Advice of the Gurus

John Vlissides
C++ Report, November/December ’98

I’d like to harp on a couple of things. The first of them regards a healthy fear of complexity and its importance to all that we do as software developers. Programmers, especially the brightest ones, seem to waste no time getting to the limits of their complexity-handling abilities. After all, that’s where the coolest challenges lie, right? And don’t tough problems demand complex solutions? Besides, if I’m one of the few who understand the system, that makes me pretty important, does it not?

Maybe. But most of the complexity in your garden-variety program isn’t a necessary evil; it is not intrinsic to the problem being addressed. It’s a product of developmental entropy—too many people doing too many things to too much code, over not enough time. Ergo the pervasive tendency toward design-by-accretion, rather than discretion. No system is immune to it. I’ve seen weeds of complexity rise up and choke even well-tended and thriving software.

So developers must be proactive against complexity. If I could prescribe a drug for that, it would be one that induced a craving for minimalism. It would inoculate the patient against complexity in all its guises, be it creeping featurism, overgeneralization, boil-the-ocean design tendencies—whatever. The complexity of C++ makes a minimalist attitude all the more vital.

The other thing I want to underscore here is how to go about reading *Design Patterns*, a.k.a. the “GoF” book. Many people feel that to fully grasp its content, they need to read it sequentially. But GoF is really a reference book, not a novel. Imagine trying to learn German by reading a Deutsch-English dictionary cover-to-cover—it just won’t work! If you want to master German, you have to immerse yourself in German culture. You have to *live* German. The same is true of design patterns: you must immerse yourself in software development before you can master them. You have to *live* the patterns.

Read *Design Patterns* like a novel if you must, but few people will become fluent that way. Put the patterns to work in the heat of a software development project. Draw on their insights as you encounter real design problems. That’s the most efficient way to make the GoF patterns your own.