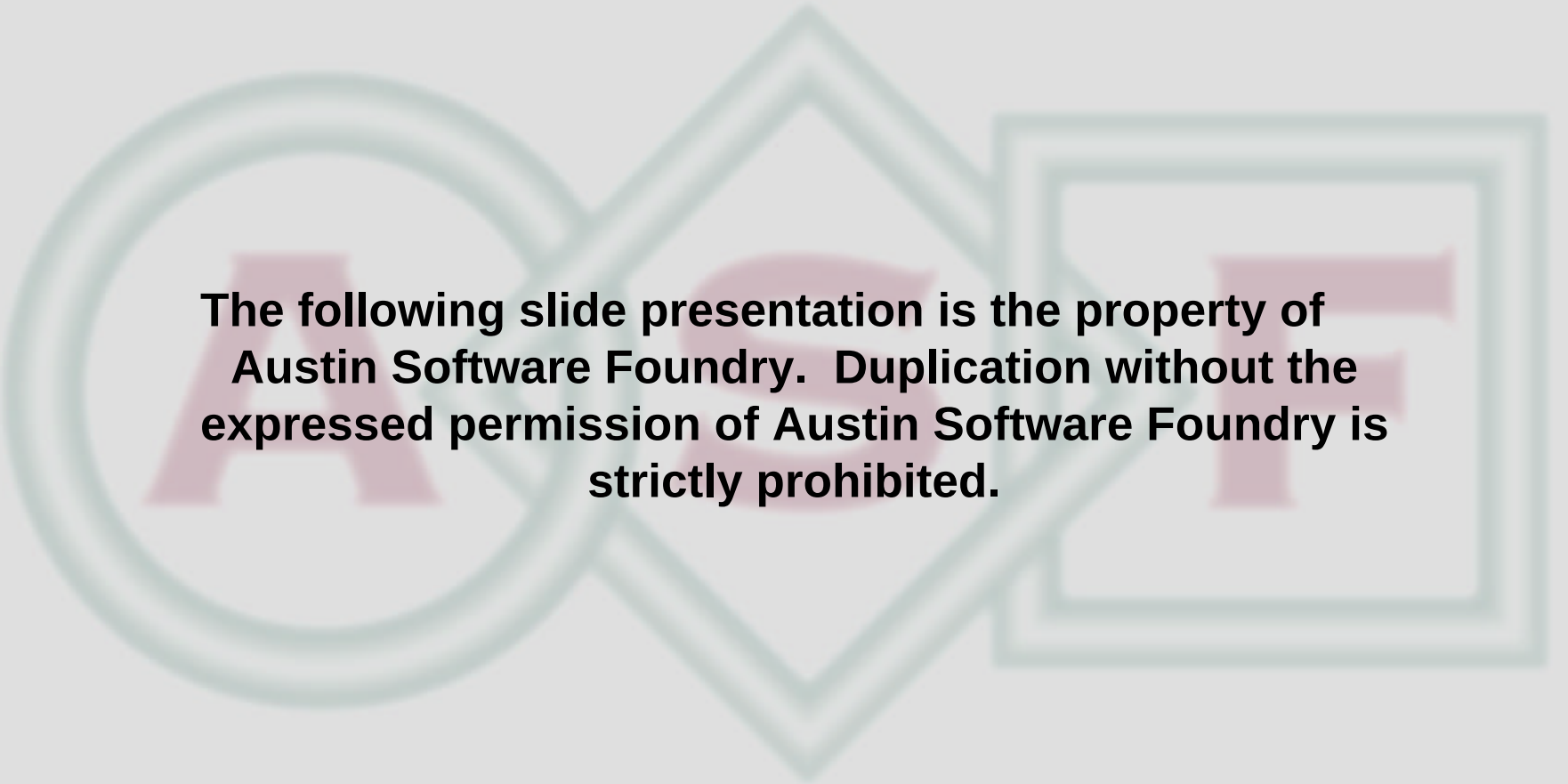


The following slide presentation is the property of Austin Software Foundry. Duplication without the expressed permission of Austin Software Foundry is strictly prohibited.

Architectures for Object-Oriented Development

Topics

- **Architecture Overview**
- Pattern Overview
- Visual Patterns
- Domain Patterns
- Design Patterns
- Applying Patterns to Web Development

Software Development Challenges

- **Software development organizations are *not keeping up* with the demand for new applications, or maintenance of existing applications**
- **Software *quality is elusive* even for the best development organization**
- **Line of business software applications are *too expensive***
- **Demands for *collaborative and distributed* applications is increasing**
- **True distributed application development (including “the Web”) is *much more complex* than client/server development**

Software Development Challenges

- General techniques for designing software do not address the solution of specific problems, for example:
  - System analysis and design guidelines that objects are found by discovering people, places and things used in the domain
  - System design and construction guidelines for using inheritance, polymorphism, encapsulation
- General tools for constructing software do not provide solutions for specific problems
  - PowerBuilder doesn't come with pre-defined classes for handling collections of business objects
  - Rational Rose doesn't come with templates for classes and interactions used to model a publish/subscribe architecture

Software Development Challenges

- **Architects have consistently demonstrated the ability to design and construct amazingly complex, elegant structures**
  - **The Golden Gate Bridge**
  - **The Empire State Building**
- **Architects use observed patterns to solve specific architectural problems not addressed by general guidelines**

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

  Christopher Alexander, professor of architecture at the University of California at Berkeley in *The Timeless Way of Building*

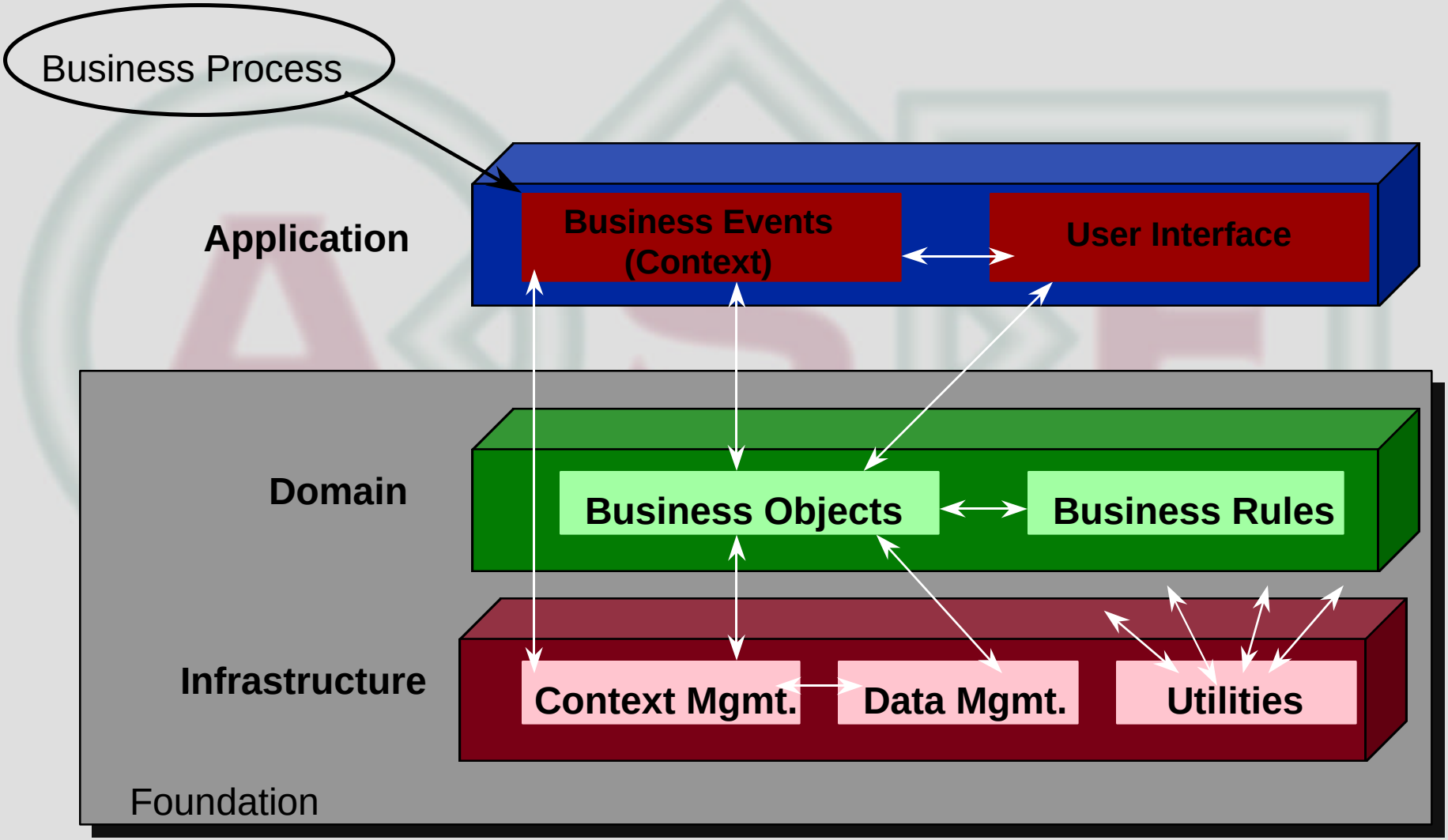
Architecture-Driven Software

- **Complex architectures are based on combinations and sequences of patterns**
- **Architecture-Driven Software uses *business and application architectures* to design and construct complex systems reliably**
- **Architecture-Driven Software is based upon**
  - **Application partitioning and layering patterns and techniques**
  - **Accepted "*Visual patterns*" that can be used to create intuitive, easy to use applications**
  - **Accepted "*Domain patterns*" that can be applied to many business problems consistently**
  - **Accepted "*Design patterns*" for solving system infrastructure problems reliably and predictably**

What is a “Business Architecture”?

- **Business architecture is the *combination/sequence* of business patterns that defines boundaries between business processes and entities**
- **Business architectures are expressed as *interactions* between the organizational entities, structures and processes that enable a business to create and deliver value to its customers**
- **A business architecture should use at least two *structural business (or domain patterns)* to provide a foundation business architecture**
  - **Use Case: event/entity/rule**
  - **Process: input/activity/output**

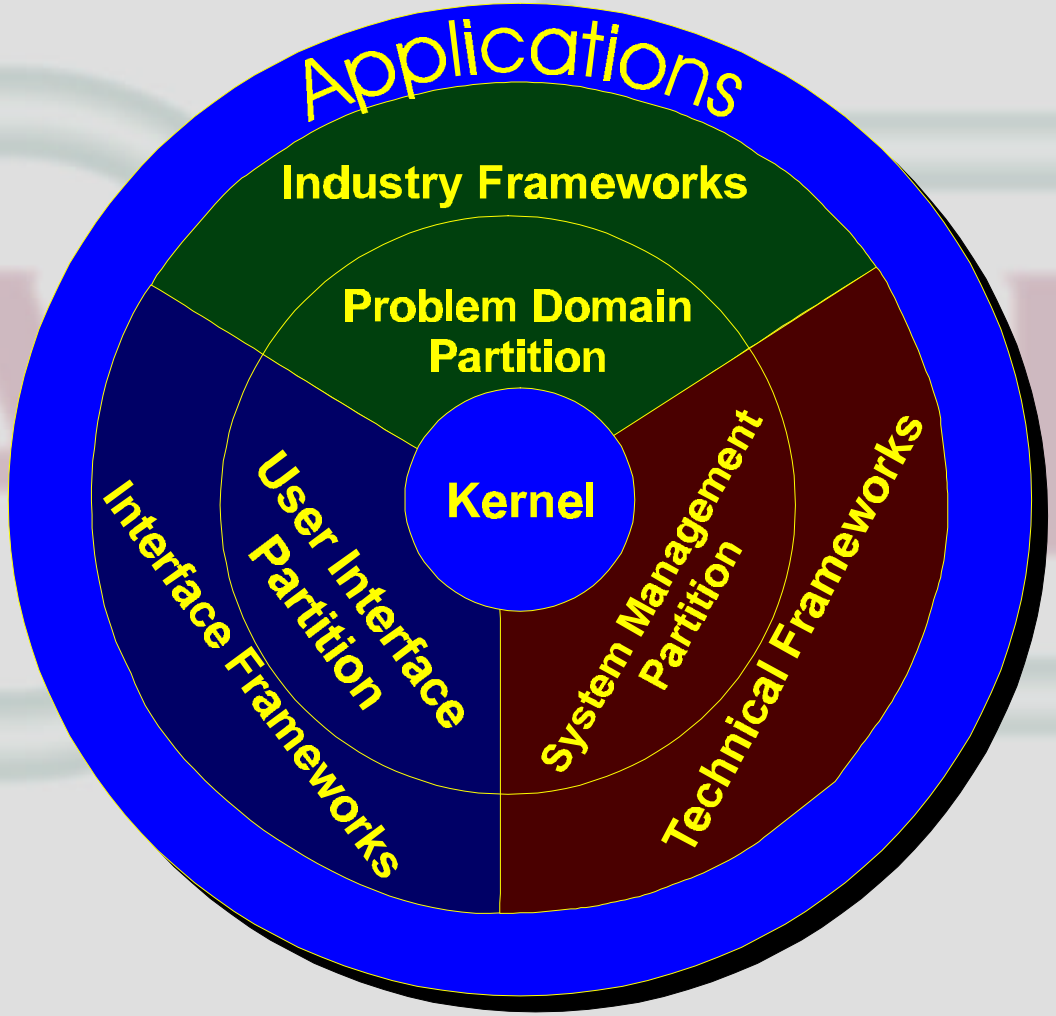
What is a “Business Architecture”



What is an “Application Architecture”?

- **Application architecture is the *combination/sequence* of system patterns that defines boundaries between application elements (classes, components, frameworks)**
- **These boundaries are expressed as *interfaces*, including:**
  - **Conventions for invoking/ordering operations across boundaries**
  - **Descriptions of operations included in the interface to components and frameworks**
- **Distributed systems must use at least two *structural system (or design patterns)* to provide a foundation application architecture**
  - **Layers**
  - **Partitions**

What is an “Application Architecture”



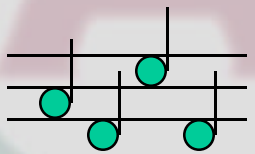
Topics

- Architecture Overview
- **Pattern Overview**
- Visual Patterns
- Domain Patterns
- Design Patterns
- Applying Patterns to Web Development

What is a pattern?

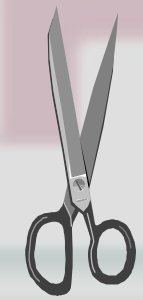
- **People apply patterns in many disciplines**

Pattern. A fully realized form, original, or model accepted or proposed for imitation.: something regarded as a normative example to be copied; archetype; exemplar [Webster's]



Song Structure
[8 and 12 bar blues]

Architectural Design
[Spanish Colonial, Victorian]



Dress Pattern
[Bridal Gown, Pant Suit]

What is a pattern?

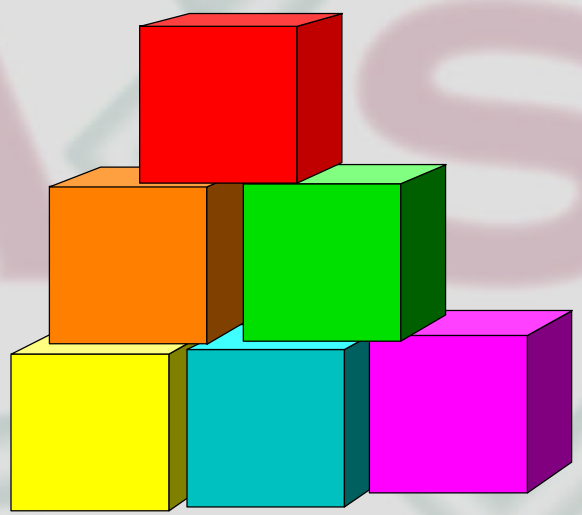
- **The current use of the term "*pattern*" is derived from the writings of architect Christopher Alexander who has written several books on the topic as it relates to building architecture**

> **"Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution. The pattern is, in short, at the same time a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it. It is both a process and a thing; both a description of a thing which is alive, and a description of the process which will generate that thing."**
>
> *From "The Timeless Way of Building", By Christopher Alexander*

- **His work applies to many other disciplines, including software development**

Patterns as Building Blocks

- **"With each pattern the *lowest-level building block* is standardized into a larger chunk or unit"**

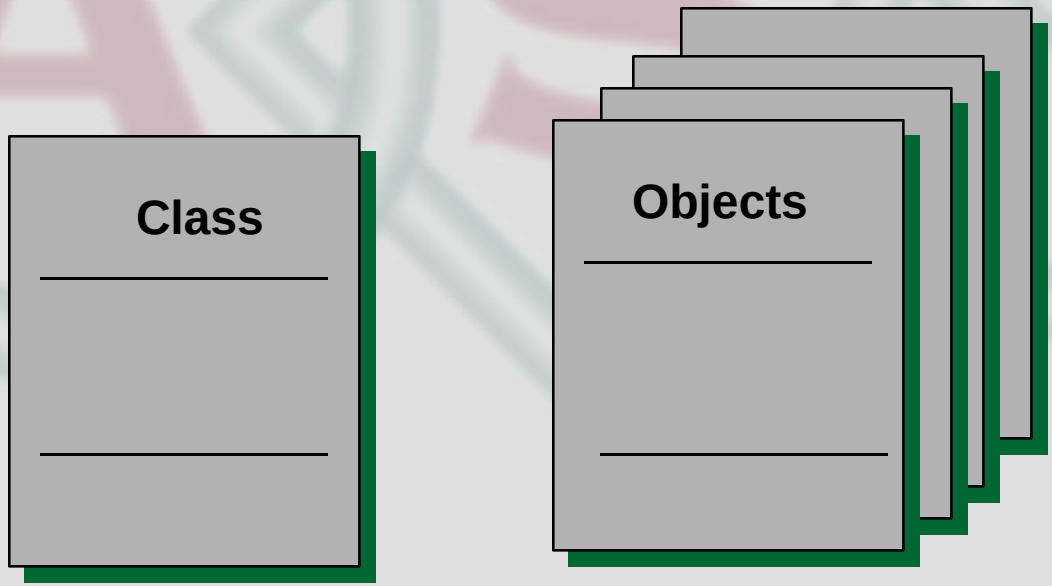


"As one uses patterns, one begins to think with that new building block, rather than with littler pieces"

From "The Timeless Way of Building", By Christopher Alexander

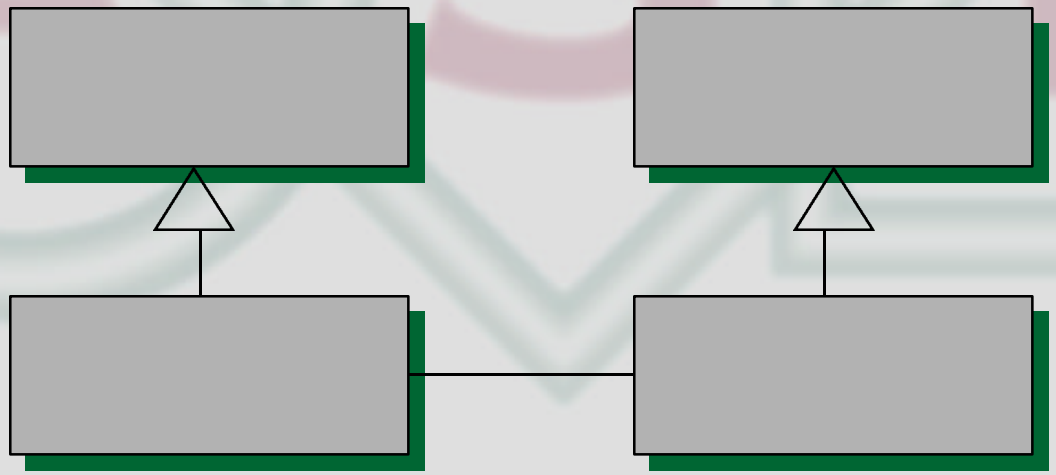
Object-oriented Patterns

- **The lowest level building block in object-oriented development is a *class* and its objects**



Object-oriented Patterns

- **An *object-oriented pattern* is an abstraction of a small grouping of classes and their relationships that is likely to be helpful again and again in object-oriented development.**



Types of Object-oriented Patterns

- **Visual Patterns**
  - Help to solve problems in user interface design
  - VERY new area of patterns research
  - http://www.concentric.net/~tmandel/
  - http://c2.com/ppr/ui.html
- **Domain Patterns**
  - Help to solve problems in the business domain
  - These patterns are being drawn from data modeling experiences
  - Fowler, M., Analysis Patterns: Reusable Object Models, Addison-Wesley, Reading, MA.
- **Design Patterns**
  - Help to solve problems in software and systems infrastructure design
  - Most of the effort on patterns to date has been here
  - http://st-www.cs.uiuc.edu/users/patterns/

Pattern Systems

- **A pattern system for software architecture is a *cohesive set of related patterns* which work together to support the construction and evolution of whole architectures.**
- **A pattern system adds deep *structure*, rich pattern *interaction*, and *uniformity* to a catalog of patterns.**

Software Pattern History

- 1992: Coad, P., "Object-Oriented Patterns," CACM 35(9), September, pp. 152-159.
- 1993: Booch, G., "Patterns," Object Magazine 3(2), July-August, pp. 24-27.
- 1994: Lea, D., "Christopher Alexander: An Introduction For Object-Oriented Designers," Software Engineering Notes 19(1), January, pp. 39-45.
- 1995: Gamma, E., Helm, R., Johnson, R., and Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, MA. (Gang of Four)
- 1996: Buschmann, F., et al., Pattern-Oriented Software Architecture, John Wiley and Sons, New York, NY. (Gang of Five, or “the Siemanns book”)
- 1997: Fowler, M., Analysis Patterns: Reusable Object Models, Addison-Wesley, Reading, MA.

Topics

- Architecture Overview
- Pattern Overview
- **Visual Patterns**
- Domain Patterns
- Design Patterns
- Applying Patterns to Web Development

Visual Pattern System Example

Pattern Category / Problem Category	Presentation Style	Interaction	Platform Specific Implementation
Structure	Command Line, Menu, WIMP, Objects	Cursor, Controller, Menu, Scrollbar, Button	
Navigation	SDI, MDI, Project, Workbook, Workspace, Browser	Pulldown Menu, Popup Menu, Direct Manipulation	
Data Layout	Grid, Freeform, Chart	Drag and Drop	
Page Layout	Header/Body/Footer Master/Detail	Hypertext	
Graphic Layout			

Browser User Interface Pattern

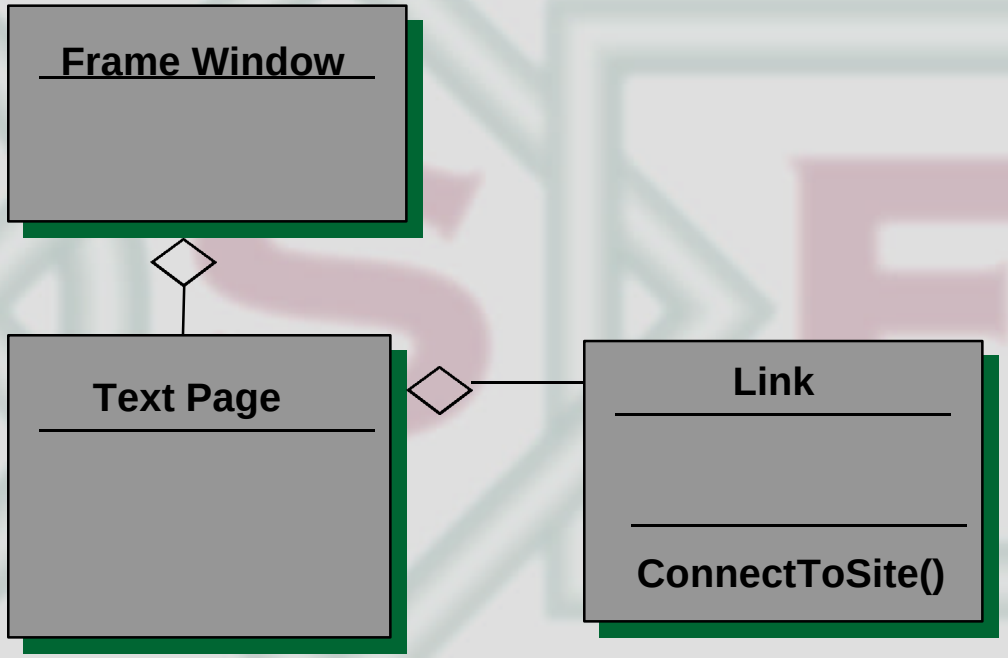
- **Problem:** You want to create a standard web user-interface with text on it and links to other web sites

- **Example:** Your company web site has pages for products, services, and personnel. Each page can have multiple links to other web sites.

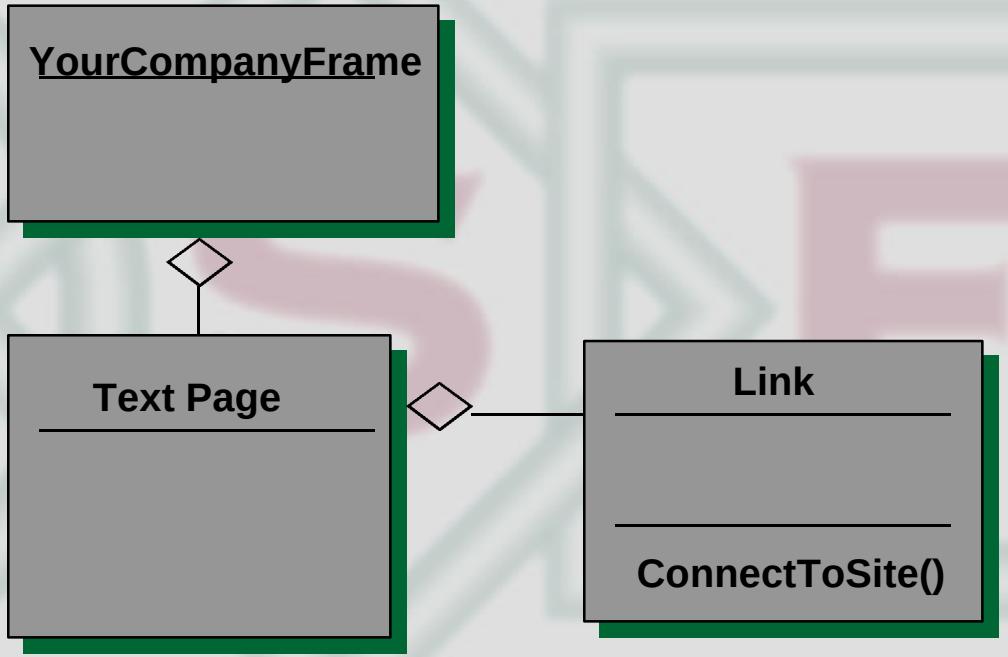
Browser User Interface Pattern

- **Solution:** Create a frame object that acts as a container for multiple text pages. Each text page can have multiple links associated with it
- **Pattern Name:** Browser
- **Intent:** The Browser user interface pattern provides a way to define the portions of a visual browser that can be reused many times with varied content

Browser User Interface Pattern



Browser User Interface Pattern Example



Topics

- Architecture Overview
- Pattern Overview
- Visual Patterns
- **Domain Patterns**
- Design Patterns
- Applying Patterns to Web Development

Domain Pattern System Example

Problem Category \ Pattern Category	Organization	Process	Domain Specific Implementation
Structure	Use Case		
Accountability	Player/Player Role	Authorization	
Planning	Plan	Proposal/Implementation Action	
Inventory & Accounting	Account Item/Item Description	Transaction Billing	
Measurement	Quantity Conversion Ratio	Event/Thing Remembered	
Diagnosis & Treatment	Observation	Hypothesis/Projection	
Trading	Contract Portfolio Quote	Valuation	

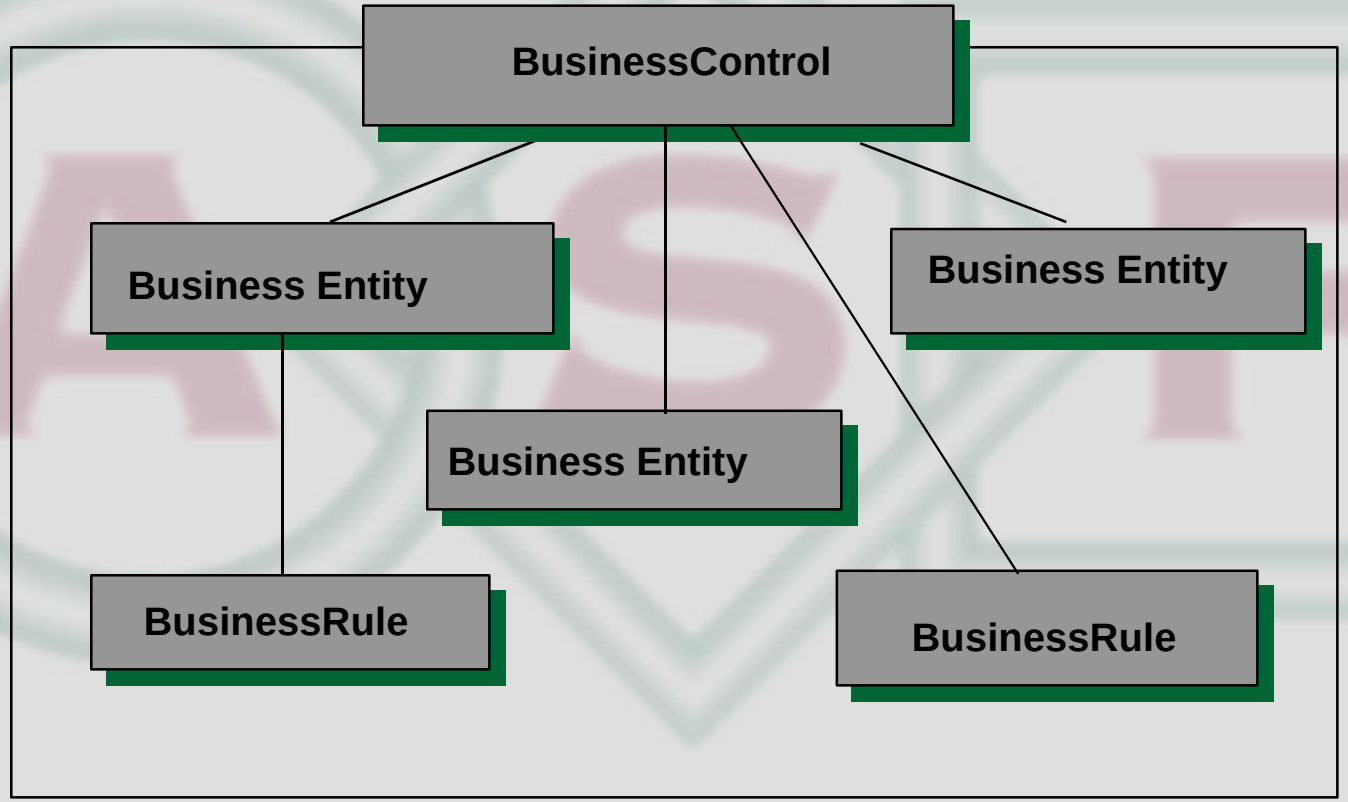
Use Case Pattern

- **Problem:** How do you translate the definition of a use case: the events, business responses, actors, etc. into domain objects and relationships?
  - How are business objects involved in the use case managed and coordinated?
  - How are logical business transactions defined and completed?
- **Example:** A business event occurs when a customer calls to place an order. Certain inventory and ordering rules are applied as a clerk takes the order and requests that the product(s) be pulled from inventory for shipping.

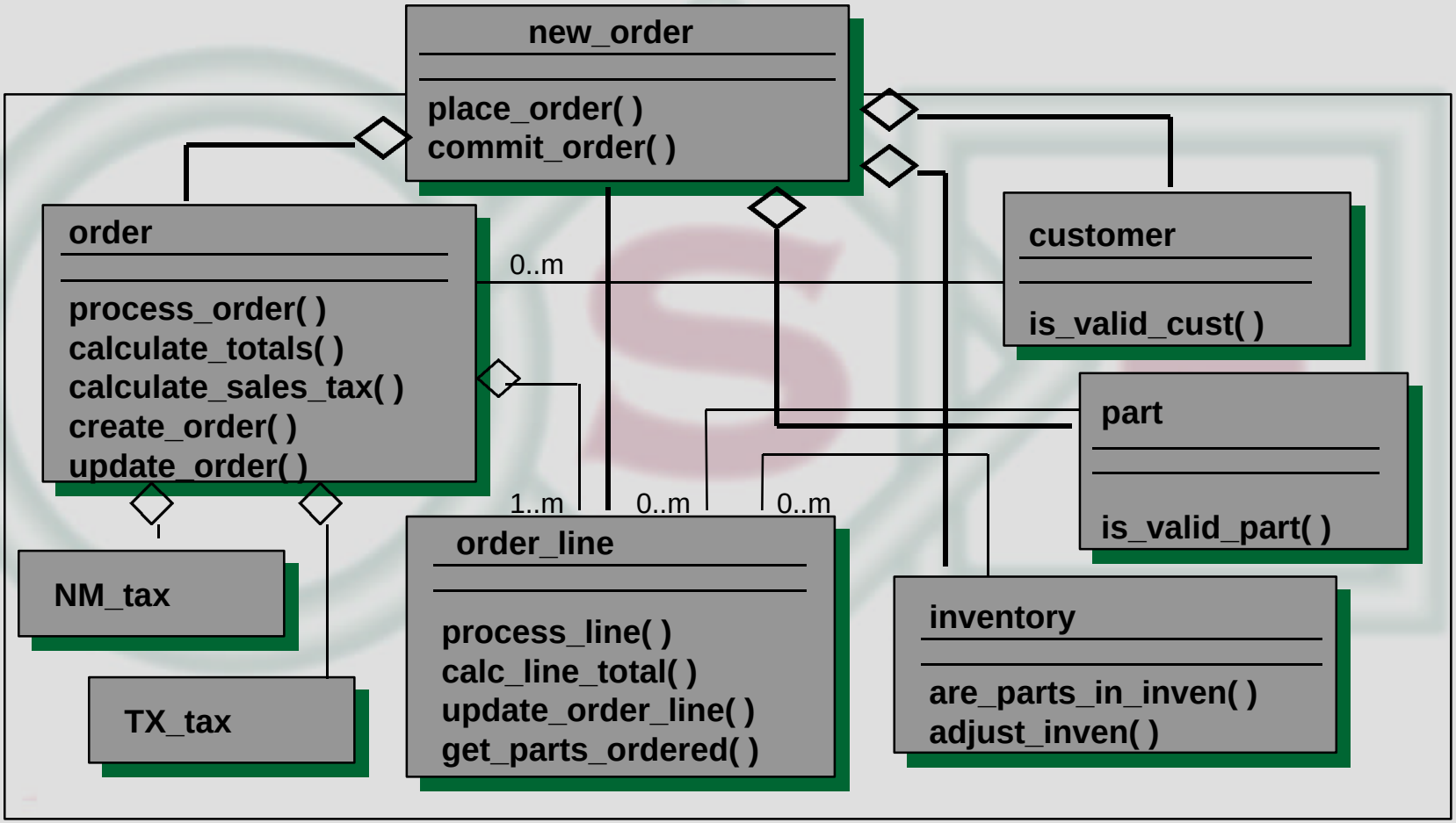
Use Case Pattern

- **Solution:** **Create a business component by defining three distinct elements:**
  - **Business Control** **(one or more processes triggered by a business event). Business control objects have a one-to-one relationship to the *use cases* defined for an application.**
  - **Business Entity** **(the data and behavior of real world things that are handled in the business). Business entities are defined by the business objects identified during domain analysis (I.e. subject/verb/direct object, crc cards)**
  - **Business Rule** **(the deterministic or heuristic rules that may alter/dictate the behavior of a Business Entity in a given use case context)**
- **Pattern Name:** **Use Case**
- **Intent:** **The use case domain pattern provides a way to map the definition of a use case into business control, business entity, and business rule domain objects.**

Use Case Domain Pattern



Use Case Domain Pattern Example



Topics

- Architecture Overview
- Pattern Overview
- Visual Patterns
- Domain Patterns
- **Design Patterns**
- Applying Patterns to Web Development

Design Pattern System Example

Problem Category \ Pattern Category	Architecture	Design	Language Specific Implementation
Structure	Layer Partition	Extension Context	
Distribution	Broker	Authorization	
Interaction	Model-View-Controller		
Adaptation	Microkernel	Adapter	
Creation		Object Factory Singleton	
Organization		Master/Slave	
Access		Proxy Facade Iterator Contract	
Management		Command Processor View Handler	
Communication		Observer	

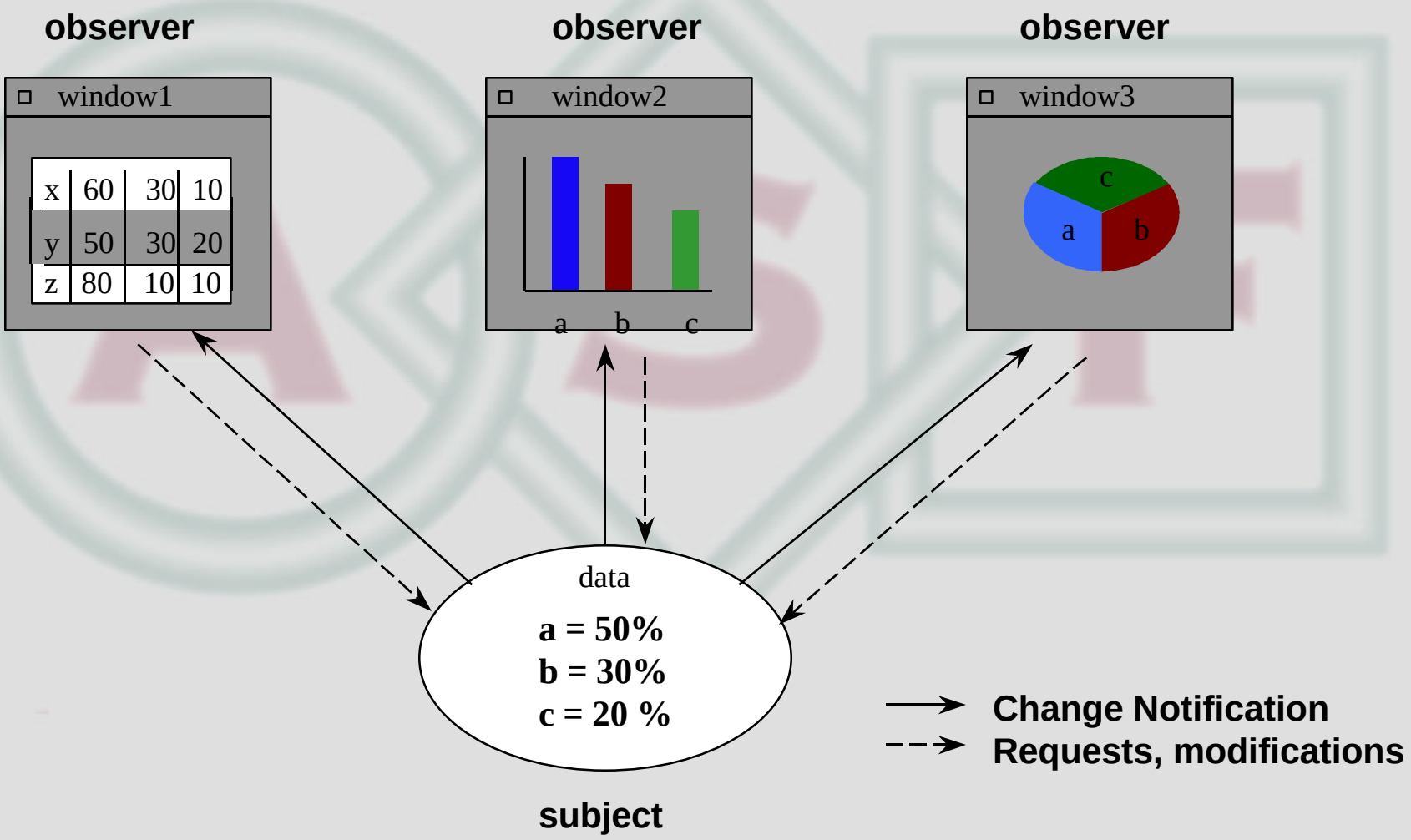
Observer Pattern

- **Problem:** **Data changes in one place in an application, but these changes need to be propagated throughout the system to other objects who are interested in this data.**
- **Example:** **You would like to centralize database access for an applications on a server, but have multiple clients using different styles of user interfaces be able to view and update data in the database. When one user changes the data, other users should see the changes immediately.**

Observer Pattern

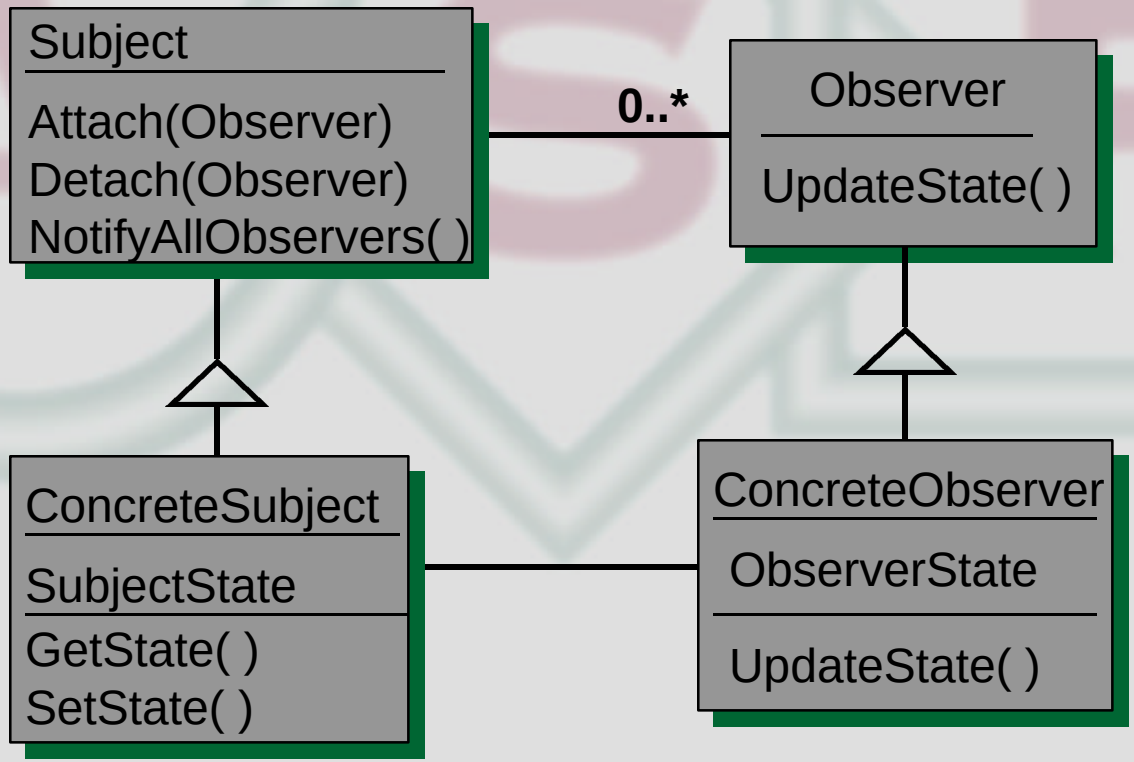
- **Solution:** Define subject objects with dependent observers that can be notified when a subject changes
- **Pattern:** Observer (publish-subscribe)
- **Intent:** The observer pattern allows a one-to-many dependency between objects to be defined so that when one object changes state, all its dependents are notified and updated automatically.

Observer Pattern Example



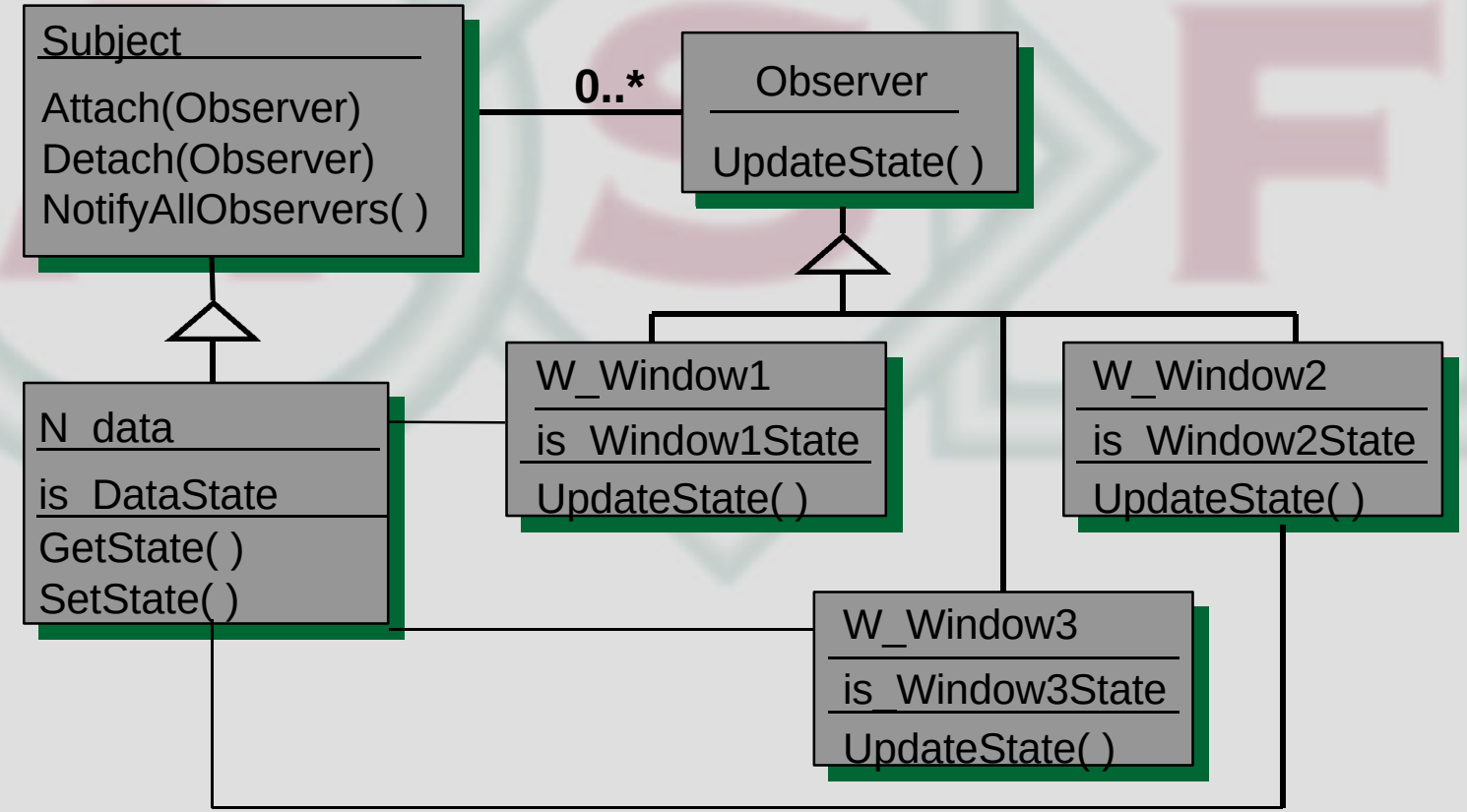
Observer Pattern

- **ConcreteSubject notifies its observers whenever a change occurs that could make its observers' state inconsistent with its own**



Observer Pattern Example

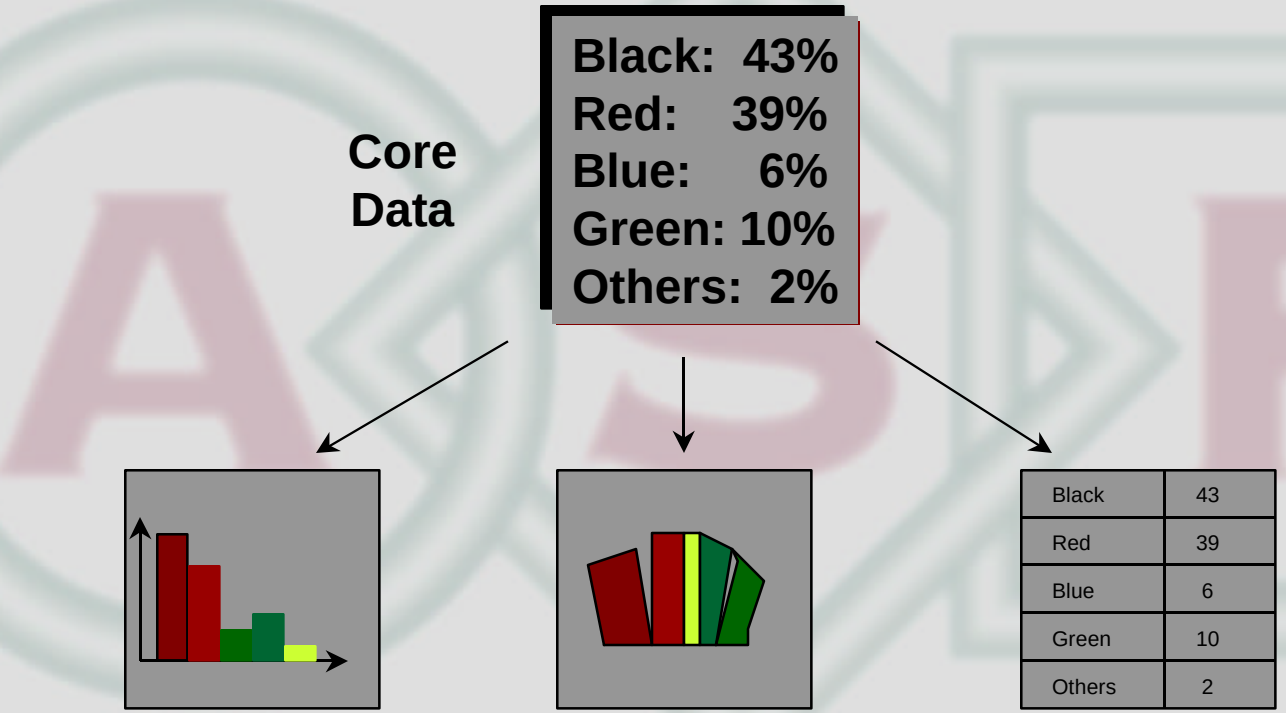
- **N_data notifies W_Window1, W_Window2, and W_Window3 whenever a change occurs**



Model-View-Controller Pattern

- **Problem:** **You would like to design a partitioned application that is loosely coupled with the flexibility to physically separate the user interface from application logic so the user interface can be modified, extended, or distributed independently of the application logic.**
- **Example:** **Your current system design allows users the choice of displaying data in a spreadsheet window, bar graph window, and pie chart window. When any user changes the spreadsheet (data), the other "views" should reflect the changes immediately.**

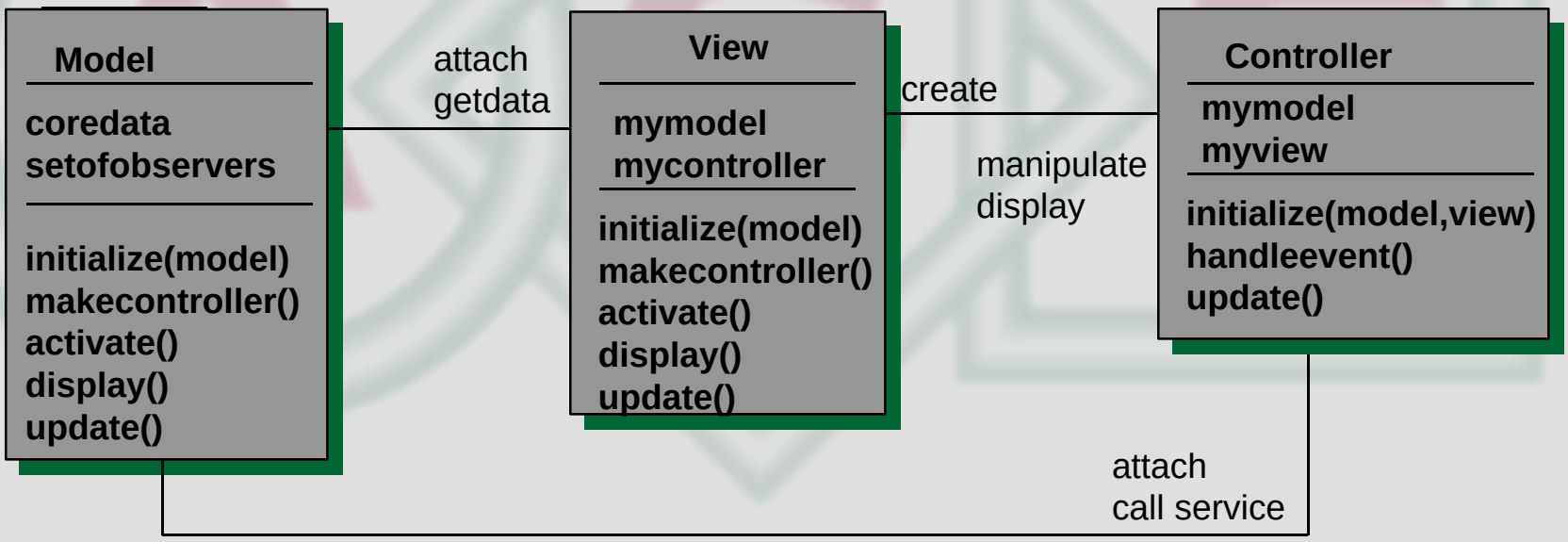
Model-View-Controller Pattern



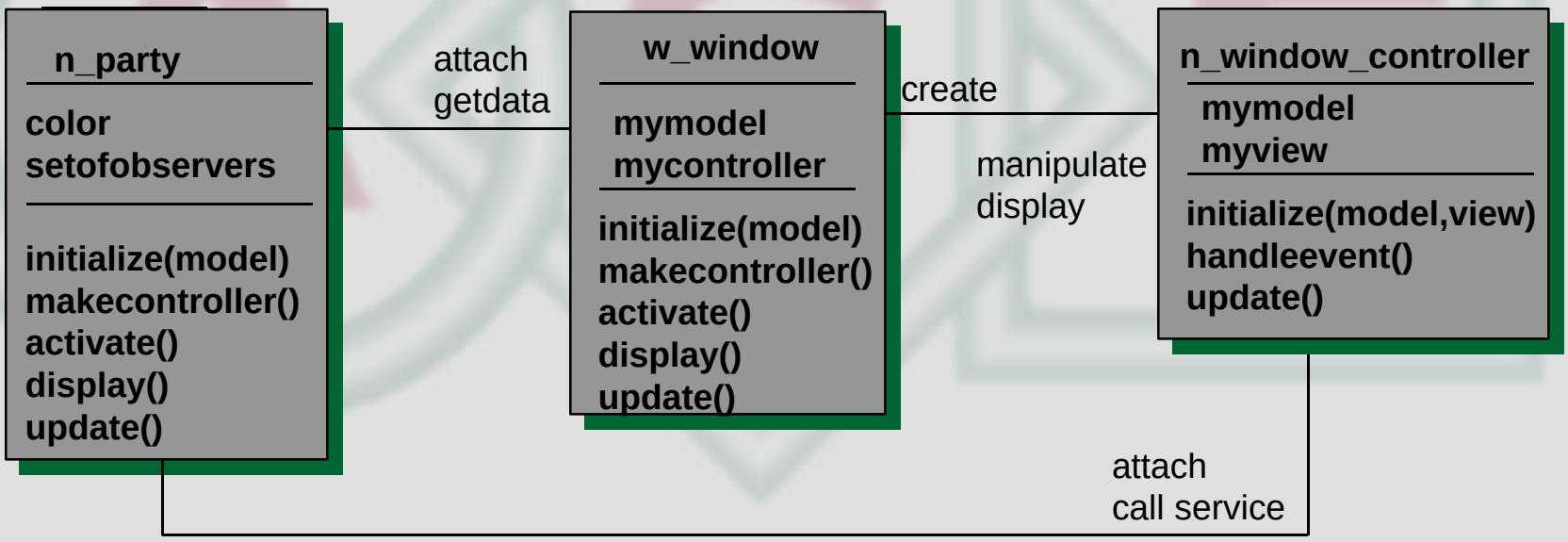
Model-View-Controller Pattern

- **Solution:** **Divide an interactive application into three types components.**
  - **The *model* contains the business contexts and business entities**
  - **The *view* is a collection of windows to display information to the user**
  - **The *controller* acts as a translation device between the view and model**
- **Pattern:** **Model-view-controller**
- **Intent:** **In order for a user(s) to request customized look-and-feel, or to communicate across a distributed connection, the business logic (model) must be separated from the user interface (view-controller)**

Model-View-Controller Pattern



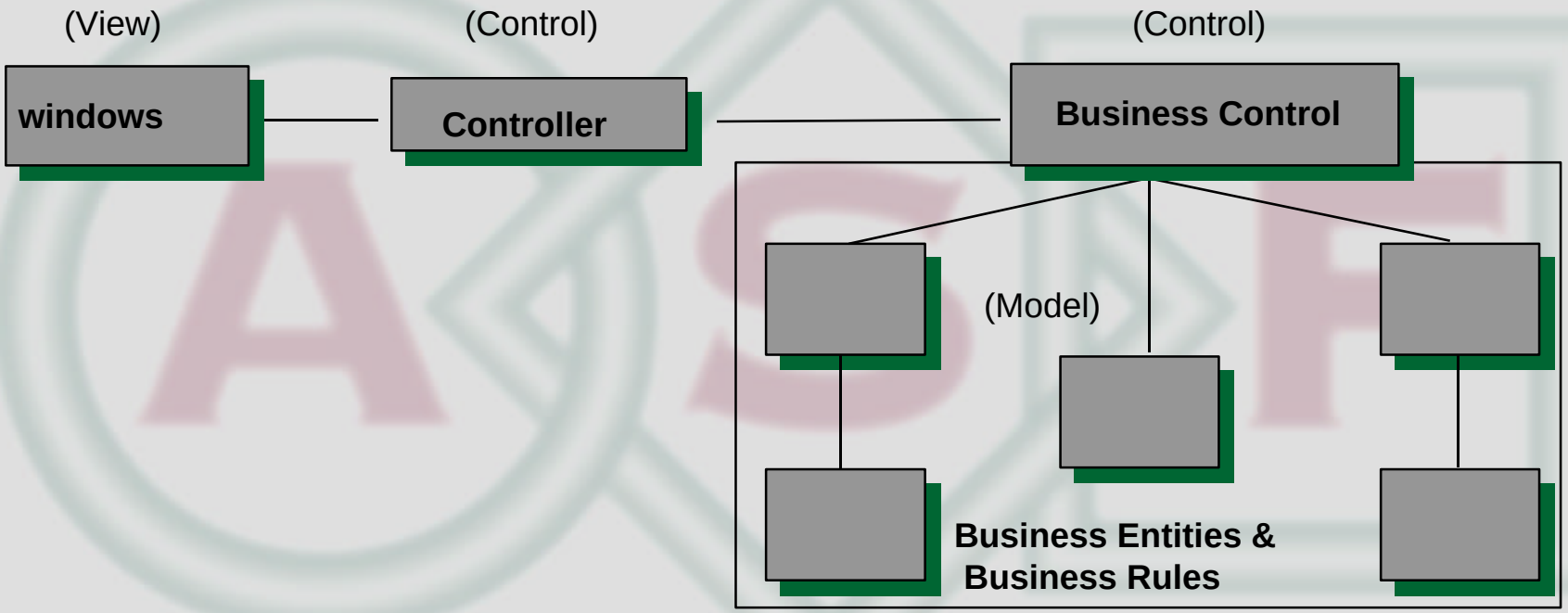
Model-View-Controller Pattern



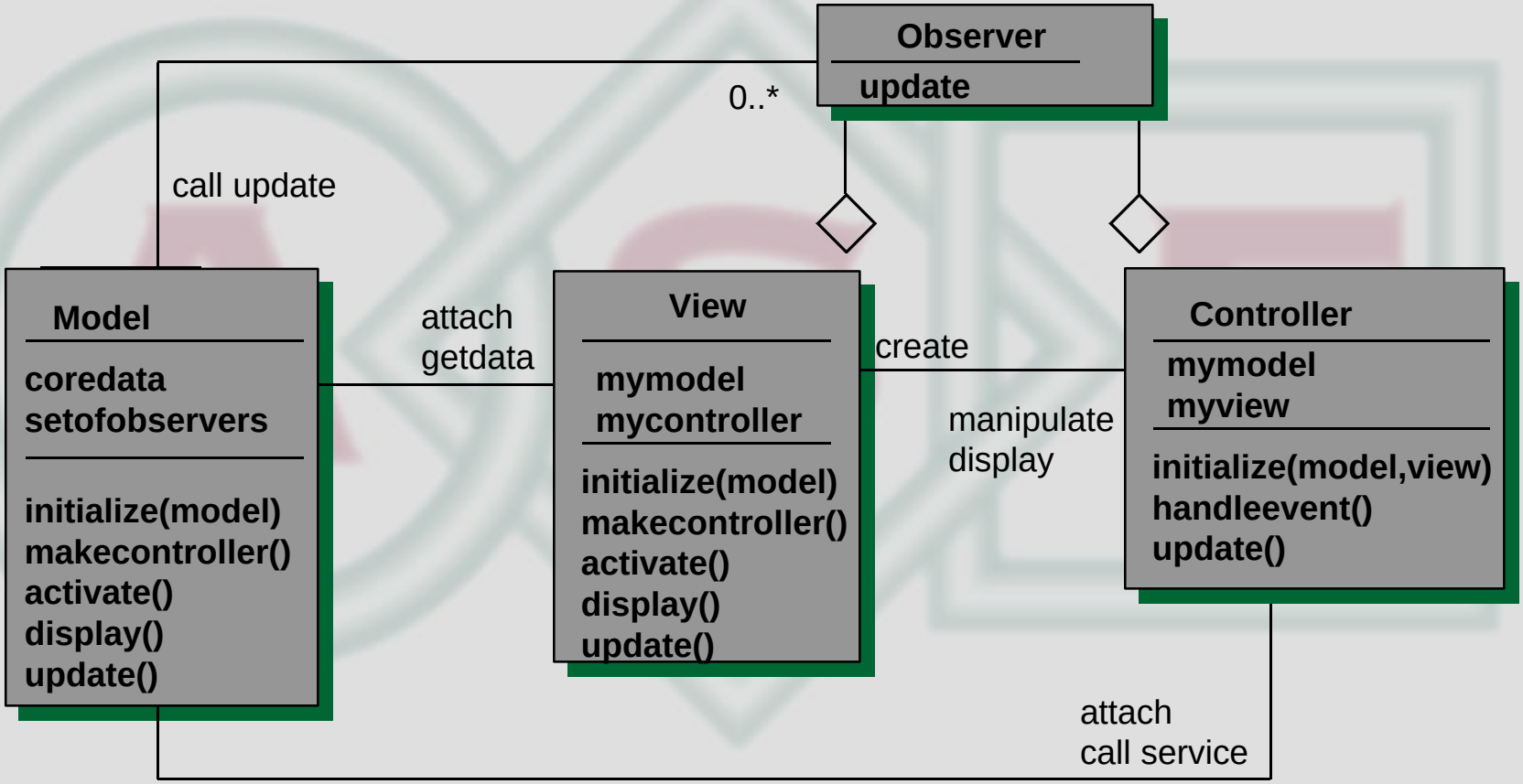
Topics

- Architecture Overview
- Pattern Overview
- Visual Patterns
- Domain Patterns
- Design Patterns
- **Applying Patterns to Web Development**

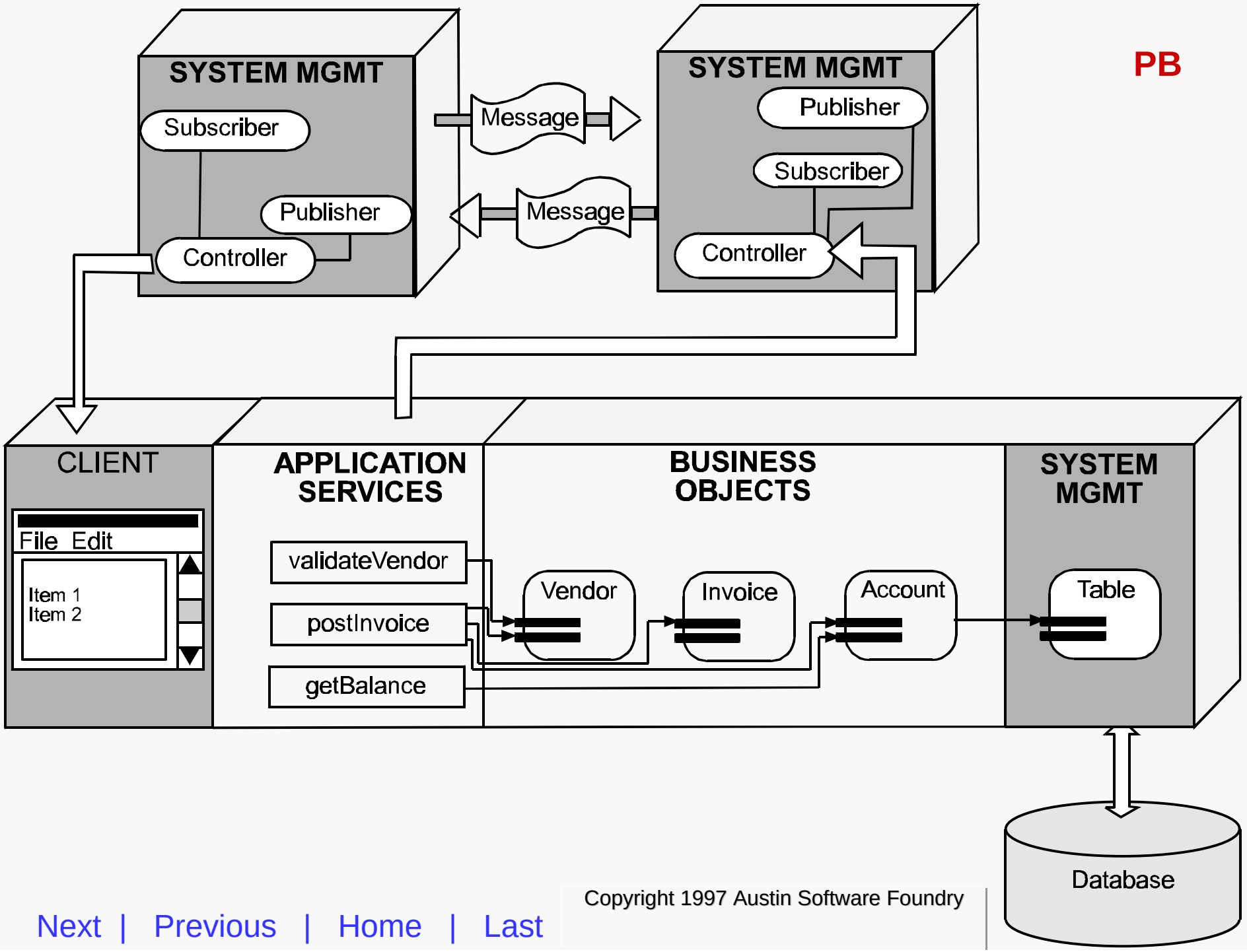
M-V-C + Use Case Pattern

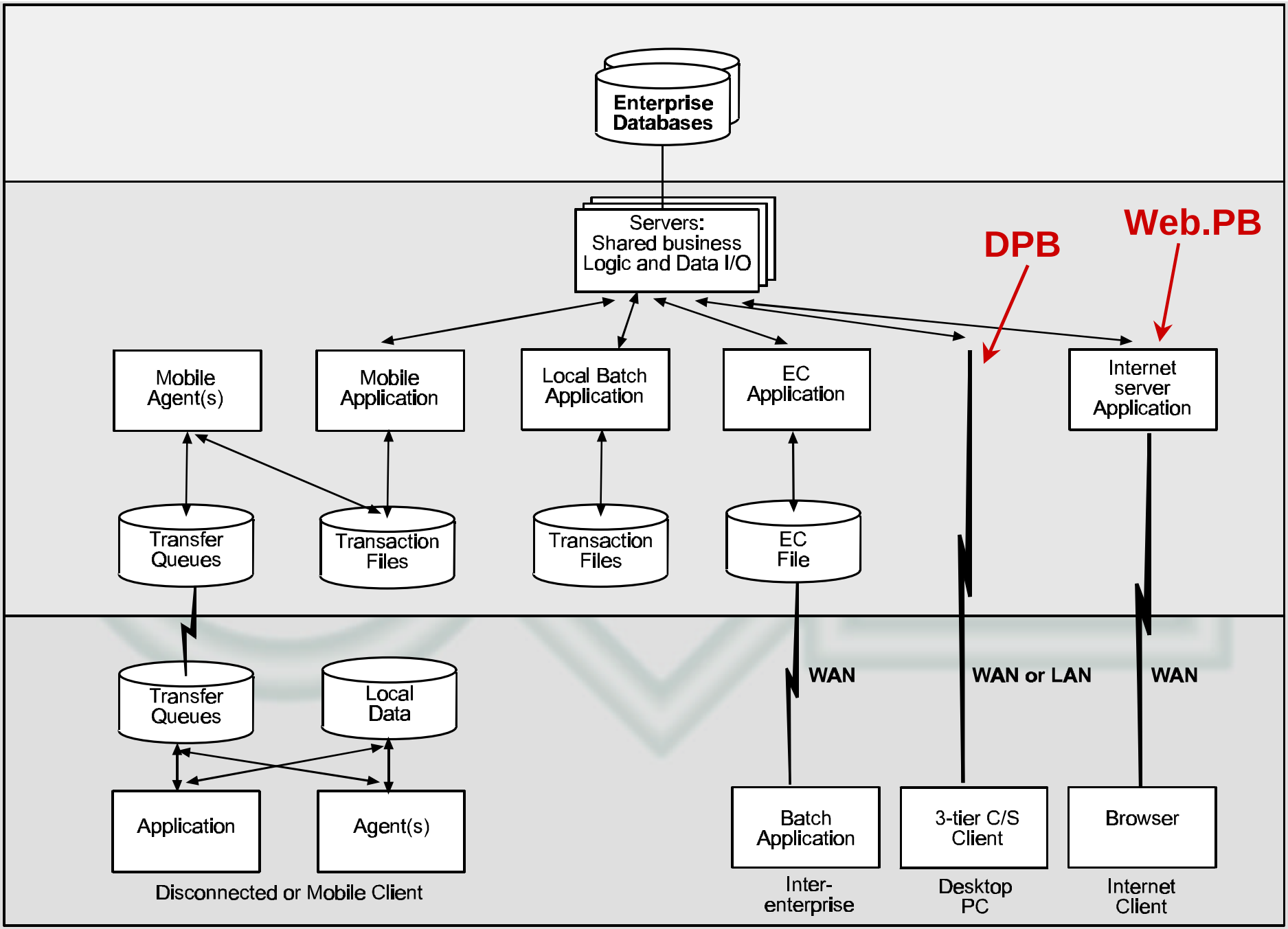


M-V-C + Observer Pattern

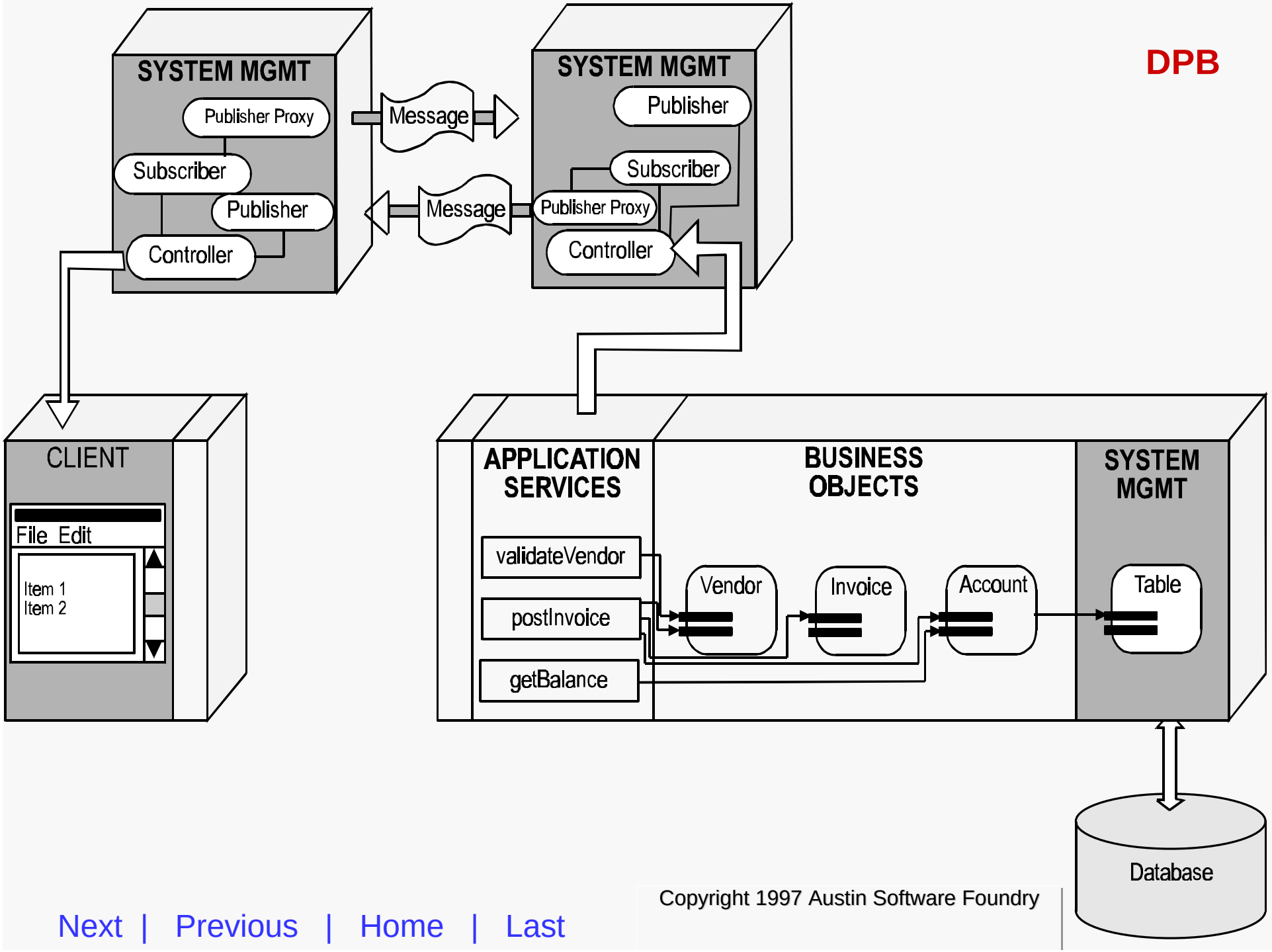


PB

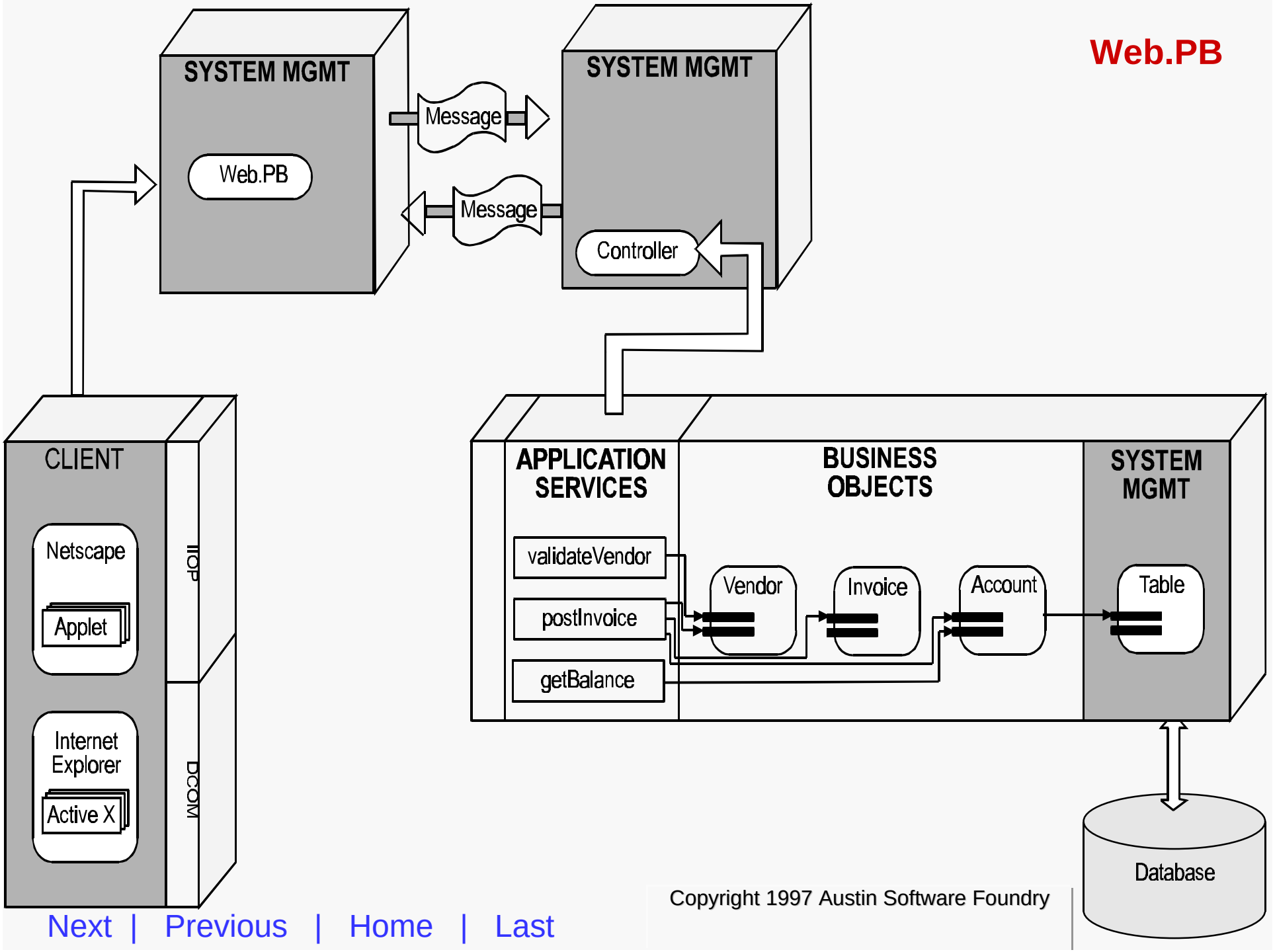




DPB



Web.PB



Questions?