Pattern Languages

• How can we organize what we know about design?
  … So that it provides useful guidance in design work?
• Design guidelines
  – Abstract apple pie, hard to instantiate
• Examples of good results
  – Often depend on application details, hard to generalize
• Design theories?

Origins of pattern languages

• Christopher Alexander
  – Notes on the synthesis of form (1964)
  – “new design methods” - Reitman, Rittel, Jones, Simon
• A Pattern Language (1979)
  – names for 253 design archetypes, at various scales
• actual impact in architecture somewhat in doubt!
• Wide impact in software and UI design
  – OO design patterns: e.g., http://c2.com/ppr/
  – software process patterns
    http://www.bell-labs.com/user/cope/Patterns/Process/
  – HCI patterns
    http://www.mit.edu/~jtidwell/interaction_patterns.html
Example: Street Café

- example: picture, description of experience
- relationship to larger/smaller scale patterns
  - larger scale - Identifiable Neighborhood, Small Public Squares, Activity Nodes, ...
  - smaller scale - Opening to the Street, Place to Wait, Different Chairs, Stair Seats, Canvas Roof, ...
- rationale (viz the experience of the space)
  - low-key social mixing, safe to relax, nurse a beer, ...
- implementation
  - specializing lower-level patterns (wide Opening to the Street, make terrace a Place to Wait, …)

How you use a Pattern Language

(following Alexander)

- “Spreading activation” in a pattern network
  - select best pattern
  - identify all patterns that support that pattern, all patterns that support each of them, …
  - identify all patterns supported by any selected pattern (and patterns they support, etc.)
  - modify any pattern as appropriate

- Not rules to be followed
- Not templates to be “instantiated”
How pattern languages can help

- Capture collective wisdom
  - Combines design rationale with suggested actions
  - Can grow by specialization for contexts
- *lingua franca* for design
  - Standardize terms and concepts
  - Ease participation of clients in design
- Reuse and extend ideas instead of just copy
  - Organize solutions to common problems
  - “Think outside the toolkit” -- slightly higher level of abstraction than particular widgets
- Converge on designs that *work* for people
  - “The quality without a name”

Elements of typical pattern schema

- **Examples:** of pattern in application
- **Context:** in which pattern is employed
- **Problem:** that pattern solves or addresses
- **Forces:** requirements/tradeoffs that constrain possible solutions
- **Solution:** the pattern
- **Resulting context:** having employed the pattern, what do you need to do now?
- **Notes:** any useful annotations
Multiple Coordinated Views

- Examples: Smalltalk debugger, View Matcher, and many other programming environments.
- Context: complex analysis tasks
- Problem: understand/manipulate complex and dynamic structures
- Forces: User needs to build a comprehensive causal representation of underlying data. Seeing dynamic inter-relationships facilitates this.
- Solution: show multiple views of underlying data and coordinate so that a change in one view is reflected in all other views
- Resulting context: support for multiple coordinated views encourages a more interactive/opportunistic approach to problem analysis, which can lead to non-optimal solutions.
Choice from a small set  
(from Tidwell)

- **Examples:** Set of radio buttons; Combo box (drop-down list); "Circle one: Mr / Mrs / Ms" on a paper form

- **Context:** Display/set a value out of a small set of possible values (10 or fewer).

- **Problem:** How should the artifact indicate what kind of information should be supplied?

- **Forces:**
  - User should see all the possible values, to put the actual value in context.
  - If the user needs to set the value (not just look at it), they should know what choices are available.
  - Small numbers of things can be taken stock of quickly, and don't take up much space.

- **Solution:** Show all possible choices up front, show clearly which choice(s) have been made, indicate clearly whether one or several values can be chosen. Provide a choice for "Other" or "None of the above," if that will ever be an issue -- never prevent the user from providing correct information.
• Resulting Context: Good Defaults may let the user look at the default value, judge it to be OK, and move on without even bothering to set the value. If the choices are pictorial, or are cryptic in some other way, Short Description may be needed to describe the choices further.

• With physical or electronic artifacts, a single selection can be enforced by causing the previous choice to "unselect" when a choice is made. Old car radios and GUI radio boxes do this.

• Notes: For a small number of choices, it is often pointless to hide choices, as in a combo box -- if it won't cost a huge amount of space; show all the possibilities so the user can see them without going through an extra step to reveal them.

Extending the pattern concept

• Levels of patterns
  – Conceptual model patterns ⇒ detail patterns
  – Genre patterns
• Layout versus interaction patterns
• Patterns for civic computing
UI Genres as patterns

• Central work surface
  – Homepage, application window
  – Not good for multi-application sessions

• Tiled work surface
  – HTML frames, multiple Emacs buffers
  – Spatially inefficient, but can’t lose windows

• Stack of work surfaces
  – Tabbed dialogs/notebooks
  – Breaks down after more than a few tabs

• Pile of work surfaces
  – Mac/Windows “desktop” OS

Patterns of Interaction in Mateo
(Erickson)

• bypassed by the highway to the Outer Banks
  – economic decline, especially in tourism
  – development; preserving town’s character

• surveys and interviews with residents
  – “revealed” usual small-town values

• “behavior mapping” (ethnography)
  – hanging out at the docks, debating politics in Dutchess Restaurant
  – place-based rituals of everyday life
Sacred Structure of Mateo

• residents ranked “important places”
  – marshes, restaurant, parking lot, hand-made sign

• consequences
  – recognized that sacred structure was shared through the community
  – legitimized talking about it
  – made preservation discussion more concrete

Interaction Patterns

• wedding
  – Collecting Bride/Bridesmaids Personal Objects
  – Taping Cards to Gifts

• office
  – Staying Synchronized ⇒ Mapping Out the Day, Closing the Loop
  – Closing the Loop ⇒ Scanning, Prioritizing, Reminding
  – Mapping Out the Day ⇒ Focused Work

• design consultancy
  – Maintaining Mutual Awareness ⇒ Blanket Email, Doing a Walkabout, Open Offices,
Community-based IT workforce development

Patterns as a design language

- concrete prototypes vs abstract principles
  - more accessible (but harder to generalize?)
- socially grounded
  - emphasize human activity and experience
- incorporate values
  - Individually-Owned Shops, Bike Paths and Racks,
- supports piecemeal adoption & use
  - don’t have to adopt/use the whole framework, inherently open-ended