

Jonathan Starr

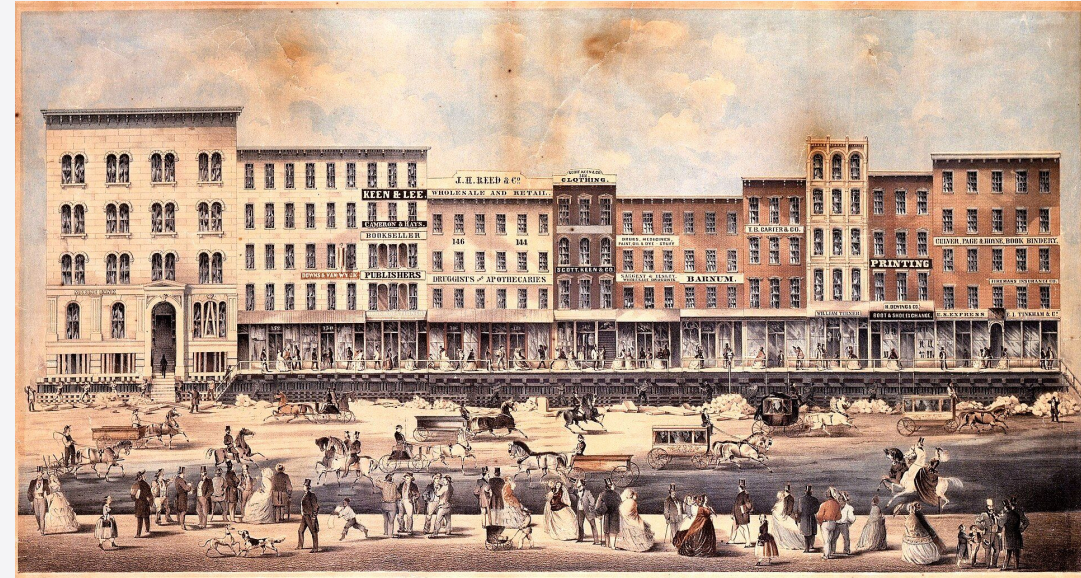
SciOS · The Institute of Open Science Practices · The Open
Source Endowment

scios.tech · iosp.science · endowment.dev



Raising Chicago

What 1850s Chicago knows about
research software



THE BEFORE



Chicago, 1850s. Fastest-growing city in America. Flat, wet, a few feet above the lake.

Everyone solved their own lot.

Private drains. Raised doorsteps. Planks across their own mud.

1849 – 1854

1,400+

Chicagoans dead of cholera in 1854 alone

Six straight years of epidemics. Typhoid and dysentery were simply ambient.

ILLINOIS LEGISLATURE · BOARD OF SEWERAGE COMMISSIONERS · FEBRUARY 14, 1855

The patching ends. A body forms.

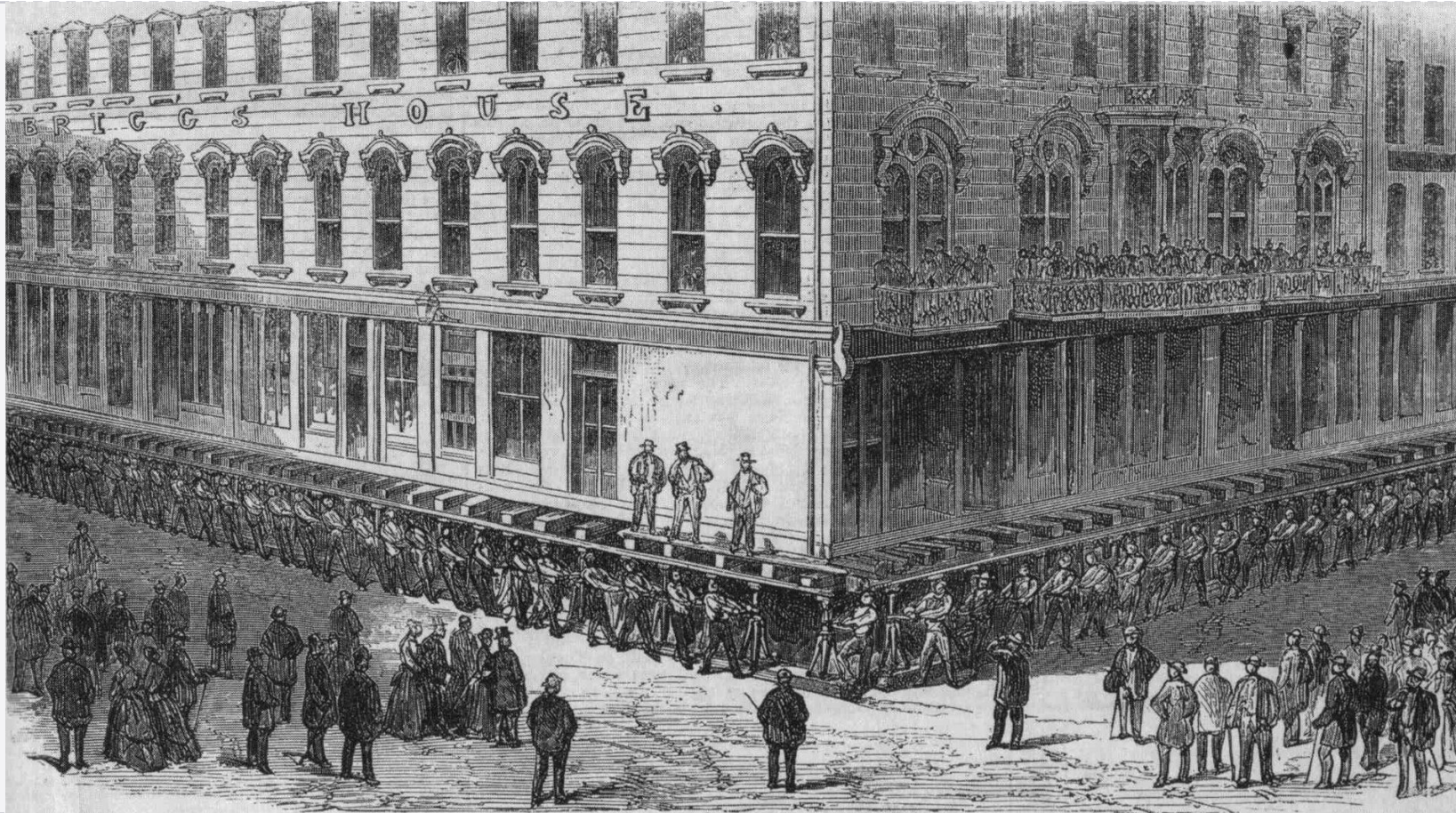
Three commissioners, one engineer. For the first time, somebody's job is to think about the whole city.

CHESBROUGH · ONE PLAN FOR THE WHOLE CITY · 1855

**You can't fight gravity,
so make it work for you.**

Build the sewers above the streets. Raise the city to meet them.

THE RAISING · 1858–1866



One person per screw. Six to ten feet up. Guests still inside

The goal: make the city livable.

The method: raise the city.

The process: ten thousand simultaneous
turns of ten thousand screws.

The body of judgement: The Board of
Sewerage Commissioners

THE RESULT



53.9

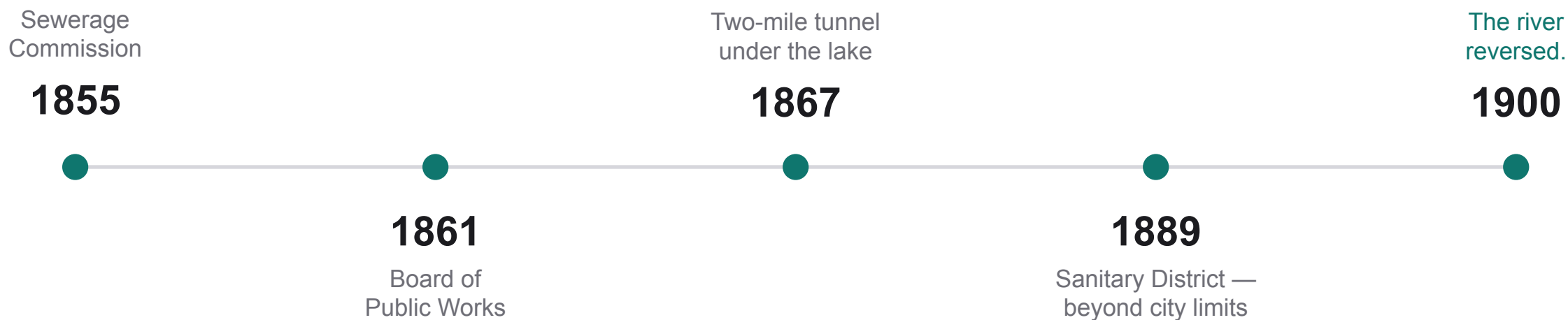
deaths per 1,000 · 1854

18.8

deaths per 1,000 · 1860

**Gravity, the thing that had been killing them,
now ran the sewers.**

The body evolved. The judgment survived. The creativity was encouraged.



Chesbrough died in 1886.

Every city has pipes under it.
So does every discipline.

Ours pipes are made of software.

**What projects do you
contribute to?**

**How much software
exists in your field?**

**How much, typically,
in a single domain?**

NO ONE KNOWS, BUT WHAT I DO KNOW IS...

7 in 10

researchers say their work would not be practical without software

SSI survey · 406 researchers, 15 UK universities

>50%

of NSF- and NIH-funded investigators build that software themselves

Carver et al. 2022 · 1,149 respondents

29%

include maintenance in their proposals
(48% budget new builds · 22% reuse)

Carver et al. 2022

45%

of funder-reported software links point nowhere. The products are missing or broken

Carlin & Rainer 2025

SIDE-NOTE

<https://compute.scios.tech>

Contribute your idle computation cycles to the Great Research Extraction Project (GREP) to find out what software is used in research.

Research software is not special

**Why does research
software fail?**

THE FAILURE MODES OF RESEARCH SOFTWARE

The grant ended

The question got answered; the paper shipped

The users left the field

Duplication

The postdoc left

The PI moved to the next funded thing

Maintenance was never a deliverable

A dependency moved; nobody paid to chase it

Institutional friction – cheaper to reimplement than contribute

Nobody could inherit it – undocumented, untested

Copy-pasted elsewhere, never credited, maintained nowhere

Maintainer burnout – too popular to ignore, too unfunded to sustain

168

tools for visualizing data in Python

pyviz.org/tools.html · updated July 2026

350+

static site generators

one job: Markdown → HTML

958

Linux distributions

637 of them discontinued

THE FAILURE MODES OF OPEN SOURCE SOFTWARE

Built as one-offs by people whose job is something else

Duplication

Reuse isn't fundable

Maintenance is nobody's job

Novelty bias

Silent loss

Bus factor

Burnout

Failed succession

**Everyone works on their
own lots**

**We fund people to work on
their own lots**

2014

Heartbleed

Two-thirds of the internet's encryption ran through OpenSSL maintained with one full-time developer and about two thousand dollars a year in donations.

One underfunded team missed one bug and set the whole internet on fire.

2016

left-pad

Eleven lines of code. One frustrated maintainer deleted them, and that afternoon, build pipelines stopped around the planet.

Nobody knew they depended on one person, until what they depended on was gone.

2018

event-stream

A burned-out maintainer wanted out. There was nobody to hand the project to, so he gave it to the one stranger who asked, then added malware that attacked Copay, a bitcoin wallet app.

There was no where for a dying project to port its capabilities, so they let an untrusted party take the keys.

2021

log4j

A logging library inside effectively every Java system on earth, maintained by sixteen unpaid volunteers.

Log4Shell: log one malicious string, attacker owns the server. 800K attacks in 72 hours

Even a group of 16 might not be enough...

2024

XZ

The sole maintainer of a compression tool inside every Linux machine burned out in public

The only volunteer who showed up was an attacker, who spent two years earning his trust, then planted a backdoor in SSH.

One person to maintain, one person to decide who to trust almost destroyed the internet.

That's the state of the commons.

Nobody knows what they depend on. Nothing ever gets cleaned up.

No one outside a domain knows what goes on inside the domains.

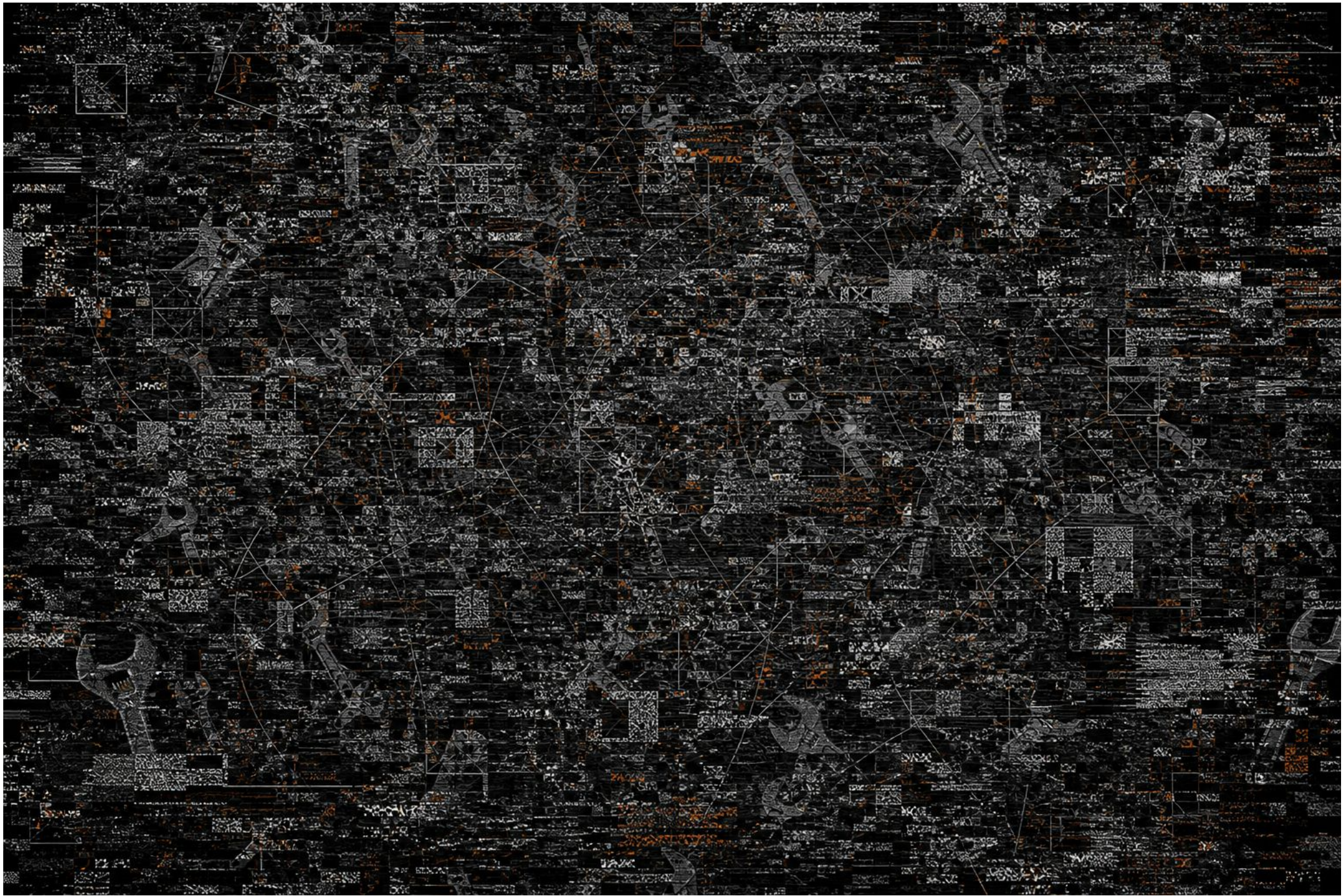
Projects get remade. Contributors burn out. Projects fail. Funders give up.

Everyone suffers for it.

So... why are things this way...

Domain Noise





Domain Noise

Too much software run by too many different projects.

It impossible for anyone, even ourselves, to see what's going on in the space.

Which leads to not knowing what already exists.

Which leads to having to write your own software.

Which leads to funders funding new software.

Which leads to having too many different software projects.

Which slows everything down.

Novelty is the Greatest Good

Not everything is groundbreaking.

Not everything *should* be.

Sometimes the right tool is the one that has been around forever and does 80% of what you need

And that tool, almost always, is the one nobody will pay to support or extend.

Pride in Lines

As beautiful as it is, raising the city of Chicago by ten feet on ten thousand screws is not the purpose of all the engineering we do.

The purpose is to pipe the sewage out of the city. If someone else's code does that better even if mine has done it well for twenty years... that's a good thing!

The Modern Funding Cycle

I have a problem I want solved.

I am surrounded by the most brilliant minds in the world...

Experts in every discipline needed to solve my problem quickly, effectively, and for the long term.

Now: let's split them into tiny groups and make them fight over the money, make new things instead of improving the existing, and then give them enough money to complete one tiny aspect of the ten year project in three years.

48%

new construction

29%

upkeep

22%

reuse

Every failure mode traces to two conditions.

**No collaborative body to hold judgment.
Funding fights collaboration.**

We are stuck in Chicago, pre-1855.

Not everything sucks

pplib

Sandia hands 25 modelers identical inputs: weather data and hardware specs for three real systems. Same published equations. Task: predict the power output.

The answers scattered, up to 15% apart: same company, same named model, same data.

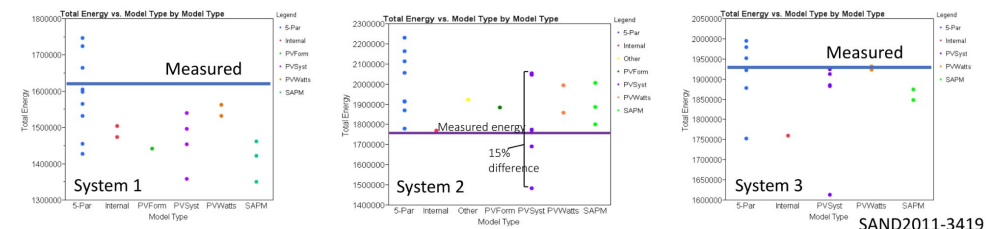
The cause: a "model" was a paper, and each modeler implement its code privately. 25 different implementations of the same equations.

No deliverable said "agree with each other." The modelers compared their code anyway. pplib grew out of it: one shared, tested, public implementation.

2010 1st Blind PV Modeling Comparison



- Weather and system design data provided for 3 systems. Modelers asked to predict power output.
- 25 contributors
- In one case differences of 15% were seen between two modelers using the same model from the same company!



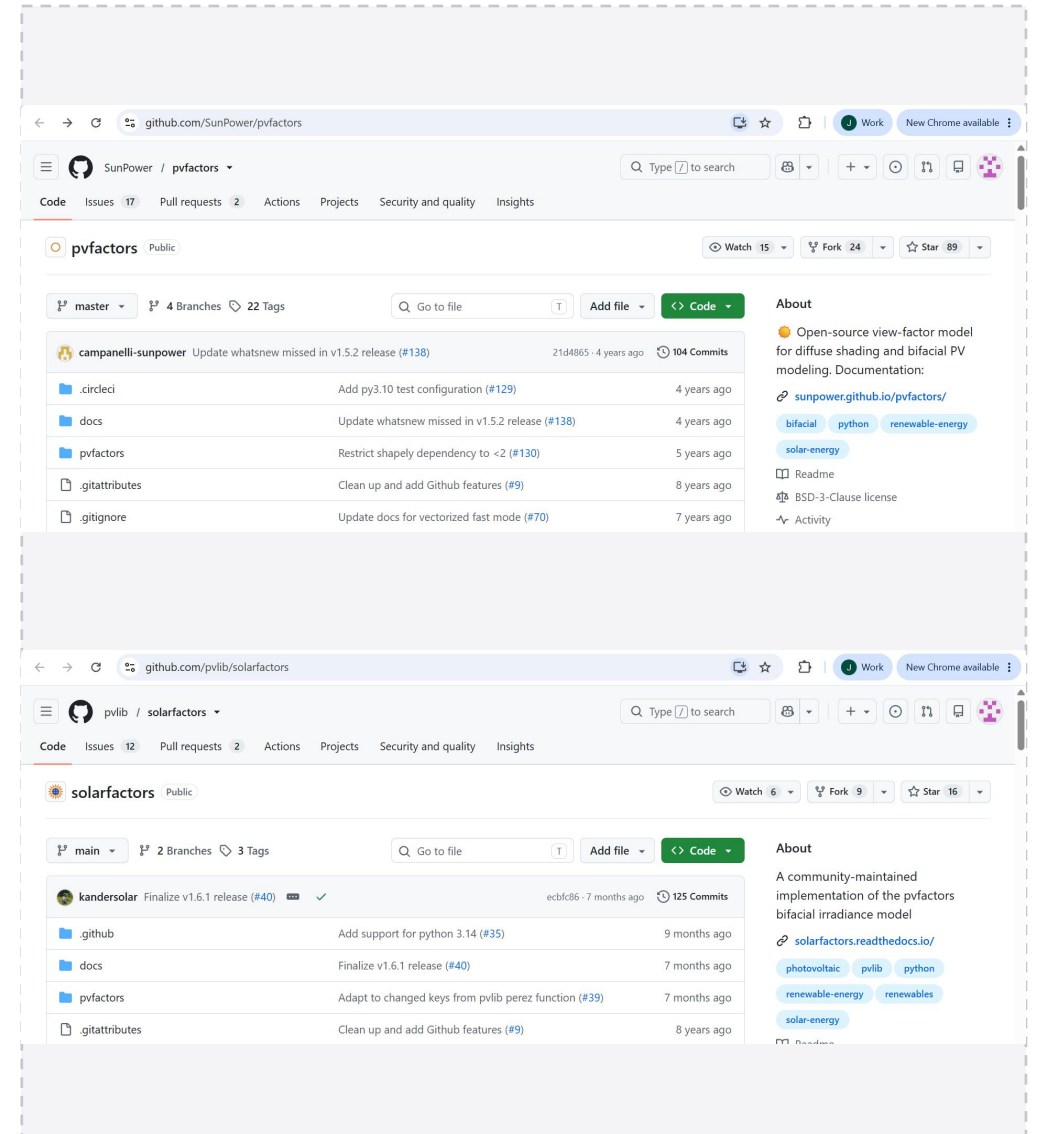
pvfactors

The bifacial math pvlib depended on, built at SunPower.

Abandoned quietly. Last touched March 2022. It rotted; new Python versions wouldn't install it.

No grant covered “adopt a company’s abandoned package.”

Volunteers forked it anyway → solarfactors. Kept the old import name so nothing anyone had written ever broke.



Solar Forecast Arbiter

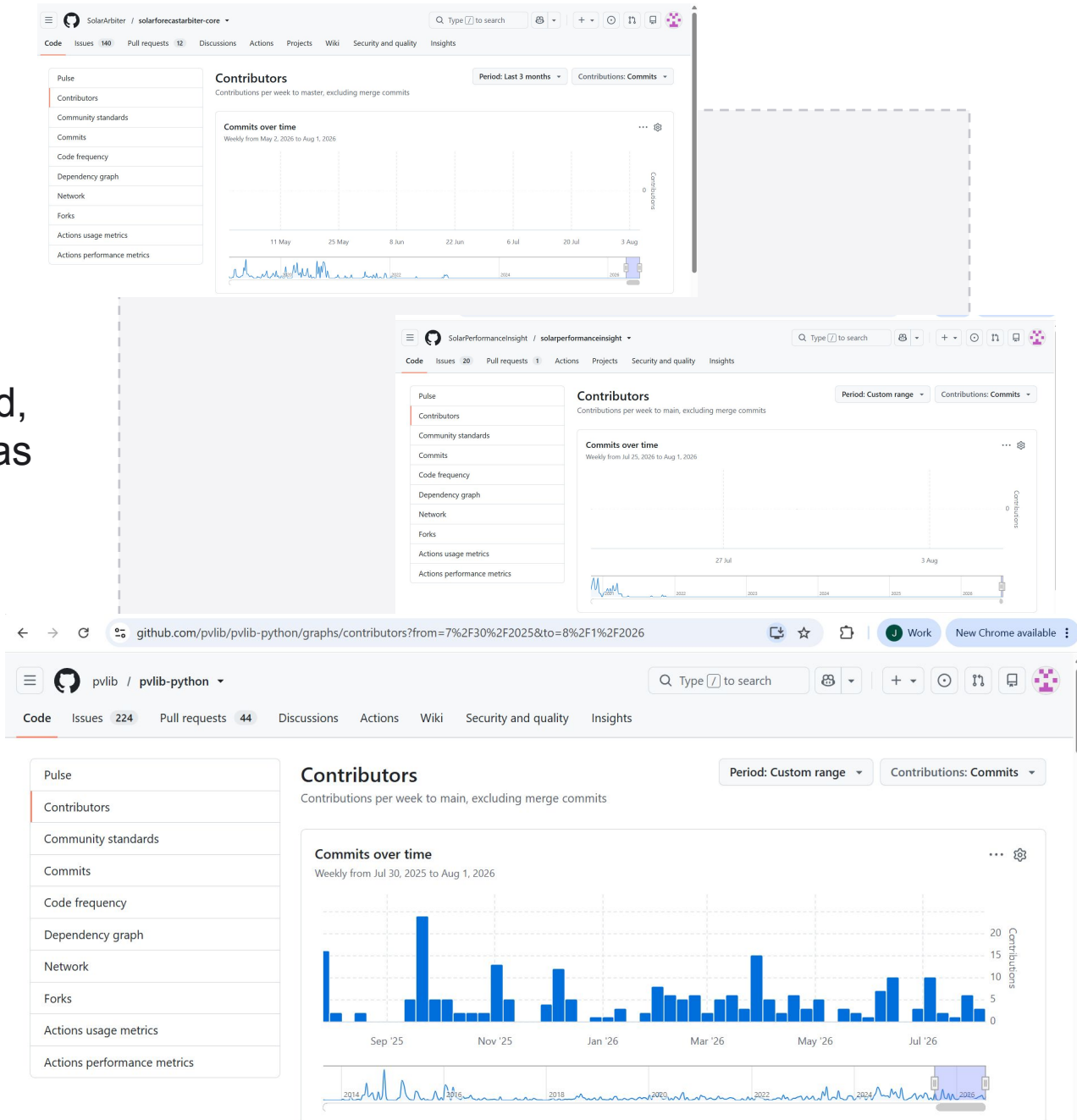
Solar Performance Insight

Each funded to ship a specific deliverable.

To build them, both teams needed pvlib to be improved, so they pushed improvements upstream. None of it was in anyone's work scope.

The grants ended. Now the deliverables sit dormant, quiet since 2021 and 2023.

The spillover in pvlib is still alive.



Research Software is Special

IT COMES WITH A SHARED MISSION, HIGHER THAN THE CODE ITSELF

Scattered genomics scripts → **Bioconductor** · genomics · 2001

Two rival array libraries → **NumPy** · scientific Python · 2005

Everybody's own FITS reader → **Astropy** · astronomy · 2011

Twenty-five silently disagreeing modelers → **pvlb** · 2015

Again and again, people are drawn together.

No mandate, no grant, no deliverable said “merge your work.” The groups formed anyway.

It's not culture. It's not virtue.

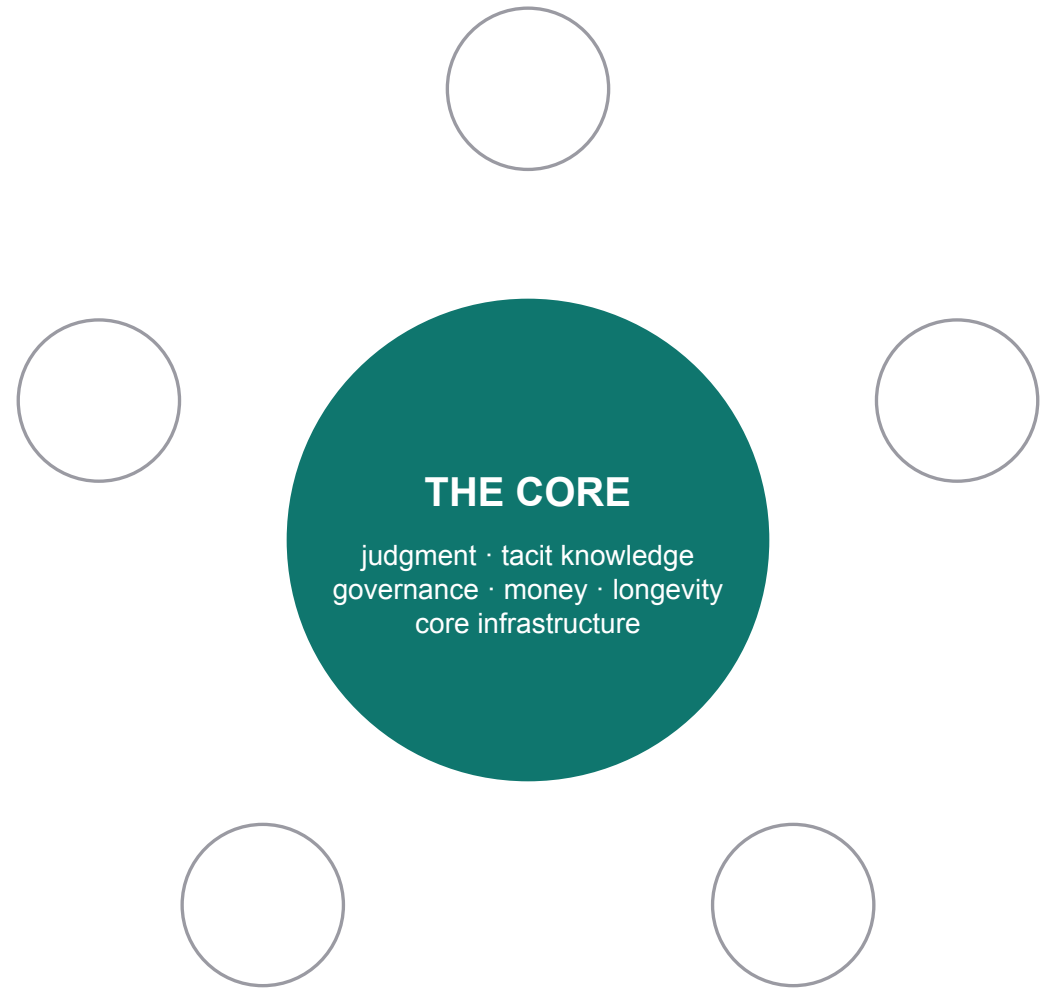
It's Gravity.

**You can't fight gravity,
so make it work for you.**



Chicago's solution was a funded, group of experts tasked with building and maintaining the core infrastructure of the city, expressing their judgement over future infrastructure needs, managing money, and governing those projects..

Core & satellite



Satellites: Niche, temporary, or experimental projects.

What a core does

Holds and, critically, expresses the tacit knowledge of a domain.

Encourages experimentation, and absorbs sunsetting packages, or their features when it is time.

Answers the questions everyone, human or AI agent, asks.

We know our neighborhood.

Dependency graphs can only go so far.

Data and metrics show only so much.

You know your neighbors, their work, and only people that live the mission understand the atmosphere of its neighborhood, in a way a machine never could.

Projects are supposed to innovate. Projects are supposed to end.

A project that ends well is a success. The loss is its capability dying with it.

Not everyone wants to maintain their project that has become popular. You're allowed to plan the handoff, and move on, guilt-free, with the credit you earned.

Knowing where the used features will go when your project dies gives you the freedom to experiment, makes this process of sunseting easier, and makes the ecosystem more stable.

Agents need guidance.

So does everybody else.

Someone wants to do Thing A, what should they use? Where should they start? Does 90% of Thing A already exist somewhere? Did someone start work on Thing A five years ago but had to stop, and should this new person just pick up from where they left off?

Users need to be told what to use. Builders need to be told what was already built. Funders need to be told what needs support, and where gaps exist (and don't). Agents need to be told... literally everything.

Code is getting cheap.

The human element is becoming scarce.

Judgment: what's canonical, what's critical, what's struggling.

Tacit knowledge: why the library is designed that way, which dependency everything quietly rests on, which fork died and why.

Creativity: the leaps that fit on no proposal form. An agent will never tell you to "reverse the river."

Chicago didn't buy a sewer. It bought a mission.

A collaborative body of experts in one room, responsible for the whole city. They were funded to make Chicago livable, not to complete a project.

One screw lifts nothing.
Ten thousand, aligned,
coordinated screws raise a city.

It matters less which screw you turn. What matters is that the sewage flows and the water is clean.

Chat more

Jonathan Starr · jon@scios.tech · scios.tech, iosp.science, endowment.dev