

A peer-reviewed version of this preprint was published in PeerJ on 19 September 2016.

[View the peer-reviewed version](https://peerj.com/articles/cs-86) (peerj.com/articles/cs-86), which is the preferred citable publication unless you specifically need to cite this preprint.

Smith AM, Katz DS, Niemeyer KE, FORCE11 Software Citation Working Group. 2016. Software citation principles. PeerJ Computer Science 2:e86 <https://doi.org/10.7717/peerj-cs.86>

SOFTWARE CITATION PRINCIPLES

ARFON M. SMITH¹, DANIEL S. KATZ², KYLE E. NIEMEYER³, 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 focused 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 main contribution of this document are the software citation principles, written fairly concisely in this section and discussed further later in the document (§6). In addition, we also motivate the creation of these principles (§2), describe the process by which they were created (§3), summarize use cases related to software citation (§4), and review related work (§5). We also lay out the necessary future work (§7).

- (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.

Corresponding author: Daniel S. Katz², d.katz@ieee.org.

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

² National Center for Supercomputing Applications (NCSA) & Electrical and Computer Engineering (ECE) Department & School of Information Sciences (iSchool), University of Illinois at Urbana–Champaign, Urbana, IL, USA.

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

- 25 (5) **Accessibility**: Software citations should facilitate access to the software itself and to its as-
 26 sociated metadata, documentation, data, and other materials necessary for both humans and
 27 machines to make informed use of the referenced software.
- 28 (6) **Specificity**: Software citations should facilitate identification of, and access to, the specific
 29 version of software that was used. Software identification should be as specific as necessary,
 30 such as using version numbers, revision numbers, or variants such as platforms.

31

2. MOTIVATION

32 As the process of research¹ has become increasingly digital, research outputs and products have
 33 grown beyond simply papers and books to include software, data, and other electronic components
 34 such as presentation slides, posters, (interactive) graphs, maps, websites (e.g., blogs and forums), and
 35 multimedia (e.g., audio and video lectures). Research knowledge is embedded in these components.
 36 And papers and books themselves are also becoming increasingly digital, allowing them to become
 37 executable and reproducible. As we move towards this future where research is performed in and
 38 recorded as a variety of linked digital products, the characteristics and properties that developed
 39 for books and papers need to be applied to all digital products and possibly adjusted. Here, we are
 40 concerned specifically with the citation of software products. The challenge is not just the textual
 41 citation of software in a paper, but the more general identification of software used within the
 42 research process. This work focuses on making software a citable entity in the scholarly ecosystem.
 43 While software products represent a small fraction of the sum total of research output, this work
 44 together with other efforts such as the FORCE11 Data Citation Principles [12, 55] collectively
 45 represent an effort to better describe (and cite) all outputs of research.

46 Software and other digital resources currently appear in publications in very inconsistent ways.
 47 For example, a random sample of 90 articles in the biology literature found seven different ways that
 48 software was mentioned, including simple names in the full-text, URLs in footnotes, and different
 49 kinds of mentions in references lists: project names or websites, user manuals, publications that
 50 describe or introduce the software [27]. Table 1 shows examples of these varied forms of software
 51 mentions and the frequency with which they were encountered. Many of these kinds of mentions
 52 fail to perform the functions needed of citations, and their very diversity and frequent informality
 53 undermines the integration of software work into bibliometrics and other analyses. Studies on data
 54 and facility citation have shown similar results [28, 42, 47].

TABLE 1. Varieties of software mentions in publications, from Howison and Bullard [27].

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%

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

- 57 • **Understanding Research Fields**: Software is a product of research, and by not citing it, we leave
 58 holes in the record of research of progress in those fields.

¹We 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.

- 59 • Credit: Academic researchers at all levels, including students, postdocs, faculty, and staff,
60 should be credited for the software products they develop and contribute to, particularly when
61 those products enable or further research done by others.² Non-academic researchers should
62 also be credited for their software work, though the specific forms of credit are different than
63 for academic researchers.
- 64 • Discovering Software: Citations enable the specific software used in a research product to be
65 found. Additional researchers can then use the same software for different purposes, leading to
66 credit for those responsible for the software.
- 67 • Reproducibility: Citation of specific software used is necessary for reproducibility, but is not
68 sufficient. Additional information such as configurations and platform issues are also needed.

69 3. PROCESS OF CREATING PRINCIPLES

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

72 *The software citation working group is a cross-team committee leveraging the perspectives*
73 *from a variety of existing initiatives working on software citation to produce a consolidated*
74 *set of citation principles in order to encourage broad adoption of a consistent policy for*
75 *software citation across disciplines and venues. The working group will review existing*
76 *efforts and make a set of recommendations. These recommendations will be put of for*
77 *endorsement by the organizations represented by this group and others that play an*
78 *important role in the community.*

79 *The group will produce a set of principles, illustrated with working examples, and*
80 *a plan for dissemination and distribution. This group will not be producing detailed*
81 *specifications for implementation although it may review and discuss possible technical*
82 *solutions.*

83 The group gathered members (see Appendix A) in April and May 2015, and then began work in
84 June. This materialized as a number of meetings and offline work by group members to document
85 existing practices in member disciplines; gather materials from workshops and other reports; review
86 those materials, identifying overlaps and differences; create a list of use cases related to software
87 citation, recorded in Appendix B; and subsequently draft an initial version of this document. The
88 draft Software Citation Principles document was discussed in a day-long workshop and presented at
89 the FORCE2016 Conference [20] in April 2016. Members of the workshop and greater FORCE11
90 community gave feedback, which we recorded here in Appendix C. This discussion led to some
91 changes in the use cases and discussion, although the principles themselves were not modified. We
92 also plan to initiate a follow-on working group that will work with stakeholders to ensure that these
93 principles impact the research process.

94 The process of creating the software citation principles began by adapting the FORCE11 Data
95 Citation Principles [12]. These were then modified based on discussions of the FORCE11 Software
96 Citation Working Group (see Appendix A for members), information from the use cases in §4, and
97 the related work in §5.

98 We made the adaptations because software, while similar to data in terms of not traditionally
99 having been cited in publications, is also different than data. In the context of research (e.g., in
100 science), the term “data” usually refers to electronic records of observations made in the course of
101 a research study (“raw data”) or to information derived from such observations by some form of
102 processing (“processed data”), as well as the output of simulation or modeling software (“simulated

²Providing recognition of software can have tremendous economic impact as demonstrated by the role of Text REtrieval Conference (TREC) in information retrieval [48].

103 data”). Some confusion about the distinction between software and data comes in part from the
104 much wider scope of the term “data” in computing and information science, where it refers to
105 anything that can be processed by a computer. In that sense, software is just a special kind of data.
106 Because of this, citing software is not the same as citing data. A more general discussion about
107 these distinctions is currently underway [37].

108 The principles in this document should guide further development of software citation mecha-
109 nisms and systems, and the reader should be able to look at any particular example of software
110 citation and see if it meets the principles. While we strive to offer practical guidelines that acknowl-
111 edge the current incentive system of academic citation, a more modern system of assigning credit is
112 sorely needed. It is not that academic software needs a separate credit system from that of academic
113 papers, but that the need for credit for research software underscores the need to overhaul the system
114 of credit for all research products. One possible solution for a more complete description of the
115 citations and associated credit is the transitive credit proposed by Katz and Smith [33, 38].

116 In the next section (§4), we provide some detailed context in which software citation is important,
117 by means of use cases. In §5, we summarize and analyze a large amount of previous work and
118 thinking in this area. In §6, we discuss issues related to the principles stated in §1, and finally, in
119 §7 we discuss the work needed to lead to these software citation principles being applied.

120 4. USE CASES

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

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

149 All use cases assume the existence of a citable software object, typically created by the authors/
 150 developers of the software. Developers can achieve this by, e.g., uploading a software release to
 151 figshare [15] or Zenodo [24] to obtain a DOI. Necessary metadata should then be included in a
 152 CITATION file [62] or machine-readable CITATION.jsonld file [38]. When software is not freely
 153 available (e.g., commercial software) or when there is no clear identifier to use, alternative means
 154 may be used to create citable objects as discussed in §6.9.

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 plus signs (+) indicate that the use case would benefit from that metadata if available.

Use case	Basic requirements										Example stakeholder(s)	
	Unique identifier	Software name	Author(s)	Contributor role	Version number	Release date	Location/repository	Indexed citations	Software license	Description		Keywords
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. Find software to implement task	●	●	●				●	●	+	+	+	Researcher, software engineer
8. Publish software paper	●	●	●		●	●	●					Publisher
9. Publish papers that cite software	●	●	●		●	●	●	●				Publisher
10. Build catalog of software	●	●	●		●	●	●	●	+	+	+	Indexer
11. Build software catalog/registry	●	●	●				●		+	+		Domain group, library, archive
12. Show scientific impact of holdings	●	●						●				Repository
13. Show how funded software has been used	●	●						●				Funder, policy maker
14. Evaluate contributions of researcher	●	●	+		●			●				Evaluator, funder
15. Store software entry	●	●	●		●	●	●		+	+		Citation manager
16. Publish mixed data/software packages	●	●	●		●	●	●		+	+	+	Repository, library, archive

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

158

5. RELATED WORK

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

164 **5.1. General community/non domain-specific activities.** Some of the most actionable work has
 165 come from the UK Software Sustainability Institute (SSI) in the form of blog posts written by their
 166 community fellows. For example, in a blog post from 2012, Jackson discusses some of the pitfalls
 167 of trying to cite software in publications [31]. He includes useful guidance for when to consider
 168 citing software as well as some ways to help “convince” journal editors to allow the inclusion of
 169 software citations.

170 Wilson suggests that software authors include a CITATION file that documents exactly how the
171 authors of the software would like to be cited by others [62]. While this is not a formal metadata
172 specification (e.g., it is not machine readable) this does offer a solution for authors wishing to give
173 explicit instructions to potential citing authors and as noted in the motivation section (§2), there is
174 evidence that authors follow instructions if they exist [28].

175 In a later post on the SSI blog, Jackson gives a good overview of some of the approaches package
176 authors have taken to automate the generation of citation entities such as BibTeX entries [32], and
177 Knepley et al. do similarly [39].

178 While not usually expressed as software citation principles, a number of groups have developed
179 community guidelines around software and data citation. Van de Sompel et al. [57] argue for
180 registration of all units of scholarly communication, including software. In “Publish or be damned?
181 An alternative impact manifesto for research software” [9], Chue Hong lists nine principles as part
182 of “The Research Software Impact Manifesto.” In the “Science Code Manifesto” [5], the founding
183 signatories cite five core principles (Code, Copyright, Citation, Credit, Curation) for scientific
184 software.

185 Perhaps in recognition of the broad range of research domains struggling with the challenge
186 of better recognizing the role of software, funders and agencies in both the US (e.g., NSF, NIH,
187 Alfred P. Sloan Foundation) and UK (e.g., SFTC, JISC, Wellcome Trust) have sponsored or hosted
188 a number of workshops with participants from across a range of disciplines, specifically aimed at
189 discussing issues around software citation [56, 2, 53, 45, 49, 3]. In many cases these workshops
190 produced strong recommendations for their respective communities on how best to proceed. In
191 addition, a number of common themes arose in these workshops, including (1) the critical need
192 for making software more “citable” (and therefore actions authors and publishers should take to
193 improve the status quo), (2) how to better measure the impact of software (and therefore attract
194 appropriate funding), and (3) how to properly archive software (where, how, and how often) and
195 how this affects what to cite and when.

196 Most notable of the community efforts are those of WSSSPE [63] and SSI [54], who between them
197 have run a series of workshops aimed at gathering together community members with an interest
198 in (1) defining the set of problems related to the role of software and associated people in research
199 settings, particularly academia, (2) discussing potential solutions to those problems, (3) beginning to
200 work on implementing some of those solutions. In each of the three years that WSSSPE workshops
201 have run thus far, the participants have produced a report [34, 35, 36] documenting the topics
202 covered. Section 5.8 and Appendix J in the WSSSPE3 report [36] has some preliminary work
203 and discussion particularly relevant to this working group. In addition, a number of academic
204 publishers such as APA [43] have recommendations for submitting authors on how to cite software,
205 and journals such as *F1000Research* [14], *SoftwareX* [52], *Open Research Computation* [46], and
206 the *Journal of Open Research Software* allow for submissions entirely focused on research software.

207 **5.2. Domain-specific community activities.** One approach to increasing software “citability” is
208 to encourage the submission of papers in standard journals describing a piece of research software,
209 often known as software papers (see §6.2). While some journals (e.g., Transactions on Mathematical
210 Software (TOMS), Bioinformatics, Computer Physics Communications, F1000Research, Seismo-
211 logical Research Letters, Electronic Seismologist) have traditionally accepted software submissions,
212 the American Astronomical Society (AAS) has recently announced they will accept software pa-
213 pers in their journals [1]. Professional societies are in a good position to change their respective
214 communities, as the publishers of journals and conveners of domain-specific conferences; as pub-
215 lishers they can change editorial policies (as AAS has done) and conferences are an opportunity to
216 communicate and discuss these changes with their communities.

217 In astronomy and astrophysics: The Astronomy Source Code Library (ASCL) [4], is a website
218 dedicated to the curation and indexing of software used in the astronomy-based literature. In
219 2015, the AAS and GitHub co-hosted a workshop [45] dedicated to software citation, indexing,
220 and discoverability in astrophysics. More recently, a Birds of a Feather session was held at the
221 Astronomical Data Analysis Software and Systems (ADASS) XXV conference [3] that included
222 discussion of software citation.

223 In the life sciences: In May 2014, the NIH held a workshop aimed at helping the biomedical
224 community discover, cite, and reuse software written by their peers. The primary outcome of this
225 workshop was the Software Discovery Index Meeting Report [59] which was shared with the com-
226 munity for public comment and feedback. The authors of the report discuss what framework would
227 be required for supporting a Software Discovery Index including the need for unique identifiers,
228 how citations to these would be handled by publishers, and the critical need for metadata to describe
229 software packages.

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

235 **5.3. Existing efforts around metadata standards.** Producing detailed specifications and recom-
236 mendations for possible metadata standards to support software citation was not within the scope
237 of this working group. However some discussion on the topic did occur and there was significant
238 interest in the wider community to produce standards for describing research software metadata.

239 Content specifications for software metadata vary across communities, and include DOAP [13],
240 an early metadata term set used by the Open Source Community, as well as more recent commu-
241 nity efforts like Research Objects [6], The Software Ontology [41], EDAM Ontology [29], Project
242 CRediT [11], the OpenRIF Contribution Role Ontology [25], Ontosoft [23], RRR/JISC guide-
243 lines [22], or the terms and classes defined at Schema.org related to the `SoftwareApplication`
244 class. In addition, language-specific software metadata schemes are in widespread use, including
245 the Debian package format [30], Python package descriptions [58], and R package descriptions [60],
246 but these are typically conceived for software build, packaging, and distribution rather than citation.
247 CodeMeta [8] has created a crosswalk among these software metadata schemes and an exchange
248 format that allows software repositories to effectively interoperate.

249

6. DISCUSSION

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

251 **6.1. What software to cite.** The software citation principles do not define what software should
252 be cited, but rather, how software should be cited. What software should be cited is the decision
253 of the author(s) of the research work in the context of community norms and practices, and in
254 most research communities, these are currently in flux. In general, *we believe that software*
255 *should be cited on the same basis as any other research product such as a paper or book*; that is,
256 authors should cite the appropriate set of software products just as they cite the appropriate set of
257 papers, perhaps following the FORCE11 Data Citation Working Group principles, which state, “In
258 scholarly literature, whenever and wherever a claim relies upon data, the corresponding data should
259 be cited.” [12]

260 Some software which is, or could be, captured as part of data provenance may not be cited.
261 Citation is partly a record of software important to a research outcome³, where provenance is a
262 record of all steps (including software) used to generated particular data within the research process.
263 Research results, including data, increasingly depend on software [26], and thus may depend on
264 the specific version used [50, 61]. Furthermore, errors in software or environment variations can
265 affect results [44, 51]. This implies that for a data research product, provenance data will include
266 some of the cited software. Similarly, the software metadata recorded as part of data provenance
267 will overlap the metadata recorded as part of software citation for the software that was used in the
268 work. The data recorded for reproducibility should also overlap the metadata recorded as part of
269 software citation. In general, we intend the software citation principles to cover the minimum of
270 what is necessary for software citation for the purpose of software identification. Some use cases
271 related to citation (e.g., provenance, reproducibility) might have additional requirements beyond the
272 basic metadata needed for citation, as Table 2 shows.

273 **6.2. Software papers.** Currently, and for the foreseeable future, software papers are being pub-
274 lished and cited, in addition to software itself being published and cited, as many community norms
275 and practices are oriented towards citation of papers. As discussed in the Importance principle (1)
276 and the discussion above, *the software itself should be cited on the same basis as any other research*
277 *product; authors should cite the appropriate set of software products.* If a software paper exists and
278 it contains results (performance, validation, etc.) that are important to the work, then the software
279 paper should also be cited. We believe that a request from the software authors to cite a paper
280 should typically be respected, and the paper cited *in addition to* the software.

281 **6.3. Derived software.** The goals of software citation include the linked ideas of crediting those
282 responsible for software and understanding the dependencies of research products on specific
283 software. In the Importance principle (1), we state that “software should be cited on the same basis
284 as any other research product such as a paper or a book; that is, authors should cite the appropriate
285 set of software products just as they cite the appropriate set of papers.” In the case of one code that is
286 derived from another code, citing the derived software may appear to not credit those responsible for
287 the original software, nor recognize its role in the work that used the derived software. However, this
288 is really analogous to how any research builds on other research, where each research product just
289 cites those products that it directly builds on, not those that it indirectly builds on. Understanding
290 these chains of knowledge and credit have been part of the history of science field for some time,
291 though more recent work is suggesting more nuanced evaluation of the credit chains [11, 38].

292 **6.4. Software peer review.** Adherence to the software citation principles enables better peer
293 reviews through improved reproducibility. However, since the primary goal of software citation is
294 to identify the software that has been used in a scholarly product, the peer review of software itself
295 is mostly out of scope in the context of software citation principles. For instance, when identifying
296 a particular software artifact that has been used in a scholarly product, whether or not that software
297 has been peer-reviewed is irrelevant. One possible exception would be if the peer-review status of
298 the software should be part of the metadata, but the working group does not believe this to be part
299 of the minimal metadata needed to identify the software.

300 **6.5. Citation format in reference list.** Citations in references in the scholarly literature are for-
301 matted according to the citation style (e.g., AMS, APA, Chicago, MLA) used by that publication.
302 (Examples illustrating these styles have been published by Lipson [40]; the follow-on Software
303 Citation Implementation Group will provide suggested examples.) As these citations are typically

³Citation can be used for many purposes, including for software: which software has been used in the work, which software has influenced the work, which software is the work superseding, which software is the work disproving, etc.

304 sent to publishers as text formatted in that citation style, not as structured metadata, and because
305 the citation style dictates how the human reader sees the software citation, *we recommend that all*
306 *text citation styles support the following: a) a label indicating that this is software, e.g., [Software],*
307 *potentially with more information such as [Software: Source Code], [Software: Executable], or*
308 *[Software: Container], and b) support for version information, e.g., Version 1.8.7.*

309 **6.6. Citations limits.** This set of software citation principles, if followed, will cause the number of
310 software citations in scholarly products to increase, thus causing the number of overall citations to
311 increase. Some scholarly products, such as journal articles, may have strict limits on the number of
312 citations they permit, or page limits that include reference sections. Such limits are counter to our
313 recommendation, and *we recommend that publishers using strict limits for the number of citations*
314 *add specific instructions regarding software citations to their author guidelines to not disincentivize*
315 *software citation. Similarly, publishers should not include references in the content counted against*
316 *page limits.*

317 **6.7. Unique identification.** The Unique Identification principle (3) calls for “a method for identifi-
318 cation that is machine actionable, globally unique, interoperable, and recognized by a community.”
319 What this means for data is discussed in detail in the “Unique Identification” section of a report by
320 the FORCE11 Data Citation Implementation Group (DCIG) [55], which calls for “unique identifica-
321 tion in a manner that is machine-resolvable on the Web and demonstrates a long-term commitment
322 to persistence.” This report also lists examples of identifiers that match these criteria including
323 DOIs, PURLs, Handles, ARKS, and NBNs. For software, *we recommend the use of DOIs as the*
324 *unique identifier due to their common usage and acceptance, particularly as they are the standard*
325 *for other digital products such as publications.*

326 While we believe there is value in including the explicit version (e.g., Git SHA1 hash, Subversion
327 revision number) of the software in any software citation, there are a number of reasons that a
328 commit reference together with a repository URL is not recommended for the purposes of software
329 citation:

- 330 (1) Version numbers/commit references are not guaranteed to be permanent. Projects can be
331 migrated to new version control systems (e.g., SVN to Git). In addition, it is possible to
332 overwrite/clobber a particular version (e.g., force-pushing in the case of Git).
- 333 (2) A repository address and version number does not guarantee that the software is available at
334 a particular (resolvable) URL, especially as it is possible for authors to remove their content
335 from, e.g., GitHub.
- 336 (3) A particular version number/commit reference may not represent a “preferred” point at which
337 to cite the software from the perspective of the package authors.

338 We recognize that there are certain situations where it may not be possible for follow the
339 recommended best-practice. For example, if (1) the software authors did not register a DOI
340 and/or release a specific version, or (2) the version of the software used does not match what is
341 available to cite. In those cases, falling back on a combination of the repository URL and version
342 number/commit hash would be an appropriate way to cite the software used.

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

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

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

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

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

364 **6.8. Types of software.** The principles and discussion in this document have generally been written
365 to focus on software as source code. However, we recognize that some software is only available as
366 an executable, a container, or a virtual machine image, while other software may be available as a
367 service. We believe the principles apply to all of these forms of software, though the implementation
368 of them will certainly differ based on software type. *When software is accessible as both source
369 code and another type, we recommend that the source code be cited.*

370 **6.9. Access to software.** The Accessibility principle (5) states that “software citations should
371 permit and facilitate access to the software itself.” This does not mean that the software must be
372 freely available. Rather, the metadata should provide enough information that the software can be
373 accessed. If the software is free, the metadata will likely provide an identifier that can be resolved
374 to a URL pointing to the specific version of the software being cited. For commercial software, the
375 metadata should still provide information on how to access the specific software, but this may be a
376 company’s product number or a link to a web site that allows the software be purchased. As stated
377 in the Persistence principle (4), we recognize that the software version may no longer be available,
378 but it still should be cited along with information about how it was accessed.

379 **6.10. What an identifier should resolve to.** While citing an identifier that points to, e.g., a GitHub
380 repository can satisfy the principles of Unique Identification (3), Accessibility (5), and Specificity
381 (6), such a repository cannot guarantee Persistence (4). *Therefore, we recommend that the software
382 identifier should resolve to a persistent landing page that contains metadata and a link to the
383 software itself, rather than directly to the source code files, repository, or executable.* This ensures
384 longevity of the software metadata—even perhaps beyond the lifespan of the software they describe.
385 This is currently offered by services such as figshare [15] and Zenodo [24], which both generate
386 persistent DataCite DOIs for submitted software. In addition, such landing pages can contain both
387 human-readable metadata (e.g., the types shown by Table 2) as well as content-negotiable formats
388 such as RDF or DOAP [13].

389 **6.11. Updates to these principles.** As this set of software citation principles has been created
390 by the FORCE11 Software Citation Working Group, which will cease work and dissolve after
391 publication of these principles, any updates will require a different FORCE11 working group to
392 make them. As mentioned in §7, we expect a follow-on working group to be established to
393 promote the implementation of these principles, and it is possible that this group might find items
394 that need correction or addition in these principles. *We recommend that this Software Citation
395 Implementation Working Group be charged, in part, with updating these principles during its*

396 *lifetime, and that FORCE11 should listen to community requests for later updates and respond by*
397 *creating a new working group.*

398 7. FUTURE WORK

399 Software citation principles without clear worked-through examples are of limited value to
400 potential implementers, and so in addition to this principles document, the final deliverable of this
401 working group will be an implementation paper outlining working examples for each of the use
402 cases listed in §4.

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

408 ACKNOWLEDGEMENTS

409 While D. S. Katz prepared this material while employed at the NSF, any opinion, finding, and
410 conclusions or recommendations expressed in this material are those of the authors and do not
411 necessarily reflect the views of the NSF.

412 APPENDIX A. WORKING GROUP MEMBERSHIP

413 Alberto Accomazzi, Harvard-Smithsonian CfA
414 Alice Allen, Astrophysics Source Code Library
415 Micah Altman, MIT
416 Jay Jay Billings, Oak Ridge National Laboratory
417 Carl Boettiger, University of California, Berkeley
418 Jed Brown, University of Colorado Boulder
419 Sou-Cheng T. Choi, NORC at the University of Chicago & Illinois Institute of Technology
420 Neil Chue Hong, Software Sustainability Institute
421 Tom Crick, Cardiff Metropolitan University
422 Mercè Crosas, IQSS, Harvard University
423 Scott Edmunds, GigaScience, BGI Hong Kong
424 Christopher Erdmann, Harvard-Smithsonian CfA
425 Martin Fenner, DataCite
426 Darel Finkbeiner, OSTI
427 Ian Gent, University of St Andrews, recomputation.org
428 Carole Goble, The University of Manchester, Software Sustainability Institute
429 Paul Groth, Elsevier Labs
430 Melissa Haendel, Oregon Health and Science University
431 Stephanie Hagstrom, FORCE11
432 Robert Hanisch, National Institute of Standards and Technology, One Degree Imager
433 Edwin Henneken, Harvard-Smithsonian CfA
434 Ivan Herman, World Wide Web Consortium (W3C)
435 James Howison, University of Texas
436 Lorraine Hwang, University of California, Davis
437 Thomas Ingraham, F1000Research
438 Matthew B. Jones, NCEAS, University of California, Santa Barbara
439 Catherine Jones, Science and Technology Facilities Council
440 Daniel S. Katz, University of Illinois (co-chair)

441 Alexander Konovalov, University of St Andrews
442 John Kratz, California Digital Library
443 Jennifer Lin, Public Library of Science
444 Frank Löffler, Louisiana State University
445 Brian Matthews, Science and Technology Facilities Council
446 Abigail Cabunoc Mayes, Mozilla Science Lab
447 Daniel Mietchen, National Institutes of Health
448 Bill Mills, TRIUMF
449 Evan Misshula, CUNY Graduate Center
450 August Muench, American Astronomical Society
451 Fiona Murphy, Independent Researcher
452 Lars Holm Nielsen, CERN
453 Kyle E. Niemeyer, Oregon State University (co-chair)
454 Karthik Ram, University of California, Berkeley
455 Fernando Rios, Johns Hopkins University
456 Ashley Sands, University of California, Los Angeles
457 Soren Scott, Independent Researcher
458 Frank J. Seinstra, Netherlands eScience Center
459 Arfon Smith, GitHub (co-chair)
460 Kaitlin Thaney, Mozilla Science Lab
461 Ilian Todorov, Science and Technology Facilities Council
462 Matt Turk, University of Illinois
463 Miguel de Val-Borro, Princeton University
464 Daan Van Hauwermeiren, Ghent University
465 Stijn Van Hoey, Ghent University
466 Belinda Weaver, The University of Queensland
467 Nic Weber, University of Washington iSchool

468 APPENDIX B. SOFTWARE CITATION USE CASES

469 This appendix records an edited, extended description of the use cases discussed in §4, originally
470 found in [19]. This discussion is not fully complete, and in some cases, it may not be fully self-
471 consistent, but it is part of this paper as a record of one of the inputs to the principles. We expect
472 that the follow-on Software Citation Implementation Group will further develop these use cases,
473 including explaining in more detail how the software citation principles can be applied to each as
474 part of working with the stakeholders to persuade them to actually implement the principles in their
475 standard workflows.

476 **B.1. Researcher who uses someone else's software for a paper.** One of the most common use
477 cases may be researchers who use someone else's software and want to cite it in a technical paper.
478 This will be similar to existing practices for citing research artifacts in papers.

479 "Requirements" for researcher:

- 480 • Name of software
- 481 • Names of software authors/contributors
- 482 • Software version number and release date, or download date
- 483 • Location/repository, or contact name/email (if not publicly available)
- 484 • Citable DOI of software
- 485 • Format for citing software in text and in bibliography

486 Possible steps:

- 487 (1) Software developers create CITATION file and associate with source code release/repository.
488 (2) Researcher finds and uses software for research paper.
489 (3) Researcher identifies citation metadata file (e.g., “CITATION” file) associated with down-
490 loaded/installed software source code or in online repository/published location. CITATION
491 file includes necessary citation metadata. CITATION file may include BibTeX entry, suggested
492 citation format
493 (4) Researcher cites software appropriately, e.g., in methodology section; reference included in
494 bibliography.

495 **B.2. Researcher who uses someone else’s software for new software.** In this case, a researcher
496 develops new software that incorporates or depends on existing software. In order to credit the
497 developer(s), the researcher will include citations in his/her source code, documentation, or other
498 metadata in a similar manner to papers

499 Requirements for researcher:

- 500 • Name of software
- 501 • Names of software authors/contributors
- 502 • Software version number and release date
- 503 • Location/repository
- 504 • Citable DOI of software
- 505 • Format for citing software in source code, documentation, or citation metadata file

506 Possible steps:

- 507 (1) Assume that software developers have created a CITATION file and associated with the source
508 code release/repository.
509 (2) Researcher finds and uses software in the development of new software.
510 (3) Researcher identifies citation metadata file (e.g., “CITATION” file) associated with down-
511 loaded/installed software source code or in online repository/published location. CITATION
512 file includes necessary citation metadata. CITATION file may include BibTeX entry, suggested
513 citation format.
514 (4) Researcher cites software in source code, documentation, or other metadata-containing file.

515 **B.3. Researcher who contributes to someone else’s software (open source project).** A re-
516 searcher wants to contribute to someone else’s software in the manner in which their contributions
517 will be accepted and recognized.

518 Possible steps:

- 519 (1) Researcher finds information about the software, and how contributors will be recognized
520 (2) Researcher possibly submit a Contributor License Agreement (CLA) or Copyright Assignment
521 Agreement (CAA) to allow the contributed content to be distributed with the software being
522 contributed to
523 (3) Researcher contributes to the software
524 (4) Software maintainers accept contribution, recognize researcher’s contribution, and update the
525 software metadata as appropriate

526 **B.4. Researcher who wants to know who uses the researcher’s software.** This case is similar to
527 a researcher who wants to find other papers/publications that cite a particular paper. A researcher
528 wants to gauge the usage of her software within or across communities and measure its impact on
529 research for both credit and funding.

530 Requirements:

- 531 • Uniquely identify software
- 532 • Indexed citations of software

- 533 • Indexed papers that use software

534 Steps:

- 535 (1) Researcher finds software official name or unique DOI in metadata associated with down-
536 loaded/installed source code or in online repository/published location.
537 (2) Researcher searches for software, may use online indexer (e.g., Scopus, Web of Science, Google
538 Scholar) using software name or DOI.
539 (3) Online indexer presents entry for software with list of citations, if any. Ideally, entry will also
540 include metadata contained in software CITATION file and citation example.

541 **B.5. Researcher gets credit for software development at the academic/governmental institu-**
542 **tion, in professional career, etc.** This case describes the need for a researcher who has contributed
543 to software (by design, software engineering, development, testing, patching, documentation, train-
544 ing, evangelizing, etc.) to have their software work recognized by their employer or colleagues for
545 the purpose of career advancement and increased professional reputation.

546 Requirements for researcher:

- 547 • Name of software
548 • Names of software authors/contributors
549 • Location/repository
550 • Citable DOI of software
551 • Format for citing software in an official CV, in a departmental/institutional review report, etc.
552 • Role in the software creation, that is linked to version or component
553 • Role in contributing to the software as a “package” (not just lines of code) development of
554 benchmarks, testing, documentation, tutorials etc.

555 **B.6. Researcher who wants to “reproduce” another person/group’s analysis.** When a re-
556 searcher wants to understand or verify a research results from another researcher, they would
557 like to use the same software. Note that accessing the exact same software is necessary but not
558 sufficient for reproducibility.

559 Requirements for researcher:

- 560 • Name of software
561 • Location/repository for the exact release that was used
562 • DOI or other persistent handle for that specific release
563 • Release has all components necessary for reproducing the work (Note: this ideally also means
564 sample inputs and outputs)

565 **B.7. Researcher who wants to find a piece of software to implement a task.** This is the case
566 where a research is looking for software to use but wants to understand whether it is being used in
567 a scholarly fashion. For example, a researcher searches through a software repository and finds a
568 package that might be useful. They look to find whether it has been used by others in the scientific
569 literature.

570 Requirements

- 571 • Either the software documentation page has a reference to existing literature that makes use of
572 it.
573 • There is a mechanism to look it up.

574 **B.8. Publisher wants to publish a software paper.** This case asks what information regarding
575 software is needed for a publisher who wants to publish a paper describing that software.

576 Requirements

- 577 • Name of software

- 578 • Names of software authors/contributors
- 579 • Location/repository
- 580 • Citable DOI of software
- 581 • Format for citing software in JATS, for example, as well as references in the text itself

582 **B.9. Publisher who wants to publish papers that cite software.** This case asks what information
583 regarding software is needed for a publisher who wants to publish papers that cite that software.

584 Requirements for publisher:

- 585 • Name of software
- 586 • Names of software authors/contributors
- 587 • Location/repository
- 588 • Citable DOI of software
- 589 • Format for citing software in, e.g., JATS, as well as references in the text itself

590 **B.10. Indexer (e.g., Scopus, WoS, Scholar, MS Academic Search) who wants to build a catalog
591 of software.** Provide an index over the software that is used within the research domain. Track how
592 that software is being used by different groups of researchers and to what ends.

593 Requirements:

- 594 • Uniquely identify pieces of software used by the research literature
- 595 • Connect authors and organizations to that software
- 596 • Connect various software versions together

597 **B.11. Domain group (e.g., ASCL, bioCADDIE), Libraries, and Archives (e.g., University
598 library, laboratory archive, etc.) wants to build a catalog/registry of institutional or domain
599 software.** There are two different examples here: One is building a catalog/archive of software
600 produced by those affiliated with the institution. The other is along the lines of Sayeed Choudhury's
601 note that "data are the new special collections." An institution may choose to build a catalog/archive
602 of many things within a single topic or subject in order to secure all the software on a certain topic
603 or build a collection that may draw users to their establishment, much like special collections now
604 do for university libraries and archives.

605 **B.12. Repository showing scientific impact of holdings.** A repository that archives and/or main-
606 tains a collection of software. The repository would like to address usage and impact of software
607 in its holding. Usage would aid potential users whether the software is being actively maintained
608 or developed or has been superseded. Both would help repository know how to direct resources,
609 e.g., maintenance, training etc. This is similar to the case of a funder wanting to know the impact
610 of funded work.

611 Requirements:

- 612 • Code name, or a unique identifier
- 613 • Relationships to previous versions
- 614 • Connect to repository
- 615 • Connect to research

616 **B.13. Funder who wants to know how software they funded has been used.** This use case is
617 similar to "Repository showing scientific impact of holdings", where a funder wants to find out the
618 use and impact and software that they supported. It is also similar to "Researcher who wants to
619 know who uses the researcher's software."

620 **B.14. Evaluator or funder wants to evaluate contributions of a researcher.** In this use case, an
621 evaluator (e.g., academic administrator) or funder wants to evaluate the contributions of a researcher
622 who develops software. This case is related to those where researchers want to get credit for software
623 development, or where organizations want to evaluate the impact of software itself.

624 **B.15. Reference management system used by researchers to author a manuscript.** Reference
625 management systems may need to be updated to internally understand that there is a software
626 reference type, and to be able to output references to software in common formats.

627 Requirements for reference manager:

- 628 • Names of software authors/contributors
- 629 • Software version number and release date
- 630 • Location/repository
- 631 • Citable DOI of software or paper recommended for citation
- 632 • Format for citing software in citation metadata file
- 633 • Citation metadata tags embedded in DOI landing page/software project page for easy ingest

634 Possible steps:

- 635 (1) Reference management system such as EndNote, Mendeley, Zotero, etc. builds affordances for
636 software references.
- 637 (2) Researcher finds software citation and adds it to their reference manager library, by (a) importing
638 from the CITATION file (e.g., BibTeX, RIS), or (b) clicking on, e.g., an “add to Zotero library”
639 widget in web browser.
- 640 (3) Researcher writes a paper and uses the reference manager to generate citations or bibliography.

641 **B.16. Repository wants to publish mixed data/software packages.** Domain and institutional data
642 repositories have both data and software artifacts, and want to link these together in a provenance
643 trace that can be cited. Sometimes the software is a separately identified artifact, but at other times
644 software is included inside of data packages, and the researcher wants to cite the combined product.

645 **Use cases not adopted in the table:**

646 **Researcher who benchmarks someone else’s software with or without modification on one or
647 many hardware platforms for publication.** This case describes the need for a researcher who
648 has contributed to software (by design, software engineering, development, testing, patching, doc-
649 umentation, training, evangelizing, etc.) to have their software work recognized by their employer
650 or colleagues for the purpose of career advancement and increased professional reputation.

651 Requirements for researcher:

- 652 • Name of software
- 653 • Names of software authors/contributors
- 654 • Software version number and release date
- 655 • Location/repository
- 656 • Citable DOI of software or paper recommended for citation
- 657 • Format for citing software in source code or citation metadata file

658 Possible steps:

- 659 (1) Software developers create CITATION file and associate with source code release/repository.
- 660 (2) Researcher finds and uses software in the development of new software.
- 661 (3) Researcher identifies citation metadata file (e.g., CITATION file) associated with down-
662 loaded/installed software source code or in online repository/published location. CITATION
663 file includes necessary citation metadata. CITATION file may include BibTeX entry, suggested
664 citation format.

665 (4) Researcher cites software in source code, documentation, or other metadata-containing file.

666 After review of this use case, we decided that based on the title this falls under use case 1, where
667 a researcher uses someone else's software for a paper. Unlike use case 1, which is general in terms
668 of the use of software, here the use leads to a benchmarking study—but the outcome in both cases
669 is a paper that needs to cite the software.

670 **Researcher who wants to publish about a piece of software.** The research wants to publish about
671 a version of software they have produced. A key part of this use case is to be able to connect the
672 given narrative to a specific version of the software in questions and connect that in large story.

673 Requirements:

- 674 • Name of software
- 675 • Names of software authors/contributors
- 676 • Location/repository
- 677 • Citable DOI of Software
- 678 • Links to older versions of software

679 This is similar to use case 1, other than the fact that the software developer(s) and paper author(s)
680 will likely be the same person/people here.

681 **Researcher wants to record the software that generated some data.** This is the case where a
682 researcher is using some software to perform an analysis, either of a physical sample or of data. The
683 researcher needs to know which version was used, for example in case a bug was fixed. Note that
684 knowing the software and its version is not sufficient to determine the “conditions” of the analysis,
685 but they are essential.

686 Requirement: The analysis, or the generated data, has information about the software used.

687 This is also similar to use case 1, except in that case the research output is a paper, while here the
688 output is a dataset.

689 **Researcher who wants to reproduce experience of use of a particular software implementation
690 in context.** Researcher is engaged in historical/cultural research, e.g., a study of video games as
691 cultural artifacts.

692 Requirements:

- 693 • Name of software
- 694 • Software version number
- 695 • Documentation of the execution environment/context
- 696 • Location/repository for virtual machine (or equivalent) comprising both software and execution
697 environment/context
- 698 • Persistent identifier associated with virtual machine instance (or equivalent) comprising both
699 software and execution environment/context

700 Possible steps:

- 701 (1) Researcher obtains persistent ID from citation
- 702 (2) Research uses a persistent ID resolution service to resolve ID to a location of an executable VM
703 instance in a repository
- 704 (3) Researcher obtains VM in the repository, executes it, and interacts with software

705 This overlaps use case 6 (reproducing analysis), and so we decided not to include this as a distinct
706 use case.

707

APPENDIX C. FEEDBACK FOLLOWING FORCE2016

708 This appendix contains a record of comments made by the FORCE11 community on the draft
709 Software Citation Principles, either directly via Hypothesis on the draft document⁴ posted following
710 the FORCE2016 conference [20] or via GitHub issues⁵, and the responses to these comments.

711 **C.1. On unique identification:** I know this suggestion of a single unique identifier comes from
712 the DOI perspective where it works pretty well, but I'm wondering if something different in the way
713 of identification should be used for software. For creative works generally there is the FRBR model
714 (https://en.wikipedia.org/wiki/Functional_Requirements_for_Bibliographic_Records) which de-
715 fines several levels for a creative entity - "work", "expression", "manifestation", and "item". I
716 think something along these lines are particularly relevant for software - it is useful to be able to
717 locate all uses of a particular piece of software no matter what version (the "work" level - software
718 identified by a particular name and purpose over a period of time), but it is also important to specify
719 the particular version used in any given work ("expression" - the source code at the time of use)
720 and in some cases also the platform ("manifestation" - the compiled bytes including libraries, for
721 example a docker image). "Item" probably isn't relevant for software. That is, I think a software
722 citation perhaps could use THREE distinct unique identifiers, one for the work itself, one for the
723 specific version (source code), and possibly an additional one for the actual downloadable binary
724 image that can be run. Rather than leave it implicit I think recognizing the different levels of citable
725 record would be helpful here. #F11SC

726 *Reply:* I interpret the requirement for "global uniqueness" as referring to the identifier itself. Two
727 different people can have the same name (not globally unique) but cannot share a single ORCID
728 (globally unique). Global uniqueness of the identifier does not preclude multiple identifiers pointing
729 to the same person. I think the suggestion of differentiating between different software expressions/
730 manifestations/items is a reasonable one, but I don't think it relaxes the requirement for identifiers
731 to be globally unique.

732 **Our response:** We agree that there are valid points here, but on balance we don't feel that the
733 rewards from implementing this outweigh the practical challenges.

734 **C.2. On accessibility:** Should this document address this in further detail? For example, "permit
735 and facilitate access" could be explored further. Should this be done through open access licensing?
736 repositories? Who's responsible for providing this access?

737 I am also wondering if this is a separate issue since "citing" traditionally pointed to publications
738 but did not necessarily address access. DOI, for example is stated, but doesn't guarantee "access",
739 so does this simply restating point 3, or should it provide something new?

740 **Our response:** We agree that accessibility should receive further attention, which the follow-on
741 group focusing on implementation will provide. However, this is out of scope for the document
742 outlining the principles.

743 To the second point, accessibility provides information about access, but does not guarantee
744 access itself (e.g., paywalled article).

745 **C.3. On specificity:** I am wondering if this should be folded into number 3 "Unique Identification".
746 Both seem to deal with the issue of identification and access.

747 **Our response:** A unique software identifier **can** point to the specific version/variant of software,
748 but it can also identify other things (collection of versions, repository, etc.), while this principle
749 deals with the need to identify the specific version of software used (via citation).

⁴<https://www.force11.org/software-citation-principles>

⁵<https://github.com/force11/force11-scwg/issues>

750 **C.4. On academic credit:** A lot of software that were developed by non-academic engineers also
751 contribute to academic research indirectly. Their names and contributions should also be credited.
752 So removing “Academic” makes more sense?

753 *Reply:* This is a good point, though I think academic and non-academic credit are different, so
754 perhaps we can add to this regarding non-academic credit, rather than removing “academic”.

755 *Reply:* I agree with Daniel on this. Keep Academic and add non-academic.

756 **Our response:** We’ve made the bullet more general, just about credit, discussing academic credit
757 and adding a sentence about non-academic credit as well.

758 **C.5. On citations in text:** Although the focus here is on citations in the references, as a publisher,
759 our experience is that most common practice of “citation” of data and software for authors is
760 typically in the main body of the text. In order to encourage software to be treated and valued
761 as a first-class research object, it is important that citations to it be positioned in the references as
762 citations to articles and books are. However, it would be a missed opportunity if we did not leverage
763 current practices of authors. This will also likely arise during implementation, as it has for the Data
764 Citation Implementation Publisher Early Adopters Pilot. This could be addressed in future work on
765 implementation.

766 **Our response:** In the principles, we propose that software should be cited in the references
767 list, to recognize the primary role of software in research. However, this practice is not mutually
768 exclusive with **also** referencing/citing software in the main body of a paper—as long as the software
769 is cited in the references.

770 **C.6. On unique identification:** Clearer instructions will be needed for authors on which version
771 to cite. For BioMed Central journals, we ask authors to cite two versions of the software, an
772 archived version (e.g., on Zenodo) as well as the current version (e.g., on GitHub). This is to ensure
773 accessibility. However, if repositories and archives were to include a persistent link to the current
774 version of the software, publishers could then instruct authors to cite only software with a UID,
775 which wouldn’t point to a current version, but would point to the version(s) used and would be a more
776 accurate version of scientific record. Related to this point is the idea of group object identifiers. A
777 need for group identifiers has been identified in the area of data (e.g., in the case of meta-analyses),
778 and one could also identify a use case for these in the case of software, collecting metadata around all
779 versions of a given software package. See blog here ([https://blog.datacite.org/to-better-understand-
780 research-communication-we-need-a-groid-group-object-identifier/](https://blog.datacite.org/to-better-understand-research-communication-we-need-a-groid-group-object-identifier/)).

781 **Our response:** We recommend citing the specific version of the software that was used. We
782 expect that the unique identifier (e.g., DOI) will point to a landing page that directs to the repository/
783 current version. However, this is more of a convenience issue that the software developers should
784 address, rather than the author citing the software they used.

785 **C.7. On future work:** For implementation we would recommend both consulting with adopters
786 as well as developing metadata standards simultaneously rather than developing metadata standards
787 and then pursuing early adopters implementation. The work early adopters are doing now for data
788 citation will be able to be leveraged for software citation and the changes needed to do so could
789 happen now. There is no need to wait on approval of new tagging for a specific metadata standard.
790 Many publishers will have their own preferred metadata standards and so implementation could
791 begin now with publishers, as long as we know what we want to capture. Future implementation
792 groups might also consider levels of contribution. This is particularly relevant for software. Who
793 is considered an author? For example, to what extent should authors of pull requests receive
794 attribution? This might be considered in an FAQs group, or possibly an early adopters group.

795 **Our response:** We agree that metadata standards should be developed with the input of adopters,
796 and have updated this text accordingly.

797 **C.8. Additional thoughts (not sure what section this applies to):** The principles do not address
798 virtual machines. As these are becoming more common and relevant when addressing the repro-
799 ducibility of research, it is important this “form” of software is acknowledged. The question remains
800 in which cases should authors cite the current version, which the static archived version, and in
801 which the virtual machine? In this way software is very much a unique evolving research object
802 and might not fit perfectly into the same citation practices and structure as other research objects.
803 In addition, software citation could possibly occur within the virtual machine. This could be added
804 as a use case.

805 **Our response:** We feel this has been addressed in Section 5.8, with the explicit addition of virtual
806 machines in addition to executables and containers. This is also an issue that should be addressed
807 further by the follow-on implementation working group.

808 **C.9. On persistence of identifier vs. persistence of software:** The persistence principle outlined
809 in (4) is a key element in making software citeable. Where software has become part of the record
810 of science not only the identifier and metadata of the software should be persistent, it should also
811 be the goal to keep a persistent copy of the source code, where applicable. This links with the
812 accessibility principle (5).

813 There are still many open questions about how to resolve package dependencies in the long term,
814 therefore I would not make the persistent access to code a hard requirement but may add something
815 more specific towards preserving the record of science.

816 **Our response:** Our goal is for software citations to point to (persistent) archived source code,
817 but we are not—nor could we—require this.

818 **C.10. Granularity of the citation:** One of the key issues with any citation, whether document,
819 individual, or software is the specificity of what is being cited. In the case of publications, there is
820 almost zero specificity most of the time.

821 It’s very easy to cite an entire package even though one function was used. Part of this problem
822 is being solved in the Python world through this project (<https://github.com/ducredit/ducredit>).

823 Any citation should have the ability to specify more than just the obvious, but even the obvious
824 would be a good starting point.

825 The citation/url should therefore allow for greater specificity within a code base. In general
826 though, a provenance record of the workflow would be significantly more useful than a citation from
827 a research perspective.

828 **Our response:** We agree that greater specificity is desirable in some cases, but we do not believe
829 this rises to the level of what should be specified or discussed in the principles at this time.

830 **C.11. “Software citations should permit . . . access to the software itself”:** Under the “Access”
831 header, the data declaration states that:

832 Data citations should facilitate access to the data themselves

833 Under the same header, the software declaration states:

834 Software citations should permit and facilitate access to the software itself

835 The addition of “permit” suggests that software citations should also grant the user with permission
836 to access the software. Is this intentional?

837 It doesn’t seem like a good idea to make access a requirement for discovery, so “permit” might
838 not be helpful in this sentence.

839 **Our response:** To avoid confusion, we removed “permit and” from the accessibility principle.

840 **C.12. Access to software: free vs commercial:** The section talks about software that is “free” as
841 well as “commercial” software. I am not sure whether this is about free as in freedom (or just gratis
842 or freely available), since it is compared with commercial software, which is unrelated in general,
843 see <http://www.gnu.org/philosophy/words-to-avoid.html#Commercial>

844 I suppose that “free” should be replaced by “gratis” and “commercial” be replaced by “non-free”
845 in that section.

846 **Our response:** We think this is sufficiently clear as written.

REFERENCES

- 847
- 848 [1] AAS Editorial Board. Policy statement on software. <http://journals.aas.org/policy/software.html>. Accessed: 2016-
849 02-17.
- 850 [2] S. Ahalt, T. Carsey, A. Couch, R. Hooper, L. Ibanez, R. Idaszak, M. B. Jones, J. Lin, and E. Robinson. NSF
851 workshop on supporting scientific discovery through norms and practices for software and data citation and
852 attribution. Technical report, National Science Foundation, Apr. 2015. Available at: <http://dl.acm.org/citation.cfm?id=2795624>.
- 853 [3] A. Allen, G. B. Berriman, K. DuPrie, J. Mink, R. Nemiroff, T. Robitaille, L. Shamir, K. Shortridge,
854 M. Taylor, P. Teuben, and J. Wallin. Improving software citation and credit. Technical report, arXiv, 2015.
855 arXiv:1512.07919 [cs.DL].
- 856 [4] Astrophysics Source Code Library. <http://ascl.net>. Accessed: 2016-02-21.
- 857 [5] N. Barnes, D. Jones, P. Norvig, C. Neylon, R. Pollock, J. Jackson, V. Stodden, and P. Suber. Science code
858 manifesto. <http://sciencecodemanifesto.org>. Accessed: 2016-04-18.
- 859 [6] S. Bechhofer, I. Buchan, D. D. Roure, P. Missier, J. Ainsworth, J. Bhagat, P. Couch, D. Cruickshank, M. Delderfield,
860 I. Dunlop, M. Gamble, D. Michaelides, S. Owen, D. Newman, S. Sufi, and C. Goble. Why linked data is not
861 enough for scientists. *Future Generation Computer Systems*, 29(2):599–611, 2013.
- 862 [7] biomedical and healthCAre Data Discovery Index Ecosystem (bioCADDIE). <https://biocaddie.org>. Accessed:
863 2016-03-06.
- 864 [8] C. Boettiger and M. B. Jones. Minimal metadata schemas for science software and code, in JSON and XML.
865 <https://github.com/codemeta/codemeta>. Accessed: 2016-03-25.
- 866 [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>.
867 Accessed: 2016-02-17.
- 868 [10] Computational Infrastructure for Geodynamics. <https://geodynamics.org>.
- 869 [11] Consortia Advancing Standards in Research Administration Information. <http://casrai.org/CRedit>. Accessed:
870 2016-02-17.
- 871 [12] Data Citation Synthesis Group, M. Martone (ed). Joint declaration of data citation principles. Final document,
872 FORCE11, San Diego CA, 2014. <https://www.force11.org/group/joint-declaration-data-citation-principles-final>.
- 873 [13] E. Dumbill. DOAP: Description of a project. <https://github.com/edumbill/doap/>. Accessed: 2016-03-31.
- 874 [14] F1000Research. <http://f1000research.com/for-authors/article-guidelines/software-tool-articles>. Accessed: 2016-
875 03-28.
- 876 [15] figshare. <https://figshare.com/>. Accessed: 2016-06-23.
- 877 [16] FORCE11 Resource Identification Initiative. <https://www.force11.org/group/resource-identification-initiative>.
- 878 [17] FORCE11 Resource Identification Technical Specifications Working Group. [https://www.force11.org/group/
879 resource-identification-technical-specifications-working-group](https://www.force11.org/group/resource-identification-technical-specifications-working-group).
- 880 [18] FORCE11 Software Citation Working Group. <https://www.force11.org/group/software-citation-working-group>.
- 881 [19] FORCE11 Software Citation Working Group. Software citation use cases. [https://docs.google.com/document/d/
882 1dS0SqGoBIFwLB5G3HiLLEOSAAgMdo8QPEjYUaWCvIU](https://docs.google.com/document/d/1dS0SqGoBIFwLB5G3HiLLEOSAAgMdo8QPEjYUaWCvIU), 2016. Accessed: 2016-02-10.
- 883 [20] FORCE2016 Conference. Portland, OR, <https://www.force11.org/meetings/force2016>.
- 884 [21] P. Fox and R. Signell. NSF geo-data informatics: Exploring the life cycle, citation and integration of geo-
885 data workshop report. Final document, Rensselaer Polytechnic Institute, 2011. [http://tw.rpi.edu/web/workshop/
886 community/GeoData2011](http://tw.rpi.edu/web/workshop/community/GeoData2011).
- 887 [22] I. Gent, C. Jones, and B. Matthews. Guidelines for persistently identifying software using DataCite. a JISC
888 research Data Spring project. [http://rrr.cs.st-andrews.ac.uk/wp-content/uploads/2015/10/guidelines-software-
889 identification.pdf](http://rrr.cs.st-andrews.ac.uk/wp-content/uploads/2015/10/guidelines-software-identification.pdf), Sept. 2015. Accessed: 2016-04-25.

- 892 [23] Y. Gil, V. Ratnakar, and D. Garijo. OntoSoft: Capturing scientific software metadata. In *Proceedings of the Eighth*
893 *ACM International Conference on Knowledge Capture (K-CAP)*, Oct. 2015. [http://dx.doi.org/10.1145/2815833.](http://dx.doi.org/10.1145/2815833.2816955)
894 2816955.
- 895 [24] GitHub. Making your code citable with GitHub & Zenodo. <https://guides.github.com/activities/citable-code/>,
896 2014. Accessed: 2016-03-10.
- 897 [25] K. Gutzman, S. Konkiel, M. White, M. Brush, V. Ilik, M. Conlon, M. Haendel, and K. Holmes. Attribution of
898 work in the scholarly ecosystem. *figshare*, Apr. 2016. <http://dx.doi.org/10.6084/m9.figshare.3175198.v1>.
- 899 [26] J. E. Hannay, H. P. Langtangen, C. MacLeod, D. Pfahl, J. Singer, and G. Wilson. How do scientists develop and
900 use scientific software? In *Proc. 2009 ICSE Workshop on Soft. Eng. for Computational Sci. and Eng.*, SECSE,
901 pages 1–8, Vancouver, BC, 2009. IEEE.
- 902 [27] J. Howison and J. Bullard. Software in the scientific literature: Problems with seeing, finding, and using software
903 mentioned in the biology literature. *Journal of the Association for Information Science and Technology*, 2015. In
904 press. <http://dx.doi.org/10.1002/asi.23538>.
- 905 [28] Y.-H. Huang, P. W. Rose, and C.-N. Hsu. Citing a data repository: A case study of the protein data bank. *PLoS*
906 *ONE*, 10(8):1–17, 08 2015. <http://dx.doi.org/10.1371/journal.pone.0136631>.
- 907 [29] J. Ison, M. Kalaš, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer, and P. Rice.
908 EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*,
909 29(10):1325–1332, 2013. <http://dx.doi.org/10.1093/bioinformatics/btt113>.
- 910 [30] I. Jackson and C. Schwarz. Debian policy manual. [https://www.debian.org/doc/debian-policy/ch-controlfields.](https://www.debian.org/doc/debian-policy/ch-controlfields.html)
911 [html](https://www.debian.org/doc/debian-policy/ch-controlfields.html). Accessed: 2016-04-17.
- 912 [31] M. Jackson. How to cite and describe software. <http://www.software.ac.uk/how-cite-and-describe-software>. Ac-
913 cessed: 2016-02-17.
- 914 [32] M. Jackson. Oh research software, how shalt I cite thee? [http://www.software.ac.uk/blog/2014-07-30-oh-research-](http://www.software.ac.uk/blog/2014-07-30-oh-research-software-how-shalt-i-cite-thee)
915 [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.
- 916 [33] D. S. Katz. Transitive credit as a means to address social and technological concerns stemming from citation and
917 attribution of digital products. *Journal of Open Research Software*, 2(1):e20, 2014. [http://dx.doi.org/10.5334/jors.](http://dx.doi.org/10.5334/jors.be)
918 [be](http://dx.doi.org/10.5334/jors.be).
- 919 [34] D. S. Katz, S.-C. T. Choi, H. Lapp, K. Maheshwari, F. Löffler, M. Turk, M. Hanwell, N. Wilkins-Diehr, J. Hether-
920 ington, J. Howison, S. Swenson, G. Allen, A. Elster, B. Berriman, and C. Venters. Summary of the first workshop
921 on sustainable software for science: Practice and experiences (WSSSPE1). *Journal of Open Research Software*,
922 2(1):e6, 2014. <http://dx.doi.org/10.5334/jors.an>.
- 923 [35] D. S. Katz, S.-C. T. Choi, N. Wilkins-Diehr, N. Chue Hong, C. C. Venters, J. Howison, F. J. Seinstra, M. Jones,
924 K. Cranston, T. L. Clune, M. de Val-Borro, and R. Littauer. Report on the second workshop on sustainable
925 software for science: Practice and experiences (WSSSPE2). *Journal of Open Research Software*, 4(1):e7, 2016.
926 <http://doi.org/10.5334/jors.85>.
- 927 [36] D. S. Katz, S. T. Choi, K. E. Niemeyer, J. Hetherington, F. Löffler, D. Gunter, R. Idaszak, S. R. Brandt, M. A.
928 Miller, S. Gesing, N. D. Jones, N. Weber, S. Marru, G. Allen, B. Penzenstadler, C. C. Venters, E. Davis, L. Hwang,
929 I. Todorov, A. Patra, and M. de Val-Borro. Report on the third workshop on sustainable software for science:
930 Practice and experiences (WSSSPE3). Technical report, arXiv, 2016. arXiv:1602.02296 [cs.SE].
- 931 [37] D. S. Katz, K. E. Niemeyer, A. M. Smith, W. L. Anderson, C. Boettiger, K. Hinsin, M. Hucka, F. Löffler, T. Pollard,
932 and F. Rios. Software vs data. <https://github.com/danielskatz/software-vs-data>, Accessed 2016-08-21.
- 933 [38] D. S. Katz and A. M. Smith. Implementing transitive credit with JSON-LD. *Journal of Open Research Software*,
934 3:e7, 2015. <http://dx.doi.org/10.5334/jors.by>.
- 935 [39] M. G. Knepley, J. Brown, L. C. McInnes, and B. F. Smith. Accurately citing software and algorithms used in
936 publications. *figshare*, <http://dx.doi.org/10.6084/m9.figshare.785731.v1>, 2013.
- 937 [40] C. Lipson. *Cite Right, Second Edition: A Quick Guide to Citation Styles—MLA, APA, Chicago, the Sciences,*
938 *Professions, and More*. Chicago Guides to Writing, Editing, and Publishing. University of Chicago Press, 2011.
- 939 [41] J. Malone, A. Brown, A. L. Lister, J. Ison, D. Hull, H. Parkinson, and R. Stevens. The Software Ontology (SWO):
940 a resource for reproducibility in biomedical data analysis, curation and digital preservation. *Journal of Biomedical*
941 *Semantics*, 5(1):1–13, 2014. <http://dx.doi.org/10.1186/2041-1480-5-25>.
- 942 [42] M. Mayernik, K. Maull, and D. Hart. Tracing the use of research resources using persistent citable identifiers.
943 https://share.renci.org/SI2PI2015/2015_SI2PI_Posters/mayernik_SI2poster_Feb2015.pdf, 2015. Poster presented
944 at NSF SI2 PI Meeting, Arlington, VA, Accessed: 2016-03-03.
- 945 [43] T. McAdoo. <http://blog.apastyle.org/apastyle/2015/01/how-to-cite-software-in-apa-style.html>.
- 946 [44] A. Morin, J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, and P. Sliz. Shining light into black boxes. *Science*,
947 336(6078):159–160, 2012.

- 948 [45] L. Norén. Invitation to comment on a proposal for a cohesive research software citation-enabling platform.
949 <http://astronomy-software-index.github.io/2015-workshop/>. Accessed: 2016-02-17.
- 950 [46] Open Research Computation. <http://www.openresearchcomputation.com>. Accessed: 2016-03-28.
- 951 [47] M. A. Parsons, R. Duerr, and J.-B. Minster. Data citation and peer review. *Eos, Transactions American Geophysical*
952 *Union*, 91(34):297–298, 2010. <http://dx.doi.org/10.1029/2010EO340001>.
- 953 [48] B. R. Rowe, D. W. Wood, A. N. Link, and D. A. Simoni. Economic impact assessment of NIST’s Text REtrieval
954 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].
- 955 [49] Software for Science: Getting Credit for Code. <https://geodynamics.org/cig/projects/saga/>, 2015. Accessed: 2016-
956 04-06.
- 957 [50] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig. Ten simple rules for reproducible computational research.
958 *PLoS Comp Biol*, 9(10):e1003285, Oct. 2013.
- 959 [51] D. A. W. Soergel. Rampant software errors may undermine scientific results [version 2; referees: 2 approved].
960 *F1000Research*, 3:303, 2015.
- 961 [52] SoftwareX. <http://www.journals.elsevier.com/softwarex/>. Accessed: 2016-03-28.
- 962 [53] Software Credit Workshop. <http://www.software.ac.uk/software-credit>, 2015. Accessed: 2016-04-06.
- 963 [54] SSI Workshops. <http://www.software.ac.uk/community/workshops>. Accessed: 2016-03-31.
- 964 [55] J. Starr, E. Castro, M. Crosas, M. Dumontier, R. R. Downs, R. Duerr, L. Haak, M. Haendel, I. Herman, S. Hodson,
965 J. Hourclé, J. E. Kratz, J. Lin, L. H. Nielsen, A. Nurnberger, S. Proell, A. Rauber, S. Sacchi, A. Smith, M. Taylor,
966 and T. Clark. Achieving human and machine accessibility of cited data in scholarly publications. *PeerJ Computer*
967 *Science*, 1:e1, 5 2015. <https://dx.doi.org/10.7717/peerj-cs.1>.
- 968 [56] S. Sufi, N. P. Chue Hong, S. Hettrick, M. Antonioletti, S. Crouch, A. Hay, D. Inupakutika, M. Jackson, A. Pawlik,
969 G. Peru, J. Robinson, L. Carr, D. De Roure, C. Goble, and M. Parsons. Software in reproducible research: Advice
970 and best practice collected from experiences at the collaborations workshop. In *Proc. 1st ACM SIGPLAN Work. on*
971 *Reproducible Research Methodologies and New Publication Models in Comp. Eng.*, TRUST ’14, pages 2:1–2:4,
972 Edinburgh, United Kingdom, June 2014. ACM.
- 973 [57] H. Van de Sompel, S. Payette, J. Erickson, C. Lagoze, and S. Warner. Rethinking scholarly communication: Build-
974 ing the system that scholars deserve. *D-Lib Magazine*, 10(9), Sept. 2004. [http://www.dlib.org/dlib/september04/](http://www.dlib.org/dlib/september04/vandesompel/09vandesompel.html)
975 [vandesompel/09vandesompel.html](http://www.dlib.org/dlib/september04/vandesompel/09vandesompel.html).
- 976 [58] G. Ward and A. Baxter. Distributing python modules. [https://docs.python.org/2/distutils/setupscript.html#](https://docs.python.org/2/distutils/setupscript.html#additional-meta-data)
977 [additional-meta-data](https://docs.python.org/2/distutils/setupscript.html#additional-meta-data). Accessed: 2016-04-17.
- 978 [59] O. White, A. Dhar, V. Bonazzi, J. Couch, and C. Wellington. NIH Software Discovery Index Meeting Report.
979 Copies of archived content from final document, NIH, 2014. <http://www.softwarediscoveryindex.org/> & <https://gist.github.com/mhucka/44921ea1e9a01697dbd0591d872b7b22>.
- 980 [60] H. Wickham. *R Packages*. O’Reilly Media, Sebastopol, CA, first edition, 2015.
- 981 [61] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff,
982 I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson. Best practices for scientific computing.
983 *PLoS Biol*, 12(1):e1001745, 2014.
- 984 [62] 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.
- 985 [63] WSSSPE Workshops. <http://wssspe.researchcomputing.org.uk/>. Accessed: 2016-03-16.