Simplicity is the Key

A simple design always takes less time to finish than a complex one. So always do the simplest thing that could possibly work next. If you find something that is complex replace it with something simple. It's always faster and cheaper to replace complex code now, before you waste a lot more time on it.

Many people try to measure simplicity. Simple defies measurement because it is a very subjective quality. One person's simple is another person's complex. Adding an advanced technology can simplify one application and make a complete mess of another.

Within your project the team decides what is simple. Together you judge your code subjectively. I recommend four subjective qualities; Testable, Understandable, Browsable, and Explainable (TUBE).

Testable means you can write unit tests and acceptance tests to automatically check for problems. This impacts the overall design and coupling of objects in your application. Break your system into small testable units.

Browsable is the quality of being able to find what you want when you want it. Good names helps you find things. Using polymorphism, delegation, and inheritance correctly helps you find things when you need to.

Understandable is obvious, but highly subjective. A team that has been working with a system a long time will understand it even though someone new is completely baffled. So I also recommend explainable as a quality that means it's easy to show new people how everything works.

Many people recommend a measure of simplicity as: Once and only once. It is important to remember this has two parts. Everyone seems to understand the "only once" part but not the "once and" part. The first part means express all the intention of your code even if it means duplication.

Consider a simple example; you multiply by 100 to turn a fraction into percentage and you also multiply by 100 to turn meters into centimeters. Should you have a single function that multiplies by 100 called convertToPercentOrMeters(x)? NO! Not even if it would remove some duplication. You want two methods; converToPercent(aFraction), and convertMetersToCentimeters(aLength). You want two because they tell you different things. Not just that you will multiply by 100 but also why you will multiply by 100 and what kinds of numbers are valid inputs.

One thing about simple designs is that they require knowledge to recognize. Knowledge is different from information, you can have plenty of information and no knowledge. Knowledge is insight into your problem domain that develops over time.

Simple designs often come after the project has been running for a while. The best approach is to create code only for the features you are implementing while you search for enough knowledge to reveal the simplest design. Then refactor incrementally to implement your new understanding and design.

Keep things as simple as possible as long as possible by never adding functionality before it is scheduled. Beware though, keeping a design simple is hard work.

Portland Pattern XProgramming.com Portland Pattern

ExtremeProgramming.org home | XP Rules | System Metaphor | About the Author

AARP
Focus on What Really Matters
Get instant access to member benefits and helpful resources, so you can get back to what's important.
LEARN MORE