Модуль 1: Знакомство с Windows Communication Foundation

{Спицын Александр Геннадьевич}
Ведущий разработчик отдела разработки ПО ФТС, MCPD(EA)
{e-mail: sag@acs-it.ru}

Содержание

- Проектирование приложений в стиле SOA
- Обзор архитектуры WCF
- Использование интерфейсов как сервисных контрактов.
- Реализация простого WCF сервиса в Visual Studio 2008
- Реализация простого WCF клиента в Visual Studio 2008
- Лабораторная работа: Создание простого WCF сервиса.

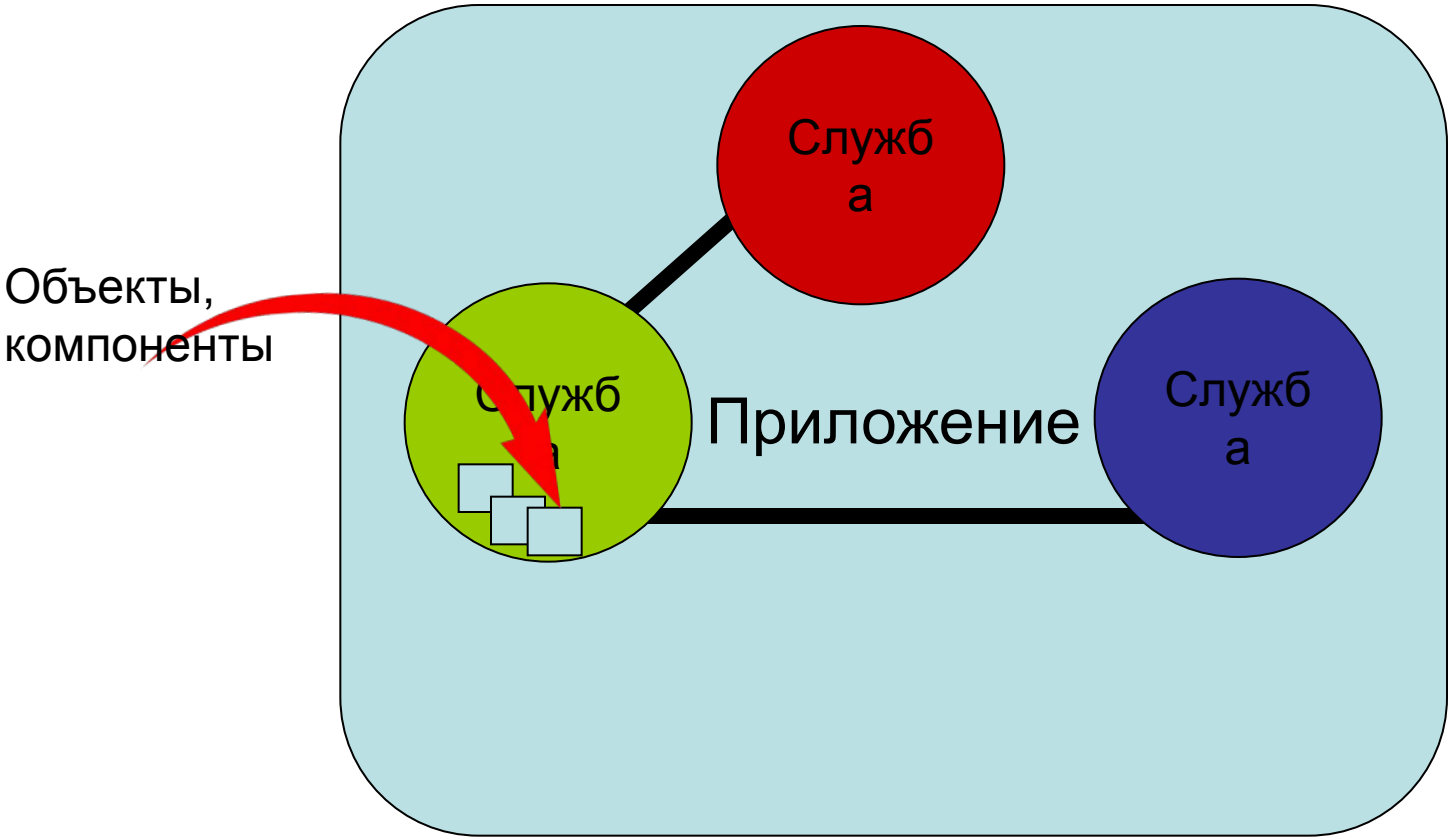
Урок 1: Разработка приложений в стиле COA

- Преимущества сервис ориентированной архитектуры
- Разработка SOA приложений
- WCF и SOA
- WCF в контексте SOA

История развития методологий

- Функциональный подход
- Объектно-ориентированное программирование
- Компонентно-ориентированное программирование
- Служебно-ориентированное программирование

Служебно-ориентированные приложения



Преимущества сервис ориентированной архитектуры

Взаимодействие базируется на межотраслевых стандартах.

Служебно-ориентированные программы более устойчивы к ошибкам.

Избавляют разработчика от возни с вторичным кодом и позволяет сосредоточиться на бизнес-логике.

Межтехнологическая совместимость. При написании обычно не нужно думать на какой платформе написан клиент.

Беспроблемный переход через границы(границы безопасности, географические, организационные, временные, транзакционные …) благодаря стандарту, основанному на базе обмена сообщениями

WCF & SOA

- WCF поддерживает SOA:
  - Разработчикам нет необходимости изучать WSDL
  - Множества функционала реализует среда выполнения
  - Существуют приложения облегчающие создание сервисов и клиентов.
- **Однако:**
  - То, что вы лишь предоставляете функционал службы через Web сервис не означает, что вы создаете приложение в стиле SOA
  - SOA помогает проектировать сервисы, а WCF предоставляет возможность реализовывать данные сервисы.

Каноны и принципы

- Четкие границы служб.
- Автономность служб.
- Службы предоставляют контракты операций и схемы данных, а не метаданные, специфические для конкретных типов технологий.
- Совместимость служб определяется политикой.

Практические принципы

- Службы должны быть безопасными.
- Службы должны оставлять систему в стабильном состоянии.
- Службы должны быть потоково-безопасными.
- Службы должны быть надежными.
- Службы должны быть устойчивы к ошибкам.

Необязательные принципы

- Службы должны быть совместимы
- Службы должны быть масштабно-инвариантны.
- Службы должны быть доступны.
- Службы должны обладать доступным временем отклика.
- Службы должны работать в нормальных временных рамках.

WCF в контексте SOA

Пример: приложение для абстрактного заказчика

Уровень языка:

- Объекты заказчика, хранимые в базе данных
- Бизнес правила (например хранимые процедуры)
- Workflow
- Стандартные типы .NET Framework

Уровень службы:

- Напоминает бизнес фасад с иными, более дружественными к сервисам типами
- Методы службы больше похожи на определенные бизнес функции и скрывают обращение к нескольким функциям уровня языка

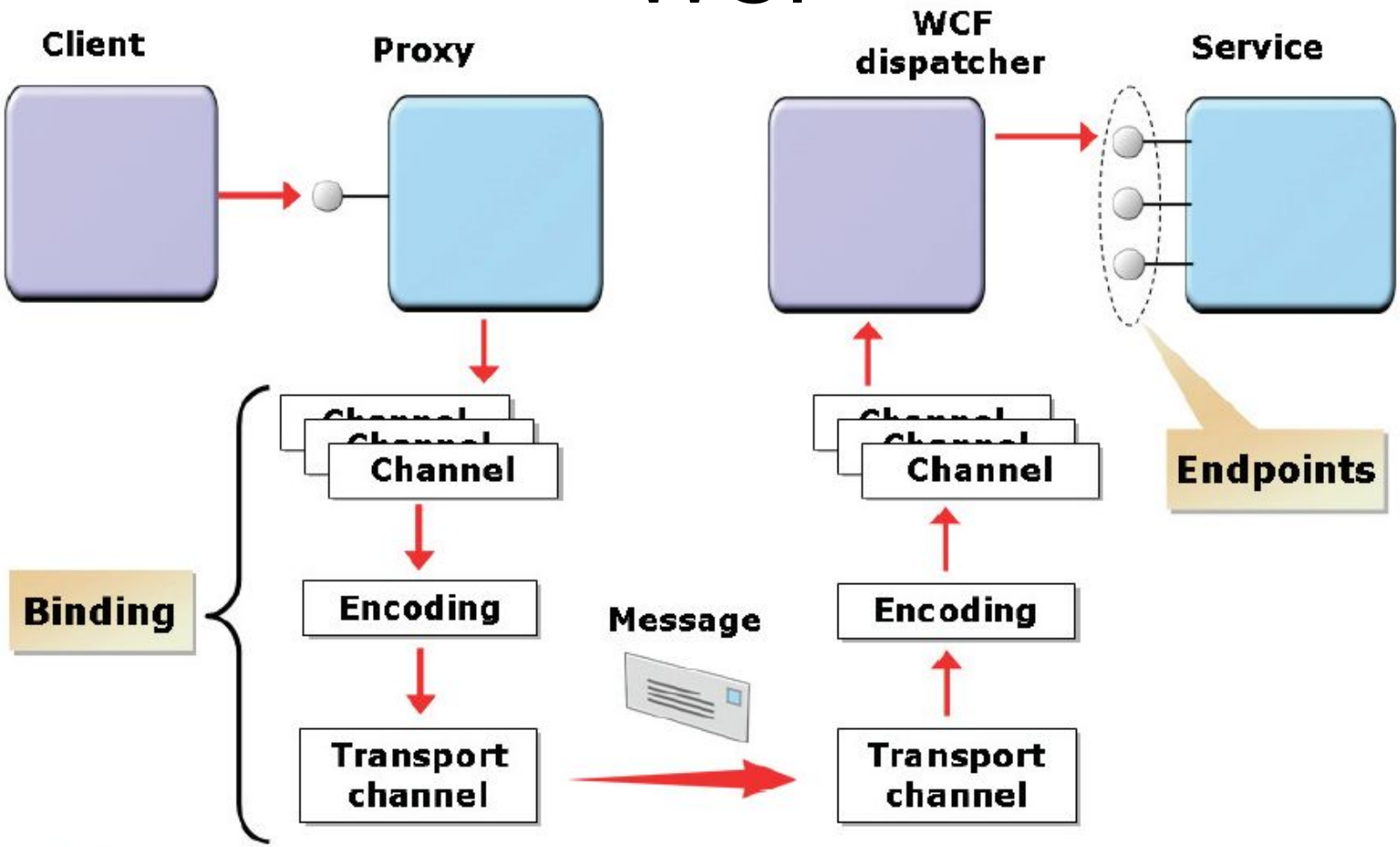
Урок 2: Обзор архитектуры WCF

- Разработка сервисно-ориентированных приложений с помощью WCF
- Отправка сообщений WCF
- ABC конечных точек
- Структура сервиса
- Единая программная модель
- Взаимодействие с не WCF приложениями

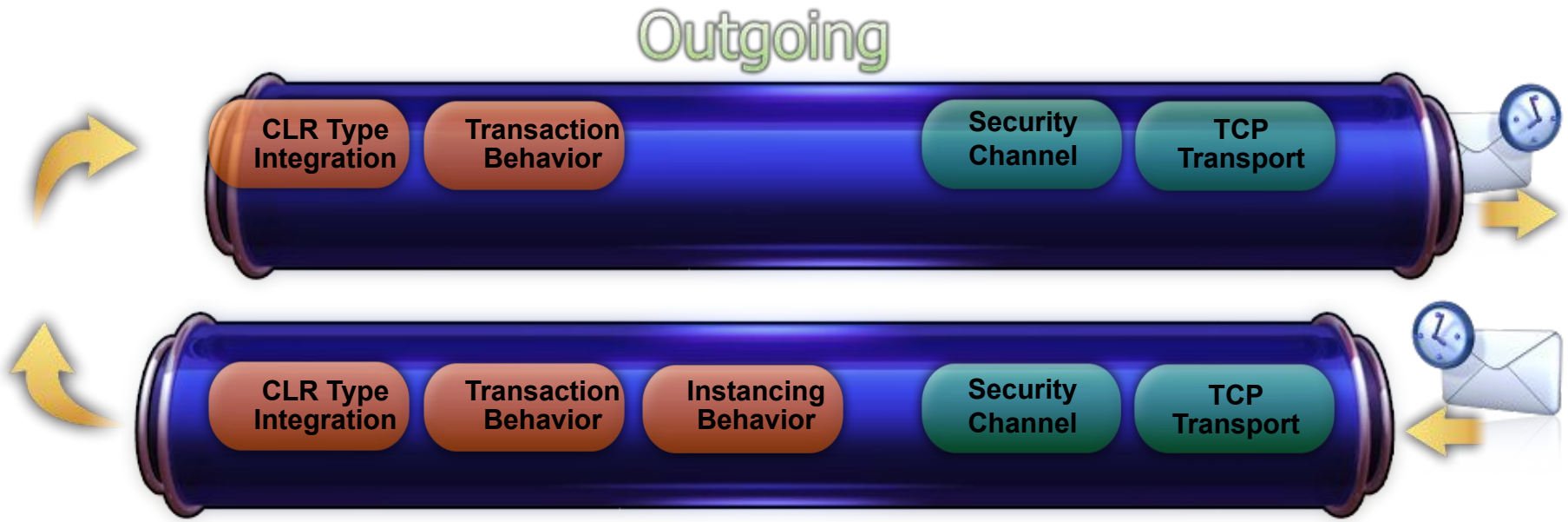
Разработка WCF сервис-ориентированных приложений

- WCF позволяет создавать сервисы единообразно, вне зависимости от сетевого протокола взаимодействия
- Клиент обращается к сервису для получения определенного функционала
- Клиенты и сервисы взаимодействуют посредством обмена сообщениями
- Предназначение клиентов и сервисов – не предоставлять друг другу единое адресное пространство
- Сервис-ориентированная архитектура – концепция и стиль проектирования сервисов

Организация взаимодействия в WCF



Архитектура



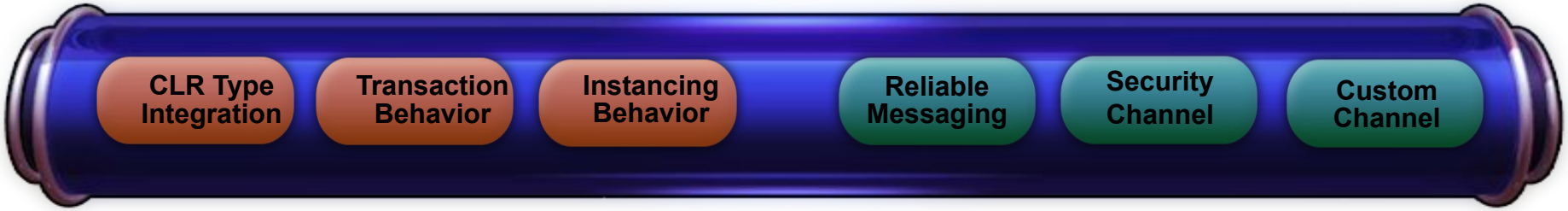
Модель сервиса

Оборачивает код сервиса в систему приема и передачи сообщений

Модель сообщений

Передает сообщения вперед и назад, реализуя логику транспорта

Компонентная архитектура



Поведение

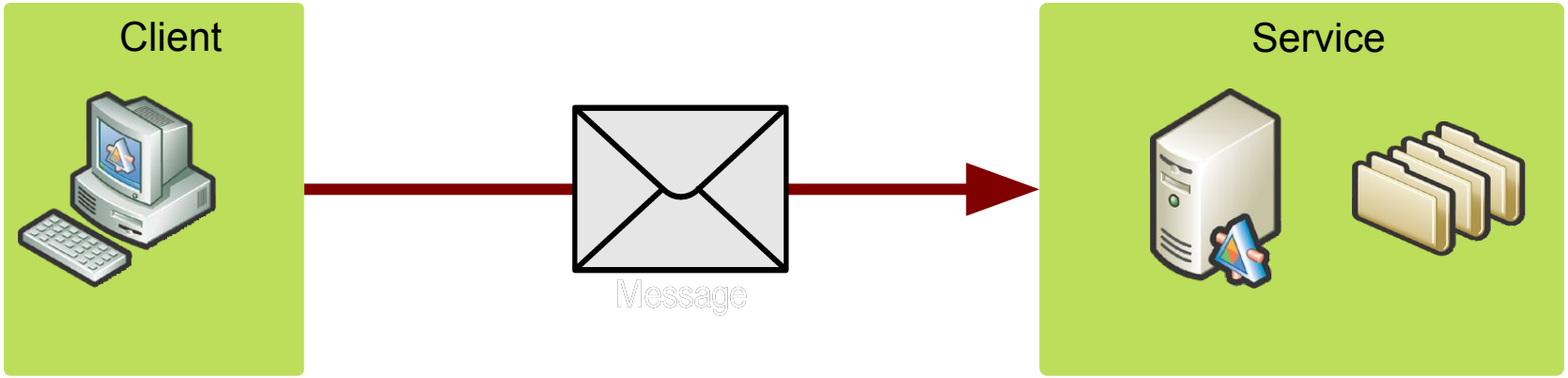
- Metadata Behavior
- Instancing Behavior
- Error Handling Behavior
- Transaction Behavior
- Concurrency Behavior
- CLR Type Integration
- Custom Behavior
- Throttling Behavior

Каналы

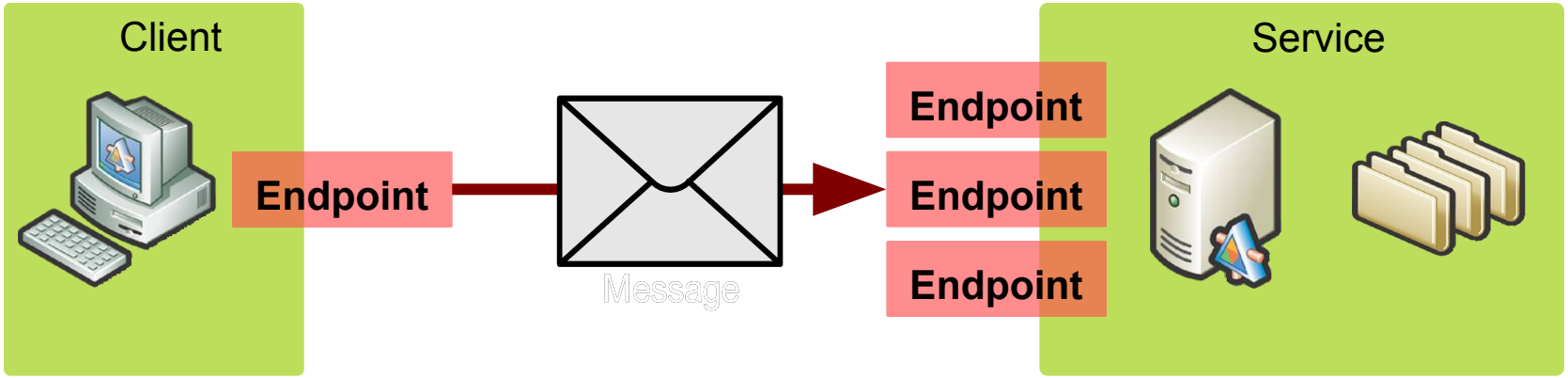
- Security Channel
- HTTP Transport
- Reliable Messaging
- TCP Transport
- MSMQ Integration
- Named Pipes
- Custom Channel
- Queue Transport

ACS

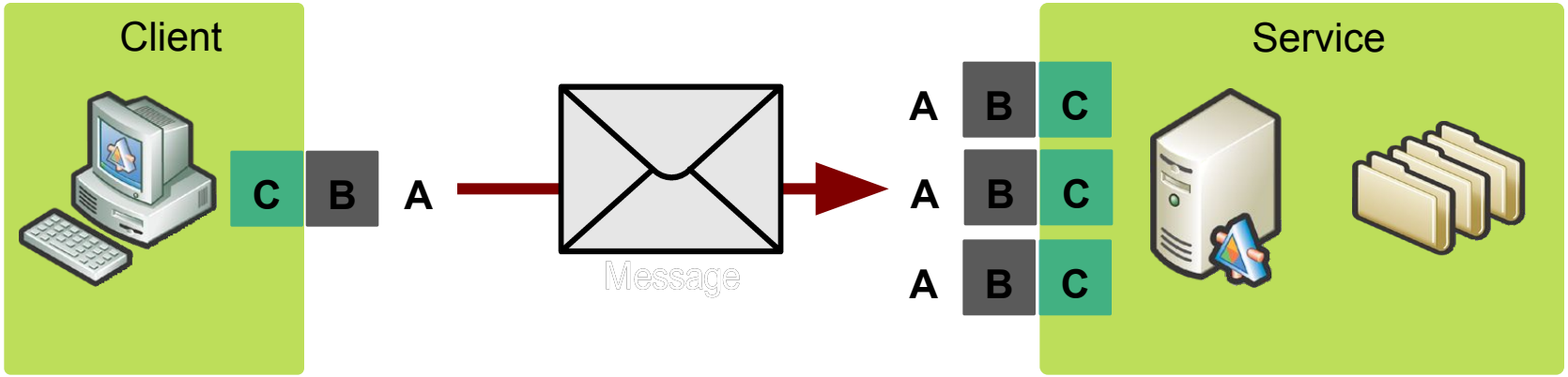
Взаимодействия клиента и сервиса



Конечные точки Endpoints



Адрес, привязка, контракт



Address

Где?

Binding

Как?

Contract

Что?

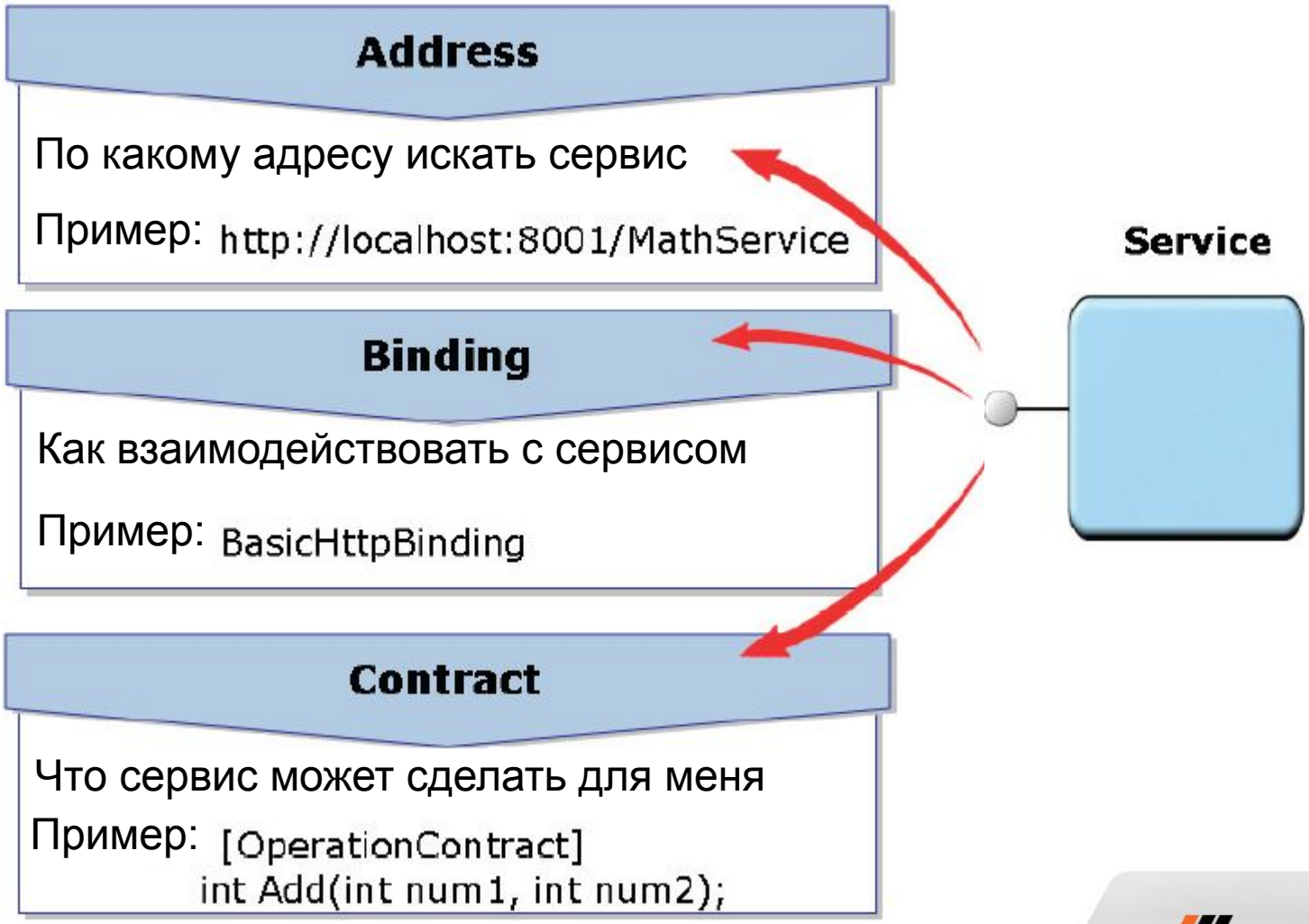
Endpoi
nt

ABC конечных точек (Endpoints)

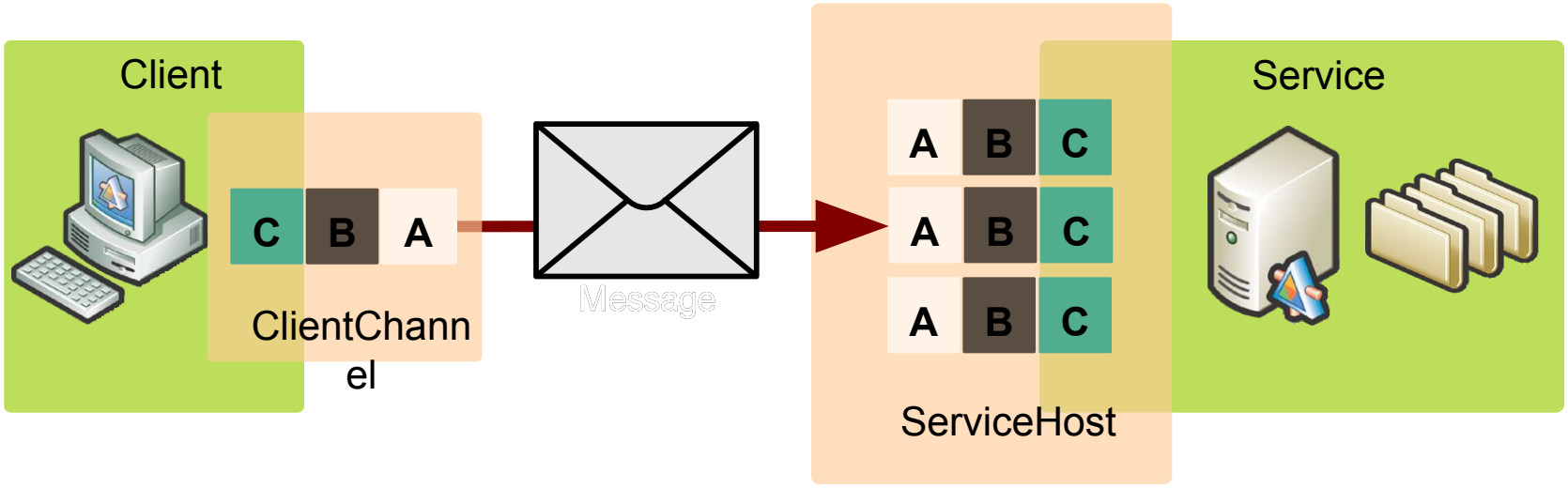
A

B

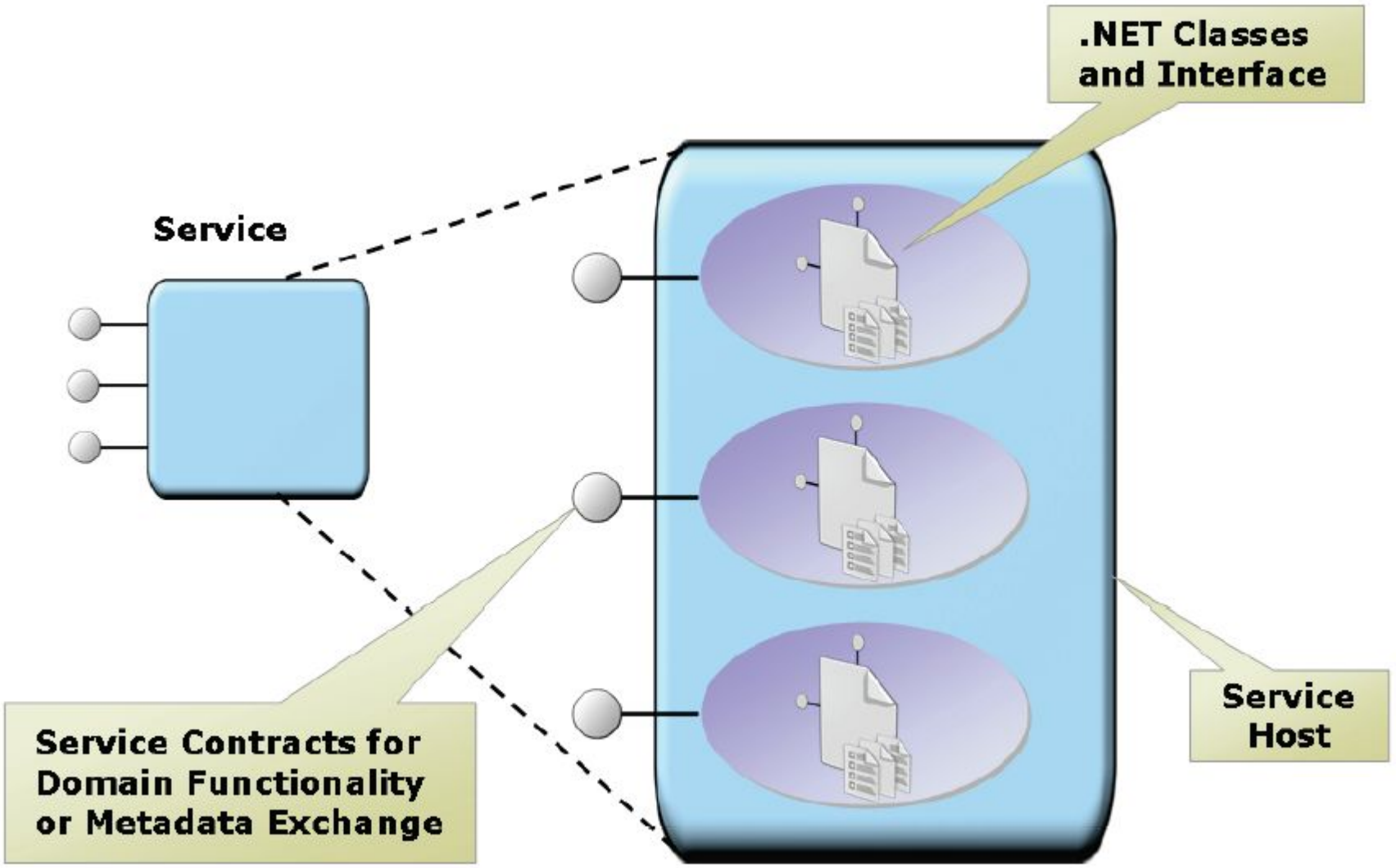
C



Создание точек взаимодействия

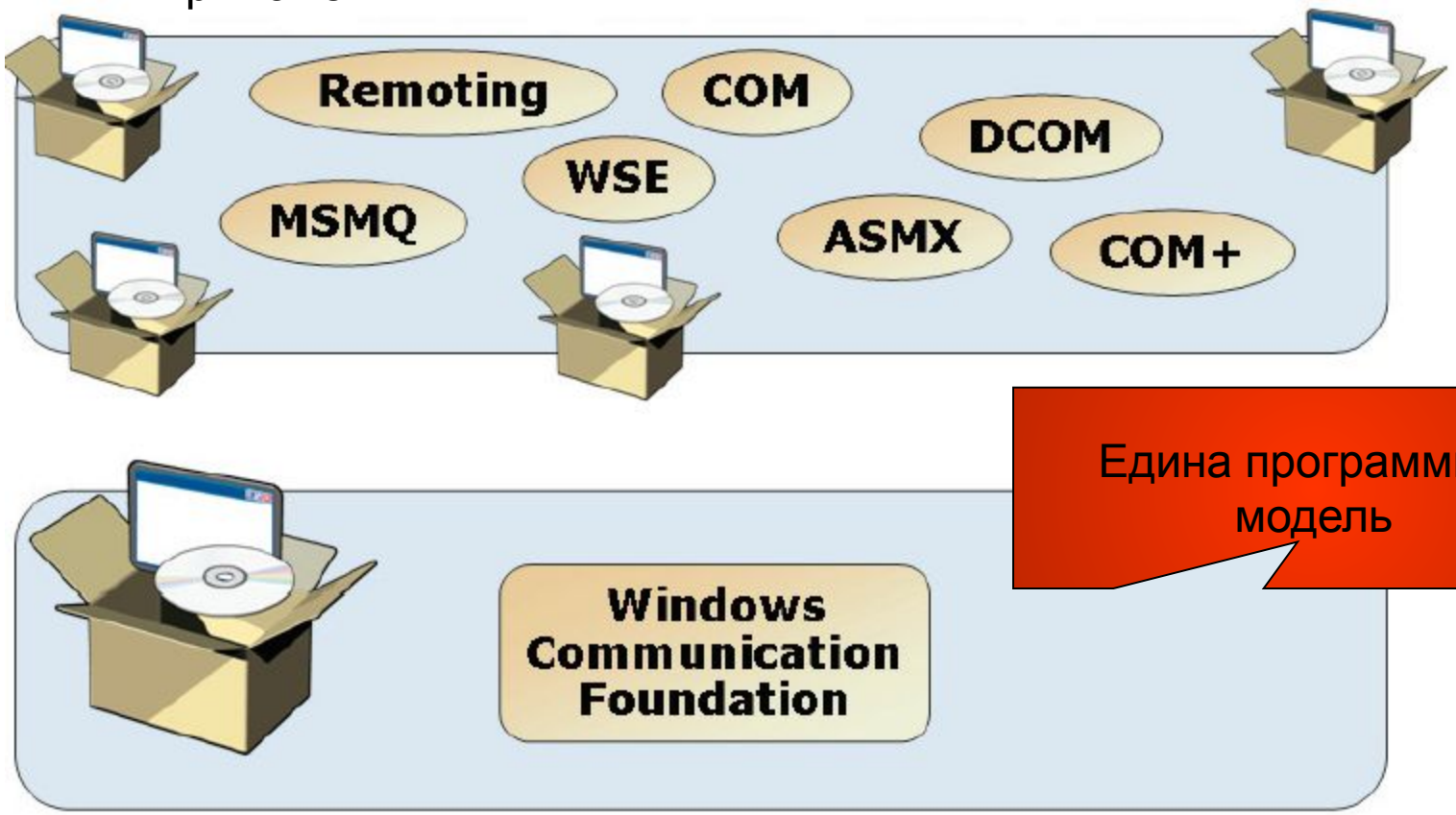


Структура сервиса



Единая программная модель

Множество технологий разработки распределенных приложений



Взаимодействие с не WCF приложениями

- Web сервисы
  - Старые Web сервисы такие как ASP.NET ASMX сервисы
  - Другие Web сервисы, поддерживающие протоколы WS-*
  - POX (plain old xml) сервисы

- Другие протоколы взаимодействия:
  - MSMQ (существует несколько привязок - bindings)
  - .NET Remoting
  - COM+

Урок 2: Использование интерфейсов как сервисных контрактов

- Пример простого контракта
- Атрибут ServiceContract
- Атрибут OperationContract
- Данные и сообщения
- Контракты, метаданные и артефакты

Пример контракта

```
using System;
using System.ServiceModel;

namespace ConnectedWCF
{
    [ServiceContract(Namespace="http://myuri.org/Simple") ]
    public interface IBank
    {
        [OperationContract]
        decimal GetBalance(string account);

        [OperationContract]
        void Withdraw(string account, decimal amount);

        [OperationContract]
        void Deposit(string account, decimal amount);
    }
}
```

Principal namespace for WCF

Attributes control exposure of types and methods

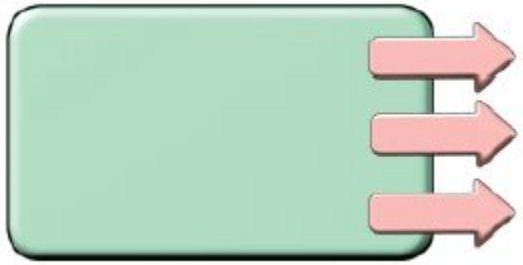
ACS

Атрибут ServiceContract

- Используется для обозначения WCF сервиса
- Накладывается на интерфейсы или на классы
- Сервис доступен клиенту, если он «экспортирован»
- Используются свойства Name и Namespace
- **Крайне желательно применять атрибут к интерфейсу а не к классу!**

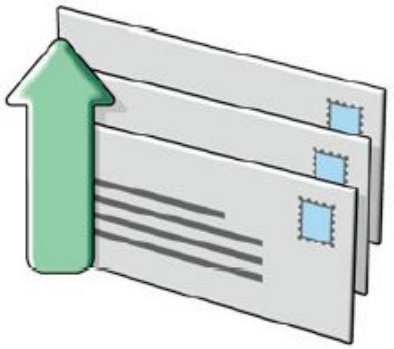
Атрибут OperationContract

- Атрибут накладывается только на методы класса!
- Методы сервиса доступны клиенту если они отмечены атрибутом OperationContract

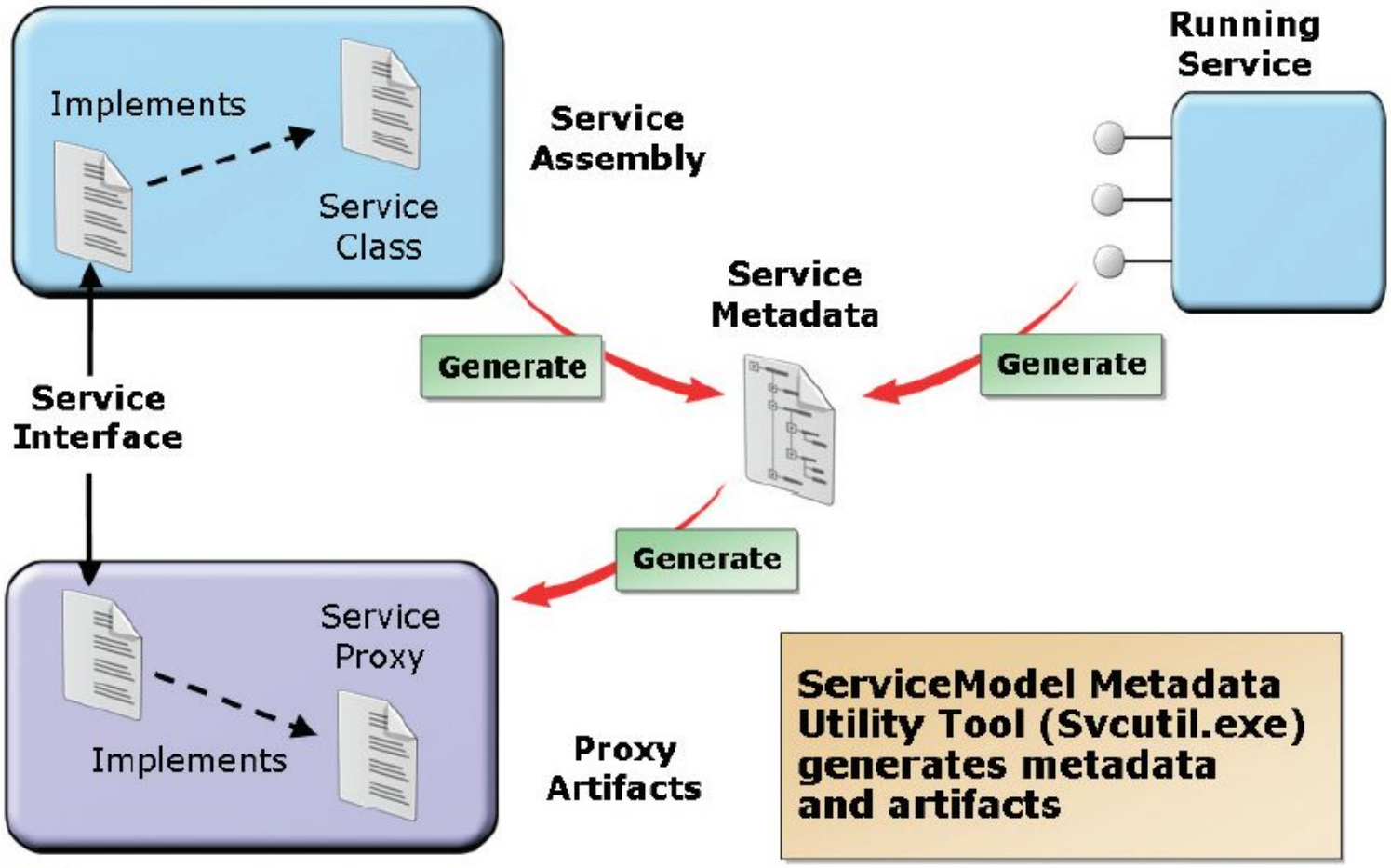


Данные и сообщения

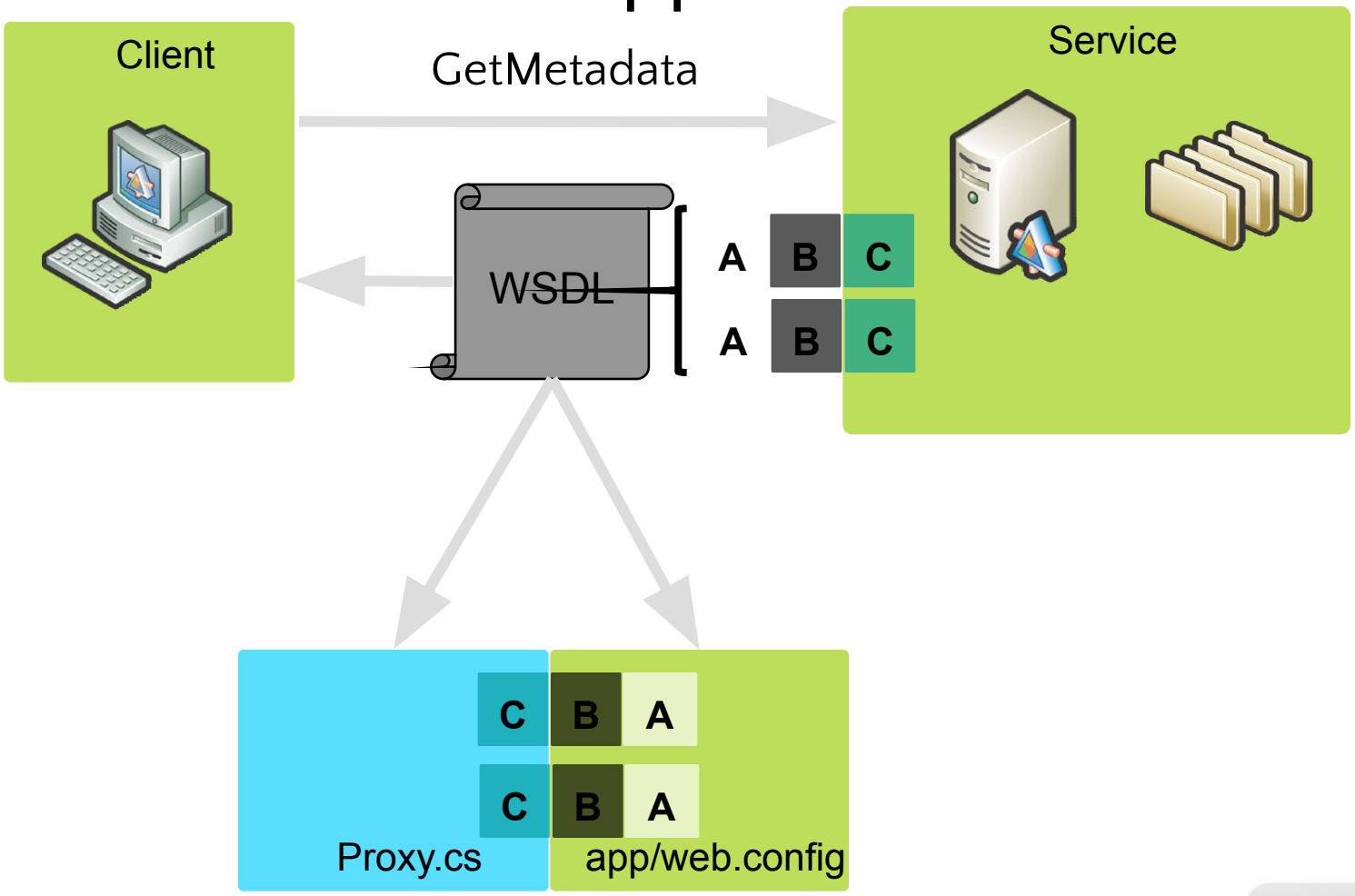
- CLR типы преобразуются к инфонабору XML при сериализации
- WCF позволяет определить собственный способ сериализации
- Состав и структура сообщения должна быть понятна клиенту и серверу
- Контракты данных и контракты сообщений предназначены для контроля утверждений



Контракты, метаданные и артефакты



Описываем точки взаимодействия



Урок 4: Создание простого WCF сервиса

- Определение сервисного контракта и его реализация в классе
- Хостинг сервиса
- Конфигурация сервиса
- Демонстрация: создание простого WCF сервиса

Создание сервиса

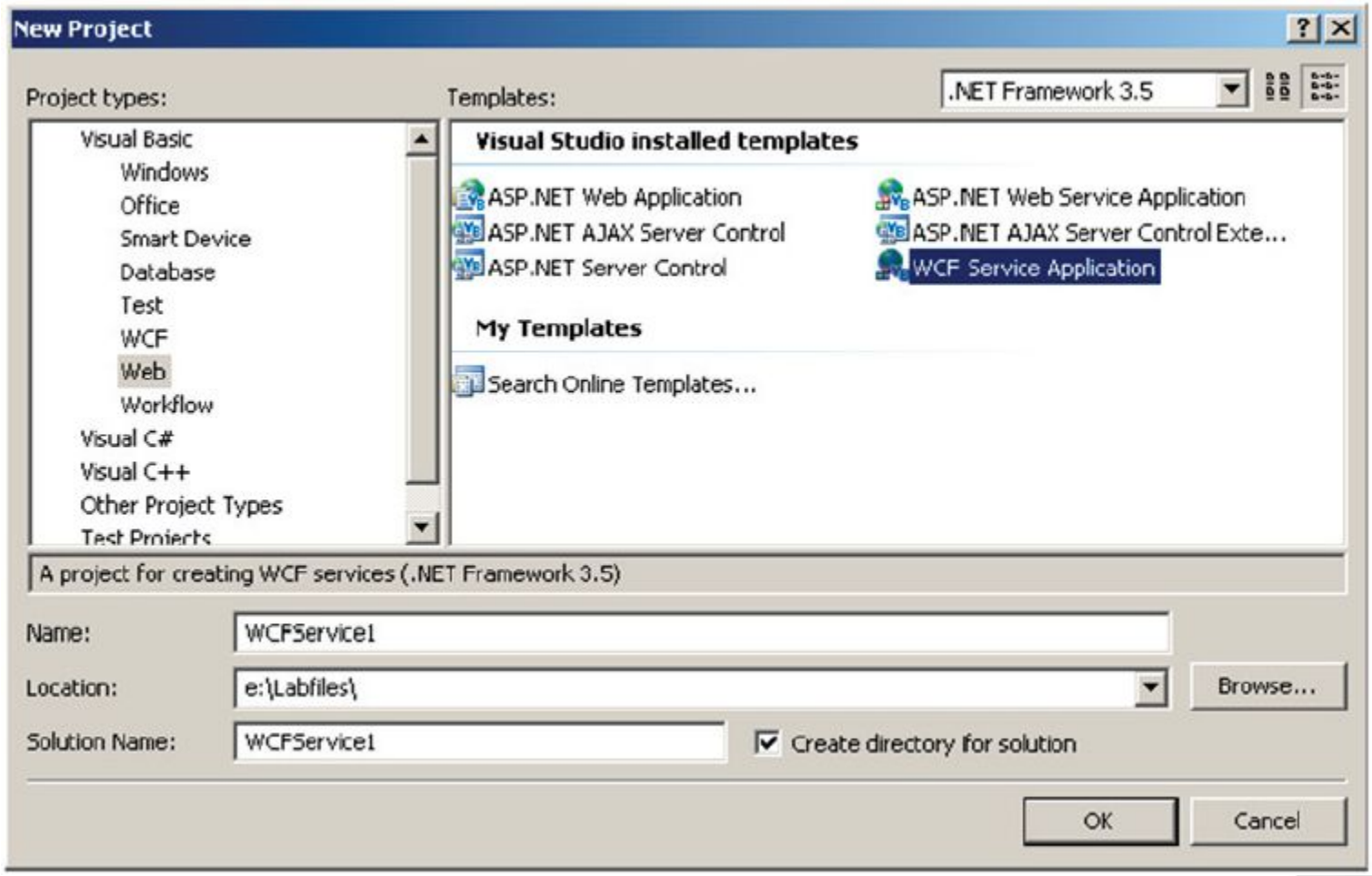
```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    string GetData(int intParam);

    [OperationContract]
    CompositeType GetDataUsingDataContract(CompositeType composite);
}
```

```
public class Service1 : IService1
{
    public string GetData(int intParam)
    {
        return string.Format("You entered: {0}", intParam);
    }

    public CompositeType GetDataUsingDataContract(CompositeType composite)
    {
        return composite;
    }
}
```

Хостинг сервиса



Виды хостинг сервиса

Внутри процесса

```
class HelloHost
{
  static void Main(string[] args)
  {
    ServiceHost host =
      new ServiceHost(typeof(HelloService));
    host.Open();
    // Wait until done accepting connections
    Console.ReadLine();
    host.Close();
  }
}
```

В IIS

http://localhost/HelloService/HelloService.svc

```
<%@ Service Language="C#" Class="HelloService" %>
```

Конфигурация сервиса

```
<configuration>
  <system.serviceModel>
    <services>
      <service name="WCFService1.Service1"
               behaviorConfiguration="WCFService1.Service1Behavior">
        <endpoint address="" binding="wsHttpBinding"
                  contract="WCFService1.IService1"/>
        <endpoint address="mex" binding="mexHttpBinding"
                  contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="WCFService1.Service1Behavior">
         <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

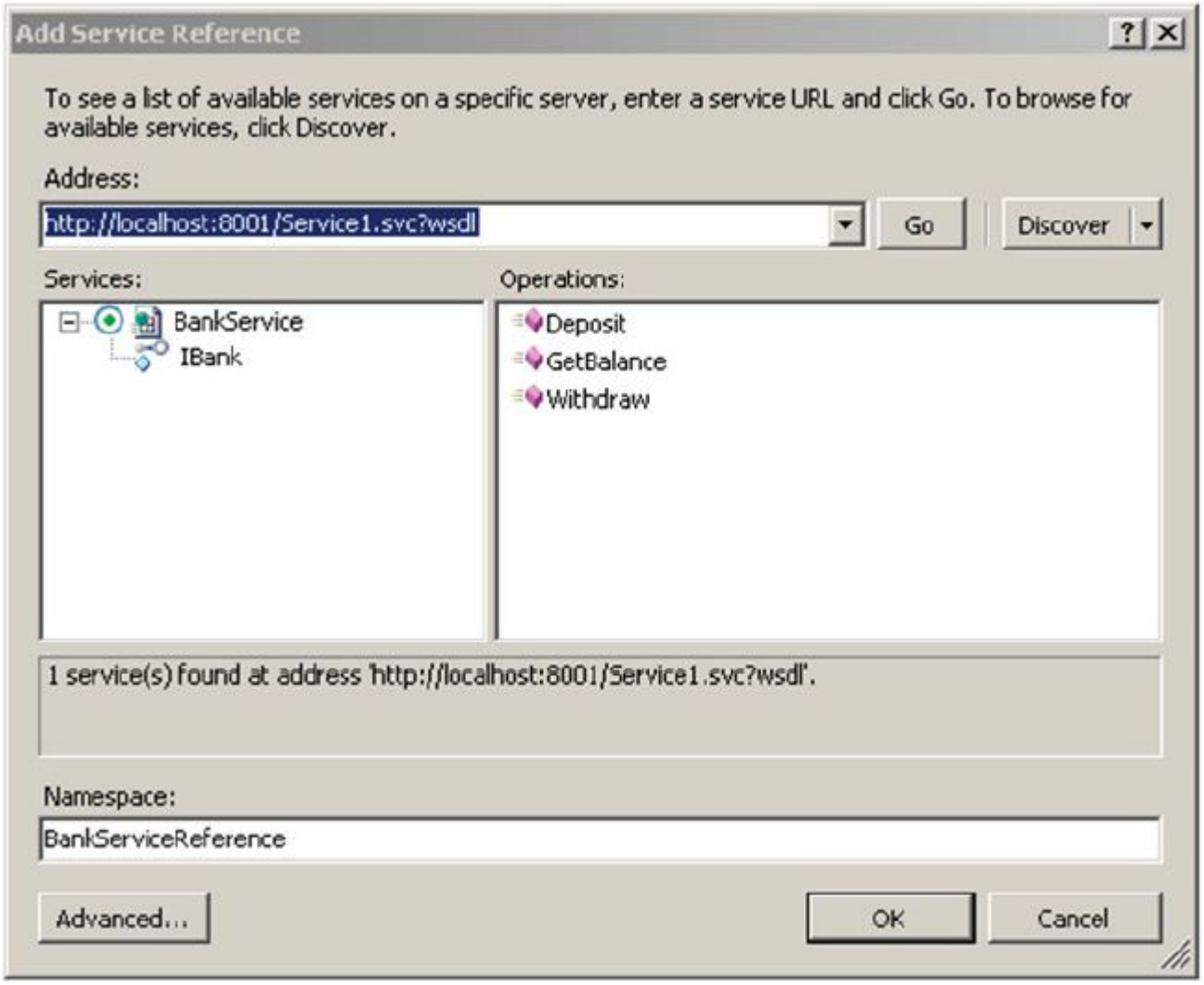
Демонстрация

- Создание простого WCF сервиса

Занятие 5: Создание простого WCF клиента

- Импорт метаданных
- Вызов сервиса используя прокси
- Демонстрация: вызов простого WCF сервиса

Импорт метаданных



Вызов сервиса посредством прокси

- Импорт метаданных создает классы на стороне клиента, которые представляют сервис и сервисный контракт
  - <ServiceName> Клиент это проси объект сервиса
  - <ServiceContractName> - представление типа контракта
- Классы на клиентской стороне, при добавлении ссылки на сервис, создаются в текущем пространстве имен

```
using BankServiceClient.BankServiceReference;
...
IBank proxy = new BankClient("WSHttpBinding_IBank");
...
double balance = proxy.GetBalance(1234);
```

ACS

Реализация клиента

```
class Client
{
    static void Main()
    {
        IHello proxy = ChannelFactory.
       CreateChannel<IHello>(“HelloEndpoint”);
        string r = proxy.Hello(“Beat”);
        Console.WriteLine(r);
    }
}
```

```
<system.serviceModel>
  <client>
    <endpoint configurationName=“HelloEndpoint"
      address="http://localhost/HelloService"
      binding="basicHttpBinding"
      contract="IHello" />
  </client>
</system.serviceModel>
```

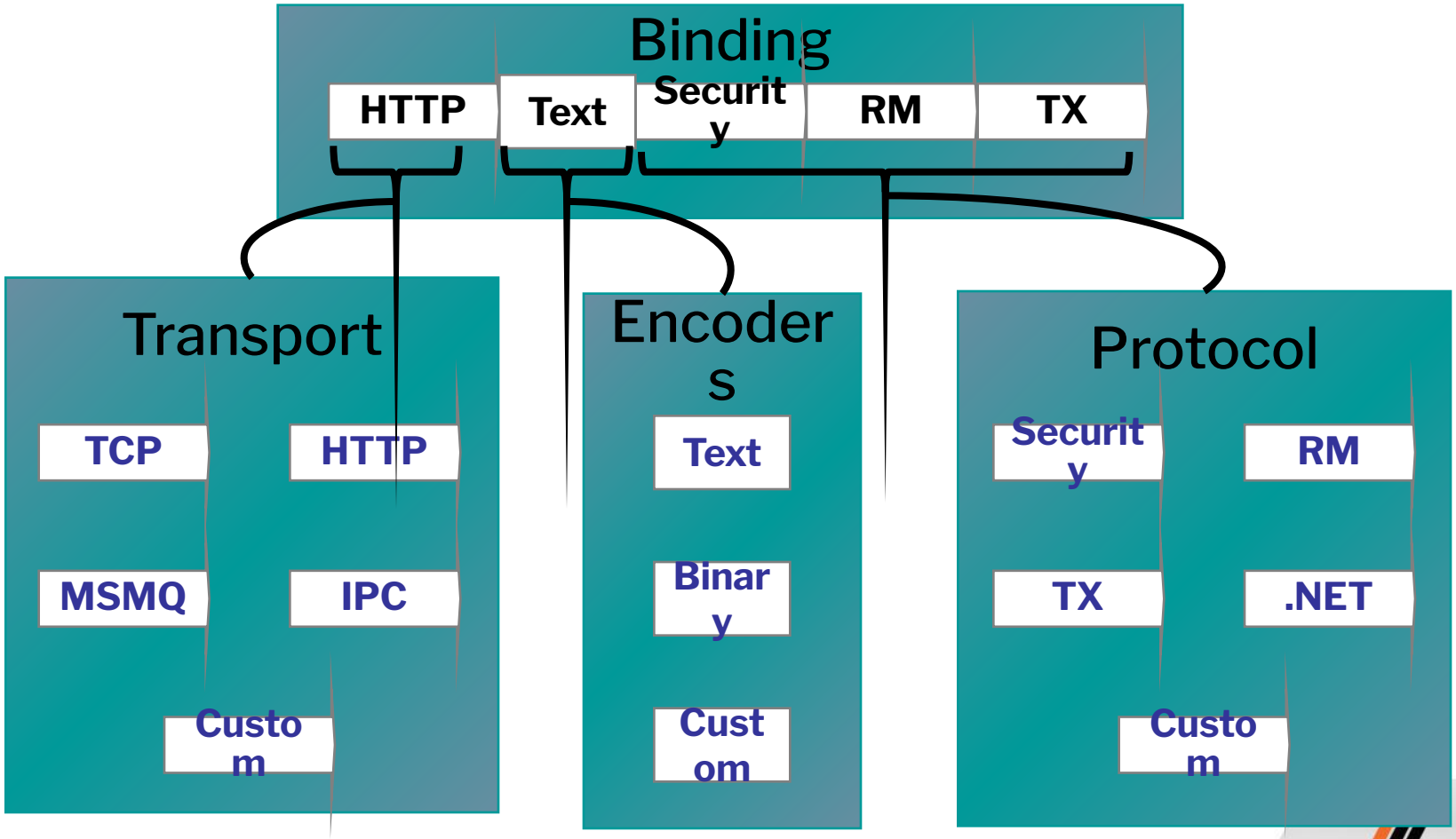
Демонстрация

- Вызов WCF сервиса

Контракты*

- Контракты сервисов и операций
  - Дуплексные, С сохранением сессии
  - Однонаправленные, Запрос/Ответ, Открытие/Закрытие, Сбои
  - Версионность протокола
- Контракты сообщений и данных
  - Схема сообщения
  - Версионность схемы
- Тонкое управление
  - Действие, Направление, Заголовки, Тело сообщения, Обертки, Кодировка

Связывание: вид изнутри*

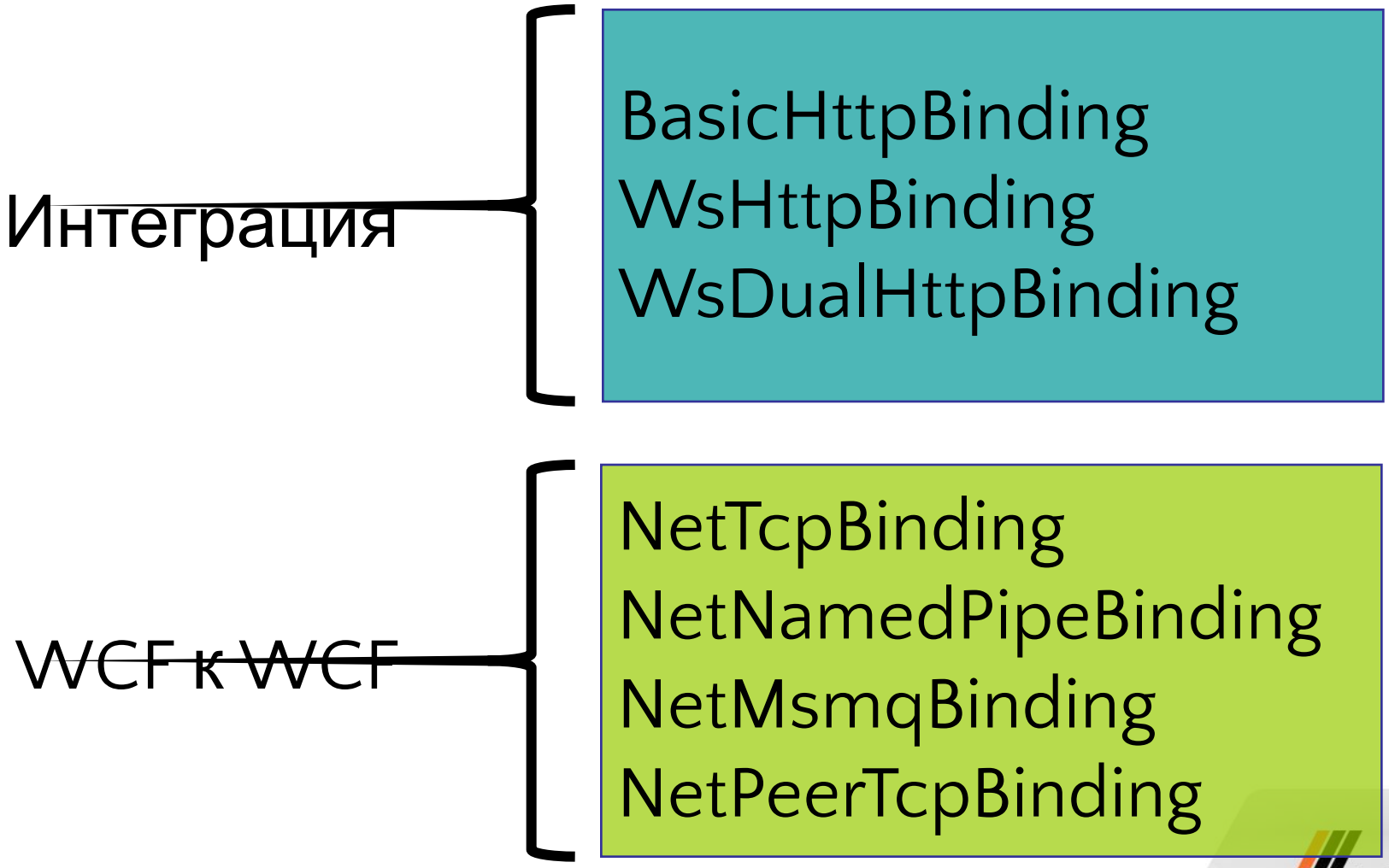


Стандартные наборы

	Interop	Security	Session	Transactions	Duplex	Streaming
BasicHttpBinding	BP 1.1	T				
WsHttpBinding	WS	T | S	X	X		
WsDualHttpBinding	WS	T | S	X	X	X	
NetTcpBinding	.NET	T | S	X	X	X	O
NetNamedPipesBinding	.NET	T | S	X	X	X	O
NetMsmqBinding	.NET	T | S	X	X		
NetPeerTcpBinding	.NET	T | S				

T = Transport Security S = WS-Security | O = One-Way Only

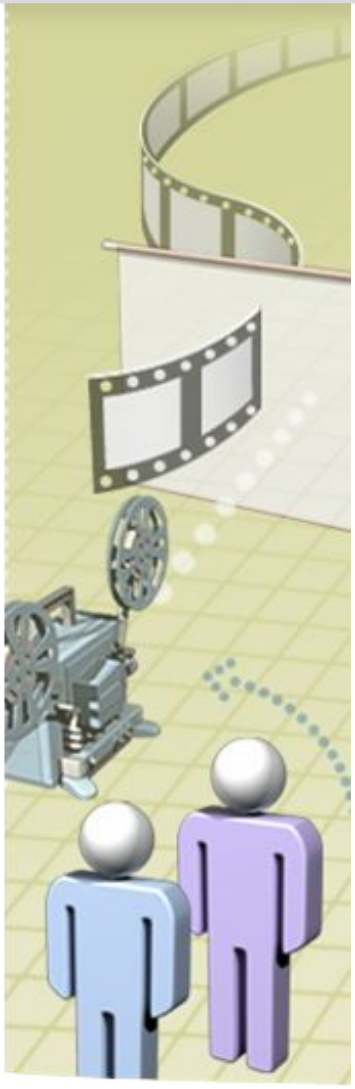
Связывание: взгляд снаружи



Возможности связывания

- Кодировка
  - Text, Binary, Custom
- Транспорт
  - TCP, HTTP, Named Pipes, P2P, MSMQ, Свой собственный
- Безопасность
  - Шифрование, подпись
  - Аутентификация: X509, User/Pwd, Kerberos, SAML, InfoCard, Свой
- Гарантированная доставка
  - Вне зависимости от выбранного транспорта (по порядку, ровно 1 раз)
  - Надежная и незаметная проверка доступности
- Транзакции
  - Общие транзакции для “синхронных” операций
  - Очереди транзакций для “асинхронных” операций

Подведение итогов



В этом модуле рассмотрели:

- Проектирование приложений в стиле SOA
- Архитектура WCF
- Использование интерфейсов как сервисных контрактов.
- Реализация простого WCF сервиса в Visual Studio 2008
- Реализация простого WCF клиента в Visual Studio 2008

Лабораторная работа

- Написать простой сервис единственный метод которого возвращает клиенту строку HelloWorld
- Разместить сервис в консольном приложении
- Предусмотреть возможность получения метаданных сервиса
- Реализовать клиента сервиса

Спасибо за внимание

ЗАО «Эй-Си-Эс»

адрес:620014
Екатеринбург
Радищева, 12
тел: +7 (343) 253-53-00

{ **Спицын Александр Геннадьевич**
e-mail: sag@acs-it.ru,
http://www.acs-it.ru}