SOFTWARE CITATION PRINCIPLES

ARFON M. SMITH[1], DANIEL S. KATZ[2], KYLE E. NIEMEYER[3], AND THE FORCE11 SOFTWARE CITATION WORKING GROUP

ABSTRACT. Software is a critical part of modern research and yet there is little support across the scholarly ecosystem for its acknowledgement and citation. Inspired by the activities of the FORCE11 working group focussed on data citation, this document summarizes the recommendations of the FORCE11 Software Citation Working Group and its activities between June 2015 and April 2016. Based on a review of existing community practices, the goal of the working group was to produce a consolidated set of citation principles that may encourage broad adoption of a consistent policy for software citation across disciplines and venues. Our work is presented here as a set of software citation principles, a discussion of the motivations for developing the principles, reviews of existing community practice, and a discussion of the requirements these principles would place upon different stakeholders. Working examples and possible technical solutions for how these principles can be implemented will be discussed in a separate paper.

1. Software citation principles

The principles in this section are written fairly concisely, and discussed further later in this document (§5). Here, for example, we do not define what software should be cited, but how it should be cited, and we talk about how such decisions might be made in the discussion section (§5).

(1) **Importance**: Software should be considered a legitimate and citable product of research. Software citations should be accorded the same importance in the scholarly record as citations of other research products, such as publications and data; they should be included in the metadata of the citing work, for example in the reference list of a journal article, and should not be omitted or separated. Software should be cited on the same basis as any other research product such as a paper or a book, that is, authors should cite the appropriate set of software products just as they cite the appropriate set of papers.

(2) **Credit and Attribution**: Software citations should facilitate giving scholarly credit and normative and legal attribution to all contributors to the software, recognizing that a single style or mechanism of attribution may not be applicable to all software.

(3) **Unique Identification**: A software citation should include a method for identification that is machine actionable, globally unique, interoperable, and recognized by at least a community of the corresponding domain experts, and preferably by general public researchers.

(4) **Persistence**: Unique identifiers and metadata describing the software and its disposition should persist – even beyond the lifespan of the software they describe.

(5) **Accessibility**: Software citations should facilitate access to the software itself and to its associated metadata, documentation, data, and other materials necessary for both humans and machines to make informed use of the referenced software.

Corresponding author: Daniel S. Katz[2], d.katz@ieee.org.

[1] GitHub, Inc., San Francisco, CA, USA.

[2] National Center for Supercomputing Applications & Electrical and Computer Engineering Department & School of Information Sciences, University of Illinois at Urbana–Champaign, Urbana, IL, USA.

[3] School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2169v2 | CC BY 4.0 Open Access | rec: 6 Jul 2016, publ: 6 Jul 2016

(6) **Specificity**: Software citations should facilitate identification of, and access to, the specific version of software that was used. Software identification should be as specific as necessary, such as using version numbers, revision numbers, or variants such as platforms.

These software citation principles were originally based on an adaptation of the FORCE11 Data Citation Principles [12], and then were modified based on discussions of the FORCE11 Software Citation Working Group (see Appendix A for members), information from the use cases in §3, and the related work in §4. The adaptations have been made because software, while similar to data in terms of not traditionally having been cited in publications, is also different than data in that it can be used to express or explain concepts, it is updated more frequently, and it is executable. Also, while software can be considered a type of data, the converse is not generally true.

2. Motivation

As the process of research[1] has become increasingly digital, research outputs and products have grown beyond simply papers and books to include software, data, and other electronic components such as presentation slides, posters, (interactive) graphs, maps, websites (e.g., blogs and forums), and multimedia (e.g., audio and video lectures). Research knowledge is embedded in these components. And papers and books themselves are also becoming increasingly digital, allowing them to become executable and reproducible. As we move towards this future where research is performed in and recorded as a variety of linked digital products, the characteristics and properties that developed for books and papers need to be applied to all digital products and possibly adjusted. Here, we are concerned specifically with the citation of software products. The challenge is not just the textual citation of software in a paper, but the more general identification of software used within the research process.

Software and other digital resources currently appear in publications in very inconsistent ways. For example, a random sample of 90 articles in the biology literature found seven different ways that software was mentioned, including simple names in the full-text, URLs in footnotes, and different kinds of mentions in references lists: project names or websites, user manuals, publications that describe or introduce the software [26]. Table 1 shows examples of these varied forms of software mentions and the frequency with which they were encountered. Many of these kinds of mentions fail to perform the functions needed of citations, and their very diversity and frequent informality undermines the integration of software work into bibliometrics and other analyses. Studies on data and facility citation have shown similar results [27, 39, 43].

Table 1. Varieties of software mentions in publications, from Howison and Bullard [26].

Mention Type	Count (n=286)	Percentage
Cite to publication	105	37%
Cite to users manual	6	2%
Cite to name or website	15	5%
Instrument-like	53	19%
URL in text	13	5%
In-text name only	90	31%
Not even name	4	1%

There are many reasons why this lack of both software citations in general and standard practices for software citation are of concern:

[1] We use the term "research" in this document to include work intended to increase human knowledge and benefit society, in science, engineering, humanities, and other areas.

- Understanding Research Fields: Software is a product of research, and by not citing it, we leave holes in the record of research of progress in those fields.
- Credit: Academic researchers at all levels, including students, postdocs, faculty, and staff, should be credited for the software products they develop and contribute to, particularly when those products enable or further research done by others.[2] Non-academic researchers should also be credited for their software work, though the specific forms of credit are different than for academic researchers.
- Discovering Software: Citations enable the specific software used in a research product to be found. Additional researchers can then use the same software for different purposes, leading to credit for those responsible for the software.
- Reproducibility: Citation of specific software used is necessary for reproducibility, but is not sufficient. Additional information such as configurations and platform issues are also needed.

The FORCE11 Software Citation Working Group [18] was created in April 2015 with the following mission statement:

> *The software citation working group is a cross-team committee leveraging the perspectives from a variety of existing initiatives working on software citation to produce a consolidated set of citation principles in order to encourage broad adoption of a consistent policy for software citation across disciplines and venues. The working group will review existing efforts and make a set of recommendations. These recommendations will be put of for endorsement by the organizations represented by this group and others that play an important role in the community.*
>
> *The group will produce a set of principles, illustrated with working examples, and a plan for dissemination and distribution. This group will not be producing detailed specifications for implementation although it may review and discuss possible technical solutions.*

The group gathered members (see Appendix A) in April and May 2015, and then began work in June, with a number of meetings and some off-line work by group members to gather materials documenting existing practices in member disciplines; gather materials from workshops and other reports; review those materials, identifying overlaps and differences; and subsequently draft this resulting document, which will be presented and discussed at the Force2016 Conference [20] in April 2016. We expect that this discussion may lead to a second, final version, and we also plan to have a follow-on working group that will work with stakeholders to ensure that these principles impact the research process.

The principles in this document should guide further development of software citation mechanisms and systems, and the reader should be able to look at any particular example of software citation and see if it meets the principles. Please note that while we strive to offer practical guidelines that acknowledge the current incentive system of academic citation, a more modern system of assigning credit is sorely needed. It is not that academic software needs a separate system from academic papers, but that the need for credit for application software underscores the need to overhaul the system of credit for all research products.

In the next section (§3), we provide some detailed context in which software citation is important, by means of use cases. In §4, we summarize and analyze a large amount of previous work and thinking in this area. In §5, we discuss issues related to the principles stated in §1, and finally, in §6 we discuss the work needed to lead to these software citation principles being applied.

[2] Providing recognition of software can have tremendous economic impact as demonstrated by the role of Text REtrieval Conference (TREC) in information retrieval [44].

3. Use cases

We have documented and analyzed a set of use cases related to software citation in [19]. Table 2 summarizes these use cases and makes clear what the requirements are for software citation in each case. Each example represents a particular stakeholder performing an activity related to citing software, with the given metadata as information needed to do that. In that table, we use the following definitions:

- "Researcher" includes both academic researchers (e.g., postdoc, tenure-track faculty member) and research software engineers.
- "Publisher" includes both traditional publishers that publish text and/or software papers as well as archives such as Zenodo that directly publish software.
- "Funder" is a group that funds software or work using software.
- "Indexer" examples include Scopus, Web of Science, Google Scholar, and Microsoft Academic Search.
- "Domain group/library/archive" includes the Astronomy Source Code Library (ASCL) [4], bioCADDIE [7], Computational Infrastructure for Geodynamics (CIG) [10], libraries, institutional archives, etc.
- "Repository" refers to public software repositories such as GitHub, Netlib, Comprehensive R Archive Network (CRAN), and institutional repositories.
- "Unique identifier" refers to unique, persistent, and machine-actionable identifiers such as a DOI, ARK, or PURL.
- "Description" refers to some description of the software such as an abstract, README, or other text description.
- "Keywords" refers to keywords or tags used to categorize the software.
- "Reproduce" can mean actions focused on reproduction, replication, verification, validation, repeatability, and/or utility.
- "Citation manager" refers to people and organizations that create scholarly reference management software and websites including Zotero, Mendeley, EndNote, RefWorks, BibDesk, etc., that manage citation information and semi-automatically insert those citations into research products.

All use cases assume the existence of a citable software object, typically created by the authors/developers of the software. Developers can achieve this by, e.g., uploading a software release to figshare [15] or Zenodo [24] to obtain a DOI. Necessary metadata should then be included in a `CITATION` file [55] or machine-readable `CITATION.jsonld` file [35]. When software is not freely available (e.g., commercial software) or when there is no clear identifier to use, alternative means may be used to create citable objects as discussed in §5.9.

In some cases, if particular metadata are not available, alternatives may be provided. For example, if the version number and release date are not available, the download date can be used. And the contact name/email is an alternative to the location/repository.

4. Related work

With approximately 50 working group participants (see Appendix A) representing a range of research domains, the working group was tasked to document existing practices in their respective communities. A total of 47 documents were submitted by working group participants, with the life sciences, astrophysics, and geosciences being particularly well-represented in the submitted resources.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2169v2 | CC BY 4.0 Open Access | rec: 6 Jul 2016, publ: 6 Jul 2016

TABLE 2. Use cases and basic metadata requirements for software citation, adapted from [19]. Solid circles (•) indicate that the use case depends on that metadata, while open circles (∘) indicate that the use case would benefit from that metadata if available.

	Basic requirements											
Use case	Unique identifier	Software name	Author(s)	Contributor role	Version number	Release date	Location/repository	Indexed citations	Software license	Description	Keywords	Example stakeholder(s)
1. Use software for a paper	•	•	•		•	•	•		∘			Researcher
2. Use software in/with new software	•	•	•		•	•	•		∘			Researcher, software engineer
3. Contribute to software	•	•	•	∘	•	•	•		∘	∘		Researcher, software engineer
4. Determine use/citations of software	•	•						•				Researcher, software engineer
5. Get credit for software development	•	•	•	∘		•	•					Researcher, software engineer
6. “Reproduce” analysis	•	•			•	•	•		∘	∘		Researcher
7. Benchmark software	•	•			•	•	•		∘	∘		Researcher, software engineer
8. Find software to implement task	•	•	•				•	•	∘	∘	∘	Researcher, software engineer
9. Publish software paper	•	•	•		•	•	•					Publisher
10. Publish papers that cite software	•	•	•		•	•	•	•				Publisher
11. Build catalog of software	•	•	•		•	•	•	•	∘	∘	∘	Indexer
12. Build software catalog/registry	•	•	•				•			∘	∘	Domain group, library, archive
13. Show scientific impact of holdings	•	•						•				Repository
14. Show how funded software has been used	•	•						•				Funder, policy maker
15. Evaluate contributions of researcher	•		•	∘		•		•				Evaluator, funder
16. Store software entry	•	•	•		•	•	•	•				Citation manager
17. Publish mixed data/software packages	•	•	•		•	•	•		∘	∘	∘	Repository, library, archive

4.1. **General community/non domain-specific activities.** Some of the most actionable work has come from the UK Software Sustainability Institute (SSI) in the form of blog posts written by their community fellows:

In a blog post from 2012, Jackson discusses some of the pitfalls of trying to cite software in publications [30]. He includes useful guidance for when to consider citing software as well as some ways to help “convince” journal editors to allow the inclusion of software citations.

Wilson suggests that software authors include a `CITATION` file that documents exactly how the authors of the software would like to be cited by others [55]. While this is not a formal metadata specification (e.g., it is not machine readable) this does offer a solution for authors wishing to give explicit instructions to potential citing authors and as noted in the motivation section (§2), there is evidence that authors follow instructions if they exist [27].

In a later post on the SSI blog, Jackson gives a good overview of some of the approaches package authors have taken to automate the generation of citation entities such as BibTeX entries [31], and Knepley et al. do similarly [36].

While not usually expressed as software citation principles, a number of groups have developed community guidelines around software and data citation. Van de Sompel et al. [51] argue for registration of all units of scholarly communication, including software. In “Publish or be damned? An alternative impact manifesto for research software” [9], Chue Hong lists nine principles as part of “The Research Software Impact Manifesto.” In the “Science Code Manifesto” [5], the founding signatories cite five core principles (Code, Copyright, Citation, Credit, Curation) for scientific software.

Perhaps in recognition of the broad range of research domains struggling with the challenge of better recognizing the role of software, funders and agencies in both the US (e.g., NSF, NIH, Alfred P. Sloan Foundation) and UK (e.g., SFTC, JISC, Wellcome Trust) have sponsored or hosted a number of workshops with participants from across a range of disciplines, specifically aimed at discussing issues around software citation [50, 2, 47, 41, 45, 3]. In many cases these workshops produced strong recommendations for their respective communities on how best to proceed. In addition, a number of common themes arose in these workshops, including (1) the critical need for making software more "citable" (and therefore actions authors and publishers should take to improve the status quo), (2) how to better measure the impact of software (and therefore attract appropriate funding), and (3) how to properly archive software (where, how, and how often) and how this affects what to cite and when.

Most notable of the community efforts are those of WSSSPE [56] and SSI [48], who between them have run a series of workshops aimed at gathering together community members with an interest in(1) defining the set of problems related to the role of software and associated people in research settings, particularly academia, (2) discussing potential solutions to those problems, (3) beginning to work on implementing some of those solutions. In each of the three years that WSSSPE workshops have run thus far, the participants have produced a report [32, 33, 34] documenting the topics covered. Section 5.8 and Appendix J in the WSSSPE3 report [34] has some preliminary work and discussion particularly relevant to this working group. In addition, a number of academic publishers such as APA [40] have recommendations for submitting authors on how to cite software, and journals such as *F1000Research* [14], *SoftwareX* [46], *Open Research Computation* [42], and the *Journal of Open Research Software* allow for submissions entirely focused on research software.

4.2. **Domain-specific community activities.** One approach to increasing software "citability" is to encourage the submission of papers in standard journals describing a piece of research software, often known as software papers (see §5.2). While some journals (e.g., Transactions on Mathematical Software (TOMS), Bioinformatics, Computer Physics Communications, F1000Research, Seismological Research Letters, Electronic Seismologist) have traditionally accepted software submissions, the American Astronomical Society (AAS) has recently announced they will accept software papers in their journals [1]. Professional societies are in a good position to change their respective communities, as the publishers of journals and conveners of domain-specific conferences; as publishers they can change editorial policies (as AAS has done) and conferences are an opportunity to communicate and discuss these changes with their communities.

In astronomy and astrophysics: The Astronomy Source Code Library (ASCL) [4], is a website dedicated to the curation and indexing of software used in the astronomy-based literature. In 2015, the AAS and GitHub co-hosted a workshop [41] dedicated to software citation, indexing, and discoverability in astrophysics. More recently, a Birds of a Feather session was held at the Astronomical Data Analysis Software and Systems (ADASS) XXV conference [3] that included discussion of software citation.

In the life sciences: In May 2014, the NIH held a workshop aimed at helping the biomedical community discover, cite, and reuse software written by their peers. The primary outcome of this workshop was the Software Discovery Index Meeting Report [53] which was shared with the community for public comment and feedback. The authors of the report discuss what framework would be required for supporting a Software Discovery Index including the need for unique identifiers, how citations to these would be handled by publishers, and the critical need for metadata to describe software packages.

In the geosciences: The Ontosoft [23] project describes itself as "A Community Software Commons for the Geosciences." Much attention was given to the metadata required to describe, discover, and execute research software. The NSF-sponsored Geo-Data Workshop 2011 [21] revolved around data lifecycle, management, and citation. The workshop report includes many recommendations for data citation.

4.3. Existing efforts around metadata standards.

Producing detailed specifications and recommendations for possible metadata standards to support software citation was not within the scope of this working group. However some discussion on the topic did occur and there was significant interest in the wider community to produce standards for describing research software metadata.

Content specifications for software metadata vary across communities, and include DOAP [13], an early metadata term set used by the Open Source Community, as well as more recent community efforts like Research Objects [6], The Software Ontology [38], EDAM Ontology [28], Project CRediT [11], the OpenRIF Contribution Role Ontology [25], Ontosoft [23], RRR/JISC guidelines [22], or the terms and classes defined at Schema.org related to the `SoftwareApplication` class. In addition, language-specific software metadata schemes are in widespread use, including the Debian package format [29], Python package descriptions [52], and R package descriptions [54], but these are typically conceived for software build, packaging, and distribution rather than citation. CodeMeta [8] has created a crosswalk among these software metadata schemes and an exchange format that allows software repositories to effectively interoperate.

5. Discussion

In this section we discuss some the issues and concerns related to the principles stated in Section 1.

5.1. What software to cite.

The software citation principles do not define what software should be cited, but rather, how software should be cited. What software should be cited is the decision of the author(s) of the research work in the context of community norms and practices, and in most research communities, these are currently in flux. In general, *we believe that software should be cited on the same basis as any other research product such as a paper or book*; that is, authors should cite the appropriate set of software products just as they cite the appropriate set of papers, perhaps following the FORCE11 Data Citation Working Group principles, which state, “In scholarly literature, whenever and wherever a claim relies upon data, the corresponding data should be cited.” [12]

Note that some software which is or could be captured as part of data provenance may not be cited. Citation is a record of software that is important to a research outcome, where provenance is a record of all steps (including software) used to generated particular data within the research process. This implies that for a data research product, provenance data will include all cited software, but not necessarily vice versa. Similarly, the software metadata that is recorded as part of data provenance should be a superset of the metadata recorded as part of software citation. The data recorded for reproducibility should also be a superset of the metadata recorded as part of software citation. These statements may also be true for software products. In general, we intend the software citation principles to cover the minimum of what is necessary for software citation for the purpose of software identification. Other use cases (e.g., provenance, reproducibility) may lead to additional requirements (i.e., enhanced metadata).

5.2. Software papers.

Currently, and for the foreseeable future, software papers are being published and cited, in addition to software itself being published and cited, as many community norms and practices are oriented towards citation of papers. As discussed in the Importance principle (1) and the discussion above, *the software itself should be cited on the same basis as any other research product; authors should cite the appropriate set of software products*. If a software paper exists and it contains results (performance, validation, etc.) that are important to the work, then the software paper should also be cited. In addition, if the software authors ask that a paper should be cited, that should typically be respected and the paper cited *in addition to* the software being cited.

5.3. **Derived software.** The goals of software citation include the linked ideas of crediting those responsible for software and understanding the dependencies of research products on specific software. In the Importance principle (1), we state that "software should be cited on the same basis as any other research product such as a paper or a book; that is, authors should cite the appropriate set of software products just as they cite the appropriate set of papers." In the case of one code that is derived from another code, citing the derived software may appear to not credit those responsible for the original software, nor recognize its role in the work that used the derived software. However, this is really analogous to how any research builds on other research, where each research product just cites those products that it directly builds on, not those that it indirectly builds on. Understanding these chains of knowledge and credit have been part of the history of science field for some time, though more recent work is suggesting more nuanced evaluation of the credit chains [11, 35].

5.4. **Software peer review.** Adherence to the software citation principles enables better peer reviews through improved reproducibility. However, since the primary goal of software citation is to identify the software that has been used in a scholarly product, the peer review of software itself is mostly out of scope in the context of software citation principles. For instance, when identifying a particular software artifact that has been used in a scholarly product, whether or not that software has been peer-reviewed is irrelevant. One possible exception would be if the peer-review status of the software should be part of the metadata, but the working group does not believe this to be part of the minimal metadata needed to identify the software.

5.5. **Citation format in reference list.** Citations in references in the scholarly literature are formatted according to the citation style (e.g., AMS, APA, Chicago, MLA) used by that publication. (Examples illustrating these styles have been published by Lipson [37]; the follow-on Software Citation Implementation Group will provide suggested examples.) As these citations are typically sent to publishers as text formatted in that citation style, not as structured metadata, and because the citation style dictates how the human reader sees the software citation, *we recommend that all text citation styles support the following: a) a label indicating that this is software, e.g., [Software], potentially with more information such as [Software: Source Code], [Software: Executable], or [Software: Container], and b) support for version information, e.g., Version 1.8.7.*

5.6. **Citations limits.** This set of software citation principles, if followed, will cause the number of software citations in scholarly products to increase, thus causing the number of overall citations to increase. Some scholarly products, such as journal articles, may have strict limits on the number of citations they permit, or page limits that include reference sections. Such limits are counter to our recommendation, and *we recommend that publishers using strict limits for the number of citations add specific instructions regarding software citations to their author guidelines to not disincentivize software citation.* Similarly, *publishers should not include references in the content counted against page limits.*

5.7. **Unique identification.** The Unique Identification principle (3) calls for "a method for identification that is machine actionable, globally unique, interoperable, and recognized by a community." What this means for data is discussed in detail in the "Unique Identification" section of a report by the FORCE11 Data Citation Implementation Group (DCIG) [49], which calls for "unique identification in a manner that is machine-resolvable on the Web and demonstrates a long-term commitment to persistence." This report also lists examples of identifiers that match these criteria including DOIs, PURLs, Handles, ARKS, and NBNs. For software, *we recommend the use of DOIs as the unique identifier due to their common usage and acceptance, particularly as they are the standard for other digital products such as publications.*

Note that the "unique" in a UID means that it points to a unique, specific software. However, multiple UIDs might point to the same software. This is not recommended, but is possible. *We strongly recommend that if there is already a UID for a version of software, no additional UID should be created.* Multiple UIDs can lead to split credit, which goes against the Credit and Attribution principle (2).

Software versions and identifiers. There are at least three different potential relationships between identifiers and versions of software.

(1) An identifier can point to a specific version of a piece of software.
(2) An identifier can point to the piece of software, effectively all versions of the software.
(3) An identifier can point to the latest version of a piece of software.

It is possible that a given piece of software may have identifiers of all three types. And in addition, there may be one or more software papers, each with an identifier.

While we often need to cite a specific version of software, we may also need a way to cite the software in general, or the latest version, and to link multiple releases together, perhaps for the purpose of understanding citations to the software. The principles in §1 are intended to be applicable at all levels, and to all types of identifiers, such as DOIs, RRIDs, etc., though we again recommend when possible the use of DOIs that identify specific versions of source code. We note that RRIDs were developed by the FORCE11 Resource Identification Initiative [16] and have been discussed for use to identify software packages (not specific versions), though the FORCE11 Resource Identification Technical Specifications Working Group [17] says "Information resources like software are better suited to the Software Citation WG." There is currently a lack of consensus on the use of RRIDs for software.

5.8. **Types of software.** The principles and discussion in this document have generally been written to focus on software as source code. However, we recognize that some software is only available as an executable, a container, or a virtual machine image, while other software may be available as a service. We believe the principles apply to all of these forms of software, though the implementation of them will certainly differ based on software type. *When software exists as both source code and another type, we recommend that the source code be cited.*

5.9. **Access to software.** The Accessibility principle (5) states that "software citations should permit and facilitate access to the software itself." This does not mean that the software must be freely available. Rather, the metadata should provide enough information that the software can be accessed. If the software is free, the metadata will likely provide an identifier that can be resolved to a URL pointing to the specific version of the software being cited. For commercial software, the metadata should still provide information on how to access the specific software, but this may be a company's product number or a link to a web site that allows the software be purchased. As stated in the Persistence principle (4), we recognize that the software version may no longer be available, but it still should be cited along with information about how it was accessed.

5.10. **What an identifier should resolve to.** While citing an identifier that points to, e.g., a GitHub repository can satisfy the principles of Unique Identification (3), Accessibility (5), and Specificity (6), such a repository cannot guarantee Persistence (4). *Therefore, we recommend that the software identifier should resolve to a persistent landing page that contains metadata and a link to the software itself, rather than directly to the source code files, repository, or executable.* This ensures longevity of the software metadata—even perhaps beyond the lifespan of the software they describe. This is currently offered by services such as figshare [15] and Zenodo [24], which both generate persistent DataCite DOIs for submitted software. In addition, such landing pages can contain both human-readable metadata (e.g., the types shown by Table 2) as well as content-negotiable formats such as RDF or DOAP [13].

5.11. **Updates to this document.** As this set of software citation principles has been created by the FORCE11 Software Citation Working Group, which will cease work and dissolve after these principles have been published, any updates will require a different FORCE11 working group to make them. As mentioned in §6, we expect a follow-on working group to be established to promote the implementation of these principles, and it is possible that this group might find items that need correction or addition in these principles. *We recommend that this Software Citation Implementation Working Group be charged, in part, with updating these principles during its lifetime, and that FORCE11 should listen to community requests for later updates and respond by creating a new working group.*

6. Future work

Software citation principles without clear worked-through examples are of limited value to potential implementers, and so in addition to this principles document, the final deliverable of this working group will be an implementation paper outlining working examples for each of the use cases listed in §3.

Following these efforts, we expect that FORCE11 will start a new working group with the goals of supporting potential implementers of the software citation principles and concurrently developing potential metadata standards, loosely following the model of the FORCE11 Data Citation Working Group. Beyond the efforts of this new working group, additional effort should be focused on updating the overall academic credit/citation system.

Appendix A. Working Group Membership

Alberto Accomazzi, Harvard-Smithsonian CfA
Alice Allen, Astrophysics Source Code Library
Micah Altman, MIT
Jay Jay Billings, Oak Ridge National Laboratory
Carl Boettiger, University of California, Berkeley
Jed Brown, University of Colorado Boulder
Sou-Cheng T. Choi, NORC at the University of Chicago & Illinois Institute of Technology
Neil Chue Hong, Software Sustainability Institute
Tom Crick, Cardiff Metropolitan University
Mercè Crosas, IQSS, Harvard University
Scott Edmunds, GigaScience, BGI Hong Kong
Christopher Erdmann, Harvard-Smithsonian CfA
Martin Fenner, DataCite
Darel Finkbeiner, OSTI
Ian Gent, University of St Andrews, recomputation.org
Carole Goble, The University of Manchester, Software Sustainability Institute
Paul Groth, Elsevier Labs
Melissa Haendel, Oregon Health and Science University
Stephanie Hagstrom, FORCE11
Robert Hanisch, National Institute of Standards and Technology, One Degree Imager
Edwin Henneken, Harvard-Smithsonian CfA
Ivan Herman, World Wide Web Consortium (W3C)
James Howison, University of Texas

Lorraine Hwang, University of California, Davis
Thomas Ingraham, F1000Research
Matthew B. Jones, NCEAS, University of California, Santa Barbara
Catherine Jones, Science and Technology Facilities Council
Daniel S. Katz, University of Illinois (co-chair)
Alexander Konovalov, University of St Andrews
John Kratz, California Digital Library
Jennifer Lin, Public Library of Science
Frank Löffler, Louisiana State University
Brian Matthews, Science and Technology Facilities Council
Abigail Cabunoc Mayes, Mozilla Science Lab
Daniel Mietchen, National Institutes of Health
Bill Mills, TRIUMF
Evan Misshula, CUNY Graduate Center
August Muench, American Astronomical Society
Fiona Murphy, Independent Researcher
Lars Holm Nielsen, CERN
Kyle E. Niemeyer, Oregon State University (co-chair)
Karthik Ram, University of California, Berkeley
Fernando Rios, Johns Hopkins University
Ashley Sands, University of California, Los Angeles
Soren Scott, Independent Researcher
Frank J. Seinstra, Netherlands eScience Center
Arfon Smith, GitHub (co-chair)
Kaitlin Thaney, Mozilla Science Lab
Ilian Todorov, Science and Technology Facilities Council
Matt Turk, University of Illinois
Miguel de Val-Borro, Princeton University
Daan Van Hauwermeiren, Ghent University
Stijn Van Hoey, Ghent University
Belinda Weaver, The University of Queensland
Nic Weber, University of Washington iSchool

References

[1] AAS Editorial Board. Policy statement on software. http://journals.aas.org/policy/software.html. Accessed: 2016-02-17.

[2] S. Ahalt, T. Carsey, A. Couch, R. Hooper, L. Ibanez, R. Idaszak, M. B. Jones, J. Lin, and E. Robinson. NSF workshop on supporting scientific discovery through norms and practices for software and data citation and attribution. Technical report, National Science Foundation, Apr. 2015. Available at: http://dl.acm.org/citation.cfm?id=2795624.

[3] A. Allen, G. B. Berriman, K. DuPrie, J. Mink, R. Nemiroff, T. Robitaille, L. Shamir, K. Shortridge, M. Taylor, P. Teuben, and J. Wallin. Improving software citation and credit. Technical report, arXiv, 2015. arXiv:1512.07919 [cs.DL].

[4] Astrophysics Source Code Library. http://ascl.net. Accessed: 2016-02-21.

[5] N. Barnes, D. Jones, P. Norvig, C. Neylon, R. Pollock, J. Jackson, V. Stodden, and P. Suber. Science code manifesto. http://sciencecodemanifesto.org. Accessed: 2016-04-18.

[6] S. Bechhofer, I. Buchan, D. D. Roure, P. Missier, J. Ainsworth, J. Bhagat, P. Couch, D. Cruickshank, M. Delderfield, I. Dunlop, M. Gamble, D. Michaelides, S. Owen, D. Newman, S. Sufi, and C. Goble. Why linked data is not enough for scientists. *Future Generation Computer Systems*, 29(2):599–611, 2013.

[7] biomedical and healthCAre Data Discovery Index Ecosystem (bioCADDIE). https://biocaddie.org. Accessed: 2016-03-06.

[8] C. Boettiger and M. B. Jones. Minimal metadata schemas for science software and code, in JSON and XML. https://github.com/codemeta/codemeta. Accessed: 2016-03-25.

[9] N. Chue Hong. Publish or be damned? An alternative impact manifesto for research software. http://www.software.ac.uk/blog/2011-05-02-publish-or-be-damned-alternative-impact-manifesto-research-software. Accessed: 2016-02-17.

[10] Computational Infrastructure for Geodynamics. https://geodynamics.org.

[11] Consortia Advancing Standards in Research Administration Information. http://casrai.org/CRediT. Accessed: 2016-02-17.

[12] Data Citation Synthesis Group, M. Martone (ed). Joint declaration of data citation principles. Final document, FORCE11, San Diego CA, 2014. https://www.force11.org/group/joint-declaration-data-citation-principles-final.

[13] E. Dumbill. DOAP: Description of a project. https://github.com/edumbill/doap/. Accessed: 2016-03-31.

[14] F1000Research. http://f1000research.com/for-authors/article-guidelines/software-tool-articles. Accessed: 2016-03-28.

[15] figshare. https://figshare.com/. Accessed: 2016-06-23.

[16] FORCE11 Resource Identification Initiative. https://www.force11.org/group/resource-identification-initiative.

[17] FORCE11 Resource Identification Technical Specifications Working Group. https://www.force11.org/group/resource-identification-technical-specifications-working-group.

[18] FORCE11 Software Citation Working Group. https://www.force11.org/group/software-citation-working-group.

[19] FORCE11 Software Citation Working Group. Software citation use cases. https://docs.google.com/document/d/1dS0SqGoBIFwLB5G3HiLLEOSAAgMdo8QPEpjYUaWCvIU, 2016. Accessed: 2016-02-10.

[20] FORCE2016 Conference. Portland, OR, https://www.force11.org/meetings/force2016.

[21] P. Fox and R. Signell. NSF geo-data informatics: Exploring the life cycle, citation and integration of geo-data workshop report. Final document, Rensselaer Polytechnic Institute, 2011. http://tw.rpi.edu/web/workshop/community/GeoData2011.

[22] I. Gent, C. Jones, and B. Matthews. Guidelines for persistently identifying software using DataCite. a JISC research Data Spring project. http://rrr.cs.st-andrews.ac.uk/wp-content/uploads/2015/10/guidelines-software-identification.pdf, Sept. 2015. Accessed: 2016-04-25.

[23] Y. Gil, V. Ratnakar, and D. Garijo. OntoSoft: Capturing scientific software metadata. In *Proceedings of the Eighth ACM International Conference on Knowledge Capture (K-CAP)*, Oct. 2015. http://dx.doi.org/10.1145/2815833.2816955.

[24] GitHub. Making your code citable with GitHub & Zenodo. https://guides.github.com/activities/citable-code/, 2014. Accessed: 2016-03-10.

[25] K. Gutzman, S. Konkiel, M. White, M. Brush, V. Ilik, M. Conlon, M. Haendel, and K. Holmes. Attribution of work in the scholarly ecosystem. *figshare*, Apr. 2016. http://dx.doi.org/10.6084/m9.figshare.3175198.v1.

[26] J. Howison and J. Bullard. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. *Journal of the Association for Information Science and Technology*, 2015. In press. http://dx.doi.org/10.1002/asi.23538.

[27] Y.-H. Huang, P. W. Rose, and C.-N. Hsu. Citing a data repository: A case study of the protein data bank. *PLoS ONE*, 10(8):1–17, 08 2015. http://dx.doi.org/10.1371/journal.pone.0136631.

[28] J. Ison, M. Kalaš, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer, and P. Rice. EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, 29(10):1325–1332, 2013. http://dx.doi.org/10.1093/bioinformatics/btt113.

[29] I. Jackson and C. Schwarz. Debian policy manual. https://www.debian.org/doc/debian-policy/ch-controlfields.html. Accessed: 2016-04-17.

[30] M. Jackson. How to cite and describe software. http://www.software.ac.uk/how-cite-and-describe-software. Accessed: 2016-02-17.

[31] M. Jackson. Oh research software, how shalt I cite thee? http://www.software.ac.uk/blog/2014-07-30-oh-research-software-how-shalt-i-cite-thee. Accessed: 2016-02-17.

[32] D. S. Katz, S.-C. T. Choi, H. Lapp, K. Maheshwari, F. Löffler, M. Turk, M. Hanwell, N. Wilkins-Diehr, J. Hetherington, J. Howison, S. Swenson, G. Allen, A. Elster, B. Berriman, and C. Venters. Summary of the first workshop on sustainable software for science: Practice and experiences (WSSSPE1). *Journal of Open Research Software*, 2(1):e6, 2014. http://dx.doi.org/10.5334/jors.an.

[33] D. S. Katz, S.-C. T. Choi, N. Wilkins-Diehr, N. Chue Hong, C. C. Venters, J. Howison, F. J. Seinstra, M. Jones, K. Cranston, T. L. Clune, M. de Val-Borro, and R. Littauer. Report on the second workshop on sustainable

software for science: Practice and experiences (WSSSPE2). *Journal of Open Research Software*, 4(1):e7, 2016. http://doi.org/10.5334/jors.85.

[34] D. S. Katz, S. T. Choi, K. E. Niemeyer, J. Hetherington, F. Löffler, D. Gunter, R. Idaszak, S. R. Brandt, M. A. Miller, S. Gesing, N. D. Jones, N. Weber, S. Marru, G. Allen, B. Penzenstadler, C. C. Venters, E. Davis, L. Hwang, I. Todorov, A. Patra, and M. de Val-Borro. Report on the third workshop on sustainable software for science: Practice and experiences (WSSSPE3). Technical report, arXiv, 2016. `arXiv:1602.02296 [cs.SE]`.

[35] D. S. Katz and A. M. Smith. Implementing transitive credit with JSON-LD. *Journal of Open Research Software*, 3:e7, 2015. http://dx.doi.org/10.5334/jors.by.

[36] M. G. Knepley, J. Brown, L. C. McInnes, and B. F. Smith. Accurately citing software and algorithms used in publications. figshare, http://dx.doi.org/10.6084/m9.figshare.785731.v1, 2013.

[37] C. Lipson. *Cite Right, Second Edition: A Quick Guide to Citation Styles–MLA, APA, Chicago, the Sciences, Professions, and More*. Chicago Guides to Writing, Editing, and Publishing. University of Chicago Press, 2011.

[38] J. Malone, A. Brown, A. L. Lister, J. Ison, D. Hull, H. Parkinson, and R. Stevens. The Software Ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation. *Journal of Biomedical Semantics*, 5(1):1–13, 2014. http://dx.doi.org/10.1186/2041-1480-5-25.

[39] M. Mayernik, K. Maull, and D. Hart. Tracing the use of research resources using persistent citable identifiers. https://share.renci.org/SI2PI2015/2015_SI2PI_Posters/mayernik_SI2poster_Feb2015.pdf, 2015. Poster presented at NSF SI2 PI Meeting, Arlington, VA, Accessed: 2016-03-03.

[40] T. McAdoo. http://blog.apastyle.org/apastyle/2015/01/how-to-cite-software-in-apa-style.html.

[41] L. Norén. Invitation to comment on a proposal for a cohesive research software citation-enabling platform. http://astronomy-software-index.github.io/2015-workshop/. Accessed: 2016-02-17.

[42] Open Research Computation. http://www.openresearchcomputation.com. Accessed: 2016-03-28.

[43] M. A. Parsons, R. Duerr, and J.-B. Minster. Data citation and peer review. *Eos, Transactions American Geophysical Union*, 91(34):297–298, 2010. http://dx.doi.org/10.1029/2010EO340001.

[44] B. R. Rowe, D. W. Wood, A. N. Link, and D. A. Simoni. Economic impact assessment of NIST's Text REtrieval Conference (TREC) program. Final report, National Institute of Standards and Technology, 2010. http://trec.nist.gov/pubs/2010.economic.impact.pdf [Accessed 2016-04-17].

[45] Software for Science: Getting Credit for Code. https://geodynamics.org/cig/projects/saga/, 2015. Accessed: 2016-04-06.

[46] SoftwareX. http://www.journals.elsevier.com/softwarex/. Accessed: 2016-03-28.

[47] Software Credit Workshop. http://www.software.ac.uk/software-credit, 2015. Accessed: 2016-04-06.

[48] SSI Workshops. http://www.software.ac.uk/community/workshops. Accessed: 2016-03-31.

[49] J. Starr, E. Castro, M. Crosas, M. Dumontier, R. R. Downs, R. Duerr, L. Haak, M. Haendel, I. Herman, S. Hodson, J. Hourclé, J. E. Kratz, J. Lin, L. H. Nielsen, A. Nurnberger, S. Proell, A. Rauber, S. Sacchi, A. Smith, M. Taylor, and T. Clark. Achieving human and machine accessibility of cited data in scholarly publications. *PeerJ Computer Science*, 1:e1, 5 2015. https://dx.doi.org/10.7717/peerj-cs.1.

[50] S. Sufi, N. P. Chue Hong, S. Hettrick, M. Antonioletti, S. Crouch, A. Hay, D. Inupakutika, M. Jackson, A. Pawlik, G. Peru, J. Robinson, L. Carr, D. De Roure, C. Goble, and M. Parsons. Software in reproducible research: Advice and best practice collected from experiences at the collaborations workshop. In *Proc. 1st ACM SIGPLAN Work. on Reproducible Research Methodologies and New Publication Models in Comp. Eng.*, TRUST '14, pages 2:1–2:4, Edinburgh, United Kingdom, June 2014. ACM.

[51] H. Van de Sompel, S. Payette, J. Erickson, C. Lagoze, and S. Warner. Rethinking scholarly communication: Building the system that scholars deserve. *D-Lib Magazine*, 10(9), Sept. 2004. http://www.dlib.org/dlib/september04/vandesompel/09vandesompel.html.

[52] G. Ward and A. Baxter. Distributing python modules. https://docs.python.org/2/distutils/setupscript.html#additional-meta-data. Accessed: 2016-04-17.

[53] O. White, A. Dhar, V. Bonazzi, J. Couch, and C. Wellington. NIH Software Discovery Index Meeting Report. Copies of archived content from final document, NIH, 2014. http://www.softwarediscoveryindex.org/ & https://gist.github.com/mhucka/44921ea1e9a01697dbd0591d872b7b22.

[54] H. Wickham. *R Packages*. O'Reilly Media, Sebastopol, CA, first edition, 2015.

[55] R. Wilson. Encouraging citation of software – introducing CITATION files. http://www.software.ac.uk/blog/2013-09-02-encouraging-citation-software-introducing-citation-files. Accessed: 2016-02-17.

[56] WSSSPE Workshops. http://wssspe.researchcomputing.org.uk/. Accessed: 2016-03-16.