Arlo Being Bloody Stupid

Conclusively demonstrating why you should listen to others…

Naming is a process, part 3: Nonsense to Honest

August 25, 2015 by Arlo

In part 1 we talked about naming as a process. We talked about how legacy code is really defined by its poor legibility, and that reading is the core of coding. And we talked about how working effectively with legacy code is simply the process of having an insight, writing it down, and checking it in.

Later parts have gotten us to a Nonsense name.

Now let's look deeply at the second transition in names, from Nonsense to Honest.

We have a thing with a name. But its name is useless. In my FAA system example, the method was named preLoad(). This tells me when the system calls the method (not very useful), but doesn't tell me what the method does or why one would want to call it.

I want to make the name honest. It doesn't need to be completely honest. It just needs to tell me one honest thing. I look at the method to figure out one thing that appears to be key to its execution. I just read through the body and see what appears central.

> **Where to look:** inside the body. Look for patterns that repeat here or between this and other methods.

I find it useful to look for variables that are system components ("database," "screen," "network," "service"), look where the return value comes from, and see if something is used many times.

In the example, I noticed that there was a global named db that was used a lot. Several places created recordset objects, set values, and read or wrote them using the db. Since it mixed reads and writes, I named the method doSomethingEvilToTheDatabase().

> **Our insight:** one thing the code does. And perhaps a judgment of how easy it will be to work with, how safe it is, or other useful traits.

That name isn't complete, but it is honest.

It is honest in several ways. It actually records multiple insights I'd had:

1. The main effect of the method seems to involve the database.
2. I don't yet understand what it is doing to the database.
3. I am getting all judgy about how it is treating the poor database. I don't like it.

The second and third insights are as important as the first. Even if I wandered away at this point, a future reader of the code would have some insight. They wouldn't know what the method did, but they'd be properly wary of it.

They'd know that no one knows exactly what it does, it does that thing to the central database, and the last person to touch it thought it was doing some nasty stuff. Good! I've transmitted all the knowledge that I have about this method. I've put them in the proper frame of mind for working with it.

> **What we write down:** a better name, which says one honest thing and is clear about what we don't know yet.

To write down the insight we use one more refactoring:

- Rename

Be specific!

The most common way to screw up at this point is to be too general in your name. Specific is good!

The intent of naming in this way is to allow a reader to understand a chunk of code without looking at it, just by looking at the name for that chunk of code. That reader needs specific detailed knowledge. So the name has to be specific.

For example, I could have named my function handleFlightInfoSomehow(). That would have been honest. But it doesn't give a clear insight about what the code is doing.

Looking ahead just a bit, the next step will be to make the name convey everything the code is doing (be Honest and Complete). If we are specific, then the only route to completeness is to add to the name. I keep the same precision, but add more description.

If I am overly general / imprecise now, then it most or all of the things the function does will be already roughly describable by the name. This will make it a lot harder for me to see the things that aren't yet conveyed in the name, which will make it harder to create a name that allows me to not read the code behind it.

Not just for functions

This step also applies when naming classes or variables. Common nonsense names for classes are anything that ends in -Manager or -Operations. Nonsense variables are usually named the same thing as their type.

In either case, we need to have one insight about this thing. What is one important responsibility of the class? For a variable, what differentiates this one Foo instance from all the other Foo instances that might be out there? How does the code using the variable think of this one? Whatever that insight is, write it down.

The Naming is a Process blog series

1. Good naming is a process, not a single step
2. Missing to Nonsense
3. Nonsense to Honest (this entry)

4. Honest to Honest and Complete
5. Honest and Complete to Does the Right Thing
6. Does the Right Thing to Intent
7. Intent to Domain Abstraction
8. Summary and Learning Path (will publish Tuesday, 9/1/2015)

Tags: design, legacy code, naming, naming is a process, refactoring, tdd

I really liked the practical approach to refactoring methods and using naming as...
» 3 weeks ago

The last comments for
Good naming is a process, not a single step

 abelshee 50p

I'm glad you are finding it useful. Awareness of technical debt and how to work with it is the ...
» 3 weeks ago

The last comments for
Llewellyn Falco - What makes a good test suite?

 kingroot apk new 14p

i thing granularity is the most importont among the 4 factors
» 4 weeks ago

The last comments for
Good naming is a process, not a single step

 Filip

Oh man, this is just what I was looking for. What you describe as “indebted code” is something...
» 6 weeks ago

The last comments for
Naming is a Process, Part 7: Intent to Domain Abstraction

 abelshee 50p

It will. But it is currently blocked on a software project. I mentioned the learning path? Well, I'm...
» 6 weeks ago

The last comments for
The Core 6 Refactorings

 abelshee 50p

Interesting. Very different context / direction. I'm not looking for best.

I'm looking...
» 6 weeks ago

The last comments for
Naming is a Process, Part 6: Does the Right Thing to Intent

 abelshee 50p

I hear you. I've heard others with the same request. I think it would make it better. And I'm...
» 6 weeks ago

Comments by IntenseDebate

Top 5 Posts

- **The No Mocks Book** (33 comments)
- **Is Pair Programming for Me?** (23 comments)
- **How I Learned to Stop Worrying and Love the Mock** (20 comments)
- **Scaling Agile - the Easy Way** (15 comments)
- **That's Not Agile** (13 comments)

Comments by IntenseDebate

Theme: MistyLook by Sadish.