

Software Heritage

How to use it for Science and how to contribute

Roberto Di Cosmo

`roberto@dicosmo.org`

May 23rd, 2018



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

- 1 Software all around us
- 2 Source code for Science
- 3 The Software Heritage initiative
- 4 Using the Software Heritage archive for Science
- 5 Building for the long term
- 6 Conclusion





Software embodies our collective **Knowledge** and **Cultural Heritage**

Harold Abelson, Structure and Interpretation of Computer Programs (1st ed.)

1985

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

Quake 2 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[1][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 also 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

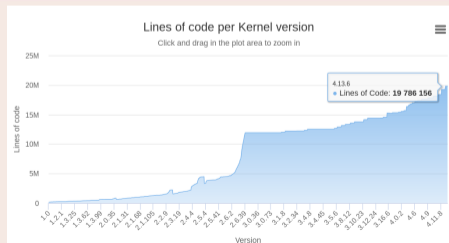
Apollo 11 Guidance Computer (~60.000 lines), 1969



"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!

are we taking care of all this?

Software is spread all around

Debian CPAN
Sourceforge Gitorious
Maven Inria
Bitbucket
Git GitHub
BerliOs CTAN
GoogleCode GitLab Adullact CRAN



A word cloud centered on a faint world map background. The words are of various sizes and colors (purple, brown, blue, green). The largest word is 'damage' in brown. Other prominent words include 'disaster' in purple, 'malicious' in brown, 'deletion' in blue, 'obsolete' in purple, 'attack' in blue, 'format' in green, 'reference' in blue, 'storage' in brown, 'dangling' in blue, 'wear' in brown, 'corruption' in brown, 'encryption' in blue, 'dependencies' in blue, 'aging' in blue, 'media' in brown, and 'tear' in purple. The background also features a decorative pattern of red and orange triangles on the right side.

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

- 
- 1 Software all around us
 - 2 Source code for Science
 - 3 The Software Heritage initiative
 - 4 Using the Software Heritage archive for Science
 - 5 Building for the long term
 - 6 Conclusion

Analysis of 613 papers

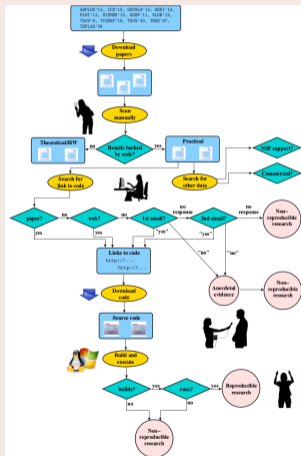
- 8 ACM conferences: ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12
- 5 journals: TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34

all very practical oriented

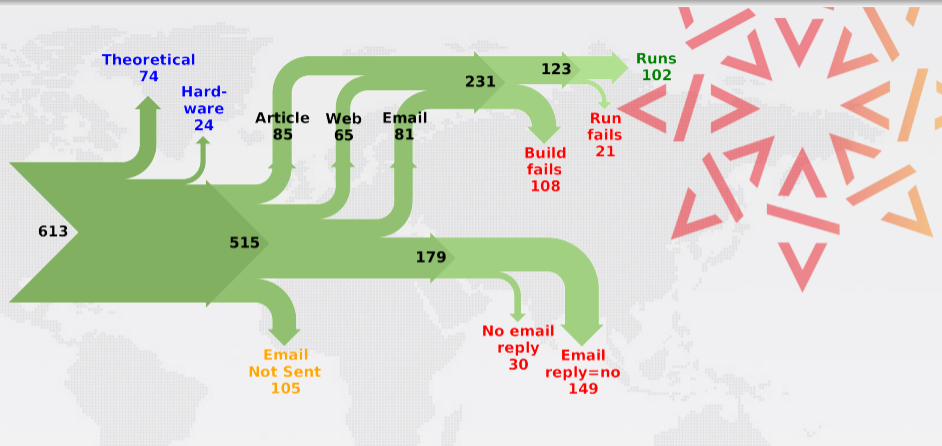
The basic question

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!

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.

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.

An example from Astronomy

Domain	links (broken)	.html	.txt	.dat	.gz	.tar	.fits	tilde
oac.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)
esc.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	1	0	2
www.eso.org	186 (58)	54 (22)	1 (1)	0	0	0	0	4 (1)
isa.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
rmm.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](https://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 ...

Mining Component Dependencies for Installability Issues

View Document 1 Page 45 Full Text Issues

Related actions: Add to Favorites, Add to My Library, Add to My Lists, Add to My Alerts

6 Authors: PAVAN KUMAR, PRADEEP D. CHAKRA, LOUIS DEBIEVE, PIERRE LA FRAISSE, YVES FATHALLAH, DENIS DELCOURT

Abstract Authors Figures References Citations Keywords Metrics Media

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 as 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, *Discover*, that uses component metadata to identify all the components in a repository that cannot be installed (e.g., due to unresolvable dependencies), provides detailed information to help developers understanding the source of the problem, and fix it in the repository. We report about detailed analyses of several repositories: the Debian distribution, the Chrome package collection, and Cplusplus modules. In each case, *Discover* is able to efficiently identify not installable components and provide valuable explanations of the issues. Our experience provides solid ground for generating the size of *Discover* to other component repositories.

Published in: Mining Software Repositories (MSR), 2015 IEEE/ACM LBNV Workshop Conference on

Date of Conference: 01-17 May 2015

Date added to IEEE Xplore: 01 Aug 01 2015

Electronic ISSN: 1570-1700-0204-0

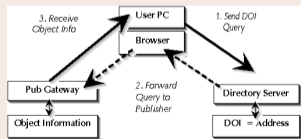
INSPEC Accession Number: 15376700

DOI: 10.1109/MSR.2015.10

Publisher: IEEE

Download PDF Read the Full Document

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*

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!

- 
- 1 Software all around us
 - 2 Source code for Science
 - 3 The Software Heritage initiative**
 - 4 Using the Software Heritage archive for Science
 - 5 Building for the long term
 - 6 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

Cultural Heritage



Industry



Research



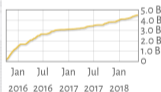
Education



Software Heritage

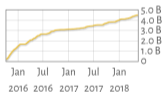
Source files

4,536,067,027



Commits

1,024,675,748



Projects

83,801,775



Technology

- transparency and FOSS
- replicas all the way down

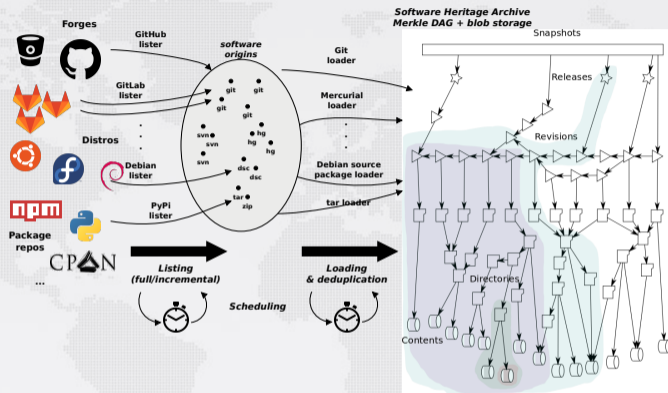
Content

- intrinsic identifiers
- facts and provenance

Organization

- non-profit
- multi-stakeholder

Architecture (simplified)



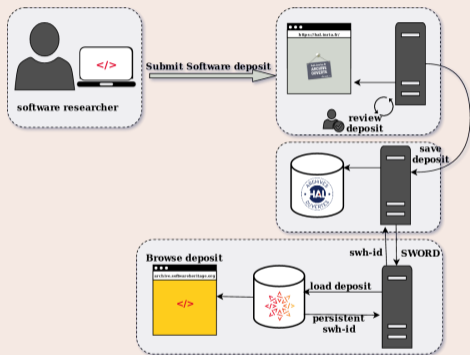
- full development history permanently archived
- origins: GitHub (automated), Debian (automated), Gitorious, Google Code, GNU
- ~200Tb raw contents, ~10Tb graph (7+Bn nodes, 60+Bn edges)

- 1 Software all around us
- 2 Source code for Science
- 3 The Software Heritage initiative
- 4 Using the Software Heritage archive for Science**
- 5 Building for the long term
- 6 Conclusion



Deposit software in HAL

<http://bit.ly/swhdeposithalen>



Generic mechanism:

- SWORD based
- review process
- versioning

How to do it:

- today: deposit .zip file
- tomorrow:
 - *provide SWH id and metadata*
 - *provide SWH id, metadata is extracted*
 - ...

The way to go for publishing research software, spread the word!



Large scale *repeatable* software studies...

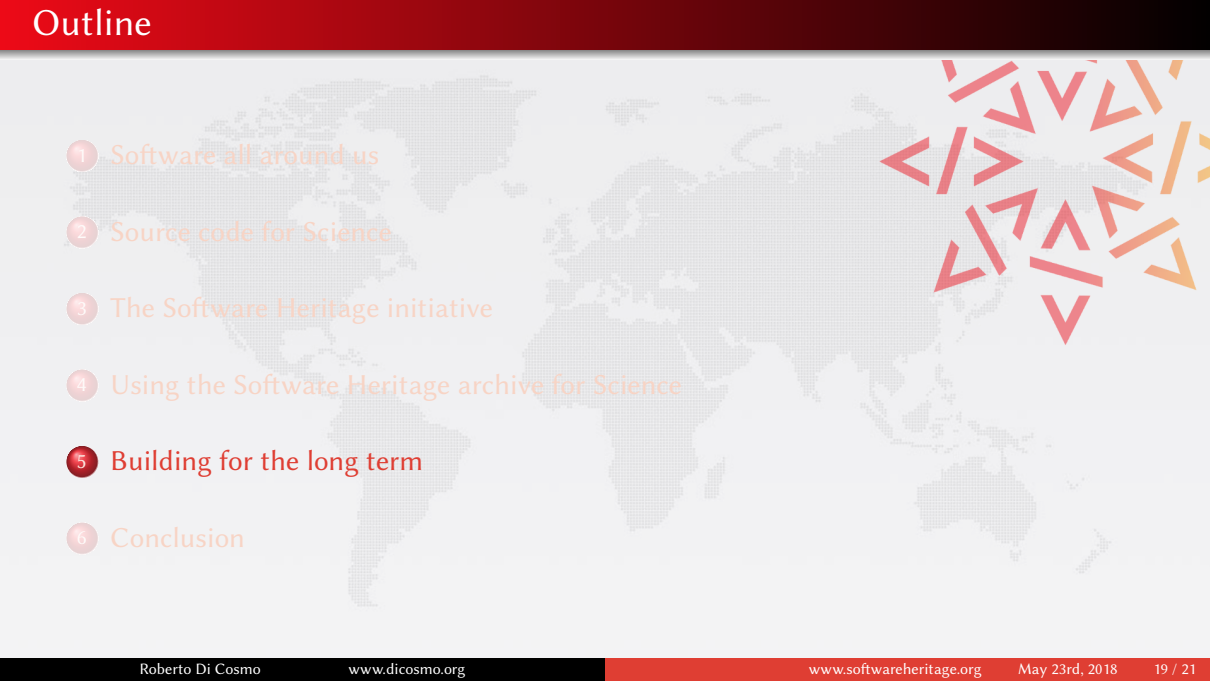
- vulnerability detection
- dependency analysis
- pattern elicitation
- automatic classification ...

... need a uniform representation

Software Heritage has **one data model** for all forges/VCS...

... yes, we do **data normalization** of software evolutiona!

Coming soon to a platform near you!

- 
- 1 Software all around us
 - 2 Source code for Science
 - 3 The Software Heritage initiative
 - 4 Using the Software Heritage archive for Science
 - 5 Building for the long term**
 - 6 Conclusion

Landmark Inria Unesco agreement, April 3rd, 2017



Sharing the vision



Contributing to the mission



You can help!

Connect Inria's forges!

see <http://bit.ly/swhlisters>

- ★★★ install FusionForge plugin, develop a lister
- ★★★ develop Gitlab plugin, develop lister (also for framagit!)

Contribute to the development see <http://forge.softwareheritage.org>

- ★★ listers/loaders for other unsupported forges, VCS
- ★★ Web UI improvements

Spread the word!

- help research teams *use* the archive
- tell everybody about Software Heritage

Funding

- pester *companies* to become sponsors : sponsorship.softwareheritage.org
- give *your own contribution* : www.softwareheritage.org/donate

- 1 Software all around us
- 2 Source code for Science
- 3 The Software Heritage initiative
- 4 Using the Software Heritage archive for Science
- 5 Building for the long term
- 6 Conclusion



Come in, we're open!



Software Heritage



www.softwareheritage.org

@swheritage

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