Stained Page News Subscribe

What Recipes Have in Common With Clean Code

A software engineer explores cookbooks

Stained Page News 49 min ago 3

Howdy cookbook fans!

It's not every day an article gives you an entirely new framework to think about and discuss recipes, but that's just what software engineer **Alaina Kafkes** has done in writing the piece below. When I review cookbooks, I often struggle to describe exactly *why* good recipes are good and bad recipes are bad—simply saying one works and one doesn't isn't quite enough—but by thinking about recipes through the lens of clean code practices, suddenly I have a new vocabulary for what makes a clear, easy-to-use recipe. And what, well, doesn't. I hope you glean as much as I did from this exploration of recipes and code (even if you're not quite tech-literate). Alaina, take it away!

What Recipes Have in Common With Clean Code

Photo by Compare Fibre on Unsplash

As a software engineer, I spend my days dreaming up steps for iOS apps. But writing code that Just Works™ is too low a bar to clear; it is imperative that my colleagues can work with my instructions. Recipes have steps, too, and if a recipe step can be defined as an isolated subset of instructions (e.g., "1. Bring a large pot of water to a boil."), every cookbook author seems to approach it differently. Some stick to skimmable one-liners. Other authors overload, stuffing each step so full that the home cook must stop, drop their spatula, and read before proceeding. Here be dragons, not necessarily standards.

While cooking a dish from someone else's recipe isn't *exactly* the same as reading through someone else's code, I do see a few key principles of clean code echoed in recipes with simple steps. Concepts like **single responsibility**, **clarity over brevity**, **navigability**, and **consistency** guide software engineers to craft code that is straightforward for humans to understand and build upon.

The Single Responsibility Principle (SRP)

I'll start with the **single responsibility principle** (**SRP**). Software engineers strive to design codebases such that each reusable component fulfills one clear-cut job. Think of the "add to cart" button on your favorite clothier's website: you'll find the same one on any item page (rendering it a "reusable component"), where it performs the same simple action (its one "clear-cut job"). That non-engineers can describe the button's job—updating your cart with an item—so sparingly, reflects the crispness of the code needed to make it work.

Clean code cannot exist without the SRP. I'd argue that recipes can't, either. The cookbooks I rely on pare down single steps to single tasks (or get damn close). In ***The Flavor Equation*'s** Roasted Broccolini + Chickpea Pancakes, for instance, author **Nik Sharma** separates the preparations of equipment, broccolini, and batter into their own respective unnumbered paragraph breaks. Other writers might shove all setup steps into one big paragraph, which belies the reality that each of these tasks are completed independently of the others. I also bemoan recipes in which the prep work has been obfuscated in the ingredients list, with instructions like "blanched" or "cooled" dangled after each ingredient like an afterthought. From a home cook's perspective, gathering ingredients is a separate task from cooking them.

Author **Hetty McKinnon** ups the SRP ante. In Cumin-Roasted Sweet Potato With Harissa Chickpeas and Spinach, one of her salad recipes from ***Neighborhood***, she kindly reminds the reader to preheat the oven in a paragraph that stands above and apart from all other text. Oven readiness is but one box the home cook must check while preparing a recipe, and McKinnon's recipes reflect that reality.

As with all rules, extenuating circumstances merit exceptions to the SRP. **Grace Young** does her best to tamp down the whistle-stop stir-fry into numbered steps throughout ***Stir-frying to the Sky's Edge***, but, since a single wok move might happen within mere seconds of its predecessor, sweeping several into a shared paragraph might prevent rather than foment home cooking errors. But flouting the SRP seldom helps the home cook. When authors use the SRP to whittle a step down to its essence, readers benefit from instructions that are easier to digest and check off on the fly.

Clarity Over Brevity (COB)

The principle of **clarity over brevity** (**COB**) can help further deepen code and recipe comprehension. This phrase has been incanted by my colleagues countless times to emphasize the importance of writing code that makes sense to *people* instead of short, smarty-pantsy code that just talks to the *computer*. Code quality isn't scored like golf: fewer keystrokes do *not* guarantee better code.

Code tends to be written in collaborative, asynchronous environments: colleagues working together towards a shared goal at different times, in different places. I'd count cookbooks among collaborative, asynchronous environments, too. Authors devise recipes before publication; readers cook from them many moons later. In these circumstances, clarity—helping someone else understand your instructions when you are not around to explain them—trumps brevity.

Consider the ternary operator. **Ternaries** offer a typically useful shorthand option to developers writing if/else instructions. For example, "if the user is in America, display temperature in Fahrenheit; else show it to them in Celsius." Where the logic is elementary, a one-line ternary can be preferable, as in the following pseudocode example:

```
var temperature = user.isInAmerica() ? temperatureInFahrenheit :
temperatureInCelsius
```

The equivalent if/else statement would take up multiple lines of code, as follows, which might impede a developer from gleaning its meaning at a glance:

```
var temperature

if user.isInAmerica() {

  temperature = temperatureInFahrenheit

} else {

  temperature = temperatureInCelsius

}
```

But the ternary ceases to be a sensible choice as soon as instructions deviate from the straightforward. Convoluted calculations with multiple special cases *could be* communicated in one mega-ternary, but an engineer would likely break this logic down into simpler statements that others can comfortably scan. Their teammates present and future stand a better chance of understanding this logic when written as a longer chunk of code. Here, clarity trumps brevity.

COB is crucial to recipes, too: after all, readers don't buy cookbooks that further confound them in the kitchen. That many modern releases feature photos alongside recipe steps attests to COB's centrality. Pictures are worth a thousand words, or so the idiom goes, and can serve recipe newbies as a lodestar across space and time. I would've struggled to tuck apricot slices into concentric circles had **Lauren Ko** not shared the process behind her Swirled Peace Tart in both words *and* pictures. Her pictorial ***Pieometry*** epitomizes COB.

But pictures are, well, only one part of the picture. I've seen COB at work in an heirloom copy of *The Food of Greece*, a seventies cookbook *without* illustrations. Author **Vilma Liacouras Chantiles** compensates by dovetailing Greek (albeit romanized) food names with verbose English explainers. She demystifies the pastry diples—which, in my opinion, handily best the better-known baklava—with the description "crisp bubble bows, knots, and rolls honeyed with nuts and cinnamon." Though these extra words don't allay the difficulties of actually *wrangling* diples, they act as a lifesaver for the lost home baker to cling to as they endeavor to shape a pastry that looks right (more or less).

Someone once called naming things one of the two hardest problems in computer science; naming recipes can be equally fraught. Clarity doesn't mean oversimplifying a recipe to bend it to thoughtless "mainstream" (white) comprehension. Verbose descriptions can clue in the curious reader without sacrificing a recipe's integrity. At the outset of each of her mother's Afghan recipes, ***Parwana*** author **Durkhanai Ayubi** prominently displays her father's handwritten Farsi alongside its romanization. She then follows up with a few paragraphs that anatomize the dish and shed light on its sociocultural context. ***In Bibi's Kitchen*** by **Hawa Hassan** marches to a similar beat, which made my first time cooking a new-to-me dish like Kunde (Black-Eyed Peas and Tomatoes in Peanut Sauce) manageable, informative, and straight-up delightful.

Navigability and Consistency

Engineers regard navigability and consistency as two other key proxies for code cleanliness. Calling a codebase **navigable** connotes a sea of files and components through which a developer can independently and confidently maneuver to find what they're looking for. A navigable codebase might, say, mandate that each file order its components in a particular way. Amongst developers of iOS applications, it is common to spotlight the order with something called **pragma marks**, marginalia under which we nest kindred code à la a desk drawer organizer. For example, one pragma mark might precede all code needed to arrange tab buttons in a tab bar, while another might engird the code that handles what happens when a user taps on them. Pragma marks serve as the section headers of long code files.

Consistency is companion to navigability: using similar stylistic choices to design similar code components prevents the reader from needing to familiarize themselves with both format *and* functionality before their eureka moment hits. In the codebase for an iOS application, each file that represents a user-facing screen—like a home feed, search page, or user profile—might be arranged with the same set of pragma marks that accommodate similar pieces of logic.

Reframing navigability and consistency as techspeak for tidiness might set your brain abuzz with ideas of how they crop up in cookbook recipes. In my book, **Stella Parks** reigns as the queen of these two principles. She stipples each recipe in her eponymous ***BraveTart*** with something like the pragma mark. Within the recipe body, she erects headers for steps with a shared goal: in "Snickerdoodles," they read "make the dough" and "dust and bake the cookies." Outside of the recipe body, she proposes ways in which a home baker might tinker with the steps that she's written: I, for one, followed her instructions to transmogrify her base recipe into banana bread snickerdoodles. She'll sometimes throw experimental bakers an additional bone with a chart like "Custom Snickerdoodles," which crumbles the cookie to its components: a "sugar sprinkle," "primary flavor," and "aromatic." Her recipes all abide by this format, which helps readers like me home in on the process rather than its periphery.

Comments as Common Ground

I've rhapsodized at length about some of the stylistic parallels between clean coding and straightforward recipe writing. But there's one more likeness I've observed that, though admittedly tangential, complements both of these practices: levity! A ready example lies in the **code comment**, a developer's annotation. The funnest one that I've ever encountered prefaces a particularly thorny code block with three doomed words: "hear me out." I love humorous comments like this one, but I also see them as having a higher purpose: they humanize the process of creating software. Coders write comments for people, not machines, and that awareness reminds them to make their instructions human-friendly with marginalia when they can't do so with code alone.

Repartee likely plays a similar role in recipe writing. It'd be harder to pick out a cookbook that *doesn't* stray from full-on formality with a few comments—after all, starch is for eating, not reading!—but I'll toss out two examples. **Bryant Terry** gifts the home cook with a song that harmonizes with the steps of each recipe in ***Vegetable Kingdom***. In *Six Seasons*, **Joshua McFadden** wraps up his final step for Brined and Roasted Almonds with a quip: you *could* store them for a fortnight in an airtight container, but only "if you don't eat them all sooner." Cookbook authors' casual commentary breaks the fourth wall and evinces a wholesome truth: they wrote their steps with real-life readers in mind.

Whether or not you buy into that twee take, I hope that my juxtaposition of code and cookbook recipes has charmed you. But, for the nerdiest food industry folks and home cooks out there, I do harbor one more hope: may my comparison grant you a novel lens through which to consider your own preferences as a recipe reader and perhaps even guide your practices as a recipe writer. We all stand to benefit from your step wisdom.

Alaina Kafkes is a writer and software engineer based in San Francisco. Read more of her vegetable-forward home cooking stories on salad days, or, if you're not in a food mood, check out the latest entries on her catch-all blog. Follow her on Twitter at @alainakafkes.

3

Subscribe

← Previous

Write a comment...

Ready for more?

Subscribe

© 2021 Paula Forbes. See privacy, terms and information collection notice

Publish on Substack

Stained Page News is on Substack – the place for independent writing