INTRODUCING WEB SERVICES

SAPIENZA
UNIVERSITÀ DI ROMA

Massimo Mecella
MSE-CS

Service-Oriented Paradigm

- The service-oriented paradigm to programming utilizes *services* as the constructs to support the development of rapid, low-cost and easy composition of distributed applications.
- A Web service is a ***programmatically available application logic exposed over the Internet.***
  - Any piece of code and any application component deployed on a system can be transformed into a network-available service.
- Services reflect a new "service-oriented" approach to programming, based on the idea of composing applications by discovering and invoking network-available services rather than building new applications or by invoking available applications to accomplish some task.

What are Web Services?

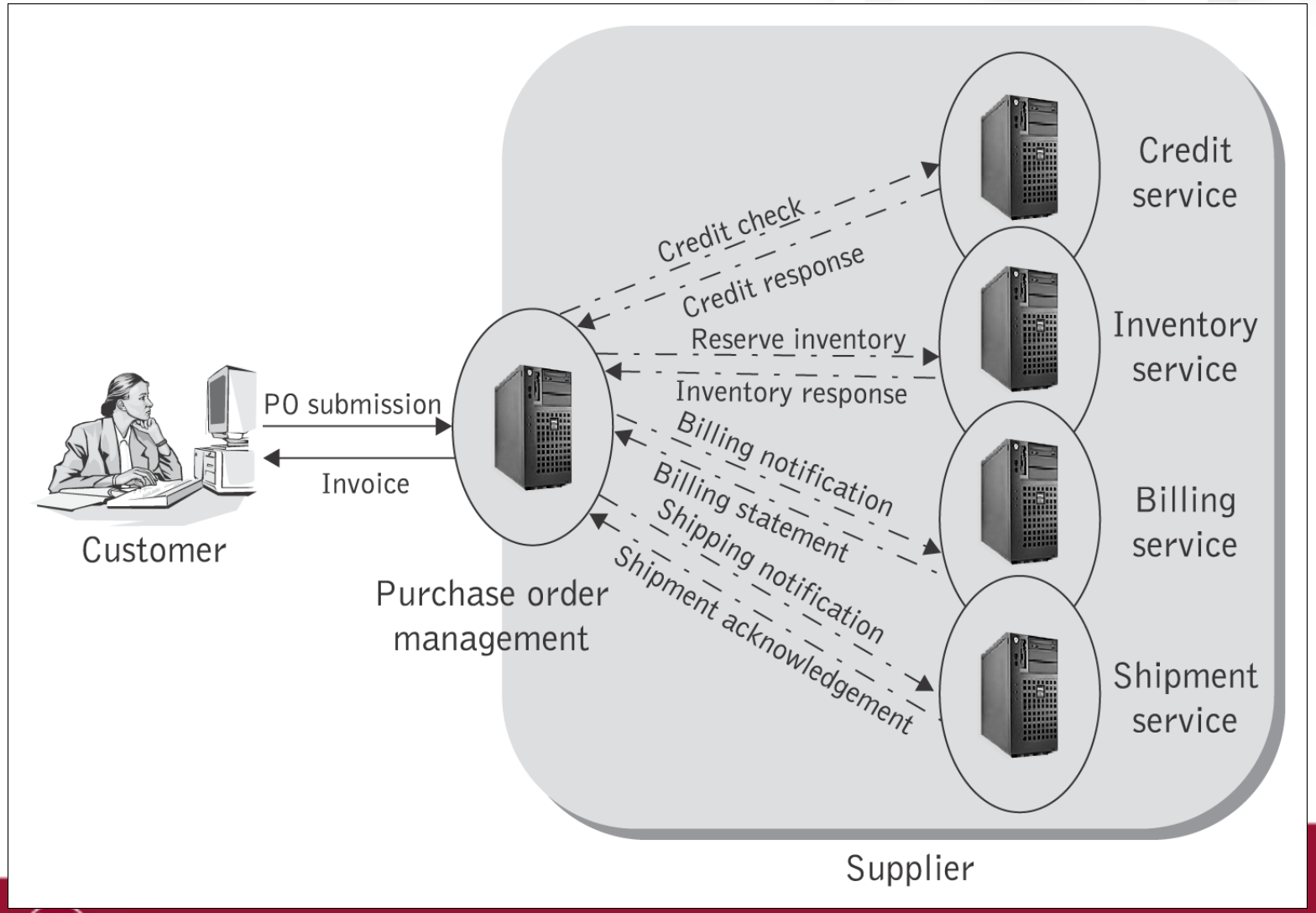
Web services perform encapsulated business functions such as:

1. a self-contained business task – a funds withdrawal or funds deposit service;
2. a full-fledged business process – the automated purchasing of office supplies;
3. an application – a life insurance application or demand forecasts and stock replenishment; or
4. a service-enabled resource – access to a particular back-end database containing patient medical records.

What are Web Services?

- Can be mixed and matched to create a complete process:
  - Enable integration with decreased human interaction, e.g., create complete enterprise processes, such as supply chain management, procurement, logistics, etc.
  - Both new and extensions to existing applications
- Available to a variety of clients (platform independent)
- Pricing (rating) model determines subscriber rates based on subscription and usage events, e.g., calculate charges for services based on the quality and precision of the service.
- Billing model:
  - "pay per use", "lease it", "pay for it" basis.

A purchase order application involving interacting web services



Application Service Providers

- The concept of software-as-a-service is evolutionary and appeared first with the ASP (application service provider) software model:

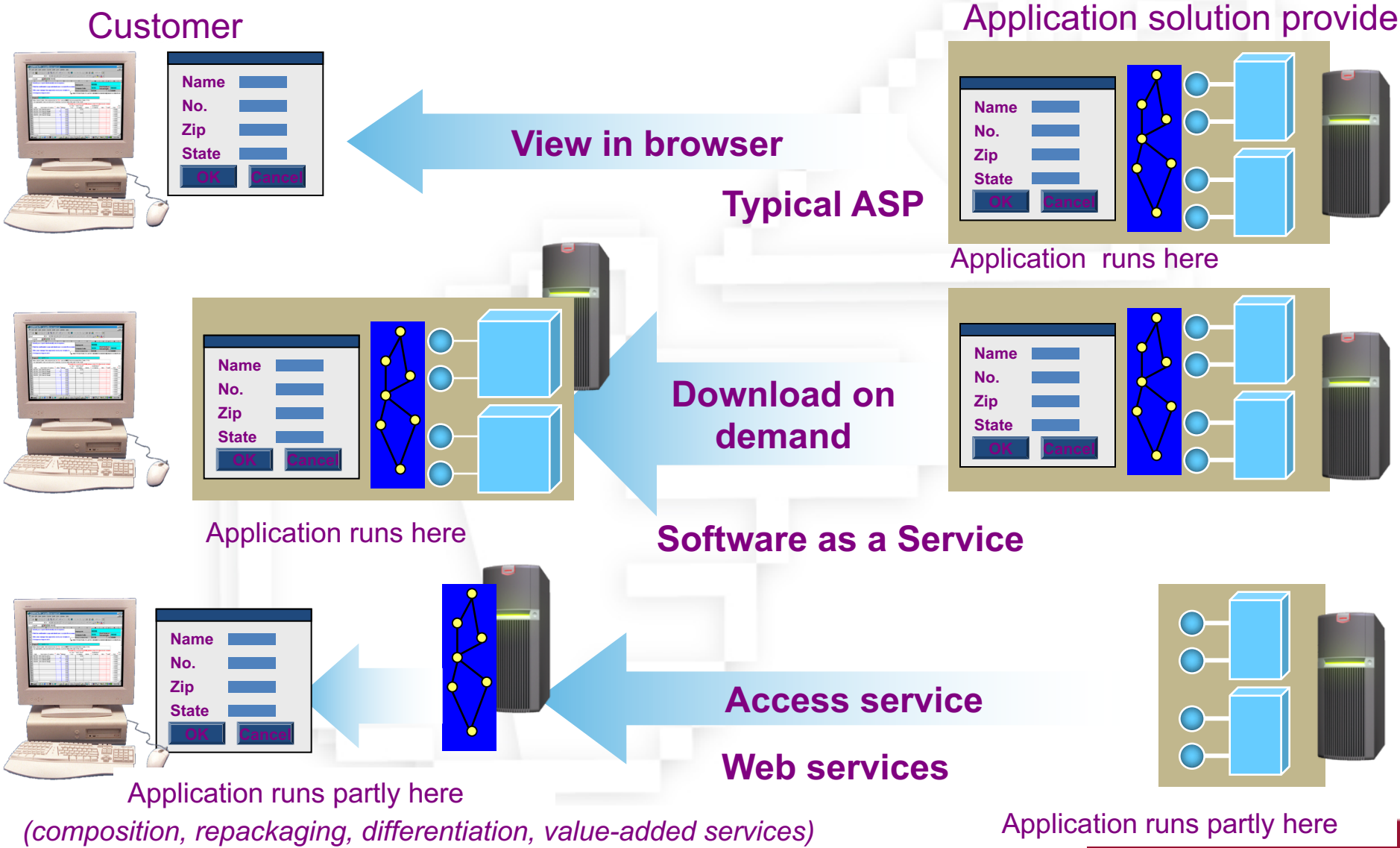
  An ASP "rents" applications to subscribers.

  – The whole application is developed in terms of the user interface, workflow, business, and data components that are all bound together to provide a working solution.
  – An ASP hosts the entire application and the customer has little opportunity to customize it beyond the appearance of the user interface, e.g., adding company logos.
  – An alternative of this is where the ASP is providing a software module that is downloaded to the customer's site on demand – this is for situations where the software does not work in a client/server fashion, or can be operated remotely via a browser.

ASP vs. Web Services

- Although the ASP model introduced the concept of software- as-a-service first, it suffered from several inherent limitations such as
  - inability to develop highly interactive applications;
  - inability to provide complete customizable applications and;
  - inability to integrate applications.
- Today we are in the midst of another significant development in the evolution of software-as-a-service. The new architecture allows for:
  - loosely coupled asynchronous interactions
  - on the basis of eXtensible Markup Language (XML) standards
  - with the intention of making access to, and communications between, applications over the Internet easier, by means of appropriate standards.

Are ASP, Software-as-a-Service, and Web Service the same?



Where are Services used?

- Within an enterprise (enterprise application integration)

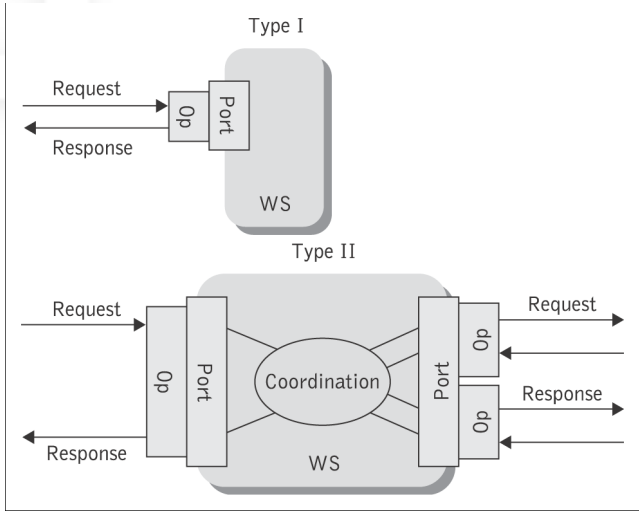
  - ➡ Accelerate and reduce the cost of integration
  - ➡ Save on infrastructure deployment and management costs
  - ➡ Reduce skill requirements
  - ➡ Improve reuse

- Between enterprises (e-Business integration)

  - ➡ Providing service to a company's customers
    - e.g., an Insurance company wishes to link its systems to the systems of a new institutional customer
  - ➡ Accessing services from a company's partners and suppliers
    - e.g., dynamically link to new partners and suppliers to offer their services to complement the value the company provides
  - Standards and common infrastructure reduce the barriers
  - Simplicity accelerates deployment
  - Dynamics opens new business opportunities

Types of Web Services

- Informational services are services of relatively simple nature. They either provide access to content interacting with an end-user by means of simple request/response sequences, or expose back-end business applications to other applications. Examples include:
  - content services such as weather report info., simple financial info., stock quote info., news items
  - simple trading services that can provide a seamless aggregation of information across disparate systems and data sources.
- Complex services that involve the assembly and invocation of many pre-existing services possibly found in diverse enterprises to complete a multi-step business interaction:
  - a supply-chain application involving order taking, stocking orders, sourcing, inventory control, financials, and logistics.



Service Properties and State

- Functional and non-functional properties:
  - The functional service description details the operational characteristics that define the overall behavior of the service.
  - The non-functional description targets service quality attributes, e.g., service metering and cost, performance metrics (response time or accuracy), security, authorization, authentication, scalability, availability, etc.
- Stateless or stateful services:
  - Services that can be invoked repeatedly without having to maintain context or state they are called stateless.
    - Simple informational services are stateless.
  - Services that require their context to be preserved from one invocation to the next are called stateful.
    - Complex services (business processes) typically involve stateful interactions.

Loose Coupling and Granularity

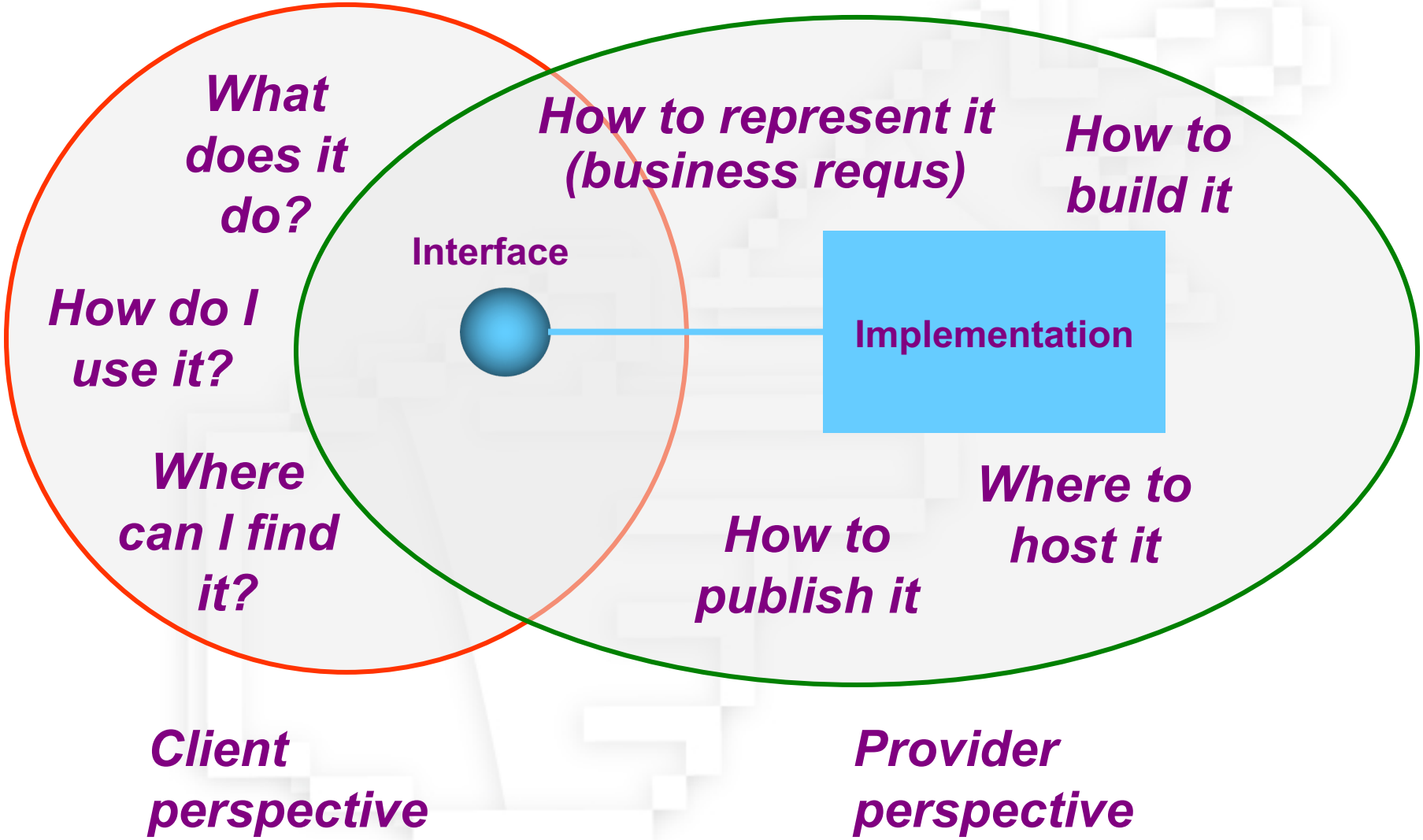
- Loose coupling:
  - Coupling indicates the degree of dependency any two systems have on each other.
  - Web services are loosely coupled: they connect and interact more freely (across the Internet). They need not know how their partner applications behave or are implemented.
- Service granularity:
  - Simple services are discrete in nature, exhibit normally a request/reply mode of operation and are of fine granularity, i.e., they are atomic in nature.
  - Complex services are coarse-grained, e.g., a SubmitPurchaseOrder process. These involve interactions with other services and possibly end-users in a single or multiple sessions.
  - Coarse-grained communication implies larger and richer data structures (viz. those supported by XML).

Synchronicity and Well-definedness

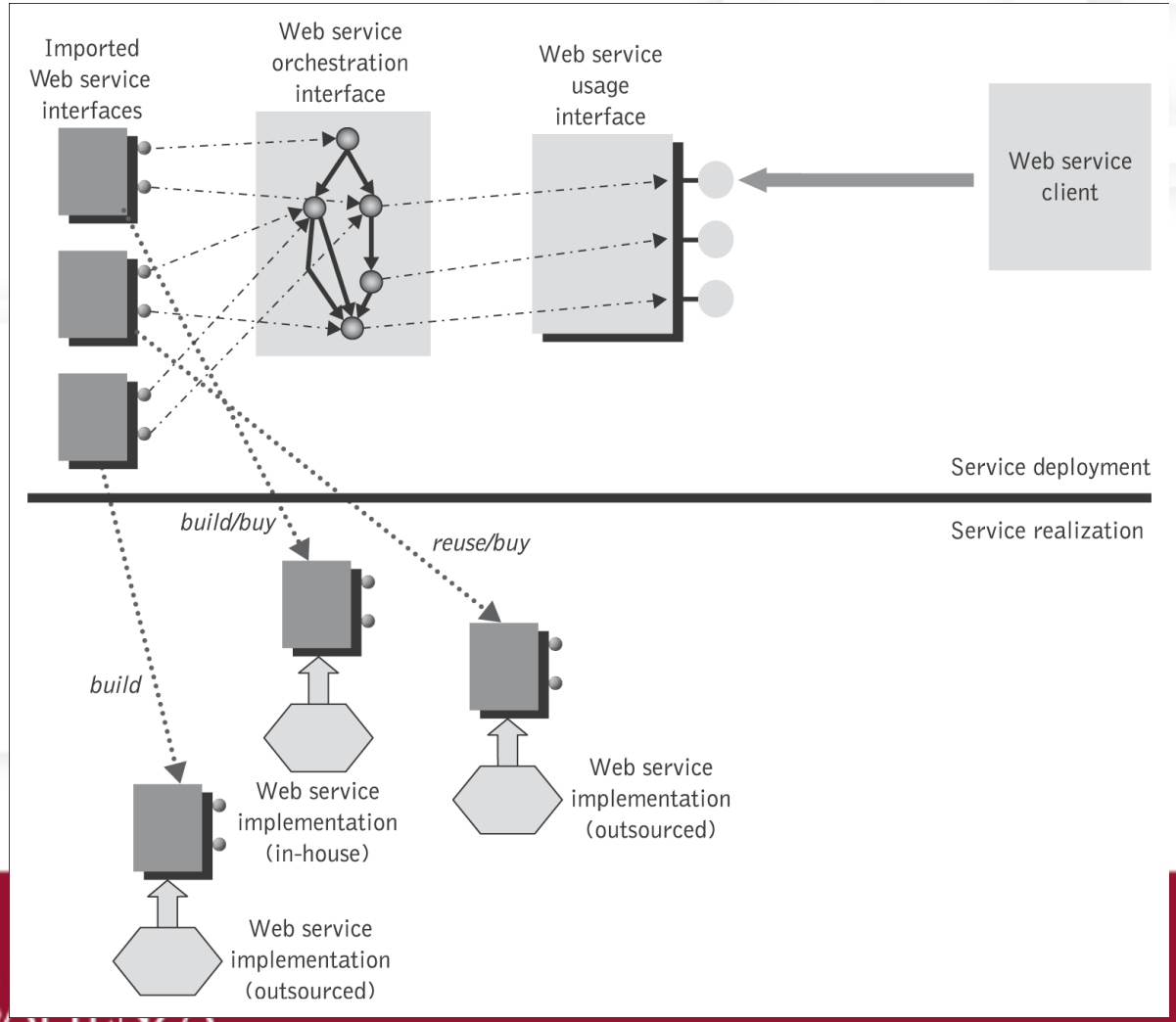
- Sychronicity: There are two programming styles for services:
  - Synchronous or remote procedure call (RPC)-style: Clients of synchronous services express their request as a method call with a set of arguments, which returns a response containing a return value.
    - Simple informational services, e.g., returning the current price for a given stock; providing the current weather conditions in a particular location; or checking the credit rating of a potential trading partner.
  - Asynchronous or message (document)-style: When a client invokes a message-style service, it typically sends it an entire document, e.g., a purchase order, rather than a discrete set of parameters.
    - Business processes, e.g., processing a purchase order; responding to a request for quote order from a customer; or responding to an order placement by a particular customer.
- Well-definedness:
  - The service interaction must be well- defined. The web services description language (WSDL) allows applications to describe the rules for interfacing and interacting to other applications.

Service Interface and Implementation

- The service interface defines service functionality visible to the external world and provides the means to access this functionality.
  - The service describes its own interface characteristics, i.e., the operations available, the parameters, data-typing and the access protocols, in a way that other software modules can determine what it does, how to invoke its functionality, and what result to expect in return.
- The service implementation realizes a specific service interface whose implementation details are hidden from its users.
  - Different service providers using any programming language of their choice may implement the same interface.


What does it do?
How do I use it?
Where can I find it?
How to represent it (business requs)
How to build it
Interface
Implementation
How to publish it
Where to host it
Client perspective
Provider perspective

Service deployment vs. service realization

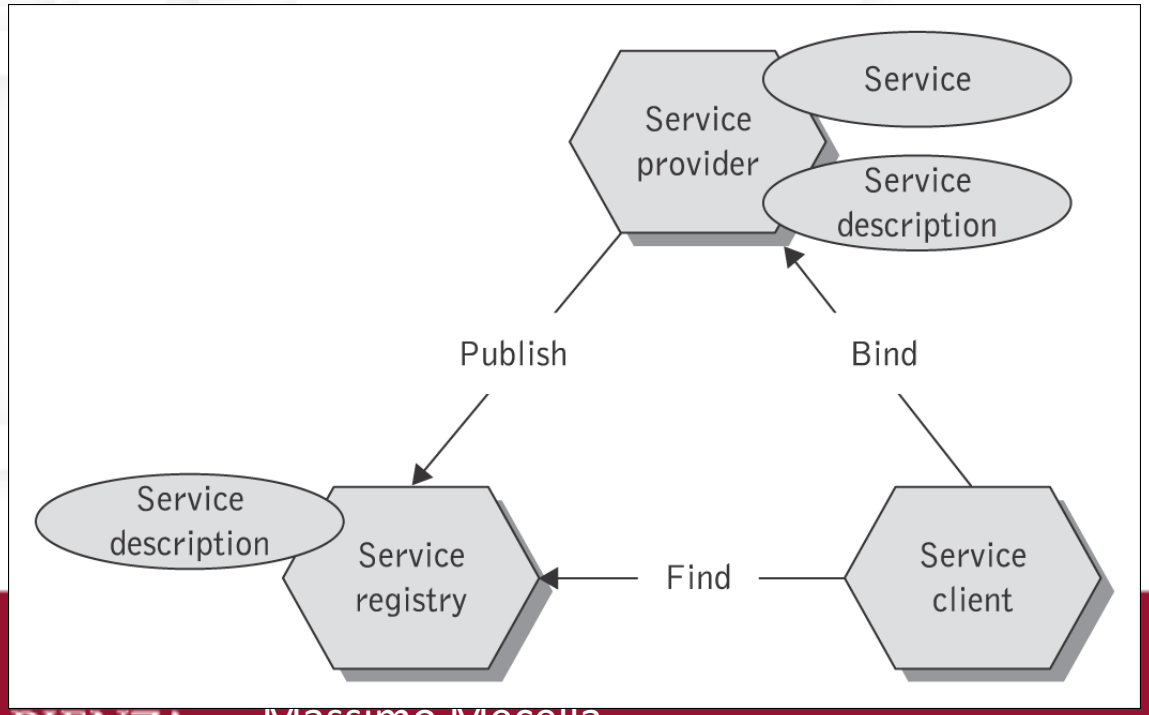


Roles

- The service model allows for a clear distinction to be made between
  - ***service providers*** (organizations that provide the service implementations, supply their service descriptions, and provide related technical and business support);
  - ***service clients*** (end-user organizations that use some service);
  - ***service registry*** (a searchable directory where service descriptions can be published and searched).
    - Service requestors find service descriptions in the registry and obtain binding information for services.
    - This information is sufficient for the service requestor to contact, or bind to, the service provider and thus make use of the services it provides.

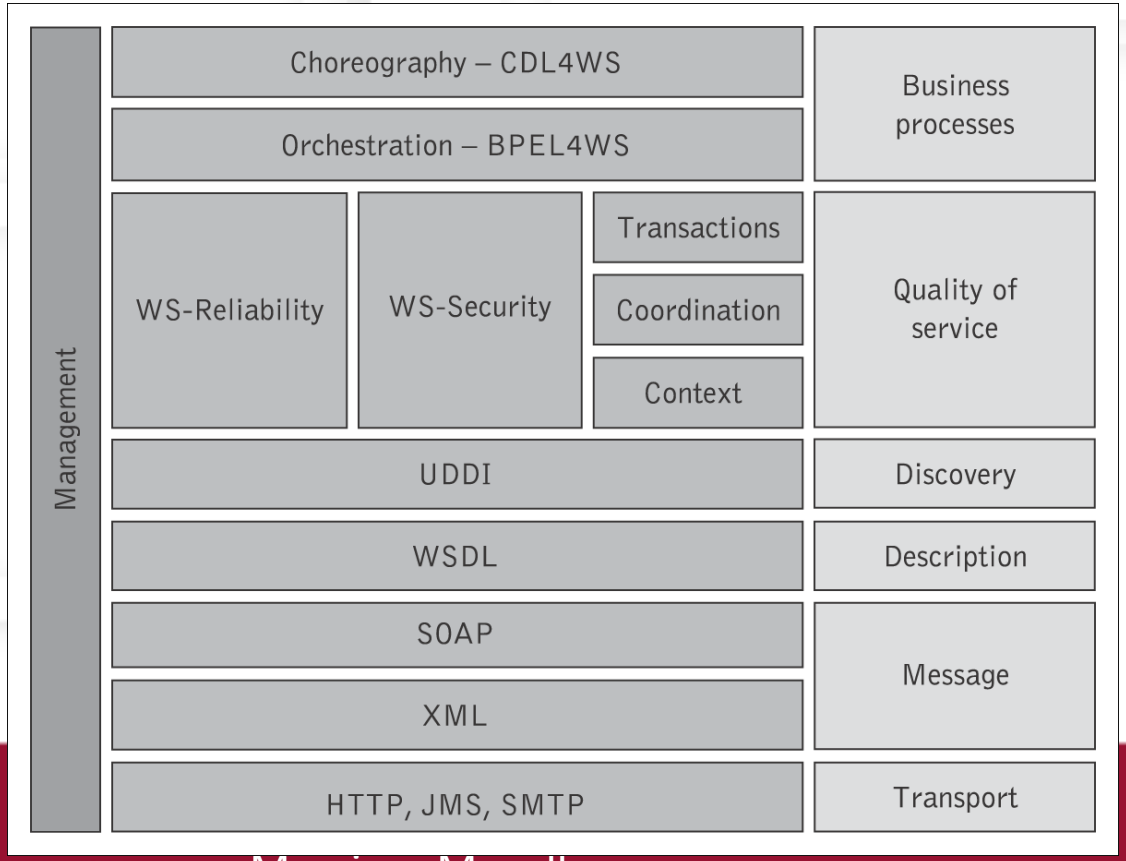
Service-Oriented Architecture

- SOA is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces.



Web Services Technology Stack

- Web services are implemented by a collection of several related technologies and standards.



Quality of Service (QoS)

- QoS refers to the ability of a Web service to respond to expected invocations and perform them at the level commensurate to the mutual expectations of both its provider and its customers.
- The key QoS in a Web services environment are:
  - availability
  - accessibility
  - conformance to standards
  - integrity
  - performance
  - reliability
  - scalability
  - security
  - Transactionality

Service Level Agreements (SLAs)

- An SLA is a formal agreement (contract) between a provider and client, formalizing the details of use of a Web service (contents, price, delivery process, acceptance and quality criteria, penalties, etc in measurable terms) in a way that meets the mutual understandings and expectations of both providers and clients.
- An SLA may contain the following parts:
  - purpose
  - parties
  - validity period
  - scope
  - restrictions
  - service-level objectives
  - penalties
  - exclusion terms
  - administration authority.

Impact of Web Services

- The most appealing characteristic of Web services is that they are evolving to embrace the convergence of e-Business, EAI, traditional middleware, and the Web. Web services offer:
  - a standard way to expose legacy application functionality as a set of reusable services;
  - a standard, easy, and flexible way to help overcome application integration issues;
  - a standard way to develop and/or assemble Internet-native applications for both the internal and the extended enterprise;
  - a common facade for cross-enterprise specific systems, making it easier to create the service-level agreements needed for business-to-business integration.

COMMUNICATION & SOAP

SAPIENZA
UNIVERSITÀ DI ROMA

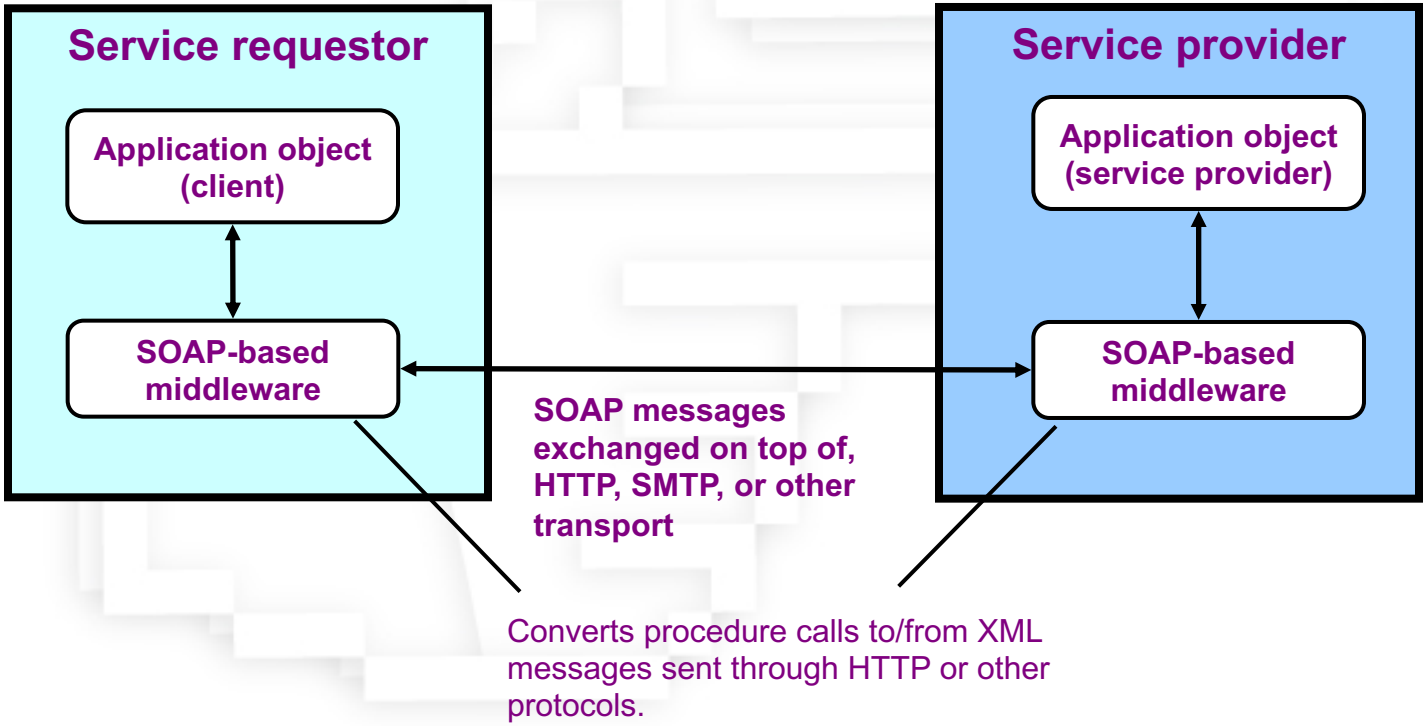
Massimo Mecella
MSE-CS

Inter-Application Communication

- Conventional distributed applications use distributed communication technologies, e.g., CORBA, DCOM/.NET, Java/RMI, based on object RPC (ORPC) protocols that attempted to marry object orientation and network protocols.
  - ORPC request is an identifier or symbolic name that the server could use to locate the target object inside the server process.
- Weaknesses
  - Both ends of the communication link would need to be implemented under the same distributed object model (Java/RMI or CORBA/IIOP)
  - Difficulty of getting these protocols to work over firewalls or proxy servers, e.g, most firewalls are configured to allow hypertext transfer protocol (HTTP) to pass across, but not IIOP.
- To address the problem of overcoming proprietary systems running on heterogeneous infrastructures, Web services rely on SOAP, an XML-based communication protocol for exchanging messages between computers regardless of their operating systems, programming environment or object model framework.

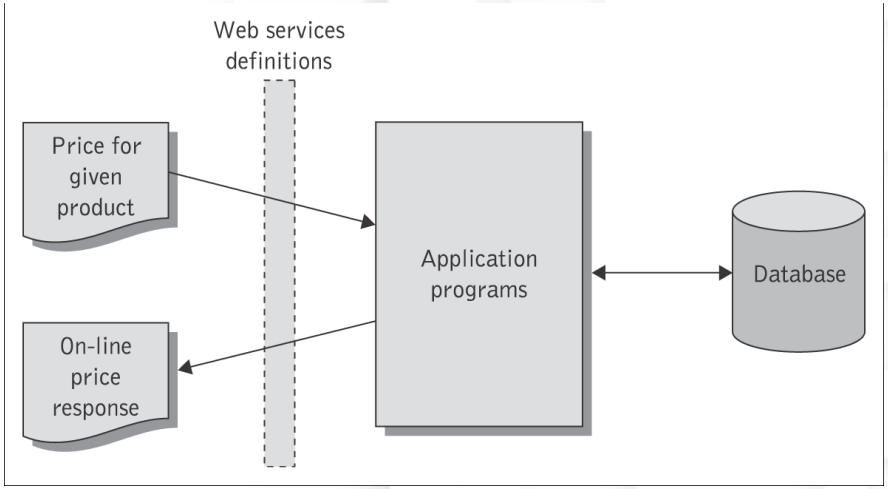
What is SOAP?

- SOAP is the standard messaging protocol used by Web services. SOAP's primary application is inter application communication. SOAP codifies the use of XML as an encoding scheme for request and response parameters using HTTP as a means for transport.

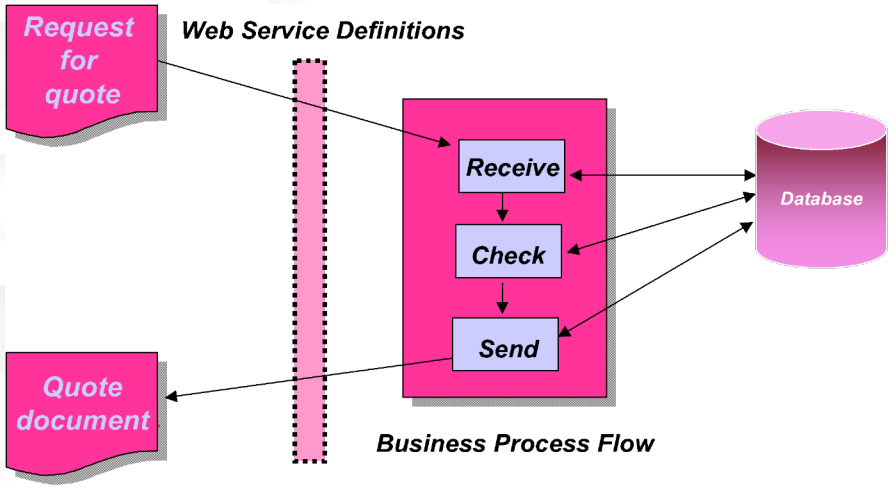


The SOAP Communication Model

- SOAP supports two possible communication styles:
  - remote procedure call (RPC) and
  - document (or message).



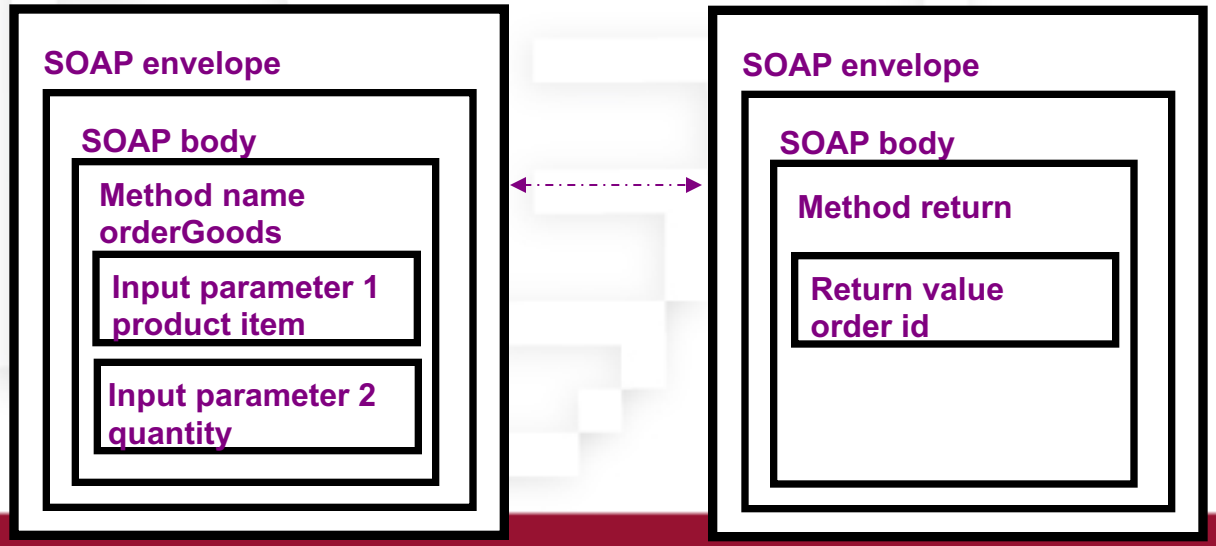
RPC-style interaction



Document-style interaction

RPC-style SOAP Services

- A remote procedure call (RPC)-style Web service appears as a remote object to a client application. The interaction between a client and an RPC-style Web service centers around a service-specific interface. Clients express their request as a method call with a set of arguments, which returns a response containing a return value.



RPC-style web services

```
<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>
```

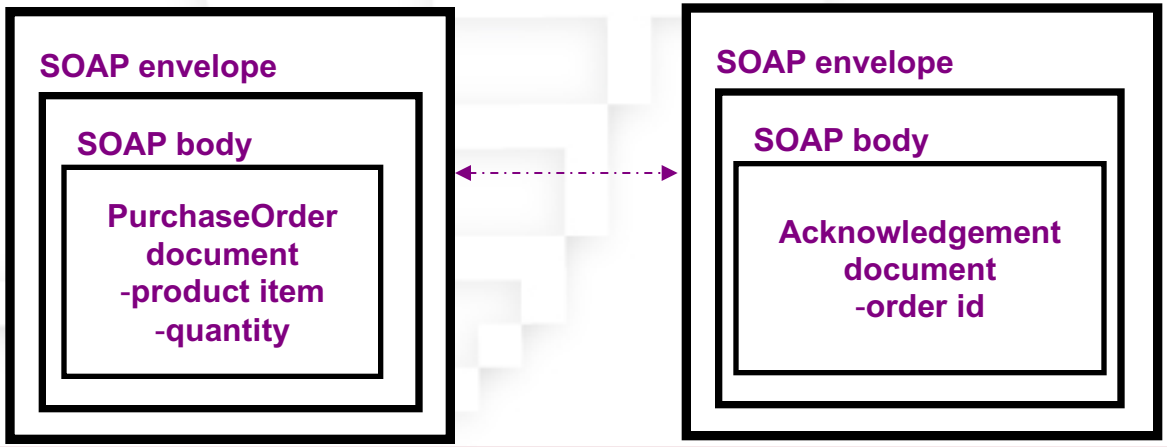
Example of RPC-style SOAP body

Example of RPC-style SOAP response message

```
<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>
```

Document (Message)-style SOAP Services

- In the document-style of messaging, the SOAP <Body> contains an XML document fragment. The <Body> element reflects no explicit XML structure.
- The SOAP run-time environment accepts the SOAP <Body> element as it stands and hands it over to the application it is destined for unchanged. There may or may not be a response associated with this message.



Example of document-style SOAP body

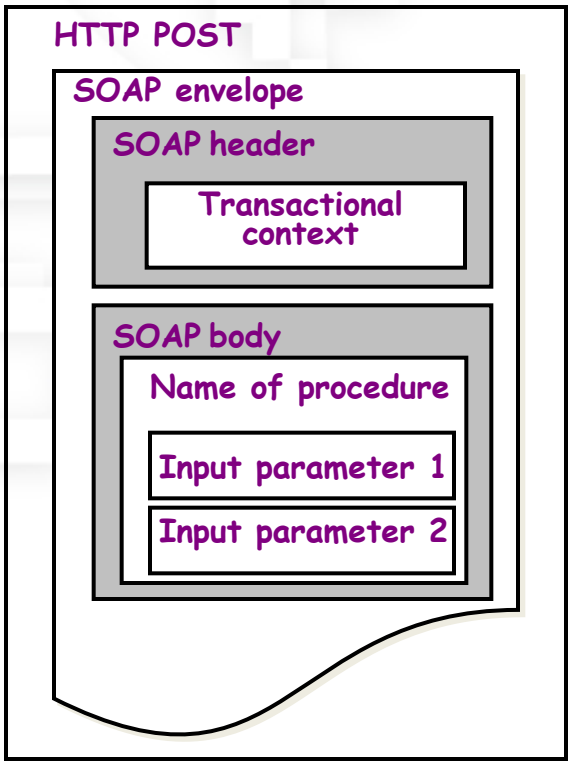
```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">

    <env:Header>
      <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
          env:mustUnderstand='1'>
              512
    </env:Header>
    <env:Body>
      <po:PurchaseOrder oderDate="2004-12-02"
           xmlns:m="http://www.plastics_supply.com/POs">
           <po:from>
             <po:accountName>   RightPlastics   </po:accountName>
            <po:accountNumber>   PSC-0343-02  </po:accountNumber>
           </po:from>
           <po:to>
             <po:supplierName>  Plastic Supplies Inc.  </po:supplierName>
             <po:supplierAddress>  Yara Valley Melbourne  </po:supplierAddress>
           </po:to>
           <po:product>
               <po:product-name> injection molder </po:product-name>
               <po:product-model> G-100T </po:product-model>
               <po:quantity> 2 </po:quantity>
           </po:product>
      </ po:PurchaseOrder >
    </env:Body>
</env:Envelope>
```

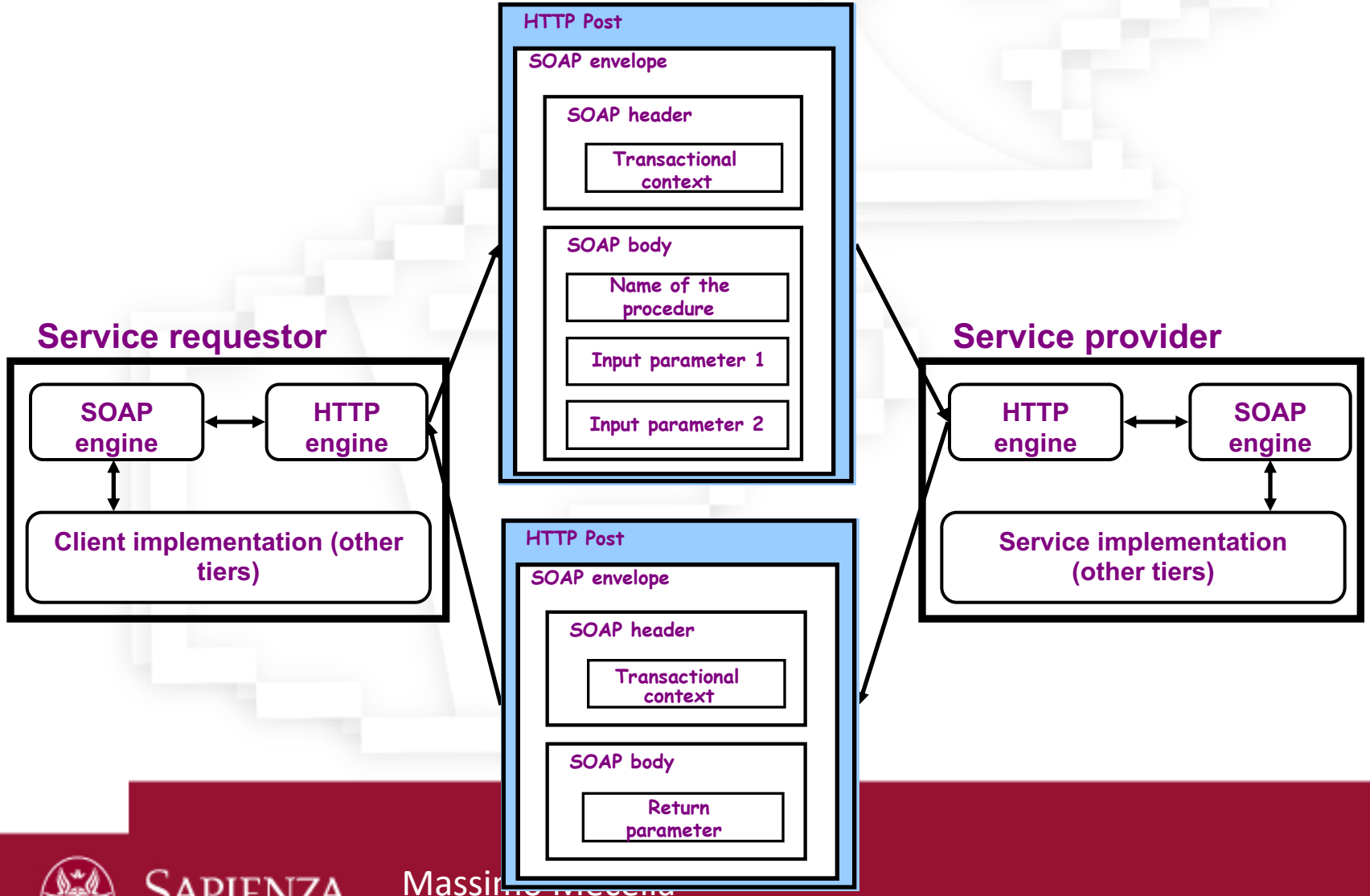
Example of document-style SOAP body

SOAP and HTTP

- A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent using that transport protocol.
- The typical binding for SOAP is HTTP.
- SOAP can use GET or POST. With GET, the request is not a SOAP message but the response is a SOAP message, with POST both request and response are SOAP messages (in version 1.2, version 1.1 mainly considers the use of POST).
- SOAP uses the same error and status codes as those used in HTTP so that HTTP responses can be directly interpreted by a SOAP module.



RPC call using SOAP over HTTP



Advantages and disadvantages of SOAP

- Advantages of SOAP are:
  - Simplicity
  - Portability
  - Firewall friendliness
  - Use of open standards
  - Interoperability
  - Universal acceptance.
- Disadvantages of SOAP are:
  - Too much reliance on HTTP
  - Statelessness
  - Serialization by value and not by reference.

SERVICE DESCRIPTION & WSDL

SAPIENZA
UNIVERSITÀ DI ROMA

Massimo Mecella
MSE-CS

Why is a Service description needed?

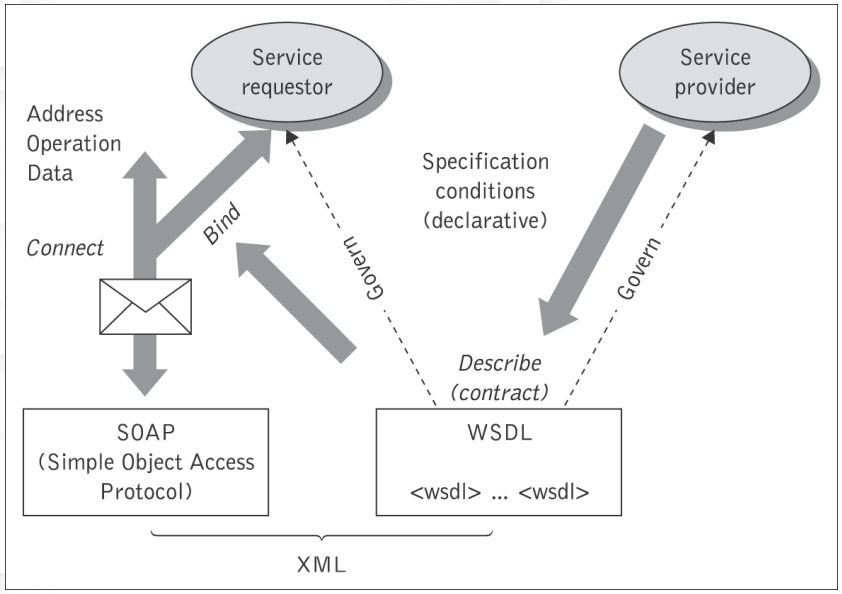
- Web services must be defined in a consistent manner to be discovered and used by other services and applications. Web service consumers must determine the precise XML interface of a web service:
  - XML Schema alone cannot describe important additional details involved in communicating with a Web service.
- **Service description** reduces the amount of required common understanding and custom programming and integration:
  - It is a machine understandable standard describing the operations of a Web service.
  - It specifies the wire format and transport protocol that the Web service uses to expose this functionality.
  - It can also describe the payload data using a type system.
- **Service description + SOAP infrastructure** isolates all technical details, e.g., machine- and implementation language-specific elements, away from the service requestor's application and the service provider's Web service.

Web Services Description Language

- The web services description language (WSDL) is the XML-based service representation language used to describe the details of the complete interfaces exposed by Web services and thus is the means to accessing a Web service.
  - For instance, neither the service requestor nor the provider should be aware of each other's technical infrastructure, programming language, or distributed object framework (if any).

WSDL as a contract

- A Web service description in WSDL is an XML document that describes the mechanics of interacting with a particular Web service.
- It is inherently intended to constrain both the service provider and the service requestor that make use of that service. This implies that **WSDL represents a "contract" between the service requestor and the service provider**
- WSDL is platform and language independent and is used primarily (but not exclusively) to describe SOAP-enabled services. Essentially, WSDL is used to describe precisely
  - ***what*** a service does, i.e., the operations the service provides,
  - ***where*** it resides, i.e., details of the protocol-specific address, e.g., a URL, and
  - ***how*** to invoke it, i.e., details of the data formats and protocols necessary to access the service's operations.



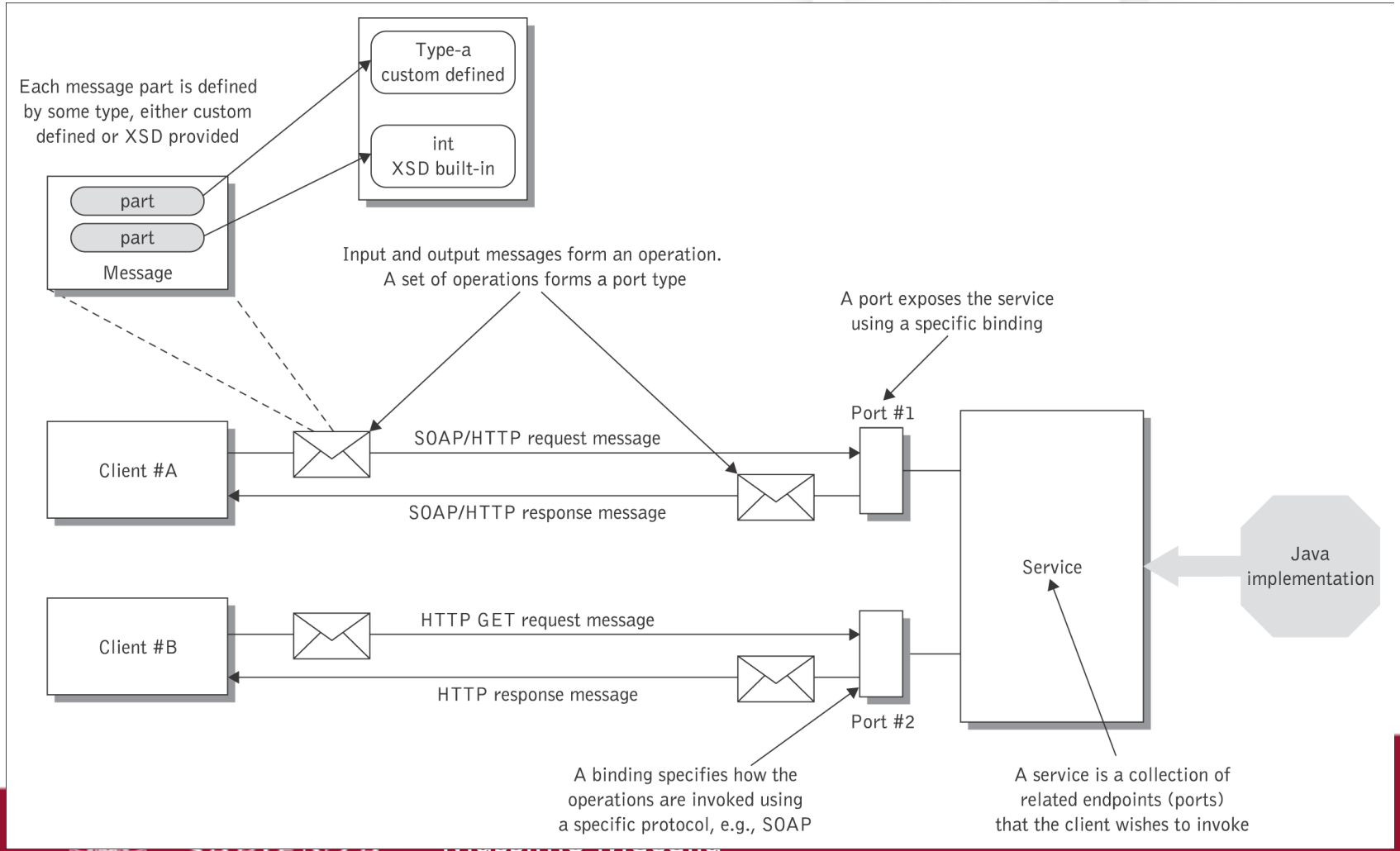
Structure of WSDL documents

- WSDL documents can be separated into distinct sections:
  - The ***service-interface definition*** describes the general Web service interface structure. This contains all the operations supported by the service, the operation parameters, and abstract data types.
  - The ***service implementation part*** binds the abstract interface to a concrete network address, to a specific protocol, and to concrete data structures.
    - A web service client may bind to such an implementation and invoke the service in question.
- This enables each part to be defined separately and independently, and **reused** by other parts.
- The combination of these two parts contains **sufficient information** to describe to the service requestor how to invoke and interact with the Web service at a provider's site.
  - Using WSDL, a requestor can locate a web service and invoke any of the publicly available operations.

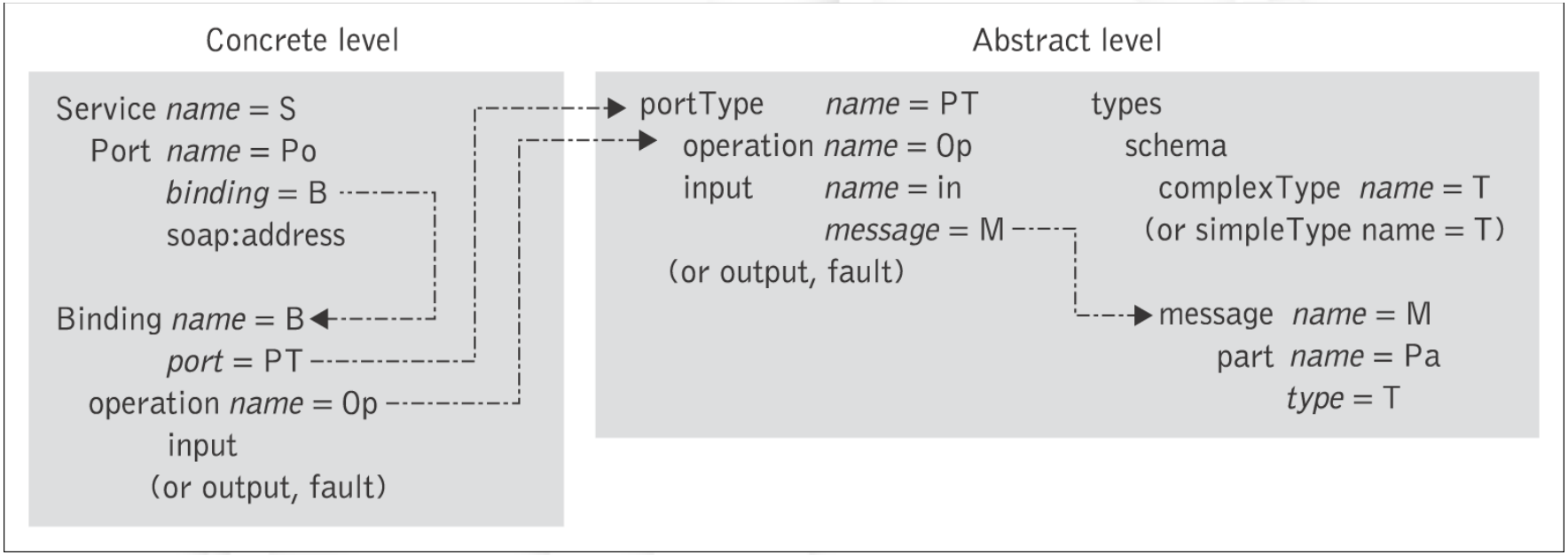
WSDL document content

- Abstract (interface) definitions
  - <types> data type definitions
  - <message> operation parameters
  - <operation> abstract description of service actions
  - <portType> set of operation definitions
- Concrete (implementation) definitions
  - <binding> operation bindings
  - <port> association of an endpoint with a binding
  - <service> location/address for each binding
- Also:
  - <import> used to reference other XML documents

Elements of WSDL as part of requestor–service interaction

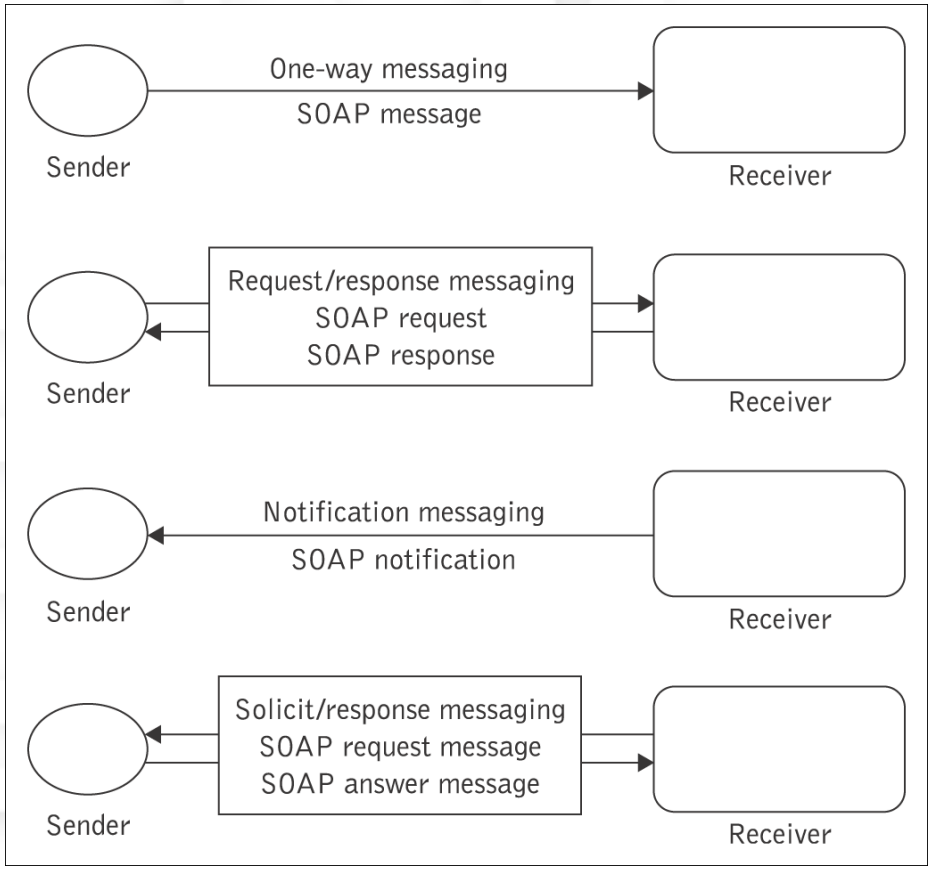


Connecting the abstract and concrete levels of a Web service.



WSDL Message Exchange Patterns

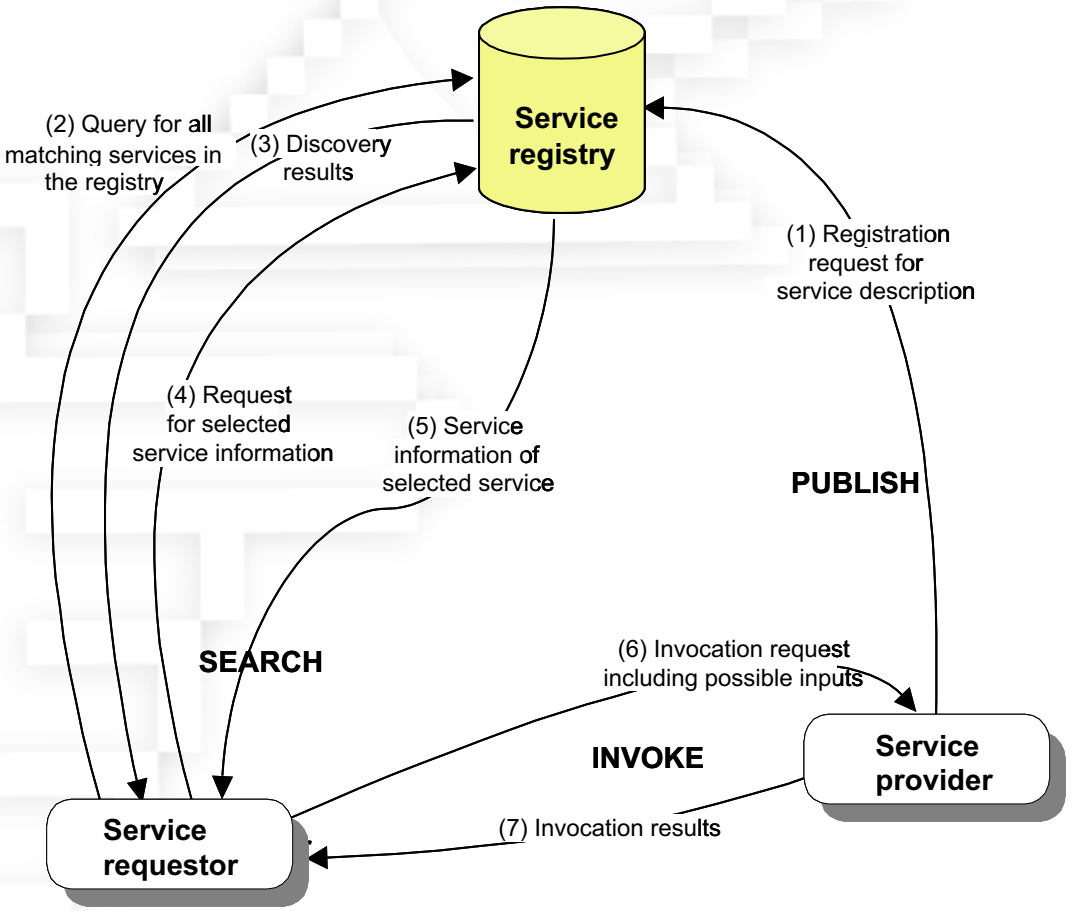
- WSDL interfaces support four common types of operations that represent possible combinations of input and output messages
- The WSDL operations correspond to the incoming and outgoing versions of two basic operation types:
  - an incoming single message passing operation and its outgoing counterpart ("one-way" and "notification" operations),
  - the incoming and outgoing versions of a synchronous two-way message exchange ("request/response" and "solicit response").
- Any combination of incoming and outgoing operations can be included in a single WSDL interface:
  - these four types of operations provide support for both push and pull interaction models at the interface level.



REGISTRY AND UDDI

SOA interactions between actors

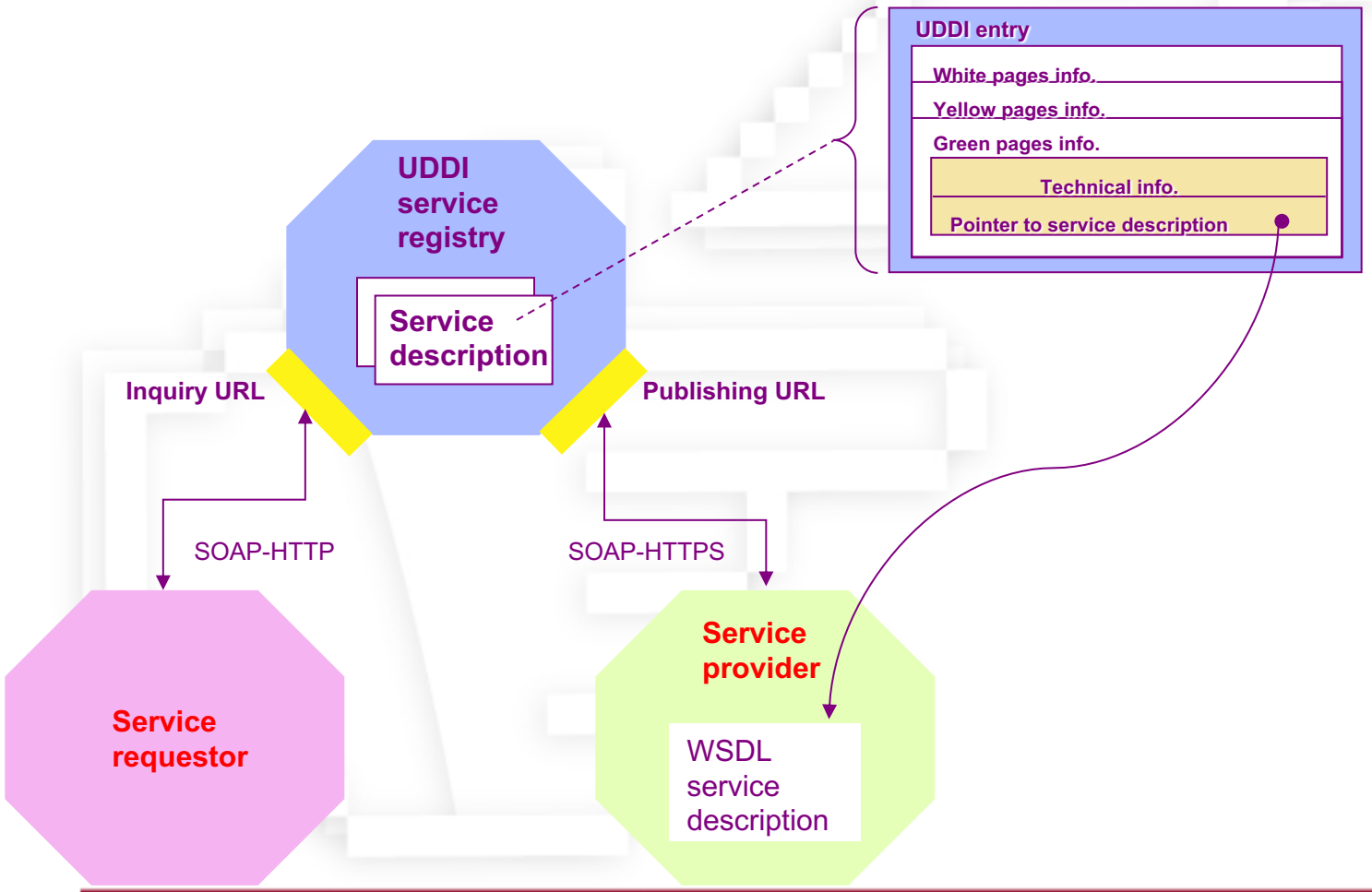
- Problem to solve:
  - How to find the service a client wants among a large collection of services and providers.
  - The client does not necessarily need to know which provider provides the service.



What is UDDI?

- The universal description, discovery, and integration is a registry standard for Web service description and discovery together with a registry facility that supports the WS publishing and discovery processes.
- UDDI enables a business to:
  - **describe** its business and its services;
  - **discover** other businesses that offer desired services;
  - **integrate (interoperate)** with these other businesses.
- Conceptually, a UDDI business registration consists of three inter-related components:
  - “white pages” (address, contact, and other key points of contact);
  - “yellow pages” classification info. based on standard industry taxonomies; and
  - “green pages”, the technical capabilities and information about services.

UDDI and WSDL



Pitfalls of Web Services

- As the business requirements that drive Web services become even more complex, Web services technologies require additional capabilities to handle demanding situations that severely test the most obvious current shortcomings of Web services. These include:
  - performance issues,
  - lack of appropriate support for sophisticated transaction management,
  - lack of expressing business semantics and, especially,
  - achieving a widespread agreement and harmonization on a wide range of existing and emerging standards.
    - There is an overwhelming number of existing and emerging standards.
    - Unless these standards mature enough to the degree that they are harmonized and can act together, the long-term viability of Web services applications being used in mission-critical, production environments will be severely tested.

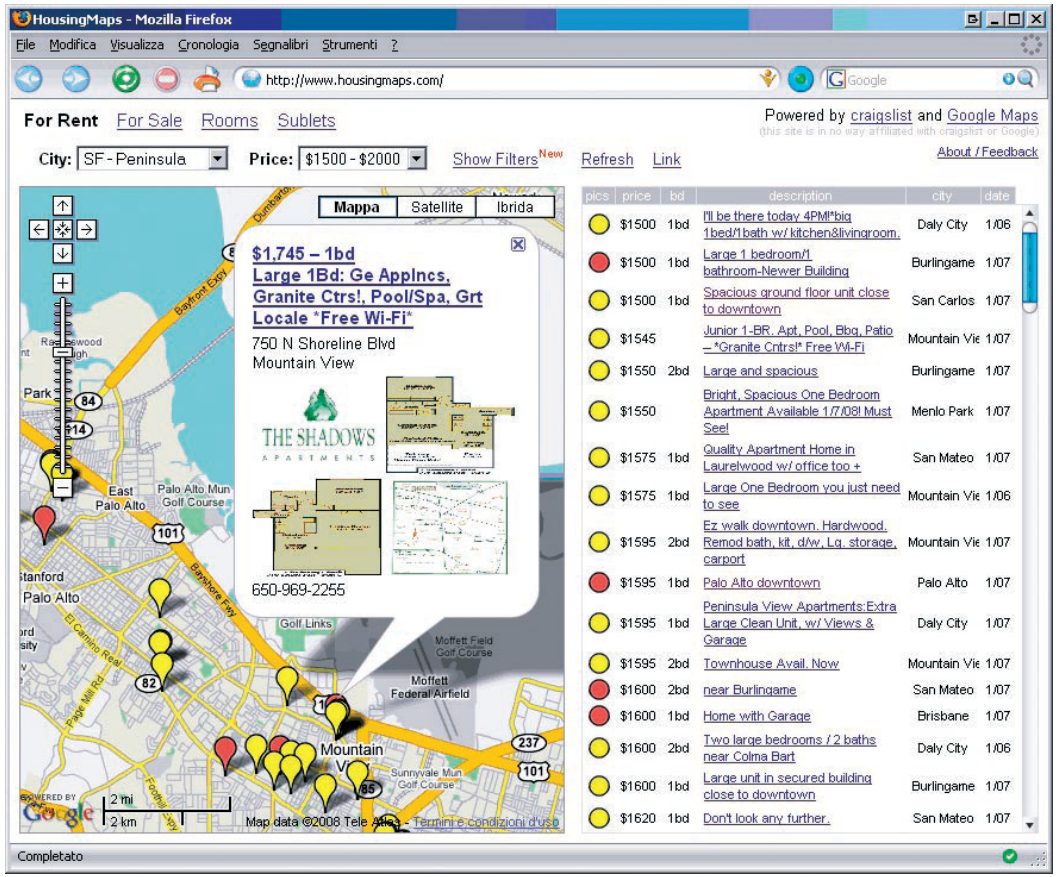
DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Services Mash-up (1)

Web application that combines data from one or more sources into a single integrated tool

- easy, fast integration, frequently done by access to open APIs and data sources to produce results that were not the original reason for producing the raw source data.
- E.g., cartographic data from Google Maps to add location information to real estate data, thereby creating a new and distinct e-Service that was not originally provided by either source
- Bottom-up, developers-driven approach



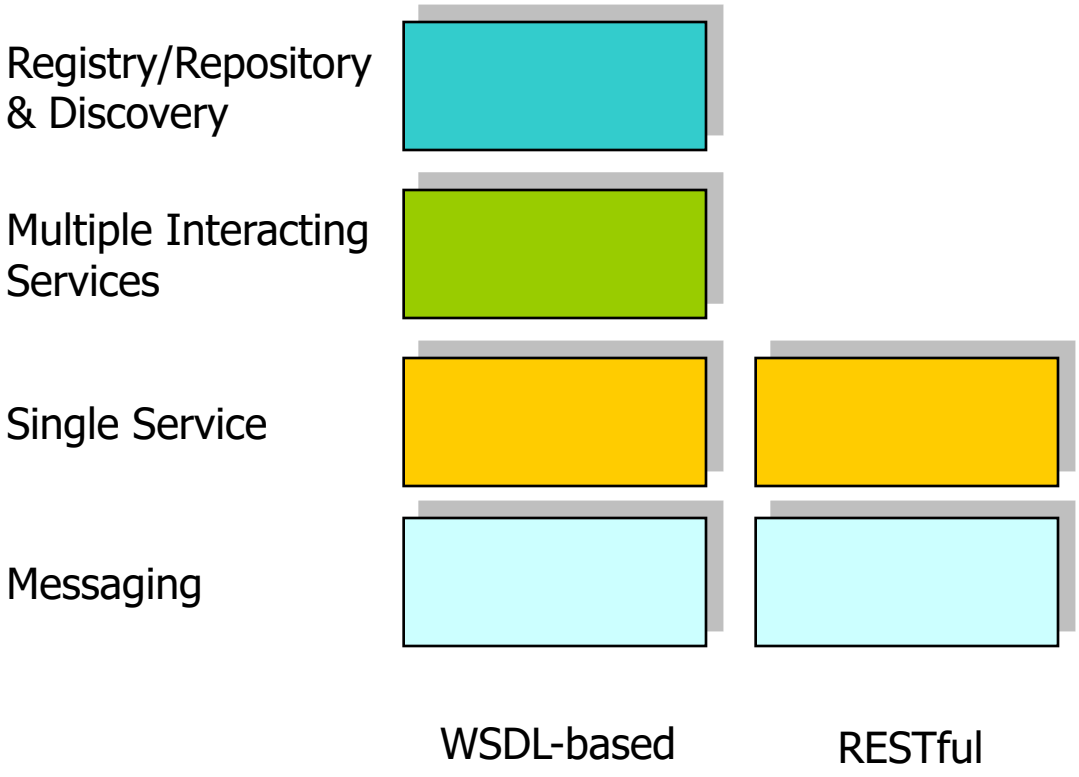
D. Benslimane, S. Dustdar, A. Sheth (eds.). Services Mashups – Special Issue. IEEE Internet Computing, vol. 12, no. 5, 2008.

Services Mash-up (2)

- Based on various technologies
  - Web services
    - SOAP
    - RESTful
    - Atom/RSS
- Basically a lightweight form of composition

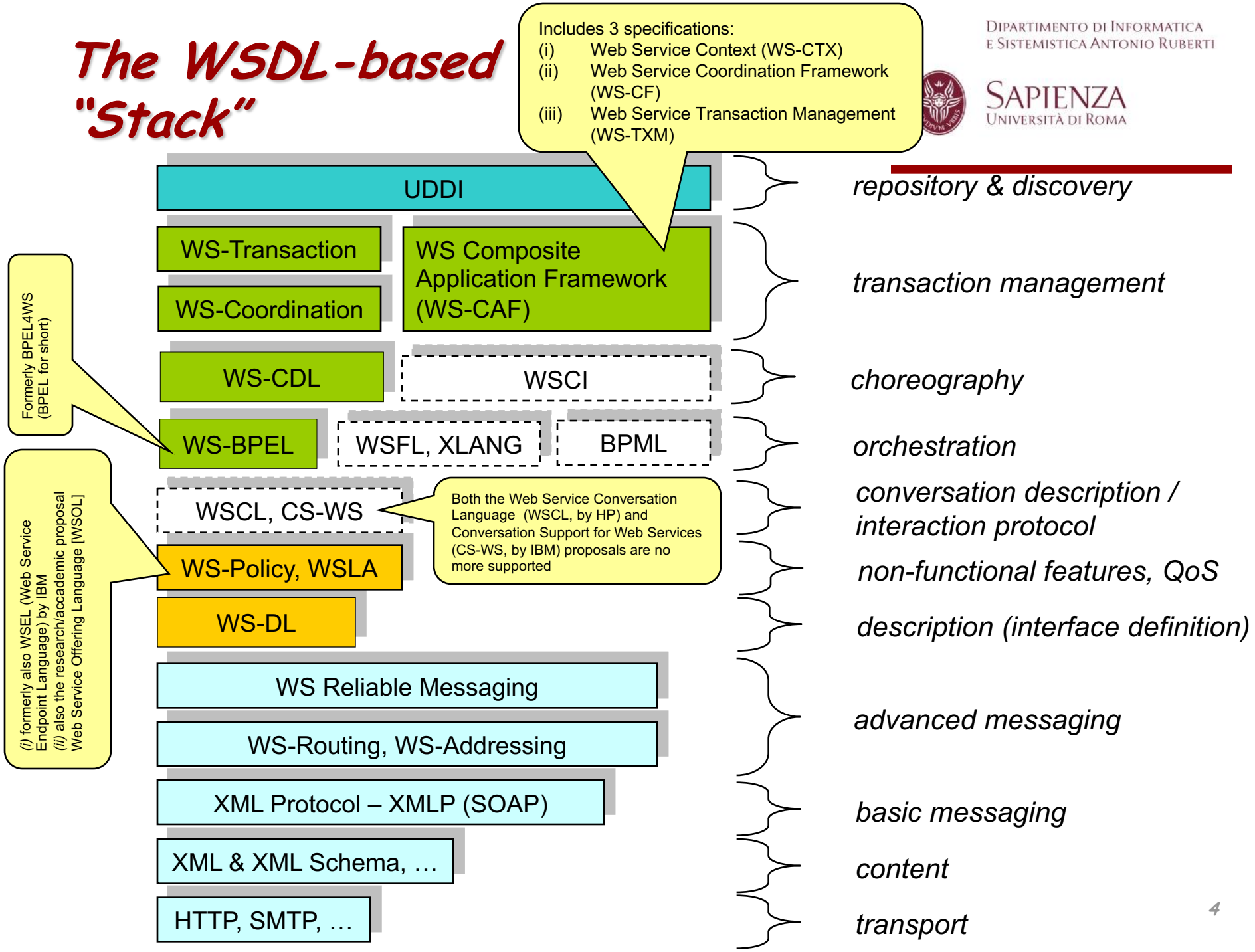
The "Stacks" of Service Technologies

Dipartimento di Informatica
e Sistemistica Antonio Ruberti
Sapienza
Università di Roma



The WSDL-based "Stack"

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma



RESTFUL SERVICES

RESTful Services (1)

Dipartimento di Informatica
e Sistemistica Antonio Ruberti

Sapienza
Università di Roma

- REST refers to simple application interfaces transmitting data over HTTP without additional layers as SOAP
  - Web page meant to be consumed by program as opposed to a Web browser or similar UI tool
  - require an architectural style to make sense of them (the REST one), because there's no smart human being on the client end to keep track

RESTful Services (2)

Dipartimento di Informatica
e Sistemistica Antonio Ruberti

Sapienza
Università di Roma

- Metaphor based on nouns and verbs
  - URIs ~ nouns
  - Verbs describe actions that are applicable to nouns
    - GET -- retrieve information / READ, SELECT
    - POST (PUT) – add/update new information / CREATE, INSERT, UPDATE
    - DELETE -- discard information / DELETE
- State means the application/session state, maintained as part of the content transferred (in XML) from client to server back to client

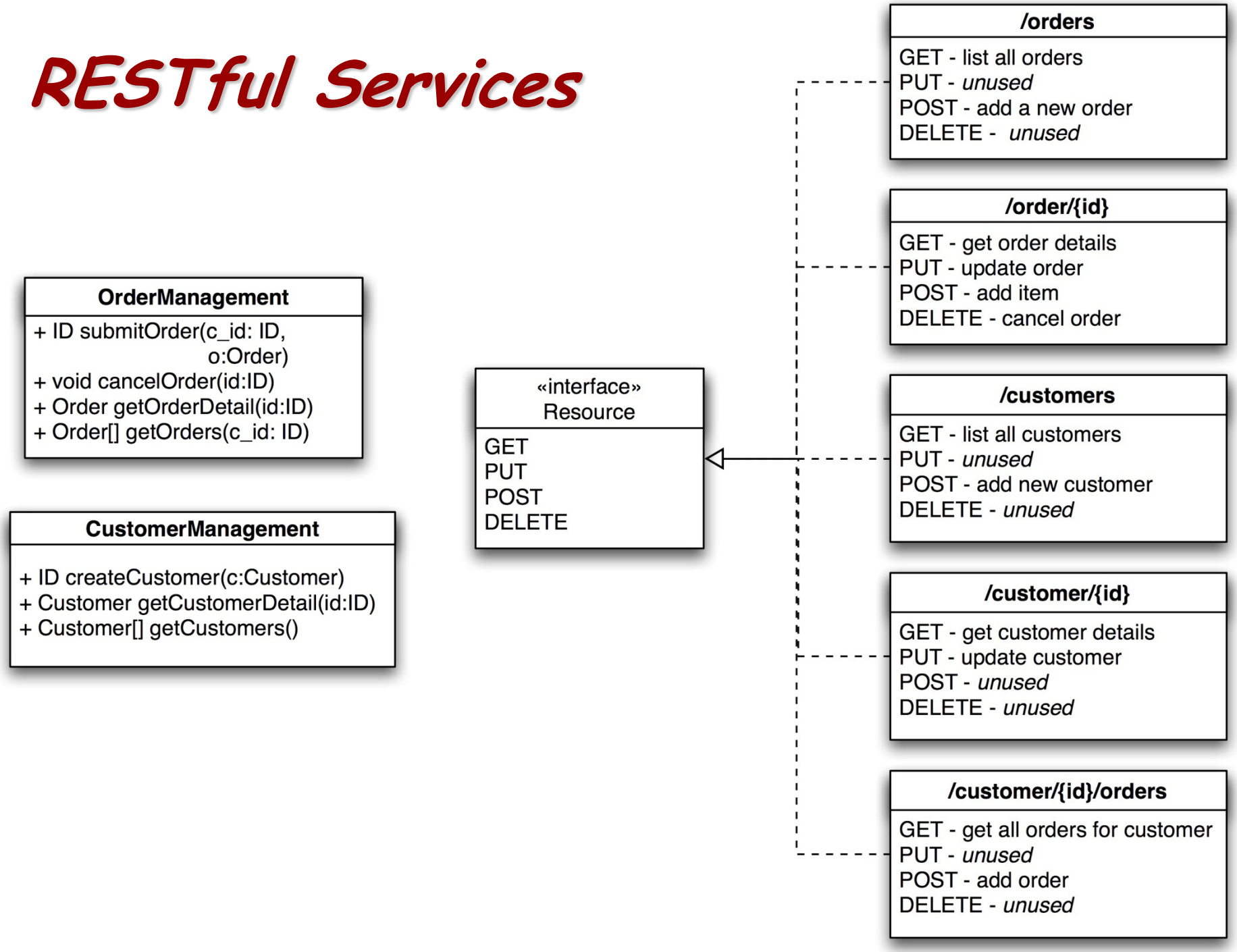
RESTful Services (3)

Dipartimento di Informatica
e Sistemistica Antonio Ruberti

Sapienza
Università di Roma

- REST is, in a sense, a kind of RPC, except the methods have been defined in advance
  - Consider the stock example of a remote procedure called "getStockPrice"
  - It's not clear what what it means to GET, PUT, and POST to something called "getStockPrice"
  - But if we change the name from "getStockPrice" to "CurrentStockPrice" all is well !!

RESTful Services



RESTful Principles

Dipartimento di Informatica
e Sistemistica Antonio Ruberti
Sapienza
Università di Roma

- Addressability (each **resource** an URL)
- Uniform Interface (HTTP GET, PUT, …)
- Stateless Interactions (no session)
- Self-Describing Messages (metadata)
- Hypermedia (hyper-links)

RESTful Services (4)

Dipartimento di Informatica
e Sistemistica Antonio Ruberti
Sapienza
Università di Roma

- REST is incompatible with "end-point" RPC -- Either you address data objects or you address "software components"
  - REST does the former
  - End-point RPC does the latter

```
POST /purchase_orders HTTP/1.1
Host: accounting.mycompany.com
content-type:
application/purchase-order+xml
....
<po>...</po>
```

```
POST /generic_message_handler
content-type: application/SOAP+XML
<soap:envelope>
   <soap:body>
     <submit-purchase-order>
       <destination>accounting.mycompany.com
       </destination>
       <po>...</po>
     </submit-purchase-order>
   </soap:body>
<soap:envelope>
```

Example (1)

Operation	HTTP Request	HTTP Response	Java Technology Method
Create	POST /restfulwebservice-war/poservice/ HTTP/1.0 Accept: */* Connection: close Content-Type: text/xml Content-Length: 618 Pragma: no-cache <tns:PurchaseOrderDocument xmlns:tns="urn:PurchaseOrderDocument"> <billTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </billTo> <createDate>2004-03-27T12:21:02.055-05:00</createDate> <poID>ABC-CO-19282</poID> <items> <itemname>Copier Paper</itemname> <price>10</price> <quantity>2</quantity> </items> <items> <itemname>Toner</itemname> <price>920</price> <quantity>1</quantity> </items> <shipTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </shipTo> </tns:PurchaseOrderDocument>	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml Date: Fri, 21 Jul 2006 17:07:15 GMT Connection: close <?xml version="1.0" encoding="UTF-8"?> <ns2:Status xmlns:ns2="urn:Status" xmlns:ns3="urn:PurchaseOrderDocument" xmlns:ns4="urn:POProcessingFault"> <orderid>ABC1153501634787</orderid> <timestamp>Fri Jul 21 13:07:14 EDT 2006</timestamp> </ns2:Status>	public PurchaseOrderStatus acceptPO(PurchaseOrder order)

Example (2)

Read	GET /restfulwebservice-war/poservice/ABC1153501634787 HTTP/1.0 Connection: close Content-Type: text/xml	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml Connection: close <?xml version="1.0" encoding="UTF-8"?><ns3:PurchaseOrderDocument xmlns:ns3="urn:PurchaseOrderDocument" xmlns:ns2="urn:Status" xmlns:ns4="urn:POProcessingFault"><billTo><street>1 Main Street</street><city>Beverly Hills</city><state>CA</state><zipCode>90210</zipCode></billTo><createDate>2006-07-21T13:08:37.505-04:00</createDate><items><itemname>Copier Paper</itemname><price>10</price><quantity>2</quantity></items><items><itemname>Toner</itemname><price>920</price><quantity>1</quantity></items><poID>/ABC1153501634787</poID><shipTo><street>1 Main Street</street><city>Beverly Hills</city><state>CA</state><zipCode>90210</zipCode></shipTo></ns3:PurchaseOrderDocument>	public PurchaseOrder retreivePO (String orderID)
	GET /restfulwebservice-war/poservice/ HTTP/1.1 Connection: keep-alive	HTTP/1.1 400 Bad Request X-Powered-By: Servlet/2.5 Content-Type: text/xml <?xml version="1.0" encoding="UTF-8"?><ns4:POProcessingFault xmlns:ns4="urn:POProcessingFault" xmlns:ns2="urn:Status" xmlns:ns3="urn:PurchaseOrderDocument"><message>Unable to retrieve the order associated with the orderid you specified</message></ns4:POProcessingFault>	Indicates a problem finding the order

Example (3)

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Update	PUT /restfulwebservice-war/poservice/ HTTP/1.0 Connection: close Content-Type: text/xml Content-Length: 620 Pragma: no-cache <tns:PurchaseOrderDocument xmlns:tns="urn:PurchaseOrderDocument"> <billTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </billTo> <createDate>2004-03-27T12:21:02.055-05:00</createDate> <poID>ABC-CO-19282</poID> <items> <itemname>Copier Paper</itemname> <price>10</price> <quantity>2</quantity> </items> <items> <itemname>Toner</itemname> <price>920</price> <quantity>1</quantity> </items> <shipTo> <street>1 Main Street</street> <city>Beverly Hills</city> <state>CA</state> <zipCode>90210</zipCode> </shipTo> </tns:PurchaseOrderDocument>	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml <?xml version="1.0" encoding="UTF-8"?><ns3:PurchaseOrderDocument xmlns:ns3="urn:PurchaseOrderDocument" xmlns:ns2="urn:Status" xmlns:ns4="urn:POProcessingFault"> <billTo><street>1 Main Street</street><city>Beverly Hills</city><state>CA</state><zipCode>90210</zipCode> </billTo> <createDate>2004-03-27T12:21:02.055-05:00</createDate> <items><itemname>Copier Paper</itemname> <price>10</price><quantity>2</quantity></items> <items><itemname>Toner</itemname><price>920</price> <quantity>1</quantity></items> <poID>ABC-CO-19282</poID><shipTo><street>1 Main Street</street><city>Beverly Hills</city> <state>CA</state><zipCode>90210</zipCode></shipTo> </ns3:PurchaseOrderDocument>	public PurchaseOrder updatePO(PurchaseOrder order)
Delete	DELETE /restfulwebservice-war/poservice/ABC-CO-19282 HTTP/1.0 Connection: close Content-Type: text/xml Content-Length: 0 Pragma: no-cache	HTTP/1.1 200 OK X-Powered-By: Servlet/2.5 Content-Type: text/xml Date: Fri, 21 Jul 2006 17:10:38 GMT Server: Sun Java System Application Server Platform Edition 9.1 Connection: close <?xml version="1.0" encoding="UTF-8"?>	public void cancelPO(String orderID)

Maturity of RESTful Services

Dipartimento di Informatica
e Sistemistica Antonio Ruberti

Sapienza
Università di Roma

Levels:

- 0. HTTP as a tunnel
- 1. Resources
- 2. HTTP Verbs
- 3. Hypermedia

Why so trendy ?

- Easy and lightweight
- Amazon, Yahoo, Google offer their Web services as RESTful

- … but nothing really new for us, basically the same abstractions apply, you can consider the operations as a whole or you can start modeling the data flowing through the service

A FINAL TERMINOLOGICAL NOTE

e-Services, Web Services, Services … (1) - Historically

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- An e-Service is often defined as an application accessible via the Web, that provides a set of functionalities to businesses or individuals. What makes the e-Service vision attractive is the ability to automatically discover the e-Services that fulfill the users' needs, negotiate service contracts, and have the services delivered where and when users needs them

  Guest editorial. In [VLDBJ01]

- e-Service: an application component provided by an organization in order to be assembled and reused in a distributed, Internet-based environment; an application component is considered as an e-Service if it is: (i) open, that is independent, as much as possible, of specific platforms and computing paradigms; (ii) developed mainly for inter-organizations applications, not only for intra-organization applications; (iii) easily composable; its assembling and integration in an inter-organizations application does not require the development of complex adapters. e-Application: a distributed application which integrates in a cooperative way the e-Services offered by different organizations

  M. Mecella, B. Pernici: Designing Wrapper Components for e-Services in Integrating Heterogeneous Systems. In [VLDBJ01]

e-Services, Web Services, Services … (2) - Historically

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols

Web Services Architecture Requirements, W3C Working Group Note, 11 Feb. 2004,
http://www.w3.org/TR/wsa-reqs/

e-Services, Web Services, Services … (3) - Historically

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma

- Services are self-describing, open components that support rapid, low-cost composition of distributed applications. Services are offered by service providers — organizations that procure the service implementations, supply their service descriptions, and provide related technical and business support.

  Since services may be offered by different enterprises and communicate over the Internet, they provide a distributed computing infrastructure for both intra and cross-enterprise application integration and collaboration.

  Service descriptions are used to advertise the service capabilities, interface, behavior, and quality. Publication of such information about available services provides the necessary means for discovery, selection, binding, and composition of services. In particular, the service capability description states the conceptual purpose and expected results of the service (by using terms or concepts defined in an application-specific taxonomy). The service interface description publishes the service signature (its input/output/error parameters and message types). The (expected) behavior of a service during its execution is described by its service behavior description. Finally, the Quality of Service (QoS) description publishes important functional and nonfunctional service quality attributes […]. Service clients (end-user organizations that use some service) and service aggregators (organizations that consolidate multiple services into a new, single service offering) utilize service descriptions to achieve their objectives.
- The application on the Web (including several aspects of the SOA) is manifested by Web services

Guest editorial. In [CACM03]

And Today ?

- e-Service
  - e-Service is the provision of a service via the Internet (the prefix "e" standing for "electronic")
  - True Web jargon, meaning just about anything done online
  - Basically whichever Web application usable by a human, through a user interface
- Web service
  - software component available on the Web, to be invoked by some other client application/component
  - A way of building Web-scale component-based distributed systems
- For building an e-Service, a designer may need to use/invoke many Web services