Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings

Volume 17 *Boston, USA*

Article 29

2017

Towards OSGeo Best Practices for Scientific Software Citation: Integration Options for Persistent Identifiers fn OSGeo Project Repositories

Peter Löwe
German Institute for Economic Research Mohrenstraße 58, 10117 Berlin, Germany

Markus Neteler
Mundialis GmbH & Co. KG Kölnstraße 99, 53111 Bonn, Germany

Jan Goebel
German Institute for Economic Research Mohrenstraße 58, 10117 Berlin, Germany

Marco Tullney
Technische Informationsbibliothek Welfengarten 1B, 30167 Hannover, Germany

Recommended Citation
Löwe, Peter; Neteler, Markus; Goebel, Jan; and Tullney, Marco (2017) "Towards OSGeo Best Practices for Scientific Software Citation: Integration Options for Persistent Identifiers fn OSGeo Project Repositories," *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*: Vol. 17 , Article 29.
DOI: https://doi.org/10.7275/R5542KSK
Available at: https://scholarworks.umass.edu/foss4g/vol17/iss1/29

Towards OSGeo Best Practices for Scientific Software Citation: Integration Options for Persistent Identifiers fn OSGeo Project Repositories

Peter Lowe[a,*], Markus Neteler[b], Jan Goebel[c], Marco Tullney[d]

[a] *German Institute for Economic Research Mohrenstraße 58, 10117 Berlin, Germany*

[b] *Mundialis GmbH Co. KG Kolnstraße 99, 53111 Bonn, Germany*

[c] *German Institute for Economic Research Mohrenstraße 58, 10117 Berlin, Germany*

[d] *Technische Informationsbibliothek Welfengarten 1B, 30167 Hannover, Germany*

Abstract: As a contribution to the currently ongoing larger effort to establish Open Science as best practices in academia, this article focuses on the Open Source and Open Access tiers of the Open Science triad and community software projects. The current situation of research software development and the need to recognize it as a significant contribution to science is introduced in relation to Open Science. The adoption of the Open Science paradigms occurs at different speeds and on different levels within the various fields of science and crosscutting software communities. This is paralleled by the emerging of an underlying futuresafe technical infrastructure based on open standards to enable proper recognition for published articles, data, and software. Currently the number of journal publications about research software remains low in comparison to the amount of research code published on various software repositories in the WWW. Because common standards for the citation of software projects (containers) and versions of software are lacking, the FORCE11 group and the CodeMeta project are recommending to establish Persistent Identifiers (PIDs), together with suitable metadata sets to reliably cite research software. This approach is compared to the best practices implemented by the OSGeo Foundation for geospatial community software projects. For GRASS GIS, a OSGeo project and one of the oldest geospatial open source community projects, the external requirements for Digital Object Identifier (DOI)-based software citation are compared with the projects software documentation standards. Based on this status assessment, application scenarios are derived on how OSGeo projects can approach DOI-based software citation, both as a standalone option and also as a means to foster open access journal publications as part of reproducible Open Science.

[*] Corresponding author
Email address: `ploewe@diw.de` (Peter Lowe)

1. Research Code, The Workhorse of Science

Software has become the cross-disciplinary workhorse (Brett et al. 2017). Software developed for research is usually created by scientists themselves, instead of through hired professional software developers. According to a survey among 24 universities presented in (BreBrett et al. 2017), scientific software was used by 92% of the researchers, 67% stating that this is fundamental to their research, and 50% developing their own software. This is a potential bottleneck or opportunity for improvement for science, as the coding skills of the researchers directly affect the quality of the scientific results. Effectively, research is becoming dependent upon advances in software, which consists of a diverse range of types of research tools, including operating systems, applications, models, algorithms, middleware and code libraries (Katz 2017). Scientists should be provided with sufficient motivation to hone their programming skills, publish their research software as part of the scientific process, and care about its long term provision to science infrastructure. This motivation should be provided in a suitable coin of the realm of academia, namely recognition for impact and reuse by scientific credit, i.e. citation. This must in turn be footed in reliable social and technical infrastructures. Otherwise, scientific code is in most cases only advanced to the readiness level of a demonstrator/pilot (works for me), as the scope of research projects does not include a drive for code longevity, including refactoring or post-project maintenance.

Scientists should know to find all relevant software packages for his or her field of work, either as a starting point for their individual work, or as a long term repository for the legacy of their resulting research code. This is a challenge, as software projects and code repositories (e.g: GitHub, Sourceforge or Bitbucket) on the web follow lifecycles: They appear, rise in popularity and eventually fall out of use when the user base moves on to next generation software, starting the cycle once again. A pragmatic solution would be to put reasonable trust in software projects, which have been used over a long period in time, having stable user- and developer-communities to ensure updates of codebase and features, reliable and open communication channels and governance models. Creating a personal survey of such state of the art software projects would result in a significant effort for the individual scientist, yet such surveys for geospatial tools have only been infrequently published in journals, like Steiniger and Hunter 2013. A viable alternative can be provided by self-organized federations of software projects, such as the OSGeo umbrella organization which act as impartial arbiters and provide comparable assessments of software alternatives.

2. Open Science

William Gibsons' observation (https://en.wikiquote.org/wiki/William_Gibson) that the future has already arrived, yet remains unevenly distributed holds true for science, libraries and software projects: While some Open Science related paradigms, both on the social and technical level, are already common practice in some fields of science, they remain to be implemented in others. On the positive side, other fields of science have just started to embrace approaches like addressing research software for Open Science. It is possible to benefit from best practices previously developed in these fields, without having to go to the same development curve. As software from geospatial software projects being adopted as novel research infrastructure by neighboring science disciplines like sociology and economics, these sciences can immediately profit by applying best practices for scientific recognition for research software, which did already evolve in other domains.

If at some point the recommendations of Open Science have succeeded to permeate through all sciences, researchers should be trained by, and expected to adhere to these best practices:

- Discovery of existing research code as potential building blocks to serve their research questions.
- Make informed decisions which code building blocks fits their needs best regarding programming language and architecture
- Be able to trust in the quality of code created by others (individuals and groups), as best practices such as coding standards are adhered to.
- Enable others to trust in and reuse one's own research output by transparency and suitable open publication formats.
- Receive due recognition in a suitable currency (i.e. scientific citation), driving career advances including improving code development skills.

These social best practices would be complemented by a scientific work environment based on integrated technology, standards and best IT practices. These include sharing text, data, and code as openly as possible and linking these different types of published research output openly and semantically. Open practices like these support the implementation of a sustainable and open infrastructure (Bilder et al. 2015). Benefits of these desiderata would be an acceleration of the scientific process and scientific discovery, making science a sensible investment for the benefit of the global society.

3. Publishing Research Software

3.1. Open Access Journals for Geospatial Research Software

Several open access journals already provide platforms to address research software topics including the Journal of Open Research Software (JORS) (http://openresearchsoftware.metajnl.com/) and the Journal of Open Source Software (JOSS) (http://joss.theoj.org/). In comparison to the actual magnitude of research code being produced, only a fraction is being communicated via these journals. As a result, advances in scientific software are not being properly communicated and therefore remain inaccessible to other scientists. The effort to write an article about scientific software is not different than for any other scientific topic. Yet what is still lacking is adequate (from the authors' perspective) compensation through recognition by citation. For the time being, scientists continue to focus their efforts on the generally accepted publication channels (i.e. traditional research papers in high impact journals) which provide the desired kind of gratification. A noteworthy exception is the R project community, which has established in 2009 its own open access journal The R Journal (https://journal.r-project.org/) as a scientific publication. This is a best practice example for other project communities.

Even if the number of journal publications on research code could be significantly increased, this will not solve an underlying issue: A journal publication presents a static freeze frame image of a software object, which is often highly dynamic. As Rother 2015 puts it, a publication does not tell you whether the authors are still developing their program further, whether they have stopped maintaining it, or whether the developers have switched fields altogether. From the perspective of a potential user/contributor the following criteria should be applied to assess the quality of research code: Availability on a reliable repository enabling easy installation and source code access, documented recent application, responsive code authors, proof of fitness for use, including sample data and documentation. This could be provided via novel open and responsive formats for journal publications like executable papers for example based on Jupyter notebook technology (http://jupyter.org/).

3.2. Persistent Identifiers for Geospatial Research Software

There are currently no established standards or best practices how to cite research code within a journal publication, like this article. Several approaches to software citation exist, including citations of publications, user manuals, websites, by name, etc. There is also the need to make references to a software project as a general container (e.g. the GRASS GIS project), but in parallel also to refer to specific versions of the code (e.g. GRASS 7.2. versus GRASS 4.2.1). Metadata for software containers and software versions must not be necessarily identical. So independent means are needed to distinctly cite and reference them.

To mitigate this situation the FORCE11 Software Citation group (https://www.force11.org/group/software-citation-working-group) developed software citation principles from data citation principles and existing community practices. These principles comprise the following paradigms: Recognition of scientific software as a legitimate and citable product of research. Scholarly credit for scientific software should be given by credit and attribution. Unique and machine actionable global identification must be ensured, and also persistent use of identifiers and metadata. Access both to identifiable versions of the software and its metadata must be provided. According to (Sperber 2017) the situation is complicated by persistence issues (publications have PIDs, software is not persistent), inherent dynamics of distributed software development, and the absence of standardized (bibliographic) metadata for code.

While the FORCE11 are being embraced as a constructive advance, unresolved issues remain, including the referencing of software containers and versions. The FORCE11 principles call for unique and machine-actionable global identification by persistent identifiers (PIDs). From the user perspective, PIDs enable permanent references to web-objects, regardless whether its web-location should change over time, which would break conventional references by URL.

PIDs are widely used to enable citation of journal publications, research data and research software. Several kinds of PIDs exist, depending on the intended usage scenario (e.g. URN in the German library domain, handles as PIDs with a potentially limited lifespan). PIDs usually point to a human readable landing page, which provides relevant metadata and the most recent link to actual content. Apart from the underlying resolving mechanisms which enable PIDs to link to permanent landing pages, any PID is primarily a social promise to the PID users that this service will be permanently offered. Of the various kinds of PIDs, only Digital Object Identifiers (DOI) currently satisfy the FORCE11 criteria for reliable and machine-actionable software citation. DOIs for research data and software are provided by the DataCite organization (http://www.datacite.org).

The definition of the required metadata provided on landing pages for software is addressed by the CodeMeta project (https://github.com/codemeta/codemeta), which has developed a set of terms to describe software by metadata. These concepts are also used by established repositories and citation services like DataCite, figshare, GitHub, and Zenodo. The terms defined by CodeMeta are intended as a means to ensure the exchange of software descriptions among stakeholders including software users, repositories, communities, citation services, and other parties. They comprise of three object classes: containers, versions, and agents. The latter can be individuals or organizations. Each class has a distinct set of metadata fields. The CodeMeta set of terms is currently not fully consistent with the Software Citation Principles described previously.

4. OSGeo

The OSGeo Foundation (http://www.osgeo.org) is a community driven international umbrella organization, which currently hosts 22 software projects and numerous community projects. Each project is managed and organized by a community of volunteers. Project results like code, data and documentation are provided under free and open licenses for data, software and documents.

Each OSGeo project had to undergo an assessment process referred to as incubation. It assures that a basic set of best practices for open source geospatial projects is met. The current incubation criteria evolved from long duration community software projects. Their goal is to ensure transparent and meritocratic project governance and work practices, i.e. a minimal set standards including openness, and copyright and license, processes, documentation (https://wiki.osgeo.org/wiki/Project_Graduation_ChecklistDocumentation). Being observed by the OSGeo board, the incubation process is not static and evolves over time according to the lessons learned during previous incubation processes and community needs. This makes it a strategic point to introduce and anchor new best practices, like citation principles, within OSGeo for all new projects.

Since 2007, OSGeo has been publishing the online OSGeo Journal (http://www.osgeo.org/journal). It is currently only referenced by ISSN. For the last years, the proceedings of the annual FOSS4G conferences have been published in this journal. Due to the current lack of DOIs, only standard textual references to the proceeding articles can be made. In parallel, a growing number OSGeo-related audiovisual content has been published on several platforms, including video portals such as YouTube and Vimeo. This also includes conference recordings from the FOSS4G conference series. Since 2014, the German national library for science and technology (TIB: Technische Informationsbibliothek) is actively collecting conference recordings from several OSGeo conference series in Europe and North America to ensure their preservation as part of the scientific record (Lowe et al. 2015). The conference recordings are long term preserved, including enriched metadata derived from audiovisual content analysis and receive DOIs which are registered with DataCite. The conference videos are provided through the TIB video portal (https://av.tib.eu). DOIs for audiovisual content allow the citation of both the full video and also selected scenes, which can be referenced by appending a media fragment identifier (MFI) containing time codes to a DOI.

OSGeo has embraced this DOI-enabled service by including it as a best practice for conference video recording in its conference handbook (http://wiki.osgeo.org/wiki/FOSS4G_Handbook). For the OSGeo organization, this has led to a comprehensive archive of content related to its projects and their applications. On the individual level, OSGeo members and other conference attendees are provided with a means to scientifically cite their work, which can be accessed not just through the TIB Infrastructure, but also web-wide search mechanisms provided by the DataCite organization.

5. Case Study: GRASS GIS

The Geoinformation System (GIS) Geographic Resources Analysis Support System (GRASS GIS) is one of the oldest continuously active open source projects, being continuously developed for last 34 years, starting in 1983. The project rules have been improved over time, making it one of the most advanced OSGeo projects and having influenced the shaping of current OSGeo best practices including project incubation.

GRASS GIS currently consists of over 350 modules, which can be combined as needed (reflecting the UNIX toolbox paradigm for modular shell programming) (Neteler et al. 2012). Each GRASS GIS module is required to have a standardized manual page (Figure 1). The manual pages are partially auto-generated from the source code during compilation into machine executable binary programs, leading to a standardized manual style. Manual pages provide a human readable description and metadata of the individual module's functionality, flags and parameters. Latest changes are indicated by a timestamp; each page includes links pointing towards both the recent source code but also to the development history log covering all past versions and authors. If a manual page becomes outdated, additional references to the latest version are included to ensure convenient navigation of software versions.

New functionality for GRASS GIS is being added continuously, often as the result of research projects, or as part of campaigns such as code sprints or Google Summer of Code. This is often done by early career scientists. The transfer of such add-on code from the private domain of the scientist into the project software repository, and from there possibly onward into other such OSGeo projects is done in defined steps. From the perspective of the project, the initial step is the code deposition in a repository for add-on modules, which requires the granting of write access following a public request by the coder. The coder agrees to adhere to minimal standards (https://trac.osgeo.org/grass/wiki/Submitting) for code and documentation. Add-ons are not part of the canonical GRASS distribution, but can be downloaded and automatically installed within a GRASS release version. In a second step, after a review of the add-on's code structure and confirmation of general interest in its functionality, the add-on module can enter the core codebase, becoming a canonical functionality in GRASS GIS.

Once a sufficient amount of development goals have been reached at a given point of time, the ongoing GRASS development activities on a particular version are temporarily stopped ("frozen") by the developer community. Following a testing phase, the code snapshot of the actual project codebase is released as a new production grade GRASS GIS version, assigned with a distinctive version number and provided via the GRASS GIS Website. All versions of the GRASS code base since 1990 are provided in the distributed OSGeo server infrastructure which ensures reliable access and documentation to the evolving versions of the code base, allowing for retrospective transparent access, in the sense of Open Science. While users can also directly access the latest version of the unstable development code base in the repository instead of the officially published releases, this practice is discouraged due to the current fast pace of production grade releases (Neteler 2014). Despite GRASS GIS add-ons can be published anywhere in the internet, the authors are encouraged to make use the of official GRASS GIS add-ons repository to become part of automated provision of installable executables, to be included to the list of add-ons on the main project website and to become part of the source code peer-review mechanisms established within the project.

Currently, the GRASS GIS project homepage provides advice, how to cite versions of GRASS GIS in scientific publications and also provides a BiBTeX entry (https://grasswiki.osgeo.org/wiki/GRASS_Citation_Repository). In parallel, GRASS related code publications have started to appear on the open access research data repositories like Zenodo (https://zenodo.org/search?page=1size=20q=GRASS%20GIS) since 2015, which provide citation via DOIs.

From a technical perspective, the GRASS GIS codebase can be hosted on a variety of repository platforms: While copies of the GRASS codebase are available via the Git repository hosting service GitHub (https://travis-ci.org/GRASS-GIS/grass-ci), the primary GRASS code repository uses Apache Subversion (https://subversion.apache.org/) as its central software versioning and revision control system, hosted within the OSGeo IT infrastructure. Currently

GRASS add-on modules published on external repositories are not automatically merged with the primary GRASS repository, so sharing of such external-hosted GRASS code would require personal effort by the developers by submitting it to the main GRASS repository to benefit from the project's code review and testing.

A discussion of the usage of PIDs for software citation picked up in 2016 within the GRASS Project Steering Committee
(https://lists.osgeo.org/pipermail/grass-psc/2016-November/001606.html,
https://lists.osgeo.org/pipermail/grass-psc/2017-January/001628.html). Recent software repositories used in the social sciences were proposed as analogies and potential best practices for the GRASS GIS community. In addition, an exchange with the DataCite organization introduced the option of either the GRASS GIS project or the OSGeo umbrella organization becoming DOI registration services themselves, enabling novel citation services within the OSGeo communities. According to DataCite, the GRASS project infrastructure, meets and exceeds the minimal requirements for DataCite-approved DOI-registering infrastructures. Manual pages for GRASS modules already meet the requirements for DOI landing pages regarding identifier, creator, title, publisher, publication year and resource type (Figure 2). The discussion addressed the variety of approaches for software citation currently existing. To give the GRASS project's software due recognition (by easing scientific citation), several use cases have to be considered: citing the project as a container, irrespective of a particular software version (i.e. GRASS GIS),

1. citing a specific version (e.g. GRASS GIS 7.2), and
2. citing a particular version of a GRASS module (e.g. the g.access
(https://grass.osgeo.org/grass72/manuals/g.access.html) module in GRASS GIS 7.2.). For this purpose, a consistent approach is needed to cite a GRASS module when it is considered merely an add-on module and also after it has been included in the main GRASS functionality portfolio. The handling of software versioning has already been introduced by the Zenodo repository (http://blog.zenodo.org/2017/05/30/doi-versioning-launched). Consensus within the GRASS project community on how to proceed remains to be reached.

The current GRASS GIS repository meets these requirements for references to all published production versions of GRASS GIS and individual modules. This is demonstrated by the visual analysis (software container level) by (Neteler 2011, Figure 2) showcasing the evolution of the complexity of the GRASS release versions over 12 years, between 1999 and 2011, including the agents (individuals) as suggested by the CodeMeta project.

6. Summary and Outlook

As journal publications continue to be the standard form of scientific exchange, it is crucial to support the change of citation culture towards a more prominent recognition for code and data to foster Open Science. For geospatial research, both the GRASS GIS software project and the OSGeo umbrella organization have the capabilities to adopt DOI-based software citation due to advanced status of their internal best practices. OSGeo has already embraced DOI-citation as best practice for conference recordings since 2016. DOIs for code submissions can become an additional infrastructure offering for developers of research software. This will motivate submissions of additional research code to OSGeo projects and also foster journal publications about the code.

The approach towards DOIs for software within OSGeo needs to be independent from the current software repository situation, as it must be open towards future changes of underlying technical infrastructures. Bottle-necks and potential vendor lock-in must be avoided, as in the past.

As a follow up step, acceptance and usage of unique identifiers also for persons, like ORCID, can become a strong driver to distinguish and trace code submissions by individuals and its reuse as altmetrics, based on open standards and irrespective of underlying technical infrastructures. While such mechanisms are currently being developed for data and code citation by DataCite, they are already provided by the Crossref organization for DOI-based journal publications and ORCID (https://www.crossref.org/blog/crossrefs-doi-event-tracker-pilot/).

Based on the current state of technology and established best practices within OSGeo, the following scenarios appear feasible for the near future:

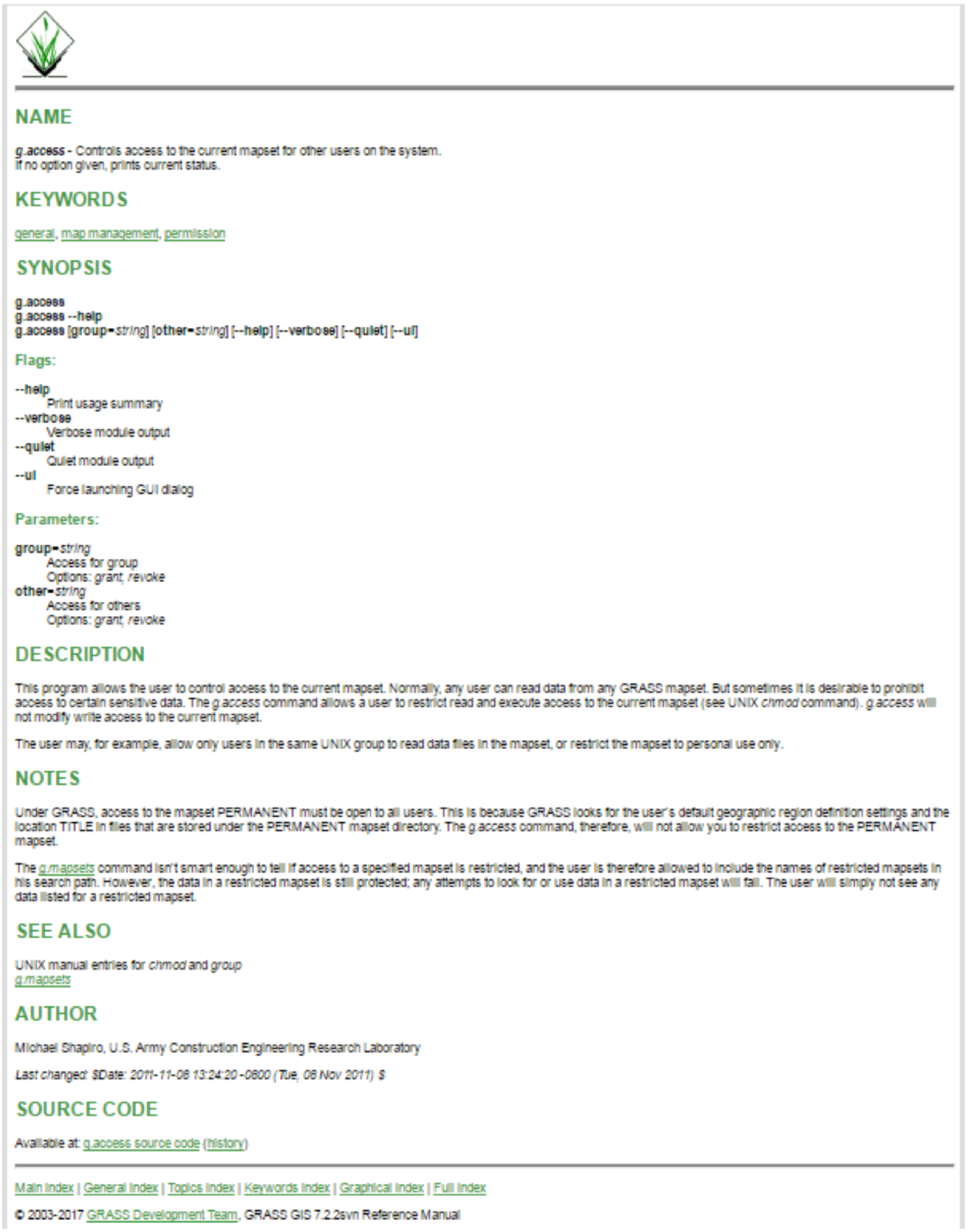
DOI-application within the GRASS GIS Project and impact on OSGeo (see Figure 3 for overview):

1. The GRASS GIS project has DOIs minted both for the overall GRASS GIS container and also individual code releases (e.g. GRASS7.0, GRASS7.2, etc.). This requires limited effort as only a very small number of DOIs is required.
2. In addition, individual GRASS modules also receive DOIs for each new version. The manual pages of the modules serve as landing pages. For this, recommendations by the Research Data Alliance (RDA) regarding the citation of evolving data could be applied to evolving code (Rauber et al. 2015).
3. Best practices for other OSGeo projects are derived from the lessons learned by the discussion within the GRASS project regarding DOI use.
4. Software citation by PID or other means is included in the OSGeo incubation roster for project quality assessment.

Management of DOI registration for OSGeo projects:

1. The GRASS GIS project becomes accredited as a data center via an allocating member of DataCite. Such a member organization could be like TIB or USGS. Other OSGeo projects can become independently accredited by other DataCite allocating members, to receive their own DOIs. All these DOIs will not be distinguishable as being related to OSGeo projects.
2. OSGeo joins the DataCite organization as an allocating member and becomes able to mint an unlimited amount of DOIs with its own distinctive DOI-prefix . Such OSGeo-DOIs can be provided to all OSGeo community projects for code, data and documentation. An example for this approach within DataCite is CodeOcean (https://search.datacite.org/members/ocean), which serves both as data center and DataCite allocating member.

In addition to the described bottom-up options and needs which have been described for the GRASS GIS project and OSGeo, top-down support by international research organizations and similar bodies is needed, to ensure that Open Science becomes an established paradigm in science: Openness and sustainability in science require an infrastructure of reliable future-safe repositories for data and code, but also the acknowledgement of their significance as scientific output. This significance needs to be reflected in the overall reputation system, to ensure viable incentives for early career scientists, to ensure that their code and data are properly published.



NAME

g.access - Controls access to the current mapset for other users on the system.
If no option given, prints current status.

KEYWORDS

general, map management, permission

SYNOPSIS

g.access
g.access --help
g.access [**group**=*string*] [**other**=*string*] [**--help**] [**--verbose**] [**--quiet**] [**--ui**]

Flags:

--help
Print usage summary
--verbose
Verbose module output
--quiet
Quiet module output
--ui
Force launching GUI dialog

Parameters:

group=*string*
Access for group
Options: *grant, revoke*
other=*string*
Access for others
Options: *grant, revoke*

DESCRIPTION

This program allows the user to control access to the current mapset. Normally, any user can read data from any GRASS mapset. But sometimes it is desirable to prohibit access to certain sensitive data. The *g.access* command allows a user to restrict read and execute access to the current mapset (see UNIX *chmod* command). *g.access* will not modify write access to the current mapset.

The user may, for example, allow only users in the same UNIX group to read data files in the mapset, or restrict the mapset to personal use only.

NOTES

Under GRASS, access to the mapset PERMANENT must be open to all users. This is because GRASS looks for the user's default geographic region definition settings and the location TITLE in files that are stored under the PERMANENT mapset directory. The *g.access* command, therefore, will not allow you to restrict access to the PERMANENT mapset.

The *g.mapsets* command isn't smart enough to tell if access to a specified mapset is restricted, and the user is therefore allowed to include the names of restricted mapsets in his search path. However, the data in a restricted mapset is still protected; any attempts to look for or use data in a restricted mapset will fail. The user will simply not see any data listed for a restricted mapset.

SEE ALSO

UNIX manual entries for *chmod* and *group*
g.mapsets

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2011-11-08 13:24:20 -0800 (Tue, 08 Nov 2011) $

SOURCE CODE

Available at: g.access source code (history)

Main index | General index | Topics index | Keywords index | Graphical index | Full index

© 2003-2017 GRASS Development Team, GRASS GIS 7.2.2svn Reference Manual

Figure 1: Example of a GRASS GIS standardized manual page for a module (g.access module).

This type of manual page can also serve as a landing page for a DOI, including a high level description, metadata, and links to the recent source code and previous versions. The module is an example for code which has been maintained by the project community for a long time after the original developer left the project (GIS 2017).

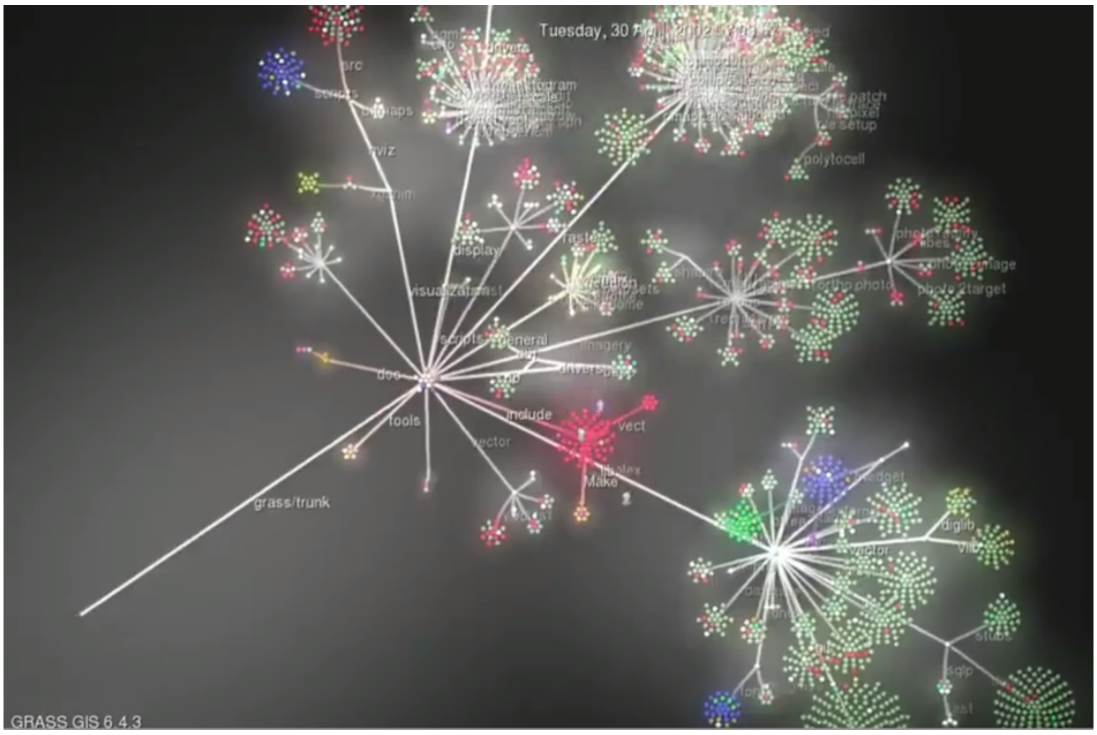


Figure 2: Still image from GRASS GIS source code evolution visualization (Neteler, M. 2013: https://doi.org/10.5446/14652t=00:52,09:33).

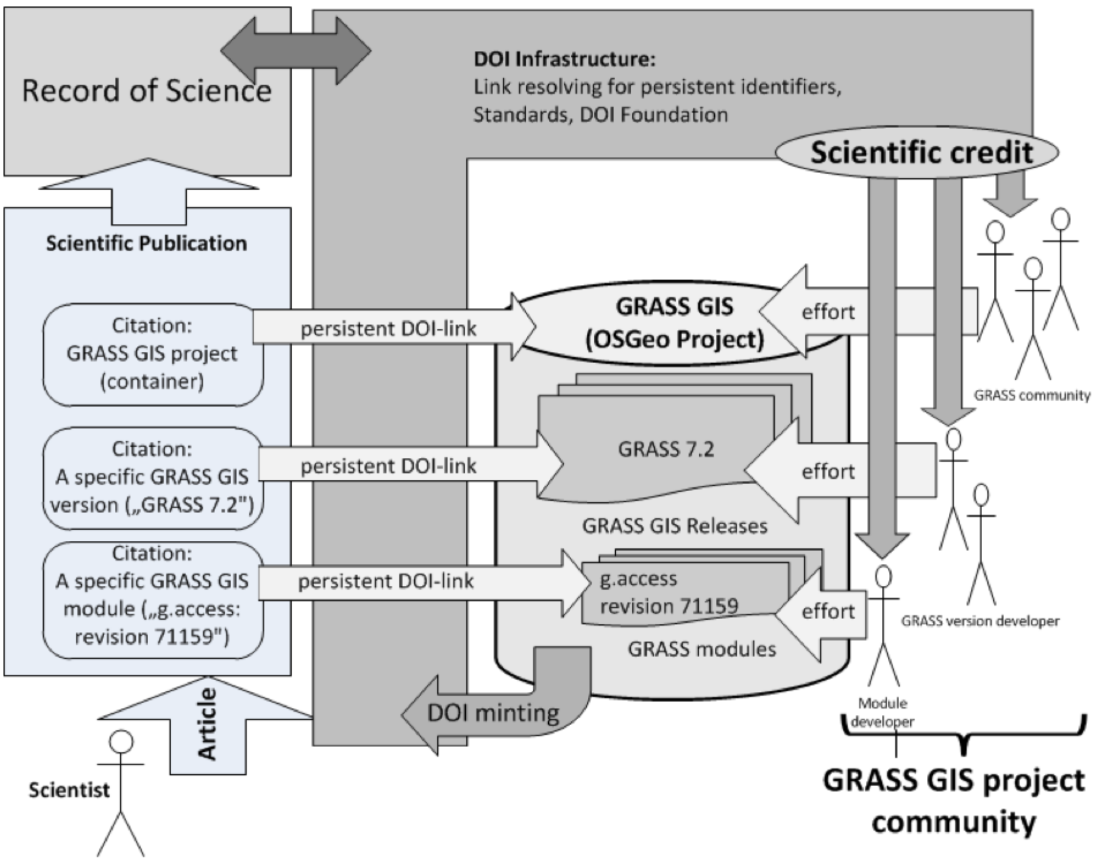


Figure 3: Proposed adoption of DOIs for software by the GRASS GIS project (high level view).

The animated visualization of the source code development (generated with the Gource software, http://gource.io/) shows as a tree the software development over time. In the beginning, the center is the root directory of the initial software project which then evolves over time. Newly created subdirectories are shown as branches with their files as leaves. The activity of software development is shown through small agents in the visualization which represent the various developers working on the source code over time. The GRASS GIS development visualization video summarizes fifteen years of development in which a major source code reorganization happened.

This high level view highlights the requirements, resulting new capabilities and emerging benefits resulting from the adoption of Digital Object Identifiers (DOI) by the GRASS GIS project. Once the GRASS GIS project starts minting DOIs via a member of a DOI registration agency, these DOIs can be used for scientific citation in journal publications and books, which become part of the record of science. The DOI infrastructure enables persistent links leading to the landing pages for software objects, on the container-, version- and code module-level. This allows to cite, and thereby provide due scientific credit, to the individuals who are maintaining and developing the codebase of the GRASS GIS project. The DOI infrastructure also enables the monitoring of citations, i.e. the re-use and possible extension of the cited software as a measure of the impact of the effort of the software authors.

References

Bilder, G., Lin, J., Neylon, C., 2015. Principles for open scholarly infrastructure-v1. Online, online; accessed May 31, 2017.
URL http://cameronneylon.net

Brett, A., Croucher, M., Haines, R., Hettrick, S., Hetherington, J., Stillwell, M., Wyatt, C., 2017. Research software engineers. State of the Nation Report, Zenodo.

GIS, G., 2017. Historical notes. Online, online; accessed May 31, 2017.
URL https://grass.osgeo.org/home/history/

Katz, D. S., 2017. Software citation: a cornerstone of software-enabled research. 2nd Conference on Non-Textual Information: Software and Services for Science.

Lowe, P., Neumann, J., Plank, M., Ziedorn, F., Lozar, R., Westervelt, J., Inman, R., 2015. Grass gis, star trek and old video tape. OSGeo Journal 14, 43–48.

Neteler, M., 2011. Grass gis 6.4 development visualization from 1999 to 2011 with gource. GRASS Development Team.

Neteler, M., Bowman, M. H., Landa, M., Metz, M., 2012. Grass gis: a multi-purpose open source gis. Environmental Modelling and Software 31, 124–130.

Rauber, A., Asmi, A., van Uytvanck, D., Proll, S., 2015. Data citation of evolving data - recommendations of the working group on data citataion (wgdc). Online, online; accessed May 31, 2017.
URL https://www.rd-alliance.org/rda-wgdc-recommendations-extended-description-tcdl-draft.html

Rother, K., 2015. How to recognize good scientific software? Online, online; accessed May 31, 2017.
URL http://www.academis.eu/posts/good_software

Sperber, W., 2017. Lost in the web? new approaches for information on mathematical software. Technische Informationsbibliothek (TIB).

Steiniger, S., Hunter, A. J. S., 2013. The 2012 free and open source gis software map - a guide to facilitate research, development and adoption. Computers, Environment and Urban Systems 39, 136–150.