Università degli Studi di Bologna
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Scuola di Ingegneria

Corso di

Reti di Calcolatori M

WEB SERVICES

Antonio Corradi

Anno accademico 2014/2015

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 organizzazione di **interfaccia standardizzata 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 aggregazioni

Service-Oriented Architecture SOA

SOA è un modello in cui ogni interazione è via **servizi che sono del tutto indipendenti dalla piattaforma e dalla comunicazione** e dalla rete
Le operazioni sono definite in modo **indipendente tra loro** e dalla **piattaforma** in cui sono realizzati e con proprietà note e negoziate prima dell'uso

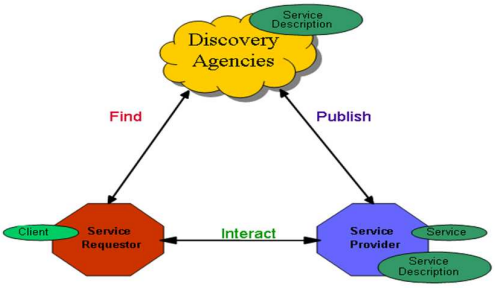
Service-Oriented Architecture SOA architettura abilitante

Un servizio deve avere una **interfaccia precisa** con cui essere invocato e deve restituire **alcuni risultati specifici**

Il formato deve essere noto a tutti gli utilizzatori e deve essere disponibile dinamicamente

SOA deve offrire le capacità
di **descrivere, trovare e comunicare**
con i servizi disponibili

I servizi sono singoli e non organizzati o composti



CONCETTO di Servizio

Un **servizio** come **astrazione** di un **processo, risorsa o applicazione, di business** che possa essere **standardizzato** come **interfaccia** e possa essere **pubblicato e noto**
I servizi sono:

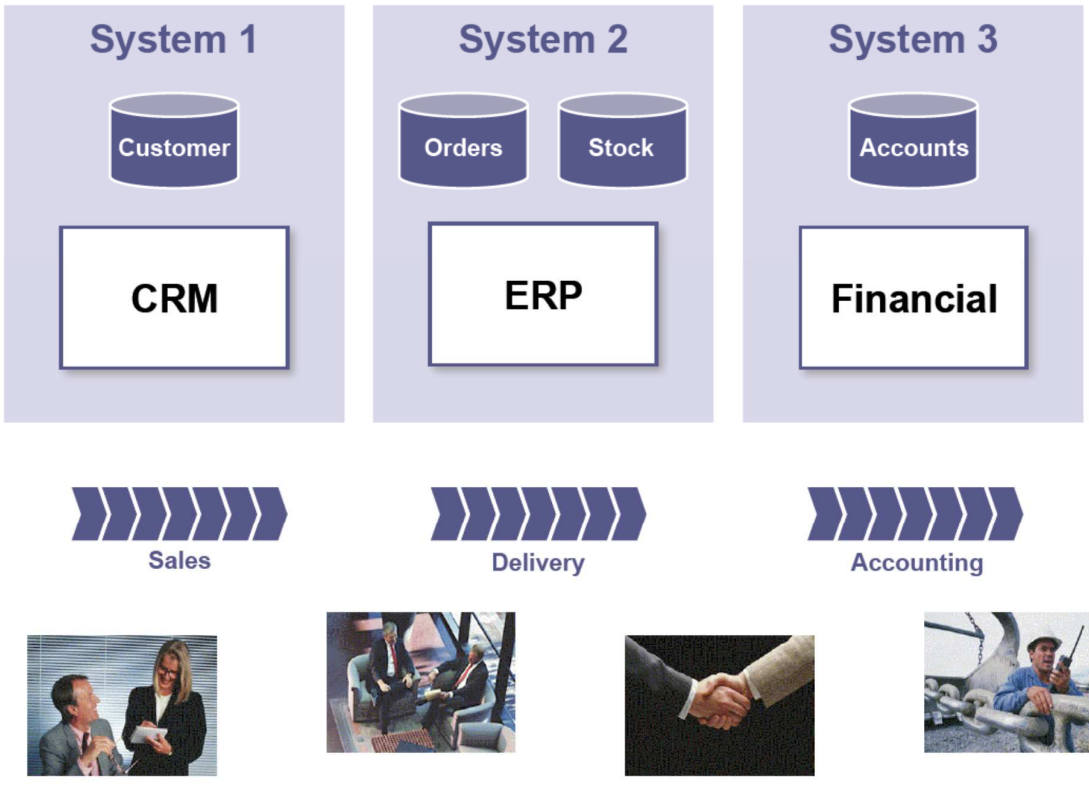
- **riusabili**, cioè possono essere riutilizzati anche in altri contesti di uso
- **formali**, sanciscono in modo non ambiguo i termini e il contratto dello scambio di informazioni
- **a scarso accoppiamento**, non presuppongono nessuna conoscenza dell'ambiente di uso
- **a black box**, nascondono la logica e i dettagli della soluzione specifica scelta

PRINCIPI dei Sistemi Orientati ai Service

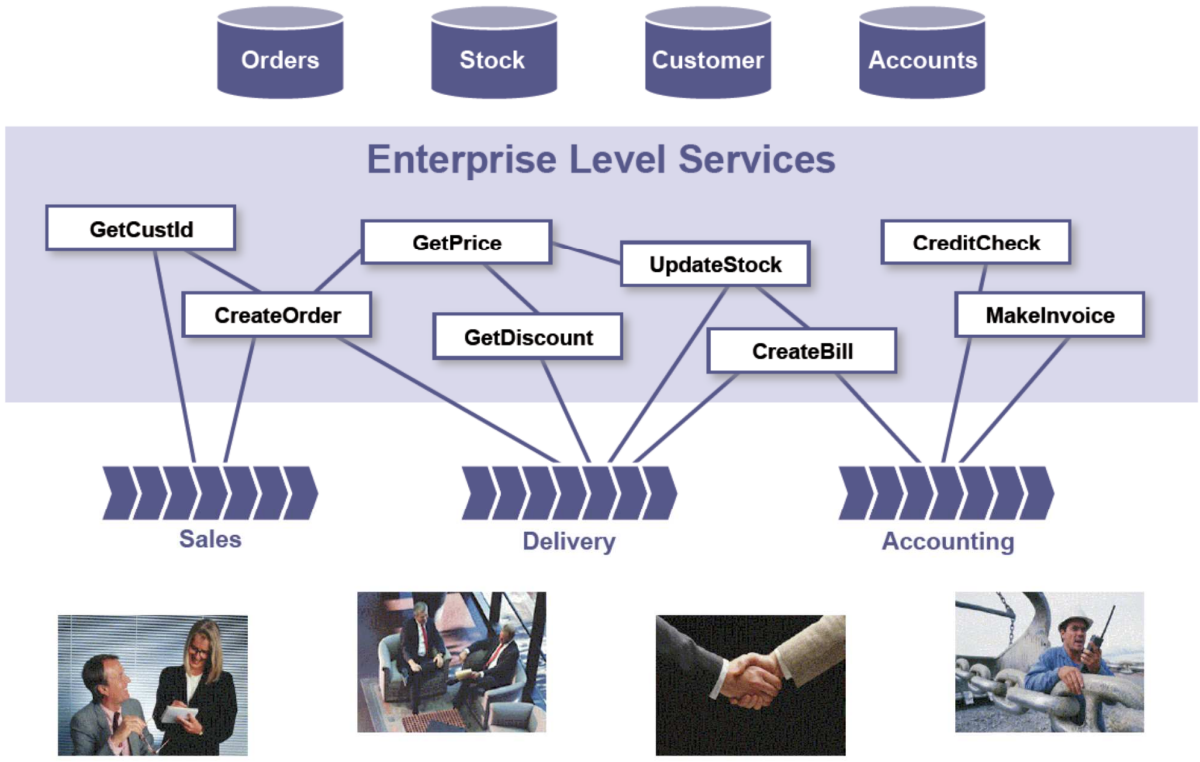
Un **servizio** deve essere fornito dalle piattaforme che lo espongono a chi ne abbia bisogno e ne faccia richiesta, purché ne conosca la interfaccia (che deve essere nota)
I servizi devono essere:

- **autonomi**, non devono dipendere dal contesto e devono essere capaci di autogestione
- **senza stato**, lo stato interno deve essere minimizzato (**stateless**) e le invocazioni non devono dipendere una dall'altra
- **soggetti a discovery**, devono potere essere ritrovati dinamicamente in base alla interfaccia, in modo standard e facile da utilizzare
- **componibili**, cioè possono consentire di formare altri servizi nuovi (**composizione**)

ARCHITETTURE basate su APPLICAZIONI

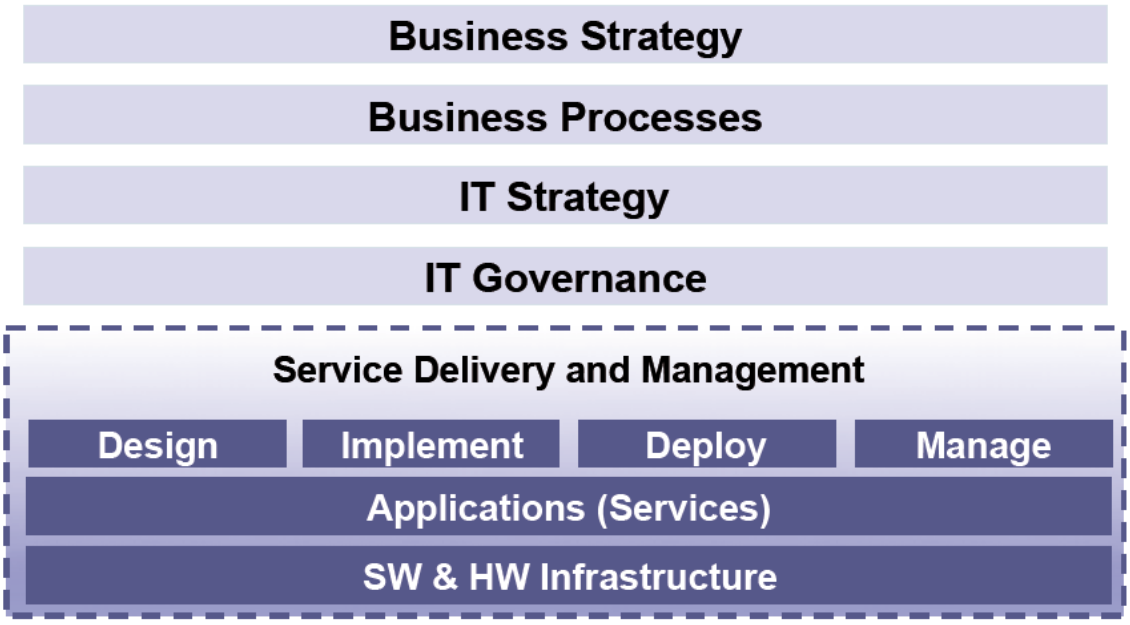


ARCHITETTURE orientate ai SERVIZI



ENTERPRISE Information Technology

Strategie Enterprise propongono **applicazioni** variabili molto rapidamente e fortemente **critiche** per l'operatività aziendale



Web Services

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 la estensione a XML (eXtensible Markup Language)
Prospettiva di massima apertura

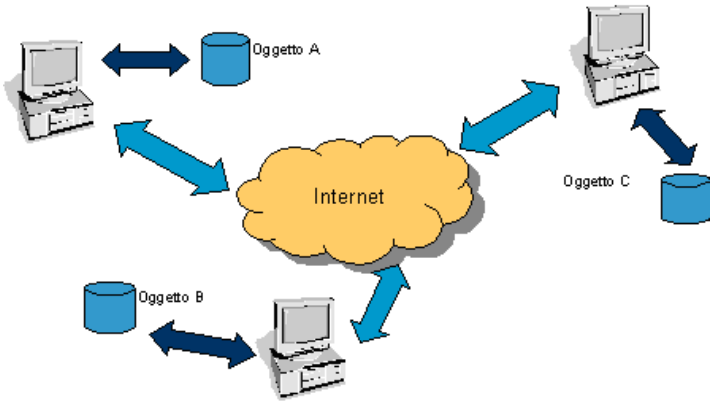
MIDDLEWARE nel supporto a sistemi

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



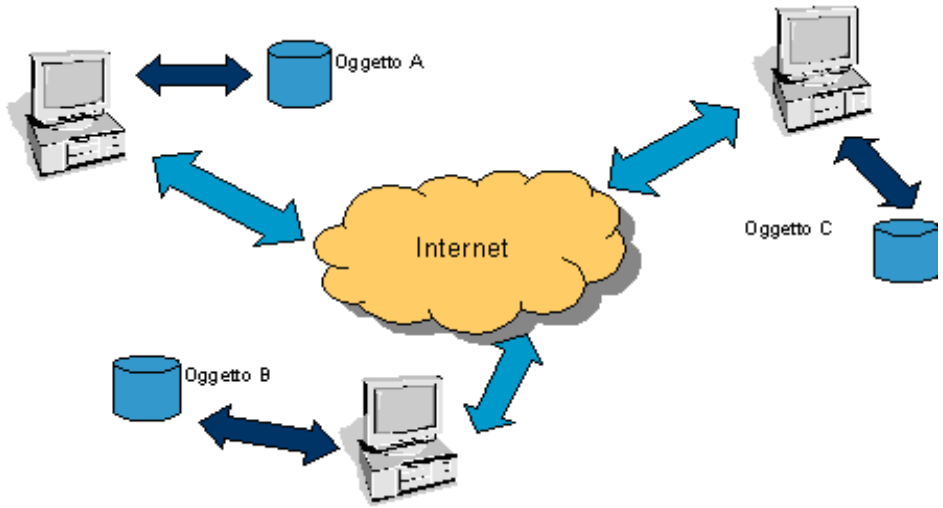
Con garanzia di superare eterogeneità e mantenere i corretti livelli di sicurezza

MIDDLEWARE ad OGGETTI

MIDDLEWARE diffusi basati

Modello Cliente-Servitore

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



MIDDLEWARE ad OGGETTI

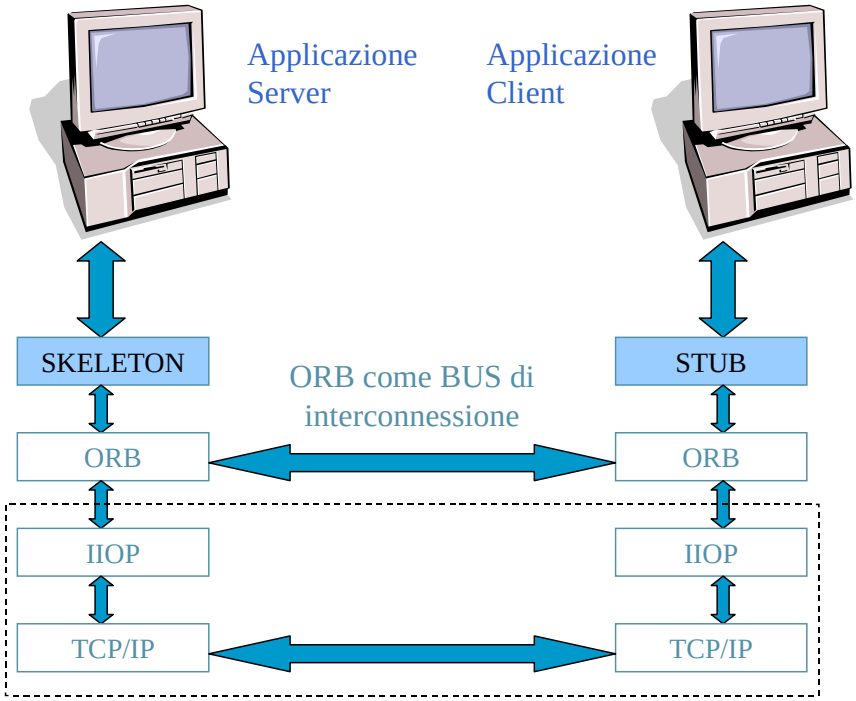
CORBA

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

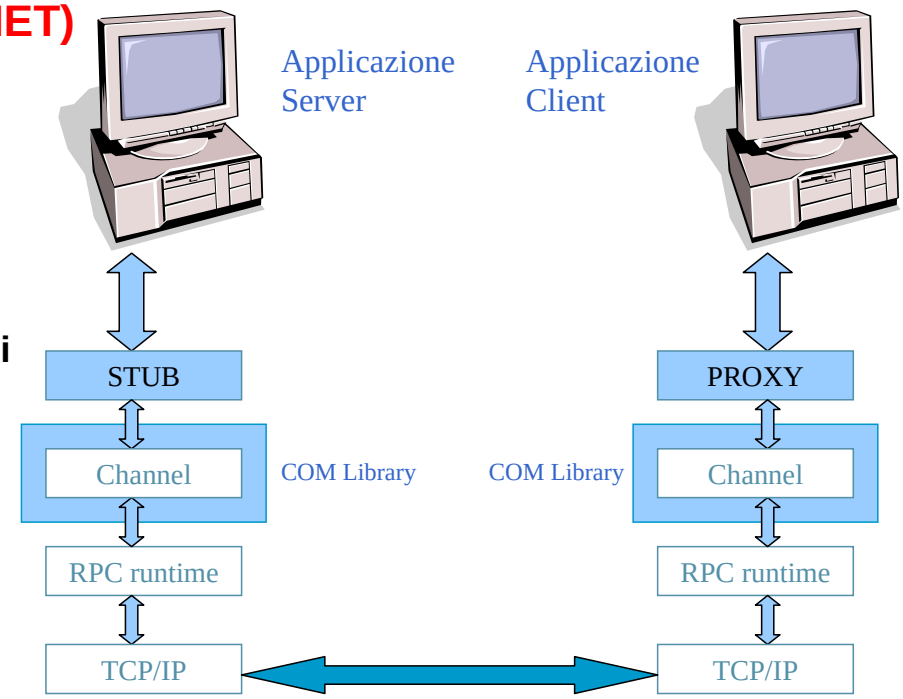
COM, DCOM,… (.NET)

Interazione sincrona

Ambiente basato su componenti multilinguaggio proprietario

Integrazione di sistemi e strumenti esistenti

Apertura verso ambienti esterni anche legacy

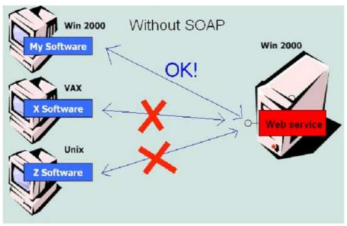


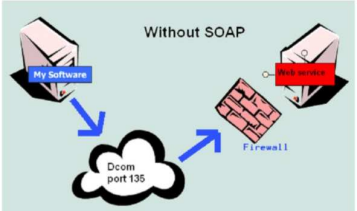
MIDDLEWARE e SICUREZZA

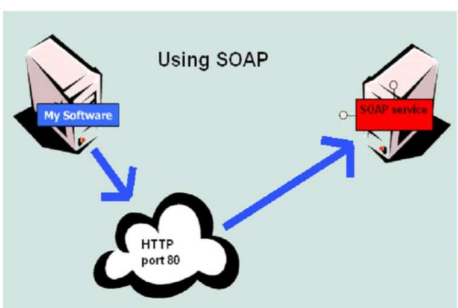
I firewall tendono a introdurre vincoli nel passaggio di funzionalità e a bloccare richieste di operazioni insicure

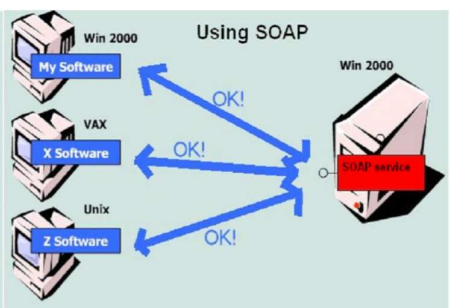
Fornitura di servizi WEB in ambiente distribuito

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









Web Services come middleware

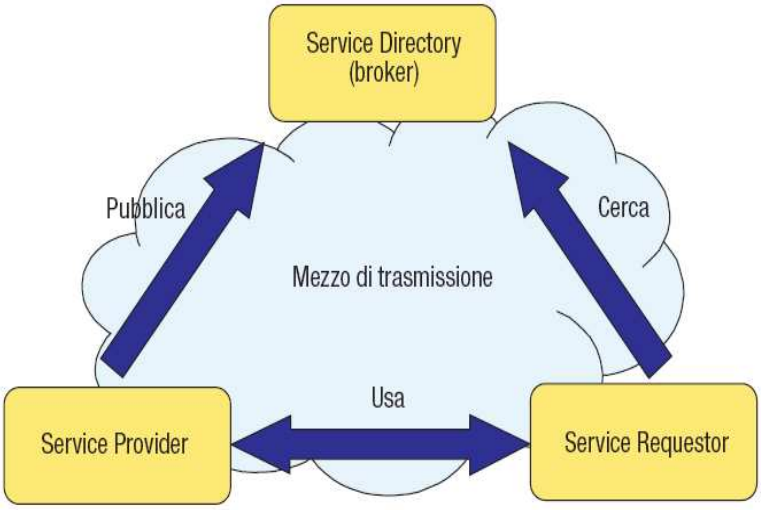
MIDDLEWARE come **organizzazione standard** per la fruizione di servizi **(Richiedente/Provider e Broker) in ambito Web-compatibile**

Servizi offerti dal Provider

Richiesti quando e se necessario
su esigenza del 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)

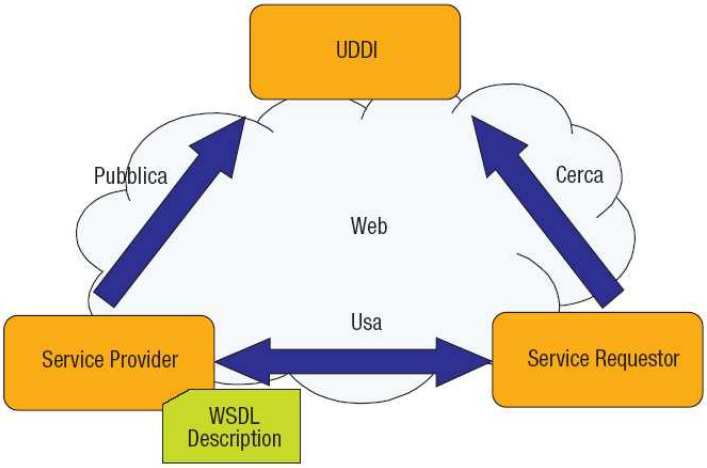
WSDL (Web Services Description Language)

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)



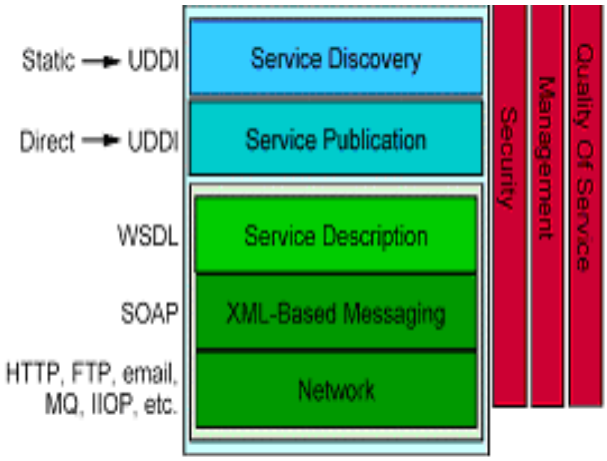
Web Services: Protocolli

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 (con personalizzazione dei tag)

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**, **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 sono specificati 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)

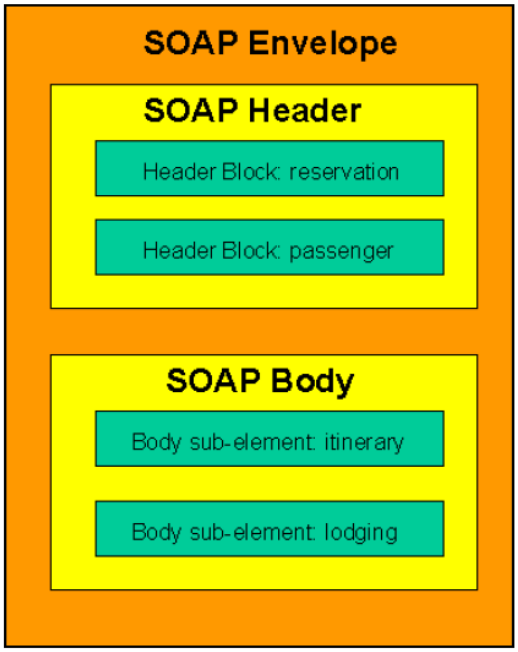
Protocollo SOAP

```
<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

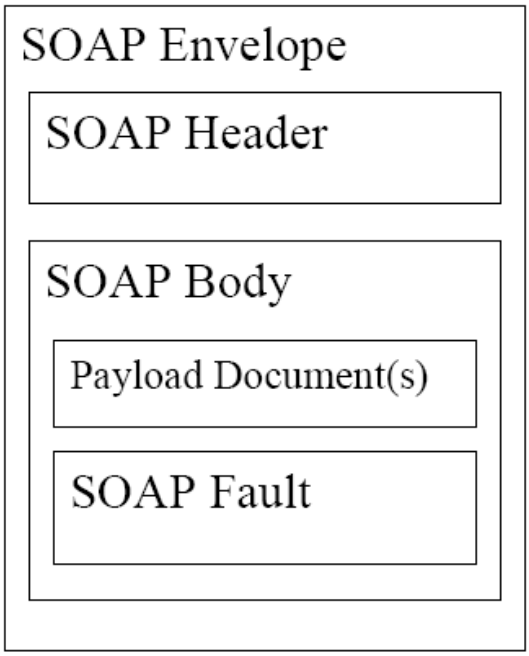
Envelope incapsula il **contenuto del messaggio**

Header destinato a contenere **informazioni aggiuntive**

Informazioni accessorie di sicurezza

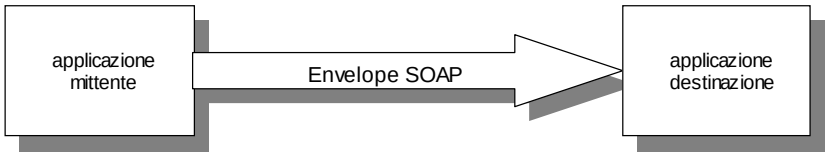
Body incapsula le **richieste e le risposte** (in genere, il messaggio da comunicare)

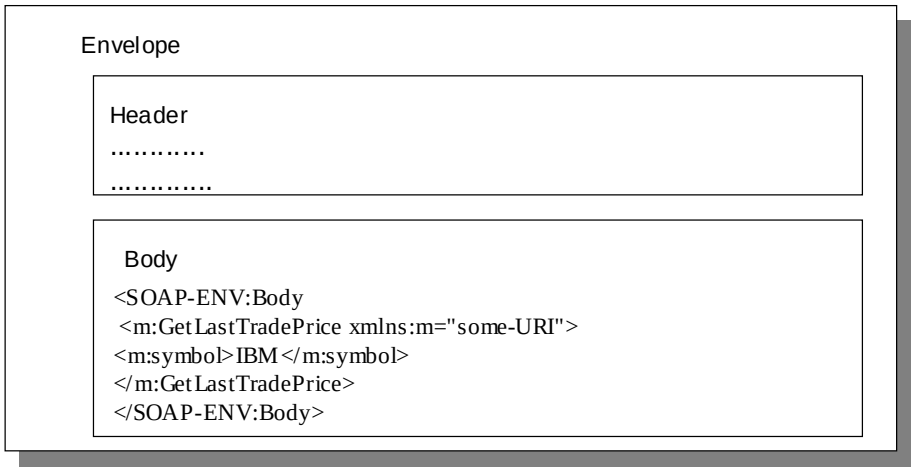
Fault incapsula eventuali casi distinti di **errore ed eccezione**



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 e XML

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

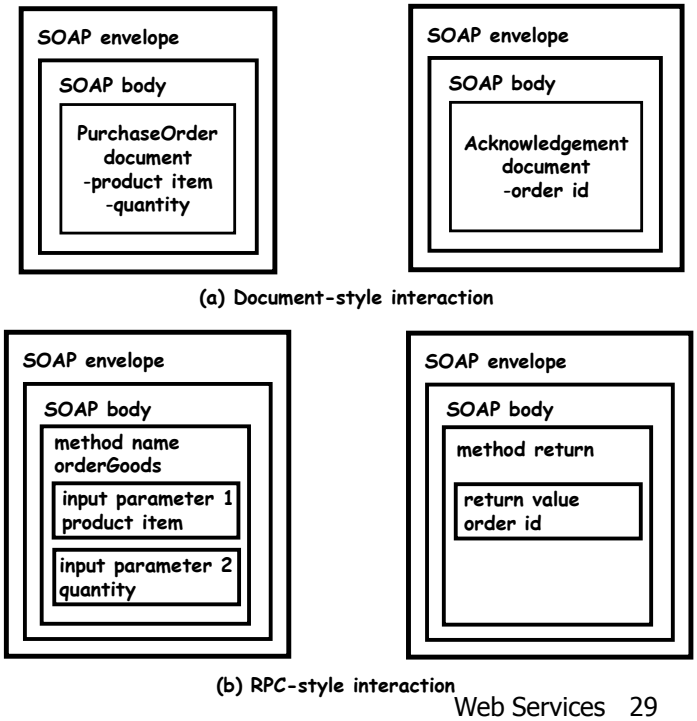
Anche **Interaction style**: **document o RPC style**

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



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">
         <Pricel>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)

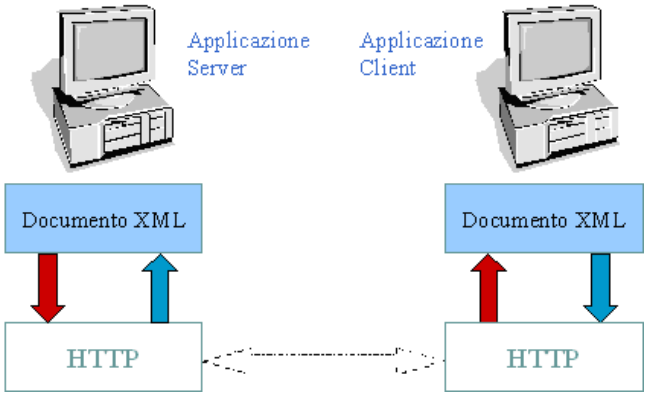
Per raggiungere la **indipendenza** dalle diverse **realizzazioni dei servizi** che stiamo richiedendo e dalla **eterogeneità delle diverse** architetture

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

SOAP

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**)



Web Services

Torniamo alla definizione iniziale

Web Services come realizzazione SOA

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

WDSL 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**

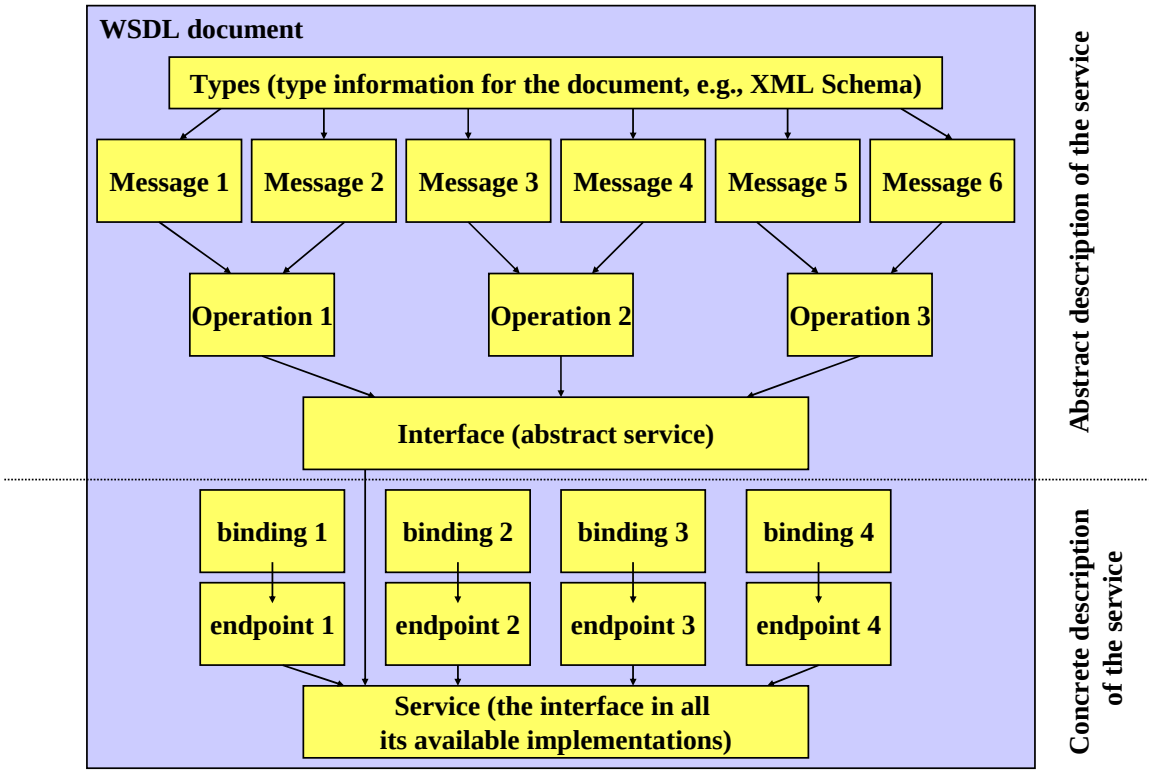
– si crea una **richiesta SOAP**

– si invia la richiesta **SOAP al servizio** e si attende la **risposta**

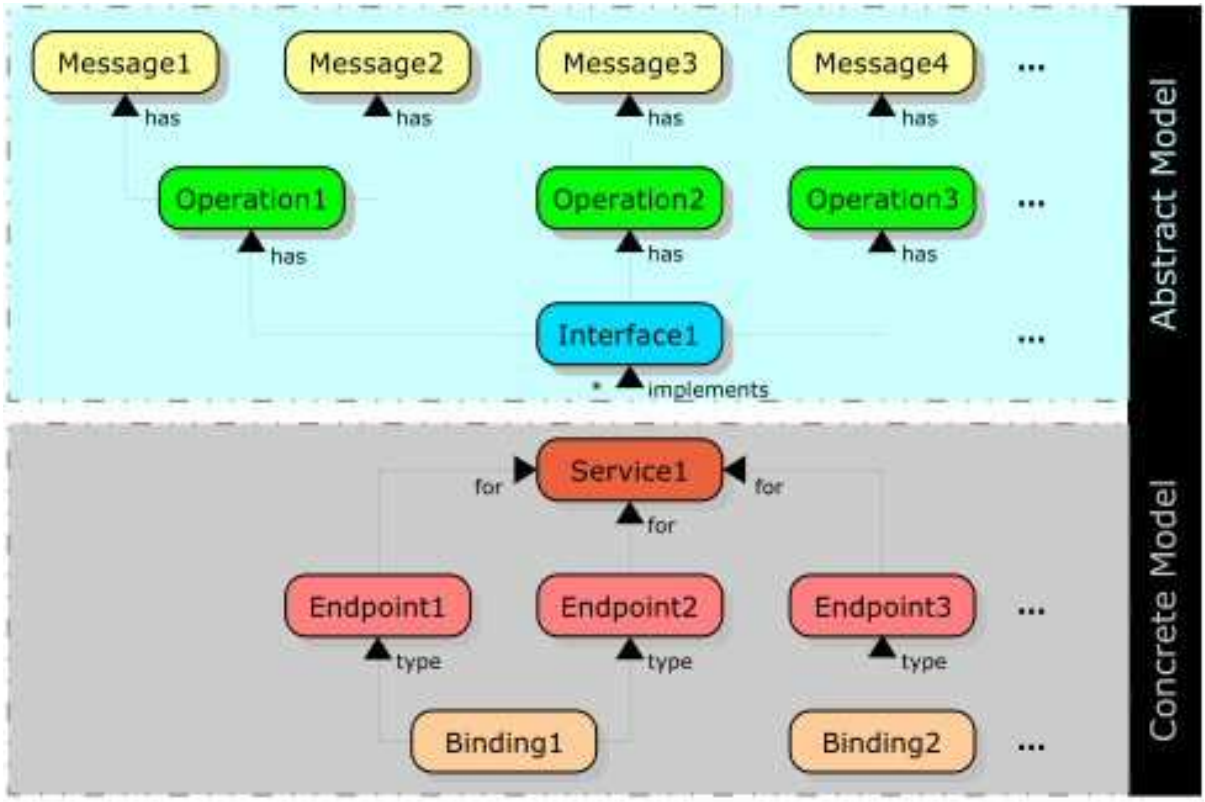
La logica è quella di avere il massimo del supporto e della facilità nel procedere, fino alla completa automazione da parte di un middleware

Alcune parti di WSDL sono molto simili a un IDL

Elementi di base di WSDL



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 (tipi)

Uno scambio di messaggi è definito operation

astratti

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)

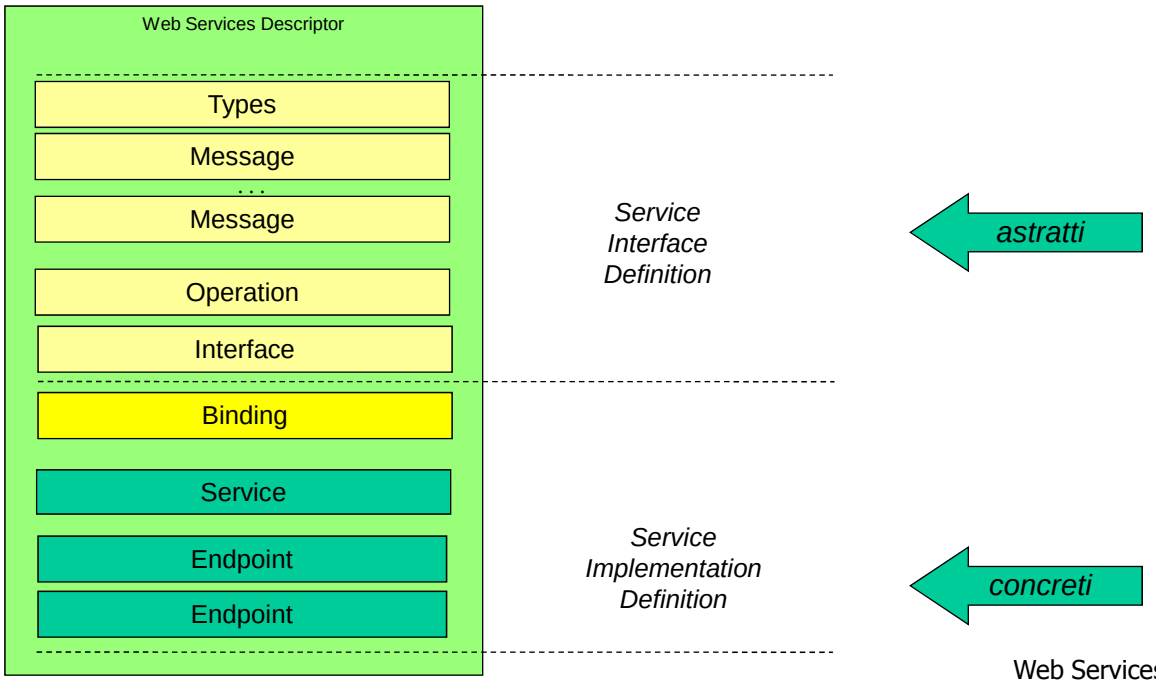
concreti

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

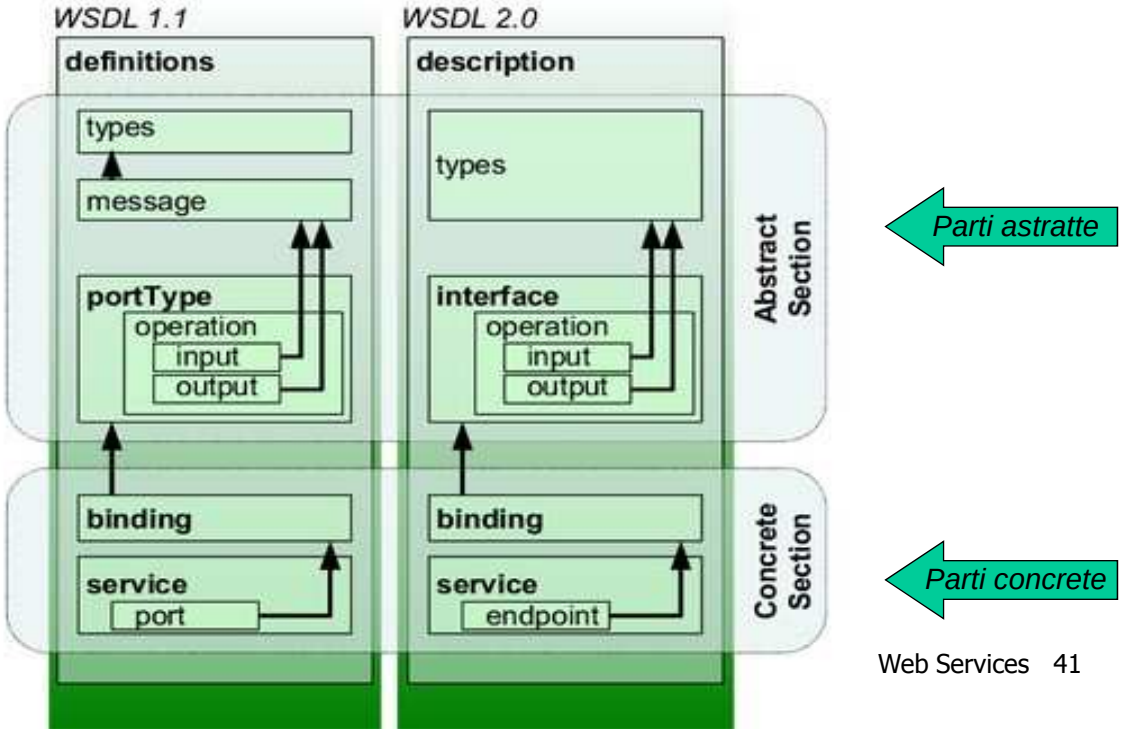
WSDL 2.0

WSDL deve descrivere sia le parti astratte sia le parti concrete



WSDL 2.0 e WSDL 1.1

Due standard WSDL in poco tempo, ma poco diversi



SERVIZIO in WSDL

Un documento WSDL è formato da questi elementi

in corrispondenza ad una parte dell'applicazione

– Parti **astratte**

Type, Message, Operation, Interface

– Parti **concrete**

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

• **type** **tipo di dato** in un messaggio usando XML Schema

• **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 di WSDL

• **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

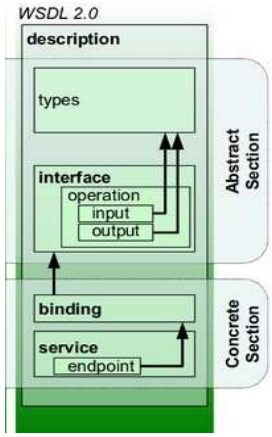
Permette di raggruppare tutte le interface, in modo che sia visibile quali siano gli endpoint supportati da un determinato servizio

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

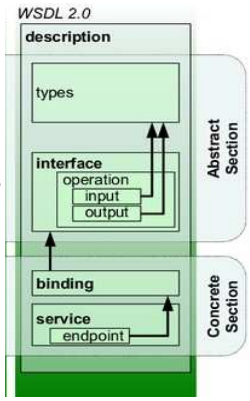
```
<types> <schema>
    <element name="TradePriceRequest">
      <complexType>
        <all>
         <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="TradePrice">
       <complexType>
         <all>
           <element name="price" type="float"/>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

WSDL Interface

Ogni Interface prevede modalità diverse

sincrone e *asincrone*

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

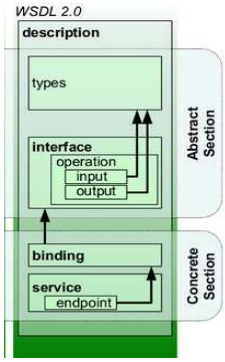
Solicit_Response: l'operazione prevede l'attesa da parte del service provider di una risposta sollecitata con una richiesta mandata dal provider al cliente

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):



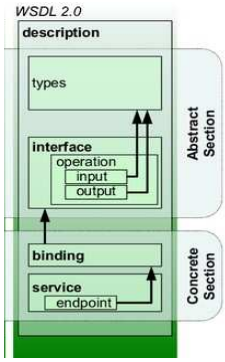

```
<binding name="StockQuoteSoapBinding“
 type="tns:StockQuoteInterface">
  <soap:binding>
    <operation name="GetLastTradePrice“>
      <soap:operation

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:StockQuoteSoapBinding">
   <soap:address location="http://www.stockquote.com"/>
  </endpoint>
</service>
```

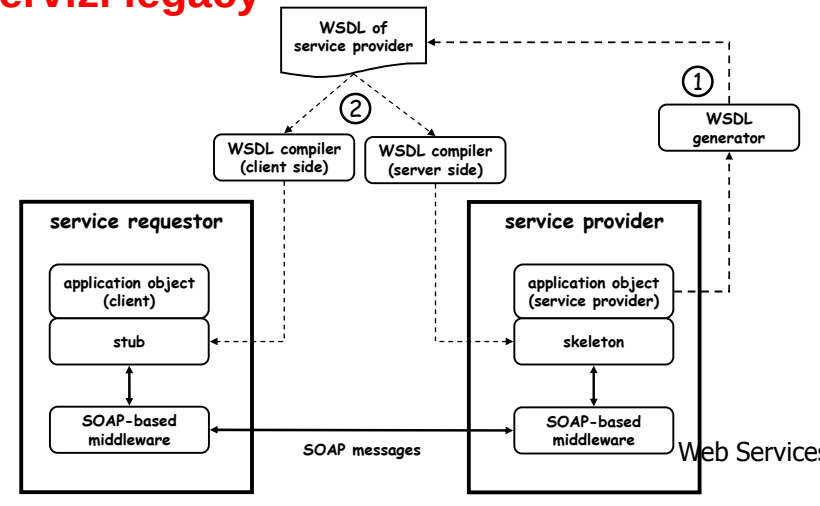
Oltre a ultimi dettagli concreti

Uso di WSDL

WSDL può essere usato:

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

Si usa XML per generare dalle **specifiche logiche** dei **wrapper per servizi legacy**



Universal Description Discovery & Integration

Universal Description Discovery & Integration language (UDDI)

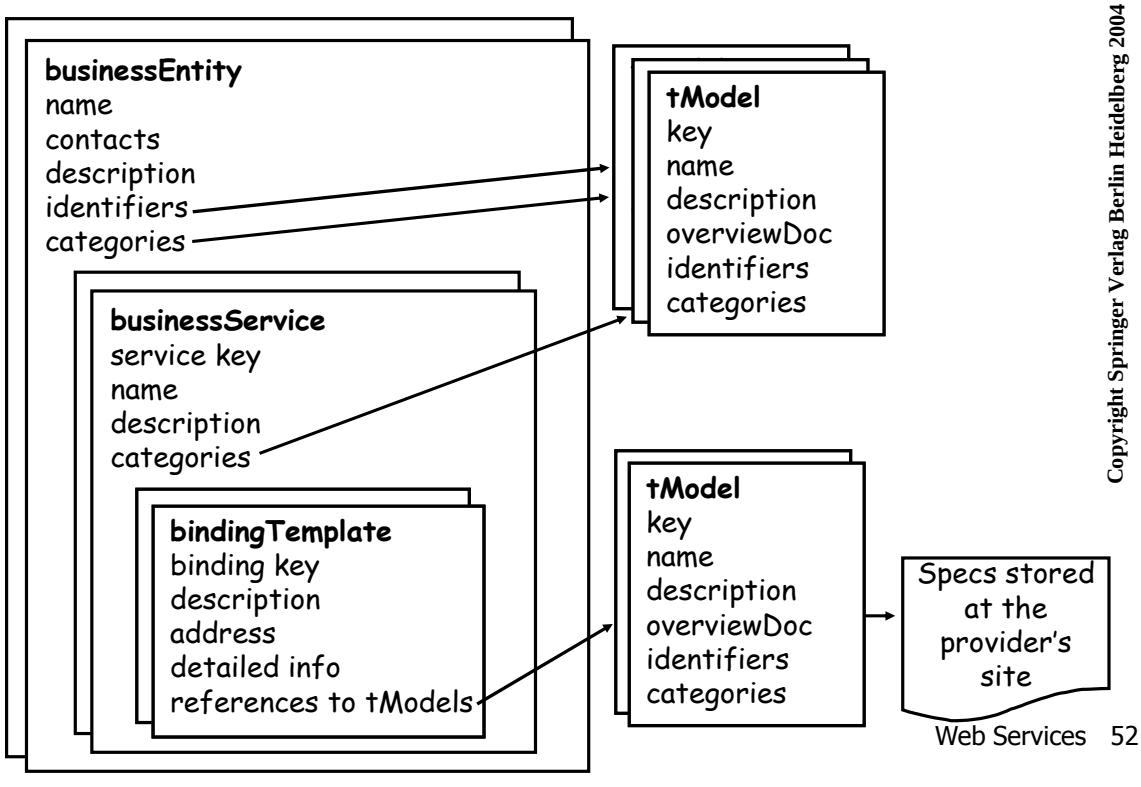
Necessità di **sistemi di discovery**, ossia di **naming/directory** per i **Web Services** visto che i servizi possono **non essere noti in descrizione o locazione** (parte **astratta o concreta**)

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

UDDI: DESCRIZIONE FORMALE



INFORMAZIONI BUSINESS 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 tecnica** 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

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**

Si noti la specializzazione delle possibilità e la specifica molto predeterminata (era necessaria? è sufficiente?)

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 dovrebbe essere un **servizio globale condiviso** tra **server differenti sparsi** in tutto il mondo, **anche se non organizzati secondo una struttura gerarchica**

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

I server non sono coordinati (come si poteva immaginare, mancando protocolli ad-hoc e anche la volontà di fare pubblicità a servizi di altri)

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

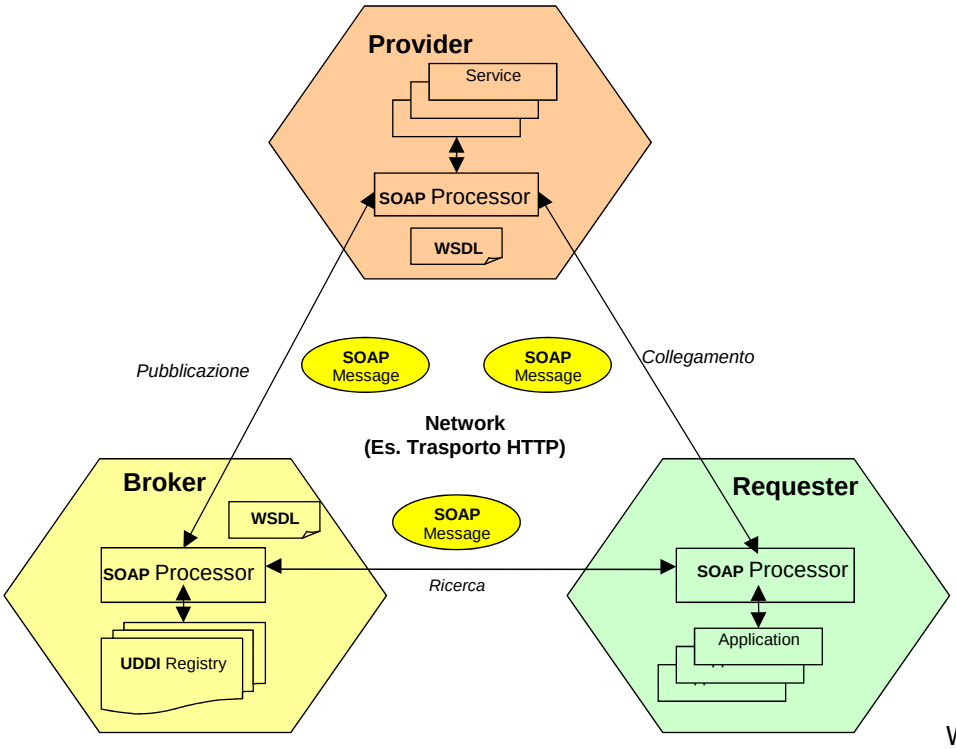
Ogni server mantiene traccia dei publisher e di cosa sia pubblicato

Solo chi ha pubblicato un servizio autorizzato a modifica/cancellazione

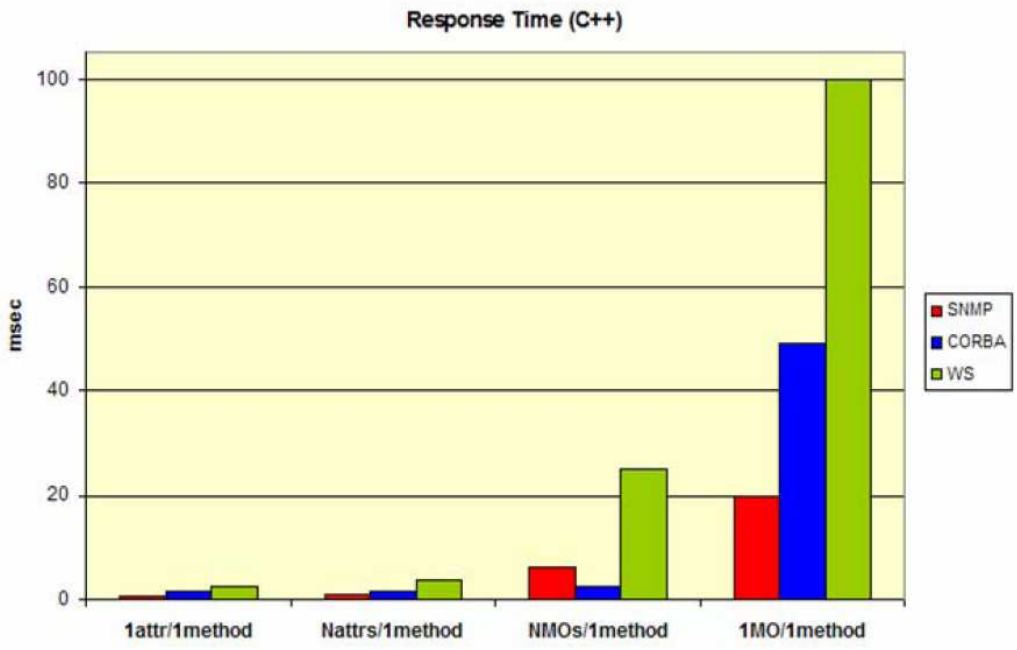
Classificazione dei Registri

pubblici, **privati**, **condivisi** (*semi-privati*)

INTEGRAZIONE PROTOCOLLI WS



Performance di Web Services



Anche più attributi e possibili intermediari, diverse management operations

Performance di Web Services

Problema critico: si consiglia di fare un **monitoraggio continuo** della **performance** con strumenti ad hoc

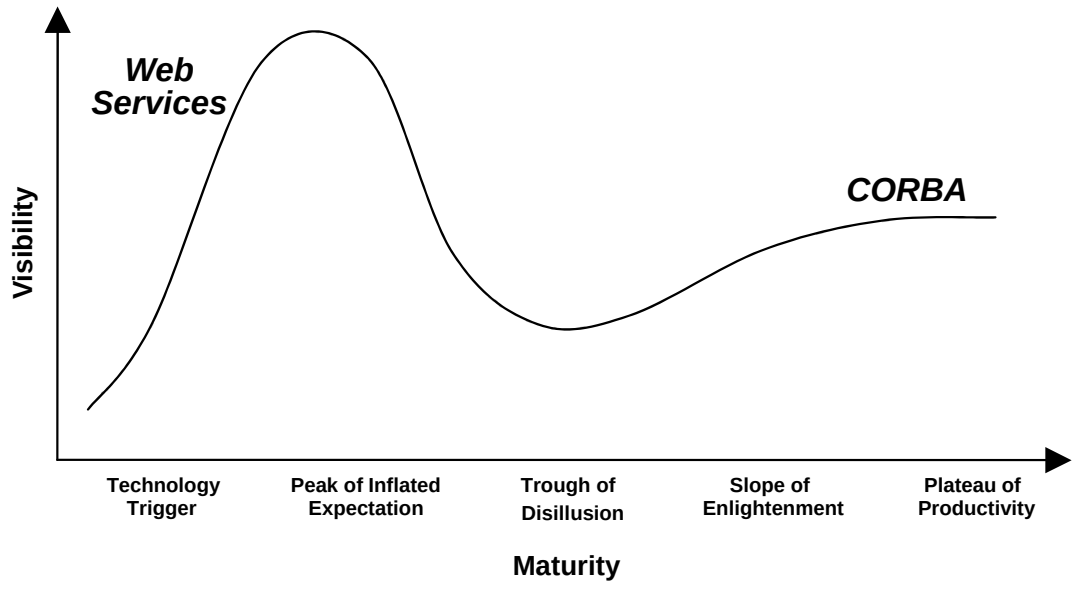
Basic considerations for a high-performance Web services application:

- Reduce Web services requests by using a few highly functional APIs, rather than several simple APIs.
- Design your WSDL file interface to limit the size and complexity of SOAP messages.
- Use the document/literal style argument when you generate the WSDL file.
- Leverage the caching capabilities offered for your Application Server.
- Test the performance of your Web service.

Additional Web services performance features that you can leverage

- In-process optimizations for Web services to optimize the communication path between a Web services client application and a Web container that are located in the same application server process.
- Access to Web services over multiple transport protocols with lower overhead
- SOAP with Attachments API for Java (SAAJ) Version 1.2 provides a programming model for Web services relative to JAX-RPC.
- The Web services tooling generates higher performance custom deserializers for all JAX-RPC beans: redeploying a V5.x application into the V6 runtime
- Serialization and deserialization runtime is enhanced to cache frequently used serializers and deserializers.
- The performance of WS-Security encryption and digital signature validation is improved

Valutazione ed evoluzione tecnologie



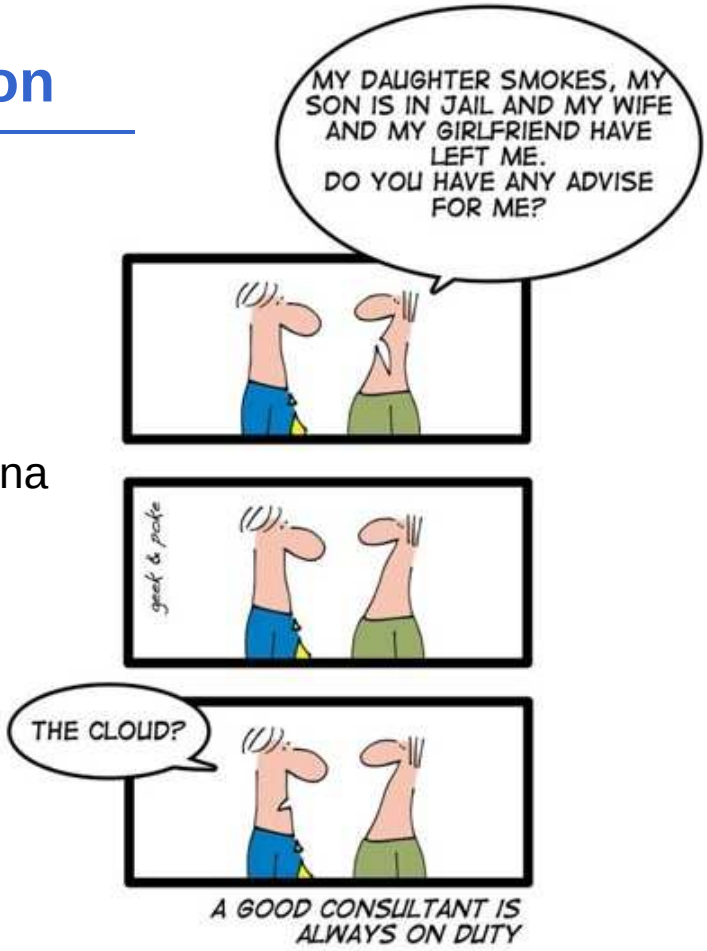
Ogni tecnologia può essere descritta con un ciclo di vita

Inflated Expectation

Stiamo parlando delle fasi di entusiasmo per una tecnologia

Quando una tecnologia può sembrare la medicina tutti i mali (panacea)

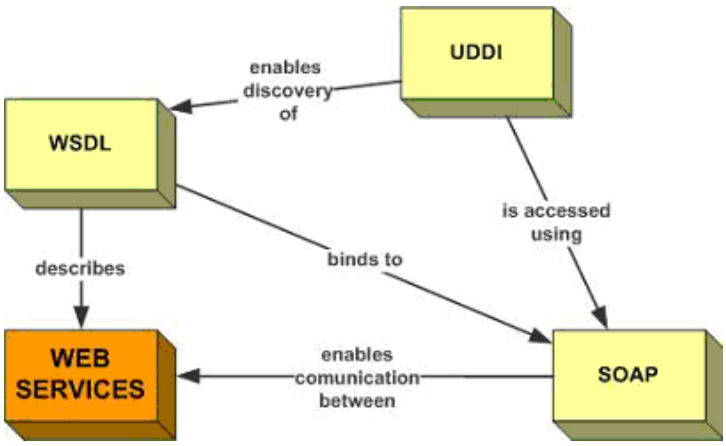
CLOUD



Architettura Web Services

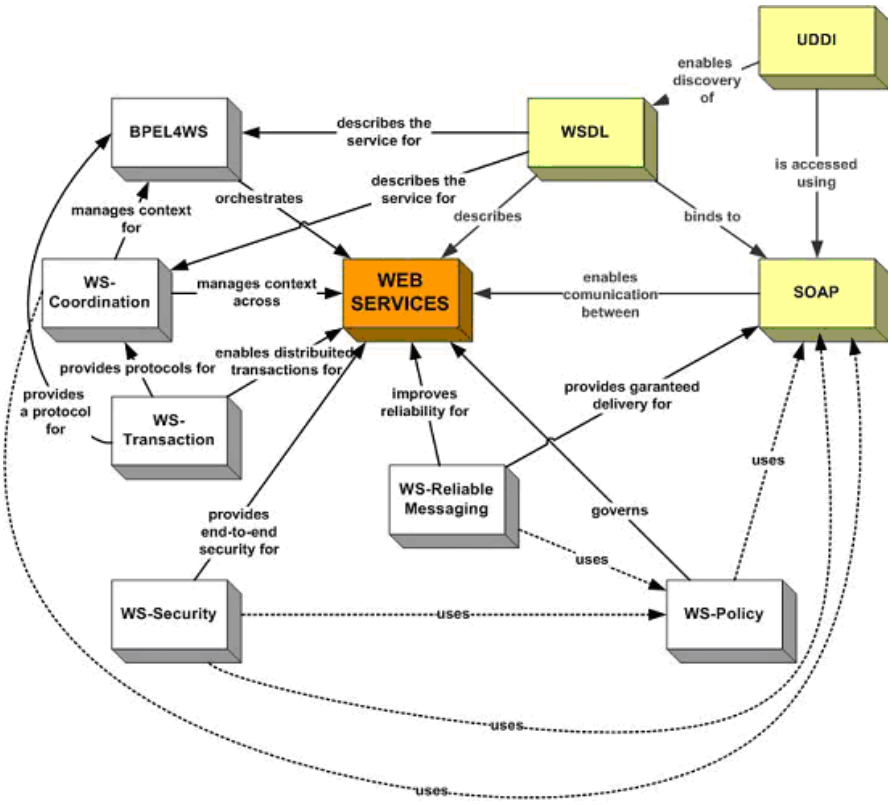
I **protocolli di base** hanno ottenuto successo e applicazione, ma manca ancora tutta la parte di specifica e descrizione del **middleware di supporto** ai diversi servizi e delle necessità di integrazione e delle relazioni tra servizi

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 com-portamenti che si possono descrivere

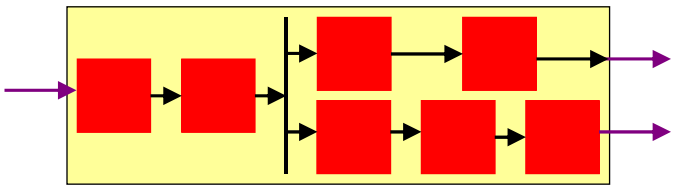
Linguaggi di Workflow

Uno dei problemi più importanti e meno risolti e standardizzati in ambito Enterprise è quello della composizione dei servizi, cioè di come riusciamo a ottenere **nuovi servizi** a partire dagli **esistenti** …

In uno scenario SOA, ci sono servizi forniti, si possono fornire servizi che richiedono altri servizi, ecc. …

I servizi si possono anche mettere insieme in un approccio di composizione

Ad esempio l'**output** di un servizio può essere l'**input** di una altro (in pipeline) o **in flussi molto più complessi**, che possono produrre **altri, nuovi, servizi**



Altri Protocolli Web Services

La **composizione di servizi**?

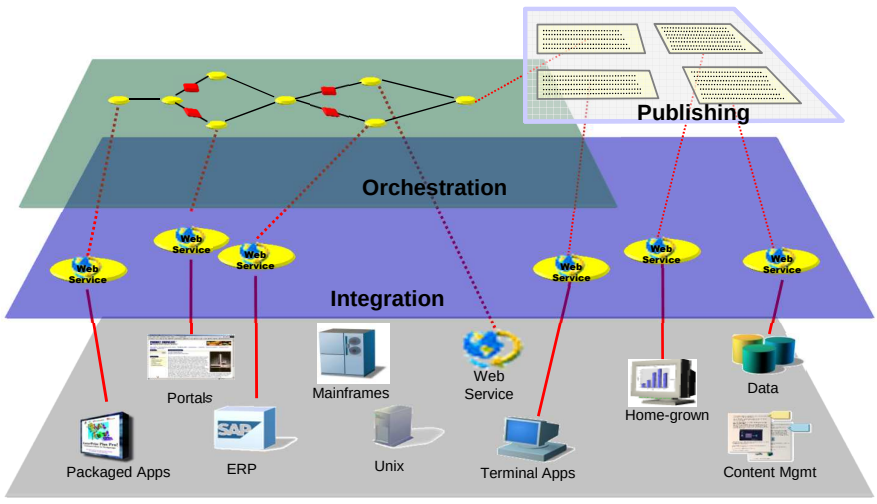
Business **P**rocess **E**xecution **L**anguage for **W**eb **S**ervices (**BPEL4WS**)

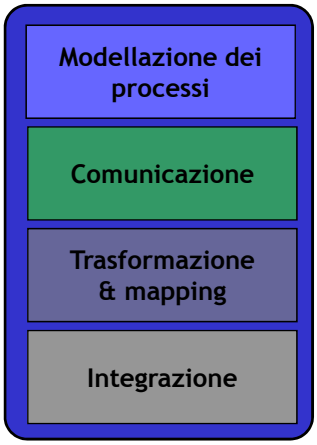
Sul modello WSDL si costruisce un linguaggio di script che permette la invocazione orchestrata dei Web Services come una applicazione di Workflow Management sulla base di costrutti di

- **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

ORCHESTRAZIONE

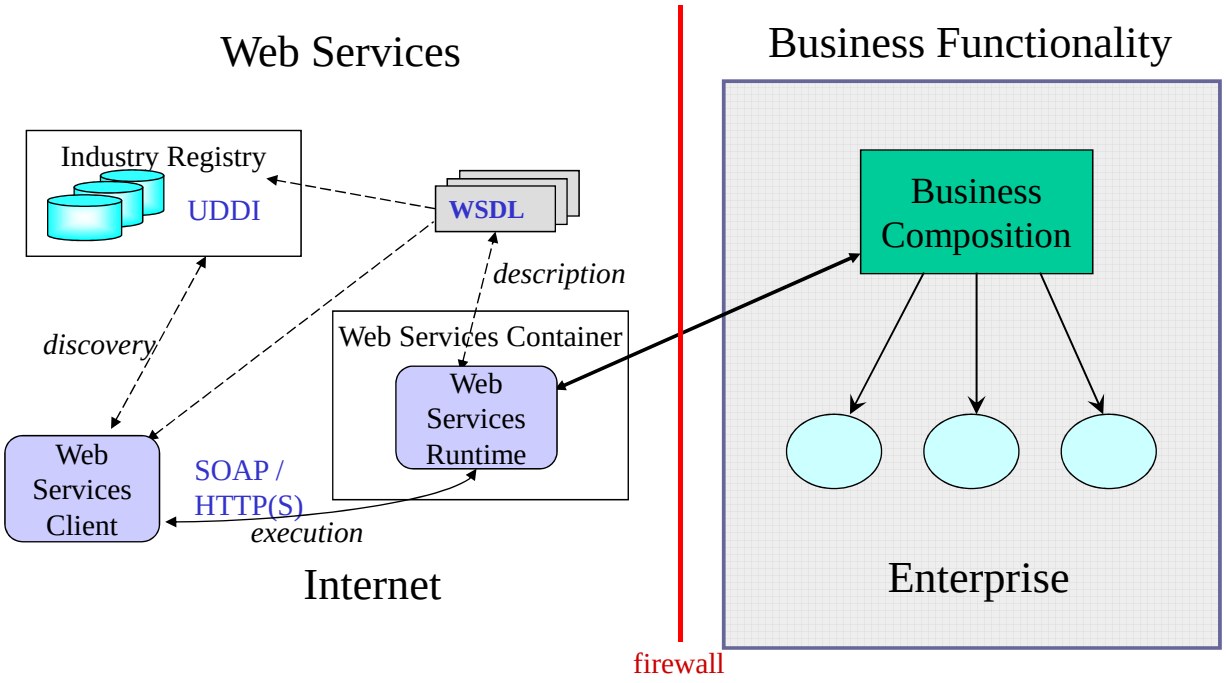
Livello aziendale di supporto alla **integrazione** e alla **orchestrazione**





Verso un'integrazione aziendale

Integrazione di WS e modello business



IN ALTERNATIVA a WS e WS-*

I protocolli **WS-* sono molto pesanti** e a volte **molto vincolanti ma**

non sono il solo modo di realizzare una SOA

Come alternativa **protocollo REST (protocolli REST)**

Protocollo leggero per lavorare in modo compatibile WEB con una granularità più fine e minori costi

Representational State Transfer (REST)

Insieme di principi per la interazione Web compatibile per consentire una **interazione leggera** e **meno vincolata** basandosi sul concetto di **risorsa Web**

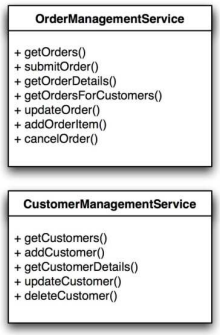
proposta nella tesi di dottorato di R.T. Fielding nel 2000

Representational State Transfer (REST)

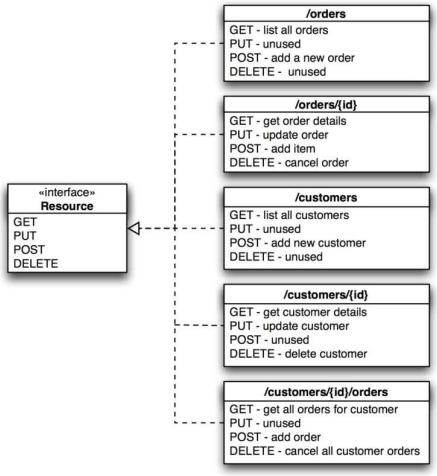
I protocolli **REST sono basati su risorse HTTP e sul C/S per le operazioni su queste:**

- Uso esplicito di **metodi HTTP**
- Protocollo applicativo **senza stato** (stateless)
- URL per **risorse strutturate come direttori** (risorse organizzate in tipici in direttori)
- Protocolli di descrizione usando **XML**, **JSON** (Javascript Object Notation) o entrambi

WS



REST



Intanto … Evoluzioni Web: WEB 2.0

I protocolli **Web sono particolarmente limitanti per la sincronicità del modello di richiesta**: per ogni piccola modifica, la pagina deve essere richiesta tutta al server e riportata al cliente che deve aspettare la risposta del server

Web 2.0

Uso di protocolli asincroni non visibili all'utente per richiedere solo le informazioni necessarie e in modo asincrono

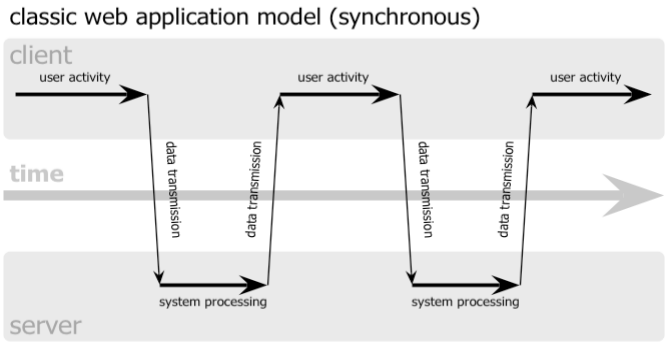
Asynchronous Javascript And XML (AJAX)

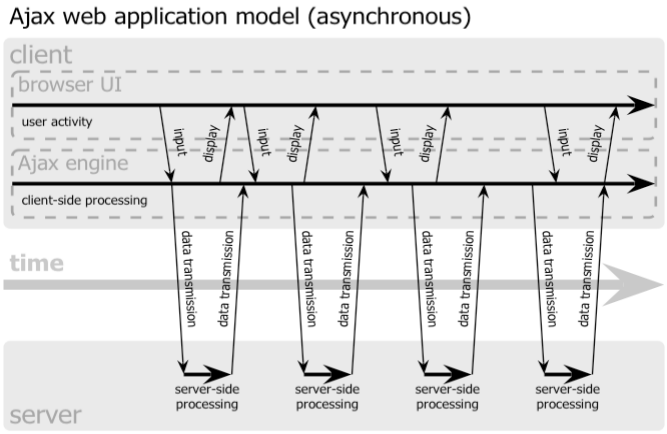
Questo nuovo modo di **usare i servizi Web** è stato anche accoppiato a **nuove applicazioni più facili da usare e collaborative**, arrivando a determinare un nuovo modo di lavoro, molto cooperativo e integrato nel distribuito

Asincronicità vs. Sincronicità

Con le *normali operazioni sincrone del Web*, un **cambiamento anche minimo** richiede una *intera* ***operazione di richiesta*** *per la* ***sostituzione della intera pagina*** *e si deve* ***attenderla in modo sincrono***

Le operazioni Ajax permettono agli **utenti di non aspettare e di fare altro**, mentre il **browser si occupa delle operazioni e della sostituzione dei dati**





AJAX - Organizzazione interna