Research Software Sustainability

Future Trends in Nuclear Physics Computing
29 September 2020

Daniel S. Katz
(d.katz@ieee.org, http://danielskatz.org, @danielskatz)
Assistant Director for Scientific Software & Applications
Research Associate Professor, CS, ECE, iSchool

ILLINOIS
NCSA | National Center for Supercomputing Applications

Why do we care about research software?

- Funding
  - ~20% of NSF projects over 11 years topically discuss software in their abstracts ($10b)
  - 2 of 3 main DOE ECP areas are research software (~$4b)
- Publications
  - Software intensive projects are a majority of current publications
  - Most-cited papers are methods and software
- Researchers
  - >90% of US/UK researchers use research software
  - ~65% would not be able to do their research without it
  - ~50% develop software as part of their research

Collected from http://www.dia2.org in 2017
Nangia and Katz; 10.1109/eScience.2017.78
"Top 100-cited papers of all time," Nature, 2014 10.1038/514550a

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

Science of research software organizations

- Software Sustainability Institute (SSI)
  - In third period of funding, 10+ years
  - Now funded by all UK research councils
- Better Scientific Software (BSSw)
  - Clearinghouse to gather, discuss, and disseminate experiences, techniques, tools, and other resources to improve developer productivity and software sustainability
  - DOE funded
- United States Research Software Sustainability Institute (URSSI)
  - Conceptualization project under NSF funding
  - Institute proposal planned in 2020
  - Interest from private foundations
- Research Software Alliance (ReSA)
  - Intended to coordinate the above & others internationally

Open source & software growth

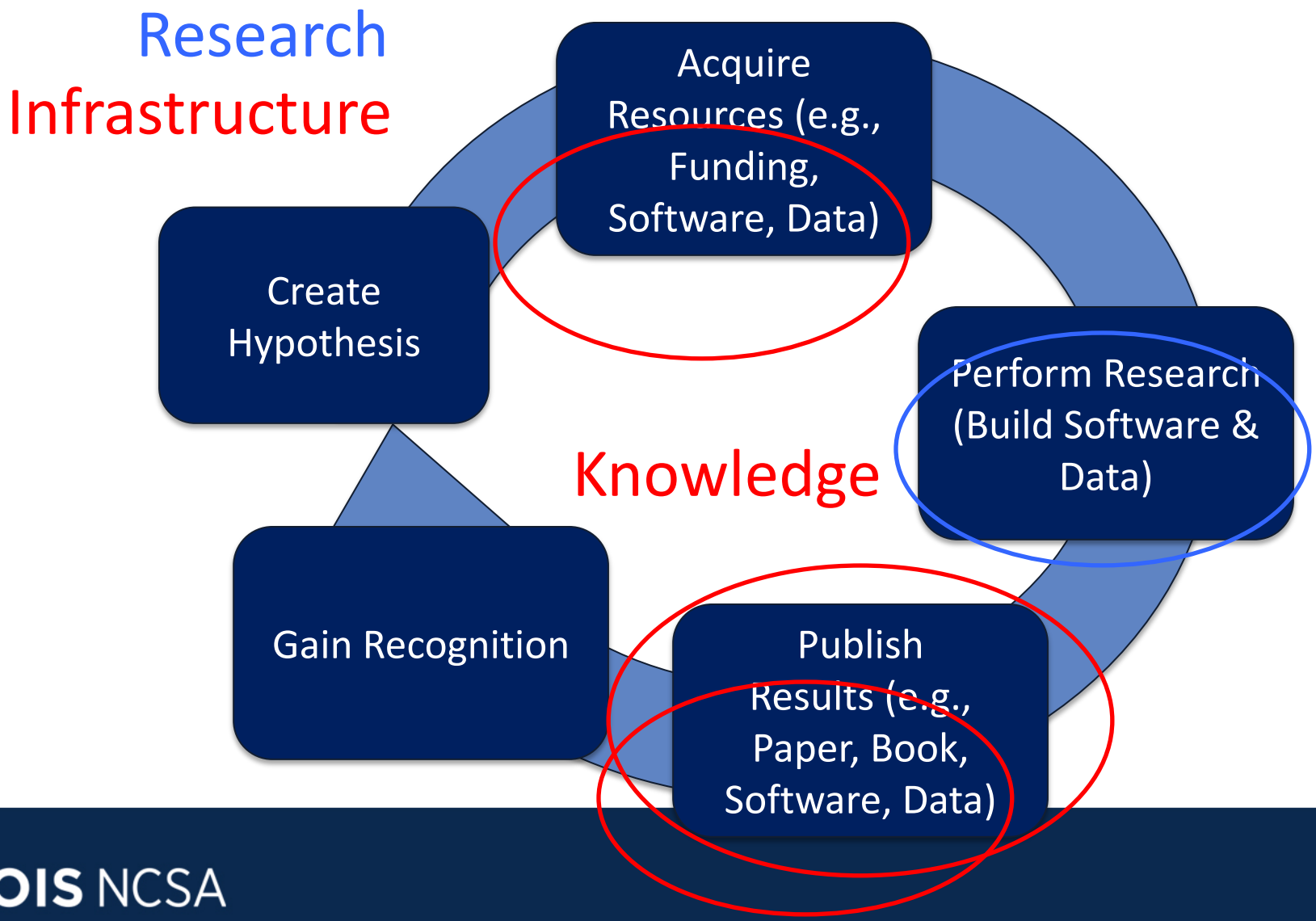
- 2001: 208K SourceForge users
- 2017: 20M GitHub users
- 2019: 37M GitHub users

- 1998: 180K downloads of Netscape (app) in 2 weeks
- 2017: 21M downloads of lodash (javascript library) in 2 weeks

- 2018 survey of scientist-developers found that 82% of respondents felt that they were spending spending “more time” or “much more time” developing software than they did 10 years ago

Credit: “Rebuilding the cathedral” @nayafia for Strange Loop 2017
Pinto at al., “How do scientists develop scientific software? an external replication,” 10.1109/SANER.2018.8330263

Software in research cycle

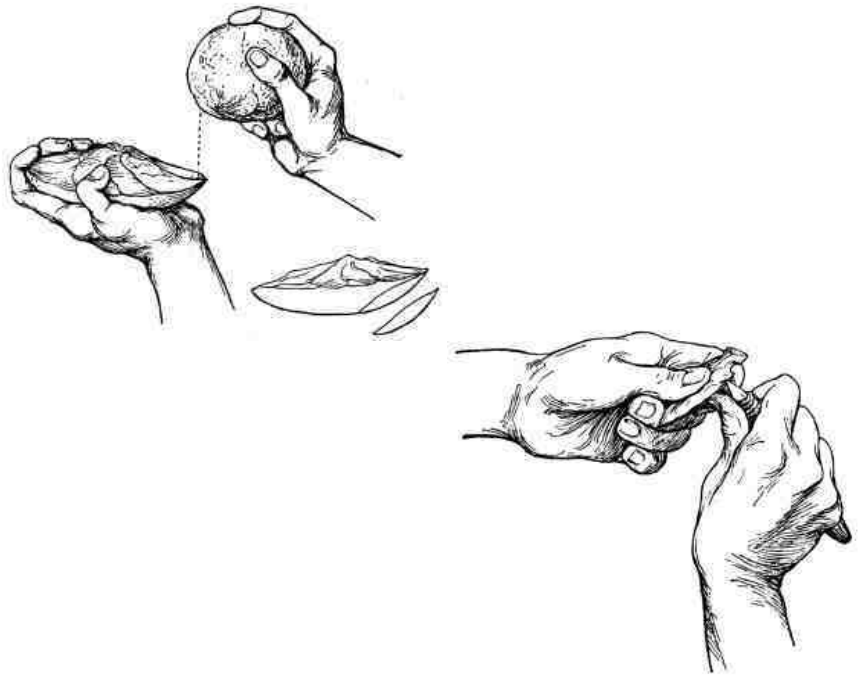


Research software vs. infrastructure software

- Some research software is intended just for research
  - Funded by many agencies, sometimes explicitly, often implicitly
  - Intended for immediate use by developer
  - Maybe archived for future use and reproducibility
  - And probably dependent on infrastructure software
- Other research software is intended as infrastructure
  - To be shared
  - Funded by some agencies, almost always explicitly
  - Intended for use by community
  - Appreciation and reward easier because of sharing
- Software intended for research can be turned into infrastructure software
  - Requires making a conscious choice
  - Has consequences, both positive and negative

Who starts new infrastructure software projects?

- Tool makers
  - To make something useful to others



- Then options:
- Accept contributions?
  and if so:
  a. Broaden focus?
     - Bring together other (related) packages
  b. Broaden governance?
     - Collaborate with other developers

Parsl: Interactive parallel programming in Python

Apps define opportunities for parallelism
- Python apps call Python functions
- Bash apps call external applications

Apps return “futures”: a proxy for a result that might not yet be available

Apps run concurrently respecting data dependencies. Natural parallel programming!

Parsl scripts are independent of where they run. Write once run anywhere!

```
pip install parsl
```

```python
@python_app
def hello ():
    return 'Hello World!'

print(hello().result())
```

```
Hello World!
```

```python
@bash_app
def echo_hello(stdout='echo-hello.stdout'):
    return 'echo "Hello World!"'

echo_hello().result()

with open('echo-hello.stdout', 'r') as f:
    print(f.read())
```

```
Hello World!
```

Parsl project summary

- Based on improving ideas in Swift workflow system/language & 10+ years of CS/infrastructure research
- Initially funded by NSF, $3m over 3 years (stretched to 4)
- 2.5 core developer FTEs, PI, co-PIs, chemistry & education application developers, undergraduate & graduate students
- Open source, intended as open community, including library of reusable workflows
- Interesting milestones
  - First outside user
  - First outside user who didn't contact us
  - First outside contributor
  - First outside contributor who didn't contact us
- Some success with purely external contributions to code, more success with collaborating projects (e.g., LSST-DESC, Thain group @ ND), some providing funding
- New follow-on NSF project (funcX) will use (and support) Parsl as dependency

Who starts new research software projects?

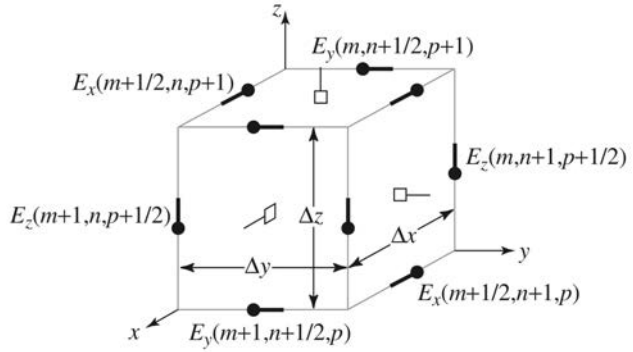
- User/Developer
  - To scratch their own itch

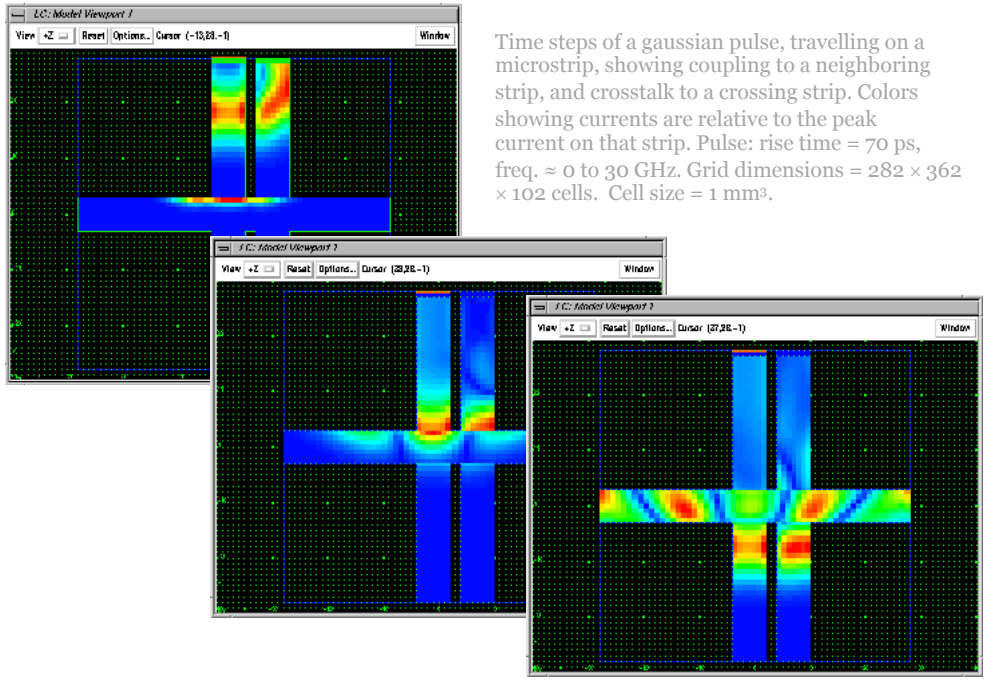
- Then options:

1. Keep it private
2. Share it
3. Accept contributions?
   and if so:
   a. Broaden focus?
      - Bring together other (related) packages
   b. Broaden governance?
      - Collaborate with other developers

FDTD electromagnetics

- My PhD dissertation was “Boundary treatments applied to the FD-TD method for solving problems of electromagnetic wave propagation” (1994)
- Adding methods to treat curved materials and infinite space to my advisor/lab’s time-marching code that solved EM problems on a finite cartesian grid
- Code originally written in late 1970s, FORTRAN77, no subroutines
- While I was working on it: heavily modified, better engineered, ported to various languages, vector computing, parallel computing
- But never shared – was viewed as part of the lab’s IP
- Algorithm was shared, leading others to redevelop code, now many versions including open source and commercial





Time steps of a gaussian pulse, travelling on a microstrip, showing coupling to a neighboring strip, and crosstalk to a crossing strip. Colors showing currents are relative to the peak current on that strip. Pulse: rise time = 70 ps, freq. ≈ 0 to 30 GHz. Grid dimensions = 282 × 362 × 102 cells. Cell size = 1 mm³.

Top figure: J. B. Schneider, Understanding the Finite-Difference Time-Domain Method, www.eecs.wsu.edu/~schneidj/ufdtd, 2010.

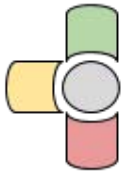
Bottom images produced at U of Colorado’s Comp. EM Lab. By Matt Larson using SGI’s LC FDTD code

Project stages

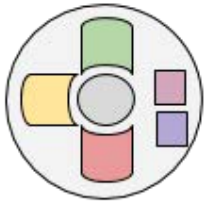
Schematic stages of open community for research software

Stage 0. Some code and a user of it. No sustained team.

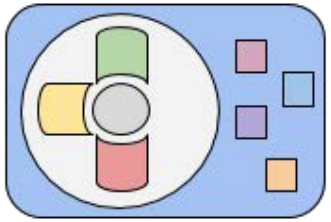
Stage 1. Software development team, internal use.



Stage 2. Multiple software teams (different institutions) on same code (team is *community*), for internal use.



Stage 3. Self-governing developer community deliberately supporting broad user community.

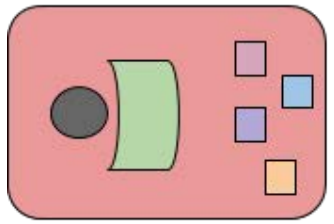


Stage 4. Self-sustaining organization dedicated to supporting user and dev community (e.g. through commercial support, events, software foundation, etc.).

Proprietary commercialization path...

Stage 0. Some code.

Stage 1. Software development team, internal use.

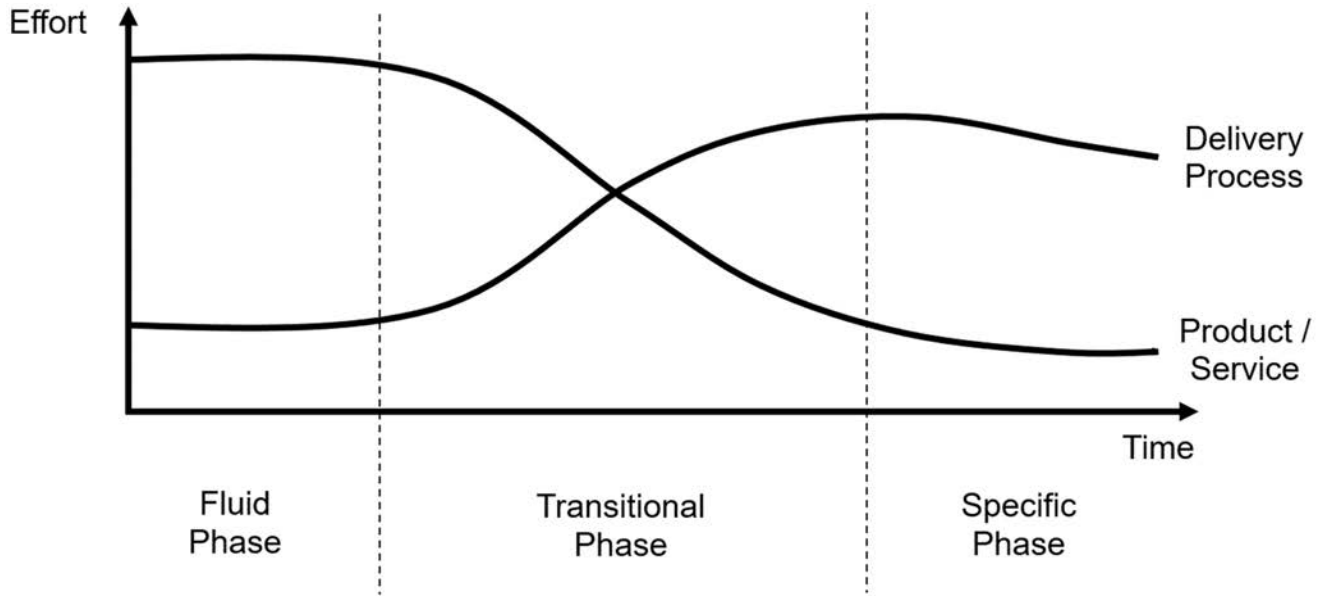


Stage X. Project goes commercial, without developing an open community.

S. P. Benthal, Software Incubator Workshop: A Synthesis, http://urssi.us/blog/2019/02/25/software-incubator-workshop-a-synthesis/

Changing stages

- At each point/stage, decide consciously to go forward
- Think about methods, goals, and consequences
- What resources are available to help
- What (type) of work will be needed?
- Are the right skills available?
- What are the incentives?
- How will success be measured?
- How will the institution(s) support this?



J. Leng , M. Shoura, T. C. B. McLeish, A. N. Real, et al. “Securing the future of research computing in the biosciences,” *PLoS Comput Biol* 15(5): e1006958, 2019. https://doi.org/10.1371/journal.pcbi.1006958

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

Software sustainability

- Software sustainability ≡ the capacity of the software to endure
  - Will the software will continue to be available in the future, on new platforms, meeting new needs?
- Software development and maintenance requires human effort
- Human effort ⇆ $
  - All human effort works (community open source)
  - All $ (salary) works (commercial software, grant funded projects)
  - Combined is hard: effort ≠ $; humans are not purely rational

Example/Problem 1: OpenSSL

- 1998: UK group built internet encryption tools: OpenSSL
- 2011: Heartbleed bug introduced
- 2014: 2/3 of web sites rely on OpenSSL
- One full-time developer: Steven Hensen, barely supported by OpenSSL Software Foundation (OSF)
  - Private, for-profit company
- 2014: Heartbleed bug discovered
- OpenSSL bug fixed
- OSF requests donations
  - $9,000 given initially
- Further campaign led to support for 4 developers for 3 years
- Today: active project with 18 committers and financial & in-kind (including staff time) support from companies

Credit: Nadia Eghbal, “Roads and Bridges,” 2016

Example/Problem 2: Bus factor

- NumFOCUS: Umbrella non-profit to support scientific software
- NumFOCUS sustainability summit annually since 2017
  - 2017 bus factor survey
  - Bimodal, ~half 1-2, ~half 4-6
  - One project's story: developer support, backlog, students
- Wider open source community
  - Two-thirds of popular projects: bus factor of 1 or 2

G. Avelino, M. T. Valente, A. Hora, "What is the Truck Factor of popular GitHub applications? A first assessment," *PeerJ Preprints* 5:e1233v3, 2017. https://doi.org/10.7287/peerj.preprints.1233v3

Research software summary

- 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

Max Planck

- A new scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die, and a new generation grows up that is familiar with it
- Or: science advances one funeral at a time
- My version: culture of science advances one funeral at a time

- Enough bad news
- What can we do?
- Wait
  - cf Planck
- Or act

12 scientific software challenges

- Incentives, citation/credit models, and metrics
- Career paths
- Training and education
- Software engineering
- Portability
- Intellectual property
- Publication and peer review
- Software communities and sociology
- Sustainability and funding models
- Software dissemination, catalogs, search, and review
- Multi-disciplinary science
- Reproducibility

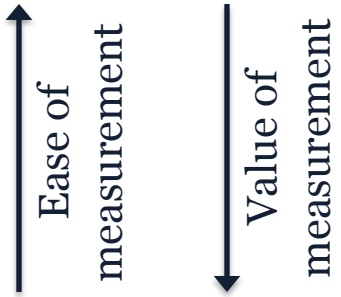
All are tied together

https://www.slideshare.net/danielskatz/scientific-software-challenges-and-community-responses

Credit for software: What to measure?

1. Developer of open source physics simulation
   - Possible metrics
     - How many downloads?
     - How many contributors?
     - How many uses?
     - How many papers cite it?
     - How many papers that cite it are cited?

impact



2. Developer of open source math library
   - Possible metrics are similar, but citations are less likely
   - May not be able to measure downloads
     - It's part of a distribution
     - It's pre-installed (and optimized) on an HPC system
     - It's part of a cloud image
     - It's a service

Measuring software impact

- Citations
  - Because citation system was created for papers/books
  - Need to jam software into current citation system
- Altmetrics
  - Not citations, but other structured measures of discussion (tweets, blogs, etc.)
- ImpactStory
  - Measures research impact: reads, citations, tweets, etc.
- Depsy (roughly ImpactStory specifically for software)
  - Measures software impact: downloads, software reuse (if one package is forked into another package), citations, tweets, etc.
- Libraries.io
  - Counts software dependencies

Software citation today

- Software and other digital resources currently appear in publications in very inconsistent ways
- Howison & Bullard: random sample 90 articles in biology literature -> 7 different types of software mentions

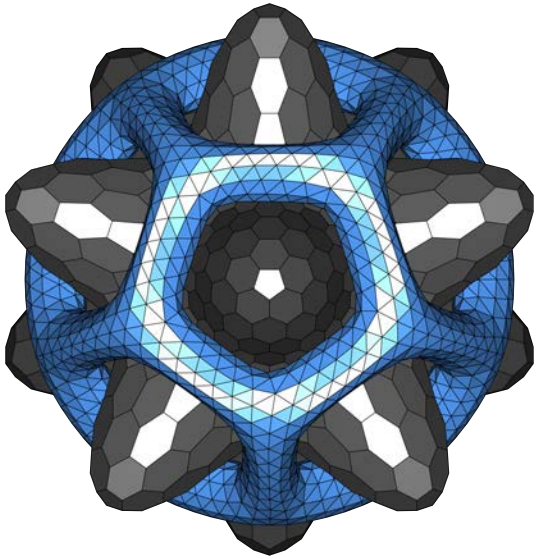
Mention Type	Count (n=286)	Percentage
Cite to publication	105	37%
Cite to users manual	6	2%
Cite to name or website	15	5%
Instrument-like	53	19%
URL in text	13	5%
In-text name only	90	31%
Not even name	4	1%

- 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. https://doi.org/10.1002/asi.23538

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: 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
- JOSS papers archived, have DOIs, increasing indexed
- First paper submitted 4 May 2016
  - 31 May 2017: 111 accepted papers, 56 under review and pre-review
  - 22 Sept 2020: 1023 accepted papers, 177 under review and pre-review
  - Pre-Covid-19 publication rate ~400 papers/year, now back to that rate
- Editors: 1 editor-in-chief and 11 editors at launch;
  1 EiC, 5 associate EiCs, 48 topic editors, 13 emeritus editors today



Software citation principles

- FORCE11 Software Citation group started July 2015
- WSSSPE3 Credit & Citation working group joined September 2015
  - ~55 members (researchers, developers, publishers, repositories, librarians)
- Reviewed existing community practices & developed use cases
- Drafted & published software citation principles
  - Started with data citation principles, updated based on software use cases and related work, updated based working group discussions, community feedback, workshop
  - 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
  - Principles: **importance, credit and attribution, unique identification, persistence, accessibility, specificity**
- Software Citation Working Group ended April 2017

Software citation implementation

- FORCE11 Software Citation Implementation Working Group in progress, started May 2017
  - Co-chairs: Neil Chue Hong, Martin Fenner, Daniel S. Katz
  - Goal is implementing software citation
  - Working with institutions, publishers, technology and service providers, funders, researchers, etc.
- Lots of good work being done, and good coordination of ongoing activities
- Metadata standards and translation (DataCite Schema 4.1, CodeMeta, citation.cff), being aligned with schema.org
- Open source archiving and identification (Software Heritage) developing
- Good work and initial acceptance in communities (astronomy, Earth science, math, …)
- Published Software Citation Checklist for Authors v0.9 document
- Published Software Citation Checklist for Developers v0.9 document
- Software Citation Checklist for Reviewers document under review
- Repositories task force developing good/best practices for registries and repositories
- Journals task force started in Jan 2020, guidance document for journals (and conferences to use) in publication

12 scientific software challenges

- Incentives, citation/credit models, and metrics
- Career paths
- Training and education
- Software engineering
- Portability
- Intellectual property
- Publication and peer review
- Software communities and sociology
- Sustainability and funding models
- Software dissemination, catalogs, search, and review
- Multi-disciplinary science
- Reproducibility

All are tied together

https://www.slideshare.net/danielskatz/scientific-software-challenges-and-community-responses

Challenge: better career paths?

- Career paths for software developers in universities unclear
- Should we give up in favor of national labs & government intramural researchers & industry?
  - More financial rewards, cohorts (others with similar problems and solutions) and promotion opportunities
- For researchers, published software or software papers make software a valued output similar to publications
- University centers, e.g. NCSA, SDSC, TACC, give programmers a home & critical mass for career paths
- Moore/Sloan Data Science program created new structure across universities
- Research Software Engineers (RSEs) …

Who are Research Software Engineers?

- Not independent researchers
  - No personal research agenda
- Facilitative, supportive, and collaborative
  - Part of the academic community
  - With professional IT skills
  - Deep engagement with research groups
  - Understand, study, and be part of group research activities
  - Can read and understand the papers
- Sustainable and long term
  - Institutional memory
  - Continuity, stability, maintenance
- But in most institutions
  - Without a formal home in academia, or a career path

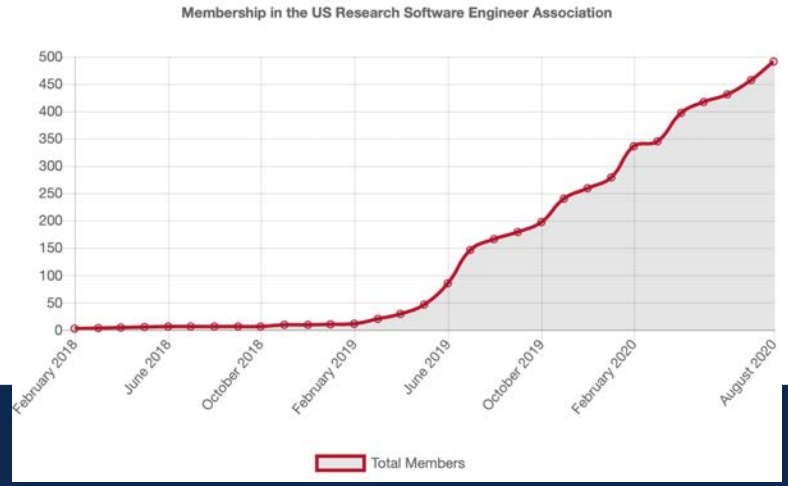
Credit: James Hetherington, “Computational Science as a Service,” http://github-pages.ucl.ac.uk/rsd-talks/rsd/experiences-reveal.html

The Story of RSEs

- April 2012: Idea & name at SSI Collaborations Workshop
- September 2012: University College London group founded
- More groups: Manchester (2014); Sheffield, Southampton, and Cambridge (2015)
- January 2016: EPSRC awards first RSE Fellowships
- RSE Conferences in the UK
  - September 2016: First RSE conference, 202 attendees, 14 countries
  - September 2017: Second RSE conference, 224 attendees
  - September 2018: Third RSE conference, 340 attendees
  - September 2019: Fourth UK RSE conference, 360 attendees
- Society of Research Software Engineering – formed 2019
  - Independent organization for RSEs: international, membership fee, voting rights
- Other countries at various stages of development
  - Germany, Netherlands, Nordic, US, South Africa, Canada, Australia/NZ, Belgium, …
  - First conferences in Germany and Netherlands occurred in 2019
  - 491 US RSE members (as of Sept 1 2020)

RSE
SOCIETY OF RESEARCH SOFTWARE ENGINEERING

US RSE
US RESEARCH SOFTWARE ENGINEER ASSOCIATION



Promotion and evaluation

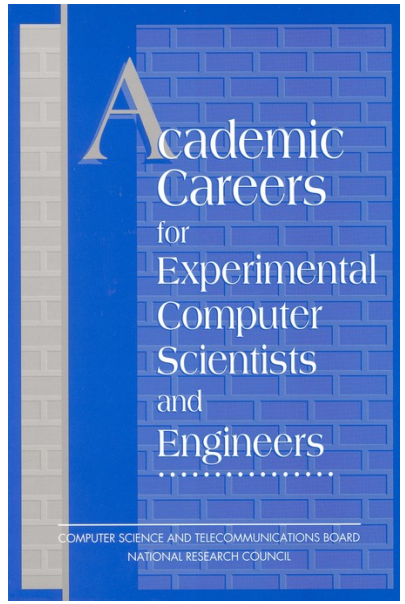
- Guidelines for promotion and evaluation important
  - Say what's valued; shape activities people undertake
  - Promotion guidelines written by senior people, how can they be changed?

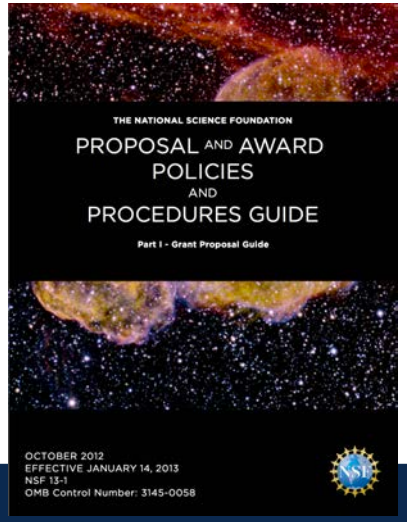
- We can influence these processes when we participate in these evaluations

- We can provide templates and guidelines for recognizing software contributions and encourage respected organizations to adopt them
  - Multiple groups working in this space

Promotion and evaluation are not fixed

- National Academies (1994): “Academic Careers for Experimental Computer Scientists and Engineers”
  - Experimental artifacts are important in CS, should be part of evaluation
  - Intended to provide a reference point for change
  - Has been quoted in many tenure recommendation letters





- NSF 2013 biosketch change: products, not publications
  - Acknowledges software contributions as a primary research product
  - Intended to signal to universities that they should do the same

https://www.nap.edu/read/2236/

https://www.nsf.gov/pubs/policydocs/pappguide/nsf13001/gpg_index.jsp

Potential general solutions

- Convince governments and funders of importance of software (and sustained funding for some of it, including maintenance)
  - via Research Software Alliance (ReSA) and others
- Encourage use of software citation to aid developers
  - via FORCE11 Software Citation Implementation Working Group
- Build better career paths for developers
  - via Research Software Engineer (RSE) movement (https://society-rse.org, http://us-rse.org)
- Develop and use software best practices
  - via Project Carpentry, Incubators (e.g. ESIP, Apache)
- Join groups working on these
  - SSI, URSSI, NumFOCUS, CS&S

Examples of potential specific solutions

- Under IRIS-HEP and HSF, held Software Sustainability and High Energy Physics workshop, https://indico.cern.ch/event/930127/, in July 2020
- Report being drafted
- Current draft recommendations include:

  (all are in the context of HEP)

  - Training
    - Create visiting fellowships to develop and “professionalize” training material
    - Increase scalability of training
  - Software
    - Explicitly consider and invest in tools that enable and support software sustainability
    - Further discuss via Snowmass and future focused workshops
  - People
    - Create award(s) for software work
    - Consider future workshop focused on career paths, with university and lab administrators

Credits

- Thanks to Arfon Smith and Kyle Niemeyer for co-leadership in FORCE11 Software Citation WG
- And Neil Chue Hong & Martin Fenner for co-leadership in FORCE11 Software Citation Implementation WG
- And colleagues Gabrielle Allen, C. Titus Brown, Kyle Chard, Ian Foster, Melissa Haendel, Christie Koehler, Bill Miller, Rajiv Ramnath
- And to the BSSw project (http://bssw.io) for a fellowship to pursue some parts of the citation work

- More of my thinking
  - Blog: http://danielskatzblog.wordpress.com
  - Tweets: @danielskatz