IJDC | *Peer-Reviewed Paper*

Citations for Software: Providing Identification, Access and Recognition for Research Software

Laura Soito
University of New Mexico

Lorraine J. Hwang
University of California, Davis

Abstract

Software plays a significant role in modern academic research, yet lacks a similarly significant presence in the scholarly record. With increasing interest in promoting reproducible research, curating software as a scholarly resource not only promotes access to these tools, but also provides recognition for the intellectual efforts that go into their development. This work reviews existing standards for identifying, promoting discovery of, and providing credit for software development work. In addition, it shows how these guidelines have been integrated into existing tools and community cultures, and provides recommendations for future software curation efforts.

Received 11 February 2016 ~ *Revision received* 15 September 2016 ~ *Accepted* 8 November 2016

Correspondence should be addressed to Laura Soito, University Libraries, MSC05 3020, 1 University of New Mexico, Albuquerque, NM 87131. Email: lsoito@unm.edu

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Introduction

From simple data processing scripts to complex databases and modelling packages, software and related digital products permeate scholarly research. In surveys, researchers indicate that using and developing scientific software is important to their work (Hannay et al., 2009) and that it would be difficult or impossible to conduct research without these tools (Hettrick et al., 2014). While software is used for many purposes, research software can be identified as the tools that uniquely assist compilation, transformation, analysis or modelling, rather than those tools that simply facilitate communication and presentation of information. Research software includes commercially available resources as well as free or open software, and may be installed locally or used on remote systems. There has been research interest in software produced in academic environments and the mechanisms to promote discoverability, reuse, and research reproducibility, as well as standards of academic rigor and credit for these works (Morin et al., 2013; National Science Foundation, 2012; Peng, 2011; Stodden, Guo and Ma, 2013).

Curation of research software aids in its discoverability and accessibility, which reduces duplication of effort when developing or using similar research methods. These practices work hand in hand with scholarly citation to ascribe value to, and provide recognition for research software. While sharing code creates efficiencies and robustness, researchers are often hesitant and resist sharing their code due to perceived issues of code quality, intellectual property rights, and fears of providing user support or creating undue competition (Barnes, 2010; Cannam, Figueira and Plumbley, 2012; Millman and Pérez, 2014). Funder, publisher, and institutional policies can help to set expectations for more open software and analogous data sharing practices, but there is also recognition that current mechanisms to provide academic credit typically do not incentivize software development, documentation, or sharing (Morin et al., 2012).

Traditionally, software has been viewed as a technical work or invention, rather than a scholarly or creative work. Many organizations maintain intellectual property rights to software, rather than allowing ownership to its creators, as would be the case for journal articles or books. However, software has many functions in the research process. It can be developed as a tool that provides utility in a workflow, such as an instrument. It can also be developed to study a specific research problem. Either way it is an intellectual contribution to the creative process of research. This contrast between technical and creative work is also reflected in cases where software support staff, such as developers and engineers, are often considered technical support rather than researchers. They are sometimes recognized for authorship of articles or technical reports documenting the creation of new software, but the significance of their role amongst a long list of authors is murky. A lack of citation culture and standards for software make it difficult for individuals and software development groups to receive credit for their contributions.

As a digital resource, software also poses challenges for being cited in a bibliocentric, or publication-based, citation system. Software may be developed over decades, with hundreds of people contributing directly or indirectly to its creation. The potential for determining software provenance is growing as the use of version control and collaborative software development systems become more prevalent, but in many cases software development contributions and revision histories have been poorly documented. Modern software commonly has dependencies and thus relies upon code

 doi:10.2218/ijdc.v11i2.390

libraries written by others. When not included in a distribution package these dependencies lead to unclear software boundaries and questions about what exactly should be cited. In addition, research software may not be formally published, but rather made available on websites or only by request. This leads to challenges in finding code as researchers change institutions, code is abandoned, or collections fall into disarray.

In academic research, software makes significant contributions towards the development of new knowledge; however, its sometimes complex creation process and ephemeral nature pose challenges for curation and appropriately crediting software development efforts. In an effort to better understand and improve practices that support open sharing of research software, this work seeks to identify existing approaches for identification, access, and recognition of these resources. More specifically, the aim is to provide software users, developers, and curators with answers to the following questions:

1. What are recommended practices or standards for citing or acknowledging software?
2. What tools have been developed to help software users more easily and accurately track and indicate how software is used in their work?
3. How have research communities encouraged recognition of software development and adoption of documentation practices?

Roles for Citations

Citations are used to serve many intertwined roles in the scholarly landscape (Ayers, 2016; Bonazzi et al., 2015; Goble, Allen, Sands and Cruse, 2016; Jones, Matthews, Gent, Griffin and Tedds, 2016; Smith, Katz, Niemeyer and FORCE11 Software Citation Working Group, 2016). The following list summarizes ways that citation roles connect to software and provides a framework for analysing the effectiveness of approaches identified in the remainder of this paper:

- Identification – Uniquely distinguish a work from others. For software this may include identifying an algorithm, and the environment in which it is implemented, compiled and executed.
- Discovery – Guide readers to related works and help identify resources that fulfil specific needs. While discovery can be facilitated by references to other works, descriptive metadata located elsewhere can also play a significant role selecting new tools.
- Access – Provide the necessary information to obtain and use a work. Beyond providing a place to download or purchase software, this may entail providing information about licensing, platform requirements, configuration, and execution.
- Credit – Recognition for the creators, contributors, and originators of a work. Beyond a code's developers, recognition may be necessary for entities that funded or provided leadership for software development, as well as those who help to maintain and preserve its continued availability.

- Appraisal – Evaluate the quality and reliability of a work. The use of citation and other metrics signal how much attention a work has received. Beyond simple citation, appraisal may include other processes to provide peer-review or document the usability and usefulness of software.
- Provenance – Provides a record of the history of the work, including how it was created, as well as its maintenance, use, and evolution. For software this may include developer identification, commit and change logs, and documentation.
- Connection – Illustrate and capture relationships between different works. Software often draws upon other works in its creation and use.

Standards for the Citation of Software

Approaches for the description and acknowledgement of software come from different information sectors, such as libraries, publishers, professional societies, and software developers. Metadata schemas for software applications[1] and source codes[2] have been established by Schema.org. The Software Ontology[3] was developed for describing software used in biomedical research (Malone et al., 2014) and the EarthCube Initiative's OntoSoft[4] project has emphasized guiding geoscience researchers through creating metadata for discovery and reuse (Gil, Ratnakar and Garijo, 2015). Software is accounted for in more general standards, for example as the software resource type in the DataCite Metadata Schema (DataCite Metadata Working Group, 2015) and computer program content type in the Resource Description and Access (RDA) cataloguing standard (Joint Steering Committee, 2013). The reuse of software is also supported by metadata that capture and allow easy sharing of software licensing terms, such as those found in the Software Package Data Exchange Specification (SPDX, 2016).

Across disciplines and contexts, there is a lack of consistency in software citation practices. Howison and Bullard (2016) found a wide range of citation forms including references to publications, user manuals, project websites, and informal mentions of the tools in their study of the biological literature. Inconsistency is also reflected in recommendations that may include referencing articles that discuss the software, direct citation of the software itself, or simply providing a link to where it can be downloaded (Figure 1). For example, the Publication Manual of the American Psychological Association (APA, 2010) suggests book or website-like entries in the reference list for specialized software and in-text descriptions of the software for standard tools. The IEEE Editorial Style Manual (2014) bases its approach to software on APA and ISO guidelines, but provides examples of citing software manuals rather than software as its own entity. Not all software will have a publication or even an associated manual, thus guides may recognize and accommodate variations in available information. The ACS Style Guide provides five different forms for citation for software that can be used on a case-by-case basis. For example, software that has been published might be cited more like a book or technical report, while software with minimal information available could be cited by providing an author or program name (Coghill and Garson, 2006). The

1 Software Application Schema: https://schema.org/SoftwareApplication
2 Software Source Code Schema: https://schema.org/SoftwareSourceCode
3 The Software Ontology: http://theswo.sourceforge.net/
4 OntoSoft: http://www.ontosoft.org/

 doi:10.2218/ijdc.v11i2.390

American Astronomical Society software policy[5] suggests two approaches: one based on the paper describing the software and one using an associated Digital Object Identifier (DOI). Citations can include both forms of citation along with links to any appropriate repositories. In addition to the style guidelines provided by many sources, there are also formalized standards for creating references like ANSI/NISO Z39.29-2005 (R2010) and ISO 690:2010.

ImageJ Documentation	Style Guides	Software Collections
• Schneider, C. A.; Rasband, W. S. & Eliceiri, K. W. (2012), "NIH Image to ImageJ: 25 years of image analysis", *Nature methods* **9(7)**: 671-675, PMID 22930834.	• **APA:** ImageJ [Computer software]. (2012). Retrieved from http://imagej.nih.gov/ij/download.html • **IEEE:** T. Ferreira and W. Rasband. (2012). *ImageJ User Guide IJ 1.46r* [Online]. Available: http://imagej.nih.gov/ij/docs/guide/index.html • **PhysRev:** W.S. Rasband, computer code ImageJ, U. S. National Institutes of Health, Bethesda, MD, 1997.	• **ASCL:** Rasband, W. S., 2012, *ImageJ,* Astrophysics Source Code Library, record ascl:1206.013 • **eagle-i:** National Institutes of Health. ImageJ. http://rsbweb.nih.gov/ij/ http://ohsu.eagle-i.net/i/0000012f-74a7-bf90-f561-6e8a80000000 • **RRID:** http://rsb.info.nih.gov/ij/index.html, RRID:SCR_003070

Figure 1. Conflicting citation guidelines are illustrated in examples of citations for the image processing software ImageJ as recommended by the software documentation,[6] the following publishing style guides: APA (American Psychological Association, 2010), IEEE (IEEE Periodicals, 2014), PhysRev (American Physical Society, 1993) and software collection guidelines: ASCL (Astrophysics Source Code Library),[7] eagle-I,[8] RRID (Resource Identification Portal).[9]

One feature of software citation that is increasingly recommended is the use of unique and persistent identifiers, such as DOI. This recommendation is consistent with the best practices recommended in the Guidelines for Transparency and Openness Promotion (TOP) in Journal Policies and Practices (TOP Guidelines Committee, 2015) and Joint Declaration on Data Citation Principles (Data Citation Synthesis Group, 2014). Force11's Software Citation Principles also emphasis the use of identification that is "machine actionable, globally unique, interoperable, and recognized by … researchers" (Smith et al., 2016). As an alternative to DOI, software associated with disciplinary databases may be associated with more specific community handles, such as ASCL used by the Astrophysics Source Code Library (Allen and Schmidt, 2015) or RRID proposed by Force11's Resource Identification Initiative (Bandrowski et al., 2015).

5 American Astronomical Society Policy Statement on Software: http://journals.aas.org/policy/software.html
6 ImageJ, Citing: http://imagej.net/Citing
7 Astrophysics Source Code Library, Citing ASCL code entries: http://ascl.net/wordpress/?page_id=351
8 Citing an eagle-i resource: https://www.eagle-i.net/get-involved/for-researchers/citing-an-eagle-i-resource/
9 Resource Identification Portal: https://scicrunch.org/resources

While there may not be full agreement on how to implement citation standards, there is reason to exhibit caution and avoid creating new standards for specific projects or yet another 'unifying' standard. Given the existing diversity in software description standards, projects to connect different standards or extend existing standards to be more compatible with software are most helpful. CodeMeta,[10] an extension of the Mozilla Science Lab's Code as Research Object project, is working to create a minimal metadata set for software that can be used to connect popular software repositories, such as Zenodo and figshare. Similarly, the Research Data Alliance's Persistent Identifier Information Types Working Group has created a framework to identify and support harmonization among different types of persistent identifiers (Weigel, DiLauro and Zastrow, 2014).

Another issue with using standard citation practices is that acknowledging software, especially open source tools developed by many people over long periods of time, may bring into question who should receive credit. The use of version control systems (such as Git, Subversion, or Mercurial) in software development allows for the tracking of individual contributions but not necessarily their intrinsic value. Recognition for software development introduces a discontinuity in that what is a valued product in the software community (i.e. open, readable, well-documented code) is not equivalent to a valued product in the academic community (i.e. peer-reviewed publication) (Millman and Pérez, 2014). Authorship of software development articles poses another challenge in whether *all* contributors, no matter how minor their role, should be authors or even acknowledged or if there should be a threshold (Crusoe et al., 2015). One emerging approach is to allow article authors to better identify their roles, for example the Paper Badger[11] project builds upon the Contributor Roles Taxonomy (CRediT)[12] and allow authors of papers to identify research contributions, including those related to software using digital badges.

Tools to Support Software Citation

Given the complex and conflicting standards for citing software, there has been an emergence of what might be called *metasoftware*, that is, software to support software use. These tools are slowly beginning to help researchers capture information that can be used to cite or otherwise document how software was used in their work, and to more thoroughly document the processes used to develop new functionality in software. They also provide new opportunities for measuring the impact of software in others' work in contexts like tenure and promotion review. Wider adoption of these tools will support many goals of software citation in the academic environment.

At a fundamental level, software developers can take steps to suggest preferred citations for their code in readme files, license agreements, landing pages, user manuals or other documentation. To streamline this, some software tools and programming languages allow users and developers to run code that outputs citation information. For example, the PETSc numerical libraries embeds code that can identify which portions of the library are used and outputs appropriate citation information (Knepley, Brown, McInnes and Smith, 2013). The statistical programming language R supports functions to assist in compiling citations, as well as information about contributors and their roles

10 CodeMeta: https://github.com/codemeta/codemeta
11 Mozilla Science Lab, Contributorship Badges: https://www.mozillascience.org/projects/contributorship-badges
12 CRediT: http://casrai.org/CRediT

 doi:10.2218/ijdc.v11i2.390

(Hornik, Murdoch and Zeileis, 2012). These practices are particularly helpful for recognizing modular pieces of code or libraries with many contributors and acknowledging software that was built upon to create a new product.

To incorporate these suggestions into required style guidelines, researchers may use templates or reference management software. The use of flexible citation templates accommodates the existing variation in mechanisms to cite software, but does not necessarily guide users to provide all the necessary information to ensure that others can consistently locate and use cited materials. In alignment with style manuals, many reference management programs[13] (e.g. EndNote, Zotero) allow users to create references to software. Templates for software mirror those for more traditional book and article sources, but may rename or add fields to account for different roles or practices, such as using 'programmer' rather than 'author' or adding fields for system and version information. Reference management packages like BibTeX and BibLaTeX do not include an explicit software style, but rather more generic 'misc' style format can be used to cite software (Lehman, Kime, Boruvka and Wright, 2015).

Some bibliographic management tools are capable of automatically capturing citation related information from source files and resource databases. General repositories, like Zenodo and figshare, provide suggested citations for software and other resources. While these approaches allow users to download some information about these resources to their citation management programs, structured metadata are not consistently available within software or from the sources where it is obtained. Repositories may also incorporate tools to help researchers find and export citations for software, such as AppCiter which is embedded in the SBGrid Consortium's collection of supported applications (Socias, Morin, Timony and Sliz, 2015). Given the variation in guidelines, these tools may provide citations that point to related works like journal articles and manuals rather than, or in addition to, the software as a discrete research object.

Indexing of computer software began in the mid-1960s and was taken online in the early 1980s (Rorvig, 1988). Over the years there have been numerous attempts to capture and index software products ranging from early efforts, like the Computer Physics Communications Program Library,[14] to the recently established Software Heritage[15] project. There has also been recent interest in incorporating code and software into generalized data repositories. For example, GitHub users are encouraged to make their code citable by obtaining a DOI and archiving their code in Zenodo,[16] whilst Dryad Digital Repository[17] facilitates software archiving during the journal submission process. Increasingly, these tools facilitate workflows which allow researchers to capture and preserve discrete versions of their software alongside their data and publications.

Going beyond citation, metasoftware can provide both greater documentation of context for computational research and new opportunities to express scholarship. The ICERM Workshop on Reproducibility in Computational and Experimental Mathematics identified tools to help integrate code into documents and e-notebooks, track code provenance, track versions and collaboration, and capture the computational environment (Stodden, Bailey et al., 2013). For example, embedding executable code

13 Wikipedia, Comparison of Reference Management Software: https://en.wikipedia.org/wiki/Comparison_of_reference_management_software
14 Computer Physics Communications Program Library: http://cpc.cs.qub.ac.uk/
15 Software Heritage: https://www.softwareheritage.org/
16 GitHub Guides, Making Your Code Citable: https://guides.github.com/activities/citable-code/
17 Dryad Repository, Submission Integration: http://datadryad.org/pages/submissionIntegration

and data into research papers not only helps make these resources more accessible to readers, but also to lower barriers for reviewers to evaluate software as an aspect of the research.[18] Similarly, the use of interactive notebooks, like the Jupyter Notebook,[19] allow computational researchers to capture code and contextual resources like input data or output visualizations. Provenance tracking tools, like Sumatra,[20] provide automation in recording details about the software environment. Going further, virtual machines and cloud computing can be used to capture and give others access to the same computational environment (Howe, 2012). Docker, a tool for creating software containers, can be used to not only capture dependent files, but also to capture how the software was installed and configured (Boettiger, 2015).

While not perfect, there are metrics that can serve as proxies for quantity and quality of software development, and tools can be used to collect these values for appraisal purposes. Examples of these metrics include: number of lines of code, number of downloads, project forks, and ratings (like other scholarly metrics these too can be gamed). Metrics can be collected via a variety of tools that interact with software, including GitHub or other software repositories that include rating systems like the MathWorks File Exchange.[21] In addition these data can be incorporated into tools such as ImpactStory,[22] which collect metrics beyond traditional citation, otherwise known as altmetrics. Depsy[23] has also emerged as a prototype for collecting data on software use and prevalence in social media. These tools help to bring software to a similar visibility as other more traditional research outputs.

Community Approaches and Practices

Many research communities, such as those centred around a discipline, funding source, research technique or programing language, have created mechanisms to help promote software development efforts. Many of these communities have established repositories or indexes to bring code developed or used in the community to one place. Some host conferences, workshops, and online forums or mailing lists to promote networking and exchange of ideas. They may also provide training and work to set standards or guidelines for work produced by community members. While helpful in bringing people together to tackle the challenges of software development, these efforts can be inhibited for reasons like those identified in the astrophysics community: lack of awareness, unwillingness to contribute, loss of project funding, and need for ongoing updates and curation (Allen and Schmidt, 2015).

Some communities have developed software collections or registries to promote more open sharing of code. To build awareness of these collections and encourage contributions, some repositories are closely tied to journals in the discipline. Journals may require or encourage that code be deposited as a condition of publication, for example, agent-based models associated with articles published in *Ecology and Society* must be archived in OpenABM,[24] the computational model library for The Network for Computational Modeling for SocioEcological Science (CoMSES Net) (Rollins, Barton,

18 Executable Paper Grand Challenge: http://www.executablepapers.com/
19 Project Jupyter: http://jupyter.org/
20 Sumatra: http://neuralensemble.org/sumatra/
21 MathWorks File Exchange: https://www.mathworks.com/matlabcentral/fileexchange/
22 ImpactStory: https://impactstory.org/
23 Depsy: http://depsy.org/
24 OpenABM: https://www.openabm.org

 doi:10.2218/ijdc.v11i2.390

Bergin, Janssen and Lee, 2014). Another practice to build awareness involves mining the scholarly literature for software used or developed in the community and adding these to the software collection, rather than relying solely on voluntary contributions, as is done with the Astrophysics Source Code Library (Allen and Schmidt, 2015). This form of active curation is a promising model for tying together an otherwise disjointed archival system.

Another way to make software easier to cite is to share the software through a familiar article-like format, often called a *software article*. Rather than describe a research problem that was studied using the software, software articles are short reports containing structured metadata and description connected to code or executable programs. These articles can be used by researchers to more directly point to the tool or algorithms used, especially in cases where software would otherwise be treated as an unpublished work that is not citable. Structured metadata may include details similar to those found in citations, such as code title, developer names, software license, programming language used, and system requirements. Narrative sections may contain context as to why the code was developed, what functionalities it provides, and how the code has been tested. Source code and executable files are archived by the article publisher or connected via a persistent identifier link to a software repository. The Software Sustainability Institute provides examples of both general and discipline-specific journals for publishing software,[25] such as *Journal of Open Research Software*,[26] *SoftwareX*[27] and *BMC Source Code for Biology and Medicine.*[28] In addition to providing venues for sharing software, this approach allows software to be indexed in the same tools that promote discoverability of other academic works.

There are also efforts to establish standards of academic rigor and procedures for evaluating code. For example, the Advanced Research Consortium has created guidelines and identifies qualified reviewers to be called upon to evaluate digital projects in terms of scholarly content and technical standards (Grumbach and Mandell, 2014). CoMSES Net incentivizes creation of high quality metadata and documentation through a peer-review process leading to certification in the OpenABM library as an alternative to formal publishing. In this process the code and documentation are reviewed for adherence to documentation guidelines and it is verified that the model can be run given provided instructions (Rollins et al., 2014). These efforts pave the way for researchers working on digital projects to obtain scholarly credit for their work.

While there are many projects and groups considering issues related to software, there are also efforts to bring people and initiatives together. Force11's Software Citation Working Group[29] and events like the Workshops on Sustainable Software for Science: Practice and Experiences (WSSSPE)[30] or the Software Sustainability Institute's Collaborations Workshops[31] involve people from many domains and disciplines. The US National Institutes of Health hosted a workshop to explore the creation of a Software Discovery Index to help researchers find, cite, and reuse software (Bonazzi et al., 2015). The US National Science and Sloan Foundations have brought together researchers working on software projects through workshops to actively engage and design pilots or

25 Software Sustainability Institute, In which journals should I publish my software? http://www.software.ac.uk/resources/guides/which-journals-should-i-publish-my-software
26 Journal of Open Research Software: http://openresearchsoftware.metajnl.com/
27 SoftwareX: http://www.journals.elsevier.com/softwarex
28 Source Code for Biology and Medicine: http://www.scfbm.org/
29 Force11 Software Citation Working Group: https://www.force11.org/group/software-citation-working-group
30 WSSSPE: http://wssspe.researchcomputing.org.uk/
31 Software Sustainability Institute Workshops: http://www.software.ac.uk/community/workshops

experiments to address software issues like discoverability and attribution (Ahalt et al., 2015; Timmes et al., 2015).

Analysis and Recommendations for Achieving Citation Goals

As illustrated in the previous sections there are many mechanisms that can be utilized to provide recognition and support curation of research software. However, these mechanisms do not universally or equally address citation goals. Moving forward in the support of curated software collections there are a variety of issues stakeholders should be aware of and many techniques that could be deployed to improve support for software as an essential research tool.

Identification

The use of unique, persistent, actionable identifiers is essential for capturing and distinguishing software products. While identifiers are increasingly required by style guides, these guidelines do not consistently recommend how to address different versions or instances of software and associated code. References that provide more specific information, such as version or platform details benefit research reproducibility, but using identifiers that more generally direct users to a software project provide greater context, improved flexibility for users, and the ability to capture collective metrics. Requiring that metadata capture relationships between software entities, such as [Software B] is a [new Version] of [Software A], is one approach to improving clarity (Jones et al., 2016). Designating software entities through the use of identifier suffixes that allow users to select a more specific or broader access point (e.g. softwareID:1234/v3) provides more flexibility, but may not fully accommodate the needs of code with many variants or an otherwise complex development history. The creation of a system that allows users to verify whether they are using the newest version of software and to alert users to known issues, similar to the information the service CrossMark[32] provides for articles, could be indispensable to those seeking citation metadata and code updates, especially as code is reused further from its original context.

Access and Discovery

Identifiers are not necessarily sufficient to provide consistent access to software. There are diverse options in identifiers and standards for citing software, which leads to a lack of consistency in describing and finding these resources. This can be exacerbated when software is used across disciplinary boundaries (e.g. the image analysis tool ImageJ noted in Figure 1) and tools acquire different identifiers and conflicting metadata from different access points. Community and publisher efforts to mandate the use of standardized repositories, as is the case for other research products (e.g. the Protein Data Bank[33] for macromolecular structural data) would be a starting point for greater and more consistent access. As software evolves, providing stable points of access through archives, registries, or (less desirably) *software article* approaches allows software to be

32 CrossMark: http://www.crossref.org/crossmark/
33 Worldwide Protein Data Bank: http://www.wwpdb.org/

 doi:10.2218/ijdc.v11i2.390

connected to something less ephemeral and more readily accessed. These systems support discovery and metadata consistency, and improve understanding and usability via the capture of software context.

Credit and Appraisal

Credit for software relies not only on standards and technology for supporting software citation, but also acceptance from multiple communities of stakeholders. Enacting citation practices is hindered by academic publishers, research review committees, and other components of the academic landscape that do not yet have systems to recognize research formats, including software, that fall outside what can be traditionally published. Creating systems that facilitate the critical evaluation and review of code, both as technical and intellectual research products, bring greater acceptance of these works as scholarship. Organizations such as universities and professional societies should consider investment in software infrastructure that parallels what these organizations once provided in terms of scholarly presses for publications. Publishers have already begun this process in the creation of quasi-new formats, such as the software article. However, by taking advantage of altmetrics or other methods to measuring software diffusion it becomes possible to bypass the use of journal article proxies for software. This will require a cultural shift, one that reimagines software as scholarship, rather than a mere tool to facilitate scholarship.

Provenance and Connection

Capturing provenance and connection can be assisted by metasoftware that incorporates citation activities into researcher workflows. Learning to do research in new ways can have a learning curve, and especially with competing demands upon researchers there may not be significant motivation to change practices that have worked in the past. Incorporating software citation into existing workflows, such as providing full templates for citation in reference managers and styles, helps researchers to begin to adapt practices that are already familiar. The machine executable nature of software and the ability for it to draw from external libraries, also uniquely positions these resources to be incorporated into automated workflows. Systems are already being used to capture the history of a code's development and can also be used to connect code to metadata for contributors, their associated institutions and roles, as well as funders, and support the collection of usage metrics. Future versions of programing languages and software development tools should better incorporate functions that assist automatic extraction of citations from software that is used. It might also be possible to create systems that facilitate the logging of software use, rather than simply registering its existence. This approach would help stakeholders better understand how software is being used, even if the results of its use are never formally published.

Conclusions

It should not be surprising that there have been challenges in capturing software within a bibliographic model, especially given the fundamental differences in publishing, use of citation, and indexing of scholarly works across disciplinary communities. One of the greatest barriers to software citation is not a lack of standards that could be used, but

rather a lack of knowledge of these standards and agreement on how to use them. Improvements to software citation systems will be facilitated by developing metasoftware that lowers thresholds to using software in a research ready state, without complex installation and configuration processes, to the community as a whole. As these tools become more incorporated into researcher workflows, they will also help to facilitate greater access, more comprehensive peer evaluation, and indirectly, understanding of software development as a scholarly process. Software development for research is inherently interdisciplinary, and communities that are willing to recognize and accept diversity of approach in the generation of new knowledge will be more successful in fostering collaborative development of new software to support their work. These communities will also recognize software as a legitimate contribution to research, and support opportunities for career advancement for researchers who chose to pursue this path.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Grant No. SMA-1448633. The authors would like to thank additional members of the Software Attribution for Geoscience Applications (SAGA) project team: Louise Kellogg, Joe Dumit, MacKenzie Smith, Allison Fish, and Eric Heien, as well as Phoebe Ayers, Karl Benedict, Sever Bordeianu, Brian Kolb, and Plato Smith for helpful suggestions.

References

Ahalt, S., Carsey, T., Couch, A., Hooper, R., Ibanez, L., Idaszak, R., … Robinson, E. (2015). NSF workshop on supporting scientific discovery through norms and practices for software and data citation and attribution. Retrieved from http://dl.acm.org/citation.cfm?id=2795624

Allen, A. & Schmidt, J. (2015). Looking before leaping: Creating a software registry. *Journal of Open Research Software*, *3*(1). doi:10.5334/jors.bv

American Physical Society. (1993). *Physical review style notation and guide – June 2011 revision*. Waldron, A., Judd, P., & Miller, V. (Eds.). Ridge, NY: American Physical Society. Retrieved from https://journals.aps.org/files/styleguide-pr.pdf

American Psychological Association. (2010). *Publication manual of the American Psychological Association* (6th ed). Washington, DC: American Psychological Association.

Ayers, P. (2016). What's a citation good for, anyway? *Medium*. Retrieved from https://medium.com/@phoebeayers/whats-a-citation-good-for-anyway-e7585bb003d#.z1t6k6sx6

Bandrowski, A., Brush, M., Grethe, J.S., Haendel, M.A., Kennedy, D.N., Hill, S., … Vasilevsky, N. (2015). The resource identification initiative: A cultural shift in publishing. *F1000Research*, *4*, 134. doi:10.12688/f1000research.6555.2

Barnes, N. (2010). Publish your computer code: It is good enough. *Nature*, *467*, 753. doi:10.1038/467753a

Boettiger, C. (2015). An introduction to Docker for reproducible research. *Operating Systems Review*, *49*(1), 71–79. doi:10.1145/2723872.2723882

Bonazzi, V., Bourne, P., Brenner, S., Brown, R., Chandramouliswaran, I., Couch, J., … White, O. (2015). *Software Discovery Index workshop report*. Retrieved from https://nciphub.org/resources/885

Cannam, C., Figueira, L. A., & Plumbley, M.D. (2012). Sound software: Towards software reuse in audio and music research. In 2012 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2745–2748). doi:10.1109/ICASSP.2012.6288485

Coghill, A.M., & Garson, L.R. (Eds.). (2006). *The ACS style guide: Effective communication of scientific information* (3rd ed). Washington, DC: Oxford ; New York: American Chemical Society; Oxford University Press.

Crusoe, M.R., Alameldin, H.F., Awad, S., Boucher, E., Caldwell, A., Cartwright, R., … Brown, C.T. (2015). The khmer software package: Enabling efficient nucleotide sequence analysis. *F1000Research*, *4*, 900. doi:10.12688/f1000research.6924.1

Data Citation Synthesis Group. (2014). *Joint declaration of Data Citation Principles*. (M. Martone, Ed.). San Diego, CA: FORCE11. Retrieved from https://www.force11.org/datacitation

DataCite, Metadata Working Group. (2015). DataCite metadata schema for the publication and citation of research data. Version 3.1. doi:10.5438/0010

Gil, Y., Ratnakar, V., & Garijo, D. (2015). OntoSoft: Capturing scientific software metadata. In *Proceedings of the Eighth ACM International Conference on Knowledge Capture* (Article 32). doi:10.1145/2815833.2816955

Goble, C., Allen, A., Sands, A., & Cruse, P. (2016). CodeMeta software use cases working document. Retrieved from https://github.com/codemeta/codemeta

Grumbach, E., & Mandell, L. (2014). Meeting scholars where they are: The Advanced Research Consortium (ARC) and a social humanities infrastructure. *Scholarly and Research Communication*, *5*(4). Retrieved from http://www.src-online.ca/index.php/src/article/view/189

Hannay, J.E., MacLeod, C., Singer, J., Langtangen, H.P., Pfahl, D., & Wilson, G. (2009). How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering* (pp. 1–8). doi:10.1109/SECSE.2009.5069155

Hettrick, S., Antonioletti, M., Carr, L., Chue Hong, N., Crouch, S., De Roure, D., … Sufi, S. (2014). *UK Research Software Survey 2014*. Zenodo. doi:10.5281/zenodo.14809

Hornik, K., Murdoch, D., & Zeileis, A. (2012). Who did what? The roles of R package authors and how to refer to them. *The R Journal*, *4*(1), 64–69.

Howe, B. (2012). Virtual appliances, cloud computing, and reproducible research. *Computing in Science Engineering*, *14*(4), 36–41. doi:10.1109/MCSE.2012.62

Howison, J., & Bullard, J. (2016). 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, 67(*9), 2137-2155. doi:10.1002/asi.23538

IEEE Periodicals. (2014). *IEEE Editorial Style Manual*. Piscataway, NJ: IEEE. Retrieved from https://www.ieee.org/documents/style_manual.pdf

Joint Steering Committee. (2013). *RDA: Resource Description and Access: 2013 revision*. Chicago, IL: American Library Association.

Jones, C.M., Matthews, B.M., Gent, I., Griffin, T., & Tedds, J.A. (2016). Persistent identification and citation of software. Presented at the 16th International Digital Curation Conference (IDCC16), Amsterdam, The Netherlands. Retrieved from http://purl.org/net/epubs/work/24496942

Knepley, M.G., Brown, J., McInnes, L.C., & Smith, B. (2013). *Accurately citing software and algorithms used in publications* (ANL/MCS-P5010-0913). Presented at the Workshop on Sustainable Software for Science: Practice and Experiences, Denver, CO. Retrieved from http://www.mcs.anl.gov/papers/P5010-0913_1.pdf

Lehman, P., Kime, P., Boruvka, A., & Wright, J. (2015). The Biblatex package: Programmable bibliographies and citations, Version 3.0. Retrieved from http://mirror.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf

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

Millman, K.J., & Pérez, F. (2014). Developing open-source scientific practice. In Stodden, V., Leisch, F., & Peng, R.D. (Eds.), *Implementing Reproducible Research* (pp. 149-184). Boca Raton: CRC Press. Retrieved from http://www.crcnetbase.com/doi/abs/10.1201/b16868-9

Morin, A., Eisenbraun, B., Key, J., Sanschagrin, P.C., Timony, M.A., Ottaviano, M., & Sliz, P. (2013). Collaboration gets the most out of software. *eLife*, *2*. doi:10.7554/eLife.01456

Morin, A., Urban, J., Adams, P. D., Foster, I., Sali, A., Baker, D., & Sliz, P. (2012). Shining light into black boxes. *Science*, *336*(6078), 159–160. doi:10.1126/science.1218263

National Science Foundation. (2012). *A vision and strategy for software for science, engineering, and education: Cyberinfrastructure framework for the 21st century* (No. nsf12113). Retrieved from http://www.nsf.gov/pubs/2012/nsf12113/nsf12113.pdf

Peng, R.D. (2011). Reproducible research in computational science. *Science*, *334*, 1226–1227. doi:10.1126/science.1213847

Rollins, N.D., Barton, C.M., Bergin, S., Janssen, M.A., & Lee, A. (2014). A computational model library for publishing model documentation and code. *Environmental Modelling & Software*, *61*, 59–64. doi:10.1016/j.envsoft.2014.06.022

Rorvig, M.E. (1988). Bibliographic control of microcomputer software. In Kent, A. (Ed.), *Encyclopedia of Library and Information Science Vol. 43, Supplement 8*. New York: Marcel Dekker.

Smith, A.M., Katz, D.S., Niemeyer, K.E., & FORCE11 Software Citation Working Group. (2016). Software citation principles. PeerJ Computer Science, 2, e86. doi:10.7717/peerj-cs.86

Socias, S.M., Morin, A., Timony, M.A., & Sliz, P. (2015). AppCiter: A web application for increasing rates and accuracy of scientific software citation. *Structure*, *23*(5), 807–808. doi:10.1016/j.str.2015.04.005

SPDX. (2016). Software Package Data Exchange (SPDX®) Specification (No. v. 2.1). Linux Foundation. Retrieved from https://spdx.org/spdx-specification-21-web-version

Stodden, V., Bailey, D.H., Borwein, J., LeVeque, R.J., Rider, W., & Stein, W. (Eds.). (2013). *Setting the default to reproducible: Reproducibility in computational and experimental mathematics*. Report of the ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, December 10-14, 2012. Retrieved from https://icerm.brown.edu/tw12-5-rcem/

Stodden, V., Guo, P., & Ma, Z. (2013). Toward reproducible computational research: An empirical analysis of data and code policy adoption by journals. *PLoS ONE*, *8*(6), e67111. doi:10.1371/journal.pone.0067111

Timmes, F., Ahalt, S., Turk, M., Idaszak, R., Schildhauer, M., Brower, R., … Gustafson, K. (2015). *Workshop report 2015 Software Infrastructure for Sustained Innovation (SI 2) principal investigators workshop*. Retrieved from http://cococubed.asu.edu/si2pimeeting2015/ewExternalFiles/Final_Report_2015_SI2_Workshop.pdf

TOP Guidelines Committee. (2015). Guidelines for transparency and openness promotion (TOP) in journal policies and practices version 1.0.1. Center for Open Science. Retrieved from https://osf.io/ud578/

Weigel, T., DiLauro, T., & Zastrow, T. (2014). PID information types: Final report. Retrieved from https://rd-alliance.org/system/files/PIT%20final%20report.pdf