource Locators. RFC 1738.

Users should beware that there is no general guarantee that a URL which at one time points to a given object continues to do so, and does not even at some later time point to a different object due to the movement of objects on servers.

The Decay and Failures of URL References

half life of web references is 4 years

Diomidis Spinellis, CACM 2003

recommendation

our preferred platform(s) should:

- provide *intrinsic* resource identifiers
- *avoid* volatile identifiers like DOI or URLs

Web links *are not* permanent (even *permalinks*)

T. Berners-Lee et al. Uniform Resource Locators. RFC 1738.

Users should beware that there is no general guarantee that a URL which at one time points to a given object continues to do so, and does not even at some later time point to a different object due to the movement of objects on servers.

The Decay and Failures of URL References

half life of web references is 4 years

Diomidis Spinellis, CACM 2003

recommendation

our preferred platform(s) should:

- provide *intrinsic* resource identifiers
- *avoid* volatile identifiers like DOI or URLs

Web links *are not* permanent (even *permalinks*)

T. Berners-Lee et al. Uniform Resource Locators. RFC 1738.

Users should beware that there is no general guarantee that a URL which at one time points to a given object continues to do so, and does not even at some later time point to a different object due to the movement of objects on servers.

The Decay and Failures of URL References

half life of web references is 4 years

Diomidis Spinellis, CACM 2003

recommendation

our preferred platform(s) should:

- provide *intrinsic* resource identifiers
- *avoid* volatile identifiers like DOI or URLs

Replication is the key

Thomas Jefferson, February 18, 1791

. . . let us save what remains: not by vaults and locks which fence them from the public eye and use in consigning them to the waste of time, but by such a multiplication of copies, as shall place them beyond the reach of accident.

recommendation

our preferred platform(s) should:

- provide easy means for making copies
- encourage the growth of a mirror network (like ArXiv did)

Replication is the key

Thomas Jefferson, February 18, 1791

. . . let us save what remains: not by vaults and locks which fence them from the public eye and use in consigning them to the waste of time, but by such a multiplication of copies, as shall place them beyond the reach of accident.

recommendation

our preferred platform(s) should:

- provide easy means for making copies
- encourage the growth of a mirror network (like ArXiv did)

Caring for the long term

not just a project

projects have limited time-frame

not commercial

business interests come and go

a shared concern

- cultural heritage
- scientific infrastructure
- industrial infrastructure

Unix philosophy

do *one* thing, do it *well*