

Software with the Quality that Has No Name

**Federico Mena Quintero
federico@gnome.org**

Desktop Summit, Berlin, Aug/2011



**Our house
in the middle of our street**

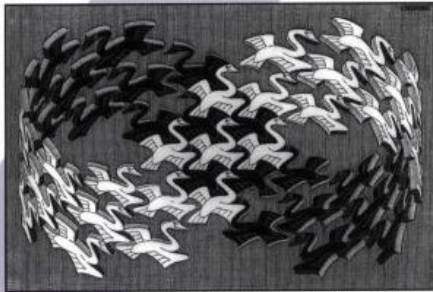
FIXME: before/after pictures

Copyrighted Material

Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Cordon Art - Baam - Holland. All rights reserved.

Foreword by Grady Booch



Copyrighted Material



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Copyrighted Material

REFACTORING

IMPROVING THE DESIGN
OF EXISTING CODE

MARTIN FOWLER

With contributions by **Kent Beck, John Brant,
William Opdyke, and Don Roberts**

Foreword by **Erich Gamma**
Object Technology International, Inc.



Copyrighted Material

1994

1999



Christopher Alexander



0976. P. 2 - VENEZIA. FACCIATA DELLA SCUOLA
DI SAN MARCO ORA OSPITALE CIVILE.



AVENUE - COLUMBIA

A Pattern Language

Towns · Buildings · Construction



Christopher Alexander

Sara Ishikawa · Murray Silverstein

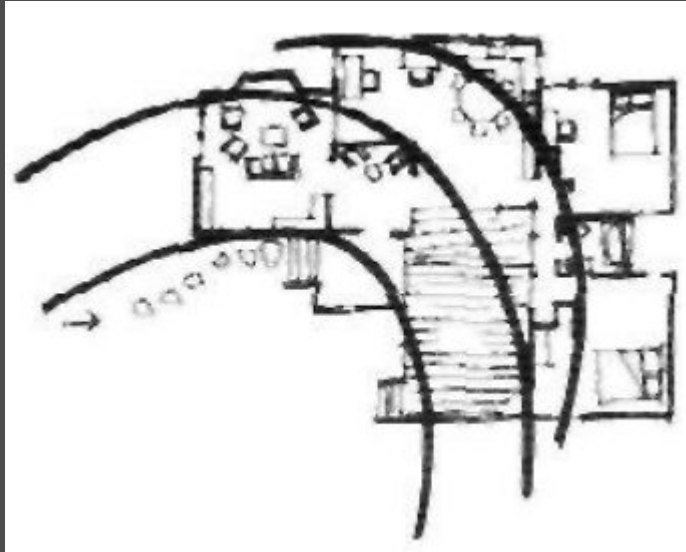
WITH

Max Jacobson · Ingrid Fiksdahl-King

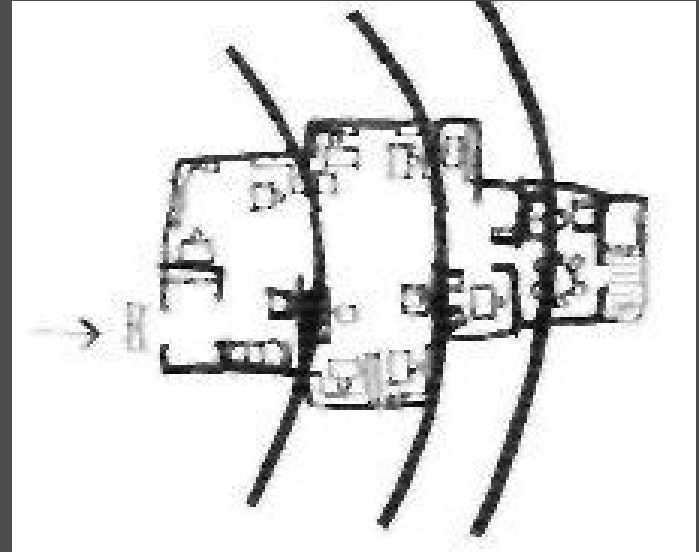
Shlomo Angel

1977

Intimacy gradient



House



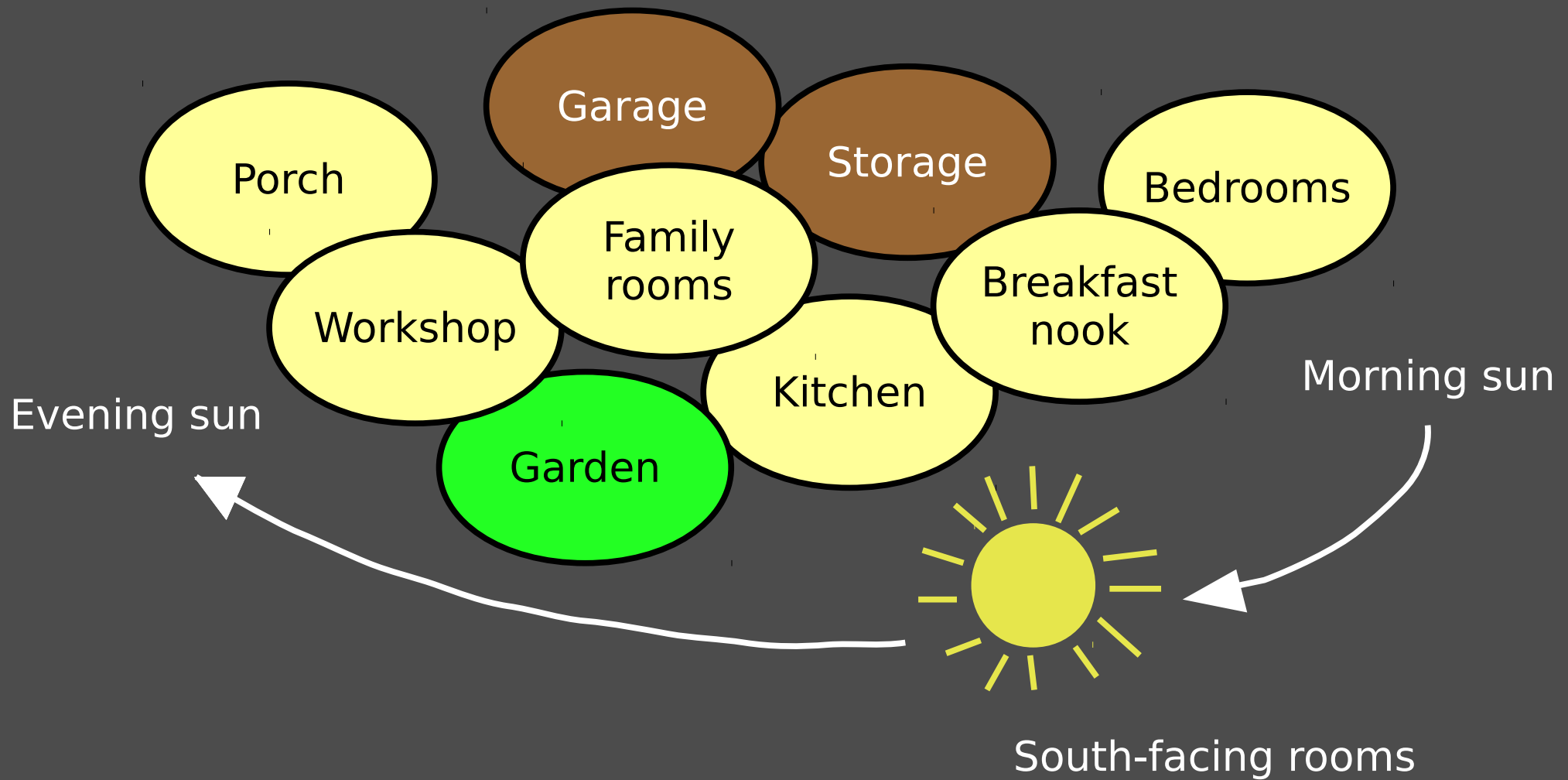
Office

Light on two sides of every room



Wrinkle the building's edge

Indoor sunlight



Pattern name

- **Super-patterns**
- **Statement of problem**
- **Discussion**
- **Summary of the solution**
- **Sub-patterns**

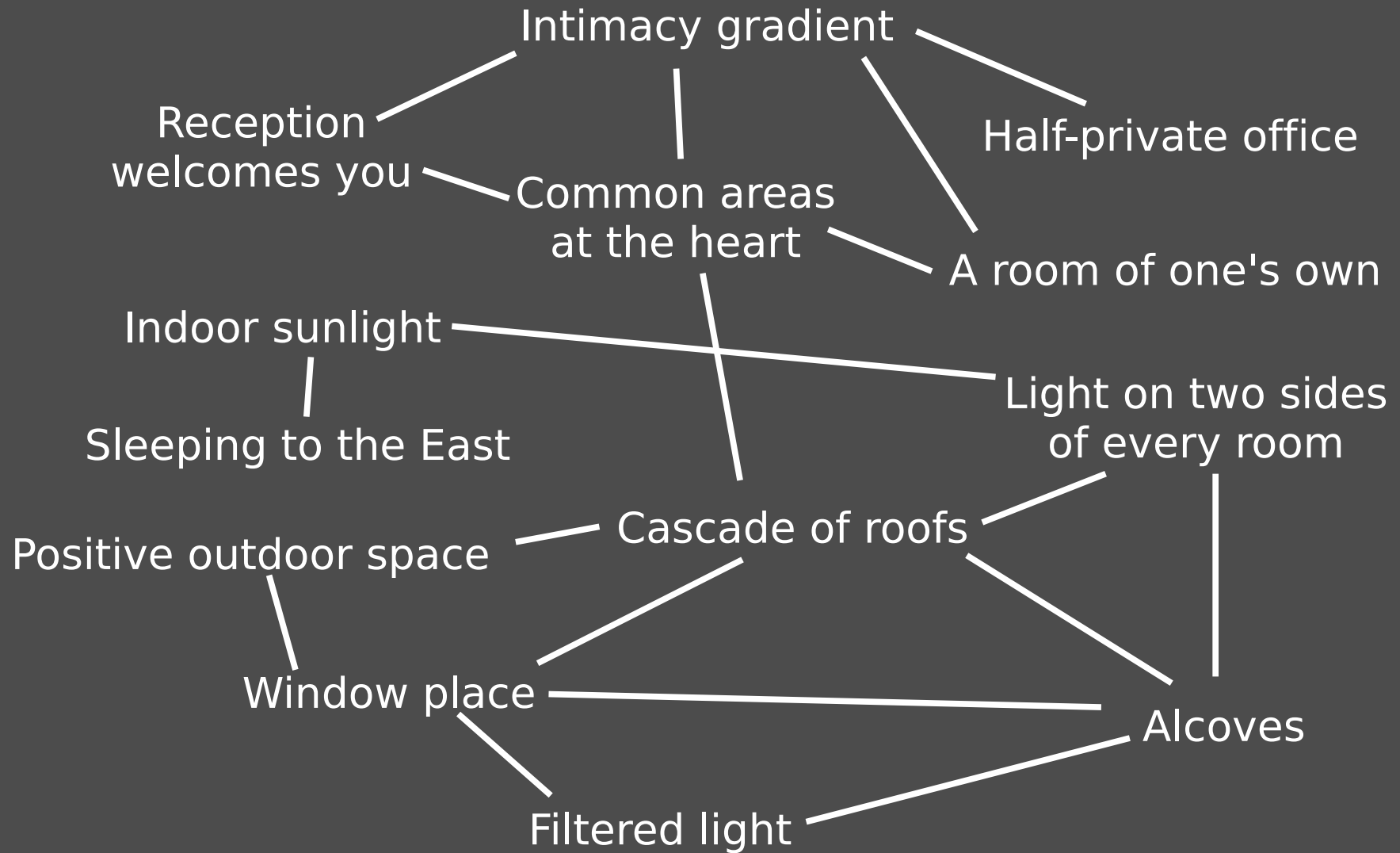
Light on two sides of every room

- **Super-patterns:** wings of light, positive outdoor space, cascade of roofs
- **Statement of problem:** People gravitate to well-lit rooms.

- **Discussion:**



- **Summary of the solution:** Light on two sides; natural light through the windows
- **Sub-patterns:** Roof layout, windows overlooking life, window place, filtered light



**Patterns
do not
give you
a final form**

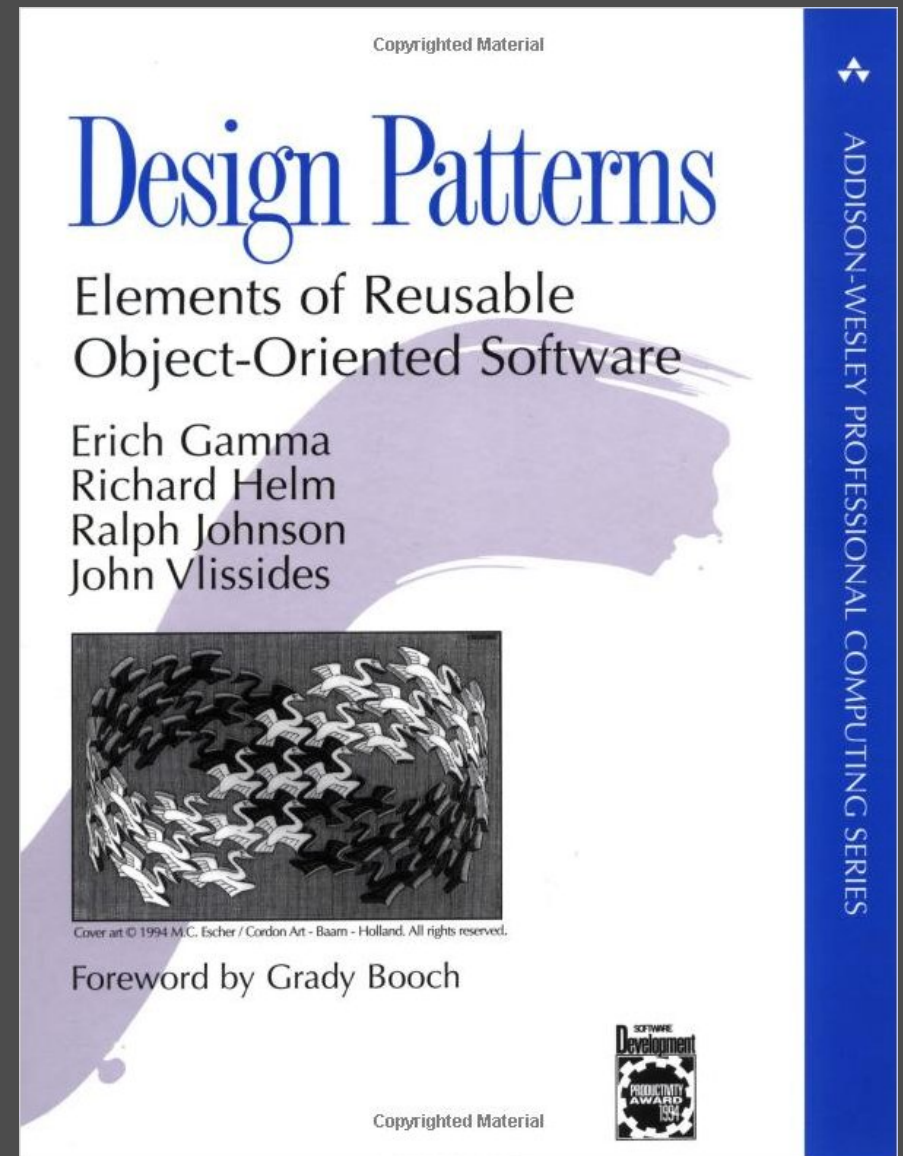
Patterns give you a vocabulary

Architecture

Alcove
Positive space
Cascade of roofs

Programming

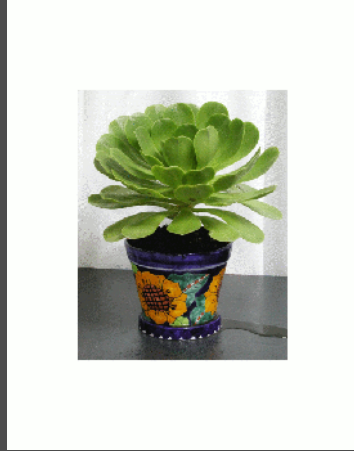
Factory
Strategy
Listener



Pattern: Zooming



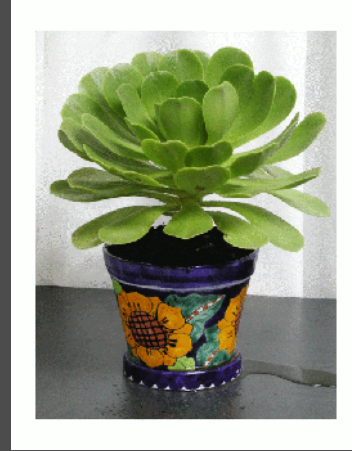
1.05^0



1.05^1



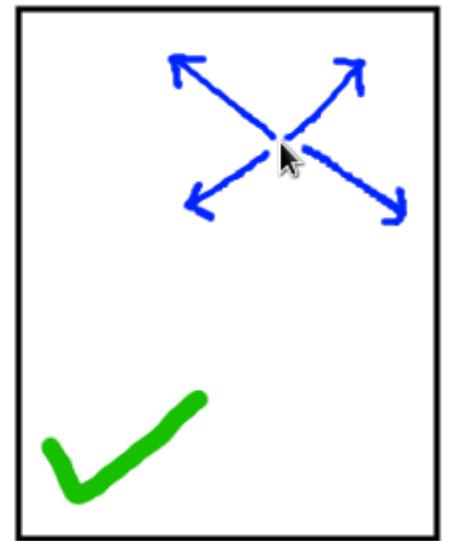
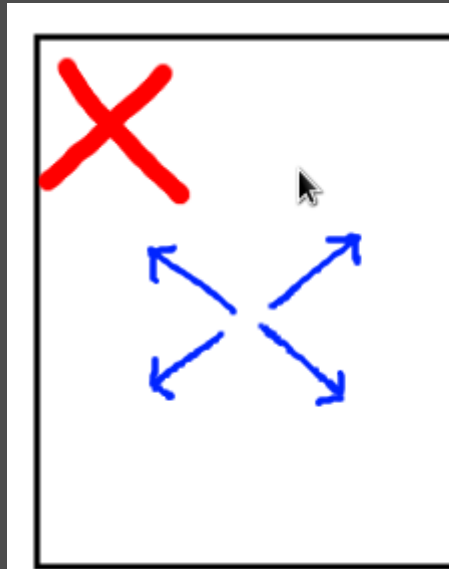
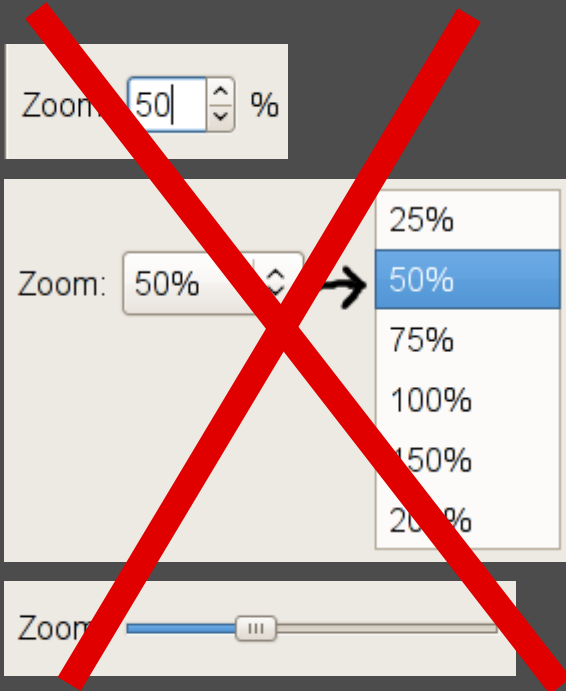
1.05^2



1.05^3



1.05^4



The ticket booth

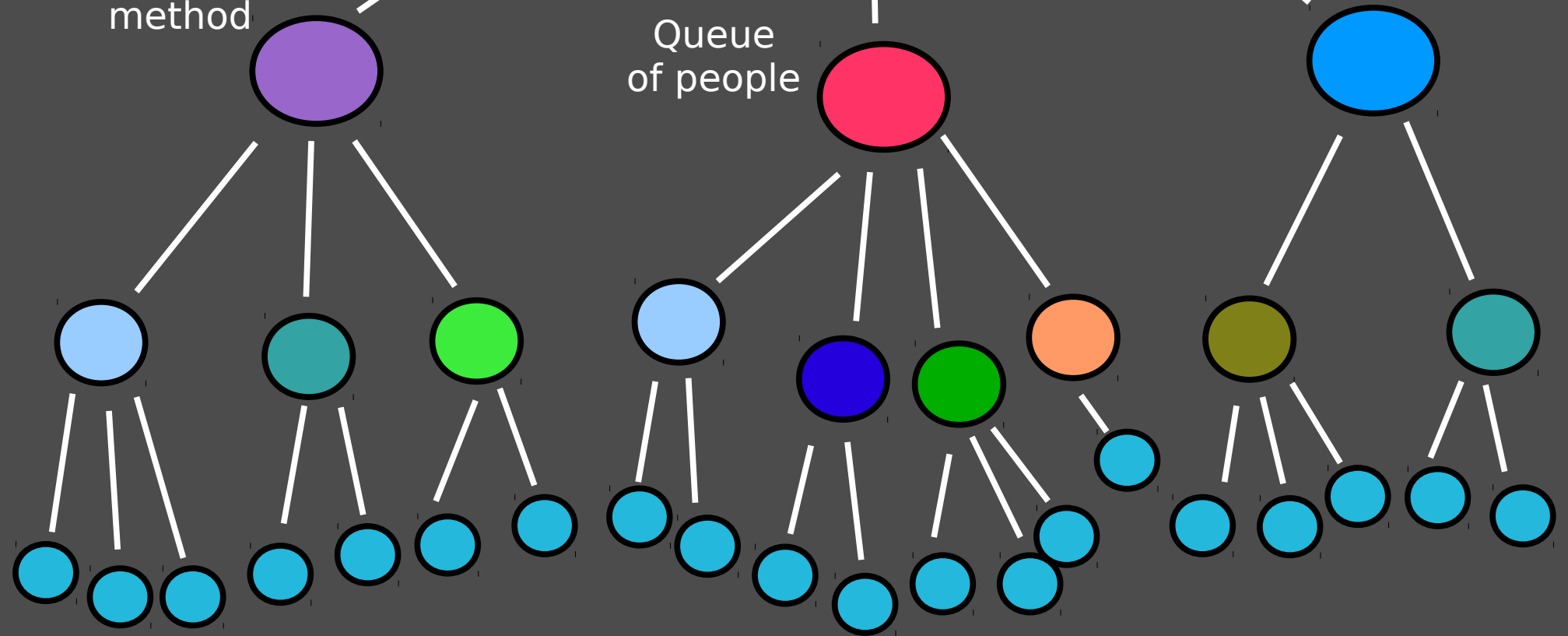


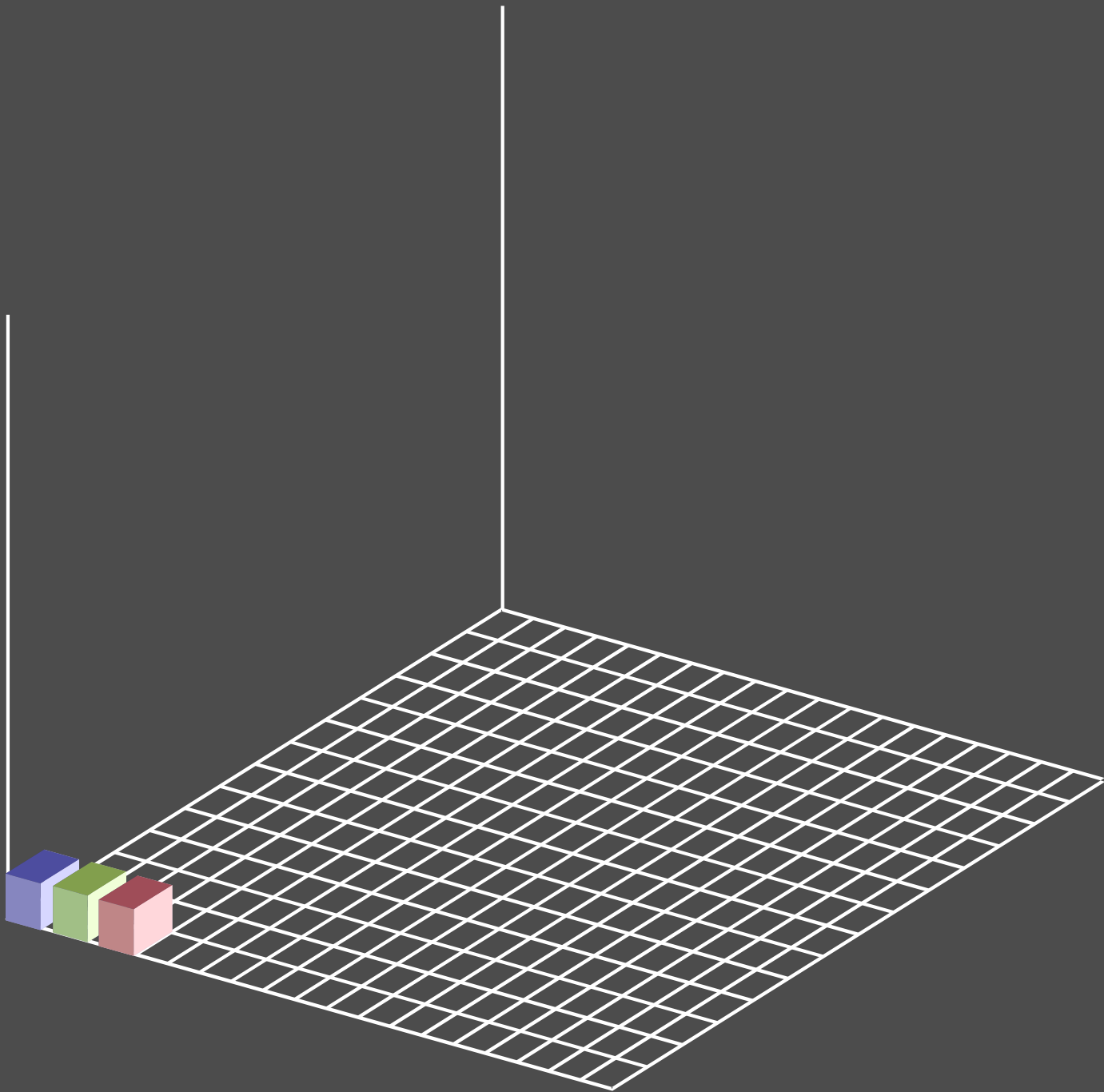
Ticket booth

Purchase
method

Queue
of people

Shape of
booth





The Quality Without A Name

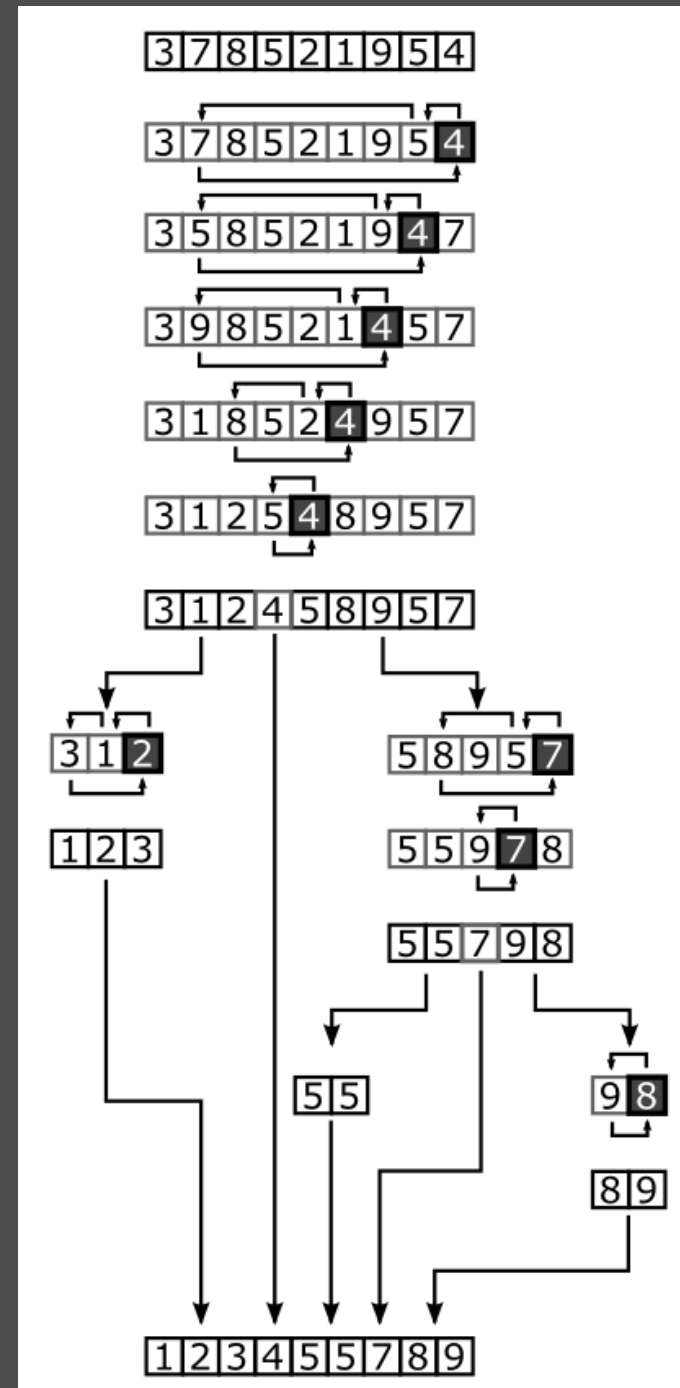


```

partition (array, left, right, pivot)
{
    pivot_value = array[pivot];
    swap (array, pivot, right);
    store = left;
    for (i = left; i < right; i++) {
        if (array[i] < pivot_value) {
            swap (array, i, store);
            store++;
        }
    }
    swap (array, store, right);
    return store;
}

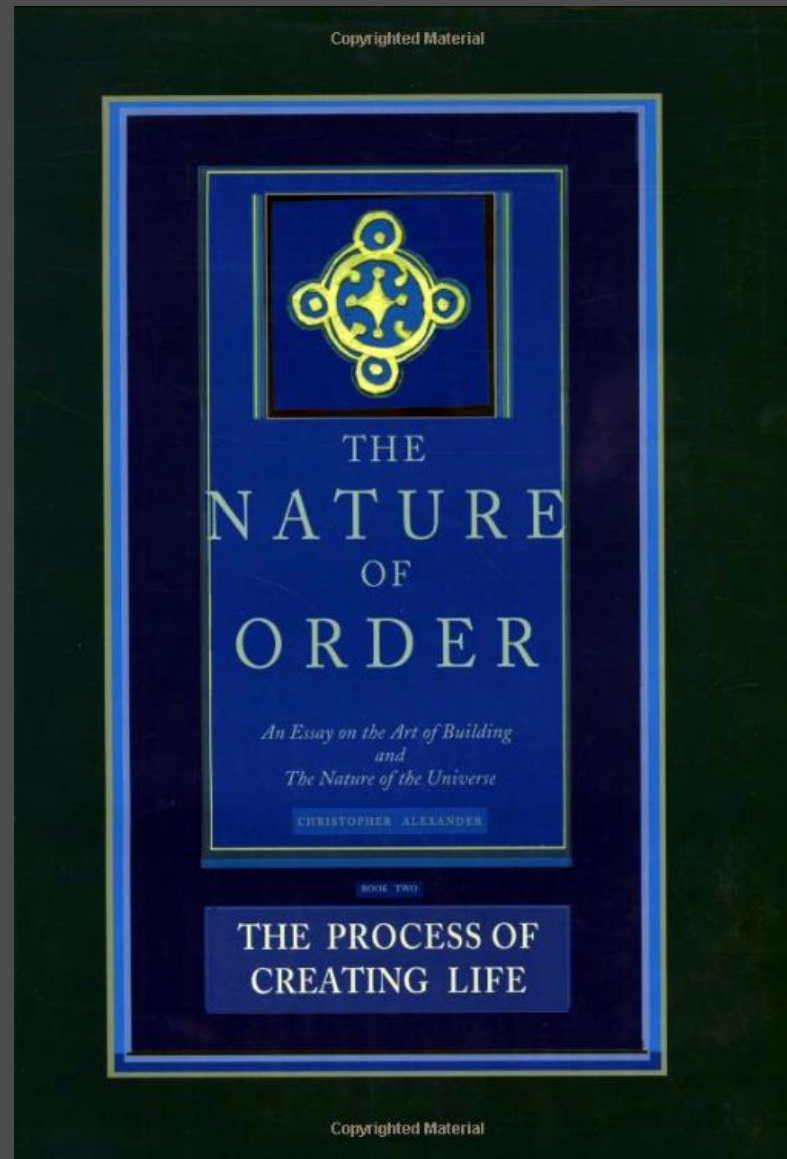
quicksort (array, left, right)
{
    if (left < right) {
        pivot = (left + right) / 2;
        new_pivot = partition (array, left, right, pivot);
        quicksort (array, left, new_pivot - 1);
        quicksort (array, new_pivot + 1, right);
    }
}

```



The Quality for Software

- *(According to Richard Gabriel)*
- It was not written to unrealistic deadline
- Any bad parts were repaired during the maintenance or are being repaired now
- If it is small, it was written by an extraordinary person, someone I would like as a friend; if it is large, it was not designed by one person, but over time in a slow, careful, incremental way
- 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
- It is like a fractal, in which every level of details is as locally coherent and as well thought as any other level
- Every part of the code is transparently clear - there are no sections that are obscure in order to gain efficiency
- Everything about it seems to be familiar
- I can imagine changing it, adding some functionality
- I am not afraid of it, I will remember it



2001-2004

15 Properties of Living Structure

Levels of scale	Strong centers	Thick boundaries
Alternating repetition	Positive space	Good shape
Local symmetries	Deep interlock and ambiguity	Contrast
Gradients	Roughness	Echoes
The void	Simplicity and inner calm	Non-separateness



N° 35

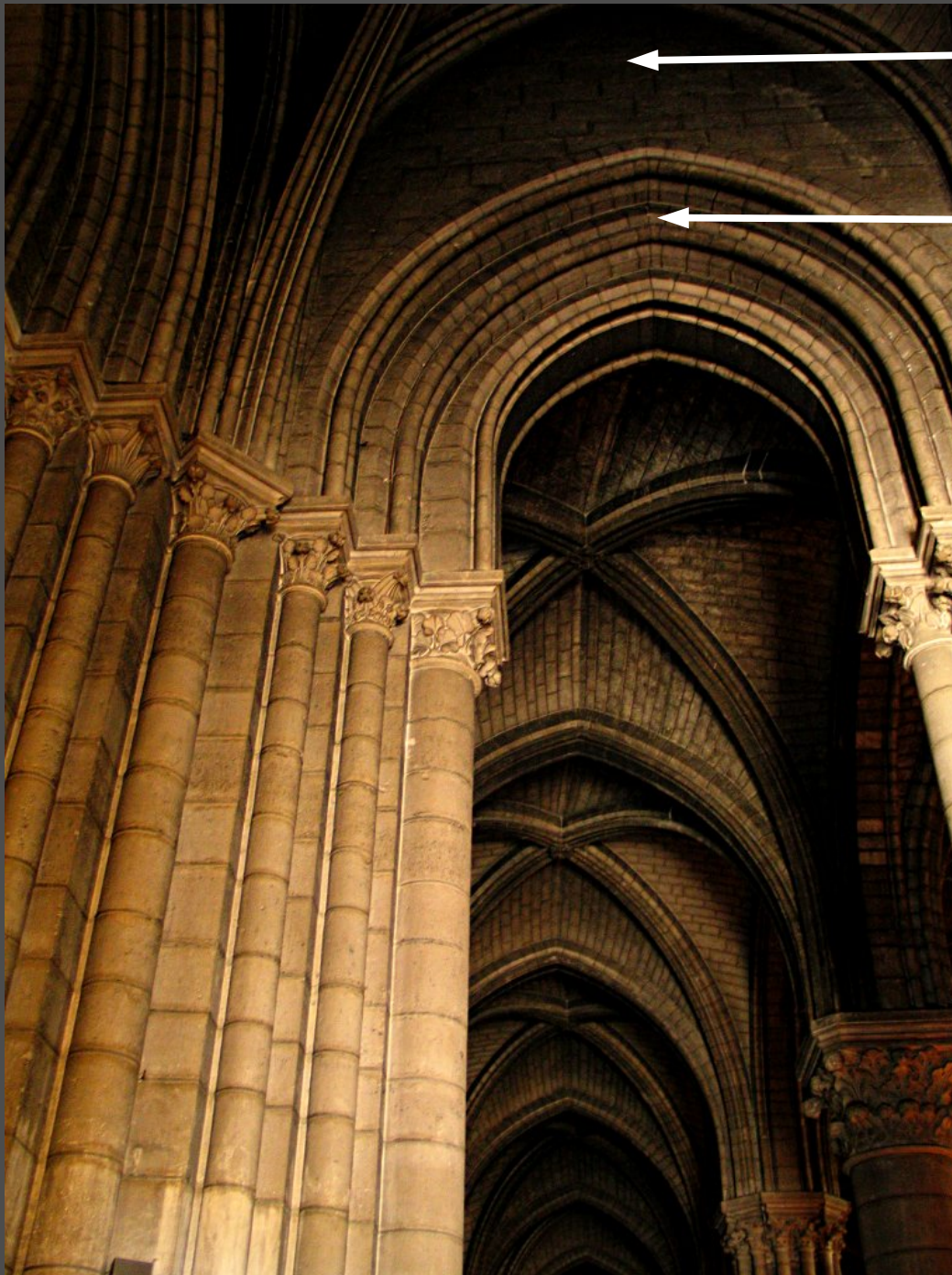
PARIS. La Cathédrale, Notre-Dame

L. P. Phot.



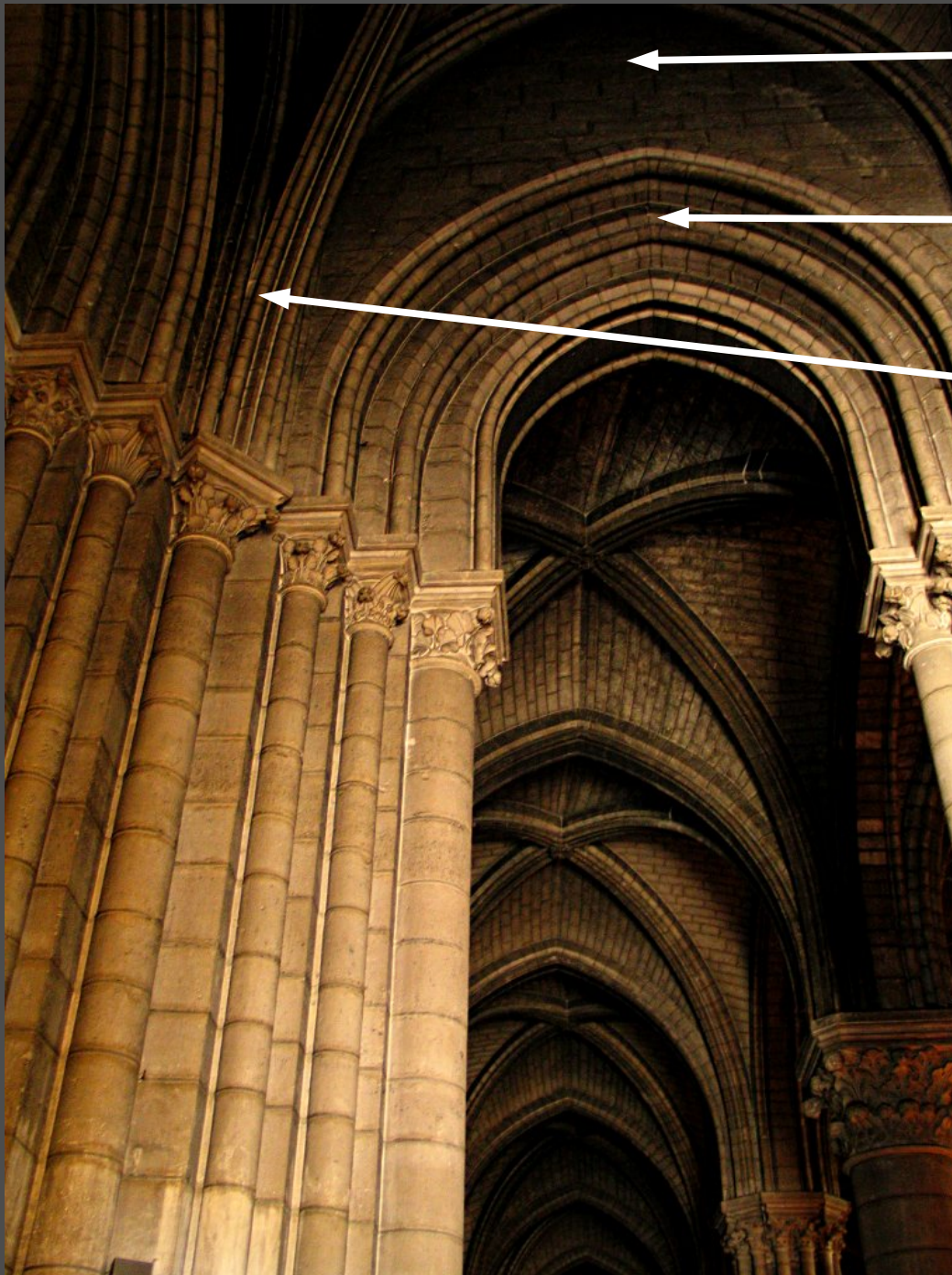


The void



← The void

← Good shape



The void

Good shape

Echoes



← The void

← Good shape

← Echoes

← Positive space



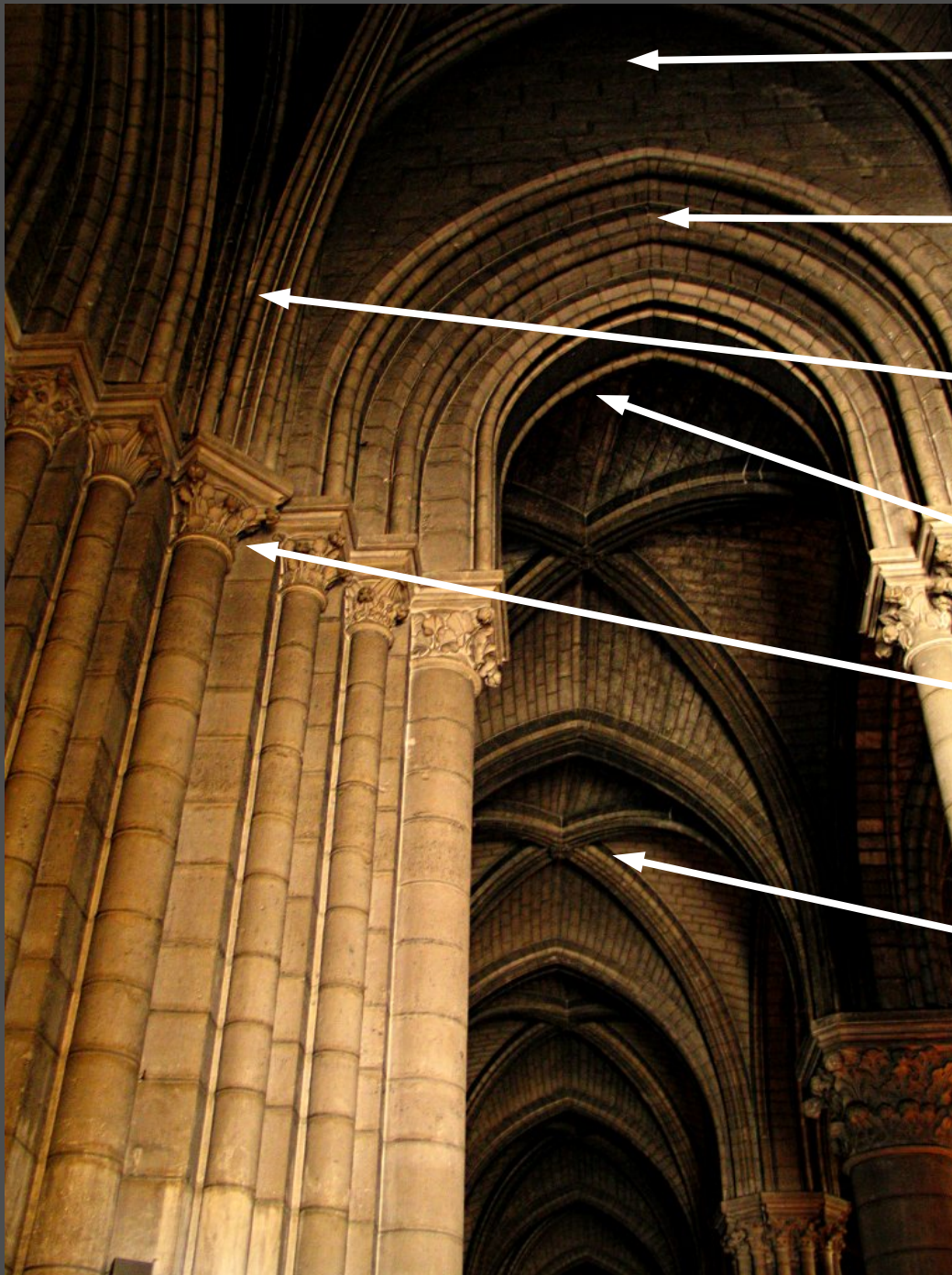
← The void

← Good shape

← Echoes

← Positive space

← Local symmetries



← The void

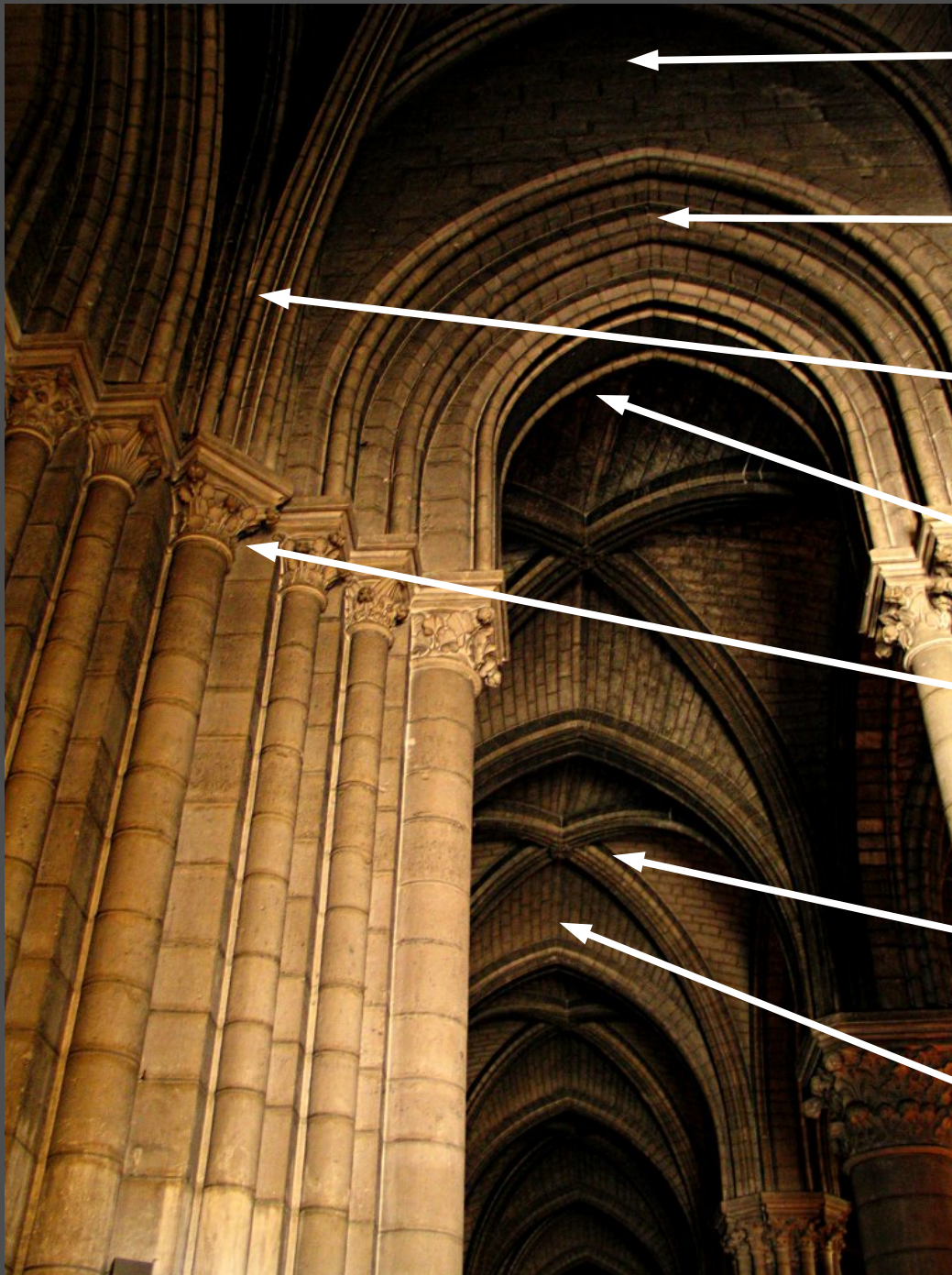
← Good shape

← Echoes

← Positive space

← Local symmetries

← Strong centers



The void

Good shape

Echoes

Positive space

Local symmetries

Strong centers

Roughness



← The void

← Good shape

← Echoes

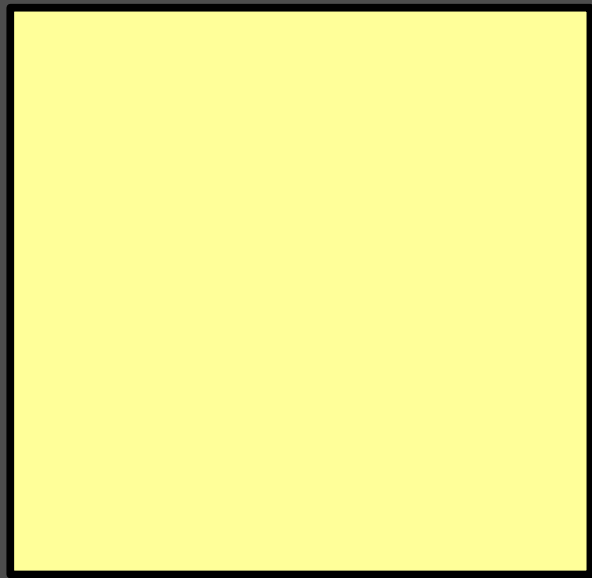
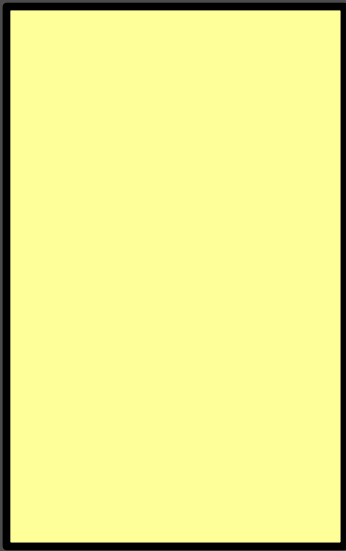
← Positive space

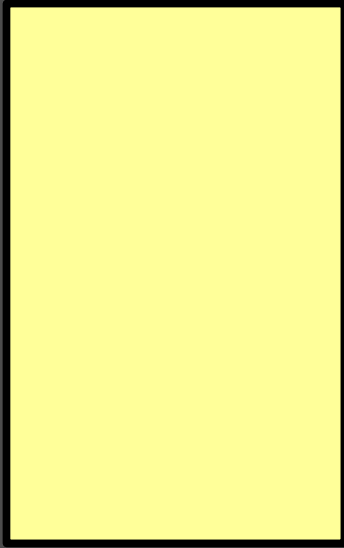
← Local symmetries

← Strong centers

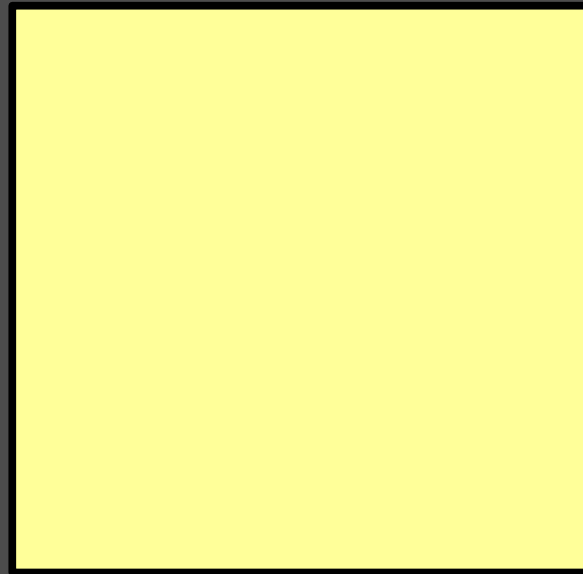
← Roughness

← Alternating repetition





Negative space – amorphous leftovers
Weak centers



GIMP
toolbar

Empty image

Negative space – amorphous leftovers
Weak centers

Layers dialog

Positive space
(convex, enclosed)

Boundary

Strong centers

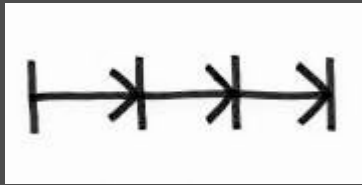
Inkscape menu/toolbar

A schematic diagram of the Inkscape interface layout. It features a central dark gray rectangle labeled 'Drawing area'. This central area is surrounded by yellow rectangular regions. At the top is a horizontal bar labeled 'Inkscape menu/toolbar'. To the right of the drawing area is a vertical bar labeled 'Tool area'. To the left of the drawing area is a vertical bar. At the bottom is a horizontal bar. All these surrounding bars are yellow and have black outlines. The central drawing area is dark gray and also has a black outline.

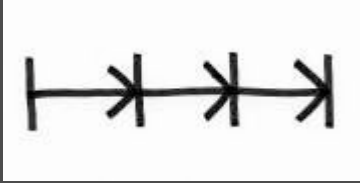
Drawing area

Tool area

Design as computation



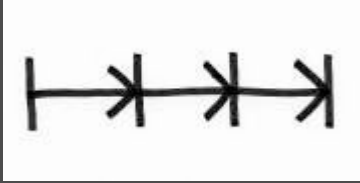
Stepwise: one step at a time



Stepwise: one step at a time



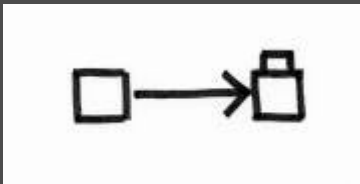
Reversible: test using models, prototypes, trial and error



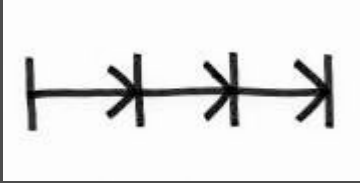
Stepwise: one step at a time



Reversible: test using models, prototypes, trial and error



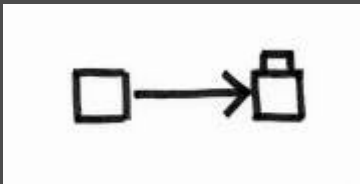
Structure-preserving: each step builds on what is already there



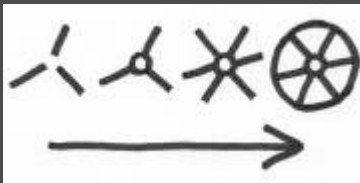
Stepwise: one step at a time



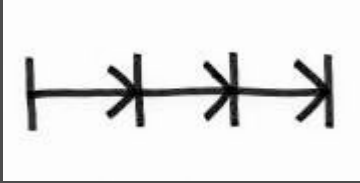
Reversible: test using models, prototypes, trial and error



Structure-preserving: each step builds on what is already there



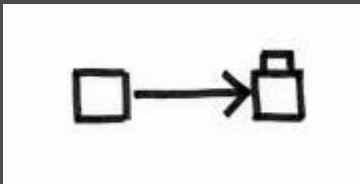
Design from weakness: each step improves coherence



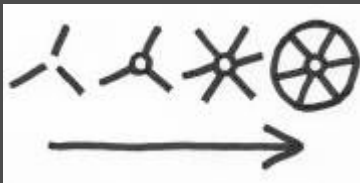
Stepwise: one step at a time



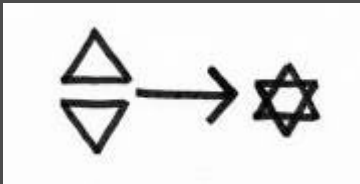
Reversible: test using models, prototypes, trial and error



Structure-preserving: each step builds on what is already there



Design from weakness: each step improves coherence



New from existing: emergent structure combines what is already there

Structure- preserving transformations

A class

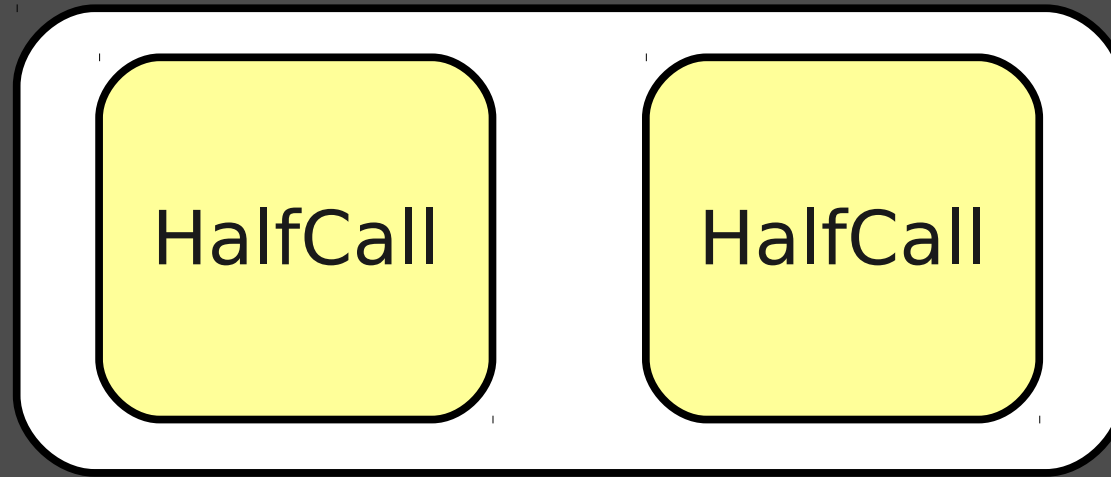
weak, latent center



PhoneCall

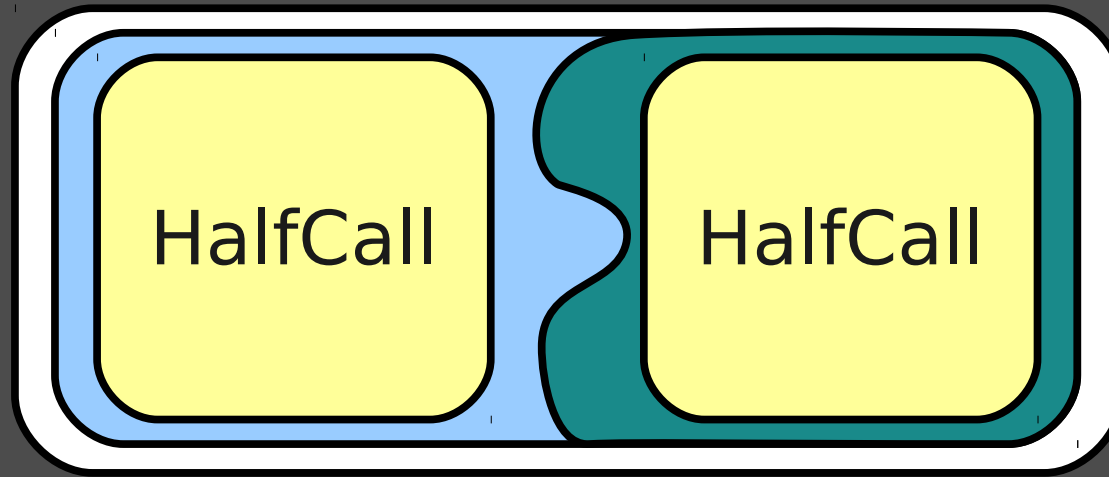
Pattern: Half-object + Protocol

Local symmetry, strong center,
levels of scale



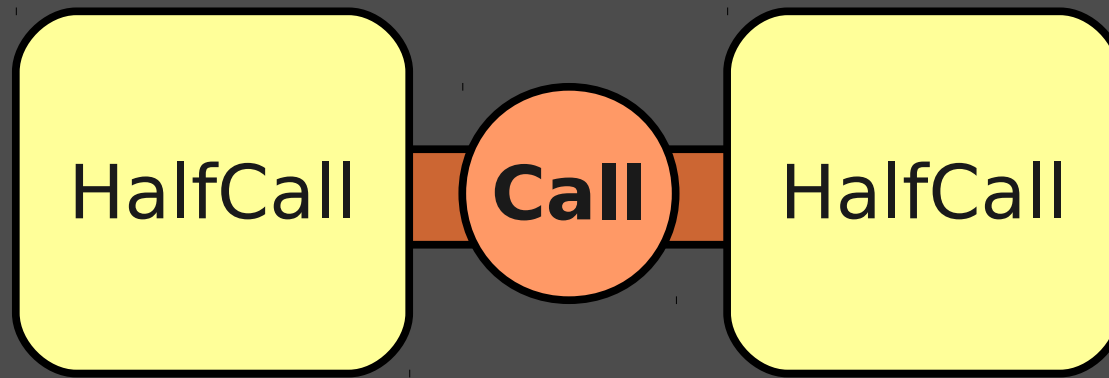
What joins to what?

Local symmetry, levels of scale, boundaries,
deep interlock and ambiguity



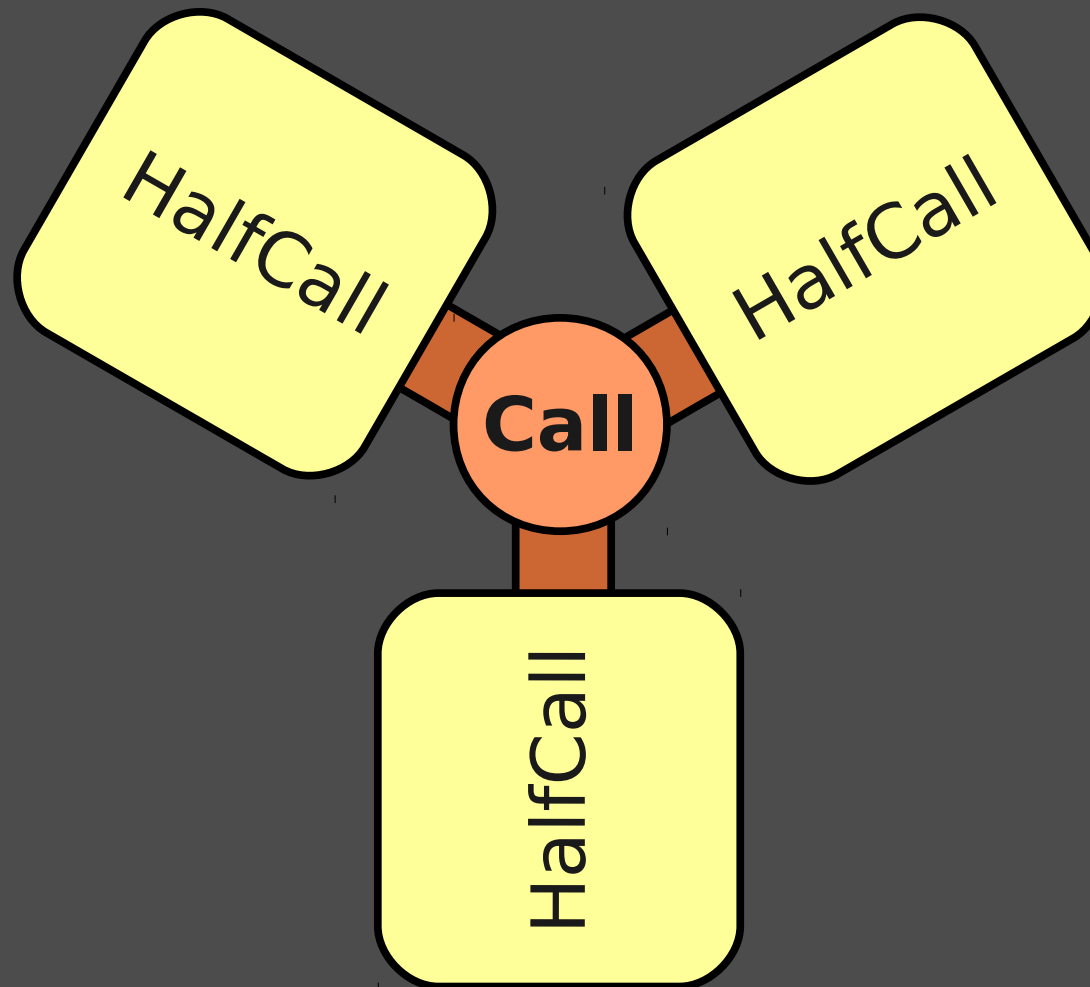
Explicit boundary

Local symmetry, deep interlock,
and this is composable



Composable elements

Multi-way calls, conference calls



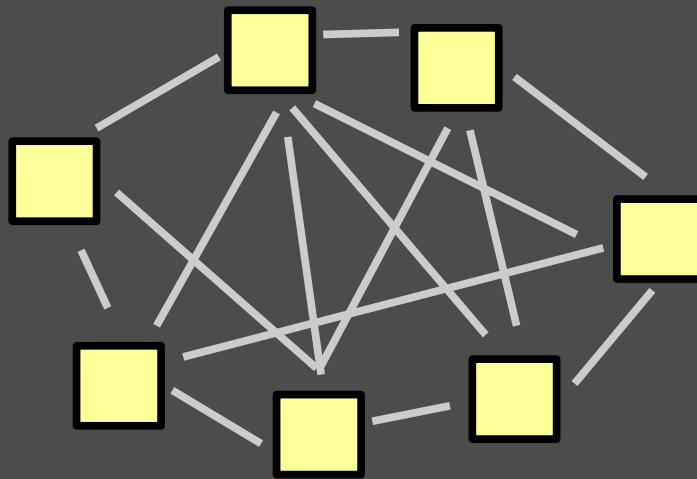
Form languages

Form language (Japan/China)

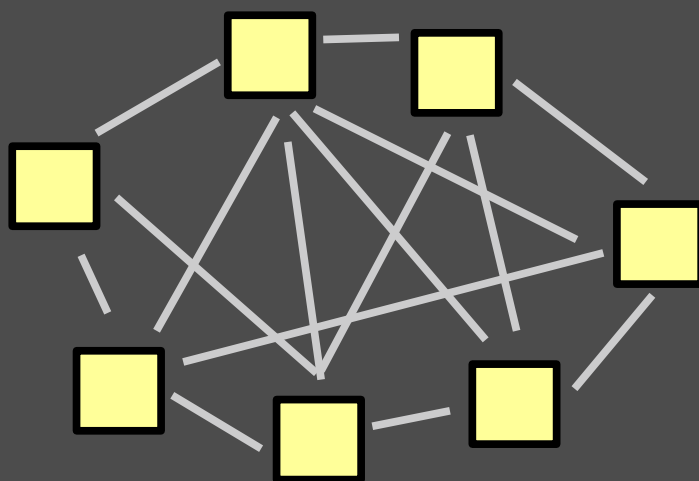


Form language (Germany)

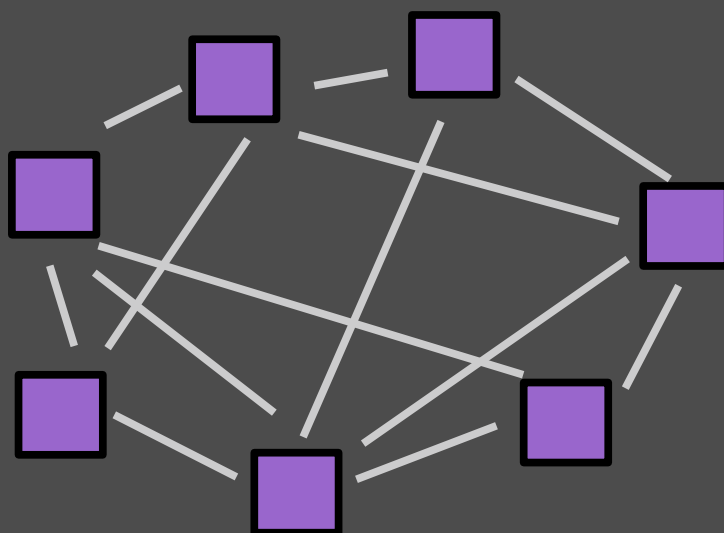




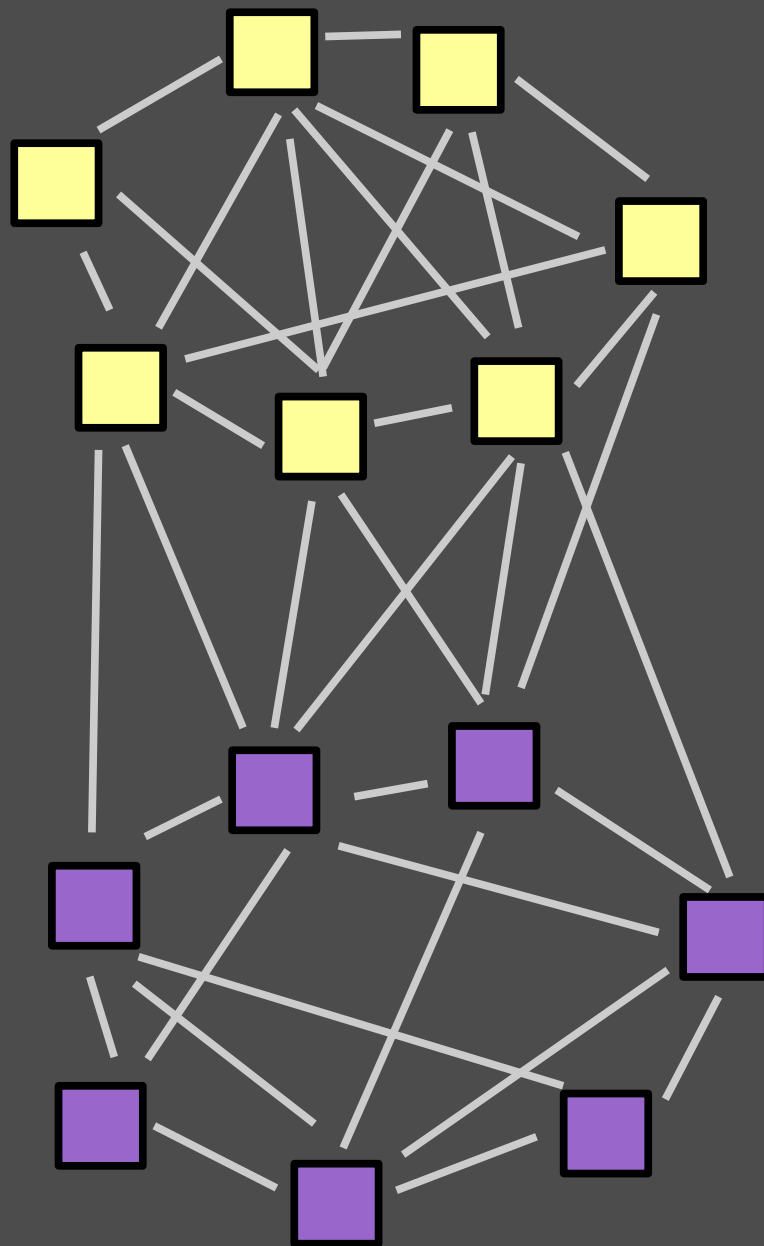
Pattern language



Pattern language

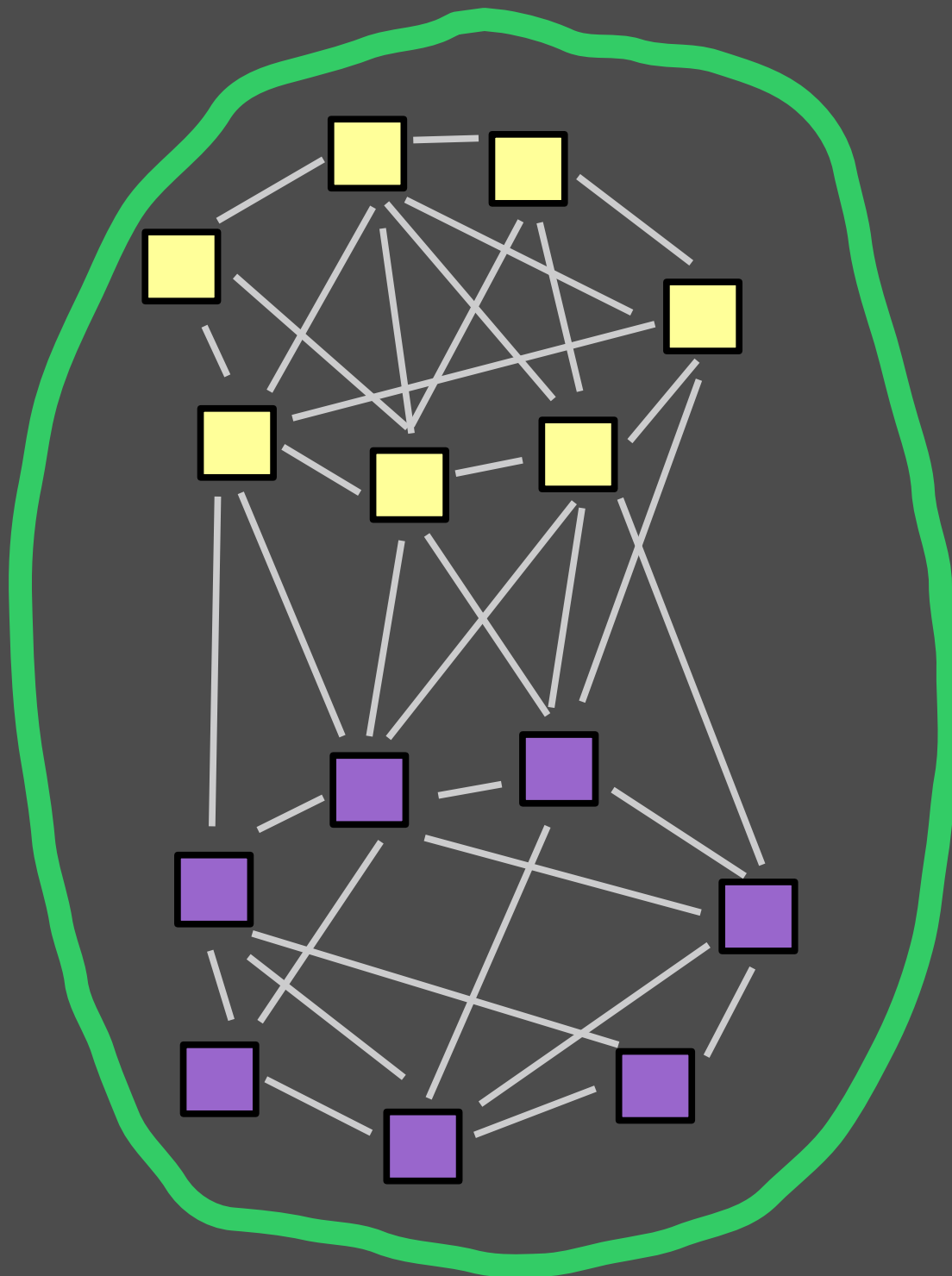


Form language



Pattern language

Form language

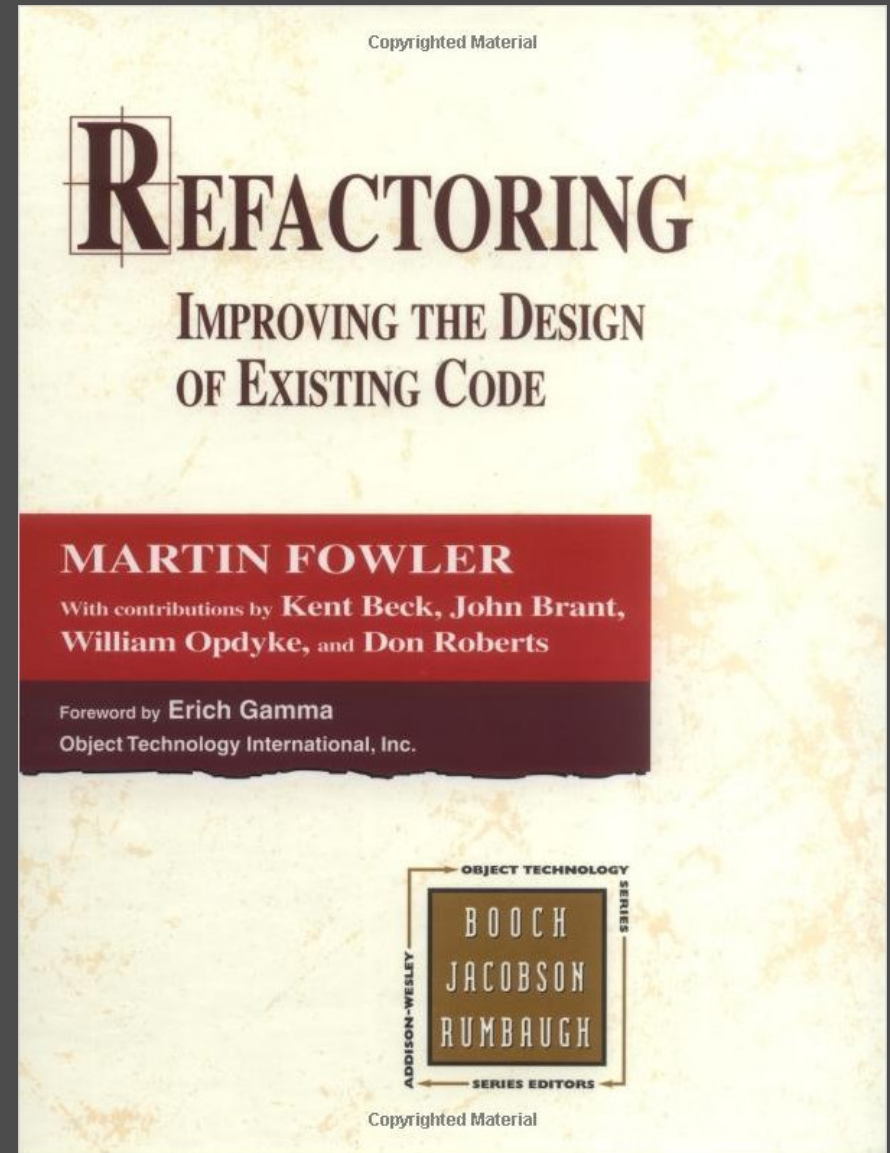


Pattern language

**Adaptive
design method**

Form language

Behavior-preserving transformations



Move common code to function

Strong center

Boundary

Move common code to function

Strong center

Boundary

Add parameter to a function

Roughness

Non-separateness

Move common code to function

- Strong center
- Boundary

Add parameter to a function

- Roughness
- Non-separateness

Replace parameter with explicit methods

- Strong centers
- Simplicity
- Non-separateness
- Deep ambiguity and interlock

Delete a bunch of code

The void

Simplicity and inner calm

Credits

- Rob Hopkins – Picture of Christopher Alexander
- Amazon.com – Book covers
- Oxford University Press – diagrams from “A Pattern Language”
- Hobbit house -
<http://www.tenchford.com/hobbit%20house.html>
- Other pictures – Flickr Creative Commons
- Japan – Lennart Poettering
- Process diagrams – Nikos Salingaros