“Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.” - Conway, 1967

Nearly 60 years ago the socio-technical nature of the production of software was becoming apparent. The consequences for the production of software are profound; if you wish to build a different kind of software, redesign your organisation and let the emergent processes do their work.

But it doesn’t seem quite sufficient, perhaps software comes to reflect the environment of its development *and of its use*. This is an indication that the many decades of user advocacy is finally paying off - the user centered design processes of development end up representing the preferences of users to the extent that it slowly shapes the software towards its own end. When engineering companies talk about seeking ‘bake time’ it’s reflecting this instinct that the software that has spent the longest time immersed in the preferences of a population will have accumulated its effect, and so be better suited.

Even software that requires large scale investment, such CAD systems and programming languages, seem to exhibit these properties. Scratch, designed for education, and mass used by children has been a starling success, but the majority of successful no/low-code systems do not use it despite some well funded attempts.

Further it seems that software can only live in tension with the epistemic power of its surroundings for brief periods of time, and at the expense of significant investments of energy by the people building it. This is perhaps most apparent with the mechanisms of funding, that capital logics cease to act solely as a market coordination mechanism between producers and consumers - and instead shape the kind of software that is produced and orient it for maximum extraction. Resisting this pressure has often seemed to only succeed for short periods of time.

Software is renowned for its ease of duplication. But somehow it never really works - clones of software always feel wrong, somehow inauthentic. As if without the thousands of person-years of careful attention and effort, they’re missing a certain something, a quality without a name that it can only collect through years of usage. So if you don’t like what your software is becoming, go find different users/customers, move to a different location, and let the emergent processes do their work.

For the context of the workshop - this leads to two questions about substrates.

1. What is it about the materiality of the software that acts as a form of cultural coral, soaking up the context of development and use and laying it down into the source code? What are the mechanisms by which this happens? Can the design of the substate of software fundamentally shift these mechanisms?
2. If a major form of epistemic power is the disciplining effect of capital, it would follow that to create more interesting software we need more variations in funding vehicles. Where might this come from? Historically shifts in the sources of patronage of the arts helped enable pretty different forms of art - what alternative sources of patronage might be available to creators of new forms of coral that absorb different properties from different environments?

********** PAPER 17 **********
AUTHORS: Luke Church and William Samosir
TITLE: Software as a socio-technical coral

++++++++++ REVIEW 1 (Jonathan Edwards) +++++++++
Socio-technical coral. What a wonderful metaphor! Rephrasing, long and extensive embrace of co-evolved software imbues a "spirit" into it that resists cloning into different contexts of use. From-scratch rewrites fail by dissipating that spirit.

The epistemic power of capital on software is a trenchant observation. I would love to hear your opinion on the impact of AI. It promises (so I am told) to change who owns the means of production.

I appreciate having your perspective add a different dimension to the conversation.

++++++++++ REVIEW 2 (Pavel Bažant) +++++++++
Dear Luke, dear William,

Your question No. 2 caught my attention:
"If a major form of epistemic power is the disciplining effect of capital, it would follow that to create more interesting software we need more variations in funding vehicles. Where might this come from? Historically shifts in the sources of patronage of the arts helped enable pretty different forms of art - what alternative sources of patronage might be available to creators of new forms of coral that absorb different properties from different environments?"
- This is a great question. I like the idea to look at the history of art patronage as a source of insight. Still, I have this intuition that our field might be able to bootstrap itself through commercial projects after all, lessening the dependence on patronage.
- I think we should re-examine funding sources that have produced interesting software in the past. The story of Xerox PARC and Smalltalk is well known, but there are other stories, too. For example, the Symbolic Lisp Machine. Or PLATO. A lot of those projects depended on "political will". Should we lobby for such will? Are there niches with "untapped" political will?
- At many points in history, when a problem was recognised as "infrastructural" (railways, the electric transmission grid), this helped build political will to establish an institution to develop and maintain that infrastructure, associated standards etc.

++++++++++ REVIEW 3 (Gilad Bracha) +++++++++
Somehow I am reminded of the theory that dogs resemble their owners; if cosmetic surgery is too expensive, consider getting a nicer dog. Not that I disagree with Conway's law, but it is rare to see the causality leveraged in the way you suggest, as a design tool. Executives/MBAs try constantly, which is why we have endless reorgs in industry. Yet the software output seems fundamentally the same.

As for your questions. (2) suggests to me we treat software as art. Now go find the Medicis.

++++++++++ REVIEW 4 (Steven Githens) +++++++++
I like the point #2 at the end about variations in funding vehicles, and what that would like if it were more diverse than either corporate funding or academic/non-profit grant funding.

Another random thought I had in reading of the "many decades of user advocacy is finally paying off" and "bake time", is true, but something that seems to have been missed is knowing when to stop. As much as building substrates and finding the needs of our users

is important, we also need to know when to stop, that what we are adding or changing is no longer welcome, or at the very least give the user the choice to exist where they are in their substrate without further evolution.