Pattern Languages For Interaction Design

by Victor Lombardi

RAZORFISH REPORTS

a publication of the science department | 04.24.00 | report 15

…The language, and the processes which stem from it, merely release the fundamental order which is native to us. They do not teach us, they only remind us of what we know already, and of what we shall discover time and time again, when we give up our ideas and opinions, and do exactly what emerges from ourselves.

- Christopher Alexander, The Timeless Way of Building

Introduction

In the late 1960s Christopher Alexander and his associates embarked on a search for a new method of architecture and planning, studying how people interacted with physical spaces. Several years later they published a series of books describing this new method. With philosophy and writing of soaring beauty, Alexander posits that our lives consist mainly of patterns of events, and that architecture which supports these patterns helps us feel more "alive" or "whole." He says, "A pattern language is really nothing more than a precise way of describing someone's experience of a building." This approach of focusing on people relates directly to the Razorfish philosophy of creating user experiences.

"The pattern discipline has become one of the most widely applied and important ideas of the past decade in software architecture and design." -James Coplien

Alexander recorded his observations in a format called "design patterns" which summarize the context of a problem and its solution. For example, one design pattern, "Opening to the Street," is illustrated with a photograph and begins with the essence of the pattern: "The site of action is an incentive for action. When people can see into spaces from the street their world is enlarged and made richer, there is more understanding; and there is the possibility for communication, learning." Notice his description focuses on the human experience. Next is a detailed elaboration of this scenario, including a report of his observations and other architectural references. The pattern continues with a solution: "In any public space which depends for its success on its exposure to the street, open it up, with a fully opening wall which can be thrown wide open, and if it is possible, include some part of the activity on the far side of the pedestrian path, so that it actually straddles the path, and people walk through it as they walk along the path…" He concludes the pattern with a line drawing and a list of related patterns.

When writing his patterns, Alexander carefully selected the level of abstraction. The patterns are concrete enough to use as rules of thumb with good results, and yet are sufficiently abstract to apply to countless situations, with an almost infinite variety of results.

A group of related design patterns are referred to as a "pattern language." The particular pattern language Alexander wrote spans the entire scale of architecture, from "Agricultural Valleys" to "Street Cafes" to "Alcoves." But this language is only one example; his hope is that every living society and to some extent every individual will identify their own pattern language and use it to design their environment.

Pattern Languages in Computer Science

The concept of pattern languages crossed over to the computer science field not via user interface designers but actually through the efforts of the programming community. The method of documenting invariant problems paired with their corresponding solutions was a convenient way to express repeating programming techniques. Additionally, the concept of associating these patterns in the hierarchical structure of a pattern language translated into interesting parallels for object-oriented software design. The most notable expression of design patterns in the software field is the landmark book, Design Patterns, by Gamma et al.

"A pattern language is nothing more than a precise way of describing someone's experience..." -Christopher Alexander

James Coplien of Bell Laboratories comments, "Focusing on objects had caused us to lose the system perspective. Preoccupation with design method had caused us to lose the human perspective. The curious parallels between Alexander's world of buildings and our world of software construction helped the ideas to take root and thrive in grassroots programming communities worldwide. The pattern discipline has become one of the most widely applied and important ideas of the past decade in software architecture and design."

The software community started a series of Pattern Languages of Programming conferences to further creation and use of design patterns. It was in these conferences that design patterns were first created for interaction design.

Pattern Languages for Interaction Design

There is a small, active community of interaction designers around the world cur-

rently creating design patterns and experimenting with their use. An example is Common Ground, a user interface pattern language created by Jenifer Tidwell at the Massachusetts Institute of Technology. Here is a summary of one of her patterns:
Pointer Shows Affordance
Examples:
• Finger pointer over a hyperlink, especially pictorial links
• Buttons whose borders change when you point to them

Context: The artifact contains a pointer, or "virtual fingertip" (mouse or pen point, for instance) that is the focal point for the user's interaction with the artifact…

Problem: How can the artifact indicate that an entity represents an action that the user may take?

Forces: Static affordances aren't always enough to indicate the presence of a manipulatable control, especially when space is tight or when an esthetic design is of paramount importance.

Solution: Change the affordance of the thing as the pointer moves over it. This can be done by changing the pointer to communicate what action can be performed or by changing the artifact to make it stand out perceptually…

Tidwell concludes this pattern with the resulting context, an illustration, references to related patterns in her language, and examples of well-known interface designs. This pattern contributes to a pattern language spanning all of user interface design, including patterns such as "Background Posture," "Status Display," and "Progress Indicator."

This pattern language shows consistency with Alexander's ideas in several ways: the focus on human interaction, the format that includes a problem, context, and solution, and the composition of a language describing human behavior within an environment. Additionally, the level of abstraction enables designers to use this design pattern repeatedly while never duplicating the details of implementation.

"Focusing on objects had caused us to lose the system perspective. Preoccupation with design method had caused us to lose the human perspective."
-James Coplien

Benefits

Interest in pattern languages for computer interaction design continues to grow for several reasons. First, design patterns provide an accessible format to document design knowledge on a personal, project, or organizational level, creating a complete but accessible reference source. Design patterns elegantly summarize a problem, context, examples, and solution in a format that is more rigorous than a heuristic and more accessible than a library of design books.

Second, the format of design patterns facilitates communication with other participants of a design project. Because design is a distributed social process, effective communication plays an important role.

Third, pattern languages serve as a tool to practice actual design. They move the designer to an abstract level to help escape repetition and encourage innovation. Because design patterns transcend particular implementations, they assist the designer in moving easily among platforms, devices, and media (visual, auditory, haptic, etc.). When used as a toolbox of techniques, design patterns can lower costs, in terms of time and usability testing, by helping designers form educated design choices at the beginning of a project.

The Challenge

At the Object-Oriented Programs, Systems, Languages, and Applications conference in 1996, Christopher Alexander lamented that architects still do not create in a generative way through observation of human experience. He pointed out that the spread of computers effectively increases the influence of computer professionals in a way that could prove more important to human experience than architecture. He challenged software professionals to create the "natural, genetic infrastructure of a living world which you/we are capable of creating, managing, making available, and which could then have the result that a living structure…becomes an attainable thing."

for more information

Alexander, Christopher.
The Timeless Way of Building. Oxford University Press, 1979.

Alexander, Christopher.
A Pattern Language: Towns, Building, Construction. Oxford University Press,1977.

Alexander, Christopher.
The Origins of Pattern Theory: The Future of the Theory, and the Generation of a Living World.
http://www.computer.org/software/so1999/s5071abs.htm

Erickson, Thomas.
The Interaction Design Patterns Page.
http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html

Gamma, Erich, et al.
Design Patterns: Elements of Reusable Object-Oriented Software.
Addison-Wesley, 1995.

Tidwell, Jenifer.
Common Ground: A Pattern Language for Human-Computer Interface Design.
http://www.mit.edu/~jtidwell/interaction_patterns.html