# On Continuing Creativity

**Colin Clark, Lead Software Architect**

Sepideh Shahi, Senior Inclusive Designer

Inclusive Design Research Centre, OCAD University

dem ave we di people inna
saying we ave no power to affec change

d’bi, r/evolution, *333*, 2011

# Inequality is Growing

- 85% of post-2008 economic growth was pocketed by the richest 1%
- The U.S. ranks 35th out of 37 OECD countries in terms of poverty and inequality
- More than **1 in every 8 Americans are living in poverty**
- Only **64% of U.S. voting-age population was registered to vote** in 2016—a smaller share of potential voters than just about any other OECD country

Philip Alston, UN OHCHR Report on extreme poverty and human rights

# Our Technologies are Complicit

“The United States is one of the world’s richest, most powerful and technologically innovative countries; but neither its wealth nor its power nor its technology is being harnessed to address the situation in which 40 million people continue to live in poverty…

Much more attention needs to be given to the ways in which new technology impacts the human rights of the poorest Americans.”

Technology is practice; "the way things are done around here."

Ursula Franklin

*The Real World Of Technology*, 1989

"Jai guru deva om
Nothing's gonna change my world,
Nothing's gonna change my world.
Nothing's gonna change my world,
Nothing's gonna change my world."

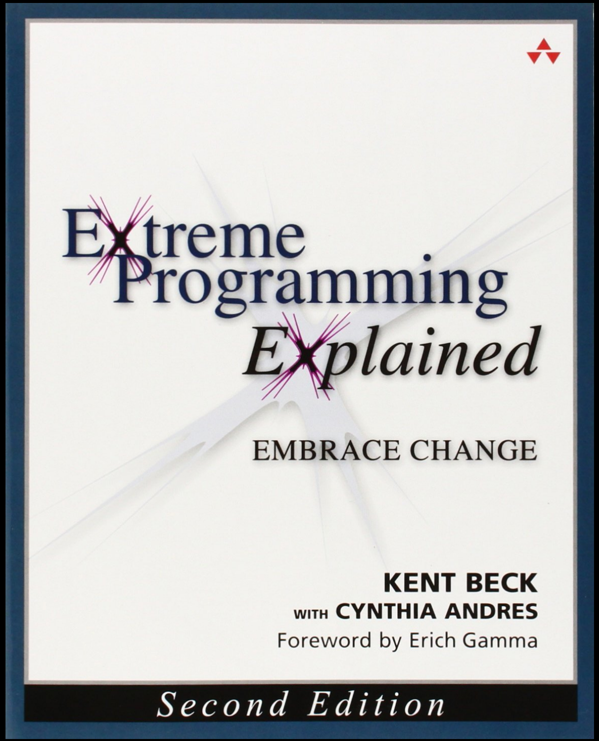
The Beatles
Across the Universe, *Let it Be*, 1970

# The Tyranny of Change

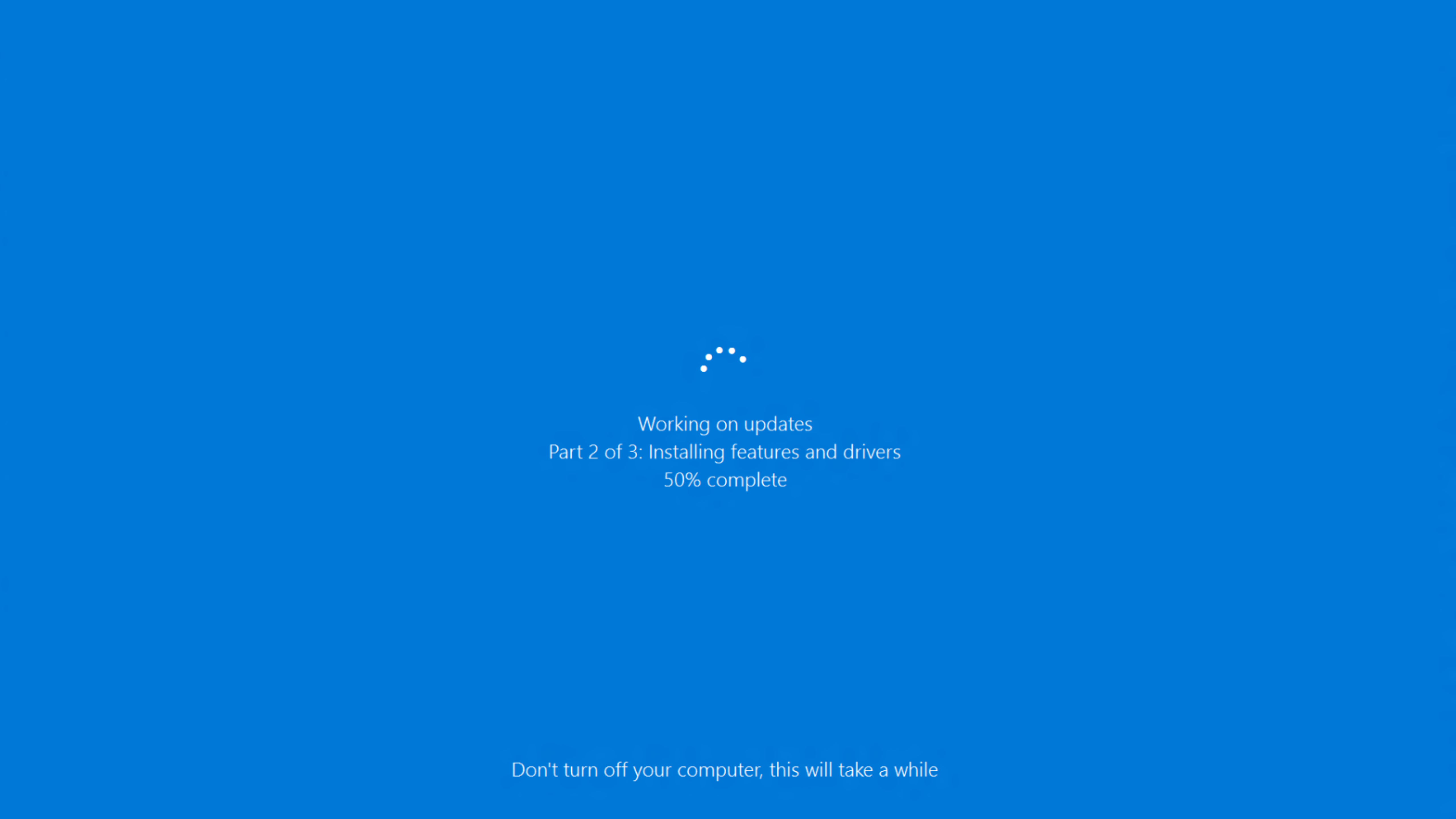
- More than 80% of the total cost of software development is devoted to software maintenance… mainly due to the need for software systems to evolve in the face of changing requirements.
- Unanticipated requirement changes… account for most of the technical complications and related costs of evolving software.
- Kniesel, et. al. (2002) “Unanticipated Software Evolution.” In J. Hernandez and A. Moreira (Eds.): ECOOP 2002 Workshops, LNCS 2548, pp. 92–106.

# Change is Hard for Developers…

- Change has, from the perspective of software practice, inordinately been treated as something to **avoid**, **minimize**, **control**, or **manage**.
- On one hand, methods for requirements management that aim to get it right from the start, and freeze it—change, here, is cast as **risk**
- On the other hand, agile methods cast their gaze inwards; teams "embrace change" but only within a bounded scope—change becomes a choice to be wielded by expert designers and developers alone



B S 灰 A 灰
有 2 有 2
5 年 JPNO 左 3 4
黒 3 黒
P
有 HY 4 1 D
5C TH 黒
3
年 N4 TUF2 B 2 P1
2 有 4 O
PRIU 米 黒 B B
TOYOTA
TOYOTA



Working on updates
Part 2 of 3: Installing features and drivers
50% complete

Don't turn off your computer, this will take a while

# …But change is Intolerable for Users

- For users, software products tend to be a “take it or leave it” proposition; if someone needs something different, often their only recourse is to look elsewhere, at other products
- Beloved features may disappear, move, or be recast by software designers at any time and without notice or permission, leaving users to adapt or relearn their hard-earned workflows
- The power to enact (or forgo) changes, to manage their impacts, scales and timing, rests overwhelmingly in the hands of those who initially created the software—this is, I believe, what constitutes **substantive ownership** of software

# The Failure of Models

- Personas typically “blur” individuals by representing multiples as one—they’re static
- However, we are all multiplicities: dynamic, evolving, and with changing needs and preferences under different circumstances and in different environments
- Industrial design literature emphasizes using personas to reduce diversity and avoid edge cases (See *About Face*, Alan Cooper)
- More importantly, if personas stand in for our absent users during the design process, **where are they**?

“The single story creates stereotypes, and the problem with stereotypes is not that they are untrue, but that they are incomplete. They make one story become the only story."

Chimamanda Ngozi Adichie,
“The Danger of a Single Story”

With revolution in the air
and people changing everywhere.
Step forward now,
no need to hide.

Bad Brains
Joshua's Song, *Rock the Light*, 1983

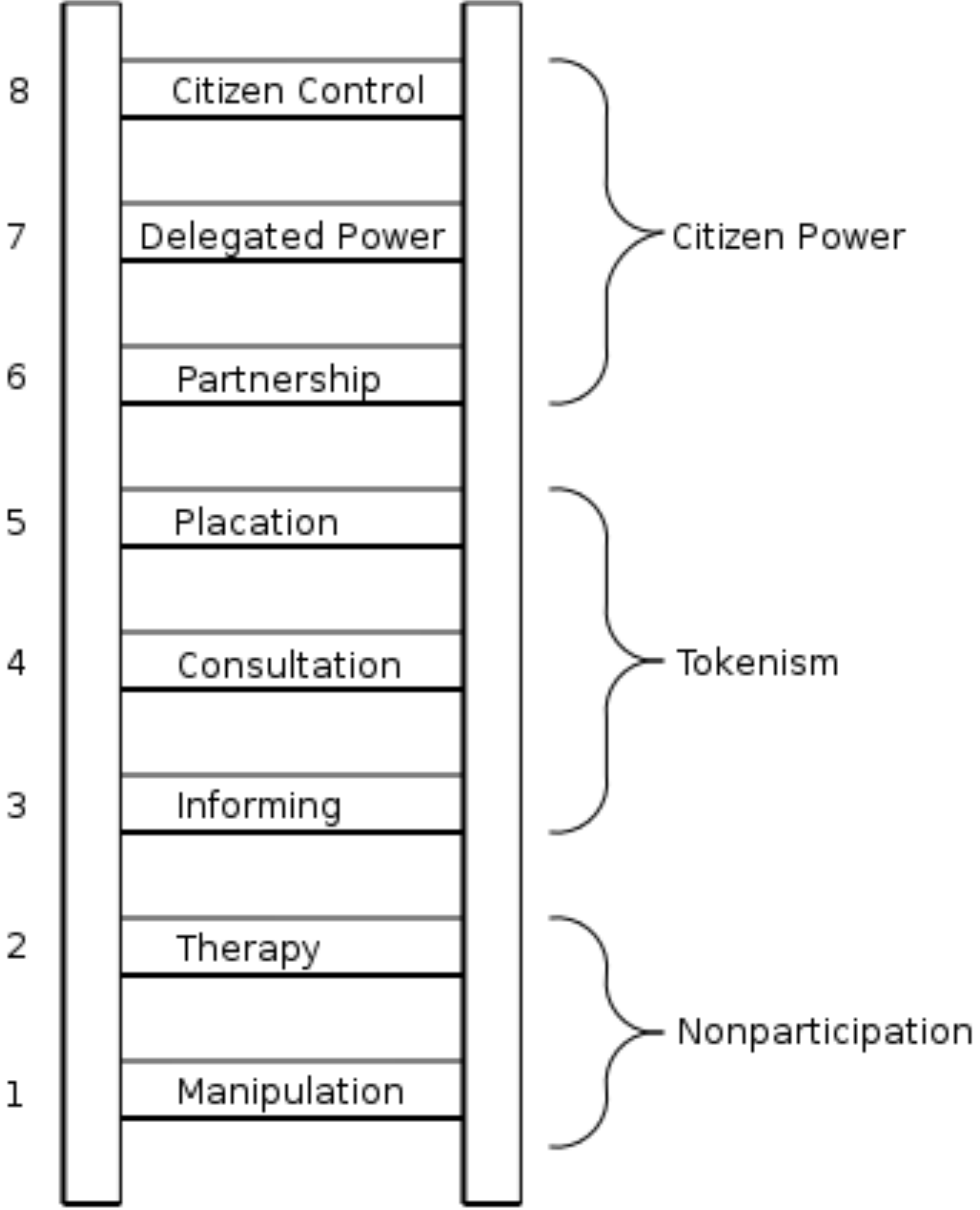
fluid

An open community for co-design, including designers, users, artists, testers, accessibility experts, thinkers, and all the other people who don’t usually fit in open source software communities.

fluidproject.org

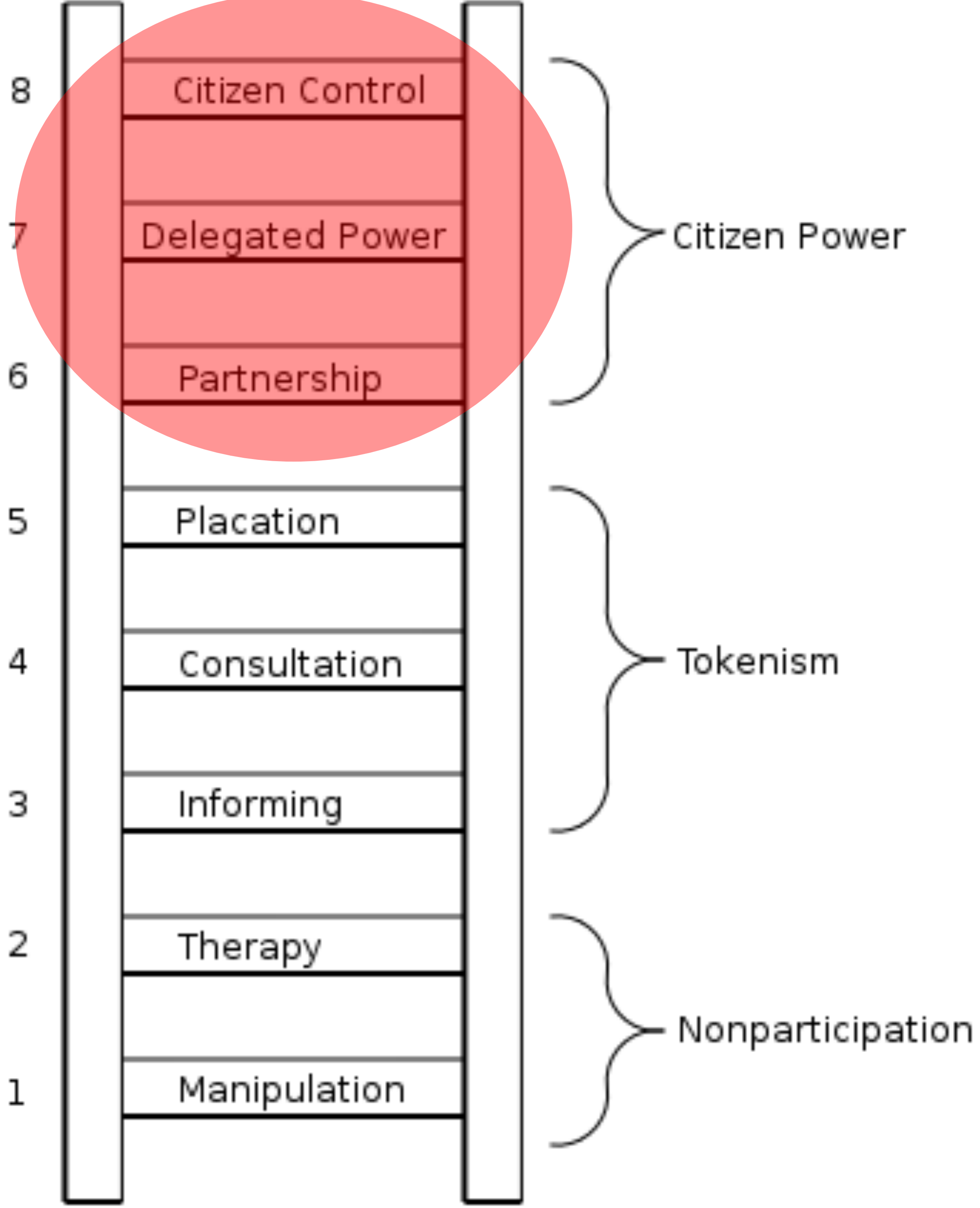
# Co-Design and Community

- Co-design is designing **with**, not simply **for**. It involves asking the people who might otherwise just be "users," particularly those on the margins of today’s technology experiences, to be part of the design process.
- Co-design typically starts with a process of discovering and negotiating roles—asking participants how, when, and how often they want to be involved, and making space to accommodate different “scales” of investment and engagement. It takes time.
- It demands that all participants have equal access to the information—plans, ideas, prototypes, and works in progress—that is essential for full decision-making and responsible contribution.
- A starting point for this involves an opening up of agile’s iterative and incremental processes and open source’s collaboration to be more porous and include a broader range of team participants and modes of engagement.

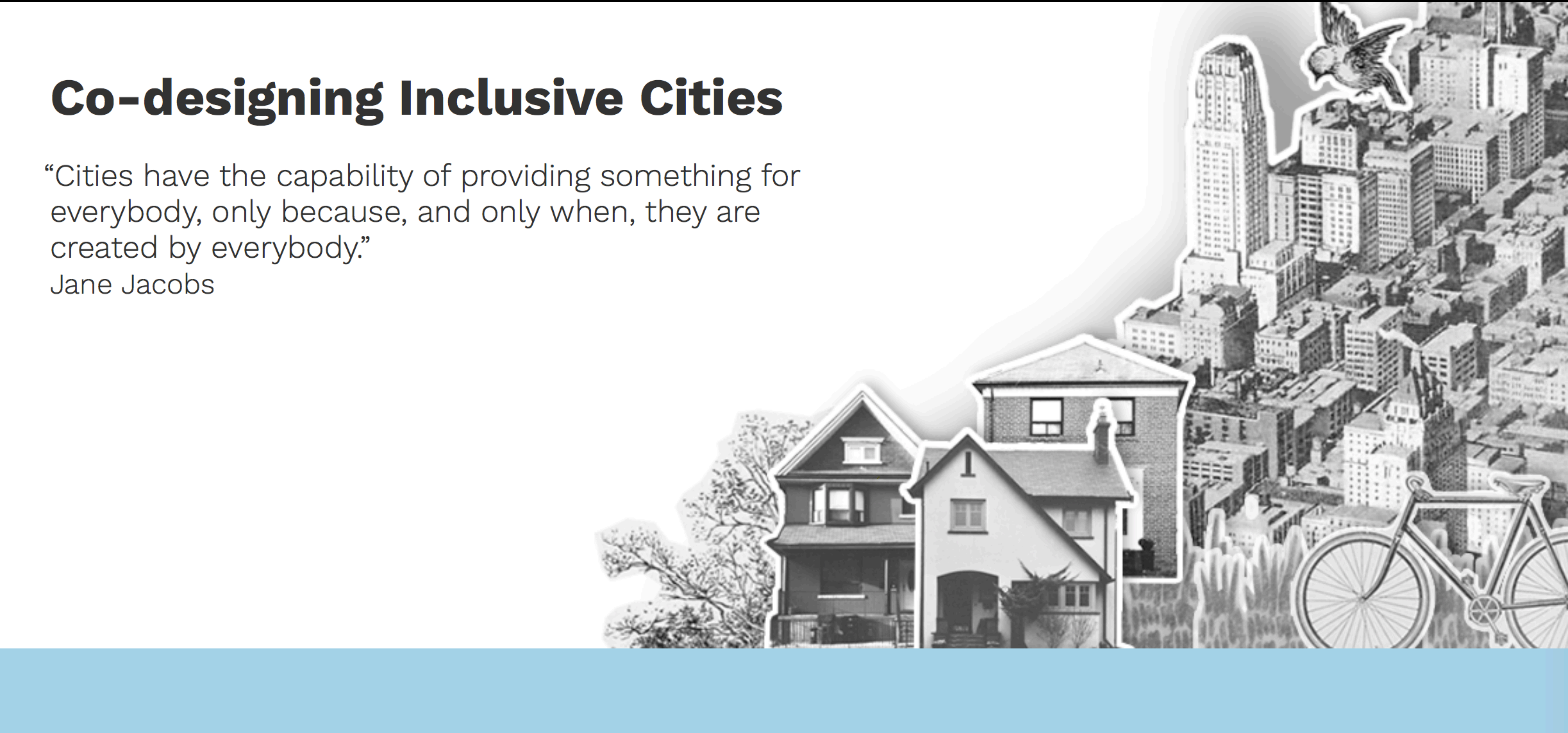

8
Citizen Control
7
Delegated Power
6
Partnership
5
Placation
4
Consultation
3
Informing
2
Therapy
1
Manipulation
Citizen Power
Tokenism
Nonparticipation

"Citizen participation is… citizen power. It is the redistribution of power that enables the have-not citizens, presently excluded from the political and economic processes, to be deliberately included in the future. It is the strategy by which the have-nots join in determining how information is shared, goals and policies are set… resources are allocated, programs are operated, and benefits… are parceled out. In short, it is the means by which they can induce significant social reform which enables them to share in the benefits."

Arnstein, Sherry. (1969). "A Ladder of Citizen Participation." AIP,
Vol. 35, No. 4, July 1969, pp. 216-224.

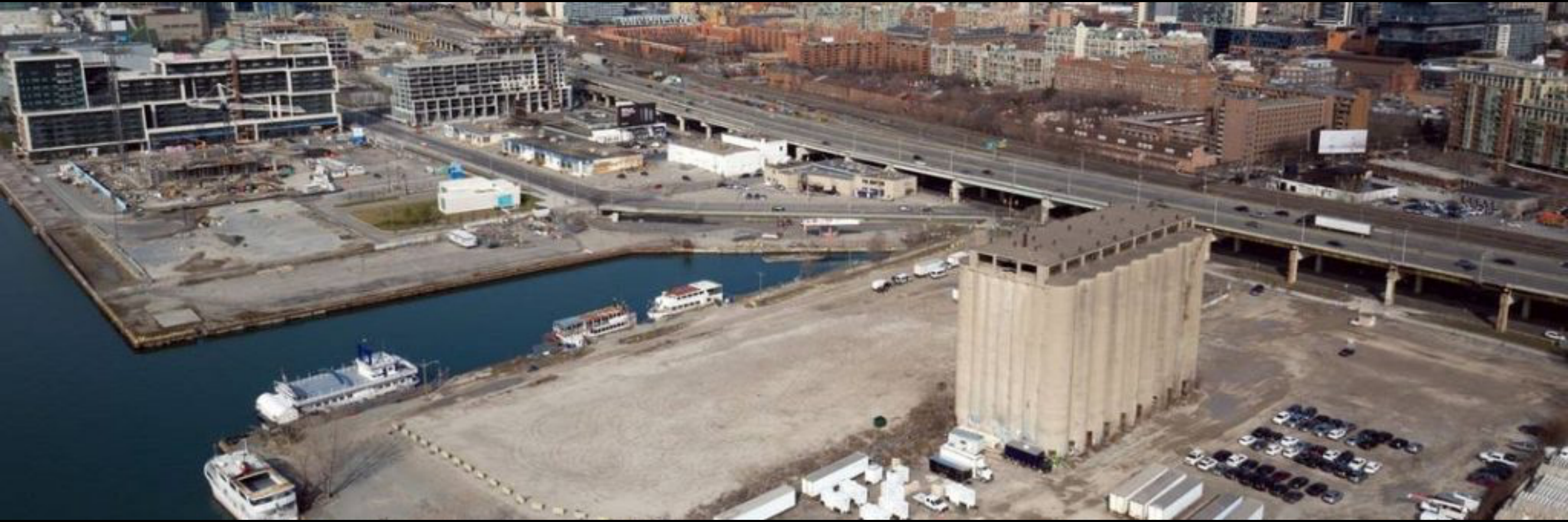

8
Citizen Control
7
Delegated Power
6
Partnership
5
Placation
4
Consultation
3
Informing
2
Therapy
1
Manipulation
Citizen Power
Tokenism
Nonparticipation

# cities.inclusivedesign.ca

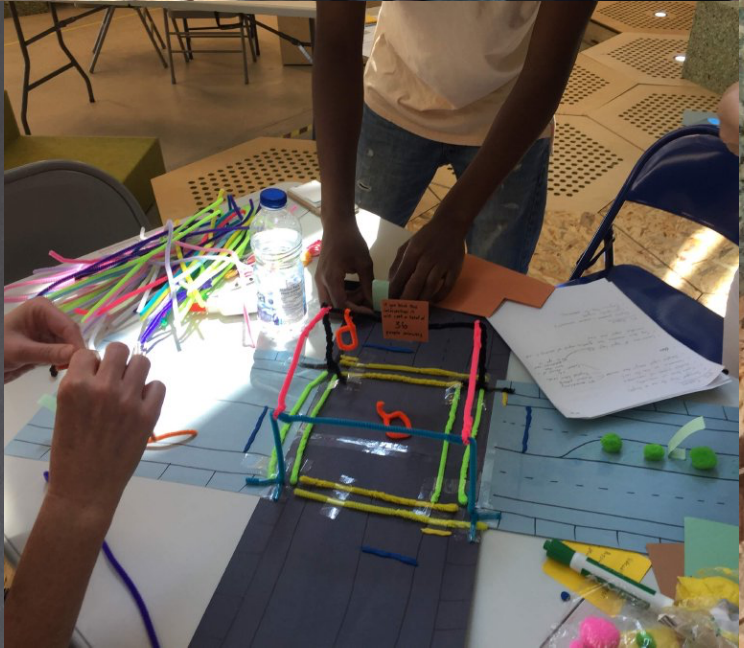


## Co-designing Inclusive Cities

“Cities have the capability of providing something for everybody, only because, and only when, they are created by everybody.”
Jane Jacobs



lights embedded (colour changing)
smooth surface
ridged area - change in

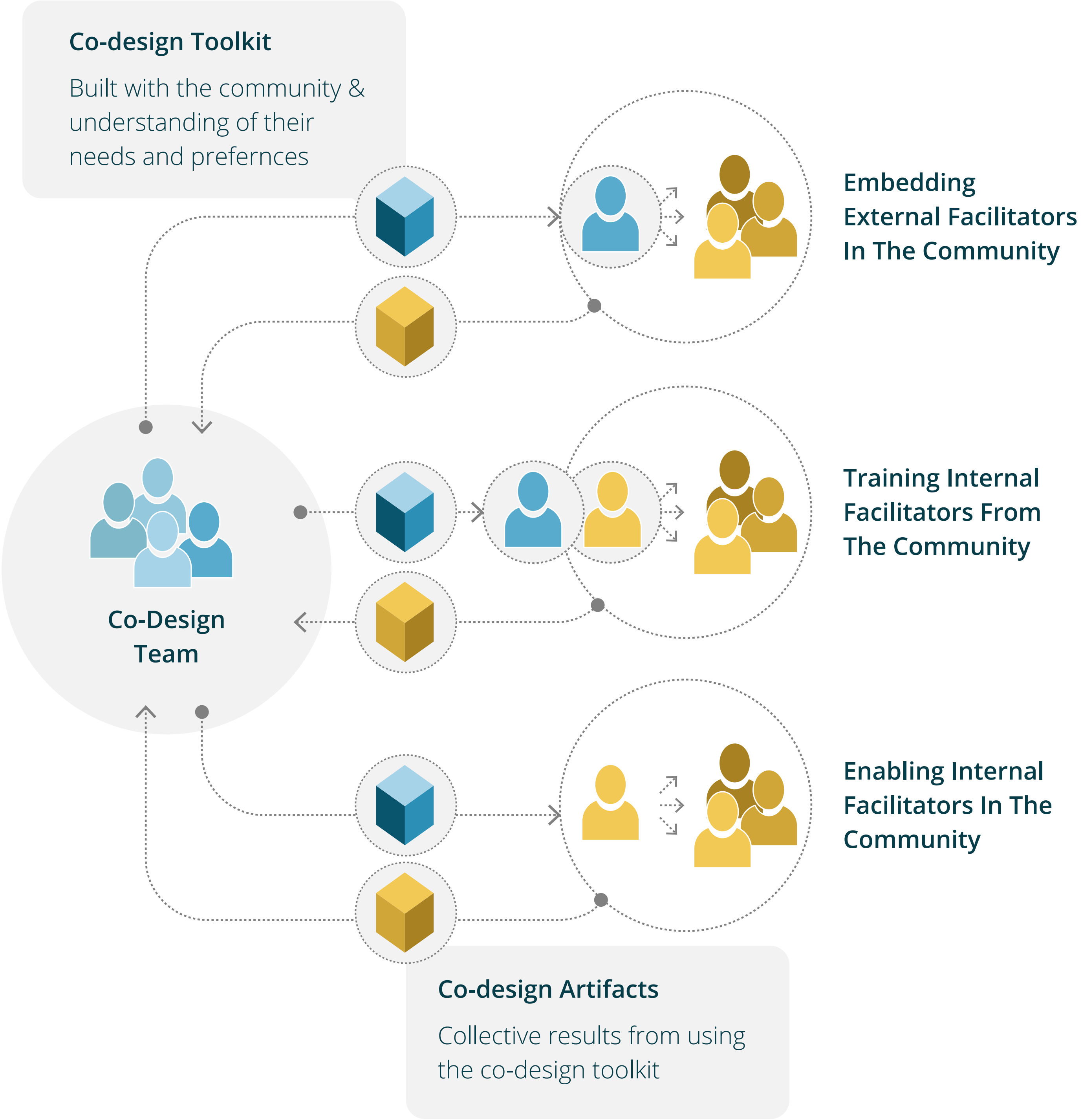


TTC info
ASK ME A QUESTION

# Modes of Co-Design

1. Workshops and synchronous events led by facilitators
2. Embedded co-design toolkits (led by community members themselves)
3. Open Studio methods and crits (open source designing)
4. Paired designer/user methods (working together on day-to-day designs)

*…Users designing it themselves*



**Co-design Toolkit**

Built with the community & understanding of their needs and prefernces

**Embedding External Facilitators In The Community**

**Training Internal Facilitators From The Community**

**Co-Design Team**

**Enabling Internal Facilitators In The Community**

**Co-design Artifacts**

Collective results from using the co-design toolkit

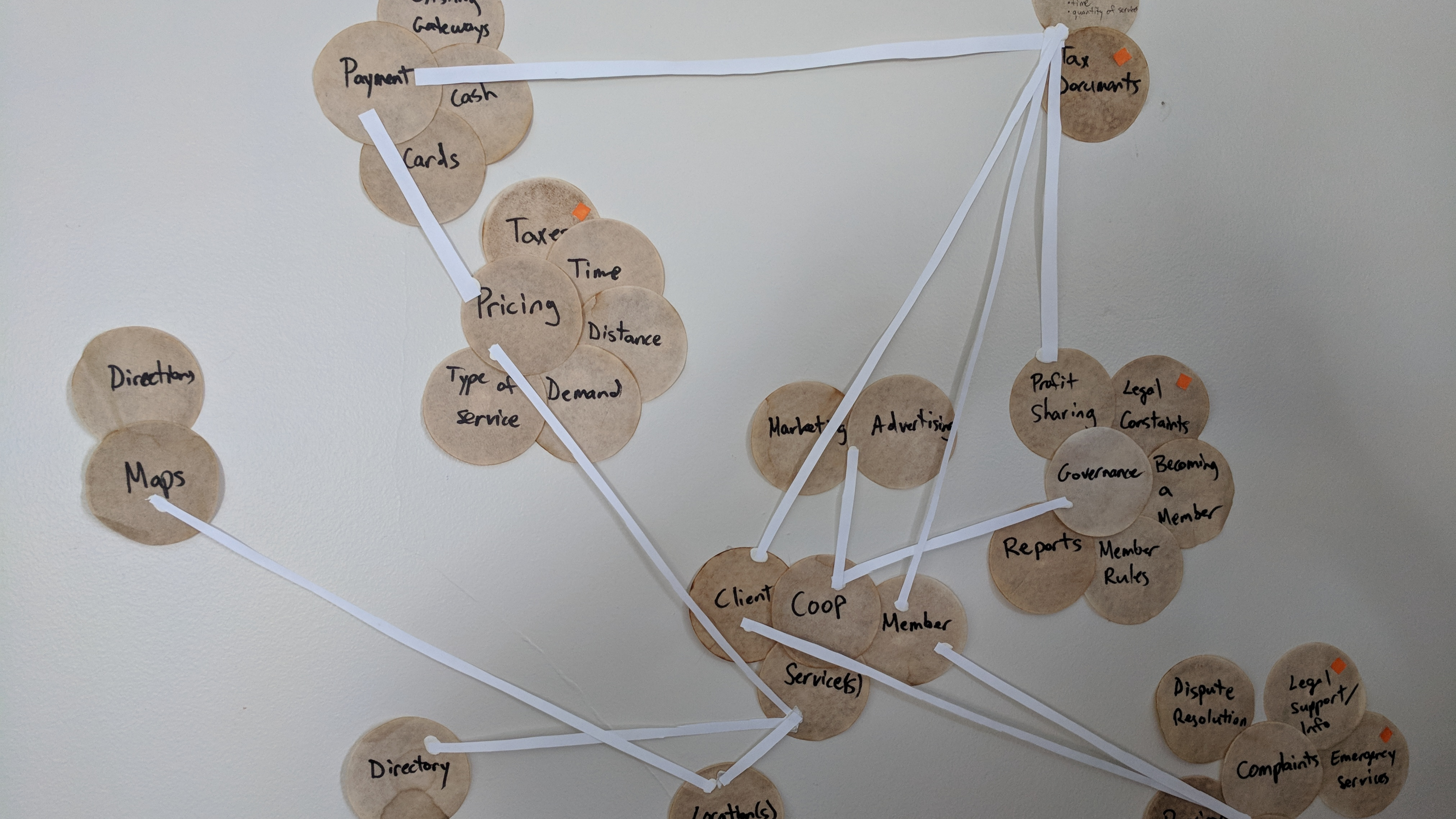
# PLATFORM CO-OPs

We connect cooperatives with the digital economy

# The Digital Economy

Imagine a digital economy that would follow the 7 co-operative principles

1. Voluntary and Open Membership
2. Democratic Member Control
3. Member Economic Participation
4. Autonomy and Independence
5. Education, Training, and Information
6. Cooperation among Cooperatives
7. Concern for Community


Existing Gateways
Payment
Cash
Cards
• time
• quantity of service
Tax Documents
Taxes
Time
Pricing
Distance
Type of service
Demand
Directions
Maps
Marketing
Advertising
Profit Sharing
Legal Constaints
Governance
Becoming a Member
Reports
Member Rules
Client
Coop
Member
Service(s)
Directory
Location(s)
Dispute Resolution
Legal support/Info
Complaints
Emergency Services

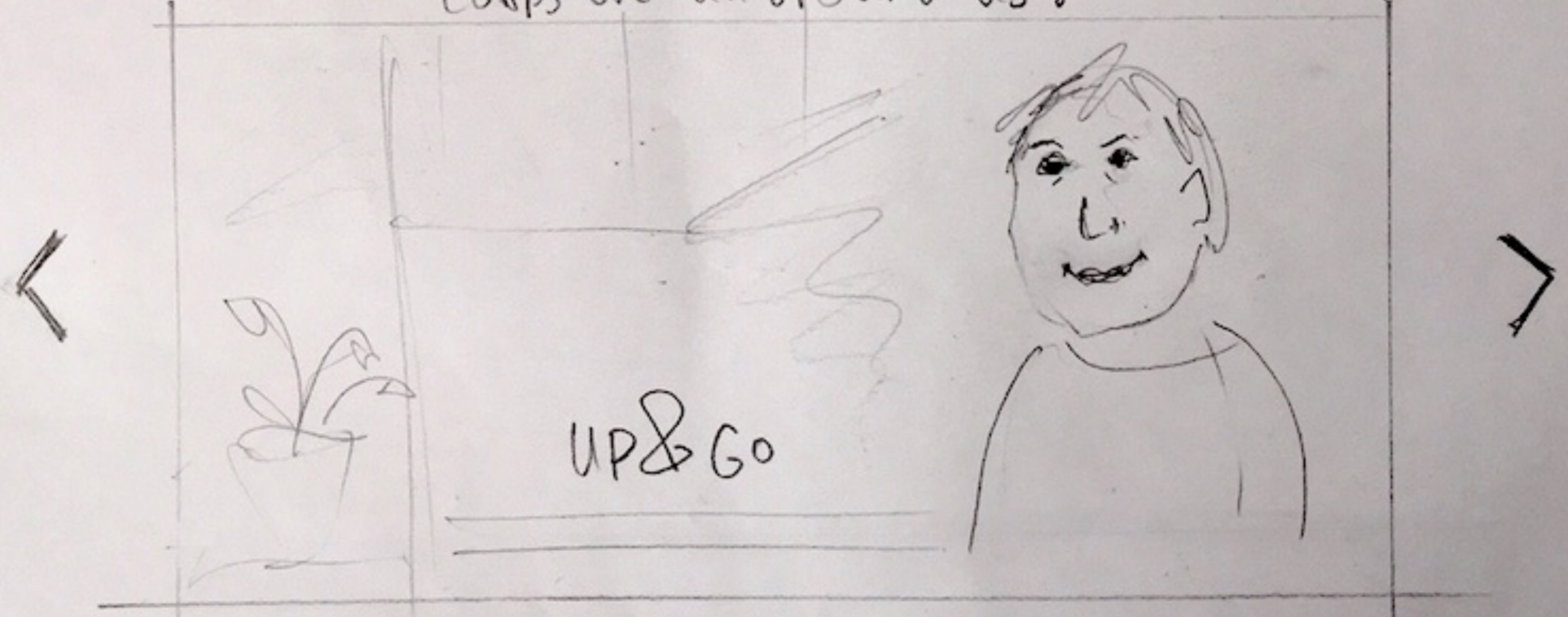
# THE PLATFORM CO-OP TOOLKIT

Combining the power of technology with worker-run co-ops to build a fair digital economy

Home    Map /Directory    Labor Platform    Tools    Resources    Contact

START A CO-OP

Coops are all around us!



LEARN MORE

# Continuing Design (after design)

# Material Case Study: Mads Dahlke

- Mads Dahlke is the host of the YouTube channel *Sail Life,* and (coincidentally) a software developer.
- He's doing something with his sailboat, I argue, that  simply can't be done with software.
- Dahlke has customized his boat substantively, including:
    - removing and replacing the boat's entire deck
    - designing custom-fit fuel tanks
    - completely reconfiguring the layout of a cabin to better suit his needs as a workspace
    - repaired and redesigned many flaws resulting from oversights or cut corners during the boat's original design and construction.
- Indeed the boat's original designers likely never conceived that their product would still be in active use today, nor did they design it with any intention of it being modified in the ways that Dahlke has accomplished
- Yet Dahlke is not a professional boat builder or expert restorer. He has pursued his project by acquiring some generalized technical skills and commodity tools, while participating in a larger community of other sailors and do-it-yourselfers who have worked on similar projects and shared their own learnings.

ARCTURUS
FOR SALE
HENRI LLOYD

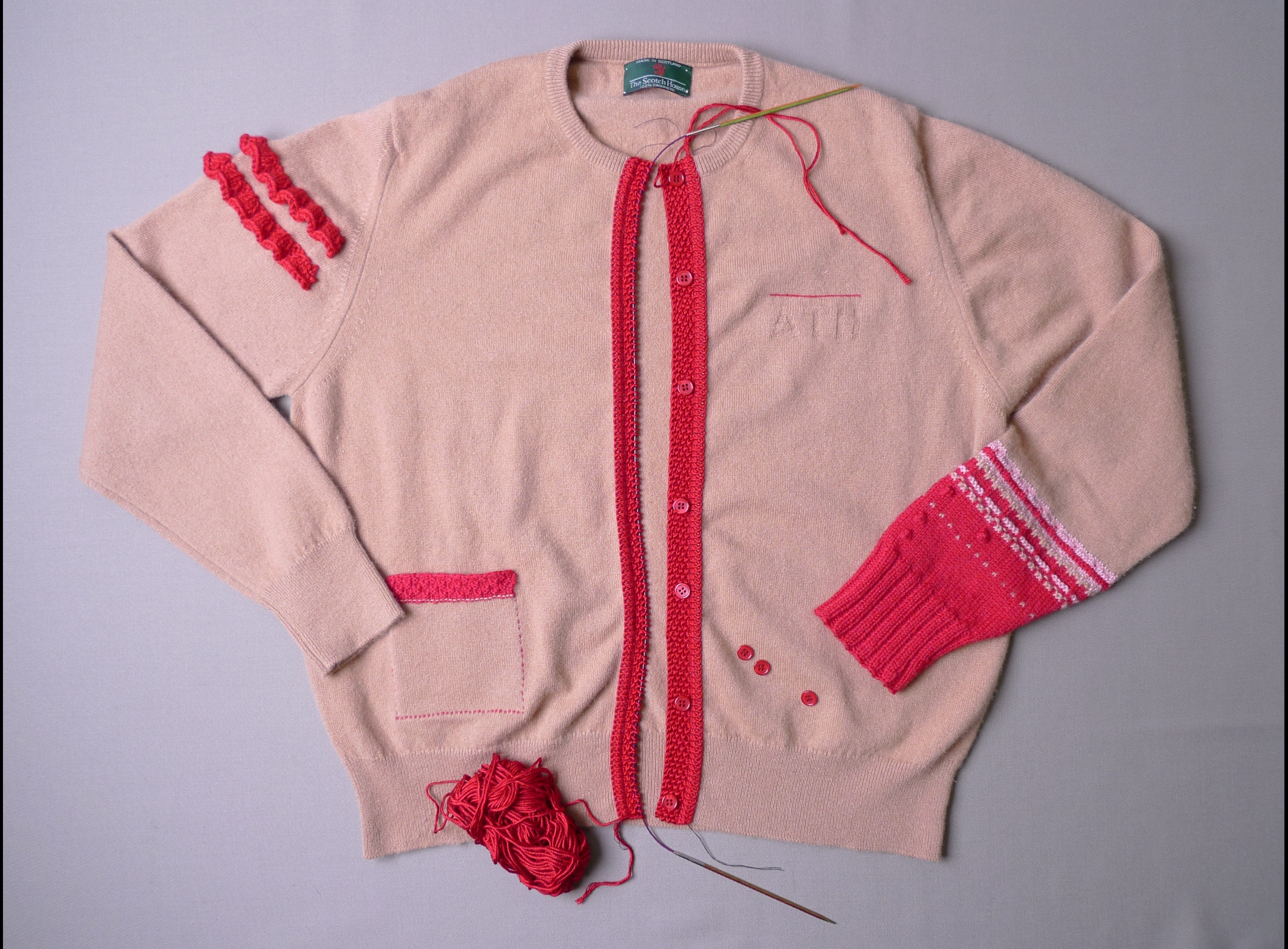
McKINLEY

How can we give an individual the power to (re)make this decision?

How can we support the serendipitous, unexpected, and informal?

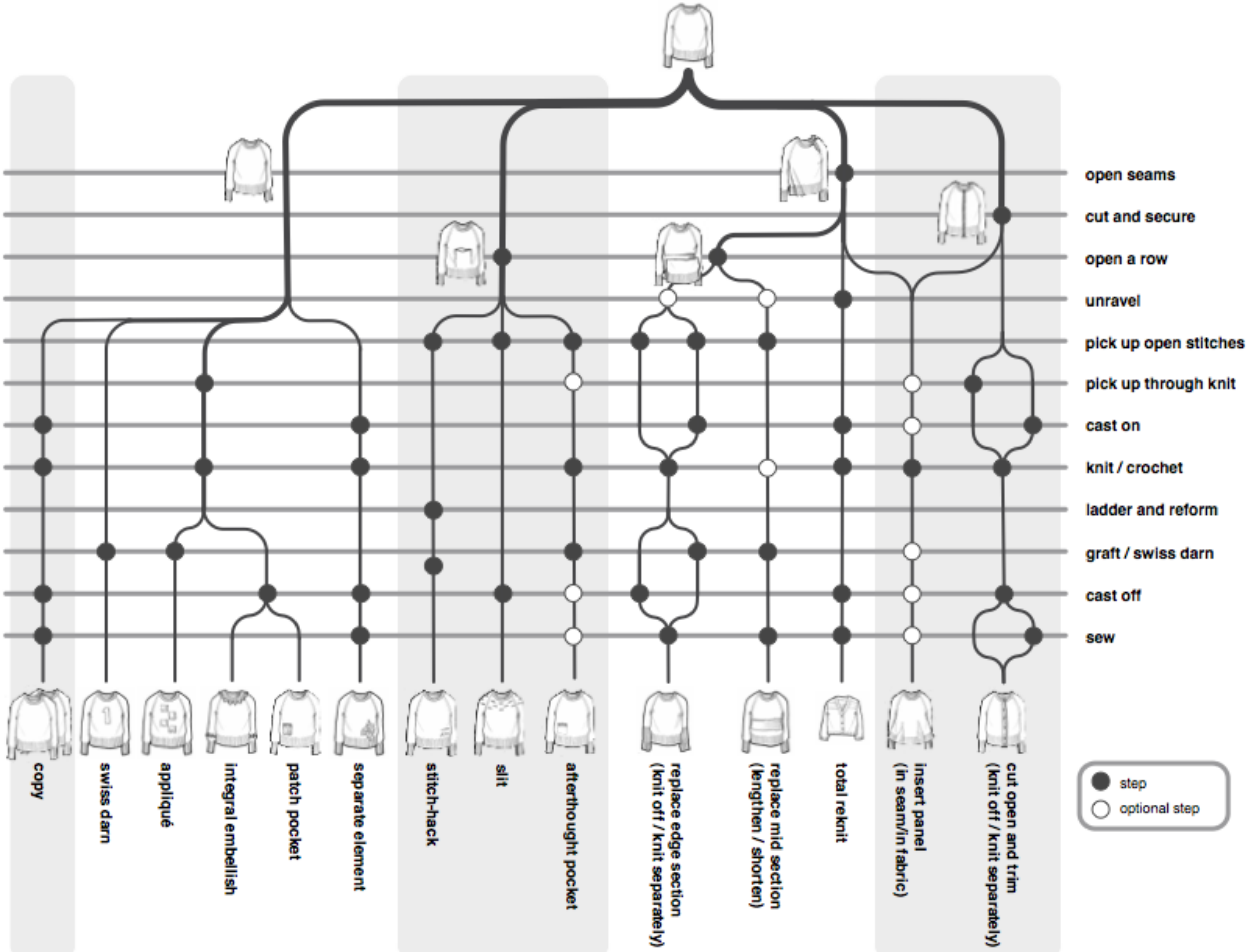
What communities might arise around this design choice?

# User Interface Options Demo


The Scotch House
ATH

# Material Case Study: Amy Twigger-Holroyd

- Amy’s *reknitting* methods provide knitters with a way to modify, add to, or subtract from already-completed knitted garments (including industrially produced, machine-made)
  - Unravelling, cutting, grafting, insertions, stitch hacks, and replacements
- Knitting, viewed materially, has a remarkably dual quality—you can take yarn and knit a sweater, and you can “frog it,” taking it back to its elements (yarn), and knit something new with its materials
- Re-knitting is not simply an idiosyncratic personal practice of Twigger Holroyd or her colleagues, but “an integral part of the practice of knitting” generally, a characteristic of the medium and the traditional methods of knitting itself
- A speculative future: what if software was like knitting?



open seams
cut and secure
open a row
unravel
pick up open stitches
pick up through knit
cast on
knit / crochet
ladder and reform
graft / swiss darn
cast off
sew
copy
swiss darn
appliqué
integral embellish
patch pocket
separate element
stitch-hack
slit
afterthought pocket
replace edge section (knit off / knit separately)
replace mid section (lengthen / shorten)
total reknit
insert panel (in seam/in fabric)
cut open and trim (knit off / knit separately)
● step
○ optional step

# Material Software

- A knitted object (software product) is a *vector*—a medium of production, communication, and becoming
- A reknitted garment (application) represents not only the content of the vector of knitting (software),  but itself a new vector, or form, for the creation of another work from and within it. This newness is not just the result of a process of copying, quotation, or appropriation, but of the possibility of the artefact itself—its ability to engender new forms and futures, "the immaterial virtuality of the material"
- Knitted objects (software programs) retain their modifiability and, as a result, have the ability to support a "community of practice" within themselves. As artefacts, they can be worked on by multiple creators and can support unanticipated uses and after-the-fact adaptation. It is this ability to be serendipitously added to, subtracted from, grafted onto, or unravelled in a form not already planned for and designed into the object that defines my concept of *materiality*, the latent and unrealized potential of software.

# Material-ing Tools

- This means, I think, that the ontological question regarding what constitutes a material vs. a tool can be set aside
- Instead, imagine an artefact that, for one, is a completed, singular tool—yet can be, for another, the raw material for generating something new or different
- The difference, then is largely contextual and perspectival, rooted in communities of practice

# Characteristics of Material Software

1. **Representation of program code as data**—material for reflection and modification by other programs or by authors who come along later in the process. An ecosystem of authoring and customization tools.
2. **Availability of landmarks**—names or other stable means of reference (e.g. via CSS selectors)—for behavioural and compositional points within program code, such that another author, working outside the original program's code, can identify these points and issue their own alternative logics.
3. **Complete externalization of program state**, also as data, in a form that it can be understood, manipulated, and modified live by programs located outside of the runtime of the software, so that software can be redesigned piecemeal and from a distance.

# Limits of Material Metaphors

- Software, we might imagine, is a “material of the mind,” yet one which is expressed in a uniquely computational form—infinitely reproducible, demanding of precision and detail
- Ideas, in system, are complex and even sometimes inconceivable outside of the context in which they were originally situated and elaborated.
- Software is not yet, and perhaps never can be, a craft. Its material qualities may be far too different, ultimately more subject to change and systematic contingencies than artefacts in the physical world—paradoxically more immaterial
- Perhaps music, sound, and poetry provide us with other hints about how materiality functions immaterially?

# In Summary

- We need to come to terms with change as a catalyzing force in our projects—not just something be managed or controlled or minimized
- Our current collaborative methods—agile (especially early forms), open source software, participatory design—are necessary but insufficient to create equitable social systems
- Our discipline(s) *need* change, and need more active agency for change within software systems—by bringing "users" into the process as serious co-creators, and by giving them new powers to continue the design process
- Material metaphors, such as craft, provide us with a crucially-needed imaginary, different from dominant mathematical and techno-scientific tropes—but may not take us far enough into a new field

It's been a long, a long time coming
But I know a change is gonna come,
oh yes it will

Sam Cooke
A Change is Gonna Come, *Ain't That Good News*, 1964