Software Sustainability and Software Citation

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

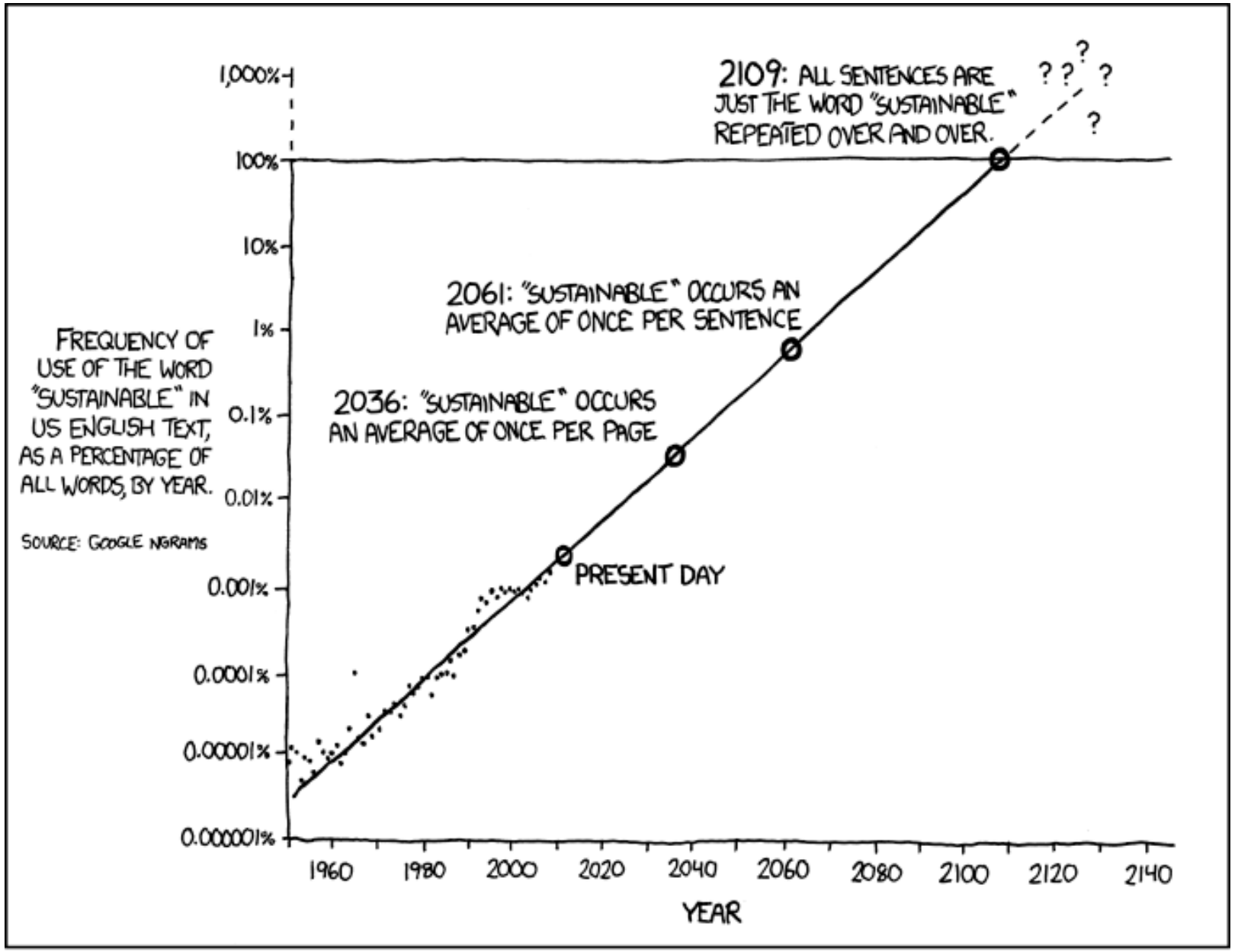
What is sustainability?

Google sustainability

All News Images Books Videos More Settings Tools View saved SafeSearch

environmental business architecture economic poster water social green definition construction design food building fashion engineering energy

ILLINOIS NCSA



THE WORD "SUSTAINABLE" IS UNSUSTAINABLE.

What is sustainability?

- Most often used in the context of ecology, often specifically in the relationship between humans and the planet
- Example: Karl-Henrik Robèrt (via Wikipedia & paraphrased)
  - Natural processes are cyclical but we process resources linearly
  - We use up resources, resulting in waste
  - Waste doesn’t find its way back into natural cycles; not reused or reassimilated
  - Call for "life-styles and forms of societal organization based on cyclic processes compatible with the Earth's natural cycles"

Software sustainability

Software sustainability for whom?

- Users
- Funders
- Managers
- Developers (Maintainers)

Software sustainability for users

- The capacity of the software to endure
- Will the software will continue to be available in the future, on new platforms, meeting new needs?

- Really:
  - Shopping
  - With elements of
    - Longevity
    - Robustness
    - Support

Software sustainability for funders

- My definition while an NSF program officer:
- “If I give you funds for this now, how will you keep this going after these funds run out?”
- “… without coming back to me for more funds”

- Really
  - Portfolio management

Software sustainability for managers

- Focused on people, not software
- How do I keep my team going?

- Really:
  - Business
  - Capitalism
  - Entrepreneurship

Software sustainability for developers

- Often focused on resources, not software
  - How do I get the resources needed to keep my software alive and up-to-date?
  - And keep myself supported / employed?
- Counterpart
  - How do I make keeping my software alive and up-to-date use less resources?

- Really
  - Entrepreneurship
  - Community building
  - Software engineering

Software collapse[1]

- 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

[1]http://blog.khinsen.net/posts/2017/01/13/sustainable-software-and-reproducible-research-dealing-with-software-collapse/

Software collapse[1]

- Options similar for house owners facing the risk of earthquakes:
  1. Accept that your house or software is short-lived; in case of collapse, start from scratch
  2. Whenever shaking foundations cause damage, do repair work before more serious collapse happens
  3. Make your house or software robust against perturbations from below
  4. Choose stable foundations
- Very short term projects might do 1 (code and throw away)
- Most active projects choose 2 (sustainability work)
- We don’t know how to do 3 (CS research needed, maybe new thinking)
- 4 is expensive & limits innovation in top layers (banks, military, NASA)

[1]http://blog.khinsen.net/posts/2017/01/13/sustainable-software-and-reproducible-research-dealing-with-software-collapse/

Common elements

- Due to software collapse, bugs, new use cases, there are lots of risks to all parties
  - Users want to make good product choices that pay off in discoveries
  - Funders want to make good investments that pay off in discoveries
  - Managers want to keep staff employed, also create discoveries
  - Developers want their software to be used in discoveries (and want a career)
- (Almost) all want to know, will this software work in the future?
  - What's the risk?
  - And how do developers get recognized?

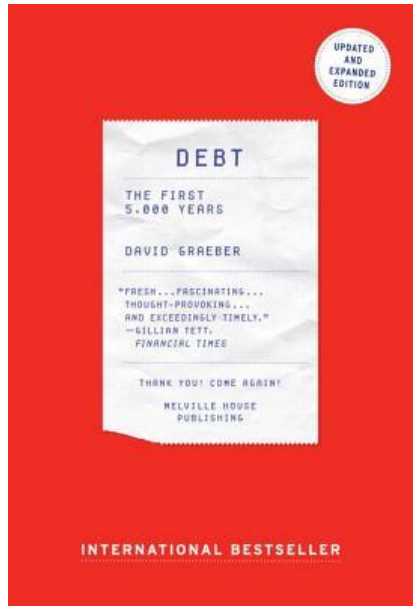
Back to sustainability, in the context of software

- Elinor Ostrom's (Governing the Commons) definition of sustainability for a common-pool resource (CPR): "As long as the average rate of withdrawal does not exceed the average rate of replenishment, a renewable resource is sustained over time."
  - Notion of a cyclic property, though cycle period not specified
  - But rate of what?
- Titus Brown[1]: "the common pool resource in open online projects is effort"
- Sustainability of effort may be appropriate for the developer
  - For effort to be available, need link to recognition, reward, position
- Sustainability of software may be appropriate for the user and funder
  - Rate of what?
- Sustainability of funding may be appropriate for the manager
  - Also helps developers
  - Rate of funding?

[1]A framework for thinking about Open Source Sustainability? http://ivory.idyll.org/blog/2018-oss-framework-cpr.html

“Equations” of software sustainability

- Software sustainability ≡ sufficient Δ software state
  - Sufficient to deal with: software collapse, bugs, new features needed
- Δ software state = (human effort in – human effort out - friction) * efficiency
  - Software stops being sustained when human effort out > human effort in over some time
- Human effort ⇆ $
  - All human effort works (community open source)
  - All $ (salary) works (commercial software, grant funded projects)
  - Combined is hard, equation is not completely true, humans are not purely rational
- Δ software state $\overset{?}{\rightarrow}$ users choose to volunteer effort or $
  - Development choices might take this into account



Debt: The First 5,000 Years
by David Graeber

Software sustainability summary

- Software sustainability means different things to different groups of people
  - Persistence of working software
  - Persistence of people (or funding)

- Can define sustainability as
  - Inflow of resources is sufficient to do the needed work
  - Those resources can (somewhat) be turned into human effort

- Challenges
  - Bring in more resources (funding, people)
  - Reduce the needed work

Why do people contribute to projects?

- 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
- Adding support and reducing friction increase engagement, and also reduce the needed work
- Supporting motivation can increase people's interest
- Hypothesis: Making software citable increases interest in software development and maintenance

*Adapted from Joseph Porcelli

Citing software

- What is software in research?
  - A tool
  - An intellectual contribution
  - An output
- How should work on software be credited?
  - Like a paper, by direct citation
  - Like an instrument, by a parenthetical comment or a footnote
  - Like a contributor, by an acknowledgement
- If software should be cited, what should actually be cited?
  - The software itself
  - A paper about the software
  - The software manual

Software citations today

- Software and other digital resources currently appear in publications in very inconsistent ways
- Howison: random sample of 90 articles in the biology literature -> 7 different ways that software was mentioned

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. http://dx.doi.org/10.1002/asi.23538.

Software Citation Principles

- Consensus after 18 months of discussions in FORCE11 working group, w/ researchers, developers, publishers, repositories, librarians
- Published as
  - 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
- Started with data citation principles, updated based on software use cases and related work, updated based working group discussions, community feedback and review of draft, workshop at FORCE2016
  1. **Importance**
  2. **Credit and Attribution**
  3. **Unique Identification**
  4. **Persistence**
  5. **Accessibility**
  6. **Specificity**
- Paper also included lots of discussion to help use principles

Software Citation Principles

- What is software in research?
  - A tool
  - An intellectual contribution
  - An output
- How should work on software be credited?
  - Like a paper, by direct citation
  - Like an instrument, by a parenthetical comment or a footnote
  - Like a contributor, by an acknowledgement
- If software should be cited, what should actually be cited?
  - The software itself
  - A paper about the software
  - The software manual

Where are we today?

- FORCE11 Software Citation Implementation Working Group in progress
- Lots of good work being done, and good coordination of ongoing activities
- Metadata standards and translation (DataCite Schema 4.1, CodeMeta, citation.cff)
- Open source archiving and identification (Software Heritage)
- Good work and initial acceptance in communities (astronomy, Earth science, math, …)
- Developed document to define ongoing challenges, should release as pre-print/paper in next few weeks
- Software Citation Checklist for Authors document drafted and under review, led by Neil Chue Hong
- Software Citation Checklist for Reviewers document, started by Neil but on hold until the author document is completed
- Repositories task force started, with good community participation
- CodeMeta gaining more community acceptance as a metadata standard, “aligned” with schema.org

What still needs to be done

- Work on documents brings up technical issues/challenges that are not resolved; not clear if they will be useful or will fail to be completed, or perhaps will just need later iterations to improve
  - Complexity of software types: open source, closed source; published, unpublished; versioned, unversioned; developed by citer, not developed by citer; services, containers, executables
  - How to uniquely identify software of each type (ideally as uniformly as possible)
    - Including via new Joint FORCE11 RDA Software Source Code Identification WG
  - How to define and store citation metadata for each type
  - How to access metadata and convert it as needed
  - How to count citations across versions
  - Realization: metadata is fundamental

How to do it

Need groups that work on implementation in context

- Disciplinary communities
- Publishers
- Repositories
- Indexers
- Funders
- Institutions

Groups need to come together, run pilots to establish norms

What you can do

- Think about software sustainability in your projects
  - Make decisions that are best in the long term
  - … that lead to decreased work and increased resources
- Support software developers and maintainers
  - When you hire/promote someone, include their software work
  - Make sure your institution provides appropriate career paths
    - E.g., see https://rse.ac.uk and http://us-rse.org
- Support software citation
  - 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

Acknowledgements

- Discussions with Neil Chue Hong and the UK SSI
- Discussions at various WSSSPE workshops
- Keynote by James Howison at RSE2018
- Discussions with Rob Haines and Caroline Jay at U. Manchester
- Feedback from Matt Turk, James Howison, Dan Sholler
- Arfon Smith, Kyle Niemeyer, my FORCE11 Software Citation Working Group co-chairs
- Neil Chue Hong, Martin Fenner, my FORCE11 Software Citation Implementation Working Group co-chairs

- D. S. Katz, “Scientific Software Challenges and Community Responses,” 2015. https://www.slideshare.net/danielskatz/scientific-software-challenges-and-community-responses
- D. S. Katz, “Fundamentals of Software Sustainability,” 2018. https://danielskatzblog.wordpress.com/2018/09/26/fundamentals-of-software-sustainability/
- C. C. Venters, C. Jay, L. Lau, M. K. Griffiths, V. Holmes, R. R. Ward, J. Austin, C. E. Dibsdale, J. Xu, “Software Sustainability: The Modern Tower of Babel,” Proceedings of Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy 2014), Karlskrona, Sweden. http://ceur-ws.org/Vol-1216/paper2.pdf
- C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, C. C. Venters, “Sustainability design and software: The Karlskrona manifesto,” 37th International Conference on Software Engineering (ICSE’15), 2015. https://doi.org/10.1109/ICSE.2015.179
- P. Johnston, M. Everard, D. Santillo, and K.-H. Robèrt, “Reclaiming the Definition of Sustainability,” *Environmental Science and Pollution Research*, v.14(1), pp. 60-66, 2007. https://doi.org/10.1065/espr2007.01.375

ILLINOIS

NCSA | National Center for
Supercomputing Applications