

PLoP® Pattern Writing Bootcamp 2012

Panama City, Panama



Joseph (Joe) Yoder joe@refactory.com

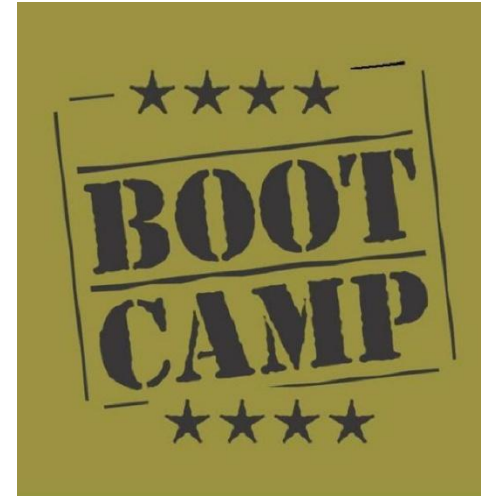
1 International Symposium
on Software Architecture
and Patterns

July 24-27 2012
Panama City, Panama

Sponsor by LACCEI and Security Group

Objectives

- At the end of this “bootcamp” you will:
 - Have knowledge about pattern writing
 - Have a better understanding of the “Quality Without A Name”
 - Have helped write a pattern
 - Be the newest members of the pattern community!



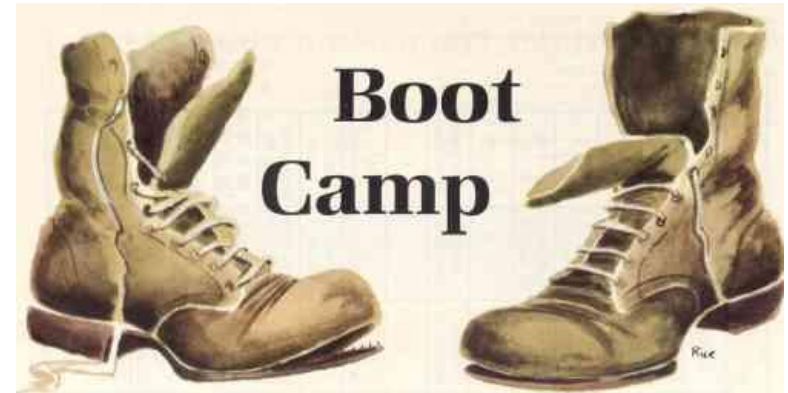
Thanks Rebecca



Thanks Linda & Bob

Agenda

- Introductions
- Why Patterns?
- Discussion of *What's a pattern?*
- Parts of a pattern
- Patterns criticism and categorization
- Choosing a topic
- Group Pattern Writing
- Group Pattern Writing Continued
- Sample Writers' Workshop
- Pattern Languages
- Pattern Ethics



Introduction to Patterns



The Reason for Patterns

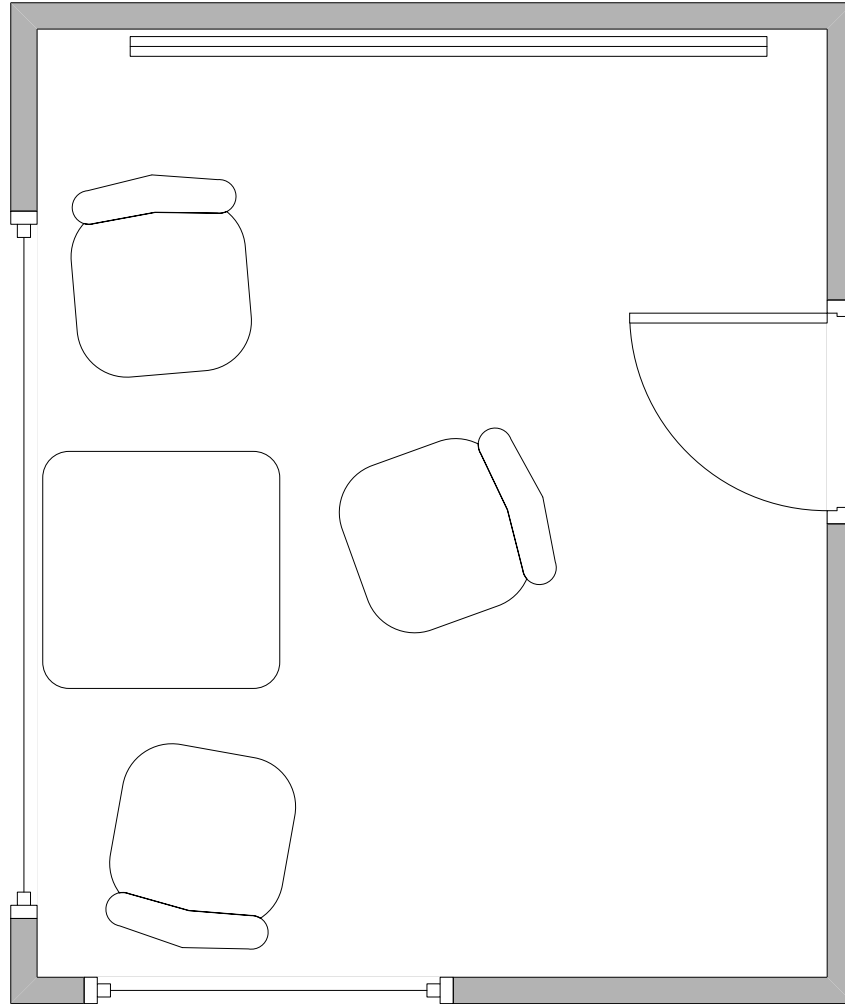
- Experts at work often use their knowledge of how they solved similar problems to solve the current problem.
- What if they could share their knowledge with others?
- Pattern descriptions record experts' wisdom so others can reuse it.



An Example of a Pattern*

Window place

*Christopher Alexander et al, *A Pattern Language-Towns.Buildings. Construction*, 1977



The Window Place Pattern

- Everybody loves window seats, bay windows, and big windows with low sills and comfortable chairs drawn up to them...A room without a place like this seldom allows you to feel comfortable
- If the room contains no window which is a “place”, a person will be drawn between:
 1. Wanting to sit down and be comfortable
 2. Being drawn toward the light
- If comfortable places are away from the windows, there is no way of overcoming this conflict...
- Therefore: In every room where you spend any length of time during the day, make at least one window into a “window place”



Part 1: Pattern Basics



Patterns

- “The pattern is, in short, at the same time a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it. It is both a process and a thing; both a description of a thing which is alive, and a description of the process which will generate that thing.”

-- Christopher Alexander,
The Timeless Way of Building



Christopher Alexander

- Building architect and theorist
- Many books on architecture and art:
 - A Pattern Language
 - The Timeless Way of Building
 - The Oregon Experiment
 - The Production of Houses
 - A Foreshadowing of 21st Century Art -- The Color and Geometry of Very Early Turkish Carpets
 - The Nature of Order (4 volumes)
- <http://www.livingneighborhoods.org>



The Quality Without A Name

- “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.”
 - Alexander, The Timeless Way of Building
- (“but with an acronym” -- Joe Davison)



What is a Pattern

Quite often you hear:

“A pattern is a proven solution to a problem in a context.”

Alexander writes:

“Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.”

Are these definitions the same?



What is a Pattern

It must be a solution to a problem in a context

You must be able to tell the problem solver what to do,
how to solve the problem

It must be a mature, proven solution (Rules of 3)

It must contribute to human comfort or quality of life

It must be something you didn't invent yourself
(Buschmann's Rule)

The solution should build on the insight of the problem solver, and
can be implemented a million times without being the same twice

It cannot be formalized or automated
(if it can, do that instead of writing a pattern)

It should have a dense set of interacting forces that are independent
of the forces in other patterns

Patterns Are:

- A form of architectural documentation
- An architectural relationship that cuts across system parts
- “Solution to a Problem in a Context”
- A way of explaining non-traditional solutions
- A family of solutions that abstractly address related problems in a specific context
- A literary form
- A resolution of forces



A Patterns Should

- Provide Facts (ref manual) about solving problem
- Be prescriptive about the solution
- Tell a good story which captures the experience of the experts
- Help to comprehend existing systems
- Help to construct new systems



What is not a Pattern

Just a simple solution to a problem in a context

- A simple rule
- A prescriptive recipe
- An algorithm
- A data structure



Patterns Will Not:

- ... make you an instant expert
- ... provide a “turn the crank” approach to software
- ... eliminate the need for intelligence and taste
- ... make you rich and famous
 - unless you become a snake oil salesman
- ... generate code
 - *Paul S. R. Chisholm, AT&T 10/94*





Part 2: A Pattern's Parts

Alexander's Pattern Definition

Each pattern describes a problem that occurs over and over again in our environment and then describes the core of the solution to that problem in such a way that you can use this solution a million times over without ever doing it the same way twice.

Alexander - building architect and author

- *The Timeless Way of Building*
- *A Pattern Language*



Parts of a Pattern (Alexander)

Problem - when to use the pattern

Solution - what to do to solve problem

Context - when to consider the pattern

Forces - pattern is a balance of forces

Consequences, positive and negative

Examples:

- Teach both problem and solution
- Are the best teacher
- Are proof of pattern-hood

Small Meeting Rooms

151 SMALL MEETING ROOMS*



Small Meeting Rooms

The **larger meetings** are, the **less people get out of them**. But institutions often put their money and attention into large meeting rooms and lecture halls.

It has been shown that the **number of people** in a group **influences** both the **number who never talk**, and the number who feel they have ideas which they have not been able to express.

There is no particularly natural threshold for group size; but it is clear that the number who never talk climbs very rapidly [with group size]. In a **group of 12, one person never talks**. In a group of 24, there are 6 people who never talk.

Make at least 70% of all meeting rooms really small -- for 12 people or less. Locate them in the most public parts of the building, evenly scattered among the workplaces.

See: Light on Two Sides of Every Room, Sitting Circle, Different Chairs, Pools of Light, The Shape of Indoor Space



Small Meeting Rooms

151 Small Meeting Rooms

151 Small Meeting Rooms*

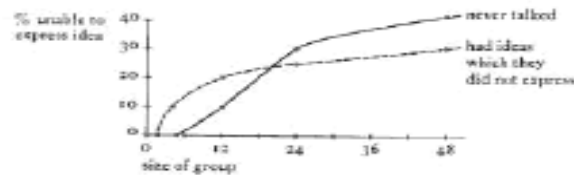


... within organizations and workplaces - University as a Marketplace (43), Local Town Hall (44), Master and Apprentices (89), Flexible Office Space (146), Small Work Groups (148), there will, inevitably, be meeting rooms, group rooms, classrooms, of one kind or another. Investigation of meeting rooms shows that the best distribution - both by size and by position - is rather unexpected.



The larger meetings are, the less people get out of them. But institutions often put their money and attention into large meeting rooms and lecture halls.

We first discuss the sheer size of meetings. It has been shown that the number of people in a group influences both the number who never talk, and the number who feel they have ideas which they have not been able to express. For example, Bernard Bass (*Organizational Psychology*, Boston: Allyn, 1965, p. 200) has conducted an experiment relating group size to participation. The results of this experiment are shown in the following graph.



As size of group grows, more and more people hold back.

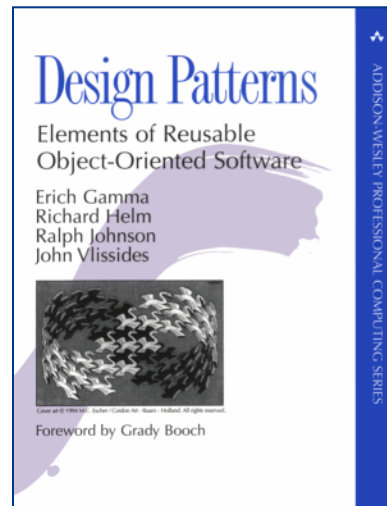
There is no particularly natural threshold for group size; but it is clear that the number who never talk climbs very rapidly. In a group of 12, one person never talks. In a group of 24, there are six people who never talk.

<http://www.stanford.edu/group/psychology/151.htm> (1 of 4) [5/2/07 7:26:57]

Design Patterns

“Design Patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.”

—Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides,
Design Patterns: Elements of Reusable Object-Oriented Software



Parts of a Pattern in *Design Patterns* (Also known as GOF for Gang of Four; Gamma et. al.)

Intent - brief description of problem and solution

Also Known As

Motivation - prototypical example

Applicability - problem, forces, context

Structure/Participants/Collaborations - solution

Consequences - forces

Implementation/Sample Code - solution

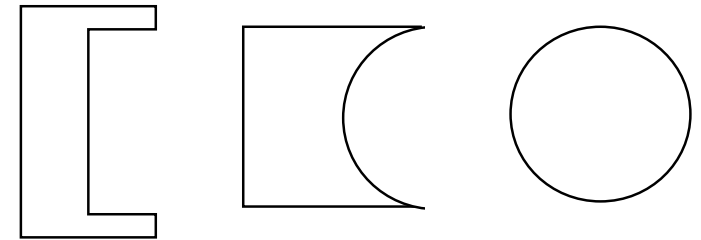
Known Uses

Related Patterns

Adapter

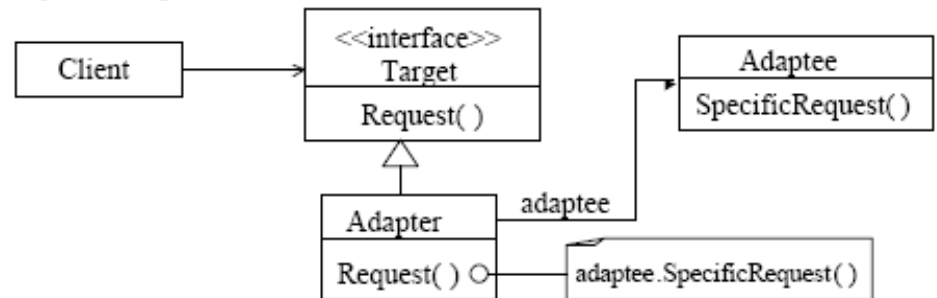
Intent

- Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
- Use Interfaces in Java and C# to get the equivalence look of multiple inheritance

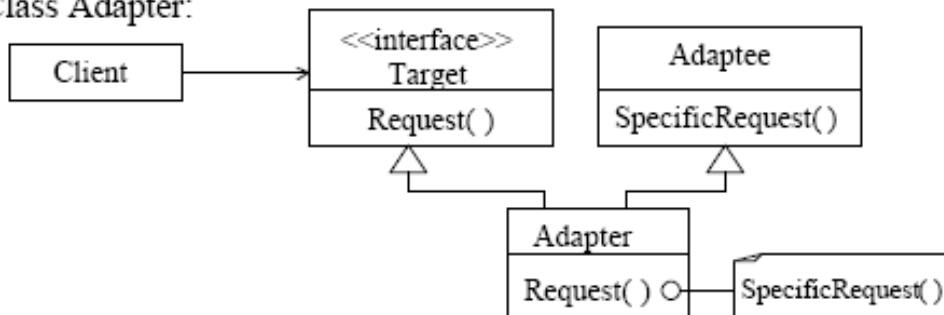


Target Adapter Adaptee

Object Adapter:



Class Adapter:



Patterns-Oriented Software Architecture (POSA) Pattern Form

Name, Aliases

Brief Description

Example

Context

Problem

Forces

Solution

Structure

Dynamics

Implementation

Example Resolved

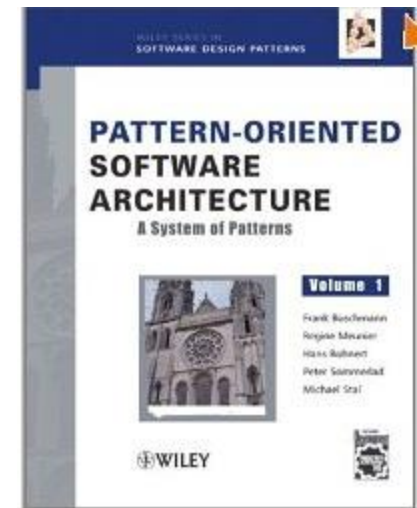
Variants

Known Uses

Consequences

Related Patterns

Credits



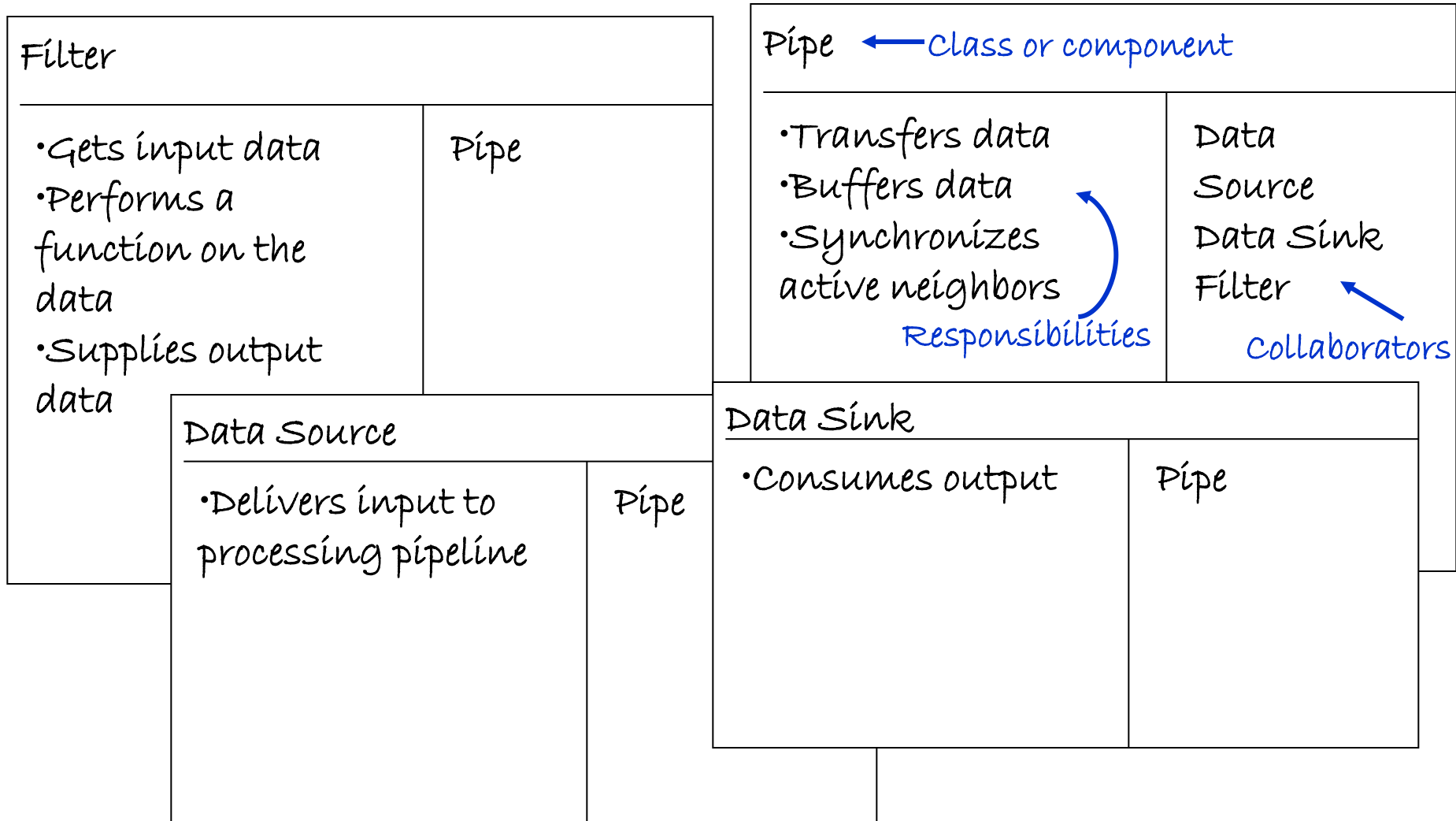
POSA Example: Pipes and Filters

Brief Description:

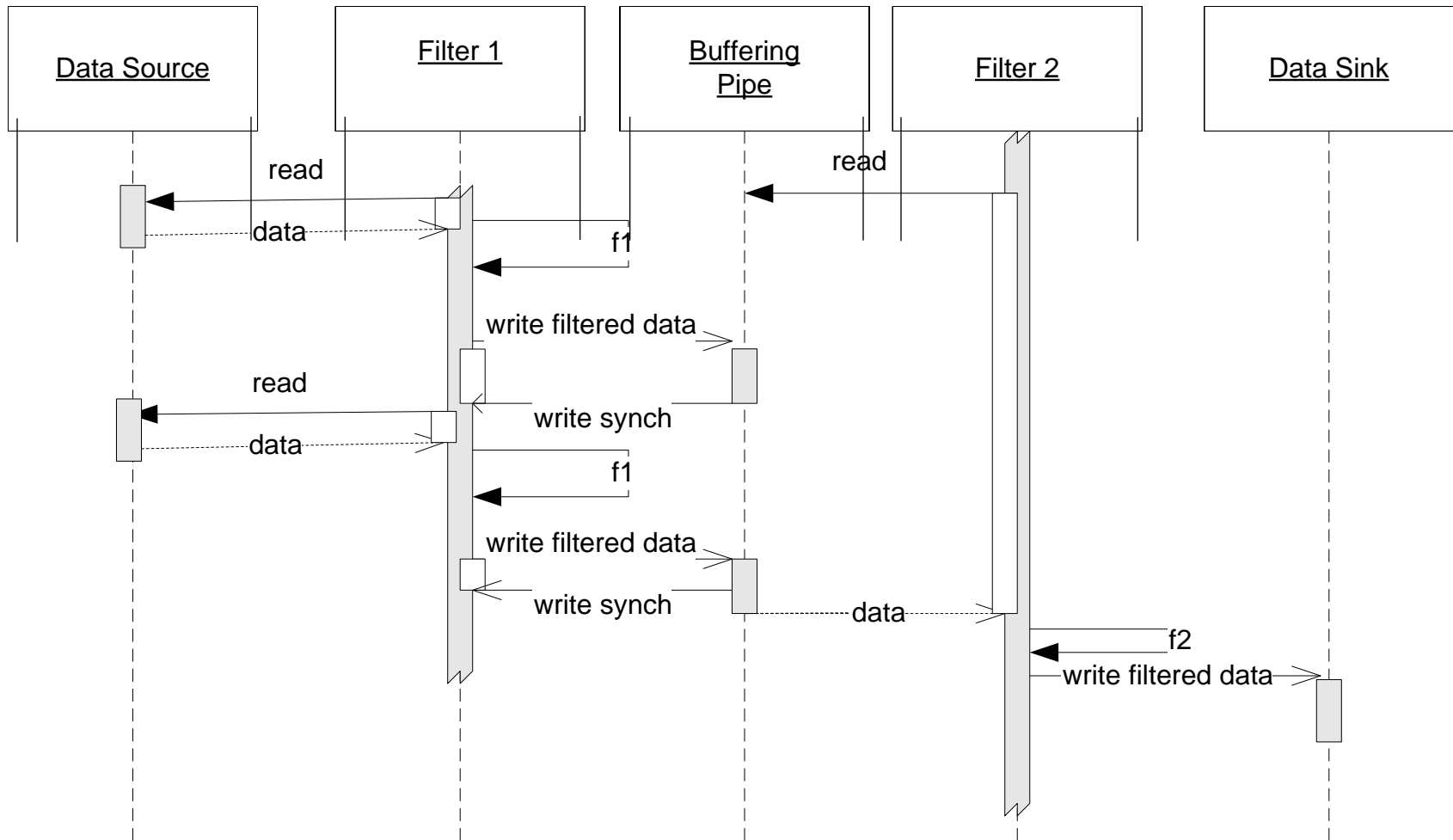
“The Pipes and Filters architectural pattern provides a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component. Data is passed through pipes between adjacent filters. Recompiling filters allows you to build families of related systems.”



POSA Example Pipes: and Filters Structure



POSA Example: Pipes and Filters Dynamics



Example Pattern Form

Name, *Aliases*

Context

Forces

Problem

Solution

Resulting Context (Consequences)

Rationale

Related Patterns

Known Uses

Sketch

Author

References

Examples

Pattern Example

How long should pattern sections be?
As short as possible but no shorter!
Albert Einstein and David Parnas



Name

Word or short phrase—the essence of the pattern,
some say noun phrase.

Naming is not trivial and very important.

Good names enhance communication—especially
when you can guess the intent from the name.

Patterns “build” something. The name should say
what the pattern builds.



Aliases

The same pattern may exist elsewhere, at another company, in a publication.

Experts use non-intuitive names that recall folklore or deeper meanings. Aliases can help novices.

An abbreviation or nickname can be helpful for discussions.



Context

The setting—target user, patterns applied, size, scope, timing, memory constraints, anything that might invalidate the solution if changed.

↪ **You're an *Evangelist* or *Dedicated Champion* who has called a meeting to introduce a new idea. Members of the user community are free to attend or not. You have resources, your own personal contribution or those of a *Local Sponsor* or *Corporate Angel*.**

Forces

Why the problem is hard.

The forces are often contradictory—create tension:

- You want to make your customers happy.
- You have limited resources.

↪ **There's always more important work to do.**

↪ **Most people are curious about new ideas.**



Problem

Short, complete statement of the problem the pattern will solve.

↳ **Usually a meeting is just another ordinary, impersonal event.**

↳ **How do we get people to want to attend our meeting?**

Solution

Your proposed method of solving the problem.

Resolve important forces determined by context; other forces may be ignored.

Keep the target audience in mind.

Best Patterns are Generative

↳ **Have food at the meeting—donuts or bagels in the morning, with coffee, tea, and juice, and cookies and drinks in the afternoon, lunch at noon-time.**

Resulting Context

What happens if the solution is applied, what forces resolved, what problems may arise, what costs and benefits.

Just “problem solved” is not enough.

↪ **Food will turn a mundane meeting, presentation or other gathering into a more special event. If offered in the beginning, it starts the meeting on a positive note.**

Rationale

Why the solution solves the problem.

Sell the pattern, teach the reader.

↪ **In Alexander’s pattern *Communal Eating* (147), “eating plays a vital role in almost all human societies as a way of binding people together and increasing the extent to which they feel like members of a group. The act of eating together is by its very nature a sign of friendship.”**

Other Sections

Known Uses: A one-time occurrence is an event. A double occurrence is a coincidence. If it occurs more than twice, it's a pattern.

Jim Coplien/Gerald Weinberg/Bunny Duhl

Related Patterns: Use, be used with, be similar to other patterns

- While the prospect of free food is nice, *Brown Bag* can be used when funding is not available. People can still eat together, even if they bring their own food.

Ward's Tips for Writing Patterns

- Pick a whole area, not just one idea
- Make a list of things you learned
- Cast each item on your list as a solution
- Now write each item as a Pattern
 - Try a four paragraph form where the second paragraph ends with the pivotal "therefore"
- Organize your patterns into sections
- Write an Introduction



Our Tips for Writing Patterns

- Take some small 3x5 or equivalent cards
- Brainstorm things you've learned from problems in a domain
- For each item, take a card and write the problem and solution as simple sentences
- Add any other ideas such as forces, related patterns, aliases, etc.
- Give each a candidate name that reflects the solution
- Organize the patterns
- Use this to start writing the patterns



Let's Write a Pattern!

Name, *Aliases*

Context

Forces

Problem

Solution

Resulting Context (Consequence)

Rationale

Related Patterns

Known Uses

Sketch

Author

References

Examples





Part 3: Pattern Languages and Generativity



Pattern Languages

- Individual patterns are useful, but . . .
- they are most powerful when combined into a language
- “Each pattern then, depends both on the smaller patterns it contains, and on the larger patterns within which it is contained.”

Alexander, TTWOB, p 312.



Pattern Languages

- Individual Patterns are good
 - No individual pattern balances *all* the tradeoffs.
- To address real sized problems, languages are really essential
- A language is a collection of patterns that work together.
- Within a pattern language the patterns build upon each other, resolving unbalanced forces from previous patterns.



Pattern Languages (cont.)

- Pattern languages must be complete in two ways
 - Morphologically
 - Patterns fit together to form a complete structure without gaps.
 - Functionally
 - Any new forces introduced by the patterns are resolved by the patterns themselves.
- If they are not complete in these ways we refer to them as a “collection”
 - The GOF and POSA books are collections

Example: Relating Patterns in *A Pattern Language*

“Choose surface colors which, together with the color of the natural light, reflected light, and artificial lights, create a warm light in the rooms.

This means that yellows, reds, and oranges will often be needed to pick out trim and lampshades and occasional details—HALF-INCH TRIM (240), ORNAMENT (249), POOLS OF LIGHT (252). Colored CANVAS ROOFS (244) and SOFT TILE AND BRICK (248) also help to make warm colored light. Blues and greens and greys are much harder to use; especially on the north side where the light is cold and grey, but they can always be used for ORNAMENT, where they help to set off the warmer colors...”*

Alexander et al, *A Pattern Language



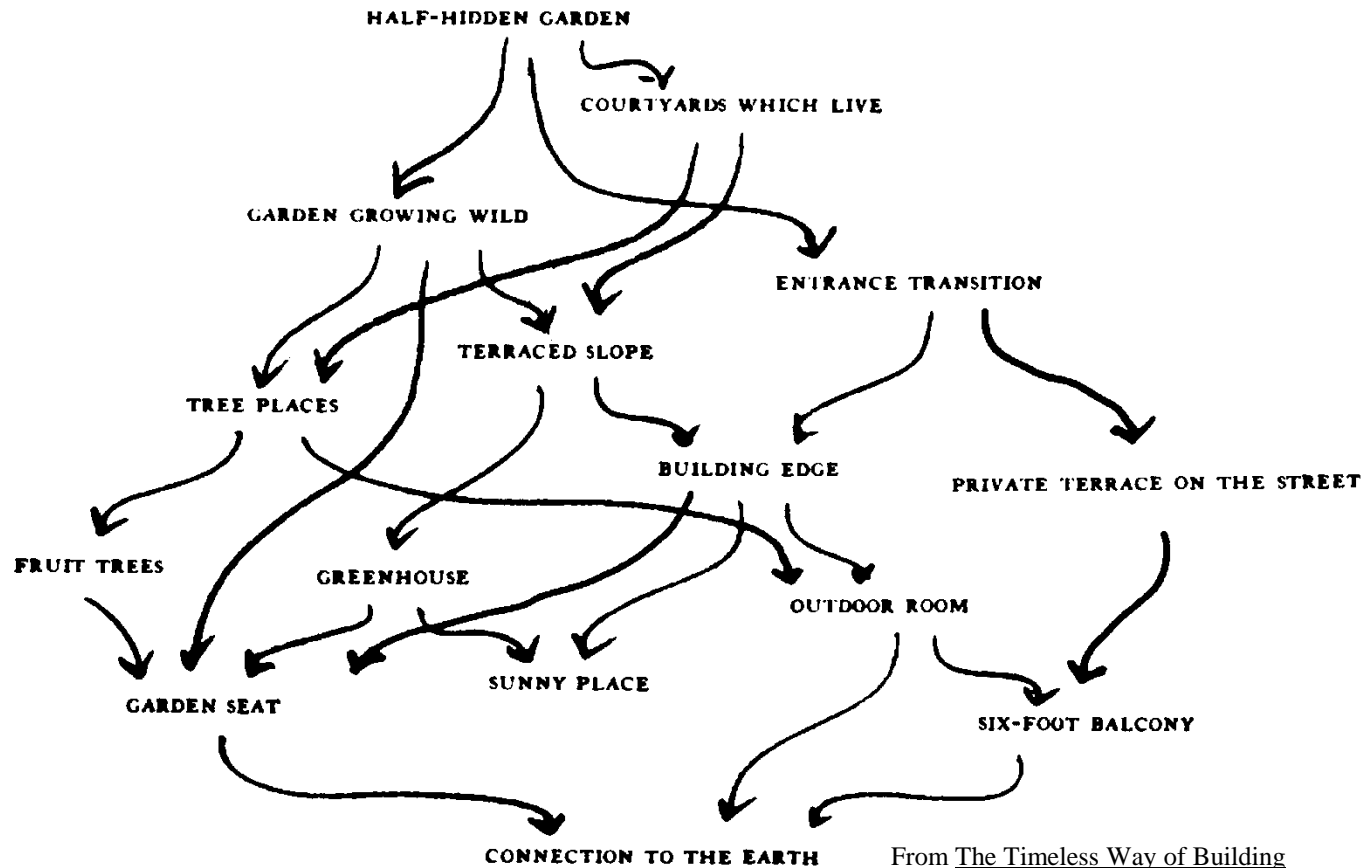
Parts of a Pattern Language

- Language intent
 - A short description of what the language is intended to accomplish
 - Like an abstract for the language
- Language map
 - A diagram that shows how the patterns build upon and relate to each other
- Language context
 - A description of how the pattern language is morphologically and functionally complete
- And of course ... the patterns that make up the language

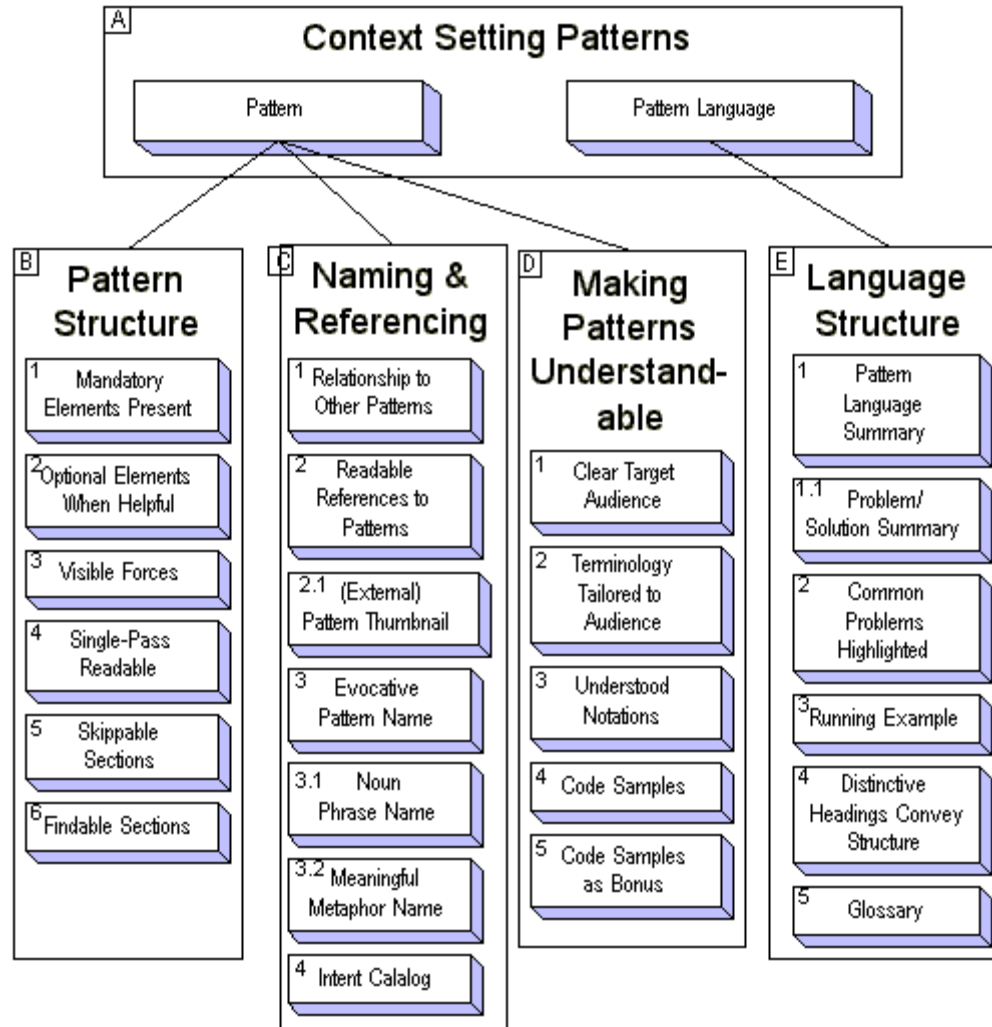
Example Pattern Languages:

- Alexander's Garden

- The only known pattern language diagram from Alexander

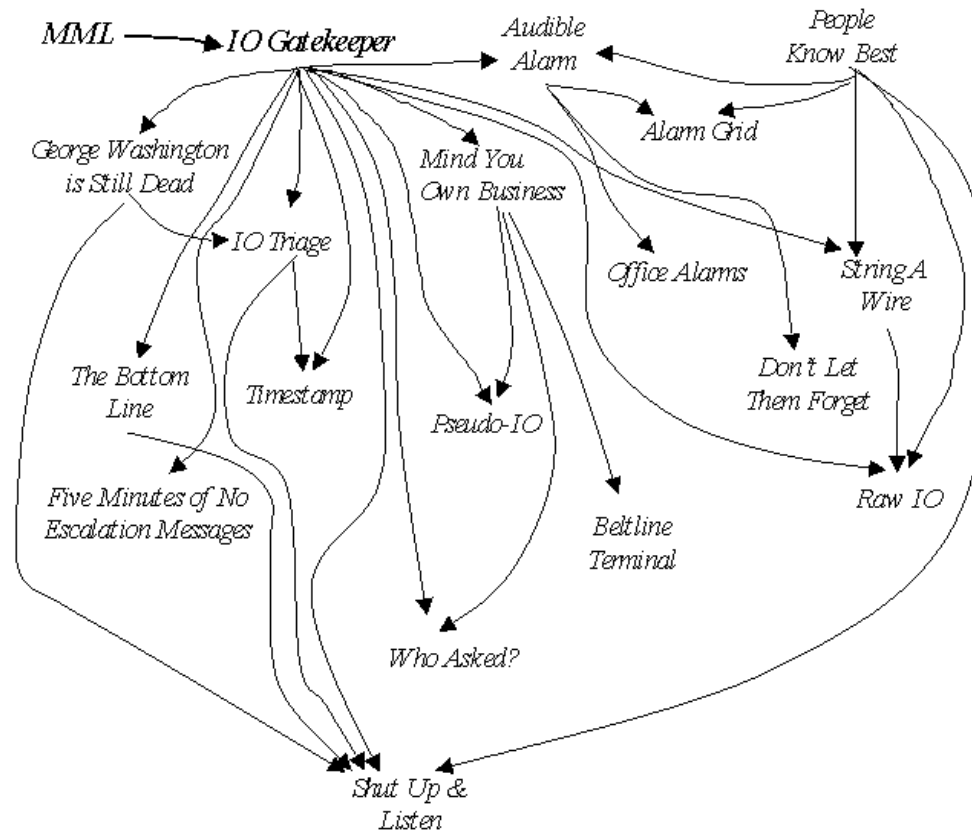


A Pattern Language for Writing Patterns (Meszaros & Doble)



Another Example

- Telecommunications I/O



From PLOPD4



1 International Symposium on Software Architecture and Patterns

July 24-27 2012
Panama City, Panama

Sponsor by LACCEI and Security Group



Part 4: Pattern Assessment

What makes a “good” pattern?

- What do you like about certain patterns?
- How do they make you feel?



Useful Patterns

- How do you read a pattern?
- How do others read patterns?
 - Title, context, problem, forces, solution?
 - Title, solution, problem, forces?
 - Title, solution, context, problem?
 - ...
- Write to your readers!



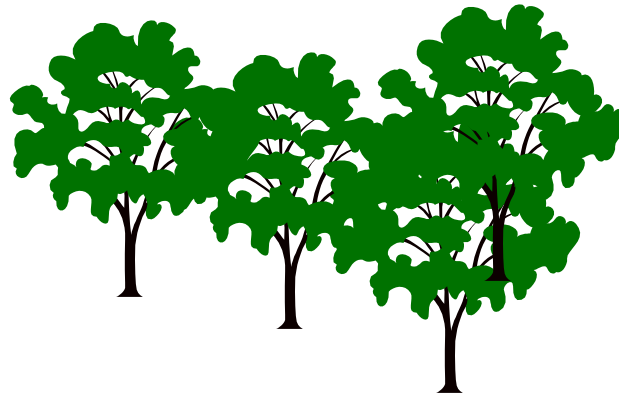
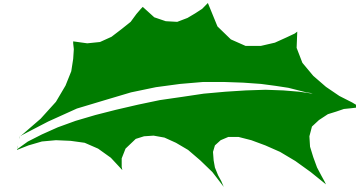
Pattern “Criticism”

- Ability to examine patterns and decide if they are useful
 - You must know how to look at the information in the handbook, or knowing where to find it won’t help
- Using patterns
 - Experienced architects and designers are “through the Gate”



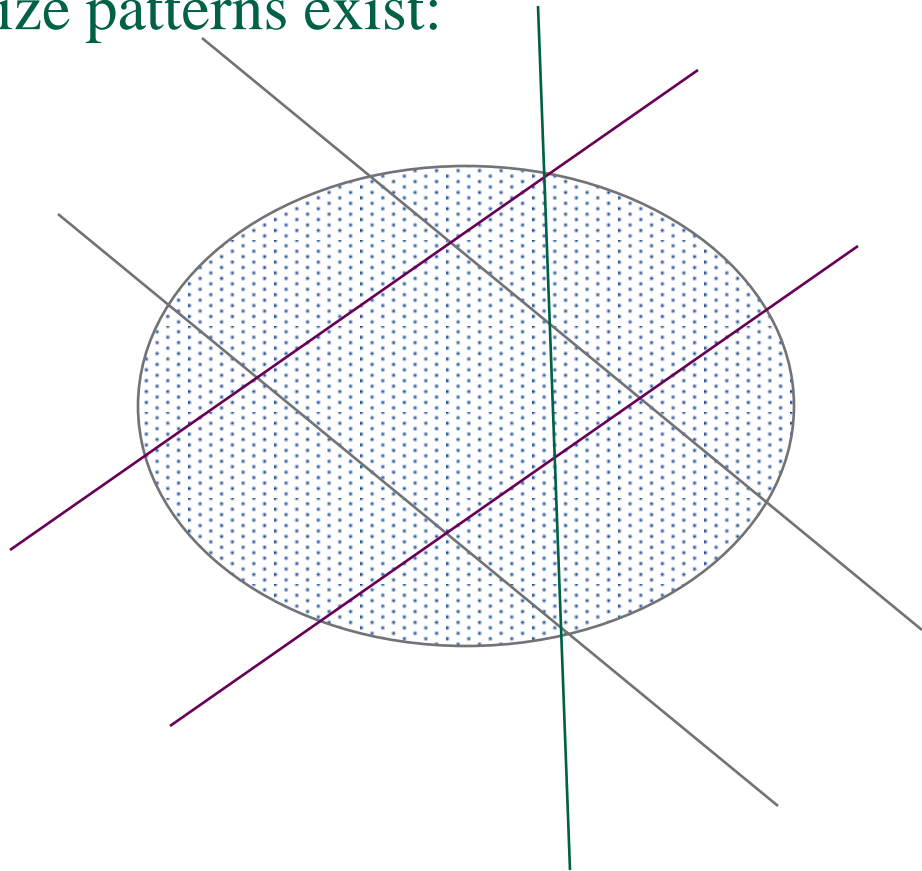
Depth of Patterns

- Some patterns are at a very low-level:
 - An example solution: Use a shift to multiple or divide by a power of 2
- Some are very general:
 - For example: *Model View Controller*



Categorizing Patterns

- Many ways to categorize patterns exist:
 - GOF:
 - Creational
 - Behavioural
 - Structural
 - Schmidt:
 - Strategic
 - Tactical
 - Coplien/Hanmer
 - Architectural
 - Design
 - Idiom



Values Underlying The 23 GOF Design Patterns

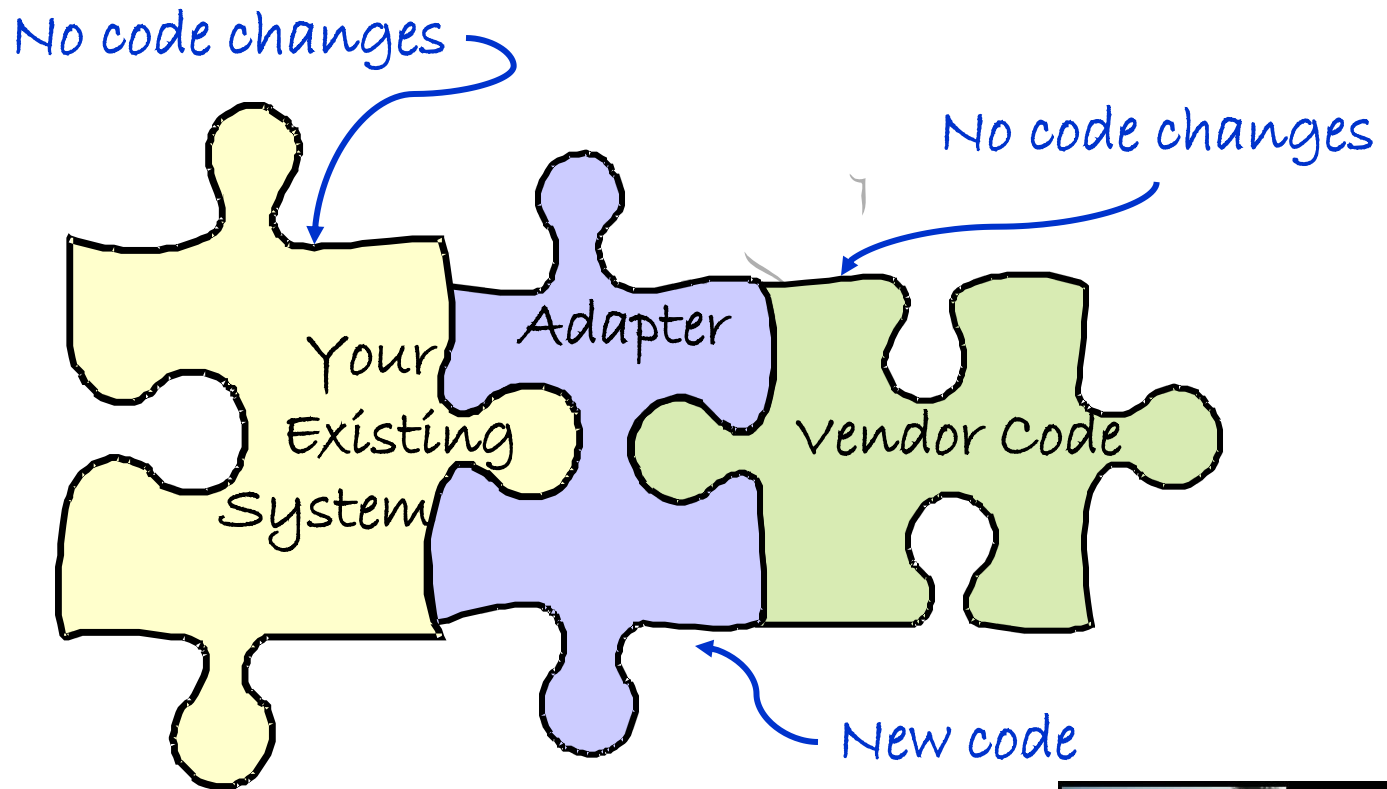
- Favoring composition over inheritance.
- Finding what varies and make it easily replaceable with another of its kind.
- Designing to interfaces, not to a particular implementation (Identify responsibilities and patterns of communication between participants. Keeping a higher-level view).



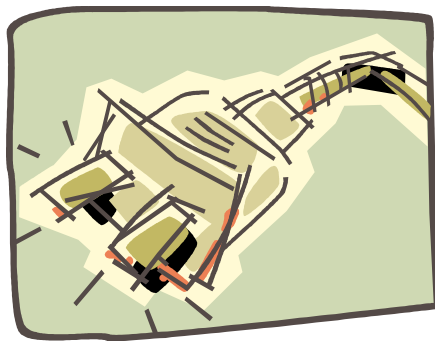
An Overview of The Adapter Pattern

p. 139 *Design Patterns*

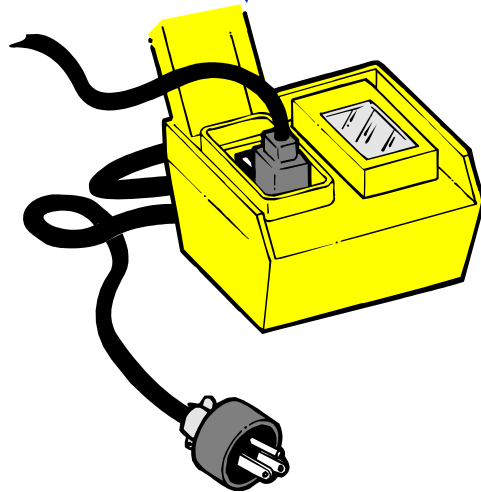
- The Problem: An existing system that needs to work with a new vendor



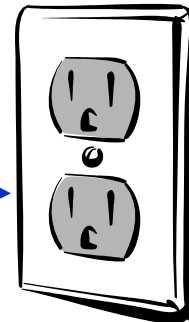
The Adapter Explained



request



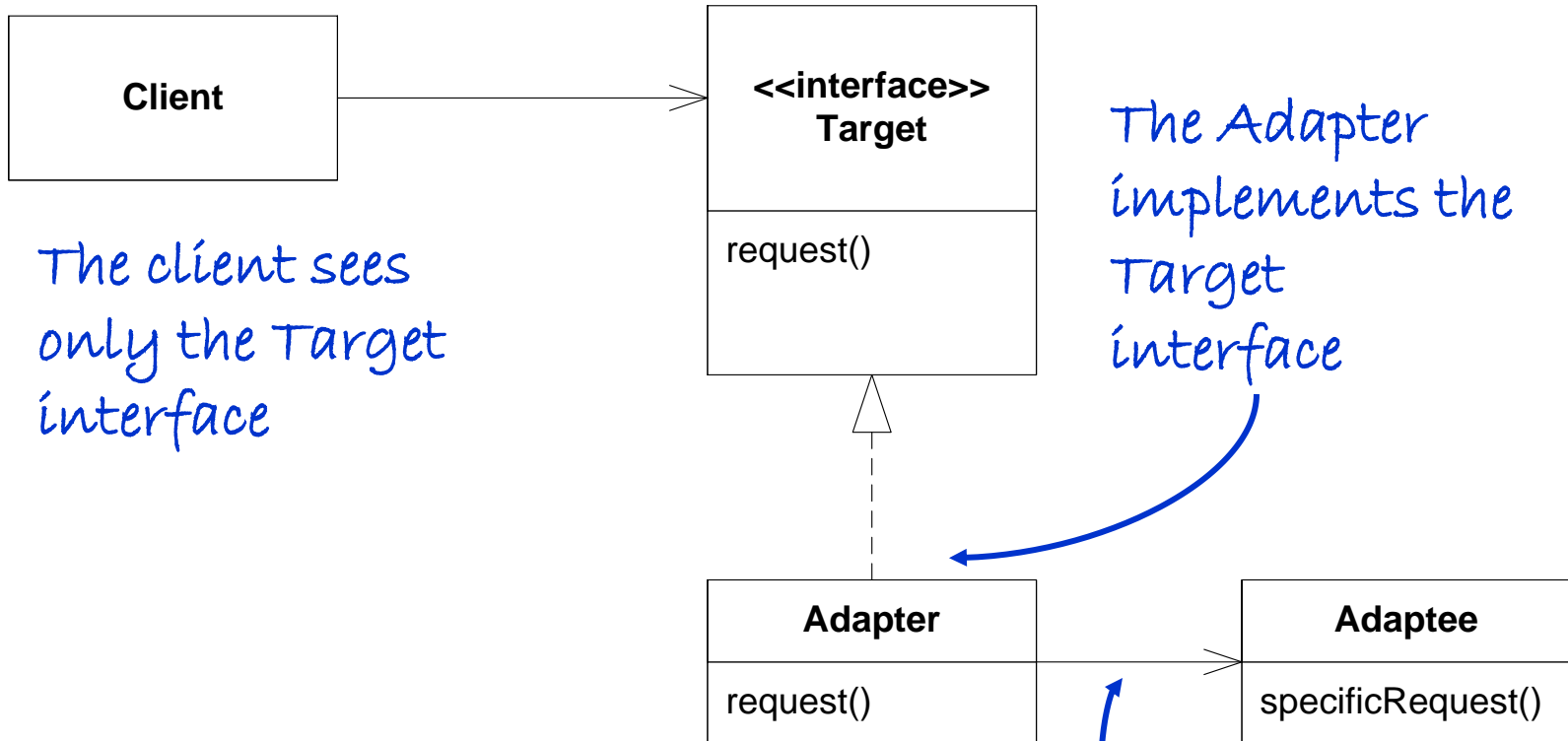
translated request



How a client can use an Adapter:

1. The client makes a request to the adapter using the target interface
2. The adapter translates that request into one or more calls on the adaptee
3. The client receives the results and never knows there is an adapter doing translation

Adapter Classes and Interfaces (The Solution)



The client sees only the Target interface

The Adapter implements the Target interface

The Adapter is composed with the Adapter

All requests get delegated to the Adaptee

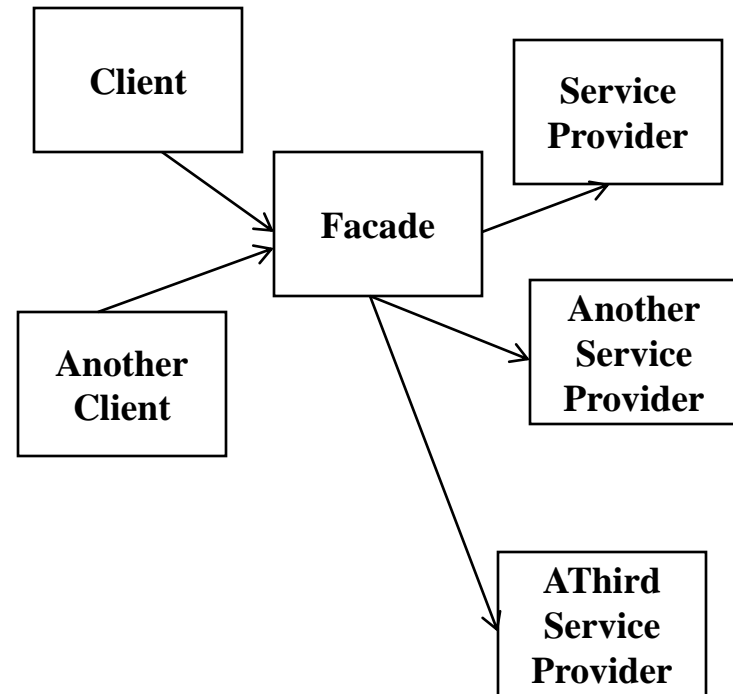
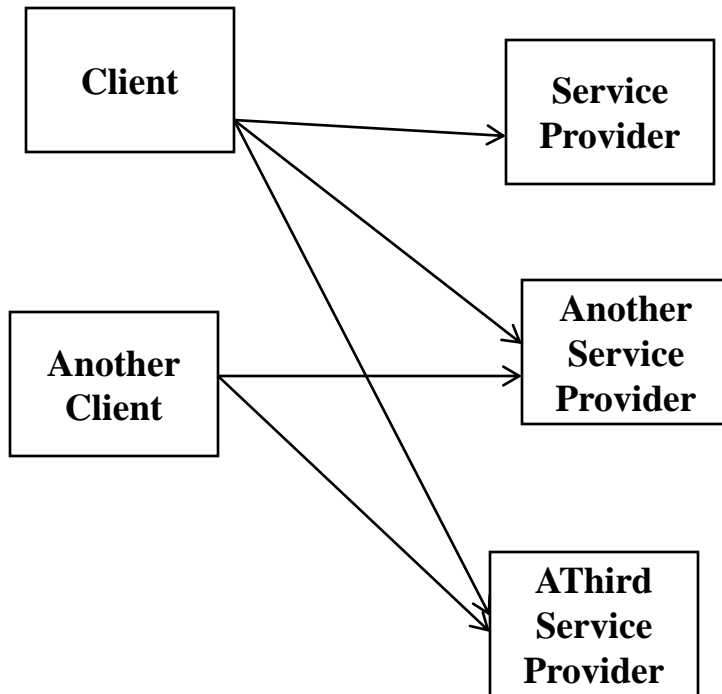
The Facade Pattern

p. 185 *Design Patterns*

- Problem: You only need a subset of a complex system or you need to interact with a complex system in a limited way.
- Solution: Present a new, simpler interface to the existing system.
- Consequences: Simplifies clients. If the facade doesn't provide a "complete" mapping of the encapsulated system's functionality, clients using the façade only get limited access to the system's full functionality.

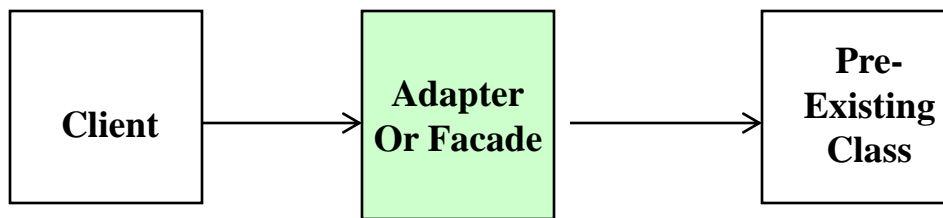
Façade

Before ... and... After



Comparing Adapter with Facade

- With both, pre-existing classes have desired behavior. And new classes are created with a desired interface.



Why are they different patterns?

Answer: Their design intent differs.

	Facade	Adapter
Pre-existing classes?	Yes	Yes
Standard interface desired?	No	Yes
Consistency/ polymorphism a goal?	No	Likely
Simplified interface a goal?	Yes	Not really



Part 5: Community, Culture, Writers Workshops, and Wrap-up



Patterns Community

- Pattern movement origins
 - 1986-1989
 - Alexander's Patterns Reading Group Tecktronix Ward Cunningham
 - OOPSLA First Pattern Workshop on Architecture Handbook Bruce Anderson
 - 1989-1992
 - Advanced C++ Programming Styles & Idioms by James Coplien
 - OOPSLA Workshops and Discussions on Patterns
 - “Center for Object-Oriented Programming”
 - 1990-1993
 - Patterns Thesis by Erich Gamma
 - Gang of Four Book
- The Hillside Group (<http://hillside.net>) (sponsored first PLoP 1994)
 - “The mission of the Hillside Group is to improve the quality of life of everyone who uses, builds, and encounters software systems-users, developers, managers, owners, educators, students, and society as a whole.
 - “Developing software is one of the most difficult human activities, and it affects every aspect of modern life. Understanding and helping the human element is critical for achieving success. The Hillside Group believes that software systems and software development can be made more humane by paying attention to real people and existing practices.
 - “The Hillside Group promotes the use of patterns and pattern languages to record, analyze, and improve software and its development, and supports any new practices that help achieve its mission.



“PLoP Conference”



- Pattern Languages of Programs (PLoP®) conference is a premier event for pattern authors and pattern enthusiasts to gather, discuss and learn more about patterns and software development. The purpose of PLoP is to promote development of pattern languages, primarily about aspects of software: design and programming, testing, software architecture, user interface design, domain modelling, and software processes.
- Pattern Languages of Programs (PLoP®) events are opportunities for pattern authors to have their pattern languages critically reviewed by fellow authors. This criticism is offered in the form of *Writer's Workshops*. This feedback allows the participants to improve their patterns to make them more useful or more publishable.

Patterns Community Gatherings

- Pattern Conferences

- **PLoP®** since 1994 at Allerton House, Monticello, IL*
 - Except: 2006: OOPSLA in Portland, OR
 - 2008: OOPSLA in Nashville, TN
 - 2009: AGILE in Chicago, IL
 - 2010: Splash in Reno, NV
 - 2011: Splash in Portland, OR
- Euro PLoP® since 1996 at Kloster Irsee, Bavaria
- SugarLoaf PLoP® since 2001, various locations in Brazil
- Viking PLoP® since 2002 rotating among Scandinavian countries
- AsianPLoP® 2010, 2011 in Tokyo Japan
- ScrumPLoP® 2009, 2010, 2011 in Scandinavian countries
- Chili PLoP® “A different Kind of PLoP”, near Phoenix, AZ since 1998
- ParaPLoP® Parallel programming patterns, held with ChiliPLoP since 2009
- Koala PLoP® 2000-2002 in Melbourne, Australia
- Mensore PLoP® 2001 Okinawa, Japan
- MetaPLoP® 2011 in Douro Valley, Portugal
- UP 1998 Mohonk Mountain House, New Paltz, NY

For more information visit <http://hillside.net/conferences>

- Transactions on Pattern Languages of Programming

- The new peer reviewed pattern journal.
- Published by Springer
- More information: <http://hillside.net/tplop>

July 25th, 2012 - 70



1 International Symposium
on Software Architecture
and Patterns

July 24-27 2012
Panama City, Panama

Online Communities

- Collect patterns focused on specific areas. Initially created using email lists which can evolve to focused conferences such as:
 - ParaPloP, MetaPloP, ScrumPloP, ...
- Initially this was the motivation behind Ward's Portland Pattern Repository wiki

ParaPloP 2011

ParaPloP 2011, a PLoP-style workshop on parallel programming patterns, will be May 10-12 in Carefree, AZ. You are invited to submit a paper. Papers should describe one or more patterns, outline a pattern language, analyze previously published patterns, describe case studies of using patterns to develop parallel software, or present experience mining patterns from significant parallel code implementations. Interested in what others have submitted in the past? Visit our [2010](#) and [2009](#) ParaPloP websites.

Similar to [PloP](#), ParaPloP will be organized as a set of writer's workshops. ParaPloP is extremely interactive and begins with the [submission process](#). Submissions are due **April 15, but if authors submit earlier**, then they will receive feedback and can resubmit. Please visit our [submit page](#) for more information.

Space permitting, non-authors will be allowed to attend ParaPloP 2011. If you are interested in attending as a non-author, please [email us](#) a brief statement about why you want to attend.

Location: As in previous years, ParaPloP 2011 will be held in coordination with [ChiliPloP 2011](#) at [Spirit in the Desert Lutheran Retreat Center](#) in Carefree, Arizona.

Questions? Contact:

- [Ralph Johnson](#)
- [Kurt Keutzer](#)
- [Tim Mattson](#)

ParaPloP Co-Sponsors:



Meta PLoP



The screenshot shows a web browser window displaying the Meta PLoP website. The browser's address bar shows the URL <http://metapl...>. The website has a navigation menu with 'HOME', 'PAPERS', and 'BLOG'. The main header features the Meta PLoP logo, which consists of a stylized orange and black drop shape next to the text 'Meta PLoP®'. To the right of the logo, the event dates and location are listed: 'June 24th - 26th, 2011' and 'Douro Valley, Portugal'. Below the logo, there is a section titled 'What is MetaPloP?' followed by a paragraph explaining the event's focus on meta-architectures, adaptive object-models, and reflective systems. Another section titled 'How to Prepare for MetaPloP?' provides instructions on reading previous work and identifying relevant patterns. A sidebar on the right contains a scenic photograph of a river valley, social media icons for AOL, Twitter, Facebook, and RSS, and a logo for 'The Refactory, Inc.' which offers training in Design Patterns, J2EE, Java, C#, C++, Refactoring, and Testing.



Scrum PLoP

Scrum Pattern Community

Search this site

ScrumTulipPLoP 2012

▼ ScrumPLoP 2013, Helsingør, Denmark

- ScrumPLoP 2012, Helsingør, Denmark
- ScrumPLoP 2011, Helsingør, Denmark
- ScrumPLoP 2010, Stora Nyteboda, Sweden

Site map

146
days until
ScrumTulipPLoP!

Recent site activity

- [ScrumPLoP 2012, Helsingør, Denmark](#)
edited by James Coplien
- [Program](#)
edited by James Coplien
- [ScrumPLoP 2013, Helsingør, Denmark](#)
edited by James Coplien
- [ScrumPLoP 2013 Attendee Roster](#)
edited by James Coplien
- [ScrumPLoP 2013, Helsingør, Denmark](#)
edited by James Coplien


[View All](#)

ScrumPLoP is a PLoP® Conference sanctioned by the Hillside Group. PLoP is a registered trademark of The Hillside Group.

ScrumPLoP 2013, Helsingør, Denmark

Mark your calendars for 19 May — 24 May, 2013, when we'll come together in [Helenekilde Badehotel](#) in Tisvildeleje, for the sixth ScrumPLoP.

- [Call for Participation](#): What are patterns and pattern languages all about, and what does one do at a PLoP?
- The [ScrumPLoP Mission](#)
- [Attendee roster](#)
- [Program](#)
- [Submission Guidelines](#): How do I write a pattern?
- [About PLoPs](#): The community where patterns grow
- The venue is Helenekilde Badehotel in Tisvildeleje, Denmark. It is on the best beach in Nordsjælland and is close to the grotesque trees of the Troll Forest. It is about 1 hour 40 minutes from Kastrup airport by train. The price per attendee will be €1360, which includes food and lodging.
- [Calendar](#)
- [Pattern Spreadsheet and links to Draft and Published Patterns](#)
- [Works in progress](#) (authors working on drafts go here)
 - [ScrumPLoP Product Backlog](#)
- [Logistics, Cost, Registration, etc.](#)





Community Peer-Reviewed Journal



- **Transactions on Pattern Languages of Programming**
 - The new peer reviewed pattern journal.
 - Published by Springer
 - *More information:* <http://hillside.net/patterns/tplop>
- **Transactions on Pattern Languages of Programming I**
- **Transactions on Pattern Languages of Programming II**
 - Special Issue on Applying Patterns



“Culture”

- “5a: the integrated pattern of human knowledge, belief, and behavior that depends upon man's capacity for learning and transmitting knowledge to succeeding generations
- “5b : the customary beliefs, social forms, and material traits of a religious, or social group
- “5c : the set of shared attitudes, values, goals, and practices that characterizes a company or corporation”
 - Merriam Webster Collegiate Dictionary on the web <http://www.m-w.com/dictionary.htm>
- Shared experiences
- Shared rituals
 - Writers’ Workshop



Writers' Workshop

Writers' Workshops & the Work of Making Things...Gabriel

A circle of interested colleagues, led by a strong, neutral moderator, that provides feedback to the author on how the pattern is understood by the group.

Roles:

- Author
- Moderator
- Summarizer
- Sympathetic Participants



Participants read the pattern before the workshop begins.

The author stands, reads a selection from the pattern, then becomes a “fly on the wall,” outside the circle. No eye contact is made. The author’s name is never mentioned; use “the author.”

Writers' Workshop

- Structured discussion of the merits and suggestions for improvement
- “Criticism” is not welcome -- offer only suggestions for improvement.
 - “I suggest the author add a reference to the paper by Beck and Auer in the Context as it will help establish how their works complement each other.”
- Praise always welcome
 - What people like about a pattern is very important.
- Author is present for note taking but does not participate except at very specific times.
- Strong moderation.

Writers' Workshop

A summary of the pattern is given by someone other than the author or moderator.

Begin with positive comments. Say what the author should leave alone.

Next, suggestions for improvement. State the opportunity and then the suggested improvement.

Those who know the pattern should not clarify or speak for the author. The pattern must stand on its own.

The moderator constrains discussion to the pattern at hand, usually with, “point noted.” Save meta issues for later.

Trivial comments and typos can be made on a marked-up copy for the author after the workshop.



Writers' Workshop

End with a positive closing comment or two.

Invite the author back into the circle.

The author thanks the group for the feedback and only asks questions to clarify comments from the group.

The author should not explain or answer any comments. The author should **never apologize** during this discussion.

The author is considered an expert and is assumed to act appropriately to suggestions. The author owns all comments.

The group stands and applauds the author's contribution.

If another pattern is to be workshopped, everyone takes a different seat and someone tells an unrelated story.

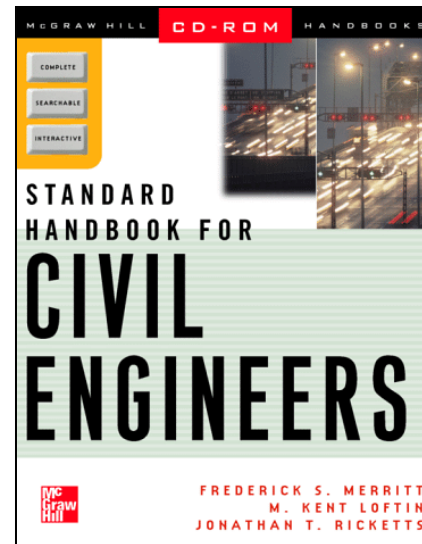
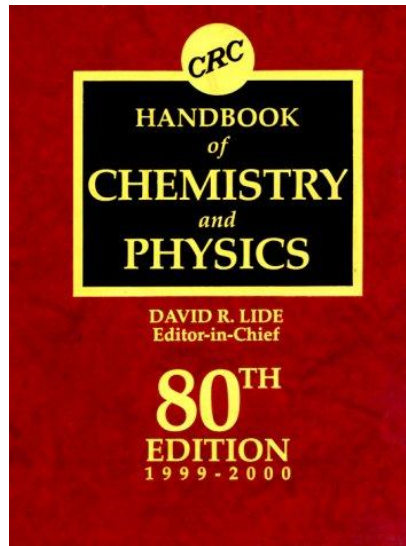


Writers' Workshop

- Richard Gabriel wrote a book on Writers' Workshops
 - <http://www.dreamsongs.com/Files/WritersWorkshop.pdf>
 - He was one of the first that brought this practice into our community as he saw the value from writers in other disciplines such as poetry and the like.
- Jim Coplien has published a pattern language for Writers' Workshops.
 - PLOPD4, chapter 25
 - <http://users.rcn.com/jcoplien/Patterns/WritersWorkshop/>
 - He (and co-author/shepherd Bobby Woolf) delve into the reasons behind way that Writers' Workshops are structured.

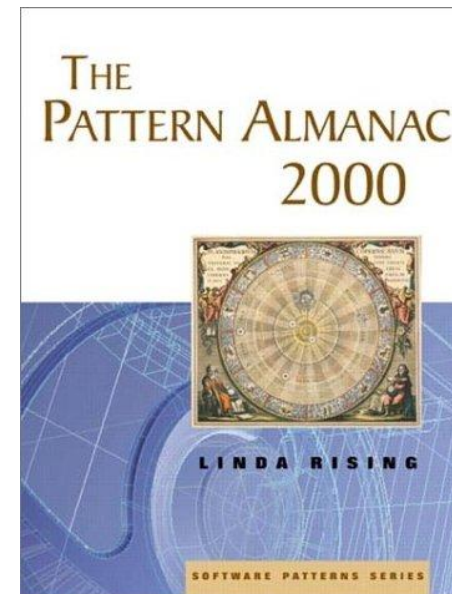
Patterns Handbook

- We use the Handbook Model
 - Patterns available in reference book easily accessible to designers
 - No need to memorize all the details -- the reference book is available



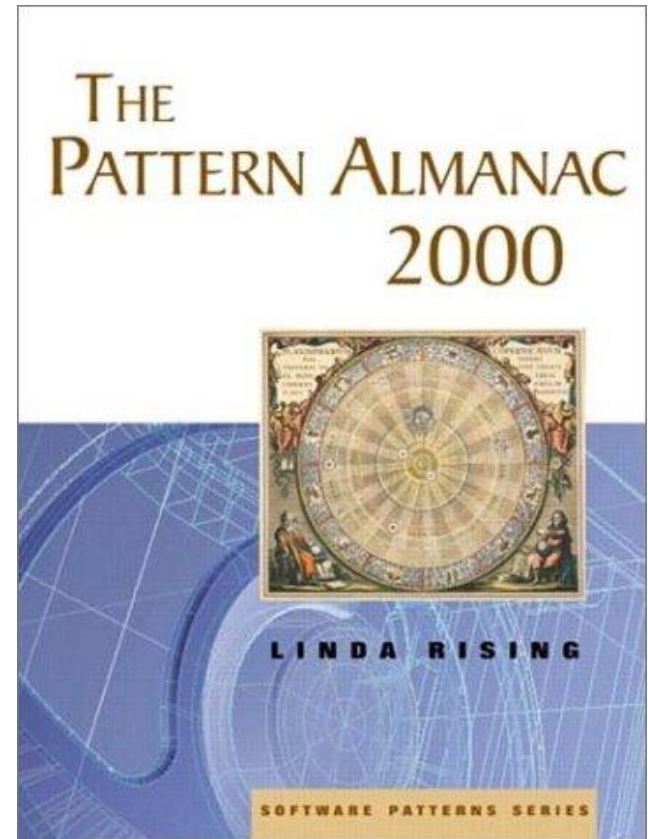
“How do I find a pattern that can help me?”

- Be familiar with existing pattern literature.
- Ask others in pattern community.
- The Pattern Almanac



The Pattern Almanac

- Published 3/2000
- Summarizes all the published, widely available patterns
- Indices based upon pattern intent



Becoming Familiar with Pattern Literature

- Read!
- Participate in online mailing lists
 - frequently new pattern sources (i.e. books or articles) are mentioned
- Attend a conference such as this one.



Pattern Ethics

- Buschmann's rule: Never capture your own ideas in a pattern
 - Focus on broad, lasting, positive patterns
- Intellectual currency paradox: Ideas are valuable if given away
- “Aggressive Disregard for Originality” (*Brian Foote*)
 - Let's encourage people to be secure in telling their secrets
 - Let's reward people who created these techniques or who first took the trouble to commit them to writing
- Patterns are solutions that have withstood the test of time
- Don't Hype!



What's Next

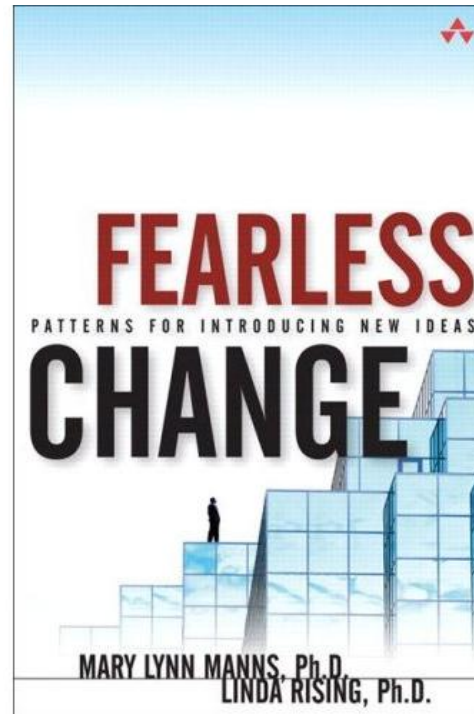
- Refine your pattern with the feedback from your Writers' Workshop
- Take your pattern (and its friends???) to future *PLoP conferences
- Continue writing patterns
- Build a pattern culture within your organization



Introducing Patterns Into Organizations

- Mary Lynn Manns and Linda Rising are collecting patterns related to introducing new technologies into a workplace.
 - <http://www.cs.unca.edu/~manns/intropatterns.html>

- Some examples:
 - *Do Food*
 - *Corporate Sponsor*
 - *Evangelist*
 - *Trial Run*



Selected Bibliography

- Apprenticeship Patterns. Hoover and Oshineye. Sebastapol: O'Reilly, 2010.
- A Pattern Language. Alexander et. Al. New York: Oxford University Press, 1977.
- Design Patterns - Elements of Reusable Object-Oriented Software. Gamma, Helm, Johnson and Vlissides (The "Gang of Four"). Reading, MA: Addison-Wesley, 1995.
- Pattern Oriented Software Architecture. Buschmann, Meunier, Rohnert, Sommerlad and Stal. Chichester, UK: Wiley & Sons, 1996.
- Pattern Oriented Software Architecture, Vol. 2. Schmidt, Stal, Rohnert, Buschmann. Chichester, UK: Wiley & Sons, 2000.
- PLOPD – PLOPD5: Pattern Languages of Program Design, volumes 1-5. Various editors. Reading, MA: Addison-Wesley, 1995-2006.
- The Pattern Handbook. Rising, ed. Cambridge: Cambridge University Press, 1998.
- PLoP Conference Proceedings: <http://hillside.net/plop/>
- The Pattern Almanac 2000. Rising. Reading, MA: Addison-Wesley, 2000.
- Fearless Change. Manns and Rising. Reading, MA: Addison-Wesley, 2004.
- The Timeless Way of Building. Alexander. New York: Oxford University Press, 1979.

For more books: <http://hillside.net/patterns/books/index.htm>



Example pattern

Reading List

Problem

The number of books you need to read is increasing faster than you can read them.

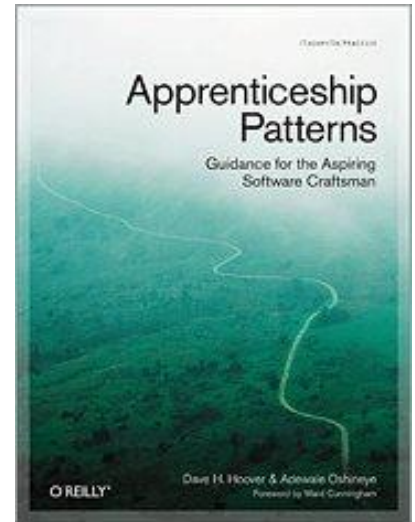
Solution

Maintain a reading list to track the books you plan to read, and remember the books you've read.

Related pattern *Study the Classics*

The ACM has identified a collection of Classics.

<http://portal.acm.org/toc.cfm?id=SERIES11430&type=series&coll=ACM&dl=ACM&CFID=1744172&CFTOKEN=77646249&qualifier=LU1007652>



Example pattern

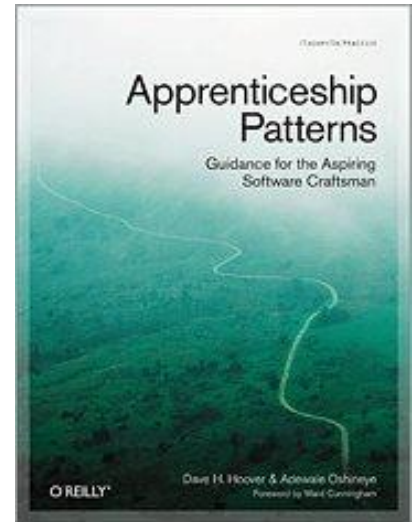
Be the Worst

Problem

Your rate of learning has leveled off

Solution

Surround yourself with developers that are better than you are. Find a stronger team where you are the weakest member and will have room to grow.



Conclusions

- Good writing is not an accident—it comes from dedication, focus, and practice.... <http://www.dreamsongspress.com/>
- Writing patterns is not necessarily hard but can take patience and iterations
- Just do it, thinking about it or theorizing about it will not write the pattern
- Ultimately Pattern Languages are more powerful and useful
- Get regular feedback specifically from the Patterns Community





THE HILLSIDE GROUP

ParaPloP 2011
Workshop on Parallel Programming Patterns
May 10-12 • Carefree, AZ

