Identifying 3.5 billions source code files
Intrinsic identifiers, the Software Heritage experience

Roberto Di Cosmo (Software Heritage, INRIA)

roberto@dicosmo.org

September 19th, 2017
Our mission
Collect, preserve and share the source code of all the software that is publicly available

Past, present and future
Preserving the past, enhancing the present, preparing the future
Our principles

Open approach
open source, transparency

In for the long haul
non profit, replication

Software Heritage

Cultural Heritage | Industry | Research | Education
Our principles

Open approach
open source, transparency

In for the long haul
non profit, replication

Cultural Heritage
Industry
Research
Education

Software Heritage

Roberto Di Cosmo
Archive coverage

Our sources

- GitHub — full, up-to-date mirror
- Debian — daily snapshots of all suites since 2005–2015
- GNU — all releases as of August 2015
- Gitorious, Google Code — almost done (Archive Team & Google)
- Bitbucket — WIP

Some numbers as a graph: 7 B nodes + 60 B edges

The richest source code archive already, . . . and growing daily!
Archive coverage

Our sources

- GitHub — full, up-to-date mirror
- Debian — daily snapshots of all suites since 2005–2015
- GNU — all releases as of August 2015
- Gitorious, Google Code — almost done (Archive Team & Google)
- Bitbucket — WIP

Some numbers

as a graph: 7 B nodes + 60 B edges

The richest source code archive already, … and growing daily!
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
only one name for each object, 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!
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
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
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, …
### 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
### 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)
**Back to basics: DIOs vs. IDOs**

<table>
<thead>
<tr>
<th>DIO (digital identifier of an object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- digital identifiers for traditional (non digital) objects</td>
</tr>
<tr>
<td>- epistemic complications (manifestations, versions, locations, etc.)</td>
</tr>
<tr>
<td>- significant governance issues, …</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDO (identifier of a digital object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- (digital) identifier for digital objects</td>
</tr>
<tr>
<td>- much simpler to build/handle</td>
</tr>
<tr>
<td>- can (and must) be intrinsic</td>
</tr>
</tbody>
</table>

**Separation of concerns**

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

---

Come in, we’re open

http://www.softwareheritage.org  
(position paper at iPres 2017)
A look at the internals

A giant (extended) Merkle DAG
A look at the internals

A giant (extended) Merkle DAG

Contents

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org>
Everyone is permitted to copy and distribute verbatim copies of
this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

The licenses for most software and other practical works are designed
to take away your freedom to share and change the works. By contrast,
the GNU General Public License is intended to guarantee your freedom
to share and change all versions of a program--to make it truly free
software--within this in a sense of being free for everyone to use,
study, change and propagate the freedoms we believe it deserves.

The General Public License subsumes most of the law that governs
software licenses today.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
demand it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender
these rights. These restrictions translate to certain responsibilities
for you if you distribute copies of the program or if you modify it.

In order to protect these rights, we need to prevent corporations
throughout the world from making a de facto property right of
exclusive proprietary control over the software.

We use the term "version" to refer to a single, identifiable
distribution of the program, be it individual source code or the full
binary distribution.

Each version is given a version number as a form of dr"
A look at the internals

A giant (extended) Merkle DAG

[Diagram of a Merkle DAG with nodes and edges representing snapshots, releases, revisions, directories, and contents.]

Roberto Di Cosmo
Identifying 3.5 billions source code files  September 19th, 2017
A look at the internals

A giant (extended) Merkle DAG

Directories

- .gitignore
- AUTHORS
- LICENSE
- MANIFEST.in
- Makefile
- Makefile.local
- README.db_testing
- README.dev
- bin
- debian
- docs
- requirements.txt
- setup.py
- sql
- swh
- utils

100644 blob 5baade4c44766042186f858c0fd63d587ebf89 .gitignore
100644 blob 20d343af6f52cfc3f6b0c2f7bd0648fbd255e77f AUTHORS
100644 blob 94a9ed02d3859793e18152e559a168bcbb5e2 LICENSE
100644 blob db2665a43a43f8a79a84e0867751dfb09577bb MANIFEST.in
100644 blob 524175c2bad0b3b5b975f79284c2f5a6d5eaf2eb4 Makefile
100644 blob 5c7e3a5bbdd0836862b7793f440492ed9678bb3 Makefile.local
100644 blob 8617988629cd24e680404f09aa749b085b3e07b README.db_testing
100644 blob 76b29f94c815e8869c41d3878d7ce08ec514e README.dev
100000 tree e1e10ecf948af0b93a0b372afa0f91e92618a bin
100000 tree 83e56d0beaf7793c77a45a345c80fc8af583013 debian
100000 tree a34c9c4ba23f0cdec67f9816348d27955577af5 docs
100644 blob f2a6d32c6135aa7287bbd76167b01df2ae4f1539 requirements.txt
100755 blob eee1473c36caf11bb2d820da8d0c26cb5b68180bc setup.py
100000 tree 224bb4c1f4c67fca1d160bfff2d60694e7e1abf3 sql
100000 tree 8631c9cd77bbe9931638107a5baf51f4c6300be swh
100000 tree 8f905b56ba8ed692f1209b2773b474c6c1d66c1 utils

id: 515f00d44e92c65322aaa9bf3fa097c00ddbb9c7d
A look at the internals

A giant (extended) Merkle DAG

Roberto Di Cosmo

Identifying 3.5 billions source code files
September 19th, 2017
### Revisions

<table>
<thead>
<tr>
<th>Details</th>
<th>Changes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA: 953634dca6ba5dc37e3ee426ba091092c267f9f6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Author: Nicolas Dandrimont <a href="mailto:nicolas@dandrimont.eu">nicolas@dandrimont.eu</a> (Thu Sep 1 14:26:13 2016)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Committer: Nicolas Dandrimont <a href="mailto:nicolas@dandrimont.eu">nicolas@dandrimont.eu</a> (Thu Sep 1 14:26:13 2016)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subject: provenance.tasks: add the revision -&gt; origin cache task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parent: fc3ab59ca1df424d860f2c29ab07fe4dc35d10: test...storage, properly pipeline origin and cont...provenance.tasks: add the revision -&gt; origin cache task</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```bash
tree 515f00d44e92c65322aa9bf3fa097c00dd9c7d
parent fc3ab59ca1df424d860f2c29ab07fe4dc35d10
author Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200
committer Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200

provenance.tasks: add the revision -> origin cache task
```

id: 963634dca6ba5dc37e3ee426ba091092c267f9f6
A look at the internals

A giant (extended) Merkle DAG
A giant (extended) Merkle DAG

Roberto Di Cosmo

Identifying 3.5 billion source code files
September 19th, 2017
A look at the internals

A giant (extended) Merkle DAG