



Web Services e Enterprise Service Bus

Alma Mater Studiorum - Università di Bologna
CdS Laurea Magistrale in Ingegneria Informatica
I Ciclo - A.A. 2022/2023

Insegnamento di Sistemi Distribuiti M

08 – Web Services (WS) e Cenni su Enterprise Service Bus (ESB)

Docente: Luca Foschini
luca.foschini@unibo.it

<http://lia.disi.unibo.it/Courses/sd2223-info/>
<https://www.unibo.it/sitoweb/luca.foschini>



Differenza tra Servizi Web e Web Services

da una parte, gli **utenti ottengono servizi Web** in un sistema integrato facendo computazione via Web, **C2B** dall'altra, i **Web services** sono una *specifica diversa e precisa* per ottenere a livello Web tutto quello che si può ottenere a livello di linguaggio di programmazione e di computazione tipicamente **B2B**

Stiamo considerando tutto l'ambito HTML-compatibile in più assumendo di usare strumenti che tengano in conto alcune estensioni come XML (eXtensible Markup Language) Prospettiva di massima apertura

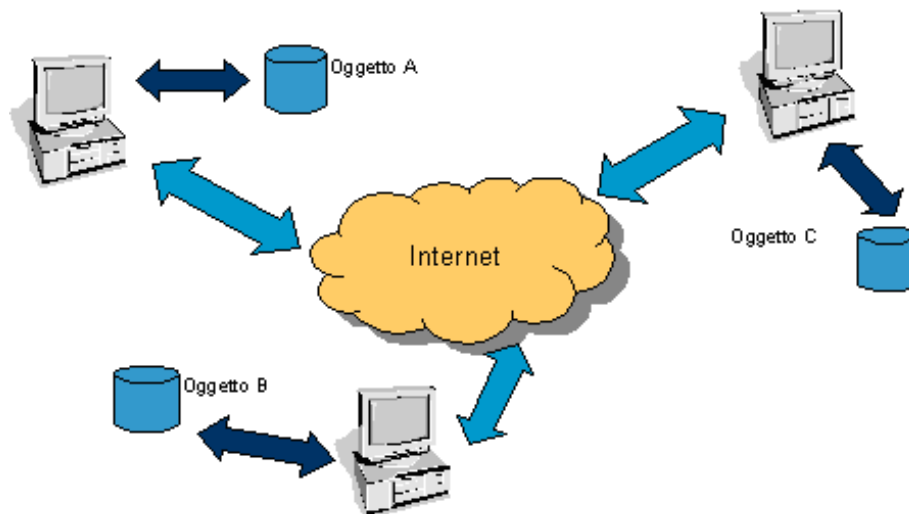
MIDDLEWARE nel supporto a sistemi

MIDDLEWARE E COMPONENTI:

direzioni di evoluzione e stato dell'arte

Fornitura di servizi in ambiente distribuito pervasivo e ubiquo

Sempre più servizi intesi come sistemi o framework per la *integrazione* e *composizione* di oggetti distribuiti

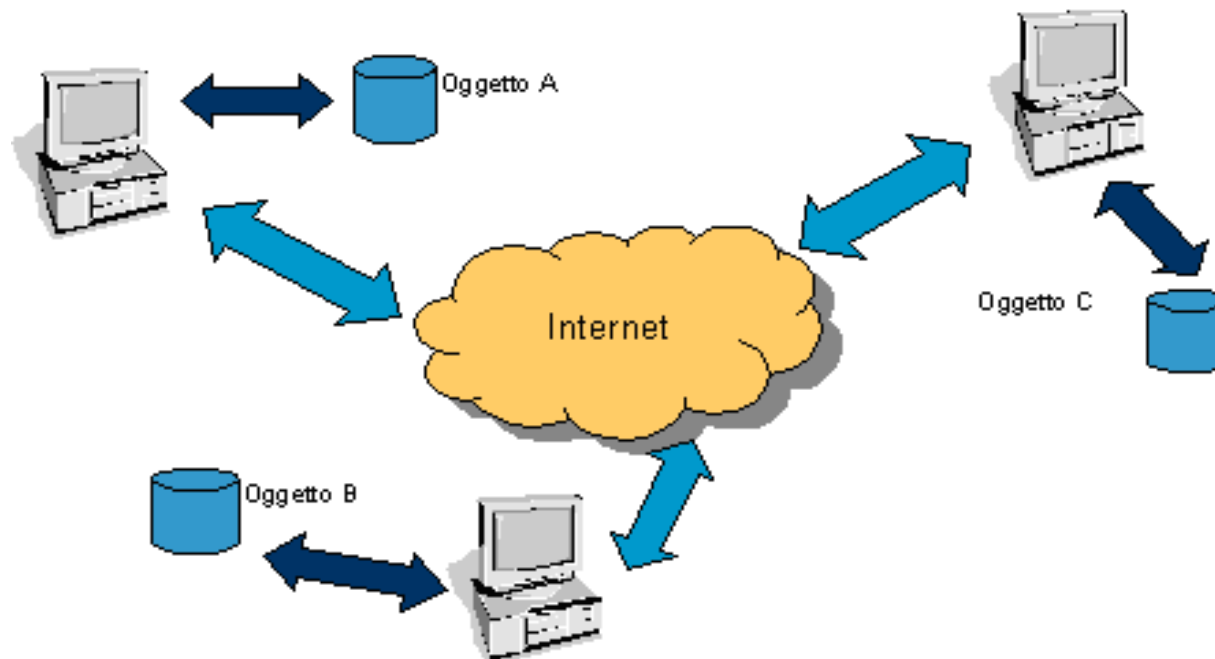


Con garanzia di
superare eterogeneità e
mantenere i corretti livelli di sicurezza

MIDDLEWARE diffusi basati

Modello Cliente-Servitore

- RPC (Remote Procedure Call)
- Attualmente: CORBA, DCOM





MIDDLEWARE ad OGGETTI

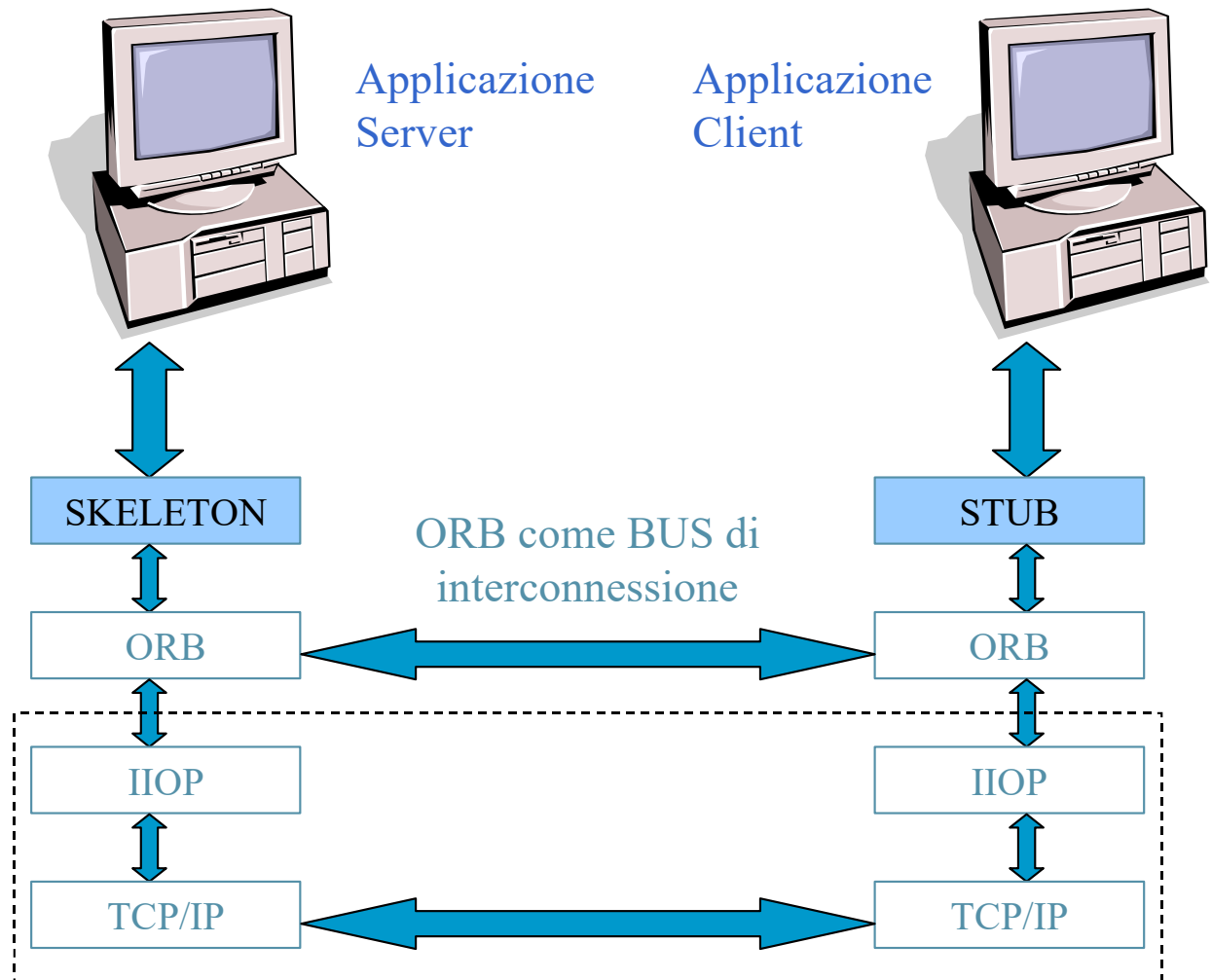
CORBA

Interazione **sincrona**

Ambiente **standard**
basato su
componenti
multilinguaggio e
multi-architettura
eterogenei

Integrazione di
sistemi e strumenti
esistenti

Apertura verso
ambienti esterni
anche legacy





MIDDLEWARE ad OGGETTI

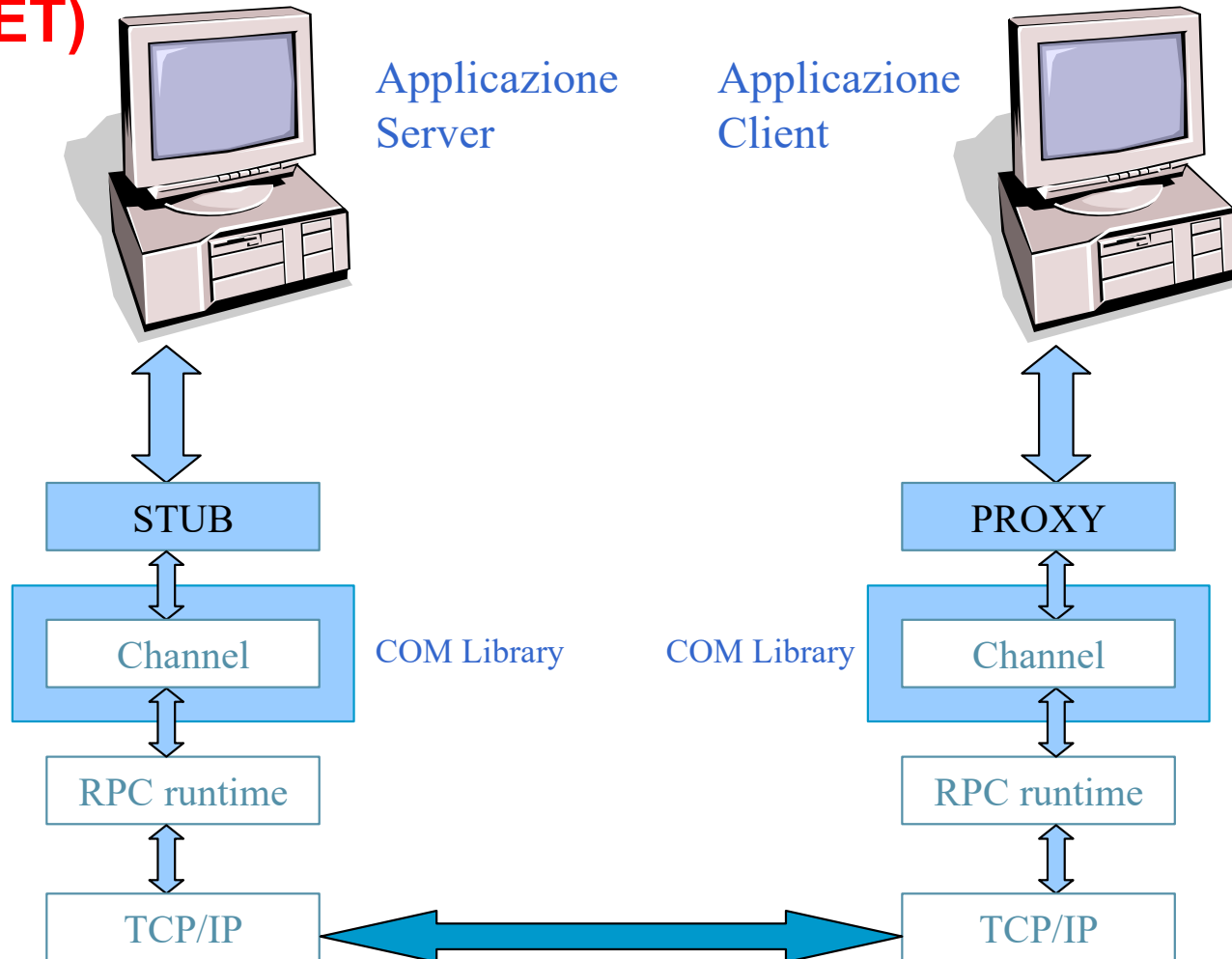
COM, DCOM,... (.NET)

Interazione **sincrona**

Ambiente basato su
componenti
multilinguaggio
proprietario

Integrazione di sistemi
e strumenti **esistenti**

Apertura verso
ambienti esterni
anche legacy





MIDDLEWARE nel supporto a sistemi

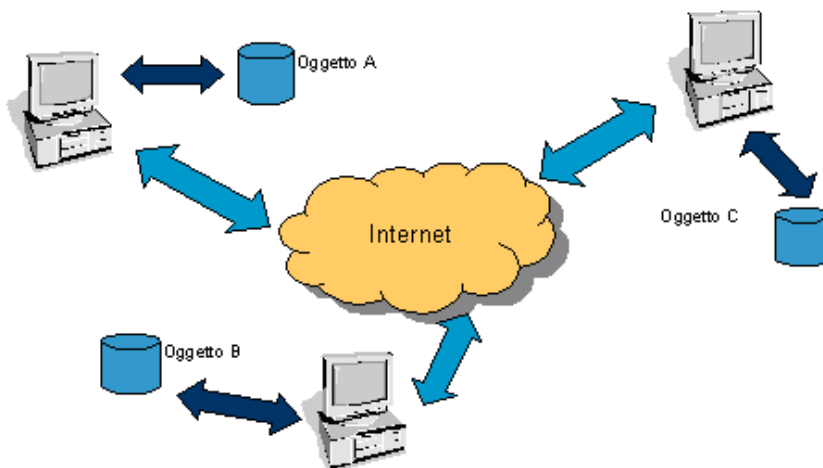
MIDDLEWARE per fornire **servizi WEB** in ambiente distribuito secondo la logica della massima integrazione

Service Oriented Architecture (SOA)

ambienti basati su interazione concordata e negoziabile

Enterprise Application Integration (EAI)

ambienti per integrare applicativi esistenti legacy





MIDDLEWARE per ENTERPRISE

MIDDLEWARE per fornire servizi di tipo business

Service Oriented Architecture (SOA)

Rivedere la **interazione** tutta in termini di **contratto astratto** di **servizi offerti e richiesti**

Interazione intesa solo come **interfaccia tra sistemi diversi**

Enterprise Application Integration (EAI)

Necessità di facilitare la **integrabilità** tra **strumenti di impresa esistenti** e la loro ampliata applicabilità e disponibilità in azienda

EAI come ambienti per la **integrazione veloce e precisa** di applicazioni e sottosistemi esistenti legacy

Anche interfacce di **rapida prototipazione** di nuove aggregazione



MIDDLEWARE e SICUREZZA

I firewall tendono a introdurre vincoli nel passaggio di funzionalità e a bloccare richieste di operazioni insicure

Fornitura di servizi WEB in ambiente distribuito

deve passare anche attraverso le politiche di sicurezza di qualunque sistema (eterogeneità) e di qualunque gestione di sicurezza



Web Services come middleware

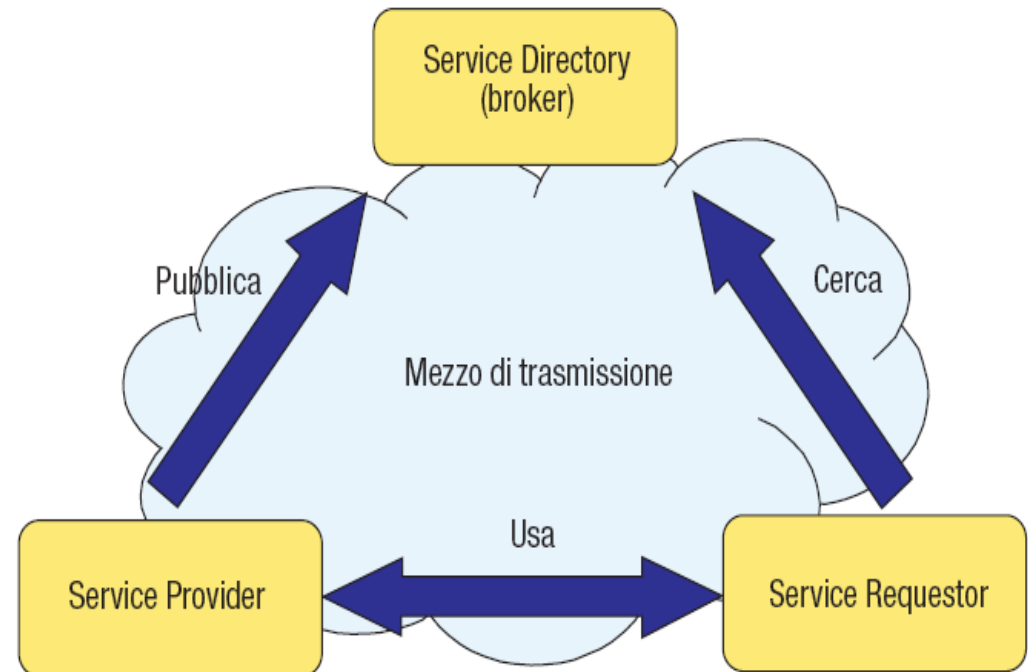
MIDDLEWARE come organizzazione standard per la fruizione di servizi (**Richiedente/Provider** e **Broker**) in ambito Web-compatible

Servizi offerti dal **Provider**

Richiesti quando e se necessario su bisogno dal **Client**

Esposti da appositi **sistemi di nomi broker** definiti **ad-hoc**

Non ORB di CORBA





Web Services come protocolli

Web Services come **MIDDLEWARE** di Integrazione (?)

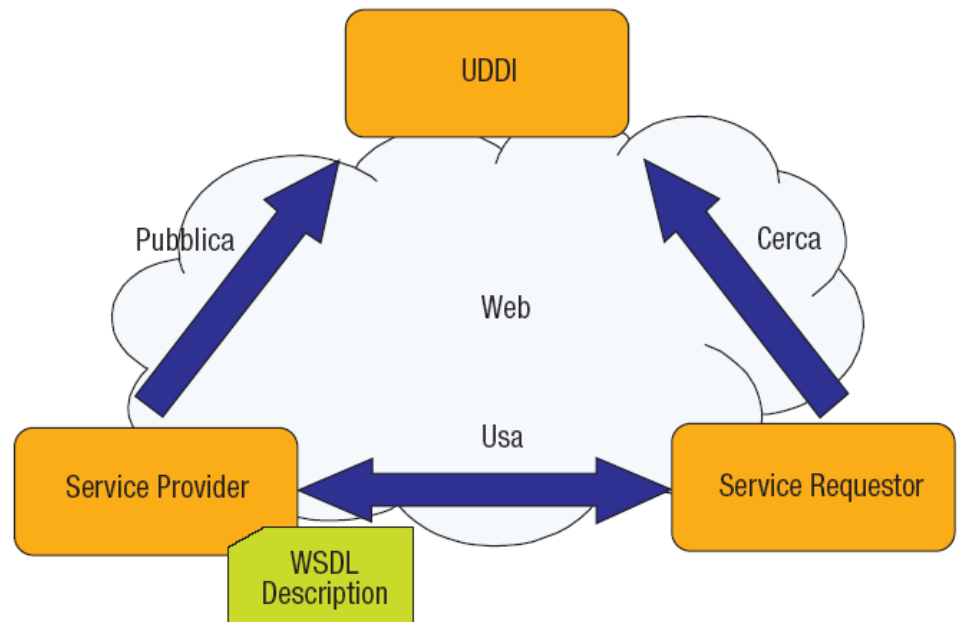
SOAP (Simple Object Access Protocol)

WSDL (Web Services Description Language)

UDDI (Universal Discovery, Description and Integration)

insieme con altre estensioni

Per ottenere la possibilità
di interoperare come
usando la
programmazione
ma attraverso il Web
(uso di XML)





Web Services: Protocolli

SOAP

Protocollo di comunicazione per la interazione sia C/S sia richiesta o risposta

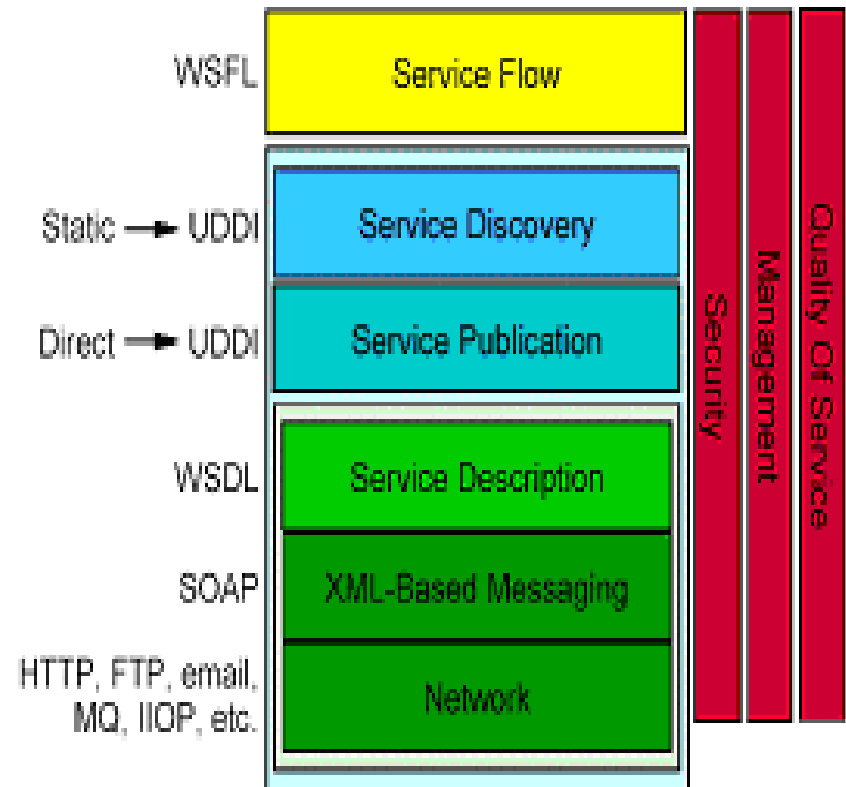
WSDL

dialetto XML per la descrizione dei servizi che si possono richiedere ed ottenere

UDDI

Sistema di nomi per esportare ed importare le proprietà dei servizi disponibili

insieme con altre estensioni





MIDDLEWARE infrastruttura di sistema

Uso di protocolli e strumenti Web compatibili

HTML considerato e usato come linguaggio di presentazione

XML **eXtensible Markup Language**

- Linguaggio di markup **aperto**
- Basato su puro **testo**
- Con informazioni di **tipo strutturale**
(?semantico?)
- Derivato da **SGML**
(**S**tandard **G**eneralized **M**arkup **L**anguage)



XML - metalinguaggio

XML come linguaggio di descrizione specializzabile per settori specifici

XML Definire **differenti layout** (compatibili con differenti dispositivi)

Definizione di **nuovi TAG applicativi**

Separazione del **contenuto** dalla **rappresentazione**

Strutturazione gerarchica delle informazioni

Generare la **grammatica** per validare dei dati

XML

Interoperabile e Orientato al Web

Compatibile con SGML

Integrabile con strumenti esistenti

Semplice e con opzioni semplificabili

Facile da usare e con vincoli per la specifica



XML – vantaggi

XML permette di conferire una **struttura ad informazioni** che sono **tipicamente non strutturate** (non significato)

XML si aggiunge in modo indolore ai formati **HTML** cui si giustappone anche per **documenti esistenti**

XML permette di omettere le **informazioni di struttura** (se esistenti e note)

XML permette di riferire e considerare **strumenti esterni per la validazione, trattamenti, e gestione** del documento

XML permette di riferire con **tecniche di imbustamento documenti oggetto** per un facile riferimento a **strutture ripetute**

XML si è affermato come uno standard per la apertura di Servizi Web ad un uso generalizzato



SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Protocollo SOAP per rispondere alla necessità di lavorare con **protocolli Web** ma permettendo di **specificare e progettare e gestire componenti e operazioni**

Soluzione per introdurre **parametri e valori nei messaggi** e per **l'invocazione remota di oggetti** basati su tecnologie Web

IPOTESI di PROGETTO

- **Uso di XML per serializzazione dei dati**
- **HTTP come protocollo di trasporto**

Esempio

```
<SOAP-ENV:Envelope>  
  <SOAP-ENV:Body>  
    <m:GetLastTradePrice>  
      <symbol>MOT</symbol>  
    </m:GetLastTradePrice>  
  </SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```



SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

SOAP come protocollo specifica:

- come effettuare una comunicazione **one-way**
- come effettuare una comunicazione **tipo C/S**
- come si devono gestire gli elementi in XML
- come si attua il solo **trasporto**

NON i dettagli locali della interazione

SOAP configura

un protocollo **stateless per la interazione**

senza fornire alcun supporto per **informazioni semantiche sul contratto di interazione**

SOAP tende ad usare *GET* e *POST* come operazioni WEB



SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Protocollo SOAP

<SOAP-ENV:Envelope>
... Header

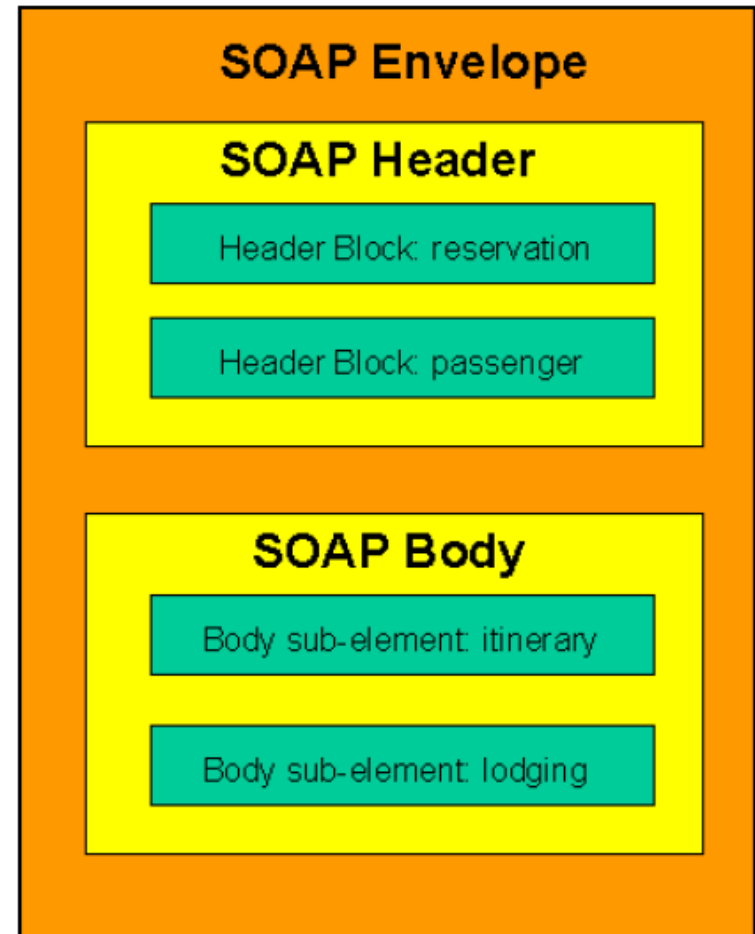
<SOAP-ENV:Body>
... Payload

<m:GetLastTradePrice>
 <symbol>MOT</symbol>

</m:GetLastTradePrice>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>





SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Protocollo SOAP

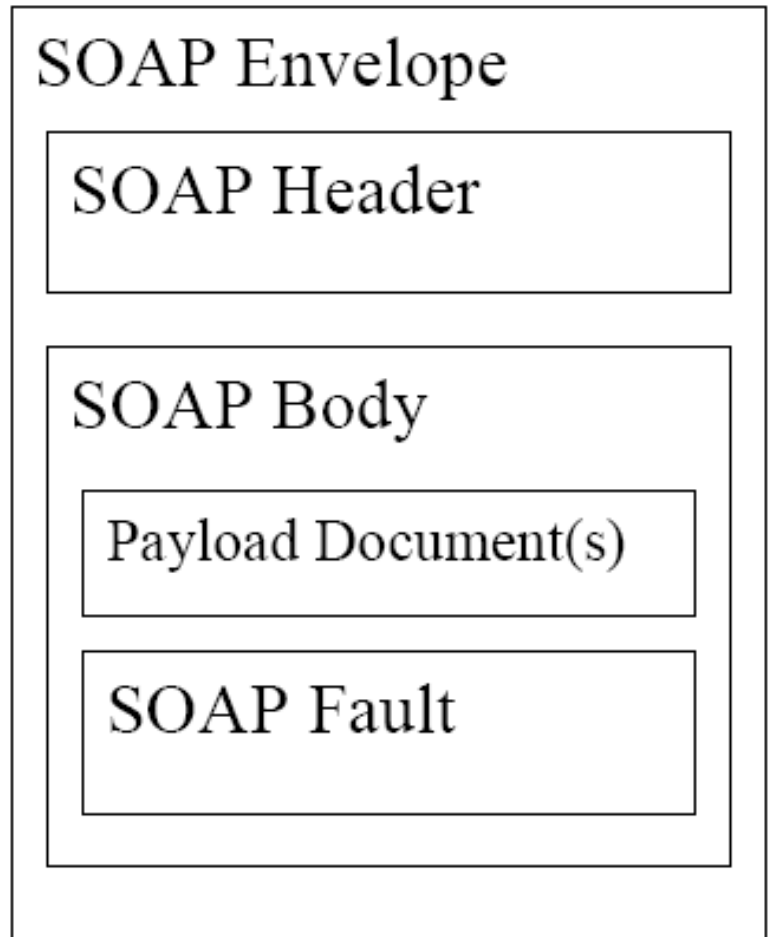
Envelope incapsula
il **contenuto del messaggio**

Header destinato a contenere
informazioni aggiuntive

Informazioni accessorie di sicurezza

Body incapsula le **richieste e le risposte** (in genere, il messaggio da comunicare)

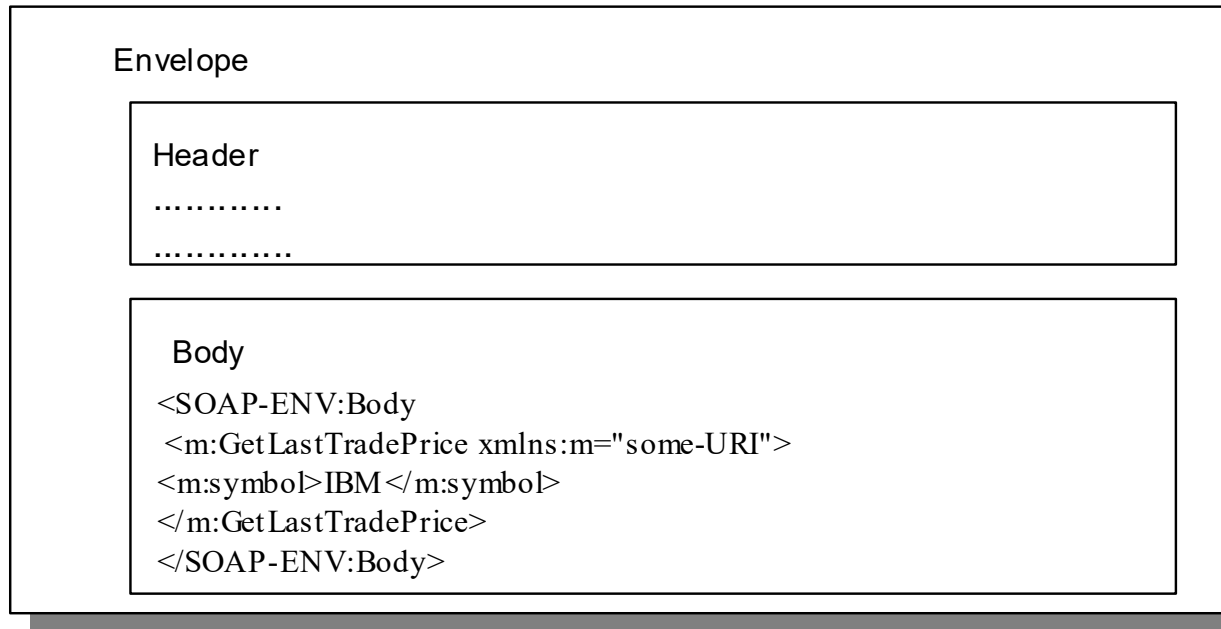
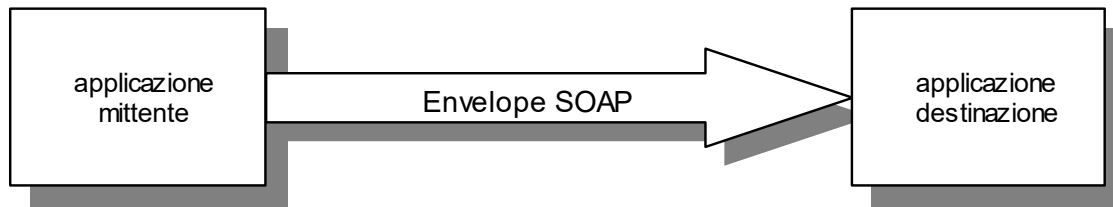
Fault incapsula eventuali casi
distinti di **errore ed eccezione**





SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Anche interazione multi attori: mittente, destinazione, e un insieme di intermediari





SOAP e COMPUTAZIONE: esempio

Un semplice esempio: un'applicazione finanziaria (client) colloquia con un servizio che fornisce in tempo reale le quotazioni di borsa

Questa interazione prevede la richiesta dell'ultima quotazione di una determinata azione e la risposta dal server

Schema del colloquio:

L'applicazione cliente costruisce una richiesta in XML usando la sintassi definita da SOAP

L'applicazione cliente trasmette **la richiesta ad un server Web** usando HTTP

Il server **riceve ed interpreta la richiesta** trasformandola in un comando che viene passato ad un'applicazione sul server

L'applicazione sul server riceve il comando e ricava dal proprio database l'informazione richiesta (ad esempio)

L'applicazione sul server **crea una risposta**, sempre in formato XML e la **restituisce al server Web**

Il server Web la restituisce **all'applicazione client come risposta HTTP**



SOAP e XML (request)

<POST /StockQuote/HTTP/1.1

Host: www.stockquoteserver.com

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

SOAPAction: "Some-URI"

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

SOAP_ENV:encodingStyle=

"http://schemas.xmlsoap.org/soap/encoding/>

<SOAP-ENV:Body>

<m:GetLastTradePrice xmlns:m="Some-URI">

<symbol>MOT</symbol>

</m: GetLastTradePrice>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>



SOAP come possibilità di riferire ambienti di nomi precisi

Si definiscono e riferiscono almeno *due namespace*:

- L'envelope SOAP trovato con **identificatore di namespace**

`"http://schemas.xmlsoap.org/SOAP/envelope/"`

- La serializzazione SOAP con un **namespace**

`"http://schemas.xmlsoap.org/SOAP/encoding/"`

Un messaggio SOAP **non deve contenere documenti** con dichiarazioni di tipo e istruzioni per processarlo

Il prologo del documento XML contiene la dichiarazione XML e del tipo di documento per identificare le informazioni specifiche da elaborare e le regole che controllano il documento completo

Anche **Interaction style: document o RPC style**

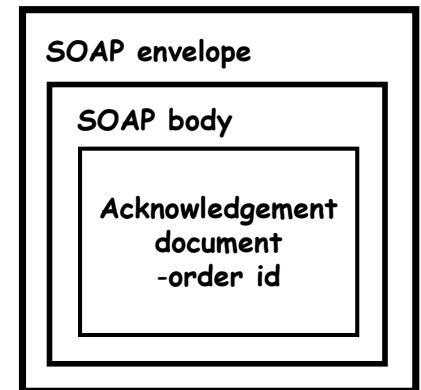
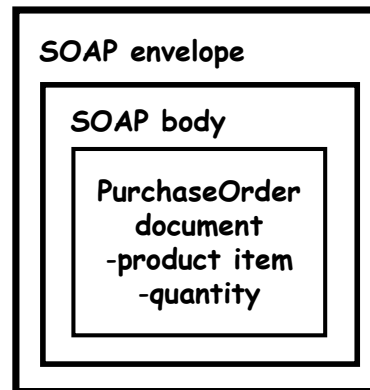


SOAP: STILE della INTERAZIONE

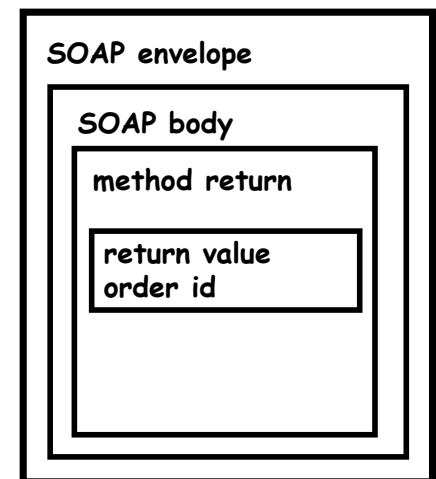
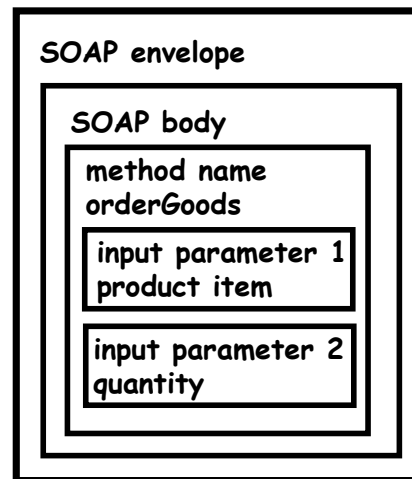
La interazione prevista prevede due forme di base:

Document-style
detta anche **one-way**
ASINCRONA

RPC-style
o anche **C/S**
SINCRONA



(a) Document-style interaction



(b) RPC-style interaction



SOAP e XML (response)

<HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

SOAP-ENV:

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<SOAP-ENV:Body>

<m:GetLastTradePriceResponse xmlns:m="Some-URI">

<Price>34.5</Price>

</m: GetLastTradePriceResponse>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>



SOAP e XML (errore)

<HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

<SOAP:Envelope

XMLns:SOAP="HTTP://schemas.XMLSOAP.org/SOAP/envelope"

SOAP:encodingStyle=

"HTTP://schemas.XMLSOAP.org/SOAP/encoding">

<SOAP:Body>

<SOAP:Fault>

<faultcode>Client**</faultcode>**

<faultstring>Invalid Request**</faultstring>**

<faultactor>unknown**</faultactor>**

<detail>during the parameter ...**</detail>**

</SOAP:Fault>

</SOAP:Body>

</SOAP:Envelope>



SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Protocollo SOAP

Con il protocollo riusciamo a lavorare come per **RPC** veicolate attraverso il protocollo HTTP

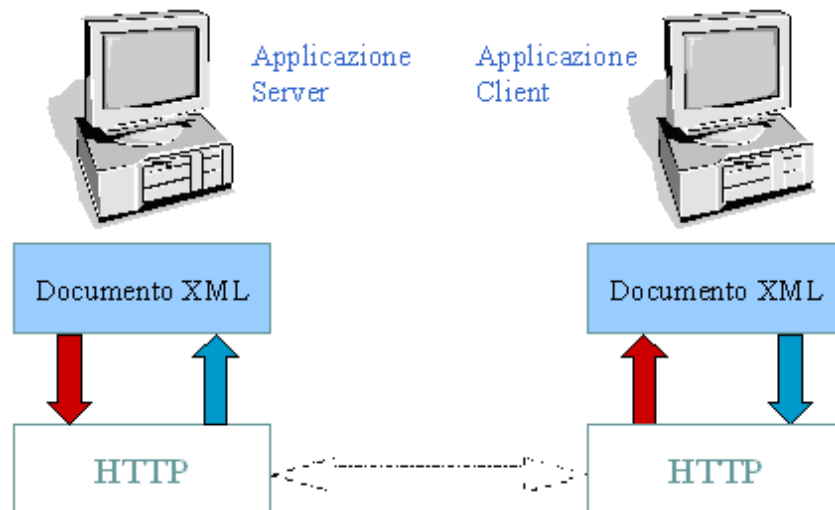
Tendiamo quindi a mandare messaggi e ricevere risposte (inserendo informazioni come da linguaggio di programmazione)

Per raggiungere la **indipendenza** dalle diverse **realizzazioni dei servizi** che stiamo richiedendo e dalla **eterogeneità delle diverse** architetture

Usando il protocollo **non ci accorgiamo di quali server stiamo richiedendo** anche se **ci muoviamo a livello applicativo molto alto** (e con **poca efficienza**)

Protocollo per comunicare dati:

- Serializzando i dati in **modo specifico**
(indipendentemente dalla piattaforma)
- Operazioni **leggere, robuste e flessibili** (?)
- Supporto **per tutte le architetture**
(**.NET, J2EE, IBM WebSphere, Sun ONE**)





Torniamo alla definizione iniziale

Web Services

Componenti software indipendenti dalla **piattaforma** e dall'**implementazione** che possono essere:

- **descritti** usando un **linguaggio di descrizione** del servizio (**WSDL**)
- **pubblicati in un registro** di servizi (**UDDI**)
- **scoperti mediante** una politica definita e meccanismi **standard di discovery** (a runtime o a tempo di progetto)
- invocati mediante un'**API remota**, solitamente tramite la rete (**SOAP**)
- **composti con altri servizi**

vedi <http://www.w3.org/2002/ws/>



Web Services: WSDL

Per i WS, oltre alla **comunicazione...**

dobbiamo anche considerare di descrivere il servizio sia in modo astratto, sia in modo concreto

WSDL (Web Services Description Language)

Una proposta in XML per **descrivere Web Services e per pubblicarli** specificando esattamente il **formato dei messaggi di richiesta e di risposta** in modo **portabile e standard** e anche i dettagli

WSDL si occupa di specificare:

- **cosa un servizio può fare** (richieste, risposte e parametri)
- **dove risiede e lo si può invocare**
- **come invocarlo**



Web Services Description Language

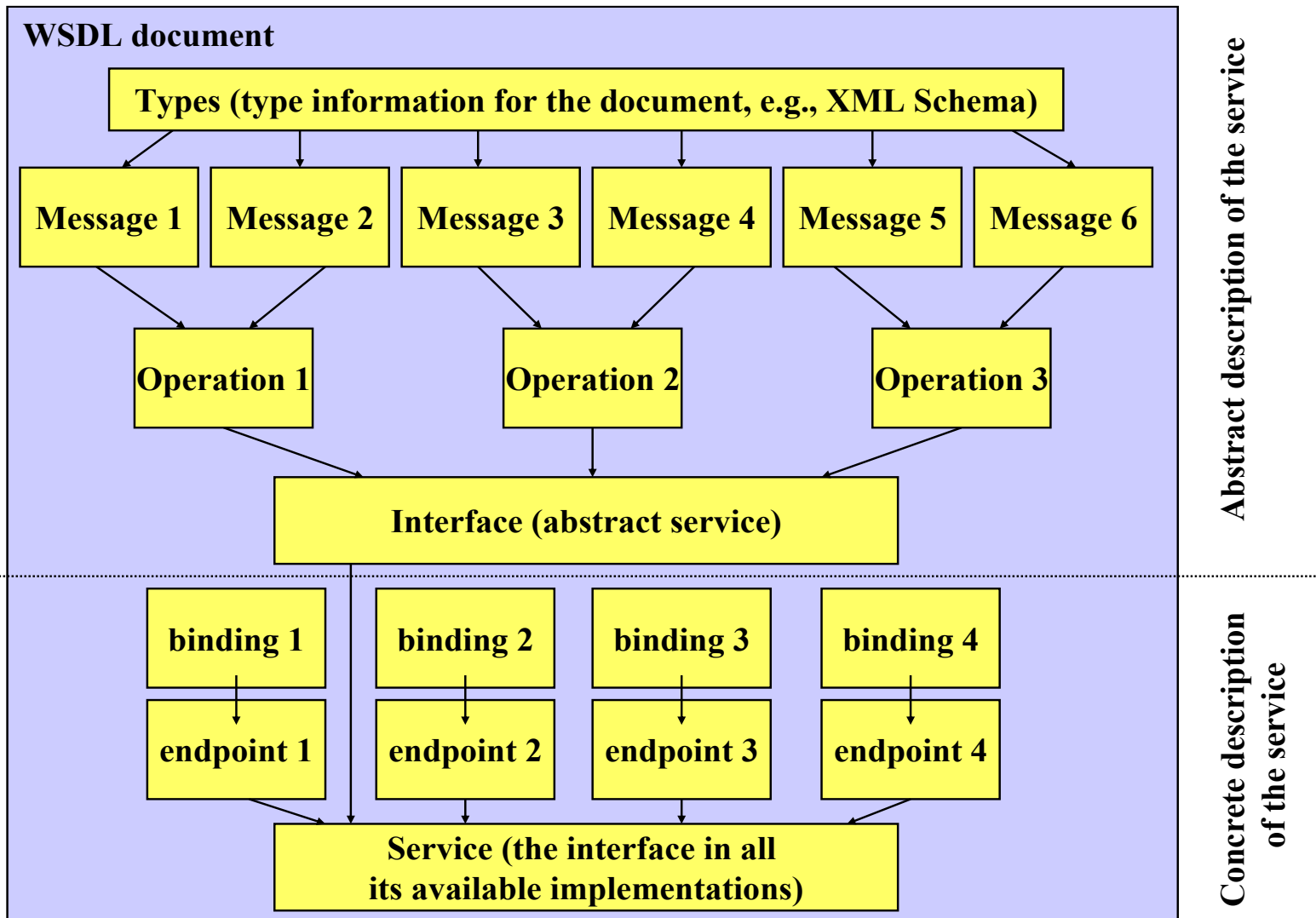
Se si vuole usare un Web service non noto

- si richiede il **file WSDL**
- si analizza il documento **WSDL** per determinare
 - **locazione del servizio**
 - **chiamate dei metodi con i parametri**
 - **come accedere ai metodi**
- si crea una **richiesta SOAP**
- si invia la richiesta **SOAP al servizio** e si attende la **risposta**

La logica è quella di avere il massimo del supporto e della facilità nel procedere, fino alla completa automazione da parte di un middleware

Alcune parti di WSDL sono molto simili a un IDL

Elementi di base di WSDL





ARCHITETTURA DI WSDL 2.0

WSDL descrive i Web Services iniziando con i messaggi da scambiare tra service Requestor e Provider

*I messaggi sono descritti a partire da una prospettiva **astratta** e poi in forma più **concreta** (protocollo e formato)*

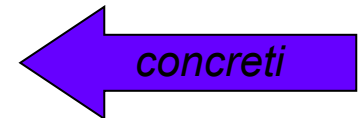
Un **messaggio** consiste in una **collezione di elementi tipati**

Uno **scambio di messaggi** è definito **operation**



Una **collezione di operation** è definita una **interface (portType v.1)**

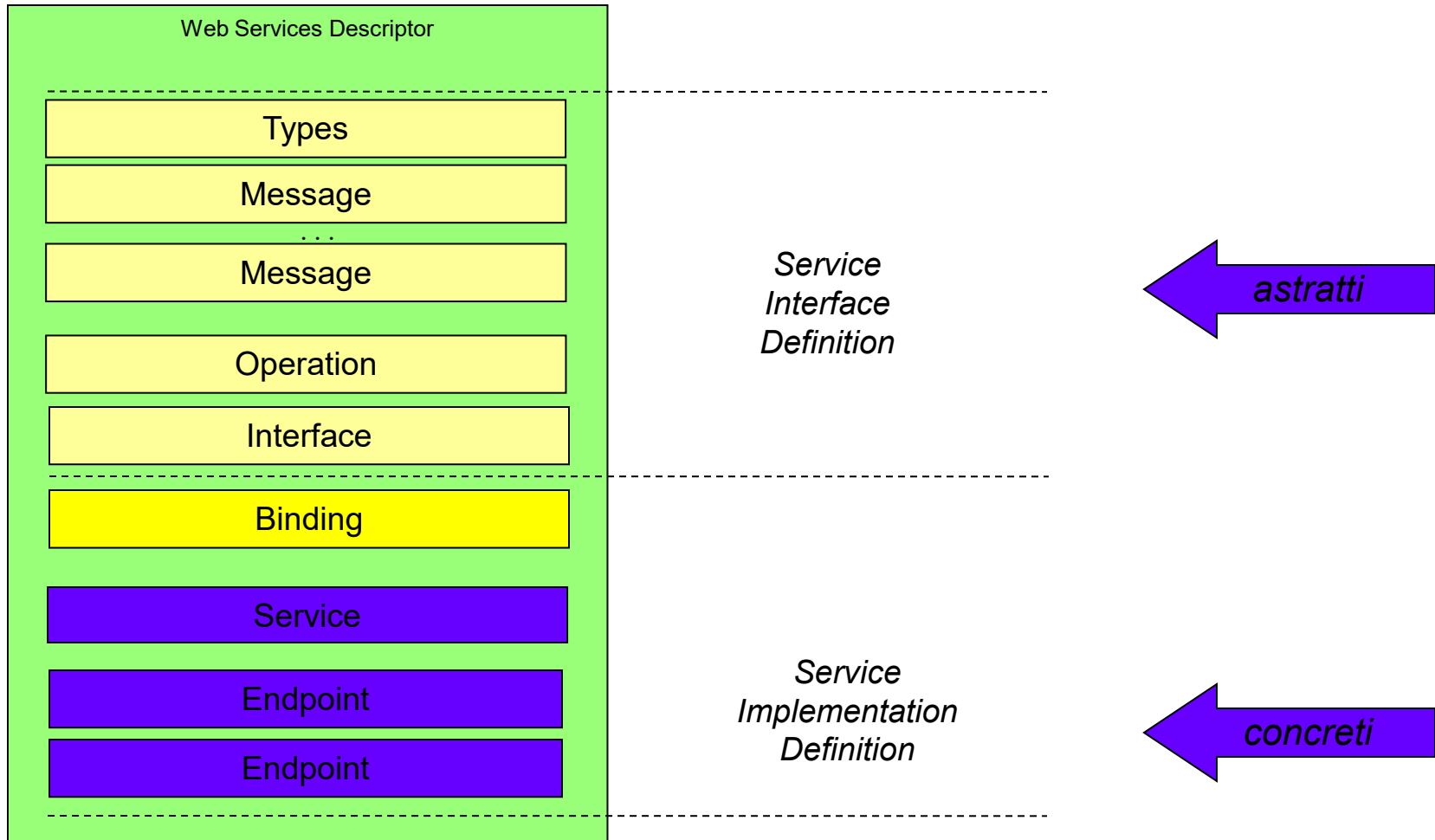
Un **service** rappresenta **l'implementazione** di una **interface** e contiene una **collezione di endpoint (port v.1)**



Un **endpoint** è **l'implementazione concreta** del servizio e include tutti i **dettagli concreti** necessari al verificarsi della **comunicazione**

Un **binding** è il legame per richiedere le **operazioni** concrete

WSDL descrive sia le parti astratte sia le parti concrete





SERVIZIO in WSDL

Un documento WSDL è formato da questi elementi
in corrispondenza ad una parte dell'applicazione

– Parti **astratte**

Message, Operation, Interface

– Parti **concrete**

Type, Binding, Endpoint, Service

WSDL definisce gli elementi astratti prima e poi le controparti concrete

La versione **astratta** del servizio è **generalizzabile, flessibile e facilmente estendibile**

Le specifiche **concrete** sono definite solo in ognuno degli **elementi costituenti il servizio**



Parti Astratte in WSDL

- **message**

informazione effettivamente inviata tra requestor e provider, con la possibile qualificazione del messaggio di input, output, o fault

- **operation**

specifica dei **nomi** delle operazioni, i **parametri** di **input** e **output** e consiste di **messaggi**

- **interface**

un insieme di **operazioni astratte** e di **messaggi** aventi un identificativo univoco

che corrisponde al servizio stesso

e che di solito si presenta in modo unico nel documento WSDL



Parti Concrete in WSDL

- **type** tipo di dato in un messaggio usando XML Schema
(inserita per prima)
- **binding** per i dettagli dell'implementazione delle operazioni contenute in un'interface

Specifica i protocolli concreti: trasporto e codifica dei dati
(**HTTP, SOAP; SMTP; FTP; ...**)

- **endpoint** per indicare l'indirizzo di rete del servizio con cui effettuare la connessione

- **service** come una collezione di endpoint correlati

Permette di raggruppare tutte le interface, in modo che sia visibile quali siano gli endpoint supportati da un determinato servizio

Ad esempio, tutti gli endpoint associati ad una transazione che richiede più passi



Web Services: WSDL types

Una prima parte del WSDL descrive i tipi necessari per le operazioni

```
<types> <schema>
  <element name="TradePriceRequest">
    <complexType>
      <all>
        <element name="tickerSymbol" type="string"/>
      </all>
    </complexType>
  </element>
  <element name="TradePrice">
    <complexType>
      <all>
        <element name="price" type="float"/>
      </all>
    </complexType>
  </element>
</schema> </types>
```



WSDL message, operation, e interface

Troviamo la descrizione dei **messaggi** e **operazioni**:

```
<message name="GetLastTradePriceInput">  
  <part name="body" element="xsd1:TradePriceRequest"/>  
</message>  
<message name="GetLastTradePriceOutput">  
  <part name="body" element="xsd1:TradePrice"/>  
</message>
```

Ogni operazione è costituita da un messaggio di richiesta e uno di risposta raggruppate in interface

```
<interface name="StockQuoteInterface">  
  <operation name="GetLastTradePrice">  
    <input message="tns:GetLastTradePriceInput"/>  
    <output message="tns:GetLastTradePriceOutput"/>  
  </operation>  
</interface>
```



Ogni Interface prevede modalità diverse sincrone e *asincrone*

Request_response: l'operazione prevede una risposta del service provider al messaggio del client

Solicit_Response: l'operazione prevede l'attesa da parte del service provider di una risposta sollecitata con una richiesta mandata dal provider al cliente

One_way: *l'operazione composta da un solo messaggio in ingresso al service provider*

Notification: *l'operazione è composta da un solo messaggio in uscita al service provider*



WSDL binding

Il binding come collegamento tra un tipo di operazione (type), un nome di operazione (name) e l'azione da eseguire (soapAction):

```
<binding name="StockQuoteSoapBinding"
  type="tns:StockQuoteInterface">
  <soap:binding>
    <operation name="GetLastTradePrice">
      <soap:operation
        soapAction="http://lia.deis.unibo.it/soap/bin/" />
      <input><soap:body use="literal"/></input>
      <output><soap:body use="literal"/></output>
    </operation>
  </binding>
```

Si riferiscono le implementazioni concrete



WSDL endpoint e service

L'ultima parte del documento descrive il servizio e l'indirizzo Web da utilizzare per accedere:

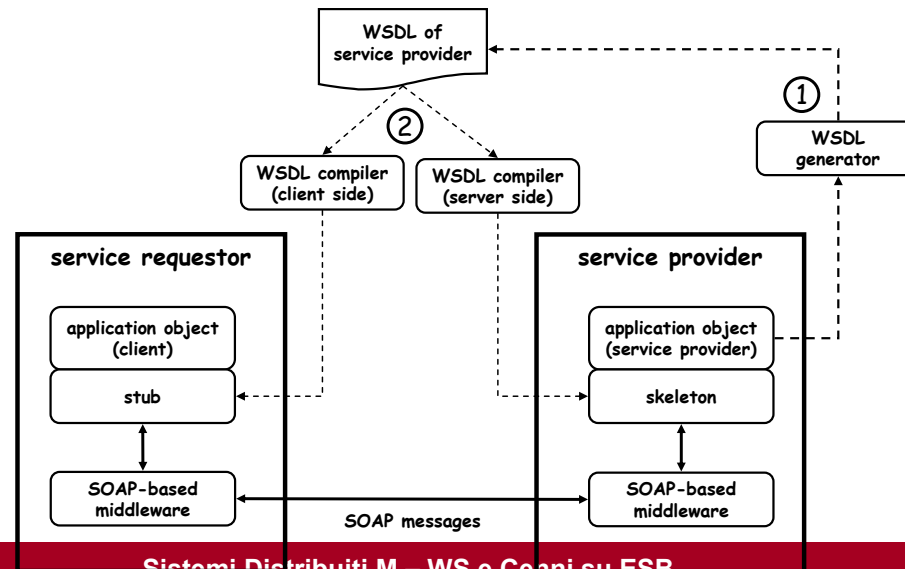
```
<service name="StockQuoteService">  
  <documentation>  
    Servizio di quotazione azioni  
  </documentation>  
  <endpoint name="StockQuoteEndPoint"  
    binding="tns:StockQuoteBinding">  
    <soap:address location="http://www.stockquote.com"/>  
  </endpoint>  
</service>
```

Oltre a ultimi dettagli concreti

WSDL può essere usato:

- come descrizione del contratto di servizio IDL
- come compilatore di stub
- come descrittore della semantica (?)

Uso di XML per generare dalle specifiche wrapper per servizi legacy





Universal Description Discovery & Integration

Universal Description Discovery & Integration language (UDDI)

Necessità di **sistemi di naming / directory** per Web Services

Un fornitore di servizi UDDI (IBM, Microsoft, SAP ecc.) gestisce un **registro elettronico** denominato UBR (UDDI Business Registry) accessibile sia per pubblicare che per rintracciare i Web Services con una suddivisione delle funzioni in:

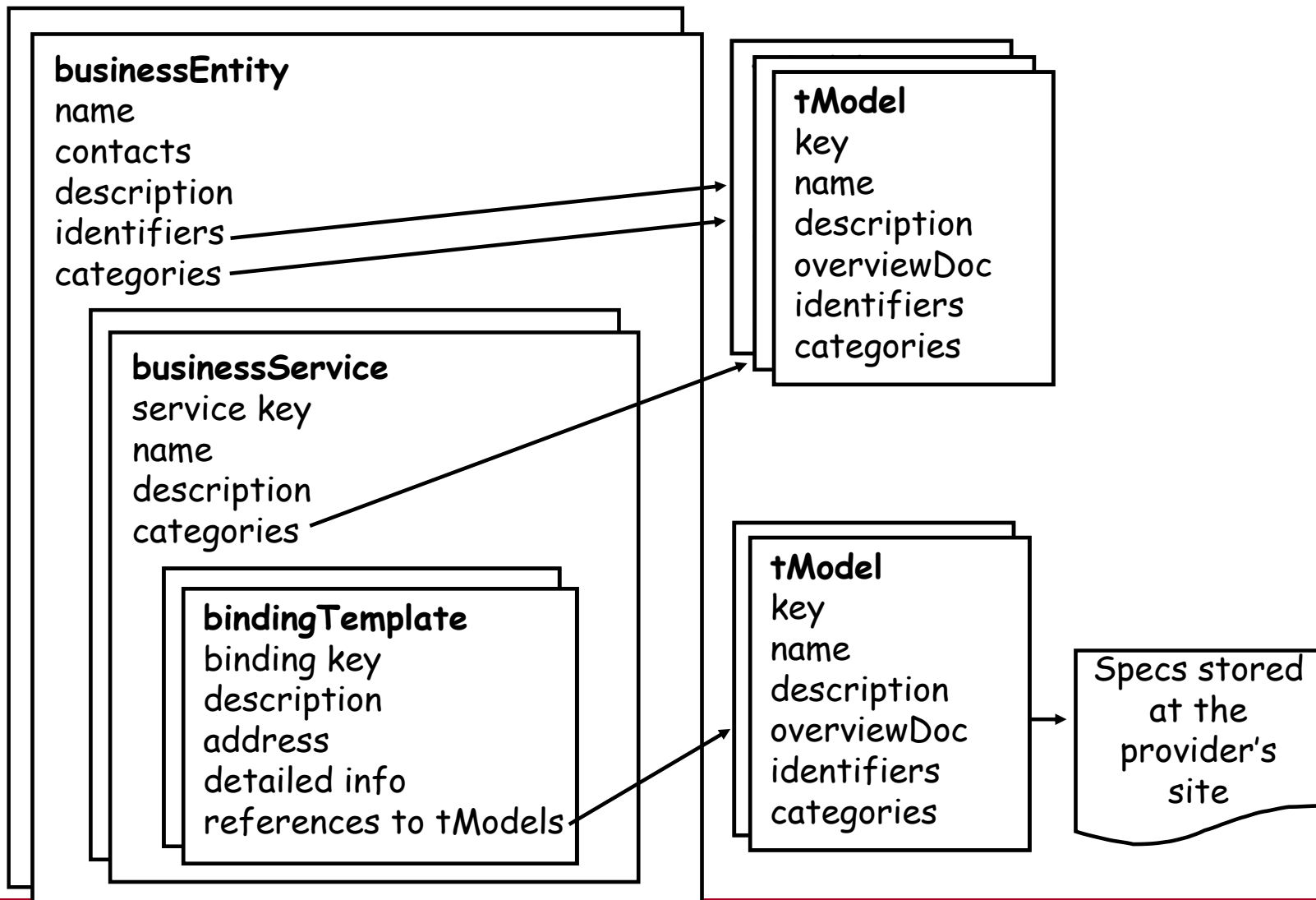
Service Type Registry informazioni sui tipi di servizio

tModel descrizioni delle tipologie dei dati con contenuto tecnico

Business Registry informazioni sulle aziende che le forniscono



Universal Description Discovery & Integration



Copyright Springer Verlag Berlin Heidelberg 2004



INFORMAZIONI di UDDI

businessEntity intesa come **descrizione della azienda produttrice** del servizio

businessService come **descrizione di un gruppo di servizi** offerti da un provider
Anche più WS nello stesso elemento

businessTemplate come **descrizione delle informazione tecniche** per richiedere il servizio

Indirizzo delle informazioni, fino ad una descrizione dei documenti coinvolti

tModel o **technical model**, inteso come contenitore di descrizioni per completare le informazioni da fornire (anche più di una per item)



Universal Description Discovery & Integration

Universal Description Discovery & Integration language (UDDI)

lo standard UDDI si propone di rintracciare i WS organizzandosi su tre tipi di servizi i cui nomi si ispirano al mondo telefonico e delle ricerche in quell'ambito

White pages: permette di trovare un **servizio per nome**

Yellow pages: permette di trovare un servizio **per categoria** nell' ambito di **molteplici classificazioni**

Green Pages: fornisce **informazioni tecniche aggiuntive** sui servizi offerti da una determinata azienda

Operazioni di base: **ricerca e pubblicazione**



Universal Description Discovery & Integration

UDDI utilizzato da due classi di utenti

- **Publisher:** compagnia che offre Web Services
- **Client:** utente o compagnia che ricerca un Web service

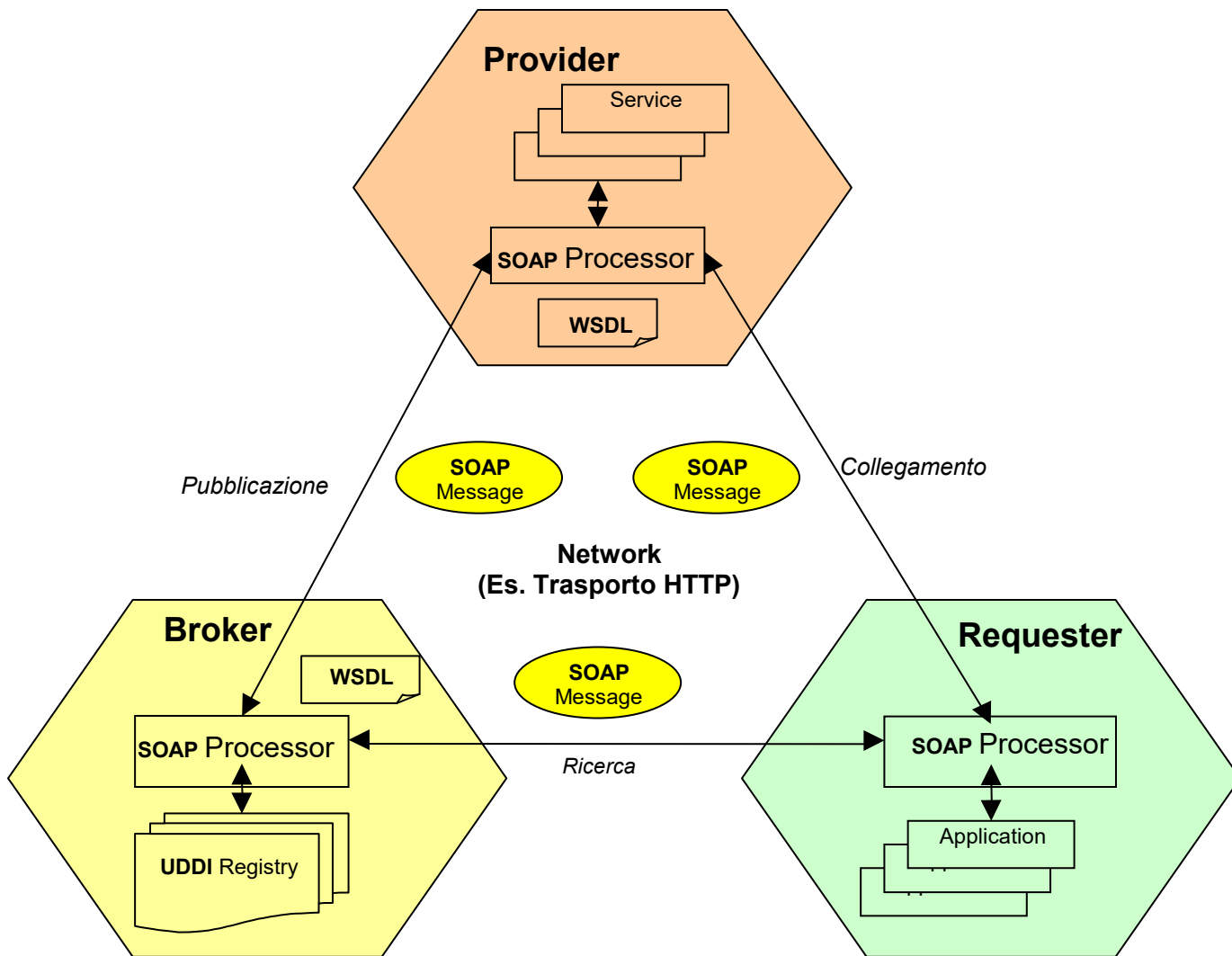
UDDI è un **servizio globale condiviso** tra **server differenti sparsi** in tutto il mondo, **anche se non organizzati secondo una struttura gerarchica: i server non sono coordinati**

I diversi server condividono i dati mediante protocolli di replicazione

UDDI come DNS (Domain Name System) con la differenza che **DNS lavora a basso livello e UDDI lavora a livello alto di servizi**

UDDI si basa su SOAP per la trasmissione dei messaggi

INTEGRAZIONE PROTOCOLLI





Universal Description Discovery & Integration

La **sicurezza** è un aspetto fondamentale in UDDI

In UDDI, le due azioni principali (**Registration** e **Discovery**)
sono a diritti diversi

- **Problema**: un concorrente potrebbe cancellare il servizio di un altro publisher
- **Soluzione**: autenticazione dei publisher

Ogni server mantiene traccia dei publisher e di cosa sia pubblicato
Solo chi ha pubblicato un servizio autorizzato a modifica/cancellazione

Classificazione dei Registri

pubblici, privati, condivisi (semi-privati)



Web Services Inspection Language (WSIL)

Web Services Inspection Language

Con **obiettivo simile a UDDI e complementare**

Informazioni sulla descrizione di un servizio che possono essere distribuite in ogni locazione usando un semplice documento in formato XML

Evita una delle **attuali difficoltà di UDDI**

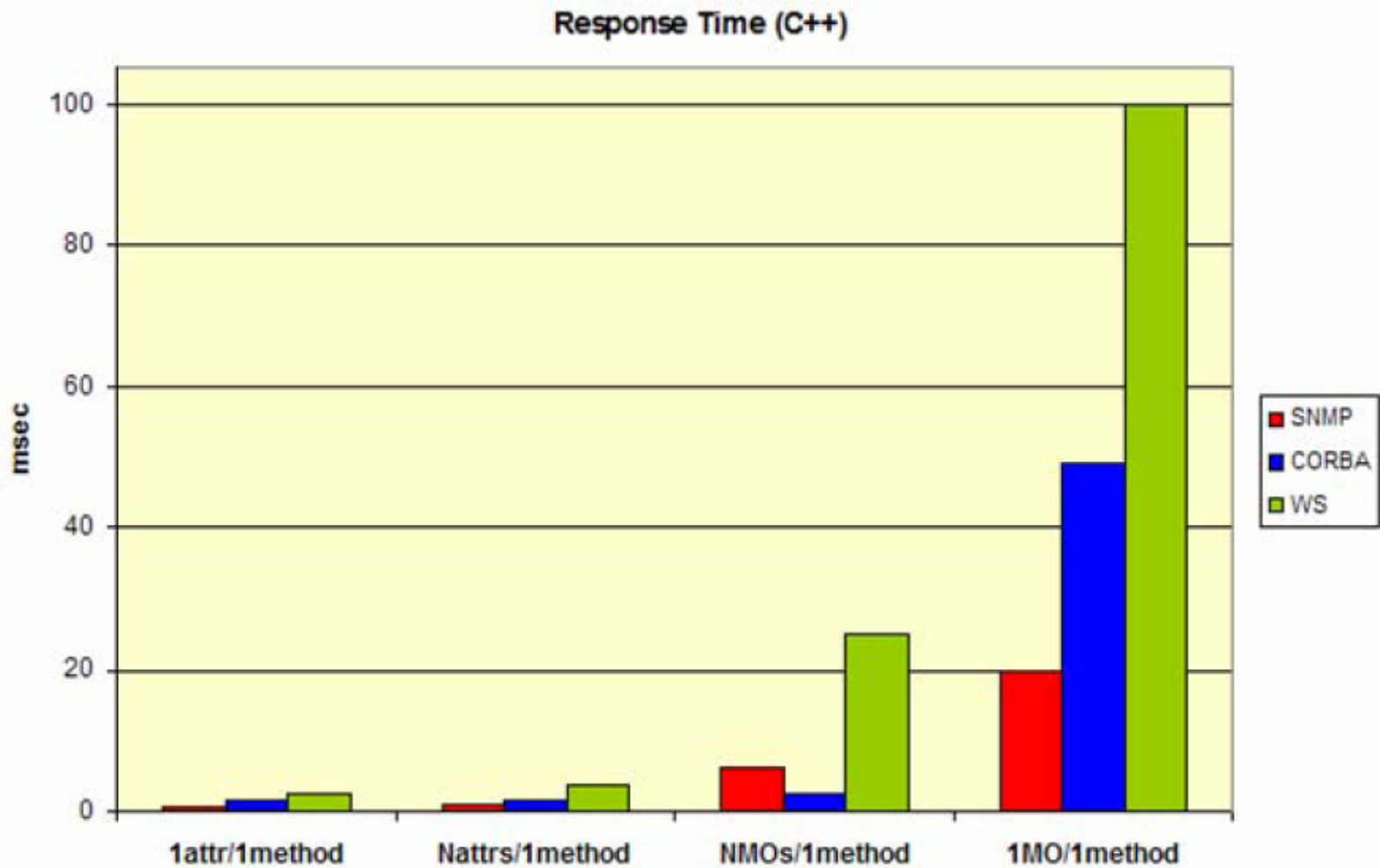
Le entry in un registro UDDI non sono moderate ed un client non può essere sicuro che un servizio appartenga effettivamente al service provider che lo ha pubblicato nel registro UDDI

Vantaggi principali: **decentralizzato, leggero**
senza focalizzare informazioni di business

Utilizza altri meccanismi per la descrizione dei servizi come WSDL
(analogamente ad UDDI)

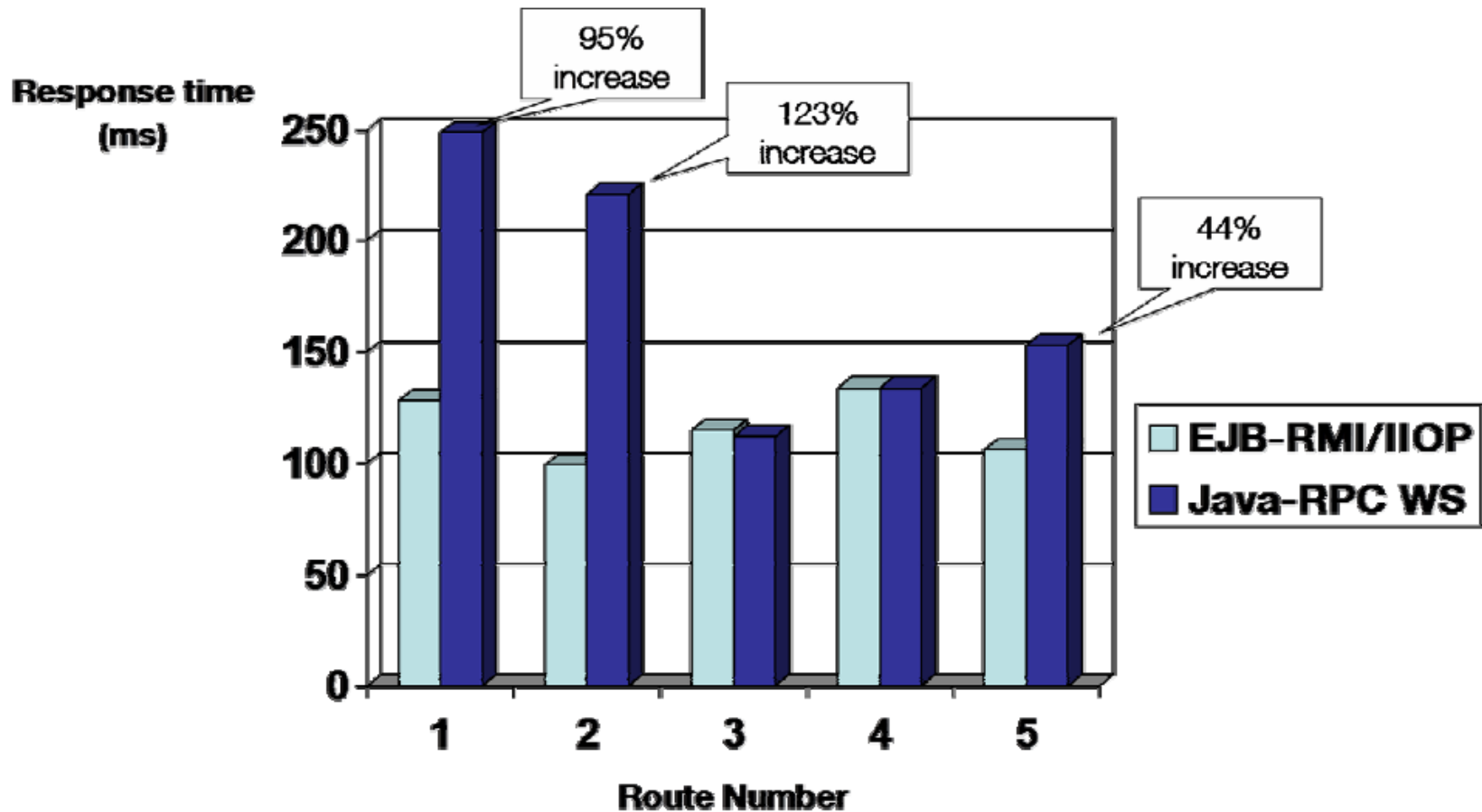


Performance di Web Services



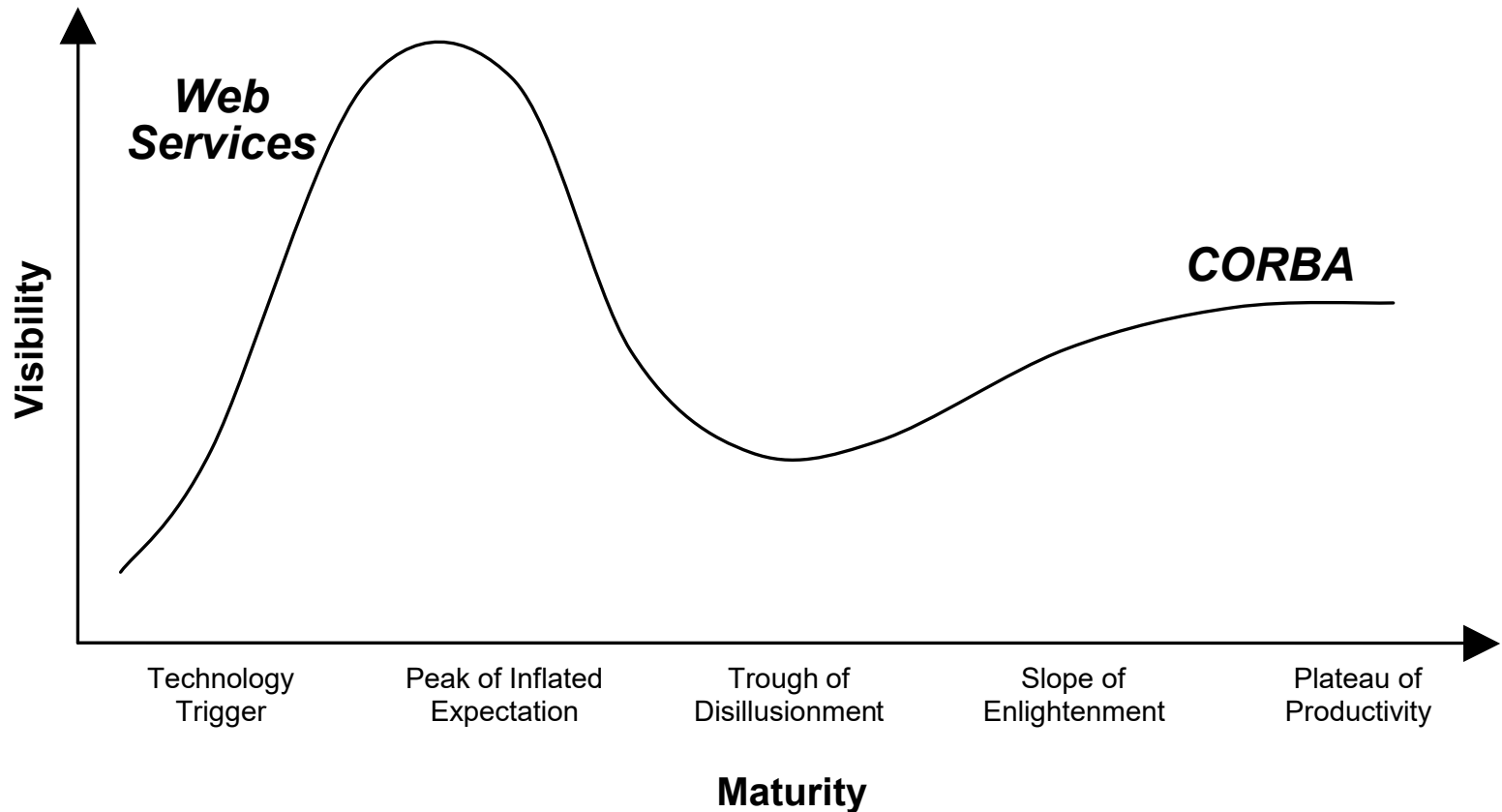


Performance di Web Services





Valutazione ed evoluzione tecnologie

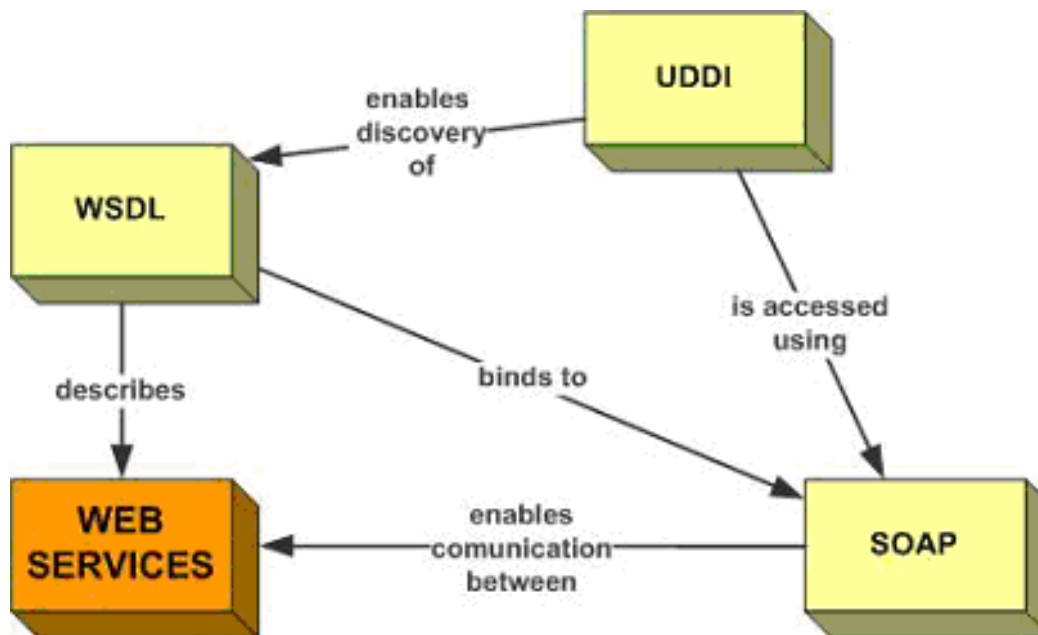


Ogni tecnologia può essere descritta con un ciclo di vita

Architettura Web Services

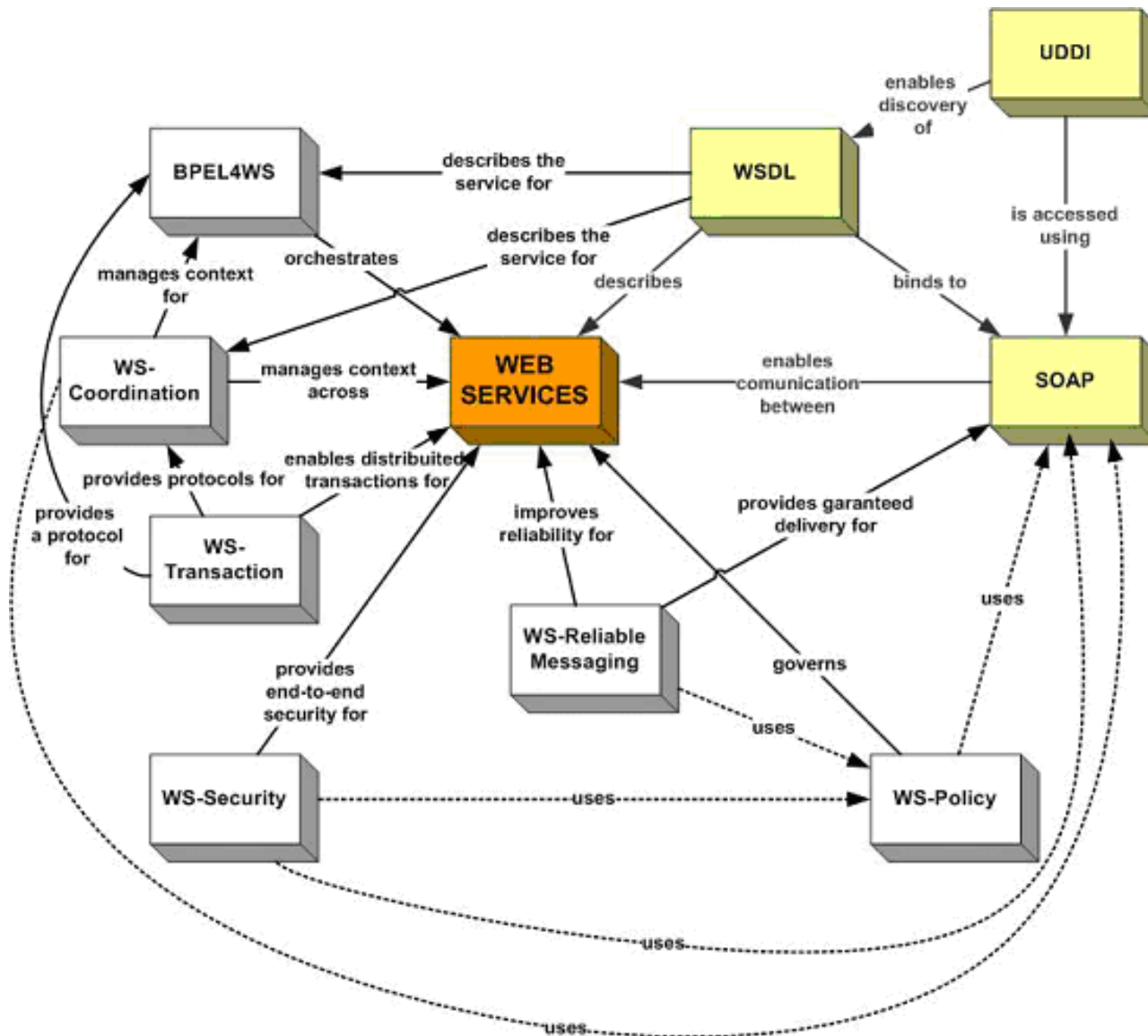
I **protocolli di base** hanno ottenuto specifiche e applicazione, ma manca ancora tutta la parte di specifica e descrizione del **middleware di supporto** ai diversi servizi

Incapacità espressiva con cui si sono scontrati gli sviluppatori



I protocolli di base **SOAP**
WSDL
UDDI
da soli hanno poca capacità
di estensione

Nuova Architettura Web Services



Sono stati introdotti e specificati nuovi protocolli di contorno per ottenere la ulteriore possibilità di allargare i comportamenti che si possono descrivere



Altri Protocolli Web Services

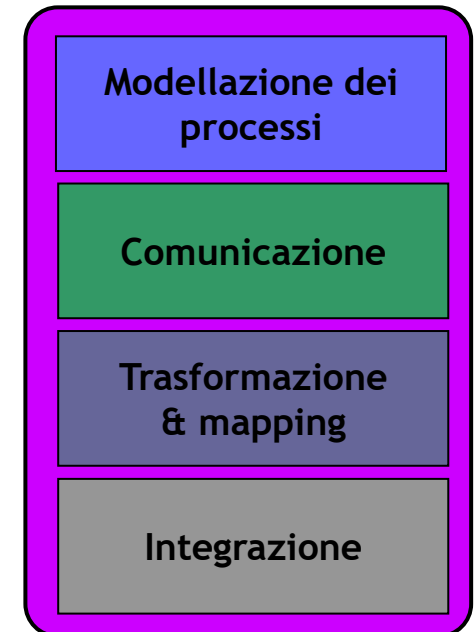
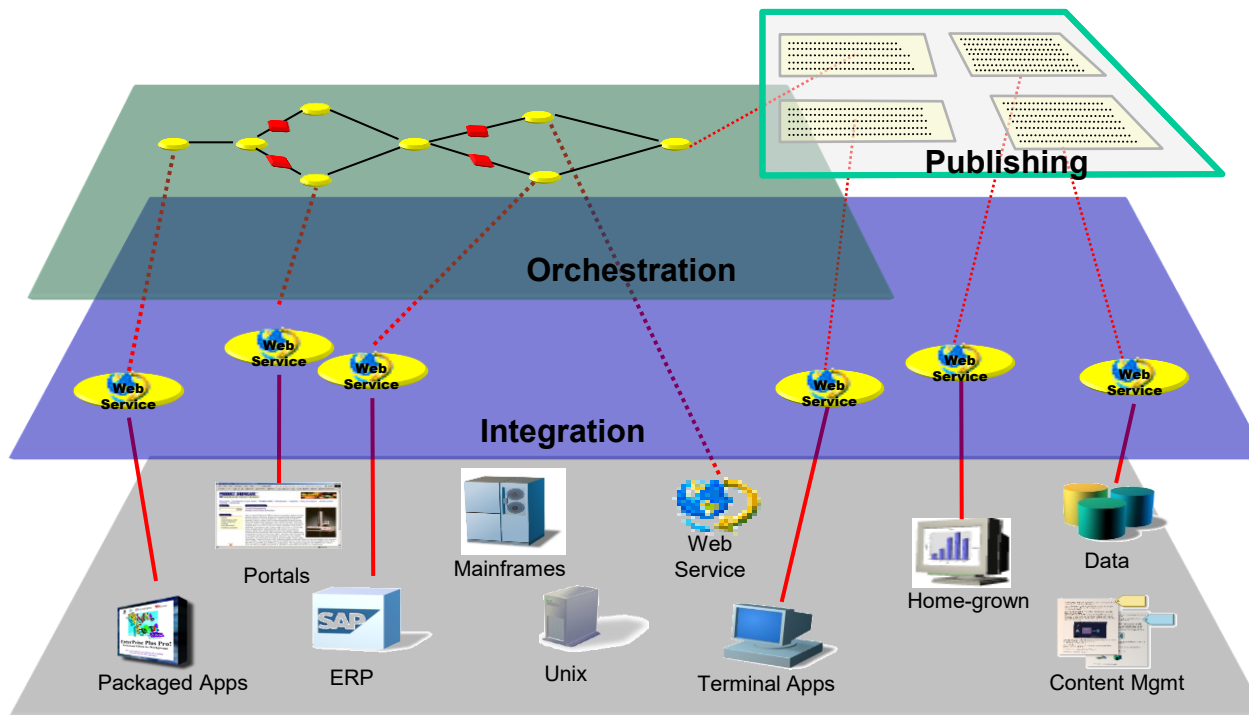
E la **composizione di servizi**?

Business Process Execution Language for Web Services
(**BPEL4WS**)

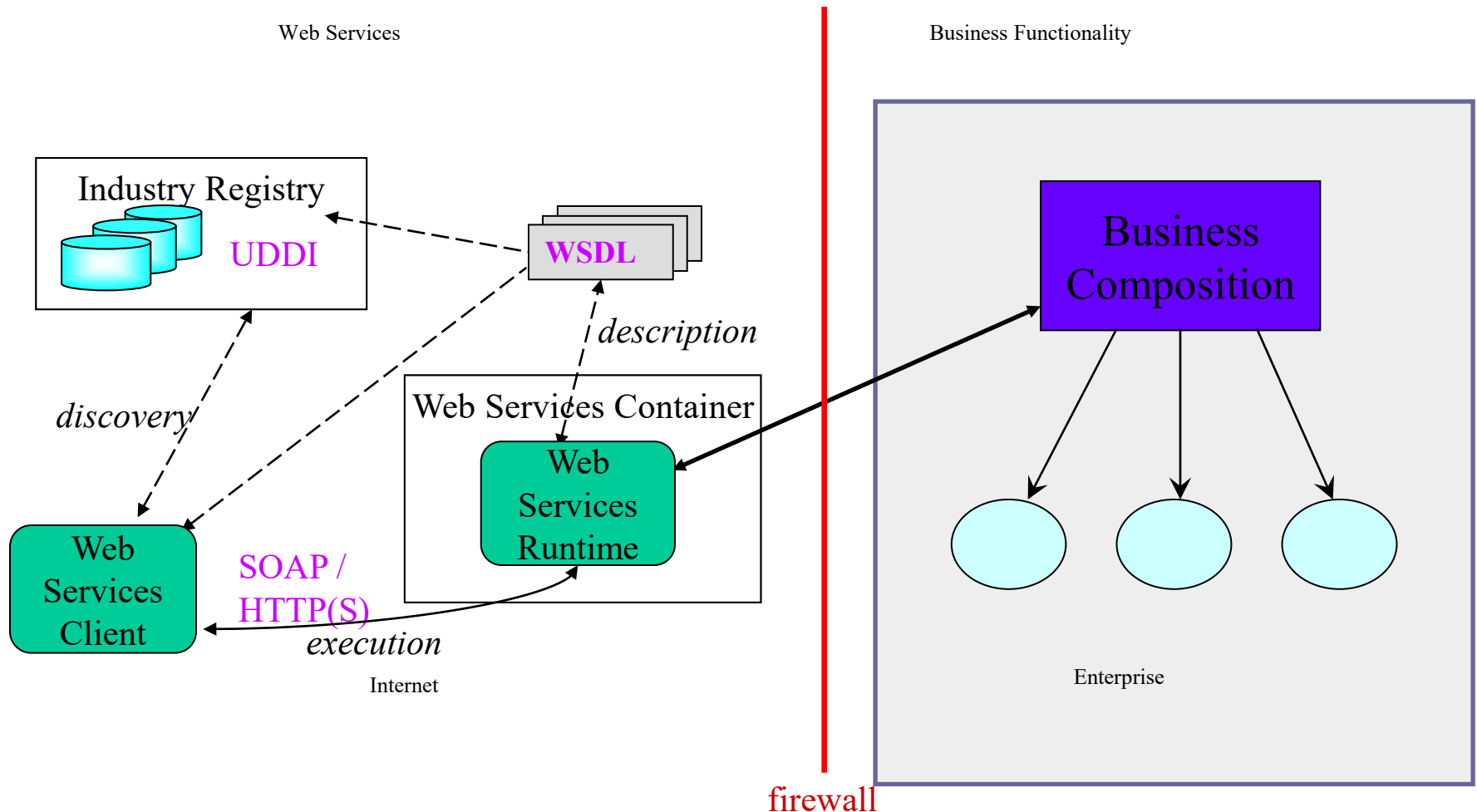
Sul modello WSDL si costruisce un linguaggio di script che permette la invocazione orchestrata dei Web Services come una applicazione di Workflow Management sulla base di costrutti di

- **sequenza**: attività una di seguito all'altra
- **parallelo** (**AND split**): attività che procedono in parallelo
- **alternativa** (**OR split**): più attività in alternativa
(condizioni di attivazione differenziate);
- **join** (**AND oppure OR**): terminazione congiunta di tutte o di una delle attività

Livello di integrazione e di orchestrazione



Integrazione di WS e modello business





CORBA e Web Services

Il confronto tra tecnologie permette di collocare le funzioni proposte in analogia se affrontano gli stessi problemi

CORBA stack	Web Services stack
IDL	WSDL
CORBA Services	UDDI
CORBA Stubs/Skeletons	SOAP Message
CDR binary encoding	XML Unicode encoding
GIOP/IIOP	HTTP
TCP/IP	TCP/IP



CORBA e Web Services

Aspect	CORBA	Web services
Data model	Object model	SOAP message exchange model
Client-Server coupling	Tight	Loose
Location transparency	Object references	URL
Type system	IDL	XML schemas
	static + runtime checks	runtime checks only
Error handling	IDL exception	SOAP fault messages
Serialization	built into the ORB	can be chosen by the user
Parameter passing	by reference	by value (no notion of objects)
	by value (<i>valuetype</i>)	
Transfer syntax	Common Data Repr. used on the wire	XML used on the wire
	binary format	Unicode
State	stateful	stateless
Request semantics	at-most-once	defined by SOAP
Runtime composition	DII	UDDI/WSDL
Registry	Interface Repository	UDDI/WSDL
	Implementation repository	
Service discovery	CORBA naming/trading service	UDDI
	RMI registry	
Language support	any language with an IDL binding	any language
Security	CORBA security service	HTTP/SSL, XML signature, WS-Security
Firewall Traversal	work in progress	uses HTTP port 80
Events	CORBA event service	N/A



Integrazione tramite ESB

- ❑ Problemi nel campo dell'integrazione
 - diversi ambienti di esecuzione, di management, ...
 - sistemi e servizi proprietari differenti, anche legacy
 - confini fisici

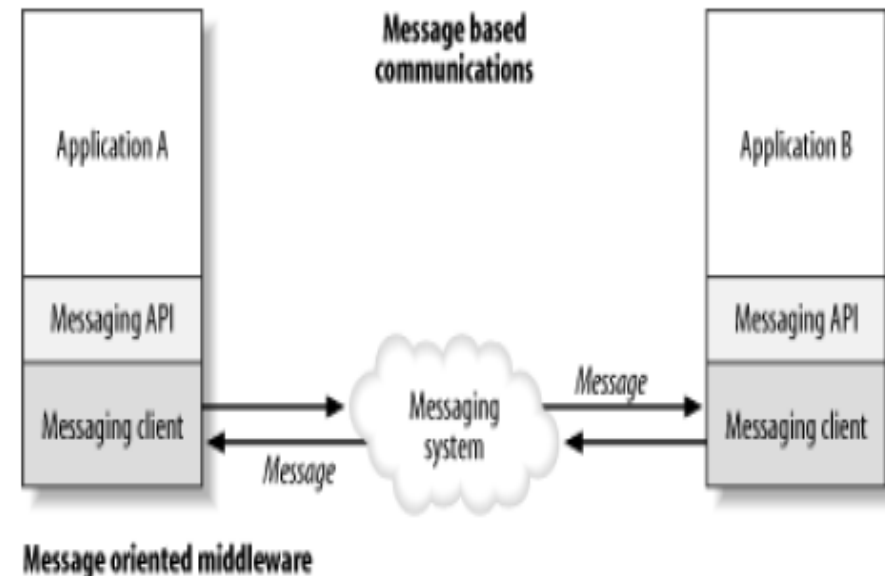
- ❑ *“A new form of **enterprise service bus (ESB)** infrastructure – **combining message-oriented middleware, Web services, transformation and routing intelligence** - will be running in the majority of enterprises by 2005.”* [Roy Schulte, Vice President of Gartner Inc., 2002]

- ❑ *ESB come infrastruttura software per l'integrazione, basata su “standard”, che combina messaging, Web services, data transformation e routing intelligence per connettere tra loro in modo **debolmente accoppiato** e affidabile un numero significativo di applicazioni eterogenee, **mappate come servizi***
[David Chappell, ESB, O'Reilly, 2004]



Message Oriented Middleware ed ESB

- ❑ Infrastruttura per la comunicazione tra applicazioni basata sullo ***scambio di messaggi***
 - Modello sincrono vs. ***modello asincrono***
 - Modello p2p vs. ***pub-sub***
- ❑ Caratteristiche generali
 - ***Disaccoppiamento***
 - Gestione dei “topic”
 - Controllo degli accessi
 - Struttura messaggi
 - ***QoS configurabile***





Service Oriented Architecture (SOA)

Paradigma basato su:

- ❑ **Servizi autonomi**
- ❑ **Interfacce** che definiscono contratti tra Consumer e Provider
- ❑ **Messaggi** che compongono le operazioni invocabili sui servizi
- ❑ **Registri** dei servizi
- ❑ Possibilità di **comporre i servizi in processi di business**

Obiettivo è ottenere:

- ❑ **Accoppiamento debole**, e quindi...
- ❑ **Flessibilità** di business
- ❑ **Interoperabilità** tra le applicazioni
- ❑ **Indipendenza** rispetto alle tecnologie di implementazione



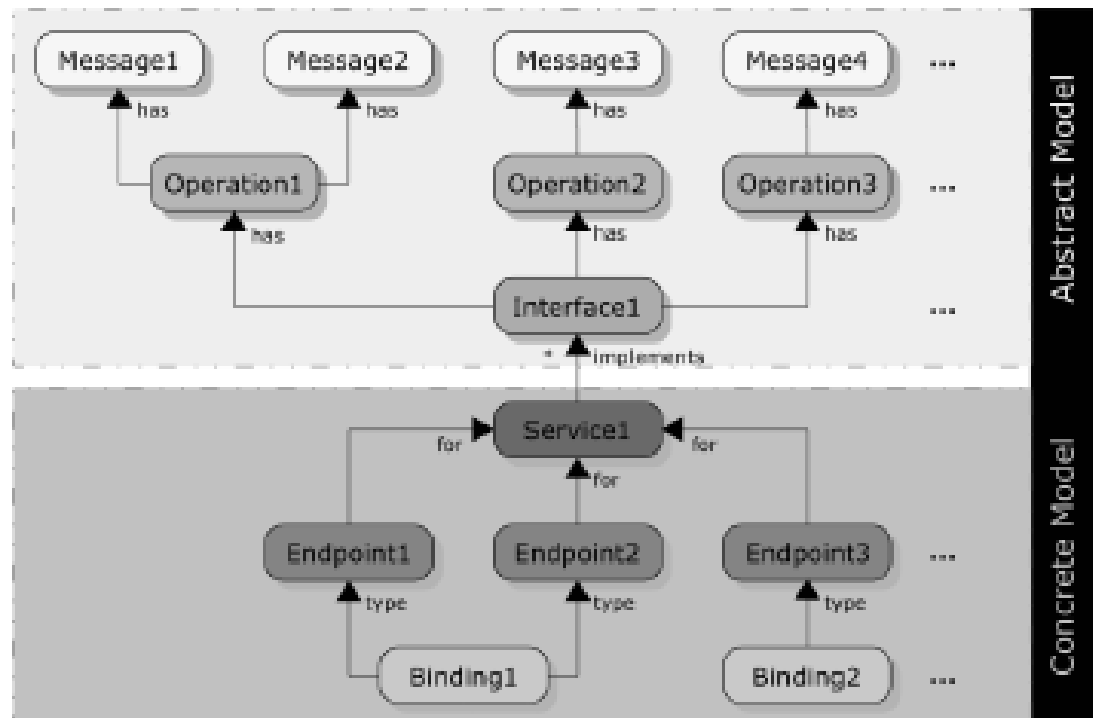
Web Service

- ❑ **Infrastruttura** per l'interazione tra applicazioni basata sul concetto di “**servizio**”
- ❑ Sfrutta **essenzialmente tre tecnologie**:
 - **SOAP** → descrizione messaggi scambiati e binding protocollo di trasporto utilizzato (usualmente HTTP)
 - **WSDL** → descrizione servizio svolto dal provider
 - **UDDI** → discovery di servizi → directory service (pattern “find-bind-invoke”)
- ❑ SOAP e WSDL si basano su XML



Web Service Description Language (WSDL)

WSDL e chiara separazione fra **livello astratto** (definizione operazioni di servizio e struttura messaggi) e **livello concreto** (binding – per ogni interfaccia, uno o più endpoint con indirizzo di rete e protocollo), tipico di tutte le soluzioni SOA





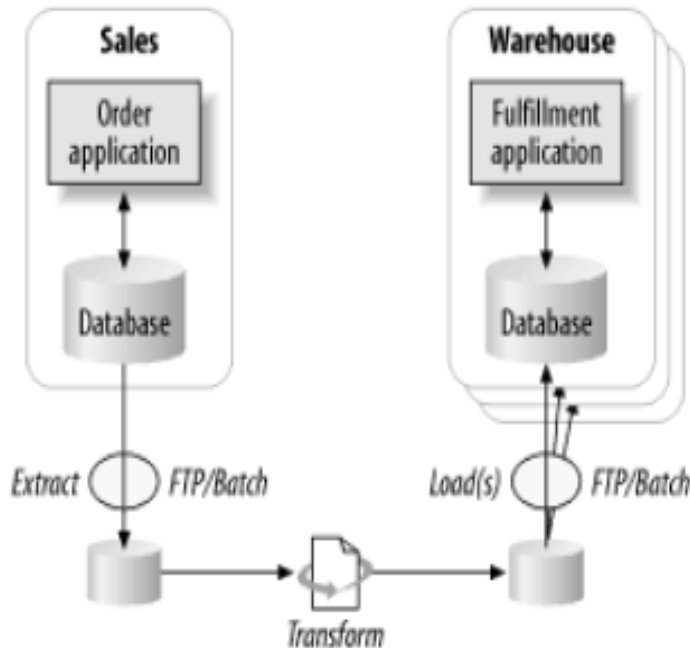
Approccio Convenzionale all'Integrazione

- ❑ Solo **10% delle applicazioni è integrato** (dati Gartner Inc.) e solo 15% di queste sfruttano middleware ad hoc...
- ❑ Com'è collegato il restante 85%? Perché le tecnologie passate si sono rivelate inadeguate?

Architettura “casuale”

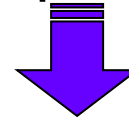
- ❑ È il risultato della composizione di diverse soluzioni adottate per i diversi sistemi nel corso degli anni
- ❑ Col tempo presenta:
 - alti costi di mantenimento
 - **rigidità** (applicazioni tightly-coupled)
 - prestazioni insoddisfacenti (**scarsa scalabilità**)

Enterprise Application Integration (EAI)



Approccio ***Extract, Transform, and Load***

- ❑ Download e upload continui

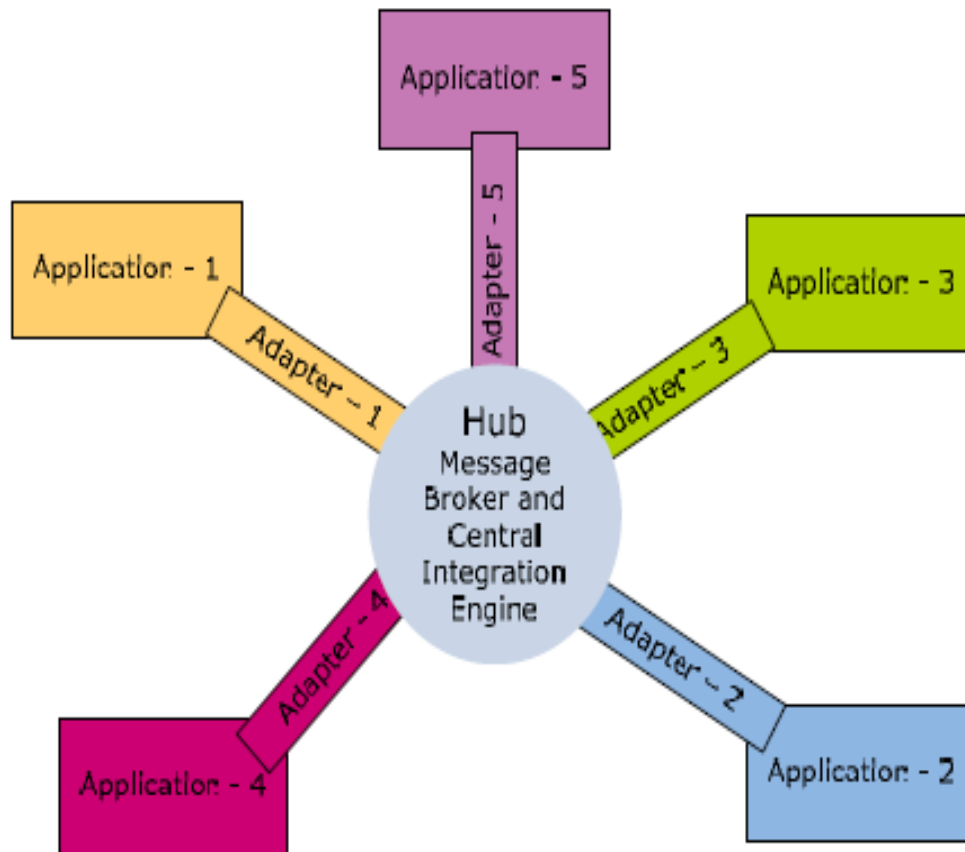


rischio di introdurre incoerenze

- ❑ Alta latenza del processo

- ❑ Applicazione ***principi architetturali*** allo scopo di ***integrare efficacemente*** applicazioni di un'organizzazione
- ❑ ***Broker + orchestration engine***
- ❑ Due topologie principali: ***hub-and-spoke o bus***
- ❑ Implementazioni generalmente proprietarie e alto costo

EAI: Hub-and-Spoke

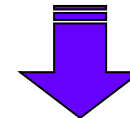


PRO:

- facilità di gestione (centralizzata)

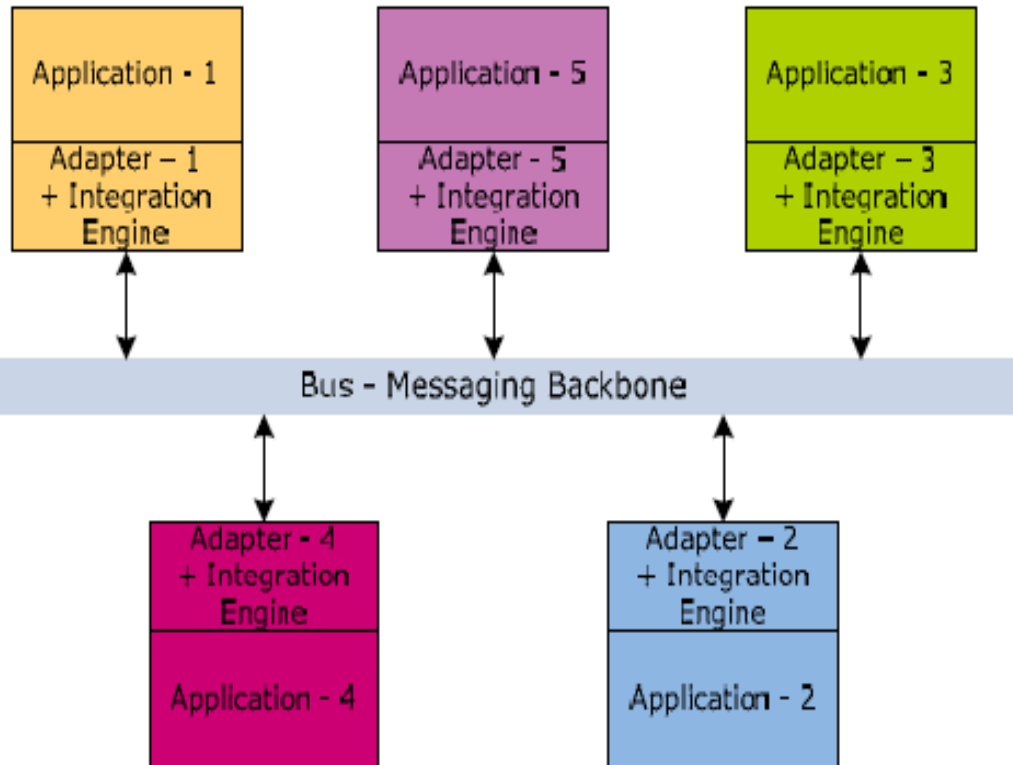
CONTRO:

- hub punto critico di centralizzazione
- ridotta scalabilità



architettura federata

EAI: Bus di Interconnessione



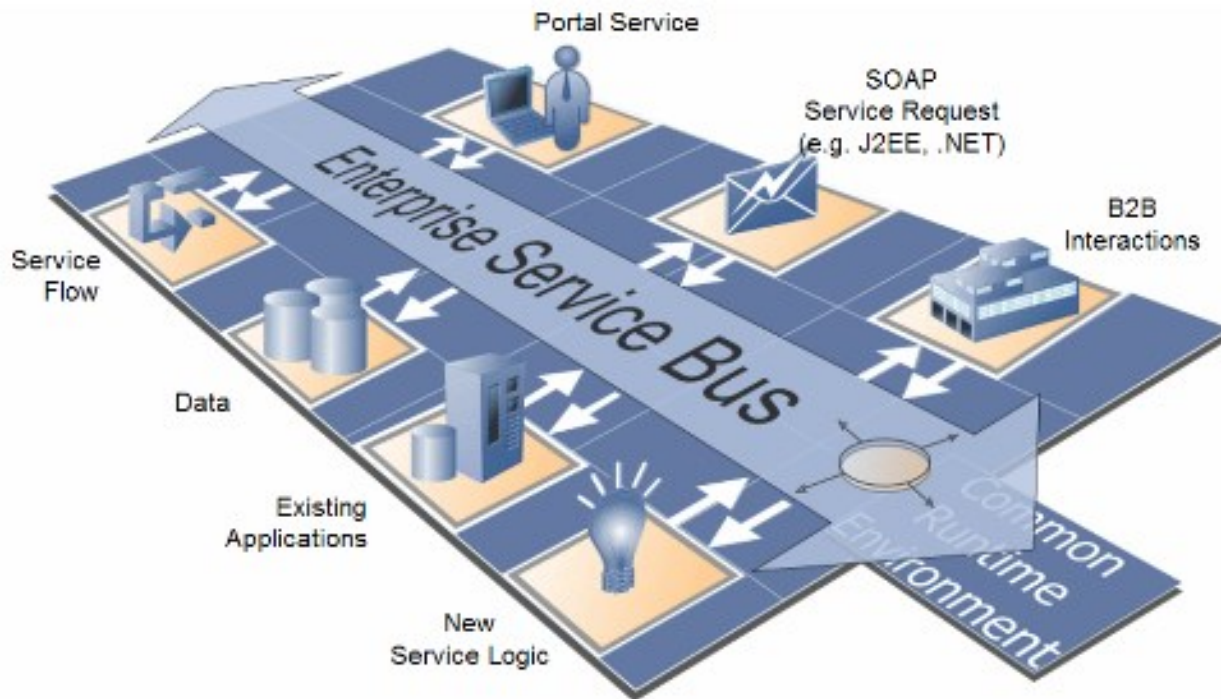
□ PRO:

- maggiore scalabilità (architettura meno centralizzata)

□ CONTRO:

- a costo di maggiori difficoltà di gestione

Enterprise Service Bus



***Middleware per
l'integrazione di
servizi basato sul
paradigma SOA***

Caratteristiche:

- ❑ ***Uniformità*** nell'accesso ai servizi
- ❑ Capacità di ***orchestrarne l'integrazione*** mediandone le incompatibilità
- ❑ Funge da ***registro dei servizi***
- ❑ Agisce come ***punto centralizzato di gestione***



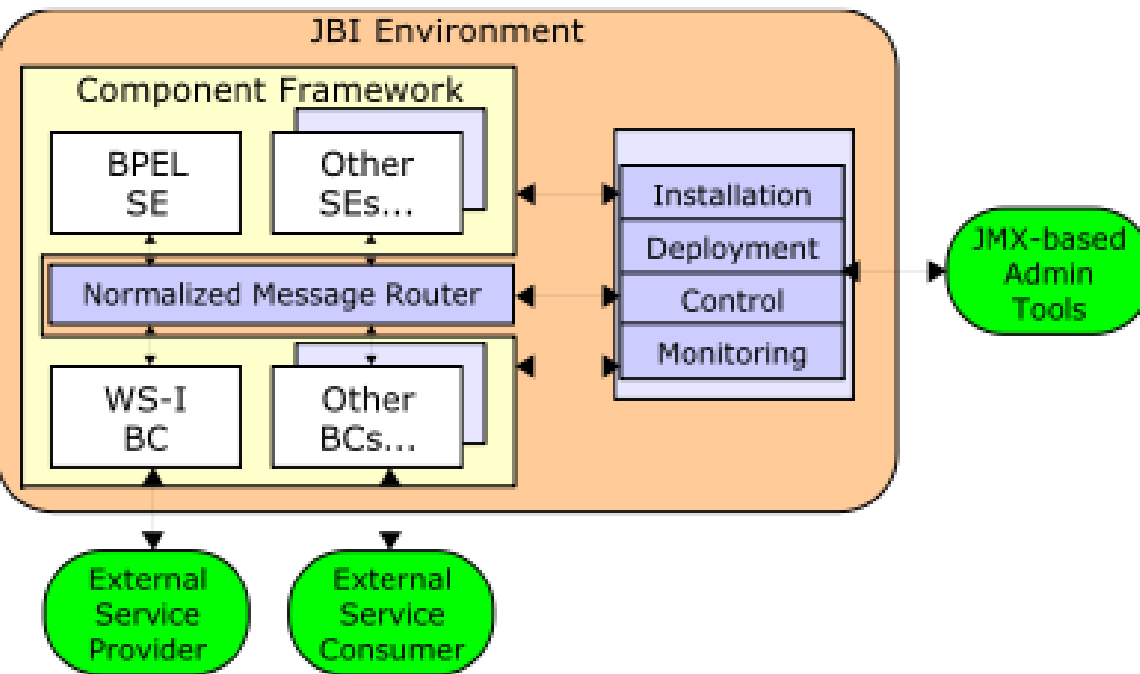
ESB: Concetti Chiave

- ❑ Architettura altamente distribuita e integrazione basata su standard
- ❑ **Servizi di orchestration**
- ❑ **Autonomia** delle singole applicazioni
- ❑ Real-time throughput; servizi di auditing e logging
- ❑ Consente adozione incrementale

Invocazione dei servizi:

- ❑ Servizi completamente disaccoppiati
- ❑ Pattern “find-bind-invoke” è gestito automaticamente dall'infrastruttura
- ❑ Progettista deve solo definire ***itinerario logico che i messaggi devono seguire***; servizi si “limitano” a inviare e ricevere messaggi...

Java Business Integration (JBI)



- ❑ **JSR 208**, Java Business Integration (JBI), 2005 (specifica Java di ESB Standard)
- ❑ **Servizi offerti e consumati da componenti**
- ❑ Interazione tra componenti mediata da **Normalized Message Router (NMR)**
- ❑ **Gestione** attraverso strumenti **JMX-compliant**

Tipologie di componenti:

- ❑ **Service Engine** – responsabili della logica di business, offrono servizi implementati in Java; forniscono **logica di integrazione e di trasformazione verso altri componenti**; a loro volta possono utilizzare i servizi degli altri SE
- ❑ **Binding Component** – **consentono fruizione di servizi esterni** all'environment da parte di servizi interni e viceversa. Fungono da **adattatori di protocollo**



Normalized Message Router (NMR)

Comunicazione tra componenti all'interno del bus ***NON*** è ***diretta***. NMR che agisce da mediatore fra i vari componenti

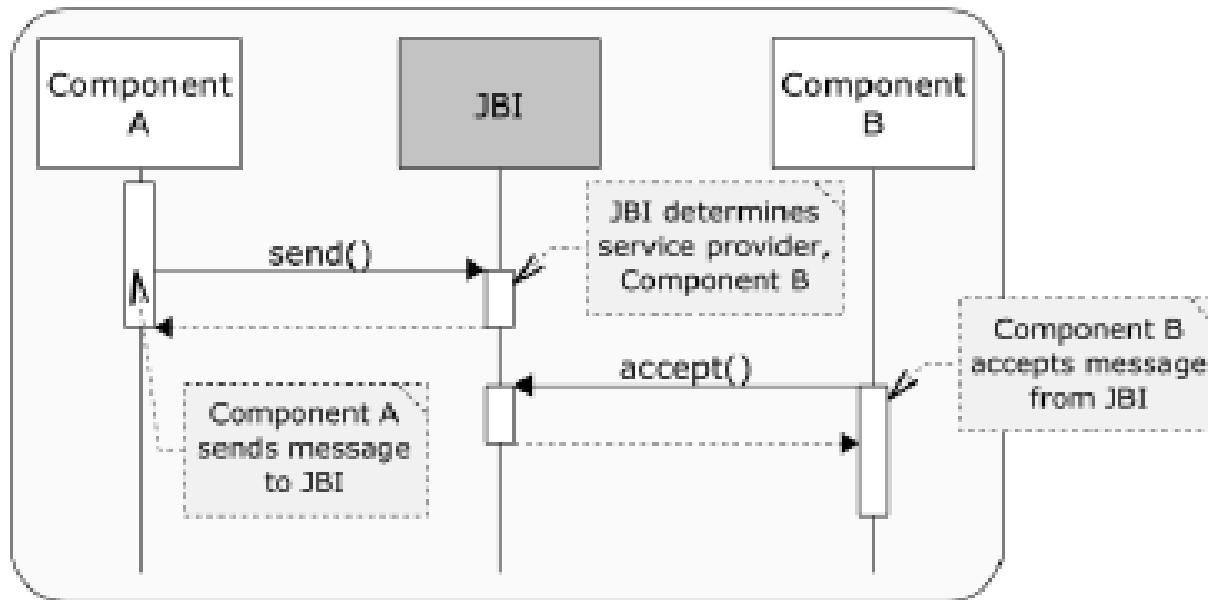
❑ ***Compito dell'NMR***

- Routing dei messaggi tra 2 o più componenti
- Disaccoppiare Service Consumer da Service Provider garantendo un basso accoppiamento tra i componenti JBI

Messaggi in formato XML

- ❑ ***Comunicazione "technology-neutral" tra endpoint.*** Normalized message scambiati sono definiti in formato indipendente e neutrale da qualsiasi specifica applicazione, tecnologia o protocollo di comunicazione
- ❑ Trasformazioni di formato → ***trasformazioni XSLT***

Interposizione di JBI nello Scambio di Messaggi



Componenti SOA e modello a scambio di messaggi basato su interposizione:

- Elevato grado di disaccoppiamento tra componenti
- Possibilità di operare su messaggi (trasformazioni) in modo trasparente

JBIM Message Exchange Pattern

JBIM supporta almeno **4 pattern di scambio messaggi**:

- ❑ **In-Only** per interazione one-way
- ❑ **Robust In-Only** per possibilità di segnalare fault a livello applicativo
- ❑ **In-Out** per interazione request-response con possibilità fault lato provider
- ❑ **In Optional-Out** per provider con risposta opzionale e possibilità di segnalare fault da provider/consumer

