CARLA 2022

LATIN AMERICA HIGH PERFORMANCE COMPUTING CONFERENCE

Sustaining Research Software: Why and How

Daniel S. Katz (d.katz@ieee.org, @danielskatz)
Chief Scientist, NCSA
Associate Research Professor, CS, ECE, iSchool
University of Illinois at Urbana Champaign

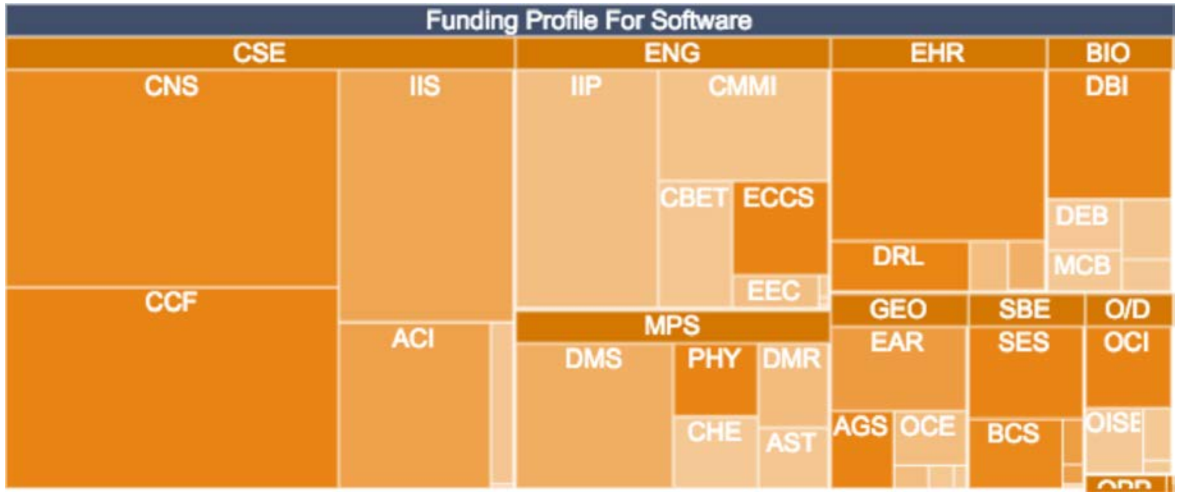
30 September 2022

ILLINOIS
NCSA | National Center for Supercomputing Applications

https://doi.org/10.5281/zenodo.7124920

Why do we care about research software?

- US NSF
  - 1995-2016: 18,592 awards totalling $9.6 billion with project abstracts that topically include "software"
  - ~20% of the overall NSF research budget
- US DOE
  - Of three ECP areas, most of two (application development & software) technology are research software
  - According to Paul Messina in 2017, "ECP is a 7-year project with a cost range of $3.5B–$5.7B"
- Digital Research Alliance of Canada
  - Three areas: advanced research computing, data management, and research software



Collected from http://www.dia2.org in 2017

Why do we care about research software?

- Surveys of UK academics at Russell Group Universities (2014) and members of (US) National Postdoctoral Research Association (2017):

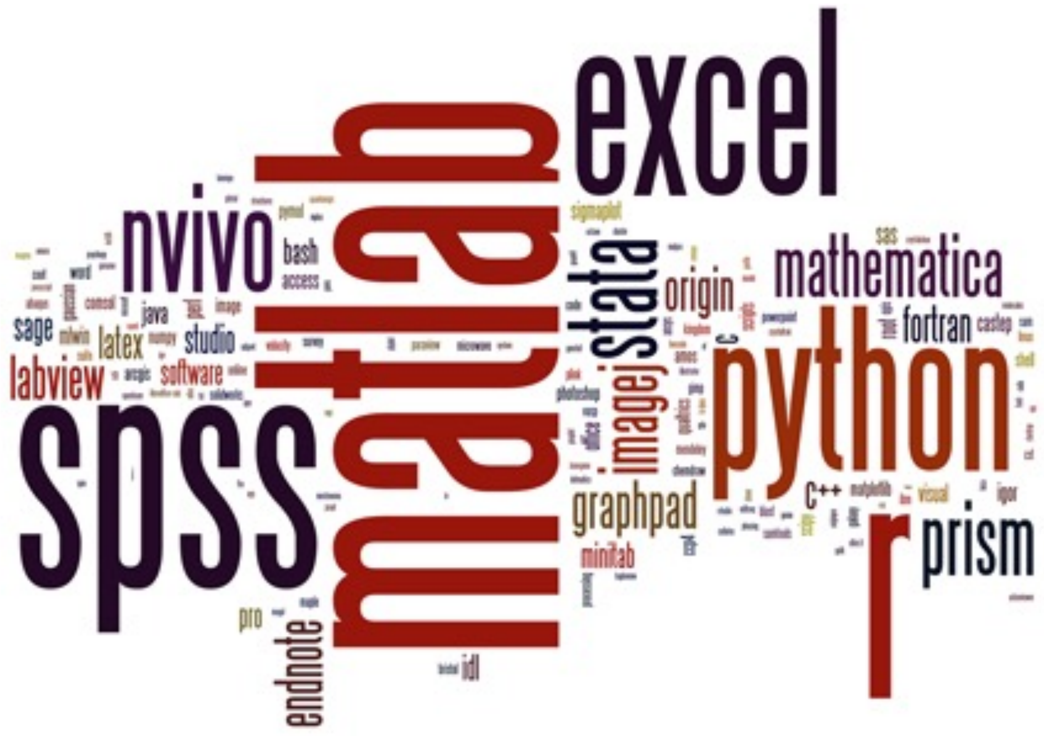
  - I use research software: 92% / 95% (UK/US)
  - My research would not be possible without software: 67% / 63%
  - My research would be possible but harder: 21% / 31%
  - I develop my own software: 56% / 28%

S. Hettrick; https://www.software.ac.uk/blog/2016-09-12-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers
S.J. Hettrick, et al.; 10.5281/zenodo.14809

U. Nangia and D. S. Katz; 10.6084/m9.figshare.5328442.v1

Why do we care about research software?

- 40 papers in Nature (Jan-Mar 2016)
  - 32 explicitly mentioned software
  - Average of 6.5 software tools/paper
  - Most of which were research software
- Top 100-cited papers:
  - 6 of top 13 are software papers
  - "… the vast majority describe experimental methods or software that have become essential in their fields."



Nangia and Katz; 10.1109/eScience.2017.78

"Top 100-cited papers of all time," Nature, 2014 10.1038/514550a

Research and research software vision

- All research software that can be is open — Open Science
- All research software is high-quality and robust — Software Engineering
- All research software is findable, accessible, and usable & used by others (for their own research) — FAIR
  - And is cited when it is used
- All contributors to research software are recognized for their work — Software Citation, JOSS
  - With good careers — RSE +
- All research software is sustained as long as it is useful — SSI, URSSI, ARDC
- All research is reproducible — Reproducibility

Note overlaps in terms of incentives and policies; all start with recognition of research software

Open Science (Collaboration)

- The free sharing of scientific ideas, methods, and results
- But not just science, rather wissenschaft (knowledge, scholarship, …)
- Initially via hand-written letters and books, mostly for other scientists
- Then more frequently via printed journals, expanding the audience
- Digitalization expanded opportunities for sharing, as well as what could be shared
- Democratization of research (public funding) and information sharing (BBS, WWW) expanded the community (at least the audience)
- Idea of knowledge as a common (societal) good
- "Open science" term documented in Science, July 1882, but having the sense of using the scientific method
- It seems to have become commonly used in the modern sense in the mid 1980s, perhaps when the AAAS Project on Secrecy and Openness in Science and Technology started in 1984

J. P. Tennant, …, D. S. Katz, …, "A tale of two 'opens': intersections between Free and Open Source Software and Open Scholarship," SocArXiv, 6 Mar. 2020. DOI: 10.31235/osf.io/2kxq8

Economics drives our lives (Competition)

- We live in a capitalistic society
- Economics drives our lives and careers
  - Where we work (hiring)
  - How we support ourselves (promotion)
  - How we get funding to do science (support, recognition)
  - Which science we do (what areas we think will lead to reward)
  - Which students we train or take advantage of (depending on your viewpoint)
- Economics: the science of allocating scarce resources to maximize the achievement of competing ends
  - Sometimes a false argument, some resources can be increased, e.g., digital

Paula Stephan, How Economics Shapes Science, Harvard University Press, 2015.

https://doi.org/10.5281/zenodo.7124920

Human behavior (Competition & Collaboration)

- Engagement: meaningful and valuable actions that produce a measurable result
- Engagement = Motivation + Support – Friction
  - Intrinsic motivation: self-fulfillment, altruism, satisfaction, accomplishment, pleasure of sharing, curiosity, real contribution to science
  - Extrinsic motivation: job, rewards, recognition, influence, knowledge, relationships, community membership
  - Support: ease, relevance, timeliness, value
  - Friction: technology, time, access, knowledge

Adapted from Joseph Porcelli

https://doi.org/10.5281/zenodo.7124920

Why not Open Science

- Sharing takes effort, immediate benefits go to others
- Mechanisms of sharing are new, not the way we work
- Metrics for evaluating products that can be shared are underdeveloped
- Intellectual property laws
- Commercial entities profit from restricting access
- Non-profit scientific societies are dependent on journal subscription fees [to support themselves/work they do]

C. Titus Brown, "What is open science?," 24 October 2016. http://ivory.idyll.org/blog/2016-what-is-open-science.html

Software collapse

- Software stops working eventually if is not actively maintained
- Structure of computational science software stacks:
  1. Project-specific software (developed by researchers): software to do a computation using building blocks from the lower levels: scripts, workflows, computational notebooks, small special-purpose libraries & utilities
  2. Discipline-specific software (developed by developers & researchers): tools & libraries that implement disciplinary models & methods
  3. Scientific infrastructure (developed by developers): libraries & utilities used for research in many disciplines
  4. Non-scientific infrastructure (developed by developers): operating systems, compilers, and support code for I/O, user interfaces, etc.
- Software builds & depends on software in all layers below it; any change below may cause collapse

K. Hinsen, "Dealing With Software Collapse," 2019. https://doi.org/10.1109/MCSE.2019.2900945

Research software

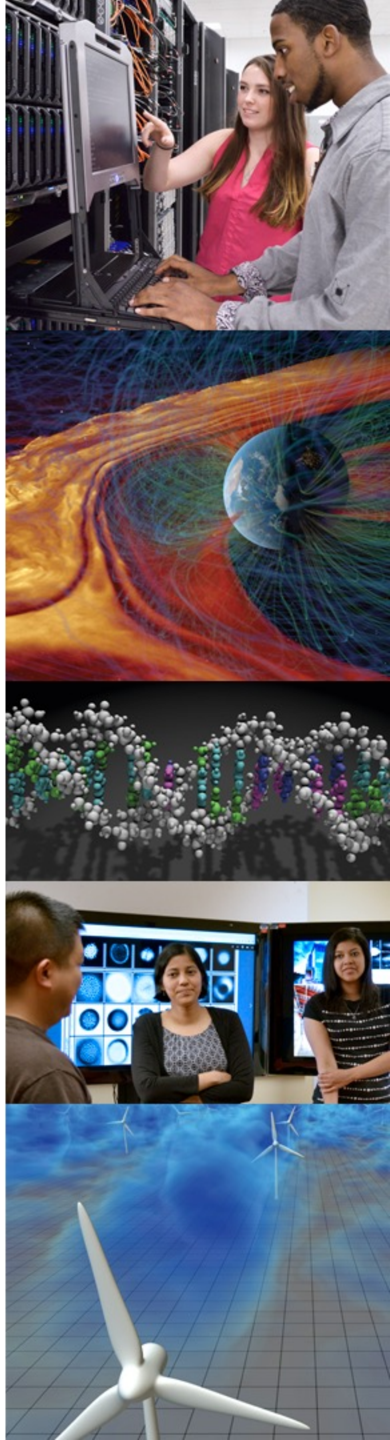
- Software developed and used for the purpose of research: to generate, process, analyze results within the scholarly process
- Increasingly essential in the research process
- But
  - Software will collapse if not maintained
  - Software bugs are found, new features are needed, new platforms arise
  - Software development and maintenance is human-intensive
  - Much software developed specifically for research, by researchers
  - Researchers know their disciplines, but often not software best practices
  - Researchers are not rewarded for software development and maintenance in academia
  - Developers don’t match the diversity of overall society or of user communities

Research software sustainability defined

Research software sustainability is the process of developing and maintaining software that continues to meet its purpose over time, which includes that the software adds new capabilities as needed by its users, responds to bugs and other problems that are discovered, and is ported to work with new versions of the underlying layers, including software as well as new hardware

https://doi.org/10.5281/zenodo.7124920

FAIR for Research Software



https://doi.org/10.5281/zenodo.7124920

The FAIR Principles

The FAIR Guiding Principles for scientific data management and stewardship

Mark D. Wilkinson, Michel Dumontier, [...] Barend Mons

Scientific Data **3**, Article number: 160018 (2016) | Cite this article

A set of principles, to ensure that data are shared in a way that enables and enhances reuse by humans and machines

Findable

F1. (Meta)data are assigned a globally unique and eternally persistent identifier.

F2. Data are described with rich metadata.

F3. (Meta)data are registered or indexed in a searchable resource.

F4. Metadata specify the data identifier.

Accessible

A1. (Meta)data are retrievable by their identifier using a standardized communications protocol.

- A1.1. The protocol is open, free, and universally implementable.
- A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
- A2. Metadata are accessible, even when the data are no longer available.

Interoperable

I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.

I2. (Meta)data use vocabularies that follow FAIR principles.

I3. (Meta)data include qualified references to other (meta)data.

Reusable

R1. (Meta)data have a plurality of accurate and relevant attributes.

- R1.1. (Meta)data are released with a clear and accessible data usage license.
- R1.2. (Meta)data are associated with their provenance.
- R1.3. (Meta)data meet domain-relevant community standards.

FAIR for non-data objects: some context

- FAIR Principles, at a high level, are intended to apply to all research objects; both those used in research and those that are research outputs
- Text in principles often includes "(Meta)data …"
  - Shorthand for "metadata and data …"
- Principles applied via dataset creators and repositories, collectively responsible for creating, annotating, indexing, preserving, sharing the datasets and their metadata
  - Assumes separate and sequential creator/publisher (repository) roles
- What about non-data objects?
  - While they can often be stored as data, they are not just data
- While high level goals (F, A, I, R) are mostly the same, the details and how they are implemented depend on
  - How objects are created and used
  - How/where the objects are stored and shared
  - How/where metadata is stored and indexed
- Work needed to define, then implement, then adopt principles

Need for FAIR for non-data objects

- FAIR Principles are intended to apply to all digital objects (Wilkinson et al. 2016)

Recommendation 5:

*Recognise that FAIR guidelines will require **translation for other digital objects** and support such efforts.*

2020: ‘Six Recommendations for Implementation of FAIR Practice’ (FAIR Practice Task Force EOSC, 2020)

Recommendation 2:

*Make sure **the specific nature of software** is recognized and not considered as “just data” particularly in the context of discussion about the notion of FAIR data.*

2019: Opportunity Note by French national Committee for Open Science's Free Software and Open Source Project Group (Clément-Fontaine, 2019)

- We focused on **adaptation and adoption** of the FAIR principles to research software

Software vs. data

- Software is data, but it's not just data
  - Software is executable, data is not
  - Data provides evidence, software provides a tool
  - Software is a creative work, scientific data are facts or observations
    - Different licensing and copyright practices
  - Software suffers from a different type of bit rot (collapse) than data
  - The lifetime of software is generally not as long as that of data
  - For open source, no natural sequential creator/publisher process & no natural publisher (repository)

D. S. Katz et al., "Software vs. data in the context of citation," PeerJ Preprints 4:e2630v1, 2016. https://doi.org/10.7287/peerj.preprints.2630v1

FAIR for Research Software (FAIR4RS)

- Working group defined consensus FAIR principles for research software
  - Led by Michelle Barker, Neil Chue Hong, Leyla Garcia, Morane Gruenpeter, Jennifer Harrow, Daniel S. Katz, Carlos Martinez, Paula A. Martinez, Fotis Psomopoulos

FORCE11
The Future of Research Communications and e-Scholarship

RDA
RESEARCH DATA ALLIANCE

<ReSA>
Research Software Alliance

https://doi.org/10.5281/zenodo.7124920

Defining Research Software

- **Research Software** includes source code files, algorithms, scripts, computational workflows and executables that were **created during the research process or for a research purpose**

- Additional software components (e.g., operating systems, libraries, dependencies, packages, scripts, etc.) that are used for research but were **not created during or with a clear research intent should be considered software in research** and not Research Software

- This differentiation may vary between disciplines

https://doi.org/10.5281/zenodo.5504016

FAIR4RS principles

Chue Hong, N. P., Katz, D. S., Barker, M., Lamprecht, A-L, Martinez, C., Psomopoulos, F. E., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., Honeyman, T., et al. (2022). FAIR Principles for Research Software version 1.0. (FAIR4RS Principles v1.0). Research Data Alliance. DOI: 10.15497/RDA00068

Findable: Software, and its associated metadata, is easy to find for both humans and machines.

F1. Software is assigned a globally unique and persistent identifier

- *F1.1. Different components of the software are assigned distinct identifiers representing different levels of granularity*
- *F1.2. Different versions of the same software are assigned distinct identifiers*

F2. Software is described with rich metadata

F3. Metadata clearly and explicitly include the identifier of the software they describe

F4. Metadata are FAIR and are searchable and indexable

Accessible: Software, and its metadata, is retrievable via standardized protocols.

A1. Software is retrievable by its identifier using a standardized communications protocol

- A1.1. The protocol is open, free, and universally implementable
- A1.2. The protocol allows for an authentication and authorization procedure, where necessary

A2. Metadata are accessible, even when the software is no longer available

Interoperable: Software interoperates with other software through exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.

I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards

I2. Software includes qualified references to other objects

Reusable: Software is both usable (it can be executed) and reusable (it can be understood, modified, built upon, or incorporated into other software).

R1. Software is described with a plurality of accurate and relevant attributes

- R1.1. Software is given a clear and accessible license
- R1.2. Software is associated with detailed provenance

R2. Software includes qualified references to other software

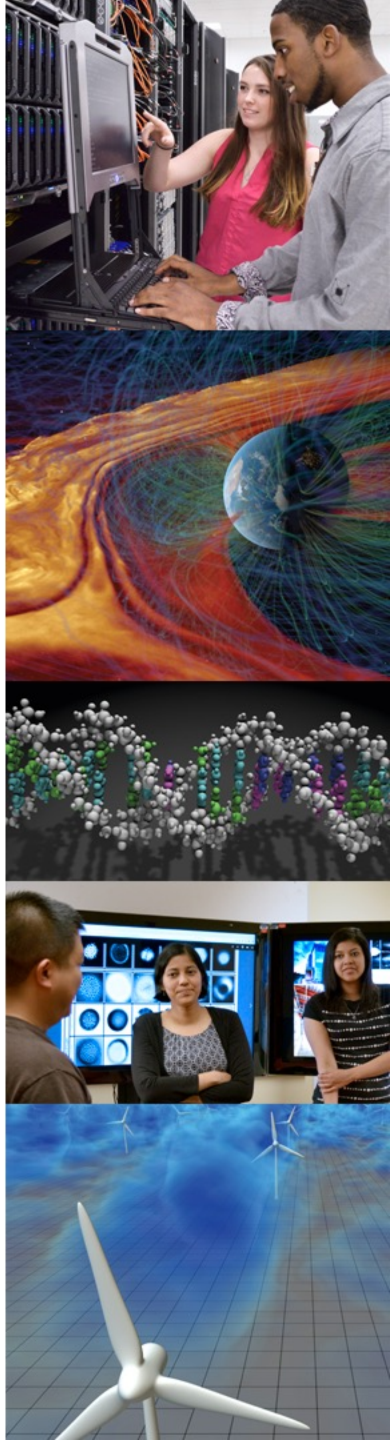
R3. Software meets domain-relevant community standards

Using FAIR4RS and what's next

- Governance (interpretation, future revisions) turned over to RDA Software Source Code Interest Group
- Survey of adoption guidelines: https://doi.org/10.5281/zenodo.6374598
- Study of adopting organizations: https://doi.org/10.5281/zenodo.6258366

- FAIR4RS exposes ecosystem gaps, particularly related to metadata, archiving, versions
  - Creator/publisher sequence doesn't typically apply
  - Where is metadata stored? (in code repository for open source?, for closed source?, in archival repository?, in registry?)
  - Where is code archived? (GitHub/Gitlab are not archival, registries are not archival, repositories? Software Heritage?)
  - Different use cases need specific version, latest version, all versions
- Lots of work beyond FAIR: quality, correctness, reproducibility, openness, …

Software citation &
Journal of Open Source Software (JOSS)



ILLINOIS NCSA

https://doi.org/10.5281/zenodo.7124920

FORCE11 Software Citation Working Group (2015-16)

- Documented differences between software and data; defined software citation challenges
  - Katz DS, Niemeyer KE, et al. (2016) Software vs. data in the context of citation. PeerJ Preprints 4:e2630v1. DOI: 10.7287/peerj.preprints.2630v1
  - Niemeyer KE, Smith AM, Katz DS. (2016) The challenge and promise of software citation for credit, identification, discovery, and reuse. ACM Journal of Data and Information Quality, 7(4):16. DOI: 10.1145/2968452
- Created software citation principles
  - 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

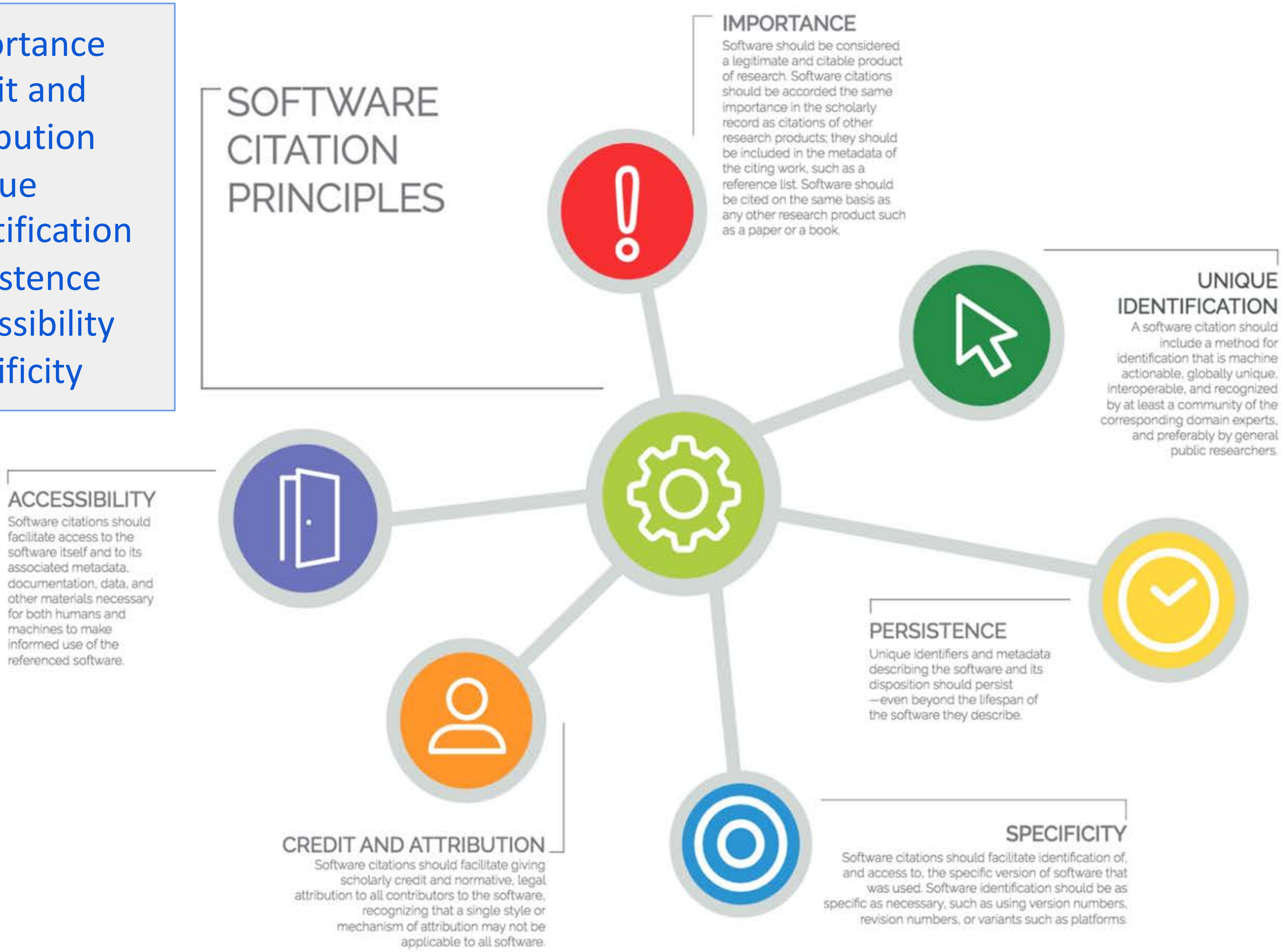
FORCE11 The Future of Research Communications and e-Scholarship

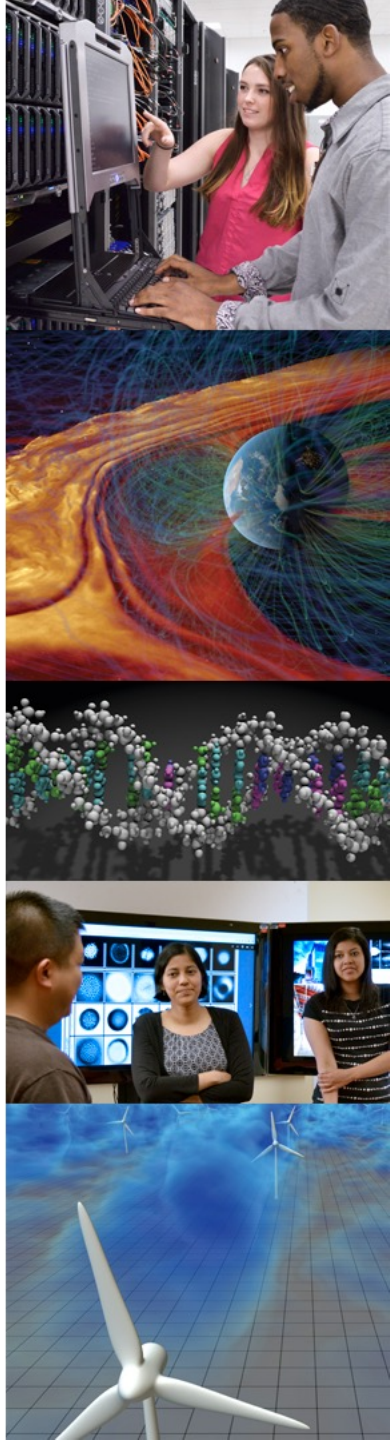
https://www.force11.org/group/software-citation-working-group

Co-Chairs: Arfon M. Smith, Daniel S. Katz, Kyle E. Niemeyer

Software is a critical part of modern research...

1. Importance
2. Credit and Attribution
3. Unique Identification
4. Persistence
5. Accessibility
6. Specificity





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

Image courtesy of DataCite

... yet there is little support for its acknowledgement and citation

FORCE11 Software Citation Implementation Working Group (2017-present)

- Initial goals:
  - Write out the “small amount” of detail needed to implement the principles
  - Coordinate research & other work going on in many areas
  - Work with communities to actually implement the principles
- Quickly realized “small amount” of detail wasn’t small, scattered progress wasn't sufficient, underlying challenges not being addressed
  - D. S. Katz, et al., "Software Citation Implementation Challenges", arXiv 1905.08674 [cs.CY], 2019.
  - Technical challenges include complexity of software types and identifiers, where to store metadata, …
  - Social challenges need groups that work on implementation in context (disciplinary communities, publishers, repositories & registries, indexers, funders, institutions) to come together and run pilots to establish norms

FORCE11 The Future of Research Communications and e-Scholarship

https://www.force11.org/group/software-citation-implementation-working-group

Co-Chairs: Neil Chue Hong, Martin Fenner, Daniel S. Katz

Responses to challenges (1)

- Guidance task force
  - For paper authors who want to cite software
    - N. P. Chue Hong, et al., "Software Citation Checklist for Authors," Zenodo, 15-Oct-2019. 10.5281/zenodo.3479198
  - For software developers who want to make their software citable
    - N. P. Chue Hong, et al., "Software Citation Checklist for Developers," Zenodo, 15-Oct-2019. 10.5281/zenodo.3482768
- CodeMeta task force
  - Following CodeMeta project
    - Aiming to understand metadata for software, not just for use in citation
    - Built a crosswalk of existing metadata standards for software
    - Then developed a CodeMeta standard to describe software based on these crosswalks
  - Updating the CodeMeta standard
  - Describing everything in CodeMeta using schema.org properties
  - Moving CodeMeta into a community group, with governance

Responses to challenges (2)

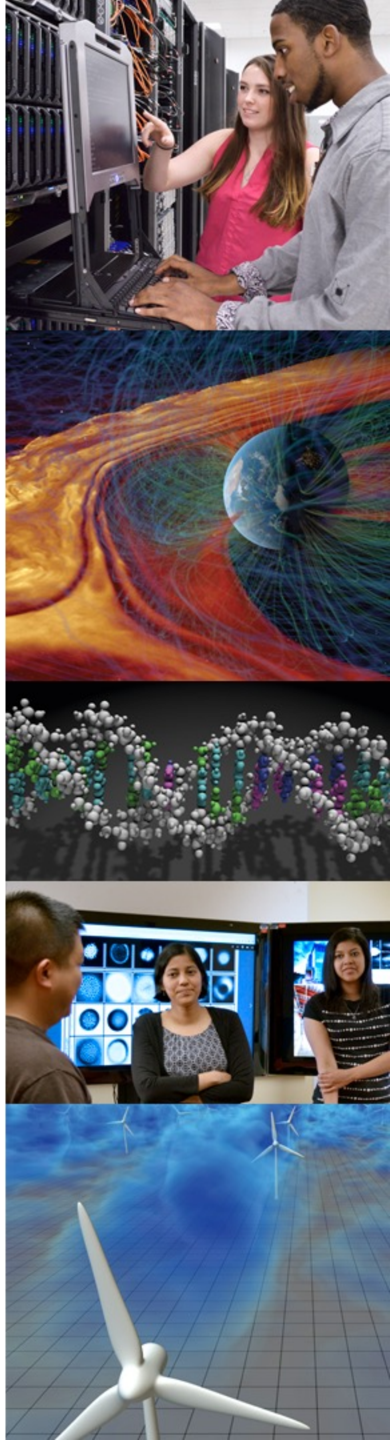
- Software Registries Task Force
  - Developed best practices document
    - Task Force on Best Practices for Software Registries, "Nine Best Practices for Research Software Registries and Repositories: A Concise Guide," 2020. arXiv 2012.13117
  - Community continuing in SciCodes: Consortium of scientific software registries and repositories, https://scicodes.net/
- Journals Task Force
  - Working with publishers to provide generic guidelines for journals and conferences to provide to authors
    - They then provide specific guidelines, with community-accepted language and examples
    - D. S. Katz, et al., "Recognizing the value of software: a software citation guide [version 2; peer review: 2 approved]," F1000Research 9:1257, 2021. 10.12688/f1000research.26932.2
    - Tracked by CHORUS in Software Citation Policy Index
  - Also working on publication processing
    - How citation information moves from author provides to internal publisher/contractor systems and then to indices
    - S. Stall, et al., "Journal Production Guidance for Data and Software Citations", in draft

Responses to challenges (3)

- Considered institutions task force
  - Institutions: places where people work
    - Universities, laboratories, industry, government, etc.
  - Want to affect policies and practices
    - How do they encourage software citation
    - How do they use software citation information in hiring & promotion
  - Collect and share examples
  - Help form communities
  - But insufficient interest from FORCE11 WG members
- Given progress to date, what else makes sense to do, and who can do it?
  - Recent IMLS-funded software citation workshop (led by Daina Bouquin) to assess and plan next steps
    - Report coming soon

https://doi.org/10.5281/zenodo.7124920

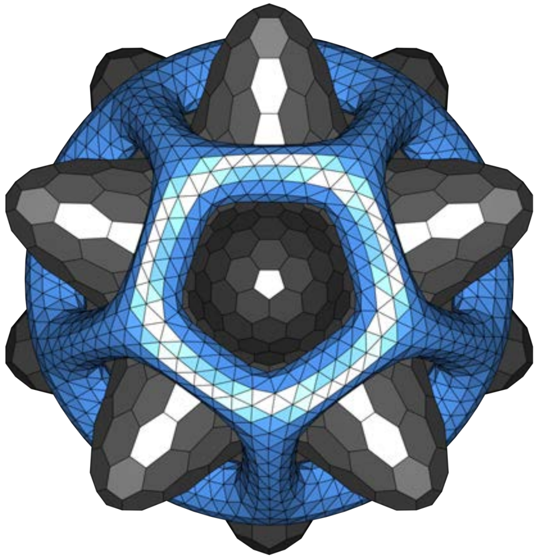
Journal of Open Source Software (JOSS)



https://doi.org/10.5281/zenodo.7124920

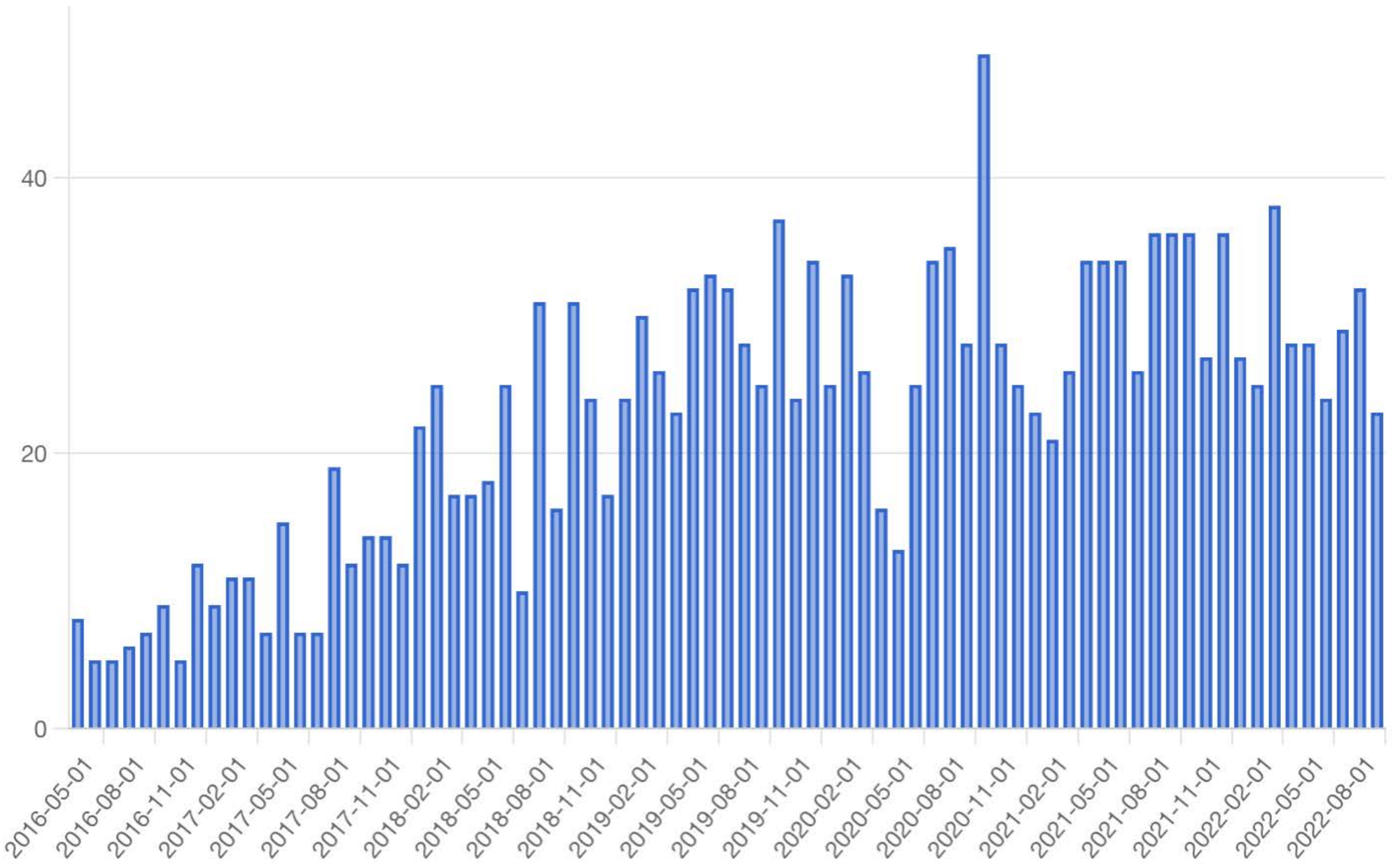
Journal of Open Source Software (JOSS)

- 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"
- Everything is open:
  - Submitted/published paper: https://joss.theoj.org
  - Code itself: where is up to the author(s)
  - Reviews & process: https://github.com/openjournals/joss-reviews
    - Adapted from rOpenSci
    - Expedited process for software already reviewed by rOpenSci & pyOpenSci
  - Code for the journal itself: https://github.com/openjournals/joss
    - Reused for Journal of Open Source Education (JOSE) and Proceedings of the JuliaCon Conferences



Journal of Open Source Software (JOSS)

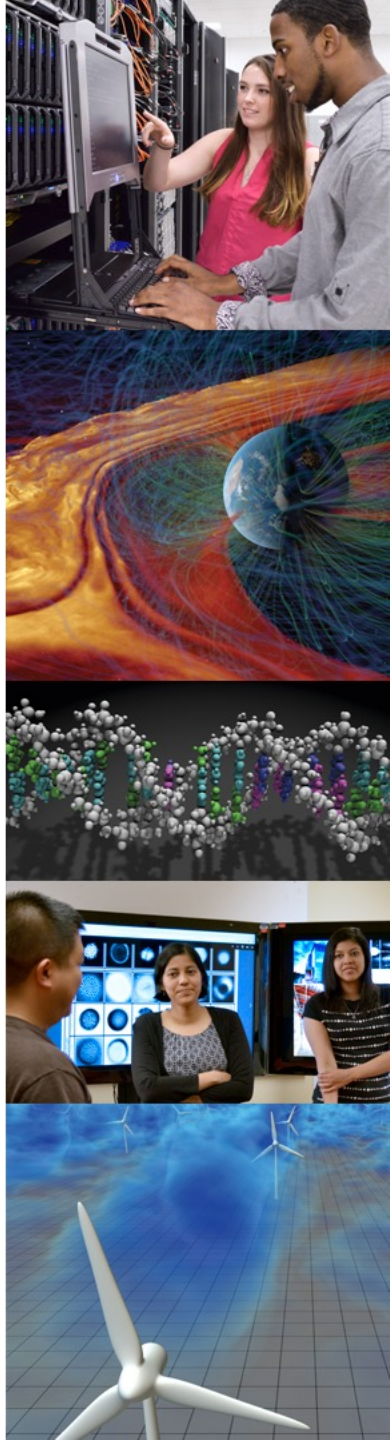
- JOSS papers archived, have DOIs, increasingly indexed
- First paper submitted 4 May 2016
- 31 May 2017: 111 accepted papers, 56 under review and pre-review
- 29 Sep 2022: 1756 accepted papers, 218 under review and pre-review
- Current publication rate: ~1 paper/day
- Editors:
  - 1 editor-in-chief and 11 editors at launch;
  - 1 EiC, 5 associate EiCs, 62 topic editors, 32 emeritus editors today
  - Moving to a track-based model soon



JOSS as a community

- Culture changes based on rules and incentives
- JOSS practices have influenced reviewers and developers in terms of what's good and what's minimally acceptable
  - Similar to rOpenSci's influence in the R community
  - Minimum criteria for successful review change over time as community best practices develop
    - E.g., packaging, testing/CI, documentation
- JOSS applies open source software practices to reviews: openness & collaboration
- JOSS provides rules, and at a high-level, tries to nudge incentives
  - Accepted software = accepted paper
- If software was cited directly, JOSS papers wouldn't be needed, but JOSS reviews and JOSS community would still have great value

Research Software Engineers



ILLINOIS NCSA

https://doi.org/10.5281/zenodo.7124920

Where does research software come from?

- As discussed, significant fraction developed in research
- From the start of computing
  - Software appears around 1948
  - Research software (weather) in early 1950s
  - Software engineering starting in late 1960s, mostly initially applied to operational software (operating system, NASA flights, etc.)
- Researchers (faculty) generally don't know good software practices
- Software engineers generally don't understand research context
- Students & postdocs generally don't know good software practices and don't stick around
- Some postdocs do stay, join staff (perhaps unofficially)
- Staff with research understanding and software engineering skills develop

https://doi.org/10.5281/zenodo.7124920

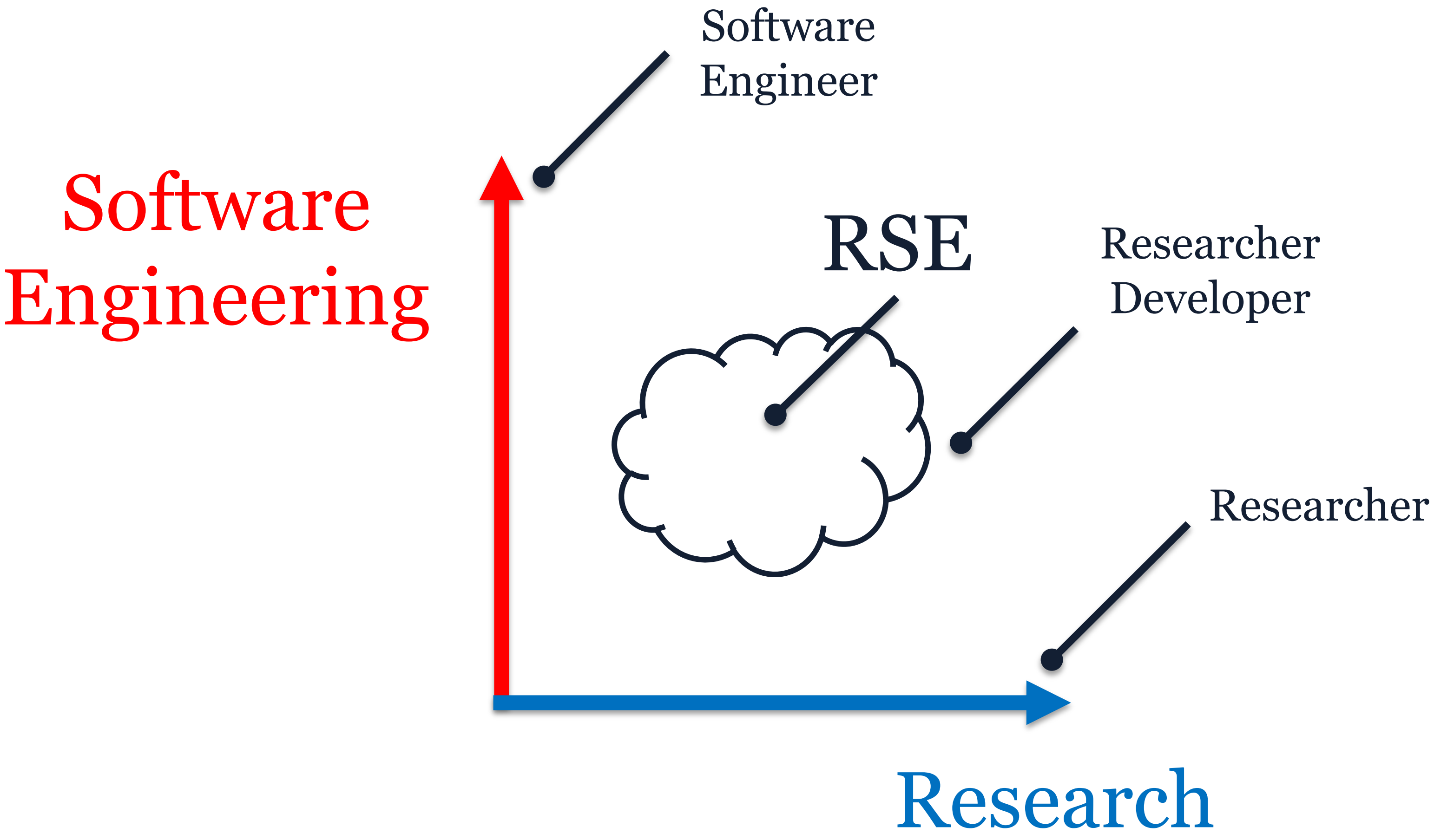
The Craftsperson and the Scholar

- Scholar: archetypical researcher driven to understand things to their fullest capability
  - Find intellectually-demanding problems
  - Curiosity-driven, work on a topic until understanding has been acquired, pass on that understanding through teaching
- Craftsperson: driven to create and leave behind an artifact that reifies their efforts in a field
  - Feels pain when things they make are fragile or ugly
  - Prefer to make things that explain themselves
  - Work requires patience, and pride in doing a job well
- Scientific software requires individuals who combine the best of both roles

http://www.software.ac.uk/blog/2012-11-09-craftsperson-and-scholar

https://doi.org/10.5281/zenodo.7124920

What is a Research Software Engineer?



https://danielskatzblog.wordpress.com/2019/07/12/super-rses-combining-research-and-service-in-three-dimensions-of-research-software-engineering/

https://doi.org/10.5281/zenodo.7124920

Collaborations Workshop 2012

- Lots of people already doing this work
- No common title
  - ✓ Research Software Engineer (RSE)
- No community
  - ✓ Associations/societies
- Not a profession
  - ✓ Career paths, structure

Credit: Simon Hettrick

10 years of RSEs

- Movement and term: Born in the UK
  - Late 2013 UKRSE Association forms with ~50 members
  - Now society, ~600 dues-paying members, ~4000-member community
- Also: Belgium, Germany, Netherlands, Nordic, Australia/New Zealand
- And US-RSE (https://us-rse.org), ~1300 members
- New associations now forming
  - Africa
  - Asia
- Associations work on local issues collectively, and can coordinate

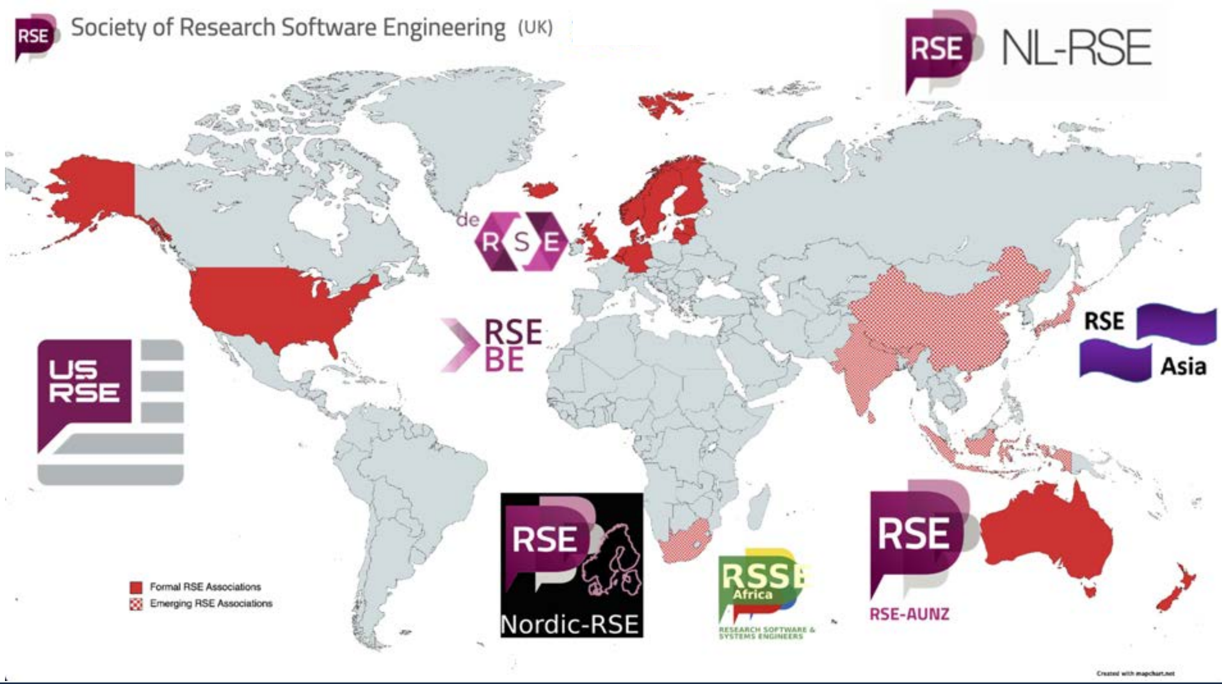


Image credit: Ian Cosden

ILLINOIS NCSA

https://doi.org/10.5281/zenodo.7124920

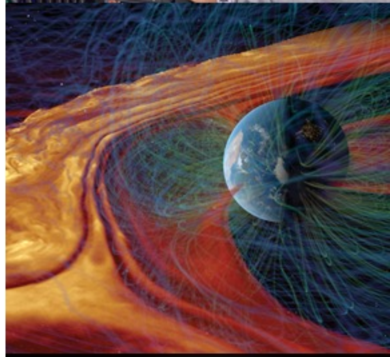
Advantages of RSEs

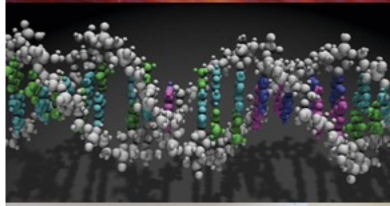
- Better software & more mature project management aspect/structure
  - Team organization, work organization, software organization, coding practices

  → better, more sustainable code & more efficient group coordination
- Institutional memory
  - Research software is becoming more important, complex, and costly, expertise is valuable
  - RSEs have longevity beyond individual projects and students
  - RSEs often mobile across domains, can translate knowledge/artifacts across communities
- Build community expertise
  - RSEs typically work on series of grant-funded (typically less than 1 RSE FTE) projects
    - Shorter duration than staff careers
  - RSE groups tend towards a form of “Matrix Management”
    - Principle Investigator axis: changes over time, funded by projects
    - RSE group manager axis: fixed, funded by overhead, tax on projects, institution, …

  → pool of experts/expertise, welcomes others with shared interests

Summary





https://doi.org/10.5281/zenodo.7124920

Conclusions

- Research software is increasingly important to research
- Open science and open source have an interwoven history; both seem to be moving forward
- FAIR (data) principles set out good goal: share data to enable and enhance reuse
  - Work is needed to apply this goal to research software, both open source and not, to fulfil the open science concepts
  - Principles have been created, with ~500 people involved & ~60 events
    - Chue Hong, et al. (2022). FAIR Principles for Research Software version 1.0. (FAIR4RS Principles v1.0). DOI: 10.15497/RDA00068
  - Work underway to create guidance, adopt principles, define governance
  - Next steps will be to create metrics and widen adoption
- Citation is the accepted mechanism for scholarly credit in academia
  - Software citation is starting to become accepted, with uptake from researchers, publishers, librarians, etc.
  - Software papers are partially a placeholder for software citation, but also have an important community function
- Research software Engineers (RSEs)
  - Recently recognized as a role, but a long history
  - Now building community & making case for formal career paths

What you can do

- Open science, open software, FAIR4RS, Citation, RSE are open communities, w/ significant overlap
  - Remember: culture of science is us – we can change it
  - You can join one or more and help!
- Support RSEs
  - Support software developers and maintainers
    - Consider how your organization does this
    - Can you change this?
  - E.g., see https://society-rse.org and http://us-rse.org
  - Is it time to start an RSE organization in Latin America, or in a single country?
- Promote software sharing and credit
  - When you are an author, cite the software you use
  - When you develop software, make it easy to cite
  - When you review, demand software be cited
  - Work to make your own software FAIR – follow the principles
- Work to make sure software work is included in hiring and promotion
- Overall, raise awareness of software as a key element of research

Credits

- Thanks to Arfon Smith and Kyle Niemeyer for co-leadership in FORCE11 Software Citation WG
- And Arfon Smith for JOSS leadership
- And Neil Chue Hong & Martin Fenner for co-leadership in FORCE11 Software Citation Implementation WG
- And Neil Chue Hong, Simon Hettrick, James Hetherington, Rob Haines, Ian Cosden, Kenton McHenry for RSE discussions
- And Morane Gruenpeter, Carlos Martinez, Neil Chue Hong, Paula A. Martinez, Michelle Barker, Leyla Jael Castro, Jennifer Harrow, Fotis Psomopoulos for co-leadership in RDA/ReSA/FORCE11 FAIR4RS
- And colleagues Gabrielle Allen, C. Titus Brown, Kyle Chard, Ian Foster, Melissa Haendel, Christie Koehler, Bill Miller, Tom Honeyman, Anna-Lena Lamprecht
- And to the BSSw project (http://bssw.io) for a fellowship to pursue some parts of the citation work
- More of Dan's thinking
  - Blog: http://danielskatzblog.wordpress.com
  - Tweets: @danielskatz