Introducing The ABCs of Windows Communication Foundation

Rick G. Garibay

Agenda

- A Brief History of Service Orientation
- White-box vs Black-box Reuse
- Services, SOA and Service Oriented Application Architecture
- Tenets of Service Orientation
- Windows Communication Foundation
- Service Contract
- Operation Contract
- Hosting Options
- Data Contracts
- WCF Clients

About Me

- Senior Consultant with Neudesic, Desert Mountain Region
- 8+ years professional experience developing Microsoft solutions for various industry verticals
- Author, speaker, WCF nut
- MCAD, MCP .NET Framework
- http://rickgaribay.net

Upcoming Events

MSDN Events – Silverlight, LINQ & WCF (Phoenix, AZ)

Wed., Sept. 26, 2007 - 1:00-5:00 PM Pacific Time

Microsoft Southwest Office - Phoenix, AZ

Building an Effective Messaging Infrastructure with WCF

Registration:

(877) 673-8368 (877-MSEVENT)

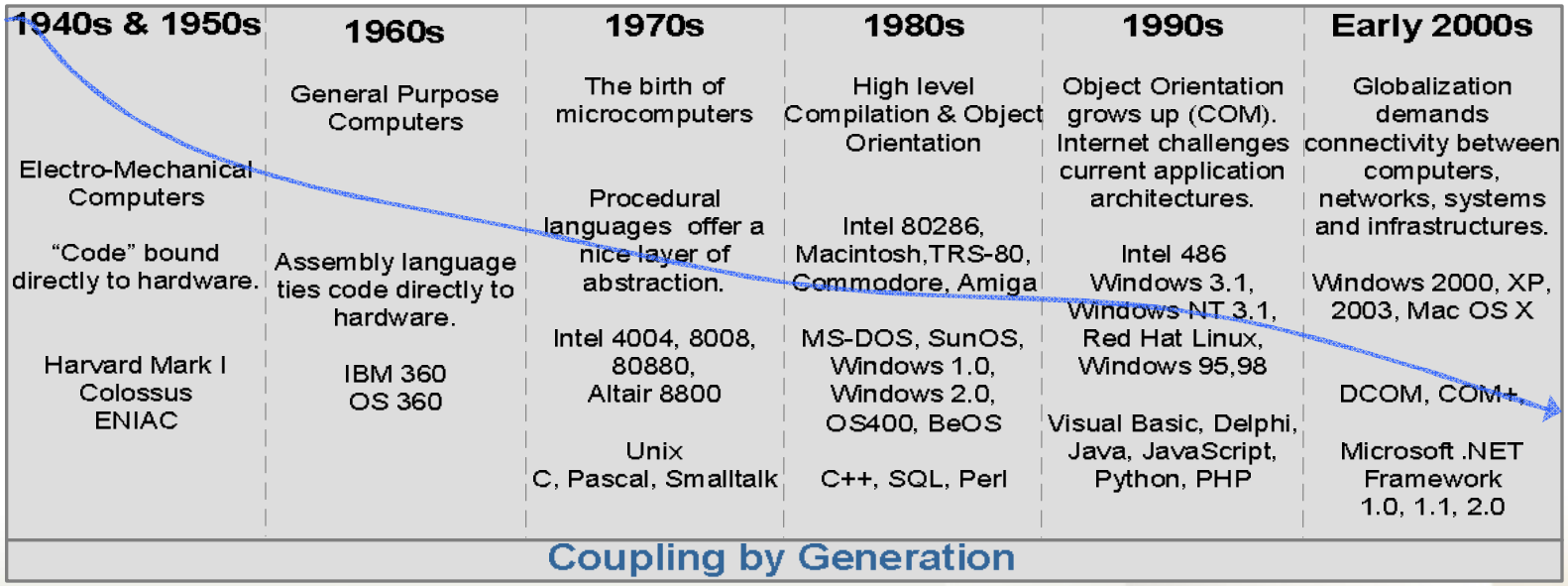
or

http://msevents.microsoft.com

Code # 1032351524

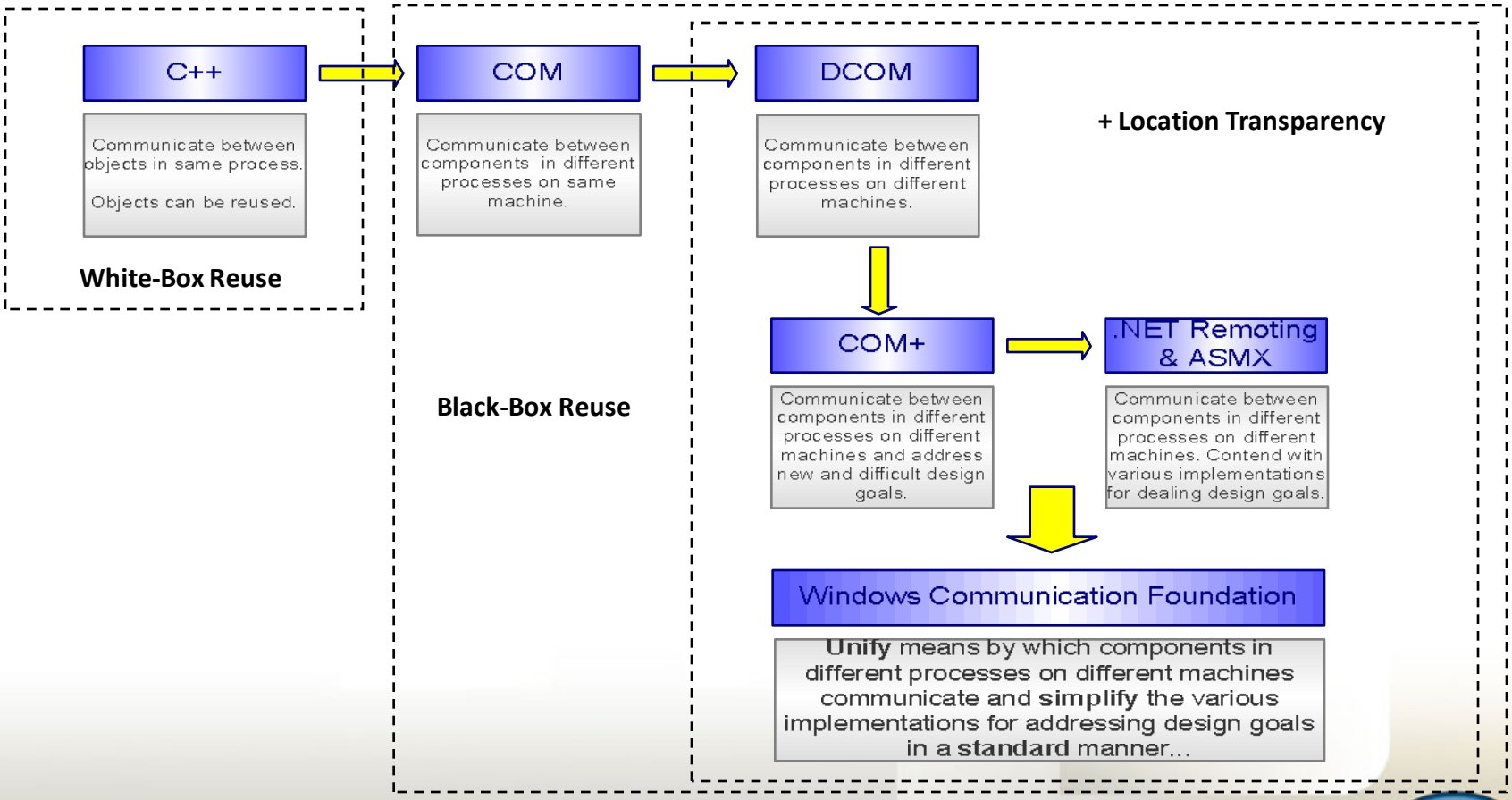
A Brief History of Service Orientation

- "Modern software engineering is ***the ongoing refinement of ever increasing degrees of decoupling***" – *Juval Lowy*



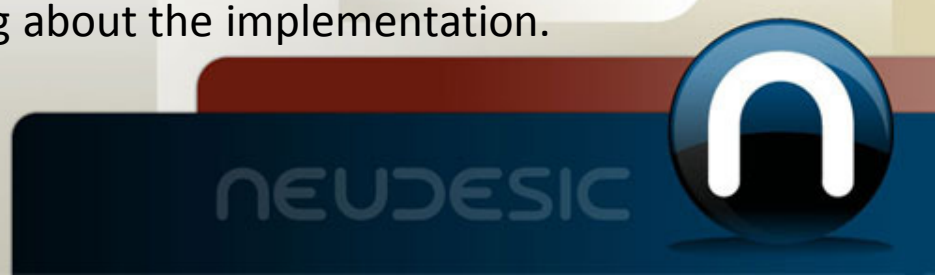
- The historical relationship between software engineering and the reduced coupling between hardware, OS, objects, components and services.

A closer look at recent developments



White-box vs Black-box Reuse

- The primary motivation for moving from procedural programming to objects was all about reuse, and thus, improved productivity.
- As it turns out, most reuse was at the expense of productivity due to the white-box nature of classic objects.
- White Box Reuse
  - Developers must be intimate with object/class internals, often within all levels.
  - Client code is tightly bound to implementation.
  - Changes in a generalized class causes hidden (and sometimes severe) impact to client code.
- Black Box Reuse
  - Developers must simply understand the contract, or interface.
  - Interfaces provide separation between definition and implementation.
  - Client code knows absolutely nothing about the implementation.



Services, SOA and Service Oriented Applications

- In general, we all agree that:
  - Coupling is a bad thing.
  - But, coupling, to some extent is unavoidable.
- **Services** (Service Oriented Architecture, and Service Oriented Applications) are the next evolution from **components**, just as interfaced-based components where an evolution from **objects**.
- SOA is, to the best of our knowledge, the best way to build distributed applications that are maintainable, where the ***right*** *aspects are decoupled yielding:*
  - *Interoperability*
  - *Productivity*
  - *Maintainability*
  - *Extensibility*
  - ***Real*** *Reuse*

Services, SOA and Service Oriented Applications

Characteristics *within* services:

- Languages (i.e. C#)
- Technologies (i.e. .NET FX, J2EE)
- Platforms (Windows, etc)
- Versions (Major.Minor.Build.Revision)

Characteristics *between* services:

- Standards/Protocols (HTTP, MSMQ, IPC, etc)
- Policies (Transactions, Reliability, etc)
- Contracts (WSDL)
- Messages (SOAP)

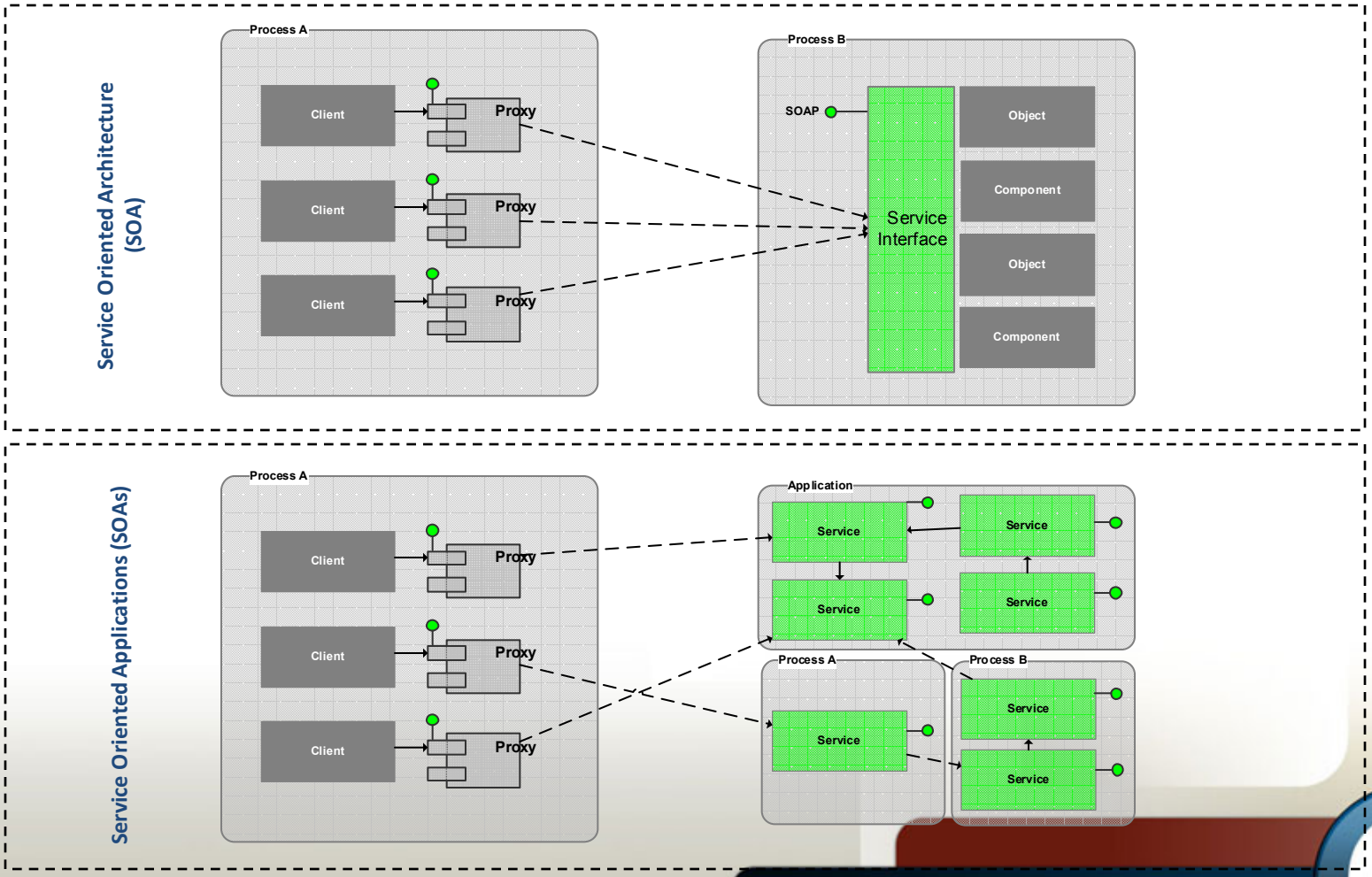
…But Web Services are Nothing New!

True, but…

- Interoperability is at the expense of reliability - can you really afford to forgo reliability in your mission critical services?
- HTTP can be limiting
- ASMX & WSE has been deprecated
- WS-* specifications are just that (constantly changing and evolving)

- WCF supports all relevant specifications via bindings:
  - WsHttpBinding and WsDualProfileBinding
    - WS-Secure Conversation
    - WS-Reliable Messaging
    - WS-Atomic Transactions
    - WS-Coordination
  - More on bindings:
    - http://windowssdk.msdn.microsoft.com/en-us/library/ms730294.aspx

Big SOA vs. Little SOA



Tenets of Service Orientation

- Boundaries are Explicit.
- Services are Autonomous.
- Services share Schema and Contract, not Class.
- Compatibility is Based on Policy.

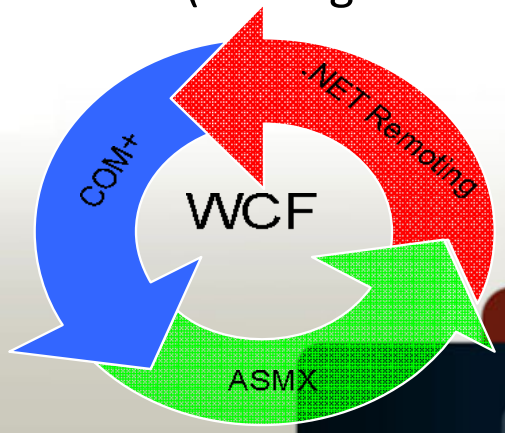
Additional SO Principles

Additional SO principles (you should adhere to simply because it is the right thing to do) :

- Services are secure.
- Services leave the system in a consistent state.
- Services are thread-safe.
- Services are reliable & robust.
  - Interoperable
  - Scalable
  - Available
  - Responsive
  - Disciplined

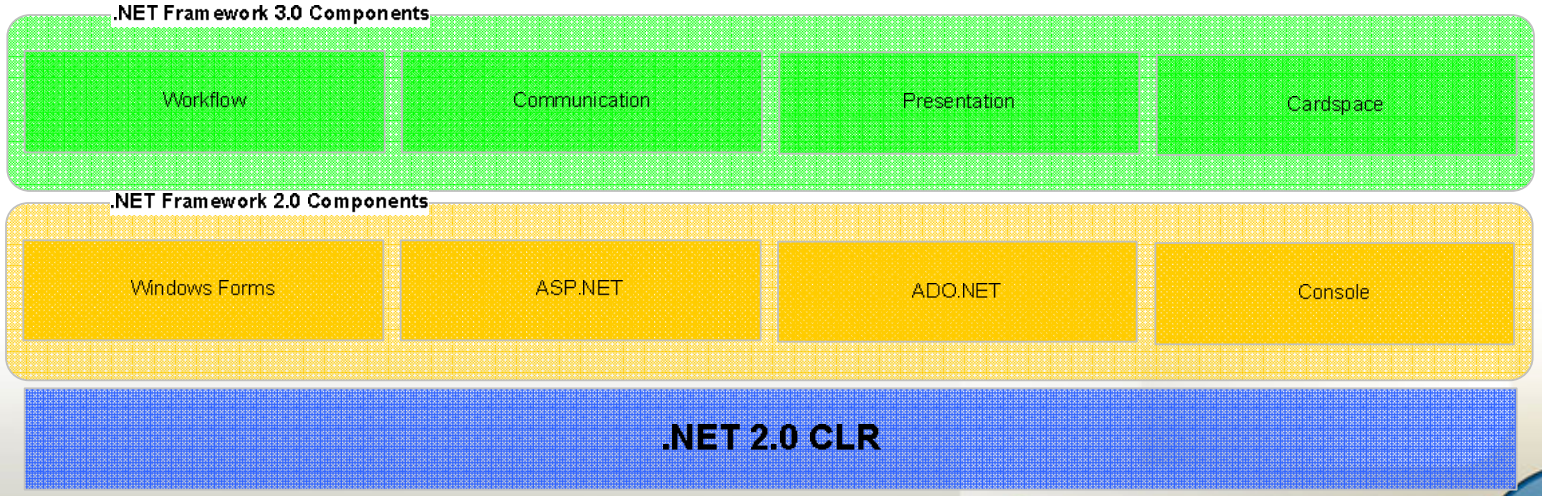
Introducing WCF

- WCF is Microsoft's implementation of industry standards to provide a communication subsystem enabling applications on one machine (process boundary) or across multiple machines to communicate.
- WCF is a core component of the .NET Framework 3.0 (formerly WinFX) which is included with Windows Vista and will support Windows 2003 and Windows XP platforms as well as the future version of Windows Server.
- The WCF API unifies ASMX Web Services, .NET Remoting, distributed transactions and messaging into a single programming model that makes true service orientation tenable (while guaranteeing interoperability).
- Fundamental to .NET



.NET 3.0 (formerly WinFx)

- .NET 3.0 is additive to .NET 2.0.
- It uses (and requires) the .NET 2.0 CLR exclusively



The ABCs of WCF

An **Address** uniquely identifies a service.
Provides the **transport** protocol, name of target **machine** (host) and **port** if applicable.
Expressed as an explicit **path** or URI:

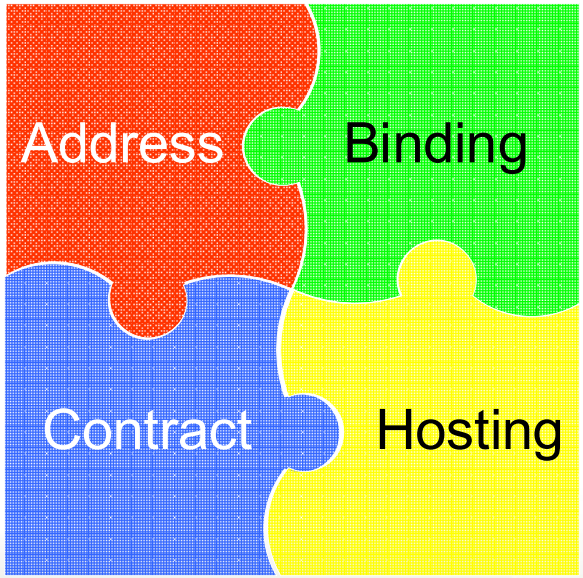
[**transport**]://[**machine**][:optional **port**]

```
http://localhost
http://localhost:8081
http://localhost:8081/Service
net.tcp://localhost:8082/Service
net.pipe://localhost/Pipe
```

All services expose a **Contract.**

WCF uses 5 types of contracts:

Service Contract – Exposes the service.
Operation Contract- Exposes the service members.
Data Contract – Describes service parameters.
Fault Contracts – Defines error handling semantics.



Bindings provide "canned" policies that implement the WCF features required to support the design goals of the service.

Some common bindings include:

```
BasicHttpBinding
NetTcpBinding
WSHttpBinding
NetMsqmqBinding
```

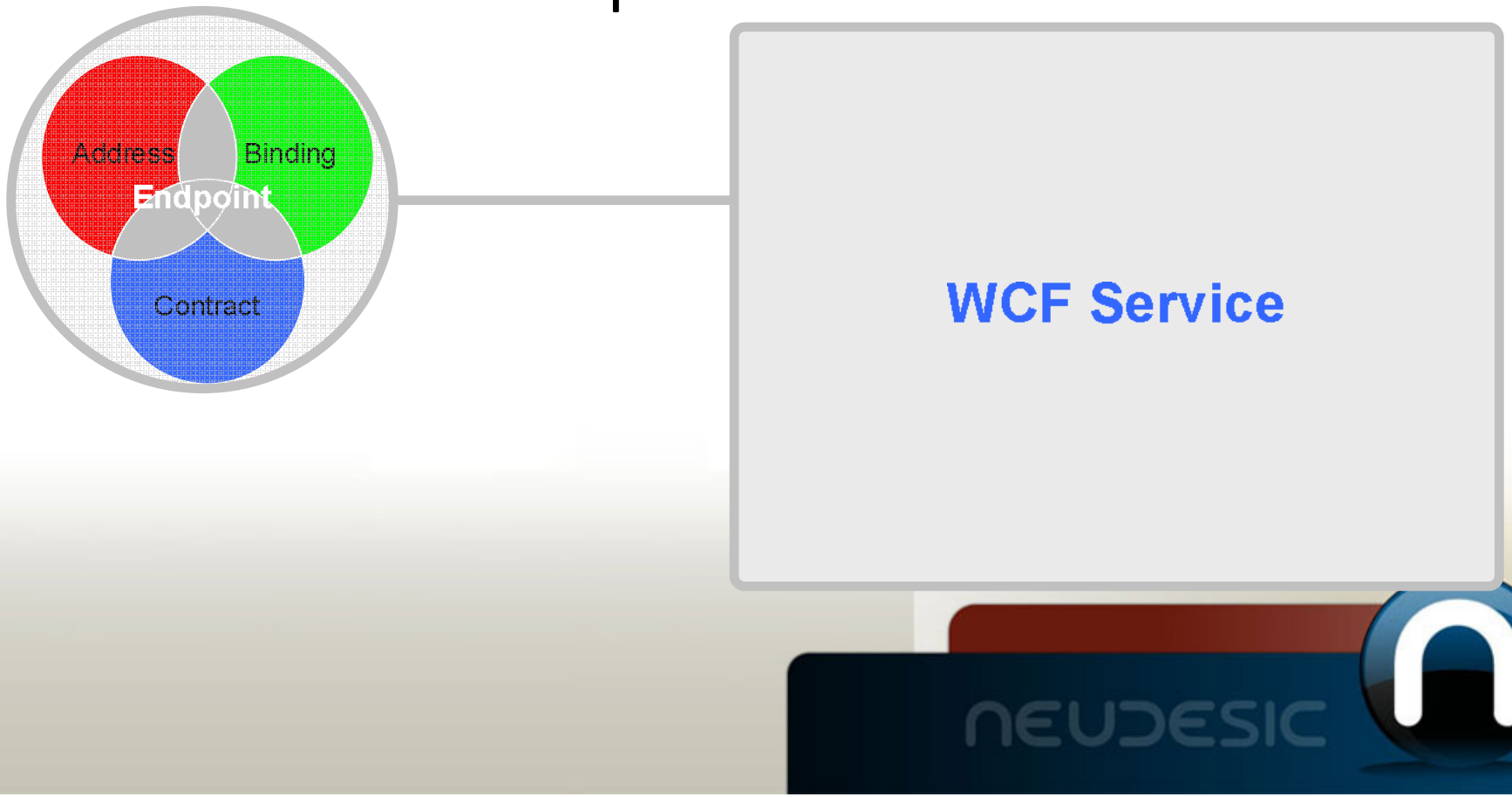
WCF services must be hosted by a Windows Process (host process).

Hosting options include:

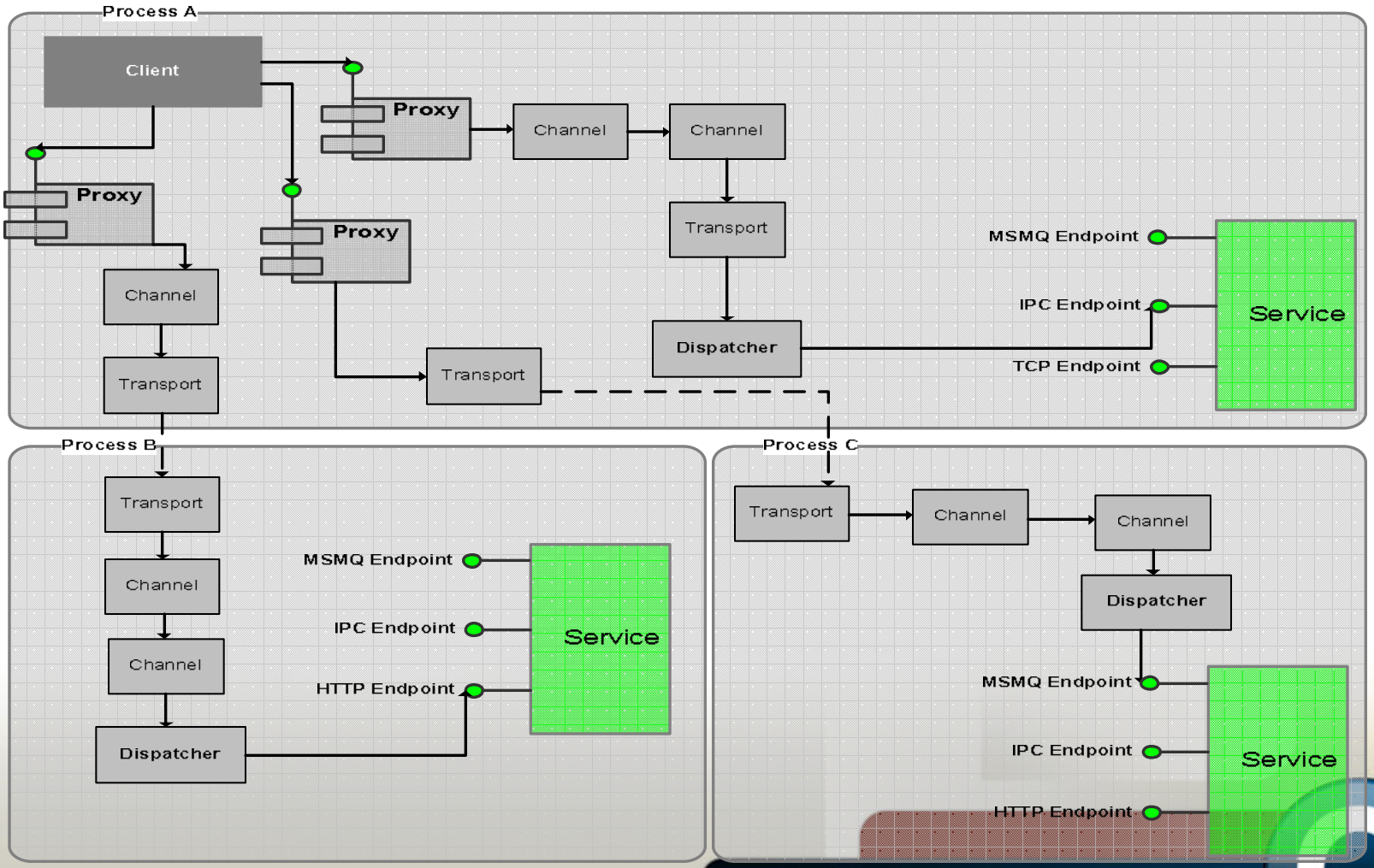
IIS 5 & 6
IIS 7 & Windows Activation Service (WAS)
SmartClient, Console or Windows Service (also called "Self-Hosting")

Endpoints

- WCF Services are deployed, discovered and consumed as endpoints.



Conceptual Architecture



Service Contracts

- .NET 3.0 WCF uses common .NET interface along with the ServiceContractAttribute class to explicitly define a contract.
- Service Contracts are implicitly public (access modifiers are a .NET notion).

```
using System;
using System.ServiceModel;

[ServiceContract()]
interface IPersonnelAction
{
  // Members

}
```

Operation Contracts

- An Operation Contract uses the OperationContractAttribute class to opt-in a method to participate in a Service Contract.

```
using System;
using System.ServiceModel;

[ServiceContract()]
interface IPersonnelAction
{
     [OperationContract]
     double AdjustSalary(int employeeId, double baseSalary, double adjustmentPercent);

     // Not a part of the contract
     bool Terminate(int employeeId);
}
```

Implementing a Service Contract

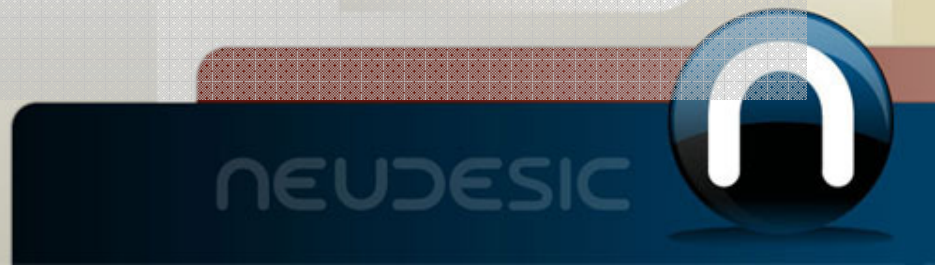
- A Service Contract is implemented just as any other .NET interface

```
using System;
using System.ServiceModel;

class HRManagerService : IPersonnelAction
{
  // Implementation
}
```

Implementing a Service Contract

- Disciplined developers will maintain the separation of contract from implementation (nothing changes here).

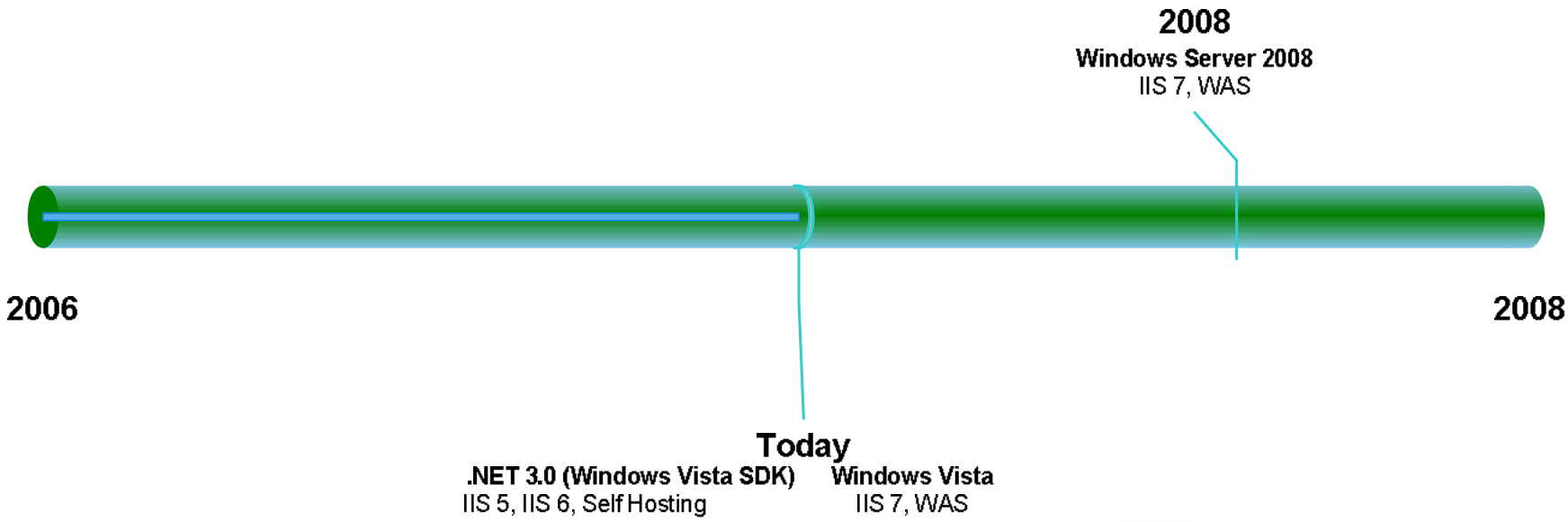
```
using System;
using System.ServiceModel;

// Avoid!!!
[ServiceContract()]
class HRManagerService
{
      [OperationContract]
      double AdjustSalary(int employeeId, double baseSalary, double adjustmentPercent)
      {
           // Implementation
      }

      // Not a part of the contract
      bool Terminate(int employeeId)
      {
         // Implementation
      }

}
```



Demo - Service Contract, Operation Contracts & Configuration

- Define a Service Contract
- Opt-In Operations
- Implement the Service Contract
- Configure the Service

Hosting Options



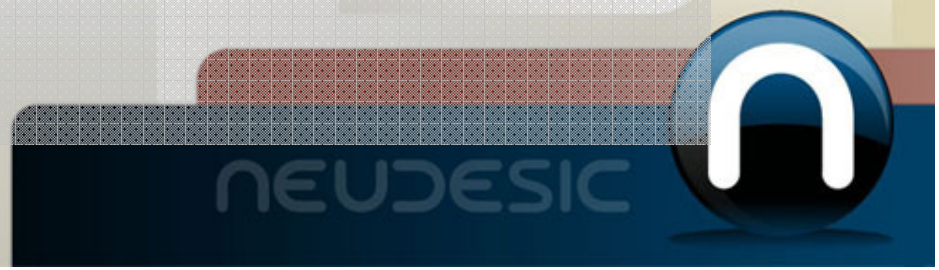
Demo - IIS Hosting, Deployment & Metadata Exchange

- Create Service stub
- Deploy Service
- Test Service
- Enable Metadata Exchange

Data Contracts

- A Data Contract marks a business entity (class) as a participant in a Service Contract and Service Operation.

```
using System;
using System.Runtime.Serialization;

[DataContract()]
public class Employee : Person
{
            private string m_Name = string.Empty;
            private int m_EmployeeId = string.Empty;

            public int EmployeeId
           {
            get
              {
                return m_EmployeeId;
              }
            set
              {
                m_EmployeeId = value;
              }
           }
 }
```



Data Members

- Unlike using the SerialziableAttribute, all entities and members are strictly opt-in.

```
using System;
using System.Runtime.Serialization;

[DataContract()]
public class Employee : Person
{
            private string m_Name = string.Empty;
            private int m_EmployeeId = string.Empty;

            [DataMember]
            public int EmployeeId
            {
            get
               {
                  return m_EmployeeId;
               }
            set
               {
                  m_EmployeeId = value;
               }
            }
 }
```

Demo - Data Contracts, Data Members & Operation Overloading

- Define a business entity
- Annotate with Data Contract Attribute
- Define Data Members
- Demonstrate Operation Overloading

Demo – Writing a WCF Client

- Add a Service Reference
- Generate a proxy
- Consume the proxy

Tip of the Iceberg

- Bindings
- Fault Handling
- Instance Management
- Security
- Transactions
- Concurrency Management
- Queued Services
- Service & Operation Factoring & Design

References & Resources

- IDesign WCF Master Class, Microsoft Silicon Valley Campus, San Jose, CA. June 2006.
- IDesign WCF Standard
- Programming WCF Services, Juval Lowy, 2007 O'Reilly Press.
- Timeline of Computing, Wikipedia: http://en.wikipedia.org/wiki/Timeline_of_computing. July 2006.
- S. Someasegar, Someasegar's Blog: http://blogs.msdn.com/somasegar/archive/2006/06/09/624300.aspx. July 2006