ch05.fm Page 71 Friday, June 8, 2001 11:59 AM

CHAPTER 5

An Introduction to Design Patterns

Overview

In this chapter

This chapter introduces the concept of design patterns.

In this chapter,

- I discuss the origins of design patterns in architecture and how they apply in the discipline of software design.
- I discuss the motivations for studying design patterns.

Design patterns and object-oriented design reinforce each other

Design patterns are part of the cutting edge of object-oriented technology. Object-oriented analysis tools, books, and seminars are incorporating design patterns. Study groups on design patterns abound. It is often suggested that people learn design patterns only after they have mastered basic object-oriented skills. I have found that the opposite is true: learning design patterns early in the learning of object-oriented skills greatly helps to improve understanding of object-oriented analysis and design.

Throughout the rest of the book, I will discuss not only design patterns, but also how they reveal and reinforce good object-oriented principles. I hope to improve both your understanding of these principles and illustrate why the design patterns being discussed here represent good designs.

Give this a chance

Some of this material may seem abstract or philosophical. But give it a chance! This chapter lays the foundation for your understanding of design patterns. Understanding this material will enhance your ability to understand and work with new patterns.

I have taken many of my ideas from Christopher Alexander's *The Timeless Way of Building.*[1] I will discuss these ideas throughout this book.

Design Patterns Arose from Architecture and Anthropology

Is quality objective?

Years ago, an architect named Christopher Alexander asked himself, "Is quality objective?" Is beauty truly in the eye of the beholder or would people agree that some things are beautiful and some are not? Now, the particular form of beauty that Alexander was interested in was one of architectural quality: what makes us know when an architectural design is good? For example, if a person were going to design an entryway for a house, how would he or she know that the design was good? Can we know good design? Is there an objective basis for such a judgment? A basis for describing our common consensus?

Alexander postulates that there is such an objective basis within architectural systems. The judgment that a building is beautiful is not simply a matter of taste. We can describe beauty through an objective basis that can be measured.

The discipline of cultural anthropology discovered the same thing. That body of work suggests that within a culture, individuals will agree to a large extent on what is considered to be a good design, what is beautiful. Cultures make judgments on good design that transcend individual beliefs. I believe that there are transcending patterns that serve as objective bases for judging design. A major branch of cultural anthropology looks for such patterns to describe the behaviors and values of a culture.[2]

1. Alexander, C., Ishikawa, S., Silverstein, M., *The Timeless Way of Building*, New York: Oxford University Press, 1979.
2. The anthropologist Ruth Benedict is a pioneer in pattern-based analysis of cultures. For examples, see Benedict, R., *The Chrysanthemum and the Sword*, Boston: Houghton Mifflin, 1946.

The proposition behind design patterns is that the quality of software systems can also be measured objectively.

How do you get good quality repeatedly?

If you accept the idea that it is possible to recognize and describe a good quality design, then how do you go about creating one? I can imagine Alexander asking himself,

> *What is present in a good quality design that is not present in a poor quality design?*

and

> *What is present in a poor quality design that is not present in a good quality design?*

These questions spring from Alexander's belief that if quality in design is objective, then we should be able to identify what makes designs good and what makes designs bad.

Look for the commonalties

Alexander studied this problem by making many observations of buildings, towns, streets, and virtually every other aspect of living spaces that human beings have built for themselves. He discovered that, for a particular architectural creation, good constructs had things in common with each other.

. . . especially commonality in the features of the problem to be solved

Architectural structures differ from each other, even if they are of the same type. Yet even though they are different, they can still be of high quality.

For example, two porches may appear structurally different and yet both may still be considered high quality. They might be solving different problems for different houses. One porch may be a transition from the walkway to the front door. Another porch might be a place for shade on a hot day. Or two porches might solve a common problem (transition) in different ways.

Alexander understood this. He knew that structures couldn't be separated from the problem they are trying to solve. Therefore, in his quest to identify and describe the consistency of quality in design, Alexander realized that he had to look at different structures that were designed to solve the same problem. For example, Figure 5-1 illustrates two solutions to the problem of demarking an entryway.

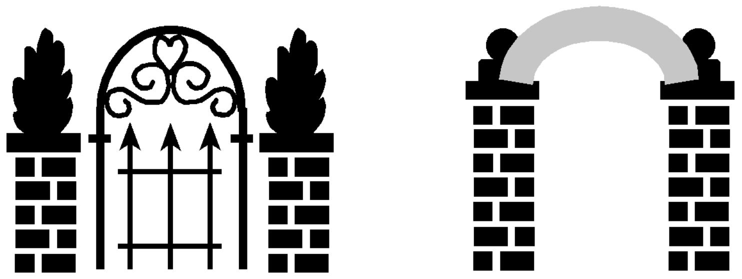


Figure 5-1 Structures may look different but still solve a common problem.

This led to the concept of a pattern

Alexander discovered that by narrowing his focus in this way—by looking at structures that solve similar problems—he could discern similarities between designs that were high quality. He called these similarities, *patterns*.

He defined a pattern as "a solution to a problem in a context."

> Each pattern describes a problem which occurs over and over again in our environment and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.[3]

3. Alexander, C., Ishikawa, S., Silverstein, M., *A Pattern Language,* New York: Oxford University Press, 1977, p. x.

Let's review some of Alexander's work to illustrate this. In Table 5-1 I will present an excerpt from his *The Timeless Way of Building,*[4] an excellent book that presents the philosophy of patterns succinctly.

Table 5-1 Excerpt from *The Timeless Way of Building*

Alexander Says . . .	My Comments . . .
In the same way, a courtyard, which is properly formed, helps people come to life in it.	A pattern always has a name and has a purpose. Here, the pattern's name is Courtyard and its purpose is to help people to come to life in it.
Consider the forces at work in a courtyard. Most fundamental of all, people seek some kind of private outdoor space, where they can sit under the sky, see the stars, enjoy the sun, perhaps plant flowers. This is obvious.	Although it might be obvious sometimes, it is important to state explicitly the problem being solved, which is the reason for having the pattern in the first place. This is what Alexander does here for Courtyard.
But there are more subtle forces too. For instance, when a courtyard is too tightly enclosed, has no view out, people feel uncomfortable, and tend to stay away . . . they need to see out into some larger and more distant space.	He points out a difficulty with the simplified solution and then gives us a way to solve the problem that he has just pointed out.
Or again, people are creatures of habit. If they pass in and out of the courtyard, every day, in the course of their normal lives, the courtyard becomes familiar, a natural place to go . . . and it is used.	Familiarity sometimes keeps us from seeing the obvious. The value of a pattern is that those with less experience can take advantage of what others have learned before them: both what must be included to have a good design, and what must be avoided to keep from a poor design.
But a courtyard with only one way in, a place you only go when you "want" to go there, is an unfamiliar place, tends to stay unused . . . people go more often to places which are familiar.	

(continued)

4. Alexander, C., Ishikawa, S., Silverstein, M., *The Timeless Way of Building*, New York: Oxford University Press, 1977.

Table 5-1 Excerpt from *The Timeless Way of Building* (continued)

Alexander Says . . .	My Comments . . .
Or again, there is a certain abruptness about suddenly stepping out, from the inside, directly to the outside . . . it is subtle, but enough to inhibit you.	
If there is a transitional space—a porch or a veranda, under cover, but open to the air—this is psychologically half way between indoors and outdoors, and makes it much easier, more simple, to take each of the smaller steps that brings you out into the courtyard . . .	He proposes a solution to a possibly overlooked challenge to building a great courtyard.
When a courtyard has a view out to a larger space, has crossing paths from different rooms, and has a veranda or a porch, these forces can resolve themselves. The view out makes it comfortable, the crossing paths help generate a sense of habit there, the porch makes it easier to go out more often . . . and gradually the courtyard becomes a pleasant customary place to be.	Alexander is telling us how to build a great courtyard . . . . . . and then tells us why it is great.

The four components required of every pattern description

To review, Alexander says that a description of a pattern involves four items:

- The name of the pattern
- The purpose of the pattern, the problem it solves
- How we could accomplish this
- The constraints and forces we have to consider in order to accomplish it

Alexander postulated that patterns can solve virtually every architectural problem that one will encounter. He further postulated that patterns could be used together to solve complex architectural problems.

Patterns exist for almost any design problem

How patterns work together will be discussed later in this book. For now, I want to focus on his claim that patterns are useful to solve specialized problems.

. . . and may be combined to solve complex problems

Moving from Architectural to Software Design Patterns

What does all of this architectural stuff have to do with us software developers?

Well, in the early 1990s some smart developers happened upon Alexander's work in patterns. They wondered if what was true for architectural patterns would also be true for software design.[5]

Adapting Alexander for software

- Were there problems in software that occur over and over again that could be solved in somewhat the same manner?
- Was it possible to design software in terms of patterns, creating specific solutions based on these patterns only after the patterns had been identified?

The group felt the answer to both of these questions was "unequivocally yes." The next step was to identify several patterns and develop standards for cataloging new ones.

5. The ESPRIT consortium in Europe was doing similar work in the 1980s. ESPRIT's Project 1098 and Project 5248 developed a pattern-based design methodology called Knowledge Analysis and Design Support (KADS) that was focused on patterns for creating expert systems. Karen Gardner extended the KADS analysis patterns to object orientation. See Gardner, K., *Cognitive Patterns: Problem-Solving Frameworks for Object Technology,* New York: Cambridge University Press, 1998.

The Gang of Four did the early work on Design Patterns

Although many people were working on design patterns in the early 1990s, the book that had the greatest influence on this fledging community was *Design Patterns: Elements of Reusable Object-Oriented Software*[6] by Gamma, Helm, Johnson, and Vlissides. In recognition of their important work, these four authors are commonly and affectionately known as the Gang of Four.

This book served several purposes:

- It applied the idea of design patterns to software design.
- It described a structure within which to catalog and describe design patterns.
- It cataloged 23 such patterns.
- It postulated object-oriented strategies and approaches based on these design patterns.

It is important to realize that the authors did not create the patterns described in the book. Rather, the authors identified these patterns as already existing within the software community, patterns that reflected what had been learned about high-quality designs for specific problems (note the similarity to Alexander's work).

Today, there are several different forms for describing design patterns. Since this is not a book about writing design patterns, I will not offer an opinion on the best structure for describing patterns; however, the following items listed in Table 5-2 need to be included in any description.

For each pattern that I present in this book, I present a one-page summary of the key features that describes that pattern.

6. Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading, Mass.: Addison-Wesley, 1995.

Table 5-2 Key Features of Patterns

Item	Description
Name	All patterns have a unique name that identifies them.
Intent	The purpose of the pattern.
Problem	The problem that the pattern is trying to solve.
Solution	How the pattern provides a solution to the problem in the context in which it shows up.
Participants and Collaborators	The entities involved in the pattern.
Consequences	The consequences of using the pattern. Investigates the forces at play in the pattern.
Implementation	How the pattern can be implemented. *Note*: Implementations are just concrete manifestations of the pattern and should not be construed as the pattern itself.
GoF Reference	Where to look in the Gang of Four text to get more information.

Consequences/Forces

The term *consequences* is used in design patterns and is often misunderstood. In everyday usage, consequences usually carries a negative connotation. (You never hear someone say, "I won the lottery! As a *consequence*, I now do not have to go to work!") Within the design pattern community, on the other hand, consequences simply refers to cause and effect. That is, if you implement this pattern in such-and-such a way, this is how it will affect and be affected by the forces present.

Why Study Design Patterns?

Design patterns help with reuse and communication

Now that you have an idea about what design patterns are, you may still be wondering, "Why study them?" There are several reasons that are obvious and some that are not so obvious.

The most commonly stated reasons for studying patterns are because patterns allow us to:

- *Reuse solutions*—By reusing already established designs, I get a head start on my problems and avoid *gotchas*. I get the benefit of learning from the experience of others. I do not have to reinvent solutions for commonly recurring problems.
- *Establish common terminology*—Communication and teamwork require a common base of vocabulary and a common viewpoint of the problem. Design patterns provide a common point of reference during the analysis and design phase of a project.

Design patterns give a higher perspective on analysis and design

However, there is a third reason to study design patterns:

> Patterns give you a higher-level perspective on the problem and on the process of design and object orientation. This frees you from the tyranny of dealing with the details too early.

By the end of this book, I hope you will agree that this is one of the greatest reasons to study design patterns. It will shift your mindset and make you a more powerful analyst.

To illustrate this advantage, I want to relate a conversation between two carpenters about how to build the drawers for some cabinets.[7]

7. This section is inspired by a talk given by Ralph Johnson and is adapted by the authors.

Example of the tyranny of details: carpenters making a set of drawers

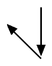
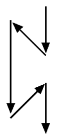
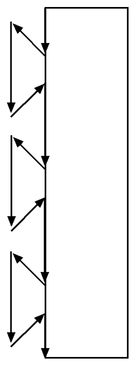
Consider two carpenters discussing how to build the drawers for some cabinets.

> **Carpenter 1:** How do you think we should build these drawers?
>
> **Carpenter 2:** Well, I think we should make the joint by cutting straight down into the wood, and then cut back up 45 degrees, and then going straight back down, and then back up the other way 45 degrees, and then going straight back down, and then . . .

Now, your job is to figure out what they are talking about!

The details may confuse the solution

Isn't that a confusing description? What is Carpenter 2 prescribing? The details certainly get in the way! Let's try to draw out his description.

Carpenter 2 Says . . .	Which Looks Like . . .
"Well, I think we should make the joint by cutting straight down into the wood, and then cut back up 45 degrees . . ."	
". . . then going straight back down, and then back up the other way 45 degrees, and then going straight back down, and then . . ."	
". . . until you end up with a *dovetail joint*. That is what I was describing!"	

This sounds like so many code reviews: details, details, details

Doesn't this sound like code reviews you have heard? The one where the programmer describes the code in terms such as,

> And then, I use a WHILE LOOP here to do . . . followed by a series of IF statements to do . . . and here I use a SWITCH to handle . . .

You get a description of the details of the code, but you have no idea *what the program is doing and why it is doing it!*

Carpenters do not really talk at that level of detail

Of course, no self-respecting carpenter would talk like this. What would really happen is something like:

> **Carpenter 1:** Should we use a dovetail joint or a miter joint?

Already we see a qualitative difference. The carpenters are discussing differences in the quality of solutions to a problem; their discussion is at a higher level, a more abstract level. They avoid getting bogged down in the details of a particular solution.

When the carpenter speaks of a miter joint, he or she has the following characteristics of the solution in mind:

- *It is a simpler solution*—A miter joint is a simple joint to make. You cut the edges of the joining pieces at 45 degrees, abut them, and then nail or glue them together (see Figure 5-2).
- *It is lightweight*—A miter joint is weaker than a dovetail. It cannot hold together under great stress.
- *It is inconspicuous*—The miter joint's single cut is much less noticeable than the dovetail joint's multiple cuts.

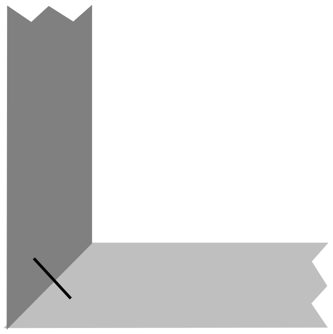


Figure 5-2 A miter joint

When the carpenter speaks of a dovetail joint (which we described how to make on page 81), he or she has other characteristics of the solution in mind. These characteristics may not be obvious to a layman, but would be clearly understood by any carpenter.

- *It is a more complex solution*—It is more involved to make a dovetail joint. Thus, it is more expensive.
- *It is impervious to temperature and humidity*—As these change, the wood expands or contracts. However, the dovetail joint will remain solid.
- *It is independent of the fastening system*—In fact, dovetail joints do not even depend upon glue to work.
- *It is a more aesthetically pleasing joint*—It is beautiful to look at when made well.

In other words, the dovetail joint is a strong, dependable, beautiful joint that is complex (and therefore expensive) to make.

So, when Carpenter 1 asked,

There is a meta-level conversation going on

> Should we use a dovetail joint or a miter joint?

the real question that was being asked was,

> Should we use a joint that is expensive to make but is both beautiful and durable, or should we just make a quick and dirty joint that will last at least as long until the check clears?

We might say the carpenters' discussion really occurs at two levels: the surface level of their words, and the *real* conversation, which occurs at a higher level (a *meta-level*) that is hidden from the layman and which is much richer in meaning. This higher level is the level of "carpenter patterns" and reflects the real design issues for the carpenters.

In the first case, Carpenter 2 obscures the real issues by discussing the details of the implementations of the joints. In the second case, Carpenter 1 wants to decide which joint to use based on costs and quality of the joint.

Who is more efficient? Who would you rather work with?

Patterns help us see the forest and the trees

This is one thing I mean when I say that patterns can help raise the level of your thinking. You will learn later in the book that when you raise your level of thinking like this, new design methods become available. This is where the real power of patterns lies.

Other Advantages to Studying Design Patterns

Improve team communications and individual learning

My experience with development groups working with design patterns is that design patterns helped both individual learning and team development. This occurred because the more junior team members saw that the senior developers who knew design patterns had something of value and these junior members wanted it. This provided motivation for them to learn some of these powerful concepts.

Improved modifiability of code

Most design patterns also make software more modifiable. The reason for this is that they are time-tested solutions. Therefore, they have evolved into structures that can handle change more readily than what often first comes to mind as a solution.

Design patterns illustrate basic object-oriented principles

Design patterns, when they are taught properly, can be used to greatly increase the understanding of basic object-oriented design principles. I have seen this countless times in the introductory object-oriented courses I teach. In those classes, I start with a brief introduction to the object-oriented paradigm. I then proceed to teach design patterns, using them to illustrate the basic object-oriented concepts (encapsulation, inheritance, and polymorphism). By the end of the three-day course, although we've been talking mostly about patterns, these concepts—which were just introduced to many of the participants—feel like they are old friends.

Adoption of improved strategies—even when patterns aren't present

The Gang of Four suggests a few strategies for creating good object-oriented designs. In particular, they suggest the following:

- Design to interfaces.
- Favor composition over inheritance.
- Find what varies and encapsulate it.

These strategies were employed in most of the design patterns discussed in this book. Even if you do not learn a lot of design patterns, studying a few should enable you to learn why these strategies are useful. With that understanding comes the ability to apply them to your own design problems even if you do not use design patterns directly.

Learn alternatives to large inheritance hierarchies

Another advantage is that design patterns allow you or your team to create designs for complex problems that do not require large inheritance hierarchies. Again, even if design patterns are not used directly, avoiding large inheritance hierarchies will result in improved designs.

Summary

In this chapter

In this chapter, I described what design patterns are. Christopher Alexander says "patterns are solutions to a problem in a context." They are more than a kind of template to solve one's problems. They are a way of describing the motivations by including both what we want to have happen along with the problems that are plaguing us.

I looked at reasons for studying design patterns. Such study helps to

- Reuse existing, high-quality solutions to commonly recurring problems.
- Establish common terminology to improve communications within teams.
- Shift the level of thinking to a higher perspective.
- Decide whether I have the right design, not just one that works.
- Improve individual learning and team learning.
- Improve the modifiability of code.
- Facilitate adoption of improved design alternatives, even when patterns are not used explicitly.
- Discover alternatives to large inheritance hierarchies.