

Web Service e Microsoft .NET Framework

Silvano Coriani (scoriani@microsoft.com)

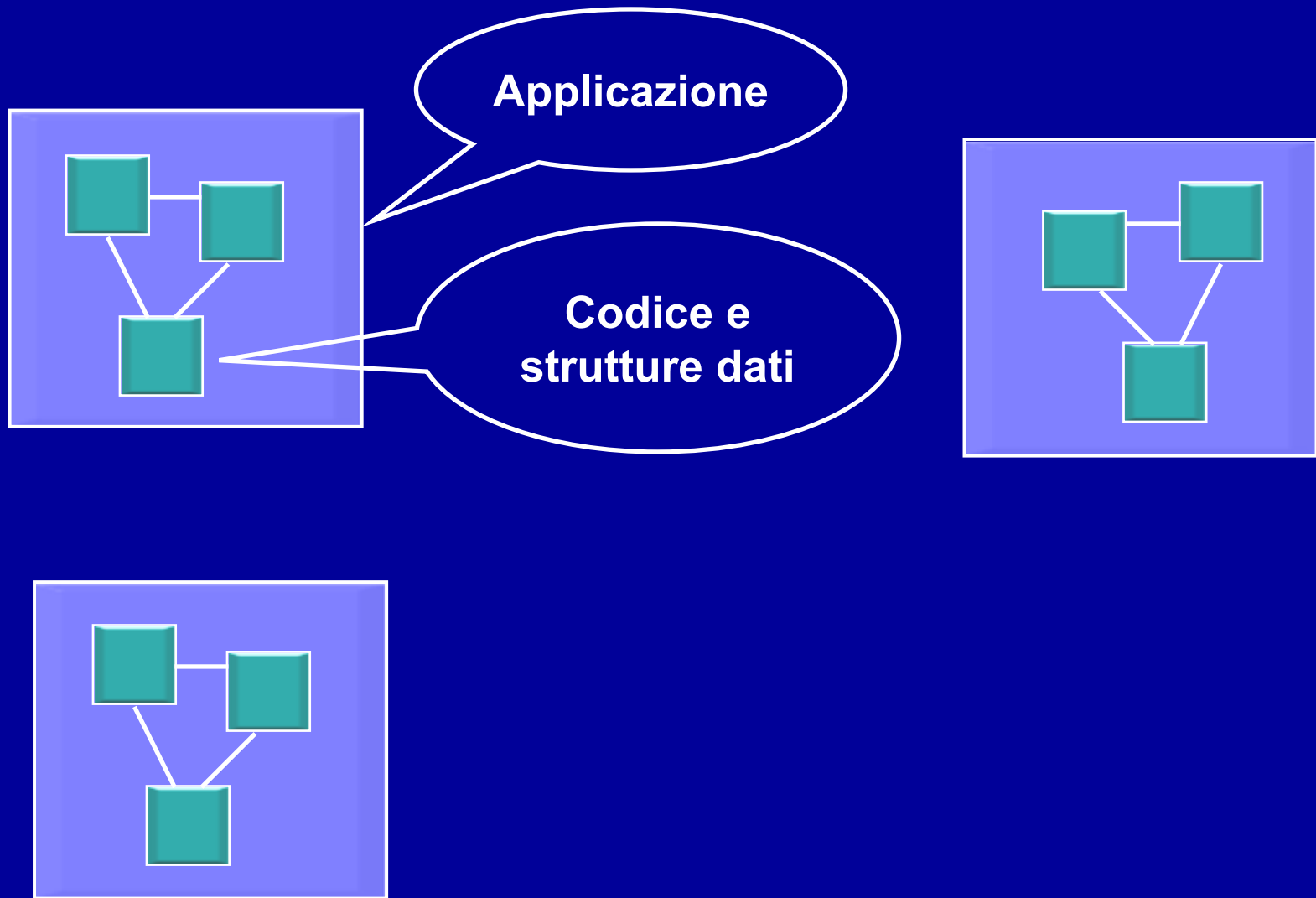
Academic Developer Evangelist
Developer & Platform Evangelism
Microsoft

Microsoft Certified Trainer
Microsoft Certified Solution Developer
Microsoft Certified Application Developer
Microsoft Certified System Engineer + Internet
Microsoft Certified DataBase Administrator

Agenda

- **Base Recall**
- **Web Service**
 - **Tecnologia:**
 - SOAP
 - WSDL
 - WSA
- **Web Service ASP.NET**
 - Architettura
 - ASMX (WebServiceHandler) + Classe sul server
 - Classe proxy sul client
- **.NET Remoting**
 - Architettura
- **Web Service o .NET Remoting?**

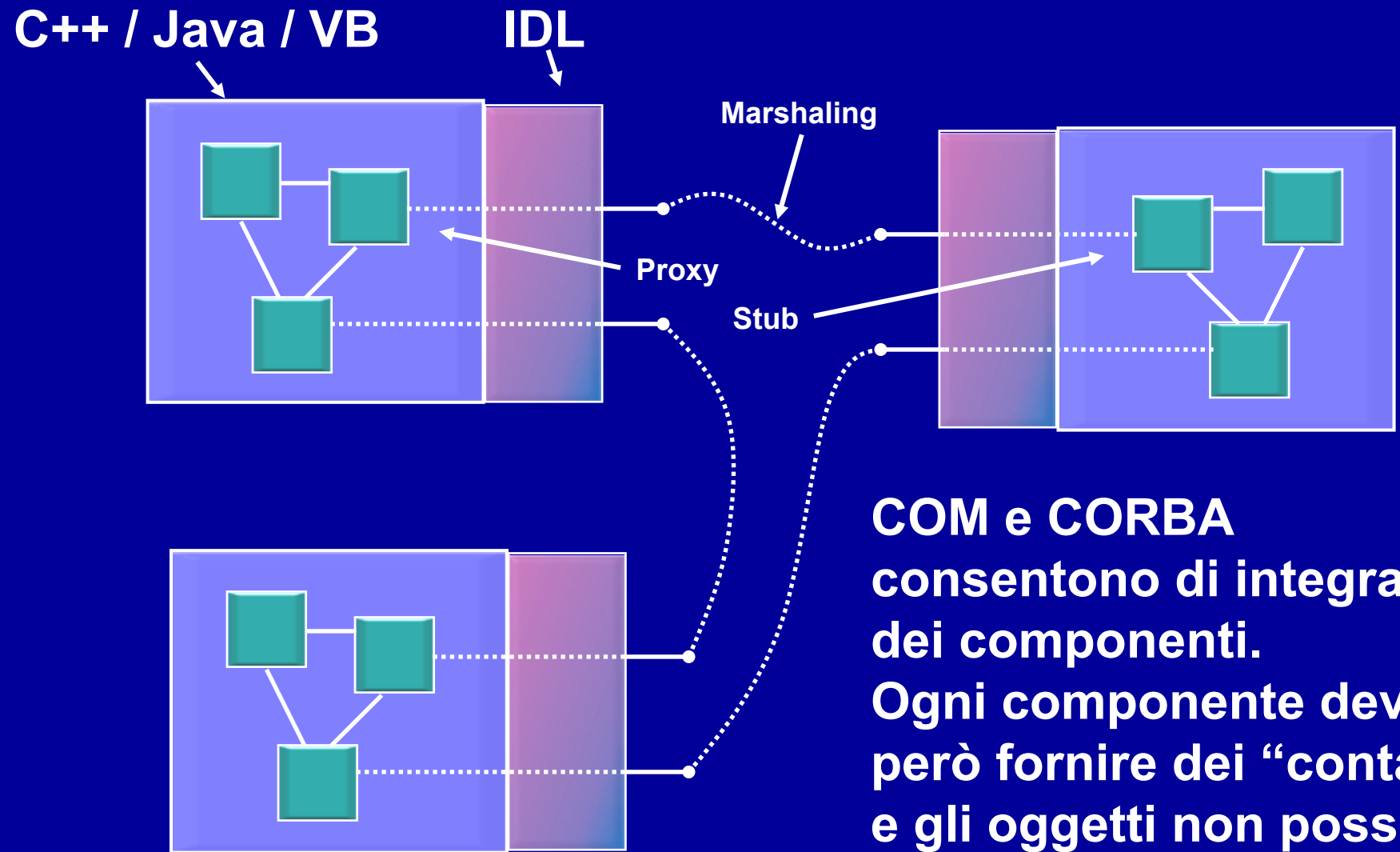
Applicazioni monolitiche



Le Dynamic Linked Library

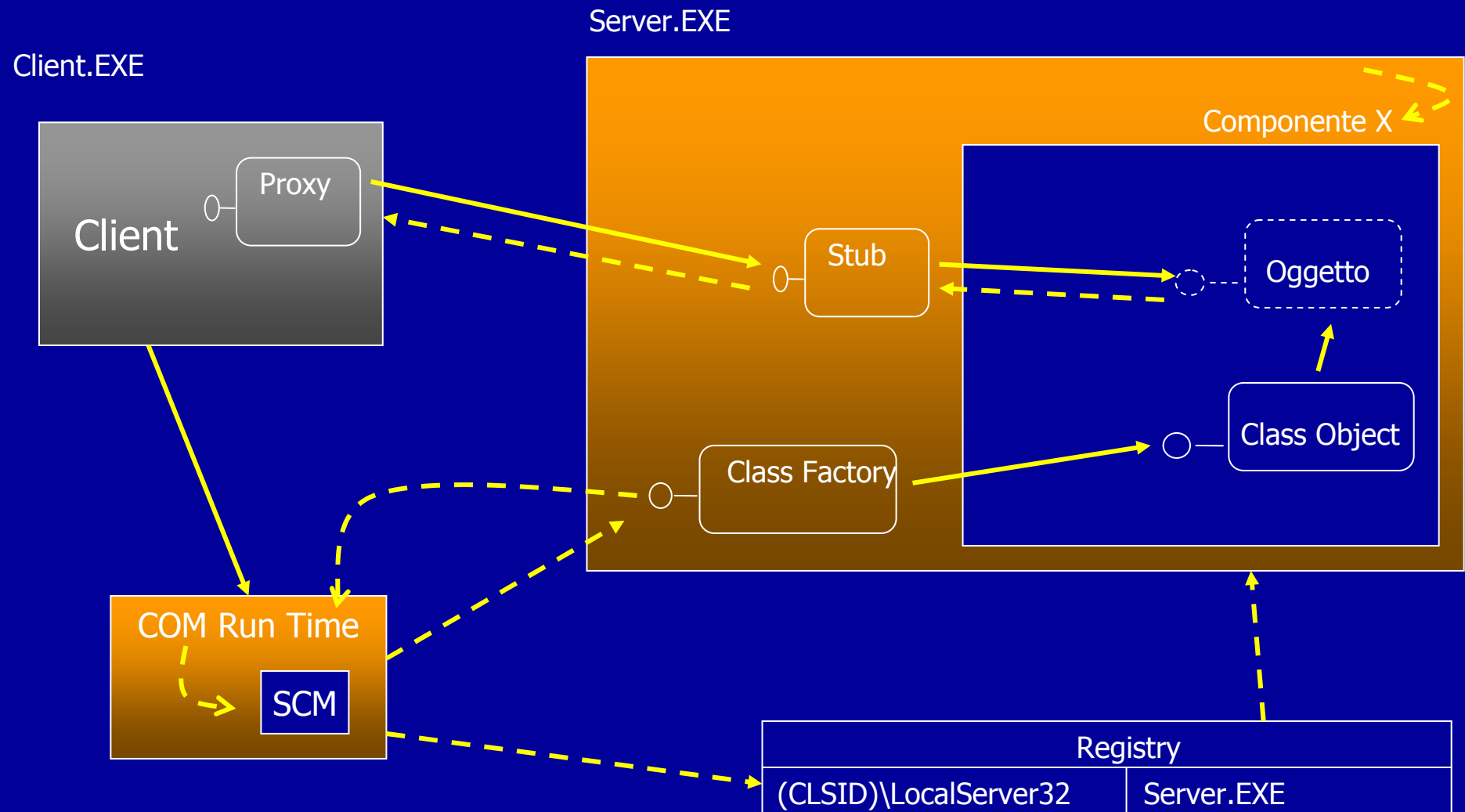
- **Prima forma di "componentizzazione"**
- **Contengono funzioni compilate in codice nativo (codice macchina)**
- **Espongono le funzionalità attraverso il concetto di "export file"**
- **Dipendenza dalla posizione fisica sul disco**
- **Type system non integrato**
 - **Nessuna possibilità controllo sull'invocazione di una funzione**
 - **Potenziale incompatibilità con alcuni linguaggi**
- **Nessun controllo sul cambiamento di versione**
 - **DLL Hell**

L'era di COM / CORBA



COM e CORBA
consentono di integrare
dei componenti.
Ogni componente deve
però fornire dei “contatti”
e gli oggetti non possono
interagire direttamente

Come funziona COM / DCOM



I problemi di COM / DCOM / CORBA

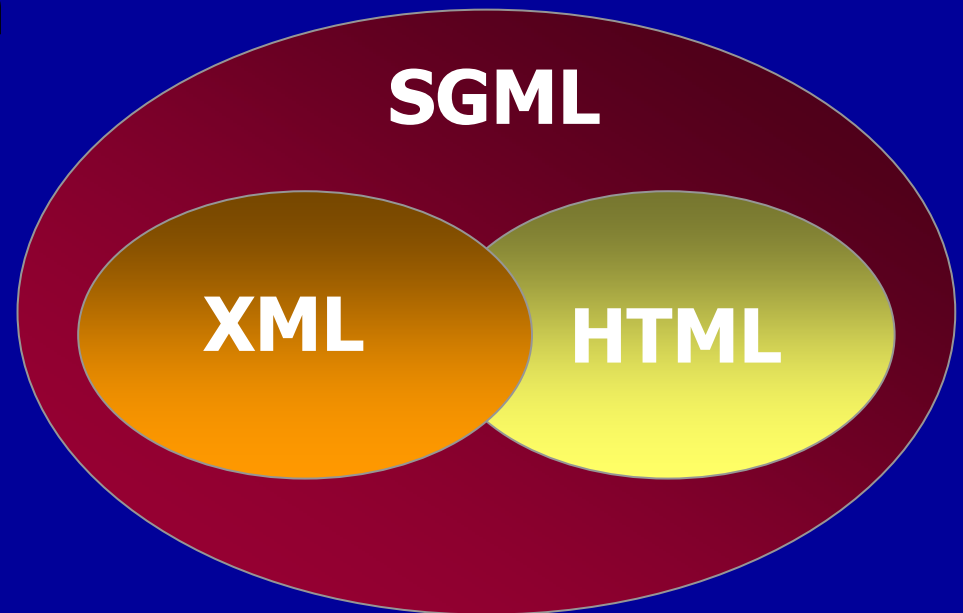
- **Dipendenza da un servizio di “registry”**
- **Problematiche di versioning dei componenti**
 - **DLL Hell**
- **Anche se potenzialmente indipendenti dal linguaggio di programmazione, dipendono dai diversi “run-time” per la comunicazione / conversione dei tipi di dati**
 - **VB <> C++ <> Java**
- **Nelle applicazioni distribuite, problematiche legate alla gestione di porte TCP dinamiche (communication endpoint)**
- **Incompatibilità tra le differenti implementazioni**

Protocolli

- **TCP/IP**
 - **Protocollo utilizzato per tutte le comunicazioni su Internet**
 - **Protocollo a due livelli: TCP e IP**
- **Hypertext Transfer Protocol (HTTP)**
 - **Protocollo per lo scambio di file**
 - **Utilizzato dai browser per accedere alle risorse**
 - **Si basa su TCP/IP**
 - **E' Stateless**

Cosa è XML

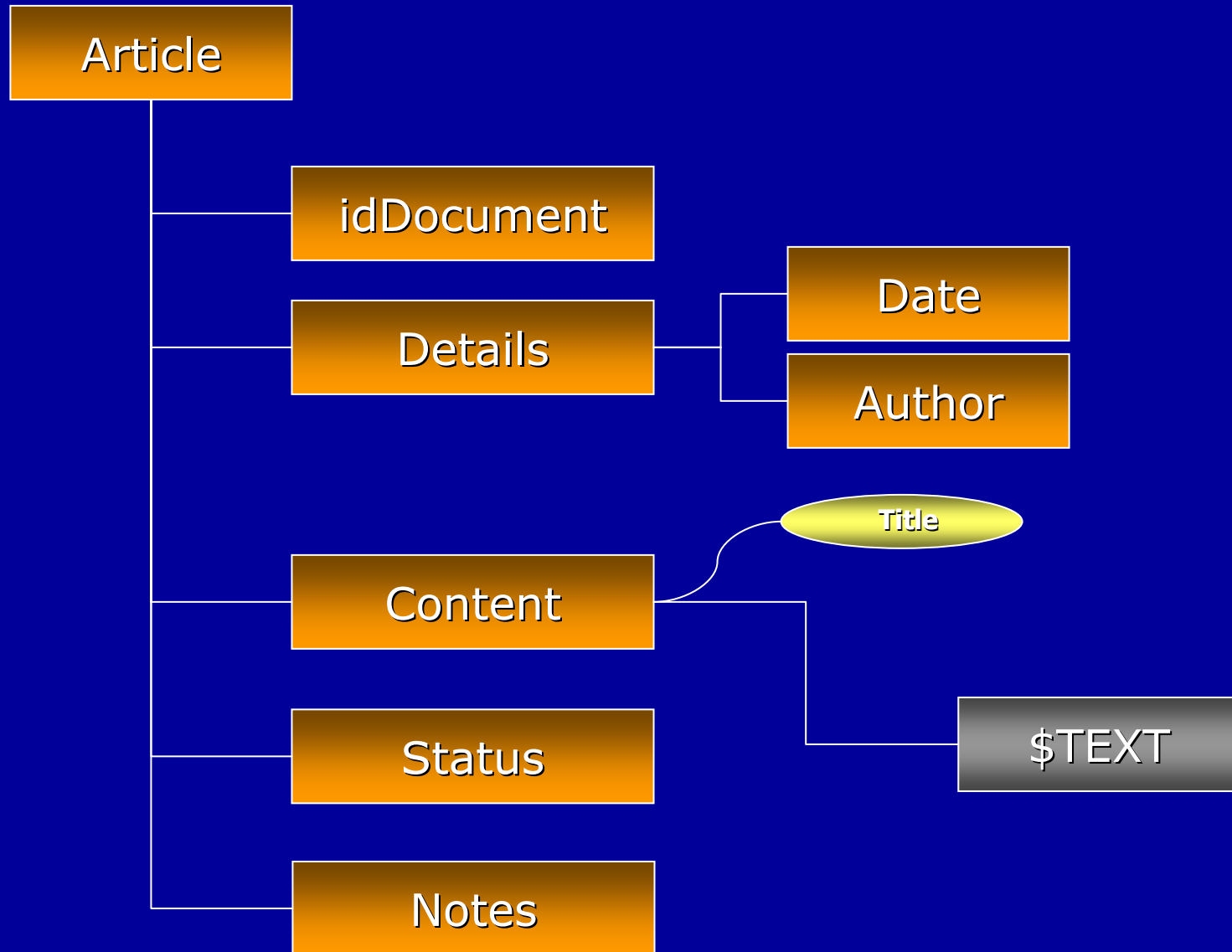
- XML = eXtensible Markup Language
- È un **METALINGUAGGIO** che ci permette di rappresentare informazioni in un formato testuale trasmissibile via Internet tra sistemi diversi.



XML Well-Formed

```
<?xml version="1.0"?>
<Article>
  <idDocument>DOC346</idDocument>
  <Details>
    <Date>14/02/2003</Date>
    <Author>paolo@devleap.it</Author>
  </Details>
  <Content Title="XML & Web Service">
    Documento che descrive XML ed i Web Service da un punto
    di vista teorico ma anche e soprattutto pratico.
  </Content>
  <Status>Closed</Status>
  <Notes />
</Article>
```

In-Memory Tree



Namespace

<?xml version="1.0"?>

<ordine

xmlns:prodotto="http://www.azienda.com/"

xmlns:cliente="http://www.cliente.com/">

<prodotto:codice>1321</prodotto:codice>

<cliente:codice>PP2301</cliente:codice>

</ordine>

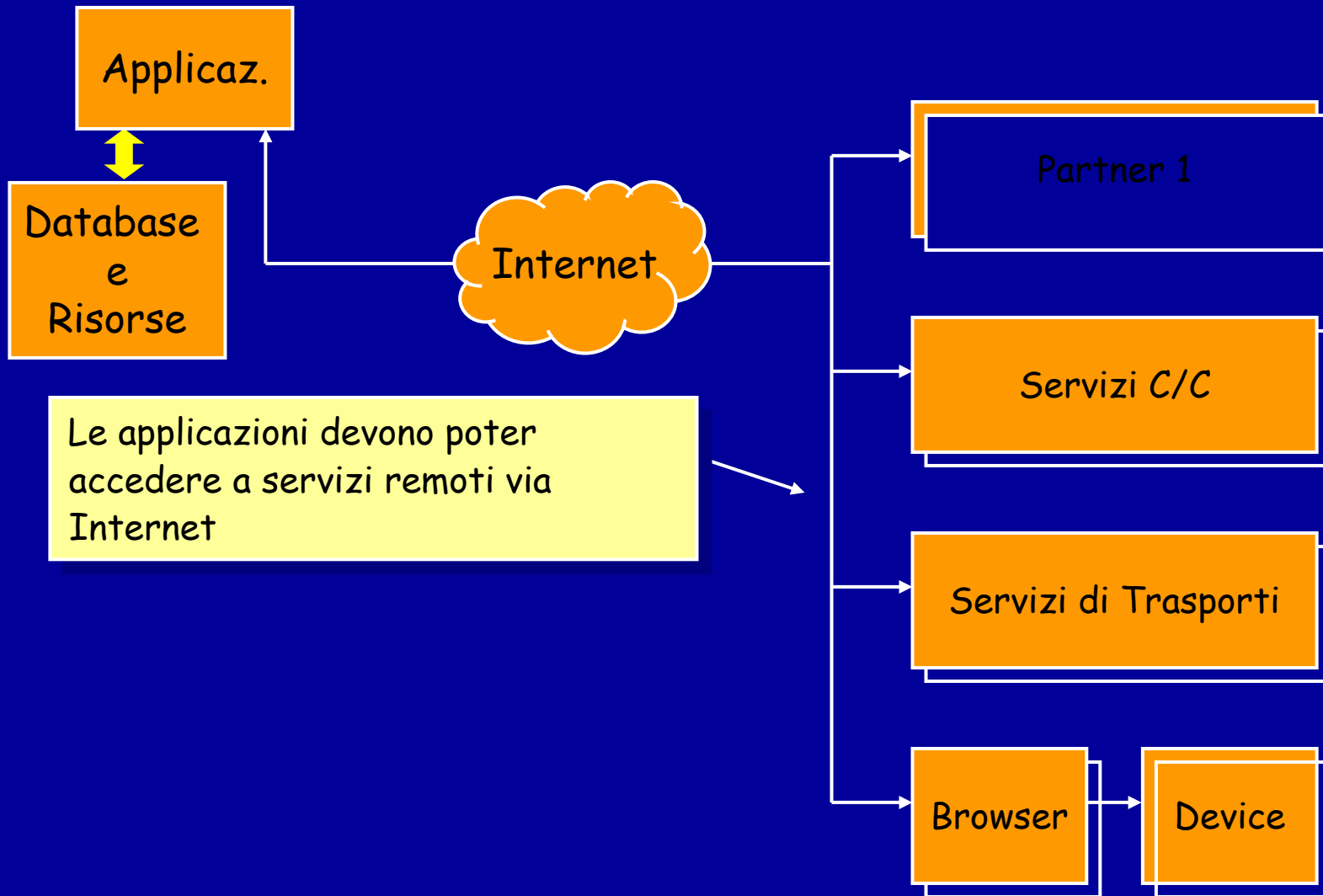
XML Schema Definition

- **Si indica spesso come XSD 1.0**
- **È una grammatica XML per descrivere altre grammatiche XML**
- **Permette di stabilire**
 - **i nomi dei tag e degli attributi**
 - **la loro posizione nel documento**
 - **i tipi di valori che possono contenere (int, dateTime, string, ecc.)**
 - **vincoli e relazioni sui dati**
 - **ecc.**

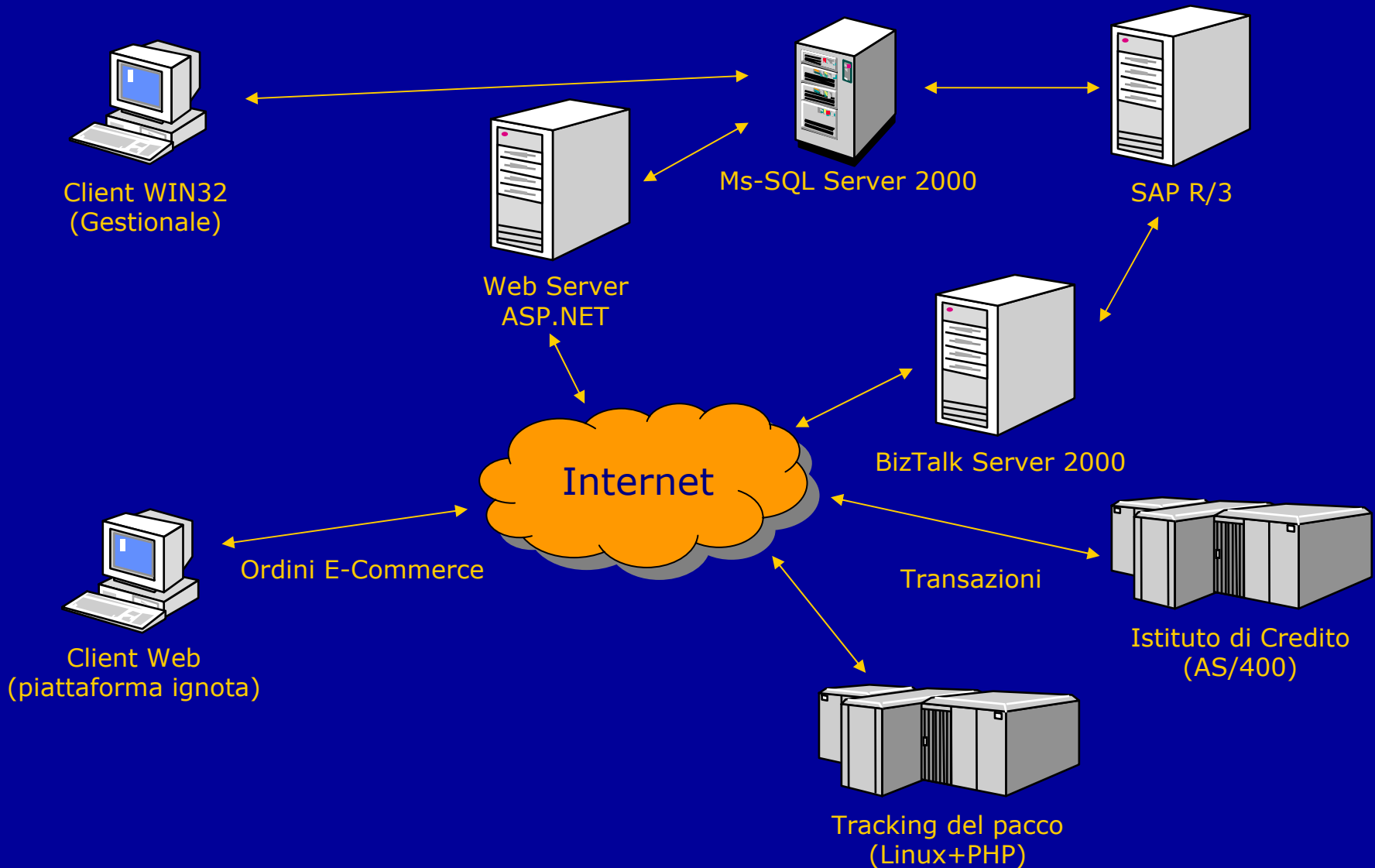
Applicazioni distribuite

- **Sempre più spesso abbiamo l'esigenza di**
 - Distribuire i carichi di lavoro (ridondanza e scalabilità)
 - Isolare attività costose (tempo/hardware)
 - Demandare attività a terzi (partner/service provider)
- **Ma senza perdere il controllo!**
 - Transazioni distribuite
 - Attività coordinate e sincronizzate
 - Mantenibilità e aggiornabilità

Le Applicazioni di oggi



Piattaforme differenti



Piattaforme differenti

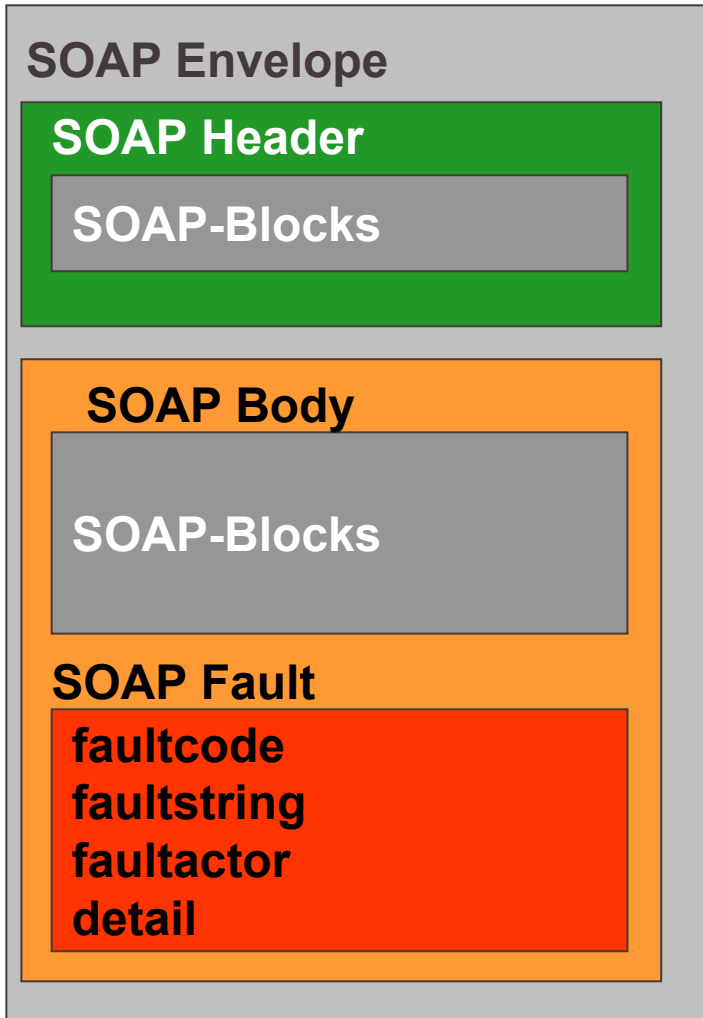
- **Come possiamo integrarle?**
- **Come evitare di inventare ogni volta la ruota?!**
- **Come riutilizzare il lavoro svolto ?**
 - **Aumentando la produttività**
 - **Diminuendo i costi**
- **Serve una lingua “franca” che tutti possano parlare e capire**

SOAP

- **Simple Object Access Protocol**
- **È una grammatica XML**
- **Pensata per descrivere richieste e risposte di servizi remoti**
- **È indipendente da**
 - **Piattaforma**
 - **Protocollo di trasporto**
 - **API o Runtime**
- **È XML !**

SOAP 1.1 - Schema

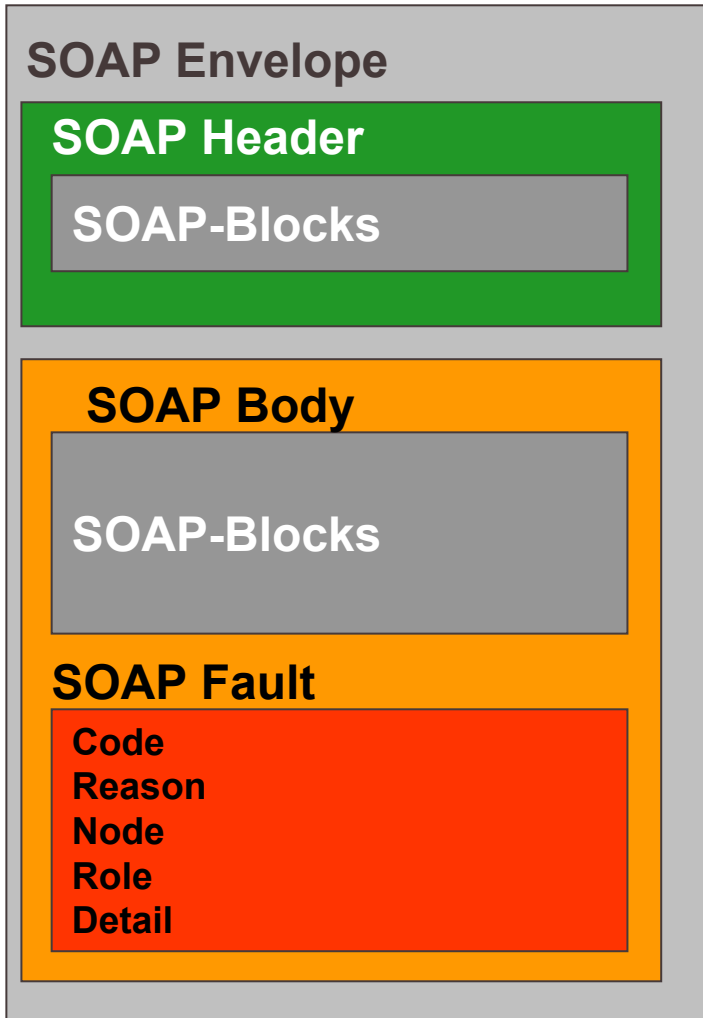
SOAP Message



```
<soap:Envelope>  
  <soap:Header>  
    <Intestazione1>...</Intestazione1>  
    ...  
    <IntestazioneN>...</IntestazioneN>  
  </soap:Header>  
  <soap:Body>  
    <Metodo>  
      <Parametro1></Parametro1>  
      ...  
      <ParametroN></ParametroN>  
    </Metodo>  
    <soap:Fault>  
      <faultcode />  
      <faultstring />  
      <faultactor />  
      <detail>  
        ...  
      </detail>  
    </soap:Fault>  
  </soap:Body>  
</soap:Envelope>
```

SOAP 1.2 - Schema

SOAP Message



```
<soap:Envelope>
  <soap:Header>
    <Intestazione1>...</Intestazione1>
    ...
    <IntestazioneN>...</IntestazioneN>
  </soap:Header>
  <soap:Body>
    <Metodo>
      <Parametro1></Parametro1>
      ...
      <ParametroN></ParametroN>
    </Metodo>
    <soap:Fault>
      <Code>...</Code>
      <Reason xml:lang=".." />
      <Node />
      <Role />
      <Detail />
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

HTTP

- **Protocollo stateless**
 - **Ad ogni richiesta**
 - **Corrisponde una risposta**
 - **E si chiude la connessione**

HTTP Request

GET /foo.htm HTTP/1.1

0

POST /foo.cgi HTTP/1.1
Content-Type: text/plain
Content-Length: 13

IdRecord=1

HTTP Response

200 OK
Content-Type: text/plain
Content-Length: xxx

Risultato della richiesta

SOAP + HTTP = Web Service

- **Se veicoliamo messaggi SOAP tramite HTTP otteniamo un Web Service**
- **Ma potremmo utilizzare qualsiasi altro protocollo in grado di trasmettere XML (SMTP, FTP, MSMQ ...)**
- **Avremmo un servizio remoto con accesso via SOAP, non un *Web Service*, ma non cambia poi molto ...**

Versioni SOAP a confronto

- **Namespace XML:**

- SOAP 1.1 : <http://schemas.xmlsoap.org/soap/envelope/>
- SOAP 1.2 : <http://www.w3.org/2002/06/soap-envelope>

- **Content-type:**

- SOAP 1.1 : text/xml
- SOAP 1.2 : application/soap+xml

- **HTTP Headers:**

- SOAP 1.1 : SOAPAction
- SOAP 1.2 : action (opzionale)

- **Notifiche errori:**

- Soap Fault differente. Più completo in SOAP 1.2.

Chi definisce il contenuto del Body?

- **Due formati di Body:**
 - **Document:** pensato per gestire documenti, non ci sono regole precostituite
 - **RPC:** la struttura è definita nella sezione 7 della specifica SOAP 1.1
- **Due stili di rappresentazione dei parametri:**
 - **Literal:** uno schema XSD descrive la struttura dei parametri
 - **Encoded:** parametri sotto forma di elementi come indicato nella sezione 5 della specifica SOAP 1.1
- **Per ottenere due formati principali:**
 - **document/literal**
 - **RPC/encoded**

WSDL

- **Web Service Description Language**
- **Grammatica XML che descrive:**
 - Servizi esposti
 - Protocolli da utilizzare per raggiungerli
 - Porte utilizzabili (operazioni esposte)
 - Formato dei messaggi (input/output)
 - Tipi di dati (secondo XSD)
- **Se ho il WSDL di uno o più servizi so come rivolgermi a loro**

WSA

- **XML Web Services Architecture**
- **Si tratta di protocolli infrastrutturali (cioè non legati ad una particolare applicazione)**
- **Sono basati sui SOAP:Header => rispettano gli standard**
- **Sono concordati tra le aziende (Microsoft, IBM, Verisign, BEA, ecc.)**
- **Alcuni in fase di approvazione dal consorzio OASIS**

Alcune specifiche di WSA

- **WS-Security: sicurezza dei messaggi SOAP**
- **WS-Routing: instradamento di messaggi SOAP**
- **WS-Transaction: transazioni distribuite**
- **WS-Coordination: coordinamento di attività distribuite su più WS, affianca WS-Transaction**
- **WS-Attachments: si basa su DIME per fornire messaggi SOAP con allegati binari**
- **WS-***

WS-Security

- **Autenticazione:**
 - Username e Password
 - Certificato X.509
 - Token custom
- **Firma digitale:**
 - Basata su Username e Password
 - Con certificato X.509
 - Con token custom
- **Encryption:**
 - Chiavi simmetriche (shared key)
 - Chiavi asimmetriche (certificati X.509 per es.)
 - Custom

WS-Security e HTTP

- **WS-Security** consente di avere

NON RIPUDIABILITÀ DEI MESSAGGI !

- **In modo indipendente**
 - Dal protocollo di trasporto
 - Dall'applicazione che ne fa uso
- **Ci sono già implementazioni multiplatforma:**
 - Microsoft WSE 2.0
 - IBM WSTK 3.3.2
 - Newtelligence Ws-Security 1.0

.NET Framework 1.1

- **Il supporto ai Web Service è nativo**
 - **System.Web.Services (Host ASP.NET)**
- **.NET Remoting**
 - Non è pensato per realizzare Web Service
 - Solo in alcuni casi e se configurato opportunamente espone dei Web Service ...
ne parliamo dopo ...

Web Service ASP.NET

- **Il motore di ASP.NET associa all'estensione ASMX un Handler apposito**
- **Alle spalle dei file ASMX ci saranno classi .NET che espongono dei metodi**
- **Questi metodi, se decorati con l'attributo WebMethod, saranno invocabili via SOAP+HTTP**

Realizzare un Web Service .NET

- Documento .ASMX pubblicato sul Web
- Namespace:
 System.Web.Services

- Direttiva:

<%@ WebService Language="???" Class="???" %>

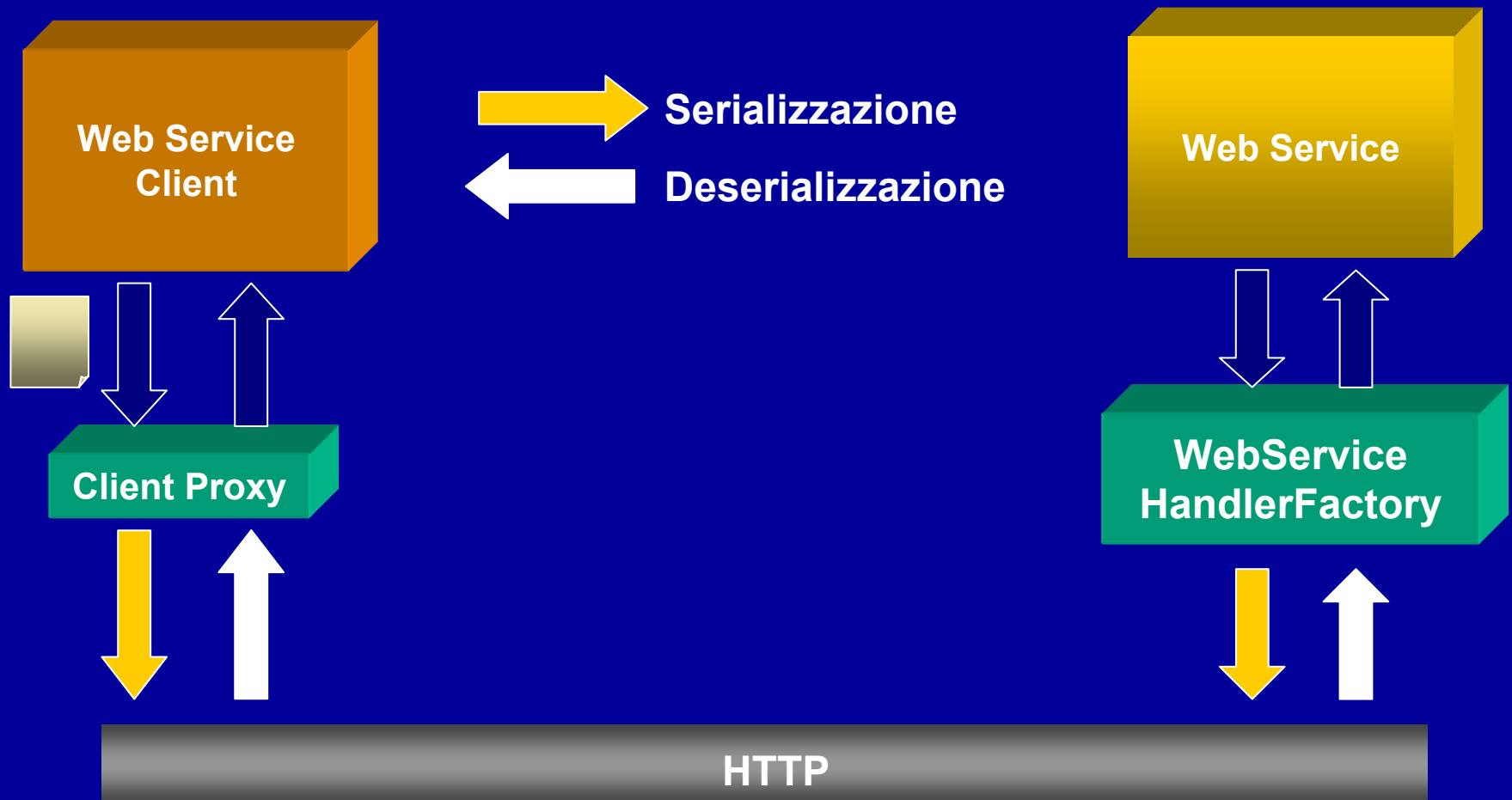
- Associare alla classe dei WebMethod:

[WebMethod()]
public String LeggiMessaggio() { ... }

Realizzare un Client .NET

- **WSDL.EXE**
 - Tool a riga di comando che genera una classe proxy verso il Web Service
- **Creare il codice client che utilizza il Web Service tramite la classe proxy**
- **Compilare il client con un riferimento al codice (o all'assembly) della classe proxy**

Architettura WS ASP.NET



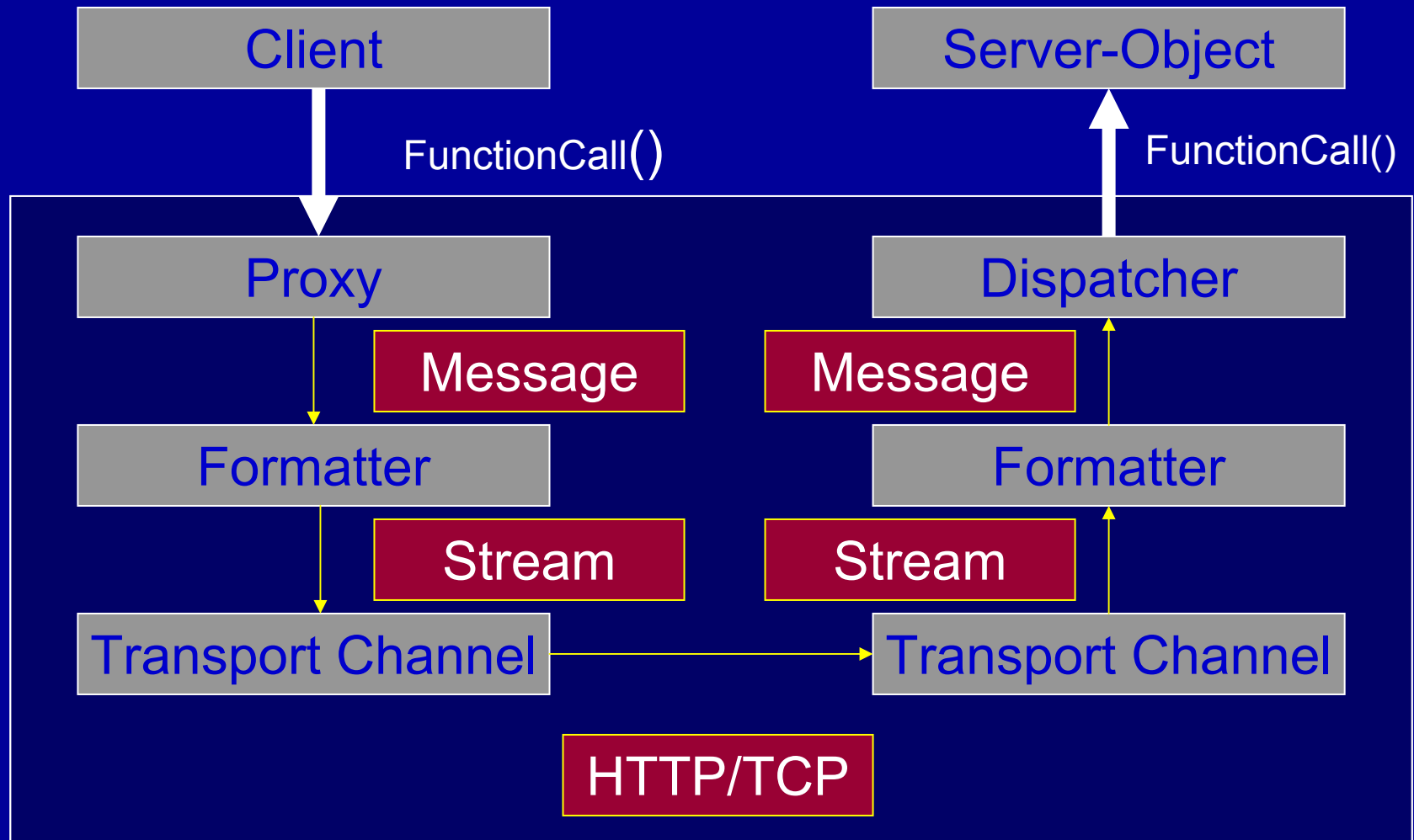
WS con Visual Studio .NET

- **Server:**
 - Template di progetto Web Service
 - Definire la classe
 - Scrivere i WebMethod
 - Compilare
- **Client:**
 - Add Web Reference (ASMX/WSDL)
 - Codice che usa l'oggetto come se fosse locale
- **Tutti i meccanismi di serializzazione/deserializzazione sono per noi trasparenti!**

.NET Remoting

- **È simile ad un meccanismo di chiamata tra processi**
 - **Intercettazione chiamata**
 - **Marshalling dei parametri**
 - **Ricostruzione stack di chiamata**
- **Meccanismo per attraversare il confine di Application Domain**
 - **AppDomain diversi su stesso PC**
 - **AppDomain su PC diversi**

Architettura di .NET Remoting



Differenze e similitudini

- **Portabilità**
 - Web Service sono multi-piattaforma
 - Remoting è solo .NET/CLI
(anche SSCLI e Mono hanno un'implementazione)
- **Remoting supporta CAO**
(Client-activated object)
- **Serializzazione**
 - .NET Remoting usa IFormatter
 - Web Service ASP.NET usano XmlSerializer

Differenze e similitudini

- **Hosting**
 - **.NET Remoting: richiede un processo host da creare oppure IIS**
 - **Web Service ASP.NET: usano IIS o comunque il motore di ASP.NET**
- **In certe configurazioni, Remoting e Web Service parlano la stessa lingua**
In configurazione HTTP/SOAP, Remoting comunica come Web Service usando SOAP RPC/encoded

Che cosa restituisce un Web Service ?

- **Tutto ciò che è serializzabile in XML**
- **Per esempio:**
 - **Tipi base .NET**
 - **Classi e Strutture**
 - **XmlDocument e XmlNode**
 - **Array**
 - **Collezioni**
 - **DataSet (typed o untyped)**
- **Vengono definiti tramite XSD nel documento WSDL che descrive il servizio**

Cosa può restituire .NET Remoting ?

- **Potenzialmente qualsiasi oggetto .NET**
- **Attenzione però**
 - **I tipi che derivano da MarshalByRefObject sono restituiti “per riferimento”**
 - **I tipi che hanno l’attributo [Serializable] sono restituiti per valore (MarshalByValue)**
 - **I tipi che non sono né l’uno né l’altro non sono utilizzabili**
 - **I tipi value base di .NET sono tutti [Serializable]**

Ancora sul confronto

- **Sul fronte prestazionale**
 - **Per ora .NET Remoting è spesso + veloce**
 - **Ma non è detto che un domani WS lo raggiunga o lo superi**
 - **Ricordiamoci che ora quando pensiamo a XML possiamo concentrarci sull'InfoSet e non sul documento fisico**
- **Nessuno dei due ha nativamente meccanismi per la gestione della security**

Quindi chi scegliere?

- **Devi esporre un servizio fruibile da qualsiasi piattaforma?**
 - **Realizza un Web Service ASP.NET**
- **Devi integrare differenti AppDomain .NET fornendo oggetti/servizi remoti?**
 - **Sfrutta .NET Remoting**
- **Sono tecnologie complementari e non alternative**

Link utili

- **MSDN Studenti**
 - <http://www.microsoft.com/italy/msdn/studenti>
- **MSDN Academic Alliance**
 - <http://www.msndaa.net>
- **MSDN Online**
 - <http://msdn.microsoft.com/webservices>
- **GotDotNET**
 - <http://www.gotdotnet.com>
- **ASP.NET**
 - <http://www.asp.net>
- **Windows Forms**
 - <http://www.windowsforms.net>
- **DevLeap**
 - <http://www.devleap.it>
- **UgiDotNet**
 - <http://www.ugidotnet.org>

