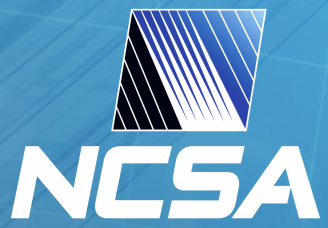
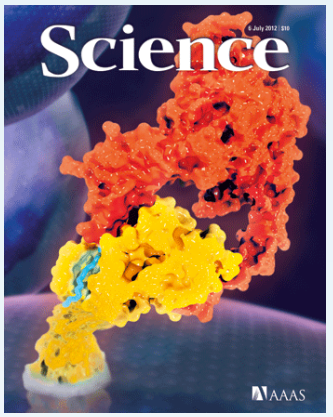
Citation and Reproducibility in Software

Daniel S. Katz
Assistant Director for Scientific Software & Applications, NCSA
Research Associate Professor, CS
Research Associate Professor, ECE
Research Associate Professor, iSchool
dskatz@illinois.edu, d.katz@ieee.org, @danielskatz



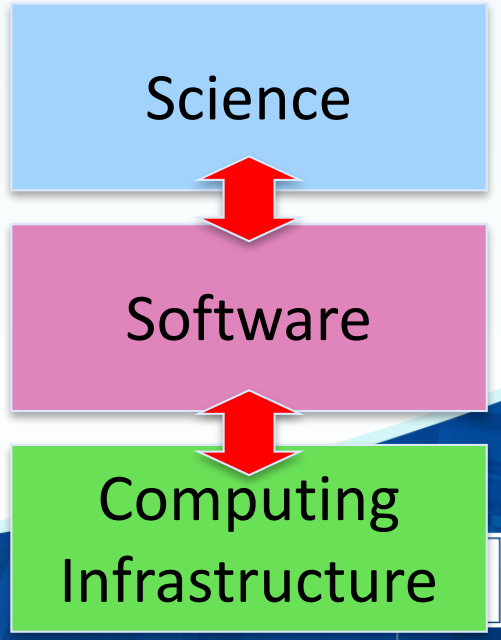
National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign

Software as infrastructure

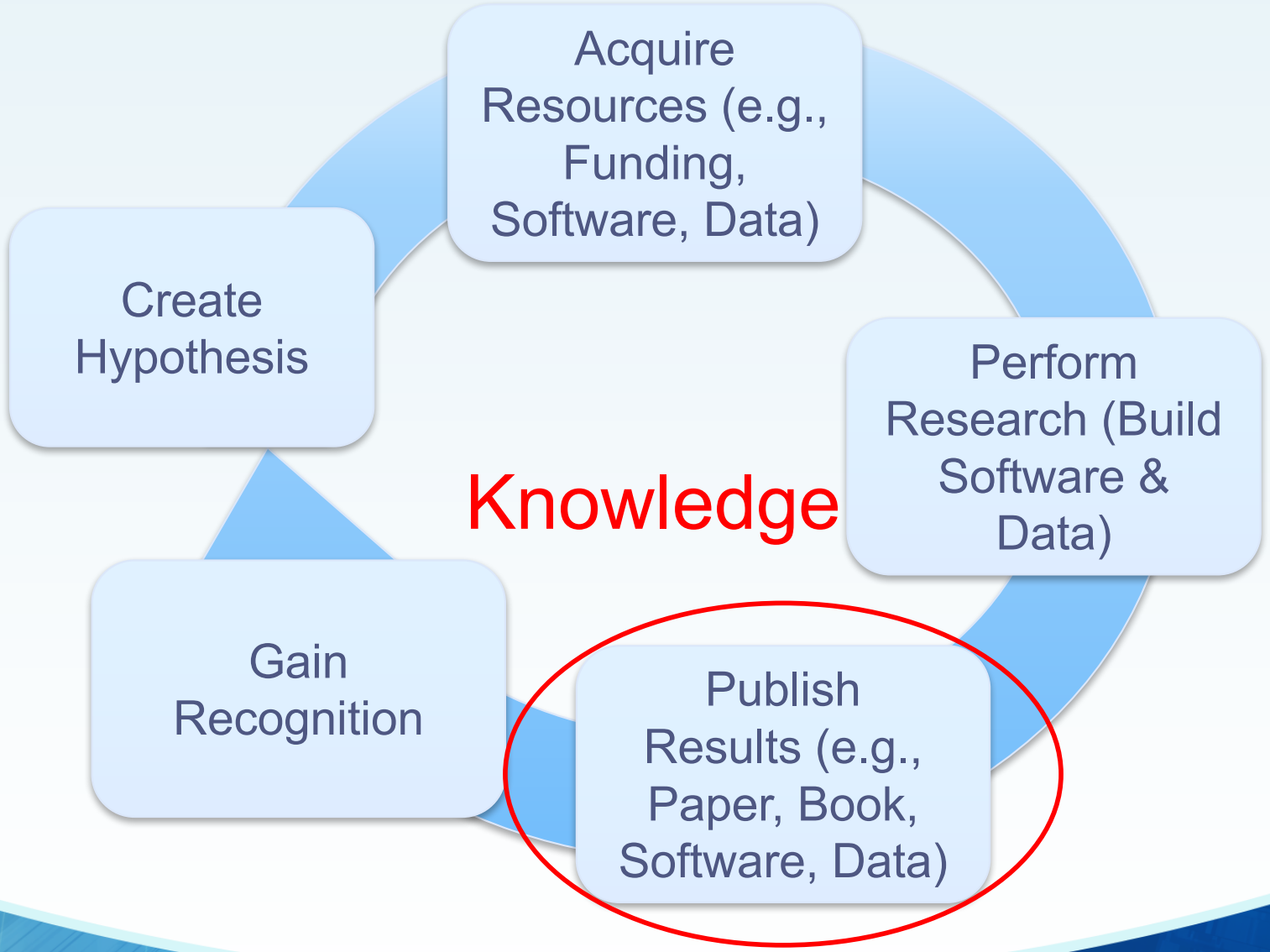


- Software (including services) essential for the bulk of science
  - About half the papers in recent issues of Science were software-intensive projects
  - Research becoming dependent upon advances in software
  - Wide range of software types: system, applications, modeling, gateways, analysis, algorithms, middleware, libraries

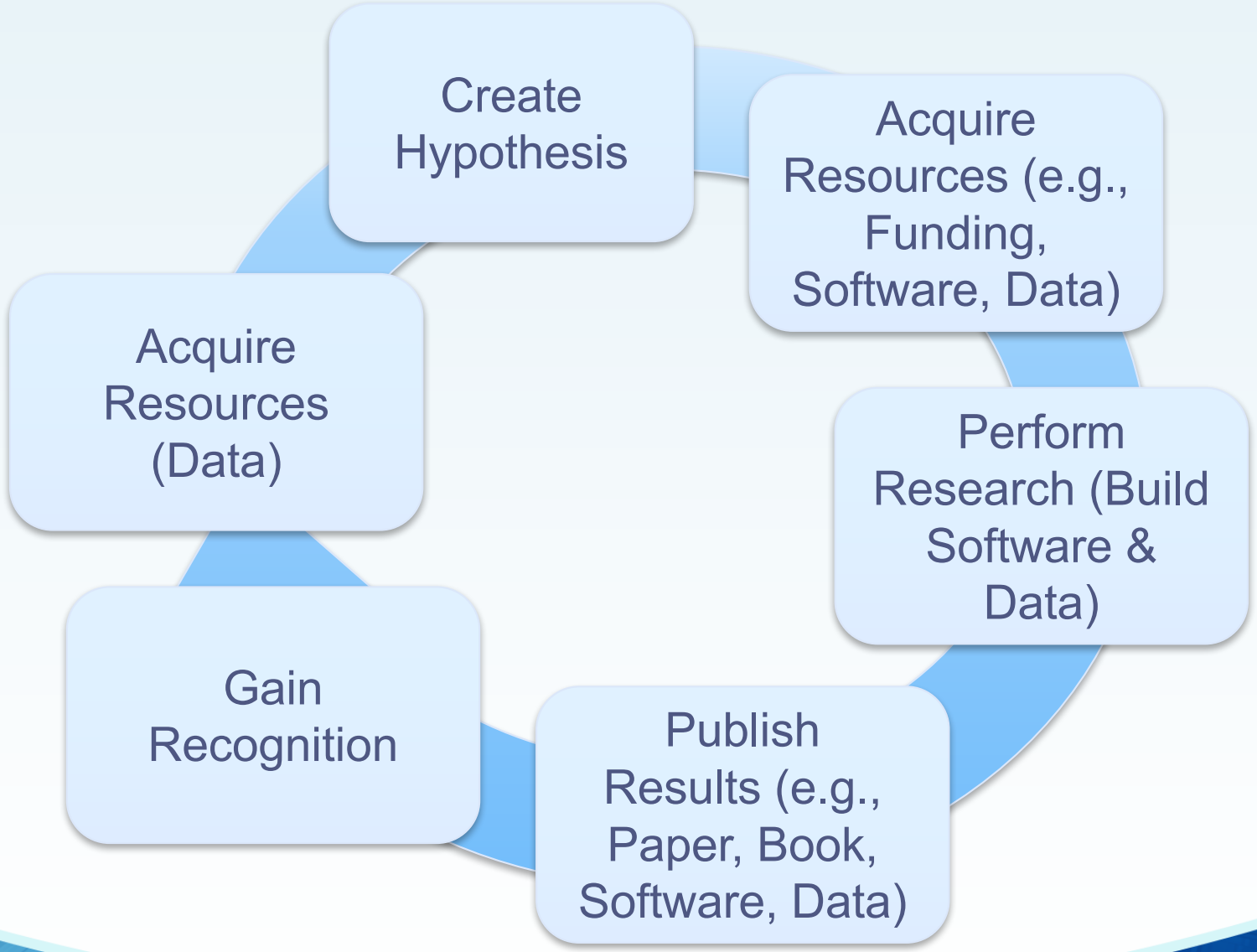
- Software is not a one-time effort, it must be sustained
  - Development, production, and **maintenance** are people intensive
  - Software life-times are long vs hardware
  - Software has under-appreciated value



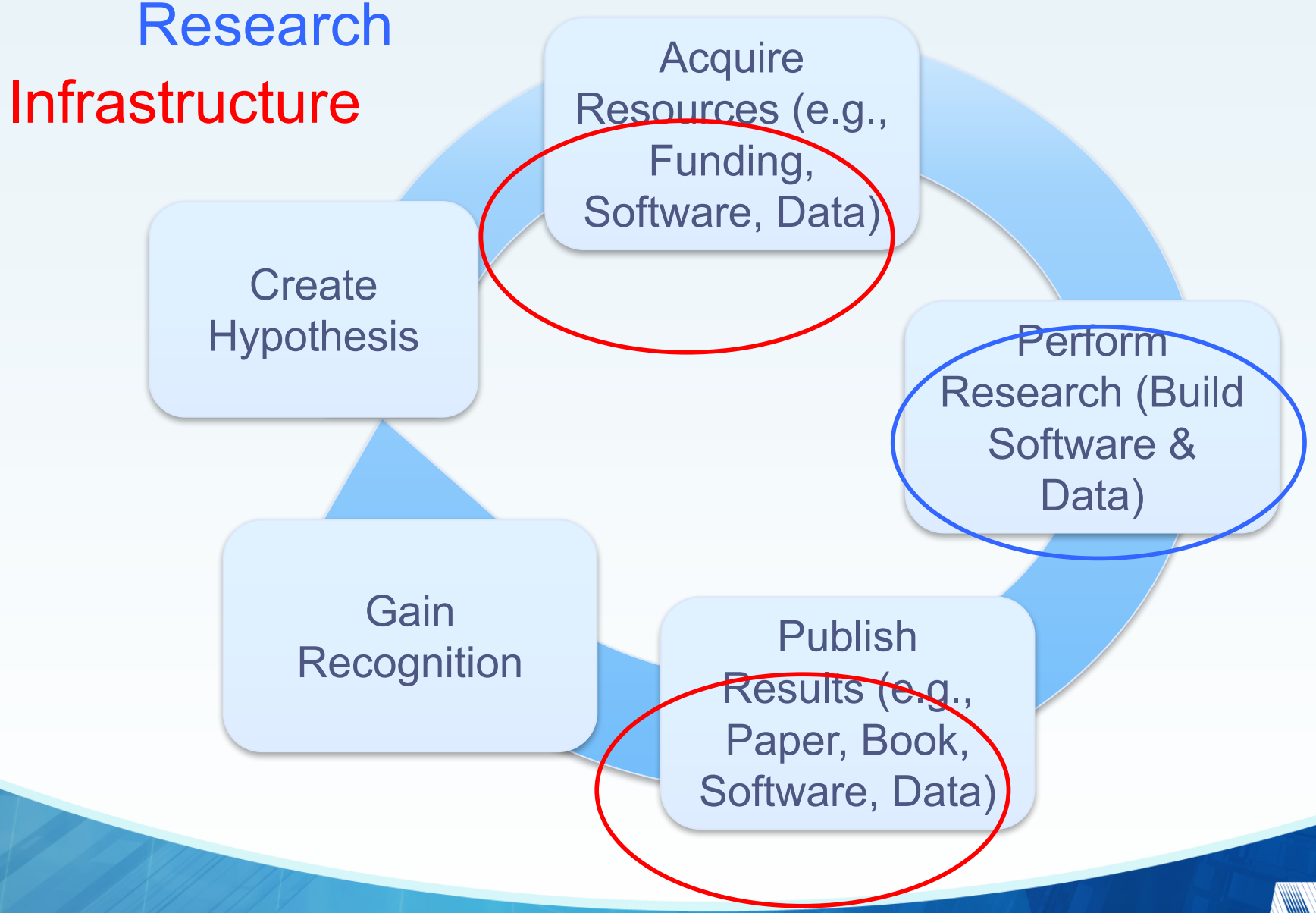
Computational science research



Data science research

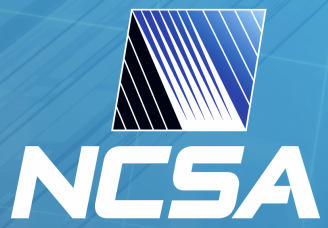


Purposes of software in research



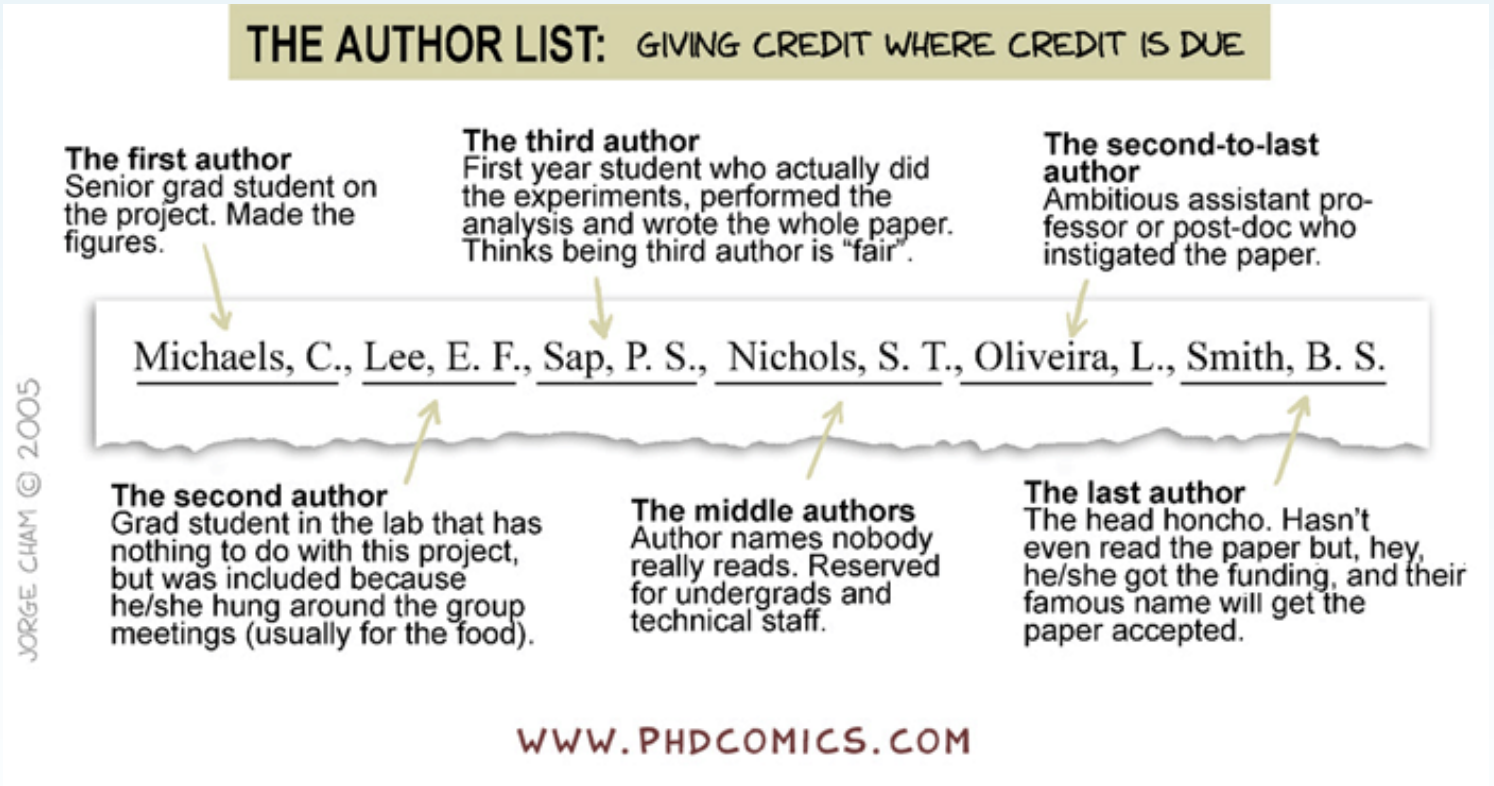
Software Citation

Software Reproducibility



National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign

Credit is a problem in Academia



http://www.phdcomics.com/comics/archive.php?comicid=562

Another Problem

COPE COMMITTEE ON PUBLICATION ETHICS

What are you looking for

Home About COPE Resources Cases Become a member Members Events News & Opinion Contact Us

Home / Cases /

What constitutes authorship?

CASE NUMBER:
10-19

CASE TEXT (ANONYMISED)

Author X submitted a paper to another journal, and included author Y, a student in the same institute, as a courtesy. Author Y had drawn two figures for the paper and discussed some of the observations (all made by author X) with author X but the paper did not deal with the thesis research of author Y.

After the original paper was returned, requiring extensive revisions, author X revised the paper and submitted it to our journal. Author X did not include author Y as an author, but listed Y's contribution in the acknowledgments. Following notification of acceptance and online publication, author Y complained to the editor, insisting that he should be added as an author by the journal without reference to author X. The editor contacted author X who indicated that author Y did not make a substantial scientific contribution to the article and that the notice in the acknowledgments was appropriate.

Authors X and Y hold similar views as to what author Y's contribution was, but different views on whether this is sufficient for authorship. The editor notified authors X, Y and the administrative head of the institute, and recommended that the parties resolve the matter among themselves. Author Y indicated that this suggested approach was unacceptable, repeated that he should be added as an author on the paper regardless of the opinion of author X and did not accept the editor's position that this matter could not be resolved arbitrarily in author Y's favour by the editor or the publisher. Author X has insisted on

Software Citation Motivation

- Scientific research is becoming:
  - More open – scientists want to collaborate; want/need to share
  - More digital – outputs such as software and data; easier to share
- Significant time spent developing software & data
- Efforts not recognized or rewarded
- Citations for papers systematically collected, metrics built
  - But not for software & data
- Hypothesis:

  Better measurement of contributions (citations, impact, metrics)

  —> Rewards (incentives)

  —> Career paths, willingness to join communities

  —> More sustainable software

Incentives, citation/credit models, and metrics (1)

1. Downloads
   - From web/repository logs/stats
2. Installation/Build
   - Add “phone home” mechanism at build level
   - e.g., add ‘curl URL’ in makefile
3. Use/Run
   - Track centrally; add “phone home” mechanism at run level
     - Per app, e.g. add ‘curl URL’ in code, send local log to server
     - For a set of apps, e.g., sempervirens project, Debian popularity contest
   - Track locally; ask user to report
     - e.g., duecredit project, or via frameworks like Galaxy

Incentives, citation/credit models, and metrics (2)

4. Impact
   - Project CRediT (and Mozilla contributorship badges)
     - Record contributorship roles
   - Force 11 Software Citation Working Group (and merged WSSSPE software credit WG)
     - Define software citation principles
   - Codemeta
     - Minimal metadata schemas for science software
   - Lots of research projects (including my own research on transitive credit - http://doi.org/10.5334/jors.by)
   - Altmetrics – not citations, but other structured measures of discussion (tweets, blogs, etc.)
   - ImpactStory
     - Measure research impact: reads, citations, tweets, etc.
   - Depsy (roughly ImpactStory specifically for software)
     - Measure software impact: downloads; software reuse (if one package is reused in another package), literature mentions (citations of a software package), and social mentions (tweets, etc.)

How to better measure software contributions

- Citation system was created for papers/books
- We need to either/both
  1. Jam software into current citation system
  2. Rework citation system
     - Focus on 1 as possible; 2 is very hard.
- Challenge: not just how to identify software in a paper
  - **How to identify software used within research process**

Software citation today

- Software and other digital resources currently appear in publications in very inconsistent ways
- Howison: random sample of 90 articles in the biology literature -> 7 different ways that software was mentioned

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%

- Studies on data and facility citation -> similar results

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

NCSA

Software citation principles: People & Process

- FORCE11 Software Citation group started July 2015
- WSSSPE3 Credit & Citation working group joined September 2015
- ~55 members (researchers, developers, publishers, repositories, librarians)
- Working on GitHub https://github.com/force11/force11-scwg & FORCE11 https://www.force11.org/group/software-citation-working-group
- Reviewed existing community practices & developed use cases
- Drafted software citation principles document
  - Started with data citation principles, updated based on software use cases and related work, updated based working group discussions, community feedback and review of draft, workshop at FORCE2016 in April
  - Discussion via GitHub issues, changes tracked
- Contains 6 principles, motivation, summary of use cases, related work, discussion & recommendations
- Submitted, reviewed and modified (many times), now published
  - Smith AM, Katz DS, Niemeyer KE, FORCE11 Software Citation Working Group.(2016) Software Citation Principles. *PeerJ Computer Science* 2:e86. DOI: 10.7717/peerj-cs.86 and https://www.force11.org/software-citation-principles

Software citation principles

- Contents (details on next slides):
  - 6 principles: Importance, Credit and Attribution, Unique Identification, Persistence, Accessibility, Specificity
  - Motivation, summary of use cases, related work, and discussion (including recommendations)
- Format: working document in GitHub, linked from FORCE11 SCWG page, discussion has been via GitHub issues, changes have been tracked
- https://github.com/force11/force11-scwg
- Reviews and responses also in *PeerJ CS* paper

Principle 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.

Principle 2. Credit and Attribution

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

Principle 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.

Principle 4. Persistence

- **Unique identifiers and metadata describing the software and its disposition should persist** – even beyond the lifespan of the software they describe.

Principle 5. Accessibility

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

Principle 6. Specificity

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

Example 1: Make your software citable

- Publish it – if it's on GitHub, follow steps in https://guides.github.com/activities/citable-code/
- Otherwise, submit it to zenodo or figshare, with appropriate metadata (including authors, title, …, citations of … & software that you use)
- Get a DOI
- Create a CITATION file, update your README, tell people how to cite
- Also, can write a software paper and ask people to cite that (but this is secondary, just since our current system doesn't work well)

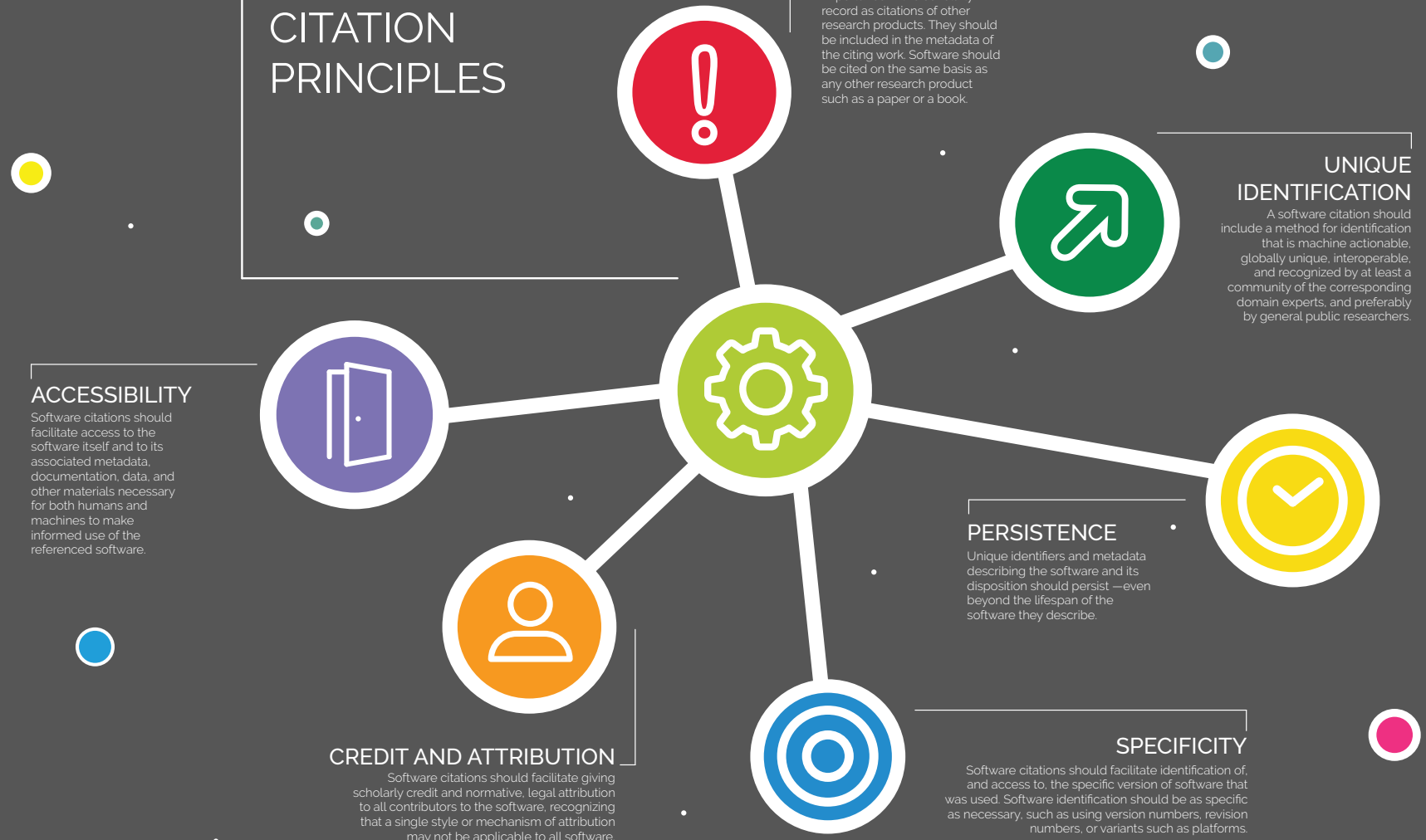
Example 2: Cite someone else's software in a paper

- Check for a CITATION file or README; if this says how to cite the software itself, do that
- If not, do your best following the principles
  - Try to include all contributors to the software (maybe by just naming the project)
  - Try to include a method for identification that is machine actionable, globally unique, interoperable – perhaps a URL to a release, a company product number
  - If there's a landing page that includes metadata, point to that, not directly to the software (e.g. the GitHub repo URL)
  - Include specific version/release information
- If there's a software paper, can cite this too, but not in place of citing the software

Working group status & next steps

- Final version of principles document published in PeerJ CS
- Considering endorsement period for both individuals and organizations (will suggest to FORCE11, might defer to implementation phase)
  - Want to endorse? Email/talk to me
- Will create infographic and 1–3 slides
  - In progress; draft infographic on next slide
- Will create white paper that works through implementation of some use cases
- Software Citation Working Group ends
- Software Citation Implementation group starts
  - Works with institutions, publishers, funders, researchers, etc.,
  - Writes full implementation examples paper?
  - Want to join? Sign up on current FORCE11 group page
    - https://www.force11.org/group/software-citation-working-group

SOFTWARE CITATION PRINCIPLES



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. They should be included in the metadata of the citing work. Software should be cited on the same basis as any other research product such as a paper or a book.

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.

ACCESSIBILITY

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

PERSISTENCE

Unique identifiers and metadata describing the software and its disposition should persist —even beyond the lifespan of the software they describe.

CREDIT AND ATTRIBUTION

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

SPECIFICITY

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

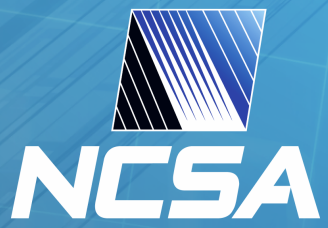
Credit: Laura Rueda, DataCite

Journal of Open Source Software (JOSS)

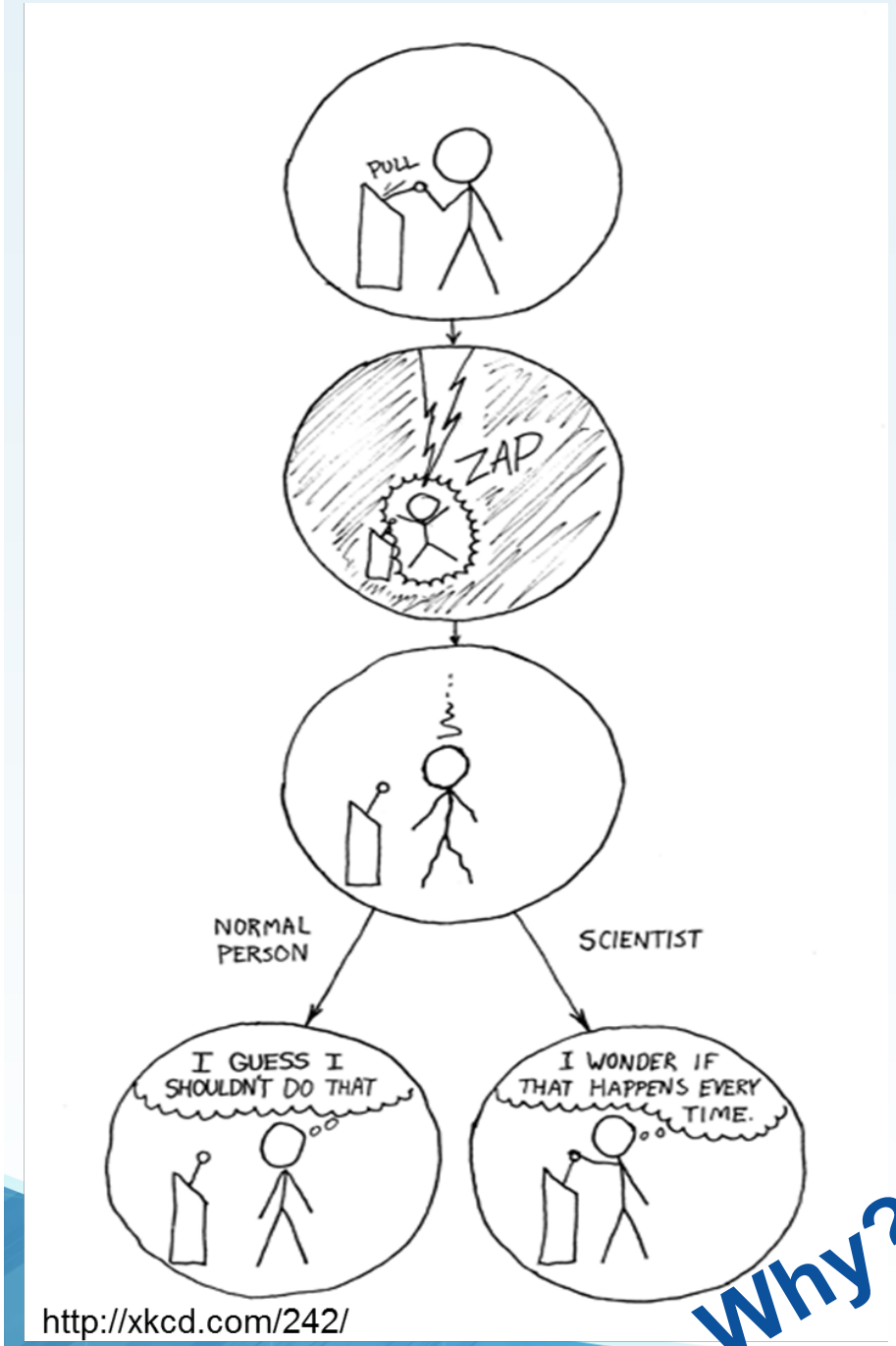
- In the meantime, there's JOSS
- A developer friendly journal for research software packages
- "If you've already licensed your code and have good documentation then we expect that it should take **less than an hour** to prepare and submit your paper to JOSS"
- Everything is open:
  - Submitted/published paper: http://joss.theoj.org
  - Code itself: where is up to the author(s)
  - Reviews & process: https://github.com/openjournals/joss-reviews
  - Code for the journal itself: https://github.com/openjournals/joss
- Zenodo archives JOSS papers and issues DOIs
- First paper submitted May 4, 2016
  - As of 18 January: 62 accepted papers, 21 under review, 17 submitted (pre-review)

Software Citation

Software Reproducibility



National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign



http://xkcd.com/242/

Why?

repeat — Defend

same experiment,
same set up, same lab

replicate — Certify

same experiment,
same set up, independent lab

reproduce — Compare

variations on experiment,
on set up, independent labs

reuse — Transfer

different experiment

Adapted from Carole Goble: http://www.slideshare.net/carolegoble/what-is-reproducibility-gobleclean

NCSA

Reproducibility

- Problems:
  - Scientific culture: reproducibility is important at a high level, but not in specific cases
    - Like Mark Twain's definition of classic books: those that people praise but don't read
    - No incentives or practices that translate the high level concept of reproducibility into actions that support actual reproducibility
  - Reproducibility can be hard due to a unique situation
    - Data can be taken with a unique instrument, transient data
    - Unique computer system was used
  - Given limited resources, reproducibility is less important than new research
    - A computer run that took months is unlikely to be repeated, because generating a new result is seen as a better use of the computing resources than reproducing the old result

Software reproducibility

- Technical concerns
  - Software collapse (coined by Konrad Hinsen[1]): the fact that software stops working eventually if is not actively maintained
  - Software stacks used in computational science have a nearly universal multi-layer structure
    - Project-specific software: whatever it takes to do a computation using software building blocks from the lower three levels: scripts, workflows, computational notebooks, small special-purpose libraries and utilities
    - Discipline-specific research software: tools and libraries that implement models and methods which are developed and used by research communities
    - Scientific infrastructure: libraries and utilities used for research in many different disciplines, such as LAPACK, NumPy, or Gnuplot
    - Non-scientific infrastructure: operating systems, compilers, and support code for I/O, user interfaces, etc.
  - Software in each layer builds on and depends on software in all layers below it
  - Any changes in any lower layer can cause it to collapse

[1]http://blog.khinsen.net/posts/2017/01/13/sustainable-software-and-reproducible-research-dealing-with-software-collapse/

Software reproducibility

- Project-specific software (developed by scientists)
- Discipline-specific research software (developed by scientists & developers)
- Scientific infrastructure (developed by developers)
- Non-scientific infrastructure (developed by developers)

- Where to address reproducibility?
- Hinsen[1]: Just addressing project-specific software isn't enough to solve software collapse; enemy is changes in the foundations
- Options are similar for house owners facing the risk of earthquakes
  1. Accept that your house or software is short-lived; in case of collapse, start from scratch
  2. Whenever shaking foundations cause damage, do repair work before more serious collapse happens
  3. Make your house or software robust against perturbations from below
  4. Choose stable foundations
- Active projects choose 1 & 2
- We don't know how to do 3 (CS research needed, maybe new thinking[2])
- 4 is expensive & limits innovation in top layers (banks, military, NASA)

[1]http://blog.khinsen.net/posts/2017/01/13/sustainable-software-and-reproducible-research-dealing-with-software-collapse/
[2]Greg Wilson colloquium at NCSA: https://www.youtube.com/watch?v=vx0DUiv1Gvw

Software reproducibility

- Titus Brown[1]: “Archivability is a disaster in the software world”
  - Can’t we just use containers/VMs?
    - Docker itself isn’t robust[2]
    - VMs and docker images provide bitwise reproducibility, but aren’t scientifically useful; big black boxes don't really let you reuse or remix the contents
  - Options:
    - Run everything all the time
      - Hinsen’s option 2
      - Aka continuous analysis[3], similar to continuous integration
    - Acknowledge that exact repeatability has a half life of utility, and that this is OK
      - We don’t build houses to last forever…

[1]http://ivory.idyll.org/blog/2017-pof-software-archivability.html
[2]https://thehftguy.com/2016/11/01/docker-in-production-an-history-of-failure/
[3]Beaulieu-Jones & Greene, https://doi.org/10.1101/056473

NCSA

Software reproducibility

- Costs and benefits
  - If software-enabled results could be made reproducible at no cost, do so
  - If cost is 10x cost of original work, don't
  - At 1x, probably still no
  - What about at 0.1x (+10%)? Or +20%?
  - How to balance reproducibility cost vs lost opportunity of new research?
- Could be general question about the culture of science or specific question about any one experiment/result
- If not practical to make everything reproducible, could use cost:benefit ratio to determine what to do
- Also, could factor in confirmation depth (David Soergel[1]) as a measure of reproducible scientific research (i.e., benefit)
  - Shallowest form of confirmation: peer review
  - Deeper forms (different software -> different approach -> different inputs): more confidence in results

[1]http://davidsoergel.com/posts/confirmation-depth-as-a-measure-of-reproducible-scientific-research

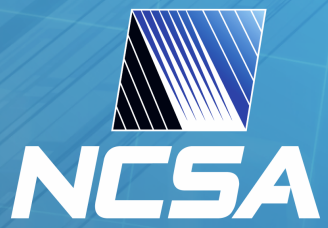
NCSA

Conclusions

- Software
  - Important today, essential tomorrow
- Credit
  - A known problem for papers, worse for software
- Citation
  - We know what to do (mostly), now need to do it
- Reproducibility
  - We think we know what we want to do
  - But don’t know how to do it
- What you can do
  - Cite the software you use, make it easy for others to cite the software you write (and see the “I solemnly pledge” manifesto – http://ceur-ws.org/Vol-1686/WSSSPE4_paper_15.pdf)
  - Work towards reproducibility as much as possible

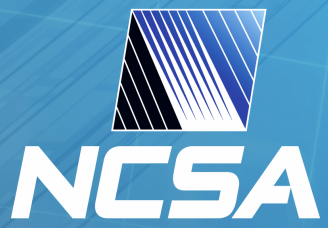
Citation and Reproducibility in Software

Daniel S. Katz
Assistant Director for Scientific Software & Applications, NCSA
Research Associate Professor, CS
Research Associate Professor, ECE
Research Associate Professor, iSchool
dskatz@illinois.edu, d.katz@ieee.org, @danielskatz



National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign

Additional Software Citation Principle Slides: discussion topics



National Center for Supercomputing Applications
University of Illinois at Urbana–Champaign

Discussion: What to cite

- Importance principle: "…**authors should cite the appropriate set of software products just as they cite the appropriate set of papers**"
- What software to cite decided by author(s) of product, in context of community norms and practices

- POWL: "Do not cite standard office software (e.g. Word, Excel) or programming languages. Provide references only for specialized software."
- i.e., if using different software could produce different data or results, then the software used should be cited

Purdue Online Writing Lab. Reference List: Electronic Sources (Web Publications). https://owl.english.purdue. edu/owl/resource/560/10/, 2015.

Discussion: What to cite (citation vs provenance & reproducibility)

- Provenance/reproducibility requirements > citation requirements
- Citation: software important to research outcome
- Provenance: all steps (including software) in research
- For data research product, provenance data includes all cited software, not vice versa
- Software citation principles cover minimal needs for software citation for software identification
- Provenance & reproducibility may need more metadata

Discussion: Software papers

- Goal: Software should be cited
- Practice: Papers about software ("software papers") are published and cited
- Importance principle (1) and other discussion: **The software itself should be cited on the same basis as any other research product; authors should cite the appropriate set of software products**
- Ok to cite software paper too, if it contains results (performance, validation, etc.) that are important to the work
- If the software authors ask users to cite software paper, can do so, *in addition to citing to the software*

Discussion: Derived software

- Imagine Code A is derived from Code B, and a paper uses and cites Code A
- Should the paper also cite Code B?
- No, any research builds on other research
- Each research product just cites those products that it directly builds on
- Together, this give credit and knowledge chains
- Science historians study these chains
- More automated analyses may also develop, such as transitive credit

Discussion: Software peer review

- Important issue for software in science
- Probably out-of-scope in citation discussion
- Goal of software citation is to identify software that has been used in a scholarly product
  - Whether or not that software has been peer-reviewed is irrelevant
- Possible exception: if peer-review status of software is part of software metadata
- Working group opinion: not part of the minimal metadata needed to identify the software

Discussion: Citations in text

- Each publisher/publication has a style it prefers
  - e.g., AMS, APA, Chicago, MLA
- Examples for software using these styles published by Lipson
- Citations typically sent to publishers as text formatted in that citation style, not as structured metadata
- Recommendation: **text citation styles should support:**
  - **a) a label indicating that this is software, e.g. [Computer program]**
  - **b) support for version information, e.g. Version 1.8.7**

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

NCSA

Discussion: Citation limits

- Software citation principles
- –> more software citations in scholarly products
- –> more overall citations
- Some journals have strict limits on
  - Number of citations
  - Number of pages (including references)
- Recommendations to publishers:
  - **Add specific instructions regarding software citations to author guidelines to not disincentivize software citation**
  - **Don’t include references in content counted against page limits**

Discussion: Unique identification

- **Recommend DOIs for identification of published software**
- However, identifier can point to
  1. a specific version of a piece of software
  2. the piece of software (all versions of the software)
  3. the latest version of a piece of software
- One piece of software may have identifiers of all 3 types
- And maybe 1+ software papers, each with identifiers
- Use cases:
  - Cite a specific version
  - Cite the software in general
  - Link multiple releases together, to understanding all citations

Discussion: Unique identification (cont.)

- Principles intended to apply at all levels
- To all identifiers types, e.g., DOIs, RRIDs, ARKS, etc.
- Though again: **recommend when possible use DOIs that identify specific versions of source code**
- RRIDs developed by the FORCE11 Resource Identification Initiative
  - Discussed for use to identify software packages (not specific versions)
  - FORCE11 Resource Identification Technical Specifications Working Group says “Information resources like software are better suited to the Software Citation WG”
  - Currently no consensus on RRIDs for software

Discussion: Types of software

- Principles and discussion generally focus on software as source code
- But some software is only available as an executable, a container, or a service
- Principles intended to apply to all these forms of software
- Implementation of principles will differ by software type
- **When software exists as both source code and another type, cite the source code**

Discussion: Access to software

- Accessibility principle: “software citations should permit and facilitate access to the software itself”
- Metadata should provide access information
- Free software: metadata includes UID that resolves to URL to specific version of software
- Commercial software: metadata provides information on how to access the specific software
  - E.g., company’s product number, URL to buy the software
- If software isn’t available now, it still should be cited along with information about how it was accessed
- Metadata should persist, even when software doesn’t

Discussion: Identifier resolves to …

- Identifier that points directly to software (e.g., GitHub repo) satisfies Unique Identification (3), Accessibility (5), and Specificity (6), but not Persistence (4)
- Recommend that **identifier should resolve to persistent landing page that contains metadata and link to the software itself, rather than directly to source code**
- Ensures longevity of software metadata, even beyond software lifespan
- Point to figshare, Zenodo, etc., not GitHub

Last Problem

proxies for success in the U.S. economics labor market (tenure at highly ranked schools, fellowship in the Econometric Society, and to a lesser extent, Nobel Prize and Clark Medal winnings) **are correlated with surname initials**, favoring economists with surname initials earlier in the alphabet. These patterns persist even when controlling for country of origin, ethnicity, and religion. We suspect that **these effects are related to the existing norm** in economics **prescribing alphabetical ordering of authors' credits**. Indeed, there is no significant correlation between surname initials and tenure at departments of psychology, where authors are credited roughly according to their intellectual contribution. The economics market participants seem to react to this phenomenon. Analyzing publications in the top economics journals since 1980, we note two consistent patterns: authors with higher surname initials are significantly less likely to participate in projects with more than three authors and significantly more likely to write papers in which the order of credits is non-alphabetical.

- Liran Einav and Leeat Yariv (2006), What's in a Surname? The Effects of Surname Initials on Academic Success (or free pdf here)