Problems Before Patterns: A Different Look at Christopher Alexander and Pattern Languages

Molly Wright Steenson
Princeton University School of Architecture | molly@girlwonder.com

Interaction and system designers alike gravitate to the idea of pattern languages. The notion of patterns comes from the work of architect Christopher Alexander, who with his associates Sara Ishikawa and Murray Silverstein of the Center for Environmental Structure, published *A Pattern Language* in 1977. The book defines a set of fundamentals for building and planning urban and architectural projects that can be used by non-expert designers. "Each pattern describes a problem which occurs over and over again in our environment," wrote Alexander and his coauthors, "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 [1]." While the authors addressed architectural and urban problems—in effect, spatial problems—the approach offered (and continues to offer) ready parallels with the design problems faced by all designers.

Alexander has long influenced interaction and software designers. Pattern languages have made numerous appearances in previous issues of *interactions*, explored by Aaron Marcus, Shelley Evenson, Hugh Dubberly, and Rick Robinson, to name a few [2,3]. Alan Cooper's approach to design was strongly inspired by pattern languages. Kent Beck and Ward Cunningham not only cite Alexander's influence on the development of object-oriented programming languages at Xerox PARC in the early 1990s, but also on extreme programming during the later part of the decade [4, 5]. And Erin Malone and Christian Crumlish are currently writing a book about patterns for social software, titled *Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience.*

For designers of many disciplines, pattern languages are attractive because they offer a way to identify the core design problem and because they seek replicable rules and building blocks in their solutions. Alexander and his colleagues even envisioned the kinds of sharing mechanisms central to contemporary pattern libraries. As early as the mid-1960s, they thought that patterns should be shared via an ever-growing, open database of design problems and solutions [6].

While pattern literature often focuses on patterns, there's an even greater focus on the reproducible solution to a design problem. As patterns move to online reference models, they concentrate less on outlining the problem and the context, and more on the object, component, or interface solution. Where this might help someone find a quick reference, it can be done at the detriment of a problem statement that offers expertise and context. John Vlissides, one of the four authors of *Design Patterns: Elements of Reusable Object-Oriented Software*, noted in a 1997 article that one of the primary offerings of patterns as a whole is their usefulness in addressing recurring problems. "In short," he wrote, "patterns are primarily food for the brain, not fodder for a tool [7]." Skimping on defining the problem makes it more difficult to critique, share, or build upon the learnings of the pattern.

The *Pattern Manual* deals with the issue of the design problem. This little-known text by Alexander and his colleagues defined the landscape of the design problem in 1967—a decade earlier than the publi-

[1] Alexander, C., S. Ishikawa, and M. Silverstein. *A Pattern Language: Towns, Buildings, Construction.* New York: Oxford University Press, 1977.

[2] Aaron, M. "Patterns Within Patterns." *interactions* 11, no. 2 (2004): 28-34.

[3] Dubberly, H., S. Evenson, and R. Robinson." The Analysis-Synthesis Bridge Model." *interactions* 15 no. 2 (2008): 57-61.

[4] Beck, K. "Embracing Change with Extreme Programming." *Computer* 32, no. 10 (1999): 70-77.

[5] Beck, Kent. <http://c2.com/ppr/about/author/kent.html>.

[6] Alexander, C., S. Ishikawa, and M. Silverstein. *Pattern Manual*, Berkeley, 1967.

[7] Vlissides, J."Patterns: The Top Ten Misconceptions." *Object Magazine* 7 no.1 (1997): 30-33. <http://www.research.ibm.com/designpatterns/pubs/top10misc.html.> Accessed 24 November 2008

Photograph of Christopher Alexander by Richard Morgenstein

cation of the more familiar *A Pattern Language*. The methodology in the manual specifies a structure for setting up design problems in order to find generalities, particularities, and eventual solutions. The authors considered it a "minimal and natural" format: the what, where, and how of a situation; in other words, the problem, the context, and the resulting pattern [6]. Shifting the focus to the definition of the design problem and not just its resulting pattern helps to ensure the pattern properly addresses the situation, particularly in complex environments.

Alexander long maintained an interest in defining a design methodology in the face of complexity. *Notes on the Synthesis of Form*, originally published in 1964, more than 20 years before *A Pattern Language*, outlines the difficulty of designing for a series of intermeshing, interacting systems, even when the final designed object itself might not look complicated. "In spite of their superficial simplicity," Alexander wrote, "even these problems have a background of needs and activities which is becoming too complex to grasp intuitively;" needs and activities that sit within a growing ecosystem of other pressures, whether social, cultural, or informational [8]. In this setting, Alexander found no place for the secret, intuitive processes traditionally claimed by many designers, ones which did not take the intricacies of their contexts into consideration. Instead, he advocated a logical, objective approach to design, in which form fit context by addressing a set of design requirements. With these requirements, Alexander expanded the architectural notion of program (it specifically means the set of functions fulfilled by a room, space, or building). It is a program, he wrote, "because it provides directions or instructions to the designer [8]." If this sounds like engineering language, it is no surprise. Alexander developed design-requirement data sets in the early 1960s that were complex enough to necessitate an IBM 704 mainframe computer for analysis. With his colleagues at the Center for Environmental Structure, Alexander moved away from such a byzantine analysis of requirements, instead seeking a method for creating straightforward descriptions of the program—that is, the design problem—in the *Pattern Manual*.

[8] Alexander, C. *Notes on the Synthesis of Form*. Cambridge, MA: Harvard University Press, 1971.

The manual defines a grammatical structure that maps to a designer's mental model. A designer follows three steps when developing a pattern, "or, for that matter, [when he] entertains any idea about the physical environment…. He considers a problem, invents a pattern to solve the problem, and makes a mental note of the range of contexts where the pattern will solve the problem [6]." Contexts and problems are paired with each other—wherever a particular context appears, so too does its problem. The context modifies the pattern in the way that an adverb modifies a verb: It says how the pattern works and in which circumstances it is valid. The problem statement provides the reasoning behind the pattern and context. It can be much lengthier, offering an explanation of the situation, a "common-sense description of the problem, as it exists today [6]." The pattern, then, is a set of parts that relate to each other in space. Patterns can address anything from the appropriate layout for a kitchen, to freeway ramps, to designs for users of a certain income or educational level, to furniture design, to structures that hold up houses [6]. Where they can address a huge variety of problems, they themselves seek to be reductive and essential, offering only what is necessary. Where patterns might not provide the only solution to the problem, without it or an equivalent, "the problem will go unsolved [6]."

Although titled the *Pattern Manual*, its heart is the design problem statement—the most important element "from a human standpoint [6]." Problems subsume the considerations that system designers address, called "functional demands… [that] at one time or another [have] been called requirements, needs, performance standards, facts, tendencies, objectives, constraints, activities, technical data, and so forth." Yet the functional demands do not stop with what a system should do: They address a wide variety of issues surrounding the ecology of a system. "They may concern human behavior, economics, the state of technology, the political climate, whatever. No limits can be placed on the kinds of elements necessary to describe a problem properly [6]."

If that sounds vast, it is. Patterns address an astonishingly wide variety of elements that are organized in space in some manner. The *Pattern Manual* offers an expansive list that includes "all kitchens; dormitory kitchens; efficiency apartment kitchens; …all industrial sites larger than two acres; a 2x4… residential areas with 40 percent of their population under 25 and median incomes between $6,000 and $8,000; garden paths; cobblestone paths; a doorknob; any freeway; freeway exit ramps; bookshelves [6].” Any of these patterns provides a solution to a problem that exists in space, whether the demographics of a neighborhood, the kind of structure required for a house, a transportation issue, or the optimal setup of a dormitory.

As an example, the *Pattern Manual* describes the difficulty of reading house numbers from a moving vehicle. It states the context tersely and specifically: “Freestanding house on a street where cars move at speeds between 5 miles per hour and 30 miles per hour [6].” The problem statement is much longer—in this case, three pages—and sets out the series of issues the pattern will need to address, beginning with: “House numbers are very hard to see from moving cars, especially for the driver. Many signs are parallel to the road (on the house face, or garden gate), so that they can’t be seen from up the street [6].” The rest of the problem statement includes facts about house numbers and signage, references to studies on driver vision, and the limits of potential positioning of signs—in essence, the evidence for a case to support the problem. The following pattern, for instance, addresses the house-number problem:

• Two house signs, each at about 45 degrees to the street, facing up and down the street, respectively.

• If the house is one of a regular sequence of houses all using this pattern, then the sign letters are at least 6 inches high.

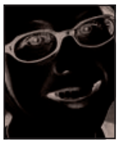
• If the house is isolated, or is one of a regular sequence of houses not using this pattern, then the sign letters are at least 12 inches high [6].

Consequently, a simple pattern that addresses angle and direction of signage and the size of letters tackles a broader design problem. It notes different use cases—sequences of similar houses versus isolated or nonsequenced houses—and offers different variables for the solutions. While a designer could simply use the pattern, the richness of the framework lies in the overall problem statement and context.

Furthermore, the goal of pattern libraries is not only to offer solutions to design problems, but also to solicit critique and invite improvement. “We want our ideas to improve under public scrutiny,” wrote Alexander’s team, “and we want our good ideas to be potentially combined with other good ideas [6].” The Center for Environmental Structure first sought to publish its patterns under the rubric of a catalog to which anyone could submit patterns using the format described in this article. An editorial board would select patterns; catalog subscribers would receive the patterns. Alexander and his colleagues thought that by 1970, patterns could be stored in a computer and offered to subscribers—a central feature to contemporary pattern libraries for games, object-oriented programming, or Web design [9].

Through their straightforward approach to describing a complex network of design considerations, Alexander, Ishikawa, and Silverstein all anticipated and inspired contemporary methods for design thinking. By seeking to provide “a natural way of expressing thoughts about the physical environment,” the authors offered a vital means to articulate the richness not only of a design solution, but its problem and its context [6]. At the same time, the earlier publication of *Pattern Manual* serves as a reminder for elements of patterns that often receive less focus. At the heart of every pattern is a design problem. When well defined, the design problem represents the designer’s collective expertise of issues, information, and problem context, making for better patterns and design solutions. In examining how the pattern language developed, we see how important the latter parts of that sentence were to Alexander and his colleagues—and to the continued evolution of design thinking in general. With straightforward language, the problem and pattern language continue to bring a systematic approach to design to the wider audience who practiced it, improved upon its elements, and continue to develop the concept today.

[9] Alexander, C. *Theory, Organization, Activities.* [Pamphlet], Berkeley, Center for Environmental Structure, 1968.



ABOUT THE AUTHOR Molly Wright Steenson is a design researcher and architectural historian who studies interactivity and responsiveness in architecture and urbanism. She is a Ph.D. candidate in architecture at Princeton University and contributing editor for *interactions*. Steenson also conducts research and develops design strategy for mobile, Web, and urban projects and blogs at Active Social Plastic (activesocialplastic.com).

DOI: 10.1145/1487632.1487637
© 2009 ACM 1072-5220/09/0300 $5.00