Pedagogical and Elementary Patterns

Joe Bergin

Pace University

http://csis.pace.edu/~bergin

Patterns Overview

Thanks to Eugene Wallingford for some material here.
Jim (Cope) Coplien also gave advice in the design.

Patterns--What they DO

- Capture Expert Practice
- Communicate Expertise to Others
- Solve Common Recurring Problems
- Vocabulary of Solutions.
- Bring a set of forces/constraints into equilibrium
- Work with other patterns

Where Patterns Came From

- Christopher Alexander --A Pattern Language
  - Architecture
    - Structural Relationships that recur in good spaces
- Ward Cunningham, Kent Beck (oopsla ‘87), Dick Gabriel, Jim Coplien, and others
- Johnson, Gamma, Helm, and Vlisides (GOF)
    - Structural Relationships that recur in good programs

Patterns Branch Out

- Software Design Patterns (GOF...)
- Organizational Patterns (XP, SCRUM...)
- Telecommunication Patterns (Hub…)
- Pedagogical Patterns
- Elementary Patterns

Patterns--What they ARE

- A Thing
- A Description of a Thing
- A Description of how to Create the Thing
- A Solution of a Recurring Problem in a Context
  - Unfortunately this “definition” is only useful if you already know what patterns are

Patterns--What they ARE (2)

- Structural relationships between components of a system that brings into equilibrium a set of demands on the system
- A way to generate complex (emergent) behavior from simple rules
- A way to make the world a better place for humans -- not just developers or teachers...

Elements of a Pattern

- Problem
- Context
- Forces
- Solution
- Examples of Use (several)
- Consequences and Resulting Context

Exercise

- Problem: Build a stairway up a castle tower
- Context: 12th Century Denmark. The region is often at war
- Forces: Confined space. Must be defended against attack from below. Warriors are mostly right handed.
- Solution:

Exercise

- Problem: Design the intersection of two or more rural roads
- Context: Denmark
- Forces: Low maintenance. Land is cheap. Electricity is expensive to deliver. Traffic is light. Don’t cause an unnecessary stop.
- Solution:

Alexander Example

- Light On Two Sides of Every Room (159)
- Major rooms have been placed...
- People gravitate to such rooms (converse)
- Shadows, glare, softer light gradient...
- Locate every room so that it has outdoor space adjacent on at least two sides...

Alexander Example (2)

- Light on Two Sides of Every room:
- A Thing -- a configuration of rooms with the property
- A Description of the Thing -- a picture or paragraph describing it.
- A Description of how to build the thing-- Place rooms in the corners of the building. Juxtapose small rooms with large ones. Take care not to destroy the roof layout. Make the windows open onto something beautiful.

GOF Example

- Composite
- Building a complex part-whole hierarchy
- Many parts must be composed flexibly
- Tradeoff simplicity and flexibility
- Structure objects so that a container of components is also itself a component

Composite(2)

- A Thing -- Panel in Java or (3 * (4 +5))
- A Description of the Thing -- An object in a program that behaves like other objects but which also contains other objects with that behavior
- A Description of how to build the Thing -- Create a class that implements the desired interface. Give the class an instance variable that is a collection of objects that implement the interface. Implement the methods by delegating to the elements of the collection.

More Examples

- Grand- Patterns in Java
  - Delegation
  - Singleton
  - Observer
  - Decorator

Forms for Writing Patterns

- Alexandrian
  - See Seminars
- Outline (sometimes called GOF)
  - See Grand

- Outline is easier and a good place to begin
- Alexandrian is more informative and literate

Alexandrian Form (APL)

1. PATTERN NAME*
Picture giving architectural example
Introduction giving context
 ***
Problem description
Essence of Problem (1-2 sentences)
Body of problem (longest section)
Solution -- Stated in form of an instruction
Diagram giving solution graphically
***
Resulting context

Modified Alexandrian Form

1. PATTERN NAME*
Introduction giving context
 ***
Problem description
Essence of Problem (1-2 sentences)
Body of problem (longest section)
Solution -- Stated in form of an instruction
Diagram giving solution graphically
Examples of use
***
Resulting context

Patterns--What they are NOT

- A way to make software cheaper
  - Instead of software: courses, organizations...
- A way to make software faster
- Surprising to experts (mostly)
- Invented by their authors
- Just a literary form
- Stand alone incremental improvements

Patterns--What they are NOT (2)

- Just GOF
- Just OOP
- Just programming templates
- Just tricks
- Extra material for students
- New material for students

Patterns--What they are NOT (3)

- If there is only one way to do something it isn’t a pattern. (context and forces are key)
- If it doesn’t properly bring the applicable forces into equilibrium, it isn’t a pattern.
- If there is no agreement among experts that the solution is “best,” it isn’t a pattern.

Overheard

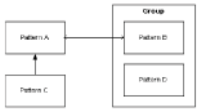
- Why don’t you just have an immutable singleton null object iterator mediate between your composite decorated facade and the conoculator valve?
- OR-- maybe bridge the flyweight proxy prototype with a factory method mememto adapter?

Pattern Languages

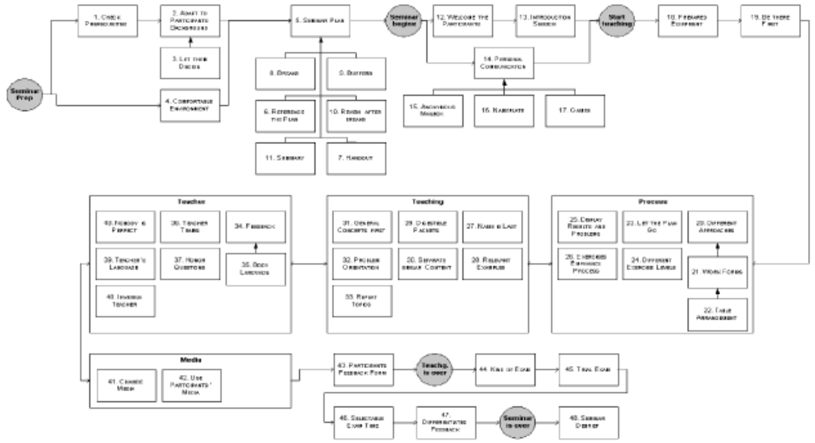
- Patterns seldom occur alone
- Seen properly they are part of pattern languages
- It is the languages that provide the emergent behavior, not the individual patterns
- Context of a pattern is the consequences of one or more other patterns.
  - Patterns are applied in sequence

Pattern Languages (2)

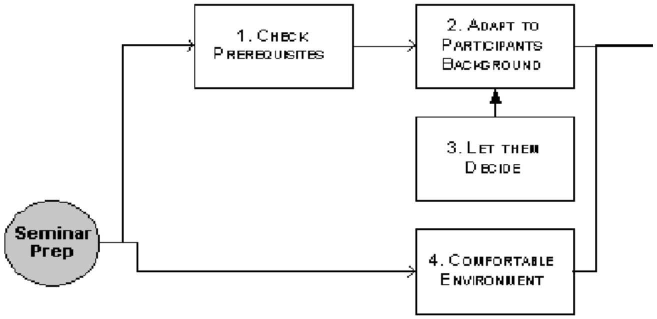
- Patterns are defined at different time or space “scale”
- Patterns at different scale are all of similar complexity
- Large scale patterns (towns)as simple as small scale (pictures on your wall)
- Work from large scale down to small within a given context



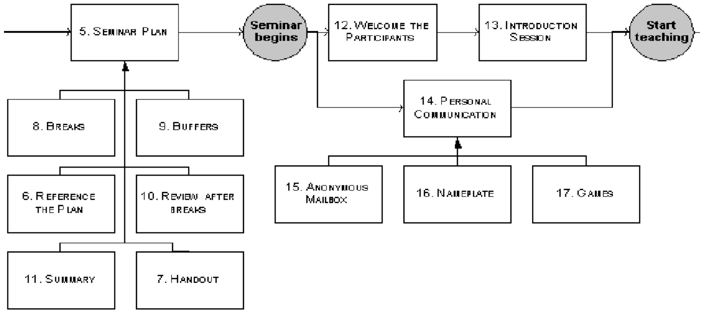
Seminars: Fricke/Voelter



Seminars


1. Check Prerequisites
2. Adapt to Participants Background
3. Let them Decide
Seminar Prep
4. Comfortable Environment

Seminars



Pattern Languages are NOT

- Just catalogs of patterns
  - Patterns in a language work together to solve a complex problem whose parts are solved by the individual patterns. The Goodness and Liveness emerge from the interactions.
  - In Alexandrian terms, from the spaces defined by the elements in the pattern. Spaces in which humans live. (Architect positions walls but defines spaces between the walls.)

Patterns Community

- The patterns community tries hard to be supportive of its members
- There are many activities that work to this end
- PLoPs, Lists, Web sites
- Patterns Process

Wiki Wiki Web

- Ward Cunningham maintains a virtual meeting place for all patterns people at
  - http://c2.com/cgi/wiki/
- The Wiki Wiki Web is an interactive web site to which anyone can modify any page.

Other Views-Resources

- Brad Appleton's overview of patterns
  - http://www.enteract.com/~bradapp/docs/patterns-intro.html
- More On Christopher Alexander
  - http://g.oswego.edu/dl/ca/ca/ca.html
- Jim Coplien's Home Page
  - http://www.bell-labs.com/~cope/
- Hillside Group
  - http://hillside.net/patterns/

More Resources

- Introducing Patterns into Organizations
  - Mary Lynn Manns and Linda Rising
  - risingl@acm.org
  - Linda is spending this year in Copenhagen
  - She is also the author of the Patterns Almanac and the Patterns Handbook
- http://www.cs.unca.edu/~manns/intropatterns.html

Elementary Patterns

Overview

http://www.cs.uni.edu/~wallingf/patterns/elementary/

(There is a mailing list also)

Current Key Players

- Eugene Wallingford
- Joe Bergin
- Robert Duvall
- Owen Astrachan
- Alyce Brady
- Ed Epp
- Rick Mercer
- Doug Dechow
- Dwight Duego
- Zung Nguyen
- Viera Proulx
- Richard Rasala
- Stephen Wong
- Javier Galve-Frances
- <you>
- <and you>

Elementary Patterns

- Intended for novices
- Both
  - simpler standard design patterns and
  - Patterns created especially for novices
- Varieties of scale, but attacking problems encountered by novices

Goals

- More effective teaching (not new material)
- More effective learning
  - Students can say why they do what they do
  - Students can follow a plan
  - Students can make connections
- Introduction to the patterns way of thinking
- Efficiency when solving recurring problems

Example

- The Decorator pattern can be used to dynamically add functionality to an object defined by an interface without modifying the object
- You can use it to develop search strategies in simple lists/vectors
- It is also the basis of the design of the Java I/O libraries
- Once you understand Decorator, you understand Java I/O

How to use Elementary Patterns

- Not a syllabus item or exam item
- Tools for studying design principles
- Tools for reading and organizing programs
- Tools for writing programs
- Vocabulary for communication between instructor and student
- Benchmark for evaluating programs

Tools for studying design principles

- Use Decorator or Composite
  - to show both inheritance and delegation in a way that maximizes the benefits of both
  - to show dynamic polymorphism
  - to show the power of interface

Tools for reading & organizing programs

- Enumeration -- Java collections
- Observer -- Java events
- Decorator -- Java I/O
- Pipes & Filters -- Compiler passes
- Immutable -- Making programs simple to understand

Tools for writing programs

- What IS the problem?
- What IS the solution?
- WHY is one solution better than another?
- HOW do the pieces fit together?

- Students need to ask and answer these questions. Patterns can help.

Example of Use

- Show the same problem solved with and without patterns.
- Discuss the differences
  - Why is one better?
    - Maintainability? Clarity? ...

Example of Use

- Give an exercise in which patterns are likely to appear and a list of likely patterns
- Ask them to document patterns in their solutions
  - The list can be as short as one pattern.

Example of Use

- Prepare a Fixer-Upper (pedagogical pattern)
  - In this case a large program written without patterns
  - Have the students (in pairs, perhaps) identify opportunities for refactoring into patterns from a fixed small list of elementary patterns.
  - Have them carry out the work

How Have We Used Patterns?

- Wallingford Example
  - Balls
- Bergin Examples
  - Understanding OO
    - Polymorphism, factory, singleton, null object
  - Representing Gender
    - State, strategy, decorator
  - Decorator and Java I/O

How Have We Used Patterns?

- Owen Astrachan has begun to create a series of exercises that show the difference between pattern poor and pattern rich software in Java.
- The first in the series is hangman, a simple word game played by children
  - http://www.cs.duke.edu/csed/patterns/oofirstyear/hangman
- Watch for more in this series

Elementary Patterns

Patterns We Write

Instructor Written Patterns

- Helps the instructor understand the material
- Helps the instructor organize the material
  - See first two pages of Patterns for Selection
- Provides an outline for the students
  - http://max.cs.kzoo.edu/cs210/Labs/AquaJava/index.shtml
  - http://max.cs.kzoo.edu/AP/Fish/AquariumLabSeries/
  - http://max.cs.kzoo.edu/~abrady/CS420W00/Labs/index.shtml

Sample Pattern (Selection)

- Function For Complex Condition
  - You want to write *Simply Understood Code* but complex conditions in if and while statements are hard to grasp quickly
  - **Therefore**: Write a separate boolean valued function to express the condition.
  - Make sure its name *Expresses a Positive Condition*.

Sample Pattern (Coding)

- Brace All
  - You want to *Prepare for Change* in your programs. Your if and while statements may need additional actions added to them in the future. This can be error prone
  - **Therefore**: When you first write an if or a while, include its statement in braces, even when this isn't required by the language.

You Can Do This at Home

- Think about some topic you regularly teach
- Think about the problem you try to solve
- Think about the right solution
- Think about when/where the problem occurs
- Think about what leads you to the right solution.

What Has Been Done?

- Selection -- Java, C++
- Iteration
- Recursion -- Scheme
- Polymorphism (a start only)
- Elementary Coding -- Java
  - None of this work is perfect. It can all be revisited, refined. Some is preliminary.

Examples

- Selection from Selection
- Selection from Coding Patterns

What Needs to be Done?

- Object Selection (find the objects)
- Responsibility assignment
- Class Design
- Inheritance
- Method design and refactoring
- I/O
- Concurrency
- etc.

Organizing Everything

Community Goal

- Write a pattern language for students in the first CS course
- Assume the course will be object-oriented
- Using Java or C++
- Will involve both programming and design, as well as important CS topics
- We currently have only a few pieces of this

Patterns Process

- Author writes
- Author Submits
- Shepherd Assigned
- Shepherd Gives Feedback/Author Revises
- Paper Workshopped by Group of Authors
- Author Revises
- Paper Published

Patterns Process (2)

- There are pattern languages for
  - Writing patterns
    - http://webclass.cqu.edu.au/Patterns/Resources/writers/language/
  - Shepherding patterns
    - http://jerry.cs.uiuc.edu/~plop/plop2k/shepherding.pdf
  - Workshopping patterns
    - http://www.bell-labs.com/~cope/Patterns/WritersWorkshops/

Elementary Patterns

Patterns Students Write

Student Written Patterns

- This is an interesting but largely untested idea.
- Have students write up what they have learned on a topic in pattern form
- Outline form is probably best to start
- A revision could use Alexandrian form

Elementary Patterns

Patterns We Use

Useful Simple Patterns

- Null Object
- Decorator
- Composite
- Singleton
- Immutable Object
- Observer
- Filter
- Strategy
- Delegation
- Object Pool
- Adapter
- Pipes and Filters
- Others depending on your course, skills, and background

Pedagogical Patterns

Patterns for Educators

http://www-lifia.info.unlp.edu.ar/ppp/

http://www.pedagogicalpatterns.org

Pedagogical Patterns

- Capture expert knowledge about teaching
- Transmit the knowledge to those who need it
- Provide tools for course construction, delivery, and improvement
- Patterns at several levels of scale
  - years...minutes

Current Key Players

- Mary Lynn Manns
- Helen Sharp
- Jutta Eckstein
- Joe Bergin
- Markus Voelter
- Kent Beck
- Alan O’Callaghan
- Máximo Prieto
- Phil McLaughlin
- Many others have contributed
- <You are needed>

Sample Pattern (Large scale)

- Need To Know
  - There is too much to teach and more is added daily. If you try to teach it all you will never finish.
  - We prefer *Active Students* who learn thru doing
  - **Therefore**: Teach what the students *Need To Know* to facilitate problem solving
    - Siliconized topics can be omitted entirely
  - Other material can be dropped or deferred

Sample Pattern (Small Scale)

- Gold Star ★
  - Students should be rewarded for what they do well. It motivates them to do more and better
  - Rewards can be public as well as private
  - **Therefore**: Find a way to give small tokens to students who do something especially well
  - This can tangible or intangible.

Pattern Languages

- Seminars
- My attempts for (single) Course Design
- The community’s attempt to unify the patterns on the web site.

- This is where the real benefit will lie
  - beyond incremental improvements

Existing Work

- Too much of it is not up to current standards
  - Single use tricks
  - Too complex
    - Simpler patterns seem to be better (personal bias)
  - Too specialized
    - Authors haven’t sufficiently abstracted the essence
  - Can’t easily see how it can be integrated
  - Very little on pattern languages

BUT...

- The patterns community generally sees great promise here.
- There is potential for creating really useful pattern languages that can serve as models for others, just as the organizational patterns have done.

Seminars--Fricke & Voelter

- A pattern language for design and delivery of short courses
- Workshopped at EuroPLoP 2000
- 48 Patterns in 5 phases
- http://www.voelter.de/seminars/
- Perhaps the best current example of a pedagogical pattern language

Seminars (Prep)

- Check Prerequisites
- Adapt to Participants Background
  - Let them Decide
- Comfortable Environment
- Seminar Plan
  - Breaks
  - Buffers
  - Reference the Plan
  - Review after breaks
  - Summary
  - Manuscript

Seminars (begin)

- Welcome the Participants
- Introduction Session
- Personal Communication
  - Anonymous Mailbox
  - Nameplate
  - Games

Seminars (teaching)

- Prepared Equipment
- Be There First
- Process Group (7)
  - Let the Plan go
  - ...
- Teaching Group (7)
  - Digestible Packets
  - ...
- Teacher Group (7)
  - Nobody is Perfect
  - ...
- Media Group (2)
  - Change Media
  - Use Participants Media
- Participant Feedback Form

Seminars (teaching ends)

- Kind of Exam
- Trial Exam
- Selectable Exam Time
- Differentiated Feedback

Seminars (end)

- Seminar Debrief

Pattern Language For CS Courses

- In development
- Currently about 100 patterns
- Some overlap with Seminars and other work
- 11 categories
  - Prior to Course. Course as a whole. Starting the course and major topics. Scale of Weeks, days, hours, minutes. Evaluation. Communication. Dealing with problems.

Pattern Language For CS Courses

- Online
  - http://csis.pace.edu/~bergin/patterns/coursepatternlanguage.html
  - http://csis.pace.edu/~bergin/PedPat1.3.html
  - http://csis.pace.edu/~bergin/patterns/fewpedpats.html

Prior to Course

- New Pedagogy for New Paradigms
- Need to Know
- Abandon Systems
- Check Prerequisites (Fricke & Völter)
- Adapt to Student Background (Fricke & Völter)
- Iterative Course Development (Anthony)

Course as a Whole

- Early Bird
- Spiral (and Anthony's Chicken and Egg,which may be used at any level)
- Multi Pronged Attack
- Groups Work
- Lazy Professor (and Fricke & Völter's Different Approaches)
- Active Student
- Buddy System
- Language Reinforces Paradigm
- Write Over Read
- General Concepts First (Fricke & Völter)
- Study Groups
- Reduce Risk
- Stealth Instructor

Starting Course/Major Topics

- Set the Stage
- Lay of the Land (and Anthony's Module's Story)
- Visible Plan (and Fricke & Völter's Manuscript)
- Learn Their Names (and Fricke & Völter's Nameplate)
- Fixer Upper
- Occam
- Read Before Write
- Consistent Metaphor
- High Leverage

Scale of Weeks

- Larger Than Life
- Tool Box
- Take a Risk
- Constant Challenge
- Inlook - Outlook
- Assigning and Grading (short) Team Projects (Meyer)
-
- Explore-Present-Interact-Critique (Knudsen & Madsen)

Scale of Days

- Toy Box
- Fill in the Blanks
- Quality is Job One
- Different Exercise Levels (Fricke & Völter)
- Repeat Topics (Fricke & Völter)
- Chunks of Information (Bienhaus)
- Example Lasts One Week (Anthony)
- Acquaintance Examples (Anthony)
- Lab-Discussion-Lecture-Lab (Manns)
- Lecture-Activity-Student Presentation-Discussion (Sharp)
- The Three Bears (Beck)

Scale of Hours

- Smile and Greet (and Fricke & Völter's Welcome the Participnts)
- Reveal Yourself
- Student Design Sprint
- Mistake
- Icky Poo (and Anthony's Simulation Games)
- Expose the Process (Becker)
- Relevant Examples (Fricke & Völter)
- Digestible Packets (Fricke & Völter)
- Separate Similar Content (Fricke & Völter)
- Reference Examples (Anthony)
- Gagne'-Ausbel Pattern of Lecture (DeClue)

Scale of Minutes

- Gold Star
- Test Tube
- Media Discipline
- Honor Questions (Fricke & Völter)
- Nobody is Perfect (Fricke & Völter)
- Review After Breaks (Fricke & Völter)
- In-Line Exercises (Marler)

Evaluation

- Fair Project Grading
- Fair Team Grading
- Key Ideas Dominate Grading
- Student Online Portfolios
- Grade It Again Sam
- Student Selected Activities
- Trial Exam (Fricke & Völter)
- Self Test (Steindl)
- Debrief After Activities (Anthony)

Communication

- Rule of 1 - Rule of 2
- 24 by 7
- Constant Feedback
- Anonymous Mailbox (Fricke & Völter)
- Feedback (Fricke & Völter)
- Ask Your Neighbor (Eckstein)

Dealing With Problems

- Buffers (Fricke & Voelter)
- Prepare Equipment (and have a backup) (Fricke & Völter)
- Let the Plan Go (Fricke & Völter)
- Debrief (Fricke & Völter)
- Human Professor
- Capture Everything

Patterns and Academic Research

- Patterns are not original to their authors
- Patterns are not new or created, but discovered (*mined* is the term of choice)
- Perhaps dangerous undertaking for untenured faculty to write patterns
- However, there ARE research possibilities
  - How do we know a solution is **best**?

The Future

- Form a Patterns Reading Group in your town. Get together monthly (weekly?)
- Form a Patterns Writing Group in your town. Workshop each other’s patterns.
- Join a mailing list like patterns-discussion
- Go to wiki web to get connected and find existing groups.

Feedback About the Course

- Send me email at berginf@pace.edu
- Anonymous messages about the course can be filtered through Michael Caspersen who can pass on the content.

PLoP Conferences

Mostly workshopping of papers
(ChiliPLoP is all working groups)

Many PLoPs

- PLoP: Held in Illinois each August
  - http://jerry.cs.uiuc.edu/plop/plop2k/
- EuroPLoP: Irsee, Germany each July
  - http://www.coldewey.com/europlop2000/
  - http://WWW.hillside.net/patterns/EuroPLoP
- KoalaPLoP: Melbourne Australia , May
  - http://www.cs.rmit.edu.au/KoalaPLoP/
- ChiliPLoP: near Phoenix, Arizona each March
  - http://www.agcs.com/patterns/chiliplop/
- SugarPlop: Rio de Janeiro, Brazil, October ??

Patterns FAQ

- Doug Lea
  - http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html

Games

- Jane Chandler on Understanding Alexander Through Games
  - mailto:Jane.Chandler@port.ac.uk
  - http://www.dis.port.ac.uk/~chandler
- George Platts on Non Competitive games
  George is games-master at most PLoPs
  - mailto: ssmdgp@hantsnet.hants.gov.uk
  - Also see: http://www.deepfun.com/

Writing Patterns

Time to Be Creative

Plan

- Start with your idea
- Express it in outline form
- Have it read and commented on (Shepherd)
- Re-write it (perhaps in Alexandrian form)
- Have it workshopped
- Revise it again
- Voila--rich and famous

Patterns

Philosophical, Ethical, and Social Dimension

Alexander?

- Alexander as Sacred Text
- Alexander as Literary Form

- Even if we reject the first it may be useful to us to try to understand Alexander’s real purpose

Alexander

- Original Goals
  - Improve human life
  - Give ordinary people a vernacular of architecture so they could
    - Control their own spaces/world
    - Release and enable human potential
  - NOT improve the lives of architects
  - NOT make buildings prettier/cheaper/faster…

Exericse

- Problem: Design the intersection of two or more rural roads
- Context: Denmark
- Forces: Low maintenance. Land is cheap. Electricity is expensive to deliver. Traffic is light. Don’t cause an unnecessary stop.
- Solution:

But: A better question

- What should we build
  - NOT how should we build it
- NOT just roads
  - public transport
  - other communication systems...

QWAN

- The Quality Without A Name (TTWOB)
  - “There is a central quality which is the root criterion of life and spirit in a man, a town, a building, or a wilderness. This quality is objective and precise, but it cannot be named.
  - The search which we make for this quality, in our own lives is the central search of any person, and the crux of any person’s story.

“I can’t define it, but I know it when I see it.”

Famous quote of U.S Supreme Court justice speaking about pornography

The Gate

- “To reach the quality without a name we must then build a living pattern language as a gate.”
- Not a barrier, but an opening to help us get from where we are to where we want and need to be.

The Way

- “Once we have built the gate, we can pass through it to the practice of the timeless way.”

Consequences

- Human scale, human needs
- No prefabricated parts
- Not just beauty, but “liveness”
- “...merely release the fundamental order which is native to us.”
- “...do exactly what emerges from ourselves”
- Define WHAT should be built, not just how

Consequences

- Languages rather than patterns
- Universality rather than specificity
- Release human potential, not just make tomorrow’s software cheaper or more maintainable
  - e.g. in XP one of the rules (patterns) is
    - 40 hour week.

Consequences

- There is no pattern language for doing biological warfare.
  - Either it would have only one pattern
    - JUST SAY NO
  - Or we would recognize that one force (destroying the human race) is so strong as to be unbalance all possible combinations of other forces.

Question

- How many pattern languages of architecture are there?
  - For Alexander the answer is ONE.
  - The language doesn’t grow out of his thought and desires or specific philosophy. It grows out of basic human nature which does not change.
  - Therefore, if he has the right language (certainly it can be improved) then it is unique.

Is this the same for design patterns?

- Probably not. There are probably many pattern languages for design.
- There are probably only a few pattern languages for organizations. Different organizations have different goals.
- It many be however, that there is only one pedagogical pattern language--but we are a long way from knowing what it is.

Beyond Patterns

- Alexander is working still on new ideas. He has developed the theory of centers which I don’t understand much of. Coplien does. He also thinks that Patterns are defined by geometry (clear in Architecture at least) and that my patterns exhibit this.
  - I’m still struggling with this.
  - http://www.PatternLanguage.com/
    - Look for “The Nature of Order”

What should we build?

- Not just how
- What should we NOT build
- How can we give the ordinary person affected by IS the “vernacular” to build their own systems? If that is impossible, how can we guarantee that those affected are included in the critical design phases?

A New Project (14/11/2000)

- A Pattern Language for Textual Electronic Communication (Collaborative Project)
  - an actual pattern language that would guide a person in deciding what to build, rather than how to build it
  - the people affected by the design of these (email) systems is enormous
  - http://www.egroups.com/group/apl4tec

Textual Electronic Comm.

- Intimate Communication
- Low-overhead Messages
- Peripheral Awareness
- Automatic Connection
- Closed Door

Controversy

- Not all members of the community hold the same views about the importance of QWAN
  - Coplien, Gabriel, Bergin … do
  - Some others just want to get on with the work
- Not all members feel that patterns have a moral component

Responsibility

- Responsibility of the Teknowlegable*
  - Those who know the consequences should
    - Protect the rights of those affected
    - Include those affected
- Power (or lack of) of the IS worker
- A Pattern for protecting people’s rights
  - Accounting in the US
    - Audit required

*Author forgotten

Responsibility

- Safety Critical Systems
- Privacy Critical Systems
-
-
-
  - The affected people must have a say in the design (opt-in vs. opt-out)

Ideas for New Patterns Discussed

- Constant Challenge
  - Worst Grade Dominates
  - Big Group Projects
  - Individual Follow up
  - Group Oral Exam
  - Variable Speed
    - Graduated Projects
    - Open ended projects
- Interdependent Group Project
- Teach Me
- Group Questions
- Don’t Answer

Ideas for New Patterns Discussed

- More Loop Patterns
- Recursion in Java
- Algorithmic Patterns
- Family of Courses
- Refine Proxy pattern
- Set the Stage
  - Create an educational environment
- From UML to Java
  - When to use association, delegation…
- Class Design
  - Responsibilities…
  - & see GRASP (Larman)
- Interfaces
  - when-where-how

Ideas For New Patterns

- Bumpy Road (pedagogical pattern)
  - If your lecture is too uniform, especially if it is long, you will be boring
  - **Therefore** vary what you do. Styles of presentation, activities, question periods...

Ideas For New Patterns

- Mark Sharp Turns (pedagogical pattern)
  - – Sometimes you come to a difficult part in a lecture or presentation. Difficult to learn, difficult to explain. Maybe you don’t even know yet how to present the idea clearly and completely.
  - – **Therefore**, let the students know that this is a difficult bit and why so they are prepared to focus especially on it.

Ideas For New Patterns

- Brave Professor
  - – Sometimes you foul up grandly
  - – Sometimes there is a lesson in it
  - – **Therefore**, admit your mistake to your students and point out what led you to the mistake

Suggested by Michael Caspersen