

An excerpt from:

# **Pattern Oriented Design:**

Using Design Patterns From Analysis to Implementation

by

Alan Shalloway &  
James R. Trott

## Section 3: Design Patterns

### Section Overview

**In this section** This section introduces design patterns: what they are and how to use them. Four patterns pertinent to the CAD/CAM problem (Chapter 6) are described. They are learned individually and then related to our earlier problem. In learning these patterns, we emphasize the object-oriented strategies espoused by the Gang of Four in their seminal work: *Design Patterns: Elements of Reusable Object-oriented Software*.

Chapter 8 is an introduction to design pattern is. I introduce the concept of design patterns, discuss their origins in architecture and how they apply in the discipline of software design. Then, I discuss the motivations for studying design patterns.

Chapter 9 describes the Façade pattern. I explain what it is, where it is used and how it is implemented. Then I relate it to the CAD/CAM problem.

Chapter 10 describes the Object Adapter pattern. I explain what it is, where it is used and how it is implemented. I compare the Object Adapter pattern and the Façade pattern. Then, I relate the Object Adapter pattern to the CAD/CAM problem.

Chapter 11 describes the Bridge pattern. The Bridge pattern is quite a bit more complex than the previous patterns; it is also much more useful. While I could discuss these other patterns more conceptually, I go into great detail with the Bridge pattern. Then, I relate the Bridge pattern to the CAD/CAM problem.

Chapter 12 describes the Abstract Factory pattern. The Abstract Factory pattern focuses on creating families of objects. I describe this pattern and how it is used and implemented. Then, I relate it to the CAD/CAM problem.

**Objectives** At the end of this section, the reader will understand what design patterns are, why they are useful and will even know four specific patterns. They will also see how these patterns relate to our earlier CAD/CAM problem. This information, however, may not be enough to create a better design than the over reliance on inheritance already seen. However, the stage is set for using patterns in a way different than most design pattern practitioners use them.

## Chapter 8: An Introduction to Design Patterns

### Overview

**In this chapter** We introduce the concept of design patterns. I discuss their origins in architecture and how they apply in the discipline of software design. Finally, 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. Object-oriented analysts are expected to learn design patterns to improve their abilities. I have found that the opposite is also true: learning design patterns greatly helped to improve my basic understanding of object-oriented analysis and design.

Throughout the rest of the book, I will discuss not only design patterns, but 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** Perhaps 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 the material here will speed up your ability to obtain new patterns.

Some of this material is taken from Christopher Alexander's Trilogy (Alexander 1979, 1977, 1970)

### 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 was going to design an entrance-way 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?

If there was not some sort of objective basis, we would not be able to make judgments. What is regarded as good for someone might be bad for someone else.

Now, this book is not a treatise in cultural anthropology, but 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<sup>1</sup>.

**How do we get good quality repeatedly?**

If we accept that we can even say we have or do not have a good quality design, how do we go about creating them? Alexander asked himself,

“What is present in a good quality design that is not present in a poor 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 quantify what makes designs good and what makes designs bad.

**Look for the commonalities**

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 commonalities with each other.

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

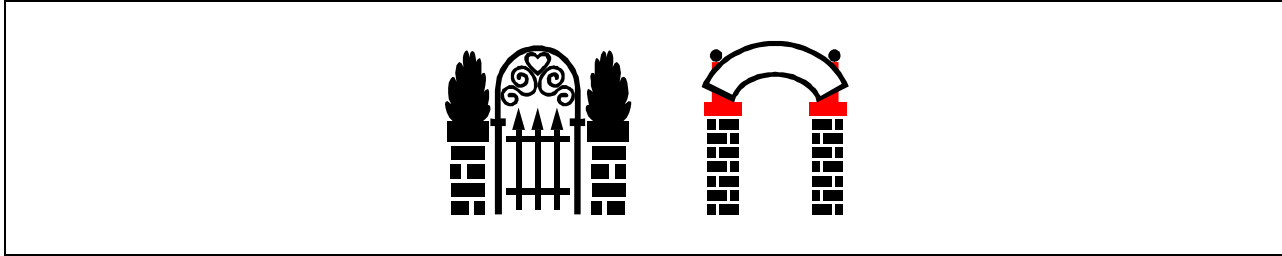
Architectural structures differ from each other, even if they are of the same type. Yet even though they are different, they can still be high quality.

For example, porches may appear different, may not have the same structure, and 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 structures that were trying to solve the same problem.

---

<sup>1</sup> The anthropologist Ruth Benedict is a pioneer in pattern-based analysis of cultures.



**Figure 8-1. 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—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.”<sup>i</sup>

*“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.”<sup>ii</sup>*

**An example pattern: the Courtyard**

Let’s read some of Alexander’s work to illustrate this. I will present an excerpt from his *The Timeless Way of Building*, an excellent book which presents the philosophy of patterns succinctly.

<b>Alexander says...</b>	<b>My comments...</b>
<p>In the same way, a courtyard which is properly formed helps people come to life in it.</p>	<p><i>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.</i></p>
<p>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.</p>	<p><i>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..</i></p>
<p>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.</p>	<p><i>He points out a difficulty with the simplified solution which we may have thought of</i></p> <p><i>...and then gives us a way to solve the problem that he has just pointed out.</i></p>

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.

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.

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...

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.

*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..*

*A solution to a possibly overlooked challenge to building a great courtyard*

*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.

**Patterns exist for almost any design problem and may be combined to solve complex problems**

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

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.

## Moving from Architectural to Software Design Patterns

### **The Hillside Group adapted Alexander for software**

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

Well, in the early 90s some experienced developers happened upon Alexander's work in patterns. They wondered if what was true for architectural patterns would also be true for software design.<sup>2</sup>

- 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.

### **The Gang of Four did the early work on Design Patterns**

Although many people were working on design patterns in the early 90s, the book that had the greatest influence on this fledging community was Design Patterns: Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson, Vlissides [Gamma 1995]. 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 defined what design patterns were within the area of 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 patterns described in the book were not created by the authors. 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. In some circles, the Gang of Four's structure is considered to be obsolete. Since

---

<sup>2</sup> The ESPRIT consortium in Europe was doing similar work in the 1980's. Project 1098 and Project 5248 developed a pattern-based design methodology called Knowledge Analysis and Design Support (KADS). See Cognitive Patterns: Problem-Solving Frameworks for Object Technology, Karen Gardner, et.al. Cambridge University Press. 1998.

this book 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 need to be included in any description.

Item	Description
<b>Name</b>	All patterns have a unique name we use to identify them with.
<b>Intent</b>	The purpose of this pattern.
<b>Problem</b>	The problem that the pattern is trying to solve.
<b>Content</b>	The context in which this problem shows up.
<b>Solution</b>	How the pattern provides a solution to this problem in the context in which it shows up.
<b>Consequences/ Forces</b>	The consequences of using this pattern. Investigates the forces at play in the pattern.
<b>Implementation</b>	How this pattern can be implemented. <u>Note:</u> Implementations are just concrete manifestations of the pattern and should not be construed as the pattern itself.

**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 means “cause and effect.” That is, if you implement this pattern in such-and-such a way, this is how it affects 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 then some that are not so obvious.

The most commonly stated reasons for studying patterns are to:

- Reuse solutions. By reusing already established designs, we get a head start on our 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 objects**

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

To give you a higher level perspective on the problem and on the process of design and object-orientation. To free you from the tyranny of dealing



with the details too early.

By the end of this book, I hope you see that this is the greatest reason 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.<sup>iii</sup>

**Example of the  
tyranny of  
details:  
Carpenters  
making a joint**

To illustrate this advantage, consider a conversation between two carpenters about how to build the drawers for some cabinets.<sup>iv</sup>



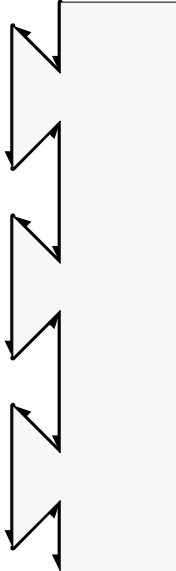
**Carpenter 1:** How do you think we should build these drawers?

**Carpenter 2:** Well, I think we should make the joint by cutting straight down into the wood, and then cutting back up 45 degrees, and then going straight back down, and then backup the other way 45 degrees, and then going straight back down and then...

Now, figure out what they are talking about doing.

**...The details can 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 Two says...	which looks like...
"Well, I think we should make the joint by cutting straight down into the wood, then cutting back up 45 degrees ..."	
"... then straight back down, then backup the other way 45 degrees, then straight back down and then ...."	
"until you end up with...a <i>dove tail joint</i> . 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 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.!*

**...but carpenters do not really talk at that level of detail**

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

**Carpenter 1:** Should we use a miter joint or a dovetail 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 mere details of a particular solution.

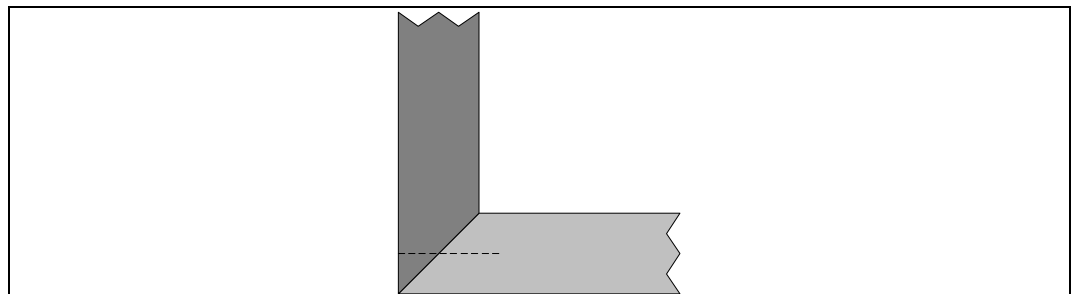
When the carpenter speaks of a miter joint, he or she has the following characteristics of the solution in mind.

- 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 8-2).
- However, it is also a weaker joint.

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

- It is a more evolved 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 does not depend on glue or other fastening systems. In fact, dovetail joints do not even depend upon glue to work.
- It is a more aesthetically pleasing joint. Beautiful to look at when made well.

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



**Figure 8-2. A miter joint**

**...there is a meta-level conversation going on.**

So, when **Carpenter 1** asked,

“Should we use a miter joint or a dovetail joint?”

the real question that was being asked was,

“Should we use a joint that is expensive to make but is both beautiful and

lasts a long time 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 several levels: the surface level of their words and the real conversation that is at a higher level (a “meta-level”) that is hidden but much richer. This higher level is the level of “carpenter patterns” and reflects the real design issues for the carpenters.

- **Carpenter 1** wants to decide on which joint to use based on costs and quality of the joint.
- **Carpenter 2** simply obscures the real issues by discussing the details of the implementations of the joints.

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

**Patterns help us see the forest for the trees**

This is one way I mean when I say that patterns can help raise our level of thinking. We will learn later in the book that when we raise our 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 they wanted it. This provided motivation for them to learn some of these powerful concepts.

**Improved modifiability of code**

Design patterns also provide for built in modifiability of software. The reason for this is that they are time tested solutions. They therefore 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 where we start using design patterns on the first day. By the end of the three day course, although we’ve been mostly talking about patterns, the concepts of encapsulation, polymorphism and inheritance – 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**

In Design Patterns: Elements of Reusable Object-Oriented Software, the Gang of Four suggest a few strategies for creating good object-oriented designs. In particular, they suggest the following:

- find what varies and encapsulate it
- favor composition over inheritance

These strategies occur in most of the design patterns described. 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

We have looked at 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 our motivations by including both what we want to have happen along with the problems that are plaguing us.

We looked at why we should study design patterns. Studying design patterns helps to:

- Reuse existing, high quality solutions to commonly recurring problems.
- Establish common terminology to improve communications within teams.
- Shift our level of thinking to a higher perspective.
- 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.

---

<sup>i</sup> A Pattern Language, Christopher Alexander

<sup>ii</sup> *ibid.*

<sup>iii</sup> This section is inspired by a talk given by Ralph Johnson and adapted by the authors.

<sup>iv</sup> This section is inspired by a talk given by Ralph Johnson and adapted by the authors.