Foreword

by John Crupi

What do you do when a new technology arrives? You learn the technology. This is exactly what I did. I studied J2EE (being from Sun Microsystems, it seemed to be the logical choice). Specifically, I focused on the EJB technology by reading the specifications (since there were no books yet). Learning the technology, however, is just the first step—the real goal is to learn how to effectively apply the technology. The nice thing about platform technologies is that they constrain you to performing certain tasks. But, as far as the technology is concerned, you can do whatever you want and quite often get into trouble if you don't do things appropriately.

One thing I've seen in the past 15 years is that there seem to be two areas that software developers obsess over: programming and designing—or more specifically, programming and designing effectively. There are great books out there that tell you the most efficient way to program certain things in Java and C#, but far fewer tell you how to design effectively. That's where this book comes in. When Deepak Alur, Dan Malks, and I wrote *Core J2EE Patterns*, we wanted to help J2EE developers "design" better code. The best decision we made was to use patterns as the artifact of choice. As James Baty, a Sun Distinguished Engineer, puts it, "Patterns seem to be the sweet spot of design." I couldn't agree more, and luckily for us, Gregor and Bobby feel the same way.

This book focuses on a hot and growing topic: integration using messaging. Not only is messaging key to integration, but it will most likely be the predominant focus in Web services for years to come. There is so much noise today in the Web services world, it's a delicate and complex endeavor just to identify the specifications and technologies to focus on. The goal remains the same, however—software helps you solve a problem. Just as in the early days of J2EE and .NET, there is not a lot of design help out there yet for Web services. Many people say

Web services is just a new and open way to solve our existing integration problems—and I agree. But, that doesn't mean we know how to design Web services. And that brings us to the gem of this book. I believe this book has many of the patterns we need to design Web services and other integration systems. Because the Web service specifications are still battling it out, it wouldn't have made sense for Bobby and Gregor to provide examples of many of the Web service specifications. But, that's okay. The real payoff will result when the specifications become standards and we use the patterns in this book to design for those solutions that are realized by these standards. Then maybe we can realize our next integration goal of designing for service-oriented architectures.

Read this book and keep it by your side. It will enhance your software career to no end.

John Crupi
Bethesda, MD
August 2003

Foreword

by Martin Fowler

While I was working on my book *Patterns of Enterprise Application Architecture*, I was lucky to get some in-depth review from Kyle Brown and Rachel Reinitz at some informal workshops at Kyle's office in Raleigh-Durham. During these sessions, we realized that a big gap in my work was asynchronous messaging systems.

There are many gaps in my book, and I never intended it to be a complete collection of patterns for enterprise development. But the gap on asynchronous messaging is particularly important because we believe that asynchronous messaging will play an increasingly important role in enterprise software development, particularly in integration. Integration is important because applications cannot live isolated from each other. We need techniques that allow us to take applications that were never designed to interoperate and break down the stovepipes so we can gain a greater benefit than the individual applications can offer us.

Various technologies have been around that promise to solve the integration puzzle. We all concluded that messaging is the technology that carries the greatest promise. The challenge we faced was to convey how to do messaging effectively. The biggest challenge in this is that messages are by their nature asynchronous, and there are significant differences in the design approaches that you use in an asynchronous world.

I didn't have space, energy, or frankly the knowledge to cover this topic properly in *Patterns of Enterprise Application Architecture*. But we came up with a better solution to this gap: find someone else who could. We hunted down Gregor and Bobby, and they took up the challenge. The result is the book you're about to read.

I'm delighted with the job that they have done. If you've already worked with messaging systems, this book will systematize much of the knowledge that you and others have already learned the hard way. If you are about to work with messaging systems, this book will provide a foundation that will be invaluable no matter which messaging technology you have to work with.

Martin Fowler
Melrose, MA
August 2003