Preface

Design patterns and object-oriented programming. They hold such promise to make your life as a software designer and developer easier. Their terminology is bandied about every day in the technical and even the popular press. But it can be hard to learn them, to become proficient with them, to understand what is really going on.

Perhaps you have been using an object-oriented or object-based language for years. Have you learned that the true power of objects is not inheritance but is in "encapsulating behaviors"? Perhaps you are curious about design patterns and have found the literature a bit too esoteric and high-falutin. If so, this book is for you.

It is based on years of teaching this material to software developers, both experienced and new to object orientation. It is based upon the belief—and our experience—that once you understand the basic principles and motivations that underlie these concepts, why they are doing what they do, your learning curve will be incredibly shorter. And in our discussion of design patterns, you will understand the true mindset of object orientation, which is a necessity before you can become proficient.

As you read this book, you will gain a solid understanding of the ten most essential design patterns. You will learn that design patterns do not exist on their own, but are supposed to work in concert with other design patterns to help you create more robust applications. You will gain enough of a foundation that you will be able to read the design pattern literature, if you want to, and possibly discover patterns on your own.

Most importantly, you will be better equipped to create flexible and complete software that is easier to maintain.

From Object Orientation to Patterns to True Object Orientation

In many ways, this book is a retelling of my personal experience learning design patterns. Prior to studying design patterns, I considered myself to be reasonably expert in object-oriented analysis and design. My track record had included several fairly impressive designs and implementations in many industries. I knew C++ and was beginning to learn Java. The objects in my code were well-formed and tightly encapsulated. I could design excellent data abstractions for inheritance hierarchies. I thought I knew object-orientation.

Now, looking back, I see that I really did not understand the full capabilities of object-oriented design, even though I was doing things the way the experts advised. It wasn't until I began to learn design patterns that my object-oriented design abilities expanded and deepened. Knowing design patterns has made me a better designer, even when I don't use these patterns directly.

I began studying design patterns in 1996. I was a C++/object-oriented design mentor at a large aerospace company in the northwest. Several people asked me to lead a design pattern study group. That's where I met my co-author, Jim Trott. In the study group, several interesting things happened. First, I grew fascinated with design patterns. I loved being able to compare my designs with the designs of others who had more experience than I had. I discovered that I was not taking full advantage of designing to interfaces and that I didn't always concern myself with seeing if I could have an object use another object without knowing the used object's type. I noticed that beginners to object-oriented design—those who would normally be deemed as learning design patterns too early—were benefiting as much from the study group as the experts were. The patterns presented examples of excellent object-oriented designs and illustrated basic object-oriented principles, which helped to mature their designs more quickly. By the end of the study session,

I was convinced that design patterns were the greatest thing to happen to software design since the invention of object-oriented design.

However, when I looked at my work at the time, I saw that I was not incorporating *any* design patterns into my code.

I just figured I didn't know enough design patterns yet and needed to learn more. At the time, I only knew about six of them. Then I had what could be called an epiphany. I was working on a project as a mentor in object-oriented design and was asked to create a high-level design for the project. The leader of the project was extremely sharp, but was fairly new to object-oriented design.

The problem itself wasn't that difficult, but it required a great deal of attention to make sure the code was going to be easy to maintain. Literally, after about two minutes of looking at the problem, I had developed a design based on my normal approach of data abstraction. Unfortunately, it was very clear this was not going to be a good design. Data abstraction alone had failed me. I had to find something better.

Two hours later, after applying every design technique I knew, I was no better off. My design was essentially the same. What was most frustrating was that I knew there was a better design. I just couldn't see it. Ironically, I also knew of four design patterns that "lived" in my problem but I couldn't see how to use them. Here I was—a supposed expert in object-oriented design—baffled by a simple problem!

Feeling very frustrated, I took a break and started walking down the hall to clear my head, telling myself I would not think of the problem for at least 10 minutes. Well, 30 seconds later, I was thinking about it again! But I had gotten an insight that changed my view of design patterns: rather than using patterns as individual items, I should use the design patterns together.

Patterns are supposed to be sewn together to solve a problem.

I had heard this before, but hadn't really understood it. Because patterns in software have been introduced as *design* patterns, I had always labored under the assumption that they had mostly to do with design. My thoughts were that in the design world, the patterns came as pretty much well-formed relationships between classes. Then, I read Christopher Alexander's amazing book, *The Timeless Way of Building*. I learned that patterns existed at all levels—analysis, design, and implementation. Alexander discusses using patterns to help in the understanding of the problem domain (even in describing it), not in creating the design after the problem domain is understood.

My mistake had been in trying to create the classes in my problem domain and then stitch them together to make a final system, a process which Alexander calls a particularly bad idea. I had never asked if I had the right classes because they just seemed so right, so obvious; they were the classes that immediately came to mind as I started my analysis, the "nouns" in the description of the system that we had been taught to look for. But I had struggled trying to piece them together.

When I stepped back and used design patterns and Alexander's approach to guide me in the creation of my classes, a far superior solution unfolded in only a matter of minutes. It was a good design and we put it into production. I was excited—excited to have designed a good solution and excited about the power of design patterns. It was then that I started incorporating design patterns into my development work and my teaching.

I began to discover that programmers who were new to object-oriented design could learn design patterns, and in doing so, develop a basic set of object-oriented design skills. It was true for me and it was true for the students that I was teaching.

Imagine my surprise! The design pattern books I had been reading and the design pattern experts I had been talking to were saying that you really needed to have a good grounding in object-oriented design before embarking on a study of design patterns. Nevertheless, I saw, with my own eyes, that students who learned object-oriented design concurrently with design patterns learned object-oriented design faster than those just studying object-oriented design. They even seemed to learn design patterns at almost the same rate as experienced object-oriented practitioners.

I began to use design patterns as a basis for my teaching. I began to call my classes *Pattern Oriented Design: Design Patterns from Analysis to Implementation*.

I wanted my students to understand these patterns and began to discover that using an exploratory approach was the best way to foster this understanding. For instance, I found that it was better to present the Bridge pattern by presenting a problem and then have my students try to design a solution to the problem using a few guiding principles and strategies that I had found were present in most of the patterns. In their exploration, the students discovered the solution—called the Bridge pattern—and remembered it.

In any event, I found that these guiding principles and strategies could be used to "derive" several of the design patterns. By "derive a design pattern," I mean that if I looked at a problem that I knew could be solved by a design pattern, I could use the guiding principles and strategies to come up with the solution that is expressed in the pattern. I made it clear to my students that we weren't really coming up with design patterns this way. Instead, I was just illustrating one possible thought process that the people who came up with the original solutions, those that were eventually classified as design patterns, might have used.

A slight digression.

The guiding principles and strategies seem very clear to me now. Certainly, they are stated in the "Gang of Four's" design patterns book. But I it took me a long time to understand them because of limitations in my own understanding of the object-oriented paradigm. It was only after integrating in my own mind the work of the Gang of Four with Alexander's work, Jim Coplien's work on commonality and variability analysis, and Martin Fowler's work in methodologies and analysis patterns that these principles became clear enough to me to that I was able to talk about them to others. It helped that I was making my livelihood explaining things to others so I couldn't get away with making assumptions as easily as I could when I was just doing things for myself.

My abilities to explain these few, but powerful, principles and strategies improved. As they did, I found that it became more useful to explain an increasing number of the Gang of Four patterns. In fact, I use these principles and strategies to explain 12 of the 14 patterns I discuss in my design patterns course.

I found that I was using these principles in my own designs both with and without patterns. This didn't surprise me. If using these strategies resulted in a design equivalent to a design pattern when I knew the pattern was present, that meant they were giving me a way to derive excellent designs (since patterns are excellent designs by definition). Why would I get any poorer designs from these techniques just because I didn't know the name of the pattern that might or might not be present anyway?

These insights helped hone my training process (and now my writing process). I had already been teaching my courses on several levels. I was teaching the fundamentals of object-oriented analysis and design. I did that by teaching design patterns and using them to

illustrate good examples of object-oriented analysis and design. In addition, by using the patterns to teach the concepts of object orientation, my students were also better able to understand the principles of object orientation. And by teaching the guiding principles and strategies, my students were able to create designs of comparable quality to the patterns themselves.

I relate this story because this book follows much the same pattern as my course (pun intended). In fact, from Chapter 3 on, this book is very much the first day of my two-day course: *Pattern Oriented Design: Design Patterns from Analysis to Implementation*.

As you read this book, you will learn the patterns. But even more importantly, you will learn why they work and how they can work together, and the principles and strategies upon which they rely. It will be useful to draw on your own experiences. When I present a problem in the text, it is helpful if you imagine a similar problem that you have come across. This book isn't about new bits of information or new patterns to apply, but rather a new way of looking at object-oriented software development. I hope that your own experiences, connected with the principles of design patterns, will prove to be a powerful ally in your learning.

Alan Shalloway
December, 2000

From Artificial Intelligence to Patterns to True Object Orientation

My journey into design patterns had a different starting point than Alan's but we have reached the same conclusions:

- Pattern-based analyses make you a more effective and efficient analyst because they let you deal with your models more

abstractly and because they represent the collected experiences of many other analysts.

- Patterns help people to learn principles of object orientation. The patterns help to explain why we do what we do with objects.

I started my career in artificial intelligence (AI) creating rule-based expert systems. This involves listening to experts and creating models of their decision-making processes and then coding these models into rules in a knowledge-based system. As I built these systems, I began to see repeating themes: in common types of problems, experts tended to work in similar ways. For example, experts who diagnose problems with equipment tend to look for simple, quick fixes first, then they get more systematic, breaking the problem into component parts; but in their systematic diagnosis, they tend to try first inexpensive tests or tests that will eliminate broad classes of problems before other kinds of tests. This was true whether we were diagnosing problems in a computer or a piece of oil field equipment.

Today, I would call these recurring themes patterns. Intuitively, I began to look for these recurring themes as I was designing new expert systems. My mind was open and friendly to the idea of patterns, even though I did not know what they were.

Then, in 1994, I discovered that researchers in Europe had codified these patterns of expert behavior and put them into a package that they called Knowledge Analysis and Design Support, or KADS. Dr. Karen Gardner, a most gifted analyst, modeler, mentor, and human being, began to apply KADS to her work in the United States. She extended the Europeans work to apply KADS to object-oriented systems. She opened my eyes to an entire world of pattern-based analysis and design that was forming in the software world, in large part due to Christopher Alexander s work. Her book, *Cognitive Patterns* (Cambridge University Press, 1998) describes this work.

Suddenly, I had a structure for modeling expert behaviors without getting trapped by the complexities and exceptions too early. I was able to complete my next three projects in less time, with less rework, and with greater satisfaction by end-users, because:

- I could design models more quickly because the patterns predicted for me what ought to be there. They told me what the essential objects were and what to pay special attention to.
- I was able to communicate much more effectively with experts because we had a more structured way to deal with the details and exceptions.
- The patterns allowed me to develop better end-user training for my system because the patterns predicted the most important features of the system.

This last point is significant. Patterns help end-users understand systems because they provide the context for the system, why we are doing things in a certain way. We can use patterns to describe the guiding principles and strategies of the system. And we can use patterns to develop the best examples to help end-users understand the system.

I was hooked.

So, when a design patterns study group started at my place of employment, I was eager to go. This is where I met Alan who had reached a similar point in his work as an object-oriented designer and mentor. The result is this book.

I hope that the principles in this book help you in your own journey to become a more effective and efficient analyst.

James Trott
December, 2000

A Note About Conventions Used in This Book

In the writing of this book, we had to make several choices about style and convention. Some of our choices have surprised our readers. So, it is worth a few comments about why we have chosen to do what we have done.

Approach	Rationale
First person voice	This book is a collaborative effort between two authors. We debated and refined our ideas to find the best ways to explain these concepts. Alan tried them out in his courses and we refined some more. We chose to use the first person singular in the body of this book because it allows us to tell the story in what we hope is a more engaging and natural style.
Scanning text	We have tried to make this book easy to scan so that you can get the main points even if you do not read the body, or so that you can quickly find the information you need. We make significant use of tables and bulleted lists. We provide text in the outside margin that summarizes paragraphs. With the discussion of each pattern, we provide a summary table of the key features of the pattern. Our hope is that these will make the book that much more accessible.
Code fragments	This book is about analysis and design more than implementation. Our intent is to help you think about crafting good designs based on the insights and best practices of the object-oriented community, as expressed in design patterns. One of the challenges for all of us programmers is to avoid going to the implementation too early, doing before thinking. Knowing this, we have purposefully tried to stay away from too much discussion on implementation. Our code examples may seem a bit lightweight and fragmentary. Specifically, we never provide error checking in the code. This is because we are trying to use the code to illustrate concepts.
Strategies and principles	Ours is an introductory book. It will help you be able to get up to speed quickly with design patterns. You will understand the principles and strategies that motivate design patterns. After reading this book, you can go on to a more scholarly or reference book. The last chapter will point you to many of the references that we have found useful.

Show breadth and give a taste	We are trying give you a taste for design patterns, to expose you to the breadth of the pattern world but not go into depth in any of them (see the previous point). Our thought was this: If you brought someone to the USA for a two week visit, what would you show them? Maybe a few sites to help them get familiar with architectures, communities, the feel of cities and the vast spaces that separate them, freeways, and coffee shops. But you would not be able to show them everything. To fill in their knowledge, you might choose to show them slide shows of many other sites and cities to give them a taste of the country. Then, they could make plans for future visits. We are showing you the major sites in design patterns and then giving you tastes of other areas so that you can plan your own journey into patterns.

Feedback

Design patterns are a work in progress, a conversation amongst practitioners who discover best practices, who discover fundamental principles in object orientation.

We covet your feedback on this book:

- What did we do well or poorly?
- Are there errors that need to be corrected?
- Was there something that was confusingly written?

Please visit us at the Web site for this book. The URL is *http://www.netobjectives.com/dpexplained*. At this site, you will find a form that you can use to send us your comments and questions. You will also find our latest research.

Acknowledgments

Almost every preface ends with a list of acknowledgments of those who helped in the development of the book. We never fully appreciated how true this was until doing a book of our own. Such an effort is truly a work of a community. The list of people to whom we are in debt is long. The following people are especially significant to us:

- Debbie Lafferty from Addison-Wesley, who never grew tired of encouraging us and keeping us on track.
- Scott Bain, our colleague who patiently reviewed this work and gave us insights.
- And especially Leigh and Jill, our patient wives, who put up with us and encouraged us in our dream of this book.

Special thanks from Alan:

- Several of my students early on had an impact they probably never knew. Many times during my courses I hesitated to project new ideas, feeling I should stick with the tried and true. However, their enthusiasm in my new concepts when I first started my courses encouraged me to project more and more of my own ideas into the curriculum I was putting together. Thanks to Lance Young, Peter Shirley, John Terrell, and Karen Allen. They serve as a constant reminder to me how encouragement can go a long way.
- Thanks to John Vlissides for his thoughtful comments and tough questions.

Special thanks from Jim:

- Dr. Karen Gardner, a mentor and wise teacher in patterns of human thought.

- Dr. Marel Norwood and Arthur Murphy, my initial collaborators in KADS and pattern-based analysis.
- Brad VanBeek who gave me the space to grow in this discipline.
- Alex Sidey who coached me in the discipline and mysteries of technical writing.