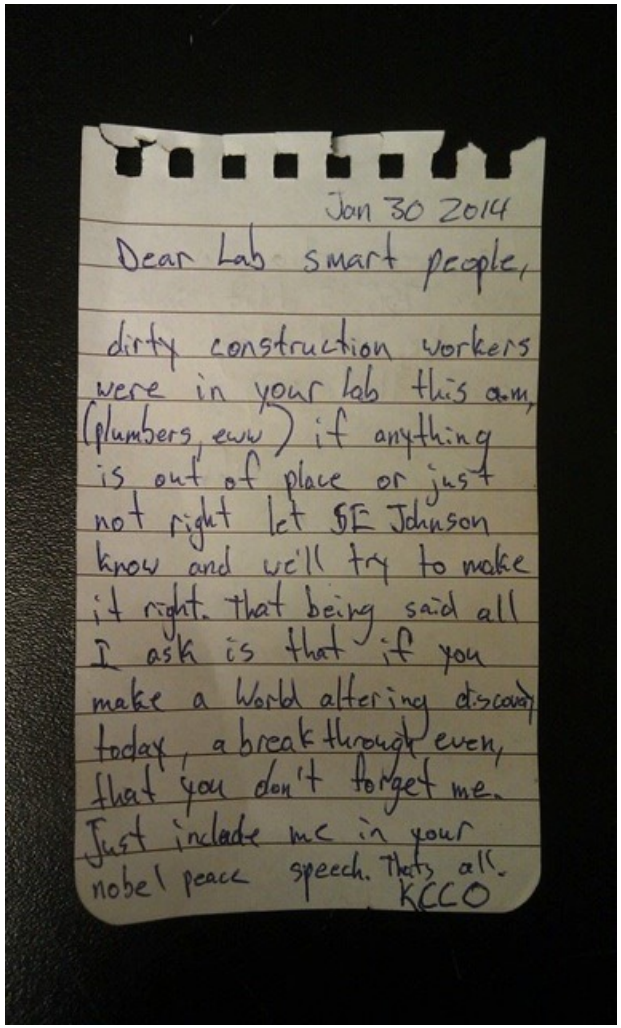
Software makes science better, but is it research?

Arguments for a research agenda in scientific software work

James Howison
Associate Professor
School of Information
University of Texas at Austin
@jameshowison

This material is based upon work supported by the US National Science Foundation under Grant Nos. SMA- 1064209 (SciSIP), OCI-0943168 (VOSS), and OAC-1453548.

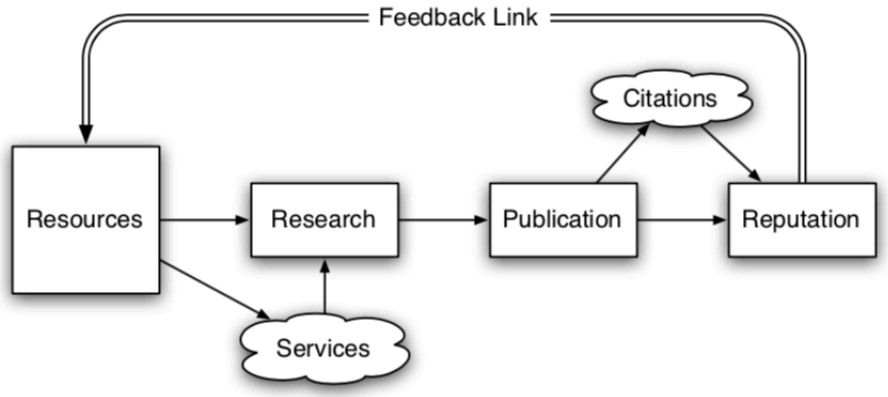


Plumbers, ewwww…

Arguments for contribution

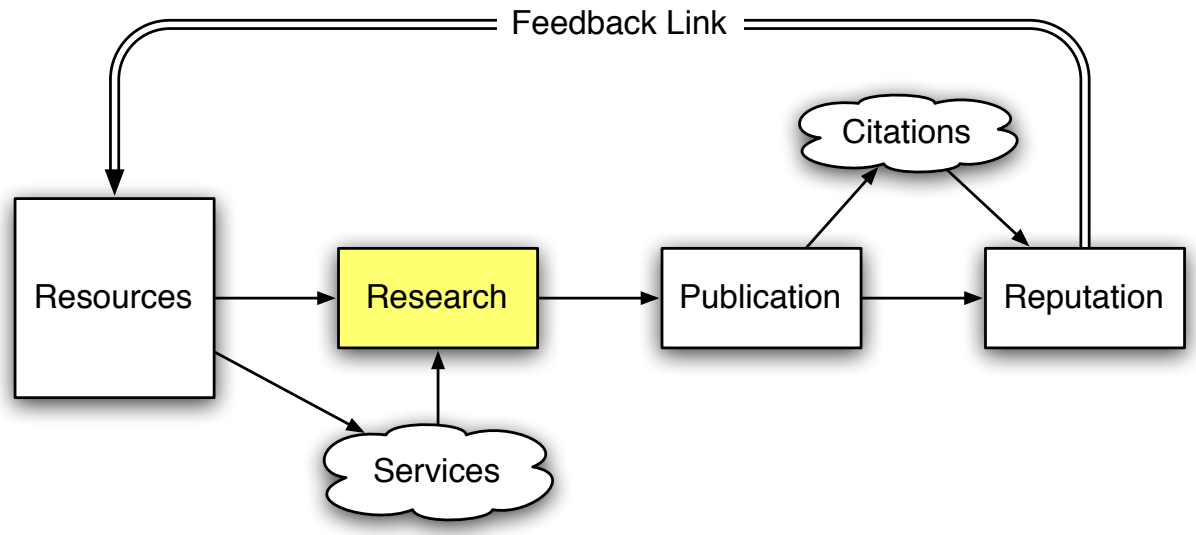
- These arguments are made (and I've observed them and their reception) in
  - In job applications
  - In tenure and promotion cases
  - In grant applications
  - In grant reports and reviews
  - In publications
- Today:
  - What arguments generally works?
  - What arguments generally struggle?
  - What ought to work but often doesn't?

A simplified scientific reputation economy

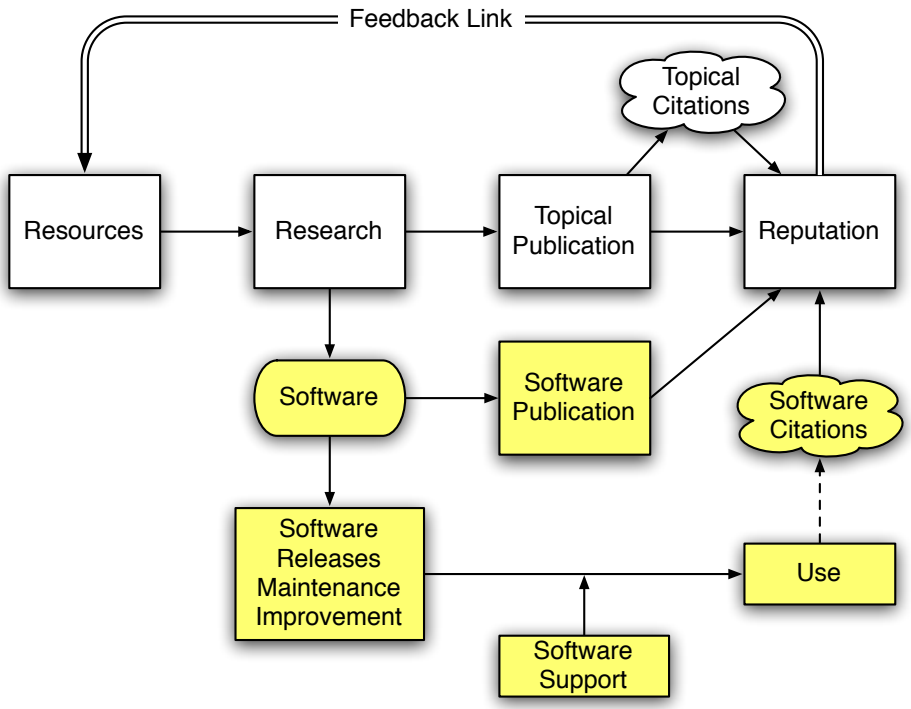


From: Howison, J., & Herbsleb, J. D. (2011). Scientific software production: incentives and collaboration. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (pp. 513–522). Hangzhou, China. https://doi.org/10.1145/1958824.1958904

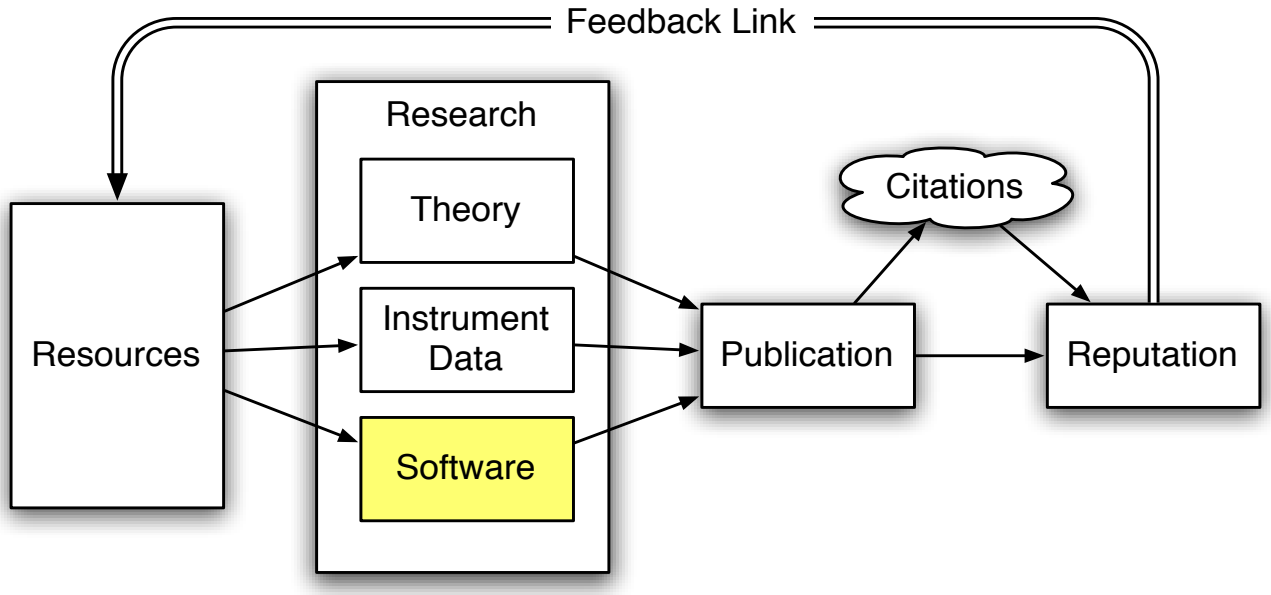
Software work as research



Or, perhaps more like this:

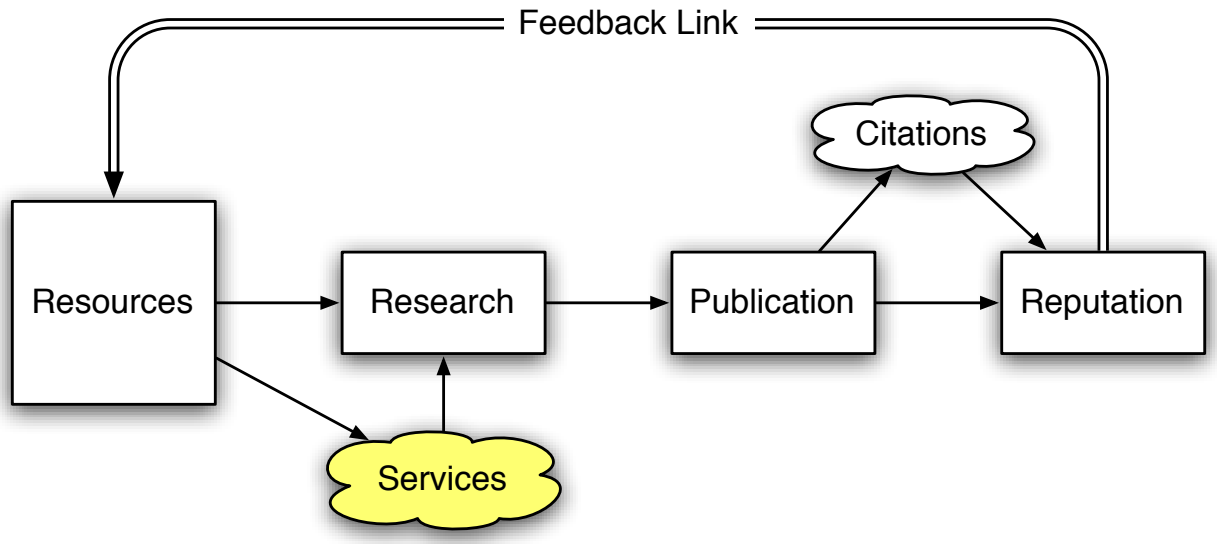


Collaboration service-work



From: Howison, J., & Herbsleb, J. D. (2011). Scientific software production: incentives and collaboration. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (pp. 513–522). Hangzhou, China. https://doi.org/10.1145/1958824.1958904

But often software work is viewed as this:



What arguments work?

“My algorithm or analysis is novel”

- Develop novel algorithms or statistical methods realized in the software.
- Here the contribution is clear, the algorithms stand apart from their particular implementation.
- 2013 Nobel Prize in Chemistry (Karplus, Warshell, Levitt)
  - Software was definitely mentioned, but it was the methods and models that were foregrounded.

Generalizable software or data techniques

- Identify and evaluate the generalizable **data or runtime software techniques** instantiated in the software
  - e.g., a new design pattern, a new data structure, new architectural move.
- Contribution to cross-cutting scientific fields, rather than domain of end-user scientists.
  - Publishing in computer science, data science, computational sub-fields.

What arguments struggle?

“The users liked it”

- Write a piece of software, write about what it does, do a survey of whether users liked it
  - Or quote letters from happy users.
- Actually very common in human computer interaction fields, a well-known troublesome claim.
- Clearly lacks generalizability, hard to know **why** users like it

“We introduce technique x from field y”

- Making a new technique available to your domain
- Often criticized as “intellectually trivial”, just lots of “grunt work”
- (I’m not sure who people think should do this needed work!)

“We have many users”

- For years I (and others) have been saying “Better measure your users”
- Even wrote a paper on how to do this:
  - Howison, J., Deelman, E., McLennan, M. J., Silva, R. F. da, & Herbsleb, J. D. (2015). Understanding the scientific software ecosystem and its impact: Current and future measures. *Research Evaluation*, *24*(4), 454–470. https://doi.org/10.1093/reseval/rvv014
- But this argument, on its own, struggles.
  - Very easy to question as intellectual merit, place into “broader impacts”
  - Very easy to characterize as service work, alongside plumbers, HR, chair suppliers.
  - Use alone is insufficient, need to discuss impact.

“Our software helps our field collaborate”

- A variant of making the science easier
- Hosting data, sharing analysis, web gateways to existing tools.
- In evaluation, often equated to running a journal or a workshop:
- Useful, appropriate for academics, but just not research, at least not enough to value over new algorithms or analyses.

“We make the science of our users easier”

- An argument of collective achievement. Seeking a portion of the credit for science realized by others.
  - Infrastructure work tends to become invisible. (Star and Ruhlender, 1994; Star and Strauss, 1999)
- Actually works inside large team science collaborations, mentioned in letters of recommendation, helps people get next jobs.
- But what about outside collaborations?
  - Most evaluation processes revert to seeking “individual contribution”
- You need to convince reviewers that *science is collective accomplishment* and that it would be so much worse off without your software.
  - And overcome the “shouldn’t they use commercial tools?” objection.

What ought to work but often doesn’t?

“We make the science of our users better”

- Not just easier or possible, but **better**.
- Forces one to discuss, argue, and demonstrate what better science is.
  - More accurate? (operating at a higher scale, reducing uncertainties)
  - Clearer?
  - Easier to share and extend?
  - Easier to falsify?
- Sometimes doesn’t work because:
  - Still viewed as a claim on collective accomplishment.
  - What evidence can be adduced to demonstrate better science by others?

Pledge from Dagstuhl on Engineering Academic Software

"I will publish the intellectual contribution of my software"

Allen, A., Aragon, C., Becker, C., Carver, J., Chis, A., Combemale, B., … Vinju, J. J. (2017). Engineering Academic Software (Dagstuhl Perspectives Workshop 16252). *Dagstuhl Manifestos*, *6*(1), 1–20. https://doi.org/10.4230/DagMan.6.1.1

- But how?

Making the case for intellectual merit

- What in your software work is difficult to accomplish?
  - Technical or social (or probably both!)
- What are your solutions? How did you arrive at them? Why are they clever and effective?
- How can an outsider judge their merit?
  - What evidence can you present?
- Who else could learn from those solutions?
  - Inside but also outside the scientific domain of the intended users.
  - Can mean speaking to unaccustomed audiences.

Generalizable contributions to other fields

- Novel requirements analysis approaches
  - e.g., new and better ways to learn about how scientists do their work.
- Novel interface approaches
  - e.g., new and better ways to suggest using ontology terms rather than free form text
- Novel community building approaches
  - e.g., novel architectures to encourage participation or that provide better incentives for integration of contributions
- Software development methodologies
  - e.g., theory and challenges for introducing agile into a lab, project as case study
- Introducing change in scientific communities through software
  - e.g., hypothesize that the software will create change, measure whether it does.

NSF definition of “Transformative Research”

Transformative research involves ideas, discoveries, or **tools** that radically **change our understanding** of an important existing scientific or engineering concept or educational practice or leads to the **creation of a new paradigm** or field of science, engineering, or education. Such research **challenges current understanding** or **provides pathways to new frontiers.**

National Science Board, 2007 “Enhancing Support of Transformative Research at the National Science Foundation”

What is to be done?

- Individuals have to decide how to characterize their contributions
  - Be very aware of how a venue/solicitation characterizes “research” (esp. CAREER)
  - Watch for slippage into generally ineffectual arguments
    - esp. facilitating and making the work of other’s easier.
  - Consider characterizing some types of work as “service to the field” (and prioritizing it as one would other service work).

- Fields have to decide if enabling the work of others is research in their field.
- Need a discourse on the intellectual content of advancing science by enabling the work of others.
  - How to publish about enabling work? What is generalizable? What types of claims can be made? What evidence is needed?

References

- Allen, A., Aragon, C., Becker, C., Carver, J., Chis, A., Combemale, B., … Vinju, J. J. (2017). Engineering Academic Software (Dagstuhl Perspectives Workshop 16252). *Dagstuhl Manifestos*, *6*(1), 1–20. https://doi.org/10.4230/DagMan.6.1.1
- Howison, J., & Herbsleb, J. D. (2011). Scientific software production: incentives and collaboration. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (pp. 513–522). Hangzhou, China. https://doi.org/10.1145/1958824.1958904
- Star, S. L., & Ruhleder, K. (1994). Steps Towards an Ecology of Infrastructure: Complex Problems in Design and Access for Large-scale Collaborative Systems. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (pp. 253–264). New York, NY, USA: ACM. https://doi.org/10.1145/192844.193021
- Star, S. L., & Strauss, A. (1999). Layers of silence, arenas of voice: The ecology of visible and invisible work. *Computer Supported Cooperative Work (CSCW)*, *8*(1), 2–30.

CiteAs.org: type software name, get requested citation