

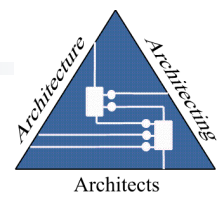


# Overview of the *Visual Architecting Process™*

Dana Bredemeyer and Ruth Malan  
**Bredemeyer Consulting**  
Email: [dana@bredemeyer.com](mailto:dana@bredemeyer.com)  
Inquiries: [training@bredemeyer.com](mailto:training@bredemeyer.com)  
Tel: 1-812-335-1653  
Fax: 1-812-335-1652  
Web: <http://www.bredemeyer.com>

## **Restrictions on Use**

This presentation may be downloaded and printed for personal use. If you quote or paraphrase fragments of this presentation in another presentation, publication or web site, please properly acknowledge this presentation as the source, with appropriate reference to our web site where it is published. If you wish to republish this presentation, in any medium, you must get written permission from Bredemeyer Consulting. Also, any commercial use must be authorized in writing by Bredemeyer Consulting.



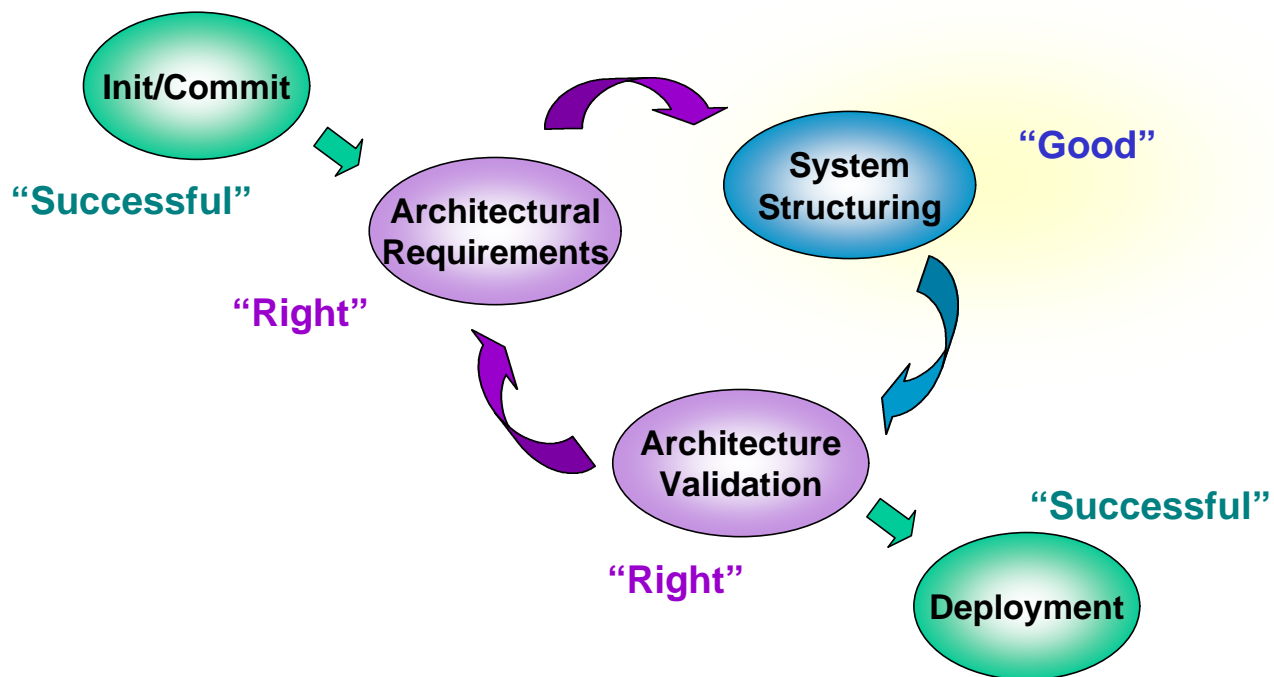
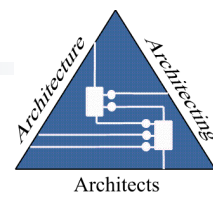
# Objectives

- Understand what architects need to do to create an architecture that is
  - *Good*: technically sound
  - ***Right***: meets the stakeholder needs (business, customers, developers, managers, etc.)
  - ***SUCCESSFUL***: actually used in building systems
- Understand what others need to do to ensure that the architecture is good, right and *successful*

## Background to this Presentation as it Appears on our Web Site

This presentation is a module from one of our seminars. As such, it is designed to be accompanied by stories and elucidation by the presenter.

# How: Overview of the *Visual Architecting Process™*



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 3

## Background

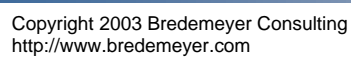
The software architecting process is

- based on our study of dozens of architecting projects
  - identified pitfalls and critical success factors
  - identified processes/steps that worked well
- used in workshops (since 1996)
- used (adapted) in numerous architecting projects
- continuously evolving

## Process Flow

We must emphasize that this is a fluid and highly iterative process--it is **NOT** a didactic and sequential process.

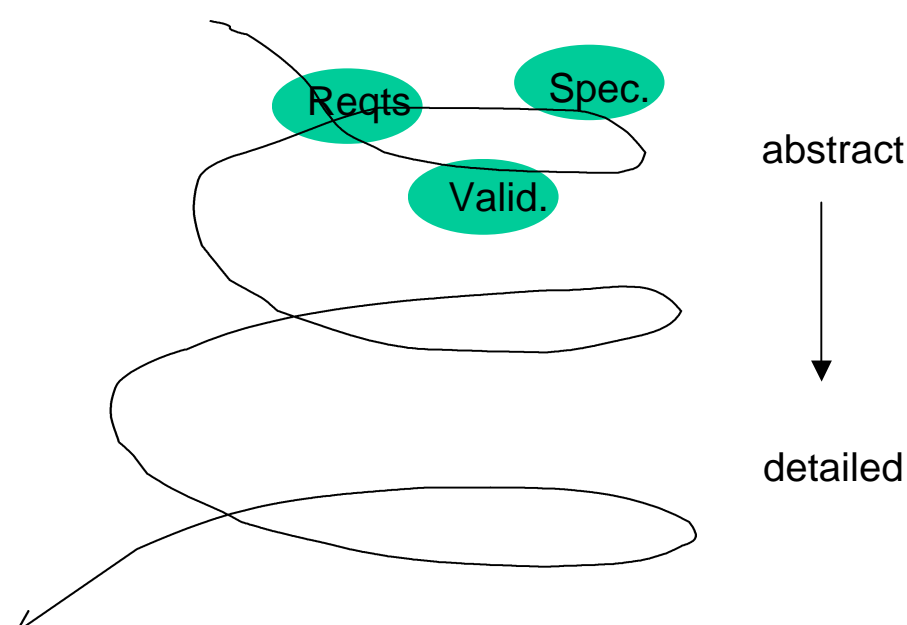
- If you are a "Mozart," you will very likely make a lot of good decisions quickly and not necessarily even consciously. The process will help you make your decisions and internal processes more visible to the people you work with.
- If you are new to the domain, it is good to do more of the requirements steps first (i.e., follow more of a sequential process, at least at first).
- If you are the average good architect, use the process to help guide you. Remember that it is an iterative process. That is, it is intended that you take quick passes through all the phases, and then, using what you've learned, go back and improve your architecture and develop it in more detail. As you do so, use your good sense, but also beware of your "good sense"! To help you avoid being blinded by your own prejudices and natural ties to your own "good ideas," validation is an explicit step. Use an expanded team who will come with an objective opinion and fresh ideas.



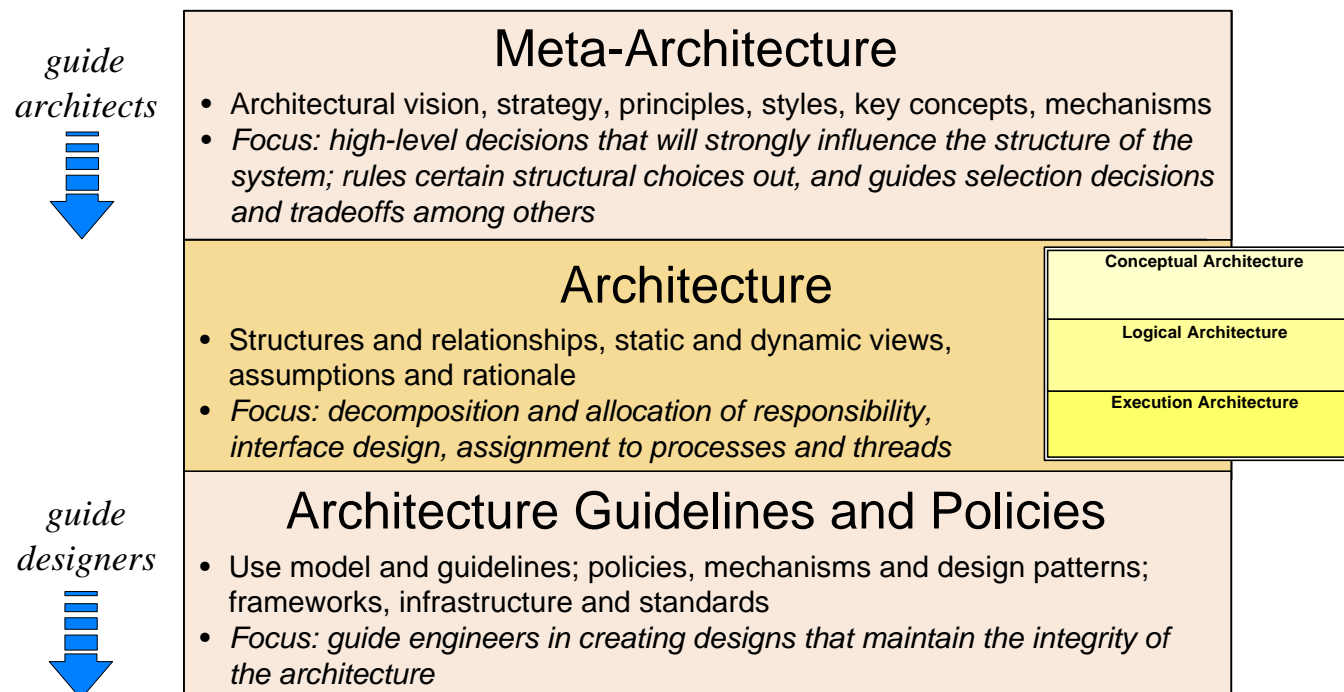
Visual Architecting Process Overview  
Slide 4

The process flows from left to right (context to requirements/“problem statement” to architectural solution) and top to bottom (abstract to detailed). This map shows where our different Action guides fit into the overall process. See <http://www.bredemeyer.com/papers.htm> for examples of our Action Guides.

Use iterations to add detail, to focus on different aspects (e.g., focus on risky areas first), widen scope, etc.



A diagram of a blue triangle. The left side is labeled "Architecture", the right side is labeled "Architecting", and the bottom is labeled "Architects". Inside the triangle, there are two rectangular boxes connected by lines. The left box is connected to the "Architecture" side, and the right box is connected to the "Architecting" side. Lines also connect the two boxes to each other and to the "Architects" base.



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 5

The meta-architecture is a set of high-level decisions that will strongly influence the integrity and structure of the system, but is not itself the structure of the system. The meta-architecture, through style, patterns of composition or interaction, principles, and philosophy, rules certain structural choices out, and guides selection decisions and trade-offs among others. By choosing communication or co-ordination mechanisms that are repeatedly applied across the architecture, a consistent approach is ensured and this simplifies the architecture. It is also very useful at this stage, to find a metaphor or organizing concept that works for your system. It will help you think about the qualities that the system should have, it may even help you think about what components you need (in Conceptual Architecture), and it will certainly help you make the architecture more vivid and understandable.

To help maintain system integrity or to address cross-cutting concerns, architects may include decisions focused at guiding or constraining lower-level design or even implementation in the architecture decision set. Recall our caution: the *only* justifiable reason for restricting the intellectual freedom of designers and implementers is demonstrable contribution to strategic and systemic properties that otherwise could not be achieved. That said, there is a fair amount that architects can valuably do to help designers and implementers in applying the architecture and in paying attention to the right characteristics of the problem so that their decisions, in turn, are in alignment with all that is explicit in the architecture and all that is implicit in the architecture concept.

Architecture

Architecting

Architects

# Meta-Architecture

## Form Guiding Principles and Strategies

Meta-Architecture

Architecture

Architectural Guidelines and Policies

guide architects

Principle Name	Give the principle a catchy name.
Description	Statement of the principle.
Rationale/Benefits	Describe the reasoning behind the principle. Where applicable, provide traceability to business or architectural objectives.
Implications	Identify implications such as actions that need to be undertaken, and constraints implied by the principle.
Counterargument	Describe the reasonable counter to this principle.

Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 6

Architectural Principles

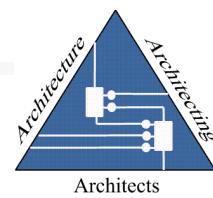
Architectural principles are “statements of preferred architectural direction or practice.” They guide the architecture team in their structuring decisions.

Architecture Strategy

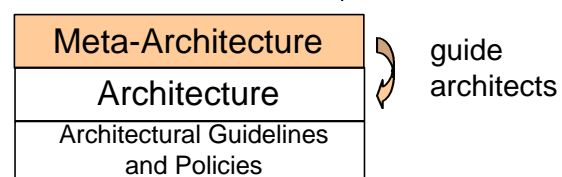
The architecture strategy sets the direction for the architecture, and is the bridge between business strategy (direction for the business) and system implementation. In creating architectural strategy, the architects must understand the context that informed the business strategy process, as well as the business strategy itself. They need to identify strategic business objectives that the architecture is intended to help meet, and formulate associated architecture objectives. It is helpful to show this visually with a strategy map that links the architecture objectives to the business objectives on the business strategy map. Measures need to be associated with the architecture objectives (a “scorecard”), and principles are defined to guide the achievement of the architecture objectives. See Malan and Bredemeyer, “Architectural Strategy”, 2003 published at <http://www.bredemeyer.com/ArchitectingProcess/ArchitectureStrategy.htm>

# Meta-Architecture

## Determine Architectural Style



- Choose styles/architectural patterns
- Design key mechanisms
- Consider an organizing metaphor
  - manager/controller
  - broker

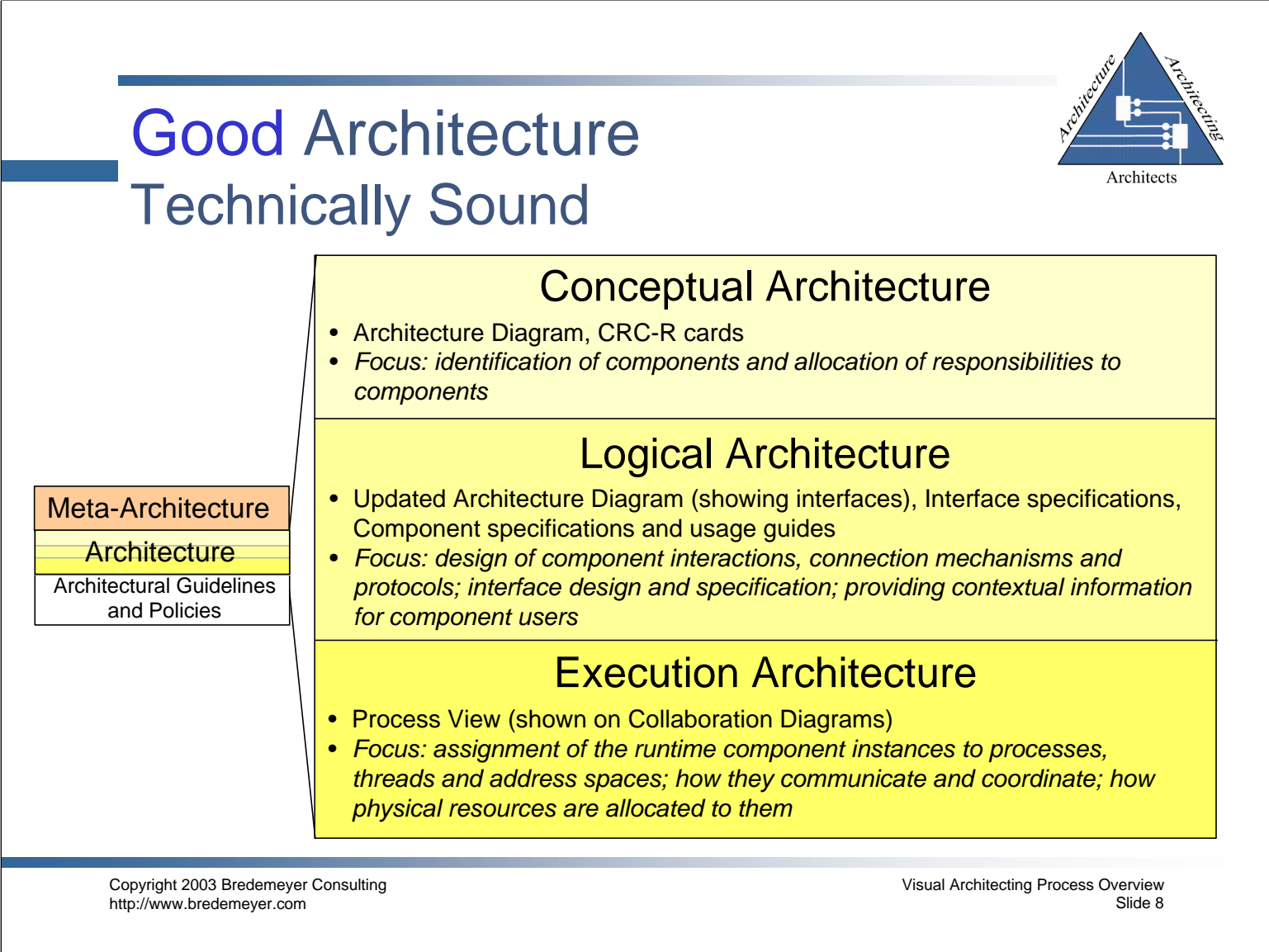


### Architectural Style

In building architecture, architectural style constrains the choices available to the architect (and subsequently the builder). For example, a “contemporary” home has “walls of windows” while sheet glass would not fit in a traditional styled home like a “colonial” or “victorian”.

Likewise, architectural styles in software, provide guidelines and constraints. In a strictly layered architecture, for example, a component in one layer cannot directly interface to a component two layers away.





**Good Architecture Documentation**

Relates to the understandability, approachablity, and usefulness of the documentation format and content to the various stakeholders. Expressing the architecture in terms of *views*, and tailoring documents/presentations to different stakeholders, enhances the usefulness of the documentation. Good documentation also includes *rationale* for the architectural decisions.

**Bad Architecture**

A bad “architecture” can have the following characteristics: monolithic, so even small changes are distributed through code and cause unpredictable results with long compile/link/debug cycle times; hidden implicit coupling in code; multiple inconsistent ways of doing the same thing; not documented and so has a high learning curve.



Architecture

Architecting

Architects

# Conceptual Architecture

## Identify Components (initial cut)

Clerk Dialog

Balance Books

Receive Payment

Enter

Accounts

Component Name	Give the component an easy-to-remember name
Responsibilities	List the responsibilities assigned to the component
Collaborators	List of other components this component depends on for (delegated) services [out-ports]
Rationale	State the rationale for allocating responsibilities to this component. Provide traceability to functional requirements and qualities or meta-architecture.
Issues and Notes	List assumptions, constraints, unknowns, etc.

Conceptual Architecture

Logical Architecture

Execution Architecture

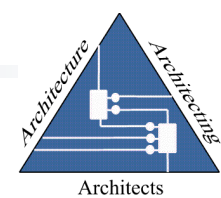
Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 9

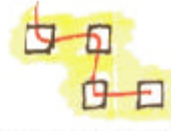
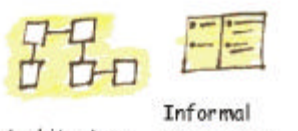
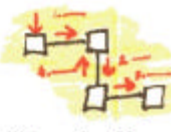
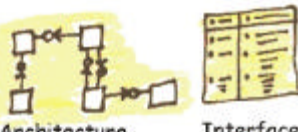
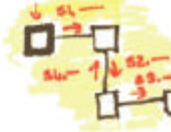
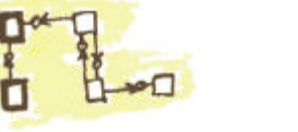
Components and Relationships

The conceptual architecture identifies the high-level components of the system, and the relationships among them. A component is a modular unit of software functionality that is accessed through one or more interfaces.

During the *conceptual architecture* phase, components are identified that obey the fundamental principles of architecting, namely information hiding and seeking high cohesion and low coupling. Responsibilities are assigned to the components with the primary concern being to ensure that each component has a cohesive set of responsibilities. This will impact the understandability and maintainability of the architecture.



# Architecture Views

	Behavioral View	Structural View
Conceptual Architecture (abstract)	 Collaboration trace	 Architecture Diagram Informal Component Specs (CRC-R)
Logical Architecture (detailed)	 Collaboration Diagrams	 Architecture Diagram with I/Fs Interface Specs
Execution Architecture (Process View and Deployment View)	 Collaboration Diagrams showing processes	 Architecture Diagram showing Active Components

Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 10

## Architecture: Structure or structure+behavior+rationale?

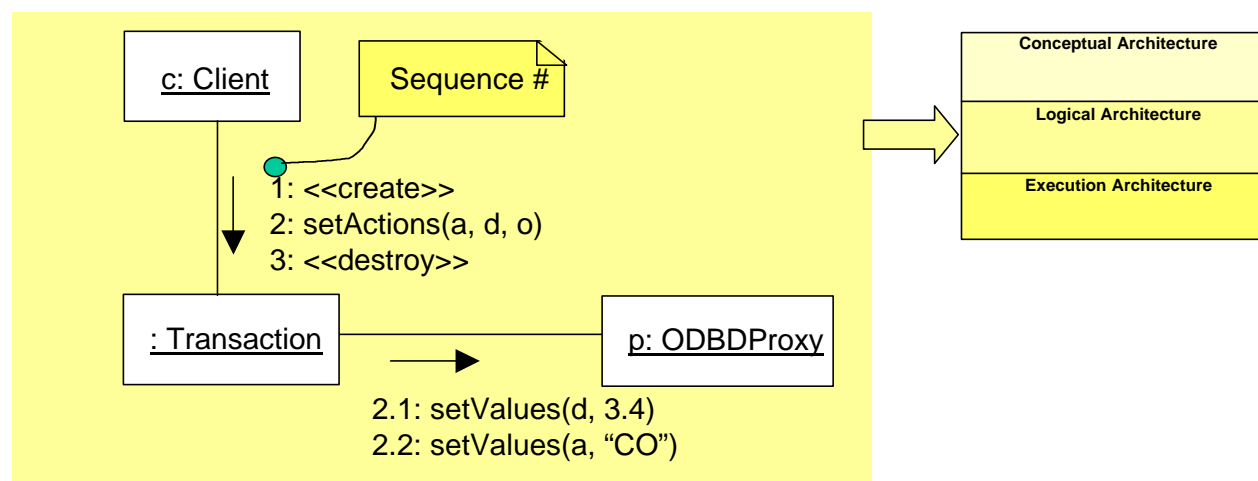
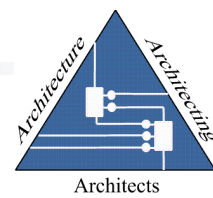
Architecture is generally defined in terms of structure. However, the behavior (or use) of the system must be considered if one is to design a good structure. Also, the structuring decisions have to explicitly address the prioritized system qualities. If not, the emergent properties of the system will fail to address stakeholder requirements. The rationale for these decisions must be captured, so that others can understand why specific choices were made (and so that you will remember why!).

## Structure and Behavior

The London Footbridge had to be retrofitted (and closed for a year to do so) to allow for its performance under high winds. The **behavior** of the system was not taken into account when the structure was designed.

# Logical Architecture

## Model System Behavior



### Key principle: Form follows Function

- Assign responsibilities to components to accomplish required services taking into account system qualities
- Key tool: **Collaboration Diagrams**

*"Form follows function"*: design structure to support required functionality (system behavior) with desired system qualities

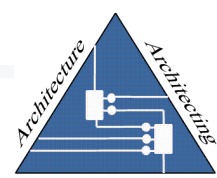
Behavioral models show how the components interact and change state.

Using architectural models, we can investigate how changes in system properties affect structure and vice-versa (Coleman and Beringer, UML World 2000). For example, in building architecture, we can investigate how changes to the floor plan affect properties such as:

- how area is distributed across the rooms
- how much sun shines into the room at different times of the year
- what view is taken in
- what access is like to the different rooms

# Logical Architecture

## Document Interfaces



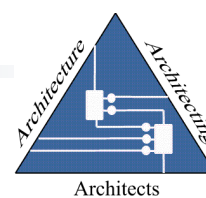
Interface signature	I/F Element	Description
	Interface name	A unique identifier for the interface
	Exceptions	The name and data content for each operation's exceptions
	Properties	The name and type of each property
	Operations	The name of each operation, together with the input and output parameters and exceptions
	Operation descriptions	Description of each operation using <ul style="list-style-type: none"><li>informal description or</li><li>pre/post condition template</li><li>example showing typical calling usage (<i>optional</i>)</li></ul>
	Protocol ( <i>optional</i> )	Constraints on the order in which operations may be called (Statechart)
	Service Level ( <i>optional</i> )	Non-functional requirements to be met by the services provided by the interface (operations)
	Notes and Issues	List of components using I/F List of issues to be resolved

### Interface Syntax AND Semantics

To make the architecture precise and actionable, the interfaces need to be documented clearly and unambiguously. Document the interface signature. Syntactic specifications are *essential* to be able to use the interface. However, they are not enough to understand the services the component offers, or to be able to use the interface properly. Without any further information, one would have to dive into the component implementation (e.g., using component introspection or having to resort to the source code) and try to figure out what each operation on the interface does and how the operations are supposed to work together. This defeats the purpose of the interface (that being encapsulation and separation of concerns). Semantic specifications address this weakness. Semantics may be added in the form of operation descriptions, pre/postconditions, protocols, and non-functional requirements/service level, etc.

# Logical Architecture

## Document Components: Specification

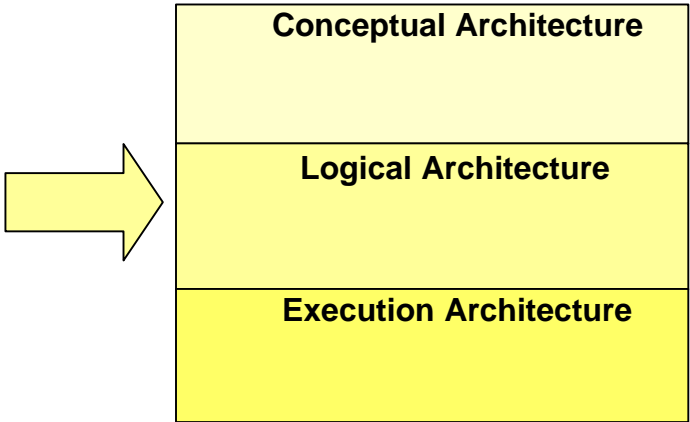


Component Element	Description
Component	A unique identifier for the component
Responsibilities and Provided I/Fs	A summary description of the component's purpose, responsibilities/principal features. List of provided interfaces with brief descriptions
Collaborators and Required I/Fs	Other components that this component interacts with, listed with the interfaces it uses to do so, and any assumptions it makes with respect to the collaborators.
Requirements/restrictions	Requirements/restrictions that apply to this component (e.g., hardware requirements for host system, OS)
Rationale	Document the underlying reasons why the component is designed the way it is
Notes	Constraints: document system-level or architectural constraints that the component must satisfy. Assumptions: document any assumptions made about the component
Issues	List of issues that must be resolved

### Not just Interfaces

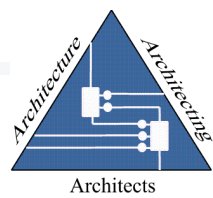
Component documentation includes specification of its interfaces **and** specification of other component-wide properties

- requirements/restrictions on the component
  - dependencies on other components
- and guidance on using the component.

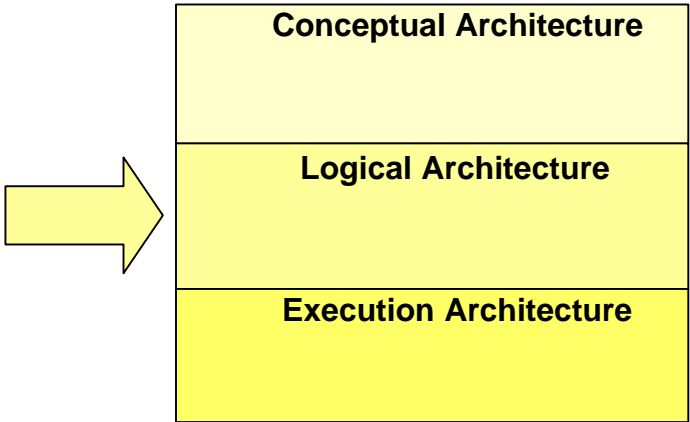


# Logical Architecture

## Document Components: Usage

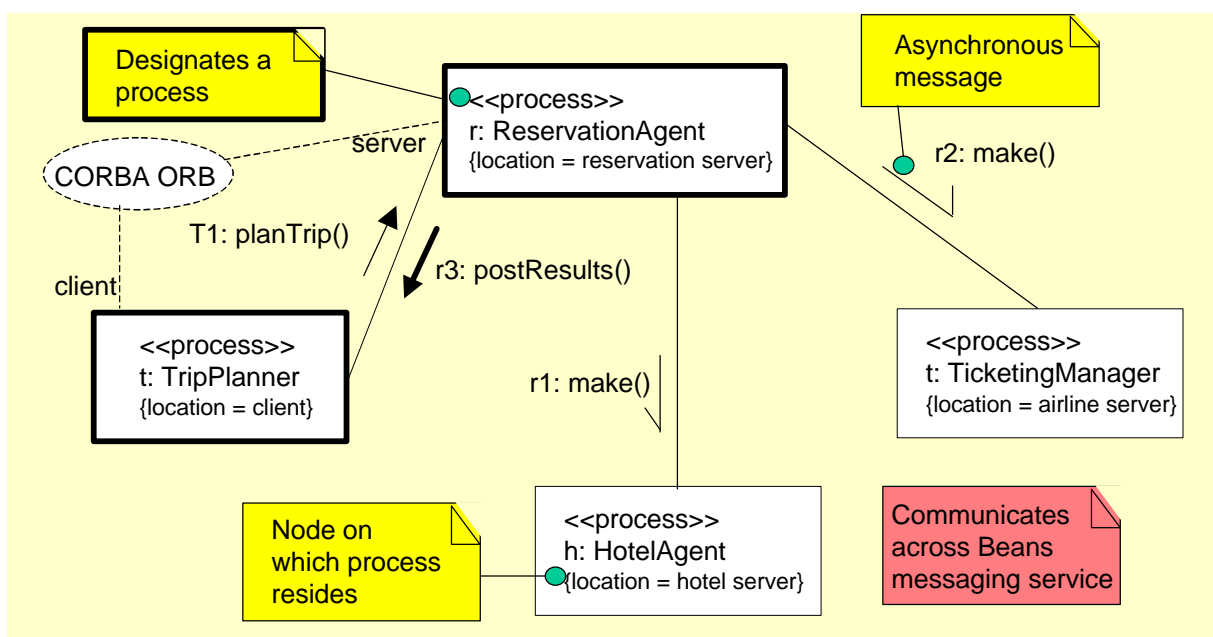
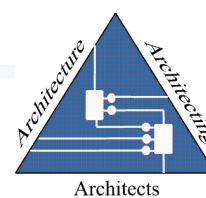


Component Element	Description
Component Personnel	Identify who to go to with questions, requests for changes, etc. (e.g., list component owner, tester, project manager, architect)
Change history	Summarize any changes to the specification/implementation and dates
Initialization	How is the component initialized? Include pre-condition assumptions.
Finalization and termination	How is the component terminated under normal and abnormal circumstances?
Interactions	How does this component interact with other components to accomplish system services? (Reference Collaboration Diagrams this component plays a major role in)
Usage guidelines	How to use the component in building an application, including a programming example
Demo	Sample mini-application showing how the component is used
Test	Testing considerations, test suite, etc.



# Execution Architecture

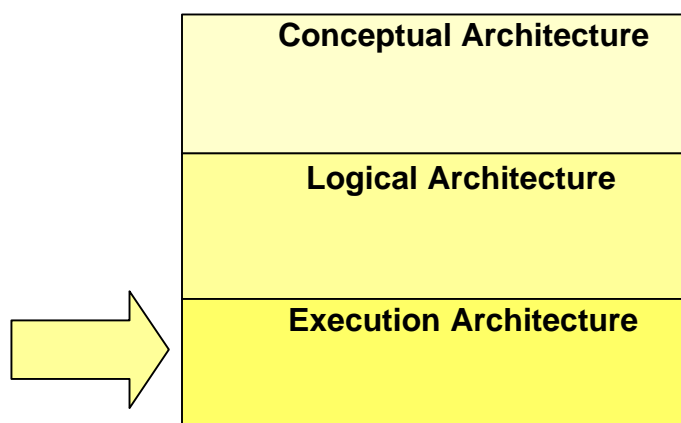
## Allocate Components to Processes



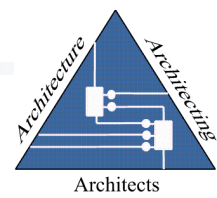
Key tool: **Collaboration Diagrams**

Diagram from Booch et al, 1999

Visual Architecting Process Overview  
Slide 15





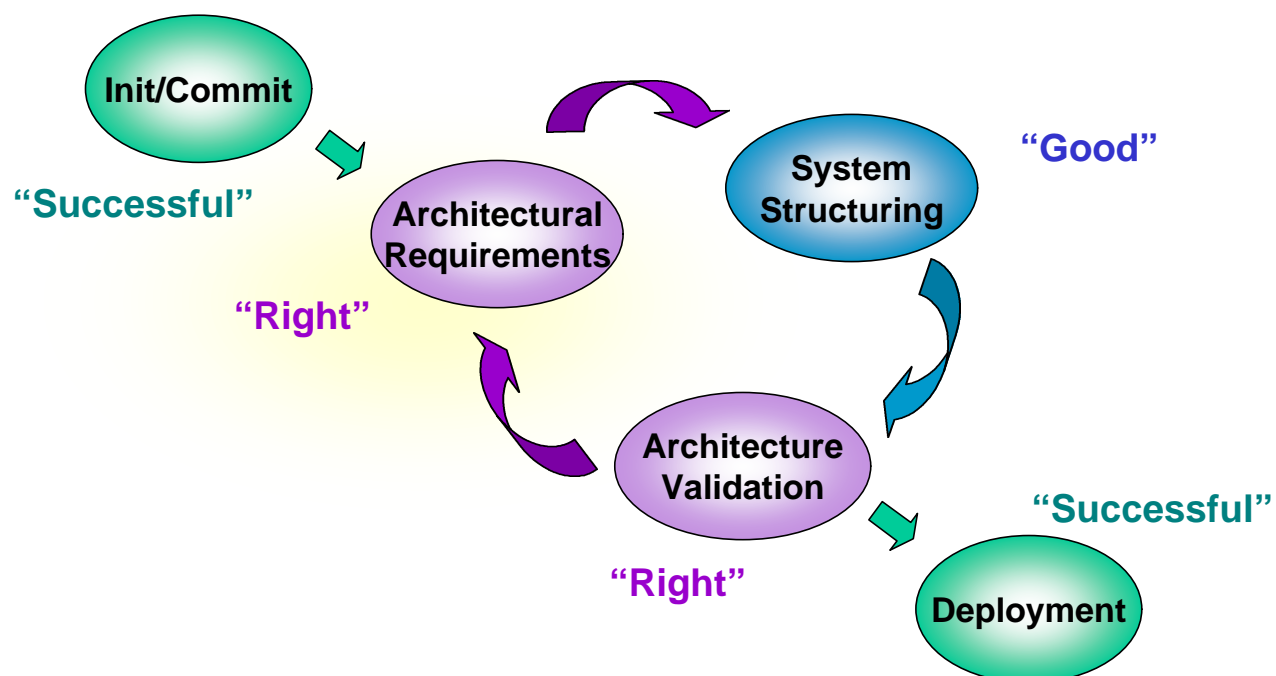
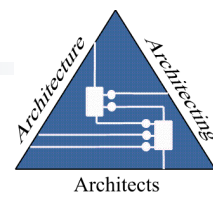


## Good Architecture

- Separation of Concerns
  - Different views for different audiences
    - see how your concerns are addressed
- Models which show
  - how the system is decomposed
    - how the structure supports the required behavior
  - how the key cross-cutting concerns are addressed
- Guide designers and implementers
  - maintain system integrity

# Visual Architecting Process

## Good, **Right** and Successful!



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

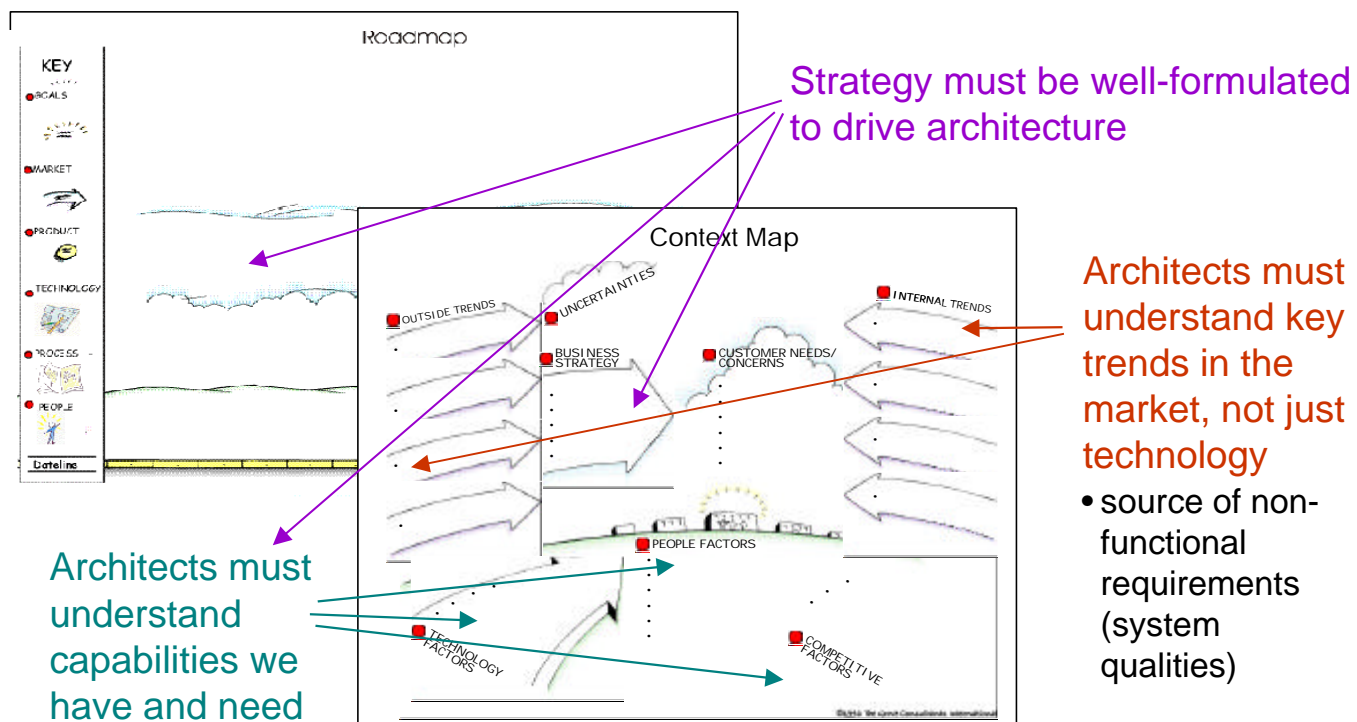
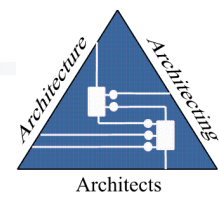
Visual Architecting Process Overview  
Slide 17

### Architectural Requirements

A “good” architecture solves the problem that it solves well, but it does not necessarily solve the right problem. In order to do so, the architects cannot just rely on past experience and intuition--though these are important too. A “right” architecture is one that will enable the development organization to implement the current business strategy, and evolve to keep pace with changes in business strategy.

To create a right architecture, the architecting team translates business objectives into architectural objectives. It understands the current organizational context, and looks at trends to make assertions about the future. It considers scenarios (good and bad, likely and unlikely). It takes stakeholder goals into account in establishing functional and non-functional requirements, and establishes the relative priority of these requirements. A solid understanding of current behavioral requirements, and a well-grounded projection of future behavioral requirements, drives the structuring choices that are at the heart of architecting. Tradeoffs are driven by the prioritized set of system qualities (or non-functional requirements), and mechanisms are designed to accomplish key cross-cutting concerns (system properties that cannot be localized to particular components).

# Architectural Requirements Link to Strategy and Context



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 18

## Strategy and Architecture

To enable the business strategy, architects need the business strategy as input to the architecting process. Often, however, the strategy is not formulated in a way that is useful to architects.

Also, the business strategy should be informed by what the organization's technical leaders, the architects, know about the technical capabilities and opportunities open to the organization. They should provide input to the strategy process and the definition of business objectives.

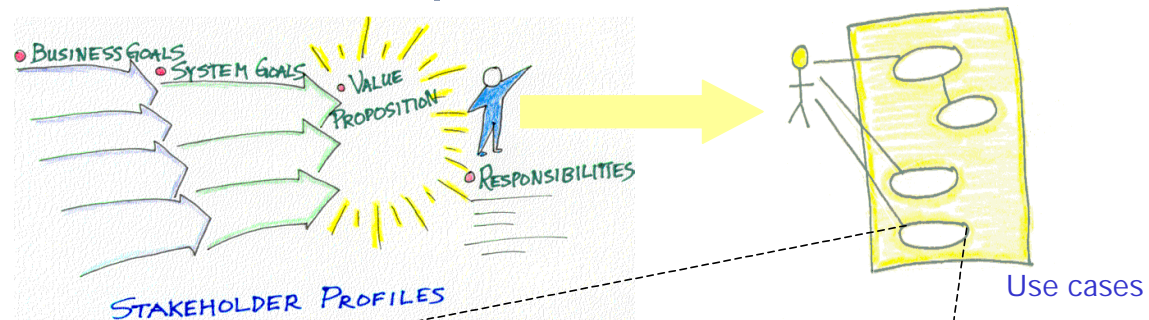
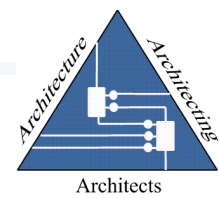
## Grove Graphic Guides®

Grove Consultants International have a number of Graphic Guides® that are useful tools for collecting strategic contextual input. See <http://www.grove.com>. We have adapted Grove's Context Map, Roadmaps, and other guides for use in the architecture context, and we also add our own *Action Guides™*.

*Graphic Guides* is a registered mark of Grove Consultants International.

# Right Architecture

## Functional Requirements



Use Case	Validate User
Actors	Customer
Steps	1. The system prompts the customer for a PIN number 2. Customer enters the PIN number 3. The Customer commits the entry 4. The system checks the PIN to see if it is valid 5. If valid, system acknowledges the entry
Variations	The Customer can cancel at any time, thus restarting the use case. No changes are made to the Customer's account.  The Customer can clear the PIN any time before committing it and re-enter the PIN.  If the Customer enters an invalid PIN, the use case restarts. If this happens 3 times in a row, the system cancels the transaction.

Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 19

### Functional Requirements and Use Cases

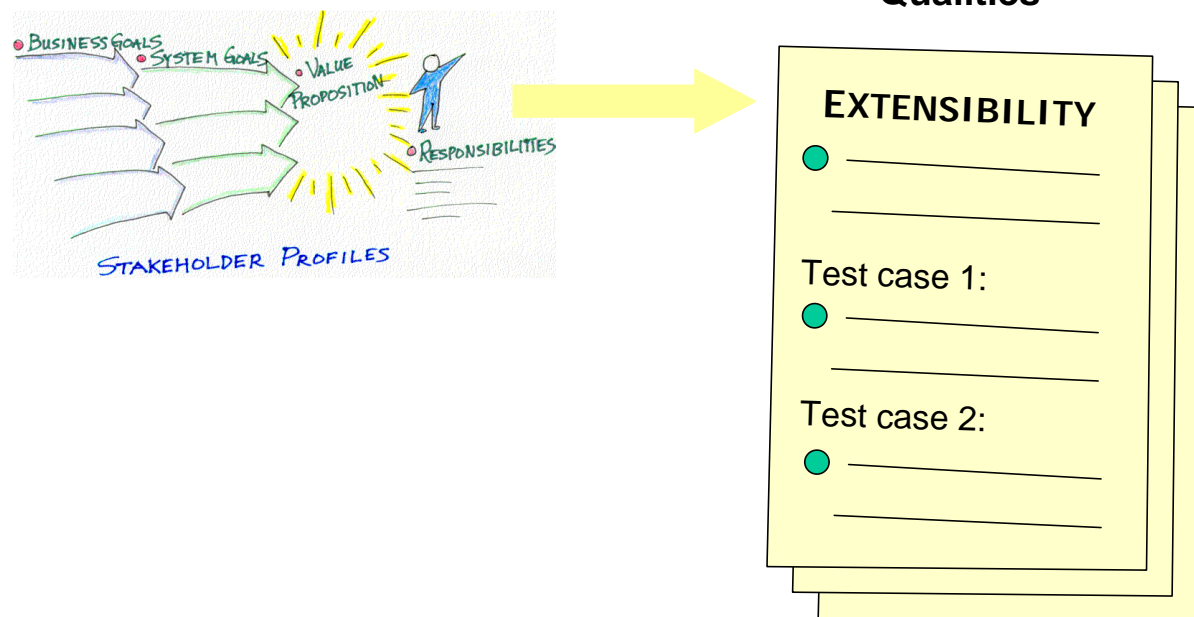
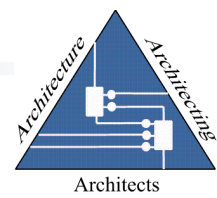
Functional requirements capture the intended behavior of the system. We use Use Cases to capture *who* (actor) does *what* (interaction) with the system, for what *purpose* (goal), without dealing with system internals. A use case defines a goal-oriented set of interactions between external actors and the system under consideration.

### References

- Kulak, Daryl, and Eamonn Guiney, *Use Cases: Requirements in Context*, Addison-Wesley, 2000.
- Malan, Ruth and Dana Bredemeyer, "Functional Requirements and Use Cases", <http://www.bredemeyer.com/functreq.pdf>, June 1999.

# Right Architecture

## Non-Functional Requirements



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 20

### Non-Functional Requirements or System Qualities

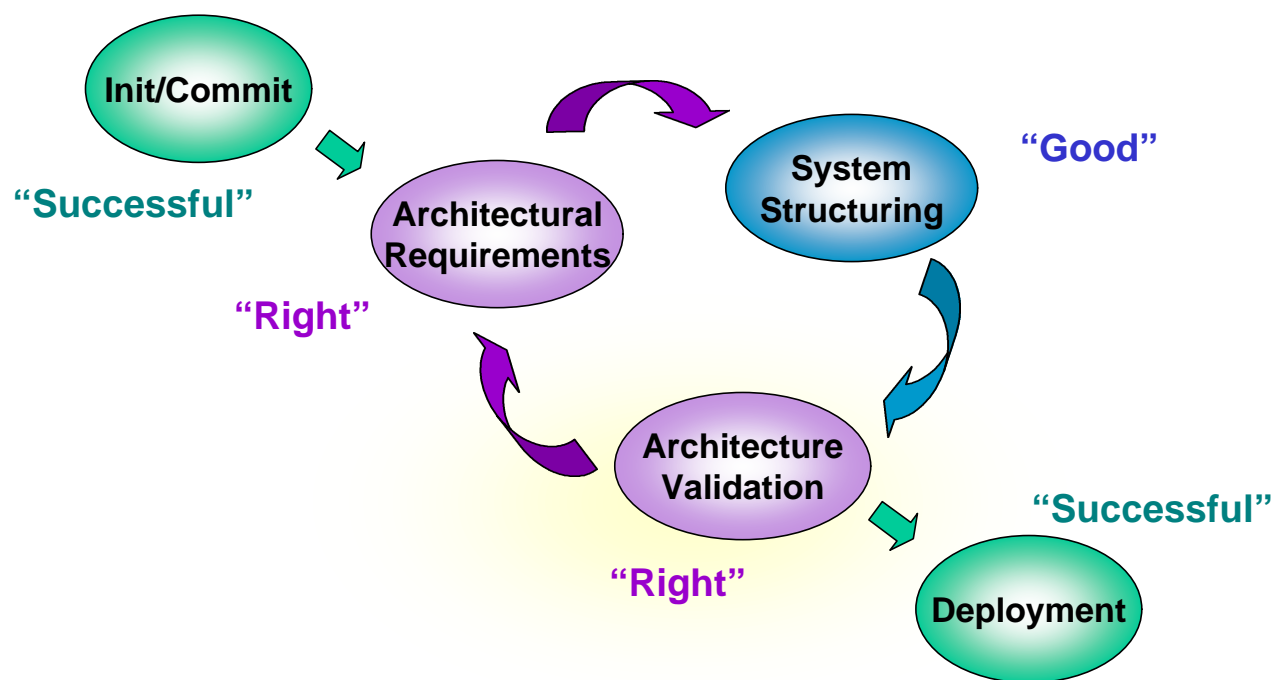
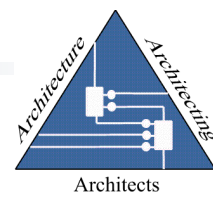
*Qualities* are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system.

#### References

- Malan, Ruth and Dana Bredemeyer, "Defining Non-Functional Requirements", <http://www.bredemeyer.com/functreg.pdf>, August 2001.

# Visual Architecting Process

## Good, **Right** and Successful!



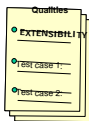
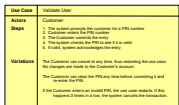
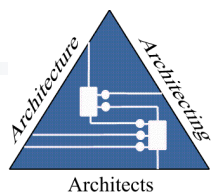
Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 21

### Architecture Validation

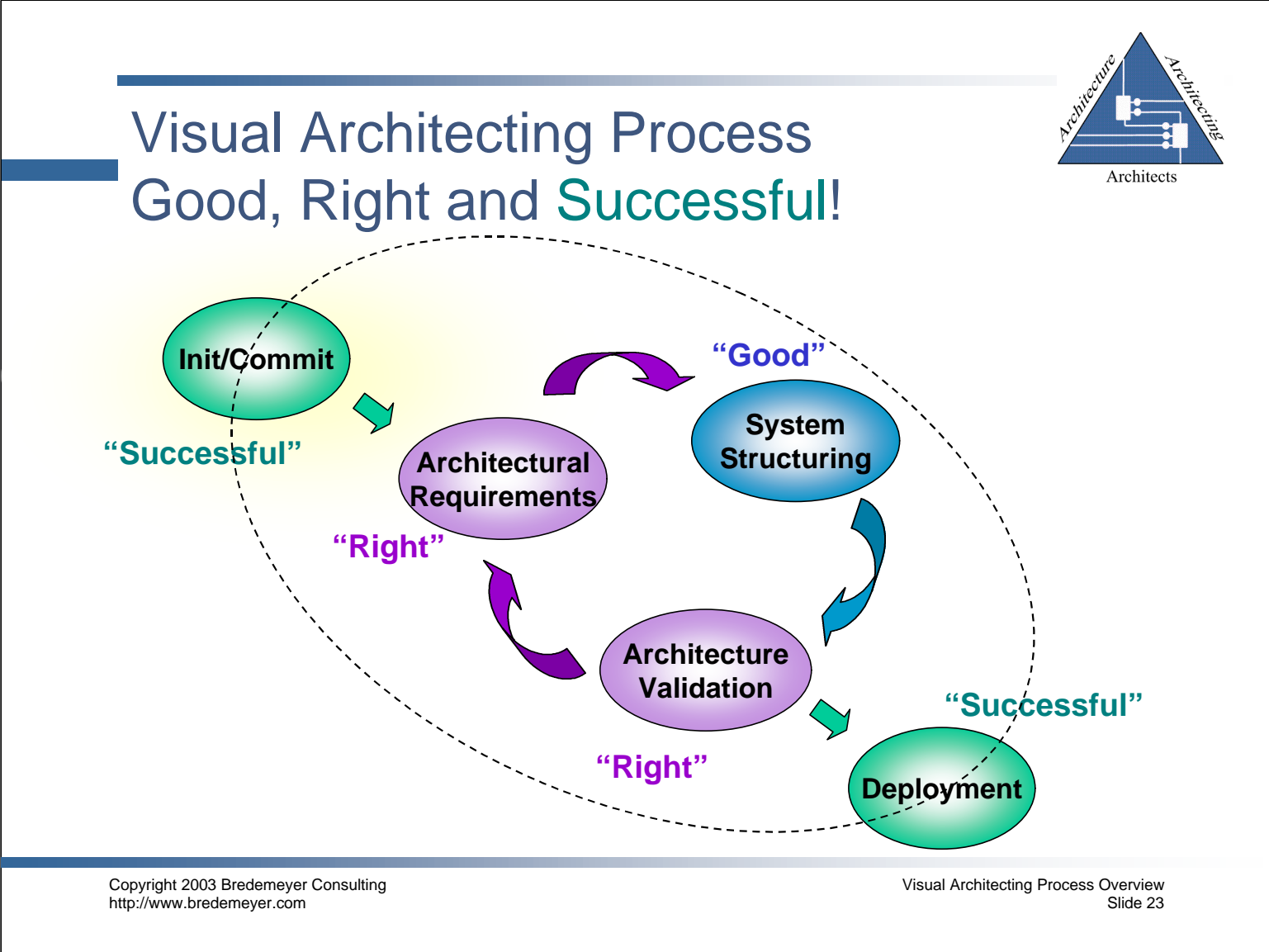
The objective of the architecture validation activity is to find (and fix) problems with the architecture early--and certainly before the architecture is implemented and cast in code. There is a tendency to think of code as very malleable, but for large systems it quickly becomes prohibitively expensive to re-architect once coding is underway.

# Validation Impact Assessment



	Satisfied	Required Changes	Impact to Project	Impact to Architecture	Importance
o Use cases					
o Test cases					
o Value Proposition					
o Scenarios					
o Context Map					
o Roadmap					
o Vision					

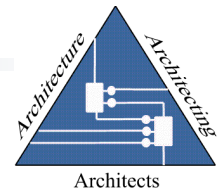




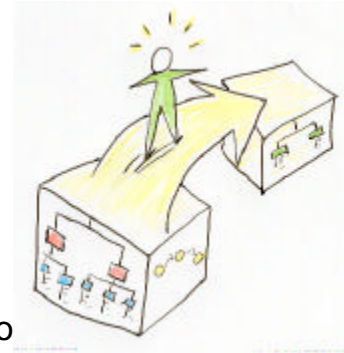
Responsibility for the Success of the Architecture

The team of architects are responsible for creating a good and right architecture, and **share** the responsibility with the rest of the organization for a **successful** architecture.

# Success: Management's Part Lead, Champion and Sponsor



- Architecture changes the way we (need to) do things
  - organization and roles; process
- Platform changes the way we (need to) do things
  - organization and roles; process
- Change *demands* leadership
  - must have a vision
  - must build enthusiasm for the vision
    - lead by example
    - walk the talk
      - tell stories that communicate the vision
      - ensure that the vision shapes decisions
  - actively champion the architecture



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 24

## Management's Role: Lead, Champion and Sponsor

Leadership is important when

- the problem is novel
- change is required
- there is lack of acceptance

Such situations require

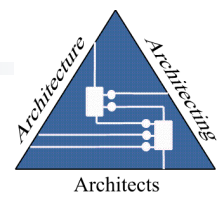
- vision to align the team and community behind the change
- passion to persevere despite the obstacles that will be faced
- a willingness to make decisions under uncertainty

"INDECISION, n. The chief element of success; "for whereas," saith Sir Thomas Brewold, "there is but one way to do nothing and divers ways to do something, whereof, to a surety, only one is the right way, it followeth that he who from indecision standeth still hath not so many chances of going astray as he who pusheth forwards..." Ambrose Bierce, *The Devil's Dictionary*

## Lead Organization and Process Changes

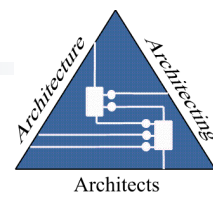
Management needs to make changes to the organization and process to support architecture, including establishing new roles and responsibilities, and changes to old roles, new process, and changes to existing processes, and a roadmap for change.

# Success: Management's Part Lead, Champion and Sponsor



- Architecture demands *commitment*
  - it takes time
  - it takes resources
    - top technical people
    - full time
- Architecture demands *sustained* commitment
  - no flagging resolve
    - it *will* meet with resistance
    - resistance *will* build as schedule pressures build
  - in it for the long term
    - invest in active architecture renewal

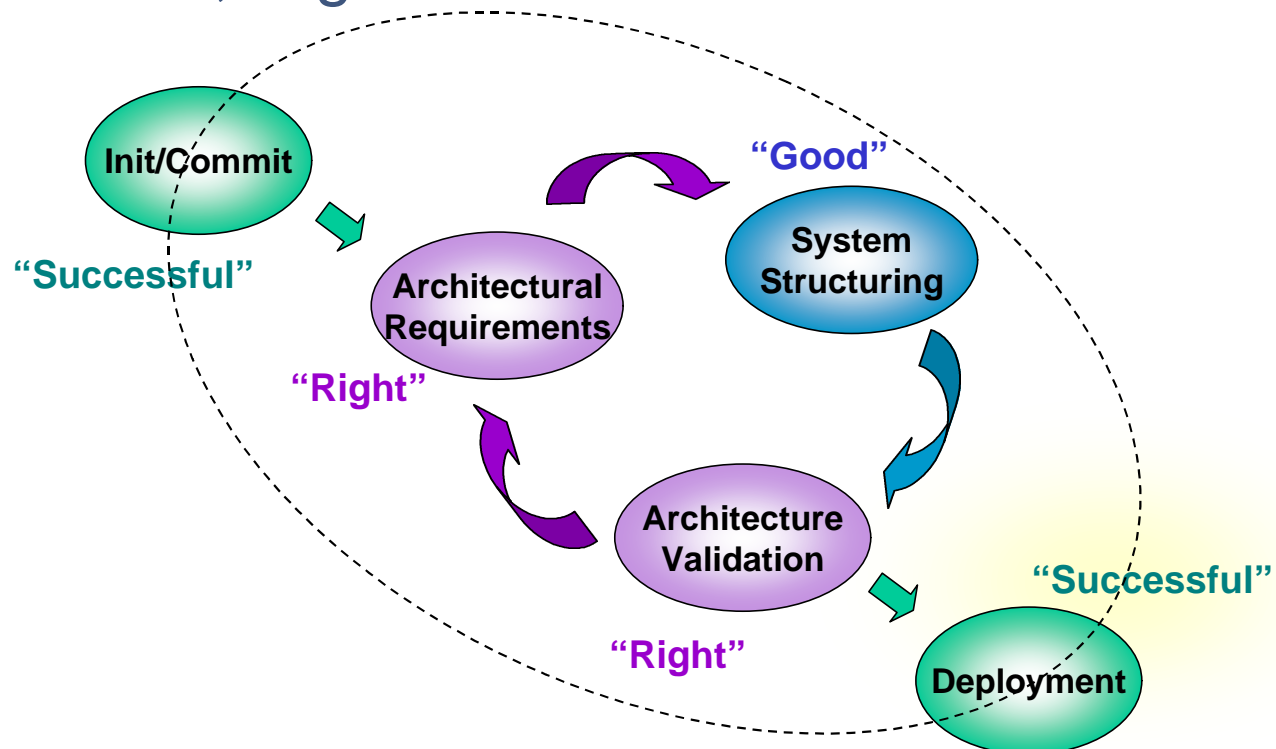
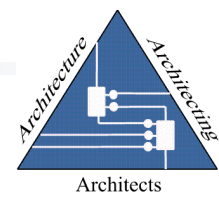
# Success: The Architect's Part Build Commitment and Lead



- Build commitment
  - create a compelling architecture vision
  - build the business case for architecture
  - create architecture roadmap
- Lead
  - the architecture team
  - engineering community
- Build the foundation for leadership
  - relationships
  - vision, excitement, alignment



# Visual Architecting Process Good, Right and **Successful**!



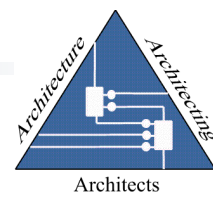
Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 27

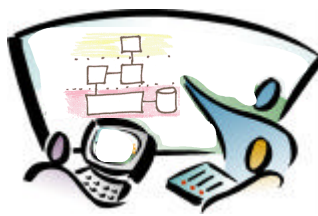
## Responsibility for the Success of the Architecture: Deployment

The team of architects are responsible for creating a good and right architecture, and **share** the responsibility with the rest of the organization for a **successful** architecture.

# Success: The Architect's Part Deployment



- Communicate to enhance understanding and build enthusiasm
  - personal conversations
  - meetings and reviews
  - presentations and walkthroughs
  - documentation
    - overviews
    - specifications
    - examples
- Provide training
  - hands-on workshops or tutorials
  - examples to work through
- Conduct demonstrations
  - how to use the architecture
  - how it solves issues and meets its objectives and requirements
- Consult



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 28

## Architect's Role in Deployment: Communicate, Educate, Consult

Not everyone in the product space will be current or even aware of your architecture, or the thought and motivation behind it. So . . .

*Collaboration, partnering and communication* is needed to:

- ♦ get buy-in to the goodness of the architectural approach (and to make it good in the first place). For certain goals, documentation is good enough. However, to ensure the confidence and commitment to adopt the architecture and make it stick, more interactive communication is needed.
- ♦ help the developers understand the architecture so that they maintain its integrity throughout the design and implementation process.

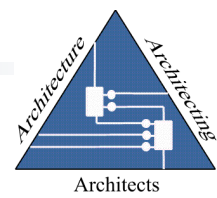
Eb Rechtin (1992) offered the following principle for ensuring success with an architecture: "Communicate, communicate, communicate", and exclaimed "There is no such thing as immaculate communication".

To be most effective, the communication process needs to start early and be ongoing during architectural requirements capture and architectural design. A careful balance must be struck between gaining the buy-in of the broader organization and "too many cooks" or failure through architecture by (extended) committee.

## Training

A training program for the developers should educate them on the role of architecture in the development process (Batman, SEI 1999) as well as the use of your architecture.

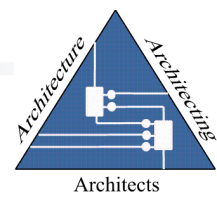
# Success: Management's Part Deployment



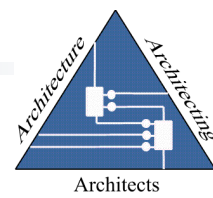
- Support the architecture
  - champion the architecture
    - believe in it
    - show enthusiasm and commitment
  - lead by following well!
- Lead the change in mindset
  - architecture decisions set context for other engineering decisions
    - individuals do not have complete autonomy
- Empower the architects



## Success: Engineer's Part Deployment



- For the architecture to have its desired impact, it must be followed
- That means engineers
  - have authority over decisions within their scope
  - but must defer to decisions made at broader scope
    - process for raising issues with decisions at broader scope



## What it Means to “Follow”

- Following is the **ACTIVE** acceptance of leadership
  - it is essential to the achievement of success
- Following involves
  - maintaining **goodwill** (the **real** silver bullet) towards the leader and fellow team members
  - accepting direction set by the leader
  - making your contribution
  - giving the leader the benefit of the doubt
    - accepting that compromises have to be made to move forward



### Example of Following

One architecture team had three of the most senior, most talented architects at HP. They knew that to be successful, all three could not try to lead. This would cause too much division in the team. They appointed a leader, and as Joe Sventek puts it, they let him be a "benevolent dictator with a baseball bat". That is, they allowed him to set direction, make difficult decisions to break logjams, and generally lead the team. This does not mean that they were not active in debates about how to solve the challenges of the architecture, but rather that they allowed the lead architect to make choices between alternatives when there was no consensus.

### Compromises Have to be Made

It is just as well to remember that:

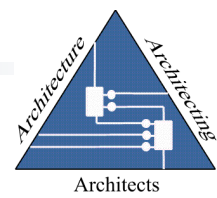
"the practice of architecture is a long and rapid succession of sub-optimal decisions, mostly made in partial light." Philippe Kruchten, 1999

and

"Good enough for each part is usually best for the whole system. When one part is maximized then there are inevitable losses for other parts." (Principles of Systems Thinking, <http://www.lambent.com/systems/sysprin.htm>)

### Some Images of Following

- US military versus Russian military
- Climbing
- Professor/student; craftsman/apprentice
- Lemmings!



## Why Following is Important

- There is no leading without following
- Many must contribute, collaborate, work to achieve the goal



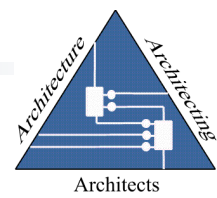
Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 32

### Collaboration is Key to Team Success

The architecture team needs to operate as a *team*: "a small number of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable" (Katzenbach and Smith, 1993)

# What it Means to “Get Out of the Way”



- “Getting out of the way” is not impeding progress
- It involves
  - Goodwill
  - trust
  - doing no harm through action or inaction



Copyright 2003 Bredemeyer Consulting  
<http://www.bredemeyer.com>

Visual Architecting Process Overview  
Slide 33

## Images of Getting Out of the Way

- Aikido

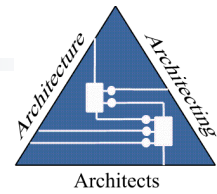
## Images of Not Getting Out of the Way

- Cow on the train tracks (bad for the cow)
- Truck on the train tracks (bad for the truck and the train)
- Boulder on the train tracks (bad for the train)

## What it Does Not Mean

This is not to say that there aren't times where it is appropriate to raise concerns and question approaches. It does mean that when you do, it should not be destructive. Indeed, there are times in the architecture lifecycle when the whole architecture initiative should be scrutinized for alignment with business need and viability. Also, for any architectural decision, there should be a decision process that takes input and provides opportunity for review and raising concerns. Once decisions are made, good following means that they are accepted (even if viewed as locally sub-optimal) and supported.

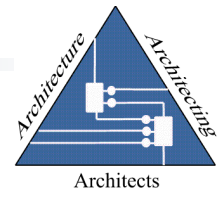
# Why Getting Out of the Way is Important



- Too many directions, targets, objectives, visions
  - dilutes attention
  - prevents traction
  - confuses
- Too many opinions slows progress
  - each additional voice takes time to be heard
  - it is harder to reach consensus
- + Focus on *your* responsibilities/get other things done
- + Create synergy, leverage

## Images of Getting Someone Out of the Way

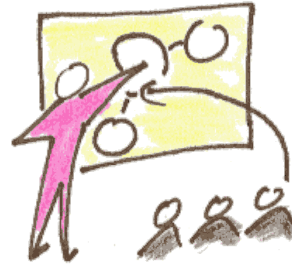
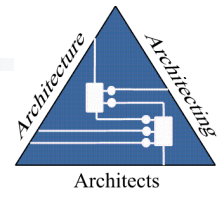
- Football blocker
- Rhode Island embargo



## Key Message

- A “good” and “right” architecture will not “just happen”
- It requires action
  - leading the effort
  - following, supporting, contributing to the effort
  - getting out of the way
- Everyone has a role to play in the success

# Review



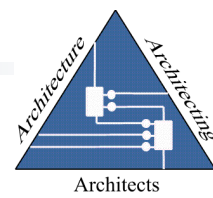
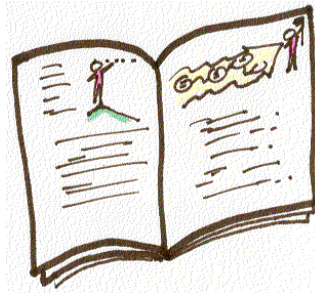
- We have covered
  - *How* to create an architecture
    - Context⇒requirements⇒system structuring (form follows function but mediated by system qualities)
    - Abstract⇒detailed specification/blueprint
    - A good, right *and successful* architecture requires good leading, following and getting out of the way

## Where we are headed

... this module is taken from one of our seminars ... and in that context this note introduces the next module...



# References



- **Architecture**

- Cheesman, Jon and John Daniels, *UML Components*, Addison-Wesley 2001.
- Hofmeister, Nord and Soni, *Applied Software Architecture*, Addison-Wesley, 2000. (especially Ch. 6 pp. 125-157)

- **Architectural Patterns**

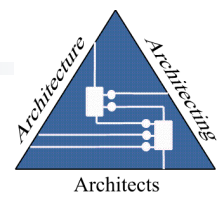
- Buschmann, et al, *Pattern-Oriented Software Architecture: A System of Patterns*, Wiley, 1996.
- Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects*, Wiley, 2000.
- Mary Shaw, Garlan David, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

## References for Modeling using UML

- Booch, Rumbaugh and Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.
- Rumbaugh, Jacobson, and Booch, *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.
- Arlow and Neustadt, *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley, 2002.
- Larman, Craig, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, Prentice-Hall, 2002.
- Kruchten, Philippe, "Modeling Component Systems with the Unified Modeling Language", *Proceedings of the 1998 International Workshop on Component-based Software Engineering*, 1998.  
<http://www.sei.cmu.edu/icse98/papers/p1.html>
- The latest Unified Modeling Language (UML) Specification (see the UML documentation page the OMG web site (<http://www.omg.org>)).

## Other

- Clark and Fujimoto, "Power of Product Integrity", *Harvard Business Review*, Nov-Dec 1990.



## Resources

- Resources for Software Architects web site
  - <http://www.bredemeyer.com>
- Enterprise-wide IT Architecture web site
  - <http://www.ewita.com>
- SEI's Software Architecture site
  - [http://www.sei.cmu.edu/ata/ata\\_init.html](http://www.sei.cmu.edu/ata/ata_init.html)
- The Gaudi Project (Philips) Architecture Site
  - <http://www.extra.research.philips.com/natlab/sysarch/index.html>