

## **My Worried Mind**

**Jerry Fitzpatrick**

**I'm** both thrilled and worried about the fate of the software industry over the next ten years.

Although we often use the term “software engineering” to describe our activities, the truth is that writing programs is still largely a craft. Advertising yourself as a software engineer is actually a crime in many parts of the United States.

Nevertheless, I believe that we stand on the threshold of a true engineering age for software. Object oriented programming is becoming the norm. The C++ language is now standardized and becoming widely used. The Unified Modeling Language is helping us communicate more effectively with one another.

But what really differentiates engineering from craft work?

By definition, engineering is the application of scientific principles to practical uses. For example, electrical engineers apply their scientific knowledge to design and build radios, telephones, computers, and so on. Unlike electronics and other hard sciences, however, we software folk have no natural laws or principles to limit us. No Ohm's Law. No Boyle's Law. No Einsteinian theories. Yet principles form the basis of all engineering disciplines.

The good news is that we are beginning to identify and apply development principles in our daily work. I don't mean nebulous principles like “use CASE tools”. I'm referring to precise and powerful maxims such as the Open-Closed Principle, the Liskov Substitution Principle, and the Dependency Inversion Principle. Although such principles do not stem from natural laws, they generally reflect decades of experience and profound clarity of thought. To qualify as a true principle, advice should not only be meaningful, it should have broad applicability. Most importantly, it should provide measurable results.

Many years ago, Tom DeMarco wrote that you can't control what you can't measure. Unless we can all agree to use a consistent set of principles and measurements, how can we ever be confident that our programs are safe, usable, reliable, and maintainable: in short, professional?

I know that some developers prefer to think of themselves as artists and have no interest in becoming engineers. Their primary objection is that the introduction of scientific principles will stifle creativity and make programming less fun.

In all fairness, principles do provide restrictions that can be viewed as confining. At the same time however such restrictions can help us pull together instead of separate directions. Moreover, the use of principles leaves plenty of room for creativity and may in fact provide new challenges. One of the most compelling benefits of a scientific framework is that it can be used to objectively compare design and implementation alternatives. Contrast this with our all-too-common discussions about the subjective “evils” of a design we happen to dislike.

Over the years, I've worked with a multitude of developers in a variety of business environments. Having analyzed, developed, and modified programs using so many different objectives and styles, I've consistently found two problems affecting the majority of applications: insufficient abstraction and insufficient modularity. As fundamental as these attributes are, why don't we have universal principles that tell us how to create appropriate abstractions and modules? Furthermore, why don't we have universal abstraction and modularity metrics that help us improve our applications?

As computer technology accelerates into the next millennium, our development tools will no doubt become more powerful and plentiful. As we begin to build ever-larger applications in shorter periods of time, one question remains: will we this power wisely?

I'm thrilled that the software industry has come this far and that we may be near an engineering age of computing. But I also worry that ignorance, territorialism, and fear will conspire to deprive us of this achievement. Whether you prefer to be called an artist or an engineer, only a principled approach to software construction will inspire respect and confidence from your customers. With this in mind, I challenge each of you to discover, learn, and apply only the finest principles of software development.