

```

getAccessTokenFromCode: aCode clientId: aClientId clientSecret:
aClientSecret
  "Complete an OAuth request from {1} with the code {2} and the client secret"
  | accessTokenString response |
  accessTokenString :=
'https://github.com/login/oauth/access_token?client_id={1}&client_secret
={2}&code={3}'
  format: {
    aClientId.
    aClientSecret.
    aCode
  }.
  response := ZnClient new
  url: accessTokenString;
  get;
  response;
  contents.

```

Discovering Alexander's Properties in Your Code

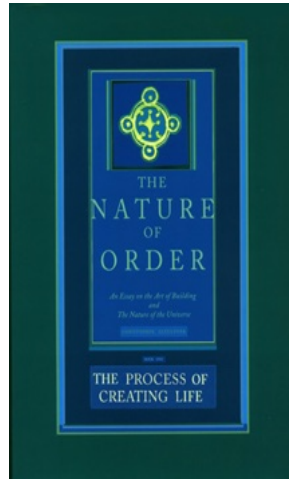
Rebecca Wirfs-Brock

1

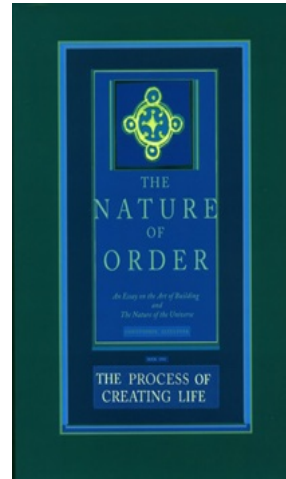


2

Christopher Alexander's Magnum Opus: 4 Volumes on *The Nature of Order*



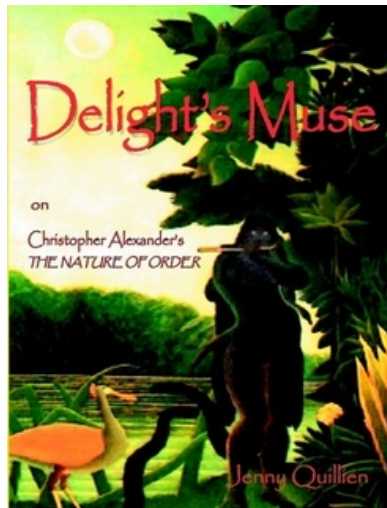
Volume 1: The fifteen properties of things that have life



Volume 2: Unfolding processes for creating "lively" things

3

An Approachable Summary and Personal Interpretation



by Jenny Quillien

4

Core principles

WHOLENESS AND CENTERS AND LIFE

5

Wholeness and Centers

There is a class of entities which I call centers appearing everywhere in space. They appear where they do, as a result of the configuration which appears in the world. Every part of the world, at every scale, has centers appearing in it.

...Although the system of centers is fluid, and changes from time to time as the configuration and arrangement and conditions all change. Still, at any given moment, these centers form a definite pattern. This pattern of all the centers appearing in a given part of space—constitutes the wholeness of that part of space. It is this structure, which is responsible for its degree of life.

—Christopher Alexander

6

Centers

In a user interface, centers include the geometrical entities . . . graphical elements, textual design elements (titles, bullets, paragraphs, sidebars)

In the actual software it depends on what we consider to be the equivalent of geometry, space, and structure: text—the source code itself, the program run trace

James Coplien, “Space: The Final Frontier,” C++ Report, March 1998

7


Alexander’s 15 Properties of Things Which Have Life

- | | |
|---------------------------------|----------------------------------|
| 1. Levels of scale | 9. Contrast |
| 2. Strong centers | 10. Gradients |
| 3. Boundaries | 11. Roughness |
| 4. Alternating repetition | 12. Echoes |
| 5. Positive space | 13. The void |
| 6. Good shape | 14. Simplicity and
Inner calm |
| 7. Local symmetries | 15. Not-separateness |
| 8. Deep interlock and ambiguity | |

8



9



Levels of Scale

relationships between centers (2:1, 3:1)

In software...

- different areas of interest in a design and relationships between the areas
- No class, object, method, or service too big*
- Assemblies of objects into components*
- Systems of software systems...*

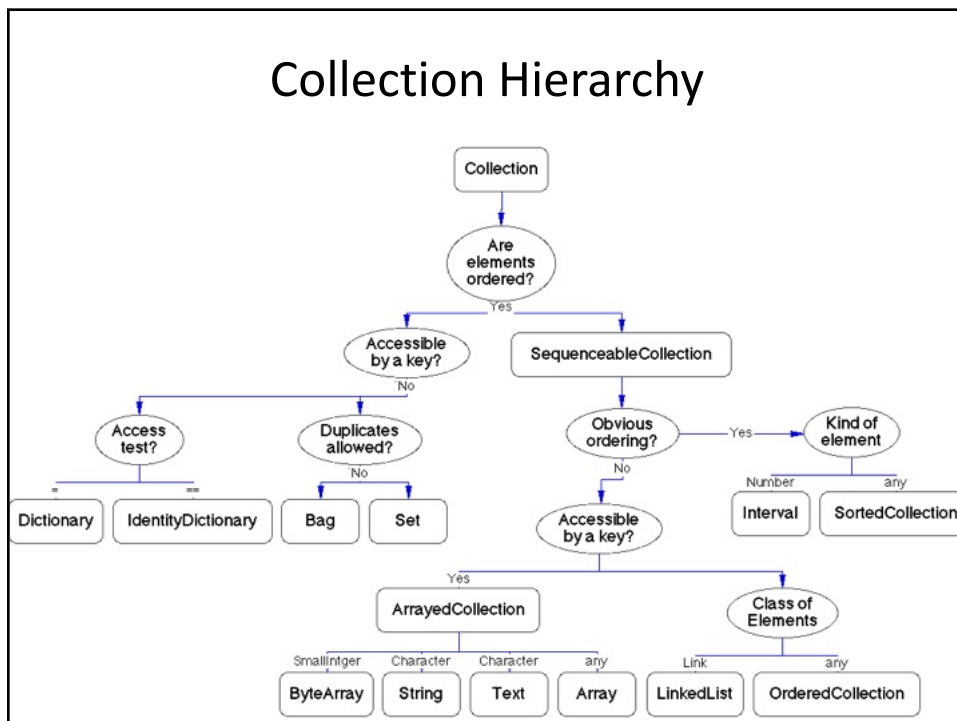
10

Scale in Building upon Core Abstractions

An Example found in the Smalltalk Collection Hierarchy

11

Collection Hierarchy



12

Collection Responsibilities

Three abstract methods as building blocks

add: anObject

**remove: anObject ifAbsent:
exceptionBlock**

do: aBlock

13

Additional Behaviors

Implemented by template methods
which use those building blocks

addAll:

remove:

removeAll:

isEmpty

includes:

ocurrencesOf:

collect:

detect: aBlock

detect: ifNone:

inject: into:

reject:

select:

14

Levels of Scale In your life...

Fearless Change Patterns

- Elevator Pitch – High level summary
- Town Hall Meeting – Interactive invite to share about the new idea and get feedback
- Personal Touch – Custom message reaching out to them on a personal level

Music Patterns

- Concerts/Piece/Movement/Riff/Phrase/Beat
- Orchestra – Strings, Brass, Woodwinds, Percussion
– Concert Violinist, Saxophonist, Conductor

15



Strong Centers

symmetrical shapes
complementary to
the whole

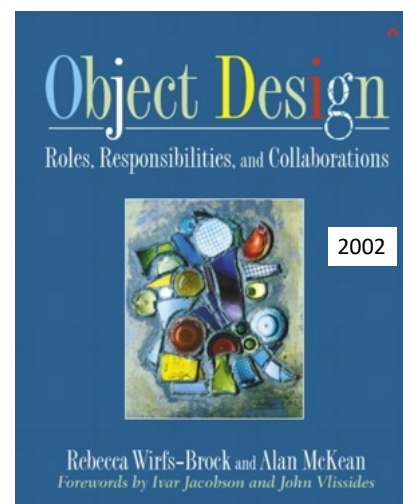
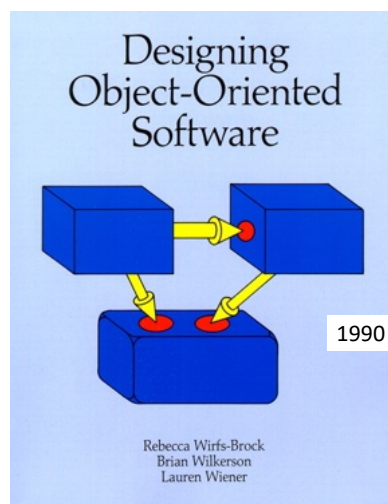
16

Centers in Software

- Well-defined roles and patterns of interactions
- Control Centers
- Domain models: A network of entities, values, aggregate roots. Domains and relationships between them
- Abstract classes and inheritance hierarchies
- Algorithms

17

Responsibility-Driven Design (RDD)



18

RDD Role Stereotypes

knowing, doing, deciding

- Typical behaviors in an object-oriented design
- **Information holder**—knows and provides information.
- **Structurer**—maintains relationships between objects.
- **Service provider**—performs work on demand.
- **Coordinator**—reacts to events by delegating to others.
- **Controller**—makes decisions and directs others' actions.
- **Interface**—transforms information and requests between distinct parts of a software system.

19

RDD Concept: Control Center
Tools for shaping collaboration/interaction style

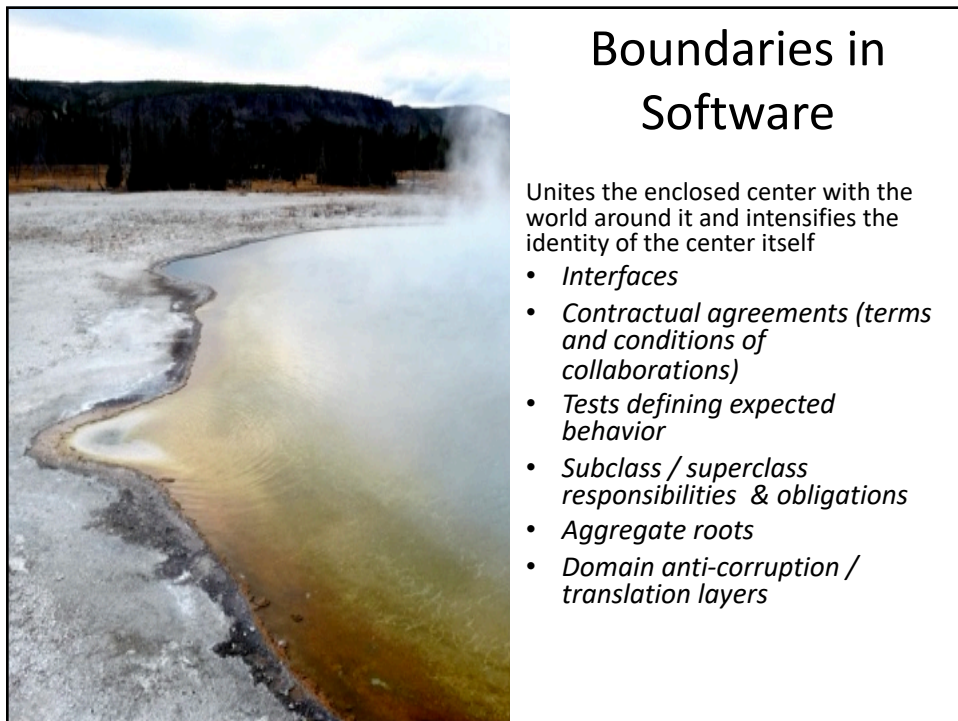
control center—a place where objects charged with controlling and coordinating reside

20



Boundaries
enclose and
strengthen centers

21

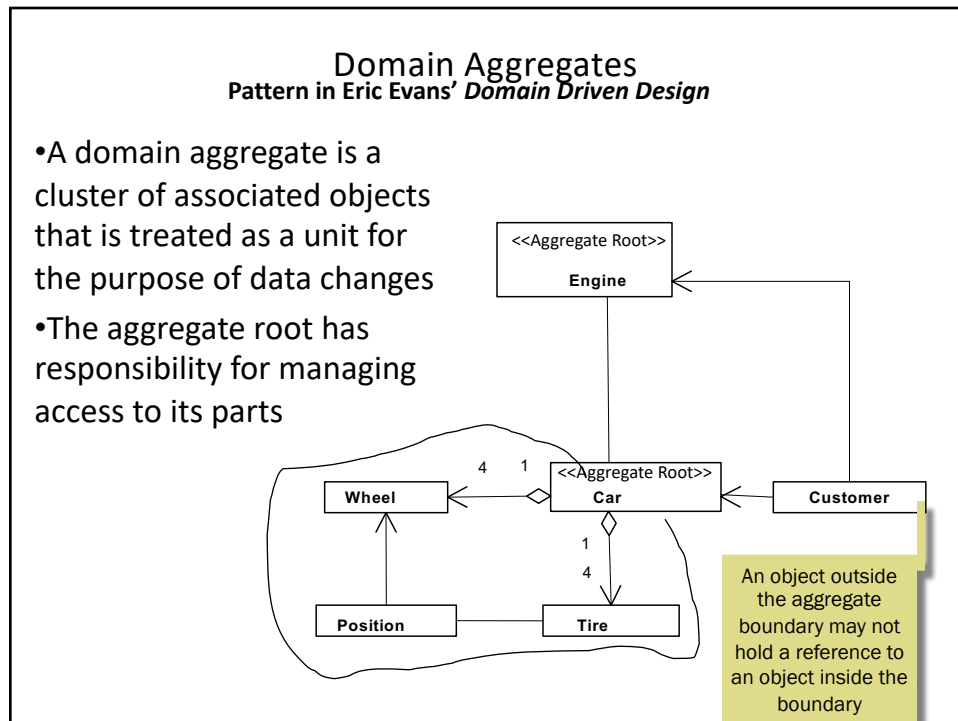


Boundaries in Software

Unites the enclosed center with the world around it and intensifies the identity of the center itself

- *Interfaces*
- *Contractual agreements (terms and conditions of collaborations)*
- *Tests defining expected behavior*
- *Subclass / superclass responsibilities & obligations*
- *Aggregate roots*
- *Domain anti-corruption / translation layers*

22



23




24



The collage consists of three distinct images. On the left, a woman in a bright green athletic shirt and black shorts is running a race, wearing a yellow visor and a bib number 122. In the center, there is a sheet of musical notation with several staves, including a title 'Tempo 2º medio e cantabile' and various musical symbols. On the right, a man in a red shirt is smiling and hugging a woman in a purple shirt from behind.

25

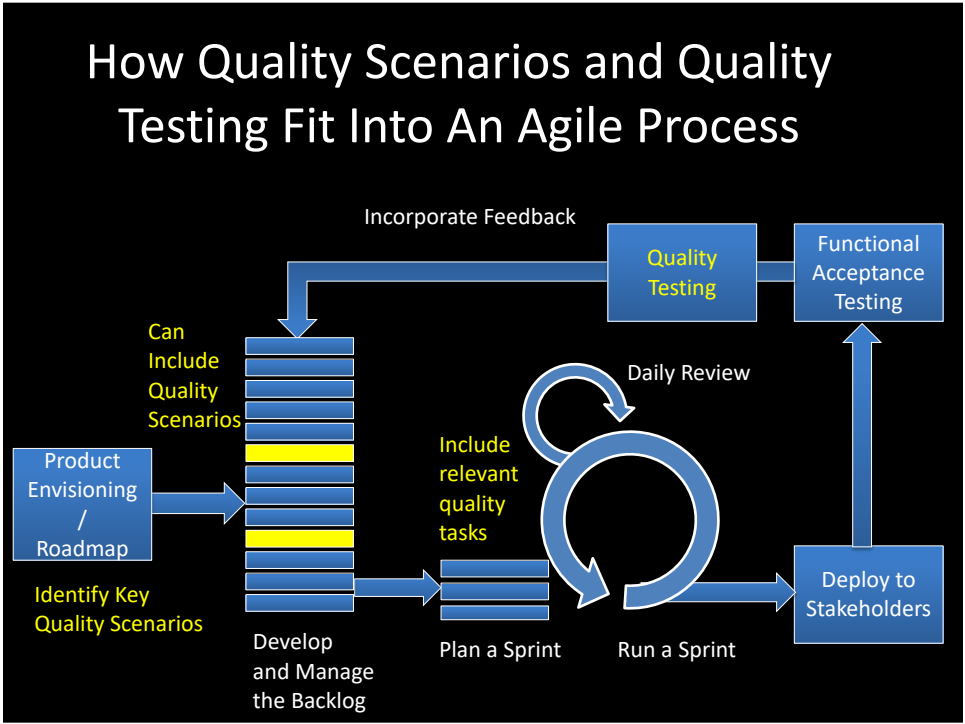


The photograph shows the corner of a red brick building. A white stone archway is visible on the right side. A black street sign with white lettering reads 'PLUMPTRE STREET'. The building features decorative stone elements, including a large corbel on the left.

Alternating Repetition in Code

- *Consistent patterns of collaboration*
- *Domain entities composed of value objects related to other domain entities*
- *Double Dispatch / Delegation*
- *Visitor / Chain of Responsibility / Composite*


26



27



28



Positive Space In Code

Every center has a coherent presence...not a fragment or a bad factoring

- *Objects with a singular purpose*
- *Role stereotypes and good stereotype “blends”*
- *Public and private responsibilities*
- *Tests for expected and exceptional behavior*
- *DRY Principle*
- *Whole Objects / Complete Constructors*
- *Parameter Object*


29



30



31



Local Symmetries

work to create coherence

- *Symmetrical behaviors*
- *Common and consistent naming*
- *Methods containing same level of code detail*
- *Collections*
 - *add: remove:*
 - *select: reject:*
- *Stacks*
 - *push / pop*
- *Do/Undo*

32



Good Shape
recursive, compact
coherent centers

33



Good Shape

a shape that comprises recursive
compact coherent centers, each
exhibiting characteristic properties

- *Roles and patterns of collaboration*
- *Adaptor, Decorator, Façade, Layers, Pipes, Filters, Services,*
- *Sub-assemblies, Modules*
- *Domains, Separation of Concerns*

34

```

render() {
  var completed = Todos.completed().length;
  var remaining = Todos.remaining().length;

  if (Todos.length) {
    this.$main.show();
    this.$footer.show();

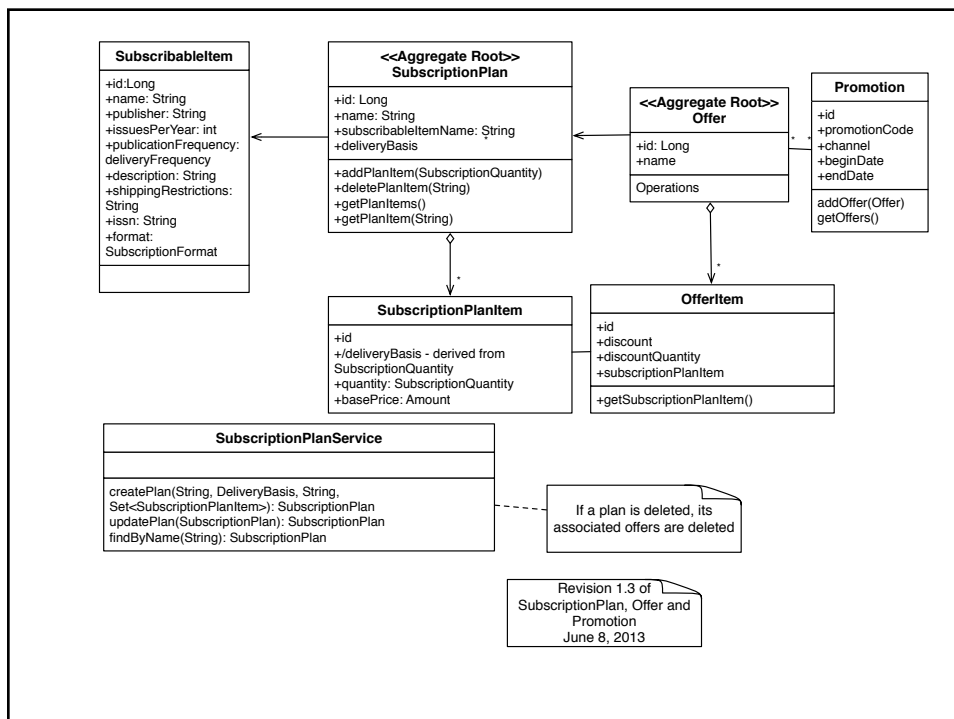
    this.$footer.html(
      this.statsTemplate({
        completed, remaining
      })
    );

    this.$('#filters li a')
      .removeClass('selected')
      .filter('[href="#"/{TodoFilter | |}"]')
      .addClass('selected');
  } else {
    this.$main.hide();
    this.$footer.hide();
  }

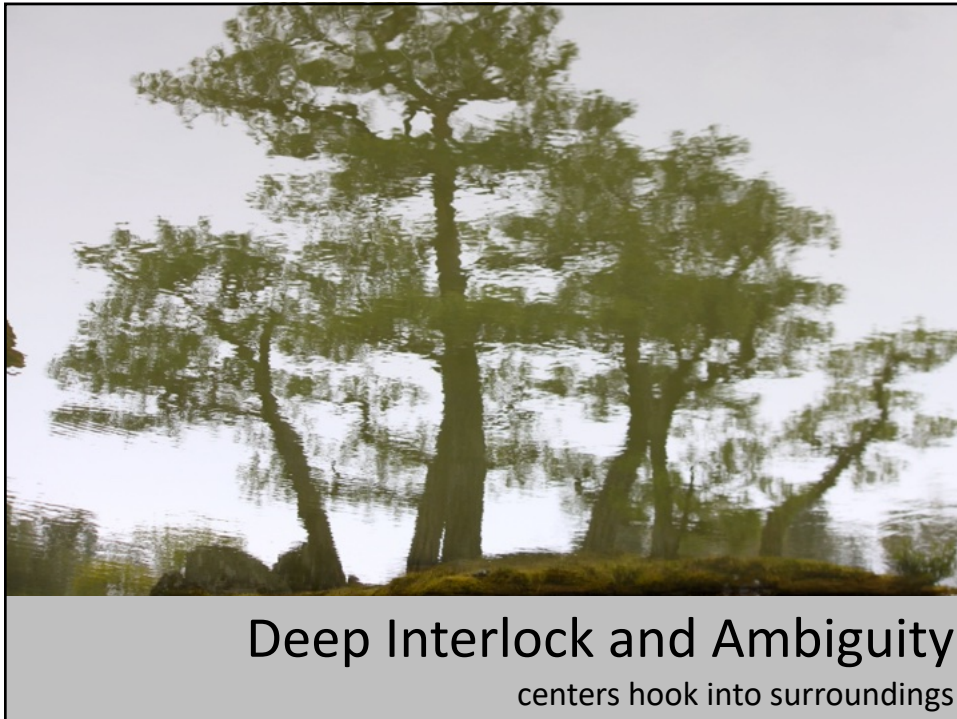
  this.allCheckbox.checked = !remaining;
}

```

35




36



37



38



Deep Interlock and Ambiguity in Code

the interface zone, both system and context, where centers hook into their surroundings

- *Collaborations between roles*
- *Interface and Contract Definitions*
- *Dynamic Client-Supplier Relationships*

39



Contrast

centers created by strong boundaries

40



41



42

Roughness in Software

- My programming differs from yours, even though we agree to the same “style guidelines”.
- Hand-crafting means you don’t blindly apply a pattern without thinking of how it should be adapted in *this* situation.
- We’re always tweaking things...for performance, scalability, aesthetics...

43



44

Echoes in Programs

- We find and repeat:
 - Recursion
 - Interfaces
 - Intentional names
 - Relationships, complex and basic structures
 - Ways of decomposing responsibilities (helper methods, classes, components, services)
 - How we handle errors and exceptions

45



46



47

Simplicity and Inner Calm in Code

- *refactored, clean code, easy to understand*
- *spare use of programming language frills*
- *lack of excessive features in a framework*
- *Fold out interfaces*
- *“Works out of the Box”*



48

Not Separateness

when a center having deep life evokes a feeling of connectedness to what surrounds it and is not cut off, isolated, or separated



49

Group Exercise



**BREAK UP INTO GROUPS AND PICK
AN AREA OF YOUR LIFE AND A FEW
PROPERTIES AND GIVE EXAMPLES
OF THEM**

50

Relating

“The key thing about these many different wholes we see, is that each of them has a relation with us, me, you. Each shape is made in such a way that you can establish a relationship with it; indeed, you want to establish a relationship with it.”

—Christopher Alexander, *The Luminous Ground*

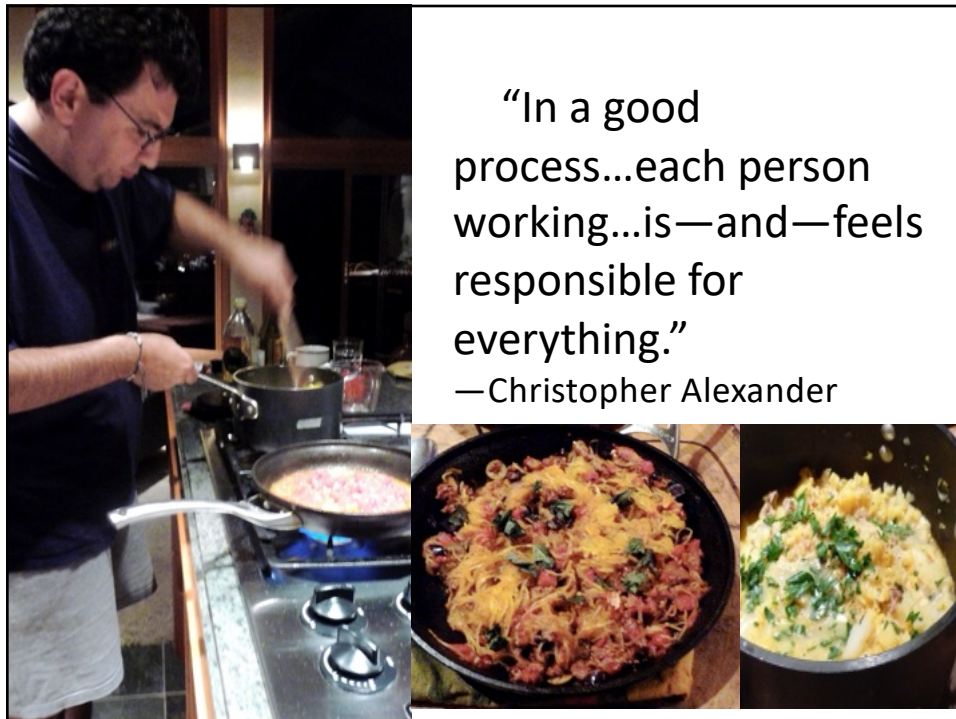
51

Living Structures

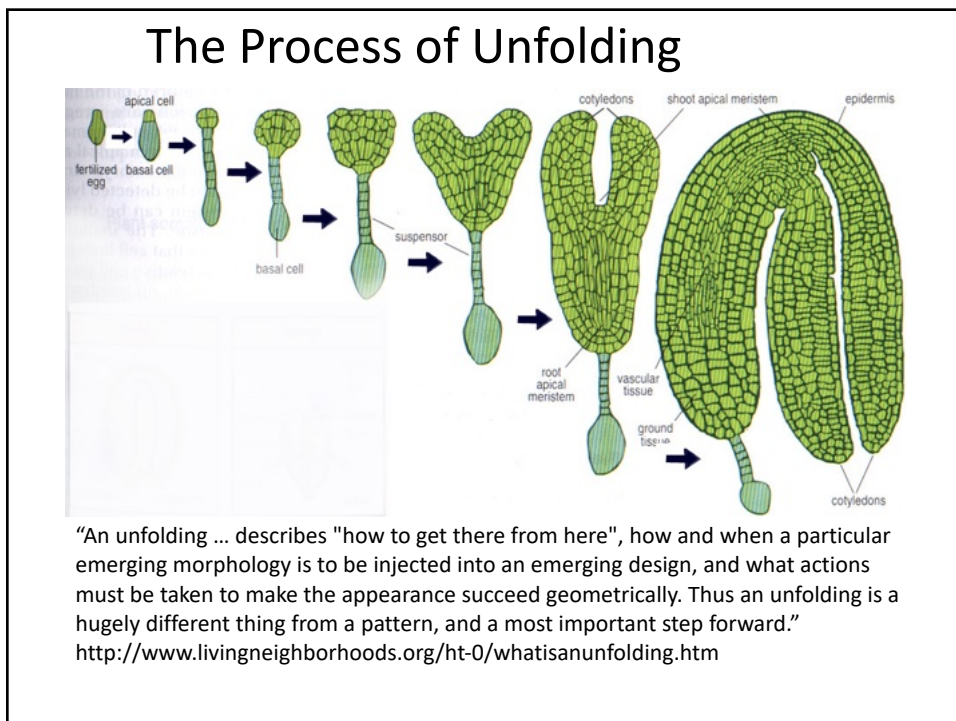
...make us
comfortable
...fit their
environment
...are made by
hundreds of small
acts



52



53



54



55

Acknowledgements

- Rebecca Wirfs-Brock photographs of Kyoto, Portugal, Greece, Sweden, and Opal Creek, Amity, and Sherwood, Oregon USA
- Joe Yoder is a friend, collaborator and fellow pattern / Alexander enthusiast whose ideas have stimulated my thinking.
- Richard Gabriel is a thinker and doer, and inspiration too.
- Allen Wirfs-Brock photographs of Oslo, Greece, and Jenny
- Brian Foote photo of Joseph Yoder/Mary Lynn Manns
- Code by TasteJS <http://blog.tastejs.com/rewriting-a-webapp-with-ecmascript-6>
- Code by Jon Tedesco <http://www.jontedesco.net/2013/05/18/github-oauth-tutorial-gists/>
- Dodgson (Stephen Dodgson 1924) - Etude - caprice (to Joan York) - guitar sheet music notes

56



57