Linked Data descriptions of Debian source packages using ADMS.SW

Olivier Berger

Telecom SudParis, Évry, France and Debian project
olivier.berger@telecom-sudparis.eu or obergix@debian.org

Abstract. The Debian Package Tracking System is a Web dashboard for Debian contributors and advanced users. This central tool publishes the status of subsequent releases of source packages in the Debian distribution.
It has been improved to generate RDF meta-data documenting the source packages, their releases and links to other packaging artifacts, using the ADMS.SW 1.0 model. This constitutes an authoritative source of machine-readable Debian "facts" and proposes a reference URI naming scheme for Linked Data resources about Debian packages.
This should enable the interlinking of these Debian package descriptions with other ADMS.SW or DOAP descriptions of FLOSS projects available on the Semantic Web also using Linked Data principles. This will be particularly interesting for traceability with upstream projects whose releases are packaged in Debian, derivative distributions reusing Debian source packages, or with other FLOSS distributions.

Keywords: ADMS.SW, Debian, Linked Data, package, Semantic Web, standard, interoperability, Open Source, Free Software, RDF, DOAP, PTS, FLOSS

1 Introduction

Asset Description Metadata Schema for Software (ADMS.SW) is a novel ontology developped for describing software packages, releases and projects, which can be applied to describe packages in a FLOSS[1] distribution, using Semantic Web techniques. We believe it is a foundational component that will allow to conduct future Quality Assurance or other large scale activities across the Linked Open Data cloud [1].

FLOSS software ecosystems are composed of many different actors collaborating around single programs, from original upstream authors to downstream packagers in distributions like Debian[2]. Descriptions of FLOSS development artifacts made with standardized and semantic formats like ADMS.SW can help trace some of the process which generally happen in various venues across the ecosystem.

[1] Free Libre and Open Source Software
[2] http://debian.org/

1.1 The Need for Linked Data Descriptions of FLOSS

Constructing models of interactions happening along the FLOSS production lines can be interesting, both for researchers and practitioners.

Research in empirical software engineering can for instance involve studies conducted by modeling properties and relations between FLOSS production artifacts and actors. Collection and correlation of facts can then be made using Semantic Web techniques [2] by third party observers distinct from the FLOSS practitioners. The extensible RDF model can be very convenient to interlink resources representing actors or artifacts belonging to different projects. It will allow to integrate in the same triple store resources with variable structures, still relying on a set of common properties and a URI nomenclature that reflects the origin of the resources.

But for FLOSS developers alike, these semantic Web Techniques can offer potential interesting applications, in particular to create new global services that need to interconnect different heterogenous project tools [3]. As an illustration, trying to correlate similar reports filed in different Linux distributions, can be helpful to offer better support responses, allowing navigation between reports which may have previously be related to each-other. Such a new "global bug tracking system" will require to interface to lots of different bugtracker APIs. Whereas standards like *Open Services for Lifecycle Collaboration* (OSLC) [4], which rely on extensible semantic formats based on RDF and REST[3] APIs, can help solve some concrete interoperability issues, they only address parts of the problem (and their deployment is not yet spectacular among FLOSS project). Even once semantically compatible data has been collected, it must be integrated in a coherent database. Nomenclature, freshness and accuracy issues will then still represent interesting challenges.

Making sure every FLOSS project is able to publish on their Web sites or development forges, even minimal, but authoritative Linked Data descriptions, either as *Descriptions Of A Project* (DOAP) [5] or ADMS.SW of their project or software artifacts, is a foundational requirement for applications described above.

1.2 Authoritative Linked Data Descriptors Produced by FLOSS Projects

We postulate that there are higher chances that meta-data is more accurate and up-to-date when it is produced closest to the very heart of the FLOSS projects, than obtained after a series of collection and conversion activities conducted by third parties. Thus, with the Linked Data principles in mind[4], significant artifacts produced by FLOSS projects ought to be complemented with descriptions available at the very same Web domains, as a minimal set of authoritative RDF resources. URIs naming these resources can then be rooted at the project's domain name, and serve as uniquely identifying its artifacts.

[3] REpresentational State Transfer

[4] `http://www.w3.org/DesignIssues/LinkedData.html`

As an illustration, resources describing projects from the Apache foundation would be downloaded from RDF descriptors at `http://projects.apache.org/` and be identified by URIs named like `<http://PROJNAME.apache.org/>`. Thus, the Debian source packaging of Apache's *geronimo* package referenced by the `<http://packages.qa.debian.org/geronimo>` RDF resource, described in a document available at `http://packages.qa.debian.org/geronimo`, would ideally refer to its upstream project as the `<http://geronimo.apache.org/>` Linked Data resource.

1.3 Goal and Structure of this Paper

This paper will introduce such a Linked Data interface, which was deployed on the Debian Package Tracking System, that will produce, using ADMS.SW, authoritative meta-data descriptions for the core artifacts produced by the Debian project: source packages. Due to Debian's respected position in the FLOSS ecosystem, such a deployment already covers a great percentage of all FLOSS programs, and can thus be inspirational for many FLOSS projects.

In section 2, we introduce the ADMS.SW specification. Then a brief introduction to the structure of Debian source packages is provided in section 3. Section 4 will document the choices adopted for generating Linked Data representations of Debian source packages and related FLOSS artifacts in the Debian PTS. Section 5 will present a quick review of similar and complementary initiatives, while section 6 will illustrate how trivial project matching can be made with collected Linked Data descriptions.

2 The ADMS.SW Specification

The *Asset Description Metadata Schema for Software* (ADMS.SW) specification[5] is described as : "[...] *a metadata vocabulary to describe software making it possible to more easily explore, find, and link software on the Web.*"

It is an outcome of the ISA programme (*Interoperability Solutions for European Public Administrations*) of the European Commission[6], elaborated by a working group of software catalogues and forges experts[7]. Although it is not specifically covering FLOSS software only, ADMS.SW has nevertheless been geared mainly at addressing meta-data of FLOSS projects hosted in public development forges that could be reused by Public Administrations.

ADMS.SW 1.0 reuses existing specifications and standards, such as DOAP [5], SPDX™ [6], ISO 19770-2 [7], ADMS [8], and the "Sourceforge Trove software map"[8]. As illustrated in Figure 1, it provides three main entities : *Software*

[5] `https://joinup.ec.europa.eu/asset/adms_foss/release/release100`
[6] `http://ec.europa.eu/isa/`
[7] the author was an active member of the working group.
[8] `http://sourceforge.net/apps/trac/sourceforge/wiki/Software Map and Trove`

Project, *Software Release*, and *Software Package* to model meta-data about software programs, their versions, and the distribution files of these.

As DOAP is already widely used, ADMS.SW reuses much of its properties. But it also extends it to propose a model for various elements related to *Software Repositories* descriptions, based on the RADion common model of ADMS, which describes generic semantic assets, in order to facilitate the maintenance of data managed by software catalogues (provenance, timestamping, etc.). ADMS.SW is also interoperable with the SPDX specification, whose main object, to date, is the description of copyright and license conditions applying to particular software packages or source files.

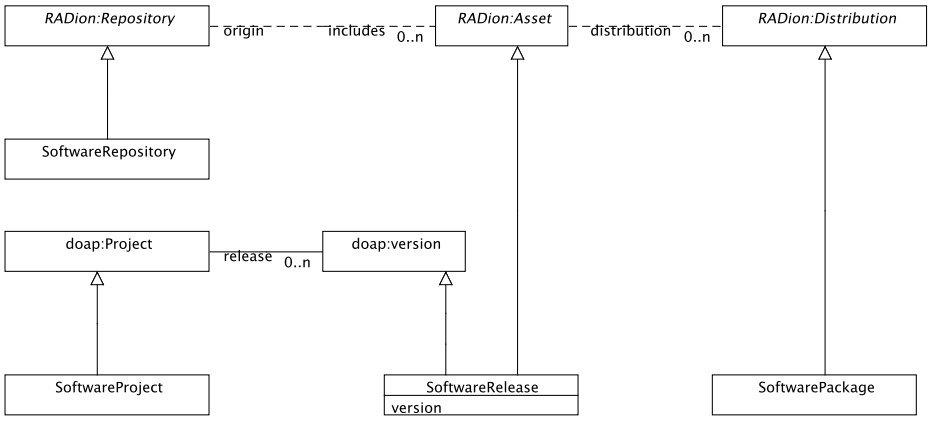

Fig. 1. Simplified UML diagram of the main ADMS.SW entities

ADMS.SW specifications are published with a complementary OWL ontology, referenced as `http://purl.org/adms/sw/`, to allow the publishing of such meta-data as RDF.

3 Debian Source Packages

The Debian project creates a Free Software distribution, which contains thousands of FLOSS *binary* packages ready to be installed on various computer architectures. Several versions of the Debian distribution are maintained in parallel, in three main *suites* : 'stable', 'testing' and 'unstable'.[9]

Debian has been studied by many authors, as it represents a good proxy for the entire FLOSS ecosystem, due to the high number of packages it contains, and since its development and Quality Assurance (QA) infrastructure is generally open or easily accessible to researchers in empirical software engineering (see for instance [9]).

[9] there are actually additional variants which don't matter too much for the current discussion

3.1 Structure of Debian Source Packages

Each binary package is actually built from a particular Debian *source* package. Source packages contain "Makefiles" for package generation, control files containing different meta-data like versions or package dependency descriptions, and other scripts necessary for installation, configuration, upgrade or removal of the binary packages [10]. In addition, it is quite common to include patches applying to the source code of the packaged program, to adjust it to Debian specificities or to include security fixes backported from later upstream releases.

Each revision of a Debian source package is then generally composed of two file archives : one for the source code of the upstream version of the packaged program (ending in `.orig.tar.gz`), complemented by another one for these Debian specific files (ending in `.debian.tar.gz`)[10]. Only the latter Debian specific files archive, and associate meta-data descriptors change between subsequent revisions of Debian source packages of the same version of an upstream program.

3.2 The Debian PTS

For every Debian source package, the *Debian Package Tracking System* (PTS) provides a Web dashboard (see a screenshot[11] in Figure 2) which displays almost all there is to know about the status of that package [11].

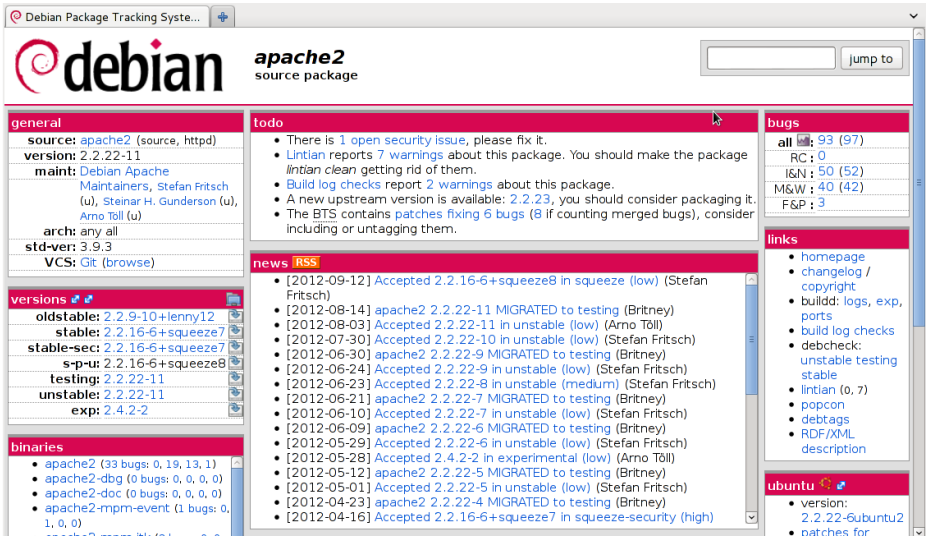

Fig. 2. Apache 2 source package status in the Debian PTS

[10] as an exception to this general rule, some packages, which are called "native packages", don't have a corresponding upstream project outside Debian and only have Debian specific files.

[11] taken from `http://packages.qa.debian.org/a/apache2.html`

However, such HTML pages are not really exploitable by machines in a direct form, should anyone need to interface the Debian QA system with other services. One such need seems quite obvious for derivative distributions constructed from Debian, like Ubuntu. Therefore, the PTS provides a custom SOAP interface[12], but the lack of standard representation of data retrieved from this API may require another ad-hoc converter to be added to any application wishing to interface with it.

As an alternative, we have started implementing a Linked Data [1] interface for the Debian PTS, using the ADMS.SW ontology to represent Debian source package facts.

4 Linked Data Representation of Debian Source Packages

We have improved the Debian PTS to add the generation of RDF descriptions for all Debian source packages, based on the same reference version as the main one documented by the PTS (i.e. usually the "latest" version in the 'unstable' suite).

Every Debian source package, which used to have an HTML page accessible at URLs like `http://packages.qa.debian.org/SRC-PKG-NAME`, now has a corresponding RDF/XML document available at the same URL, provided that the `application/rdf+xml` content-type is requested (the HTTP client being redirected to the proper HTML or RDF/XML document).

Thus, each package in Debian can be identified on the Semantic Web with a unique URI like `http://packages.qa.debian.org/apache2`, which is dereferenceable as an RDF document.

The example in Listing 1.1 is an excerpt of such an RDF description of a particular revision of the source package for `apache2` (converted to Turtle format for readability).

4.1 Modelling Debian Source Packages with ADMS.SW

This section presents the modelling choices adopted so that every Debian source package can be modeled as interlinked RDF resources. In the following, the `http://packages.qa.debian.org/...` URIs will be abbreviated as `<p.q.d.o/...>` for the sake of readability. The version numbers reflected in the resource URIs or file names below respect the Debian package versions numbering convention[13].

Figure 3 represents the main resources produced by the PTS for a particular release of the `apache2` Debian source package, as found in `http://packages.qa.debian.org/a/apache2.rdf` (in grey, the "upstream"-related resources).

[12] `http://wiki.debian.org/qa.debian.org/pts/SoapInterface`

[13] as a short rule, the Debian package revision M of version N of an upstream program P is identified in file names as `P_N-M`.

```
<#apache2_2.2.22-11>
    a admssw:SoftwareRelease ;
    rdfs:label "apache2 2.2.22-11" ;
    dcterms:description "Debian apache2 source package version
        2.2.22-11" ;
    doap:revision "2.2.22-11" ;
    admssw:project <> ;
    admssw:includedAsset <#debiansrc_2.2.22-11>, <#
        upstreamsrc_2.2.22> ;
    admssw:package <#apache2_2.2.22-11.dsc> ;
    dcterms:relation <https://launchpad.net/ubuntu/+source/
        apache2/2.2.22-6ubuntu2> .
```

Listing 1.1. RDF description of revision 11 of the source package for apache2 version 2.2.22

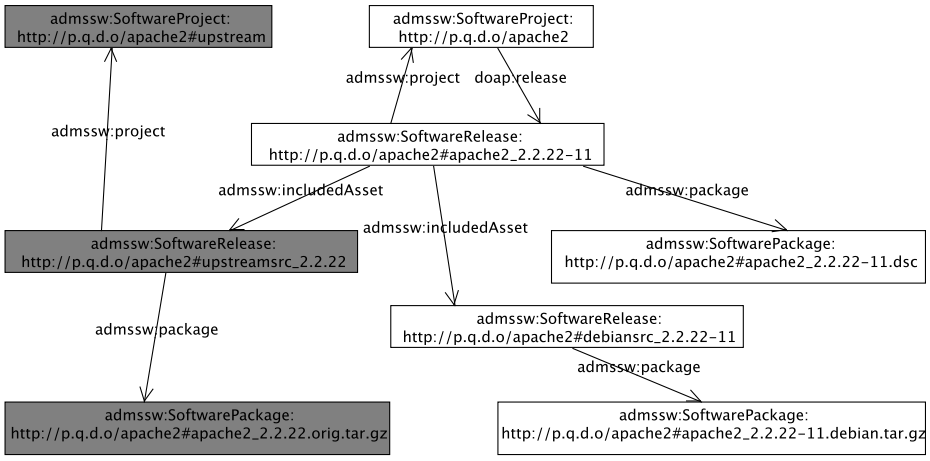

Fig. 3. Resources produced for release 2.2.22-11 of the Debian `apache2` source package

Every source package has a corresponding *source packaging project* `SoftwareProject` resource, named `<p.q.d.o/SRC-PKG-NAME>`. The different resource URIs which will be expressed below will be fragments to this base URI. *Revisions* of this source package have corresponding `SoftwareRelease` resources, named as `<#SRC-PKG-NAME_DEB-PKG-VERS>`. Only one of these (the "latest" one known by the PTS) is fully described as containing (`includedAsset`) two sub `SoftwareReleases` :

– one `SoftwareRelease` for the *upstream program's version*, named `<#upstreamsrc_UPSTR-VERS>`. It comes with additional resources for all *archive files* of the upstream sources as `SoftwarePackages` (typically named like `<#SRC-PKG-NAME_UPSTR-VERS.orig.tar.gz>`);

- one for the set of *Debian packaging files*, as `<#debiansrc_DEB-PKG-VERS>`, with resources for all files comprising the Debian *package source archive* (typically named like `<#SRC-PKG-NAME_DEB-PKG-VERS.debian.tar.gz>`).
- An additional `SoftwarePackage` resource is generated for its `SRC-PKG-NAME_UPSTR-VERS.dsc` file at a URI like `<#SRC-PKG-NAME_DEB-PKG-VERS.dsc>`.

Also produced is one `SoftwareProject` resource for the *upstream project*, named `<#upstream>` whith a `doap:homepage`, if it's known by the PTS.

Additional complementary resources are produced, and all resources have RDF properties (as mandated in ADMS.SW, mainly reused from DOAP or Dublin Core), all of which it is useless to describe here in detail.

4.2 Deployment

The author has deployed the XSLT stylesheets generating these RDF documents on the PTS service of the Debian project[14]. The RDF descriptions of Debian source packages are thus refreshed every time new revisions will appear in the Debian archive.

A full RDF dump of all the meta-data is also available to Debian members[15]. It contains around 1.5 million triples at the time of writing.

5 Complementary Efforts

In this section, we describe a few complementary initiatives which describe software packages with RDF vocabularies, using DOAP or ADMS.SW and which could be interesting for interoperability with the Debian PTS.

5.1 DOAP Packages in FLOSS Archives

A number of projects maintain public DOAP descriptions of their programs, or other RDF descriptions of meta-data about the releases they produce. They may be interested in complementing descriptions with ADMS.SW, or could offer sources of descriptions that could be interlinked with the ones produced by the Debian PTS.

A quick survey conducted by the author showed the following sources[16] :

- Gnome project
- Apache project
- PyPI (Python Package Index) directory
- CPAN (Comprehensive Perl Archive Network) directory

Listing 1.2 shows an excerpt of the DOAP description of the Apache Geronimo project as published by the project[17], and converted to Turtle for readability.

[14] see : `http://packages.qa.debian.org/common/RDF.html`

[15] on `packages.qa.debian.org:/srv/packages.qa.debian.org/www/web/full-dump.rdf`

[16] these are maintained in `https://github.com/edumbill/doap/wiki/Sites`

[17] downloaded from `http://svn.apache.org/repos/asf/geronimo/site/trunk/doap_Geronimo.rdf`

```
@prefix doap: <http://usefulinc.com/ns/doap#> .

<http://geronimo.apache.org/>
    a doap:Project .
    doap:name "Apache Geronimo"@en ;
    doap:shortdesc "Java EE Application Server"@en ;
    doap:description "Apache Geronimo is an open source server
        runtime [...]."@en ;
    doap:homepage <http://geronimo.apache.org> ;
    doap:programming-language "Java"@en ;
    doap:bug-database <http://issues.apache.org/jira/browse/
        GERONIMO> ;
    doap:download-page <http://geronimo.apache.org/downloads.
        html> ;
    doap:license <http://usefulinc.com/doap/licenses/asl20> ;
    doap:mailing-list <http://geronimo.apache.org/mailing-
        lists.html> ;
```

Listing 1.2. RDF description the Apache geronimo project

A quick review of samples from these sources show a lack of consensus on the use of certain meta-data, and that URIs adopted to reference the same projects or programs tend to vary, even for `doap:homepage` URLs (a great portion of these documents are manually crafted, and projects may have various pages that can be considered their *homepage*, in particular when the project is not hosted on its own top level domain).

5.2 Projects Hosted on FusionForge Forges

An ADMS.SW plugin for the FusionForge 5.2 software development forge has also been created by the author, in order to allow the description of projects hosted on FusionForge based development forges. It currently lacks descriptions of the files released by the projects, but already provides the means to describe projects and participants. It may be complemented by another FusionForge plugin providing FOAF profiles [12] for project participants, which can enrich the Linked Data representations.

Still, being developped for the 5.2 release of FusionForge, it will take a certain time until it is deployed on public forges hosting FLOSS projects[18], allowing interlinking of these projects with other references in software catalogues.

5.3 The Joinup Portal

The *Joinup* portal of the ISA programme aims at integrating in a single portal, FLOSS available from different Public Administration forges, by harvesting de-

[18] like Debian's own Alioth forge operated by FusionForge at `http://alioth.debian.org/`

scriptions of projects directly in their development project spaces, as ADMS.SW descriptions[19].

Whereas the current version of Joinup doesn't rely on Semantic Web techniques for collection of the projects descriptions, it is expected to be improved to evolve towards ADMS.SW consuming in the future. FLOSS Project descriptions would then complement other Semantic Assets (standards, documentation) catalogued and made available on the reference portal at Joinup as semantic assets using the ADMS vocabulary.

5.4 Interlinked developer profiles

Project descriptions aren't the only resources that can be interlinked across the FLOSS ecosystem. Iqbal shows in [13] how developer profiles can also be converted to RDF and interlinked to create a more comprehensive view of the developer communities around a project, for instance. This approach usually involves mining repositories or social sites through custom interfaces (via SOAP for instance), and later converting to RDF. But we believe there would be a great benefit in avoiding such potentially error-prone conversions if development platforms would natively produce DOAP (or FOAF) descriptions "out of the box", as explained above.

6 Applications

As with every Linked Open Data initiatives, the use of standard representations and their availability on the Semantic Web can lead to lots of different uses.

An obvious case of using such ADMS.SW description of Debian source package is the matching of Debian packages with other packages/projects described in their respective projects directories, allowing more interlinking of resources.

6.1 Matching Projects / Software Across Repositories

At the moment, the `doap:homepage` of the "upstream" `SoftwareProject` resources generated by the Debian PTS can be an obvious matching key, provided that one has a database of upstream project descriptions (as DOAP[5]).

As an illustration, by loading DOAP descriptions of projects of the Apache foundation[20], together with a dump of the Debian source package descriptions in a single triple store, one can find matches for common project homepages. The example SPARQL query in Listing 1.3 shows how to query for such matches between Debian and Apache.

Such a query currently returns 62 matched source packages and Apache upstream projects (see an excerpt in table 1, where URLs have been compacted for brievity).

[19] more details at `https://joinup.ec.europa.eu/software/federated_forge`

[20] collected from `projects.apache.org` (see `http://projects.apache.org/docs/index.html`)

```
PREFIX doap: <http://usefulinc.com/ns/doap#>

SELECT * WHERE
{
  GRAPH <http://packages.qa.debian.org/>
  {
   ?dp doap:homepage ?h
  }
  GRAPH <http://projects.apache.org/>
  {
   ?ap doap:homepage ?h
  }
}
```

Listing 1.3. SPARQL query matching Apache and Debian projects by common homepages

But the reliability of this matching method isn't very good in practice. There may be many more Apache foundation projects packaged in Debian, but the maintainers may have forgotten to add a homepage link in the package descriptors. Or the URLs mentioned may not be matching, as project homepage naming conventions can vary (and evolve in time).

An alternate matching method could be based on project names, but that isn't always feasable either, due to homonimy for instance. One will refer to [14] for an analysis of this problem.

The *distromatch* project[21], started last year, intends to try and help solve these project/packages matching issues, although it is unfortunately not maintained at the moment.

In any case, this first quick attempt allows us to plan further developments based on such meta-data, which will be tested on real life cases, for instance in constructing RDF harvesters and meta-data aggregators, and eventually merging with initiatives like distromatch.

6.2 Large Scale Perspective

The RDF-ization of the Debian PTS has just started. Next steps will include modelling of relations between source and binary packages. These will probably require extending ADMS.SW or integrating complementary ontologies.

When deployments of ADMS.SW have been made on software forges (like FusionForge servers), software catalogues (like Joinup) or in other FLOSS distributions, it will become one of the tools allowing traceability at large scale of FLOSS releases and associated artefacts, by interlinking their Linked Data resources.

[21] `http://www.enricozini.org/2011/debian/distromatch/`

Table 1. Matching upstream project homepages with Debian source packages'

dp	h	ap
ivy	ant.a.o/ivy/	ant.a.o/ivy/
apr	apr.a.o/	apr.a.o/
apr-util	apr.a.o/	apr.a.o/
libcommons-cli-java	commons.a.o/cli/	commons.a.o/cli/
libcommons-codec-java	commons.a.o/codec/	commons.a.o/codec/
libcommons-collections3-java	commons.a.o/collections/	commons.a.o/collections/
libcommons-collections-java	commons.a.o/collections/	commons.a.o/collections/
commons-daemon	commons.a.o/daemon/	commons.a.o/daemon/
libcommons-discovery-java	commons.a.o/discovery/	commons.a.o/discovery/
libcommons-el-java	commons.a.o/el/	commons.a.o/el/
libcommons-fileupload-java	commons.a.o/fileupload/	commons.a.o/fileupload/
commons-io	commons.a.o/io/	commons.a.o/io/
commons-jci	commons.a.o/jci/	commons.a.o/jci/
libcommons-launcher-java	commons.a.o/launcher/	commons.a.o/launcher/
...	...	...

Some interlinking of security advisories, patches, or bug reports (for instance combined with the OSLC-CM standard[22]) should then be easier, and diminish manual intervention needs, for the benefits of all actors along the FLOSS production chains.

7 Conclusion

We have presented a first significant deployment of an ADMS.SW 1.0 implementation, which illustrates the potential for interlinking large sets of FLOSS project descriptions on the Semantic Web. ADMS.SW allows us to describe relations between projects, programs and their releases so that such entities become part of the Linked Open Data "cloud". The way we used it or Debian can be inspirational for other FLOSS distributions, either independant, or derived from Debian. By integrating such meta-data generation in the heart of the technical infrastructure of Debian, we hope to establish such an authoritative reference for Debian source packages identification on the Semantic Web.

In [3], we envisioned some novel uses of Linked Data representations of FLOSS development artefacts, both for software engineers and researchers observing their efforts. But to achieve the full potential of that approach, the Linked Data representations must be semantically interoperable, authoritative, accurate, and using standard naming schemes for the same resources. We have achieved a first concrete step in this direction, through the current results for the Debian PTS.

[22] `http://open-services.net/wiki/change-management/`

We believe the current early result can be a driving force for more deployments around ADMS.SW as a standardization core, going in the above direction. However we think that only when novel inter-project "killer" applications making use of such Linked Data will have been developed, will it be possible to convince FLOSS projects that adoption of Linked Data standards descriptions can really be for their own benefit.

It is likely that even when lots of Linked Data descriptions of FLOSS artifacts are made available by major FLOSS projects, achieving effective interoperability will require many implementation efforts, far beyond a single actor's reach. More standardisation will be needed, and services will have to be provided to establish trusted reference catalogues of Semantic project descriptors (in the direction set by *Joinup* of the *distromatch* project for instance). Such actors will provide FLOSS "semantic hubs", or project matching "brokers" which will maintain reference interlinking relations for the concurrent Semantic descriptions which were produced in the many venues of the FLOSS ecosystem.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) **5**(3) (3 2009) 1–22
2. Howison, J.: Cross-repository data linking with RDF and OWL: Towards common ontologies for representing FLOSS data. In: WoPDaSD (Workshop on Public Data at International Conference on Open Source Software). (2008)
3. Berger, O., Vlasceanu, I.V., Bac, C., Dang, Q.V., Lauriere, S.: Weaving a semantic web across OSS repositories: Unleashing a new potential for academia and practice. International Journal of Open Source Software and Processes (IJOSSP) **2**(2) (2010) 29–40
4. Berger, O., Labbene, S., Dhar, M., Bac, C.: Introducing OSLC, an open standard for interoperability of open source development tools. In: ICSSEA, Paris, France (2011) ISSN–0295–6322
5. Dumbill, E.: Decentralizing software project registries with DOAP. In: Intelligent Search on XML Data - XML. (2004)
6. *unspecified authors*: Software Package Data eXchange specification (2011)
7. *unspecified authors*: ISO/IEC 19770-2: Software identification tag standard
8. *unspecified authors*: Asset Description Metadata Schema specification 1.00 (2012)
9. JJ Amor, G Robles, J.G.B., Rivas, F.: Measuring lenny: the size of debian 5.0. (2009)
10. Ian Jackson, C.S.e.a.: Debian policy manual version 3.9.4.0, 2012-09-19 - `http://www.debian.org/doc/debian-policy/`.
11. Michlmayr, M.: Managing debian. AUUGN, The journal of AUUG Inc. **25**(3) (9 2004)
12. Graves, M., Constabaris, A., Brickley, D.: FOAF: Connecting People on the Semantic Web. Cataloging & classification quarterly **43**(3) (April 2007) 191–202
13. Iqbal, A., Hausenblas, M.: Integrating developer-related information across open source repositories. In: Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on. (aug. 2012) 69 –76
14. Squire, M.: Integrating projects from multiple open source code forges. IJOSSP **1**(1) (2009) 46–57