Luca Cabibbo

Architettura dei Sistemi Software

# Servizi web SOAP

**dispensa asw520**

ottobre 2023

*The nice thing about standards is that you have so many to choose from.*

*Andrew S. Tanenbaum*



## - Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 30, Servizi web SOAP e REST
- Papazoglou, M.P. **Web Services**: Principles and Technology. Pearson, 2008.
- Alonso, G., Casati, F., Kuno, H., and Machiraju, V. **Web Services**: Concepts, Architectures and Applications. Springer, 2004.

# - Obiettivi e argomenti

- Obiettivi
  - presentare i servizi web SOAP e alcuni dei relativi standard
- Argomenti
  - introduzione
  - servizi web SOAP
  - discussione



# * Introduzione

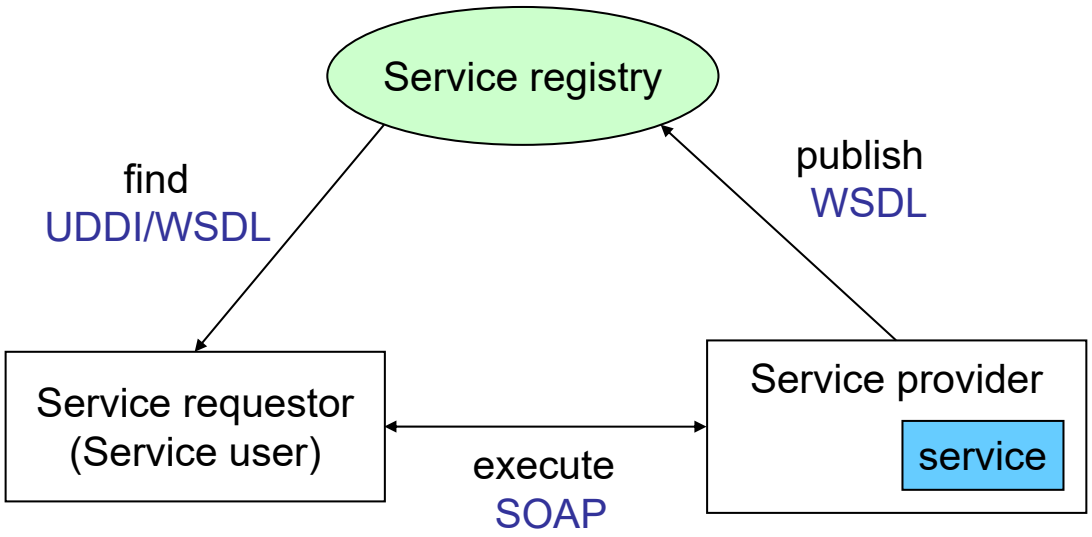
- Nell'architettura a servizi, un servizio è un componente software
  - ha lo scopo di implementare una funzionalità o un servizio di business di un'organizzazione
  - può essere scoperto e invocato dai suoi consumatori mediante un'interfaccia aperta e tramite tecnologie standard del web
  - presentiamo ora i servizi web SOAP

# * Servizi web SOAP

- I *servizi web SOAP* – *SOAP web service* o *web service WS-** o semplicemente *web service*
  - una delle prime e principali tecnologie a servizi
  - si basano su un ampio insieme di standard per l'interoperabilità – relativi sia a capacità funzionali di base che ad aspetti di qualità
    - gli standard principali sono SOAP, WSDL e UDDI – insieme ad XML
    - altri standard sono BPEL e le (numerose) estensioni WS-*
  - i principali responsabili degli standard per i servizi SOAP sono il consorzio *OASIS* (Organization for the Advancement of Structured Information Standards) e il *W3C* (World Wide Web Consortium)



# Interazione (di base) con i web service



# - Standard fondamentali per servizi web

- Standard alla base dei servizi web SOAP (WS)
  - *SOAP*
    - per lo scambio di dati e messaggi strutturati
  - *WSDL – Web Services Description Language*
    - per la descrizione dell'interfaccia dei WS
  - *UDDI – Universal Description, Discovery and Integration*
    - per la scoperta di WS
  - *XML*
    - una sintassi comune
  - un'infrastruttura minimale per i WS
    - è possibile comprendere questi standard ragionando sui problemi che intendono affrontare



# - Sintassi per WS e XML

- È necessaria una sintassi comune per gli standard per i WS
  - deve sostenere portabilità ed estendibilità
  - deve sostenere l'indipendenza dalle piattaforme e dai linguaggi di programmazione
- La sintassi scelta è *XML – eXtensible Markup Language*
  - offre generalità e flessibilità
  - tuttavia introduce un overhead nella codifica, trasmissione, decodifica dei dati

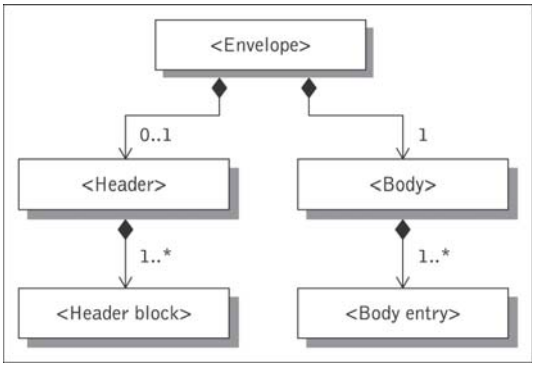
# - Interazione con i servizi web e SOAP

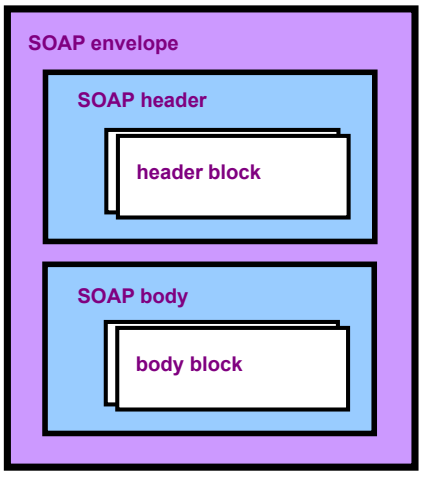
- È necessario un meccanismo per consentire l'interazione tra i WS e i suoi consumatori
  - deve supportare diverse forme di interazione
  - è necessario un formato comune per i messaggi scambiati
  - lo scambio di messaggi deve poter avvenire in modo indipendente dalla modalità di trasporto dei messaggi
- Le interazioni con e tra i WS sono basate su *SOAP*
  - la comunicazione è basata sullo scambio di *messaggi*
  - SOAP riguarda soprattutto il formato dei messaggi
  - è un protocollo stateless e one-way, che supporta diverse modalità di comunicazione ed è compatibile con diverse modalità di trasporto dei messaggi – ma il trasporto dei messaggi non è nella portata di SOAP



# Messaggi SOAP

- Un *messaggio SOAP* è costituito da una busta (*envelope*), che contiene un'intestazione e un corpo
  - il corpo (*body*) racchiude le informazioni che il messaggio vuole effettivamente trasmettere
  - l'intestazione (*header*) contiene metadati e altre informazioni "non funzionali"

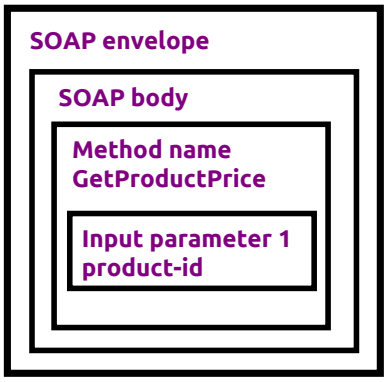


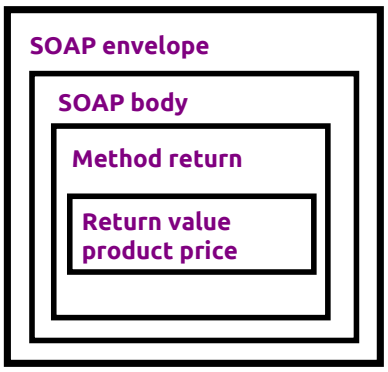
# Messaggi SOAP: esempio

- Una coppia di messaggi SOAP in un'interazione in stile RPC

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
             <tx:Transaction-id
                 xmlns:t="http://www.transaction.com/transactions"
                 env:mustUnderstand='1'>
               512
             </tx:Transaction-id>
    </env:Header>
    <env:Body>
             <m:GetProductPrice>
                <product-id> 450R6OP </product-id>
             </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

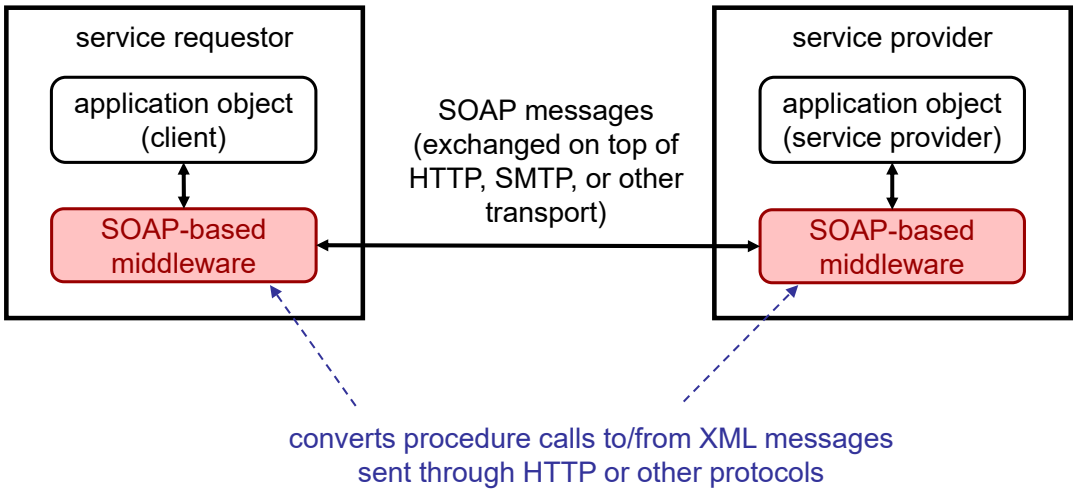



```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
             <--! - Optional context information -->
    </env:Header>
    <env:Body>
             <m:GetProductPriceResponse>
                <product-price> 134.32 </product-price>
             </m:GetProductPriceResponse>
    </env:Body>
</env:Envelope>
```






# Interazione con SOAP



# Modelli di comunicazione

- Il modello di comunicazione di SOAP prevede diverse varianti, basate su due proprietà principali
  - *stile di comunicazione* – *RPC* oppure *document*
    - lo stile *RPC* consente di definire messaggi in corrispondenza alla richiesta di esecuzione di un'operazione o alla relativa risposta
    - lo stile *document* (*message*) consente di scambiare generici documenti (messaggi)
  - *stile di codifica (encoding)* – *encoded* oppure *literal*
    - lo stile *encoded* (solitamente usato con lo stile RPC) è basato su una codifica standard dei dati trasmessi – consente, ad es., la serializzazione di grafi di oggetti
    - lo stile *literal* (solitamente usato con lo stile document) non prevede nessuna codifica dei dati trasmessi



# Alcune osservazioni su SOAP

- SOAP definisce un semplice meccanismo per codificare dati e consentirne lo scambio tra applicazioni distribuite
  - è un "wire protocol" – si occupa della forma con cui i servizi distribuiti possono scambiarsi dati (a livello applicativo)
  - non è un "transport protocol" (come TCP o UDP) – non si occupa dello scambio concreto dei dati tra servizi distribuiti
  - lo scambio di messaggi SOAP avviene sulla base di altri protocolli – ad es., HTTP
    - il binding di messaggi SOAP viene considerato nel contesto di WSDL
  - non definisce nessuna semantica – questo lo rende adatto a essere usato in una varietà di sistemi – da RPC al messaging
  - non si occupa di aspetti come il routing o lo scambio affidabile di messaggi – ma è adatto a contesti in cui è necessario scambiare informazioni in modo flessibile ed estensibile

# - Descrizione di servizi e WSDL

- È necessario anche un meccanismo per la descrizione di servizi
  - per descrivere l'interfaccia di un WS
  - per descrivere indirizzamento e la modalità di trasporto del WS
- La descrizione dei servizi web è basata su *WSDL* – *Web Services Description Language* – una specifica WSDL di un WS contiene
  - una *parte astratta* – descrizione dell'interfaccia
    - che cosa fa il WS (operazioni e porte) e come è possibile invocare il WS (messaggi e tipi di dati)
  - una *parte concreta* – descrizione dell'"implementazione"
    - lega la definizione astratta del servizio con degli indirizzi di rete concreti e dei protocolli di trasporto specifici – in termini di uno o più binding



```
<wsdl:definitions name="PurchaseOrderService"
      targetNamespace="http://supply.com/PurchaseService/wsdl"
      xmlns:tns="http://supply.com/ PurchaseService/wsdl"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
     <wsdl:types>
         <xsd:schema
             targetNamespace="http://supply.com/PurchaseService/wsdl"
            <xsd:complexType name="CustomerInfoType">
                  <xsd:sequence>
                        <xsd:element name="CusNamer" type="xsd:string"/>
                        <xsd:element name="CusAddress" type="xsd:string"/>
                    </xsd:sequence>
            </xsd:complexType>
             <xsd:complexType name="POType">
                     <xsd:sequence>
                           <xsd:element name="PONumber" type="integer"/>
                           <xsd:element name="PODate" type="string"/>
                     </xsd:sequence>
              </xsd:complexType>
              <xsd:complexType name="InvoiceType">
                     <xsd:all>
                           <xsd:element name="InvPrice" type="float"/>
                           <xsd:element name="InvDate" type="string"/>
                     </xsd:all>
              </xsd:complexType>
          </xsd:schema>
    </wsdl:types>
    <wsdl:message name="POMessage">
             <wsdl:part name="PurchaseOrder" type="tns:POType"/>
             < wsdl:part name="CustomerInfo" type="tns:CustomerInfoType"/>
    </wsdl:message>
    <wsdl:message name="InvMessage">
             <wsdl:part name="Invoice" type="tns:InvoiceType"/>
    </wsdl:message>
    <wsdl:portType name="PurchaseOrderPortType">
             <wsdl:operation name="SendPurchase">
                     <wsdl:input message="tns:POMessage"/>
                     <wsdl:output message="tns:InvMessage"/>
              </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

Abstract data type definitions

Data that is sent

Data that is returned

Port type with one operation

An operation with request (input) & response (output) message

```
<wsdl:definitions> . …
     <import namespace="http://supply.com/PurchaseService/wsdl"
      location="http://supply.com PurchaseService/wsdl/PurchaseOrder-interface.wsdl"/>
  <!--  location of WSDL PO interface from Listing-1-->
  <!-- wsdl:binding states a serialisation protocol for this service -->
  <!--  type attribute must match name of portType element in Listing-1-->
  <wsdl:binding name="PurchaseOrderSOAPBinding"
                          type="tns:PurchaseOrderPortType">

     <!-- leverage off soapbind:binding synchronous style -->
     <soapbind:binding style="rpc"
                                      transport="http://schemas.xmlsoap.org/soap/http/"/>

     <wsdl:operation name="SendPurchase">

       <!-- again bind to SOAP -->
       <soapbind:operation
          soapAction="http://supply.com/PurchaseService/wsdl SendPurchase" style="rpc"/>

       <!-- furthur specify that the messages in the wsdl:operation use SOAP -->
        <wsdl:input>
            <soapbind:body  use="literal"
              namespace="http://supply.com/PurchaseService/wsdl"/>
        </wsdl:input>
        <wsdl:output>
             <soapbind:body  use="literal"
                namespace="http://supply.com/PurchaseService/wsdl"/>
         </wsdl:output>

    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="PurchaseOrderService">
       <wsdl:port  name="PurchaseOrderPort" binding="tns:PurchaseOrderSOAPBinding">
            <!-- give the binding a network endpoint address or URI of service  -->
            <soapbind:address location="http://supply.com:8080/PurchaseOrderService"/>
      </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Bind an abstract operation to this implementation and

map the abstract input and output messages to these concrete messages

Service name

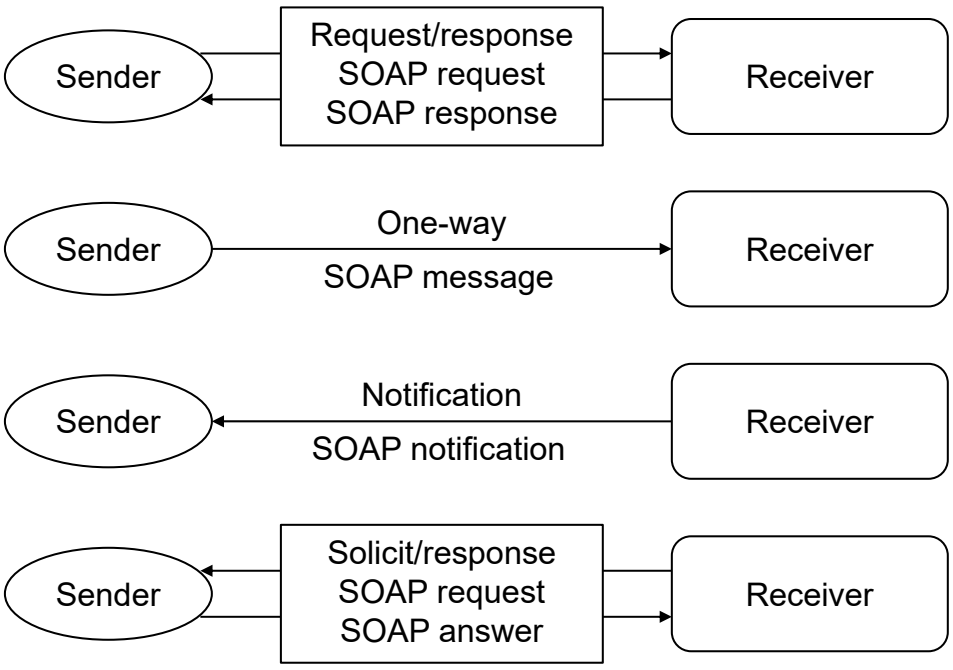
Network address of service





# Message Exchange Pattern

- In WSDL, ogni operazione può avere un solo messaggio di input e/o un solo messaggio di output
  - ci sono quattro possibili pattern per lo scambio di messaggi tra servizi (*Message Exchange Pattern* o *MEP*)



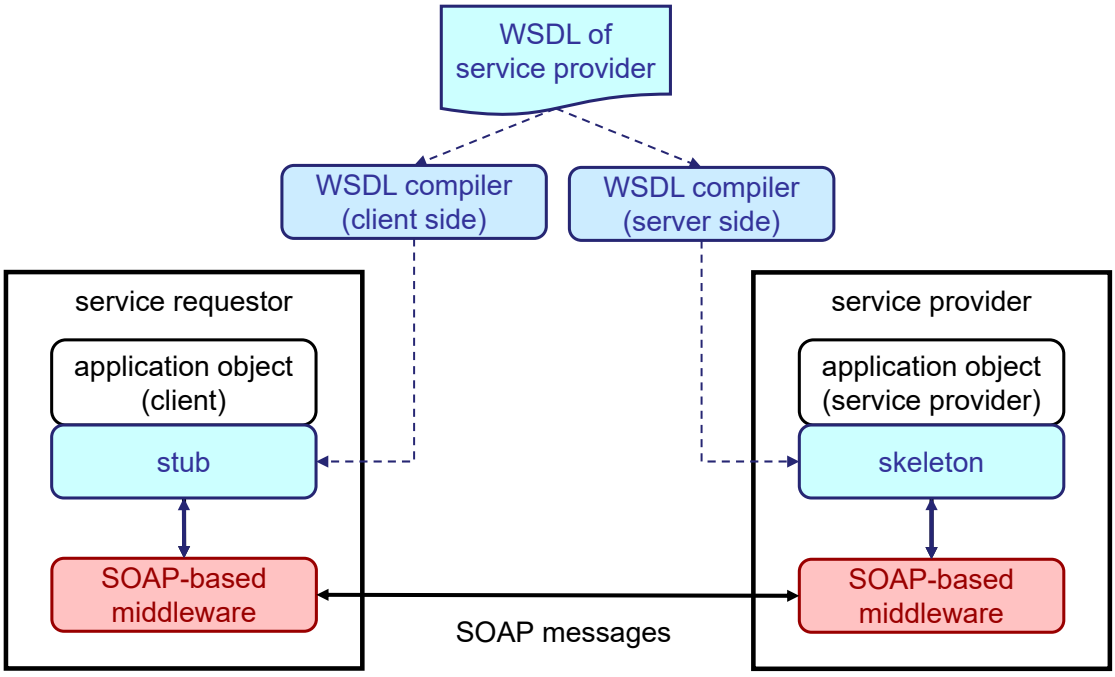
# Message Exchange Pattern

- *Request/response*
  - un'operazione in cui il servizio riceve un messaggio, e poi invia una risposta al suo client – è una forma di invocazione remota
- *One way*
  - il client invia un messaggio al servizio, ma il servizio non invia risposta al suo client – è una forma di comunicazione asincrona
- *Notification*
  - il servizio invia un messaggio a un client, senza attendere risposta – è una forma di notifica di eventi ai client
- *Solicit/response*
  - il servizio invia un messaggio a un client, e poi ne riceve una risposta – è complementare a request/response



# Uso di WSDL e SOAP

- Gli strumenti di sviluppo per WS possono utilizzare WSDL e SOAP per la costruzione automatica di oggetti proxy – sia lato del servizio che lato client



# Uso di WSDL e SOAP

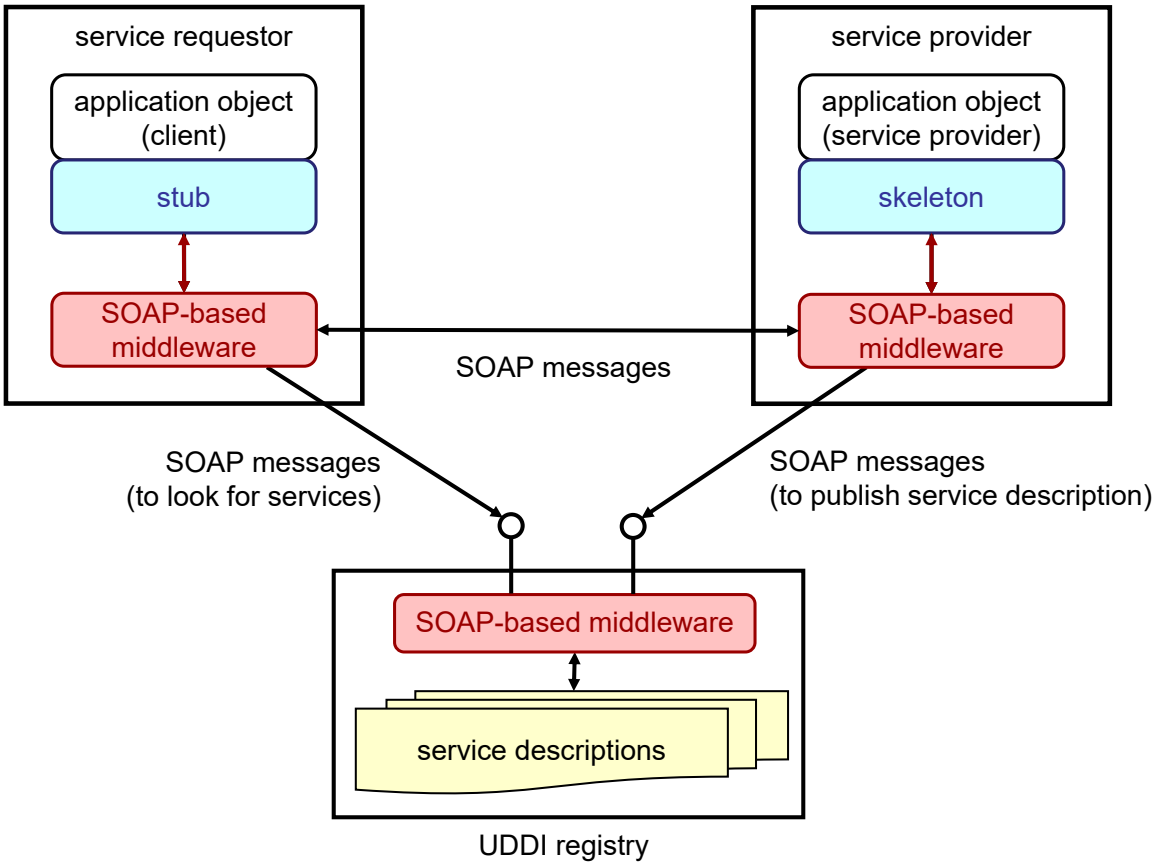
- Alcune funzionalità degli strumenti per WSDL e SOAP
  - generazione top-down del servizio – contract-first
    - da una specifica WSDL, viene generato il codice (skeleton) dell'implementazione del servizio (da completare)
  - generazione bottom-up del servizio – contract-last
    - dall'implementazione del servizio (ad es., un enterprise bean stateless esposto come servizio web), viene definito il servizio e generata la specifica WSDL
  - generazione del proxy lato client
    - da una specifica WSDL, viene generato il codice (stub) del proxy del servizio



# - Scoperta di servizi e UDDI

- È anche utile un meccanismo per la scoperta di WS – ovvero, un servizio di directory (un registry) dei servizi
- Un servizio di directory per i servizi web può essere basato sullo standard *UDDI* – *Universal Description, Discovery, and Integration*
  - due elementi
    - il registry – con diverse modalità di classificazione e ricerca dei WS
    - un'API per l'accesso al registry

# Directory per Web Services






# Uso del servizio di directory

- Alcuni possibili utilizzi di un registry di servizi
  - uso “statico”
    - utilizzato in sede di sviluppo di un’applicazione a servizi
    - supporta il riuso dei WS in più applicazioni – e la condivisione di WS tra organizzazioni
  - uso “dinamico”
    - utilizzato a runtime
    - ad es., per supportare il brokering di servizi e consentire la selezione dinamica tra servizi

# Servizi “statici”, “dinamici” e “semantici”

- Per utilizzare un servizio, un consumatore (client) deve determinare l’interfaccia del servizio, localizzarlo e poi invocarlo
  - nel binding (collegamento) statico, interfaccia e locazione del servizio sono determinate durante l’implementazione (o il deployment) del consumatore del servizio
  - nel binding dinamico, la locazione del servizio è determinata a runtime, con l’ausilio di un service registry
    - l’accoppiamento tra produttore e consumatore è ridotto
      - è possibile ad esempio che la locazione del servizio cambi – senza impatto sul consumo del servizio
    - c’è un overhead sulle prestazioni
  - talvolta, anche l’interfaccia del servizio è determinata a runtime
    - è però necessario che il consumatore e il fornitore del servizio abbiano un accordo predefinito sulla sintassi e la semantica delle interfacce



# - Ulteriori standard ed estensioni per WS

- Esistono anche altri standard (oltre 100!) per i WS – indicati complessivamente come *WS-**
  - alcuni sono relativi ad aspetti specifici dell’interazione tra servizi web – ad esempio
    - WS-Addressing si occupa di indirizzamento dei servizi, in modo neutrale rispetto ai meccanismi di trasporto
    - i WS sono solitamente stateless – ma è possibile definire servizi stateful sulla base di WS-Resource
  - molte estensioni sono relative alla gestione delle qualità dei servizi – ad esempio
    - sicurezza – WS-Security
    - affidabilità della comunicazione – WS-ReliableMessaging
    - transazioni – WS-Coordination, WS-Transaction
    - requisiti, capacità e preferenze – WS-Policy
  - alcune estensioni sono complementari, altre in competizione

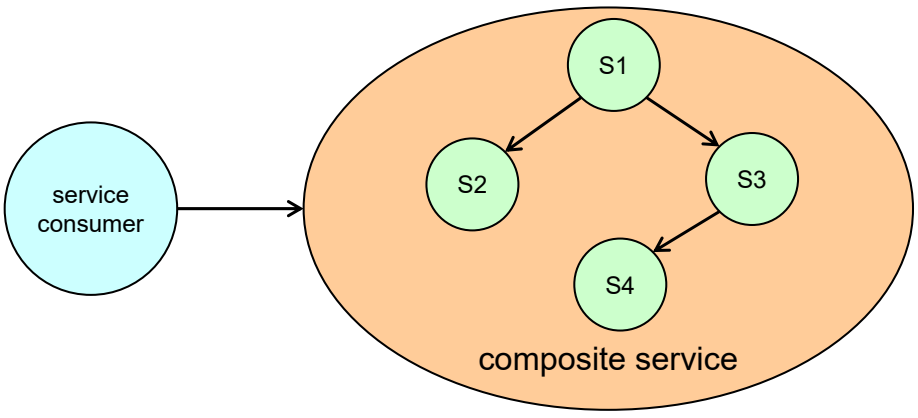
# Estensioni e profili

- Le estensioni WS-* non sono però accettate o sostenute allo stesso modo dai diversi produttori di software
  - per favorire l'interoperabilità, le estensioni sono state organizzate in *profili*
  - ad esempio
    - il profilo WS-I Basic ("Interoperability") fa riferimento soprattutto a SOAP, WSDL, UDDI, XML, insieme a XML-Schema, HTTP, HTTPS e WS-Addressing
    - il profilo Reliable-Secure-Profile estende il profilo Basic con l'adozione di WS-ReliableMessaging, WS-SecureConversation, WS-MakeConnection e WS-SecurityPolicy
  - solo alcuni profili sono accettati in modo diffuso



# - Composizione di servizi web

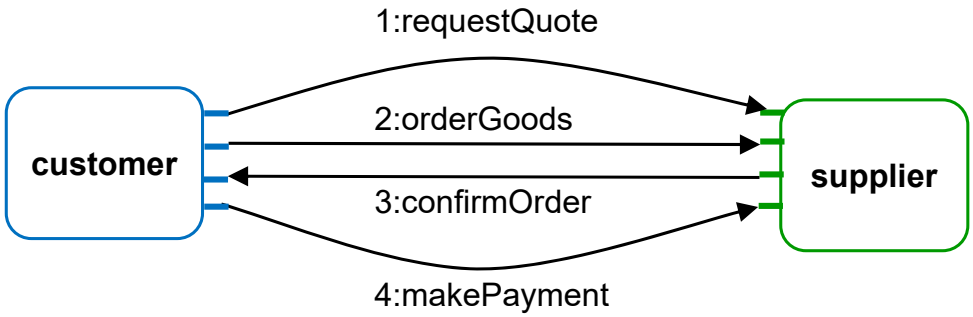
- Una capacità fondamentale delle tecnologie a servizi è poter definire nuovi servizi come *composizione* di altri servizi




  - nel caso dei WS, la composizione di servizi
    - deve specificare come un insieme di servizi "componenti" possano interagire in modo coordinato ed effettivo per dar luogo a un nuovo servizio "composto"
    - deve fornire flessibilità nella composizione – anche quando i servizi sono forniti da organizzazioni o sistemi diversi

# Composizione di servizi

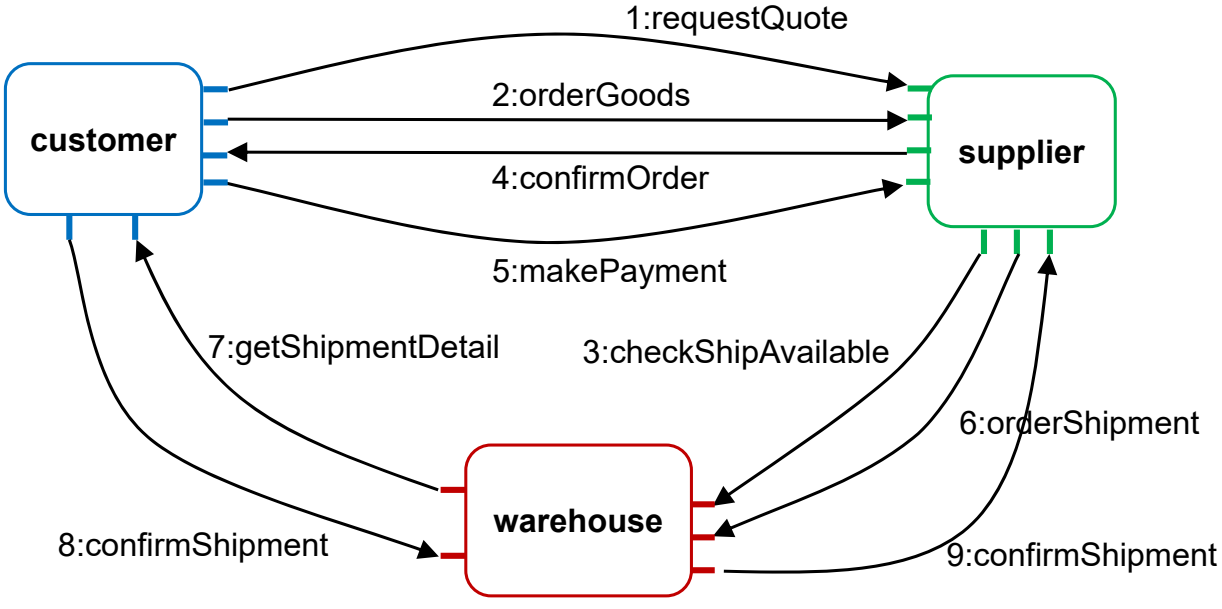
- La composizione di WS si basa sulla collaborazione e lo scambio di messaggi tra i servizi componenti
  - ad es., la conversazione tra due servizi per gestire l'approvvigionamento di prodotti






# Composizione di servizi

- La composizione di WS si basa sulla collaborazione e lo scambio di messaggi tra i servizi componenti
  - una conversazione più complessa: il fornitore delega la consegna al magazzino – il magazzino interagisce direttamente con il cliente per accordarsi sulla spedizione



# Requisiti per la composizione di servizi

- Alcuni requisiti specifici per la composizione di servizi
  - flessibilità – deve supportare la composizione di servizi forniti da organizzazioni o sistemi software diversi
  - agilità – deve essere possibile definire rapidamente un nuovo servizio come composizione di altri servizi esistenti – oppure anche ridefinire un servizio come una nuova composizione di servizi
  - deve poter essere specificata da esperti del dominio di business
  - complessivamente, la composizione di servizi deve poter essere effettuata come un'attività di assemblaggio di servizi – e non come un'attività di sviluppo



# Composizione di servizi, in pratica

- Come realizzare, in pratica, la composizione di servizi?
  - una possibilità è utilizzare dei linguaggi di programmazione tradizionali
    - è una soluzione poco flessibile – che non risponde ai requisiti per la composizione di servizi
  - una soluzione più comune (con i WS) si basa su dei linguaggi specializzati per la composizione di WS
    - *BPEL* è un linguaggio di programmazione specializzato per la composizione di servizi web SOAP
    - *BPMN* è un linguaggio visuale per la modellazione di servizi e processi di business, che può essere utilizzato direttamente dagli esperti di dominio del business
    - va anche usato un compilatore da BPMN a BPEL

# BPEL

- *Business Process Execution Language* (*BPEL* o *WS-BPEL*)
  - un linguaggio di programmazione/scripting per servizi web
    - definisce primitive per la fruizione di servizi – come invoke, receive, reply, throw e wait
    - contiene costrutti di controllo tradizionali – assegnazione, sequenza, istruzioni condizionali e ripetitive
    - contiene anche un costrutto per l'esecuzione concorrente di attività – flow
    - la sintassi è basata su XML
  - il codice BPEL può essere eseguito da un motore BPEL – ad es., in un application server – che coordina l'invocazione dei servizi, usando il codice BPEL come uno script
  - BPEL supporta soprattutto l'orchestrazione di servizi



# BPMN

- *Business Process Model and Notation* (*BPMN*)
  - un linguaggio visuale standard per la modellazione di processi di business
  - quattro categorie principali di costrutti
    - *oggetti flusso* – gli elementi principali di un diagramma, per rappresentare eventi, attività e gateway (condizioni) – le attività possono essere completamente automatizzate oppure possono richiedere un intervento di un utente umano
    - *oggetti connessione* – per collegare oggetti flusso e per descrivere le loro dipendenze
    - *corsie* – per descrivere chi è responsabile di svolgere le attività
    - *elaborati* – per descrivere i flussi di dati tra le attività
  - un diagramma BPMN può essere "compilato" in BPEL

# BPMN




Buyer
Place Order
Receive Invoice
Settle Invoice
Receive Products
Reseller
Check Order
Send Invoice
Receive Payment
Archive Order
Ship Products





# BPMN




Buyer
Place Order
Receive Invoice
Settle Invoice
Receive Products
Reseller
Order received
Check Order
Send Invoice
Receive Payment
Ship Products

# * Discussione

- I servizi web SOAP sono una delle principali tecnologie a servizi
  - si basano su un piccolo insieme di standard fondamentali (accettati in modo diffuso) – insieme a numerose estensioni
  - l'uso di SOAP garantisce indipendenza dai protocolli – sia di trasporto, che quelli relativi alle qualità, dichiarate negli header SOAP
  - l'uso di WSDL sostiene ulteriormente l'indipendenza dalla piattaforma per la definizione e la fruizione dei servizi web
  - BPEL supporta la composizione di servizi
    - insieme a BPMN, la composizione di servizi può essere effettuata in modo flessibile e agile
  - altri standard sostengono diversi attributi di qualità
  - la disponibilità di strumenti e di piattaforme per lo sviluppo e l'esecuzione di WS nasconde agli sviluppatori molta della complessità di questi standard



# Discussione

- I servizi web SOAP sono una delle principali tecnologie a servizi
  - il vantaggio principale dei servizi SOAP è la loro completezza
    - soprattutto nel supporto alla composizione e alle qualità
  - lo svantaggio principale dei servizi SOAP è la loro pesantezza
    - per l'uso estensivo di XML, ma anche per la complessità di molte estensioni
    - la semplicità con cui è possibile esporre componenti software esistenti come servizi web può dar luogo a un uso poco efficace dei WS
    - inoltre, è anche possibile che si verifichino in pratica dei problemi di interoperabilità tra un servizio e i suoi client
  - come vedremo meglio (nella seconda parte del capitolo), il vantaggio principale dei servizi REST è la loro semplicità – il loro svantaggio principale è la mancanza di un supporto diretto a diverse qualità dei servizi