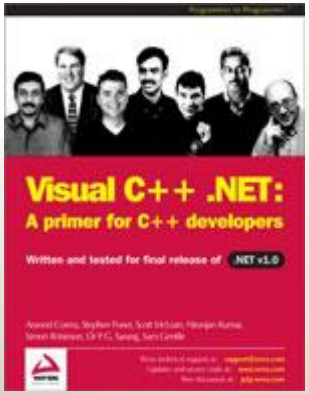
Advanced WCF: Asynchronous Messaging and Event Driven Architectures

Sam Gentile
SOA Practice Lead
Neudesic, LLC

About Sam Gentile

- SOA Practice Lead at Neudesic, Principal Consultant II,
- INETA Speaker, Connected Systems MVP
- SOA and WCF Guy
  - Led successful $2M SOA at Algorithmics
  - 4 Years on Indigo SDR at Microsoft
- Internationally known .NET guy
  - Working with CLR since 1999!
  - .NET products and expertise for Algorithmics, Adesso,
  - Microsoft, Groove, NaviSite, BCGI, others
  - Winner 2003 Jolt Award – Groove Toolkit for VS.NET
  - Patent Pending – Distributed Data Systems
- MSDN Columnist "Beyond Add Reference", "Using the .NET Framework SDK Interop Tools"
- Book Author
  - Co-author Visual C++.NET (2002)
- Major .NET/CLR Blogger ("Heart and Soul of .NET Community") at http://samgentile.com/blogs/samgentile/
- NET Portal at http://www.samgentile.com/
- **WE HAVE POSITIONS OPEN! See me after!**



About Neudesic

- Microsoft NSI, Managed, Gold Partner
- National Presence, Headquartered in SoCal
- Over 300 Clients in EPG and SMS&P
- 5 practices focused on technology
  - .Net Custom Development (Visual Studio)
  - Connected Systems (WCF, BizTalk, Neuron)
  - SQL/Business Intelligence (SQL Server)
  - IW, Portals and Collaboration (Office/SharePoint)
  - Dynamics (GP and CRM)
- Complete Microsoft Stack Coverage

Agenda

- **Review: SOA**
- Review: WCF
- Fundamentals of Message Exchange Patterns
- Event-Driven Architecture & Publish-Subscribe
- Neuron ESB and Publish-Subscribe

Diagnosis

- Misalignment
  - The world of abstract
  - Heavy Process Focus
- Silos and Monoliths
  - One world, one system
  - “Fear”
- Misguided or non-existent investment strategies
  - Businesses focus on capability. Does IT?

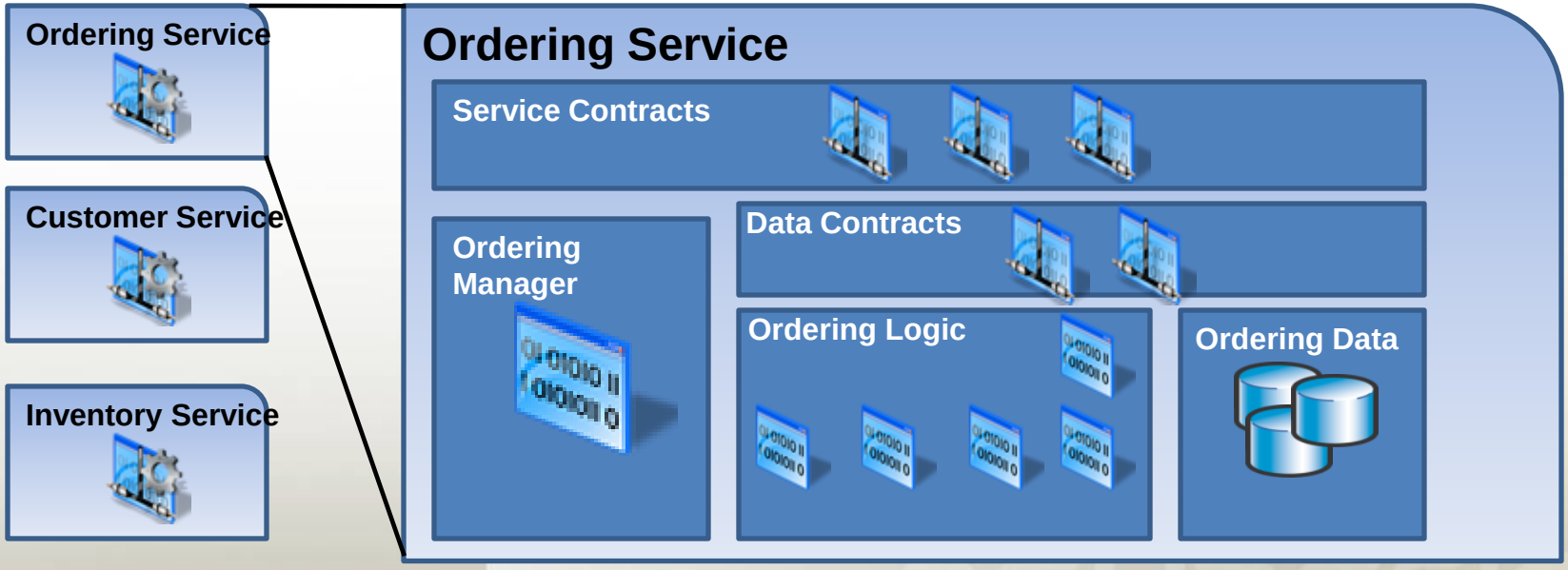
IT must become more dynamic to support the business in an agile fashion.

Service Oriented Architecture

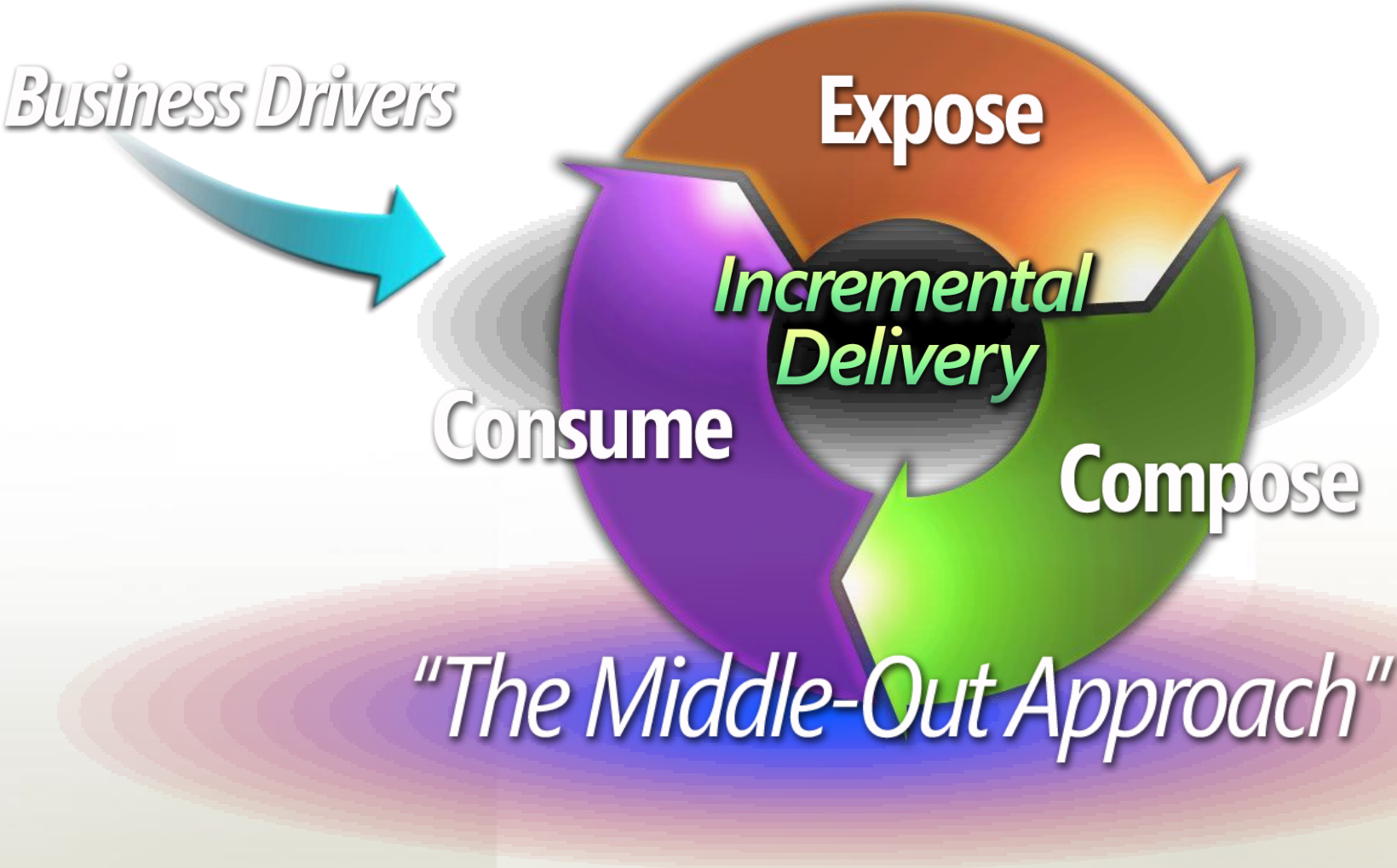
- Key Concepts:
  - Distributed systems are composed of autonomous services
  - Loosely coupled services make few assumptions about each other
  - Platform independent communication through open standards
  - Enterprise-strength communication, security, reliability, transactions
- Supporting Technologies:
  - SOAP and WS-* Standards (ongoing, composable family of standards)
  - Windows Communication Foundation (WCF) and Workflow (WF)
- Benefits:
  - Platform independence: no vendor lock-in
  - Incremental upgrades: change programs without disrupting others
  - Highly scalable: easy scale-out through routers/farms
  - Services can be locale and time independent of each other
  - Maximizes reuse: services are easily repurposed for other uses
  - Greater longevity: tolerant fabric of autonomous services is long-lasting
- Service Orientation vs. Service Oriented Architecture (SOA)
  - What is the architecture?

What is a Service?

- "…a software component of distinctive functional meaning the typically encapsulates a high-level business concept." – *Enterprise SOA: Service-Oriented Architecture Best Practices*



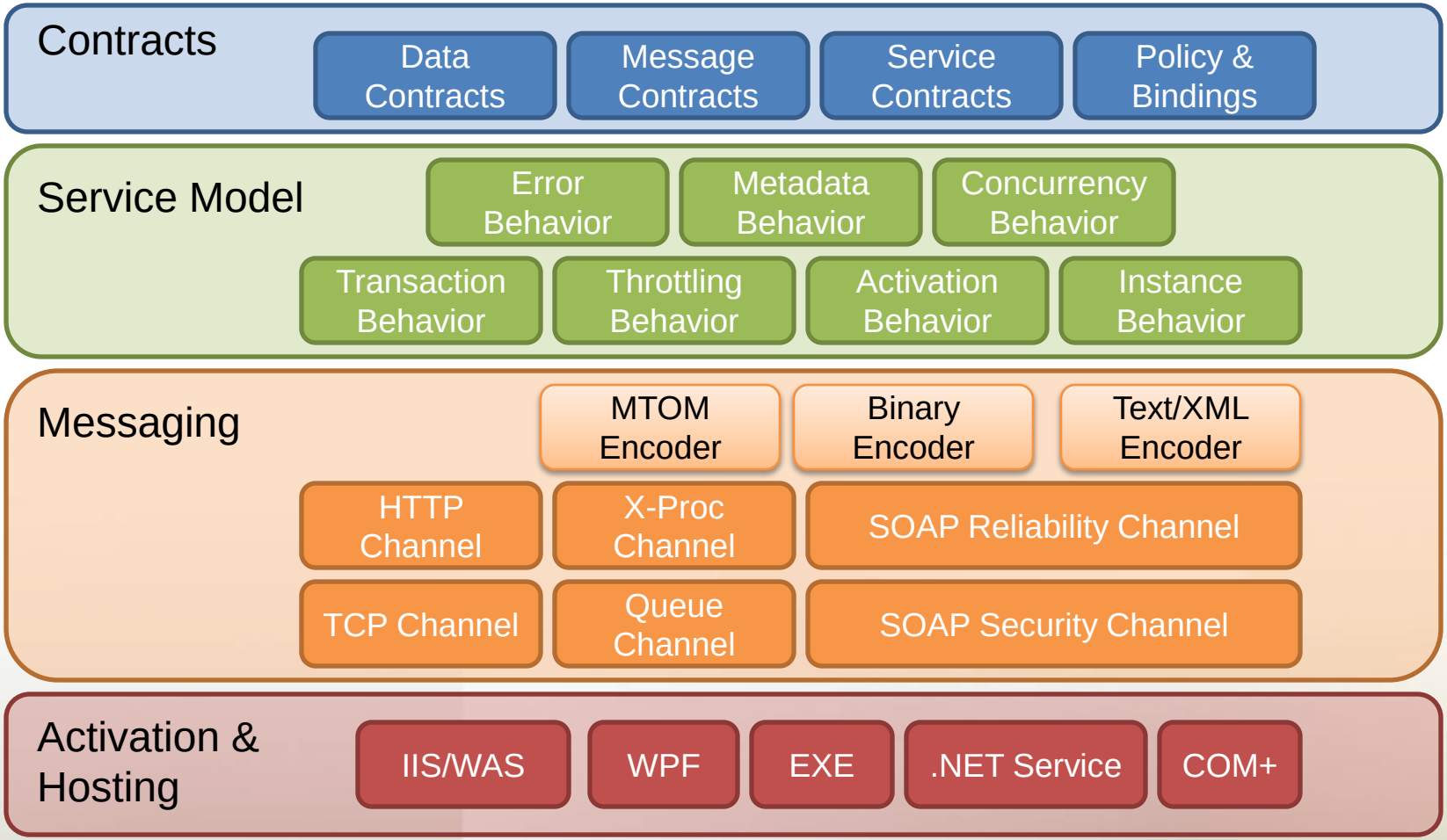
Real World SOA



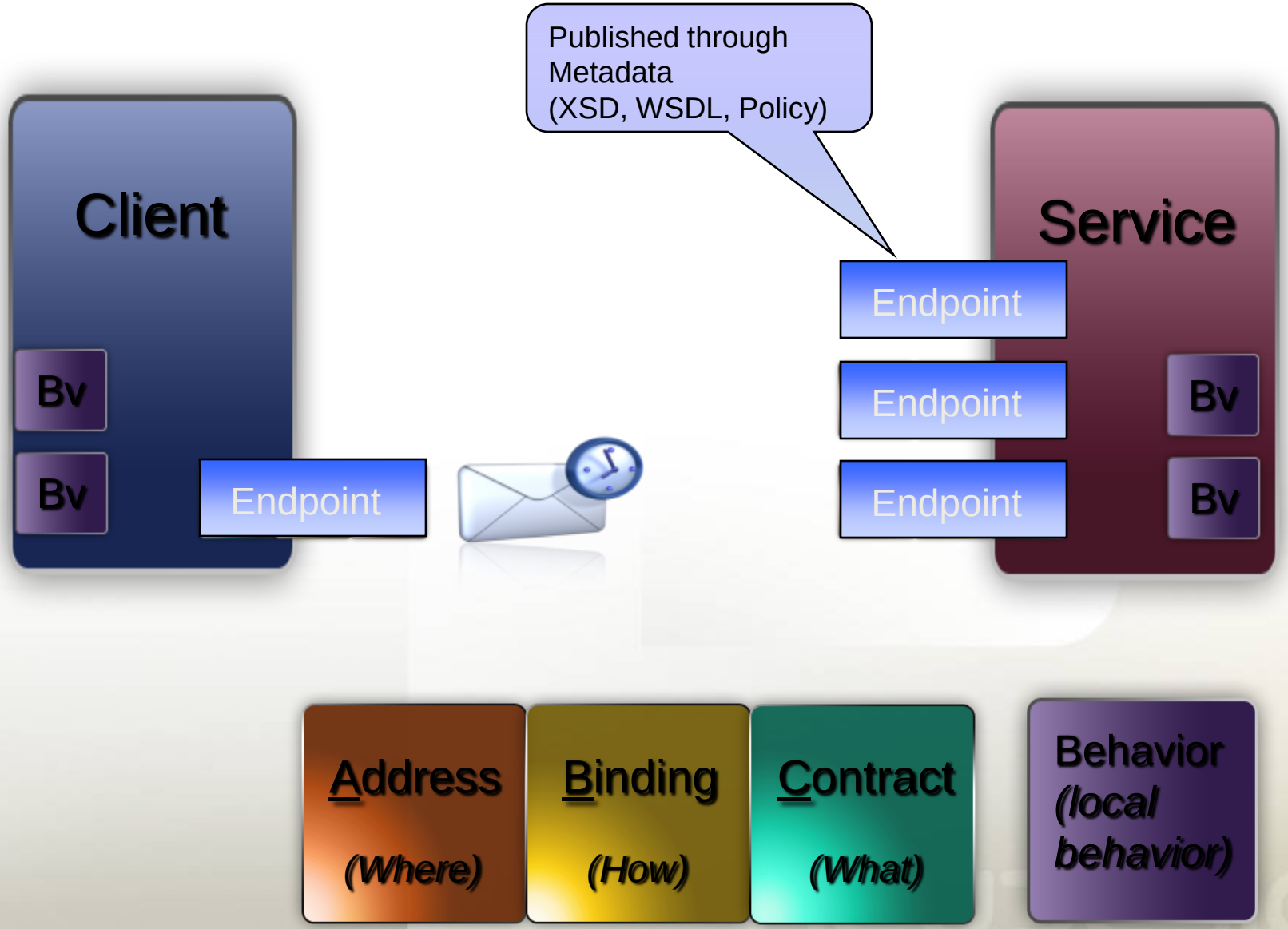
Agenda

- Review: SOA
- **Review: WCF**
- Fundamentals of Message Exchange Patterns
- Event-Driven Architecture & Publish-Subscribe
- Neuron ESB and Publish-Subscribe

WCF Architecture



The WCF ABCs


Published through
Metadata
(XSD, WSDL, Policy)
Client
Bv
Bv
Endpoint
Service
Endpoint
Endpoint
Endpoint
Bv
Bv
Address
(Where)
Binding
(How)
Contract
(What)
Behavior
(local
behavior)

Contracts

- Service Contracts
  - What Operations and MEPs a Service exposes
- Data Contracts
  - Defines data types that are passed
  - Map CLR types to XSD and back
- Fault Contracts
  - Don’t bleed CLR Exceptions out of service boundary!
- Message Contracts
  - Finer control over message body
  - Can also be used to compose MEPs based on Data Cotracts

Bindings

- Specify for communication
  - Transport (TCP, HTTP, Named Pipes, Custom)
  - Wire-level Protocol
  - Encoding
- WCF makes it easy with standard bindings
  - Combination from possible multitude of choices
- You can create custom bindings too
- Some bindings also specify Security, Reliability, Transactions

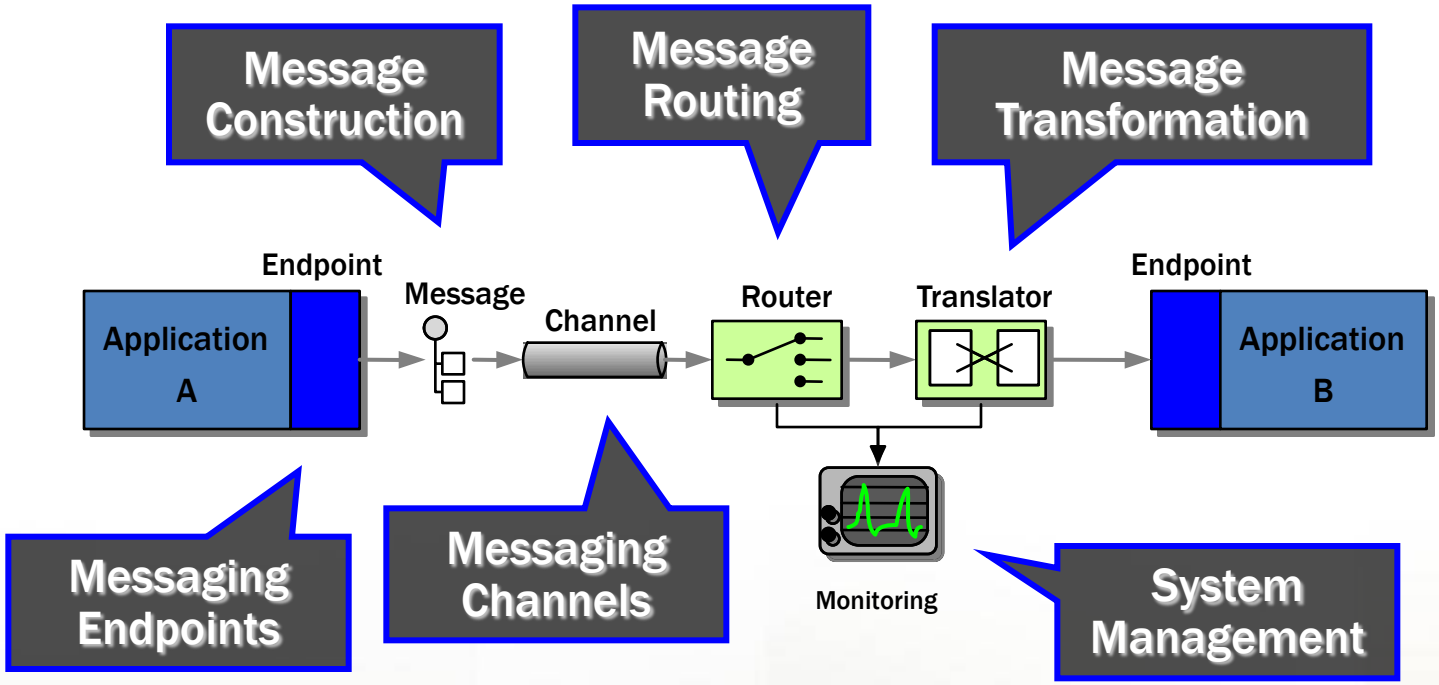
Agenda

- Review: SOA
- Review: WCF
- **Fundamentals of Message Exchange Patterns**
- Event-Driven Architecture & Publish-Subscribe
- Neuron ESB and Publish-Subscribe

Messaging

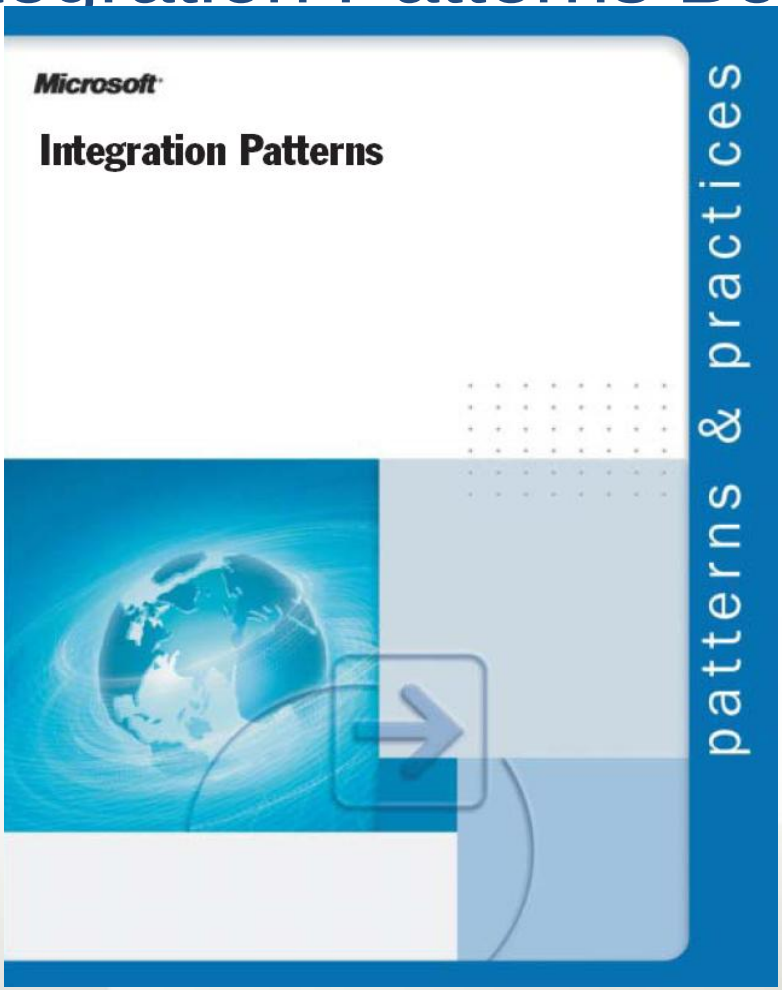
- We still need to get away from RPC and do Messaging [Hohpe04]
  - It's **messages** that travel b/w parties
- Systems communicate via Channels [Hohpe04]
- Messages grouped into Message Exchange Patterns (MEPs)
  - "a message exchange pattern is a template that establishes a pattern for the exchange of messages between two communicating parties." W3C http://www.w3.org/2002/ws/cg/2/07/meps.html
- Messages can be synchronous or asynchronous
- MEPs cataloged in [Hohpe04] and his later PAG Integration Patterns

Enterprise Integration Patterns



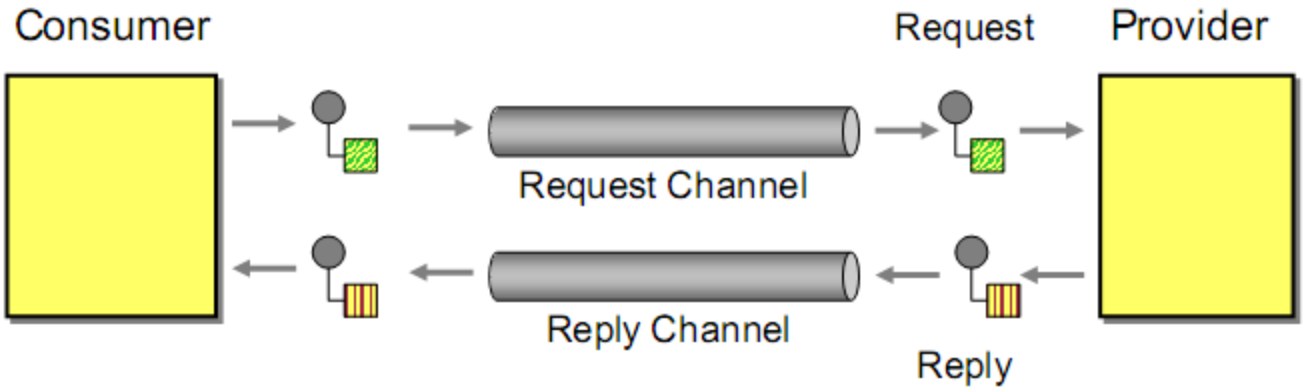
- Hohpe and Woolf 2004 defined Pattern Language of 65 Patterns

Integration Patterns Book



Available to download from
http://msdn.microsoft.com/patterns

Request-Reply MEP



- [Hohpe Page 154] is most commonly used
  - Send pair of Request-Reply messages on separate channels
- Default in WCF, can be implemented via
  - Parameter list
  - DataContract
  - MessageContract
  - Message (raw Message class)

Demo!

Request-Reply MEP

Describing Messages

- Untyped ("universal")

```
void Chat(Message m)
{
}
```

- Typed Message

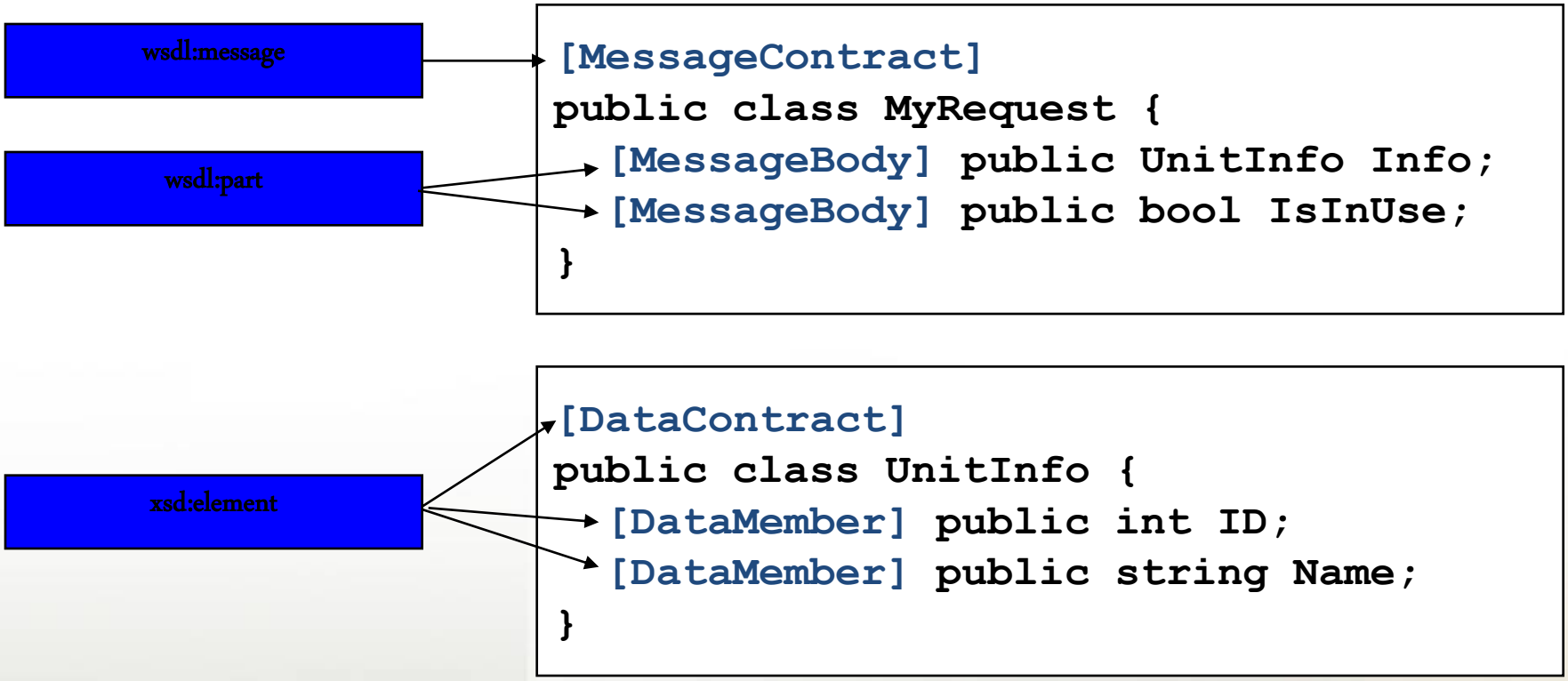
```
void Chat(ChatRequest m)
{
}
```

```
[MessageContract]
public class ChatRequest {
   [MessageHeader] public string Me;
   [MessageBody]   public string Text;
}
```

- Parameters: shorthand for [MC]

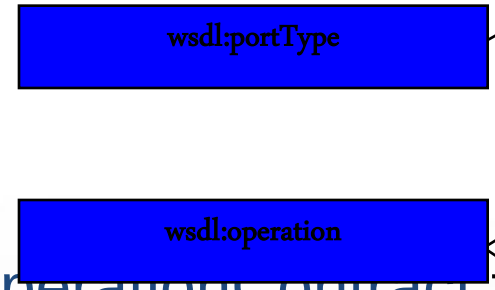
```
void Chat(string name, string text)
{
}
```

Describing Messages



Describing Message Exchanges

- ServiceContract ties together multiple operations
- OperationContract ties together Message Contract, Action




```
[ServiceContract]
public interface MyContract {
  [OperationContract(
     Action="urn:DoIt",
     ReplyAction="urn:Done")]
  MyReply DoIt(MyRequest request);

  [OperationContract(
     Action="urn:DoItAgain",
     ReplyAction="urn:DoneAgain")]
  Message DoIt(Message request);
}
```

Describing Message Exchanges

- Request/Reply

```
[OperationContract]
string Echo(string text);
```

- One-way Message

```
[OperationContract(IsOneWay = true)]
void Chat(string text);
```

- Duplex Contracts

```
[ServiceContract(CallbackContract =
  typeof(IChat)]
public interface IChat {
[OperationContract(IsOneWay = true)]
void Chat(string text);
}
```

- Actions control Dispatch

```
[OperationContract(
   Action = "Foo",
   ReplyAction = "FooResponse")]
Message Foo(Message request);
```

- "*" matches all actions

```
[OperationContract(Action = "*")]
void Dispatch(Message request);
```

Demo!

One Way Message and Message Class

Request/Reply and Asynchrony

- One-way messages modeled using OneWay=true operation contracts

```
[OperationContract(IsOneWay = true)]
void Chat(Message request);
```

- Correlated Request/Reply modeled either as synchronous method

```
[OperationContract]
Message Chat(Message request);
```

- or using IAsyncResult pattern

```
[OperationContract(AsyncPattern=true)]
IAsyncResult BeginChat(Message request,
                       AsyncCallback cb, object state);

Message EndChat(IAsyncResult call);
```

Agenda

- Review: SOA
- Review: WCF
- Fundamentals of Message Exchange Patterns
- **Event-Driven Architecture & Publish-Subscribe**
- Neuron ESB and Publish-Subscribe

Event-Driven Architecture

- Broadcast Communications
- Timeliness – systems publish events as occur
- Asynchrony – no wait before publish
- Fine-Grained Events – individual events
- Ontology – hierarchy of events
- Complex Events Processing – system understands & monitors event relationships

[Thanks to Greghor Hohpe “Programming Without a Call Stack”

Event-Driven Architecture

- Loosely-coupled communication pattern
- Better alignment of technical design and implementation with business model
- Business people talk in events
  - When an opportunity closes … do this, etc.
- Systems are connected by Pub/Sub topics that match named business conversations & events
  - Identify a Subject & maintain list of Subscribers
  - Subscribers indicate interest by register/subscribe
  - Publisher notifies interested subscribers with event

Observer and Pub-Sub Patterns

- Observer [Gamma94]
  - Decouple observers from their subject
  - Provide event notification to all interested observers no matter how many
- Publisher-Subscriber Pattern [POSA]
  - Expands on Observer by adding notion of event channel for communicating event notifications
  - Three Refinements
    - List-Based Publish/Subscribe [IntPatterns04]
    - Broadcast-Based Publish/Subscribe [IntPatterns04]
    - Content-Based Publish/Subscribe [IntPatterns04]

List-Based Publish Subscribe

- Identify subject and maintain list of subscribers
  - Upon event, notify each subscriber on list
- Classic way to implement is Observer [Gamma]
  - Interested parties register with Attach()
  - Upon changes, call each with Notify()
- Observer suited for one to many & have created instances
  - Complicated with creating many subjects
- Create many subjects & each contain list of observers
  - Must write relationships to persistent storage between process executions
- Clients provide callback address
- Service calls callback address

List-Based Pub-Sub in WCF

- Use Duplex client channel
- Specify callback contract in service contract
- Built-in support for callback registration
- Reliable Sessions must be turned on

```
[ServiceContract(Namespace="http://Microsoft.ServiceModel.Samples",
SessionMode=SessionMode.Required,
CallbackContract=typeof(ISampleClientContract))]
public interface ISampleContract
{
     [OperationContract(IsOneWay = false, IsInitiating=true)]
     void Subscribe();
     [OperationContract(IsOneWay = false, IsTerminating=true)]
     void Unsubscribe();
     [OperationContract(IsOneWay = true)]
     void PublishPriceChange(string item, double price, double change);
}
```

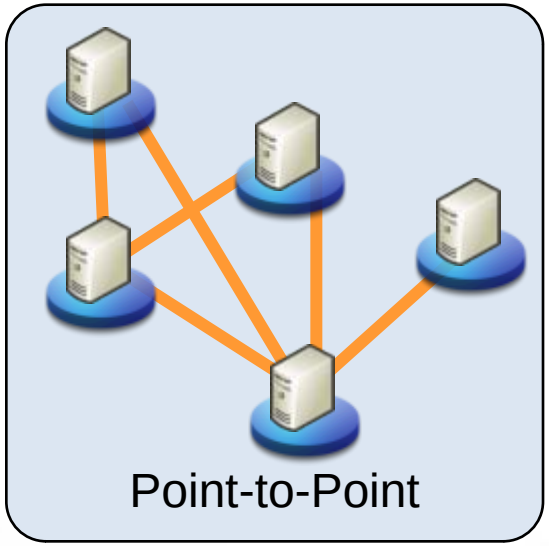
Demo!

List-Based Publish/Subscribe WCF

Agenda

- Review: SOA
- Review: WCF
- Fundamentals of Message Exchange Patterns
- Event-Driven Architecture & Publish-Subscribe
- **Neuron ESB and Publish-Subscribe**
- Service Mediation

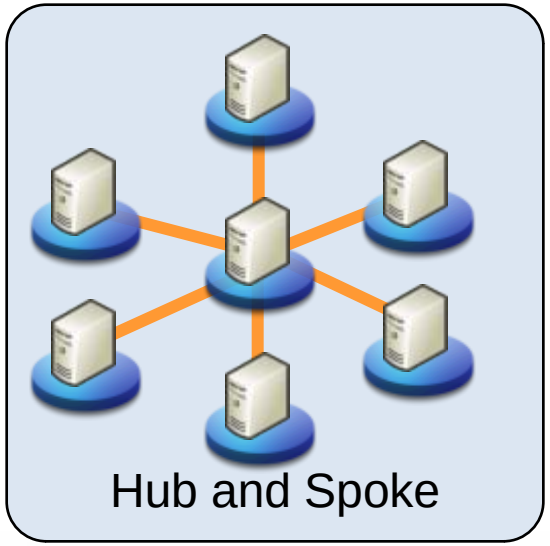
Evolution of Architectures



unmanaged / decentralized

Suitable for small environments

n^2 lines of connectivity



managed / centralized

Supports lose coupling of systems

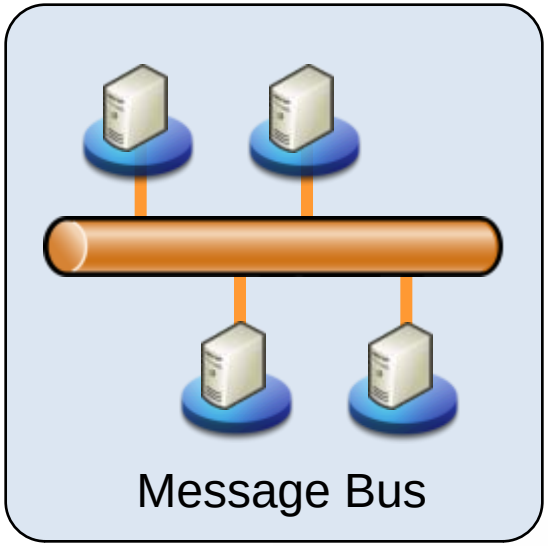
Message Broker

n lines of connectivity

Centralized management

Limited scalability

Single point of failure



managed / decentralized

Common communication infrastructure

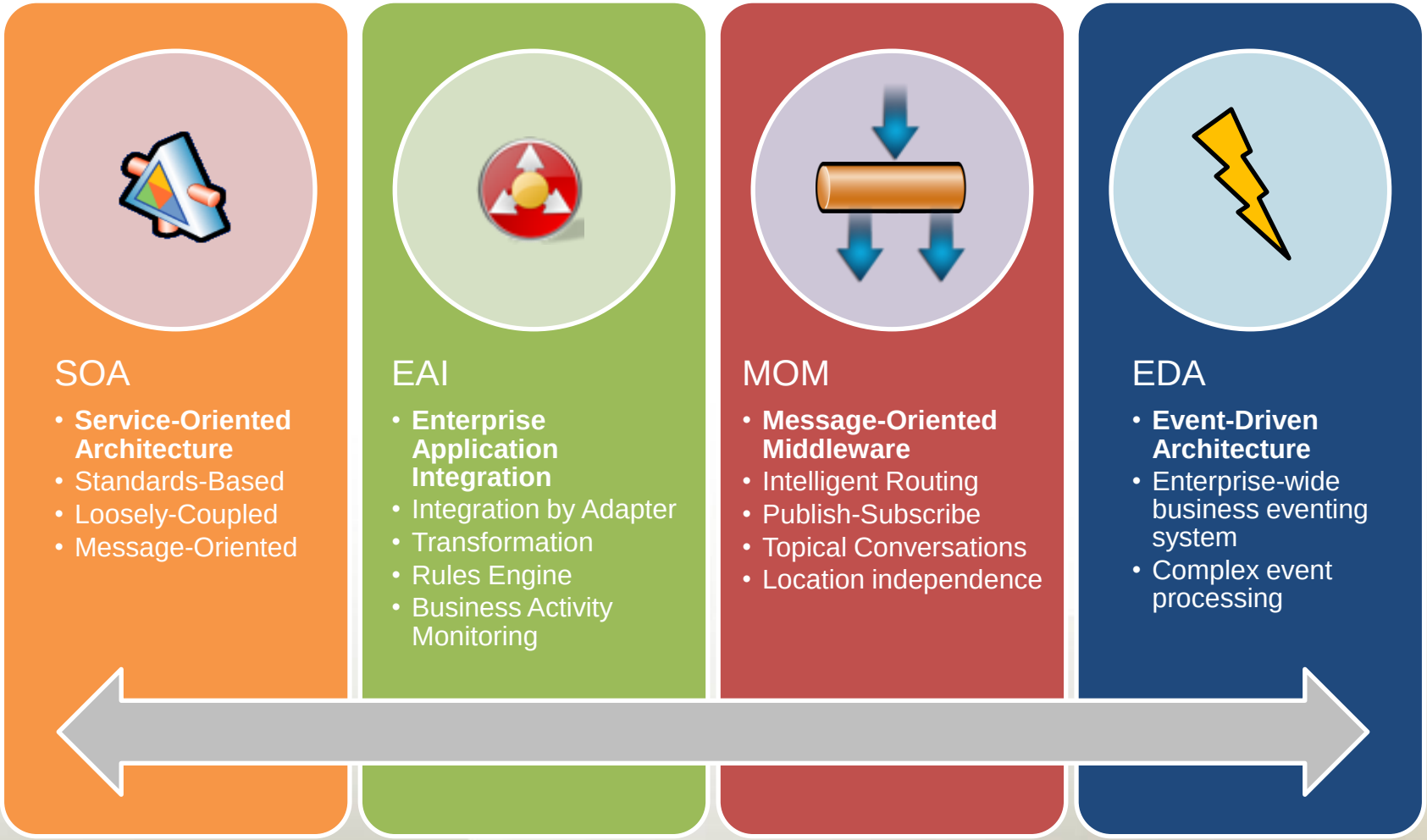
Common command infrastructure

n lines of connectivity

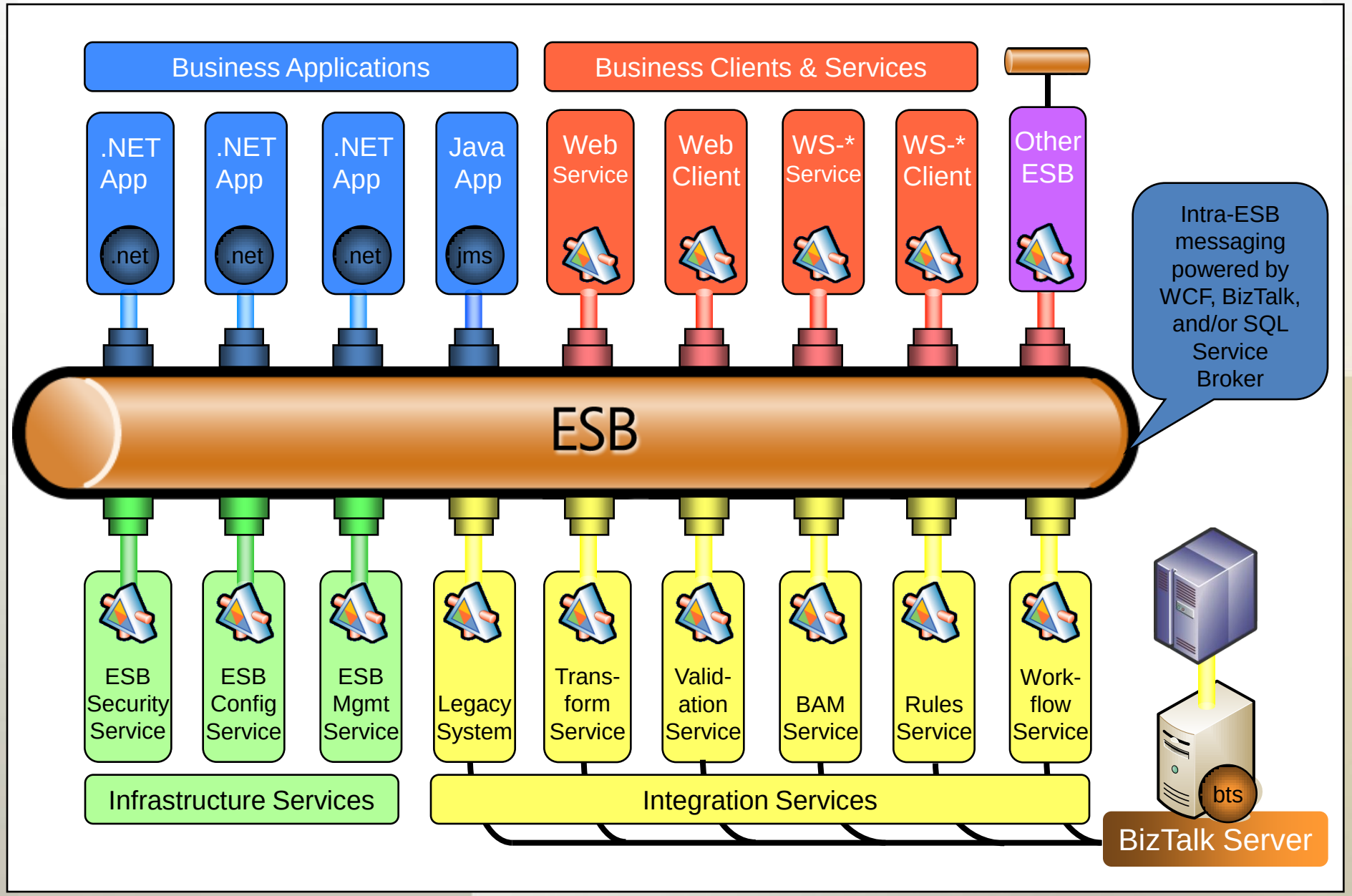
Proprietary communication protocols

Complex management

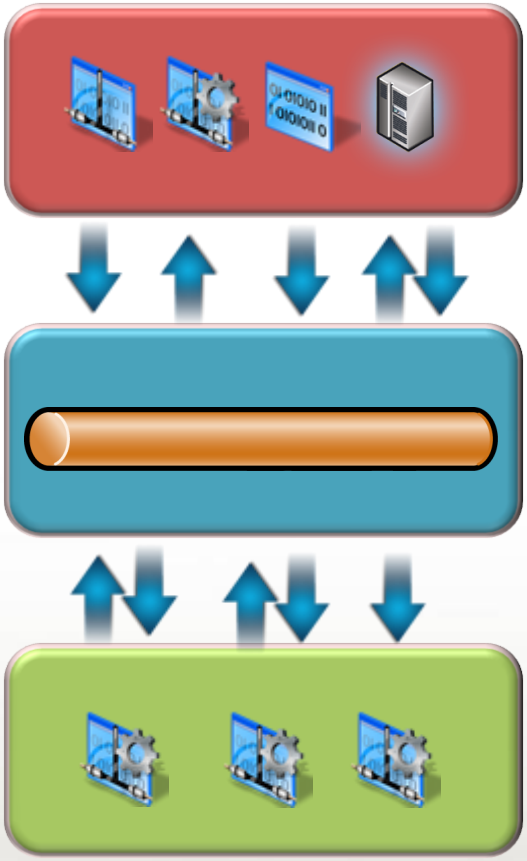
ESB is Convergence of Disciplines



A True Microsoft ESB



3 Elements of an ESB



Endpoints

Publishers & Subscribers	Services	Business Applications	Legacy systems	Human Beings

Bus

Communication & Integration Backbone	Publish-Subscribe Topical conversations	Logically centralized, Physically decentralized	Infrastructure services

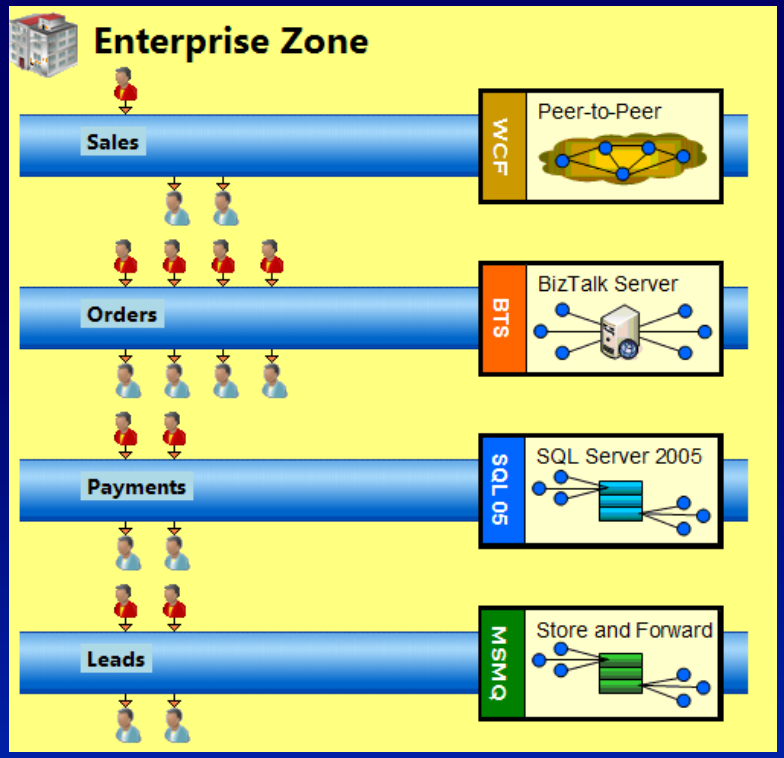
Integration Services

EAI functionality exposed as services	Invoked as messages are en route	Transform-ations	Rules Engine	Business Activity Monitoring

Publish-Subscribe

- Deluxe pub-sub implementation
- Enterprise-wide business eventing system
- Logical topics replace discrete endpoint targets
- Routing logic not embedded in applications
- Routing easily changed via central configuration
- Apps easily added/removed, even while live

Publish-Subscribe Topic Networks



ESB provides a choice of publish-subscribe implementations across the Microsoft stack

- WCF PeerChannel
- WCF HTTP / TCP
- BizTalk Server
- SQL Service Broker
- MSMQ

Each topic is a separately configured network

- Customers can leverage their investments in the Microsoft stack

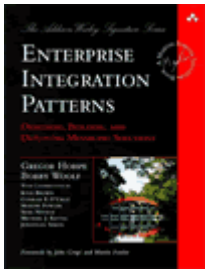
Provides a choice of QoS covering a wide spectrum of latency and reliability

- Use most appropriate technology for each business conversation
- Change any time painlessly

Demo!

Neuron Topic Pub/Sub

Resources

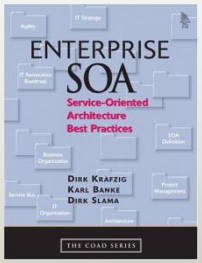


Enterprise Integration Patterns
Gregor Hohpe, Bobby Woolf
Addison-Wesley, 2004

Integration Patterns
Microsoft Patterns & Practices
2004

Patterns of Enterprise Application Architecture
Martin Fowler
Addison-Wesley, 2003

Enterprise Solution Patterns using Microsoft .NET
Microsoft Patterns & Practices
2003



Enterprise SOA
Dirk Krafzig, Karl Banke,
Dirk Slama
Prentice Hall, 2004