

Experiences in how RSE community identity leads to improved research software practices

Daniel S. Katz

Chief Scientist, NCSA

Associate Research Professor, CS, ECE, iSchool

University of Illinois at Urbana-Champaign

d.katz@ieee.org, @danielskatz@fosstodon.org



I ILLINOIS

NCSA | National Center for
Supercomputing Applications

USRSECon23, Chicago, 16 October 2023

<https://doi.org/10.5281/zenodo.10009616>



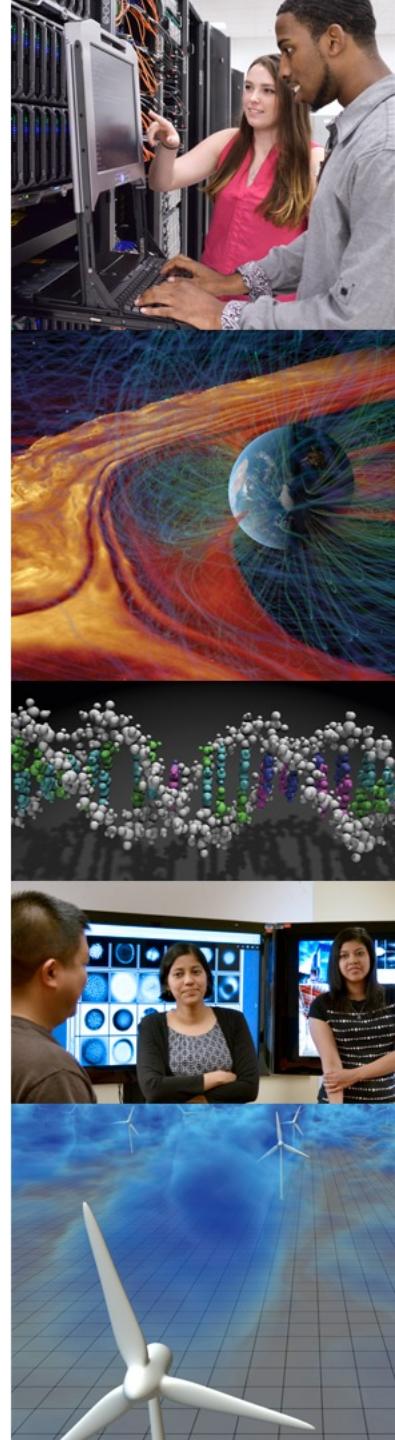
Research software once

- Research software once was a heroic endeavor, particularly in HPC
- My advisor created his own standalone simulation software in the 1960/70s
 - Depended on compilers and operating systems (and later PVM, then MPI), but otherwise had no dependencies
- When I started work with him (1987)
 - Transferred from student to student by hand (email/floppy); he kept the “master”
 - Coding practices were what he said; some pushback from students led to some changes
 - Installation was compiling, typically with a hand-typed command
 - Documentation was very few comments in the code to identify sections, lots of papers
 - Testing was mostly high-level replication of results known by theory

Research software today

- Most research software today is a social endeavor
- Most software depends on other software and is developed/maintained by multiple people
- We have best/“good enough” practices for testing, documentation, installation, version control, issues, PRs
- **Q: How do these practices form and become accepted in different contexts?**
- Research software engineering (RSEng) and research software engineers (RSEs) becoming accepted parts of the research software endeavor
- **Q: What’s the role of RSEs in creating, adapting, and infusing these practices?**
- Let’s look at communities, projects, and groups

1 - Research software community

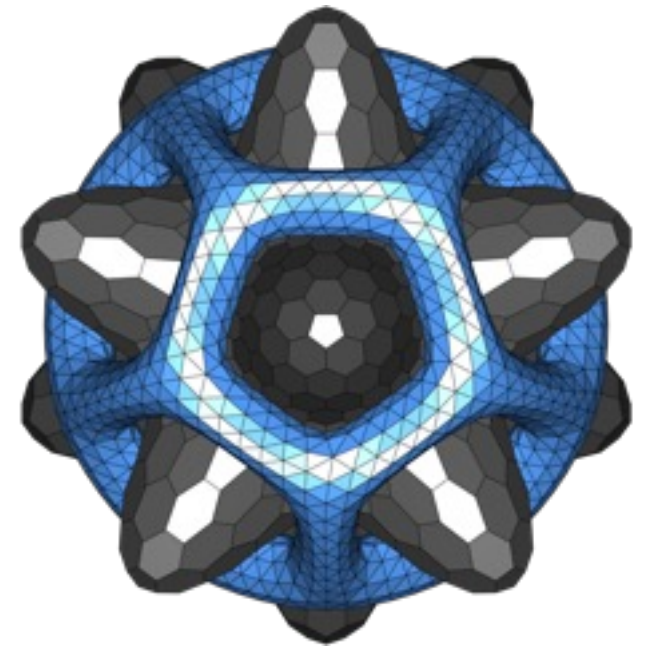


Journal of Open Source Software (JOSS)

- JOSS started to fill a gap: to better recognize software contributions
 - By finding a way to fit software into current (paper/book-centric) system
- Solution
 - Make it as easy as possible for authors to write software papers based on existing software documentation
 - Main purpose of JOSS paper is to enable citation credit to be given to authors of research software
 - A JOSS paper should be seen as equivalent to an accepted paper elsewhere
- JOSS reviews
 - On GitHub, open, non-anonymous, checklist-driven, collaborative
 - Editor's job is to find reviewers, and to help them and author come to agreement

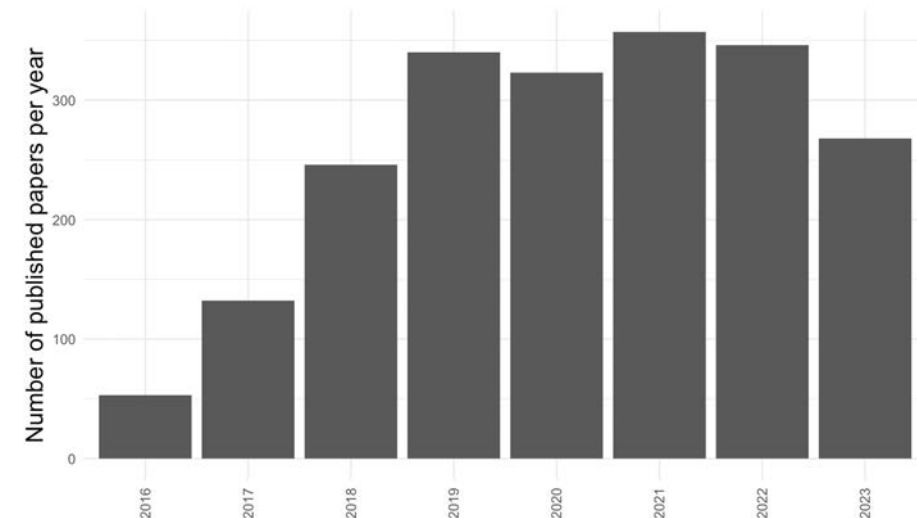
JOSS review criteria

- Some criteria have guidance
 - Installation
 - API documentation
 - Community guidelines
 - Automated testing
- These have levels, e.g., for installation
 - Good: The software is simple to install, and follows established distribution and dependency management approaches for the language being used
 - OK: A list of dependencies to install, together with some kind of script to handle their installation (e.g., a Makefile)
 - Bad (not acceptable): Dependencies are unclear, and/or installation process lacks automation



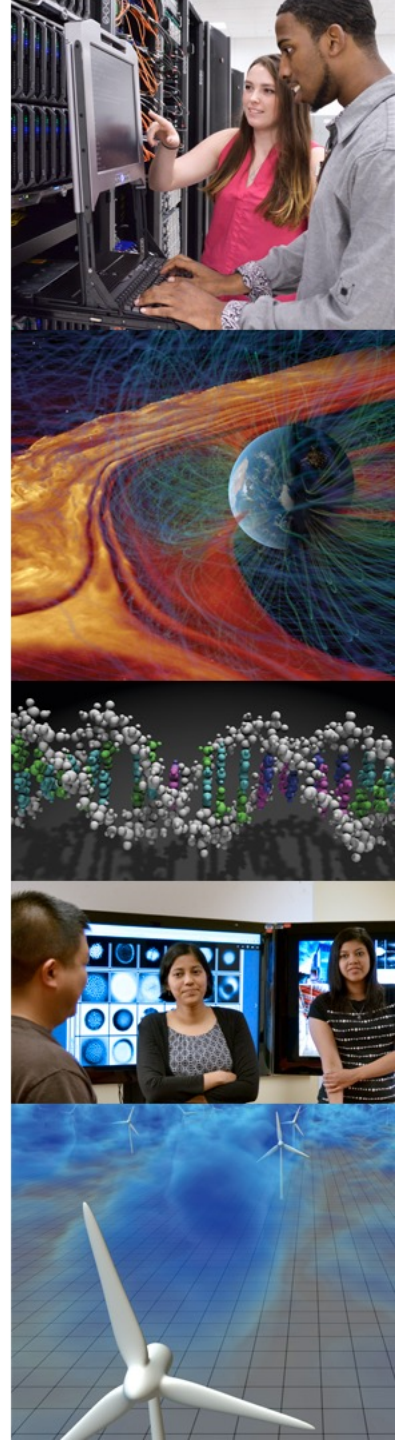
JOSS as an element of RSE community

- JOSS practices (these levels) 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
- Cultures change based on rules and incentives
- JOSS provides rules, and at a high-level, tries to nudge incentives
- Over time, community changes, and JOSS levels also change (stricter)
- Many JOSS authors and reviewers (and editors) are RSEs
- If software was cited directly, JOSS papers wouldn't be needed, but JOSS reviews and JOSS (and RSE) community would still be important in shaping values



2023-09-13

2 - Research software projects



Parsl: 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 dataflow dependencies. Natural parallel programming!

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

```
pip install parsl
```

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

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

Hello World!



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

echo_hello().result()

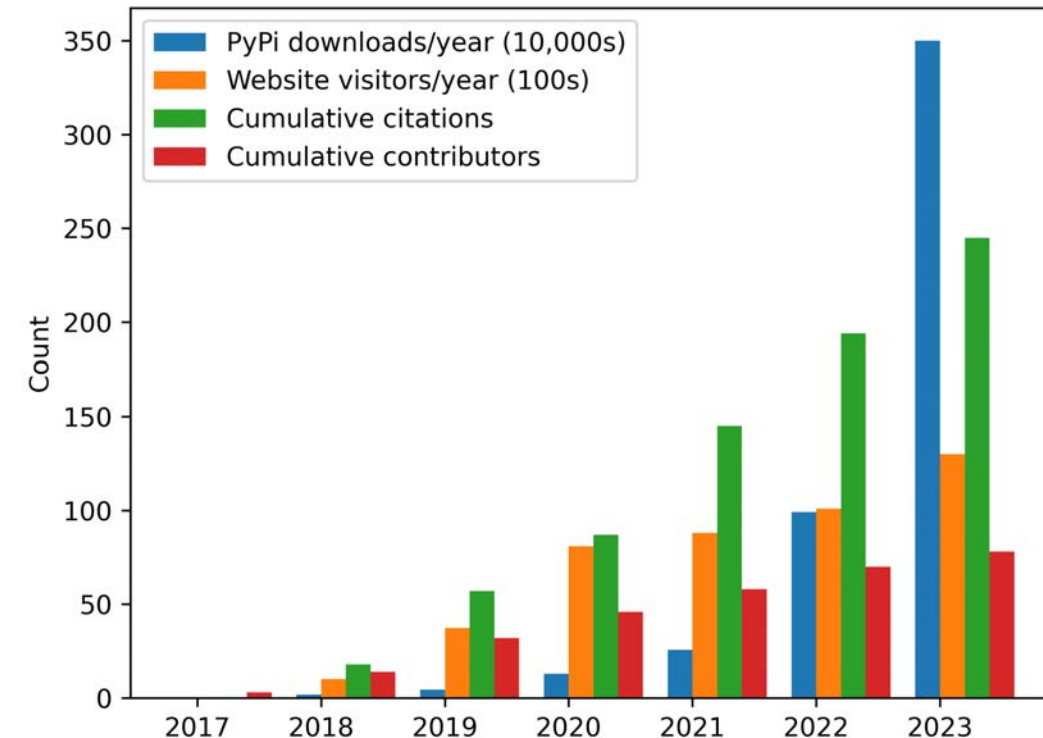
with open('echo-hello.stdout', 'r') as f:
    print(f.read())
```

Hello World!



Parsl history

- Supported by an NSF SI2 award from 2016-2022 (5 years + 1-year NCE)
- Started as an idea in 2016, based on previous project Swift (<http://swift-lang.org/main/>)
 - Basically asked if we want to do the same thing – a simple tool (language/runtime) for fast, easy scripting on big machines – today, what would we do
 - One person did some exploration/proof-of-concept – it worked
 - Build the initial usable system, was the main developer, did things the way they wanted
 - Once a second developer became active, needed to agree on and define/document processes
 - As we moved to a more open community project (2-4 FTEs/year of developers funded, and 73 contributors), these processes became more important
- Open source
- Released version 1.0 in 2020, now releasing weekly
 - Focus since v1.0 mostly maintenance rather than adding new features
- Funded development team: 2-4 people/FTEs per year
 - Current funding includes new NSF & CZI awards, contributions from projects that require Parsl



Parsl processes

- How to make design/architecture decisions?
- What code style to use?
- What testing is sufficient?
- What documentation is sufficient?
- How to engage with and support users?
- What properties do contributions and changes need to have?
- How do contributions and changes get accepted?
- How to encourage/develop contributors?
- How to mix CS research and software product development?
- Who writes papers and who is listed as co-authors?
- All of the answers have changed over the life of the project

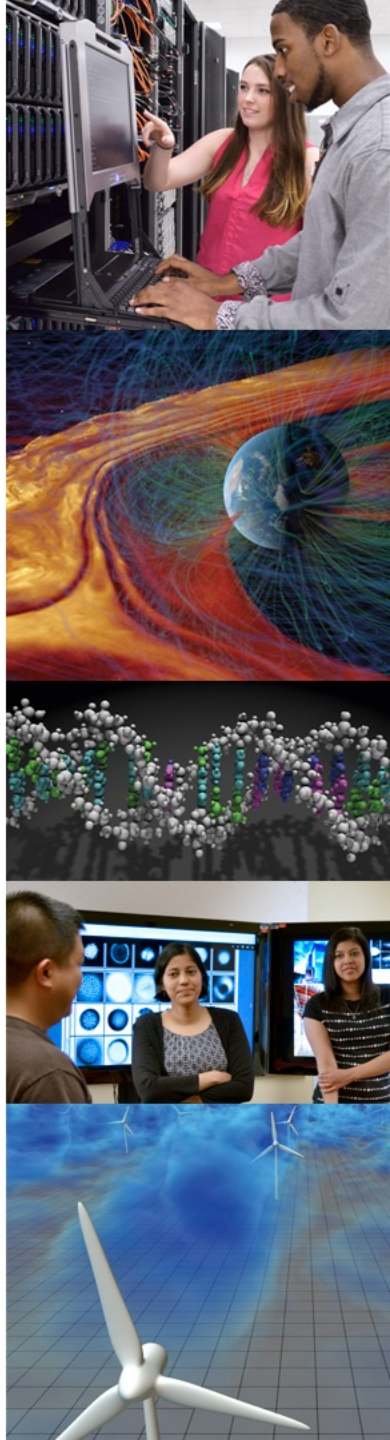


Parsl team/community



- We made specific decisions on these processes; alternative decisions could have been equally valid
- These practices are part of what makes the development team a community
- As new developers come in, there's often discussion about these practices and if they can/should be improved
- Many of these developers are RSEs, and some are undergrad/grad students or faculty
- All bring in ideas, new ideas mostly from RSEs and students
- Parsl processes impact others
 - All contributors learn from Parsl; impacts their future projects
 - RSEs also use this to influence their groups and larger RSE community
 - Students may influence other students; some students will become RSEs

3 - Research software groups



How a local RSE organization starts

A software developer hears about RSEs, and realizes they are one

- Talk about this with colleagues, leading to informal community
- Formalize the community
 - Mailing list, slack, lunches, meetings

Or perhaps software developers group together to more effectively manage changing projects/staffing needs over time and provide some semblance of a career path in a soft-funded environment

- Formalize a group (if enough are in a unit, e.g., department, IT, research)
 - Develop mission, working internally and with stakeholders
 - Discuss institutional funding support for the group (full, partial, none)
 - Create career path, including how they fit into formal HR positions

Or perhaps an administrator hears about RSEs, and realizes they either have some or should

- To compete with other universities
- To recognize work done within the company
- They formalize a group (e.g., in a department, IT, research center)
 - Stakeholders develop mission
 - Determine institutional funding support for the group (full, partial, none)
 - Create career path, including how they fit into formal HR positions
 - Hire leader and group members (or reorganize existing staff)

- Community or group and its members joins international, national, local RSE groups
- Share experiences, learn from each other
- Talk about work with HR, create proper official roles
- Everyone lives (and works) happily ever after
 - Note: a group can get too big, then needs to rethink funding, structure, ...

RSE group & community models

- RSEs work on multiple projects, each with its own choices of practices
- But multiple RSEs in a group with multiple projects means these RSE groups can collectively think at a metalevel
 - What sets of practices are good
 - Which ones from that set could be used in a new project
- RSEs also communicate across local RSE groups via national RSE organizations (e.g., US-RSE)
- And influence their organization and others
 - RSEs work with faculty, staff, and students in their organization and those on multi-institution projects
 - And infuse good practices into these parts of the community



Conclusions

- All practices, including research software practices, are tied to the communities that perform them
- RSEs are increasingly leading the research software development community
- They often are the point where new practices become accepted and then disseminated
- This happens in projects, RSE groups, and the community as a whole
- Want to change practices quickly? Get RSEs to adopt them by convincing champions who are RSEs

Acknowledgements

- Based on talk at SIAM CSE'23: <https://doi.org/10.5281/zenodo.7683966>
- JOSS discussion is from a Feb 2022 talk, with JOSS editors at that time: Gabriela Alessio Robles, Mikkel Meyer Andersen, Katy Barnhart, Juanjo Bazán, Sebastian Benthall, Eloisa Bentivegna, Monica Bobra, Frederick Boehm, Jed Brown, Pierre de Buyl, Patrick Diehl, Elizabeth DuPre, Vissarion Fisikopoulos, Martin Fleischmann, Dan Foreman-Mackey, Jarvis Moore Frost, Nikoleta Glynatsi, Jeff Gostick, Richard Gowers, Hugo Gruson, Olivia Guest, David Hagan, Jayaram Harihan, Chris Hartgerink, Bitá Hasheminezhad, Christina Hedges, Luiz Irber, Mark A. Jensen, Prashant K. Jha, Daniel S. Katz, Vincent Knight, Rachel Kurchin, Hugo Ledoux, Christopher R. Madan, Brian McFee, Melissa Weber Mendonça, Kevin M. Moerman, Kyle Niemeyer, Juan Nunez-Iglesias, Lorena Pantano, Stefan Pfenninger, Viviane Pons, Kristina Riemer, Amy Roberts, Marie E. Rognes, Ariel Rokem, Will Rowe, Kelly Rowland, David P. Sanders, Mehmet Hakan Satman, Fabian Scheipl, Jacob Schrieber, Adi Singh, Arfon Smith, Charlotte Soneson, Øystein Sørensen, Andrew Stewart, Fabian-Robert Stöter, Yuan Tang, George K. Thiruvathukal, Kristen Thyng, Tim Tröndle, Leonardo Uieda, Chris Vernon, Marcos Vital, Lucy Whalley, Bruce E. Wilson, Frauke Wiese
- Parsl team including Yadu Babuj, Kyle Chard, Ryan Chard, Ben Clifford, Ian Foster, Zhuazhao Li, Mike Wilde, Anna Woodard