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 an essential component of modern scientific research

Top 100 papers (Nature, October 2014)

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

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

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

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

**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
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)

```c
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalves = 1.5F;
    x2 = number * 0.5F;
    y = number;
    i = *( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759dfL & ( i >> 1 ); // what the fuck?
    y = *( float * ) &i;
    y = y * ( threehalves - ( x2 * y * y ) ); // 1st iteration
    // y = y * ( threehalves - ( x2 * y * y ) ); // 2nd iteration, this
    // can be removed

    return y;
}
```

Network queue in Linux (excerpt)

```c
/*
 * SFBU uses two B([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 SFBU_BUCKET_SHIFT 4
#define SFBU_NUMBUCKETS (1 << SFBU_BUCKET_SHIFT) /* N bins per level */
#define SFBU_BUCKET_MASK (SFBU_NUMBUCKETS - 1)
#define SFBU_LEVELS (32 / SFBU_BUCKET_SHIFT) /* L */

/* SFBU algo uses a virtual queue, named "b1n" */
struct sbf_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.”

Roberto Di Cosmo

Availability and traceability

November 8, 2016
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
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

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

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*


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


### An example from Astronomy

<table>
<thead>
<tr>
<th>Domain</th>
<th>links (broken)</th>
<th>.html</th>
<th>.txt</th>
<th>.dat</th>
<th>.gz</th>
<th>.tar</th>
<th>.fits</th>
<th>tilde</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.harvard.edu</td>
<td>802 (110)</td>
<td>336 (70)</td>
<td>0</td>
<td>0</td>
<td>4 (2)</td>
<td>5 (4)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>hesarc.gsfc.nasa.gov</td>
<td>640 (33)</td>
<td>421 (27)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.stsci.edu">www.stsci.edu</a></td>
<td>496 (61)</td>
<td>205 (29)</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15 (10)</td>
</tr>
<tr>
<td>ast.harvard.edu</td>
<td>471 (132)</td>
<td>212 (99)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 (1)</td>
</tr>
<tr>
<td>icpcmp.caltech.edu</td>
<td>427 (194)</td>
<td>125 (78)</td>
<td>3 (3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cfa-www.harvard.edu</td>
<td>352 (68)</td>
<td>277 (52)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>54 (17)</td>
</tr>
<tr>
<td>archive.stsci.edu</td>
<td>308 (58)</td>
<td>57 (9)</td>
<td>2</td>
<td>1 (1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.ipac.caltech.edu">www.ipac.caltech.edu</a></td>
<td>285 (14)</td>
<td>209 (12)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.mrdf.com.au">www.mrdf.com.au</a></td>
<td>211 (21)</td>
<td>12 (6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7 (5)</td>
</tr>
<tr>
<td>space.mit.edu</td>
<td>193 (10)</td>
<td>58 (5)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 (1)</td>
</tr>
<tr>
<td><a href="http://www.astronomy.edu">www.astronomy.edu</a></td>
<td>186 (6)</td>
<td>103 (1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td><a href="http://www.nrao.org">www.nrao.org</a></td>
<td>186 (58)</td>
<td>54 (22)</td>
<td>1 (1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4 (1)</td>
</tr>
<tr>
<td>apc.caltech.edu</td>
<td>163 (5)</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.sdss.org">www.sdss.org</a></td>
<td>156 (2)</td>
<td>106 (1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.harvard.edu">www.harvard.edu</a></td>
<td>125 (37)</td>
<td>42 (17)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>26 (16)</td>
</tr>
<tr>
<td>physics.nist.gov</td>
<td>125 (3)</td>
<td>63 (2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.nrao.edu">www.nrao.edu</a></td>
<td>120 (3)</td>
<td>50 (2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://www.apls.eu.es">www.apls.eu.es</a></td>
<td>116 (35)</td>
<td>23 (18)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 (1)</td>
<td>0</td>
<td>1 (1)</td>
</tr>
<tr>
<td><a href="http://www.astronomy.princeton.edu">www.astronomy.princeton.edu</a></td>
<td>115 (31)</td>
<td>43 (14)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>53 (12)</td>
</tr>
<tr>
<td>ad.usno.navy.mil</td>
<td>110 (27)</td>
<td>98 (22)</td>
<td>3 (3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 (1)</td>
</tr>
</tbody>
</table>

This table lists total number of links and broken links (HTTP status codes 50x, 40x, and 50x) 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. 

dx.doi.org/10.1371/journal.pone.0104798

---

**How Do Astronomers Share Data?**

Pepe, Goodman, Muench, Crosas, Erdmann

dx.doi.org/10.1371/journal.pone.0104798
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

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
Good citizen

Yes, we report broken links/doi's

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.
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
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.
### 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 Zacchirolu
- Nicolas Dandrimont (Engineer)
- Antoine Dumont (Engineer)
- and Jordi, Quentin and Guillaume

Scientific advisors
- Serge Abiteboul (French Science 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!
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?