Twitter Thread by GeePaw Hill



GeePaw Hill
@GeePawHill



Human, local, oriented, taken, and iterative, these are the change-harvester's bywords. In iterative change, we not only accept the reality of gradual stepwise refinement -- changing what we've already changed before -- we actually anticipate it and take advantage of it.

Before we begin: Folks, I know many are celebrating last week's results . I see them as one small pass towards the change I seek. Take a short break, then it's back to work?

Black lives matter.
Voting rights matter.
Women matter.

Stay safe, stay strong, stay angry, stay kind.

(And a *second* prefatory note: I've been not writing for six weeks. It was a combination of several factors, including illness, distraction, and the need to finish a gig. I'm happy to be back at content, now, but also very nervous. Bear with me while I regain my sea legs.)

With iteration, I think a great starting point for the concept is to watch just about any youtube video where a skilled artist draws a realistic rendering. What you will see almost inevitably is a direct implementation of iterative change.

The strokes begin quite broadly, faintly indicating the broad contours of the subject. Experience is relevant here: those who are most familiar with drawing that particular kind of subject, will make contours that seem "right" more quickly than those less experienced.

Once those contours are laid out, and possibly adjusted, the artist will begin moving around the drawing, from one spot to the next, each spot still not done in detail, just gradually refined. At some point, what is drawn starts to seriously resemble what is intended to be drawn.

The areas of artistic focus get smaller and smaller, and the details get more firmly elaborated. Some of the earlier decisions are long-gone now, others confirmed and strengthened.

The artist's loving attention to detail emerges here, and the work normally finishes with a flurry of concentrated activity, little micro-focused areas, elaborated and committed. We end with an astonishing drawing.

To understand iteration here, notice just one thing: if we lay a grid over the drawing, every single cell in that grid will have been touched, then touched again, then touched still again. The artist has iterated the changes made to the page.

Take one of those cells, and see it as a list of successive changes. One of those changes will be the last one made to that cell. But "finality" of a cell is only ever detectable in the process when that entire process is itself over.

Finality is only determined ex post facto -- after the fact of the drawing -- never in advance, and never during the process itself.

And the artist? Well. The artist knows this, & knowing it, leans into it, not just accepting it but actively structuring the workflow around it.

Let's bridge this extended metaphor into our topic of geekery. As before, we'll take a negative case and a positive case.

The negative case: 95% of the "agile transformations" I have been witness to. We attempt to install a highly detailed new process over whatever it is we are doing now. We simultaneously change every cell in the drawing to what it will be in its final state.

Some folks realize that there are a lot of cells in their process drawing. So they set aside doing all the cells at once. Instead, they take just one cell, and get it to maximum detail, and then they go to the next one.

This is just about exactly wrong.

What it's doing, btw, is confusing iteration with increment. Terminology here is muddy, tho, and I'm going to drop that seed for you to look up and think about later, and instead talk about why the cell-by-cell-perfection approach doesn't work very well.

1. Because individual cells collaborate with each other, and if they're not speaking the same language along the same channels, the perfect cell and the imperfect cell will not be able to collaborate.

2. Because what's needed to go from imperfect to perfect in a cell isn't a switch that we can throw, it's a process of evolving practice, a *learning*, with levels of development, where what is real at level C can't even be *glimpsed* until we're reasonably good at level B.

3. Because the level of detail required for perfection simply cannot be captured in any verbal description, but must be lived. Systems don't do what they say they do is a basic tenet of systems theory, and systems for making software are no exception.

4. Because the most powerful part of a mixed system involving humans, machines, and procedures, is unquestionably the humans, and human minds and behaviors can't be installed in the abstract, but must be developed in the real.

These four factors, and some others I'm probably missing, are a huge part of why "agile transformation" has become such a dirty word. They violate the change-harvester's byword "iterative".

And before we get to a positive case, let me point out I chose "process change" as my example, but I didn't have to. I see organizations do the same thing with "code change". Layered horizontal development is nearly always cell-by-cell-perfection strategy, and also doesn't work.

And I see them do the same thing, again, with "feature change": we must perfect the million-user support before we even have ten real users. We must have automatic password recovery before there's even a page to go to after they've logged in.

The positive case: let me tell you about over two years of playing Oxygen Not Included, from it's third iteration in early access to its current one, and how it stands as a terrific model of iterative user value, as well as a hella fun game.

Those of you who follow me know how much I've played it, and enjoyed it. What they might not know is how dramatically different the current game is from the one I started playing two years ago.

They started with a kind of broad brush-stroke, an AoE-style "give orders & wait" game, a graphical Hammurabi, really, tho forty years on from there, of course, far more intricate than the old text game.

Now, it is one of the most complex and sophisticated "world-builder" games there is, with a huge fanbase, tons and tons of livestream experts, tutorial videos, and even schmoes like me posting play-throughs. :)

Why did it work? The short version is that it added user value in a stable well-defined series of iterations. Most of them were released six weeks apart, though there were a couple of delays here and there.

Some of the iterations felt mostly "additive" or incremental. They actually just built on to what was already there. But some of them were huge systemic transformations. And nearly *all* of the iterations were true iterations: They changed things they had already changed before.

1. By holding fairly close to an iteration schedule, they built in their community and expectation of regularity and stability, even as they continuously changed the game.

2. By fielding their game in early access, they did two amazing things: 1) they made money that helped them keep going, 2) they got massive input from their customers.

3. By being willing to accept changing things they'd already done, they got big love from their customer base. They changed the priority system -- how we tell who what to do in what order -- no fewer than four times, and it's completely different from what we started with.

And again, what I've just described as iterative user-value, has its isomorphism in iterative process-change, and again in iterative code-change. It's all the same stuff, which insight is the beginning of the change-harvester's viewpoint.

Human, local, oriented, taken, and iterative. When we iterate we not only accept that we will be changing the same part of our system more than once, we *embrace* it. And that brings us back to the fundamental slogan of the Extreme Programming movement:

Embrace change.

Thanks for reading. It's been so long since I've written a muse, it was actually quite terrifying. I'm glad I dove in again, for better or worse.

Thanks to all for the support you've already given, & since we're already massively long, let me ask for 3 more forms of support.

Support by subscribing to my content. It's free, it's spam-free, and it comes in full-text or in audio, right to your box. It really helps me when you do this.

https://t.co/0iffwG5jrd

Support -- and get involved -- by joining the community of change-harvesters. It's called a "camerata", a kind of continuous colloquium on all matters of change, centered in but not restricted to the subject of geekery.

https://t.co/f5WhrulTsi

Support the people who are embracing change not just in geekery but out in the world.

Right now? If you don't know what to do, start with https://t.co/I1yRxTOHEc

We started a change last week. Let's keep iterating on it.