

# ARCH03 – Progettare un service layer interoperabile

**Riccardo Golia**

[ricky@aspitalia.com](mailto:ricky@aspitalia.com)

<http://ricky.aspitalia.com>



# Grazie a

5Dlabs.it

Microsoft  
Visual Studio  
with MSDN Subscription

OverNet  
EDUCATION

Sponsor

Windows Internet  
Explorer 9

SOL Server  
LIFESTYLE  
Proactively. Reactively.

Powered by  
Windows Azure

Windows Phone  
Put people first.

icubed

HOEPLI  
INFORMATICA

aspitalia.com

DomusDotNet

DotDotNet



DotNet  
Liguria

DotNET

dotnet  
MARCHE

DotNetSide

GUISA  
Gruppo Utenti Italiani  
Solution Architect

sharepointcommunity.it

UGIdotNET

Visual Basic tips&tricks  
Dal 1996 la community su VB e VB.NET

# Agenda

- Interoperabilità: concetti e problematiche
- Linee guida di progettazione di un service layer:
  - scegliere la tecnologia giusta
  - definire le operazioni necessarie
  - comunicare in modo corretto
  - gestire lo stato interno
  - interagire con tutti
- Esempi vari

# Interoperabilità 1/2

***Interoperabilità: capacità per un sistema di cooperare e scambiare informazioni strutturate con altri sistemi in maniera più o meno completa e priva di errori, con affidabilità e ottimizzazione delle risorse.***

## **Scopo:**

- facilitare l'interazione fra sistemi distribuiti;
- consentire un elevato grado di sinergia, allo scopo di ottenere nuove funzionalità derivanti dalla cooperazione e dall'integrazione di sistemi diversi;
- permettere lo scambio di dati anche fra sistemi non omogenei.

# Interoperabilità 2/2

## Tipologia:

- tecnologica (si tratta di rendere possibile la comunicazione tra sistemi non direttamente compatibili);
- funzionale / semantica (si tratta di estendere le funzionalità di un sistema in modo che possa continuare a funzionare in modo corretto e affidabile).

## Ambito:

- interazione interna al sistema (ovvero interazione tra componenti di uno stesso sistema);
- interazione con l'esterno (ovvero interazione con altri sistemi).

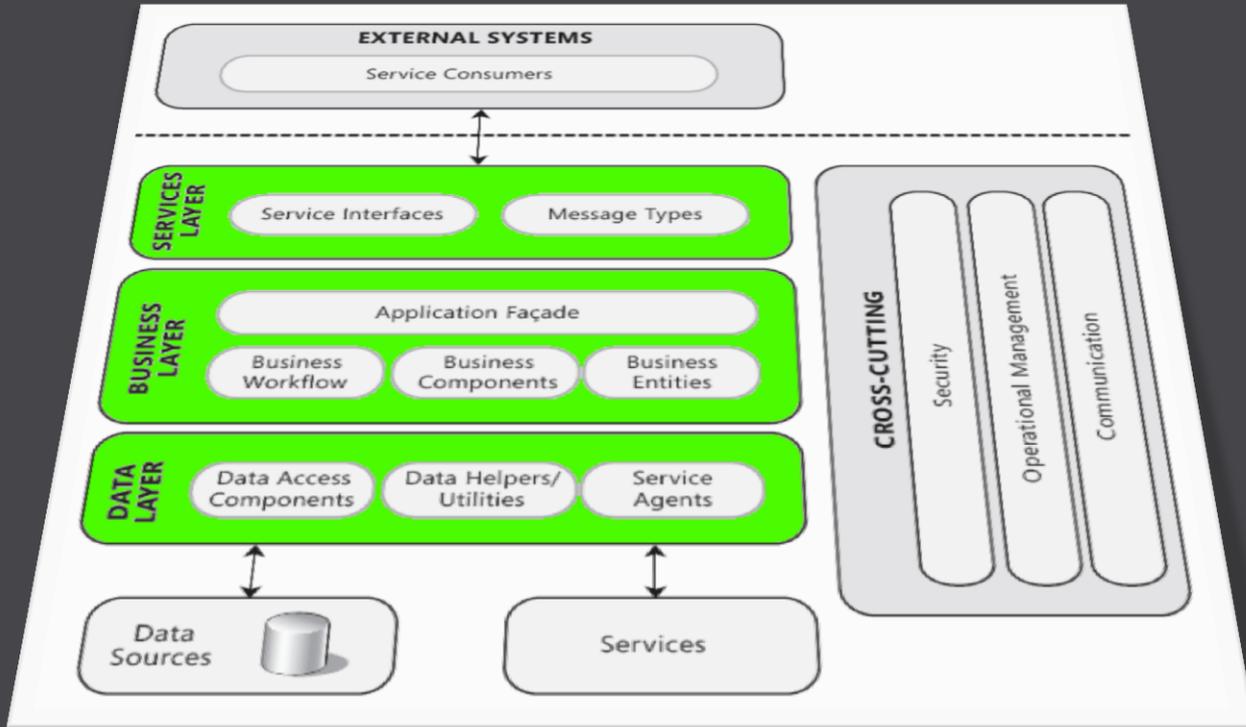
# Interazione interna

- Comunicazione attraverso lo scambio di messaggi tra parti eterogenee dello stesso sistema, con caratteristiche hardware e/o software differenti.
- Interoperabilità per lo più tecnologica.
- Progettazione orientata a superare le peculiarità delle diverse tecnologie in gioco, cercando di minimizzare il lavoro da svolgere, con la possibilità di controllare in toto la comunicazione tra le parti.
- **Esempio:** comunicazione client-server (servizio WCF lato server, client PHP/GUI Flash).

# Interazione esterna

- Comunicazione attraverso lo scambio di messaggi tra uno o più sistemi eterogenei, che internamente utilizzano tecnologie proprietarie.
- Interoperabilità sia tecnologica che funzionale.
- Progettazione orientata a ottenere la massima reperibilità delle funzionalità, la scalabilità e l'integrità/affidabilità dell'interazione, con la possibilità di controllare solo un lato della comunicazione.
- **Esempio:** comunicazione tra servizi B2B (integrazione tra due servizi di pagamento per eseguire transazioni distribuite).

# Service Layer



***Servizio:*** interfaccia pubblica che fornisce accesso a una serie di funzionalità in modo distribuito, esplicitate tramite un contratto descritto dal documento WSDL (Web Services Definition Language).

***Service Layer:*** strato applicativo che include le interfacce dei servizi e i relativi data contract.

# Tecnologie a confronto

- **WCF/SOAP:** servizi con binding *BasicHttpBinding* (messaggi SOAP)
- **WCF/WS-\*:** servizi con binding *WsHttpBinding* (messaggi SOAP)
- **WCF/REST:** servizi con binding *WebHttpBinding* (messaggi JSON)
- **WCF/POX:** binding custom sia lato server che client (messaggi XML)
- **ASP.NET WS/SOAP:** servizi ASMX (messaggi SOAP)
- **ASP.NET HTTP Handler:** response HTTP di tipo XML/JSON/Custom

***Massima interoperabilità: WCF con binding BasicHttpBinding***

# Linee guida di progettazione 1/3

- Per ciascun servizio occorre definire **sempre** una strategia di protezione appropriata (messaggio e/o trasporto).
- L'interfaccia dei servizi non deve essere composta da operazioni troppo semplici e specifiche (*fine-grained*), ma deve essere caratterizzata da operazioni a granularità grossa (*coarse-grained*).
- Il disegno dei servizi deve essere influenzato unicamente dal loro contratto. Questo significa che i servizi vanno sviluppati secondo un approccio incrementale (*side by side*).

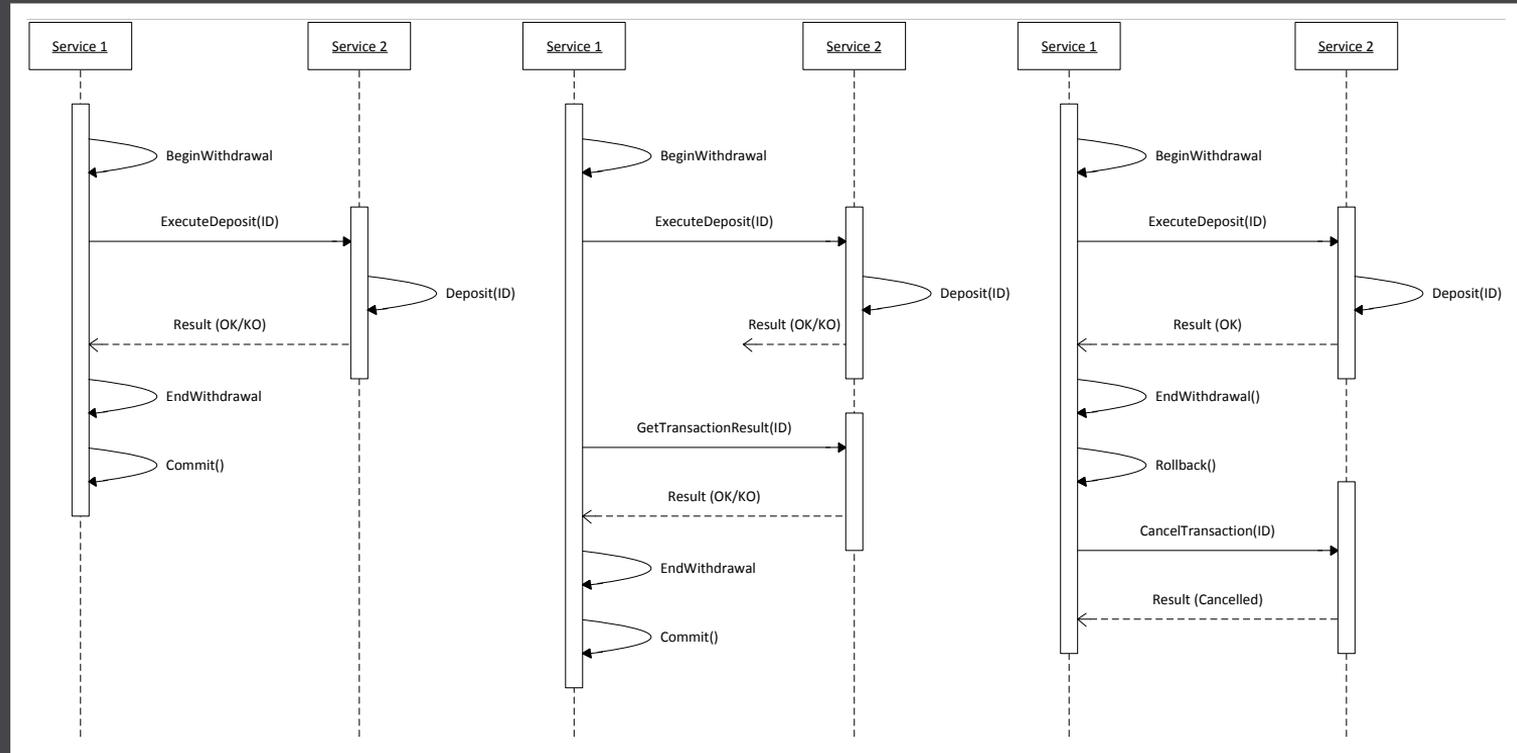
# Linee guida di progettazione 2/3

- I servizi devono essere autonomi e non dipendere dalle caratteristiche e dallo stato del chiamante.
- I dati inviati a un servizio devono essere **sempre** validati, sia a livello sintattico, sia da un punto di vista semantico. La validazione semantica solitamente fa parte delle regole di business.
- Le eccezioni non devono essere mostrate remotamente, ma devono essere gestite in modo opportuno internamente, prevedendo un meccanismo di logging e di feedback.

# Linee guida di progettazione 3/3

- Allo scopo di aumentare la scalabilità, il business layer dietro a un service layer va progettato affinché sia **stateless**, sfruttando eventualmente meccanismi di caching per cortocircuitare alcune funzionalità e/o mantenere dati cross-cutting, validi per ogni richiesta.
- Per garantire l'interoperabilità funzionale, bisogna prevedere operazioni in grado di gestire le situazioni anomale e fornire meccanismi di controllo sullo stato delle interazioni (vedi slide successiva 😊).

# Interoperabilità funzionale



# Interoperabilità tecnologica

- Porre attenzione ai namespace XML nel WSDL. Evitare di lasciare il valore di default *http://tempuri.org*.
- Evitare nel documento WSDL i tag `<xsd:import>`.
- Usare sia tipi semplici (*int, string, enum, ecc.*), sia tipi complessi (data contract). Peraltro definire i data contract in ottica service oriented, non object oriented. In particolare, evitare l'ereditarietà.
- Cercare di evitare "omonimie" per i membri dei data contract (demo).
- Le collezioni di dati vanno esposte come array (WCF ci aiuta).

# Interoperabilità "reale" con WCF 1/3

## PHP5 (NuSOAP)

```
function cashierservice_getsummaryinfo($userData) {
    $client = new nusoap_client(CASHIER_SERVICE,'wsdl','','','');
    $err = $client->getError();
    if ($err) { return null; }

    $request =
        '<v1:GetSummaryInfo ' .
        'xmlns:v1="http://www.worldmatch.com.mt/GNP2/Services/Cashier/V1/" ' .
        'xmlns:data="http://www.worldmatch.com.mt/GNP2/Services/Data/">'.
        '<v1:token>'. $userData->UserToken. '</v1:token>'.
        '<v1:userCode>'. $userData->UserCode. '</v1:userCode>'.
        '</v1:GetSummaryInfo>';

    $msg = $client->serializeEnvelope(
        $request, false, array(), 'document', 'literal');
    $action = 'http://www.worldmatch.com.mt/GNP2/Services/Cashier/V1/'.
        'ICashierService/GetSummaryInfo';

    $result = $client->send($msg, $action, 30, 30);
    $err = $client->getError();

    if (!$client->fault && !$err &&
        is_array($result) && isset($result["GetSummaryInfoResult"]))
        return $result["GetSummaryInfoResult"];
    else
        return null;
}
```

```
$internalAccountsData = cashierservice_getsummaryinfo($user);

if($internalAccountsData != null) {
    if(array_key_exists('AccountBalances',$internalAccountsData) {
        $internalAccountsBalances = $internalAccountsData['AccountBalances'];
        if(array_key_exists('SummaryBalance',$internalAccountsBalances){
            if(!array_key_exists(0,$internalAccountsBalances['SummaryBalance']))
                $internalAccountsBalances['SummaryBalance'] =
                    array($internalAccountsBalances['SummaryBalance']);
            foreach($internalAccountsBalances['SummaryBalance'] as $summaryBalance){
                if(strtolower($summaryBalance['AccountType']) == 'gambling') {
                    $gamblingData['TotalBalance'] = $summaryBalance['TotalBalance'];
                    $gamblingData['CashBalance'] =
                        $summaryBalance['TotalBalance'] - $summaryBalance['BonusBalance'];
                    $gamblingData['BonusBalance'] = $summaryBalance['BonusBalance'];
                    $gamblingData['WithdrawableBalance'] =
                        $summaryBalance['WithdrawableBalance'];
                    updateBalanceTotals($accountBalancesTotals,$gamblingData);
                }
                if(strtolower($summaryBalance['AccountType']) == 'betting'){
                    $bettingData['TotalBalance'] = $summaryBalance['TotalBalance'];
                    $bettingData['CashBalance'] =
                        $summaryBalance['TotalBalance'] - $summaryBalance['BonusBalance'];
                    $bettingData['BonusBalance'] = $summaryBalance['BonusBalance'];
                    $bettingData['WithdrawableBalance'] =
                        $summaryBalance['WithdrawableBalance'];
                    updateBalanceTotals($accountBalancesTotals,$bettingData);
                }
            }
        }
    }
}
```

# Interoperabilità "reale" con WCF 2/3

## ActionScript 3.0 (Flash 10)

```
public class WebServiceHandler extends EventDispatcher {

    public var ws : WebService;
    public var configWsData : Object;

    public function WebServiceHandler(url : String, loc : Array, language : String) {
        ws = new WebService();
        ws.wsdl = url;
        ws.GetConfiguration.addEventListener("result", GetConfiguration_onResult);
        ws.GetConfiguration.addEventListener("fault", GetConfiguration_onFault);
    }

    public function loadWSDL() : void {
        ws.loadWSDL();
        dispatchEvent(new Event("LOAD_WSDL_OK"));
    }

    public function GetConfiguration(user : String, session : String, cfg : int) : void {
        ws.GetConfiguration(user, session, cfg);
    }

    public function GetConfiguration_onResult(evt : ResultEvent) : void {
        configWsData = evt.result;
        dispatchEvent(new Event("GET_CFG_RESULT"));
    }

    public function GetConfiguration_onFault(evt : FaultEvent) : void { ... }
}
```

```
public class Main extends Sprite {

    private var WSInstance : WebServiceHandler;
    private var WSUrl : String = '...';

    public function Main() {
        // ...
        WSInstance = new WebServiceHandler(WSUrl, loc, language);
        WSInstance.addEventListener("LOAD_WSDL_OK", WSDLLoaded, false, 0, true);
        WSInstance.addEventListener("GET_CFG_RESULT", GetConfiguration_onResult, false, 0, true);
        WSInstance.loadWSDL();
    }

    private function WSDLLoaded(e : Event) : void {
        WSInstance.GetConfiguration(userToken, sessionToken, cfg);
    }

    private function GetConfiguration_onResult(e : Event) : void {
        var configurationData : Object = e.target.configWsData;

        if (configurationData.IsValid) {
            wheelsCount = configurationData.Layout.Wheels.length;
            rowsCount = configurationData.Layout.Height;
            isJackpotActive = configurationData.HasJackpot;
            betBoundsWsResult = configurationData.BetAmounts;
            betLinesWsResult = configurationData.Layout.Lines;
            // ...
        }
        else { ... }
    }
}
```

# Interoperabilità "reale" con WCF 3/3

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.worldmatch.com.mt/GNP2/Services/ClassicSlot/V2/Imports">
      <xsd:import schemaLocation="http://my.server.name/ClassicSlot2/ClassicSlotService.svc?xsd=xsd0" namespace="http://www.worldmatch.com.mt/GNP2/Services/ClassicSlot/V2/" />
      <xsd:import schemaLocation="http://my.server.name/ClassicSlot2/ClassicSlotService.svc?xsd=xsd1" namespace="http://schemas.microsoft.com/2003/10/Serialization/" />
      <xsd:import schemaLocation="http://my.server.name/ClassicSlot2/ClassicSlotService.svc?xsd=xsd2" namespace="http://www.worldmatch.com.mt/GNP2/Services/Data/" />
      <xsd:import schemaLocation="http://my.server.name/ClassicSlot2/ClassicSlotService.svc?xsd=xsd3" namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
    </xsd:schema>
  </wsdl:types>
  ...
</wsdl:definitions>
```

```
<%@ServiceHost language="C#" Service="WorldMatch.Gnp2.Services.ClassicSlot.ClassicSlotService"
  Factory="WorldMatch.Gnp2.Services.InlineWsdlserviceHostFactory"%>
```

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
  <wsdl:types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.worldmatch.com.mt/GNP2/Services/ClassicSlot/V2/">
      <xsd:element name="GetConfiguration">...</xsd:element>
      <xsd:element name="GetConfigurationResponse">...</xsd:element>
      <xsd:element name="GetBalance">...</xsd:element>
      <xsd:element name="GetBalanceResponse">...</xsd:element>
    </xsd:schema>
  </wsdl:types>
  ...
</wsdl:definitions>
```

# Q&A

- Materiale su:  
<http://www.communitydays.it/>
- Approfondimenti:  
Service layer: <http://aspitalia.com/fn>  
Interop. & WCF: <http://aspitalia.com/4x>