Software Heritage
Preserving the Free Software Commons

Stefano Zacchiroli

University Paris Diderot / IRILL / Inria
zack@upsilon.cc

14 October 2016
UPMC, M2 Informatique STL
Paris, France

Software Heritage

Outline

1. The Free Software Commons
2. Software is fragile
3. Software Heritage
4. Approach
5. Current status
6. Selected research challenges
7. Conclusion

Free Software

Definition (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 computer scientists/teachers/students, should we bother about Free/Open Source Software (FOSS)?

Why bother?

Why, as computer scientists/teachers/students, should we bother about Free/Open Source Software (FOSS)?

FOSS has *radically changed* 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 [software] commons.* [...]

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

But *where* is this commons?

But *where* is this commons?

Fashion victims

- many disparate development platforms
- a myriad places where distribution may happen
- projects tend to migrate from one place to the other over time

But *where* is this commons?

Fashion victims

- many disparate development platforms
- a myriad places where distribution may happen
- projects tend to migrate from one place to the other over time

One place to bind them…

… where can we find, track and search *all* source code?

Software is Pervasive

At the heart of *our society*

- communication, entertainment
- administration, finance
- health, energy, transportation
- education, research, politics
- ...

Software is Pervasive

At the heart of *our society*

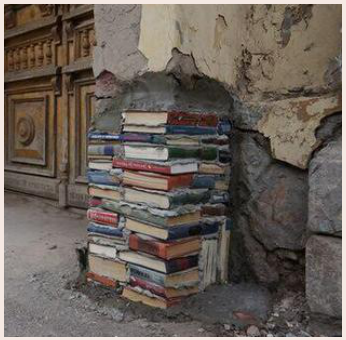
- communication, entertainment
- administration, finance
- health, energy, transportation
- education, research, politics
- …

At the heart of *technology*

- house appliances ≈ 10M SLOC
- phones ≈ 20M SLOC, *cars* ≈ 100M SLOC
- Internet of things, …

Software is Knowledge

Key mediator for accessing *all* information (c) Banski

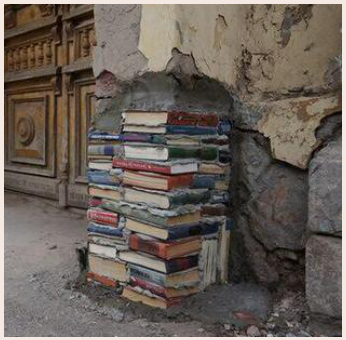


Information is a main pillar of our modern societies.

> *Absent an ability to correctly interpret digital information, we are left with […] "rotting bits" […] of no value.*
>
> *Vinton G. Cerf IEEE 2011*

Software is Knowledge

Key mediator for accessing *all* information (c) Banski



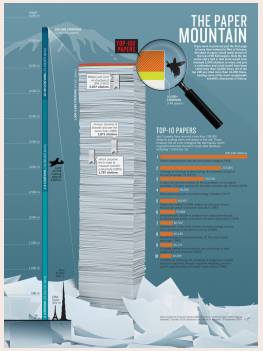
Information is a main pillar of our modern societies.

> *Absent an ability to correctly interpret digital information, we are left with […] "rotting bits" […] of no value.*
>
> *Vinton G. Cerf IEEE 2011*

Software is *an essential component* of modern scientific research

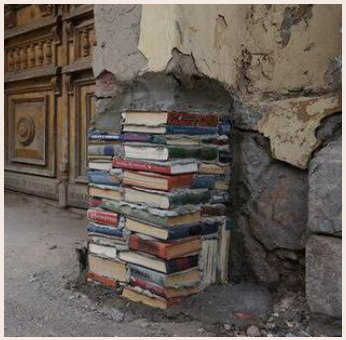
> *[…] the vast majority describe experimental methods or sofware that have become essential in their fields.*

Top 100 papers (Nature, October 2014)



Software is Knowledge

Key mediator for accessing *all* information (c) Banski



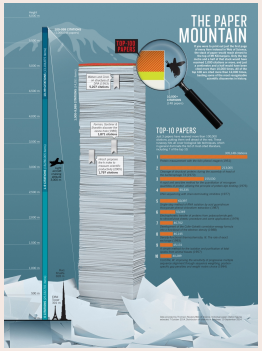
Information is a main pillar of our modern societies.

> *Absent an ability to correctly interpret digital information, we are left with […] "rotting bits" […] of no value.*
>
> *Vinton G. Cerf IEEE 2011*

Software is *an essential component* of modern scientific research

> *[…] the vast majority describe experimental methods or sofware that have become essential in their fields.*

Top 100 papers (Nature, October 2014)



Bottomline: Sofware embodies our *Knowledge* and *Cultural Heritage*

It *must* be *collected*, *preserved*, *referenced* and made *accessible*!

Outline

1. The Free Software Commons
2. Software is fragile
3. Software Heritage
4. Approach
5. Current status
6. Selected research challenges
7. Conclusion

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.

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

+1 377 Tweet 1,210 Like 404

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.

Google will continue to provide Git and Gerrit hosting for certain projects like Android and Chrome. We will also continue maintaining our mirrors of projects like Eclipse, kernel.org and others.

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: <30589491.20150416155909.552fdc4d164

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 *(note: 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.

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.

Similar findings in Lawrence, S. et al. *Persistence of Web References in Scientific Research*, IEEE Computer, 34(2), pp. 26–31, 2001.

We are at a turning point

Software is

- an *essential component* of modern scientific research
- a *key mediator* for accessing *all* information
- at the heart of our society

And yet… we are losing it, and/or traces of it…

We are at a turning point

Software is

- an *essential component* of modern scientific research
- a *key mediator* for accessing *all* information
- at the heart of our society

And yet… we are losing it, and/or traces of it…

It's time to take action!

Outline

1. The Free Software Commons
2. Software is fragile
3. Software Heritage
4. Approach
5. Current status
6. Selected research challenges
7. Conclusion

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Our mission

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

Past, present and future

Preserving the past, *enhancing* the present, *preparing* the future.

Our mission in our logo

Collect

We collect software, all of it.

Preserve

We preserve software, because it embodies our technical and scientific knowledge.

Share

We will index, organise, and make widely accessible all the Software we collect

Software Source Code is *different*

"Programs must be written for people to read, and only incidentally for machines to execute." Harold Abelson, Structure and Interpretation of Computer Programs

Distinguishing features

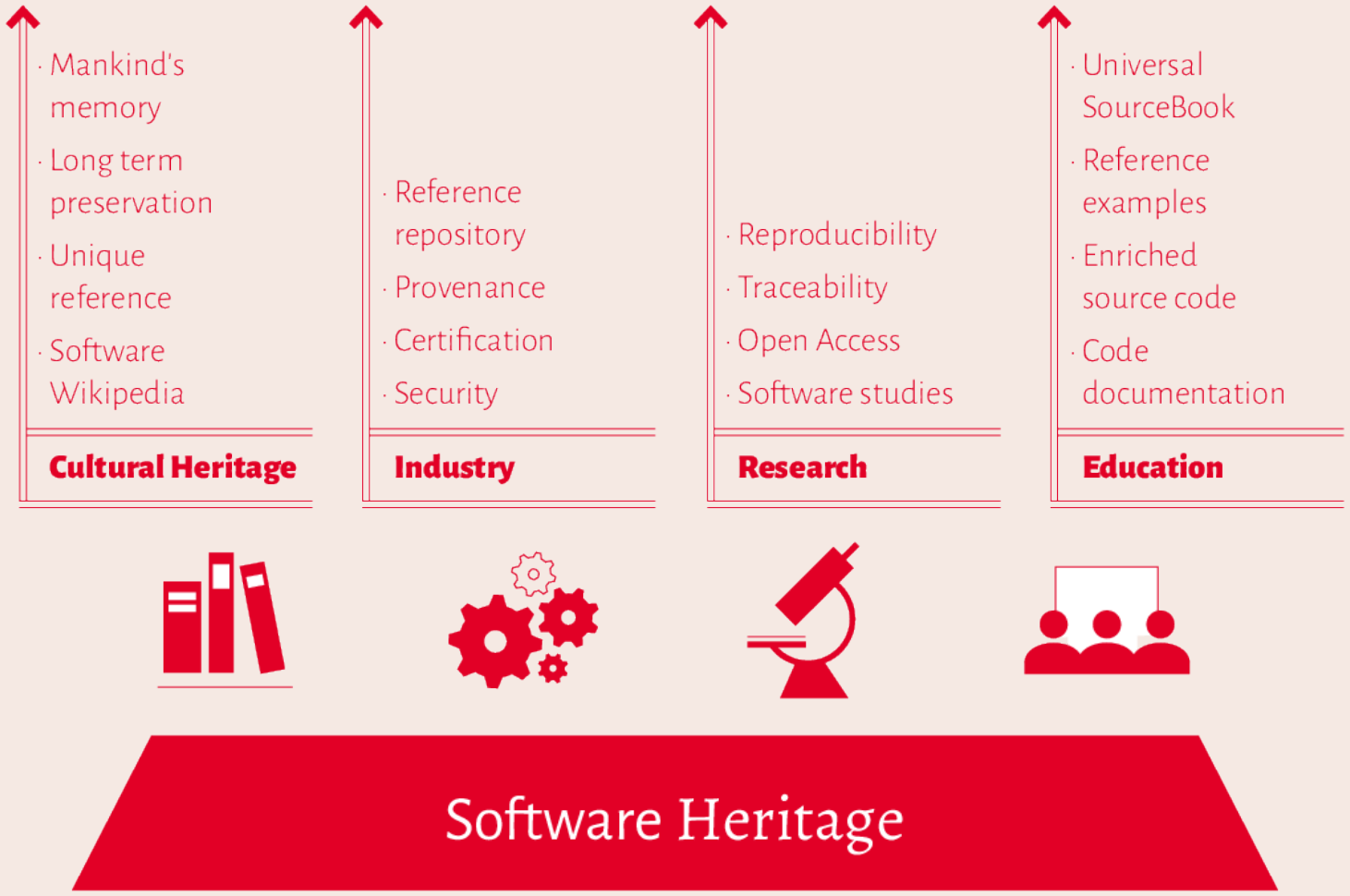
- *executable* and *human readable* knowledge (an *all time new*)
  - even hardware is… software! (VHDL, FPGA, …)
  - *text files are forever*
- naturally *evolves* over time
  - the *development history* is key to its *understanding*
- complex: large *web of dependencies*, millions of SLOCs

In a word

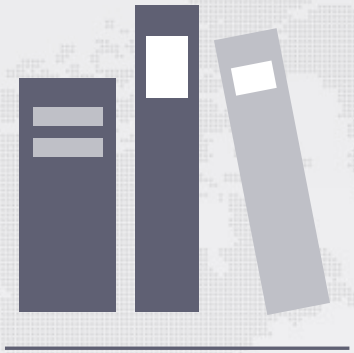
- software *is not just another* sequence of bits
- a software archive *is not just another* digital archive

We are working on the foundations

one infrastructure to build them all



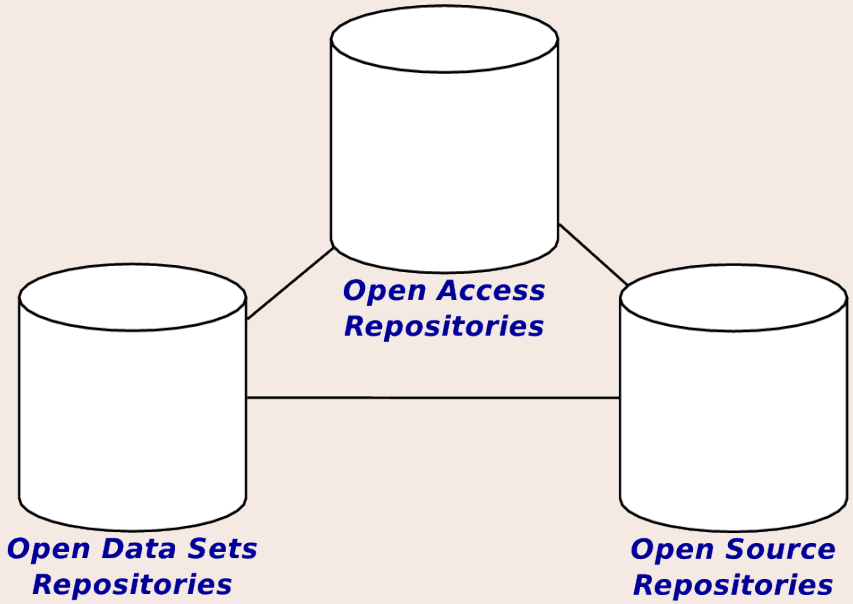
Preserving the world's software heritage



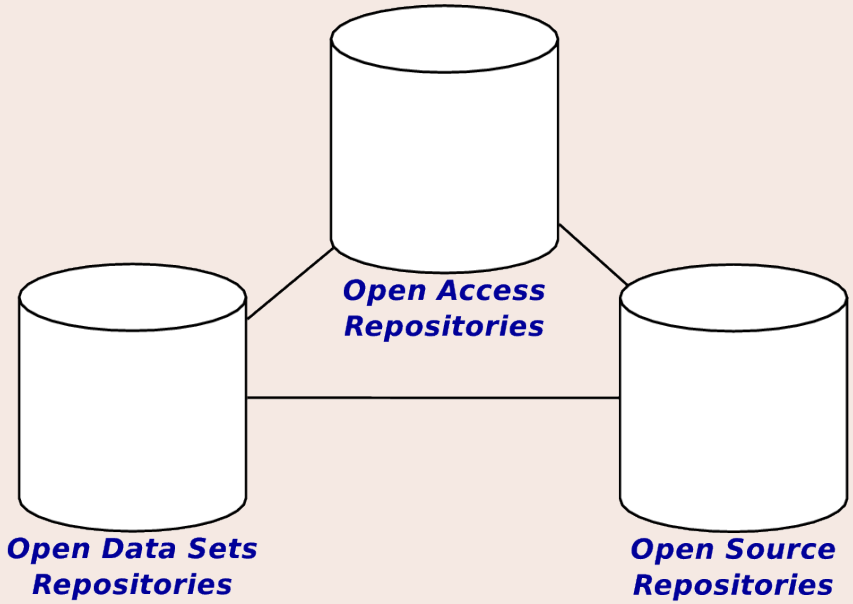
A structured archive of all of the world's software

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

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

Better software for industry and society

A unique reference catalog of all industrial software components

- a single entry point to discover, explore and reuse source code
- eases vulnerability tracking for more secure software
- simplifies traceability for better software integration
- ensures long term preservation of critical software

Compliance

Customers

The "software industry" ...

develops, maintains, uses, and/or integrates software components

Needs

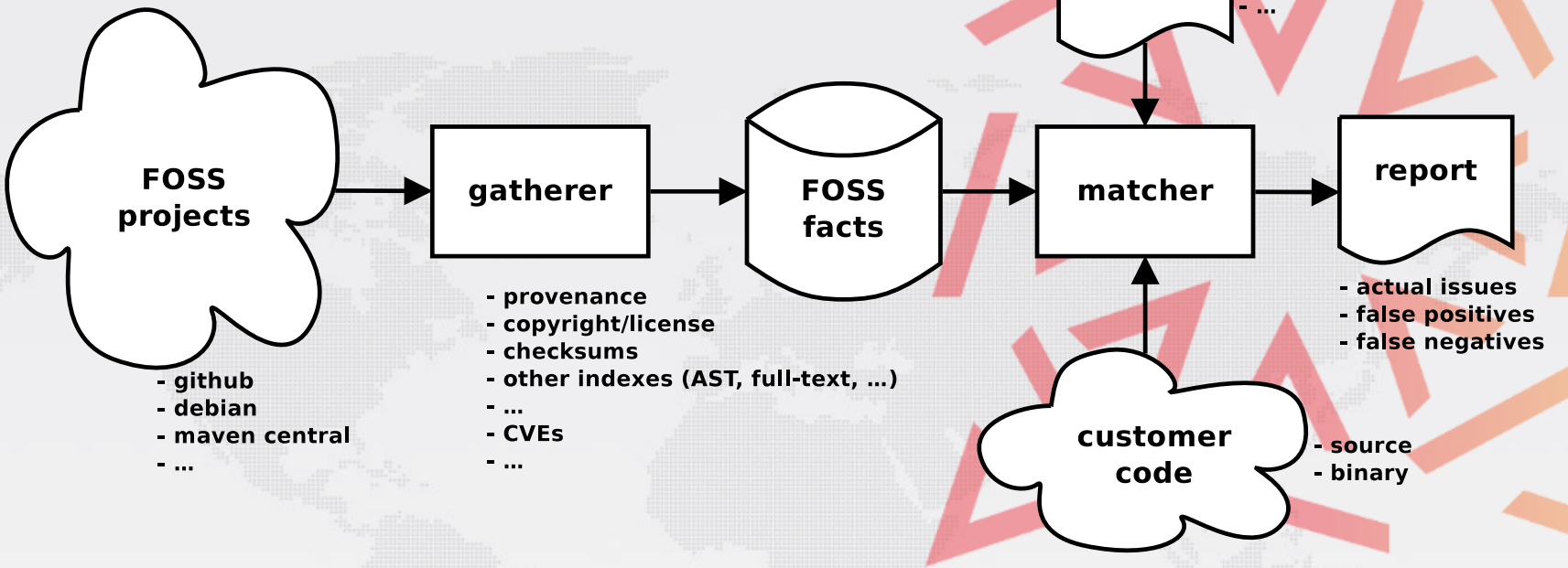
- track provenance of source code
- enforce licensing policies, avoid costly violations
- streamline supply chain management...

  acceptance process, subcontractors, etc.
- medium to long term software maintenance

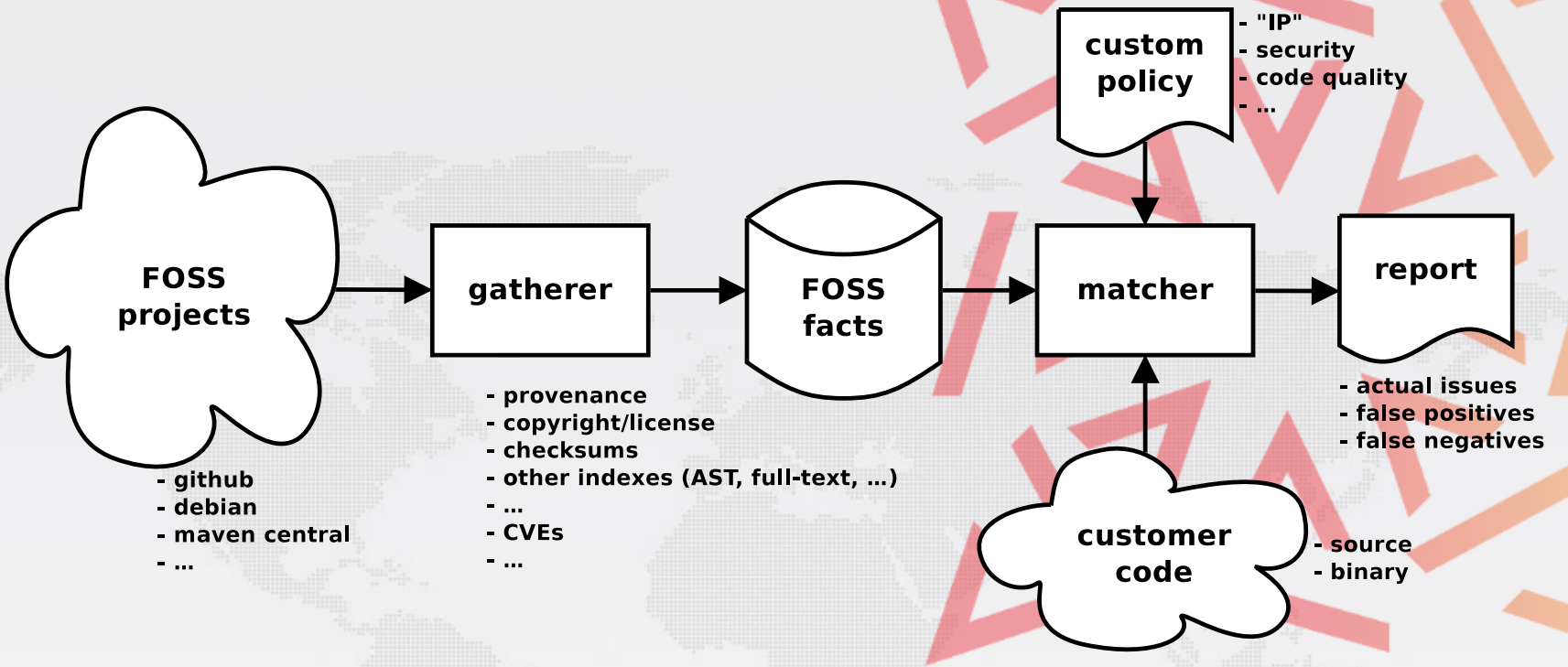
Tools

- code scanners
- bill of materials reporters

The compliance toolchain



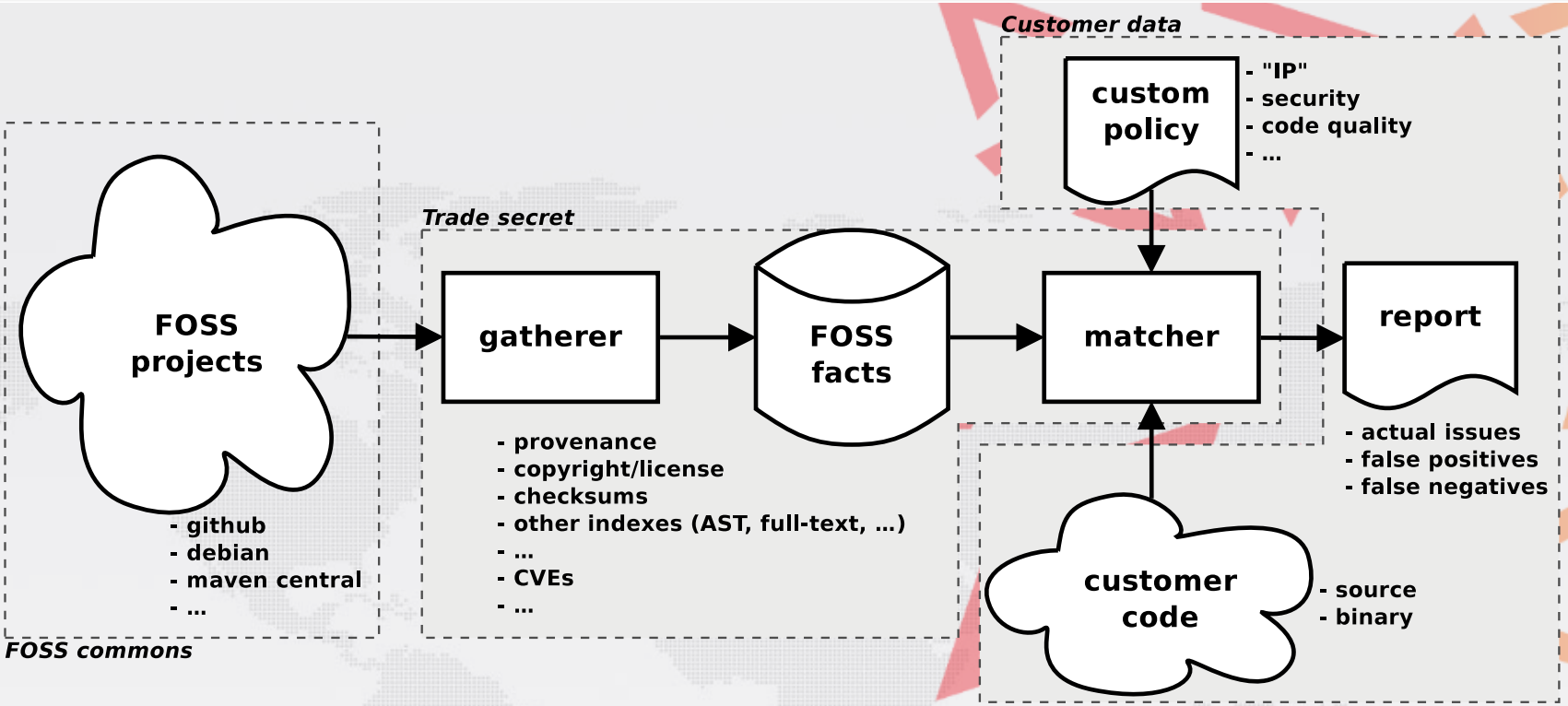
The compliance toolchain



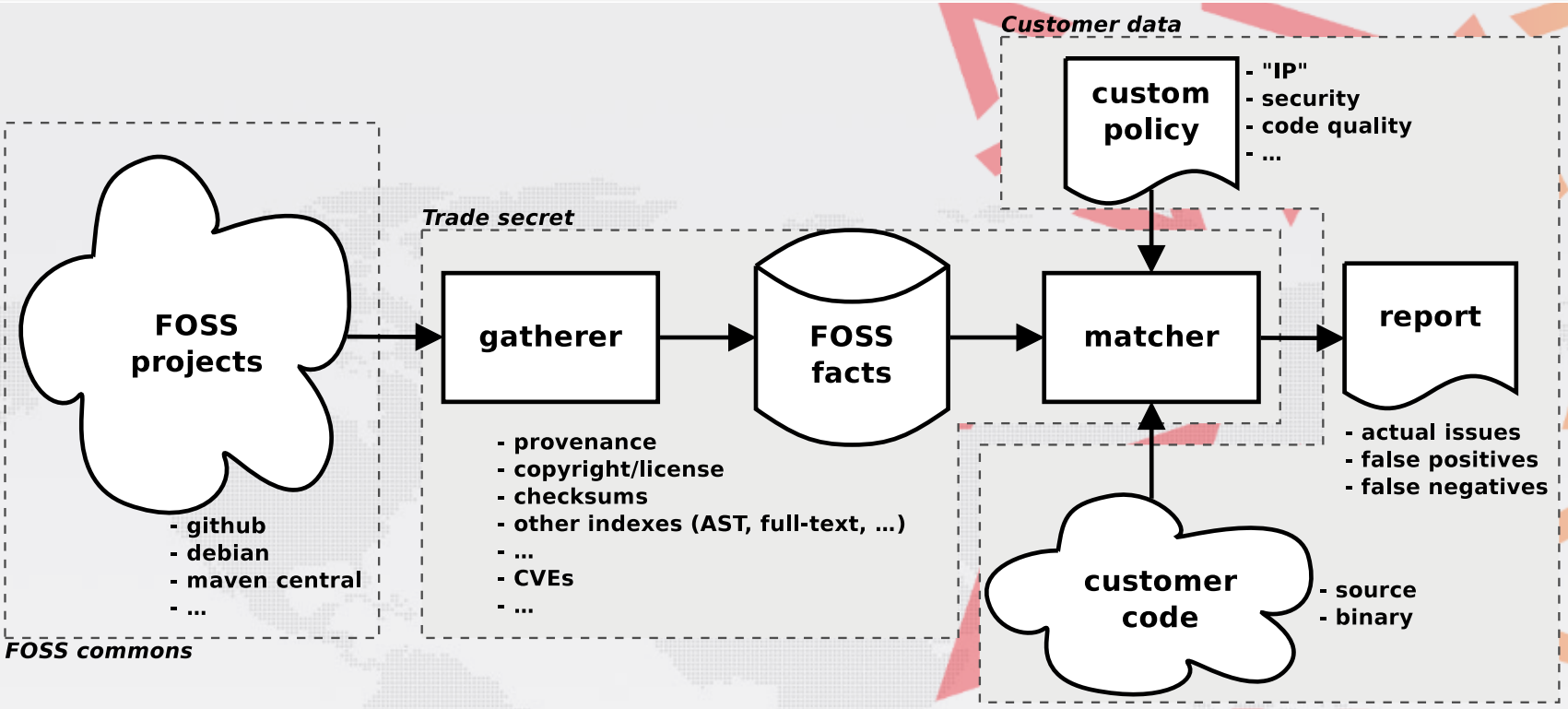
A note about *copyright-related facts*:

- ~~foo.c is under GPL3+~~
- content with SHA1 `f1d2df2...`, was last seen (UNIX timestamp 1454060276) at `http://git.bar.com/foo/`, under the name `foo.c` and the tool *FOSSology version 3.0.0* reported it as licenced under *GPL3+*

The compliance toolchain today



The compliance toolchain today

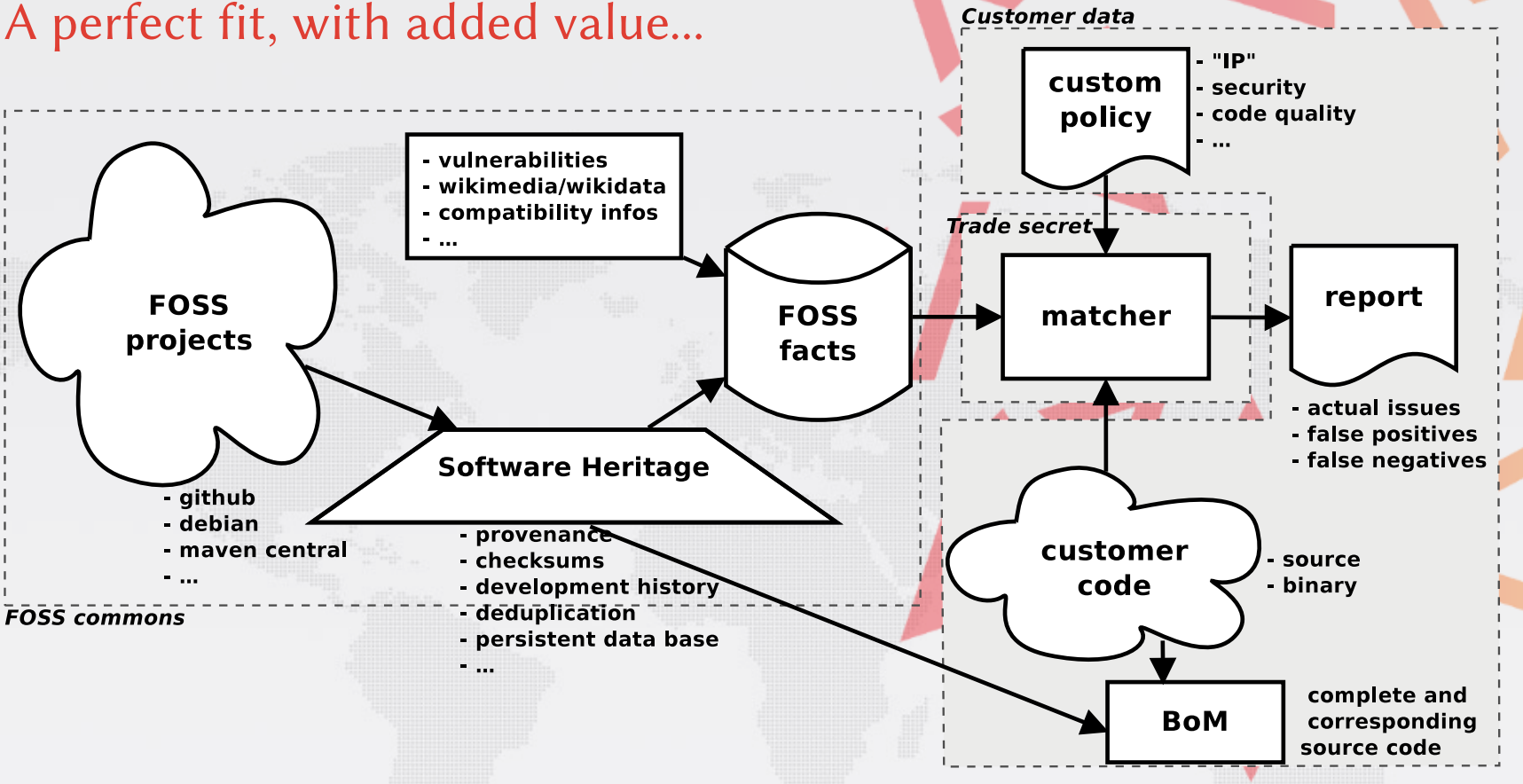


Limitations

- non-free tools, secret data bases
  - poor (or no) reproducibility
  - low quality
  - no interoperability
  - sharing *off business core* knowledge is difficult

Software Heritage in the picture

A perfect fit, with added value...



Get the complete corresponding source code

- even when upstream has gone away
- not just a legal obligation
- cornerstone for product maintenance and evolution

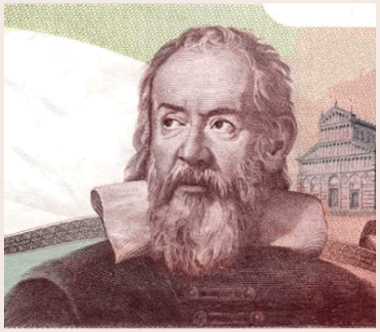
Supporting more accessible and reproducible science

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

How we built our scientific knowledge

The experimental method

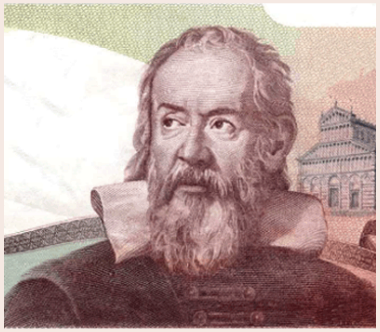


- make an *observation*
- formulate an *hypothesis*
- set up an experiment
- formulate a *theory*

And then we reproduce and verify.

How we built our scientific knowledge

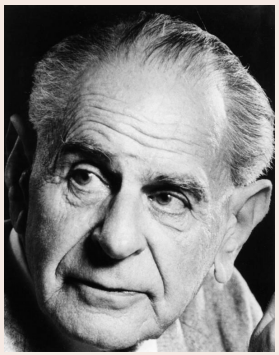
The experimental method



- make an *observation*
- formulate an *hypothesis*
- set up an experiment
- formulate a *theory*

And then we reproduce and verify.

Reproducibility is the key



non-reproducible single occurrences are of no significance to science

Karl Popper, The Logic of Scientific Discovery, 1934

Reproducibility, today

Reproducibility (Wikipedia)

the ability of an entire experiment or study to be *reproduced*, either by the researcher or *by someone else working independently*.
It is one of the main principles of the scientific method.

Why we want it

- foundation of the scientific method
- accelerator of research: allows to build upon previous work
- visibility: reproducible results are cited more often
- transparency of results eases acceptance
- necessary for industrial transfer

reproducibility is *the essence* of *industry*!

Software and reproducibility

A fundamental question

How are we doing, regarding reproducibility, in *Software*?

The case of Computer Systems Research

A field with Computer experts ... we have high expectations!
Christian Collberg set out to check them.

Measuring Reproducibility in Computer Systems Research

Long and detailed technical report, March 2014
`http://reproducibility.cs.arizona.edu/v1/tr.pdf`

Collberg's report from the trenches

Analysis of 613 papers

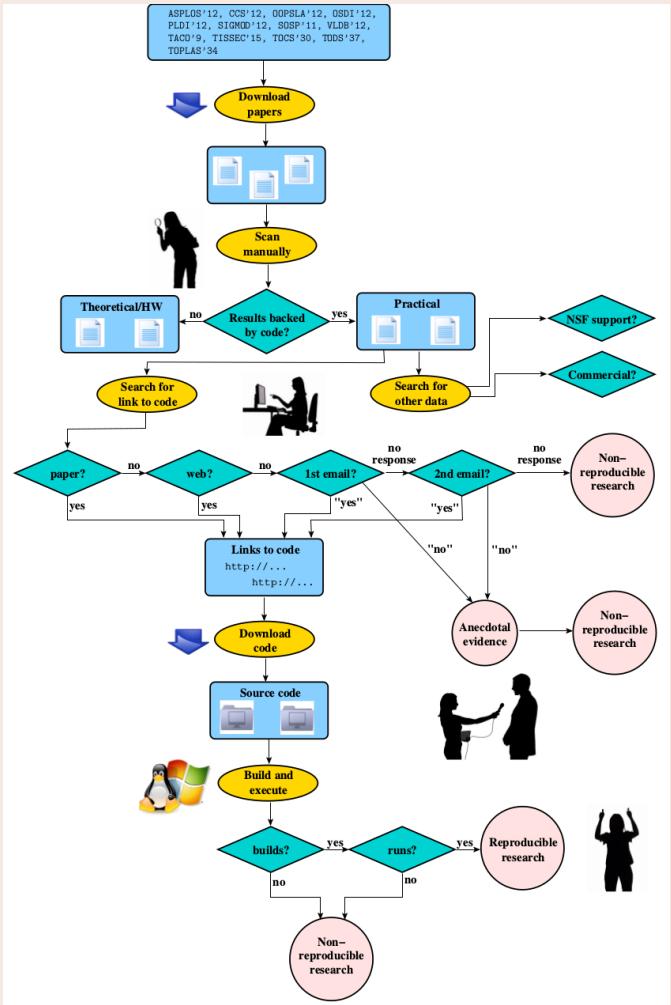
- 8 ACM conferences: ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12
- 5 journals: TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34

all very practical oriented

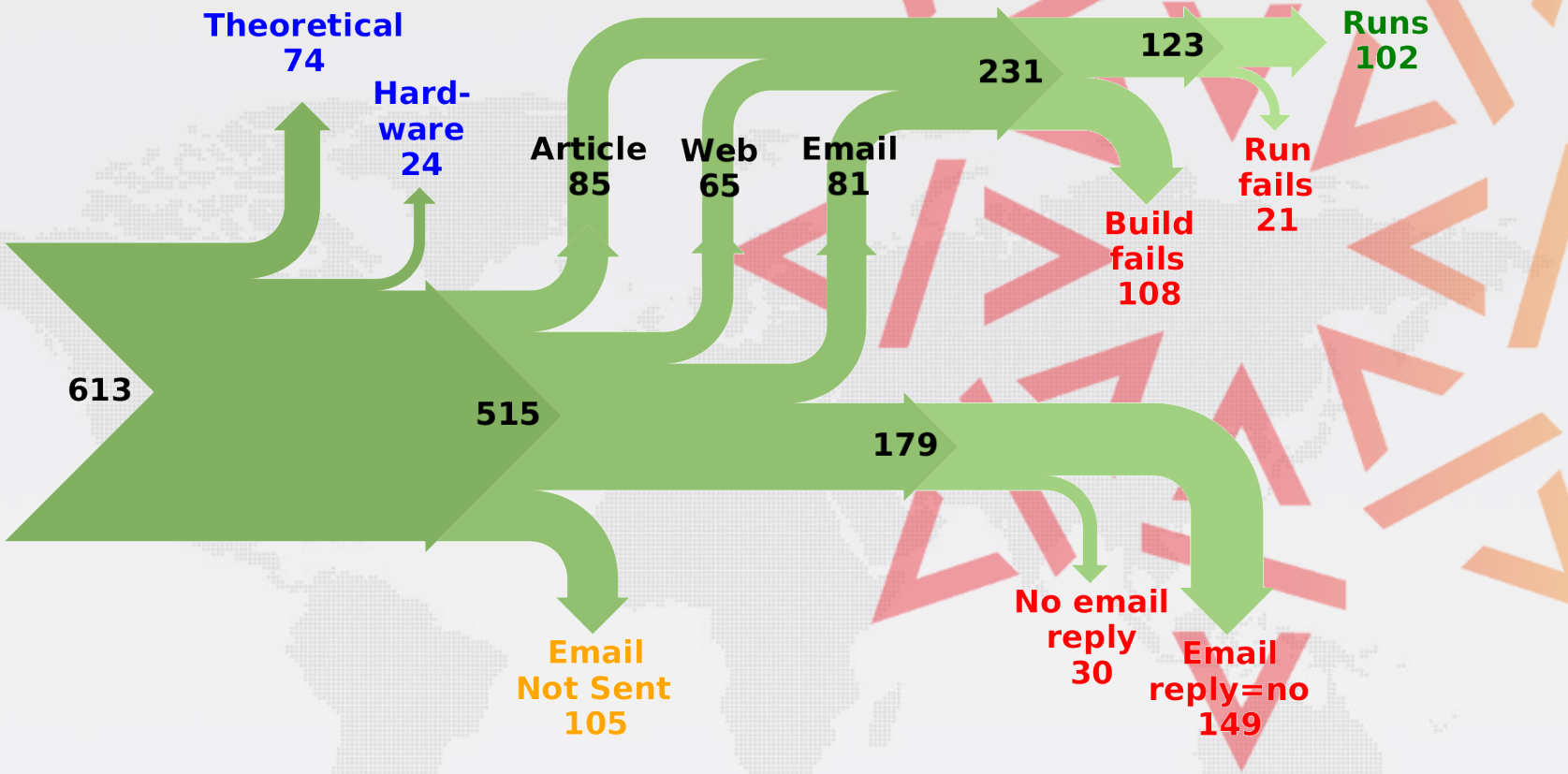
The basic question

can we get the code to build and run?

The workflow



The result



This can be debated (see `http://cs.brown.edu/~sk/Memos/Examining-Reproducibility/`), but...

... that's a whopping 81% of non reproducible works!

The reasons (or, “the dog ate my program”)

Why so much software fails to pass the test?

Many issues, nice anecdotes, and it finally boils down to

- *Availability*
- *Traceability*
- Environment
- Automation (do *you* use continuous integration?)
- Documentation
- Understanding (including Open Source)

The reasons (or, “the dog ate my program”)

Why so much software fails to pass the test?

Many issues, nice anecdotes, and it finally boils down to

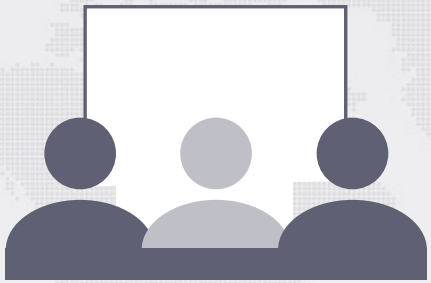
- *Availability*
- *Traceability*
- Environment
- Automation (do *you* use continuous integration?)
- Documentation
- Understanding (including Open Source)

The first two are important *software preservation issues*

Yes, code is fragile:

it can be destroyed, and we can lose trace of it

Fostering wider education to computing



A global source referencing all software

- a source book for technological education
- intrinsic persistent identifiers for stable course materials
- enables real-world, semi-automated documentation

Outline

1 The Free Software Commons

2 Software is fragile

3 Software Heritage

4 Approach

5 Current status

6 Selected research challenges

7 Conclusion

Three properties are key for Software Heritage's mission

Availability

- *all* the *history* of *all* the software
- no restrictions (technical, legal, …) on *content* or *metadata*

Three properties are key for Software Heritage's mission

Availability

- *all* the *history* of *all* the software
- no restrictions (technical, legal, …) on *content* or *metadata*

Traceability

- *unique* identifiers : *one* name for each object
- *persistent* and *intrinsic* identifiers : no middle man, no dangling pointers!

Three properties are key for Software Heritage's mission

Availability

- *all* the *history* of *all* the software
- no restrictions (technical, legal, …) on *content* or *metadata*

Traceability

- *unique* identifiers : *one* name for each object
- *persistent* and *intrinsic* identifiers : no middle man, no dangling pointers!

Uniformity

- one *standard* metadata structure, *irrespective of the origins*
- *uniform* naming *schema*

Three properties are key for Software Heritage's mission

Availability

- *all* the *history* of *all* the software
- no restrictions (technical, legal, …) on *content* or *metadata*

Traceability

- *unique* identifiers : *one* name for each object
- *persistent* and *intrinsic* identifiers : no middle man, no dangling pointers!

Uniformity

- one *standard* metadata structure, *irrespective of the origins*
- *uniform* naming *schema*

here are some bits from our drawing board

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 suffer from.

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 suffer from.

design decision

Software Heritage will:

- provide *full details* on its architecture
- release all *its own source code as FOSS*
  → `https://forge.softwareheritage.org`
- use *open standards*
- encourage a *collaborative* development process
- unleash and leverage *the power of the community*

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.*

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

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

design decision

Software Heritage will:

- provide *intrinsic* resource identifiers
- *avoid relying on* volatile identifiers (e.g., DOI, URL) for integrity

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.

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.

design decision

Software Heritage will:

- provide easy means for making copies
- encourage the growth of a mirror network
  - using *a variety* of technologies
  - spanning *multiple* continents
  - under *diverse* control structures
    - no single decisional point of failure!
      (remember Google code, Gitorious, . . .)

Caring for the long term

Build *one* infrastructure catering to *shared concerns*

cultural heritage, *scientific* research, *industrial* development, *educational* outreach

not just a project

projects have limited time-frame

nonprofit

neutrality, transparency

international

- multiple funding streams
- open to contributions

multistakeholder

- Unesco
- … you!

Unix philosophy I

do *one* thing, do it *well*

Unix philosophy II

make it *interoperable*

Outline

1 The Free Software Commons

2 Software is fragile

3 Software Heritage

4 Approach

5 Current status

6 Selected research challenges

7 Conclusion

The people

The core team

- Roberto Di Cosmo
- Stefano Zacchiroli
- Nicolas Dandrimont (Engineer)
- Antoine Dumont (Engineer)
- and *Jordi, Quentin and Guillaume*

Scientific advisors

- Serge Abiteboul (French Sience Academy)
- Jean-François Abramatic (former W3C director)
- Gerard Berry (CNRS Gold Medal, French Science Academy)
- Julia Lawall (Coccinelle, Linux Kernel, Outreachy)

The archive

Our sources

- GitHub — all public repositories as of August 2016
- Debian — daily snapshots of all suites since 2005–2015
- GNU — all releases as of August 2015
- Gitorious — retrieved full mirror from Archive Team
- Google Code — retrieved full mirror from Google

The archive

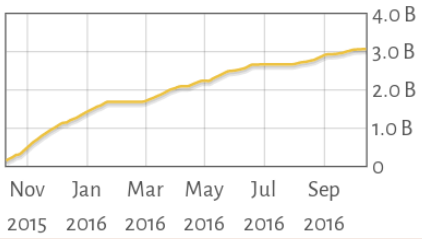
Our sources

- GitHub — all public repositories as of August 2016
- Debian — daily snapshots of all suites since 2005–2015
- GNU — all releases as of August 2015
- Gitorious — retrieved full mirror from Archive Team
- Google Code — retrieved full mirror from Google

Some numbers

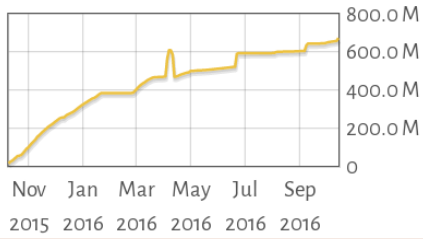
Source files

3,081,210,112



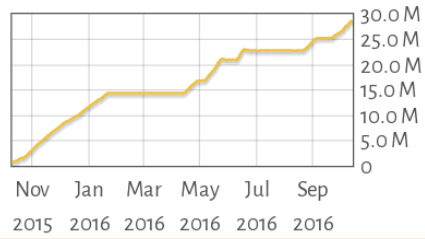
Commits

684,733,440



Projects

29,037,912

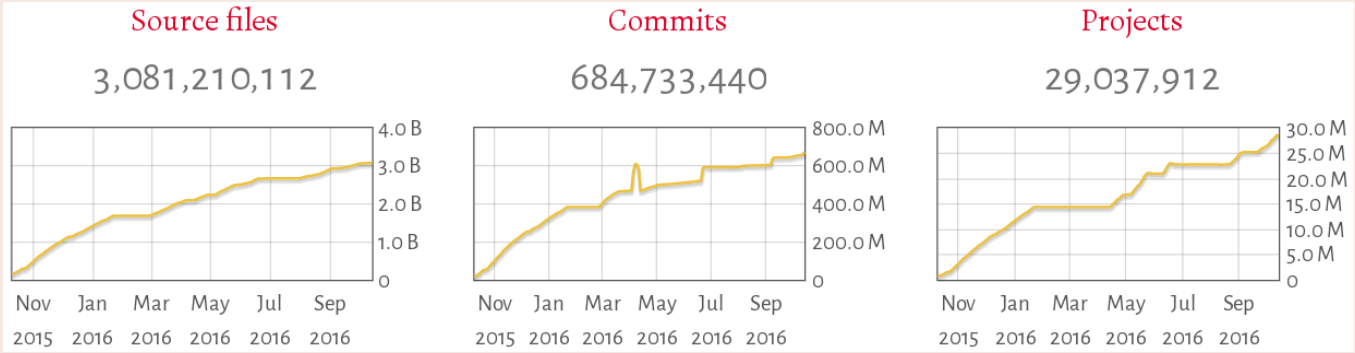


The archive

Our sources

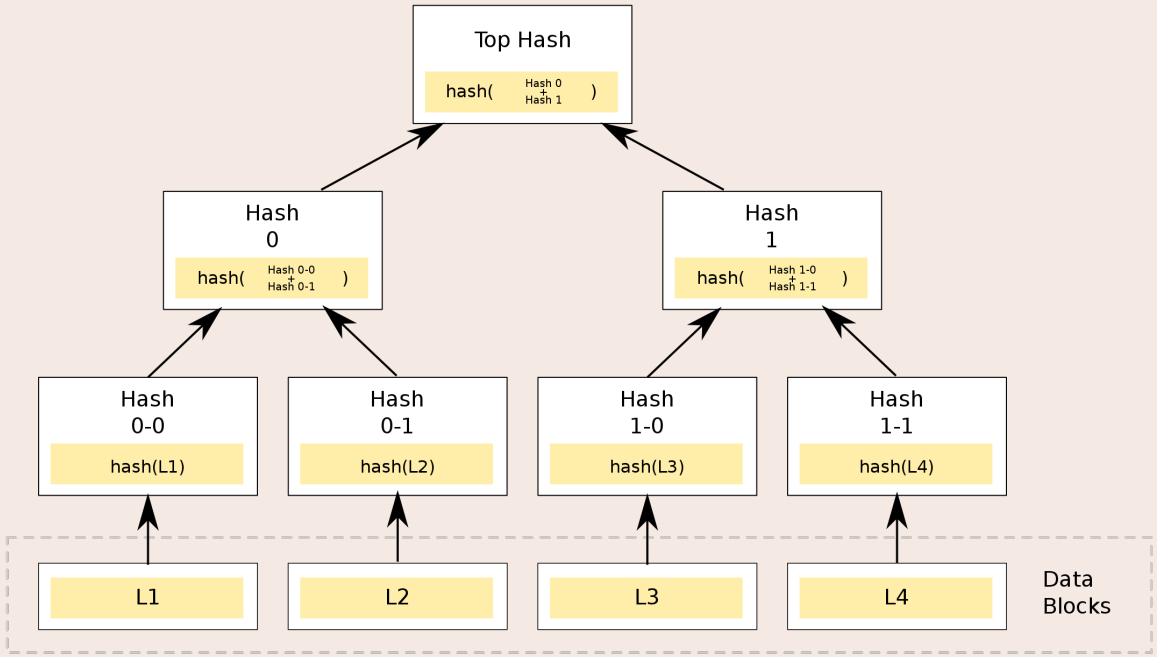
- GitHub — all public repositories as of August 2016
- Debian — daily snapshots of all suites since 2005–2015
- GNU — all releases as of August 2015
- Gitorious — retrieved full mirror from Archive Team
- Google Code — retrieved full mirror from Google

Some numbers



The *richest* source code archive already, … and growing daily!

Merkle tree (R. C. Merkle, Crypto 1979)

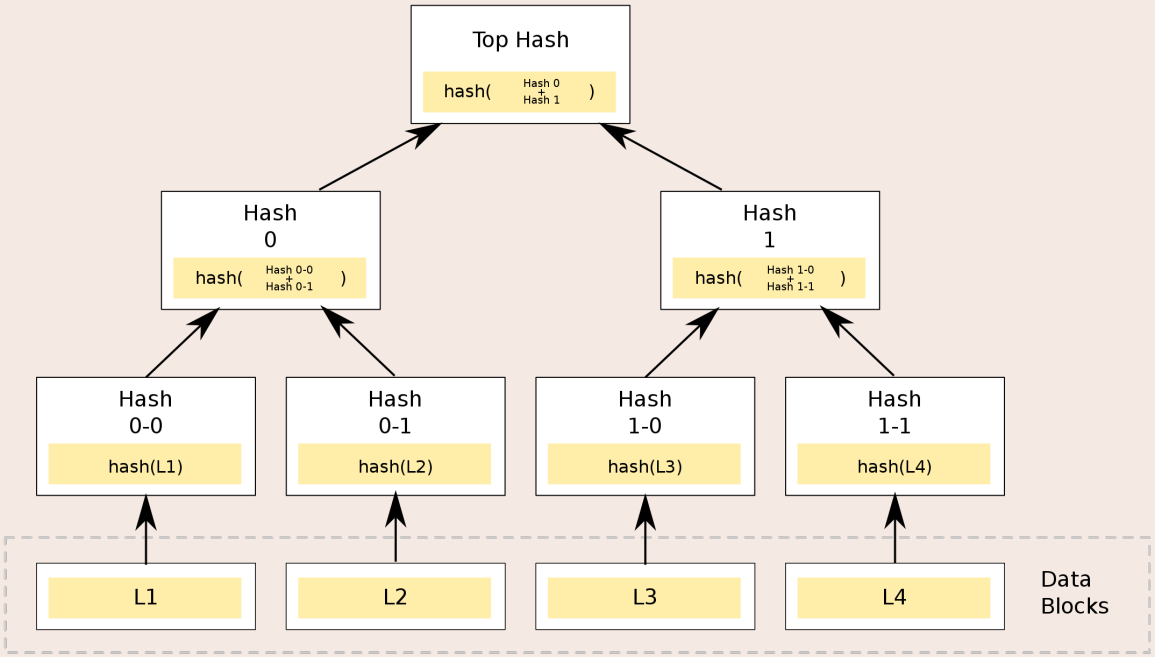


Combination of

- tree
- hash function

Merkle trees

Merkle tree (R. C. Merkle, Crypto 1979)



Combination of

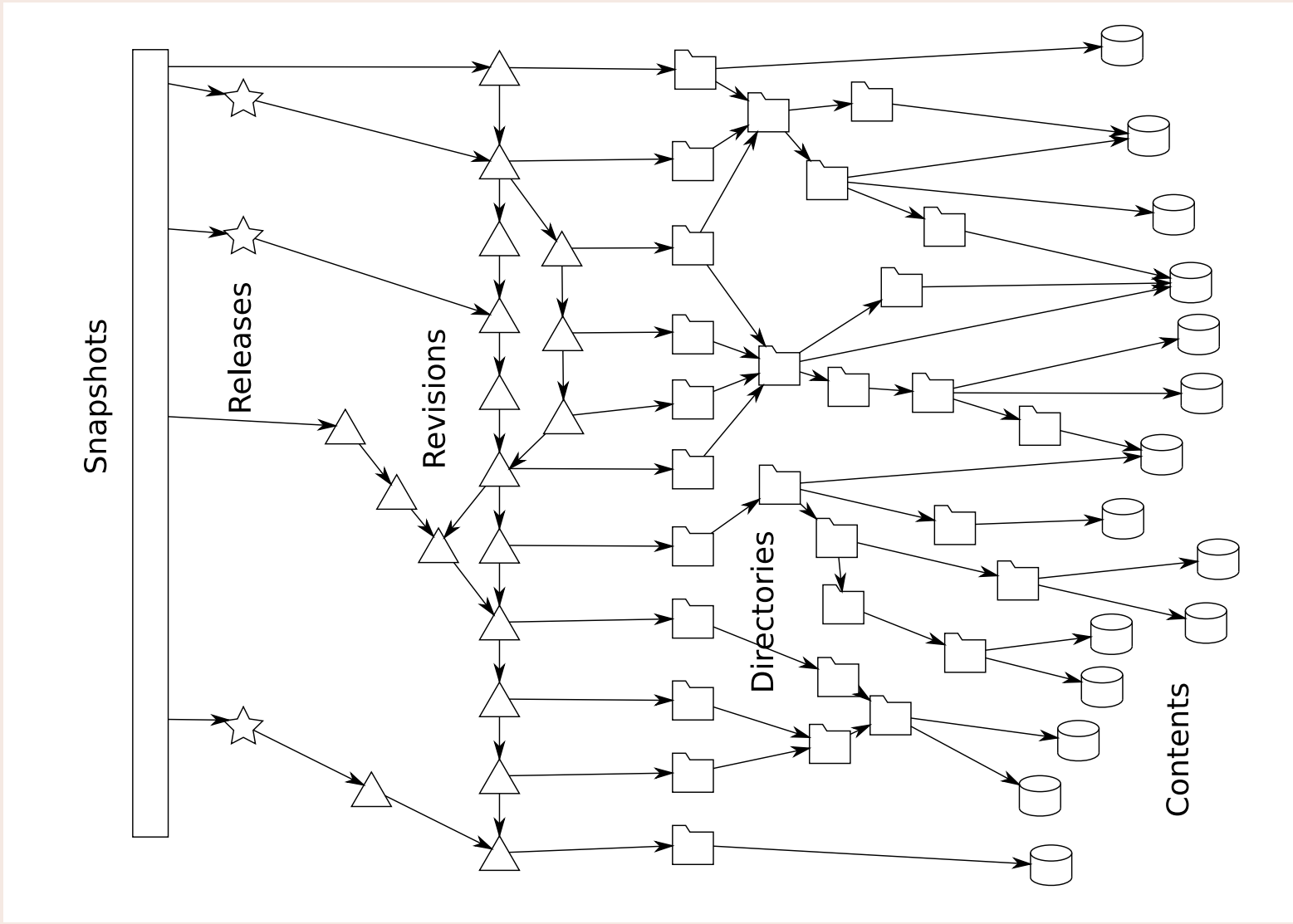
- tree
- hash function

Classical cryptographic construction

- fast, parallel signature of large data structures
- widely used by *Git*, *Bitcoin*, etc.
- natural extension: Merkle *DAG*

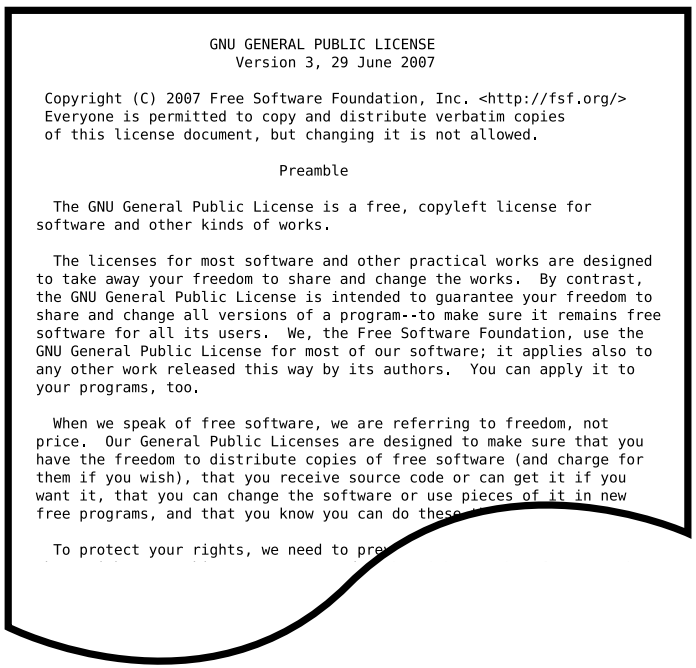
The archive in a few pictures

A giant (extended) Merkle DAG



A giant (extended) Merkle DAG

Contents



GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

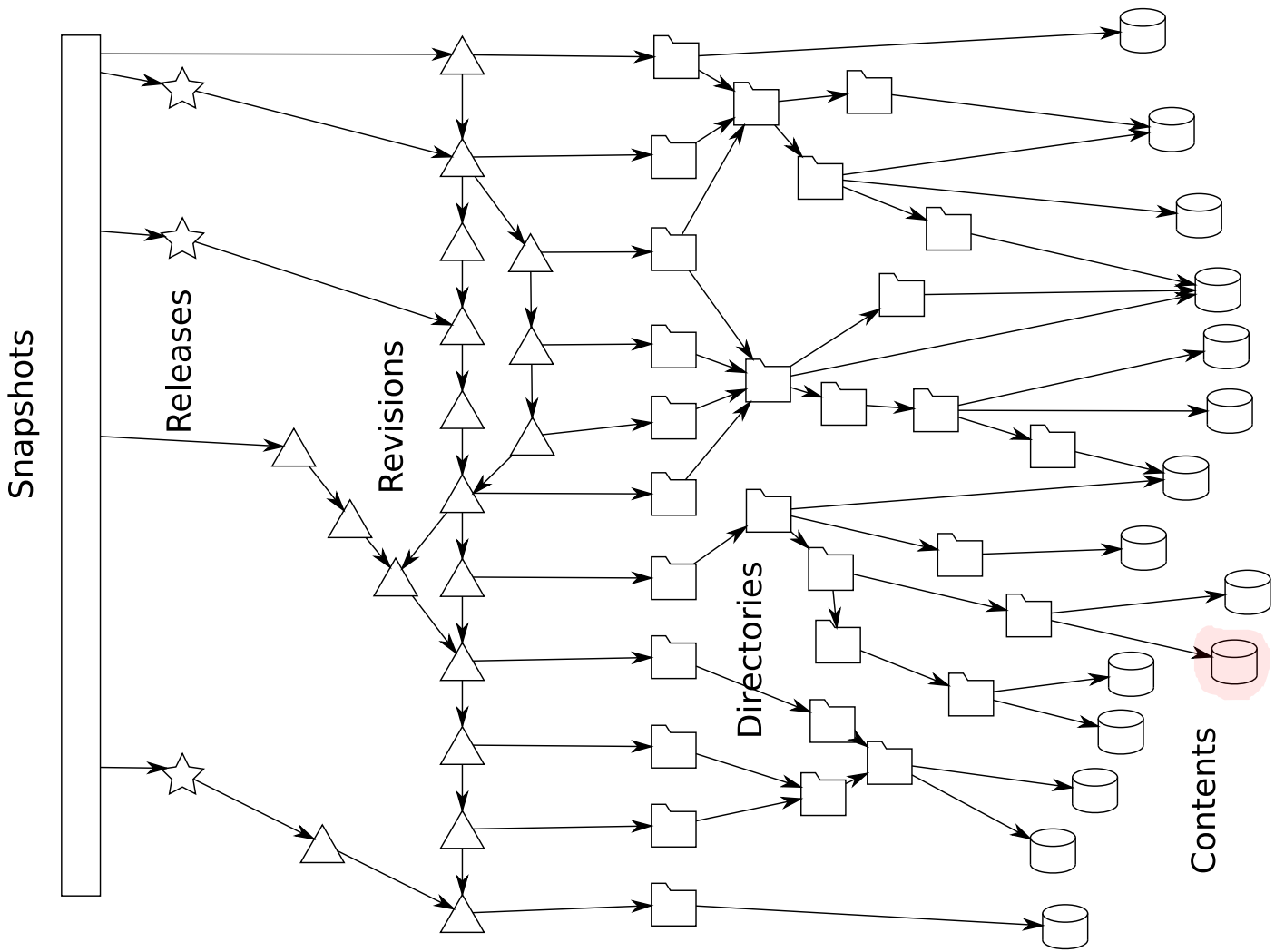
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these

To protect your rights, we need to pre

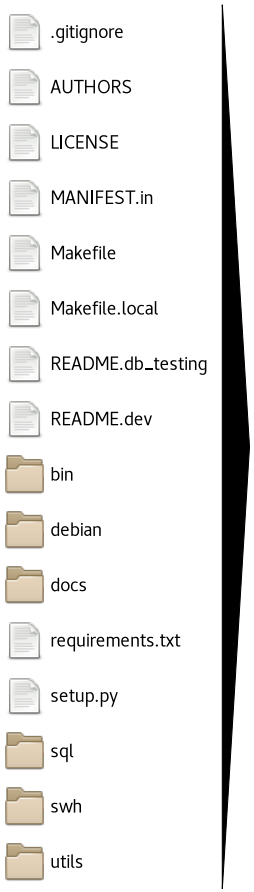
sha1: 8624bcdae55baeef...
sha256: 8ceb4b9ee5aded...
sha1_git: 94a9ed024d385...
length: 35147

The archive in a few pictures

A giant (extended) Merkle DAG



A giant (extended) Merkle DAG



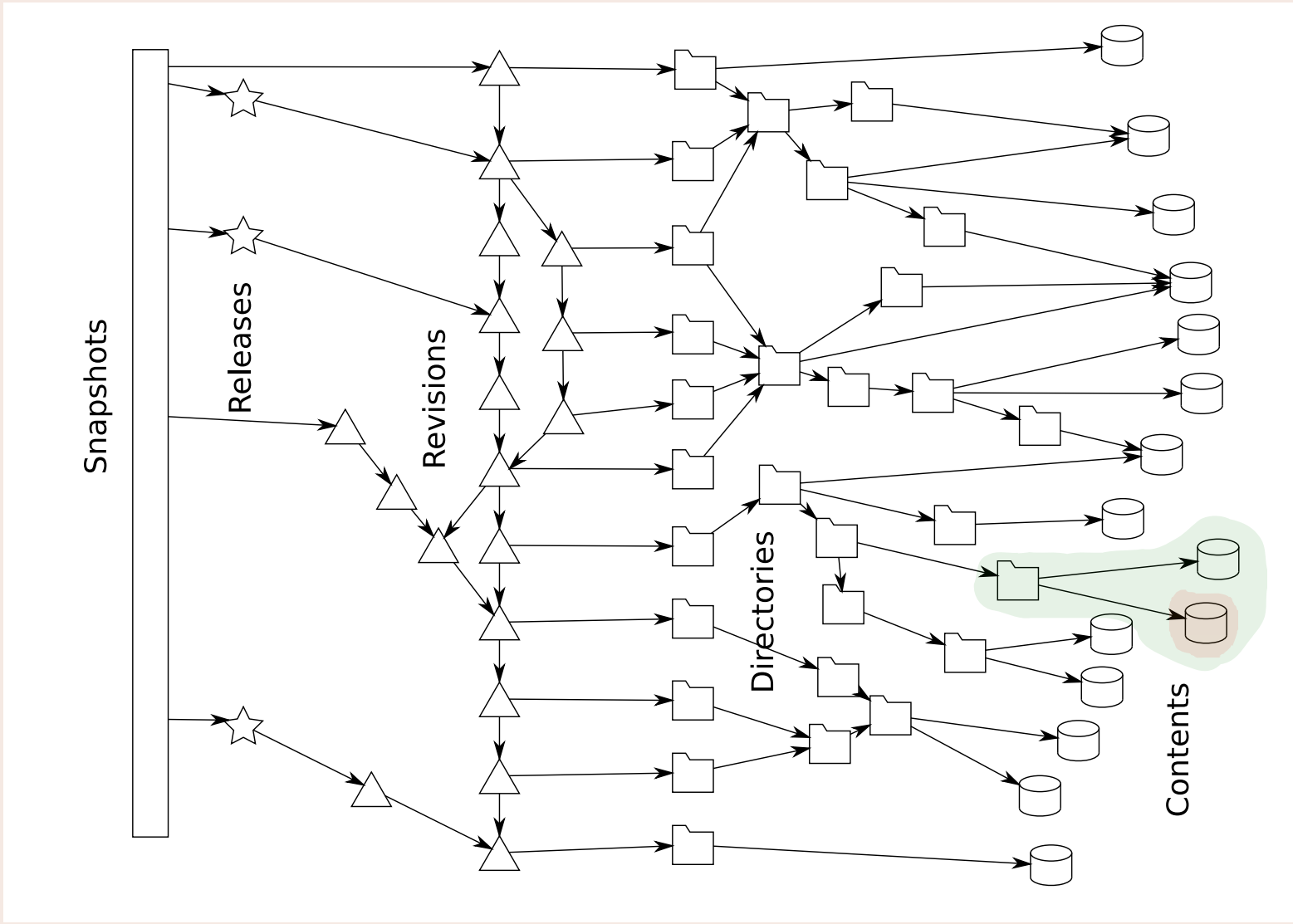
Directories

```
100644 blob c5baade4c4476042186ef858c0fd63d587ebf09 .gitignore
100644 blob 2d0a34af6f52cf3cf6b0c2f7bd0648fbd255e77f AUTHORS
100644 blob 94a9ed024d3859793618152ea559a168bbcbb5e2 LICENSE
100644 blob d9b2665a435a43f8a79a84e0867751dfb095c7bb MANIFEST.in
100644 blob 524175c2bad0b35b975f79284c2f5a6d5eaf2eb4 Makefile
100644 blob 5c7e3a5bbddb038682ba7793f440492ed9678bb3 Makefile.local
100644 blob 8617980629cd24e6080404f09aa749b085b3e07b README.db_testing
100644 blob 76b29f94cf815e0869c414d38d78d7ce08ec514e README.dev
040000 tree e1e10ecef948af0b93adb0372afc89f12e92618a bin
040000 tree 83e56d0beaf7793c77a45a345c80fcb8af503013 debian
040000 tree a34c9c4ba213f0cedc67f9816348d27955577af5 docs
100644 blob f2a6d32c6135aa7287bbd76167b01df2ae4f1539 requirements.txt
100755 blob eee147c36caf1bbc2d820da8dc026cb5b68180bc setup.py
040000 tree 224bb4c1f4c67fca1d160bffd2d06094e7e1abf3 sql
040000 tree 8631c9cd77bbe993168107ab5baf51f40c6300be swh
040000 tree 8fb905b56ba8ed692f1209b2773b474c6c1d66c1 utils
```

id: 515f00d44e92c65322aaa9bf3fa097c00ddb9c7d

The archive in a few pictures

A giant (extended) Merkle DAG

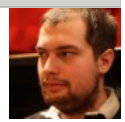


A giant (extended) Merkle DAG

Revisions

Details | Changes | Files

SHA: 963634dca6ba5dc37e3ee426ba091092c267f9f6



Author: Nicolas Dandrimont <nicolas@dandrimont.eu> (Thu Sep 1 14:26:13 2016)

Committer: Nicolas Dandrimont <nicolas@dandrimont.eu> (Thu Sep 1 14:26:13 2016)

Subject: **provenance.tasks: add the revision -> origin cache task**

Parent: fc3a8b59ca1df424d860f2c29ab07fee4dc35d10 : *test_storage: properly pipeline origin and cont...*

provenance.tasks: add the revision -> origin cache task

swh/storage/provenance/tasks.py 77

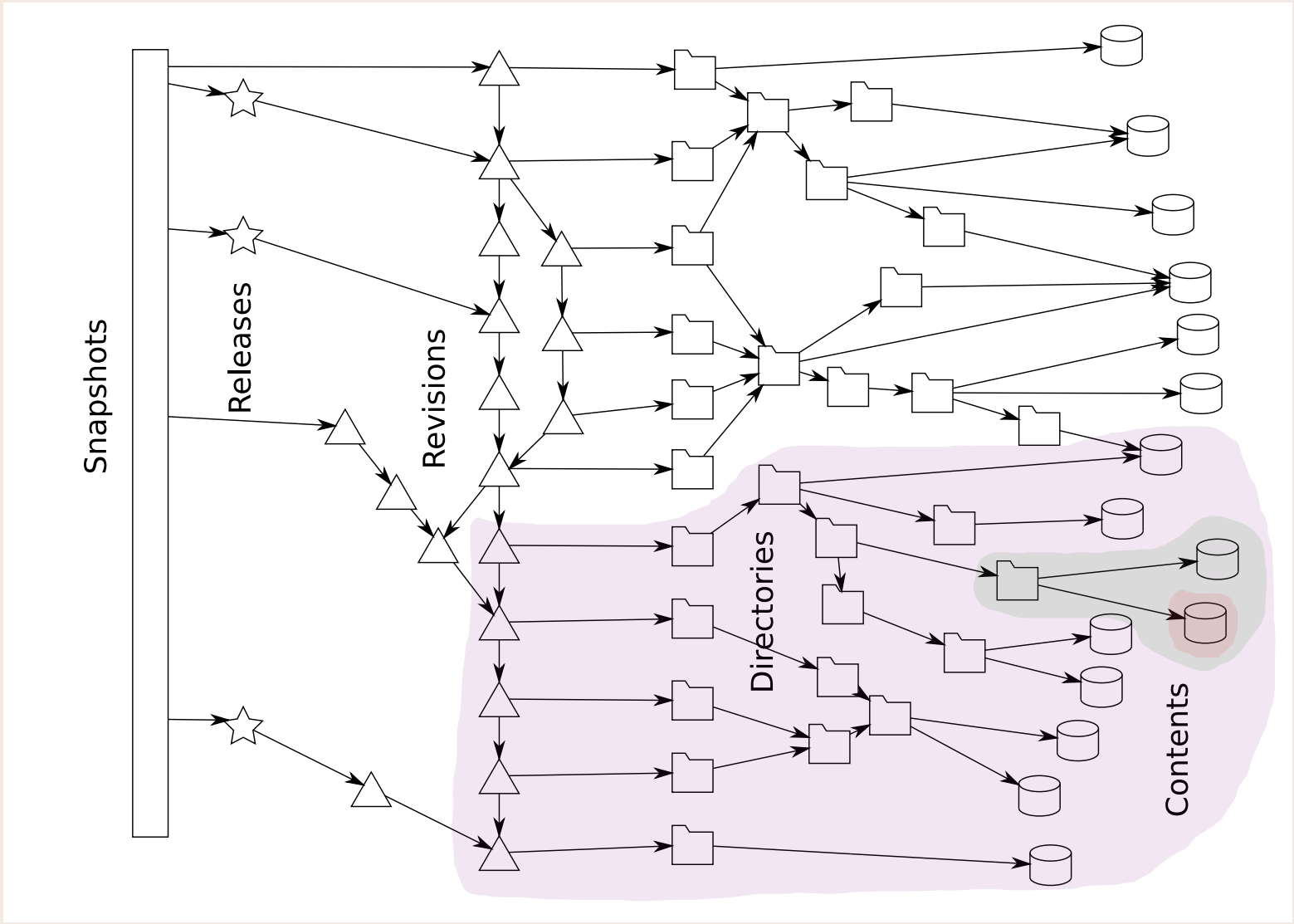
tree 515f00d44e92c65322aaa9bf3fa097c00ddb9c7d
parent fc3a8b59ca1df424d860f2c29ab07fee4dc35d10
author Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200
committer Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200

provenance.tasks: add the revision -> origin cache task

id: 963634dca6ba5dc37e3ee426ba091092c267f9f6

The archive in a few pictures

A giant (extended) Merkle DAG



A giant (extended) Merkle DAG

Releases

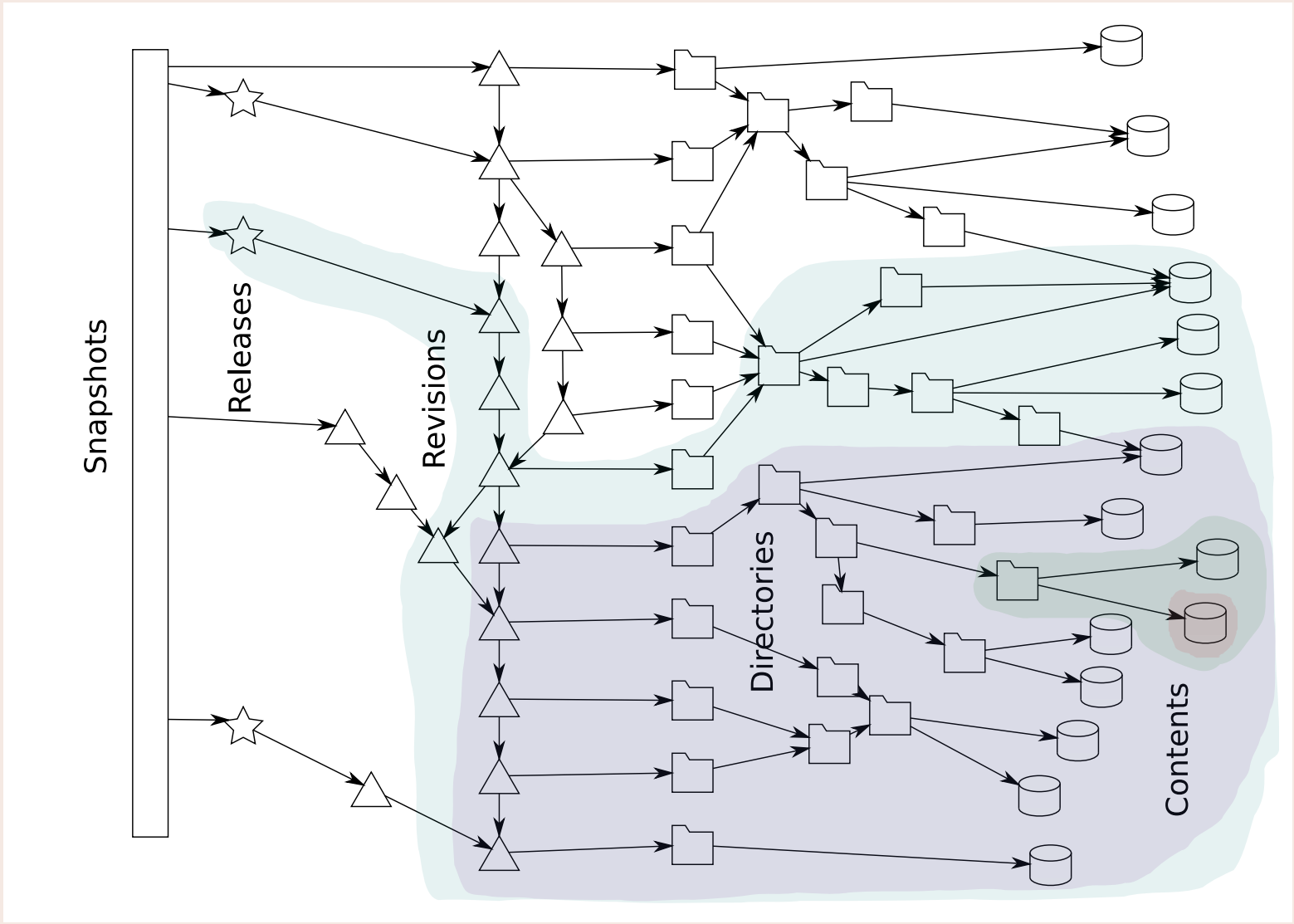
```
tag v0.0.51
Tagger: Nicolas Dandrimont <nicolas@dandrimont.eu>
Date:   Wed Aug 24 14:36:03 2016 +0200

Release swh.storage v0.0.51

 - Add new metadata column to origin_visit
 - Update swh-add-directory script for updated API
[...]

commit c0c9f16b1e134f593e7567570a1761b156e6eb1d
```

```
object c0c9f16b1e134f593e7567570a1761b156e6eb1d
type commit
tag v0.0.51
tagger Nicolas Dandrimont <nicolas@dandrimont.eu> 1472042163 +0200

Release swh.storage v0.0.51

 - Add new metadata column to origin_visit
 - Update swh-add-directory script for updated API
-----BEGIN PGP SIGNATURE-----

iQIzBAABCAAdBQJXvZTNFhxuaWNvbGFzQGRhbmRyaW1vbnQuZXUACgkQ7AWLMo2+
neqorw//aq6SOb5DijzEa+kWN3rXgVS+1K1vEVh1wNKAwx8eKJ7aX2kEiLDtt7uf
ahpZ6pz3q8nqs6aC1+YrxBfcih3L2YtrdZeWXWqr8xWNMaEoYDb8qaphwh8AD5t2
ICBIit2uJtXuCrDt93eKKPwvzZXg+hB0sMWy35Dr6jW7Z7K4Mu/PGgIyIHPY55yo
IGEndWno7VfH1Vm6t1n5qB7I5mXRaqA+becqddubTZ2xjj+jplUqC8cyqN3hm/fL
qsj2mu8kyz3t8tG/H1/pV+I5OwBInPoS5TH0tujoJEVgPK/dHSP79QuHDHZFkCao
klj6kAWyU80Mxb+nKV/jeLbrR3+yWBFj3Qp5a1/V8oOTh6E1dALcNMpEaKCoKtMt
d/gMRax1Il1/g0EDfnsW67G6sDwKPKPHhgfVLQ3nV3GaQQTnu1RpMz006H9/tAwzC
Gg/K1PdHT4hzOiI46wYPZyje0U2VXGFu6vVU9vFQ4ZR/Wjn+0zMzdcRdrlJSUOMn
RpTTfUsbXUeXHGOpkgXhSYTnvp1gdPc76USTsK0aGe84AZm1Ik0mGrwXCVfPqIYo
nhhibBSHBNMoqyF6yTSOpUbYK70tpYRRUGKWDeRK0wKSxkWKUZGtKzy6JYqljo29
gulwgZQif5qWQCB0OontAL2+HvPFaVyckMejUhg62cP/+EHlvUk=
=kOxP
-----END PGP SIGNATURE-----
```

id: 85083a5cc14a441c89dea73f5bdf67c3f9c6afdb

The archive in a few pictures

A giant (extended) Merkle DAG



Hardware

hosted by Inria

- Hypervisor with a dozen virtual machines
- High density storage array (60 * 6TB => 300TB usable)
- Copy in another server room; logical leader/follower mirroring
- Soon to enable a mirror network to duplicate our contents

Azure cloud (work in progress prototype)

- full mirror using distributed object storage
- workers for batch analyses and crawling

Software

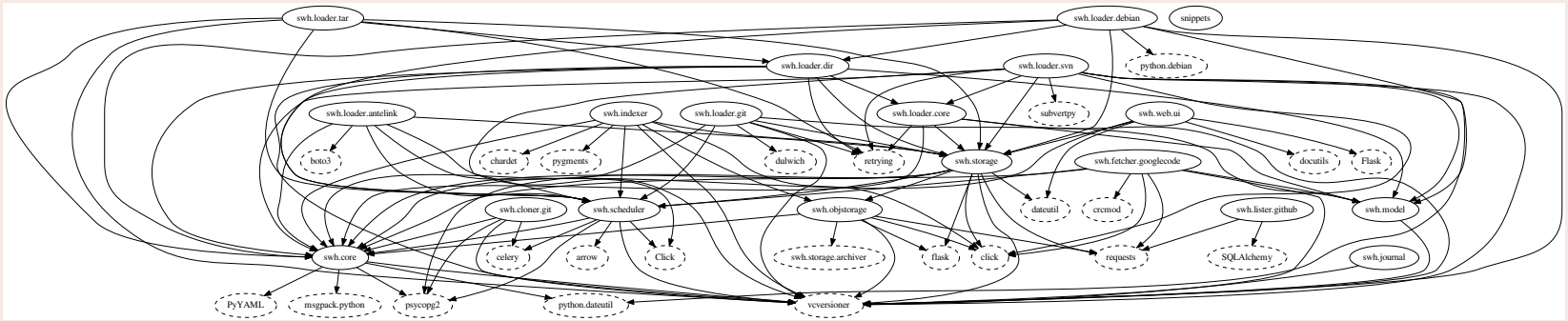
3rd party FOSS

- Debian distribution, orchestrated with Puppet
- PostgreSQL for metadata storage
- RabbitMQ for task scheduling
- Python3 and psycopg2 for the backend
- Flask and Bootstrap for the web apps
  → https://www.softwareheritage.org/jobs/
- Phabricator forge

in-house FOSS

- ~50 Git repositories (~20 Python packages, ~10 Puppet modules)
- ~20 kSLOC Python / ~10 kSLOC SQL / ~1 kSLOC Puppet
- licence choice: GPLv3 (backend) / AGPLv3 (frontend)
- https://forge.softwareheritage.org/

Module dependencies (internal + external)



let's zoom in: `http://deb.li/swhdeps`

The road ahead

Planned features...

- *lookup* by content hash (done)
- *download*: wget and git clone from Software Heritage
- *provenance information* for all archived code and metadata
- *browsing*: wayback machine for archived code and its history
- *full-text search* on all archived source code files

The road ahead

Planned features...

- *lookup* by content hash (done)
- *download*: wget and git clone from Software Heritage
- *provenance information* for all archived code and metadata
- *browsing*: wayback machine for archived code and its history
- *full-text search* on all archived source code files

... and much more than one could possibly imagine

all the world's software development history in a single graph!

that makes a 150TB archive / 5TB database already...

Software development

classic FOSS development

- language: English
- development mailing list
  `https://sympa.inria.fr/sympa/info/swh-devel`
- IRC
  #swh-devel / FreeNode
- Forge
  `https://forge.softwareheritage.org`
- Git, tasks, code review, etc.

for more information

`https://www.softwareheritage.org/community/developers/`

Outline

1. The Free Software Commons
2. Software is fragile
3. Software Heritage
4. Approach
5. Current status
6. Selected research challenges
7. Conclusion

Metadata alignment

Many concepts related to source code

- project, archive, source, language, licence, bts, mailing list, ...
- developer, committer, author, architect, ...

Metadata alignment

Many concepts related to source code

- project, archive, source, language, licence, bts, mailing list, …
- developer, committer, author, architect, …

Many existing ontologies, catalogs

- DOAP, FOAF, Appstream, schema.org, ADMS.SW, …
- Freecode (40.000+), Plume (400+), Debian (25.000+), FramaSoft (1500+), OpenHub (670.000+), …

Metadata alignment

Many concepts related to source code

- project, archive, source, language, licence, bts, mailing list, ...
- developer, committer, author, architect, ...

Many existing ontologies, catalogs

- DOAP, FOAF, Appstream, schema.org, ADMS.SW, ...
- Freecode (40.000+), Plume (400+), Debian (25.000+), FramaSoft (1500+), OpenHub (670.000+), ...

Challenge : scale up metadata to millions of projects

- *reconcile* existing ontologies
- *link* and *check* existing catalogs with Software Heritage
- handle *inconsistent data* and *provenance information*
- synthesise missing information (machine learning)

Software phylogenetics

The Software Diaspora

- Code often *migrates* across projects: forks, copy-paste
- Code gets *cloned*: reuse, language limitations, code smells
- Projects *migrate* across forges: fashion, functionality
- Projects get *cloned*: mirrors, packages

Software phylogenetics

The Software Diaspora

- Code often *migrates* across projects: forks, copy-paste
- Code gets *cloned*: reuse, language limitations, code smells
- Projects *migrate* across forges: fashion, functionality
- Projects get *cloned*: mirrors, packages

Challenge: tracing software evolution across billions of files

- rebuild the history of software artefacts
- identify code origins
- spot code clones
- build project impact graphs

Distributed infrastructure

The software graph

- files
- directories
- commits
- projects

all de-duplicated in Software Heritage

Challenge: design efficient architectures and algorithms

- replication and availability
- navigation
- what happens to CAP? (updates are nondestructive!)
- query

Code search: an old problem

A natural need

- Find the definition of a function/class/procedure/type/structure
- Search examples of code usage in an archive of source code
- you name it…

Code search: an old problem

A natural need

- Find the definition of a function/class/procedure/type/structure
- Search examples of code usage in an archive of source code
- you name it…

A natural approach

- Regular expressions

Code search: an old problem

A natural need

- Find the definition of a function/class/procedure/type/structure
- Search examples of code usage in an archive of source code
- you name it…

A natural approach

- Regular expressions

We have all used *grep* since the 1970's!

where is the challenge?

Finding a needle in a haystack: size matters!

How do we search in *millions* of source code files?

Google code search (open 2006, closed 2011)

see `https://swtch.com/~rsc/regexp/regexp4.html`
reborn in Debian `http://codesearch.debian.net/`

Finding a needle in a haystack: size matters!

How do we search in *millions* of source code files?

Google code search (open 2006, closed 2011)

see `https://swtch.com/~rsc/regexp/regexp4.html`
reborn in Debian `http://codesearch.debian.net/`

how

- build an inverted index of *trigrams* from all source files
- *map* regexps to trigrams
- *filter* files that may match
- run *grep* on each file (using the "cloud")

Finding a needle in a haystack: size matters!

How do we search in *millions* of source code files?

Google code search (open 2006, closed 2011)

see `https://swtch.com/~rsc/regexp/regexp4.html`
reborn in Debian `http://codesearch.debian.net/`

how

- build an inverted index of *trigrams* from all source files
- *map* regexps to trigrams
- *filter* files that may match
- run *grep* on each file (using the "cloud")

performance

scaled reasonably well up to *1 billion lines of codes*

Challenge: scaling up code search

What about *all the source code* in the world?

Software Heritage is *two orders of magnitude* bigger already

- over *two billion* unique source files
- *hundreds* of billions of LOCs

We need new insight for handling this.

Challenge: scaling up code search

What about *all the source code* in the world?

Software Heritage is *two orders of magnitude* bigger already

- over *two billion* unique source files
- *hundreds* of billions of LOCs

We need new insight for handling this.

Beyond regular expressions?

Advanced code search requires

- language specific *patterns*
- working on *abstract syntax trees*

Regular expressions are a nice *swiss-army knife* approximation, can we build a specific tool that scales?

Software as Big Data = "Big Code"

Remember the numbers

- 30 million repositories ingested (10M next in line)
- 700 million commits
- 3 billion unique source files / 200 TB of raw source code

… and growing by the day!

Challenge: what can machines learn here?

- programming patterns
- developer skills
- vulnerabilities
- bugs and fixes

Outline

1 The Free Software Commons

2 Software is fragile

3 Software Heritage

4 Approach

5 Current status

6 Selected research challenges

7 Conclusion

Conclusion

Software Heritage is

- a *comprehensive reference archive* of all public source code
- a promising new tool for *software research*
- an international, open, nonprofit, *shared infrastructure*
- at the service of our community and society

Conclusion

Software Heritage is

- a *comprehensive reference archive* of all public source code
- a promising new tool for *software research*
- an international, open, nonprofit, *shared infrastructure*
- at the service of our community and society

Now open

`www.softwareheritage.org` - *sponsoring, partnerships*
`wiki.softwareheritage.org` - *working groups, leads*
→ keep an eye on the `Category:Internship` page
`forge.softwareheritage.org` - *our own code*

Questions?