PRINCIPLE 1

Don't reinvent the wheel.

Now that's a good bit of advice, isn't it? Few would disagree with it, and yet we can easily fall into the trap of disregarding this fundamental principle. Here are a few common reasons:

- **We're just doing a small, simple system.** The assumption is that it isn't worth the effort or the cost to build a "real" enterprise system. If you really are just creating something trivial, then sure, do whatever works. But what starts out small and simple often grows into something more complex that must handle greater demands. If your system grows by tacking functionality onto an architecture that was meant for something far simpler, it will fail.
- **Our problem is unique.** Are you sure that your problem is different from anything anyone has ever solved before? Do you really believe that you need to invent a completely original solution? If so, that's beyond the scope of this book. Good luck!
- **We do not have time to do research.** So, you are in a big hurry, and you do not want to spend the time to find out what others have already done. But you do have time to charge off and develop something new, and then test it, and perhaps do it over again because the first attempt didn't work, and then test again, and develop again. Does this make any sense?
- **We want to invent something new.** This book is about building successful enterprise systems that will solve the problems in your domain. If your goal is to become rich and famous by devising new standards and inventing new infrastructure software, then you do not want to read this book. But I'll promise to read yours after you have become famous.

Someone Else Has Already Solved Your Problem

It is most likely that substantial portions of your problem have already been solved by others. Take advantage of their experience by reading their articles or using shareware they have contributed. Hire them if you can.

This is not to say that someone else's solution will always work for you. Every problem is different in some way, and no solution fits all. Identify what is unique about your problem and isolate those portions. Define these unique portions so that they are as small as possible. Then it will be worth your while to make the rest of your problem fit one or more of the previous solutions so that you can concentrate on your small, unique portion.

Understand What Your Added Value Is

You need to partition your problem into the portions that are unique to your domain and everything else. Your solutions to the unique portions of your problem represent your *added value*.

The added value of your enterprise system is what solves the particular problems of your domain. Everything else is mostly infrastructure, such as network protocols, security, scalability, and the like. Doing the added value right will make your system successful. Therefore, it makes sense to spend the larger amount of your time identifying, understanding, and creating your added value, and far less time on infrastructure code that others have already written.

Use Commercial Software Whenever Practicable

Commercial software packages are great at providing generic solutions to generic problems. You can configure most packages to solve a range of problems.

Use commercial software to build the nonunique portions of your system. Often this is your system's infrastructure. For example, a commercial application server can take care of network access, security, scalability, reliability, and so on. It is not worth your time, effort, and expense to build infrastructure and support code—good software vendors have already tested and validated their solutions, especially if their packages have been around for a while and have long lists of satisfied customers.

MISSION NOTES

Make vs. buy? That was an important question for the CIP middleware. Should we attempt to make a custom middleware infrastructure, or should we buy a commercially available application server?

We decided to purchase licenses for the WebLogic application server from BEA Systems to form the basis of CIP's middleware infrastructure. Our added value consists of the services we needed to develop to support the Mars rover mission, and these services run within WebLogic. Our task was not to invent and implement new middleware infrastructure standards. After all, the users benefit from the mission-related services that CIP provides, and the middleware just has to remain invisible but reliable. (I'll have more to say later about invisible middleware.)

Be wary of the trap of thinking you cannot afford a commercial solution. Carefully weigh the cost of purchasing software (and its support and maintenance) versus the cost of recreating it, especially if doing so takes resources away from working on your added value. Keep in mind that not getting the infrastructure right is a good guarantee that your enterprise system will fail—and failure can be very costly indeed.