Software Heritage

a common infrastructure for Software Engineering and Open Science

Roberto Di Cosmo

roberto@dicosmo.org

February 1st, 2019

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Short Bio: Roberto Di Cosmo

Computer Science professor in Paris, now working at INRIA

- *30 years* of research (Theor. CS, Programming, Software Engineering, Erdos #: 3)
- *20 years* of Free and Open Source Software
- *10 years* building and directing structures for the common good

1999	*DemoLinux* – first live GNU/Linux distro
2007	*Free Software Thematic Group* 150 members 40 projects 200Me
2008	*Mancoosi project* `www.mancoosi.org`
2010	*IRILL* `www.irill.org`
2015	*Software Heritage* at INRIA
2018	*National Committee for Open Science*, France

Outline

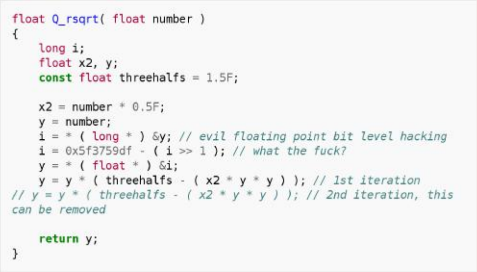
1. Introductions
2. Software is everywhere…
3. … and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Source code is *special*

Harold Abelson, Structure and Interpretation of Computer Programs

“Programs must be written for people to read, and only incidentally for machines to execute.”

Quake III source code (excerpt)

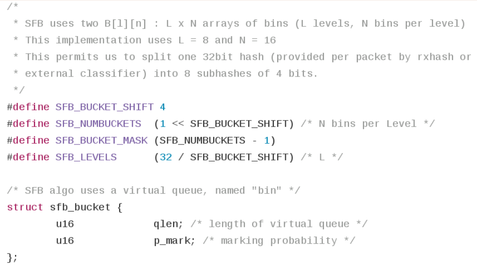


```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
// y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this
can be removed

    return y;
}
```

Net. queue in Linux (excerpt)



```
/*
 * SFB uses two B[l][n] : L x N arrays of bins (L levels, N bins per level)
 * This implementation uses L = 8 and N = 16
 * This permits us to split one 32bit hash (provided per packet by rxhash or
 * external classifier) into 8 subhashes of 4 bits.
 */
#define SFB_BUCKET_SHIFT 4
#define SFB_NUMBUCKETS  (1 << SFB_BUCKET_SHIFT) /* N bins per Level */
#define SFB_BUCKET_MASK (SFB_NUMBUCKETS - 1)
#define SFB_LEVELS      (32 / SFB_BUCKET_SHIFT) /* L */

/* SFB algo uses a virtual queue, named "bin" */
struct sfb_bucket {
        u16             qlen; /* length of virtual queue */
        u16             p_mark; /* marking probability */
};
```

Len Shustek, Computer History Museum

“Source code provides a view into the mind of the designer.”

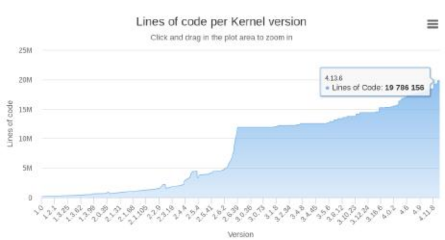
~ 50 years, a lightning fast growth

Apollo 11 Guidance Computer (~60.000 lines), 1969

"When I first got into it, nobody knew what it was that we were doing. It was like the Wild West."

Margaret Hamilton

Linux Kernel



... now in your pockets!

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Software is spread all around

Software is fragile

damage
disaster
malicious
obsolete
deletion
format
attack
storage
reference
dependencies
aging
media
tear
dangling
wear
corruption
encryption

Software lacks its own research infrastructure

Photo: ALMA(ESO/NAOJ/NRAO), R. Hills

No catalog, no archive, no references: we are at a turning point

Looking at the past

- a lot of old software misplaced, lost, or behind barriers, but...
- most founding fathers are still here, and willing to share
- urgent to collect their knowledge

Only a few years left.

Looking at the future

- software development and use skyrockets: more programmers, and more code!
- essential to provide a universal platform for all the future software source code

Every year that goes by makes the problem worse.

it is urgent to take action!

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. **Software Heritage**
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Software Heritage

Our mission

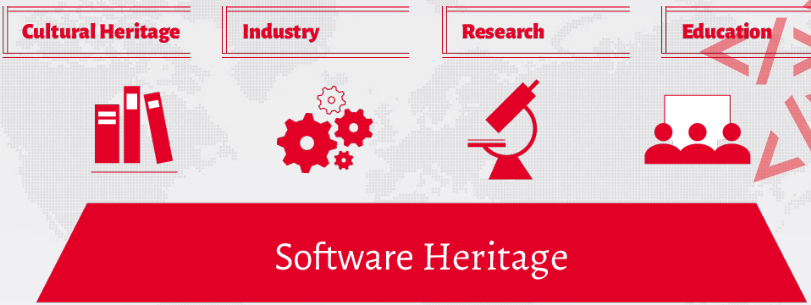
Collect, preserve and share the *source code* of *all the software* that is available

Past, present and future

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

A principled infrastructure

http://bit.ly/swhpaper



Cultural Heritage

Industry

Research

Education

Software Heritage

Technology

- transparency and FOSS
- replicas all the way down

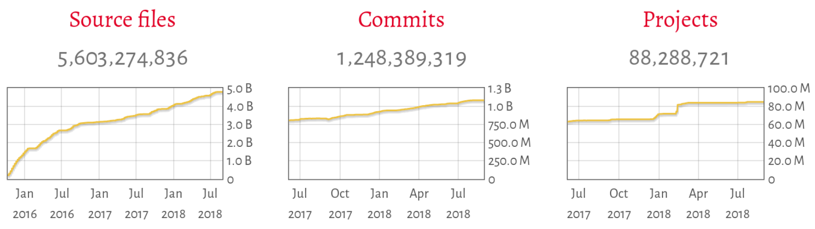
Content

- intrinsic identifiers
- facts and provenance

Organization

- non-profit
- mirror network

Coverage



Source files

5,603,274,836

Commits

1,248,389,319

Projects

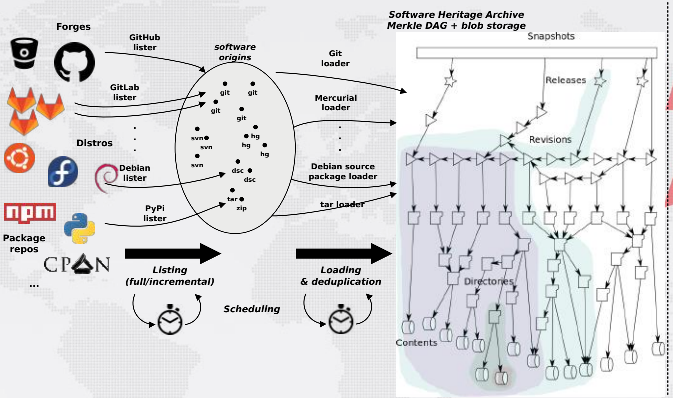
88,288,721

- 200 TB (compressed) blobs, 6 TB database (as a graph: 10 B nodes + 100 B edges)
- The *richest* public source code archive, ... and growing daily!

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. **Under the hood: architecture**
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Automation, and storage



- full development history permanently archived
- origins: GitHub (auto), Debian (auto), Gitlab.com, Gitorious, Google Code, GNU
- ~ 200Tb raw contents, ~ 10Tb graph (10Bn nodes, 100Bn edges)

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Our challenges in the PID landscape

Typical properties of systems of identifiers

uniqueness, non ambiguity, persistence, abstraction (opacity)

Key needed properties from our use cases

gratis identifiers are free (billions of objects)

integrity the associated object cannot be changed (sw dev, *reproducibility*)

no middle man no central authority is needed (sw dev, *reproducibility*)

we could not find systems with both integrity and no middle man !

An important distinction: DIOs vs. IDOs

> *The term "Digital Object Identifier" is construed as "digital identifier of an object," rather than "identifier of a digital object"* Norman Paskin. 2010

DIO (Digital Identifier of an Object) identifiers for (potentially) non digital objects

- epistemic complexity (manifestations, versions, locations, etc.)
- need an authority to ensure persistence and uniqueness

IDO (Identifier of a Digital Object) identifiers (only) for digital objects

- can provide both integrity and no middle man
- broadly used in modern software development (git, etc.)

IDOs and DIOs adress different needs

- for the core Software Heritage IDOs are enough
- we must not use DIOs for reproducibility

The Software Heritage IDO schema (see http://bit.ly/swhpids)

swh:1:**cnt**:94a9ed024d3859793618152ea559a168bbcbb5e2 full text of the GPL3 license

swh:1:**dir**:d198bc9d7a6bcf6db04f476d29314f157507d505 Darktable source code

swh:1:**rev**:309cf2674ee7a0749978cf8265ab91a60aea0f7d

a **revision** in the development history of Darktable

swh:1:**rel**:22ece559cc7cc2364edc5e5593d63ae8bd229f9f

release 2.3.0 of Darktable, dated 24 December 2016

swh:1:**snp**:c7c108084bc0bf3d81436bf980b46e98bd338453

a **snapshot** of the entire Darktable repository (4 May 2017, GitHub)

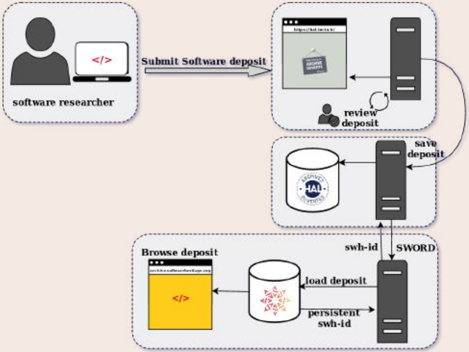
Current resolvers: archive.softwareheritage.org and n2t.org

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Deposit Scientific Software

Deposit software in HAL http://hal.inria.fr/hal-01738741



Generic mechanism:

- SWORD based
- review process
- versioning

How to do it:

- today: deposit .zip or .tar.gz file (*guide*)
- tomorrow:
  - provide *SWH id* and metadata
  - include *metadata file* for automatic metadata extraction
  - . . .

September 2018: open to all on https://hal.archives-ouvertes.fr/

The way to go to archive and reference scientific software

All features of Software Heritage *for free*

- intrinsic IDs (integrity, not dependent on resolvers!)
  - specification: `http://bit.ly/swhpids`
  - iPres2018 paper: `http://bit.ly/swhpidpaper`
- browse, download (now)
- metadata, licenses, provenance (plagiarism detection), classification (wip), ...

Coverage and uniformity

- one archive for all domains (industry included)
- reference *any* software, not just the deposited ones
- git-compatible identifiers greatly simplify workflows

Sustainability ... doors are open!

one infrastructure *independent* non profit foundation *worldwide* mirrors

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Demo time: let's highlight some features…

A "wayback machine" for software source code

- http://archive.softwareheritage.org/browse

Identification and sharing of billions of software artifacts

- http://bit.ly/swhpids for persistent identifiers

Depositing research software

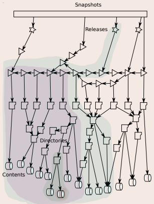
- http://bit.ly/swdepositblog

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

A revolutionary infrastructure for industry

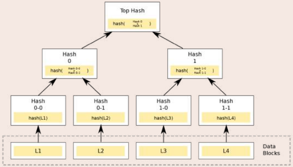
The *graph* of Software Development



All of the software development in a single graph!

- lookup by content hash
- wayback machine for software development
  - http://archive.softwareheritage.org/
- … and much more

The *blockchain* of Software Development



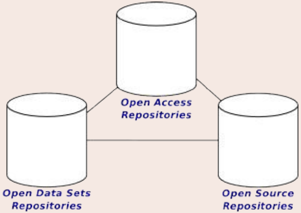
All of a software development… in a single Merkle graph!

Widely used crypto (e.g., Git, blockchains, IPFS, …)

- built-in deduplication
- intrinsic, unforgeable identifiers at all levels
- simplifies traceability (licensing, supply chain management)

A revolutionary infrastructure for research and innovation

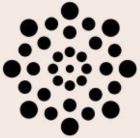
A *pillar* of Open Science



The *reference archive* of Research Software for Open Science

- curated deposit of research software
  - in collaboration with HAL, CCSD and Inria IES
  - now open *to all researchers*!
- intrinsic identifiers for reproducibility

Reference platform for *Big Code*



- unique observatory of all software development
- big data, machine learning paradise: classification, trends, coding patterns, code completion…

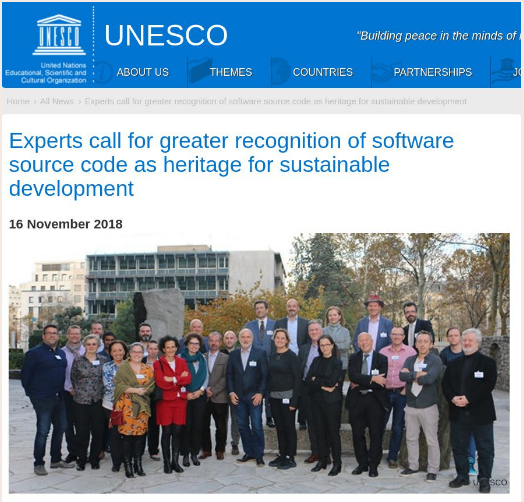
Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Raising Awareness

April 3rd 2017, Unesco Inria agreement

November 2018, Unesco Inria expert call



UNESCO
United Nations Educational, Scientific and Cultural Organization
"Building peace in the minds of
ABOUT US
THEMES
COUNTRIES
PARTNERSHIPS
JO

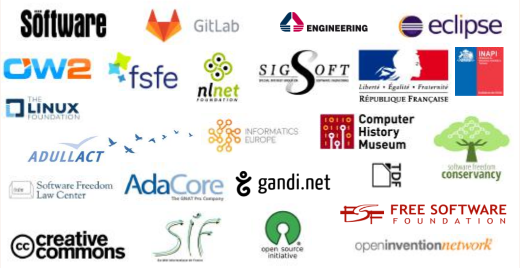
Home › All News › Experts call for greater recognition of software source code as heritage for sustainable development

Experts call for greater recognition of software source code as heritage for sustainable development

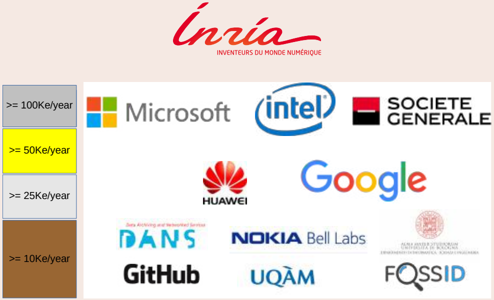
16 November 2018

Growing Support

Sharing the vision



Donors, members, sponsors



Research collaboration

Qwant

source code search engine

Global network

FOSSID

- first independent mirror
- increased reliability

The next steps

The Software Heritage Foundation

- independent
- long term mission
- multistakeholder

The community

- academia: Open Access, research
- industry: better software
- cultural heritage: all the software history

The mirror network

- resilience
- biodiversity

> *“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.”*
>
> *Thomas Jefferson*

You can help!

Many scientific and technological challenges

machine learning, classification, efficient graph queries, metadata, . . .

Reproducible Open Science

archive research software in SWH

reference it using *intrinsic identifiers*

build on SWH thematic portals for your discipline

Funding

- give *your own contribution* : `www.softwareheritage.org/donate`
- become a partner/sponsor/mirror : `sponsorship.softwareheritage.org`

Spread the word!

- *use* the archive and help others use it
- tell everybody about Software Heritage

Outline

1. Introductions
2. Software is everywhere. . .
3. . . . and we are not taking care of it!
4. Software Heritage
5. Under the hood: architecture
6. Under the hood: identifying billions of objects
7. Open Science
8. Demo time
9. Summing up: a revolutionary infrastructure
10. Building for the long term
11. Conclusion

Come in, we're open!

Library of Alexandria of code

- recover the past
- structure the future

A CERN for Software

- build better software
  - for industry
  - for society as a whole

Jean-François Abramatic, Roberto Di Cosmo, Stefano Zacchiroli
Building the Universal Archive of Source Code
Communication of the ACM, October 2018

Roberto Di Cosmo, Morane Gruenpeter, Stefano Zacchiroli
Identifiers for Digital Objects: the Case of Software Source Code Preservation
iPRES 2018: Intl. Conf. on Digital Preservation

Roberto Di Cosmo, Stefano Zacchiroli
Software Heritage: Why and How to Preserve Software Source Code
iPRES 2017: Intl. Conf. on Digital Preservation

Outline

12 Replicability/traceability

13 Strategy

14 Under the hood

URL decay disrupts 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.

URLs used in articles *decay*!

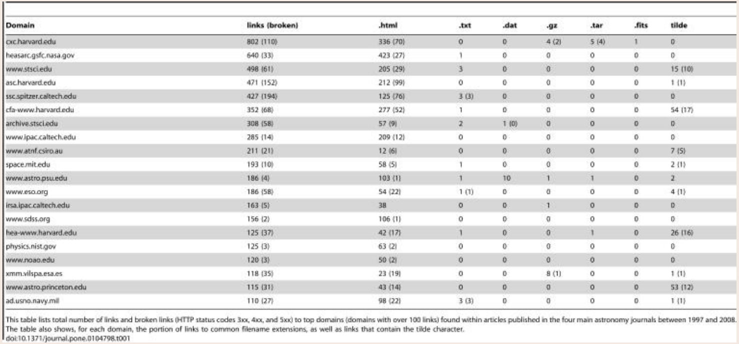
Analysis of *IEEE Computer* (Computer), and the *Communications of the ACM* (CACM): 1995-1999

- 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.

Scholar roster of broken links

An example from Astronomy



Domain	links (broken)	.html	.txt	.dat	.gz	.tar	.fits	tilde
cxc.harvard.edu	802 (110)	336 (70)	0	0	4 (2)	5 (4)	1	0
heasarc.gsfc.nasa.gov	640 (33)	423 (27)	1	0	0	0	0	0
www.stsci.edu	498 (61)	205 (29)	3	0	0	0	0	15 (10)
asc.harvard.edu	471 (152)	212 (99)	0	0	0	0	0	1 (1)
ssc.spitzer.caltech.edu	427 (194)	125 (76)	3 (3)	0	0	0	0	0
cfa-www.harvard.edu	352 (68)	277 (52)	1	0	0	0	0	54 (17)
archive.stsci.edu	308 (58)	57 (9)	2	1 (0)	0	0	0	0
www.ipac.caltech.edu	285 (14)	209 (12)	0	0	0	0	0	0
www.atnf.csiro.au	211 (21)	12 (6)	0	0	0	0	0	7 (5)
space.mit.edu	193 (10)	58 (5)	1	0	0	0	0	2 (1)
www.astro.psu.edu	186 (4)	103 (1)	1	10	1	1	0	2
www.eso.org	186 (58)	54 (22)	1 (1)	0	0	0	0	4 (1)
irsa.ipac.caltech.edu	163 (5)	38	0	0	1	0	0	0
www.sdss.org	156 (2)	106 (1)	0	0	0	0	0	0
hea-www.harvard.edu	125 (37)	42 (17)	1	0	0	1	0	26 (16)
physics.nist.gov	125 (3)	63 (2)	0	0	0	0	0	0
www.noao.edu	120 (3)	50 (2)	0	0	0	0	0	0
xmm.vilspa.esa.es	118 (35)	23 (19)	0	0	8 (1)	0	0	1 (1)
www.astro.princeton.edu	115 (31)	43 (14)	0	0	0	0	0	53 (12)
ad.usno.navy.mil	110 (27)	98 (22)	3 (3)	0	0	0	0	1 (1)

This table lists total number of links and broken links (HTTP status codes 3xx, 4xx, and 5xx) to top domains (domains with over 100 links) found within articles published in the four main astronomy journals between 1997 and 2008. The table also shows, for each domain, the portion of links to common filename extensions, as well as links that contain the tilde character.
doi:10.1371/journal.pone.0104798.t001

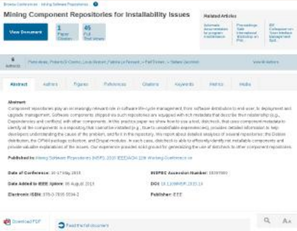
How Do Astronomers Share Data?
Pepe, Goodman, Muench, Crosas, Erdmann
dx.doi.org/10.1371/journal.pone.0104798

PLOS August 28, 2014

DOI limitations

Example: `doi:10.1109/MSR.2015.10`

- to find what 10.1109/MSR.2015.10 is, go to a *resolver* (e.g. doi.org)
- this returns `http://ieeexplore.ieee.org/document/7180064/`
- at this URL we find ...



Browse Conferences > Mining Software Repositories

Mining Component Repositories for Installability Issues

Abstract
Component repositories play an increasingly relevant role in software life-cycle management, from software distribution to end-user, to deployment and upgrade management. Software components shipped via such repositories are equipped with rich metadata that describe their relationship (e.g., dependencies and conflicts) with other components. In this practice paper we show how to use a tool, distcheck, that uses component metadata to identify all the components in a repository that cannot be installed (e.g., due to unsatisfiable dependencies), provides detailed information to help developers understanding the cause of the problem, and fix it in the repository. We report about detailed analyses of several repositories: the Debian distribution, the OPAM package collection, and Drupal modules. In each case, distcheck is able to efficiently identify not installable components and provide valuable explanations of the issues. Our experience provides solid ground for generalizing the use of distcheck to other component repositories.

Published in: Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on

Date of Conference: 16-17 May 2015

Date Added to IEEE Xplore: 06 August 2015

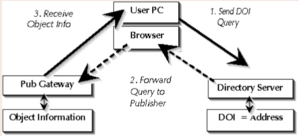
Electronic ISBN: [illegible]

INSPEC Accession Number: [illegible]

DOI: 10.1109/MSR.2015.10

Publisher: IEEE

Architecture of the DOI infrastructure



- DOI resolution *can change*
- content at URL *can change*
- no *intrinsic* way of noticing
- persistence based on *good will* of *multiple parties*

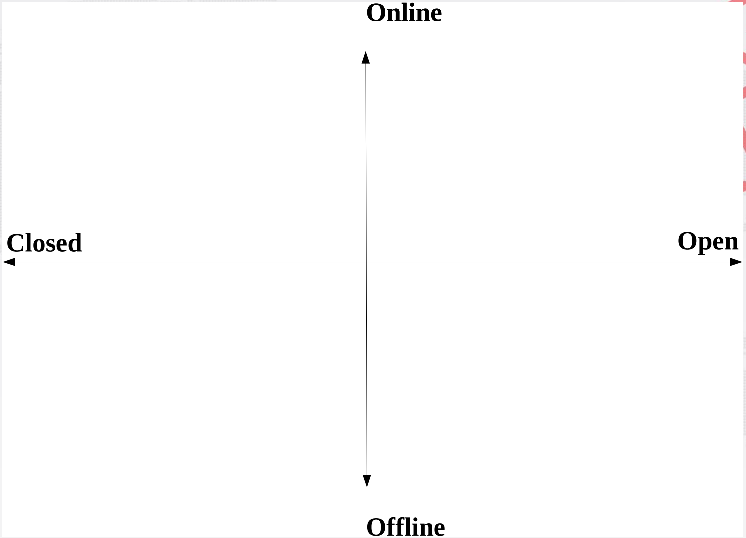
Outline

12 Replicability/traceability

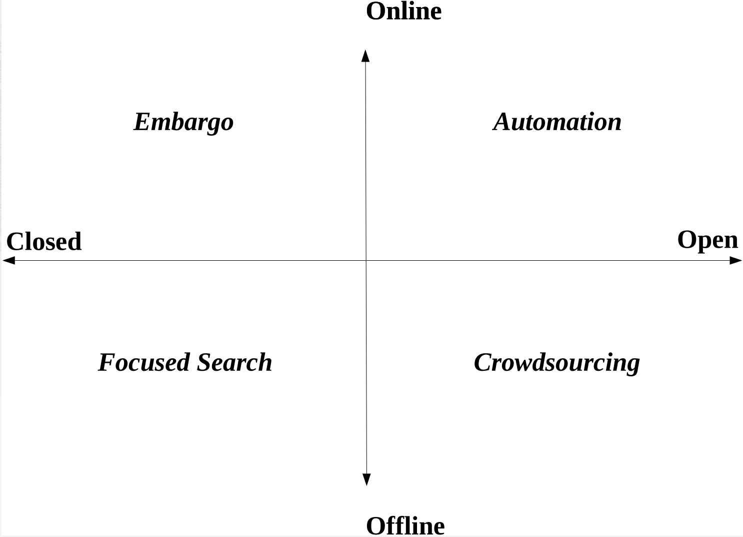
13 Strategy

14 Under the hood

All the source code



All the source code: strategy



Outline

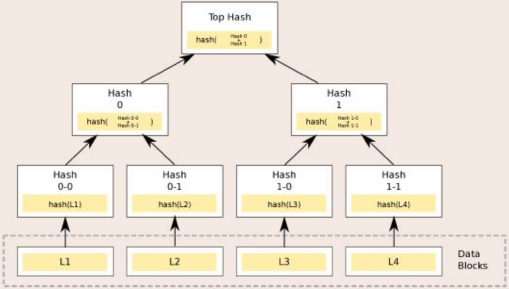
12 Replicability/traceability

13 Strategy

14 Under the hood

Much more than an archive!

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



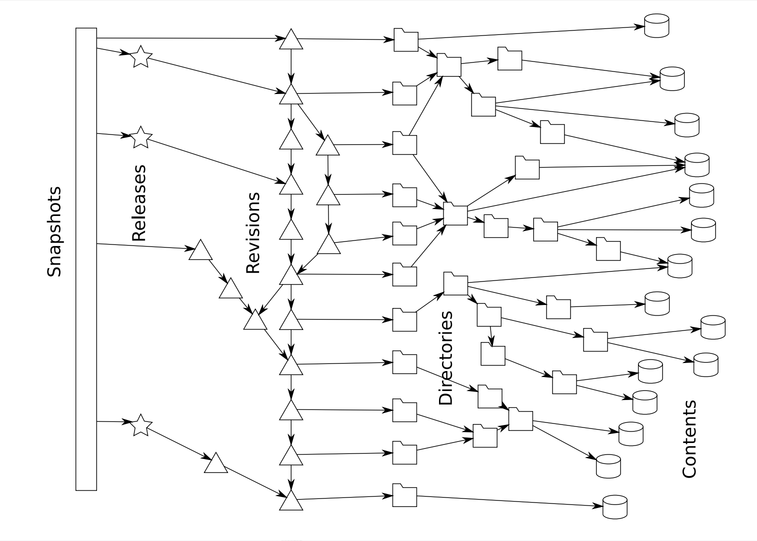
Combination of

- tree
- hash function

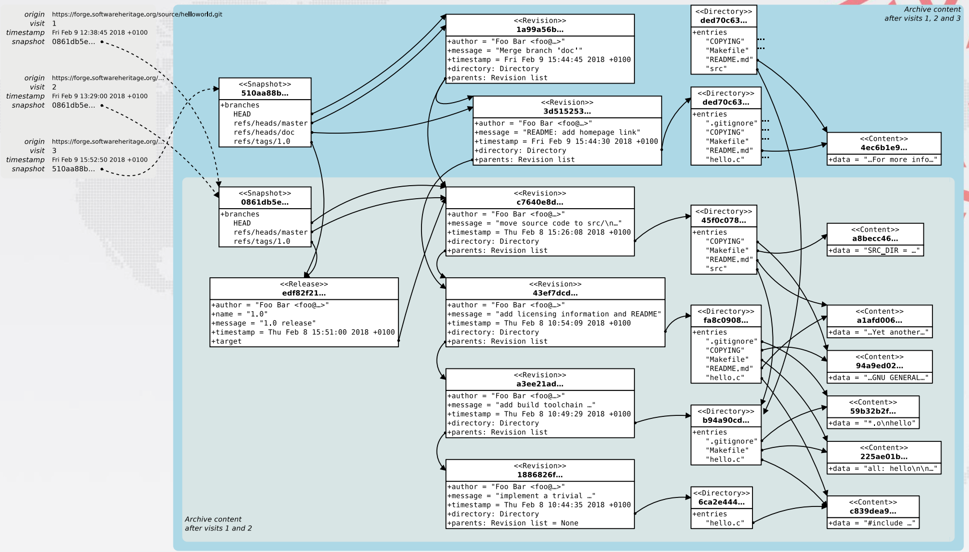
Classical cryptographic construction

- fast, parallel signature of large data structures
- widely used (e.g., Git, blockchains, IPFS, . . .)
- built-in deduplication

The archive in pictures



A bird's eye view



```
   origin  https://forge.softwareheritage.org/source/helloworld.git
    visit  1
timestamp  Fri Feb 9 12:38:45 2018 +0100
 snapshot  0861db5e...

   origin  https://forge.softwareheritage.org/...
    visit  2
timestamp  Fri Feb 9 13:29:00 2018 +0100
 snapshot  0861db5e...

   origin  https://forge.softwareheritage.org/...
    visit  3
timestamp  Fri Feb 9 15:52:50 2018 +0100
 snapshot  510aa88b...
```

Archive content after visits 1, 2 and 3

```
<<Snapshot>>
510aa88b...
+branches
   HEAD
   refs/heads/master
   refs/heads/doc
   refs/tags/1.0
```

```
<<Revision>>
1a99a56b...
+author = "Foo Bar <foo@…>"
+message = "Merge branch 'doc'"
+timestamp = Fri Feb 9 15:44:45 2018 +0100
+directory: Directory
+parents: Revision list
```

```
<<Revision>>
3d515253...
+author = "Foo Bar <foo@…>"
+message = "README: add homepage link"
+timestamp = Fri Feb 9 15:44:30 2018 +0100
+directory: Directory
+parents: Revision list
```

```
<<Directory>>
ded70c63...
+entries
   "COPYING"
   "Makefile"
   "README.md"
   "src"
```

```
<<Directory>>
ded70c63...
+entries
   ".gitignore"
   "COPYING"
   "Makefile"
   "README.md"
   "hello.c"
```

```
<<Content>>
4ec6b1e9...
+data = "…For more info…"
```

Archive content after visits 1 and 2

```
<<Snapshot>>
0861db5e...
+branches
   HEAD
   refs/heads/master
   refs/tags/1.0
```

```
<<Release>>
edf82f21...
+author = "Foo Bar <foo@…>"
+name = "1.0"
+message = "1.0 release"
+timestamp = Thu Feb 8 15:51:00 2018 +0100
+target
```

```
<<Revision>>
c7640e8d...
+author = "Foo Bar <foo@…>"
+message = "move source code to src/\n…"
+timestamp = Thu Feb 8 15:26:08 2018 +0100
+directory: Directory
+parents: Revision list
```

```
<<Revision>>
43ef7dcd...
+author = "Foo Bar <foo@…>"
+message = "add licensing information and README"
+timestamp = Thu Feb 8 10:54:09 2018 +0100
+directory: Directory
+parents: Revision list
```

```
<<Revision>>
a3ee21ad...
+author = "Foo Bar <foo@…>"
+message = "add build toolchain …"
+timestamp = Thu Feb 8 10:49:29 2018 +0100
+directory: Directory
+parents: Revision list
```

```
<<Revision>>
1886826f...
+author = "Foo Bar <foo@…>"
+message = "implement a trivial …"
+timestamp = Thu Feb 8 10:44:35 2018 +0100
+directory: Directory
+parents: Revision list = None
```

```
<<Directory>>
45f0c078...
+entries
   "COPYING"
   "Makefile"
   "README.md"
   "src"
```

```
<<Directory>>
fa8c0908...
+entries
   ".gitignore"
   "COPYING"
   "Makefile"
   "README.md"
   "hello.c"
```

```
<<Directory>>
b94a90cd...
+entries
   ".gitignore"
   "Makefile"
   "hello.c"
```

```
<<Directory>>
6ca2e444...
+entries
   "hello.c"
```

```
<<Content>>
a8becc46...
+data = "SRC_DIR = …"
```

```
<<Content>>
a1afd006...
+data = "…Yet another…"
```

```
<<Content>>
94a9ed02...
+data = "…GNU GENERAL…"
```

```
<<Content>>
59b32b2f...
+data = "*.o\nhello"
```

```
<<Content>>
225ae01b...
+data = "all: hello\n\n…"
```

```
<<Content>>
c839dea9...
+data = "#include …"
```