Patterns of engineering

James O. Coplien
Both individuals and disciplines, as a whole, learn from experience. Patterns are a way to capture that experience which is often lost in traditional design literature. Patterns work with other patterns under human guidance to support a process for building systems that employs local adaptation and piecemeal growth. The pattern language design technique has roots in architecture and has close ties to broad engineering practices; many engineering artifacts (such as the suspension bridge and the push-pull amplifier examples given here) are typical patterns. Patterns are being widely used in object-oriented software development and have pushed into other areas of software engineering. However, they are a natural fit for a broad spectrum of current engineering design practices.

Design can be defined as the process that takes one from a problem to a workable solution. Whether your design techniques help you create the reinforcements in a large dam, improve the speed or density of a VLSI circuit, or create the layout of a new neighborhood, they all have to do with creating structures that solve problems.

Students are often given self-contained problems to solve in isolation. Most solutions to these problems are supported with explicit training and practice. Practitioners in our respective trades easily can solve those example problems because the relationship between problem and solution is close in time and space, close in cause and effect. Many of the solutions to these problems can be taught as engineering rules. Such rules are often the foundation for teaching an engineering discipline.

Systems

Most interesting design problems in the real world have to do with systems. Systems are usually more complex than those self-contained problems; they are usually larger than what can easily be comprehended within a single mind. System solutions rarely precipitate from any formal method that balances all the requirements, but come instead from experience. Solutions that are based on experience draw on contextual knowledge, tacit knowledge, customization and sometimes breaking the rules. System problems and their solutions are the foundations of practical design. These design decisions make and break systems.

If you talk to the great systems designers in your field, they may tell you that experience plays a big part in what they do. Part of experience is learning when to apply a given tool or technique that you learned in college. Another part of experience is the building of confidence to try new things that aren’t yet in the textbooks. But a big part of experience is to know how to generalize solutions at the system level and to apply them again. Many of these solutions don’t naturally follow from the engineering rules of a particular discipline: they’re things you just have to know.

Patterns

Patterns have gained popularity in programming during the past decade as a way to capture and organize important engineering solutions. Patterns are enjoying great success in computer programming, possibly because computer science is a young discipline for which sound and widely accepted engineering rules have yet to come about. But engineering disciplines are rife with good patterns, patterns that encode experience, experience that comes from sound application of properties we know from formal foundations.

What is a pattern? Briefly, a pattern is a structure gleaned from experience that contributes to the overall structure of a system. The system is built for some purpose, and each pattern balances tradeoffs that an engineer must consider to achieve that purpose.

Let’s say we’re building a highway system, or perhaps renovating a shoreline area in a large city. We need a bridge across a river, gorge or perhaps a railway right-of-way, that must cover an unusually long span. The problem is how to span the obstacle with the highway. In Box A, there is a pattern that solves that problem. (A real version of this pattern would probably have more detail.)

A SUSPENSION BRIDGE is a pattern that is part of a larger system that we call a roadway. While a bridge can be a beautiful piece of engineering construction in its own right, it must fit into the system it belongs to. This means that it must fit into the overall scheme of the roadway. More generally, the bridge must also fit into the overall scheme of the regional transportation: a bridge may not be cost-effective if traffic load is particularly light, if the travel times are not critical, and if barge service is already available in the same locale.

Pattern properties

The SUSPENSION BRIDGE pattern illustrates properties that are important to all patterns:

• A pattern is always part of some larger whole.
• A pattern always refers to other patterns that complete it. A pattern is always part of the completion of some larger pattern (here, a network of roads). It comprises smaller patterns that the designer must apply to complete it.

Here, MAIN CABLE RUN, SHORESIDE ANCHOR CABLE and DECK TRUSS are such patterns. There may be other patterns such as stiffeners that reduce the torsion of the deck, or dampeners to keep the bridge from resonating to vibrations at its natural frequency.

• The pattern both describes the thing that is to be built and the process for building it. A good pattern gives its audience enough information to build it in the real world. It isn’t a document just about theory.

• It balances tradeoffs, called forces, to provide a solution to a problem.
• Using human ingenuity, a designer can tailor that solution to a particular context. Here, the designer will need to decide how wide to make the bridge, where the bridge will meet the shore, what size of cables to use, and a thousand other things that will contribute to both the functionality and beauty of the bridge.

Process and pattern languages

The notion of process is key to patterns. A pattern is an incremental, local design step. To build a roadway system requires many local construction projects, each adapted to its environment.

The bridge is one such local adaptation. The designer applies these adaptations in an order that generally starts at the largest scale and proceeds incrementally to smaller scales in one locale at a time.
SUSPENSION BRIDGE

...a network of roadways must not only provide a good surface between two potential destinations, but should ease the crossing of natural and man-made obstructions. A bridge over an obstacle solves the simple problem of allowing traffic to rise above such an obstruction as if the obstruction weren’t there. For small spans (up to 200 feet or so) a BEAM BRIDGE can easily be constructed to circumvent the obstacle, and an ARCH BRIDGE can span up to 1000 feet.

A bridge must both be beautiful and provide a convenient path for traffic to cross an obstacle. Beauty implies that the bridge harmonizes with the natural landscape, or that it complements the landscape, and that it doesn’t obstruct the traffic on the highway, railway right-of-way, or river that it might cross. The bridge structures mustn’t create blind spots or other distractions for traffic.

A bridge can be built of segments, each one of which is a BEAM BRIDGE or an ARCH BRIDGE that covers part of the total span. But that requires a footing at the end of each segment. If the bridge crosses a canyon, one of the footings would have to rise from the canyon floor perhaps several thousand feet below; if it crosses a river, we would be faced with the difficulty of building a firm footing on the muddy riverbed. Good bridge design depends on redirecting its tension into firm, dry land, and a mid-stream support is not an ideal engineering solution. Furthermore, fillings in the middle of a river form a navigation hazard for shipping and passenger vessels on the water, and they may break up a serene waterside vista or even damage native water habitat. Footings might block views for highway traffic, or offer hard walls into which traffic might crash where only soft shoulders formerly existed, presenting a safety hazard.

Therefore:

Suspend the deck from cables that are supported from above. A thick MAIN CABLE RUN can be strung between two towers, and cable suspenders connecting the road truss with the cable band can support the road truss. The forces on the towers can be balanced with SHORESIDE ANCHOR CABLE. Keep the roadway from swaying with a DECK TRUSS.

Suspension bridges can span long distances, obviating the need for artificial islands to support the bridge in the middle of a river. It provides a cost-effective and beautiful span. Their grandeur often provides for beautiful vistas over the river from mid-span. Take advantage of their location over obstacles of natural beauty with patterns PEDESTRIAN BRIDGE WALKWAY and BRIDGE-END PARK.

In building a highway, the time comes at some point to build the bridge; in building the bridge, the time comes at some point to construct the MAIN CABLE RUN and SHORESIDE ANCHOR CABLE, after which the deck can be laid and the DECK TRUSS constructed.

These patterns build on each other in a common-sense sequence that honors their structural dependency. The sequence forms a grammar of sorts, and a collection of patterns tied together by such a grammar is called a pattern language. A pattern never stands apart from other patterns that complete it. The SUSPENSION BRIDGE pattern may be part of a larger pattern that creates an effective roadway or even a harmonious transportation system. People will curse the most beautiful bridge that is accessible only via poorly engineered roadways. No single pattern adds much value in its own right; it is the emergent properties that come from combining patterns that underlie great systems.

Design is the process of transforming a system from one context to another to increase its wholeness: its utility, beauty, longevity, or some other valued attribute or property. Patterns are a way to carry out such transformations through local adaptations: many small, local changes combine to generate system-level structure and phenomena.

If the system as a whole doesn’t exhibit the qualities we seek, the quality of the one component is of little consequence and, in fact, may not be worth the effort. This holds true for the push-pull amplifier example given in Box B: it must fit with a great tuner or CD player and preamp and a good set of speakers, as well as a good power supply, to create beauty in sound. No single artifact of design has much beauty in its own right. Beauty emerges as the result of many small, local, individual acts of design—often at the hands of dozens, hundreds or thousands of individuals.

Roots in architecture—and beyond

Christopher Alexander, an urban planner and building architect, popularized the idea of design patterns in the 1970s. Much of what he says about patterns relates to the physical world: for example, he uses the word “forces” in a way we easily can tie to buildings and bridges. Most of his patterns exhibit an approximate symmetry. This bias for the physical world cause many people to question the relevance of patterns to the so-called “abstract” fields such as software.

But, in fact, the idea of patterns long predates Alexander. Also, it has not always applied to problems with corporeal structure. The concept is employed in basic texts on culture, such as Kroeber’s text (1948) that describes the basic elements of culture as patterns that solve problems in communities of people. And even Alexander talks about forces that go beyond physics: psychological forces of beauty and of human comfort dominate much of his work.

Patterns describe ideas deep within us. In fact, at the core of a pattern we often find a metaphor that transcends disciplines. We can generalize this concept of structure to any solution that we formulate as a structural design. Intrinsic to the concept of structure are the notions of symmetry, composition and fit—whether in a physical or ideo-
logical world. That fact leads us directly into many other engineering fields. Consider the well-known push-pull amplifier known to engineers, stereo enthusiasts and amateur radio operators alike. It’s a pattern, too, as demonstrated in Box B.

**Pattern languages’ relevance to engineering**

Carefully constructed pattern languages are powerful, just as human language is powerful. We can combine words in many useful ways under the rules of word composition in a language. A few words can give rise to millions of sentences. Each sentence gives context and fine shades of meaning to the words it contains: patterns themselves take on new shades of meaning in the specific systems where they are applied. Patterns work only when guided by human feedback and sensibility; that is the only way to achieve “fit” at a human scale.

Engineering has always drawn its ideas from the sciences, but it also has never been so proud as to exclude insights that come from experience. Individual patterns might build on proven engineering principles that history has proven valuable. But in pattern-guided design, as in most engineering problem solving, human ingenuity is always the key ingredient. Engineering ingenuity (which is perhaps redundant—these two words have the same etymological roots) not only depends on the human element, but also directly addresses the human element more than most disciplines do.

Engineers should be familiar with words like **symmetry**: such a concept would be lost neither on the designer of a suspension bridge nor of a push-pull amplifier. But engineering symmetry is rarely exact because it always accommodates the “fit” to its environment. The push-pull amplifier is full of symmetry, but what makes it work is the asymmetry in the voltage divider and inverter at the front end of the two active devices. A DECK TRUSS, an imperfect reflection of the suspension structure, makes a suspension bridge strong. Much of modern physics is concerned with the lack of perfect symmetry in fundamental subatomic processes. This same symmetry is fundamental to patterns and to a process of design rooted in local adaptation.

**Box B**

**PUSH-PULL AMPLIFIER**

...we want high-power output from an amplifier. Unfortunately, a single transistor or triode may not deliver all the power we want.

* * *

**The higher power rating we choose, the more distortion we find.** High-power tubes commonly have transfer functions that are nonlinear, usually in that they under-amplify low voltages. But they may also reduce their output near the top of their response.

These responses sometimes have a degree of symmetry to them: that is, they have lower slopes near both the bottom and top of their effective input ranges than they do in the middle of the range. We could operate these transistors or tubes only over the linear portion of their response curve, but that greatly limits the dynamic range of the amplifier. A wide dynamic range is important to realistic music listening, particularly for orchestral music. Yet taking advantage of the full range guarantees distortion at both low and high volumes. A great music system should exhibit a faithful response for both delicate quiet music and for the loud swells—both key elements of a moving musical experience.

Therefore:

**Balance two devices together so one is pushing while one is pulling.** Split the input signal into a balanced pair of signals, one out of phase with the other, and let each half of the input signal drive its own active device. Sum the outputs of the active devices by “stacking” their output, e.g., through an output transformer. Use a center tap on the transformer to supply the plate voltage to the tubes (or the Vcc to the transistor).

* * *

The result is that you can offer twice the power to the output transformer than if a single tube or transistor was used. If each active device is more linear at one end of its response than at the other, this push-pull effect will yield a more linear overall response than a single device.

Think of a water pump for a well: a single transistor is like a one-handled pump. If you put two transistors (or tubes) “back to back,” one can be “pushing” while the other is “pulling” (Nachbaur 2002). As the input voltage rises, one tube increases its current output while the other decreases its current output. As the input voltage falls, they swap roles.

At the extremes of their response ranges, one of the devices will usually be operating in a linear range of its response curve. If one of the tubes or transistors has a nonlinear response at an extreme of its range, the other one will tend to lessen the impact of the nonlinearity. That leads to more linear, distortion-free response at both whisper and fortissimo levels in music with wide dynamics.

**Software patterns**

Software engineering—which is not a new engineering discipline by most measures (Coplien 2001)—has too often sought formal and scientific grounding for phenomena where the formalisms have no relationship to any value system. That is, there is often no way to see how the formalisms relate to anything that anyone cares about. Software pattern practitioners have turned to patterns as a reaction against this unsatisfying aspect of the academic side of computer science that too often ignores or even discounts the role of the everyday programmer (Coplien 1996). Most patterns are common sense but, as all designers know, common sense is often so uncommon.

Thus, the software pattern commu-
ty is building a body of literature of such “common sense” techniques that, in the academic value system of publication, is a radical idea. The idea seems to be working. Why? What’s common sense to one person is an innovation to another. Most engineers achieved some familiarity with the workings of engineering things by taking apart their electric trains as a child. However, we are not yet at the point of software education where most entering students have reverse engineered the firmware in their video game systems.

That time might come. Still, most effective software developers repeatedly face system problems—perhaps even more than engineers do. Most modern practice handles local complexities such as chip power and circuit speed on the electrical engineering side, while pushing system complexity issues back to the software people. That makes pattern languages more crucial to the software practitioner even than to the engineer. This fact could be one of the reasons why patterns have gained more of an initial foothold in software than in engineering.

**PloPs and other sources**

The most common path for broad pattern publication is through one of the several software pattern conferences called PLoPs (Pattern Languages of Programs). These conferences are unlike traditional software conferences in that their purpose is to refine submitted works for later publication. In the early days of software patterns, most of these patterns were published in a book series called the PLoPD Series; four such volumes have been published (see, for example, Harrison (2000) and a fifth is in preparation). Patterns at contemporary PLoPs find their way into a wide variety of publication venues on the web, in the periodical literature and in books. There are also regionally based pattern communities that meet monthly to review and refine patterns, to read patterns, or to discuss patterns and design.

The software field is largely bereft of any true engineering, yet some patterns capture experience that is decades old. Many of the most mature software patterns come from fields like telecommunication, which has a century-old history whose structures persist in contemporary telecom software. Software organizational patterns similarly build both on a long history of experience with organizational structure and with the broad concerns of effective organizations in modern business climates.

In the field of architecture and urban design, Alexander continues to evolve pattern theory while developing a web-based mechanism that will allow such patterns to more broadly be disseminated. The most recent book from Alexander, *Nature of Order* (2003), relates insights that Alexander and his colleagues have developed since the publishing of *The Timeless Way of Building* in 1979.

In the software community, everyday software practitioners gather patterns and refine them in local workshops. Some pattern collections are used local to corporations. Some topically focused pattern collections enjoy application across corporate boundaries. Active topical areas include pedagogical patterns, organizational patterns, telecommunication patterns (Rising 2001) and many more.

There is no central clearing house for patterns: so how does one find patterns in a given area? The Pattern Almanac, edited by Linda Rising (2001), is a useful index for tracking down published patterns. There are tentative plans to migrate this index to an online web version.

The software community has learned much about writing good pattern languages in the past ten years, but at this writing there are few pattern languages in the software literature. The software community has started with small collections of patterns and has been working to integrate them into pattern languages.

To learn more about patterns:

- Try your hand at writing a pattern. Review it with your friends and improve it. Join or form a local pattern group to read, write, and critique patterns in your discipline. It doesn’t have to be software—take a risk and start something new!
- Read the Software Patterns Management Briefing, available as a PDF at <http://users.rcn.com/jcoplien/Patterns/WhitePaper/>
- Get plugged into the other pattern community resources including books, conferences, and people at the industry pattern web site <http://hillside.net/patterns/>.

**Read more about it**