

```
(defface nano-read-prompt-warning-face
  `((t :foreground ,(face-foreground 'nano-critical-i)
       :background ,(face-background 'nano-critical-i)
       :weight ,(face-attribute 'bold :weight)
       :box ,(face-background 'nano-critical-i)))
  "Face for prompt")
```

```
(defvar nano-read-with-date-map
  (define-keymap
    :parent minibuffer-mode-map
    "<tab>" #'nano-read-with-date--today
    "S-<right>" #'nano-read-with-date--forward-day
    "S-<left>" #'nano-read-with-date--backward-day
    "M-<right>" #'nano-read-with-date--forward-week
    "M-<left>" #'nano-read-with-date--backward-week
    "M-S-<right>" #'nano-read-with-date--forward-month
    "M-S-<left>" #'nano-read-with-date--backward-month
    "S-<down>" #'nano-read-with-date--backward-30m
    "S-<up>" #'nano-read-with-date--forward-30m)
  "Keymap is used in conjunction with the `nano-read-with-date'
  and allows to set the date and time on the right side.")
```

CODE

BEYOND FAIR

Roberto Di Cosmo, Sabrina Granger,
Nicolas Jullien, Konrad Hinsén,
Daniel Le Berre, Violaine Louvet,
Camille Maumet, Clémentine Maurice,
Raphaël Monat & **Nicolas P. Rougier***



```
(defvar nano-read-with-list-map
  (define-keymap
    :parent minibuffer-mode-map
    "<tab>" #'nano-read-with-list--next
    "S-<down>" #'nano-read-with-list--next
    "S-<up>" #'nano-read-with-list--prev)
  "Keymap is used in conjunction with the `nano-read-with-list'
  and allows to select next/prev item on the right side.")
```


"The importance of transparency and reproducibility"

Haibe-Kains et al., arXiv, 2020

Code share

Papers in Nature journals should make computer code accessible where possible.

the obstacles (S. M. Easterbrook *Nature Geosci.* 7, 779–781; 2014). As a leading example of transparency policies in other disciplines, the data journal *GigaScience* requires code used in its papers to be available, and hosts it in a way that allows others to analyse the data in publications. One point made by Easterbrook is that even if the code is shared, others might often make little or no use of it, but on some occasions the take-up will be large. *Nature* and the Nature journals have decided that, given the diversity of practices in the disciplines we cover, we cannot insist on sharing computer code in all cases. But we can go further than we have in the past, by at least indicating when code is available. Accordingly, our policy now mandates that when code is central to reaching a paper's conclusions, we require a statement describing whether that code is available and setting out any restrictions on accessibility. Editors will insist on availability where they consider it appropriate: any practical issues preventing code sharing will be evaluated by the editors, who reserve the right to decline a paper if important code is unavailable. Moreover, we will provide a

In their study, McKinney et al. (*Nature*, 2020) showed the high potential of artificial intelligence for breast cancer screening. However, the lack of detailed methods and computer code undermines its scientific value. We identify obstacles hindering transparent and reproducible AI research as faced by McKinney et al and provide solutions with implications for the broader field.

... More specifically, the authors' description of the model development as well as data processing and training pipelines lacks critical details. The definition of multiple hyperparameters for the model's architecture is missing. The authors did not disclose the parameters used for data augmentation; the transformations used are stochastic and can significantly affect model performance. Details of the training pipeline were also missing...

puter code in all cases. But we can go further than we have in the past, by at least indicating when code is available. Accordingly, our policy now

click on Editorials at: [go.nature.com/xhunqv](http://www.nature.com/nature/focus/reproducibility) ibility, see <http://www.nature.com/nature/focus/reproducibility>. ■

rs Limited. All rights reserved

October 2014

Can you really do less than that?

AVAILABILITY

Readers who are interested in obtaining copies of these programs are invited to send a blank diskette to the author. Readers who are interested in using these programs on machines other than the Commodore 64/128 can write to the author to obtain listings of the programs in BASIC.

Yes, but in 1988 (Russ Poldrack)

“AlphaFold3 Transparency and Reproducibility

An open letter

Wankovic et al., Zenodo, 2024

Code share

Papers in Nature journals should make computer code accessible where possible.

... for the replicability and ...

the obstacles (S. M. Easterbrook *Nature Geosci.* 7, 779–781; 2014). As a leading example of transparency policies in other disciplines, the data journal *GigaScience* requires code used in its papers to be available, and hosts it in a way that allows others to analyse the data in publications. One point made by Easterbrook is that even if the code is shared, others might often make little or no use of it, but on some occasions the take-up will be large.

Nature and the Nature journals have decided that, given the diversity of practices in the disciplines we cover, we cannot insist on sharing computer code in all cases. But we can go further than we have in the past, by at least indicating when code is available. Accordingly, our policy now mandates that when code is central to reaching a paper's conclusions, we require a statement describing whether that code is available and setting out any restrictions on accessibility. Editors will insist on availability where they consider it appropriate: any practical issues preventing code sharing will be evaluated by the editors, who reserve the right to decline a paper if important code is unavailable. Moreover, we will provide a dedicated section in articles in which any information on computer code can be placed. And we will work with individual communities to put together best-practice guidelines and possibly more-detailed rules.

For full details, see our guide for authors at go.nature.com/o5ykhe. For an archive of our content and initiatives concerning reproducibility, see <http://www.nature.com/nature/focus/reproducibility>. ■

NATURE.COM
To comment online,
click on Editorials at:
go.nature.com/xhunqv

rs Limited. All rights reserved

... In this publication, several deviations from our community's standards stand out. First, the absence of available code compromises peer review, a cornerstone of scientific publication and a standard typically upheld by journals. Indeed, one of us (RD) was a reviewer, and despite repeated requests, he was not given access to code during the review...

Nature answer: ...This was not a decision we took lightly, and this editorial briefly explains our reasoning. We think that research, regardless of the sector that does it, should be evaluated through peer review and published for the benefit of society and science. At the same time, we have no wish for this to be the final word. This is an opportunity for an important conversation among all research stakeholders at a time when the majority of global research is privately funded...



Addendum: ...This repository contains all necessary code for AlphaFold 3 inference. To request access to the AlphaFold 3 model parameters, please complete this form. Access will be granted at Google DeepMind's sole discretion...

"The Excel error that changed history

Coy, Bloomberg, 2013

Growth in a Time of Debt
Carmen M. Reinhart, Kenneth S. Rogoff
NBER Working Paper No. 15639
Issued in January 2010
NBER Program(s): IFM ME

We study economic growth and inflation at different levels of government and external debt. Our analysis is based on new data on forty-four countries spanning about two hundred years. The dataset incorporates over 3,700 annual observations covering a wide range of political systems, institutions, exchange rate arrangements, and historic circumstances. Our main findings are: First, the relationship between government debt and real GDP growth is weak for debt/GDP ratios below a threshold of 90 percent of GDP. Above 90 percent, median growth rates fall by one percent, and average growth falls considerably more. We find that the threshold for public debt is similar in advanced and emerging economies. Second, emerging markets face lower thresholds for external debt (public and private)—which is usually denominated in a foreign currency. When external debt reaches 60 percent of GDP, annual growth declines by about two percent; for higher levels, growth rates are roughly cut in half. Third, there is no apparent contemporaneous link between inflation and public debt levels for the advanced countries as a group (some countries, such as the United States, have experienced higher inflation when debt/GDP is high.) The story is entirely different for emerging markets, where inflation rises sharply as debt increases.

A non-technical summary of this paper is available in the April 2010 NBER digest. You can sign up to receive the NBER Digest by email.

This paper was revised on December 5, 2011.

Acknowledgments

Machine-readable bibliographic record - MARC, RIS, BibTeX

Document Object Identifier (DOI): 10.3386/w15639

Published: Carmen M. Reinhart & Kenneth S. Rogoff, 2010, "Growth in a Time of Debt," *American Economic Review*, American Economic Association, vol. 100(2), pages 573-78, May. [View abstract at RePEc](#)

Users who downloaded this paper also downloaded these:

Reinhart and Rogoff	w14656 The Aftermath of Financial Crises
Giovazzi and Pagano	Can Severe Fiscal Contractions Be Expansionary? Tales of Two Small European Countries
Andagna, Caselli, and Lane	w10788 Fiscal Discipline and the Cost of Public Debt Service: Some Estimates for OECD Countries
Caballero	w15636 The "Other" Imbalance and the Financial Crisis
Schulerick and Taylor	w15512 Credit Booms Gone Bust: Monetary Policy, Leverage Cycles and Financial Crises, 1870-2008

Harvard University economists C. Reinhart and K. Rogoff have acknowledged making a spreadsheet calculation mistake in a 2010 research paper which has been widely cited to justify budget-cutting. But the authors stand by their conclusion that higher government debt is associated with slower economic growth.

Reinhart and Rogoff's work showed average real economic growth slows (a 0.1% decline) when a country's debt rises to more than 90% of gross domestic product (GDP) – and this 90% figure was employed repeatedly in political arguments over high-profile austerity measures. When that error was corrected, the 0.1% decline data became a 2.2% average increase in economic growth.

“Top ten reasons to not share your code (and why you should anyway)”

Leveque, Society of Industrial and Applied Mathematics, 2013

1. The proof is too ugly to show anyone else
2. I didn't work out all the details
3. I didn't actually prove the theorem - my student did
4. Giving the proof to my competitors would be unfair to me
5. The proof is valuable intellectual property
6. Including proofs would make math papers much longer
7. Referees would never agree to check proofs
8. The proof uses sophisticated mathematical machinery that most readers don't know
9. My proof invokes other theorems with unpublished (proprietary) proofs
10. Readers who have access to my proof will want user support



*Bad code is always
better than no code
(don't be a panda)*

“FAIR Principles

Principles for scientific data management and stewardship

Wilkinson et al., Scientific Data, 2016



Credits: Ainsley Seago (CC-BY)



Credits: Sangya Pundir (CC-BY-SA)

“FAIR4RS

FAIR principles for research software

Barker et al., Nature, 2022



F — Software, and its associated metadata, is easy for both humans and machines to find. (*F1, F1.1, F1.2, F2, F3, F4*)

A — Software, and its metadata, is retrievable via standardised protocols. (*A1, A1.1, A1.2, A2*)

I — Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards. (*I1, I2*)

R — Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software). (*R1, R1.1, R1.2, R2, R3*)

“FAIR4RS

FAIR principles for research software

Barker et al., Nature, 2022



F — Software, and its associated metadata, is easy for both humans and machines to find. (*F1, F1.1, F1.2, F2, F3, F4*)

A — Software, and its metadata, is retrievable via standardised protocols. (*A1, A1.1, A1.2, A2*)

I — Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards. (*I1, I2*)

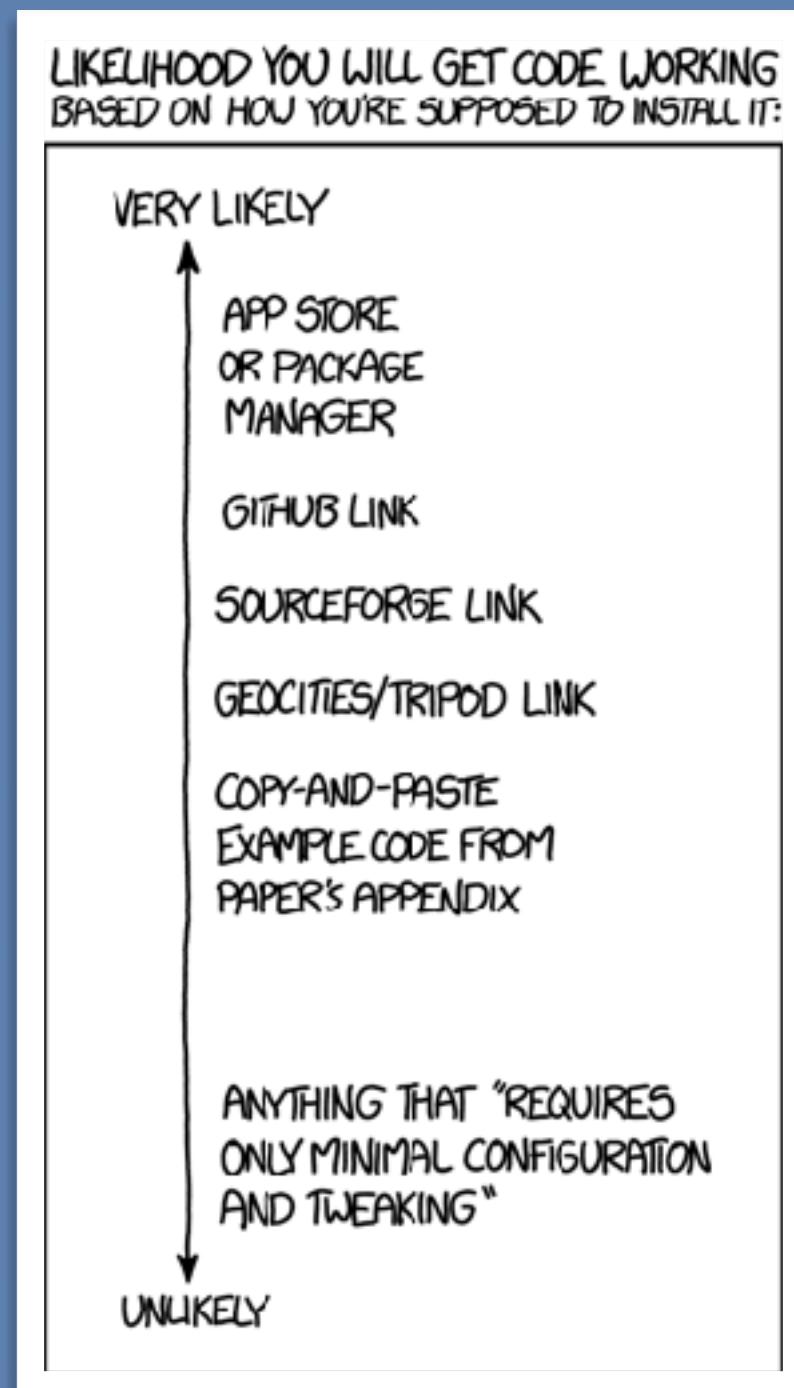
R — Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software). (*R1, R1.1, R1.2, R2, R3*)

MUCH HARDER
THAN IT SEEMS

EVEN
HARDER

“Measuring Reproducibility in Computer Systems Research

Collberg et al., Technical report, University of Arizona, 2014



xkcd.com/1742/

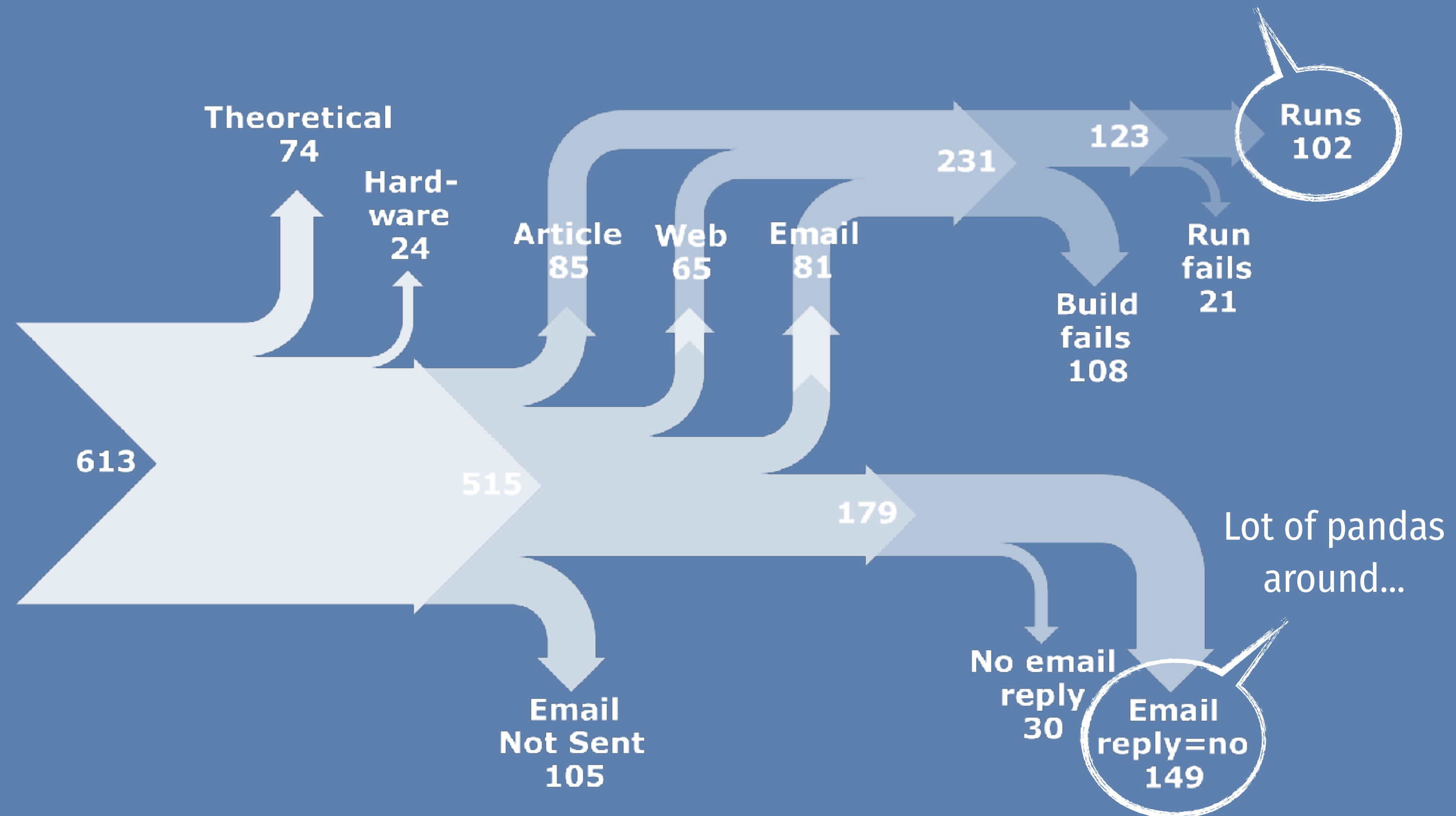
To investigate the extent to which Computer Science researchers are willing to share their code and data, and the extent to which this code will actually build with reasonable effort, during the spring and summer of 2013 we performed the following study. We downloaded 613 papers from the latest incarnations of eight ACM conferences and five journals, all with a practical orientation. For each paper we determined whether the published results appeared to be backed by source code or whether they were purely theoretical.

Next, we examined each non-theoretical paper to see whether it contained a link to downloadable code. If not, we examined the authors' websites, did a web search, examined popular code repositories such as github and sourceforge, to see if the relevant code could be found. In a final attempt, we emailed the authors of each paper for which code could not be found, asking them to direct us to the location of the source. In cases when code was eventually recovered, we also attempted to build and execute it.

“Measuring Reproducibility in Computer Systems Research

Collberg et al., Technical report, University of Arizona, 2014

~20% success runs
(without any guarantee on results)



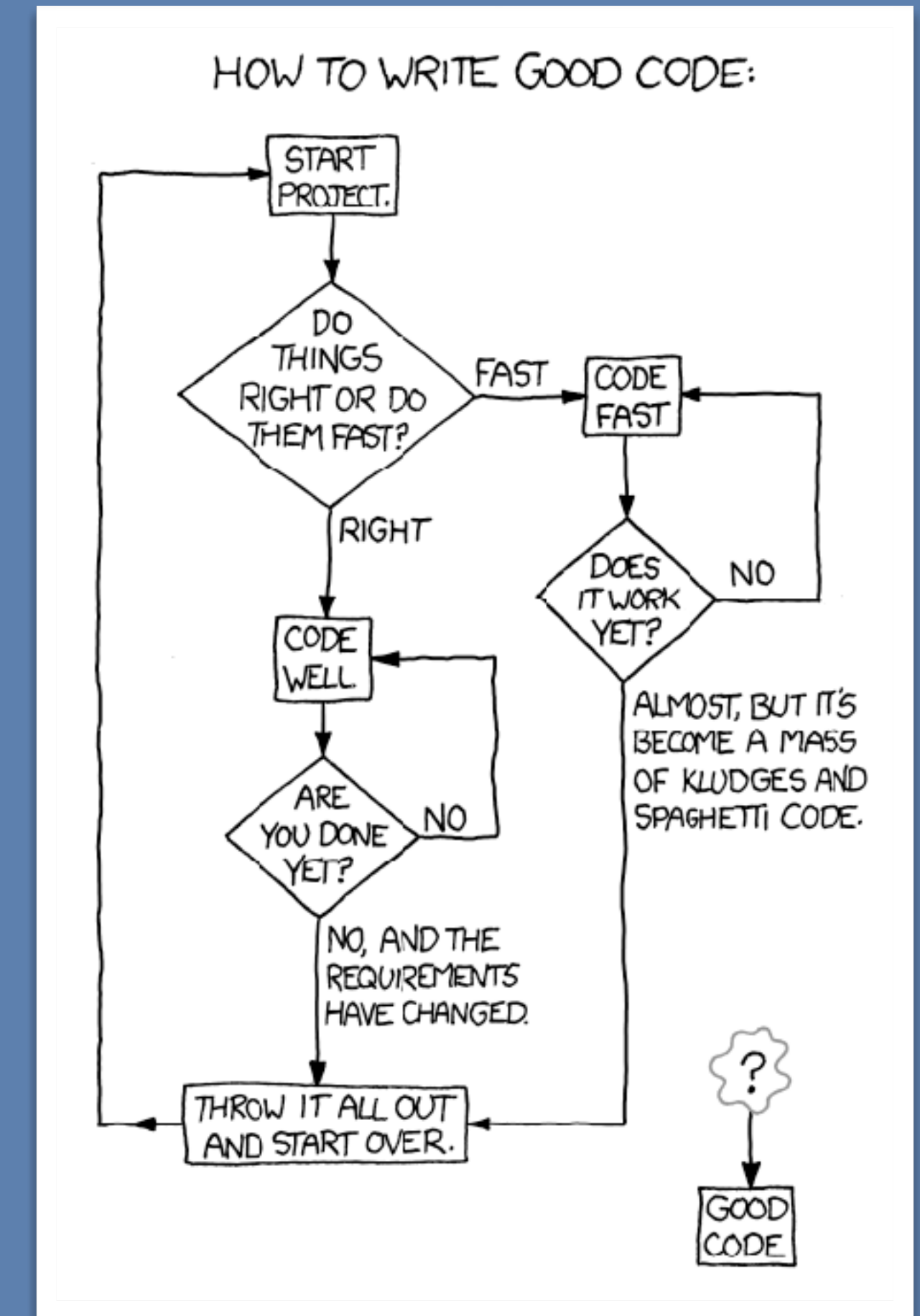
“Re-Run, Repeat, Reproduce, Re-use, Replicate Transforming Code into Scientific Contributions

Rougier & Benureau, Frontiers in Neuroinformatics, 2018

The code should be executable (re-runnable) and produce the same result more than once (repeatable); it should allow an investigator to reobtain the published results (reproducible) while being easy to use, understand and modify (reusable), and it should act as an available reference for any ambiguity in the algorithmic descriptions of the article (replicable).

See also **Terminologies for Reproducible Research**, Barba, 2018

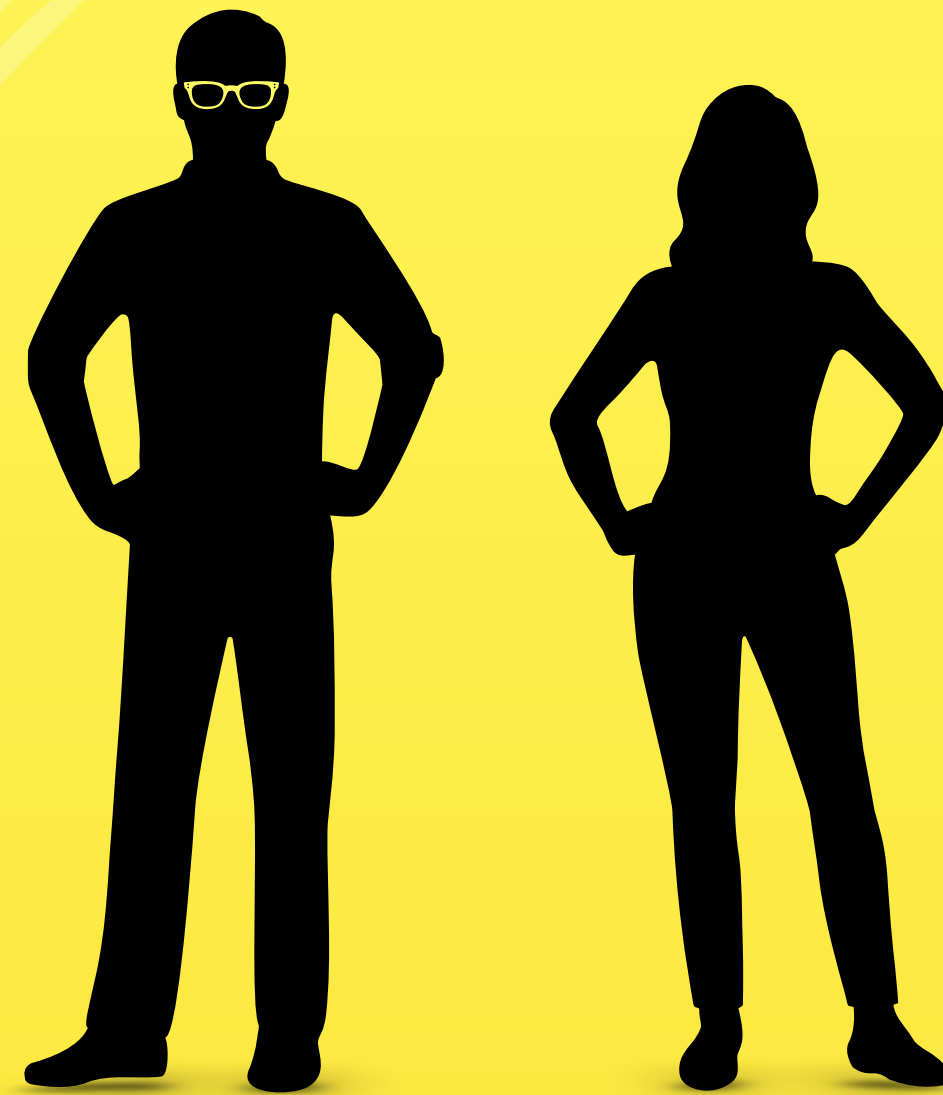
One big problem keeps coming up among those seeking to tackle the issue: different groups are using terminologies in utter contradiction with each other.



TEN YEARS REPRODUCIBILITY CHALLENGE

RESCIENCE SPECIAL ISSUE

FREE TO READ - FREE TO PUBLISH



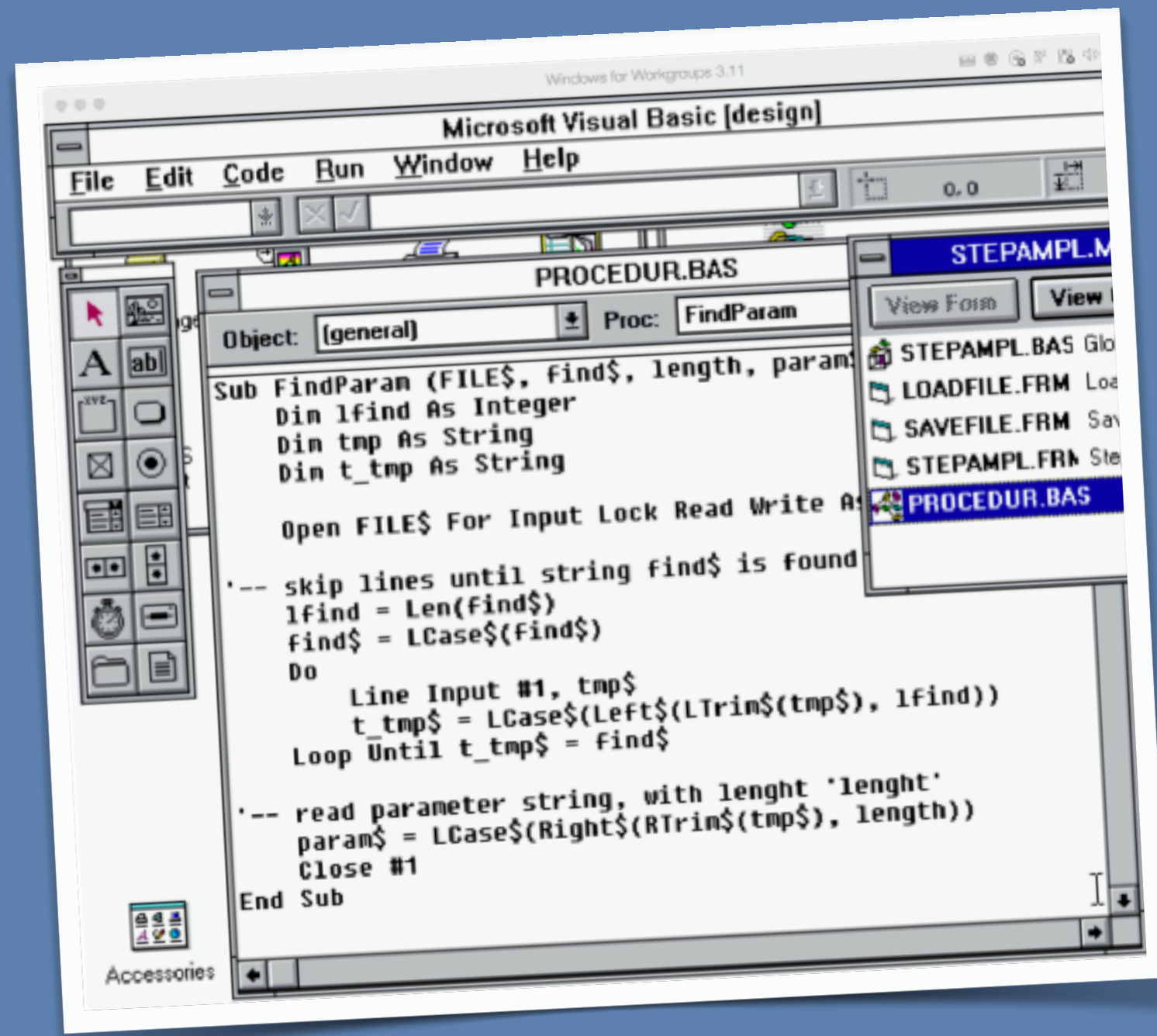
**Would you dare to run the
code from your past self ?**

(the one that does not answer mail)

“Challenge to scientists

Does your ten-year-old code still run?

Perkel, Nature, 2020



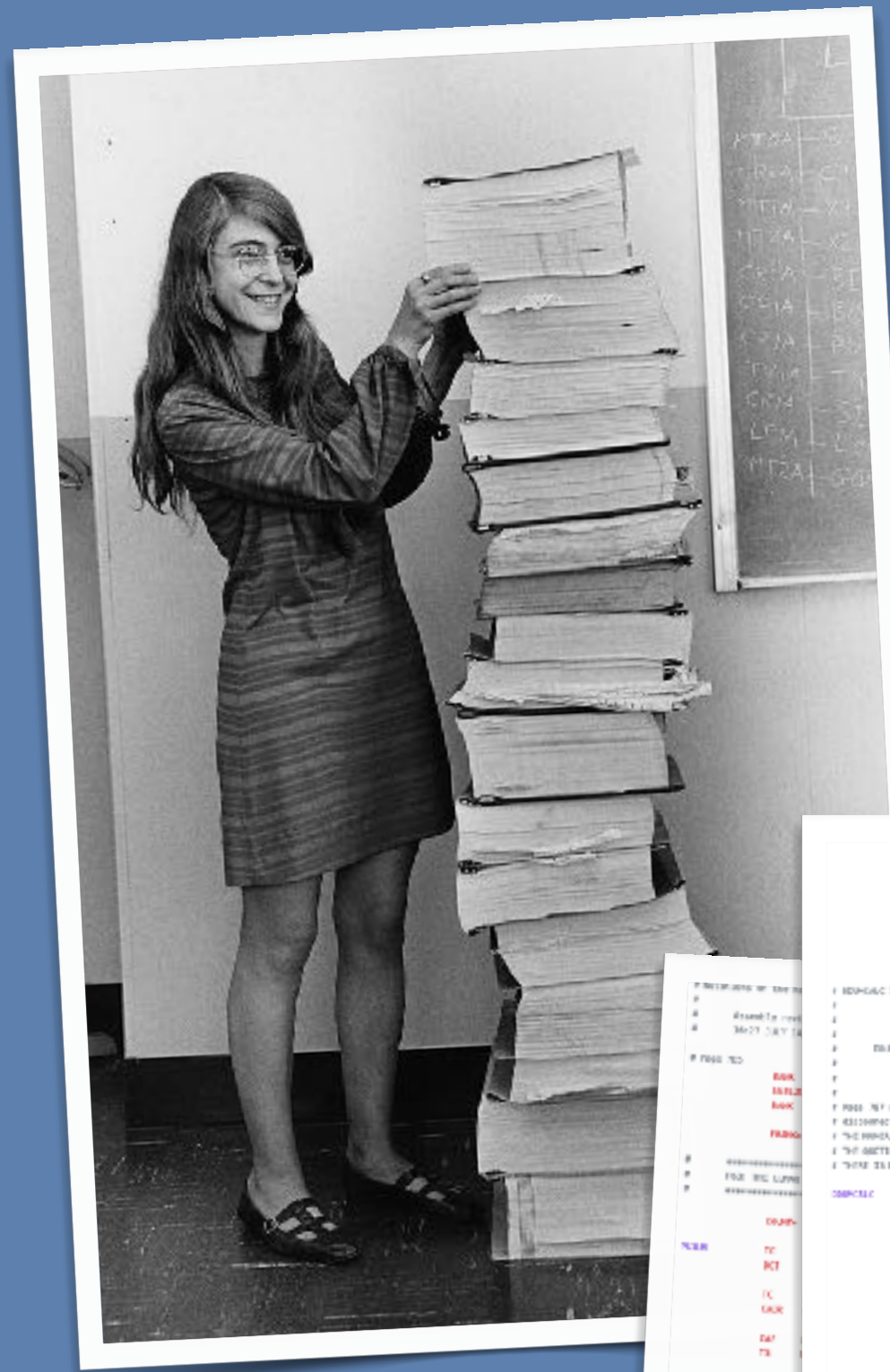
The Ten Years Reproducibility Challenge aims “to find out which of the ten-year-old techniques for writing and publishing code are good enough to make it work a decade later”, **Hinsen** says. It was timed to coincide with the 1 January 2020 ‘sunset’ date for Python 2, a popular language in the scientific community, after 20 years of support. (Development continues in Python 3, launched in 2008, but the two versions are sufficiently different that code written in one might not work in the other.)

In his reproducibility attempt, **Roberto DiCosmo**, a computer scientist at INRIA and the University of Paris, highlighted another common difficulty for challenge participants: locating their code in the first place. DiCosmo tackled a 1998 paper that described a parallel programming system called OcamlP3l. He searched his hard disk and back-ups, and asked his 1998 collaborator to do likewise, but came up empty. Then he searched Software Heritage, a service DiCosmo himself had founded in 2015. “There it was, incredible,” he says.

Software Heritage

The great library of source code

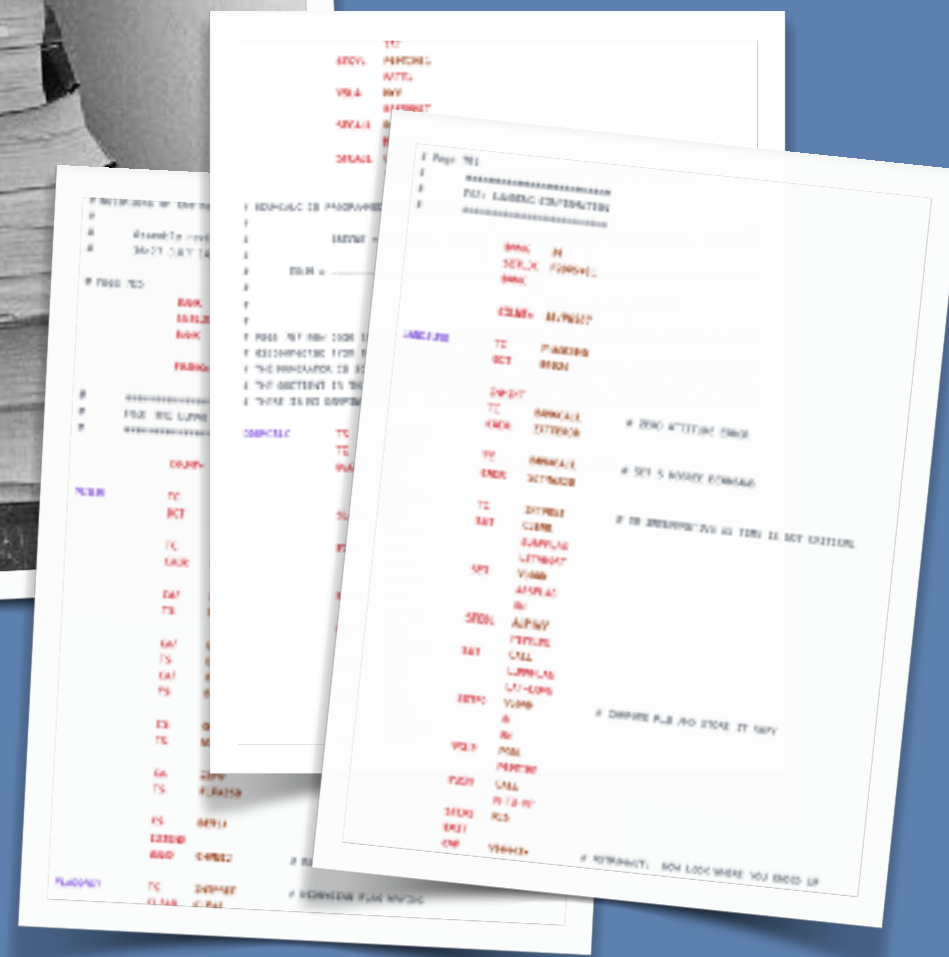
DiCosmo et al., Paris Call - Software Source Code as Heritage, 2017



Margaret Hamilton

We collect and preserve software in source code form, because software embodies our technical and scientific knowledge and humanity cannot afford the risk of losing it.

*Software is a precious part of our cultural heritage. We curate and make accessible all the software we collect, because only by **sharing** it we can guarantee its preservation in the very long term.*



50 years later, Apollo 11 code is still accessible (thanks to paper)

- Google code (RIP, 2015) → archived
- Gitorious (RIP, 2015) → archived
- **GitHub (RIP, 20??) → archived anyway**
(Embrace, Extend & Extinguish)

“Challenge to scientists

Does your ten-year-old code still run?

Perkel, Nature, 2020



Rougier’s entry reproduces the oldest code in the challenge, an image magnifier for the Apple II that he wrote aged 16 and published in a now-defunct French hobbyist’s magazine called Tremplin Micro. (The oldest scientific code in the challenge, described in an as-yet-unpublished paper submitted to ReScience C, was a 28-year-old program written in Pascal for visualizing water-quality data.) Thirty-two years later, Rougier no longer remembers precisely how the code, with its arcane AppleSoft BASIC instructions, works — “which is weird, because I wrote it”. But he was able to find it online and make it run on a web-based Apple II emulator. That, he says, was the easy bit; the hard bit was running it on an actual Apple II.



Free/Libre and Open Source Software (to the rescue)

“Free Software

Free as in speech, not free as in beer

Stallman, 1983



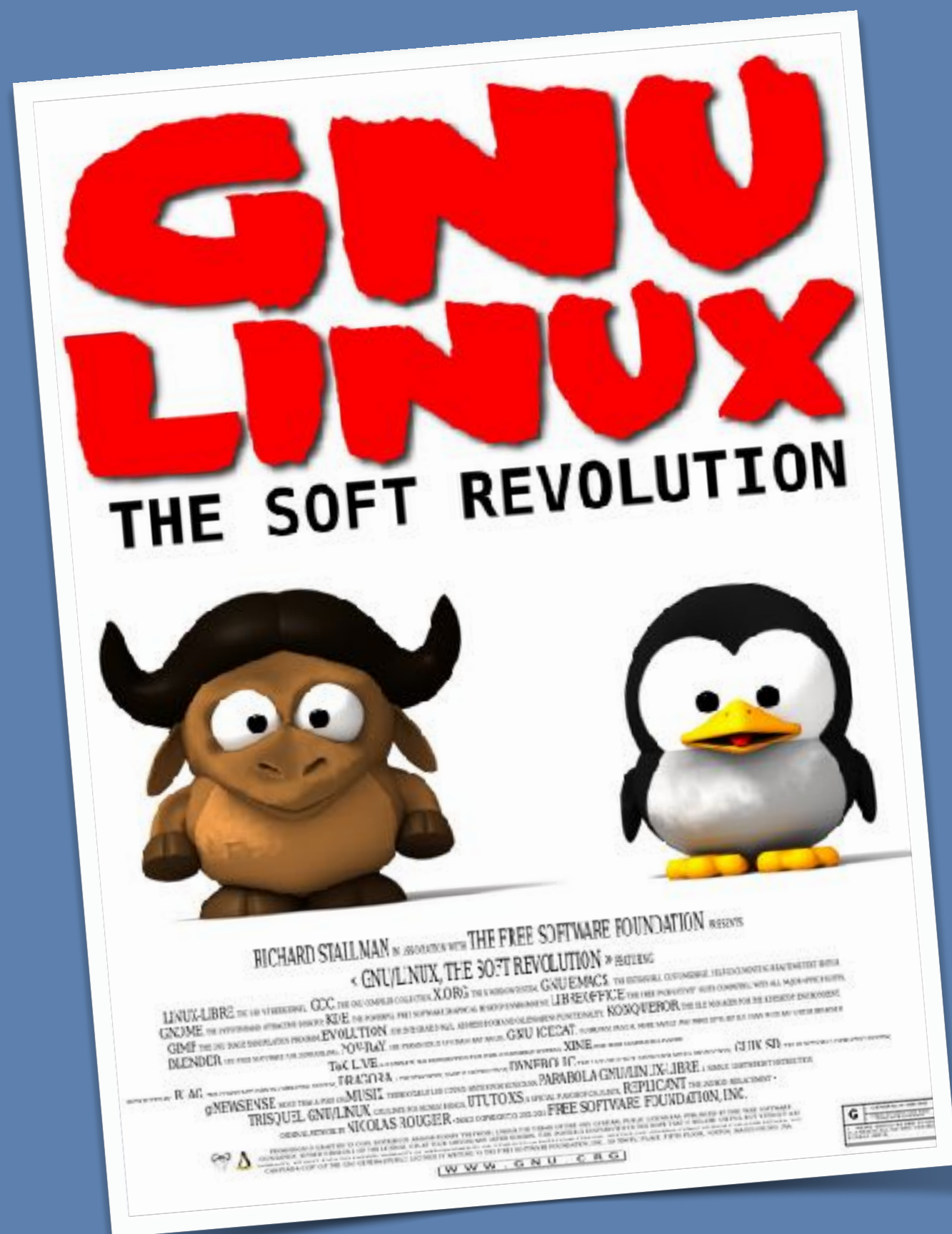
A program is “free software” if the program’s users have the 4 essential freedoms:

- 1. The freedom to run the program as you wish, for any purpose.*
- 2. The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.*
- 3. The freedom to redistribute copies so you can help your neighbor.*
- 4. The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.*

“Free Software

Free as in speech, not free as in beer

Stallman, 1983



In the early years, the distribution of GNU software was primarily done through physical media (floppies, CDs), FTP, BBS systems, and academic networks. The growth of the internet, particularly with the expansion of FTP and later CD-ROM distributions, made it easier for users around the world to access and share GNU software. Stallman and the FSF played pivotal roles in making the software available for free, ensuring that users had the legal right to use, modify, and share the software.

GNU Emacs (GPL), the extensible, customizable, free/libre text editor — and more, is almost **50 years old and still running...**

The Linux kernel (GPL), is more than **30 years old and still running...**

...Perl (1987), GCC (1987), XWindow (1987), BIND (1983), TeX (1978), dc (1970), ...

The Cathedral and the Bazaar

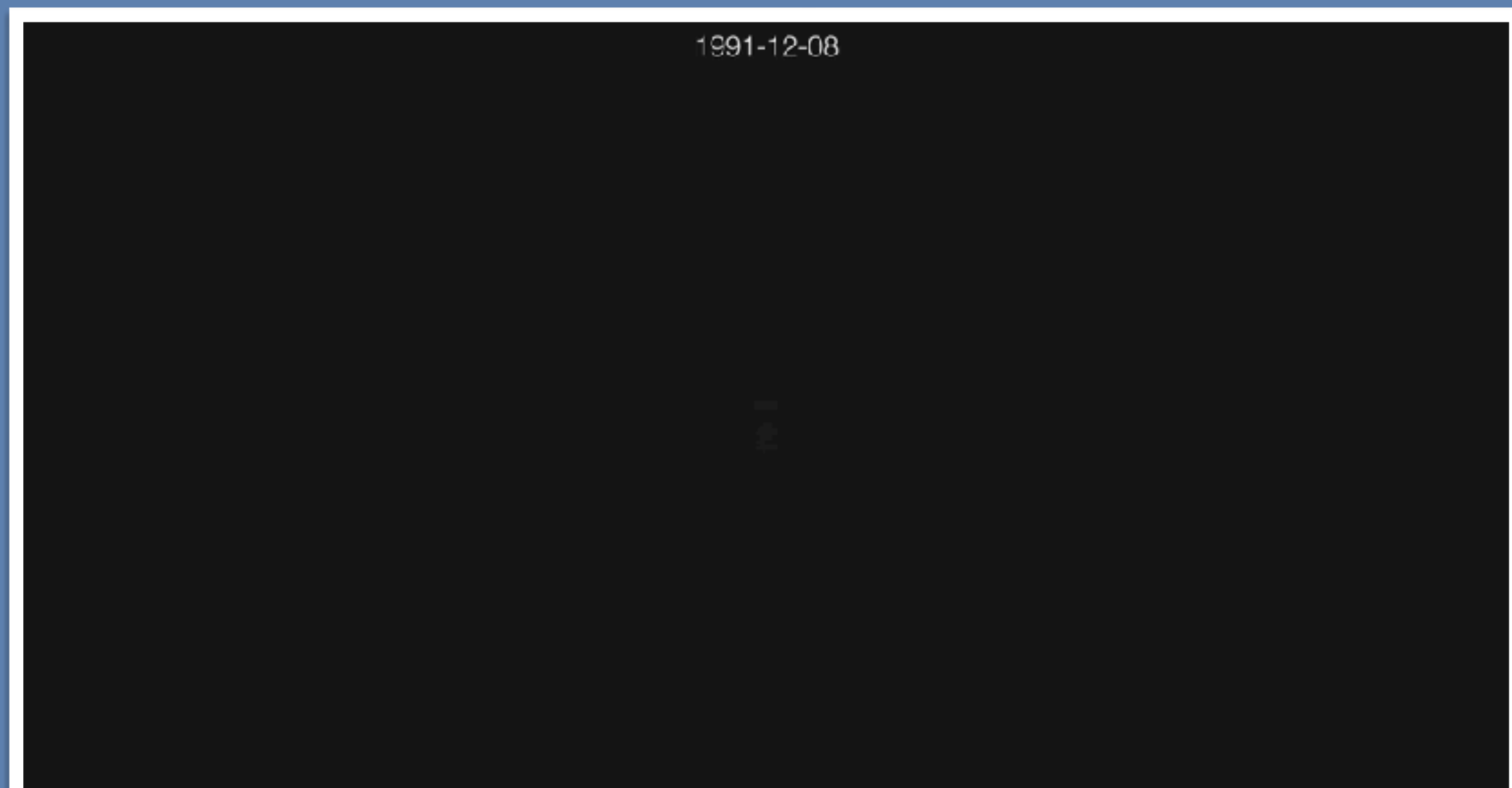
Musings on Linux and Open Source by an Accidental Revolutionary

Raymond, 1999

Found in a reviewed paper about FAIR4RS (sic)

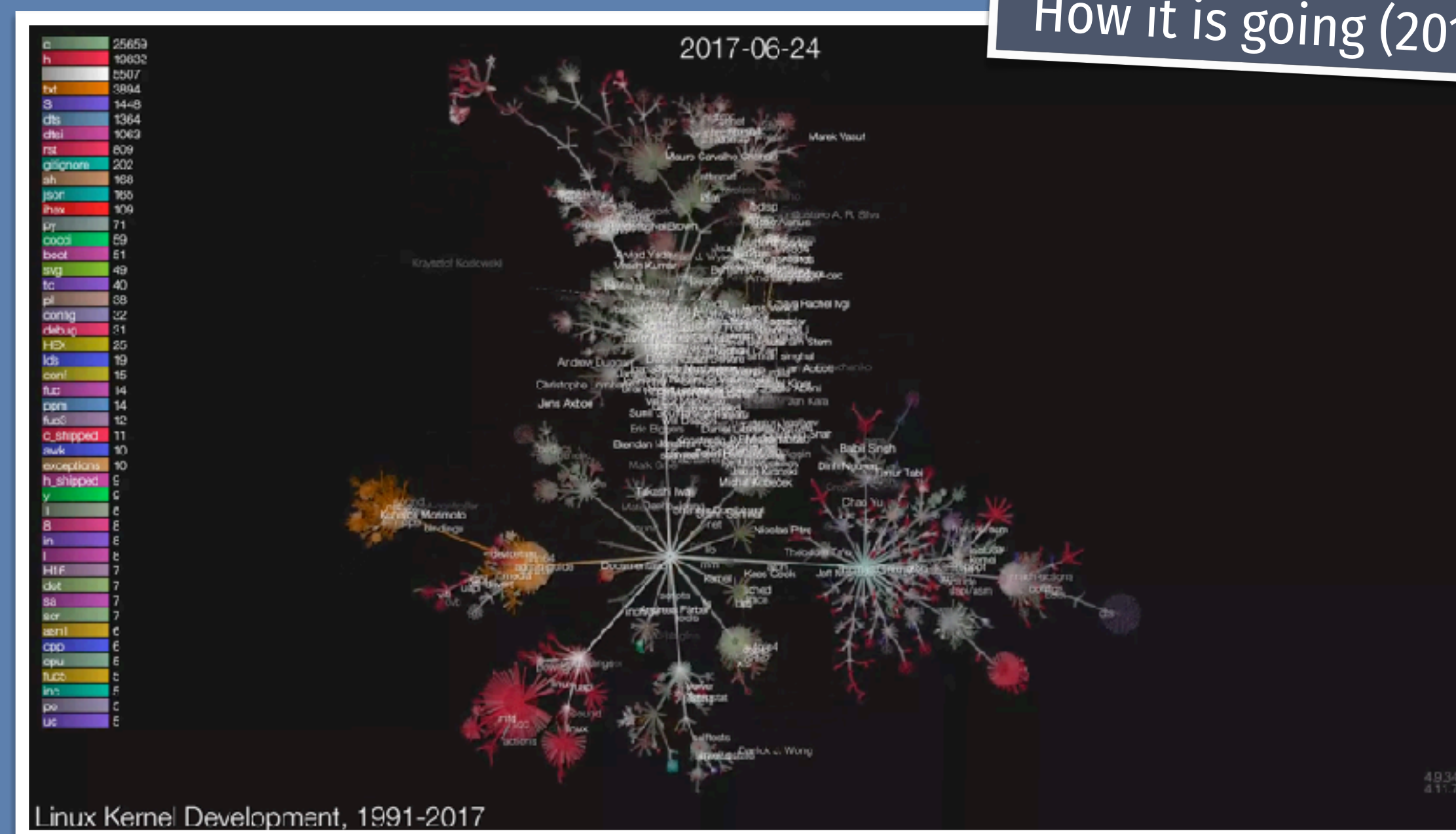
“Research software is **not** required to meet the requirements that are normally a must for other scientific methods: being peer-reviewed, being reproducible and allowing one to build upon another’s work”

~ 10k lines of code (1991)



Linux Kernel Development, 1991-2017

How it started (1991)

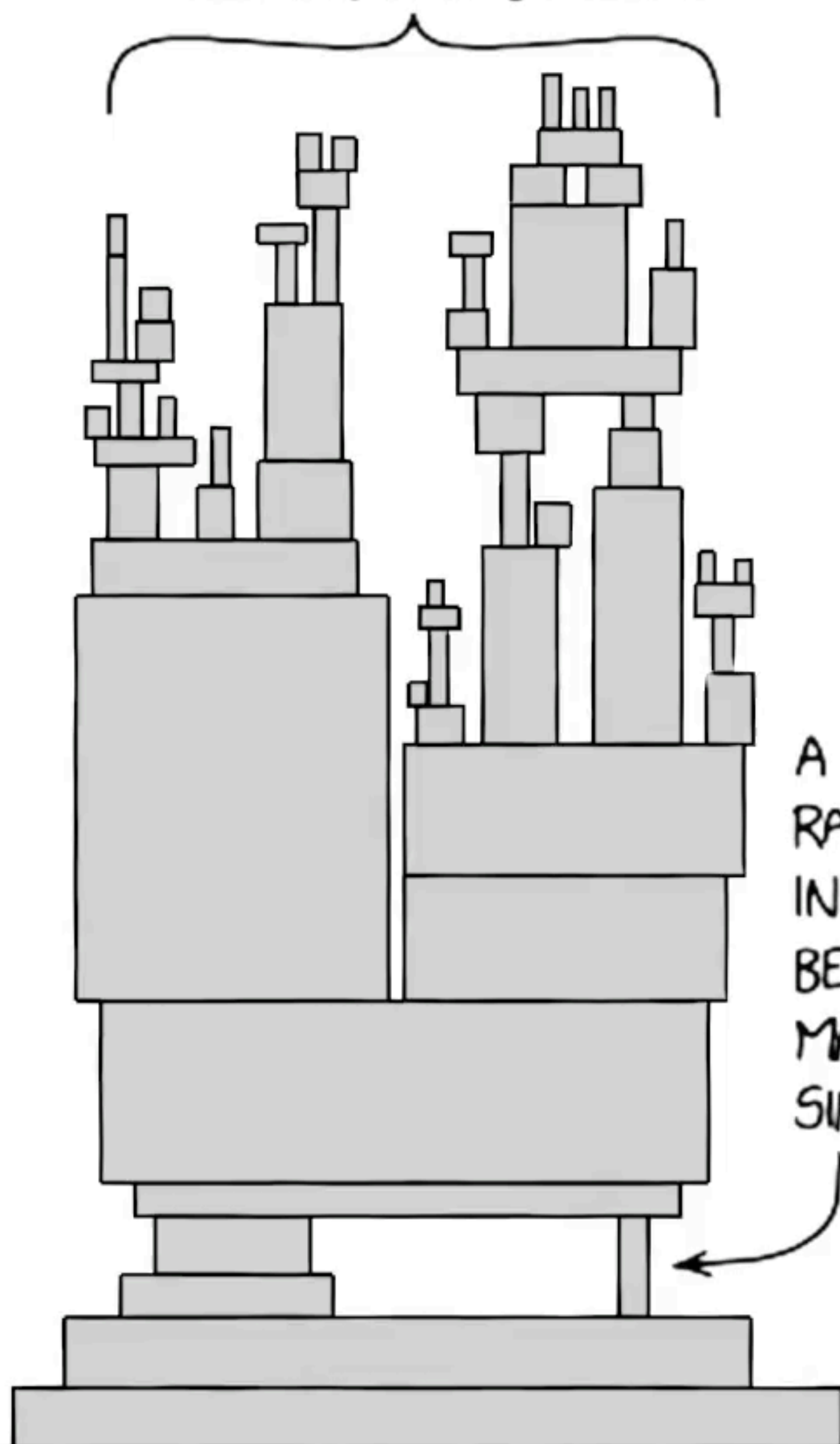


Linux Kernel Development, 1991-2017

~ 29M lines of code (2021)

How it is going (2017)

ALL MODERN DIGITAL
INFRASTRUCTURE



A PROJECT SOME
RANDOM PERSON
IN NEBRASKA HAS
BEEN THANKLESSLY
MAINTAINING
SINCE 2003



There is a real risk. There are people who have suffered real harm to their careers because of their interest in building powerful, new tools rather than writing a few more scientific papers.

Fernando Pérez

No Recognition
(and no funding either)
→ <https://postopen.org/>



for
RESEARCHERS & STAKEHOLDERS

CODE

Collaborate
Open
Document
Execute

STAKEHOLDERS



Libraries



Institutions



Funders



Publishers

RESEARCHERS & STAKEHOLDERS

OPEN

- **Publish your source code on a public forge** mandatory
- **Save your repository on dedicated archive** mandatory
- License your code with an open license (strongly) recommended
- Declare authorship & rightholders recommended

DOCUMENT

- Choose meaningful names recommended
- Comment code recommended
- Provide examples, notebooks & tutorials recommended
- Describe API optional

EXECUTE

- List software & hardware dependencies recommended
- Provide a computational environment optional
- Implement a test suite optional
- Show real-life usage example with expected results optional

COLLABORATE

- Respond to issues & feature requests optional
- Describe maintenance, features & support limits optional
- Explain how to contribute optional
- Build and animate a community optional

INSTITUTIONS

- **Support development** mandatory
- Promote software recommended
- Build institutional forges optional

FUNDERS

- **Long term support** mandatory
- Promote reproducibility recommended
- Facilitate collaborations optional

LIBRARIES

- **Archive software** mandatory
- **Curate software (metadata)** mandatory
- Build catalogs recommended

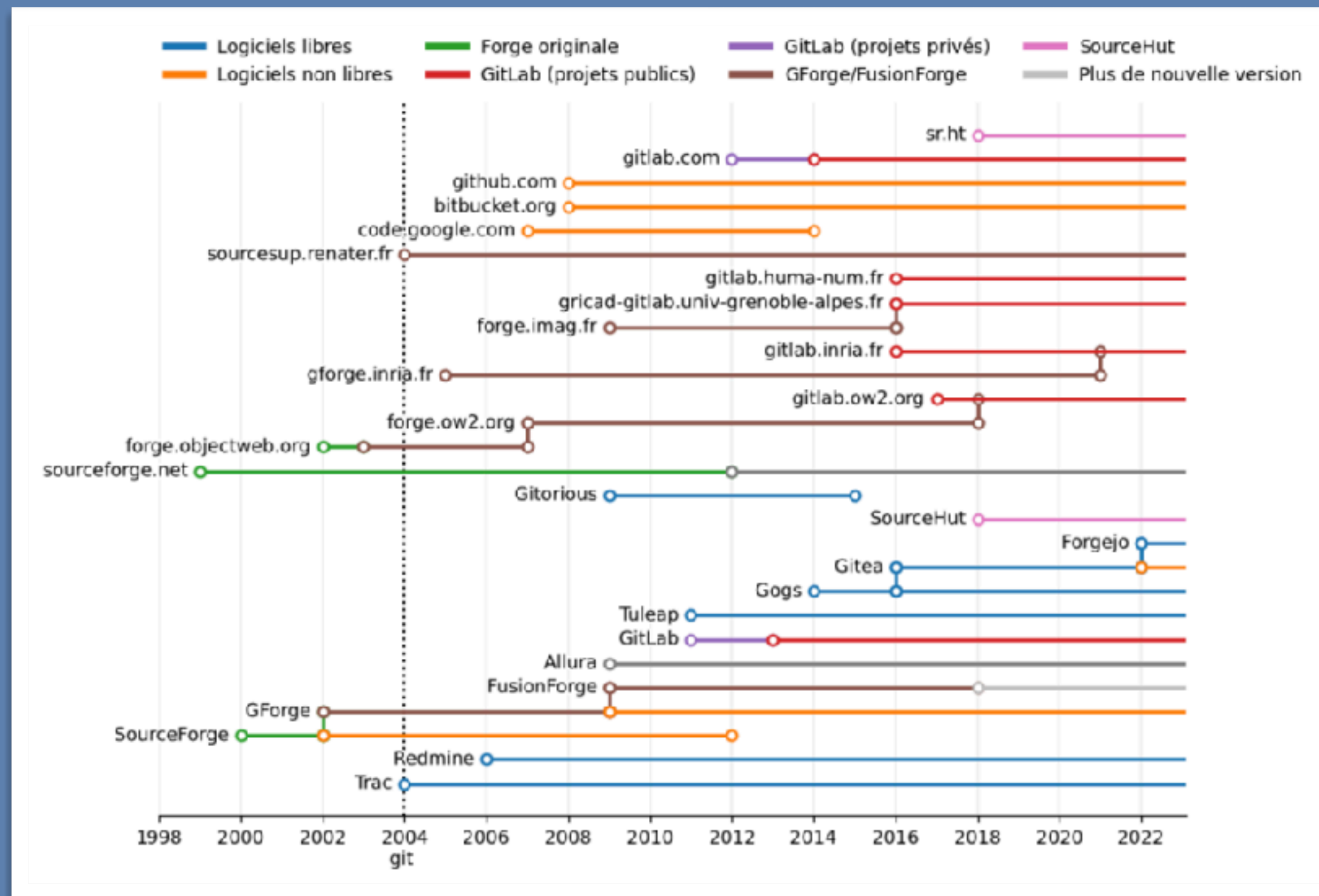
PUBLISHERS

- **Enforce open source** mandatory
- Archive software recommended
- Review software optional

“Les forges logicielles de l’ESR

Définition, usages, limitations et analyse des besoins

Le Berre et al., 2023



Dans l’enseignement supérieur et la recherche, les développeurs de logiciels soutiens ou issus de travaux de recherche ont le choix entre diverses forges pour héberger leur production logicielle. Si leur établissement dispose d’une forge, c’est la solution la plus simple, surtout si aucune interaction en dehors de l’établissement n’est nécessaire.

Quand un besoin d’interaction plus important est nécessaire, les communautés qui développent des logiciels de recherche se tournent fréquemment vers les forges commerciales en ligne, comme en témoignent les lauréats du premier prix science ouverte du logiciel libre de recherche : 9 projets hébergés sur GitHub et un projet hébergé sur SourceForge.

“Prix science ouverte du logiciel libre de la recherche Depuis 2022

Le prix science ouverte du logiciel libre de la recherche récompense les projets et les équipes qui œuvrent au développement et à la diffusion des logiciels libres, et qui contribuent ainsi à la construction d’un bien commun de première importance pour la connaissance scientifique.

Inscrit dans le deuxième Plan national pour la science ouverte et remis par le ministère de l’Enseignement supérieur et de la Recherche, le prix comporte plusieurs catégories qui distinguent les projets selon leurs dimensions scientifique et technique, leur capacité à former et animer leur communauté et la qualité de leur documentation.



CODE

Collaborate
Open
Document
Execute

for
RESEARCHERS & STAKEHOLDERS

Not all scientific software are created equal. Some are meant to be widely spread in the scientific community, to grow and to be maintained in the long term, while others serve solely as a one-shot illustration of a concept or an idea. But both are important for Science.

Imposing some high-level principles indifferently of the nature of the software and the people that create it might be counter-productive.