Twitter Thread by Patrick McKenzie

Patrick McKenzie
@patio11

A thread on HN about bad code in legacy projects both makes me think how little we've learned as a discipline over the years and, honestly, how little credit we give ourselves for some pretty major

Fun going down this list and thinking: "Hmm, plausible at a well-run modern software shop", "Hmm, possible, but requires implausible tradeoffs", "Literally disallowed by languages", and "If you were to attempt doing that our test suite wouldn't let you merge."

I think we as an industry celebrate (not quite the right word) failure too much and don't celebrate success nearly enough. There is no DailyWTF for competent execution, word of which generally stays pretty local to the source while incompetence passes into legend.

Alrighty let me try to thread the needle on being the change I want to see in the world while not giving away anything that will get me in trouble:

Ruby has wonderful developer ergonomics. Typed languages are easier for machines to guarantee the correctness of. We built a type checker for Ruby (and I believe it is slated for OSS release sometime).

c.f. https://t.co/S5XIDxFUrH

We have an infrastructure at work which allows one to specify an invariant about not just code but e.g. objects or the environment and then have a range of response options if that invariant changes.

(Parallel evolution of code: I wrote a less-well-specified one at last gig.)

Git, continuous integration, and workflow-driven mandatory code reviews are all younger that the Joel Test, at least insofar as them being common features of median-sophistication engineering shops.

It is not astonishing to start a new engineering job in 2018 and have a developer environment which reasonably approximates the production environment available on one's laptop or tested, repeatable ways to spin up and spin down a new server w/o "build it by hand."

It is highly likely that a service which is hard down learns of that fact faster than Twitter can apprise them of it, assuming that service is operated in a professional fashion.

At risk of stating the obvious: this is a relatively novel development.

The industry has decisively adopted:

* a single, common encoding for almost all human languages
* a single, parseable, human-readable data interchange format
* a default protocol for information transport

You can round to "Any new application talking to any application written by a competent team in last 10 years will be talking to it over an encrypted link which neither side had to think deeply about because the technology is reliable, ubiquitous, and uncontroversially legal."

While it's not literally the case that you could replicate an entire modern software company's deployment for zero dollars in software licenses, that can almost round to true, due to the pervasive use of OSS.

This is very good for learners.

You can get a full development environment capable of doing Hello World spun up in your well-supported language of choice in, almost certainly, less than ten minutes of effort (contingent on you using a Mac, sadly).

The majority case for libraries, APIs, and file formats of interest to you will overwhelmingly be "If you Google the thing you want you get exactly what you need very, very quickly."