Don't build it.

A GUIDE FOR PRACTITIONERS IN CIVIC TECH / TECH FOR DEVELOPMENT

Luke Jordan
Grassroot | MIT GOV/LAB

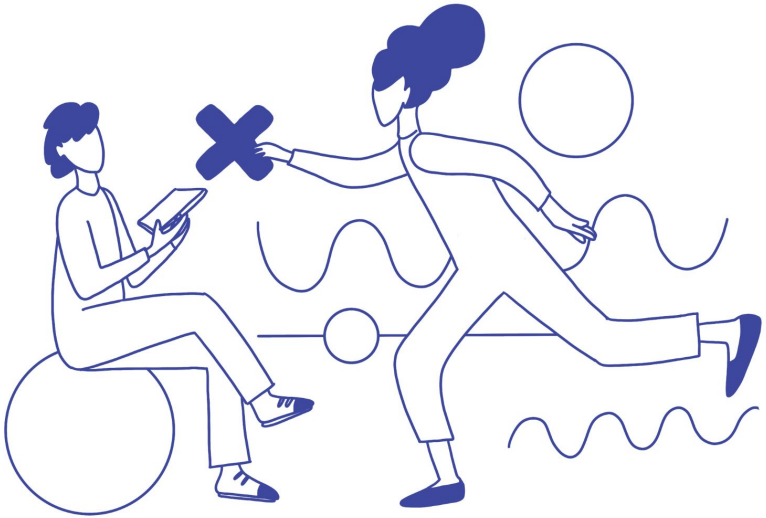
GRASSROOT

MITGOV/LAB

Quick Background

- Studied Math & CS, then worked at McKinsey and the World Bank, then picked up code again
- Founded Grassroot, a civic tech organization in South Africa
- Reached 2.5m users, over 40k activities called, used in national Presidential debates and major Covid campaigns
- Grassroot built *with* our users, from community organizers to major national campaigns
- Then founded Jupiter, a fintech for mass market saving (shut down by regulators - code now open source)
- Now a Practitioner-in-Residence at MIT GOV/LAB, allowing me to distill lessons learned over last 5 years

Why not build it?



- Are people already trying to do what the technology is supposed to help them do?
- If yes, how are they doing it now, and are you sure you know why that does not work? And why will technology make any difference to the reason their existing attempts are frustrated?
- If not, why would having technology make a difference? Why would someone who did not want to do X now want to do X just because some tech exists to do it?

Some examples

Kinds of projects

- An app to help people participate in public planning meetings
- An app to help people report local medical stock-outs or service failures
- A data dashboard to help officials gather real-time data on education outcomes

Questions to be asked first...

- If people thought participating would make a difference, or would be well managed, would a higher proportion not make a plan to attend?
- If a hospital system has such low accountability it won't install now-free inventory management software, why would it care what the app says?
- If the data was going to be used, wouldn't someone have created an Excel version already?

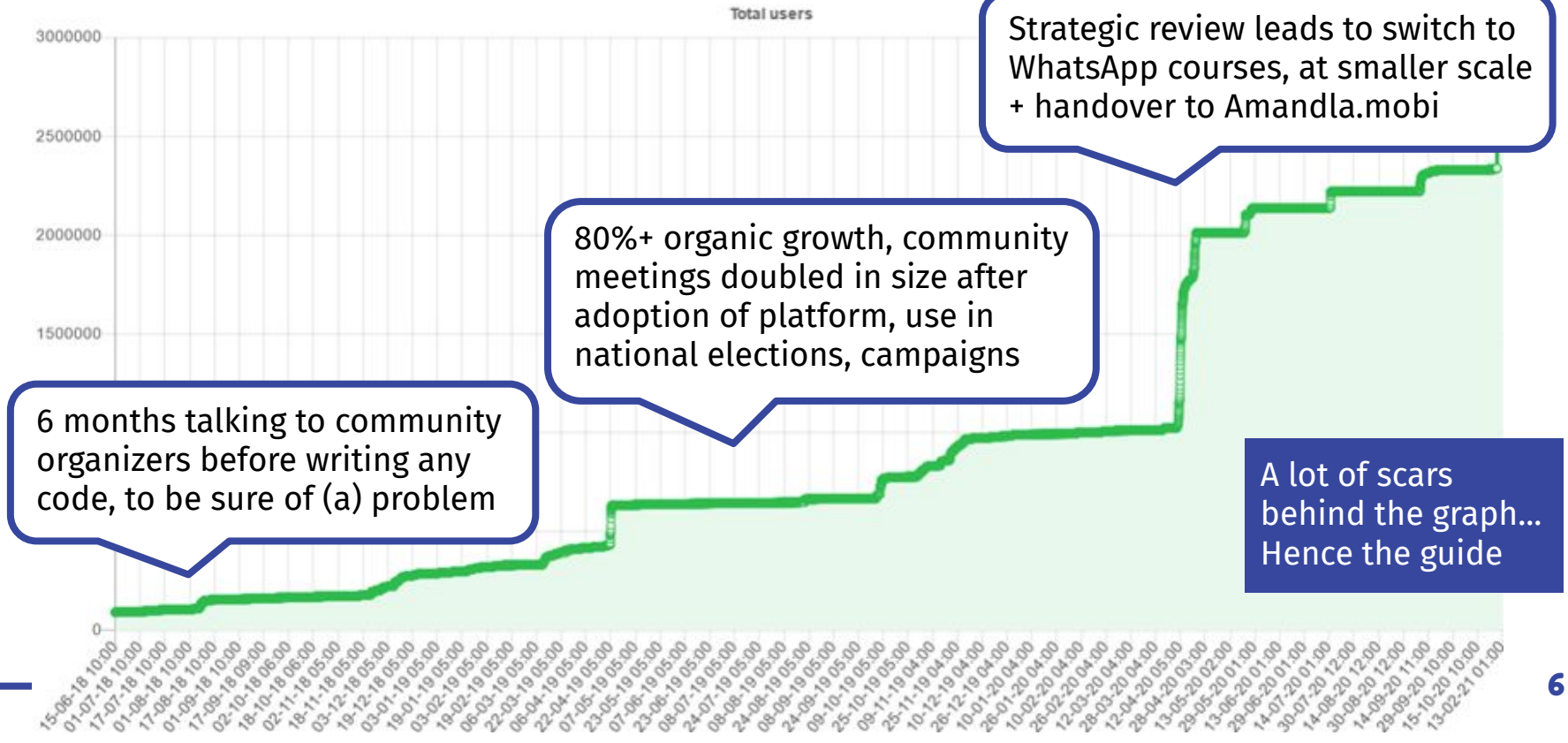
But sometimes all the questions *are* ticked, and it is time to build -- just remember that these days, no one will tell you it's a bad idea, so make sure you've asked the questions

Grassroot's origin

1. It's too costly and difficult for the poor to organize themselves
2. It's too hard for local leaders to access information they can act on and connect to each other in a meaningful way

Formal democratic systems are unaccountable because barriers to the poor in South Africa organizing themselves and participating are too high

Grassroot's story



If you have to build it...

Outsourcing is great as a tactic, but a terrible strategy.

Add full-time talent cautiously, at cost levels where you can keep them in the team and invest in their growing skills over time.

Get close to your users and to do so fast with a dedicated community engagement team.

Adaptability and speed of learning are core criteria in every role.

Set a budget that gets you off to a quick start, but allows you to keep iterating over time.

Why not just outsource?

Makes the entire project's success dependent on the single decision point of what contractor you hire

Without the ability to judge technical merits or break down a project into its components, you are almost guaranteed to under- or over-specify

A whole cottage industry of consultants hunts for bid docs written by non-technical teams, and they're not exactly who you want bidding, or building

Is the friendly developer who helped with the specs really going to hustle their network for good bidders, and then also evaluate all the bids, and then also review milestones and requests for scope changes?

If the project is not important enough to justify hiring a CTO (who can do lots of other things beside the project), should you be building it?

Hiring a CTO

Is there an example of a project or a major feature build that you successfully killed?

What's the best team member you've ever had, and how would you find more of them?

What's the worst team member you've had, and how would you avoid them?

Have you ever managed to mentor someone from being on the "worst" track to being on the "best" track? Have you come close? What did you do?

How do you decide whether a technology belongs in the stack for a project?

How do you react when a project is (very) late? Or when no one is using it?

Hiring a junior engineer

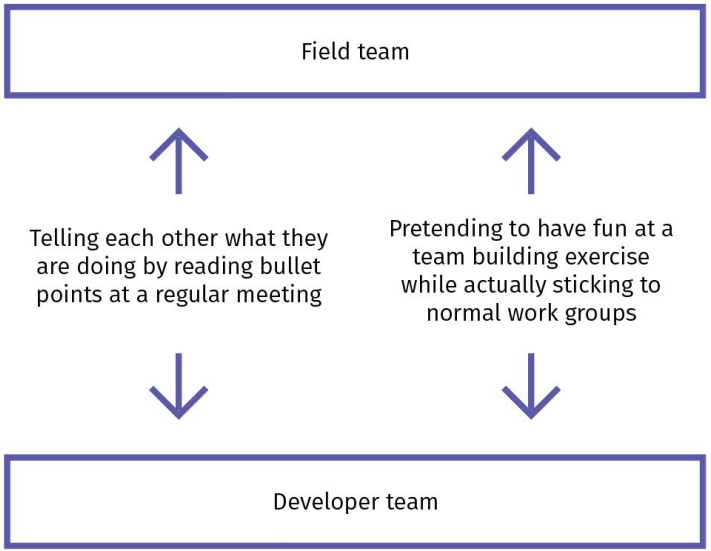
Hunger to learn. Speed of learning >> languages someone has learned. Ask about side projects, or about a team decision made on a prior project, and why it was made

Commitment to quality. Coding is about trade-offs, so selecting between the necessary shortcut and the damaging shortcut makes a big difference, and requires internal discipline. Ask about managing trade-offs, e.g., what-ifs about imagined (but concrete and plausible) scenarios.

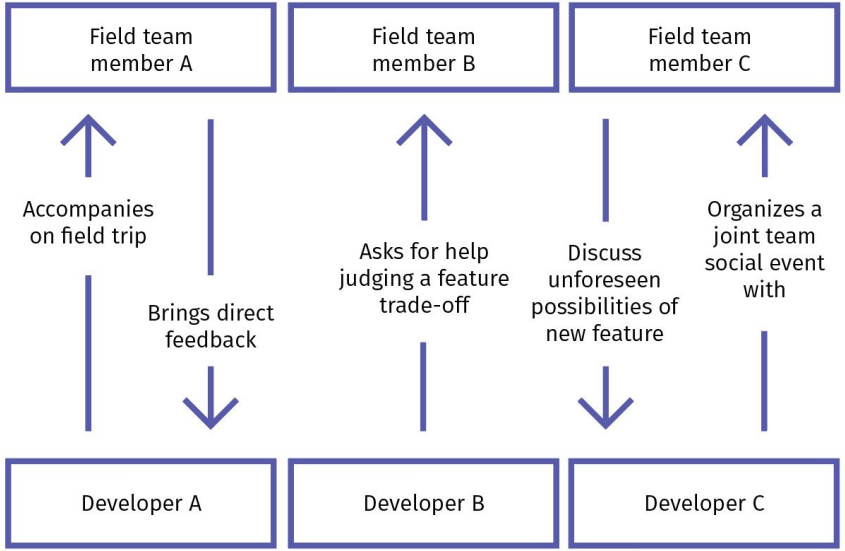
Basic technical ability. Even with experience, it is sometimes remarkable how many interviewees will not be able to code. So, some simple coding and technical questions are handy.

Tech-field connections

Artificial

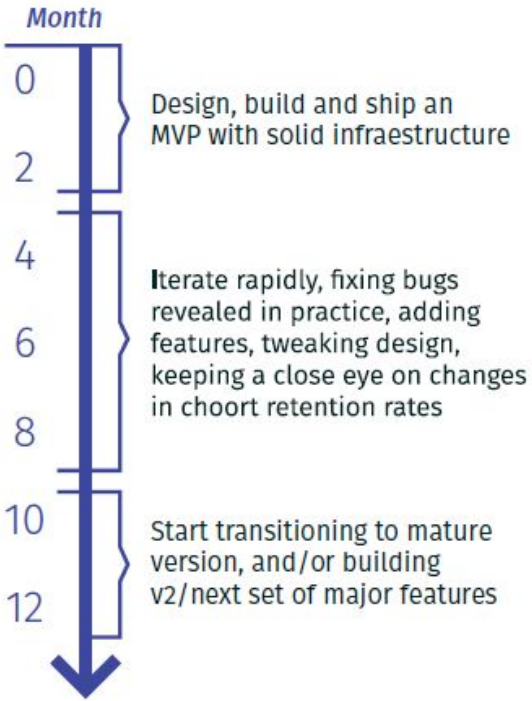


Organic

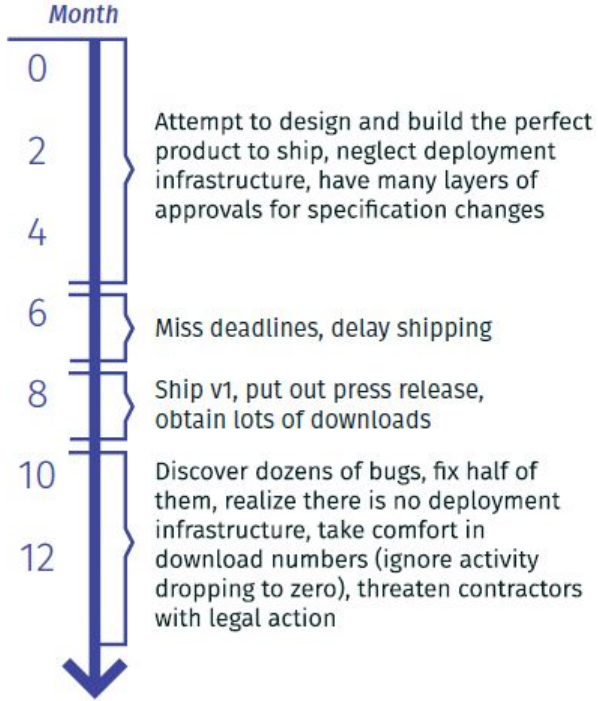


Timelines, good and bad

Good timeline



Not great timeline (planned or actual)



- Set a budget that gets you off to a quick start, but allows you to keep iterating over time
- When it's going wrong--turn the corner, then add gas, otherwise you just hit the wall faster

Grassroot today

Structural factors in South Africa

- Unemployment
- Institutions
- Knowledge

Theory, “democracy will work if people collectively engage” and “people will engage if they can organize more effectively” both failed--organizing spiked, hit a wall of non-accountability, and fissured

WhatsApp courses maintained participation and showed early signs of effectiveness, but funders unenthused (wanted more … fad + polish)

Core platform used by Amandla.mobi to run national campaigns--secured grants to low-income mothers by 500k+ credible signatures in a week during Covid lockdown--now primary channel of impact

If another problem that justifies building occurs, will build again--else, will not (currently exploring)

In all:

Can you avoid building it?

if you have to build it:

- hire a CTO
- ship early
- mature long

Don't build it

and no matter what:

- draw on a trusted crew
- build lean and fast
- get close to and build with your users as soon as possible

beyond that:

as the Bhagavad Gita says,

"you have a right to the work, not to the fruit"

All this and more...

in the guide

Plus:

- Vanity metrics versus learning
- Technology choices
- Rules of thumb on budgets

... and much more

Get it here:

https://mitgovlab.org/resources/dont-build-it-a-guide-for-practitioners-in-civic-tech/