**[The Everybody Poops Rule](http://rosstuck.com/the-everybody-poops-rule" \o "The Everybody Poops Rule)**

**AUGUST 26, 2018**

One of the hardest things to teach growing devs is when not to use their newfound skills. After putting huge effort into learning with a mentor, having that same mentor pick up a feature and say “No, just crap this one out, it’s not important” seems inconsistent and bewildering.

To help explain, I use what I call “The Everybody Poops Rule.” It goes like this:

*Everybody poops. But you don’t poop in every room in the house. You have a special room where you poop, you put a door on it and you only poop there.*

Okay, it’s crass but that makes it memorable. There’s two key points here:

First, in the same way DDD is about admitting there is no single consistent model of the world or that TDD is about admitting failures are unavoidable, the Everybody Poops rule is about admitting some parts of a system will be…well, crap. Not every part of the system could or should be top tier quality. Eventually, everybody poops. It’s a natural process.

But, and this is the second point, that doesn’t mean it’s unmanageable. DDD copes with the lack of a consistent model by embracing and integrating multiple models. TDD puts verification upfront, admitting all code is untrustworthy until proven otherwise.

The Everybody Poops rule responds to the existence of crap by embracing encapsulation. That company blog? It’s not a core system, throw it together. A shell script? Pffft, crap it out. That little framework glue code? Psha, it’ll be removed in the next framework release.

When in doubt, default to quality. But if the code is isolated, unimportant and behind an interface: who cares if it smells? We can still build a reasonable codebase, we can live in a reasonable house, as long as we only poop in specific areas and you keep the door closed.

Let’s talk about some of the implications.

**Let It Go**

First and foremost, you can’t stop people from pooping. When dealing with a sufficiently complex system, there will always be waste. Holding it in is unhealthy and impossible.

Most teams follow the Broken Window Theory, fearing even a single tradeoff starts the slide down a slippery slope. This can reduce discussion (read: dissension) in the short term but leads to arbitrary compliance or worse.

Deciding on a level of quality isn’t like deciding on a coding standard, you can’t have an off-the-shelf-always-okay answer. Quality is the place to have nuanced discussions.

**Not on The Walls**

Even when you are producing crap, there are still standards of decency. Maybe the team is fine with a God Class, or a really long method, maybe it doesn’t have all the unit tests, that’s okay but it still better have docs and pass your static analyzers.

Finding the right balance can be difficult since you’re already lowering your standards. Focus on the tradeoffs here, including the time and energy spent discussing.

**Encrapsulation**

How does one isolate crap? In practical terms, any way you normally split code. Put it behind an interface. Drop in a service layer. Pull it out to another namespace. Split it into a separate codebase. Delegate to a third-party package. Build a bridge interface. Push it behind an API.

Remember, none of this erases the existence of the poop. Putting it behind a door doesn’t somehow make it clean or better. It’s just hopefully somewhere you don’t have to touch it all the time.

This isn’t easy work. Perversely, it often takes more skill to handle poop safely than it would to make to not make crap in the first place. Poop juggling is a senior ability.

**Waste Management Systems**

Never put poop behind a door unless you have a plan to get rid of it later.

That might seem silly since the poop is, by definition, unimportant. But ask yourself: if you can’t flush it, is it really isolated? Without good waste management, poop has a nasty habit of backing up and flooding you at the worst moment. And poop rarely ages well.

Besides, it might be an unimportant detail now but times change. Make sure you’re in a good position to replace or redirect that code if the business pivots. Today’s manure is tomorrow’s fertilizer.

**Iterable Bowel Syndrome**

Other times, you might want to go in and make a deliberately crappy pass at building an important system. Even worse than a spike or proof-of-concept, these barebone first drafts may stink but they’re a great way of gaining knowledge or scaffolding other systems.

Just make a firm commitment to flush it at the end. If you name it, you (or your manager) is going to get attached to it.

**Don’t Force It**

Just because you can manage poop successfully doesn’t mean you should do it everywhere. No amount of engineering will ever make a kitchen toilet a good idea. Similarly, don’t crap in the domain layer.

Before you make the decision to crap something out, ask yourself:

* How often do you have to touch it?
* Is it going to hang around for a long time?
* Is it a core domain for your company, e.g. do you draw competitive advantage from it?
* What are the consequences if the project fails in development? At each quarter of its projected life cycle?
* What are the consequences if it exceeds its lifecycle?
* Do you know the maximum and minimum amount of complexity possible?
* Can you find a solution to keep the complexity “bounded”, i.e. within a certain range?

**Raising a Stink**

If you do make the decision to under engineer something, bring your stakeholders into the decision. They’ll know if it’s worth it and it avoids surprise. Nothing undermines confidence in an engineering team faster than someone else coming along and saying “What is this crap?!” without knowing it’s supposed to be crap.

Make sure you document it. Not only will this help cover your butt, this is the stuff you want to touch the least so it’s the most likely to be forgotten. Finally, this documentation signals to new hires that quality is the standard and YOLO is a rare deviation.

Teach the standard. Document the deviation.

**In the End**

Many have commented that the best devs often write code indistinguishable from the newest juniors. On a line by line basis, that might be true. But that’s okay. When the work is done, you’ll realize their focus was somewhere else the whole time.

They weren’t worrying about the plumbing.

They were building the house.

*Thanks to Cees-Jan Kiewiet, Igor Wiedler, Frank de Jonge, Anthony Ferrera and Matthias Noback for proofreading.*

[PHP](http://rosstuck.com/tags/index.html#php)[ARCHITECTURE](http://rosstuck.com/tags/index.html#architecture)[THE EVERYBODY POOPS RULE](http://rosstuck.com/the-everybody-poops-rule) WAS PUBLISHED ON AUGUST 26, 2018 [ROSS TUCK](http://rosstuck.com/about)

<http://rosstuck.com/the-everybody-poops-rule>