July 18, 2023

More Software Projects need Defenses of Design

Explain your choices to your users!

I was gonna write an rant about the potential probabilistic model checking, then realized I needed to look at projects besides PRISM and STORM. Then I checked out simpy and saw it had a Defense of Design (DoD):

> This document explains why SimPy is designed the way it is and how its design evolved over time.

This is great! More projects need this!

All languages and projects do "odd" things that make no sense to outsiders. And oftentimes there's a good reason, like

- It's needed for more advanced use cases. In TLA+, you have to specify every variable that's *unchanged* in the step, which adds a lot of boilerplate but makes things like refinement and spec composition less terrible.

- It's backwards compatible with something else. C originally had 0-indexed months to save some CPU cycles, and JavaScript's backwards compatible with that.

- It's a consequence of the language design. Python has mutable default arguments, which happens because "defining a function" actually creates a function object in the local namespace, and argument defaults are stored as object state.
- It's for an important, but very niche, use case. C and C++ used to have trigraphs, so you could write `??<` instead of `{`. This made them compatible with EBCDIC encoding.
- It was inspired by someone else's design, like SimPy and Simula.
- It's a workaround for some older mistake, or something that is obsolete now.
- It's more performant in a way I wouldn't think about, like using CPU caches better.

These are all design choices that didn't make sense when I first saw them, and in a lot of cases I thought "why the eff is this language so stupid." If I had a defense of design, I could have read it and understood that the designers *weren't* stupid. Maybe they're solving a problem I don't have yet or will never have, maybe they were dealing with strange constraints, maybe it looked like a good idea at the time, etc.

Here's two more potential benefits I see of DoDs:

It shows good stewardship

Stewardship is the rough notion of if a project is "in good hands", and that you can trust it to be reliable long-term. Things like "having a changelog", "regularly fixing issues and merging PRs", and "having a high bus-factor" are all signs of good stewardship. "Developers get in fights with people", "docs are out-of-date", and "the tool is someone's masters' thesis" are signs of poor stewardship. I shy away from projects with poor stewardship, even if they ostensibly solve my problems.

I think a DoD is a sign of good stewardship. It shows that they think the design problems through carefully and are able (and *willing!*) to explain their decisions. This isn't the most important sign, and someone could be really good at that but terrible at maintaining a project. But it's a sign I don't often see.

Similarly, it makes them more *credible*: I'm more likely to trust their claims about their software. This is true even if I disagree with their reasoning or design decisions. Showing your work just means a lot to me.

It breaks Chesterton's Fence

> There exists in such a case a certain institution or law; let us say, for the sake of simplicity, a fence or gate erected across a road. The more modern type of reformer goes gaily up to it and says, 'I don't see the use of this; let us clear it away.' To which the more intelligent type of reformer will do well to answer: 'If you don't see the use of it, I certainly won't let you clear it away. Go away and think. Then, when you can come back and tell me that you do see the use of it, I may allow you to destroy it. — *G. K. Chesterton*

This is used in the context of replacing legacy code. Even if the code is a mess, it could still be solving problems you don't realize are there, so don't change it until you know *why* it's there.

The Defense of Design explains *exactly* why the code's there, so then you can safely replace it.

I really don't have much more to say about this, it's just a really cool idea and I'm surprised I haven't seen it before.

If you're reading this on the web, you can subscribe [here](). Updates are once a week. My main website is [here]().

My new book, Logic for Programmers, *is now in early access! Get it [here]().*