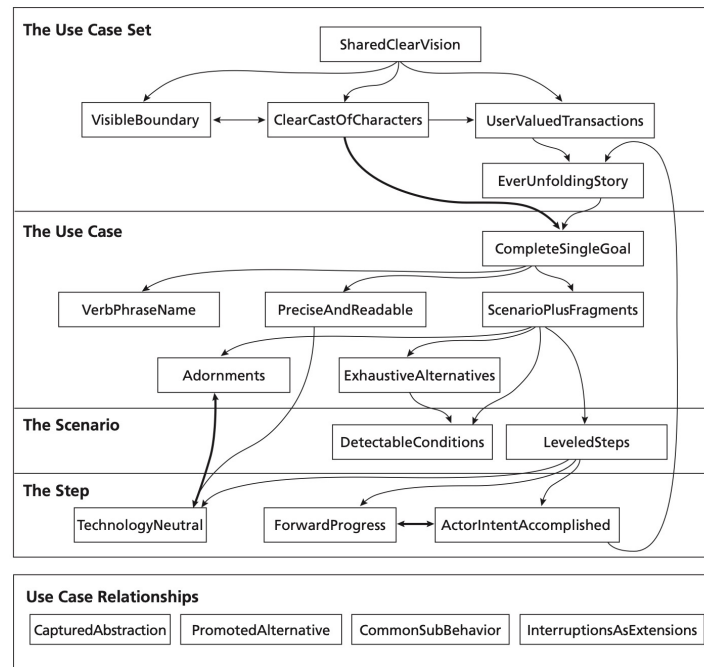


“The take-home value of a pattern is its name” & other observations from 30 years of patterns use



Alistair Cockburn



The Punchline:

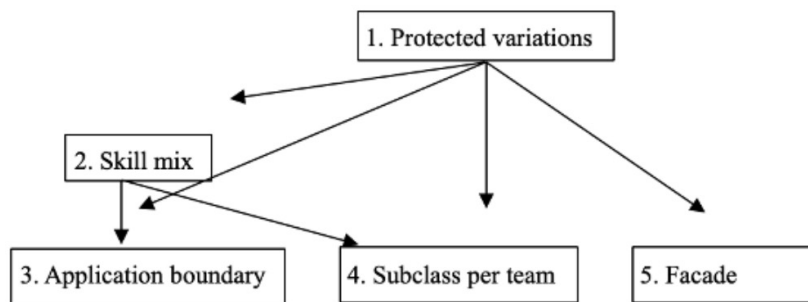
1. Patterns are write-only 😞
2. The take-home value of a pattern is its name.
3. A pattern captures a property or a strategy.
4. Capture its essence in a thumbnail.
5. Expose its side-effects and overdose effects.
6. Consider the medical/pharmaceutical metaphor.
7. Look for patterns that contradict each other.



1995: Prioritizing Forces in Software Design.

“Social Issues in Software Architecture”

Alexander-style pattern language (city-to-baseboards)



Principle Pattern	Design Decisions
1. Protected variations	1.1 Persistence Accessors
2. Skill mix	2.1 Three Subsystems
	2.2 Generic UI, Specific UI
	2.3 Generics and Specifics
3. Application boundary	3.1 Split Validation
	3.2 Edits Inside
4. Subclass per team	4.1 Model Hierarchy
5. Facade	5.1 Domain Intercepts



1996: Patterns of Project Management Risk Reduction (Medical metaphor & diagnosis catalog)

(1) OWNERSHIP: OWNER PER DELIVERABLE

Thumbnail:

Sometimes many people are working on it, sometimes nobody. So...
Make sure every deliverable has exactly one owner.

Indications:

You detect a "common area" (an area where multiple people are updating concurrently, without dominant ownership), or an "orphan area" (an area where no one works or accepts responsibility).
You hear: "What happens when two people need to program the same function?"
Classes are beginning to look like the refrigerator in a shared apartment.
No one is updating the class diagram.
You have multiple teams working on one task, or one person working on many tasks.

Forces being balanced:

You want people to share. You want consistency.
You want every deliverable to have internal integrity, to be consistent and maintained.

...but...

When people share incompletely, you get a common area.
If no one is assigned responsibility, you may get ownership, or a common or orphan area.

Recommended action:

Make someone responsible for each deliverable: project task, project end results, overall consistency. Ask and make sure you can answer, who is answerable for each of these things:

- overall system architecture / application software architecture
- user interface design quality / OO design quality / code quality
- requirements / domain model / each class / documentation/test cases.

Resulting context:

Ownership needs can conflict. You may have to set up a conflict management procedure.

Overdose effect:

The "Swiss accountability" effect: Ownership of every thing on the project can be perceived by some people as wonderful, and others as a nuisance.

Conflict management dominates the team's energy (indicates improper ownership boundaries).

Related patterns:

Function / Component Owners - addresses the particular ownership functions and components.

Team per Task - where a task is the unit of ownership and conflict.

Day Care - addresses training as a distinct deliverable.

Principles involved:

Sometimes a task is so onerous that only by making it part of a person's job responsibility can you ensure that it gets done. This is the case in updating class diagrams and creating documentation, often-found orphan areas.

People cannot track other people's intentions, making a common area hard to clean up. It is also not in any one person's local interests to do so (see Senge 95).

Diagnosis Table

Patterns appearing in this paper are underlined.

General Problem	Indications	Observations	Pattern(s)
Communication	"Our teams are not communicating." "We are doing 'throw it over the wall' development." "Our process too bureaucratic."	Teams are structured by specialty or phase deliverables, passing work to each other by written deliverables, not able to get their discoveries incorporated into connecting teams' work patterns. Hostility or lack of respect across teams.	Holistic diversity (Cockburn 96) Diversity of membership (Harrison 95) Feature Teams (McCarthy 95)
Communication	"I am doing everything"	No specialization. Each person assigned to do everything, resulting in waste from changing gears.	Holistic diversity (Cockburn 96)
Communication	"We are not united in purpose in our technical effort."	People pulling in different technical directions.	Unity of purpose (Harrison 95) Architect controls product (Coplien 94)
Communication	"We have cliques and splinter groups."	Communication distances too large, lack of a core group.	Buffalo mountain (Coplien 94)
Distractions	Team getting distracted. Distractions causing loss of design cycles	Non-primary tasks dominating time, keeping team from moving forward with their primary goal. Possibly, spending too much energy switching contexts.	<u>Someone-Always-Makes-Progress</u> <u>Team per task</u>
Distractions	"This is a distraction to our primary purpose." "This distraction is causing loss of design cycles."	Some important interruption taking time from all the team members. Interruption can be handled by one person	<u>Someone-Always-Makes-Progress</u> <u>Sacrifice One Person</u> <u>Day Care</u> Mercenary analyst (Coplien 94) Firewalls (Coplien 94) Gatekeeper (Coplien 94)
Documentation	"We have too	Too many deliverables	Owner per deliverable



1997: Surviving Object-Oriented Projects

(Patterns are just “strategies”)

1. Knowledge: Clear the Fog

You don't know the issues well enough to put together a sound plan, so...
Try to deliver something (almost anything); this tell you what the real issues are.

2. Knowledge: Early and Regular Delivery

You don't know what problems you will encounter during development, so...
Deliver something early - discover what you don't know you don't know.
Deliver regularly - improve each time.

3. Knowledge: Prototype

You don't know how some design decision will work out, so...
Build an isolated solution - discover how it really works.

4. Knowledge: Microcosm

You have to create a real plan, but have never done this sort of project, so...
Run an 8-12 week instrumented pilot to get productivity and throughput data for
r plan.

5. Teaming: Holistic Diversity

Development of a subsystem needs many skills, but people specialize, so..
Create a single team from multiple specialties.

6. Productivity: Gold Rush

You don't have time to wait for requirements to settle, so...
Start design and programming immediately, adjust their requirements weekly.
(Presupposes *Holistic Diversity*)

7. Ownership: Owner per Deliverable

Sometimes many people are working on it, sometimes nobody, so...
Make sure every deliverable has exactly one owner.
(General strategy with specializations:)

Function Owners / Component Owners
Team per Task

8. Ownership: Function Owners / Component Owners

If you organize teams by components, functions suffer, and vice versa, so...
Make sure every function has an owner, every component has an owner.

9. Distractions: Someone Always Makes Progress

Distractions constantly interrupt your team's progress, so...
Whatever happens, ensure someone keeps moving toward your primary goal.

10. Distractions: Team per Task

A big diversion hits your team, so...
Let a subteam handle the diversion, the main team keeps going.

11. Distractions: Sacrifice One Person

A smaller diversion hits your team, so...
Assign just one person to it until it gets handled.

12. Training: Day Care

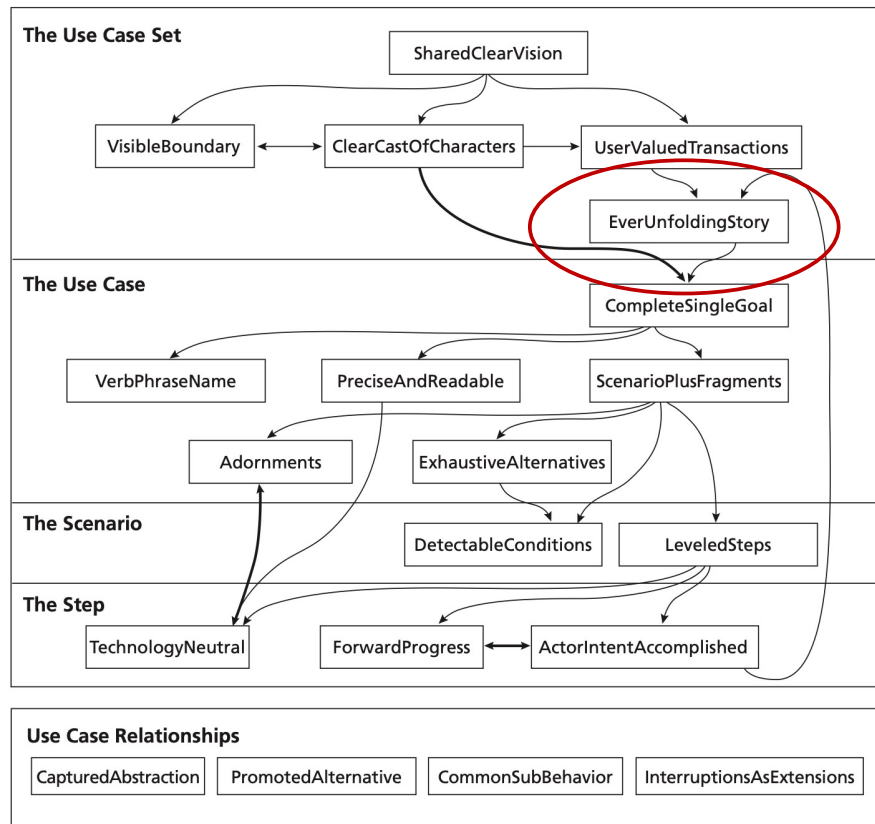
Your experts are spending all their time mentoring novices, so...
Put one expert in charge of all the novices, let the others develop the system.



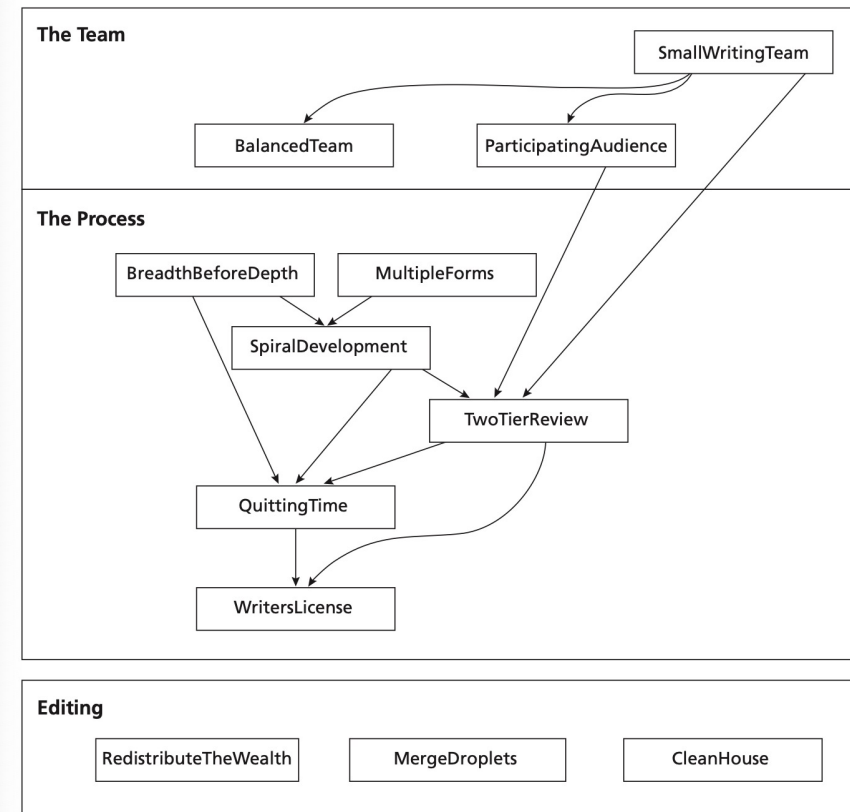
2004: Patterns for Effective Use Cases (32-pattern pattern language)

*by Steve Adolph, Paul Bramble
with Alistair Cockburn, Andy Pols*

Structural Patterns



Development Patterns



2005: Crystal Clear: (Properties vs Strategies)

Chapter 2 *Applied* (The Seven Properties)

- Property 1.* Frequent Delivery
- Property 2.* Reflective Improvement.....
- Property 3.* Osmotic Communication
- Property 4.* Personal Safety
- Property 5.* Focus
- Property 6.* Easy Access to Expert Users.....
- Property 7.* Technical Environment with Automated Tests, Configuration Management & Frequent Integration

Chapter 3 *In Practice* (Strategies & Techniques)

- Strategy 1.* Exploratory 360°
- Strategy 2.* Early Victory
- Strategy 3.* Walking Skeleton
- Strategy 4.* Incremental Rearchitecture.....
- Strategy 5.* Information Radiators



- 1 – Patterns are write-only 🙄

Why is there not more uptake on patterns?

The are dense: They capture experience in a very dense way
add vocabulary to an industry's language
take a lot of processing to learn, remember, integrate

Reading / learning is a language-acquisition activity

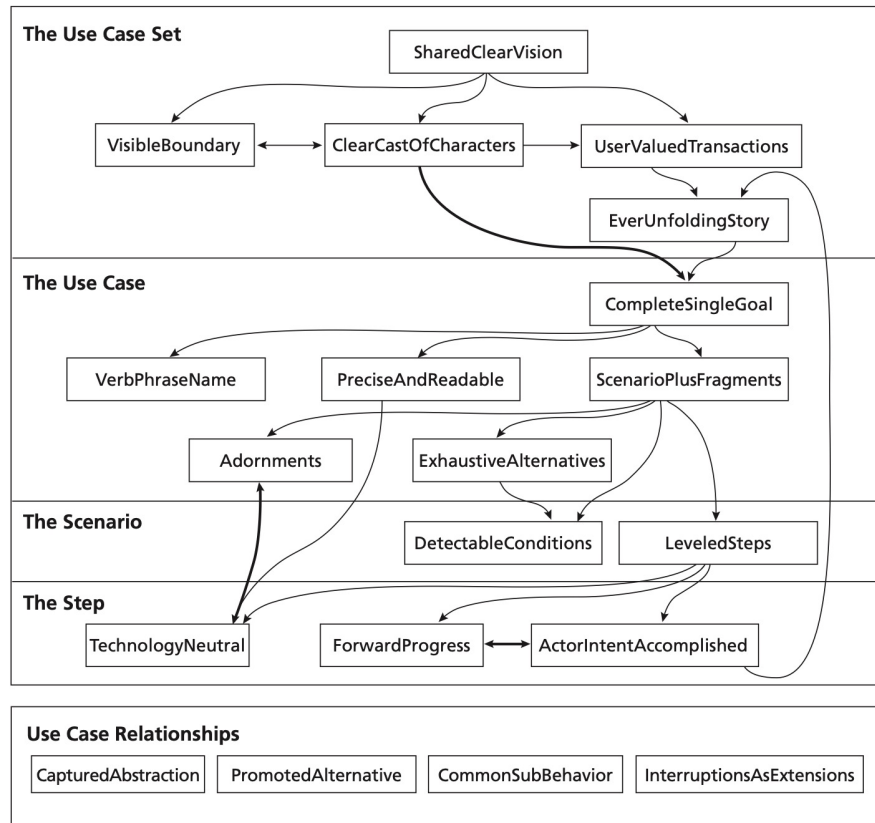
A person can learn perhaps THREE patterns in a workshop.



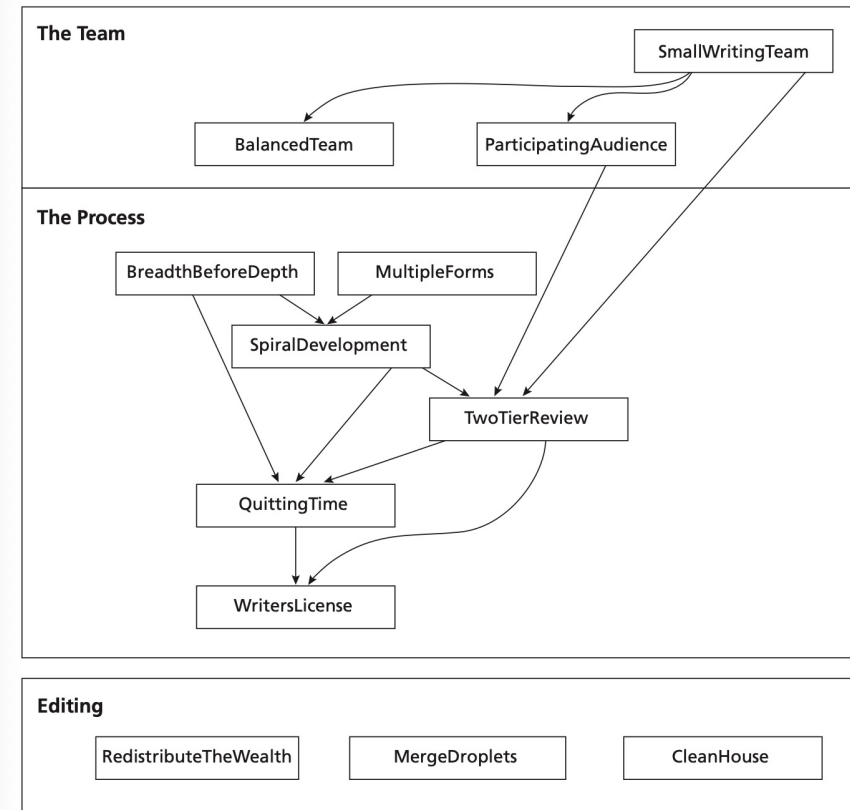
- 1 - Patterns are write-only (1/5)

by Steve Adolph, Paul Bramble
with Alistair Cockburn, Andy Pols

Structural Patterns



Development Patterns



- 1 - Patterns are write-only (2/5)

Buckets & Bath	KISS	Role Rotation
Clear Concept Practice	Learning As Task	Sacrifice One Person
Compatible Tasks	Mentor & Divide	Scrubbed Requirements
Cross SpecializedTeam	Microcosm	Short Horizon planning
Crystallizer	Osmotic Communication	Someone Always Makes Progress
Documented 'No!'(Say No with a Smile)	Owner Per Deliverable	Spare Leader Capacity (Lazy Leader)
Early and Regular Delivery	Pair Programming	Spike
Early Victory	Partner Meeting	Sufficient Demonstrator
Expert In Earshot	Part Timers As Advisors	Tasks Serialized
Folded Zippered Process	Polarized Proposal	Teamed Input
Full Spectral Sample	Pollinator	Team Generated Solution
Function Owners Component Owners	Process In View	Team per Task
Gold Rush	Process Miniature	Technology Check
Holistic Diversity	Progress Team Training Team	Timeout Regroup
Incremental Rearchitecture	Prototype	Train By Spec'ing
InMemory Imposter	Rejoining Streams	Twinning
Just One Functional Architect	Replace The Team	Walking Skeleton



- 1 - Patterns are write-only (3/5)

"Cone of Silence"

7

Anti-venom / Repair

Counterexample to *Osmotic Communication*

Related to *Focus Time*

Symptoms: Constant interruptions, can't manage *Focus Time*.

- Isolate key person/team so no one communicates with them for stretches of time.

Individual version: Move team lead upstairs (for hours each day).

Team version: Move team offsite (e.g. IBM PC development, 1980)

<http://alistair.cockburn.us/The+cone+of+silence+and+related+project+management+strategies>

Slide 14

©Alistair Cockburn 2009



8

"Sacrifice One Person"

Anti-venom / Repair

Reading: *Surviving OO Projects* (Cockburn)

- When a series of interruptions hit the project, dedicate 1 person (team) to handle it, so all the rest can *focus* on their tasks uninterrupted
- Take turns being the "sacrifice".

Better 1 person working at less than max capacity than 4 people interrupted daily.

Variations:

- : SWAT team does bug fixes on monthly rotation.
- : One person does QA assignment monthly rotation

Examples from your life:

Non-obvious, invented in various forms & variations.

Slide 19

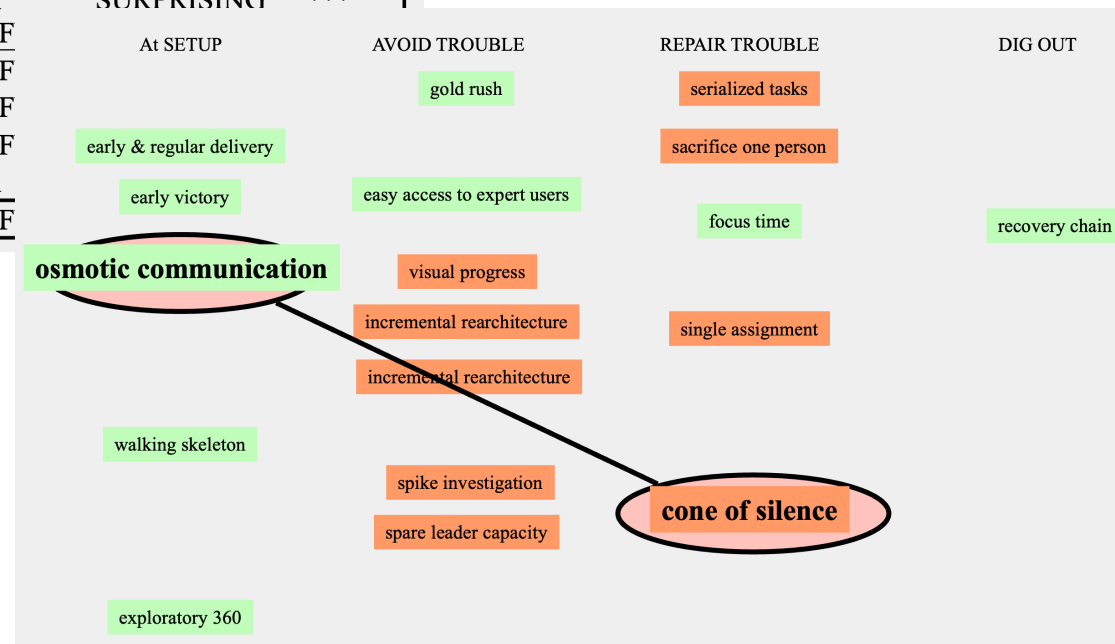
©Alistair Cockburn
2005-9

©Alistair Cockburn 2023

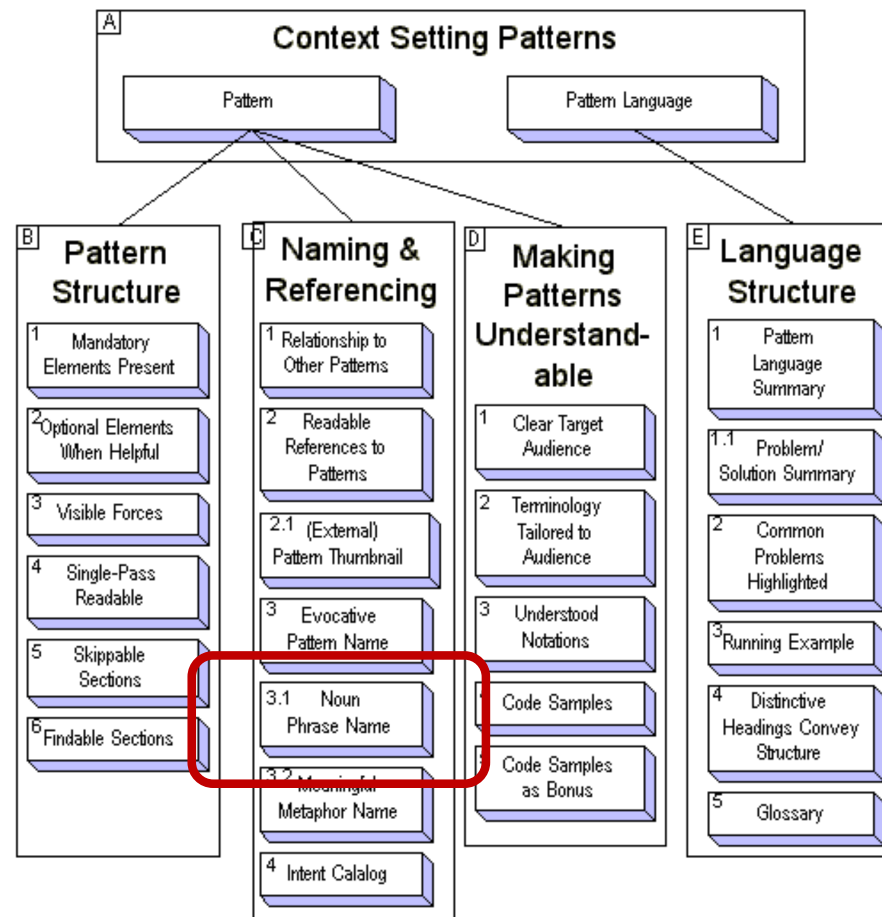


- 1 - Patterns are write-only (4/5)

Timing	Name	Value	Power	Surprise	Rank
EARLY	Early & Regular Delivery	nutrition	POWERFUL	surprising	***
	Early Victory	nutrition	POWERFUL	surprising	***
	Walking Skeleton	nutrition	powerful	SURPRISING	***
AVOID	Gold Rush	nutrition	POWERFUL	SURPRISING	****
	Burning Visibility	ANTIVENOM	POWERFUL	surprising	***
	Incremental Rearchitecture	ANTIVENOM	powerful	SURPRISING	***
	Spare Leader Capacity	ANTIVENOM	powerful	SURPRISING	***
	Easy Access to Expert Users	ANTIVENOM	POWERFUL		
REPAIR	Sacrifice One Person	ANTIVENOM	POWERFUL		
	Focus Time	nutrition	POWERFUL		
	Single Assignment	ANTIVENOM	POWERFUL		
	Cone of Silence	ANTIVENOM	powerful		
DIG OUT	Recovery Chain	nutrition	POWERFUL		



- 1 - Patterns are write-only (5/5)



Legend:

<https://hillside.net/index.php/a-pattern-language-for-pattern-writing>



- 1 - Net:

Learning patterns is a language-acquisition activity

A person can learn perhaps THREE.



- 2 -The take-home value of a pattern is its name.

It add vocabulary to a professional language.

Choose the name carefully

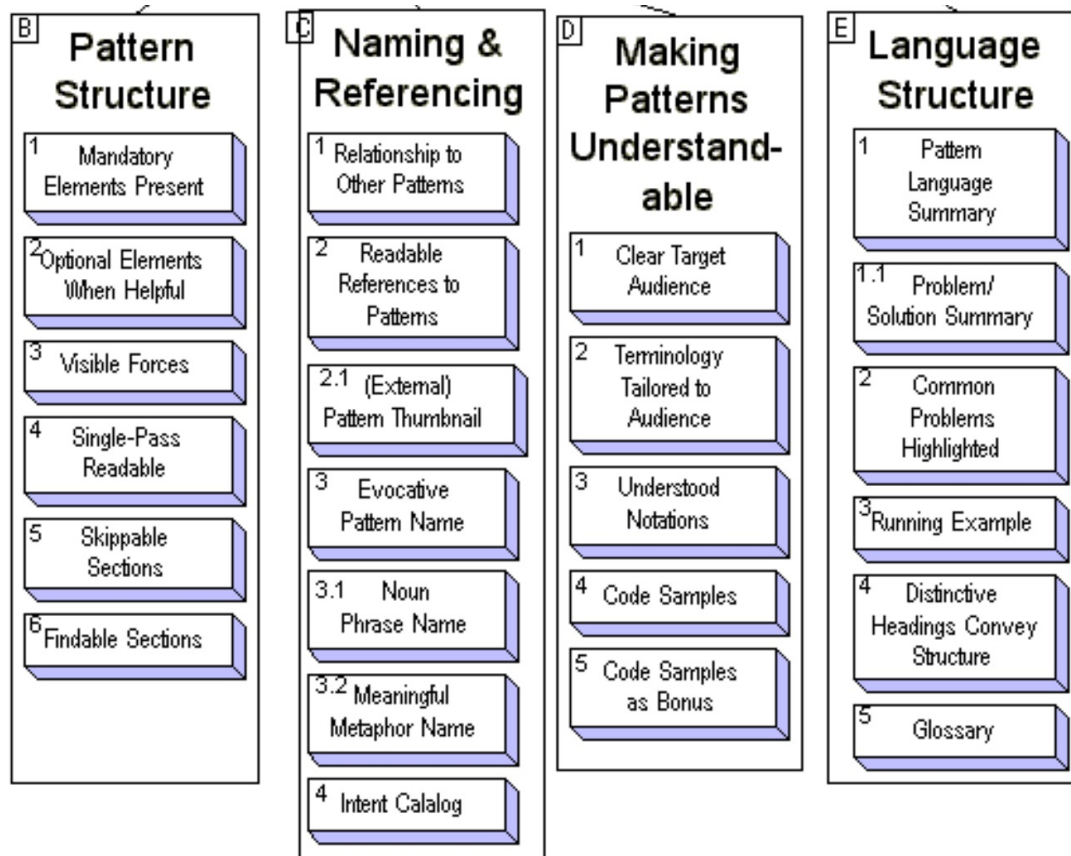
A pattern captures
common **characteristics** of good examples of _____

Generally a noun.

(Occasionally a verb for a process recommendation)



- 2 -The take-home value of a pattern is its name.



<https://hillside.net/index.php/a-pattern-language-for-pattern-writing>



- 2 -The take-home value of a pattern is its name.

Names that work

Light on two sides of a room

Walking skeleton

Incremental rearchitecture

Osmotic communication

Cone of silence

Names that don't work

Buffalo mountain

Inversion of control / Dependency inversion

Anti-corruption layer

Hexagonal architecture



- 2 - Naming riddle solved:

Phrase a process sequence as a single property

What you're thinking:

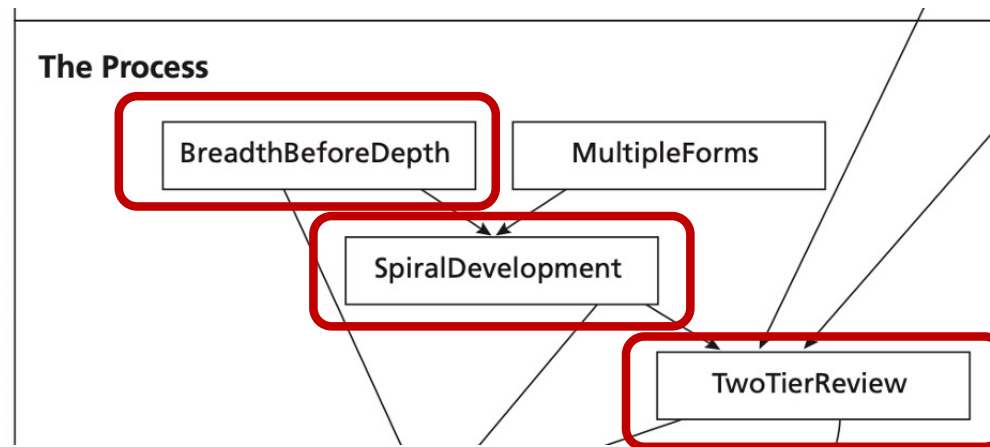
Step 1: Do this

Step 2: Do that

Step 3: Do the other

The property:

“Three-step sequence”



- 3 - A pattern captures a property or a strategy

A pattern captures
common characteristics of good examples of _____

Chapter 2 Applied (The Seven Properties)

<i>Property 1.</i>	Frequent Delivery
<i>Property 2.</i>	Reflective Improvement.....
<i>Property 3.</i>	Osmotic Communication
<i>Property 4.</i>	Personal Safety
<i>Property 5.</i>	Focus
<i>Property 6.</i>	Easy Access to Expert Users
<i>Property 7.</i>	Technical Environment with Automated Tests, Configuration Management & Frequent Integration

Light on two sides of a room
(it has this property)

Chapter 3 In Practice (Strategies & Techniques)

<i>Strategy 1.</i>	Exploratory 360°
<i>Strategy 2.</i>	Early Victory
<i>Strategy 3.</i>	Walking Skeleton
<i>Strategy 4.</i>	Incremental Rearchitecture.....
<i>Strategy 5.</i>	Information Radiators

Decorator / Composite
(we apply this strategy)



- 4 - Capture its essence in a thumbnail

Distractions: Someone Always Makes Progress

Distractions constantly interrupt your team's progress, so...
Whatever happens, ensure someone keeps moving toward your primary goal.

Distractions: Team per Task

A big diversion hits your team, so...
Let a subteam handle the diversion, the main team keeps going.

Distractions: Sacrifice One Person

A smaller diversion hits your team, so...
Assign just one person to it until it gets handled.



- 4 - Capture its essence in a thumbnail

Hexagonal Architecture / Ports & Adapters

Create your application to work without either a UI or a database so you can run automated regression-tests against the application, work when the database becomes unavailable, and link applications together without any user involvement.



- 5 - Expose its side-effects and overdose effects

(11) DISTRACTIONS: SACRIFICE ONE PERSON

Other names:

"Sacrificial Lamb"

Thumbnail:

A smaller diversion hits your team, so...

Assign just one person to it until it gets handled.

Indications:

As for *Team per Task*, except the distraction is smaller; it could seemingly be handled by one person half-time to full-time.

Forces being balanced:

Need to pay attention to every task, including small diverting ones.

Need to complete the primary task by an important date.

Desire to have to be satisfied with their jobs.

Factors affecting the balancing of the forces:

It takes significant time for people to switch between tasks.

There is such a thing as the "primary" task.

The diverting task seems to be small, but is very important.

Recommended action:

Assign one person full-time to the distraction rather than several people part time. If one person is insufficient, assign as few as possible.

Resulting context:

The main group of the team moves forward distraction free. The person assigned to the distracting task may be unhappy, so try to get that person back on the team again as soon as possible. If you feel that one person is too much to sacrifice to this task and want to make it part time work, compute the loss of flow time that would result from trying to work this distraction and some other task.

Overdose effect:

If this keeps happening, you will have no one performing the primary task, and you ought to examine why you have so many distractions in the first place.



- 5 - Expose its side-effects and overdose effects

Strategy: **Walking Skeleton**

Action: Connect the architecture with a very simple function as 1st running code.
(Grow infrastructure & function in parallel).

Benefits: **Early Victory**,
Architecture runs,
Parallel development possible

Side effect: Manage rework of infrastructure during function development

Overdose: If too simple, rework delays project

Strategy: **Early Victory**

Action: Ensure the team delivers *something* as *early* as possible, even if only a Walking Skeleton.

Benefits: Team learns each other, process
Sponsor sees team working together

Side effect: Manage expectations for early delivery

Overdose: Too simple = Not enough process exercised, not *real* confidence built

Examples: **Walking Skeleton**



- 6 - Consider the medical/pharmaceutical metaphor

Name

- Thumbnail
- Indications (& Observations)
- Counterindications
- Forces
- Recommended action
- Side effects (Resulting context)
- Overdose effects

(1) OWNERSHIP: OWNER PER DELIVERABLE

Thumbnail:

Sometimes many people are working on it, sometimes nobody. So...
Make sure every deliverable has exactly one owner.

Indications:

You detect a "common area" (an area where multiple people are updating concurrently, without dominant ownership), or an an "orphan area" (an area where no one works or accepts responsibility).

You hear: "What happens when two people need to program the same function?"

Classes are beginning to look like the refrigerator in a shared apartment.

No one is updating the class diagram.

You have multiple teams working on one task, or one person working on many tasks.

Forces being balanced:

You want people to share. You want consistency.

You want every deliverable to have internal integrity, to be consistent and maintained.

...but...

When people share incompletely, you get a common area.

If no one is assigned responsibility, you may get ownership, or a common or orphan area.

Recommended action:

Make someone responsible for each deliverable: project task, project end results, overall consistency. Ask and make sure you can answer, who is answerable for each of these things:

- overall system architecture / application software architecture
- user interface design quality / OO design quality / code quality
- requirements / domain model / each class / documentation/ test cases.

Resulting context:

Ownership needs can conflict. You may have to set up a conflict management procedure.

Overdose effect:

The "Swiss accountability" effect: Ownership of every thing on the project can be perceived by some people as wonderful, and others as a nuisance.

Conflict management dominates the team's energy (indicates improper ownership boundaries).

Related patterns:

Function / Component Owners - addresses the particular ownership functions and components.

Team per Task - where a task is the unit of ownership and conflict.

Day Care - addresses training as a distinct deliverable.

Principles involved:

Sometimes a task is so onerous that only by making it part of a person's job responsibility can you ensure that it gets done. This is the case in updating class diagrams and creating documentation, often-found orphan areas.

People cannot track other people's intentions, making a common area hard to clean up. It is also not in any one person's local interests to do so (see Senge 95).



- 6 - The medical metaphor suits large quantities of patterns

Diagnosis Table

Patterns appearing in this paper are underlined.

General Problem	Indications	Observations	Pattern(s)
Communication	"Our teams are not communicating." "We are doing 'throw it over the wall' development." "Our process too bureaucratic."	Teams are structured by specialty or phase deliverables, passing work to each other by written deliverables, not able to get their discoveries incorporated into connecting teams' work patterns.	Holistic diversity (Cockburn 96) Diversity of membership (Harrison 95) Feature Teams (McCarthy 95)
Distractions	"This is a distraction to our primary purpose." "This distraction is causing loss of design cycles."	Some important interruption taking time from all the team members. Interruption can be handled by one person	<u>Someone-Always-Makes-Progress</u> <u>Sacrifice One Person</u> <u>Day Care</u> Mercenary analyst (Coplien 94) Firewalls (Coplien 94) Gatekeeper (Coplien 94)
	Distractions causing loss of design cycles	team from moving forward with their primary goal. Possibly, spending too much energy switching contexts.	<u>Team per task</u>
Distractions	"This is a distraction to our primary purpose." "This distraction is causing loss of design cycles."	Some important interruption taking time from all the team members. Interruption can be handled by one person	<u>Someone-Always-Makes-Progress</u> <u>Sacrifice One Person</u> <u>Day Care</u> Mercenary analyst (Coplien 94) Firewalls (Coplien 94) Gatekeeper (Coplien 94)
Documentation	"We have too	Too many deliverables	Owner per deliverable

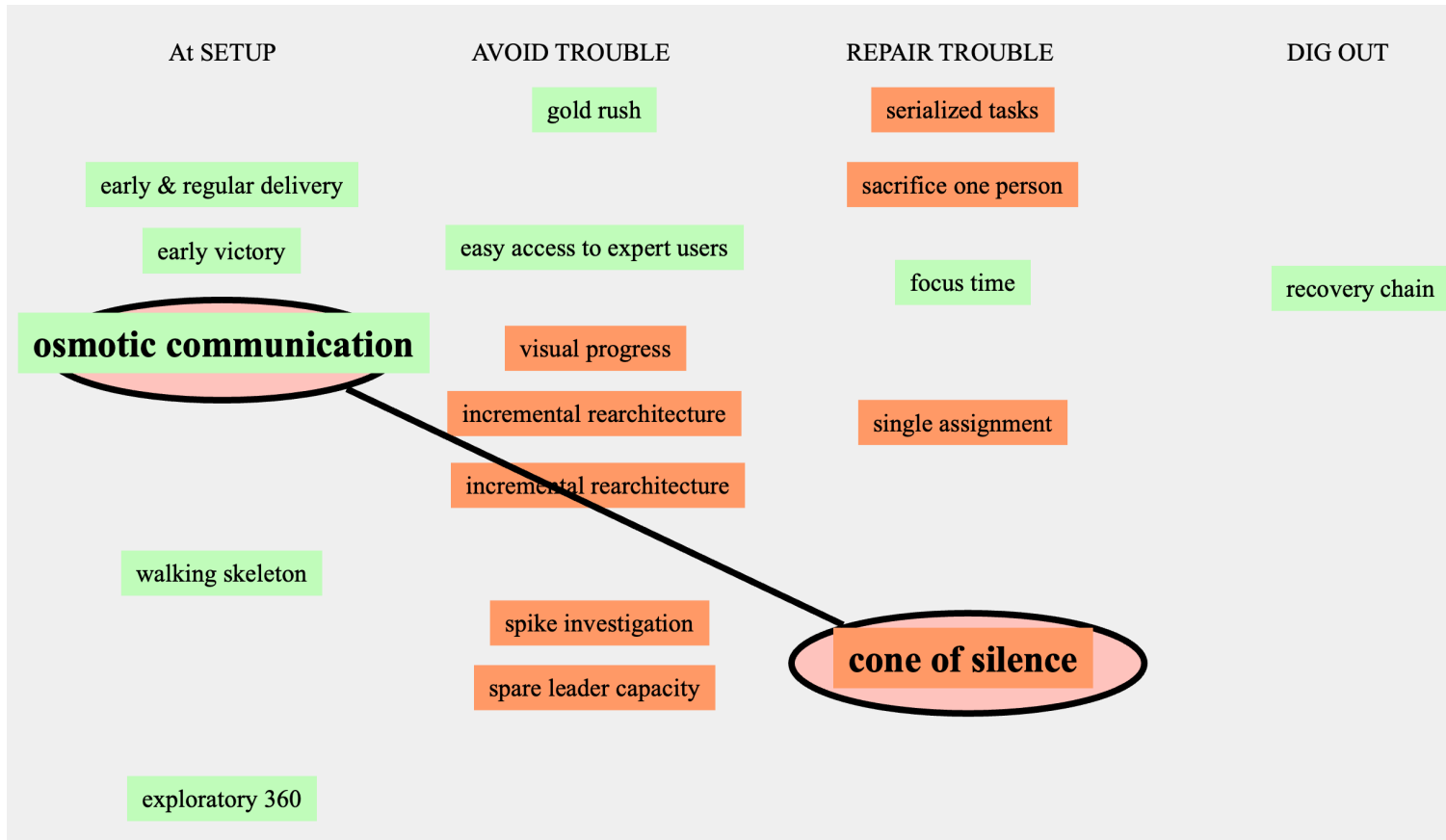


- 6 - The medical metaphor suits large quantities of patterns

#	SIG	Category	Name	Brief	AL	Indications	CounterIndications	Forces...and...	Forces...but...	Action	SideEffects	OverdoseEffect	Commentary
4	1	Design	Walking Skeleton	Start by connecting the pieces of the system, so that it runs but with the little or no functionality. Build onto that running structure over time.	0								
7	1	Distractions	Sacrifice One Person	Assign just one person to handle smaller diversions, rather than several part time.	1	This interruption is causing us to lose precious, active cycles. We are getting distracted from our primary purpose. A single non-primary task is taking noticeable time from a number of the team member, keeping them from moving forward with their primary goal.		you cannot drop the distracting task, it is somehow very important.	it is keeping the team from moving on its primary task.	Assign one person to the distraction. Let the rest of the team move forward at full speed. If one person is insufficient, assign as few as possible.	If this keeps happening, you will have no one performing the primary task, and you ought to examine why you have so many distractions in the first place.		The principle involved here is to protect the main task, where constant progress is paramount. In Situation 1, above, it was not that the design teams were complaining of the loss of flow during design - they were simply not doing design, because of the need to do scheduling.
9	1	Distractions	Spare Leader Capacity	Ensure the technical lead has some spare capacity to assist in emergencies.	1								
13	1	Knowledge	Early and Regular Delivery	Deliver something early - discover what you don't know you don't know. Deliver regularly - improve each time.	1								



- 7 - Look for patterns that contradict each other



Why didn't I write this as patterns?

Patterns are write-only

These notes are counsel



End:

1. **Patterns are write-only** 😞.
2. **The take-home value of a pattern is its name.**
3. A pattern captures a property or a strategy
4. Capture its essence in a thumbnail.
5. Expose its side-effects and overdose effects.
6. Consider the medical/pharmaceutical metaphor
7. Look for patterns that contradict each other.

