Software Heritage

A new essential infrastructure for Software Source Code

Roberto Di Cosmo

Director, Software Heritage
Computer Science full professor, Inria and IRIF
roberto@dicosmo.org

December 7th, 2017

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

Software Heritage in a nutshell

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

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

Preserving our heritage, enabling better software and better science for all

Reference catalog

find and reference all the source code

Universal archive

preserve all the source code

Research Infrastructure

Enable analysis of all the source code

Cultural Heritage | **Industry** | **Research** | **Education**

Software Heritage

Open approach

- open source
- transparency

In for the long haul

- non profit, replication
- intrinsic identifiers

Exhaustive

- all software
- open to all communities

Growing Support

Raising awareness

April 3rd, 2017: landmark Inria Unesco agreement on source code access and preservation

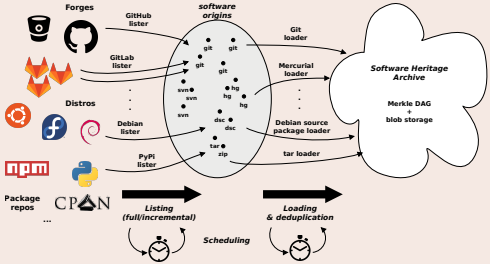
Sharing the vision

Sponsoring our work

Outline

Zoom on the collection phase

Much more than an archive!



- GitHub
- Debian, GNU
- Gitorious, Google Code
- Bitbucket (WIP), FusionForge (WIP)
- *add your own plugins!*

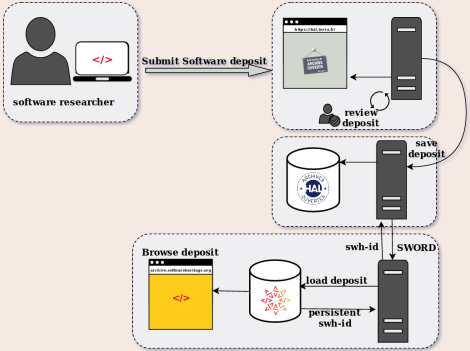
Important properties

- mission: exhaustive and up to date collection of source code, *specifically*
- strategy: automatic harvesting + *deposit* from *selected* sources

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

The research software (deposit) use case

Beta testing with HAL https://hal.archives-ouvertes.fr/



Generic mechanism:

- SWORD based
- review process
- versioning
- *industry chimes in* (details on demand)

Variants:

- just provide SWH hash and metadata
- just provide SWH hash, extract metadata
- ...

Feedback is welcome

drop me a line if you want to join the test group

Coming soon: access using intrinsic IDs

Getting close to it … click in the paper and view the source

"Our Parmap.parmap and Parmap.parfold functions may be used to seamlessly …"

Software Heritage https://archive.softwareheritage.org/browse/content/sha1_git:2a0837708151d76edf28fdbb90dc3eabc676cff3/?path=d5f31f6885306855 7d0c24639ba2a5e71c85cc9a/parmap.mli#L83-L92

```
val parmap : ?init:(int -> unit) -> ?finalize:(unit -> unit) -> ?ncores:int
  (** [parmap  ~ncores:n f (L l) ] computes [List.map f l]
      by forking [n] processes on a multicore machine.
      [parmap  ~ncores:n f (A a) ] computes [Array.map f a]
      by forking [n] processes on a multicore machine.
      If the optional [chunksize] parameter is specified,
      the processes compute the result in an on-demand fashion
      on blocks of size [chunksize]; this provides automatic
      load balancing for unbalanced computations, but the order
      of the result is no longer guaranteed to be preserved. *)

(** {6 Parallel iteration} *)

val pariter : ?init:(int -> unit) -> ?finalize:(unit -> unit) -> ?ncores:int
  (** [pariter  ~ncores:n f (L l) ] computes [List.iter f l]
      by forking [n] processes on a multicore machine.
      [parmap  ~ncores:n f (A a) ] computes [Array.iter f a]
      by forking [n] processes on a multicore machine.
      If the optional [chunksize] parameter is specified,
```

Selected unique benefits

there are more!

All features of Software Heritage *for free*

- intrinsic IDs (integrity, not just DIOs!), browse, download (now)
- metadata, licenses, provenance analysis (plagiarism detection), classification (wip)
- and many more (powerful connections with SE and Industry)

Coverage and uniformity

- one archive for all domains (industry included)
- you can reference *any* software, not just the deposited one

  (thanks D. Katz for pointing this out)
- git-compatible identifiers greatly simplify workflows

Sustainability

… doors are open!

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

Come in, we're open!

Software Heritage

www.softwareheritage.org @swheritage

Talks, slides: annex.softwareheritage.org/public/talks

Get involved

sponsoring / partnership	sponsorship.softwareheritage.org
working groups, leads	wiki.softwareheritage.org
our own code	forge.softwareheritage.org
metadata	RDA source code IG

Outline

Our challenge in the PID arena

Long term

Identifiers must be there for the long term

No middle man

Identifiers must be meaningful even if resolvers go away

Integrity, not just naming

Identifier must ensure that the retrieved object is the intended one

Uniqueness by design

one name identifies a single object, and each object has only one name

Exploring the PID landscape

A lot of options out there...

URL, URI, PURL, URN, ARK, DOI, ...

... some are widely used

- articles
- data
- even software artefacts!

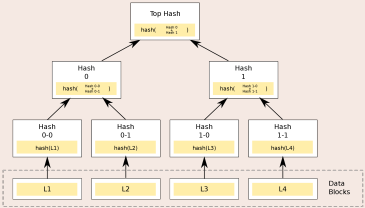
We can get no satisfaction

of all the key criteria

we adopted something radically different

Intrinsic identifiers in Software Heritage

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



Combination of

- tree
- hash function

Classical cryptographic construction

fast, parallel signature of large data structures, built-in deduplication

- satisfies all three criteria
- widely used in industry (e.g., Git, nix, blockchains, IPFS, ...)

Back to basics: DIOs vs. IDOs

DIO (digital identifier of an object)

- digital identifiers for traditional (non digital) objects
  - epistemic complications (manifestations, versions, locations, etc.)
  - significant governance issues, ...

IDO (identifier of a digital object)

- (digital) identifier for digital objects
  - much simpler to build/handle
  - can (and must) be intrinsic

Separation of concerns

- yes, we need both DIOs and IDOs
- no, we must not mistake DIOs for IDOs (and viceversa)

Working together

Example: links to *software source code* in an article

Leveraging the Software Heritage universal archive:

set of files `swh:1:tree:06741c8c37c5a384083082b99f4c5ad94cd0cd1f`
id of tree object listing all the files in a project (at a given time)

revision `swh:1:rev:7598fb94d59178d65bd8d2892c19356290f5d4e3`
id of commit object which a tree and (a pointer to) the history

metadata this *will* involve some form of DIO

- and we get all the complications back

Outline

Our role : handle *all* the *software source code*

At the end of the process

Explicit deposit, coordinated with the publisher

- store the *final* source code (no garbage)
- store only public source code
- N.B.: no embargo or access control (yet)

During the review

Access to the largest available source code base

- provenance, plagiarism detection (for new code)
- metrics (for long standing projects)

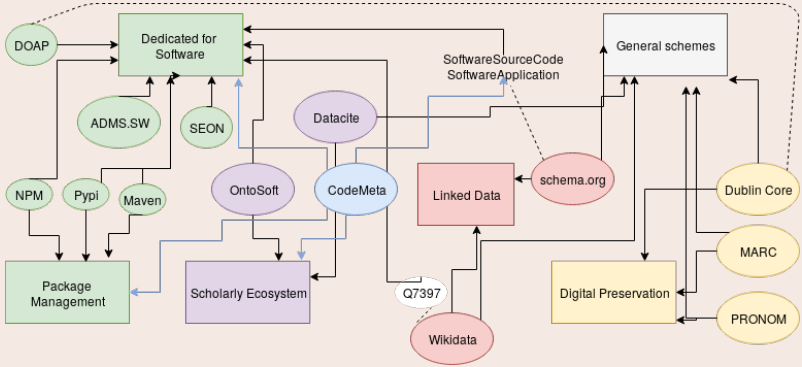
Later on

- Support embargo/access control

Outline

Collecting metadata for 60+ million projects

Landscape of Software Ontologies

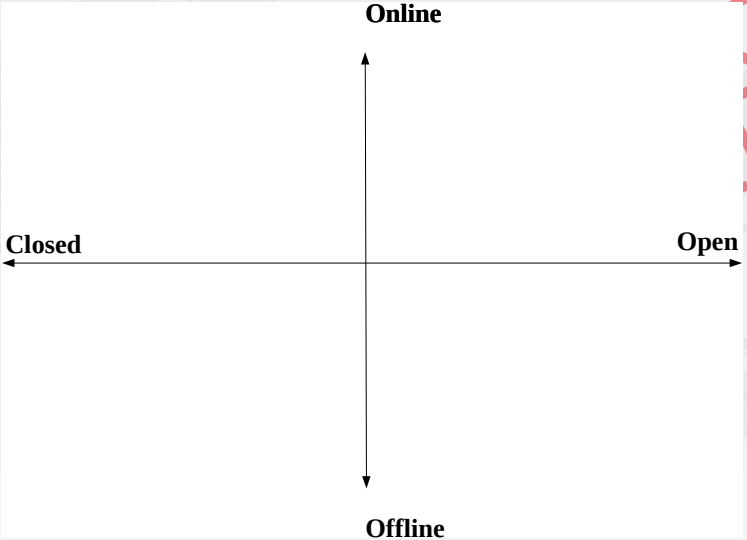


It's the real world!

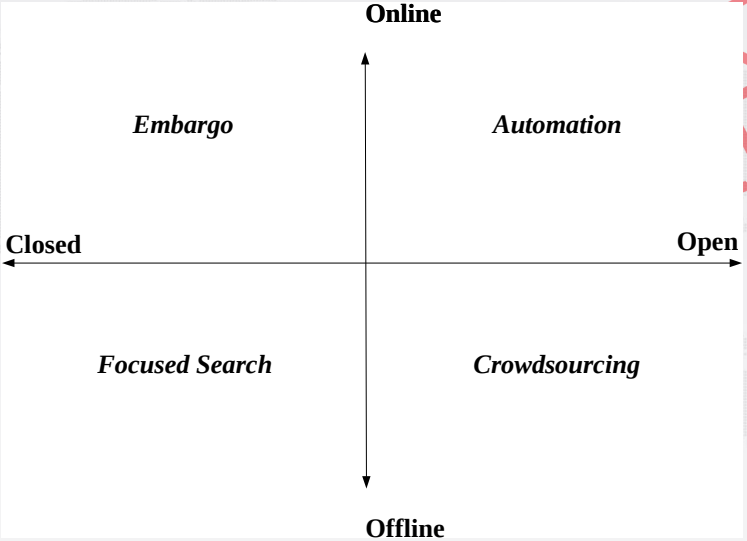
reconcile metadata from different origins, handle conflicts, synthesise missing information, classify (automatically) the projects, etc.

Outline

All the source code



All the source code, strategies



Online, open source code: automation overview