



A  
Systems  
Engineering  
Pattern  
Language

Bob Barter  
August 1998

# Here is a common problem

- You are about to give a talk on a complex subject that is new to your audience
- The audience is friendly and interested
- You do not want the audience to get lost, but you do not want to waste a lot of time on orientation material
- Consider telling them three times: a short introduction, the main message, and a summary
- Consider also: providing markers, and embedding an example in the talk

# Tell Them Three Times

## CONTEXT:

The audience is friendly and interested

## PROBLEM:

You are about to give a talk on a complex subject that is new to your audience

## FORCES AND TENSIONS:

You do not want the audience to get lost, but you do not want to waste a lot of time on orientation material

## SOLUTION:

A short introduction, the main message, and a summary

## RELATED PATTERNS:

*Providing Markers*, and *Embedding an Example* in the talk

# Agenda

- History
- Definitions
- Examples
  - Form
  - Content
- Conclusion



# “Pattern Languages” are a fairly recent development

- Christopher Alexander, et al
  - The Oregon Experiment, 1975
  - A Pattern Language: Towns, Buildings, Construction, 1977
- Eric Gamma, et al
  - Design Patterns : Elements of Reusable Object-Oriented Software, 1995
- Michael A. Beedle
  - Enterprise Architectural Patterns; Building Blocks of the Agile Company, 1998



# Agenda

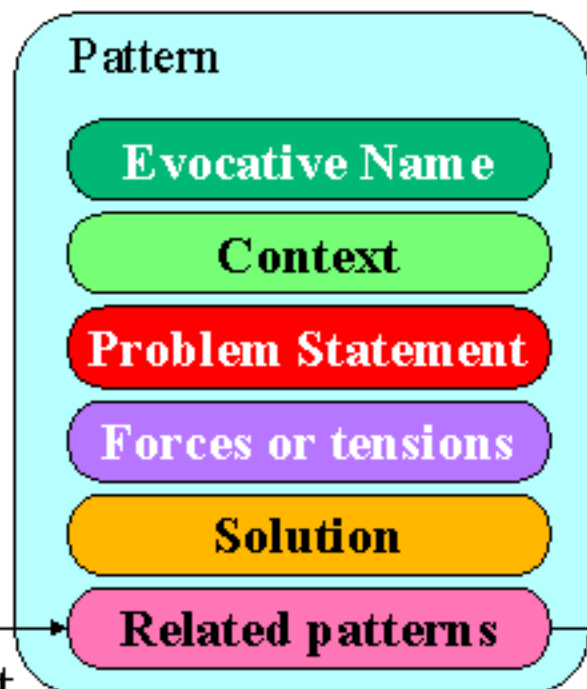
- History
- Definitions
- Examples
  - Form
  - Content
- Conclusion



# Patterns, Pattern Languages and Pattern Maps

A pattern is a solution to a problem within a context

- An evocative name
- Context
- Problem statement
- Forces or tensions
- Solution
- Related patterns



This is only one format

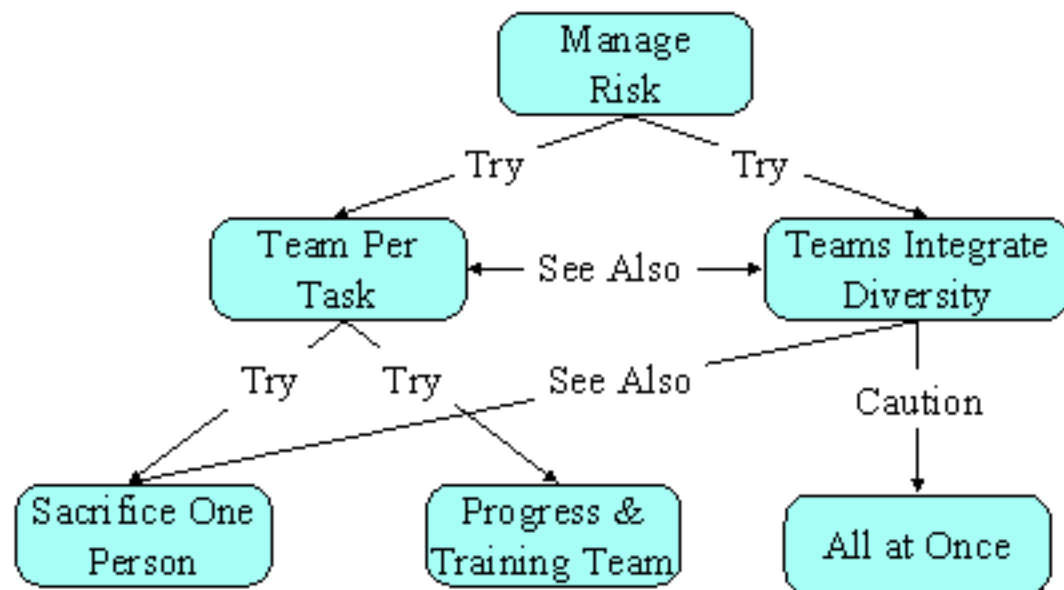
# Patterns, Pattern Languages and Pattern Maps

A Pattern Language is a collection of patterns

- Evocative names help abstract underlying meanings (Sacrifice One Person)
- Context establishes appropriate use (the team is being distracted by ...)
- Forces and tensions guide the selection of alternative patterns
- Related patterns encourage further exploration of similar concepts

# Patterns, Pattern Languages and Pattern Maps

Pattern maps help organize patterns



# Agenda

- History
- Definitions
- Examples
  - Form
  - Content
- Conclusion



# Examples of existing pattern forms

- Degenerate form - without any discernible template
- Alexanderian form
  - Name
  - Problem
  - Solution

Brown, William; Malveau, Palphael; McCormick, Hays, and Mowbray, Thomas, *Anti-Patterns, Refactoring Software, Architectures, and Projects in Crisis*, New York: John Wiley & Sons, Inc., 1997

# Examples of existing pattern forms

- Inductive mini-pattern form - focuses on the applicability of the pattern
  - Name
  - Context
  - Forces
  - Solution

# Examples of existing pattern forms

- Deductive mini-pattern form - focuses on the outcomes of the pattern
  - Name
  - Problem
  - Solution
  - Benefits
  - Consequences

# Still more examples of pattern forms

- “Gang of Four - micro architecture level

Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissides, John, *Design Patterns*, Reading, MA: Addison-Wesley, 1994

- System of Patterns - multiple levels - idioms level, applications level, system level

Buschman, Frank; Meunier, Regine; Rohnert, Hans; Somerland, Peter; and Stal, Michael, *Pattern-Oriented Architecture: A System of Patterns*, New York: John Wiley & Sons, Inc., 1996

- CORBA Design Patterns - concise sections first, free-form sections last

Mowbray, Thomas J.; and Malveau, Palphael C., *CORBA Design Patterns*, New York: John Wiley & Sons, Inc., 1997

# Software Anti-Pattern Form

- What are the two most common software problems? See:
  - The Blob
  - Poltergeist
- What can we do to fix (or refactor) bad software? See:
  - Spaghetti Code
  - Stovepipe Systems

Brown, William; Malveau, Palphael; McCormick, Hays; and Mowbray, Thomas, *Anti-Patterns, Refactoring Software, Architectures, and Projects in Crisis*, New York: John Wiley & Sons, Inc., 1997

# Agenda

- History
- Definitions
- Examples
  - Form
  - Content
- Conclusion



# The Pursuit of Wow!

- Tom Peters - "...to thrill and delight the customer in the pursuit of Wow!"
- TQM - The concept of "total customer satisfaction"
- Christopher Alexander - Quality Without A Name

Attributes of a system that make the system "live"

*Flexible*

*Freedom*

*Harmony*

*Extensible*

*Life*

*Adaptable*

*Wholeness*

*Reusable*

*Comfortability*

# Quality Without A Name

- Universally recognizable aesthetic
- Recursively nested centers of symmetry
- Wholeness
- Resilience, adaptability, and durability
- Human comfort and satisfaction
- Emotional and cognitive resonance

Timeless attributes which are considered beautiful and aesthetically pleasing to all people and all cultures

# An example from civil architecture

## Building Complex, Wings of Light, Arcades, Circulation Realms

- Name: *Paths and Goals*
- Problem: The layout of a path will seem right and comfortable only when it is compatible with the process of walking
- Solution: ... place goals at natural points of interest, ... connect the goals to one another to form the paths, ... paving should swell around the goal, ... goals should never be more than a few hundred feet apart

## Family of Entrances, Tree Places, Seat Spots, Path Shape



Alexander, Christopher; Ishikawa, Sara; Silverstein, Murray, *A Pattern Language*,  
Oxford: Oxford University Press, 1977

# An example from civil architecture

## Promenade, Raised Walk, Pedestrian Street, Paths and Goals



- Name: *Path Shape*
- Problem: Streets drive people out of the street instead of attracting them in. ... the pedestrian world outside houses must be made into the kind of place where you stay rather than move through.
- Solution: Make a bulge in the middle of a public path, so that the path forms an enclosure which is a place to stay, not just a place to pass through.

## Arcades, Activity Pockets, Stair Seats, Public Outdoor Room

Alexander, Christopher; Ishikawa, Sara; Silverstein, Murray, *A Pattern Language*,  
Oxford: Oxford University Press, 1977

# Object Oriented Modeling and Design has some interesting patterns

*RAPPeL: A Requirements Analysis Process Pattern Language for Object Oriented Development* by Bruce G. Whitenack, Jr.  
has the following patterns:

Customer Expectations	Object Aging
Customer Rapport	Object Stereotypes
Sponsor Objectives	Behavioral Requirements
Defining Requirements	Requirements Specification
Problem Domain Analysis	Business Rules
Information Needs	Pragmatic External Requirements
Finding And Defining The Domain Objects	Prototypes
Classifying Associating And Grouping The Domain Objects	Requirements Validation
Elaboration Of Domain Objects	

<http://www2.umassd.edu/SWPI/ATT/pattern/rapel.html>

# An example from RAPPel

- Name: Build the Right Things
- Problem: How do you capture, communicate and validate software requirements so that successful systems that “do the right things” can be built?
- Discussion: Customers cannot adequately express their requirements while developers have difficulty understanding what the customer needs.
- Solution: identify and categorize Requirements Sources ... interview and examine these sources ... capture and validate ...




Sponsor Objectives, Customer Expectations, Customer Rapport

# An example from RAPPel

## Build the Right Thing



- Name: Customer Expectations
  - Problem: How do you meet and manage customer expectations for a product?
  - Discussion: Requirements can always be more complete or more precisely defined. It is not possible to assure a product will completely meet customer expectations through an single massive attempt to completely specify the requirements.
  - Solution: ... produce a List of Customer Expectations ... classify them as to whether they are real requirements (must haves) or a customer wish (like to have) ...
- 

Behavioral Requirements, Pragmatic External Requirements,  
Requirements Specification

# An example from RAPPel

## Customer Expectations



- Name: Pragmatic External Requirements
- Problem: What are the non-behavioral and organizationally imposed behavioral constraints?
- Discussion: During development there can be a number of constraints placed on a system
- Solution: Use a Pragmatic Requirements Template ...  
Review the constraints with all groups that will be involved in the delivery, installation, training and implementation of the product.

# Non-Software Examples of Software Design Patterns

- Design Patterns - Elements of Reusable Object-Oriented Software<sup>1</sup> captured many software design patterns
- There is a move to extend the patterns outside the software realm<sup>2</sup>

(1) Gamma, E., Helm, R., Johnson, R., Vlissides, J.

Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995

(2) "Non-Software Examples of Software Design Patterns,"

Object Magazine, Vol. 7, No. 5, July 1997, pp. 52-57 and

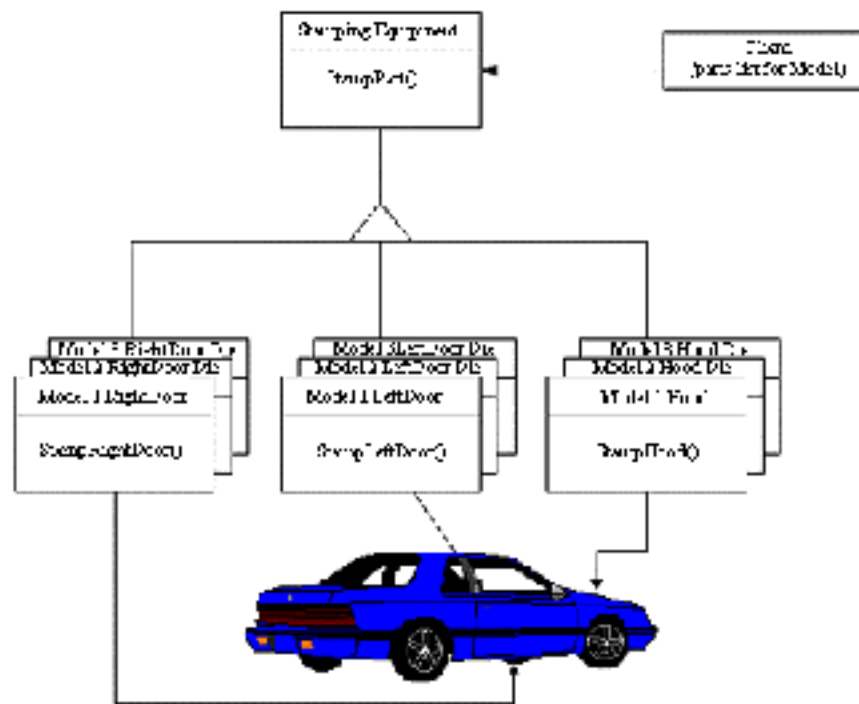
<http://www.agcs.com/patterns/papers/patexamples.htm>

# Non-Software Examples of Software Design Patterns

## Abstract Factory

Provide an interface for creating families of related objects, without specifying concrete classes.

The stamping equipment is an Abstract Factory which creates auto body parts.

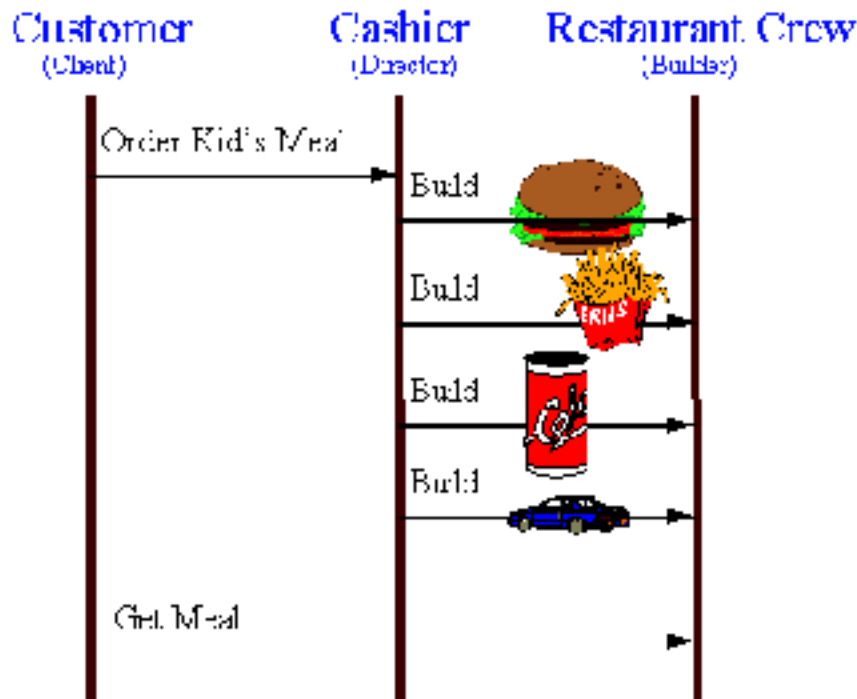


# Non-Software Examples of Software Design Patterns

## Builder

Separate the construction of a complex object from its representation, so that the same construction process can create different representations.

Used by fast food restaurants to construct children's meals.

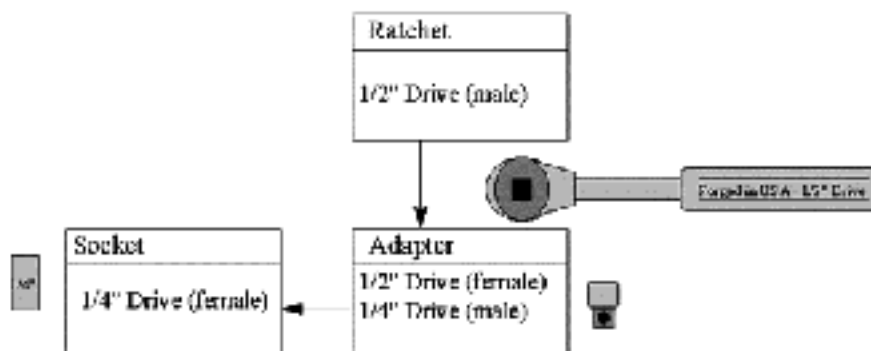


# Non-Software Examples of Software Design Patterns

## Adapter

Allow otherwise incompatible classes to work together by converting the interface of one class into an interface expected by the clients.

A socket attaches to a ratchet, provided that the size of the drive is the same.

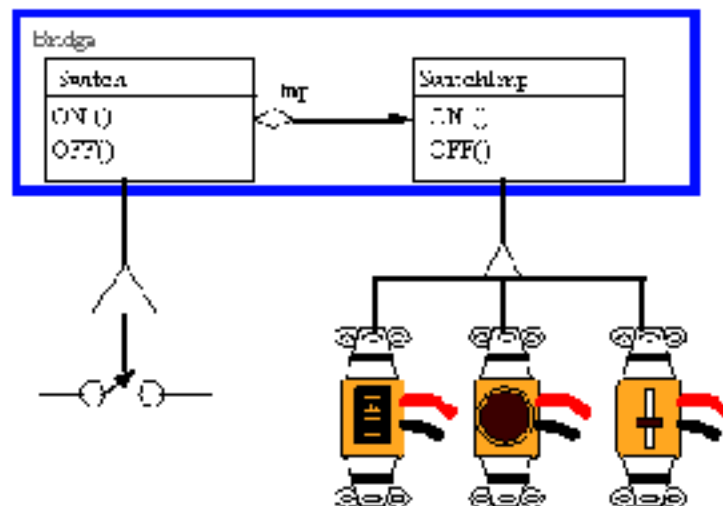


# Non-Software Examples of Software Design Patterns

## Bridge

De-couple an abstraction from its implementation, so that the two can vary independently.

The purpose of the switch is to turn a device on or off. The actual switch can be implemented as a pull chain, a simple two position switch, or a variety of dimmer switches.



# Systems Engineering already has a pattern language in DOD 4245.7-M and NAVSO P-6071

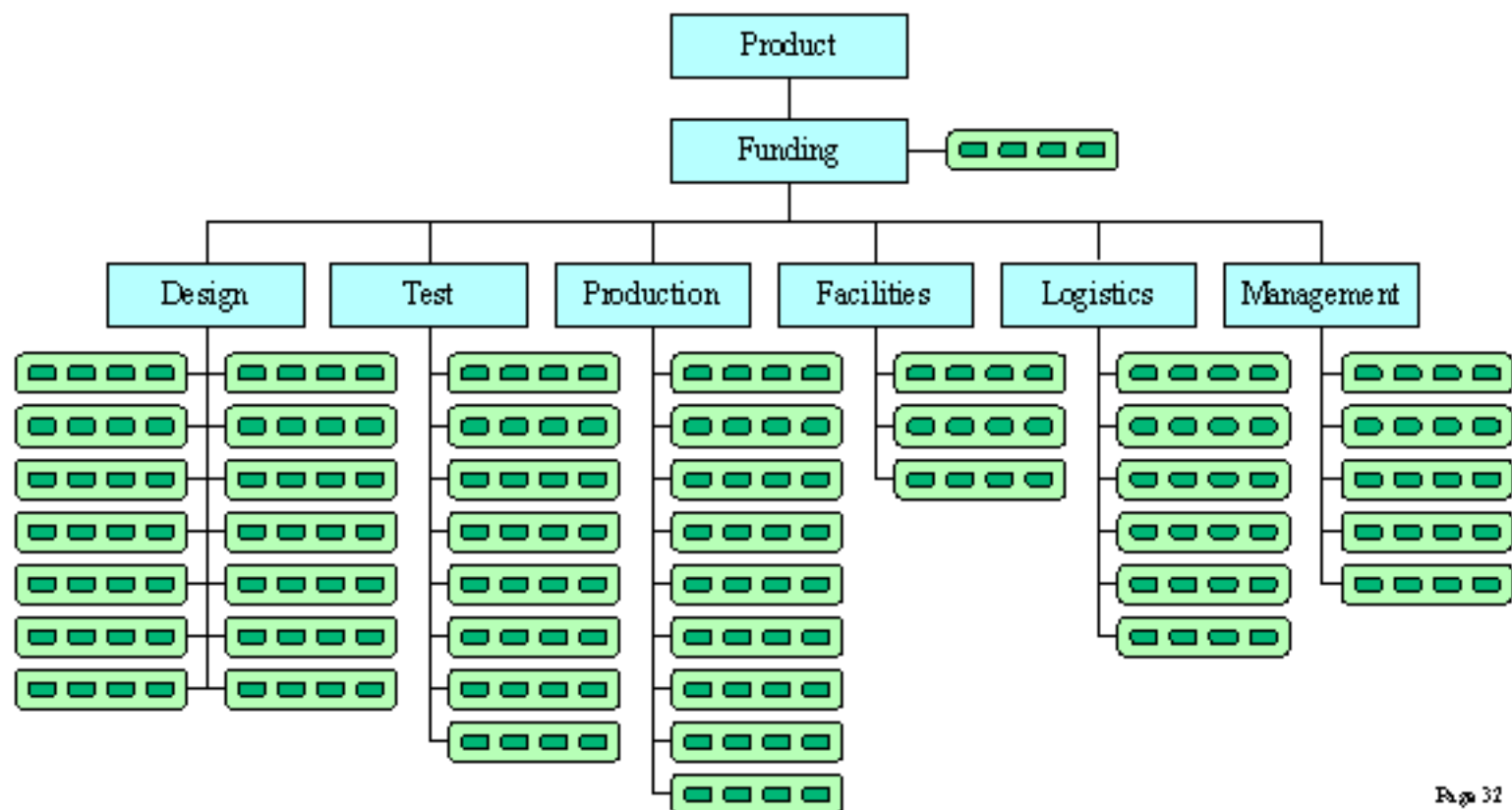
*Best Practices, How to Avoid Surprises in the  
World's Most Complicated Technical Process,  
The Transition from Development to  
Production, March 1986, NAVSO P-6071*

# “Willoughby Templates” are more properly called Anti-patterns

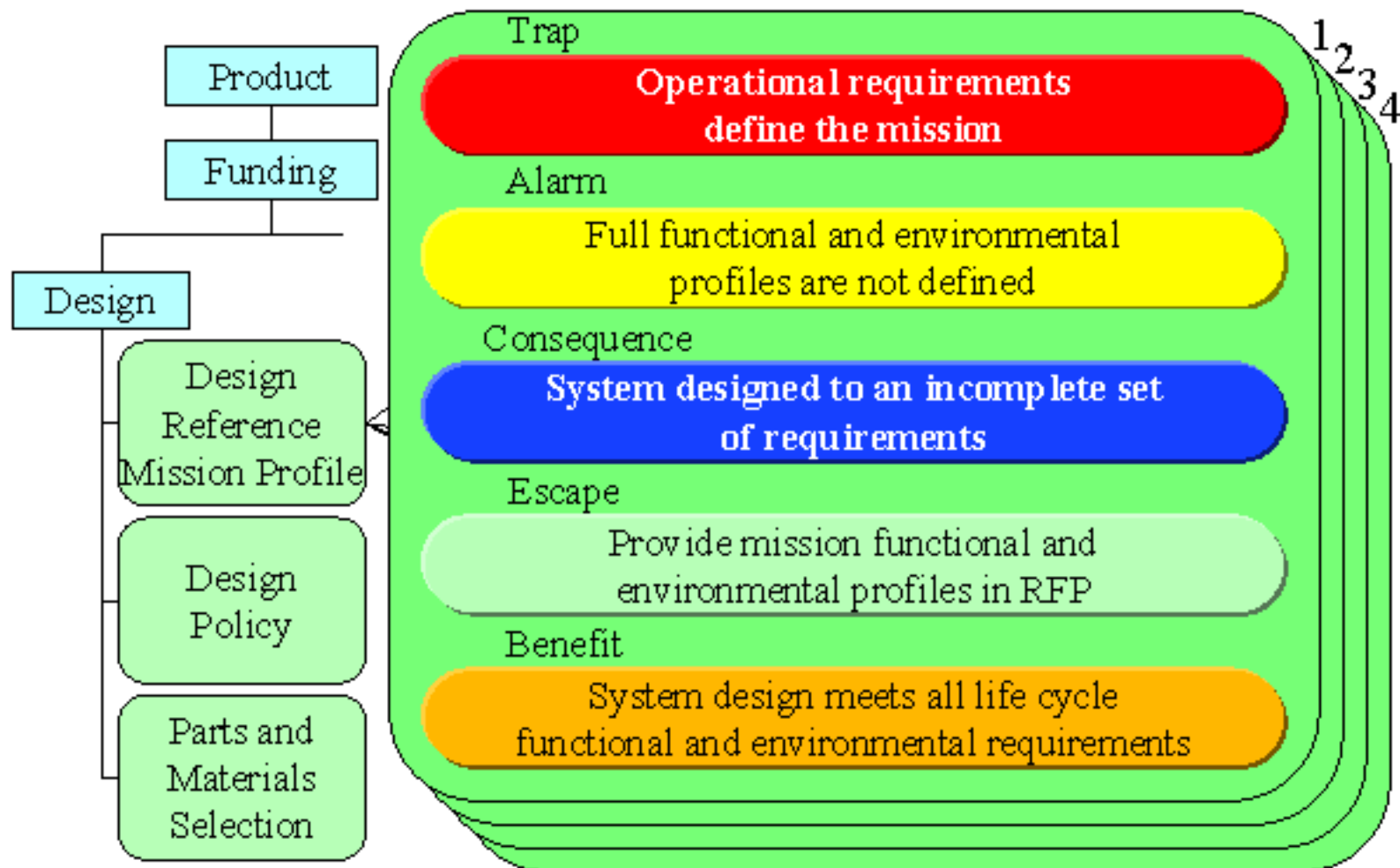
- The pattern is simple
  - Trap - that you may fall into
  - Alarm - that tells you when you are in the trap
  - Consequence - of doing nothing
  - Escape - to get out of the trap
  - Benefit - of getting out of the trap



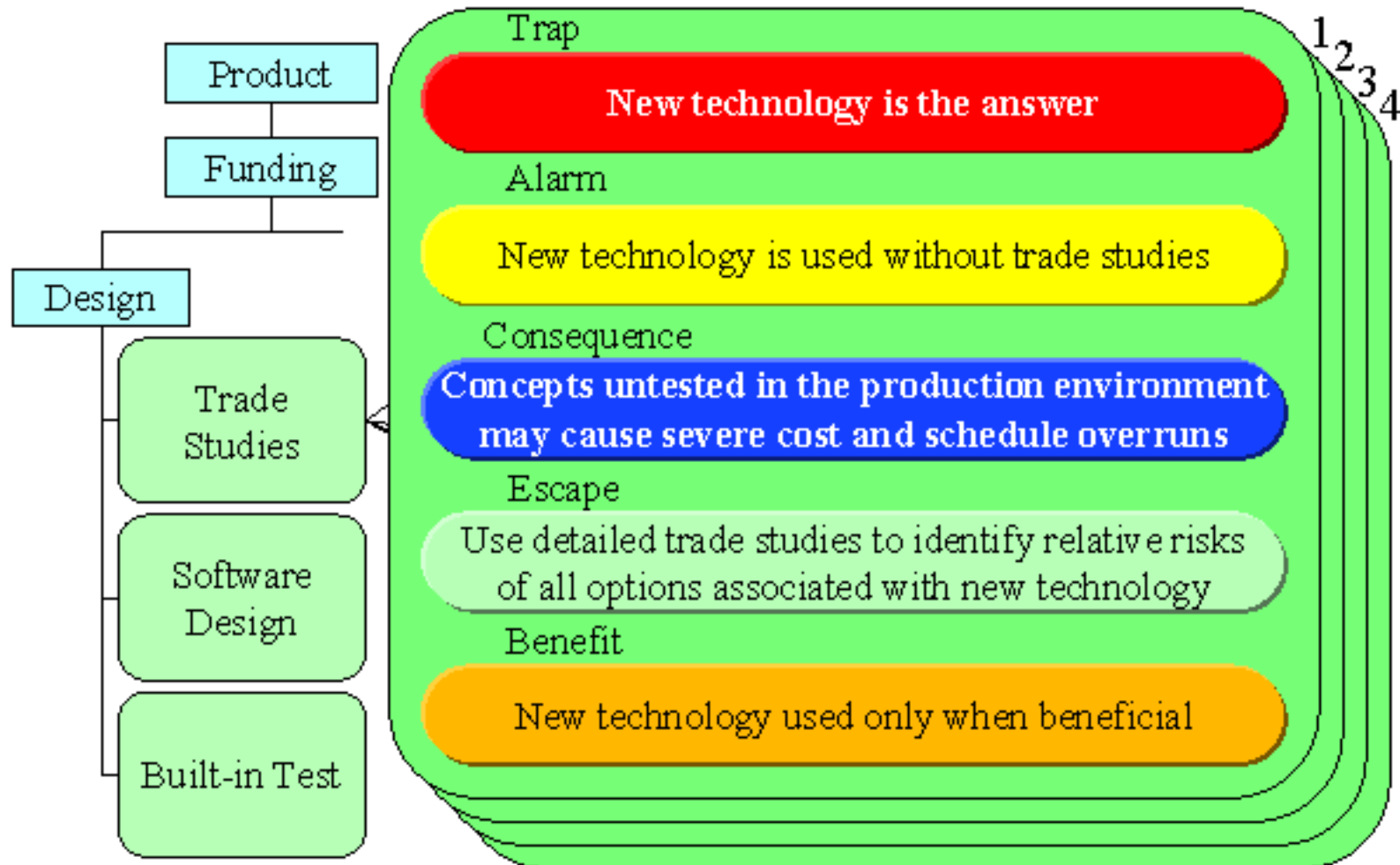
# The Pattern Map in NAVSO P-6071 is highly structured and contains 172 Patterns



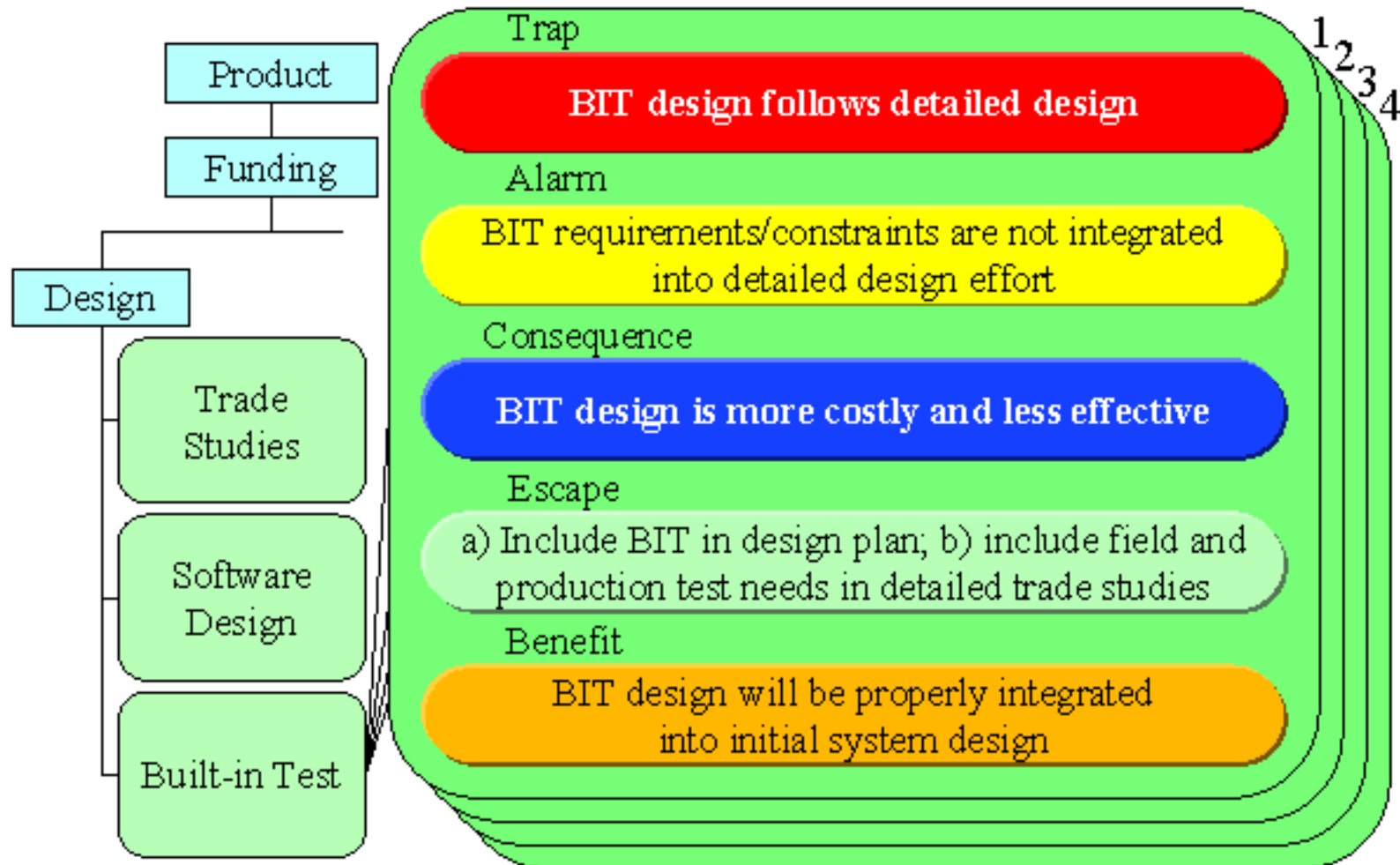
# Design Reference Mission Profile



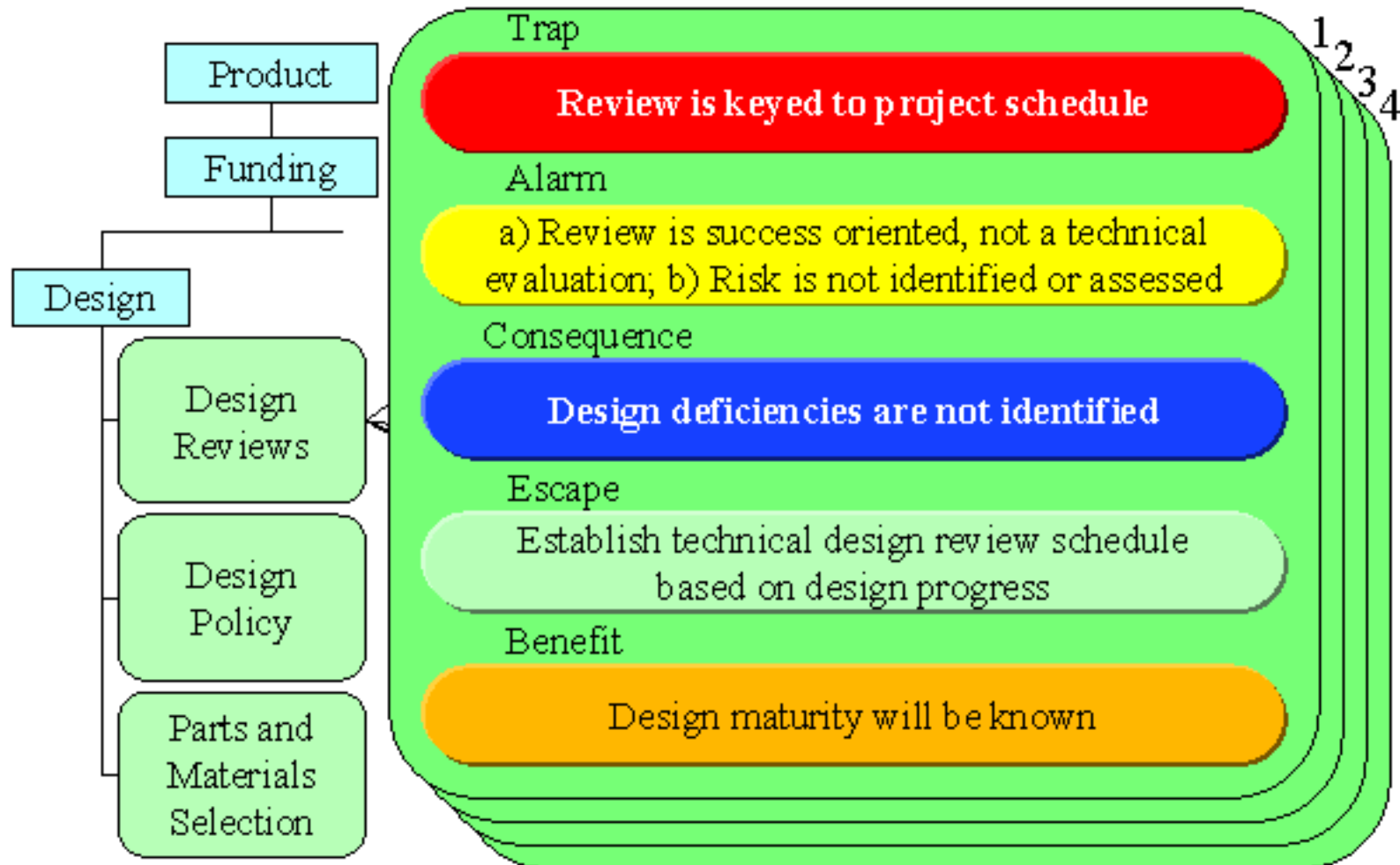
# Trade Studies



# Built-in Test

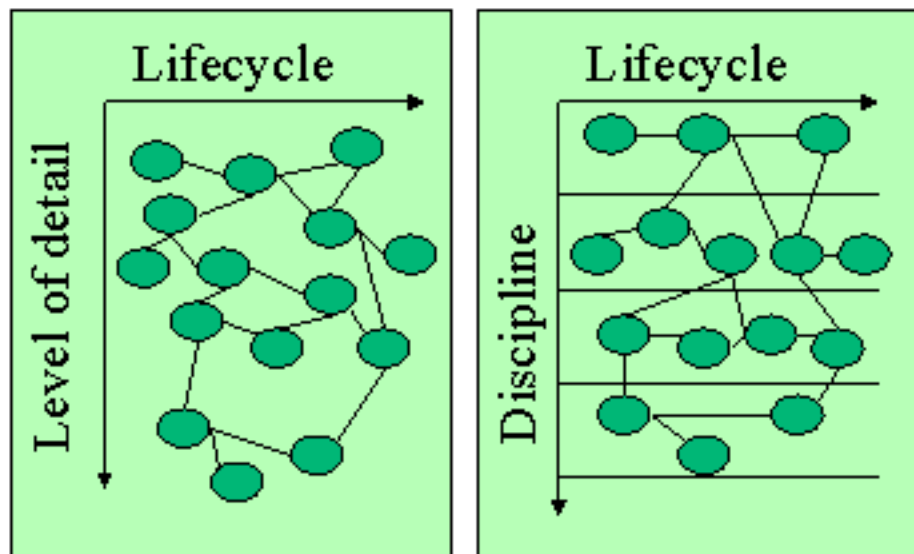


# Design Reviews



# A pattern language could be used to capture a Systems Engineering Body of Knowledge

Several pattern mappings are possible



Web technology coupled with a data repository would facilitate the capture and maintenance of a Systems Engineering Body of knowledge

# Proposal

- Form a Special Interest Group
  - Study and discuss the applicability of pattern languages to systems engineering
  - Define a work product
- If appropriate, evolve into a Special Working Group to deliver the work product

# Agenda

- History
- Definitions
- Examples
  - Form
  - Content
- Conclusion



# Pattern languages organize information for better comprehension

- Pattern languages are being used in many fields
- Some systems engineering information is already organized in patterns
- Web based pattern maps are easy to implement
- A Systems Engineering Pattern Language could capture the Systems Engineering Body of Knowledge