Pattern Languages

IDRC

February 7, 2018

Overview

- Pattern Languages
- Applications to computer software
- A Pattern Language for Infusion

Christopher Alexander book series

- Volume 1, *The Timeless Way of Building* (1979)
- Volume 2, *A Pattern Language* (1977)
- Volume 3, *The Oregon Experiment* (1975)

Patterns

“…we must begin by understanding that every place is given its character by certain patterns of events that keep on happening there.”

“These patterns of events are always interlocked with certain geometric patterns in the space. Indeed, as we shall see, each building and each town is ultimately made out of patterns in the space, and out of nothing else: they are the atoms and the molecules from which a building or town is made.”

Christopher Alexander, *The Timeless Way of Building*

Watching the World Go By

“Consider, for example, the pattern of events which we might call ‘watching the world go by.’

“We sit, perhaps slightly raised, on the front porch, or on some steps in a park, or on a café terrace, with a more or less protected, sheltered, partly private place behind us, looking out into a more public place, slightly raised above it, watching the world go by.

I cannot separate it from the porch where it occurs.”

Christopher Alexander, *The Timeless Way of Building*

Writing Generative Patterns

“Each pattern is a rule which describes what you have to do to generate the entity which it defines.”

“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.”

Christopher Alexander, *The Timeless Way of Building* and *A Pattern Language*

The Quality Without a Name

- Patterns that generate places with the quality without a name
- Alive
- Whole
- Comfortable
- Free
- Exact
- Egoless
- Eternal

The Quality in Software (excerpts)

“I can’t tell you what the quality is, but I can tell you some things about software that possesses it:

- Its modules and abstractions are not too big
- If I look at any small part of it, I can see what is going on
- If I look at any large part in overview, I can see what is going on
- I can imagine changing it, adding more functionality
- I am not afraid of it”

Richard Gabriel, *Patterns of Software*

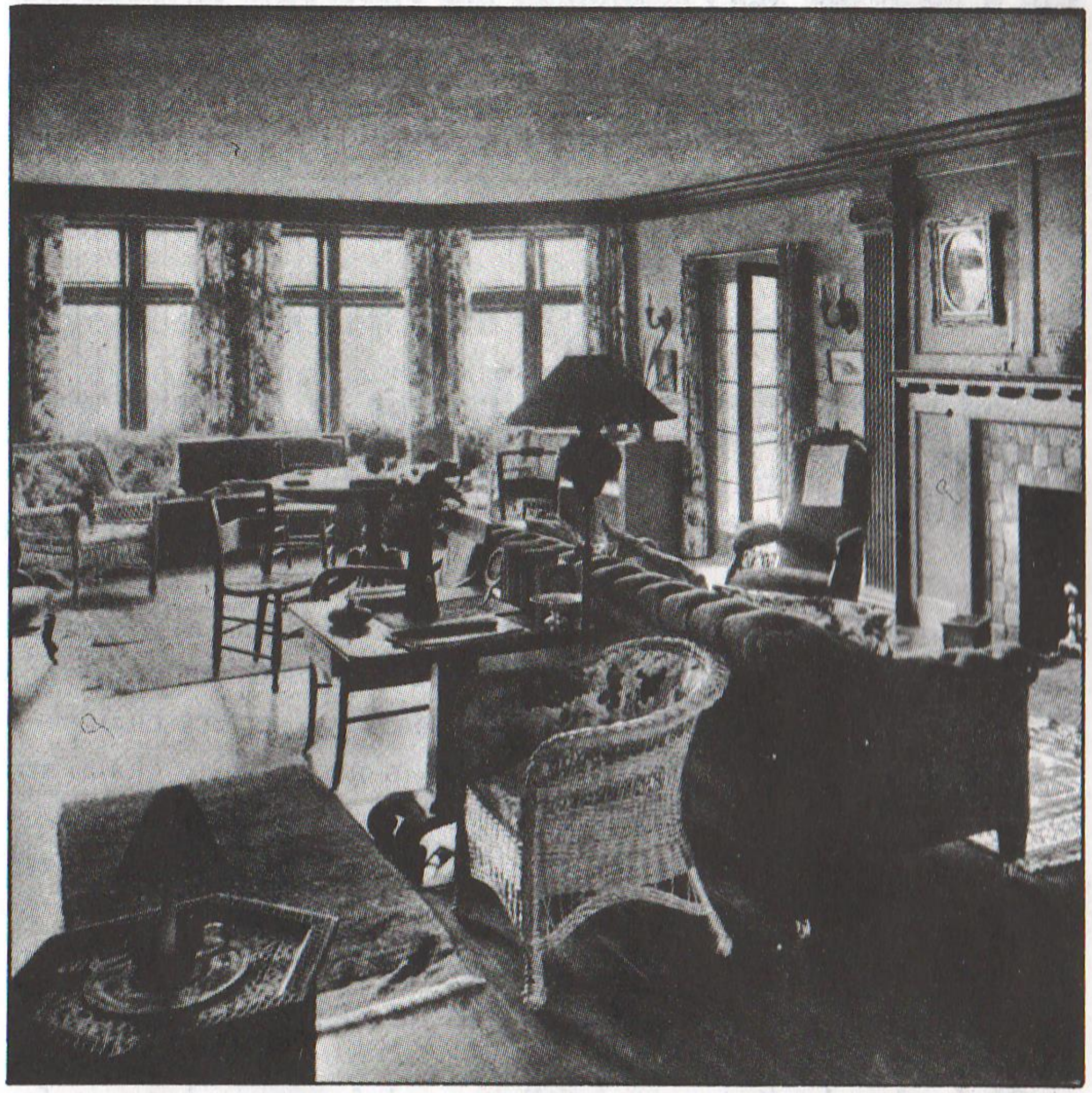
A Pattern Language

- Network of 253 patterns, structured in 3 sections:
- Towns: town and country, roads and paths, work and family, public institutions for a neighbourhood (1-94)
- Buildings: “we are dealing for the first time with patterns that are under the control of individuals or small groups of individuals, who are able to build the patterns all at once.” (95-204)
- Construction: concrete patterns for building (205-253)

Alexander Pattern Form

1. Picture
2. Introductory paragraph setting context and explaining how it helps to complete larger patterns
3. Headline (bold)
4. Body of the problem; Therefore:
5. Solution (bold)
6. Diagram
7. Final paragraph tying the pattern to smaller patterns needed to complete the pattern

251 DIFFERENT CHAIRS



. . . when you are ready to furnish rooms, choose the variety of furniture as carefully as you have made the building, so that each piece of furniture, loose or built in, has the same unique and organic individuality as the rooms and alcoves have—each different, according to the place it occupies—SEQUENCE OF SITTING SPACES (142), SITTING CIRCLE (185), BUILT-IN SEATS (202).

✣ ✣ ✣

People are different sizes; they sit in different ways. And yet there is a tendency in modern times to make all chairs alike.

Of course, this tendency to make all chairs alike is fueled by the demands of prefabrication and the supposed economies of scale. Designers have for years been creating "perfect chairs"—chairs that can be manufactured cheaply in mass. These chairs are made to be comfortable for the average person. And the institutions that buy chairs have been persuaded that buying these chairs in bulk meets all their needs.

But what it means is that some people are chronically uncomfortable; and the variety of moods among people sitting gets entirely stifled.

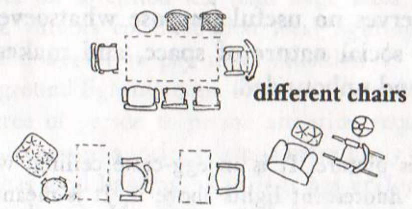
Obviously, the "average chair" is good for some, but not for everyone. Short and tall people are likely to be uncomfortable. And although situations are roughly uniform—in a restaurant everyone is eating, in an office everyone is working at a table—even so, there are important distinctions: people sitting for different lengths of time; people sitting back and musing; people sitting aggressively forward in a hot discussion; people sitting formally, waiting for a few minutes. If the chairs are all the same, these differences are repressed, and some people are uncomfortable.

What is less obvious, and yet perhaps most important of all, is this: we project our moods and personalities into the chairs we sit in. In one mood a big fat chair is just right; in another

mood, a rocking chair; for another, a stiff upright; and yet again, a stool or sofa. And, of course, it isn't only that we like to switch according to our mood; one of them is our favorite chair, the one that makes us most secure and comfortable; and that again is different for each person. A setting that is full of chairs, all slightly different, immediately creates an amosphere which supports rich experience; a setting which contains chairs that are all alike puts a subtle straight jacket on experience.

Therefore:

Never furnish any place with chairs that are identically the same. Choose a variety of different chairs, some big, some small, some softer than others, some rockers, some very old, some new, with arms, without arms, some wicker, some wood, some cloth.



✣ ✣ ✣

Where chairs are placed alone and where chairs are gathered, reinforce the character of the places which the chairs create with POOLS OF LIGHT (252), each local to the group of chairs it marks. . . .

251 Different Chairs

People are different sizes; they sit in different ways. And yet there is a tendency in modern times to make all chairs alike.

Therefore:

Never furnish any place with chairs that are identically the same. Choose a variety of different chairs, some big, some small, some softer than others, some rockers, some very old, some new, with arms, without arms, some wicker, some wood, some cloth.

Language for a Porch (page xxxv)

- Private Terrace on the Street (140)
- Sunny Place (161)
- Outdoor Room (163)
- Six-foot Balcony (167)
- Paths and Goals (120)
- Ceiling Height Variety (190)
- Columns at the Corners (212)
- Front Door Bench (242)
- Raised Flowers (245)
- Different Chairs (251)

Pattern Relationships

Common Areas at the Heart (129): **Create a single common area for every social group. Locate it at the center of gravity of all the spaces the group occupies, and in such a way that the paths which go in and out of the building lie tangent to it.**

A Room of One's Own (141): **Give each member of the family a room of his own, especially adults.**

Use this pattern as an antidote to the extremes of 'togetherness' created by Common Areas at the Heart.

Application of Patterns to Software

- 1987: Kent Beck and Ward Cunningham, Using Pattern Languages for Object-Oriented Programs
- 1992: Peter Coad, Object-oriented Patterns
- 1994: First Pattern Languages of Programs (PLoP) conference
- 1995: Jim Coplien and Doug Schmidt, Pattern Languages of Program Design
- 1995: Erich Gamma et al. Design Patterns: Elements of Reusable Object-Oriented Software
- 1996: Richard Gabriel, Patterns of Software: Tales from the Software Community

Composed Method (Kent Beck SBPP)

- How do you divide a program into methods?
- Divide your program into methods that perform one identifiable task. Keep all of the operations in a method at the same level of abstraction. This will naturally result in programs with many small methods, each a few lines long.

```
Controller>>controlActivity
        self controlInitialize.
        self controlLoop.
        self controlTerminate
```

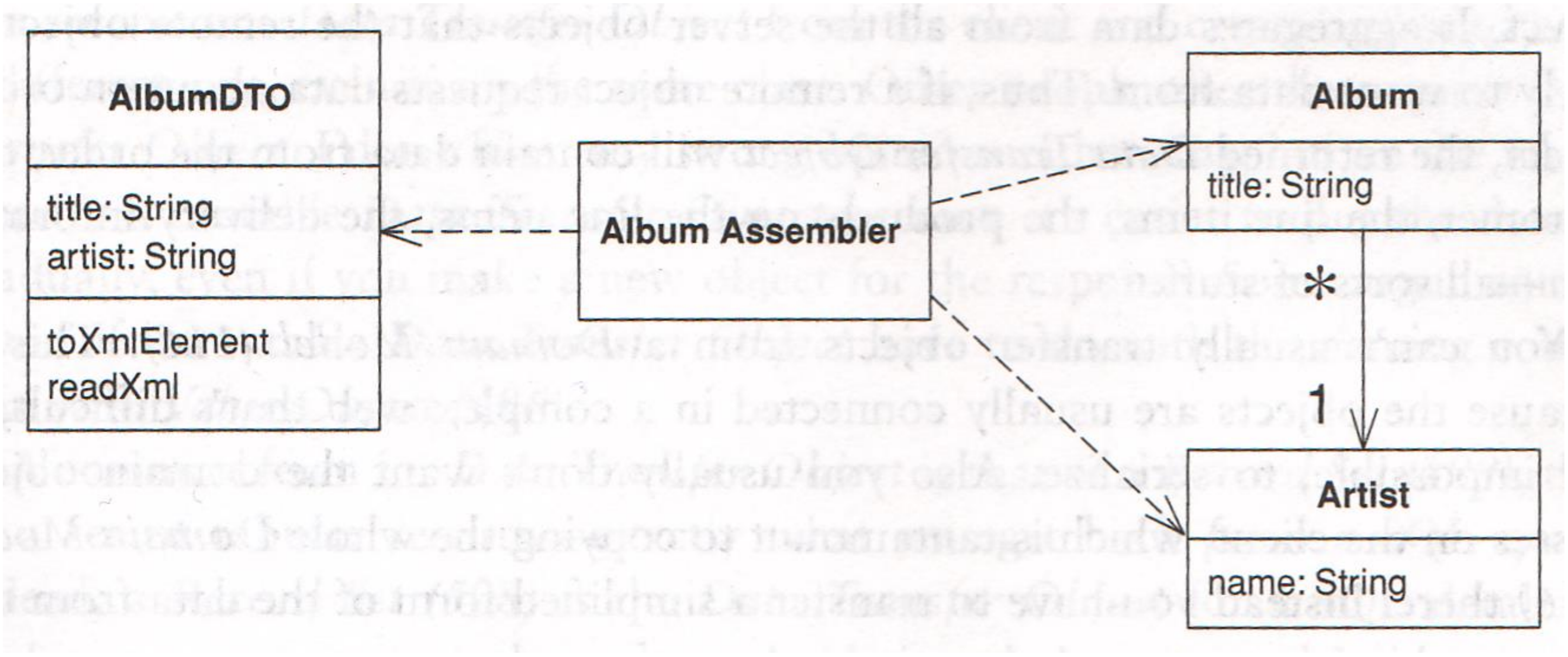
Intention Revealing Selector (SBPP)

- What do you name a method?
- Name methods after what they accomplish

Collection>>includes:

(rather than linearSearchFor:, hashedSearchFor:, or searchFor:)

Data Transfer Object (Fowler PEAA)



Patterns for End-user-programmers

"Alexander proposes homes and offices be designed and built by their eventual occupants. These people, he reasons, know best their requirements for a particular structure. We agree, and make the same argument for computer programs. Computer users should write their own programs."

Kent Beck and Ward Cunningham (1987) Using Pattern Languages for Object-Oriented Programs

Patterns for Programmers

"At least one computer scientist identified the 'user' of a piece of software as the end user. This appears to make sense at first, but when you read Alexander, it is clear that a 'user' is an inhabitant – someone who lives in the thing constructed. The thing constructed is under constant repair by its inhabitants, and end users of software do not constantly repair the software, though some might want to."

Richard Gabriel (1996) The Quality Without a Name in *Patterns of Software*

Patterns of User Experience

"As a rule of thumb, anyone who regularly refers to pattern languages of programming is not likely to be the intended audience for patterns of user experience. Patterns of user experience are, however, more closely related to the architectural interpretation of Alexander's work. The 'internal' design patterns so popular in the software patterns community might be compared to a particular pattern of screws and brackets with which two beams can be securely connected…"

Alan Blackwell and Sally Fincher (2010) PUX: Patterns of User Experience

A Pattern Language for Infusion

- For Infusion, can we make a network of patterns at different ‘scales’?
- User experience
- System integration
- Software design/architecture
- Construction

Creative User Rights

- Take a software application apart and use only one piece of it
- Combine two software applications together into one
- See the state of a software application, and present it differently inside another software application
- Customize the user interface so that the material presented inside it is easier for me to read
- Break a software application down into smaller (independently functioning) parts to understand how each part works

Resources

- Hillside Group Patterns pages
- Portland Pattern Repository
- Kent Beck and Ward Cunningham: Using Pattern Languages for Object-Oriented Programs
- Richard Gabriel: Patterns of Software [PDF]
- Blackwell and Fincher: Patterns of User Experience [PDF]
- Martin Fowler: Writing Software Patterns

Resources

- Erin Malone: A History of Patterns in User Experience Design
- Creative User Rights (Fluid wiki)
- An Infusion Pattern Language (Fluid wiki)

Acronyms

- SBPP: *Smalltalk Best Practice Patterns* by Kent Beck
- PEAA: *Patterns of Enterprise Application Architecture* by Martin Fowler