How to make software fit in research citation graphs

Stephan Druskat
German Aerospace Center (DLR), Intelligent and Distributed Systems &
Humboldt-Universität zu Berlin, Dept. of Computer Science

RSEConUK 2019, Birmingham
18 September 2019

stephan.druskat@dlr.de
@stdruskat

Aims

- **My aim:** Help integrate software in a more complete and fairer system of citation.
- **How:** Apply software engineering methods to create citation graphs for software and their dependencies.
- **This talk:** Show how modeling the output of citation can help understand the requirements for my work, and for the implementation of software citation.

Software, citation, and everything

- Software is a research product! [1, 2]
- Software citation principles! [3]
- Now what? Implementation! [4]

From the Editors

Better Software, Better Research

Carole Goble • *University of Manchester, UK*

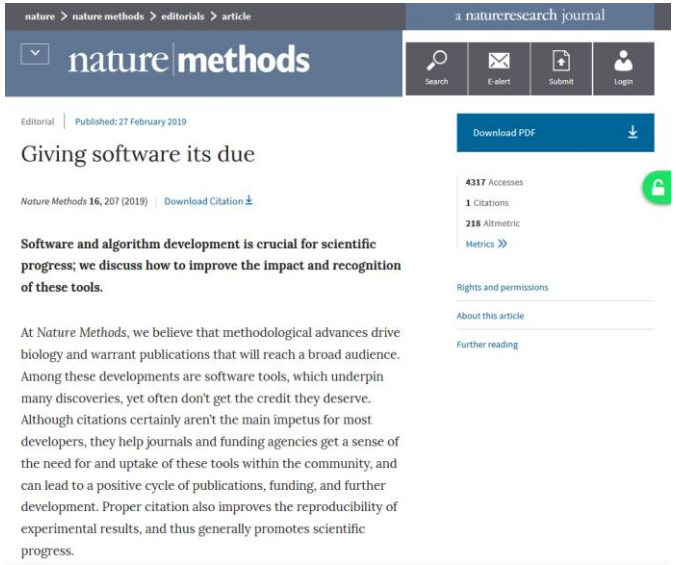
We know that modern scientific research isn't possible without software, from short, thrown-together temporary scripts and the abundance of complex spreadsheets, through to the huge software enterprises behind international efforts such as the Large Hadron Collidor and the Square Kilometer Array. And it's not just research based on simulations and computational methods. Data-driven science (the so-called "fourth paradigm"[1]) wouldn't be possible without software to access and manipulate that data, and our ability to generate insights depends on software platforms.

My personal experience suggests little difference between the size of the research community and the size of the "research software community." Of 2,000 scientists Jo Hannay and colleagues surveyed online,[2] 91 percent

Better Training, Better Production, Better Software

If your software is incorrect, so will be your science.[3] Mistakes in software happen to the best: Geoffrey Chang's discovery of the bug in his software led to his retraction of three *Science* papers.[4] We should admire him for his honest stance, and he's a better scientist for it. Many others don't even know they're wrong, or if they do, keep quiet. We can improve software quality at two major points in the research life cycle: while it's being produced and when its outcomes are subject to peer review.

Scientific software comes largely from two groups: highly trained software developers who work with research groups and are employed – typically – as postdoctoral researchers or research institute staff; and researchers self-taught in software development. Worryingly, in Hannay's



nature > nature methods > editorials > article

a natureresearch journal

nature methods

Search | E-alert | Submit | Login

Editorial | Published: 27 February 2019

Giving software its due

Nature Methods **16**, 207 (2019) | Download Citation

Download PDF

4317 Accesses
1 Citations
218 Altmetric
Metrics

Rights and permissions
About this article
Further reading

Software and algorithm development is crucial for scientific progress; we discuss how to improve the impact and recognition of these tools.

At *Nature Methods*, we believe that methodological advances drive biology and warrant publications that will reach a broad audience. Among these developments are software tools, which underpin many discoveries, yet often don't get the credit they deserve. Although citations certainly aren't the main impetus for most developers, they help journals and funding agencies get a sense of the need for and uptake of these tools within the community, and can lead to a positive cycle of publications, funding, and further development. Proper citation also improves the reproducibility of experimental results, and thus generally promotes scientific progress.

PeerJ Computer Science | About

Get ready. PeerJ to offer only peer-reviewed open access journal publishing. No more preprints. Starting Oct 1st >

PEER-REVIEWED

Software citation principles

Research article | Digital Libraries | Software Engineering

Arfon M. Smith[1], Daniel S. Katz[2], Kyle E. Niemeyer[3], FORCE11 Software Citation Working Group

September 19, 2016

Note that a Preprint of this article also exists, first published June 27, 2016.

View 151 tweets | Related research | Download | Follow | Share

Author and article information

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

Cornell University

We gratefully acknowledge support from the Simons Foundation and member institutions.

arXiv.org > cs > arXiv:1905.08674

Search... | All fields | Search | Help | Advanced Search

Computer Science > Computers and Society

Software Citation Implementation Challenges

Daniel S. Katz, Daina Bouquin, Neil P. Chue Hong, Jessica Hausman, Catherine Jones, Daniel Chivvis, Tim Clark, Mercé Crosas, Stephan Druskat, Martin Fenner, Tom Gillespie, Alejandra Gonzalez-Beltran, Morane Gruenpeter, Ted Habermann, Robert Haines, Melissa Harrison, Edwin Henneken, Lorraine Hwang, Matthew B. Jones, Alastair A. Kelly, David N. Kennedy, Katrin Leinweber, Fernando Rios, Carly B. Robinson, Ilian Todorov, Mingfang Wu, Qian Zhang

(Submitted on 21 May 2019)

The main output of the FORCE11 Software Citation working group (this https URL) was a paper on software citation principles (this https URL) published in September 2016. This paper laid out a set of six high-level principles for software citation (importance, credit and attribution, unique identification, persistence, accessibility, and specificity) and discussed how they could be used to implement software citation in the scholarly community. In a series of talks and other activities, we have promoted software citation using these increasingly accepted principles. At the time the initial paper was published, we also provided guidance and examples on how to make software citable, though we now realize there are unresolved problems with that guidance. The purpose of this document is to provide an explanation of current issues impacting scholarly attribution of research software, organize updated implementation guidance, and identify where best practices and solutions are still needed.

Subjects: Computers and Society (cs.CY); Digital Libraries (cs.DL)
Cite as: arXiv:1905.08674 [cs.CY]
(or arXiv:1905.08674v1 [cs.CY] for this version)

Bibliographic data
Select data provider: Semantic Scholar [Disable Bibex (What is Bibex?)]

References (0) | Citations (1)

Download:
• PDF only

Current browse context: cs.CY
< prev | next >
new | recent | 1905

Change to browse by: cs | cs.DL

References & Citations
• NASA ADS

DBLP - CS Bibliography
listing | bibtex
Daniel S. Katz
Daina Bouquin
Neil P. Chue Hong
Jessica Hausman
Catherine Jones

Software Citation Im...
• Daniel S. Katz
• Daina Bouquin
• Neil P. Chue Hong
• Jessica Hausman
• Catherine Jones
• Daniel Chivvis
• Tim Clark
• Mercé Crosas
• Stephan Druskat
• Martin Fenner

The citation system and its functions

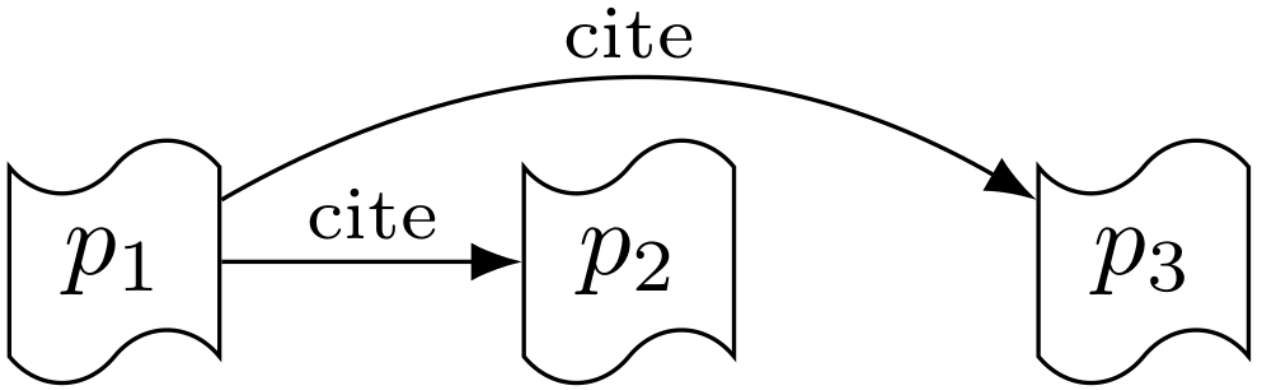
A sociotechnical system which provides

- Context
- Trust & authority
- Recognition of value, credit (for individuals and groups/entities)
- Compliance
- Discursivity
- Reproducibility

The citation system and its functions

A sociotechnical system which provides

- Context: Understand how knowledge was established and is used
- Trust & authority
- Recognition of value, credit (for individuals and groups/entities)
- Compliance
- Discursivity
- Reproducibility



The citation system and its functions

A sociotechnical system which provides

- Context: Understand how knowledge was established and is used
- Trust & authority: Trust in research and researchers, authority over research
- Recognition of value, credit (for individuals and groups/entities)
- Compliance
- Discursivity
- Reproducibility

Photo by Liane Metzler on Unsplash

The citation system and its functions

A sociotechnical system which provides

- Context: Understand how knowledge was established and is used
- Trust & authority: Trust in research and researchers, authority over research
- Recognition of value, credit (for individuals and groups/entities)
- Compliance
- Discursivity
- Reproducibility

Photo by Riccardo Annandale on Unsplash

The citation system and its functions

A sociotechnical system which provides

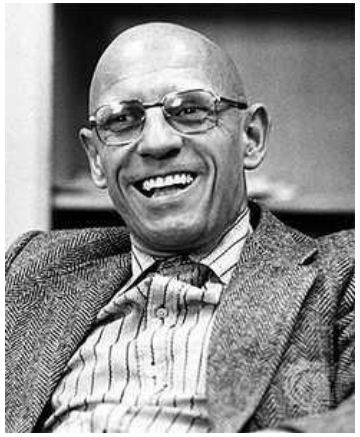
- Context: Understand how knowledge was established and is used
- Trust & authority: Trust in research and researchers, authority over research
- Recognition of value, credit (for individuals and groups/entities)
- Compliance: with established rules of scholarly practice
- Discursivity
- Reproducibility

Photo by Jessica To'oto'o on Unsplash

The citation system and its functions

A sociotechnical system which provides

- Context: Understand how knowledge was established and is used
- Trust & authority: Trust in research and researchers, authority over research
- Recognition of value, credit (for individuals and groups/entities)
- Compliance: with established rules of scholarly practice
- Discursivity: through enabling epistemic change (“re-writing of the past“)
- Reproducibility



The citation system and its functions

A sociotechnical system which provides

- Context: Understand how knowledge was established and is used
- Trust & authority: Trust in research and researchers, authority over research
- Recognition of value, credit (for individuals and groups/entities)
- Compliance: with established rules of scholarly practice
- Discursivity: through enabling epistemic change (“re-writing of the past“)
- Reproducibility: by providing provenance of research, i.e., “what was used“

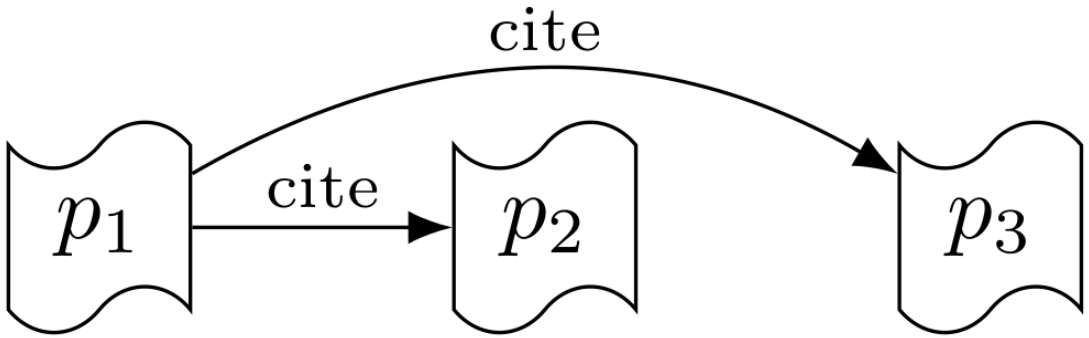
Preliminary conclusion

Software as a research product must be integrated in the citation system so that it can participate in all functions.

Modeling the citation system

Stage 1: Modeling the context function

- **Research citation graph:**
  A directed graph $G = (V, E)$
- V are vertices representing research products
- E are directed edges representing citation



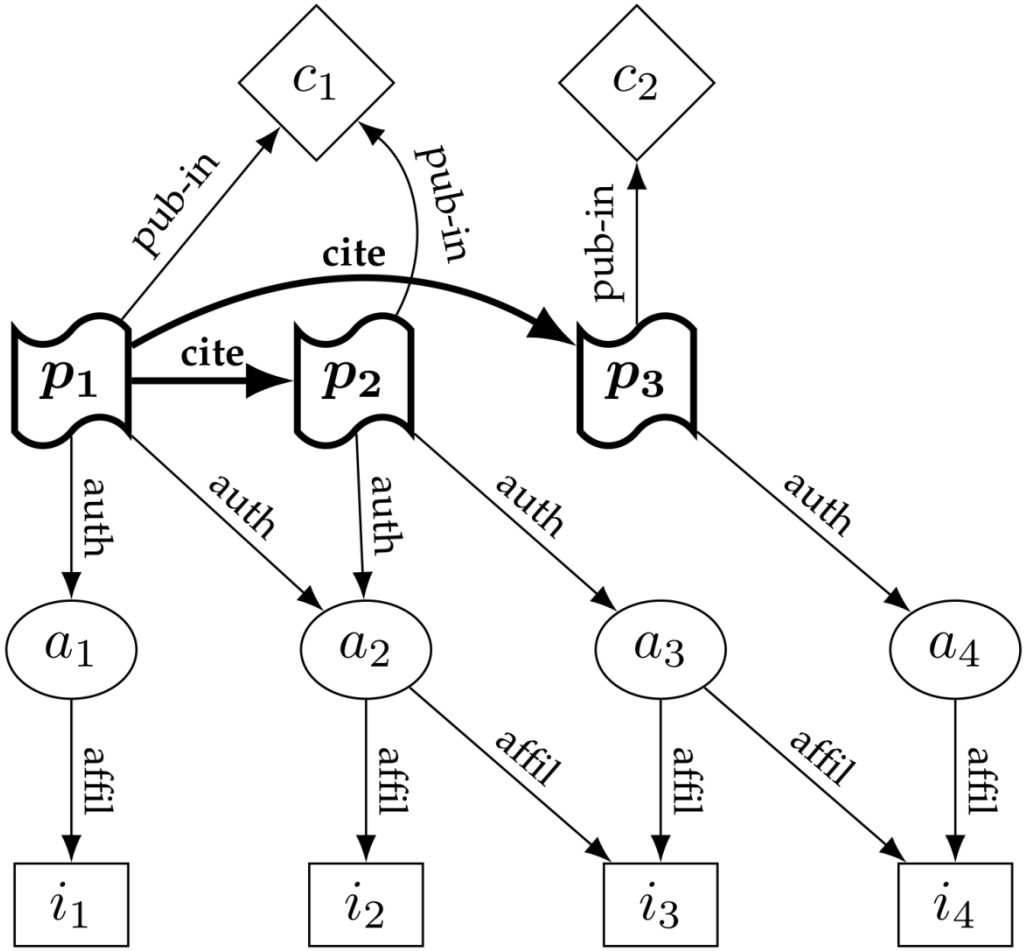
Modeling the citation system

Stage 2: Modeling the social functions: trust & authority, credit (and evaluation)

- Add:
  - Authors (and authorship relations)
  - Affiliations (and affiliation relations)
  - “Product containers“: journals, repositories, archives, etc. (and published-in relations)

$$G = (V, E)$$
$$\mathcal{V} = \{P, A, I, C\}$$
$$L : V \to \mathcal{V} \text{ to set}$$
$$L(v) = P \text{ when } v \in P \in \mathcal{V}, \text{ etc.}$$

“Pre-digitalization research citation graph”



Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take,
- its notion of finality and the relationships between its artifacts,
- the citability of its concepts,
- its dynamicity,
- the containment relationships between a product and its contributions,
- the roles which contribute to it.

Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take: source code vs. binary artifacts
- its notion of finality and the relationships between its artifacts,
- the citability of its concepts,
- its dynamicity,
- the containment relationships between a product and its contributions,
- the roles which contribute to it.

Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take: source code vs. binary artifacts
- its notion of finality and the relationships between its artifacts: versions & seriality (vs. finality)
- the citability of its concepts,
- its dynamicity,
- the containment relationships between a product and its contributions,
- the roles which contribute to it.

Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take: source code vs. binary artifacts
- its notion of finality and the relationships between its artifacts: versions & seriality (vs. finality)
- the citability of its concepts: software concepts are citable and cited
- its dynamicity,
- the containment relationships between a product and its contributions,
- the roles which contribute to it.

Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take: source code vs. binary artifacts
- its notion of finality and the relationships between its artifacts: versions & seriality (vs. finality)
- the citability of its concepts: software concepts are citable and cited
- its dynamicity: passive & functionally active (states, execution paths)
- the containment relationships between a product and its contributions,
- the roles which contribute to it.

Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take: source code vs. binary artifacts
- its notion of finality and the relationships between its artifacts: versions & seriality (vs. finality)
- the citability of its concepts: software concepts are citable and cited
- its dynamicity: passive & functionally active (states, execution paths)
- the containment relationships between a product and its contributions: dependencies are part of the product, at runtime at the latest
- the roles which contribute to it.

Modeling the citation system: factoring in software specifics

Software differs from textual research products in

- the form its artifacts can take: source code vs. binary artifacts
- its notion of finality and the relationships between its artifacts: versions & seriality (vs. finality)
- the citability of its concepts: software concepts are citable and cited
- its dynamicity: passive & functionally active (states, execution paths)
- the containment relationships between a product and its contributions: dependencies are part of the product, at runtime at the latest
- the roles which contribute to it: testers, designers, bug reporters, etc.

Modeling the citation system: requirements

- **Compliance:** Updated funders‘ guidelines/good scholarly practice guidelines require software citation

- **Reproducibility:** Complete and correct citation of used product, software should also cite products it builds on, including other software

- Model must include
  - Versions (and precedence relations)
  - Concepts (and realization relations)
  - Different contribution types

A model of research citation graphs that include software

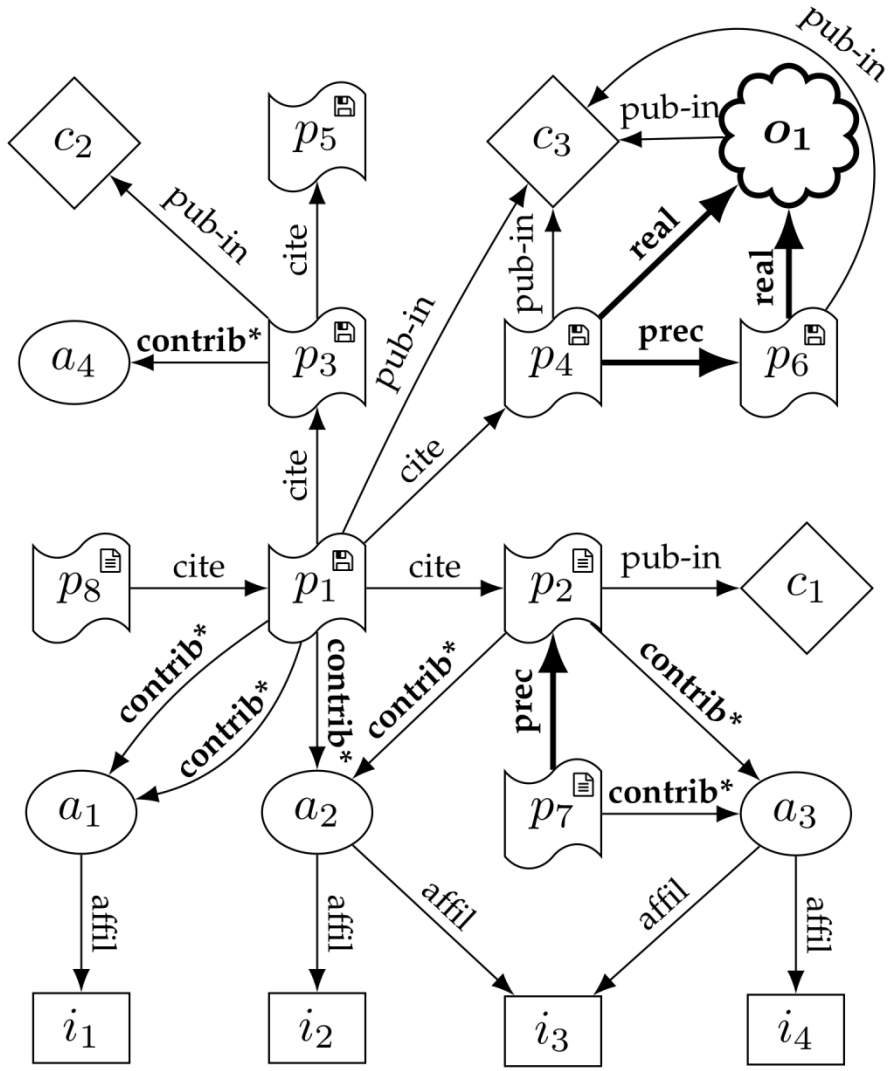
Stage 3: Model the missing functions

Compliance: software cites its references

Reproducibility: exact references are cited completely and correctly (allowing unique identification), from software and other products

Discursivity: potentially applicable to software citing its (non-software) references

$$G = (V, E)$$
$$\mathcal{V} = \{P, A, I, C, O\}$$
$$\mathcal{E} = \{E_{affil}, E_{cite}, E_{contrib}, E_{prec}, E_{pub\text{-}in}, E_{real}\}$$

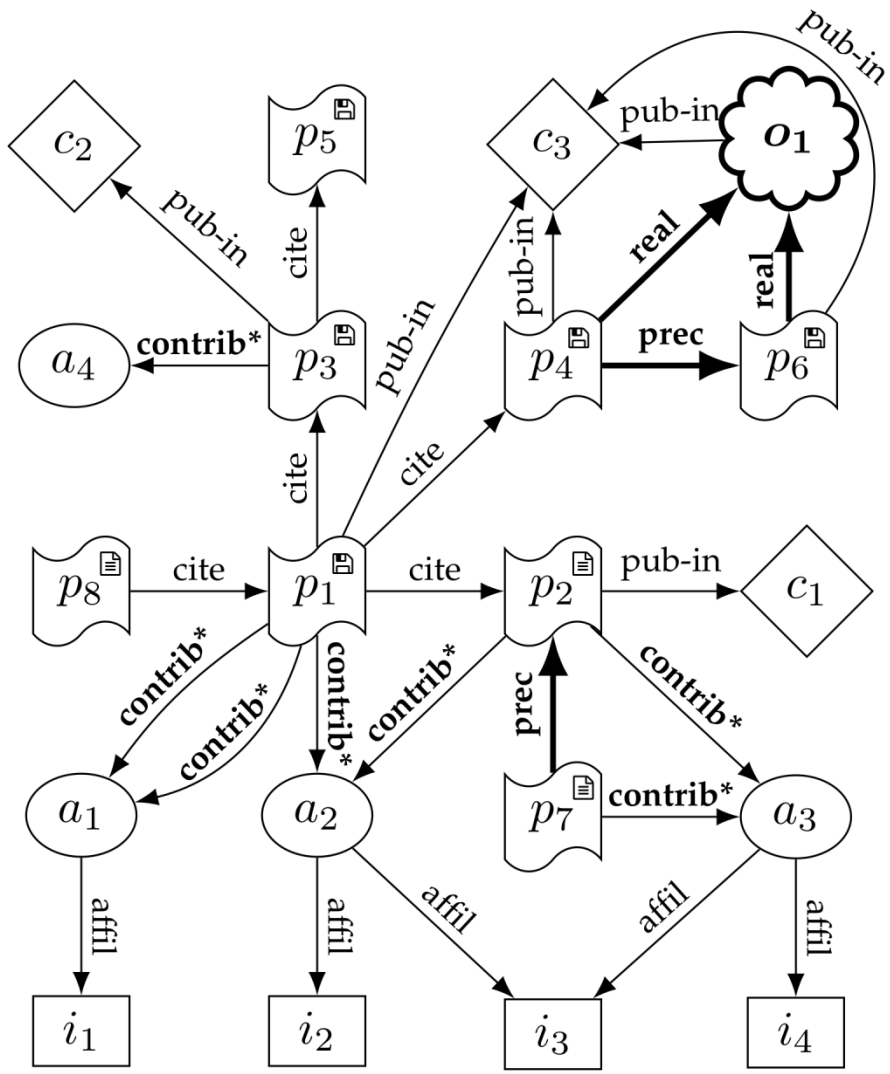


Research citation graphs: applications

- **The obvious stuff:**
  back-tracking context exploration, citation tracking, tracking of concept citation, self-citation analysis

- **The less obvious stuff:**
  Contribution role analysis, analysis of software development practices

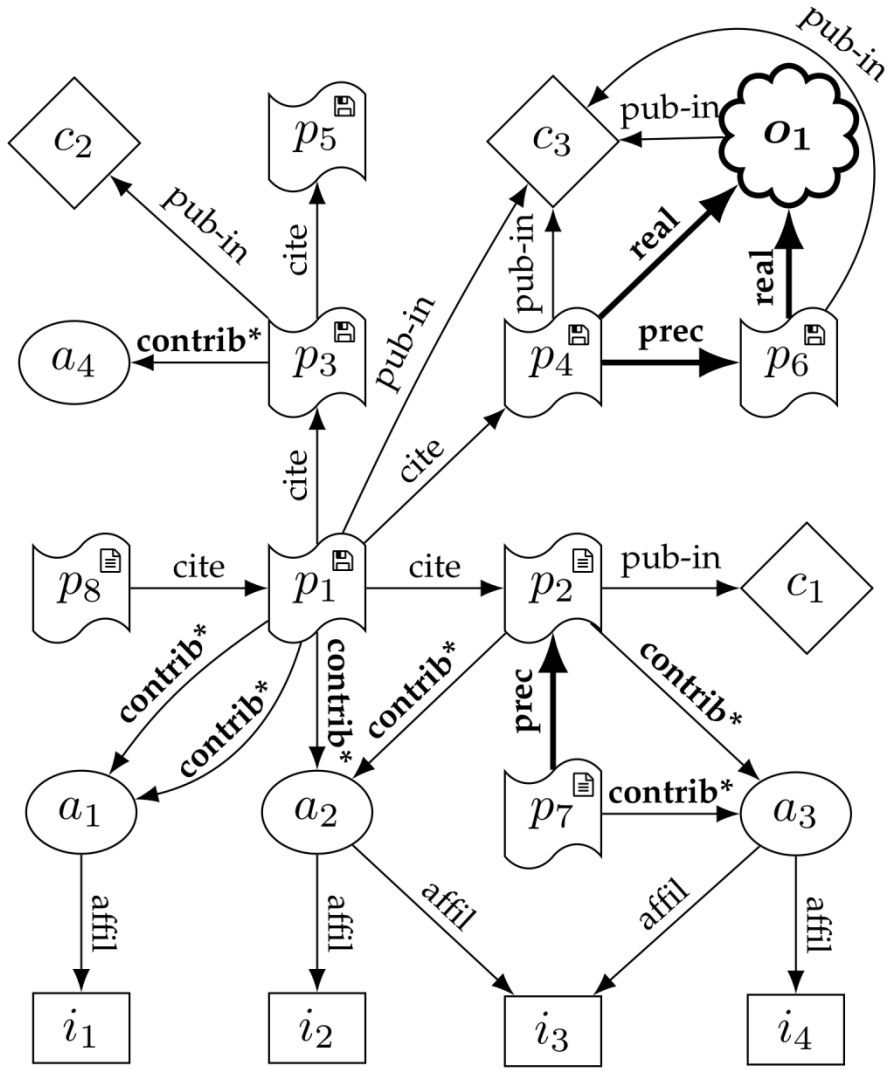
- **The cool/important/overdue stuff:**
  Credit for „hidden“ contributions to research, retrieval of transitive credit



Research citation graphs: transitive credit

- **Transitive credit [5]:**
  - Fractional credit for a research product is not distributed over authors alone, but also over referenced research products (credit map)
  - Credit maps for a product feed into the credit map for products that reference it
  - $p1$ cites $p2$, $p2$ is awarded 20% credit for $p1$ → $a3$ is awarded 50% credit for $p2$ → $a3$ is awarded 10% credit for $p1$

- **Calculating fractional credit:**
  - For contributing humans: manually, augmented
  - For contributing dependencies: programmatically
    - Software engineering: Call frequencies + complexity metrics
  - Enables evaluation methods for software dependencies



Instantiating research software citation graphs: challenges

- Cultural challenges:
  - Software as a research product (Importance principle, [3])
  - Practice of software citation
  - Unique identification of individuals and groups/entities
- Publication practice for research software:
  - Publication, formal publication
  - Unique identification
  - Incentives
- Metadata:
  - Provision, completeness, correctness, interoperability

Instantiating research software citation graphs: solutions

- Cultural challenges:
  - Software as a research product (Importance principle) – **Policy changes**
  - Practice of software citation – **CIA (Cite It Already!)**
  - Unique identification of individuals and groups/entities – **ORCID**
- Publication practice for research software:
  - Publication; formal publication – **GitHub-Zenodo, Software journals, Software Heritage; new roles for software journals? Business models?**
  - Unique identification – **DOIs**
  - Incentives – **Policy changes, evaluation practices**
- Metadata:
  - Provision, completeness, correctness, interoperability – **Citation File Format (CFF)** [6]**, CodeMeta** [7]

The role of RSEs

- **Cite It Already!** and lead by example
- **Provide citation metadata** in a CITATION.cff or codemeta.json file, help us make CFF better, build tooling to support conversion from CFF to CodeMeta
- **Publish your software** with a DOI
- **Tell your colleagues**, adapt peer reviewing practices to check for software citation

Thank you!

Preprint with details: https://arxiv.org/abs/1906.06141

Get in touch!

- stephan.druskat@dlr.de
- @stdruskat
- the coffee queue

Thanks: *Workshop on Sustainable Software Sustainability 2019 (The Hague) discussion group* - Neil Chue Hong, Gerard Coen, James Davenport, Leyla Garcia, Robert Haines, Catherine Jones, Adriaan Klinkenberg, Rachael Kotarski, Mateusz Kuzak, Brett Olivier, Esther Plomp, Shoaib Sufi, Stephanie van de Sandt, Bettine van Willigen.

References

[1] C. Goble, ‘Better Software, Better Research’, *IEEE Internet Computing*, vol. 18, no. 5, pp. 4–8, Sep. 2014.

[2] A. Doerr *et al.*, Eds., ‘Giving software its due’, *Nat Methods*, vol. 16, no. 3, pp. 207–207, Mar. 2019 [Online]. Available: https://doi.org/10.1038/s41592-019-0350-x. [Accessed: 23-Aug-2019]

[3] A. M. Smith, D. S. Katz, K. E. Niemeyer, and FORCE11 Software Citation Working Group, ‘Software citation principles’, *PeerJ Comput. Sci.*, vol. 2, no. e86, 2016 [Online]. Available: https://doi.org/10.7717/peerj-cs.86

[4] D. S. Katz *et al.*, ‘Software Citation Implementation Challenges’, *arXiv:1905.08674 [cs]*, May 2019 [Online]. Available: http://arxiv.org/abs/1905.08674. [Accessed: 18-Jun-2019]

[5] D. Katz, ‘Transitive Credit as a Means to Address Social and Technological Concerns Stemming from Citation and Attribution of Digital Products’, *Journal of Open Research Software*, vol. 2, no. 1, p. e20, Jul. 2014 [Online]. Available: http://openresearchsoftware.metajnl.com/articles/10.5334/jors.be/. [Accessed: 08-May-2019]

[6] S. Druskat, N. Chue Hong, R. Haines, and J. Baker, ‘Citation File Format (CFF) - Specifications’, Aug. 2018 [Online]. Available: https://doi.org/10.5281/zenodo.1003149

[7] M. B. Jones *et al.*, *CodeMeta: an exchange schema for software metadata. Version 2.0*. 2017 [Online]. Available: https://doi.org/10.5063/schema/codemeta-2.0