Preserving Software and Data

Ensuring Availability and Traceability

Roberto Di Cosmo
INRIA and IRIF

roberto@dicosmo.org

November 8, 2016

Software Heritage

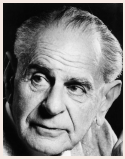
How we built our scientific knowledge

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*!

Reproducibility in the digital age

For an experiment involving software, we need

- open access to the scientific article describing it
- open data sets used in the experiment
- source code of all the components
- environment of execution
- stable references between all this

Remark

The first two items are already widely discussed!

… what about *software*?

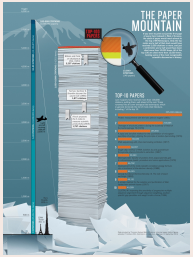
Software is Knowledge

Software is *an essential component* of modern scientific research

Top 100 papers (Nature, October 2014)

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

`http://www.nature.com/news/the-top-100-papers-1.16224`



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

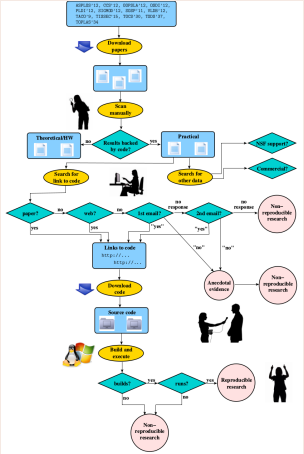
- 8 ACM conferences: ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12
- 5 journals: TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34

all very practical oriented

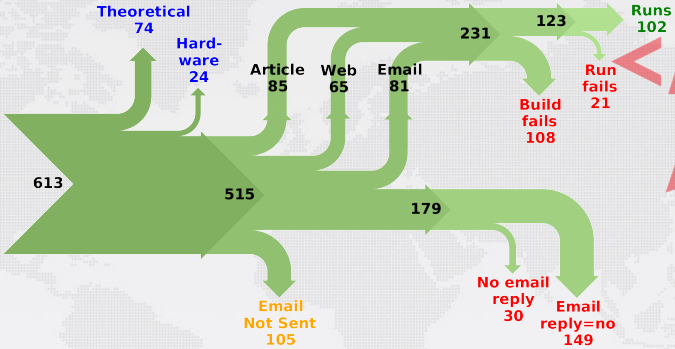
The basic question

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!

In Software Engineering

Even higher expectations, and yet similarly disappointing results `http://fr.slideshare.net/carloghezzi18/icse-2009-keynote-15919951`

Reference journal

ACM Transactions on Software Engineering and Methodology (TOSEM)

- analysis by Carlo Ghezzi, in 2009, of TOSEM from 2001 to 2006
- 60% of papers refer to a tool
- 20% only are *installable*

Reference conference

International Conference on Software Engineering (ICSE)

- analysis by Zannier, Melrik, Maurer 2006
- complete absence of replication studies

Pressure to make research code available is now raising

Evaluation of software artefacts (optional)

- tools are usable, in line with expectations
- started as a contest in 2011 (ESEC/FSE) (winner *Vouillon and Di Cosmo*)
- now going mainstream: POPL'17, POPL'16, ECOOP'16, OOPSLA'16, CGO'16, VISSOFT'16, PLDI'16, CGO'15, PPoPP'15, VISSOFT'15, ISSTA'15, OOPSLA'15, PLDI'15, POPL'15, CAV'15, ECOOP'15, FSE'15, ISSTA'14, OOPSLA'14, PLDI'14, ECOOP'14, FSE'14, SAS'13, OOPSLA'13, ECOOP'13, FSE'13, FSE'11

Use the Source, Luke!

Some people claim that having (all) the source of the code used in an experiment is *not worth the effort* (see "Replicability is not Reproducibility: Nor is it Good Science", Chris Drummond, ICML 2009)

Sure, diversity *is* important, but:

- Source code is like the proof used in a theorem: can we really accept *Fermat statements* like "the details are omitted due to lack of space"?
- modern complex systems makes even the simplest experiment depend on a wealth of components and configuration options
- access to *all* the source code is not just necessary to *reproduce*, it is also useful to *evolve and modify*, to *build new experiments* from the old ones

Software 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 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;
}
```

Network 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."

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

Software is fragile

like all digital information, FOSS is fragile

- inconsiderate and/or malicious code loss (e.g., Code Spaces)
- business-driven code loss (e.g., Gitorious, Google Code)
- for obsolete code: physical media decay (data rot)

If a website disappears you go to the Internet Archive…

… where do you go if (a repository on) GitHub goes away?

Software is spread all around

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* the source code?

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.

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

Cool URLs (should not) change

What makes a cool URI?

A cool URI is one which does not change.
What sorts of URI change?
URIs don't change: *people change them.*

Tim Berners Lee, 1998
`https://www.w3.org/Provider/Style/URI`

Yes, *people* change them...

sometimes behind your back!

Disruption of the web of reference: Inria's own Gforge

[siteadmin-Bugs][#17468] Urls of release files has silently changed

siteadmin-bugs@gforge.inria.fr via dicosmo.org 21 mai
À noreply

anglais › français Traduire le message Désactiver pou

siteadmin-Bugs [#17468] was changed at 2014-05-21 11:11 by Vincent Lefèvre
You can respond by visiting:
https://gforge.inria.fr/tracker/?funcPtail&atid 0&aid 468&group_id1

Status: Open
Priority: 3
Submitted By: Roberto Di Cosmo (robertodicosmo)
Assigned to: Nobody (None)
Summary: Urls of release files has silently changed
Category: gênant
Group: None
Resolution: None

Initial Comment:
The url of release files has silently changed: for example, the original release file

https://gforge.inria.fr/frs/download.php/file/31910/cudf-0.6.3.tar.gz

now gives an empty file when downloading it, while the actual url changed to

https://gforge.inria.fr/frs/download.php/31910/cudf-0.6.3.tar.gz

There are surely good reasons for this, but I would like to stress the fact that we *need* to be able to rely on permanent URLs for releasing our software... these urls end up embedded in other tools and software, and changing them is a source of unneeded problems.

Fixed, adding a redirection, by the Gforge team

in *1 day* this one was fixed!

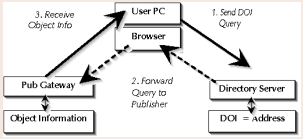
Not always that lucky, though …

The Digital Object Identifier (DOI)

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 …

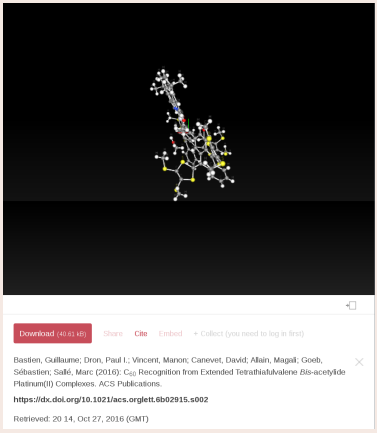
Architecture of the DOI infrastructure



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

Real world examples…

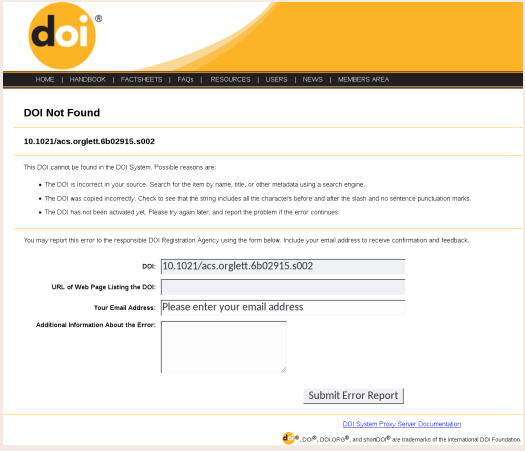
An image from *figshare*



let's click on the DOI…

… may indeed not work

DOI not found



Yes, we report broken links/dois

Thank you for reporting this error.

Problem DOI:	10.1021/acs.orglett.6b02915.s002
Publisher who owns the prefix 10.1021:	American Chemical Society
User's email address:	doi@dicosmo.org
Referring Page:	https://figshare.com
Comments:	

The DOI and comments (if provided) have been logged by CrossRef and forwarded to the publisher to correct the problem. Possible reasons for the error are:

- the DOI has been created but has not been registered by the publisher (this could be an error or it could be a timing issue and the DOI will be registered in the next few days)
- the DOI is cited incorrectly in the source
- the DOI does not resolve due to a system problem

Maintaining the integrity of DOIs is very important to CrossRef and we appreciate your help.

Stocktaking

Software is

- an *essential component* of modern scientific research
- in *all fields* of science

And yet

we are doing a very poor job at keeping (trace of) it, let alone make it *reproducible*

- no single place where to make available our source code
- no valid *intrinsic* digital identifier in sight

We can do better

and we started doing it

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.

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*

Software Heritage's approach

Availability

- collect *all* software from forges
- *replicate* the archive in a network of mirrors

Traceability

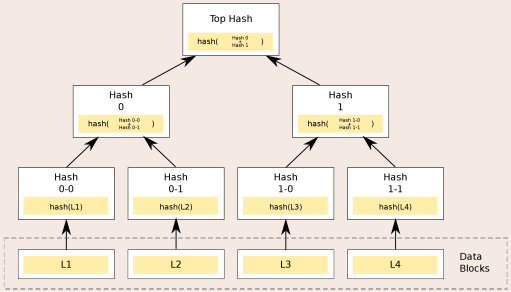
- *unique* identifiers : use *cryptographic hashes*, derived from the software itself
- access *the content* using this identifier

Uniformity

- version control data model designed to *represent all the others*

Merkle trees

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



Combination of

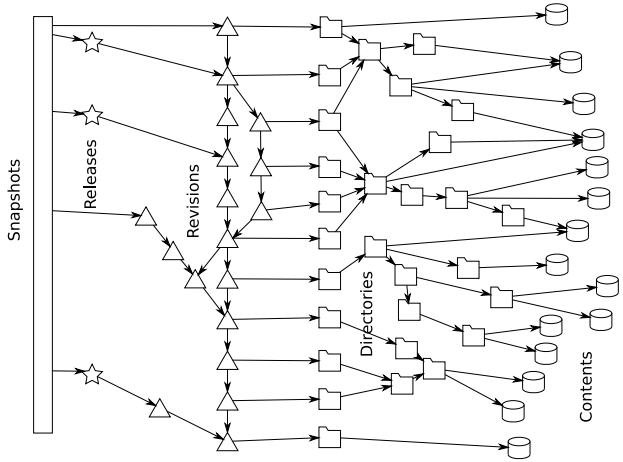
- tree
- hash function

Classical cryptographic construction

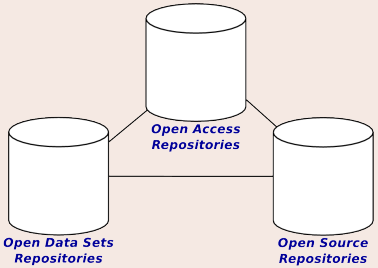
- fast, parallel signature of large data structures
- widely used by *Git*, *Bitcoin*, etc.
- natural extension: Merkle *DAG*

The archive in a few pictures

A giant (extended) Merkle DAG



The Knowledge Conservancy Magic Triangle



Legenda (links are important!)

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

What could these links look like?

Links to *software source code* in an article

Leverage Software Heritage as universal archive:

set of files `swh:tree:06741c8c37c5a384083082b99f4c5ad94cd0cd1f`
id of tree object listing all the files in a project (at a given time)

revision `swh:rev:7598fb94d59178d65bd8d2892c19356290f5d4e3`
id of commit object which a tree and (a pointer to) the history

metadata this *may* be a DOI

Links to *data* in *software source code*

- external linking mechanisms *that guarantee integrity*
  - git lfs
  - git annex
- need to extend them into a generic, VCS independent solution

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

Some numbers

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

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…

Next steps, for the research community

Create a workflow

- allow researchers to *deposit* code
- integrate with *Open Access* platform
- allow *metadata* edition/enrichment
- integrate with *WikiData* - like initiatives
- coordinate with software citation initiatives

An ambitious, worldwide initiative

Inria as initiator

- founding partner of the W3C,
- creating a non profit, international organisation

Support and *first partners*

ACM, Nokia Bell Labs, Creative Commons, DANS, Eclipse, Engineering, FSF, OSI, GitHub, GitLab, IEEE, Informatics Europe, Microsoft, OIN, OW2, SIF, SFC, SFLC, The Document Foundation, The Linux Foundation, …

Going global

building an *open, multistakeholder, nonprofit* global organisation

Come in, we're open!

Everybody is needed!

researchers	*many* scientific challenges (please ask!)
developers	Software Heritage is itself Open Source!
transversal	find the many source code repositories
partners	contribute to the effort

Now open

`www.softwareheritage.org` - *sponsoring, partnerships*
`wiki.softwareheritage.org` - *working groups, leads*
`forge.softwareheritage.org` - *our own code*

Questions?