Plain (Pattern) Language

A Pattern Language (Alexander, 1977) book review

Will Wright (2011)

“I was always interested in architecture, and so one of the original things that was a really **inspiration for The Sims** was this book, *A Pattern Language*, by Christopher Alexander.”

https://www.gamedeveloper.com/business/the-replay-interviews-will-wright

foreword from *Design Patterns* (1994)

"The importance of patterns in crafting complex systems has been long recognized in other disciplines. In particular, **Christopher Alexander** and his colleagues were perhaps the first to propose the idea of **using a pattern language to architect buildings and cities**."

Hi

I'm **Lorie den Os**

Product Tech Lead @ Yseop

java developer ~2016

linguist ~2010

simmer ~2000

book review~

- what is *A Pattern Language* and who is Christopher Alexander?
- what can we expect to find in the book?
- what is actually in the book
- can we use this “pattern language”?

what is *A Pattern Language* and who is Christopher Alexander?

Christopher Alexander (1936-2022)
Austrian-born British-American **architect** and **design theorist**

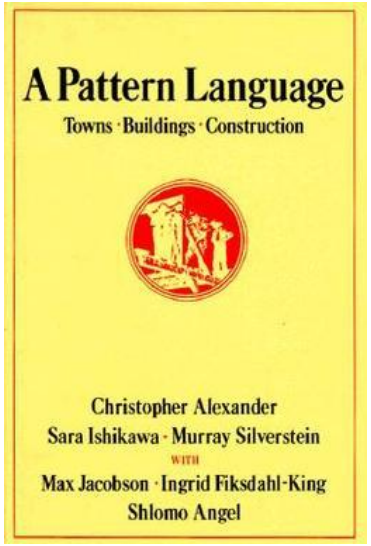
father of the **pattern language movement**: wiki's, agile software development, interaction design patterns, pedagogical patterns, permaculture, communication patterns…

what is *A Pattern Language* and who is Christopher Alexander?

written in the 70's by Alexander and his students

part of a series:
- The Oregon Experiment (1975)
- **A Pattern Language (1977)**
- The Timeless Way of Building (1979)
- (5 other books…)

"**you** can use this book to design a house **for yourself** with your family"



what did I
expect to find
in the book as
a dev

what did I expect to find in the book?

with *Design Patterns* in mind (23 patterns)

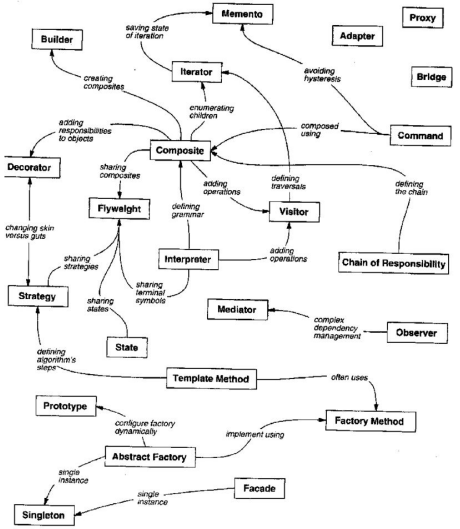
pattern = name, problem, solution, consequences

"elements of reusable object-oriented software"

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight (195) Observer State Strategy Visitor

Purpose	Design Pattern	Aspect(s) That Can Vary
Creational	Abstract Factory (87)	families of product objects
	Builder (97)	how a composite object gets created
	Factory Method (107)	subclass of object that is instantiated
	Prototype (117)	class of object that is instantiated
	Singleton (127)	the sole instance of a class
Structural	Adapter (139)	interface to an object
	Bridge (151)	implementation of an object
	Composite (163)	structure and composition of an object
	Decorator (175)	responsibilities of an object without subclassing
	Facade (185)	interface to a subsystem
	Flyweight (195)	storage costs of objects
	Proxy (207)	how an object is accessed; its location
Behavioral	Chain of Responsibility (223)	object that can fulfill a request
	Command (233)	when and how a request is fulfilled
	Interpreter (243)	grammar and interpretation of a language
	Iterator (257)	how an aggregate's elements are accessed, traversed
	Mediator (273)	how and which objects interact with each other
	Memento (283)	what private information is stored outside an object, and when
	Observer (293)	number of objects that depend on another object; how the dependent objects stay up to date
	State (305)	states of an object
	Strategy (315)	an algorithm
	Template Method (325)	steps of an algorithm
	Visitor (331)	operations that can be applied to object(s) without changing their class(es)

Table 1.2: Design aspects that design patterns let you vary



what did I expect to find in the book?

with *Design Patterns* in mind (23 patterns)

pattern = name, ~~problem~~, **solution**, consequences

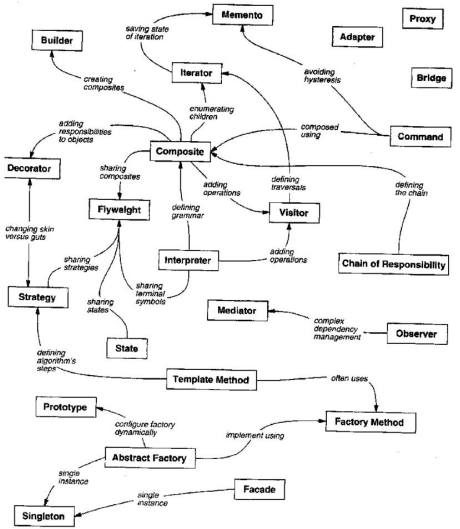
motivation + applicability

"elements of reusable object-oriented software"

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight (195) Observer State Strategy Visitor

Purpose	Design Pattern	Aspect(s) That Can Vary
Creational	Abstract Factory (87)	families of product objects
	Builder (97)	how a composite object gets created
	Factory Method (107)	subclass of object that is instantiated
	Prototype (117)	class of object that is instantiated
	Singleton (127)	the sole instance of a class
Structural	Adapter (139)	interface to an object
	Bridge (151)	implementation of an object
	Composite (163)	structure and composition of an object
	Decorator (175)	responsibilities of an object without subclassing
	Facade (185)	interface to a subsystem
	Flyweight (195)	storage costs of objects
	Proxy (207)	how an object is accessed; its location
Behavioral	Chain of Responsibility (223)	object that can fulfill a request
	Command (233)	when and how a request is fulfilled
	Interpreter (243)	grammar and interpretation of a language
	Iterator (257)	how an aggregate's elements are accessed, traversed
	Mediator (273)	how and which objects interact with each other
	Memento (283)	what private information is stored outside an object, and when
	Observer (293)	number of objects that depend on another object; how the dependent objects stay up to date
	State (305)	states of an object
	Strategy (315)	an algorithm
	Template Method (325)	steps of an algorithm
	Visitor (331)	operations that can be applied to object(s) without changing their class(es)

Table 1.2: Design aspects that design patterns let you vary



what did I expect to find in the book?

with *Design Patterns* in mind (23 patterns)

→ a catalog of patterns to address common architectural problems?

→ real-life examples, maps, blueprints?

→ ~pragmatism: no reinventing the wheel!

what did I
expect to
find in the
book as a linguist

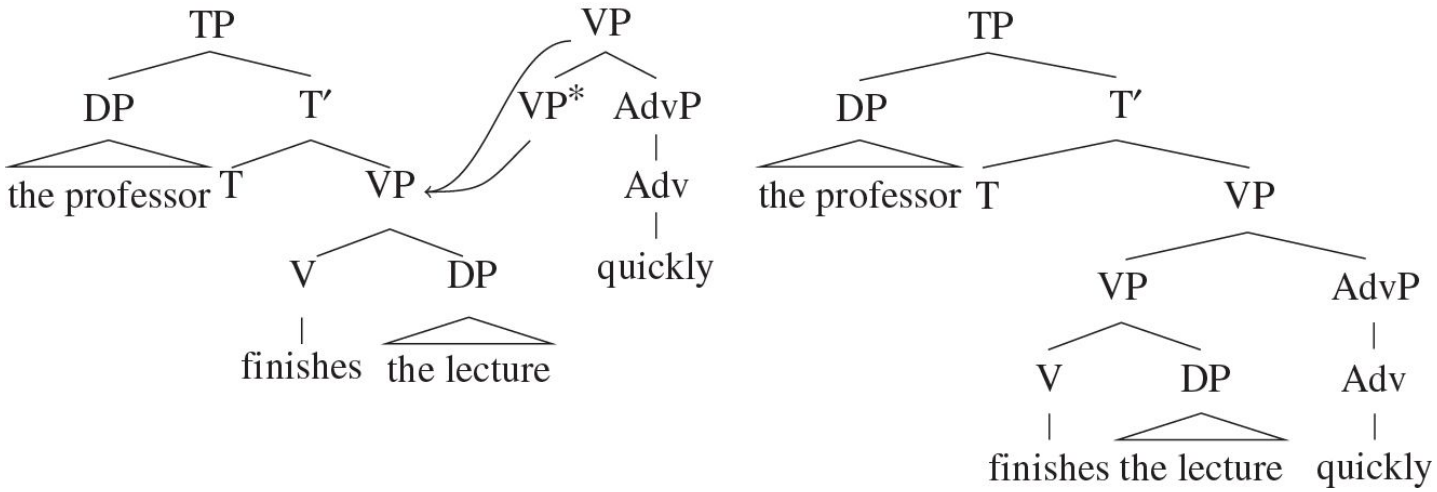
dust jacket from *A Pattern Language* (1977)

“At the core of the books too is the point that in designing their environments people always rely on certain “languages,” which, **like the languages we speak**, allow them to articulate and communicate an infinite variety of designs within **a formal system which gives them coherence**.”

what did I expect to find in the book?

as a linguist specialized in Natural Language Generation

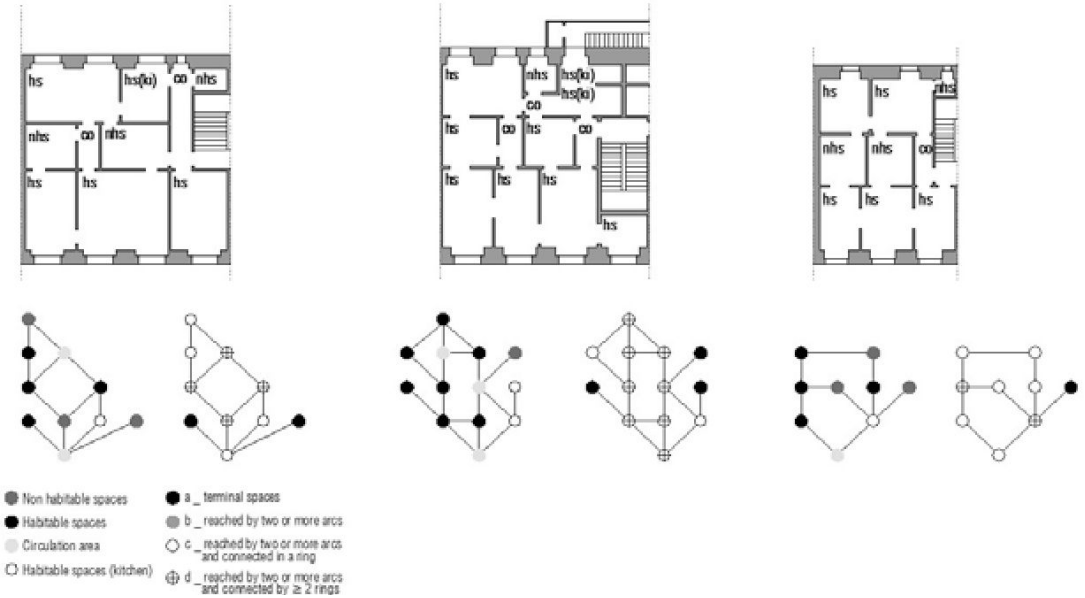
tree-adjoining grammar:



what did I expect to find in the book?

as a linguist specialized in Natural Language Generation

space syntax:



what did I expect to find in the book?

as a linguist specialized in Natural Language Generation

→ a ~formal grammar of patterns? (buildings as sentences, rooms as words?)

→ tree-like structure or graph-like structure?

→ production rules to “generate buildings”?

→ criteria to decide which buildings are *good*

what is
actually
in the book

what is **actually** in the book

→ a catalog of **253** patterns organized as “a network”

→ an “ordered hierarchy”:

- towns (patterns 1 to 94)
- buildings (patterns 95 to 204)
- construction (patterns 205 to 253)

→ a guide on how to create a “sequence” of patterns to *generate* buildings

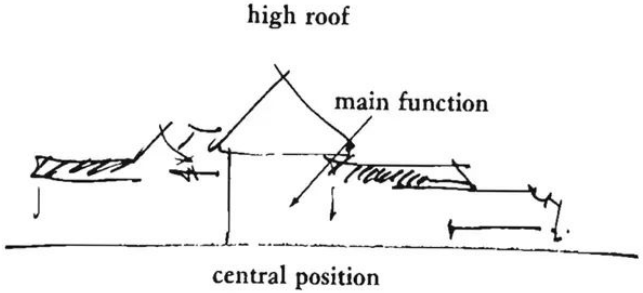
what is actually in the book

pattern =
- name
- id (1 to 253)
- asterisks (0, 1 or 2) to denote confidence in the pattern
- picture (black and white)
- context 1 (links to previous patterns)
- headline: the essence of the problem
- problem body: empirical evidence
- solution (stated as an instruction)
- diagram (handdrawn)
- context 2 (links to smaller patterns)

what is actually in the book

pattern =

- name **Main Building**
- id 99
- asterisks *
- picture
- context 1 95, 96, 98
- headline **"A complex of buildings with no center is like a man without a head."**
- problem body (3 paragraphs)
- solution (2 paragraphs)
- diagram
- context 2 116, 129, 205

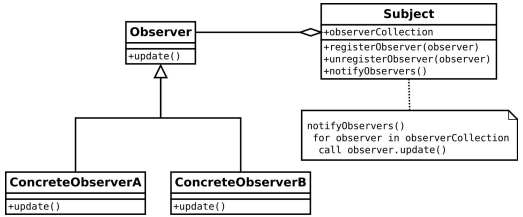


Comparison between a pattern from

Design Patterns

OBSERVER (Publish-Subscribe)

→ we don’t want tight coupling between objects



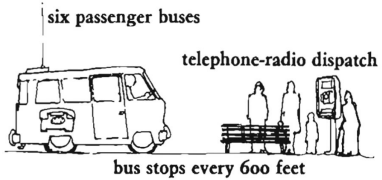
→ create a notification system and a register of notification receivers

related to MEDIATOR and SINGLETON

A Pattern Language

20 - MINI-BUSES*

→ we need public transportation to cover the full metropolitan area



→ create a system of taxi-like buses for up to 6 people, radio-controlled, optimized

related to PARALLEL ROADS, INTERCHANGE, BUS STOP

what is actually in the book

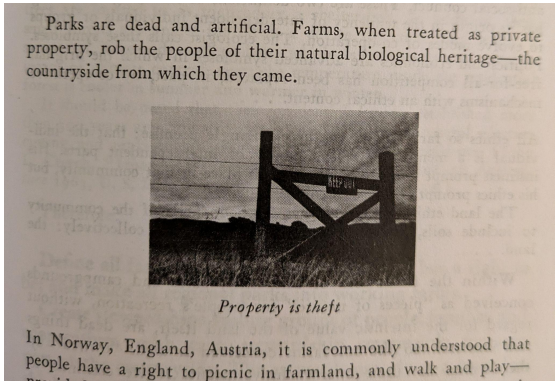
→ a proposition of patterns, a guide to build **more patterns**

→ book quotes, personal opinions, ideals

→ philosophical and political statement about the world

→ architecture as **poetry**

→ architecture for people,
to solve people's problems and
to create joy



Parks are dead and artificial. Farms, when treated as private property, rob the people of their natural biological heritage—the countryside from which they came.

Property is theft

In Norway, England, Austria, it is commonly understood that people have a right to picnic in farmland, and walk and play—

can we
use this
“pattern
language”?

can we use this “pattern language”?

→ there are A LOT of patterns

→ patterns are more elaborate than “software design patterns” and seem to capture *bigger problems*

→ this is not a grammar to generate buildings

→ it reads like…

…specifications

can we use this “pattern language”?

dev x linguist: “let’s have a look at these ***problems***”

"it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail." (attributed to Maslow, 1966)

→ a “graph of patterns” won’t help me

→ how about we separate problems from solutions…

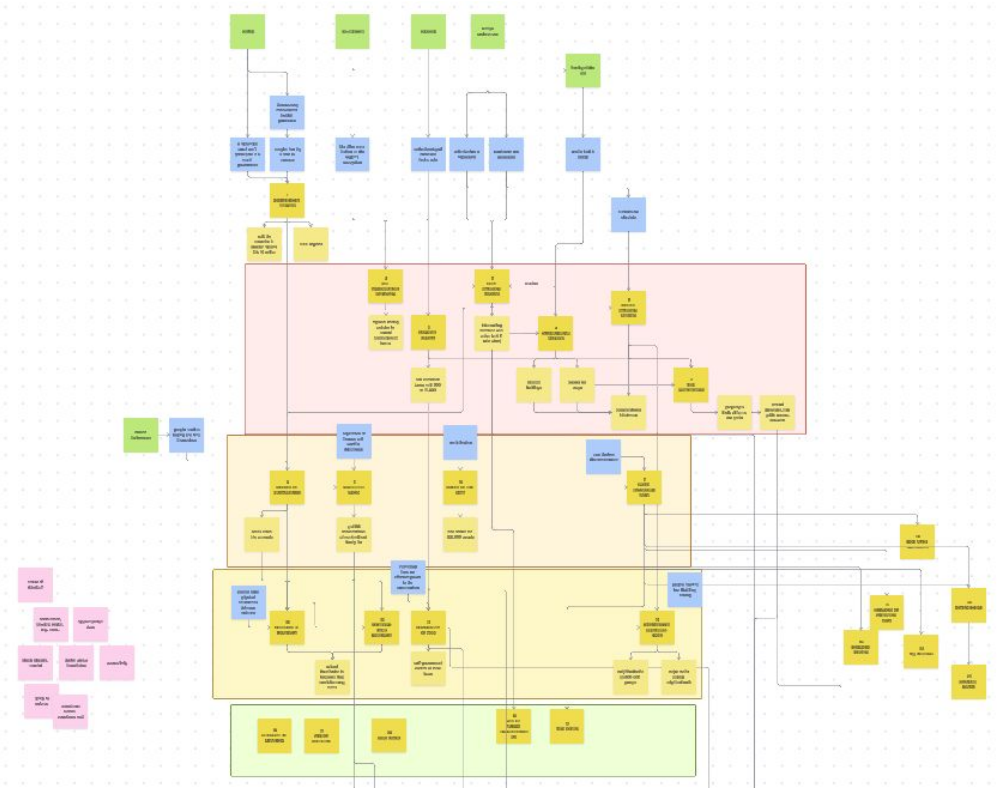
can we use this “pattern language”?

Pattern ID	Pattern Label	Page	Pattern level	Pattern stars	Problem type	Solution type	Scale	Graph-lik
69	Public outdoor room	348	towns	★★☆☆☆	lack of common land	urban planning	town	TRUE
70	Grave sites	353	towns	★☆☆☆☆	mortality, mental health	city policy and urban pla	town	FALSE
71	Still water	358	towns	★☆☆☆☆	lack of exercise (water)	urban planning	town	TRUE
72	Local sports	363	towns	★☆☆☆☆	lack of exercise	urban planning	town	TRUE
73	Adventure playground	367	towns	☆☆☆☆☆	lack of exercise (children)	urban planning	town	TRUE
74	Animals	371	towns	☆☆☆☆☆	lack of nature (animals)	city policy and urban pla	town	FALSE
75	The family	376	towns	★☆☆☆☆	mental health (family)	community	household	TRUE
76	House for a small family	381	towns	★☆☆☆☆	dependencies between people	architecture	household	TRUE
77	House for a couple	385	towns	★☆☆☆☆	lack of privacy	architecture	household	TRUE
78	House for one person	389	towns	★☆☆☆☆	lack of simplicity	architecture	household	TRUE
79	Your own home	392	towns	★★☆☆☆	lack of agency/independance	policy	household	FALSE
95	Building complex	468	buildings	★★☆☆☆	not a problem: a goal	architecture	building	FALSE
96	Number of stories	473	buildings	★☆☆☆☆	not a problem: a constraint	maths	building	TRUE
97	Shielded parking	477	buildings	★☆☆☆☆	not a problem: a goal	architecture	building	TRUE
98	Circulation realms	480	buildings	★★☆☆☆	mental health (disorientation)	urban planning	any	TRUE
99	Main building	485	buildings	★☆☆☆☆	not a problem: a constraint	architecture	any	TRUE

can we use this “pattern language”?

→ **opportunity mapping** applied to patterns

“opportunity mapping is the critical activity of taking inventory of and giving structure to the opportunity space by using an opportunity solution tree to visualize customer needs, pain points, and desires”



work in progress

- ❑ list all problems lol
  - ❑ find structural elements, building blocks…
  - ❑ find patterns, a hierarchy, semantic links between them…
- ❑ list all solutions
  - ❑ find structural elements, building blocks…
  - ❑ find patterns, a hierarchy, semantic links between them…
- ❑ find patterns, semantic links between problem/solution pairs
- ❑ look at findings as constraints
- ❑ create a meta-grammar of pattern parts
- ❑ use solver to generate patterns as a grammar
- ❑ use grammar to generate rooms, houses, towns…
- ❑ find a way to score generated houses
- ❑ profit

Will Wright (2011)

“In some sense I wanted The Sims originally to be an architecture game where it was analyzing these patterns. **So the people in The Sims originally were just there to score the architecture.**”

https://www.gamedeveloper.com/business/the-replay-interviews-will-wright

conclusion

conclusion

“I heard a rumor at breakfast that **some of the people in this room have begun to worry about their jobs**. I have no idea if that is true. But I was told there is an undercurrent of unease as to where all this—software design—is going. There is a huge expanding phenomenon of programming as an art, and yet an uneasiness about where it is all headed? What is it going to do?

My comment on this? Please forgive me, I'm going to be very direct and blunt for a horrible second. It could be thought that the technical way in which you currently look at programming is almost as if you were willing to be "guns for hire." In other words, you are the technicians. You know how to make the programs work. "Tell us what to do daddy, and we'll do it." **That is the worm in the apple.**

What I am proposing here is something a little bit different from that. It is a view of programming as 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 in our towns, houses, work places, cities, becomes an attainable thing. That would be remarkable. **It would turn the world around, and make living structure the norm once again, throughout society, and make the world worth living in again.**”

https://www.patternlanguage.com/archive/ieee.html

thank you for listening

Anouk Ricard

links and references

Dawes, M.J., Ostwald, M.J. Christopher Alexander's A Pattern Language: analysing, mapping and classifying the critical response. City Territ Archit 4, 17 (2017).
https://doi.org/10.1186/s40410-017-0073-1

Space Syntax
https://www.spacesyntax.online/applying-space-syntax/building-methods/spatial-form-analysis/

A Pattern Language as graphs
http://bekawestberg.me/blog/pattern-language/

Christopher Alexander: Patterns in Architecture (OOPSLA 1996)
https://www.youtube.com/watch?v=z_QzdKci6OY

A Pattern Language full book
https://www.patternlanguageindex.com/