Design Patterns

A workshop with examples in Ruby

A little history

- Back in the 1970's an architect named Christopher Alexander was starting to think about the qualities that many architectures share
- This lead him to two conclusions:
    1. There exists a common set of architectural patterns
    2. These patterns should be named, catalogued, and described - in other words, there was a need for a pattern language
- Alexander published his findings in 2 books...

A Pattern Language

Towns · Buildings · Construction

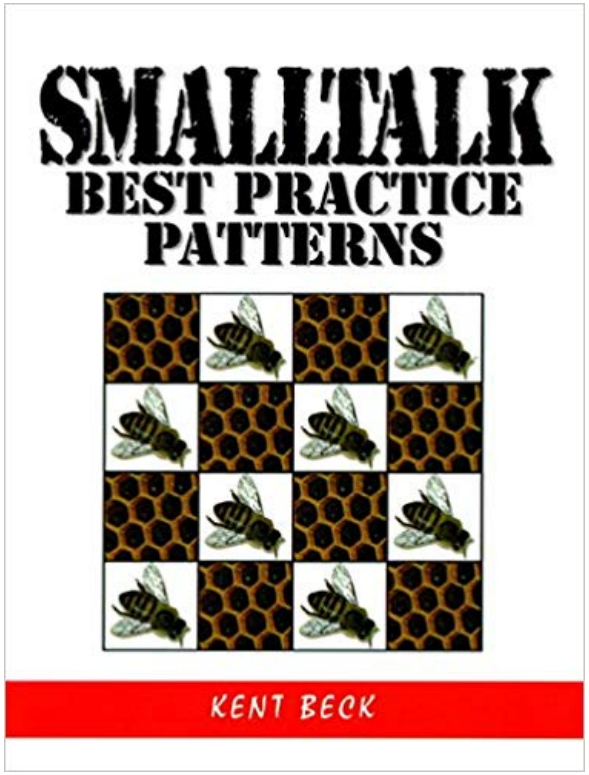
Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

The Timeless Way of Building

Christopher Alexander

Smalltalk

- Around the same time a group of Smalltalk developers led by Ward Cunningham and Kent Beck were also noticing a common set of patterns that Smalltalk develpers were using

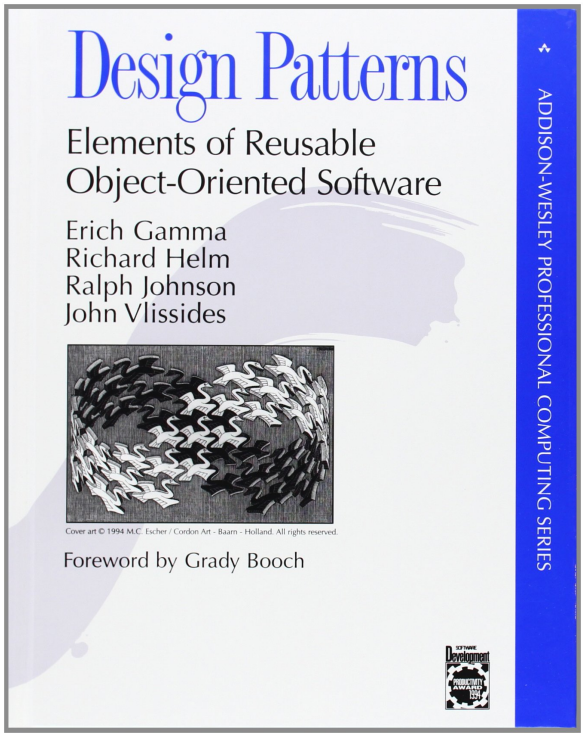


- This led Kent Beck to write what is still one of the best books ever written on software patterns

- However, prompted by the work of Alexander and the Smalltalk developers - and before Kent Beck could publish his thoughts - this happened...

Gang of Four (GoF)

- Published in 1995
- A catalogue of 23 software patterns
- All patterns are split into 3 categories:
    1. Creational Patterns
    2. Structural Patterns
    3. Behavioural Patterns



Pattern Description

Each pattern is described in 4 sections:

1. Pattern name
2. The problem - describes when to apply the pattern
3. The solution
4. Consequences (of applying the pattern)

What is a pattern?

- A reusable solution to a commonly occurring problem - an attempt to stop programmers "reinventing the wheel"
- A template of how to solve that problem
- Patterns are NOT a finished design that can be used directly as production-ready code
- Patterns are NOT a 'silver bullet'

Why do we need software patterns?

- Patterns help to speed up development
- They describe a - proven - best-practice approach to solving specific problems
- They are communication tool between developers
- They are also frequently used to screen job candidates 😉

Why are patterns useful?

They force developers to:

- Seperate out the things that change from those that don't
- Program to an interface, not an implementation
- Use composition over inheritance
- Delegate (behaviour, state, knowledge)

Possibly(?) the most common software patterns

1. Singleton *(Creational)* (*)
2. Factory Method *(Creational)* (*)
3. Strategy *(Behavioural)* (*)
4. Observer *(Behavioural)*
5. Builder *(Creational)*
6. Adapter *(Structural)* (*)
7. State *(Behavioural)*

(*) These are the patterns I personally most frequently

Workshop

https://github.com/gclikkec/design-pattern-workshop-attendee