

**Magenic**<sup>SM</sup>



# Design Highly Scalable Distributed Applications with .NET

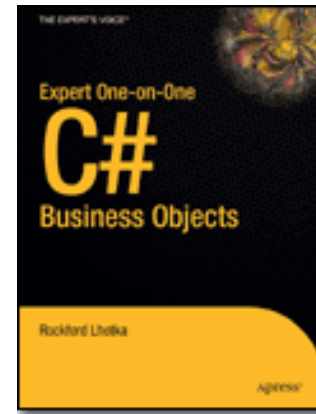
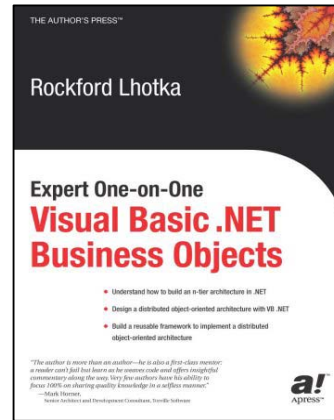
**Rockford Lhotka**

Principal Technology Evangelist  
Magenic Technologies

**Magenic**<sup>SM</sup>



# Rockford Lhotka



***MSDN Online***

***Visual Studio Magazine***

***.NET Magazine***

# Agenda

- **Distributed Objects overview**
- **N-tier distributed architecture**
- **Business object construction**
- **Data service construction**

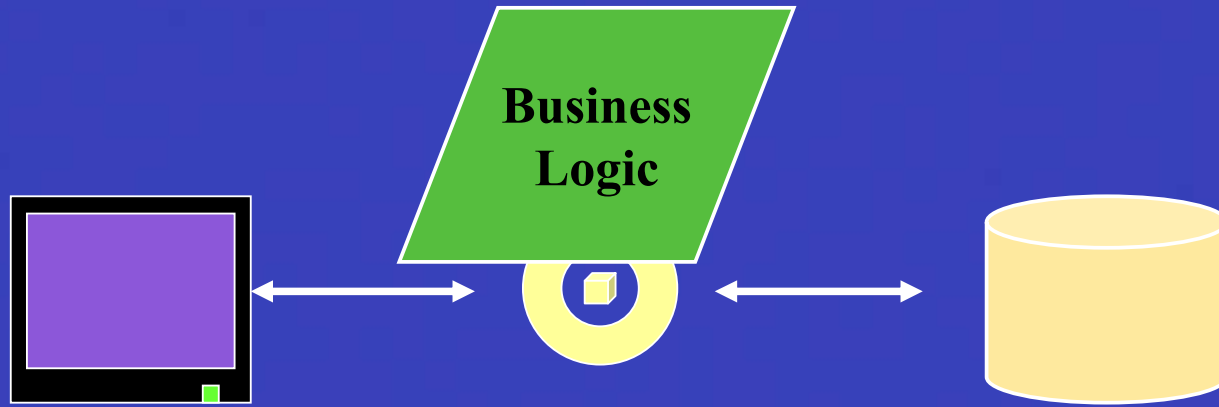
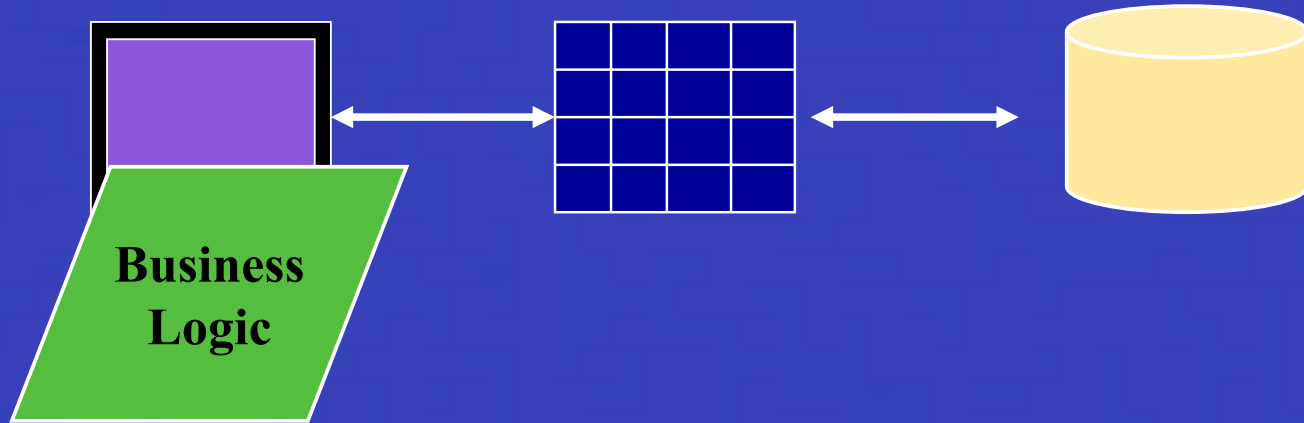
# Distributed Objects Overview

- Distributed development models
- Distributing objects vs data
- Reconciling services

# Development Models

- **Single-memory**
  - Procedural
  - Object-oriented
  - Data-centric
  
- **Multiple-memory (distributed)**
  - Distributed data
  - Distributed object
  - Message-based (service-oriented)

# Data vs Objects

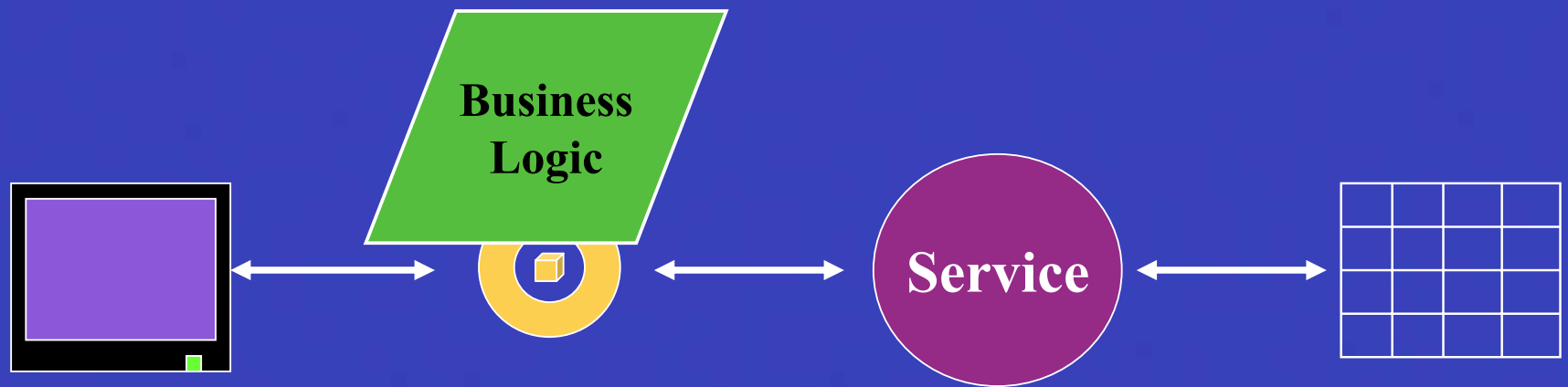


# Services

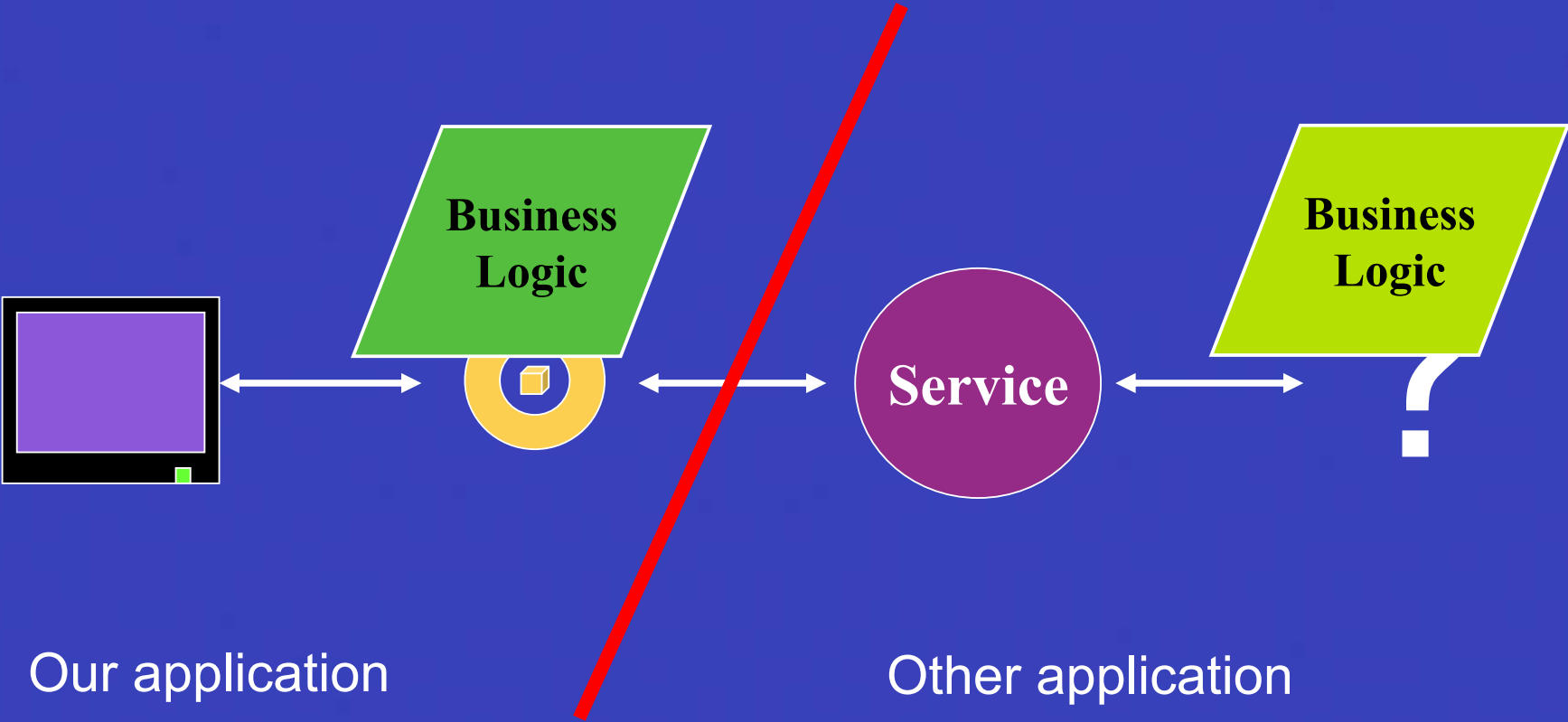
- **Internal services**
  - Service is part of our app
  - Service trusts our app
- **External services**
  - Service is part of another app
  - Service does not trust our app

**It's all about TRUST**

# Internal Services



# External Services



# Why use Objects?

- **Help us model the real world**
  - **Easier to code**
  - **Easier to maintain**
  - **Easier to relate to end user**
- **Model each concept once**
  - **Reusable code**
- **Support multiple UIs**

# UI Options

- **Windows Forms**
- **Web Forms**
- **XML Web Services**
- **All are equal, as all build on the same objects**
  - **Share identical data and logic**

# So What are Distributed Objects?

- **Objects that move around the network**
  - **Data *and* logic move to the machine where they are needed**
    - **Client for UI interaction**
    - **Server for data interaction**
- **Sometimes called ‘agents’**

**The best of both worlds**

# N-tier Distributed Architecture

# Logical Architecture

**Presentation**

**UI**

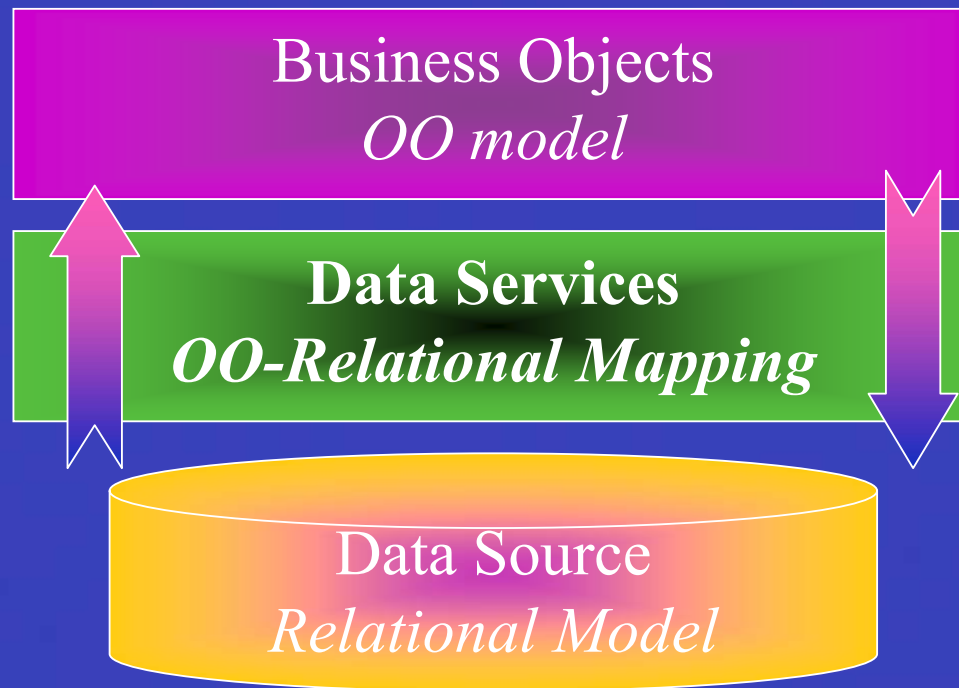
**Business Logic**

**Business Logic**

**Data Access**

**Data Storage and Management**

# OO-Relational Mapping



# Revised Logical Architecture

**Presentation**

**UI**

**Business Objects**

**Data Services**

**Data Storage and Management**

# Web App

**Browser (Presentation)**

**HTTP**

**ASP.NET (UI)**  
**Business Objects**  
**Data Services**

**Database (SQL Server, etc)**

# Web App (DMZ)

**Browser (Presentation)**

HTTP

**ASP.NET (UI)  
Business Objects**

Remoting

**Data Services**

**Database (SQL Server, etc)**

# Windows App

**Windows Forms  
Business Objects**

Remoting

**Data Services**

**Database (SQL Server, etc)**

# 2-tier Windows App

**Windows Forms  
Business Objects  
Data Services**

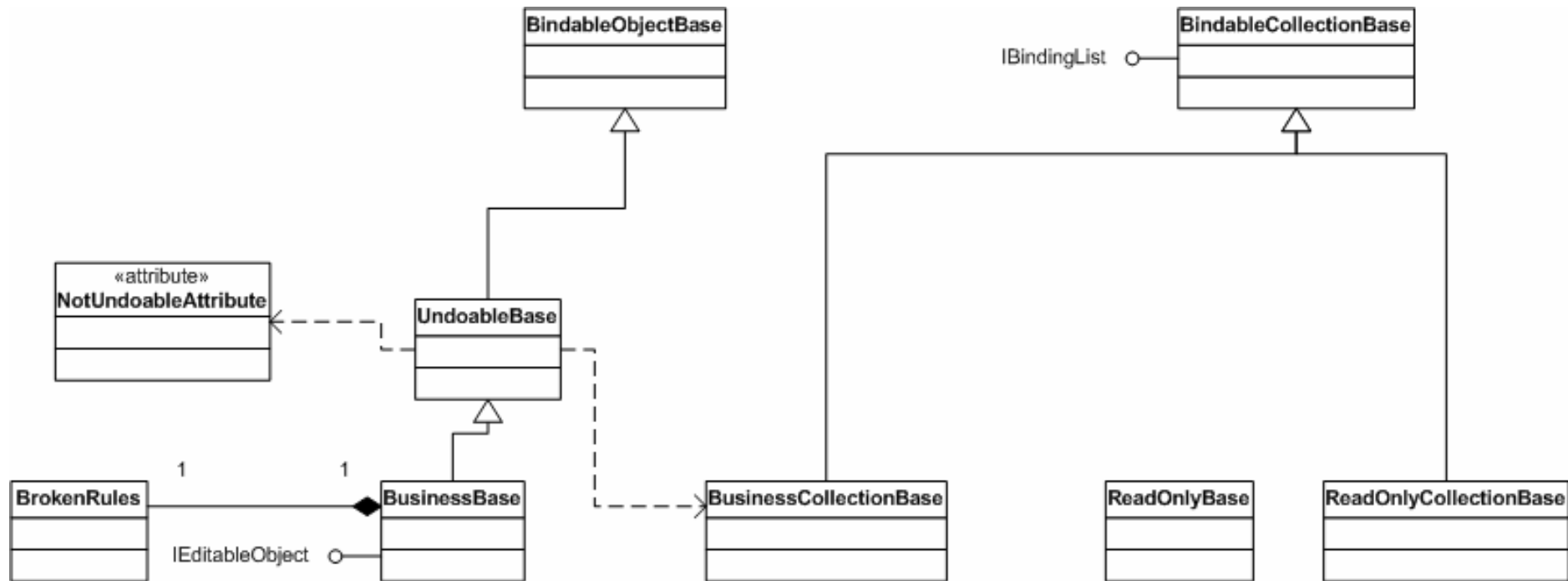
**Database (SQL Server, etc)**

# Business Object Construction

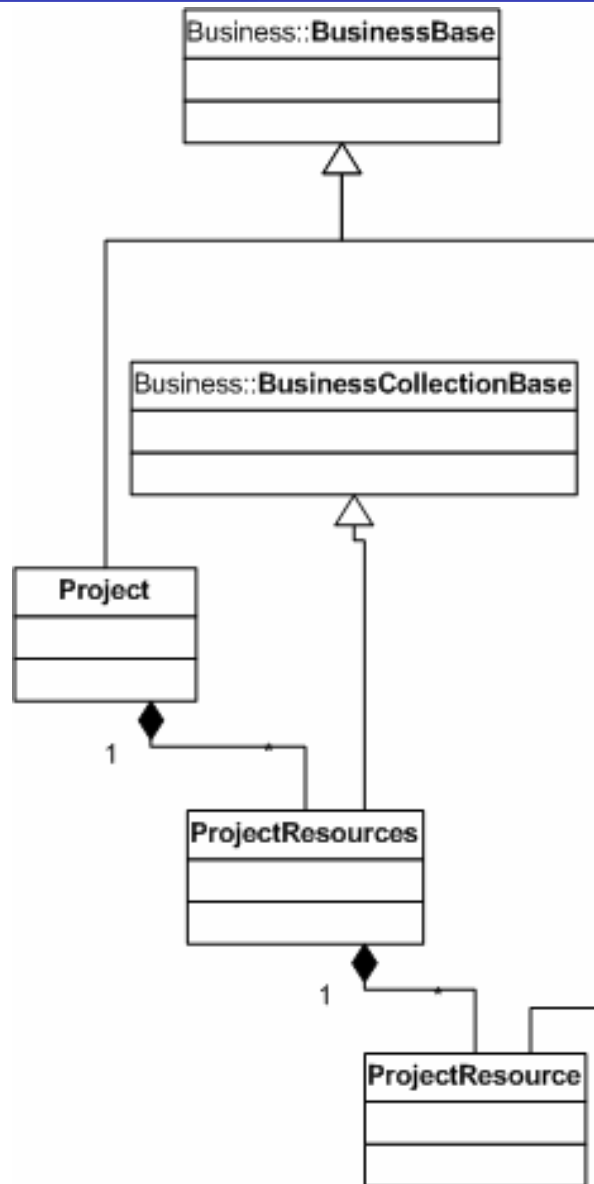
# Goals

- **High level of abstraction**
  - **Reusable, maintainable components**
- **Encapsulate all data and logic**
  - **Keep data access and business logic out of the UI**
- **N-level undo**
  - **Support rich Windows and Web UIs**

# Business Object Framework



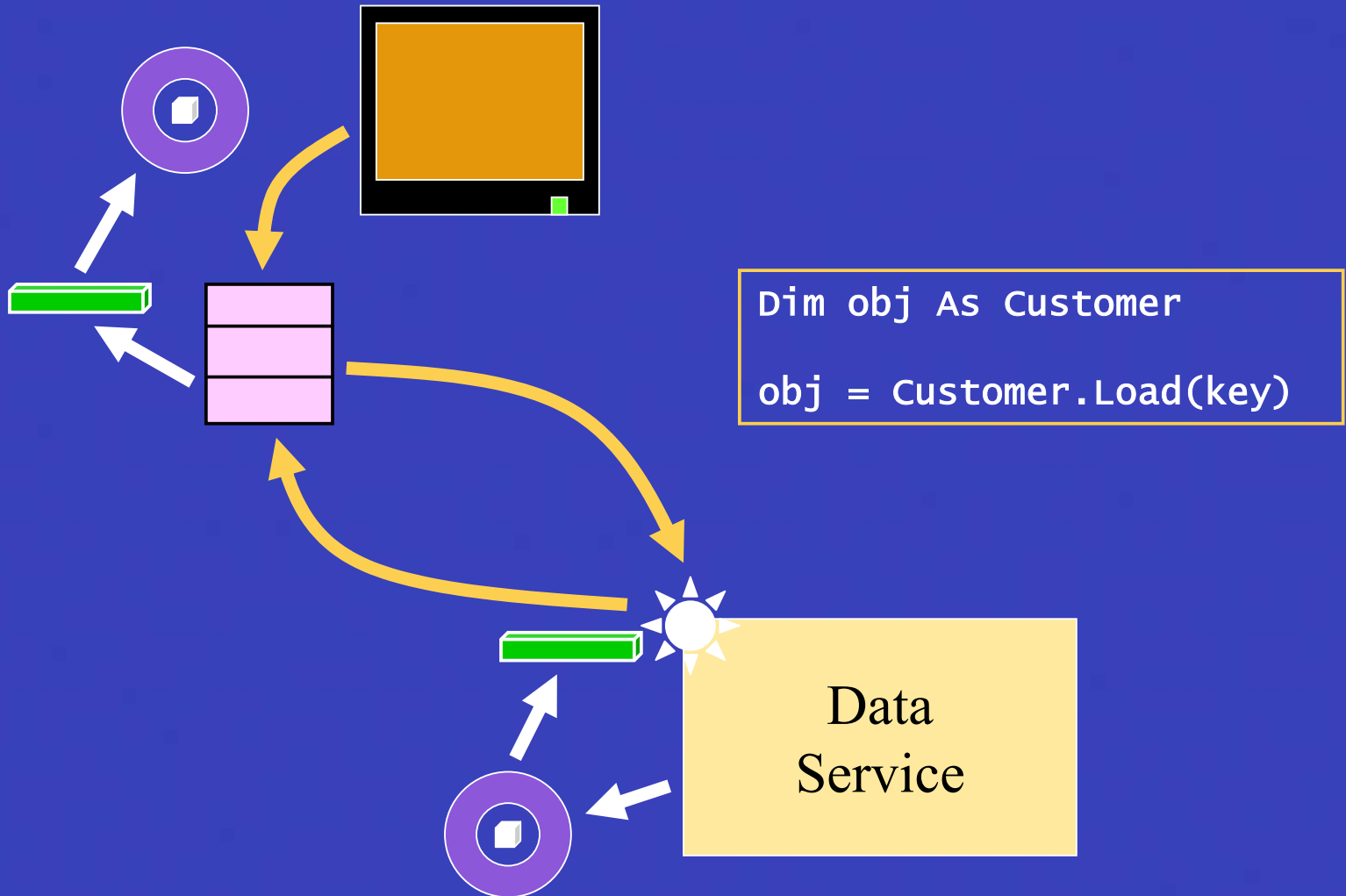
# Project Example



# Automatic Gains

- **N-level undo**
  - **BeginEdit, CancelEdit, ApplyEdit**
- **Status tracking**
  - **IsNew, IsDirty, IsDeleted**
  - **MarkNew, MarkDirty, Delete**
- **Business rule tracking**
  - **IsValid, BrokenRules**

# “Class in Charge”

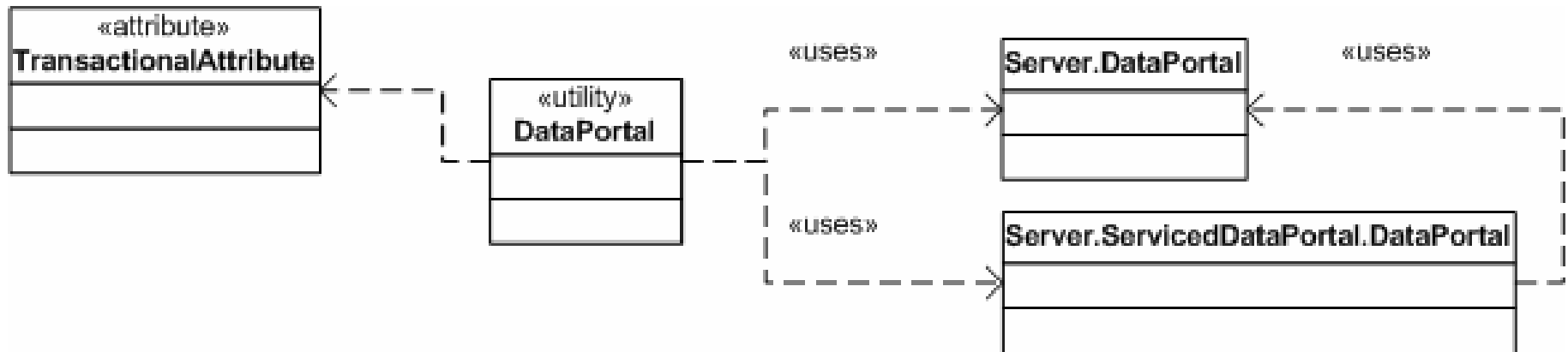


# “Class in Charge”

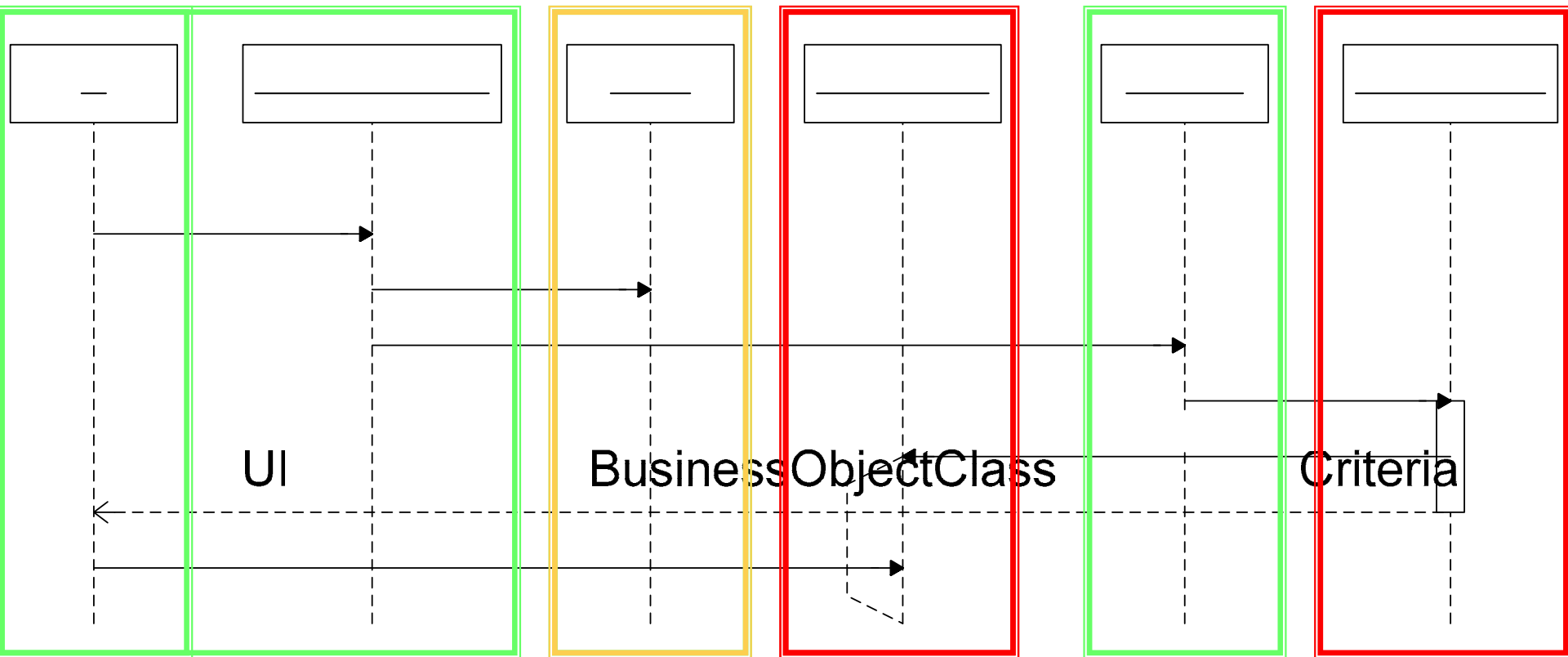
- Directly use .NET serialization
- Simpler UI code
- Objects are ‘truly’ distributed
- Easier reuse of objects
- More abstract model
- Objects encapsulate *all* business logic

# Data Service Construction

# DataPortal Framework



# Data Retrieval



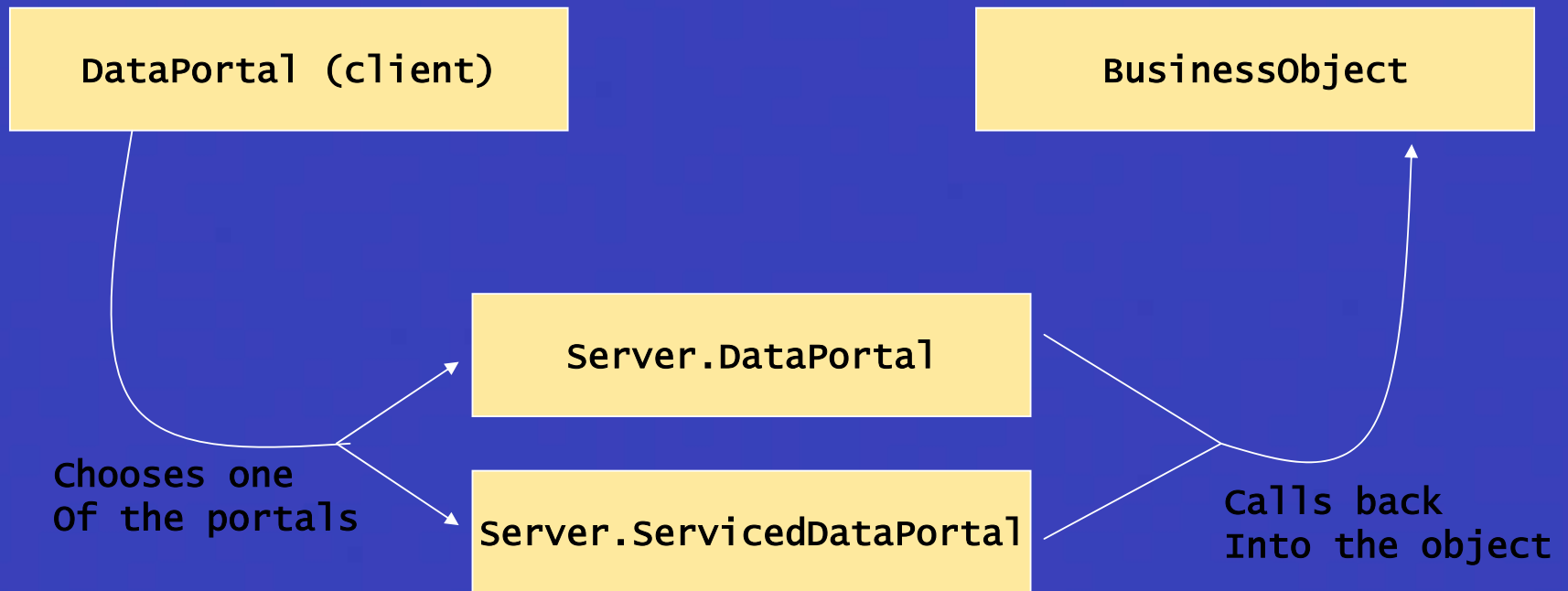
GetBusinessObject

New

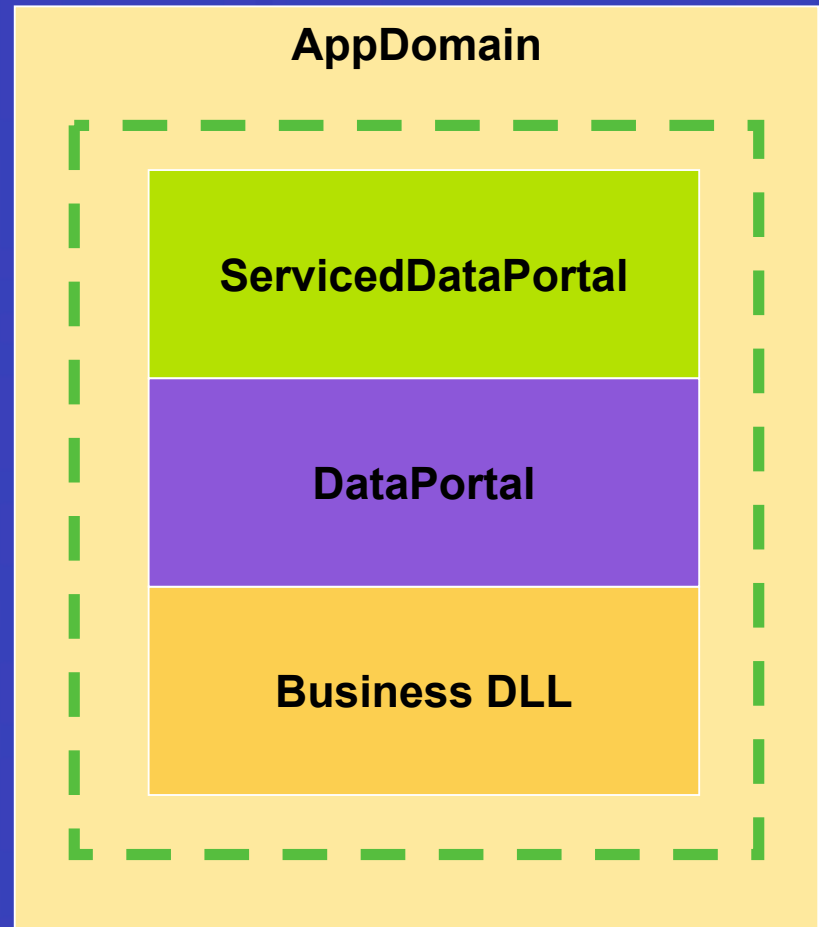
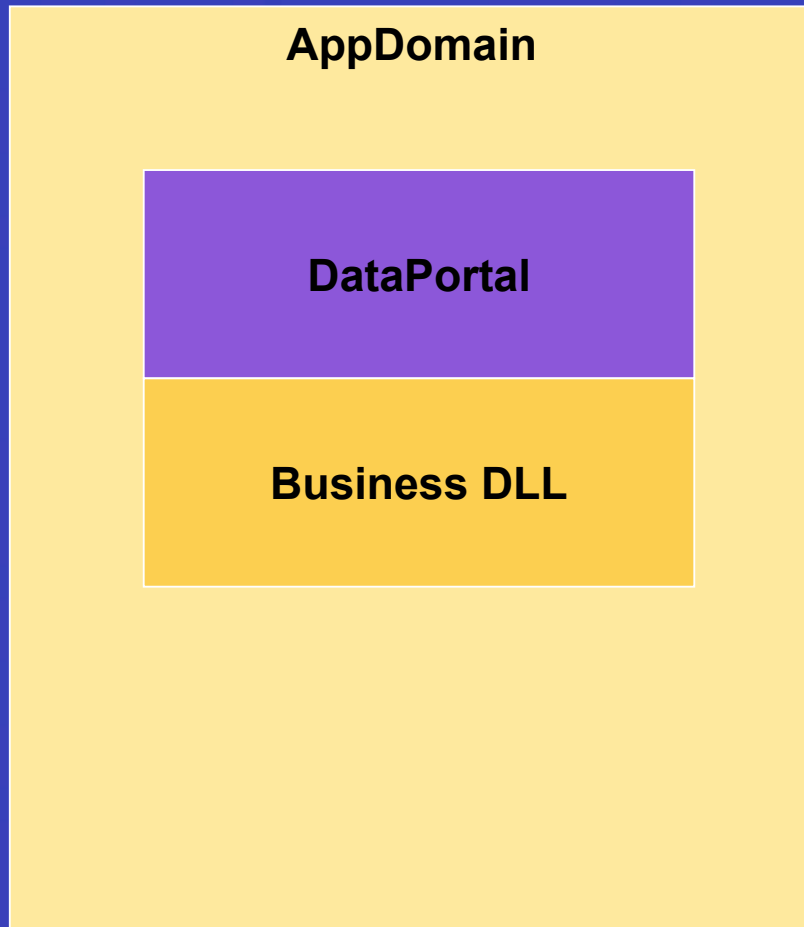
# The Great Part?

- If the DataPortal is designed right you don't need to worry about any of these details.
- All data access code goes in the object
  - DataPortal\_Fetch
  - DataPortal\_Update
  - DataPortal\_Delete
  - DataPortal\_Create

# DataPortal Design



# Making this work with COM+



# How does the object move?

- **Serialization**
- **Remoting**

# Xml vs 'Real' Serialization

- **XmlSerializer**

- Copies values of read-write properties from an object
- XML Web Services

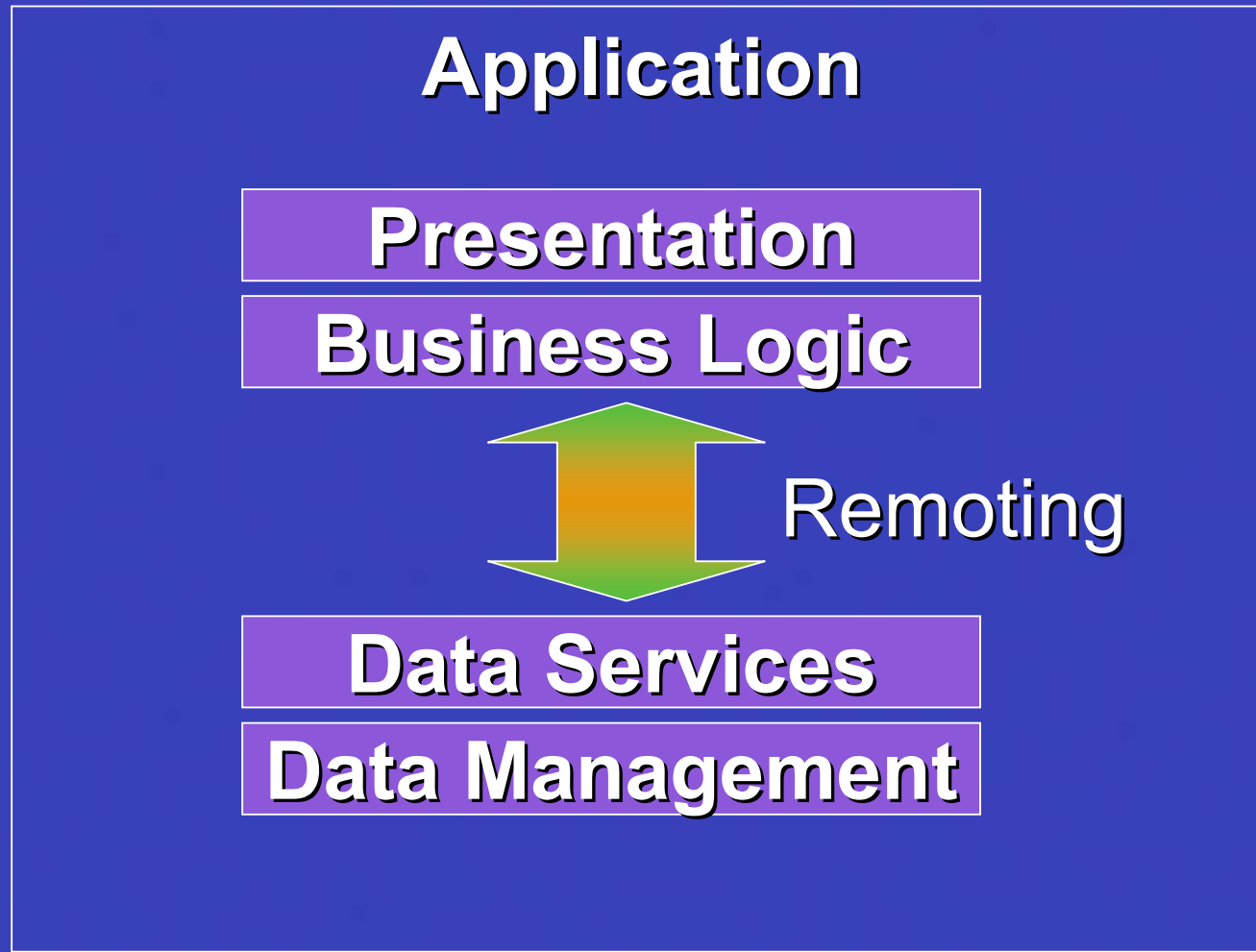
- **BinaryFormatter/SoapFormatter**

- Copies actual variable values from *within* object
- .NET Remoting

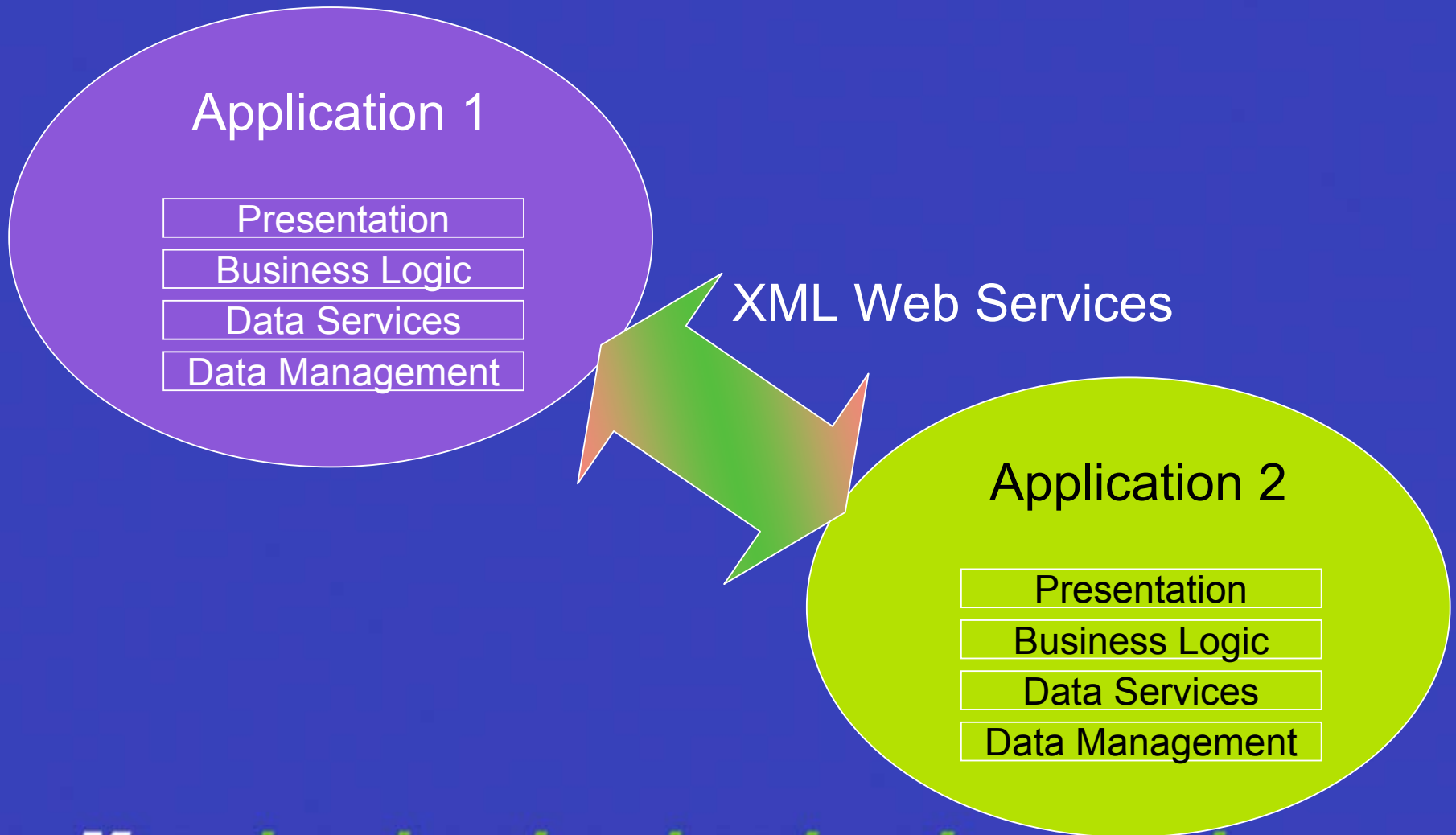
# Web Services vs Remoting

	Web Svc	Remote
Data-centric	●	●
Object-oriented		●
XML/SOAP	●	●
Binary		●
Pass thru firewall	●	●
IDE support	●	
Code controlled		●
Peer to peer		●
Events/callbacks		●
Stateful objects		●

# Where does Remoting fit?



# Where do Web services fit?



# Summary

- **Distributed objects**
  - Provide the power and maintainability of OO
  - Leverage intelligent data rather than raw data
  - Support multiple UI types with the same logic
  - Gain high performance through use of services
  - Effectively use all computing power available

**Magenic**<sup>SM</sup>  
*Nobody's more serious about Microsoft.*

ineta

**Thank you!**

**Rockford Lhotka**  
**rocky@lhotka.net**

**www.lhotka.net**

**www.magenic.com**