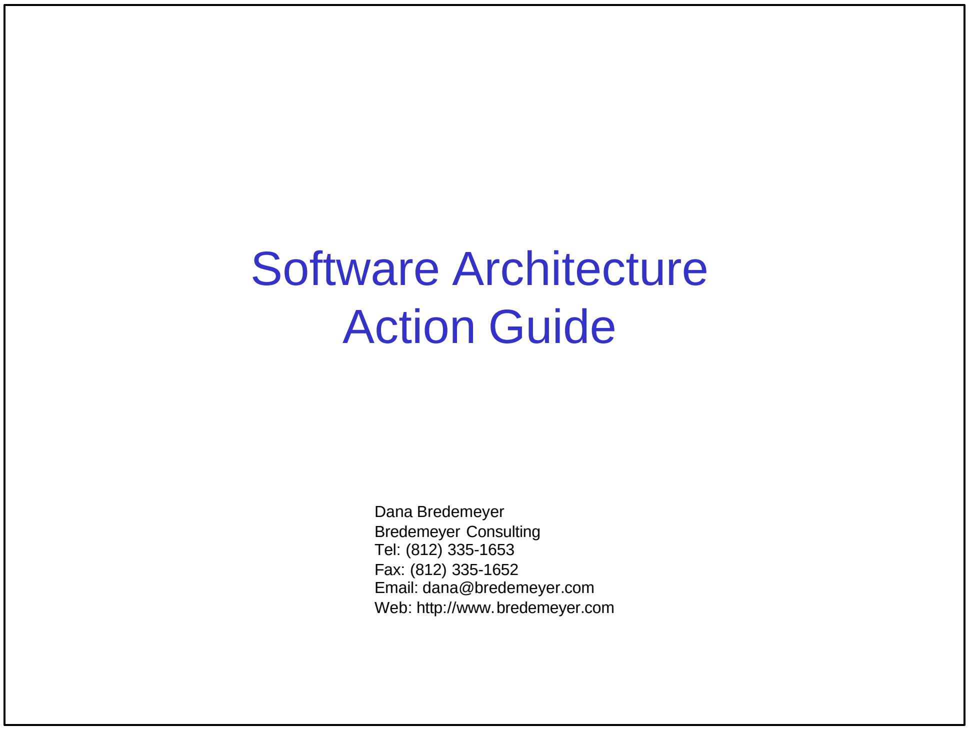
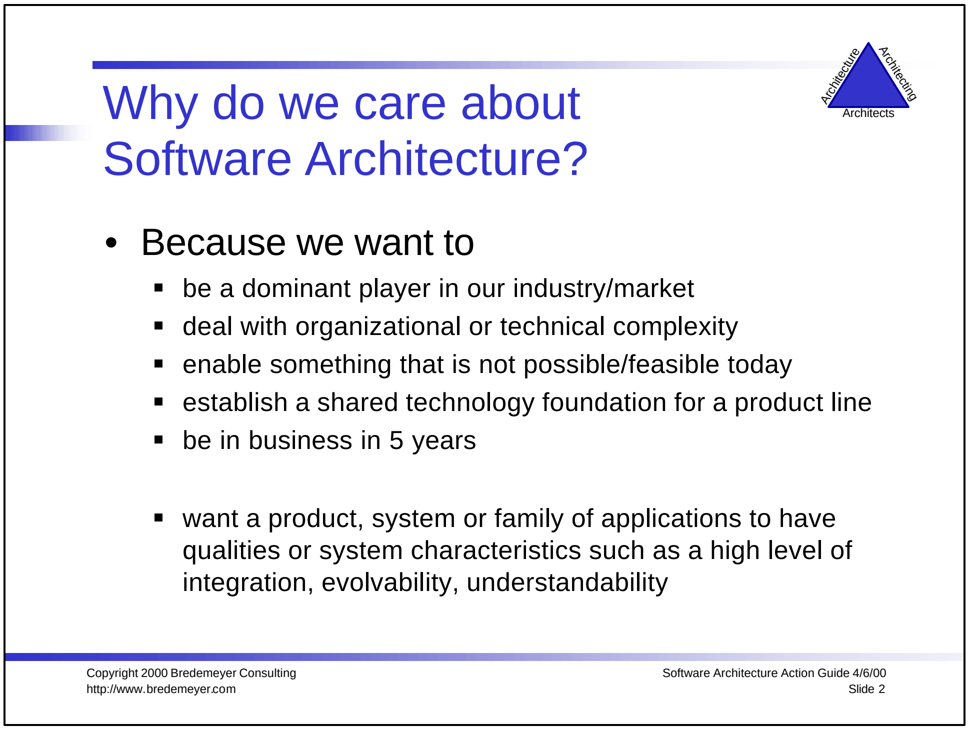


Software Architecture Action Guide

Dana Bredemeyer
Bredemeyer Consulting
Tel: (812) 335-1653
Fax: (812) 335-1652
Email: dana@bredemeyer.com
Web: http://www.bredemeyer.com



Architecture
Architecting
Architects

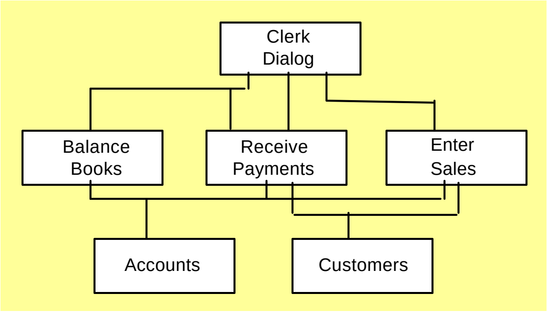
Why do we care about Software Architecture?

- Because we want to
  - be a dominant player in our industry/market
  - deal with organizational or technical complexity
  - enable something that is not possible/feasible today
  - establish a shared technology foundation for a product line
  - be in business in 5 years

  - want a product, system or family of applications to have qualities or system characteristics such as a high level of integration, evolvability, understandability

Architecture Architecting Architects

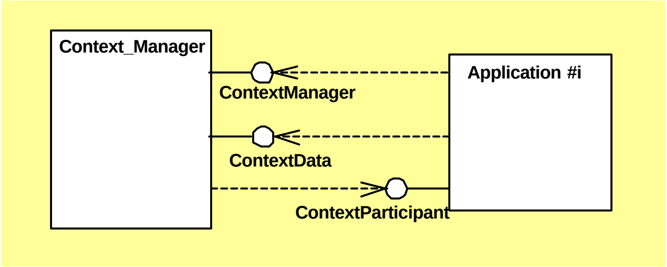
Software Architecture
Components and Relationships



Conceptual Architecture
- Abstract, system-wide view
- Basis for communication

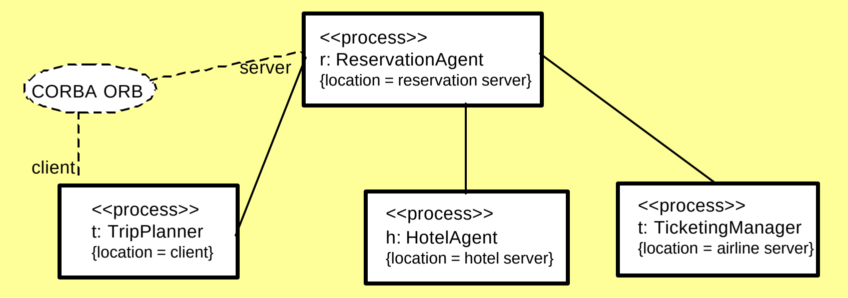
Architecture Architecting Architects

Software Architecture
Components and Interfaces



Logical Architecture
- “Blueprint”: Precise, unambiguous, actionable
- Basis for supplier/client contract

Software Architecture
Components and Location



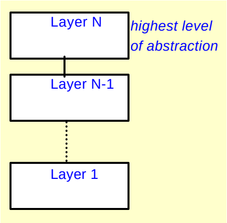
Execution Architecture
- Configuration of components at run-time
- Basis for system tuning during design

Software Architecture
System-Level Concerns

Principles

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.*

Style



Key Mechanisms and Concepts

e.g., component interconnection mechanisms

Meta-Architecture
- Guiding principles and strategies
- Basis for system decomposition and composition

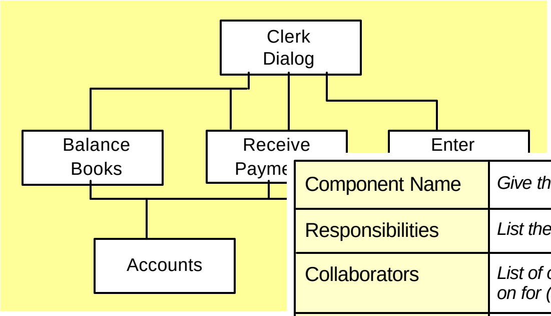
Architecting How To

Guiding Principles and Strategies

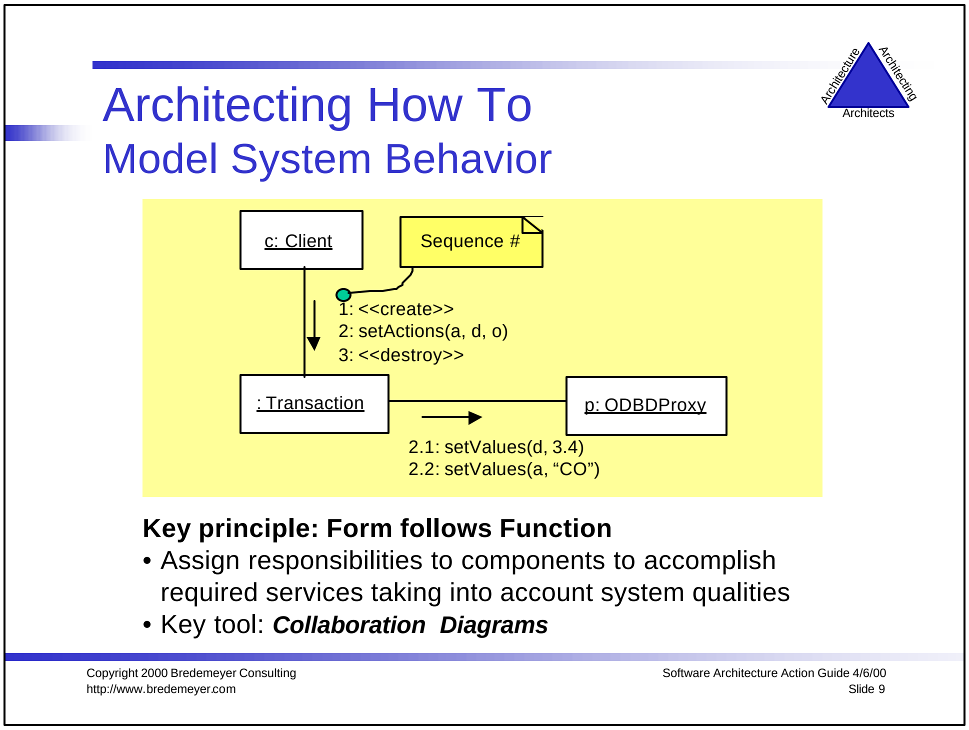
Principle Name	*Give the principle a catchy name.*
Description	*Statement of the principle.*
Rationale/Benefits	*Describe the reasoning behind the principle. Where applicable, provide traceability to business or architectural objectives.*
Implications	*Identify implications such as actions that need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

Architecting How To

Identify Components (initial cut)



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*.



Architecture Architecting Architects

Architecting How To

Model System Behavior

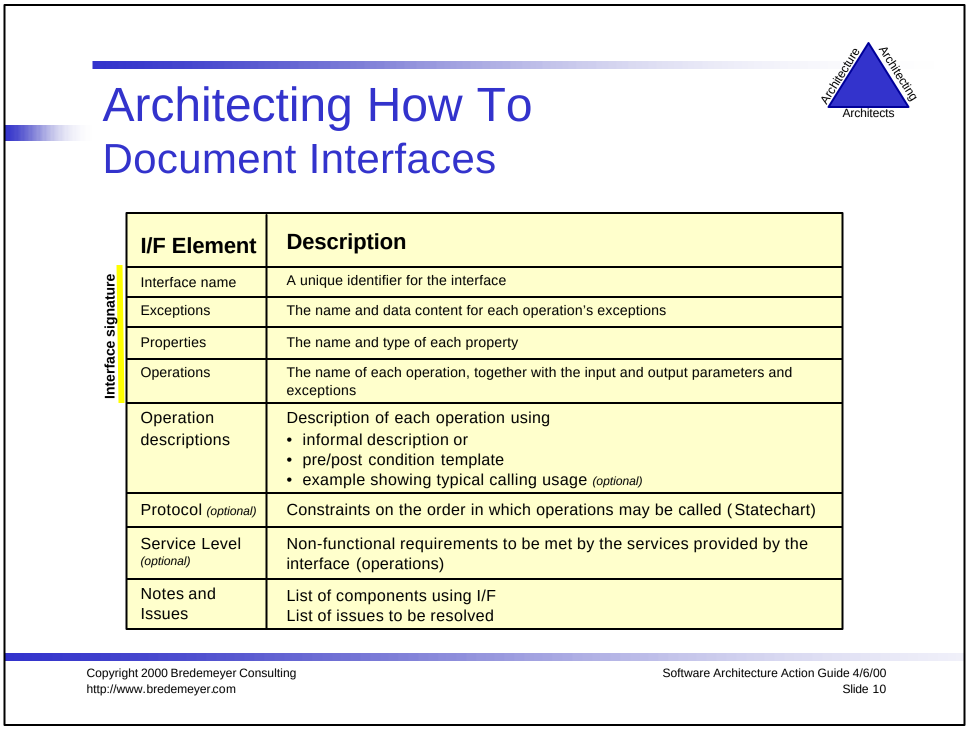
c: Client
Sequence #
1: <<create>>
2: setActions(a, d, o)
3: <<destroy>>
: Transaction
p: ODBDProxy
2.1: setValues(d, 3.4)
2.2: setValues(a, “CO”)

Key principle: Form follows Function

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

http://www.bredemeyer.com

Software Architecture Action Guide 4/6/00
Slide 9



Architecture Architecting Architects

Architecting How To

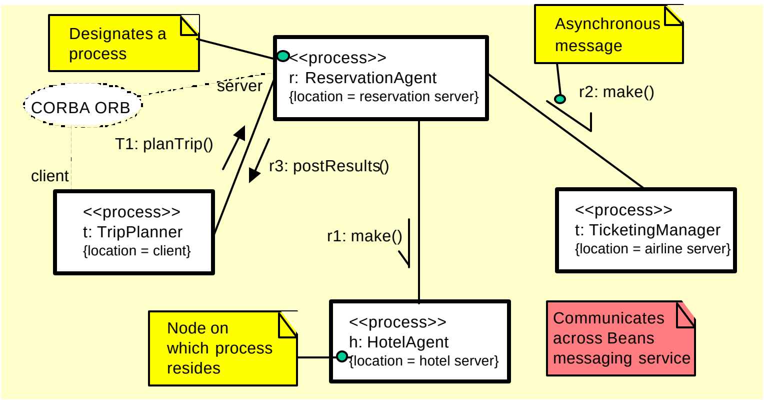
Document Interfaces

	I/F Element	Description
Interface signature	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 • informal description or • pre/post condition template • example showing typical calling usage *(optional)*
	Protocol *(optional)*	Constraints on the order in which operations may be called (Statechart)
	Service Level *(optional)*	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

http://www.bredemeyer.com

Software Architecture Action Guide 4/6/00
Slide 10

Architecting How To
Allocate Components to Processes

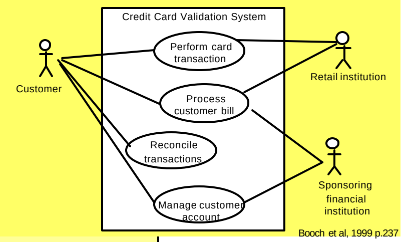


Key tool: ***Collaboration Diagrams***

Diagram from Booch et al, 1999

Architecting How To
Functional Requirements

Use Case	Validate User
Actors	Customer
Steps	1. The system prompts the customer for a PIN nur 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 commiting 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.



Booch et al, 1999 p.237

http://www.bredemeyer.com

Architecting How To

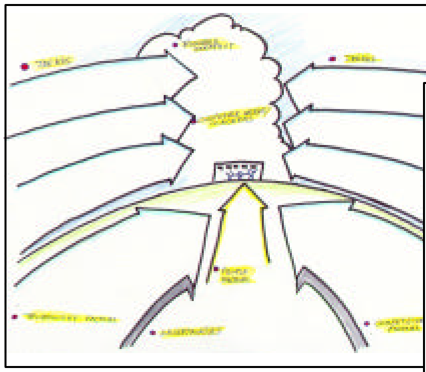
Non-Functional Requirements

Use Case	*Use case identifier and reference number*
Description	*Goal to be achieved by use case and sources for requirement*
Actors	*List of actors involved in use case*
Assumptions	*Conditions that must be true for use case to terminate successfully*
Steps	*Interactions between actors and system that are necessary to achieve goal*
Variations *(optional)*	*Any variations in the steps of a use case*
Non-Functional	*List of non-functional requirements that the use case must meet* The nonfunctional requirements are listed in the form: <**keyword**> : < requirement> Non-functional keywords include, but are not limited to **Performance**, **Reliability, Fault Tolerance**, **Frequency,** and **Priority**. Each requirement is expressed in natural language or an appropriate formalism.
Issues	*List of issues that remain to be resolved*

Service Level → Non-Functional

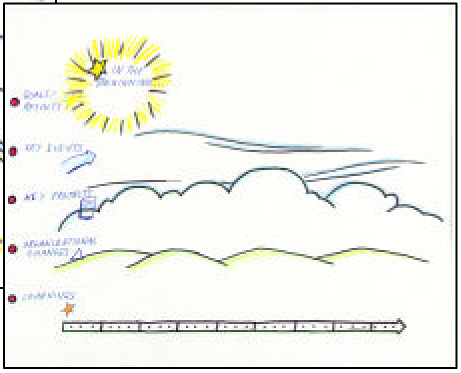
Architecting How To

Architecture Context

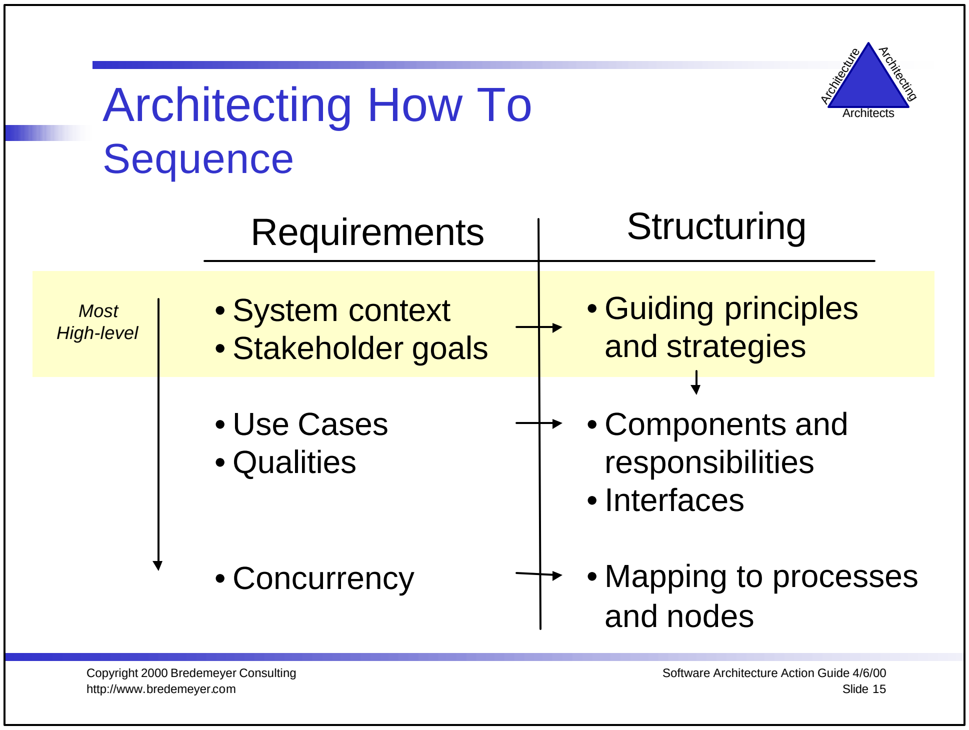


Context Map

Graphical History



Based on Grove Graphics Guides



Architecting How To
Sequence

Requirements

Structuring

Most High-level

- System context
- Stakeholder goals

- Guiding principles and strategies

- Use Cases
- Qualities

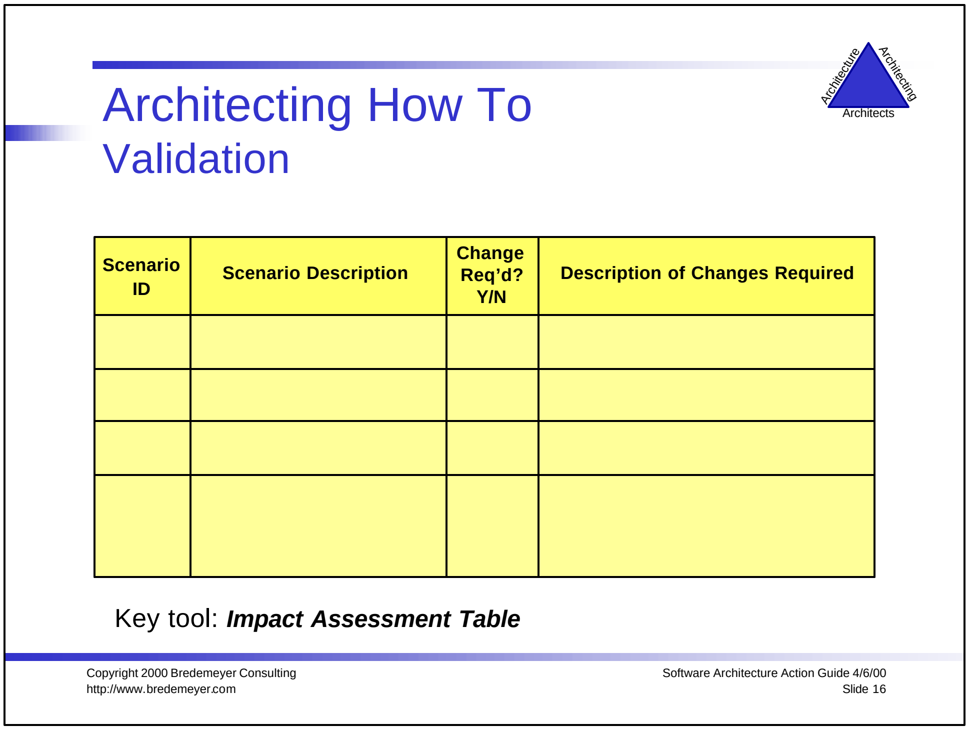
- Components and responsibilities
- Interfaces

- Concurrency

- Mapping to processes and nodes

http://www.bredemeyer.com

Software Architecture Action Guide 4/6/00
Slide 15



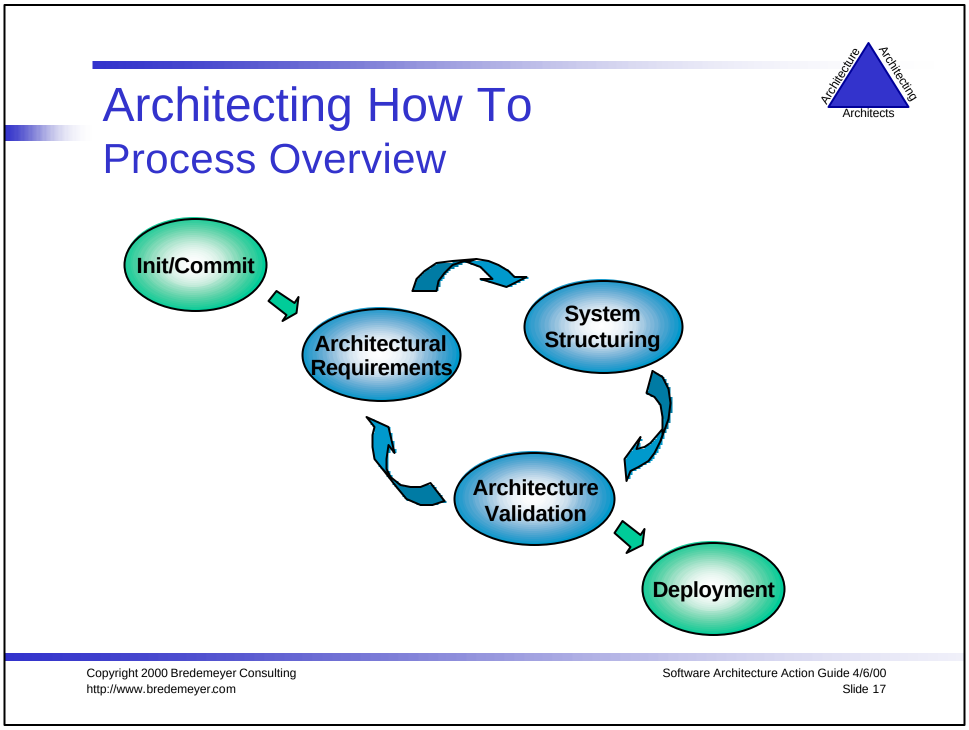
Architecting How To
Validation

Scenario ID	Scenario Description	Change Req'd? Y/N	Description of Changes Required

Key tool: ***Impact Assessment Table***

http://www.bredemeyer.com

Software Architecture Action Guide 4/6/00
Slide 16

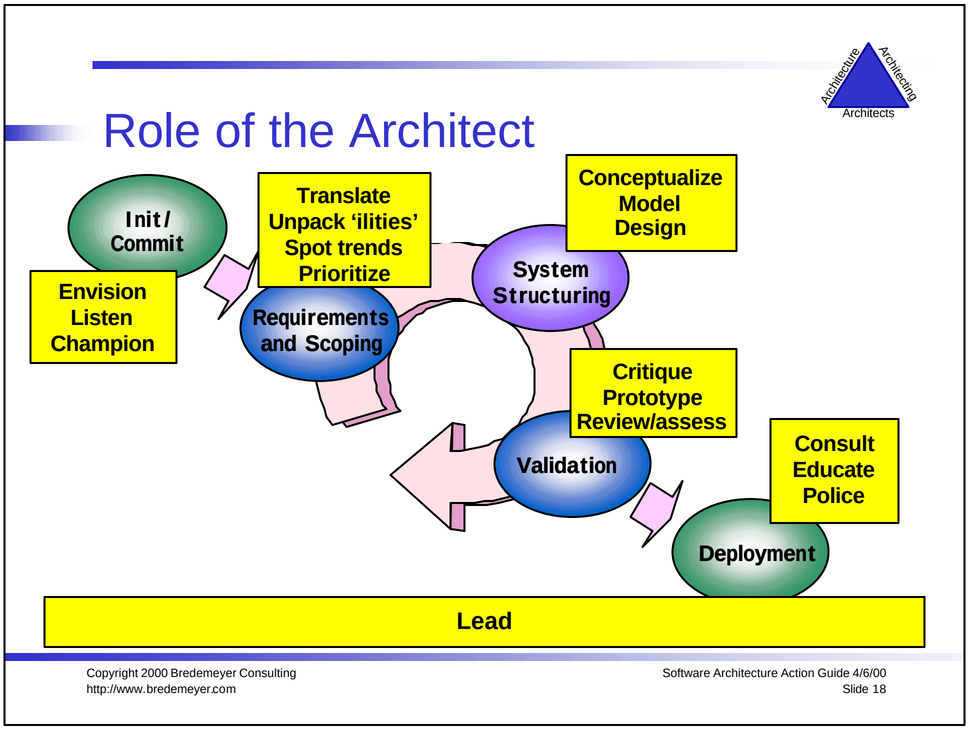


Architecting How To

Process Overview

Copyright 2000 Bredemeyer Consulting
http://www.bredemeyer.com

Software Architecture Action Guide 4/6/00
Slide 17



Role of the Architect

Copyright 2000 Bredemeyer Consulting
http://www.bredemeyer.com

Software Architecture Action Guide 4/6/00
Slide 18