5. Web Services

5.0 Service-Oriented paradigm

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.

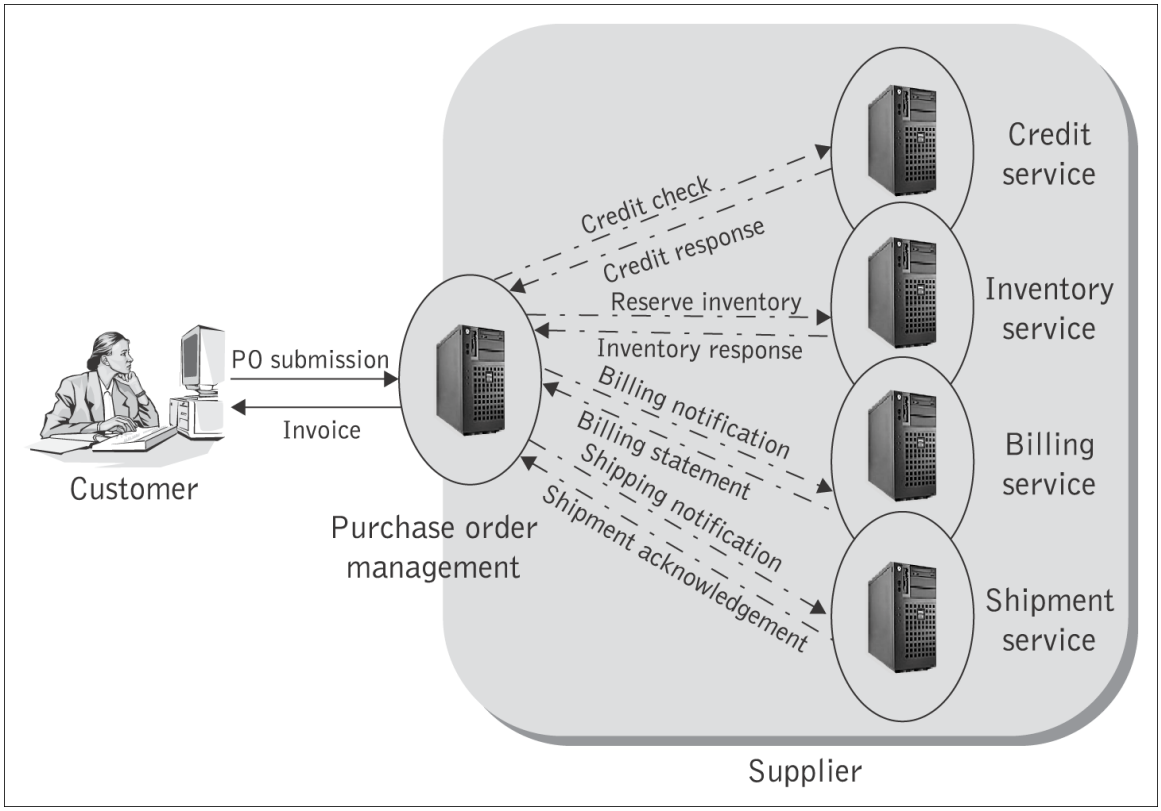
Any piece of code and any application component deployed on a system can be transformed into a network-available service.

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.

5.1 What are Web Services?

Web services perform encapsulated business functions, can be mixed and matched to create a complete process (enable integration, can be applied to new and extensions to existing apps).

Platform independent.



Purchase order app involving Web services.

5.1.1 ASP (Application Service Providers)

First concept of sw-as-a-service

An ASP rents apps to subscribers:

The whole app is developed in terms of the UI, workflow, business and data components that are all bound together.

An ASP hosts the entire app and the customer has little customize option (such as the logo).

An alternative is to provide a sw module downloadable.

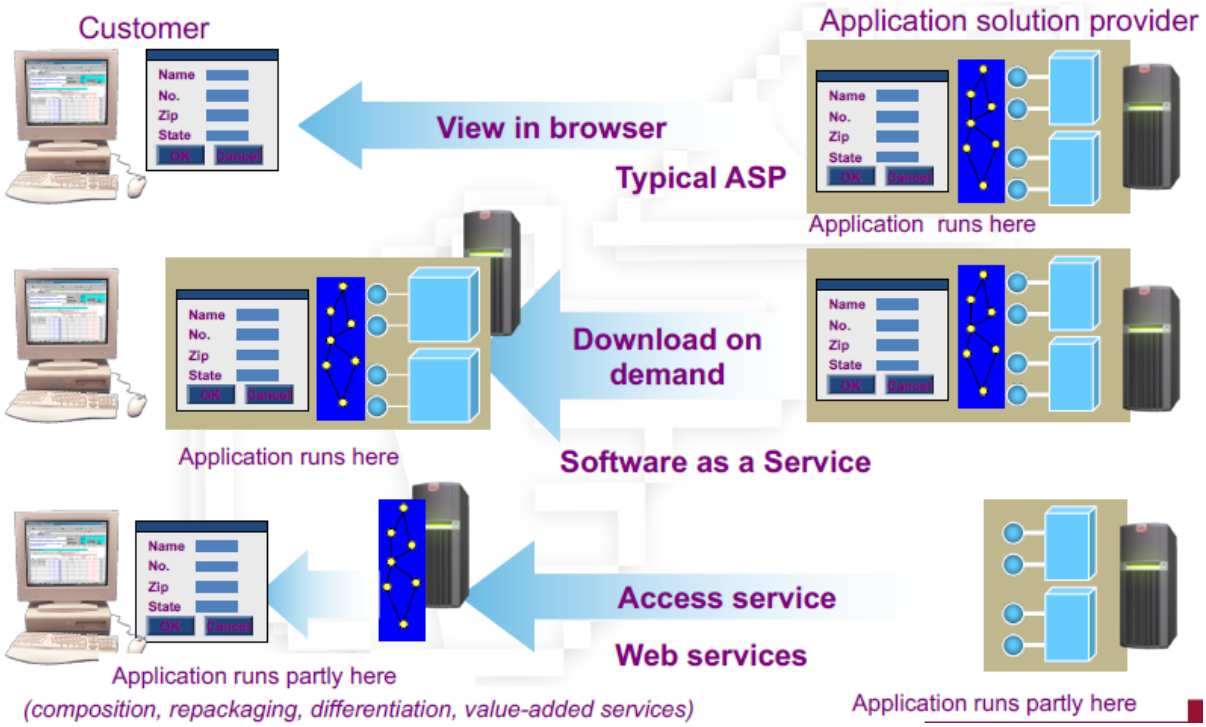
5.1.2 ASP vs. Web Services

ASP suffers limitations: inability of:

develop highly interactive apps, provide complete customizable apps, integrate apps.

Web services allow to:

loosely couples asynchronous interactions, based on XML standards, easier access and communication between apps over the internet.

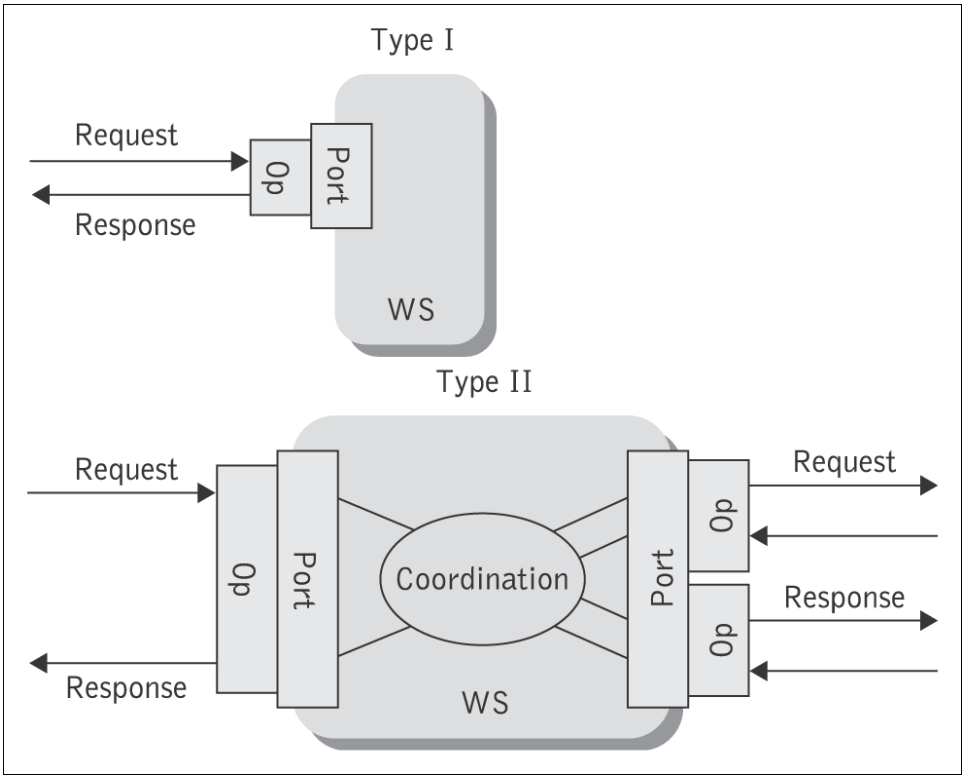


5.1.3 Where are Web Services used?

- Within enterprise (for integration)
- Between enterprises (e-Business integration)

5.1.4 Web Services Types

- Informational: either provide access to content interacting with an end-user by means of simple request/response sequences, or expose back-end business aps to other apps.
- Complex: involve the assembly and invocation of many pre-existing services possibly found in diverse enterprises to complete a multi-step business interaction.



5.1.5 Properties and States

Properties:

- the functional description details the operational characteristics that define the overall behavior of the service.
- the non-functional description targets service quality attributes.

States:

- stateless: services that can be invoked repeatdly without having to maintain context or state.
- stateful: require their context to be perserved

5.1.6 Loose coupling and Granularity

Loose coupling:

- Coupling indicates the degree of dependency, Web Services are loosely coupled: they connect and interact freely.

Service granularity:

- simple services are discrete, exhibit normally a request/reply; unlike, complex services are coarse-grained, these involve interactions with other services.
- Coarse-grained communication implies larger and richer data structures.

5.1.7 Synchronicity and Well-definedness

Synchronicity:

- Synchronous (or RPC-style): clients express their request as a method call, which returns a response containing a return value.
- Asynchronous (or message-style): when a client invokes a message-style service, it sends an entire document.

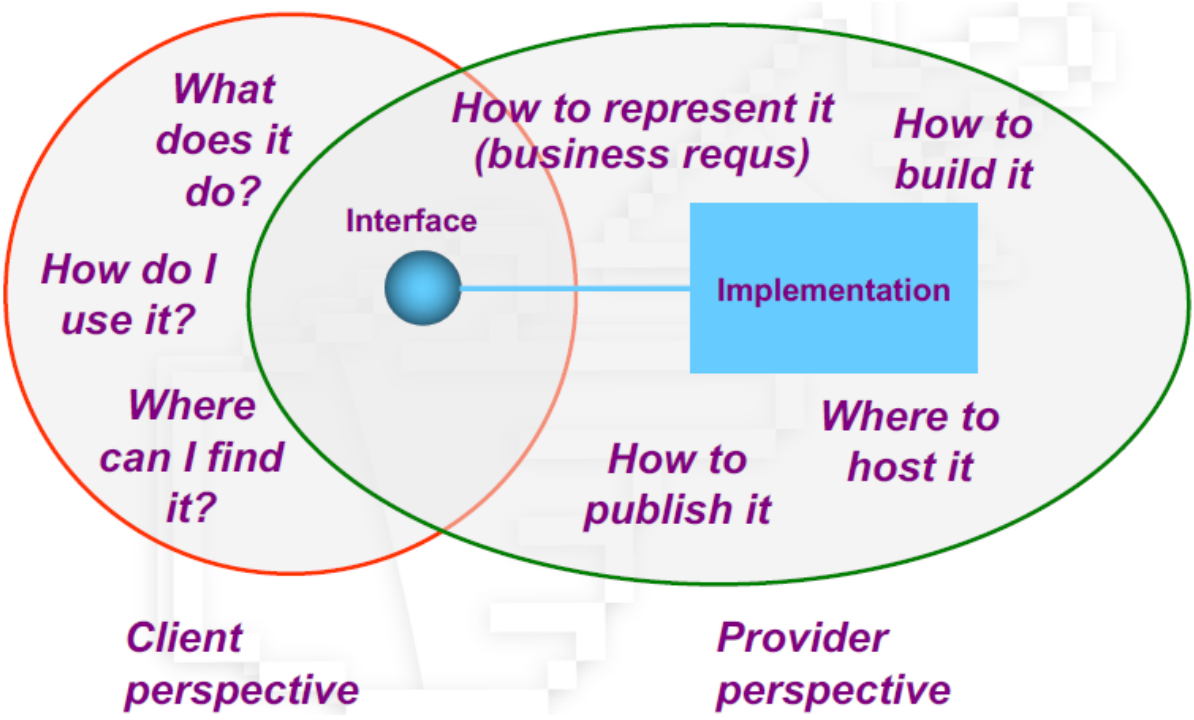
Well-definedness:

- the service must be well-defined; the Web Services Description Language (WSDL) allows applications to describe the rules for interfacing and interacting to other apps.

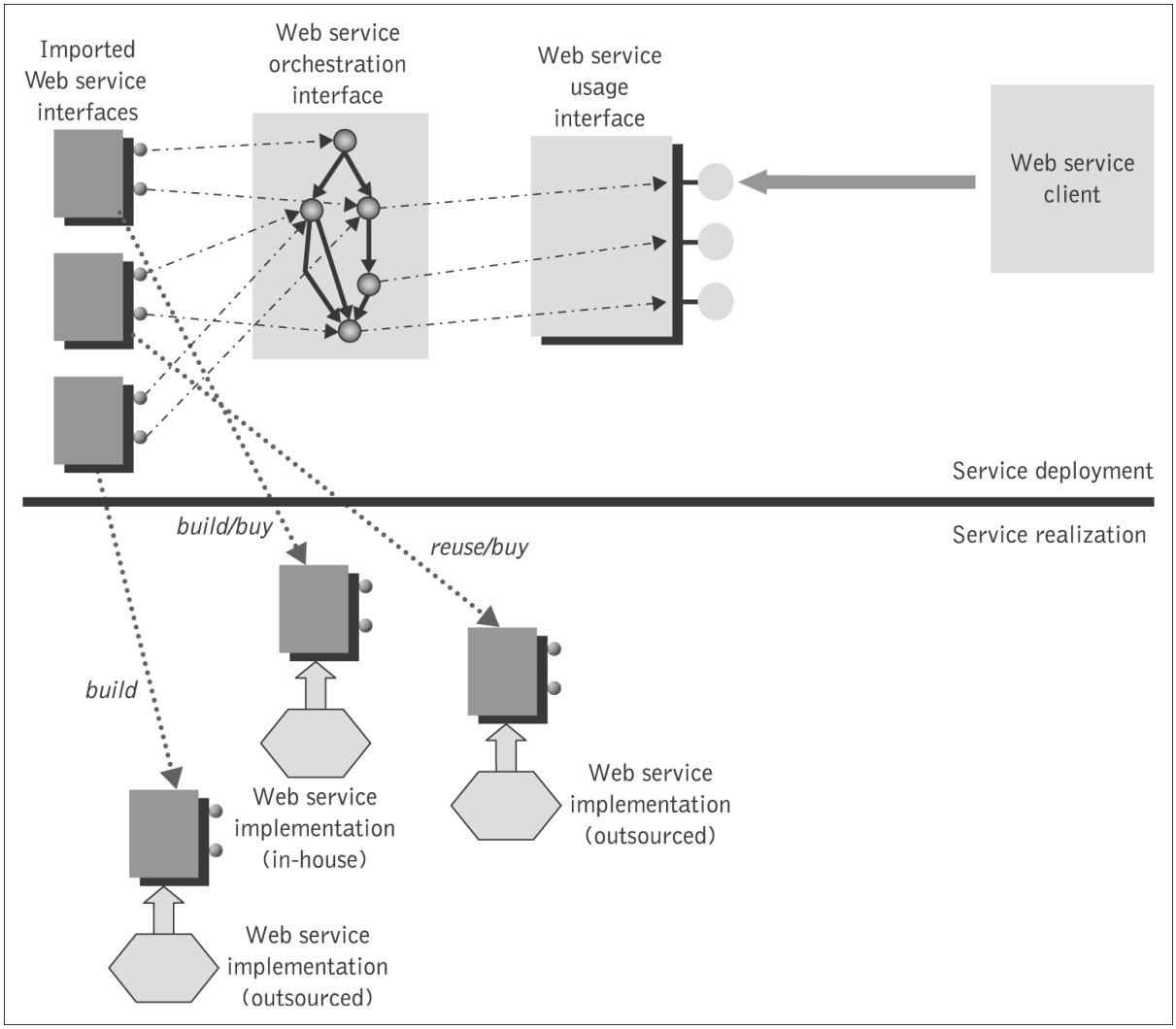
5.1.8 Service interface and implementation

The service interface defines service functionality visible to the external world and provides the means to access this functionality.

The service implementation realizes a specific service interface whose implementation details are hidden from its users.



5.1.9 Service deployment vs. service realization



5.1.10 Roles

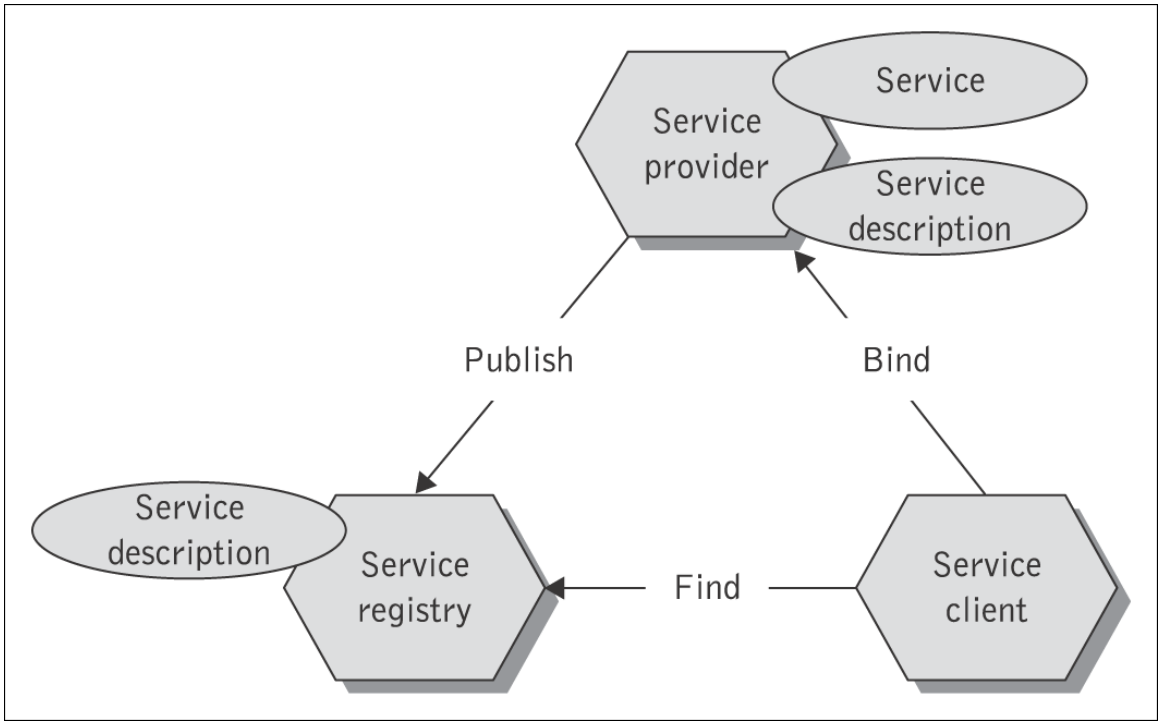
Distinction between:

- service providers: organizations that provide the service implementations;
- service clients: end-user organizations;
- service registry: searchable directory where service descriptions can be published and searched:
  - service requestors find service descriptions in the registry and obtain information for services.

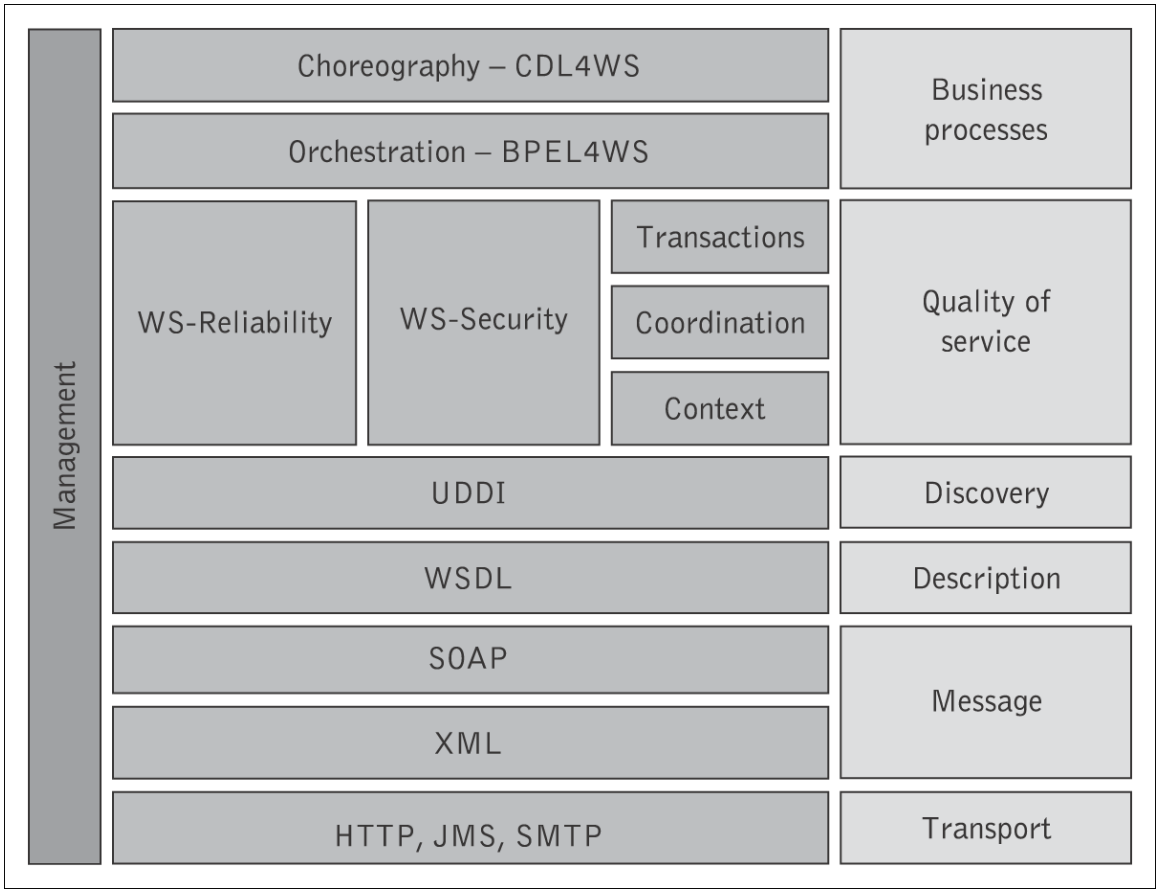
SOA

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



Quality of Service

QoS refers to the ability of a Web service to respond to expected invocations and perform them at the level commensurate.

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.

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

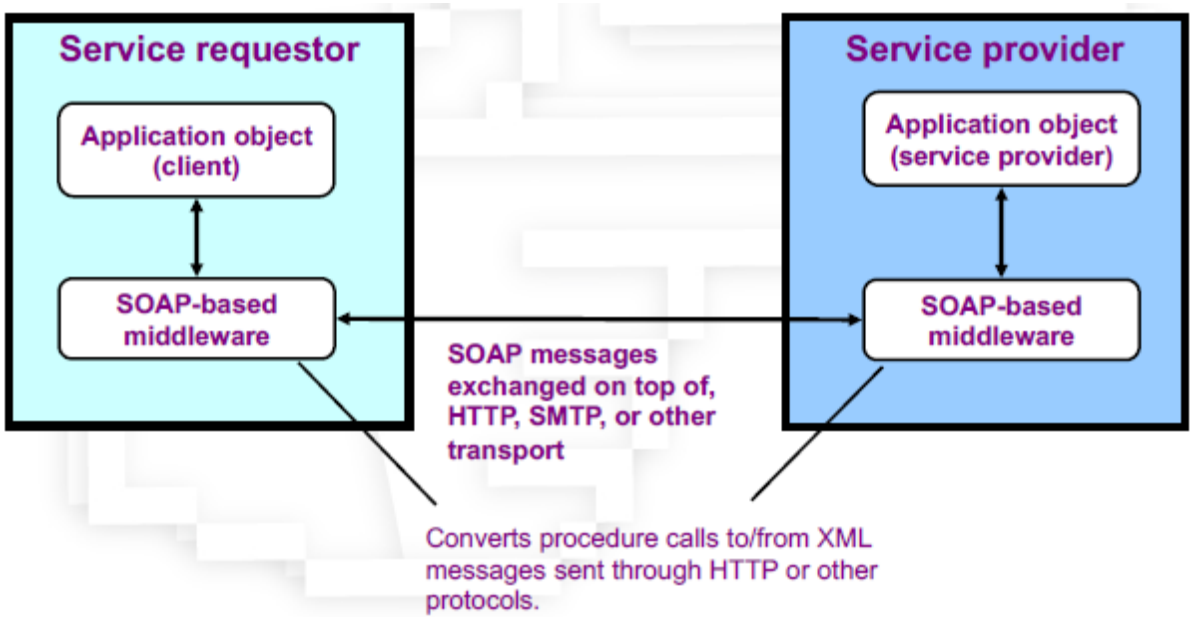
both the internal and the extended enterprise;
– a common facade for cross-enterprise specific systems, making it easier to create the service-level agreements needed for business-to-business integration.

5.2 Communication & SOAP

To address the problem of overcoming proprietary systems running on heterogeneous infrastructures, Web Services rely on SOAP.

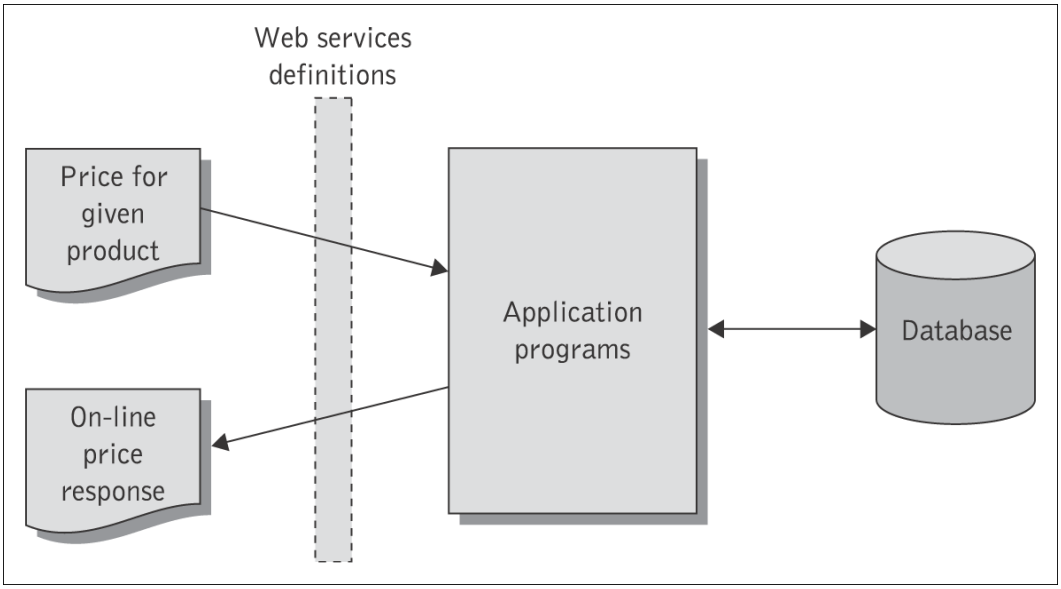
5.2.1 What is SOAP?

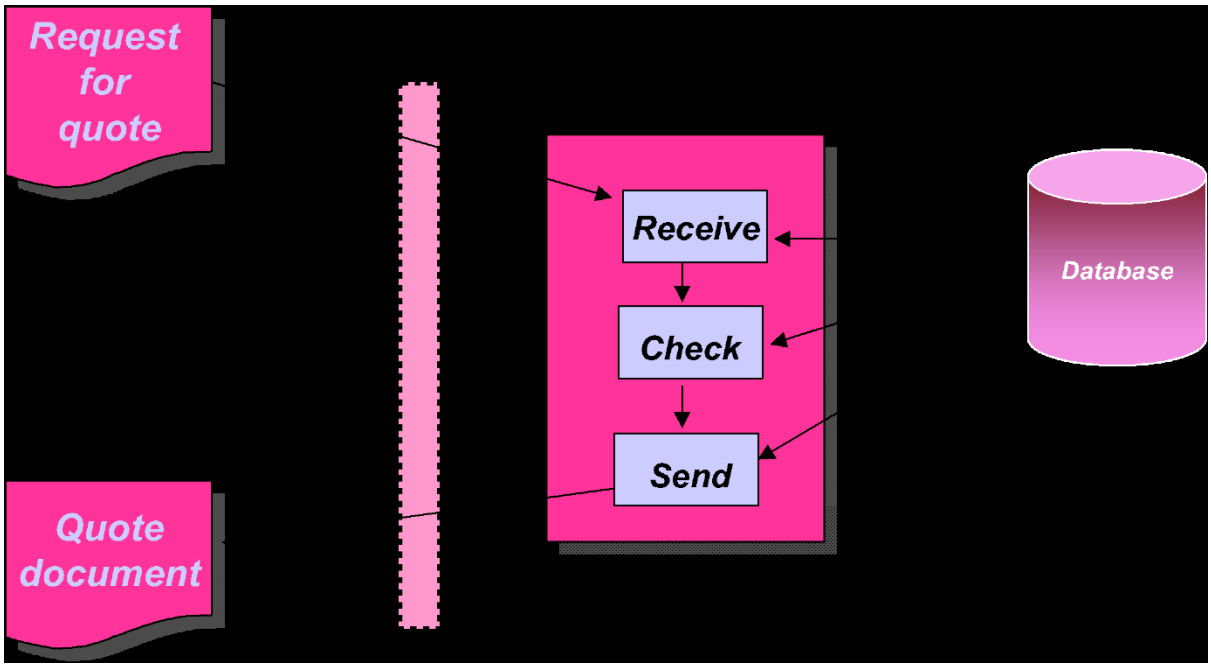
SOAP is an XML-based messaging protocol used by Web Services for exchanging messages inter-application.



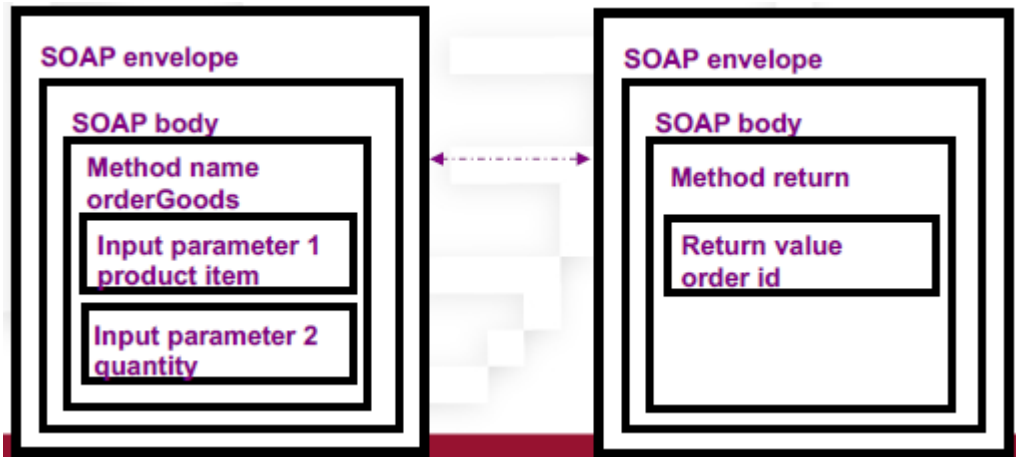
5.2.2 SOAP Communication model

5.2.2.1 RPC-style





Web Service appears as a remote object to a client application. Clients express their request as a method call with a set of arguments, which returns a response containing a return value.



An example:

```
<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

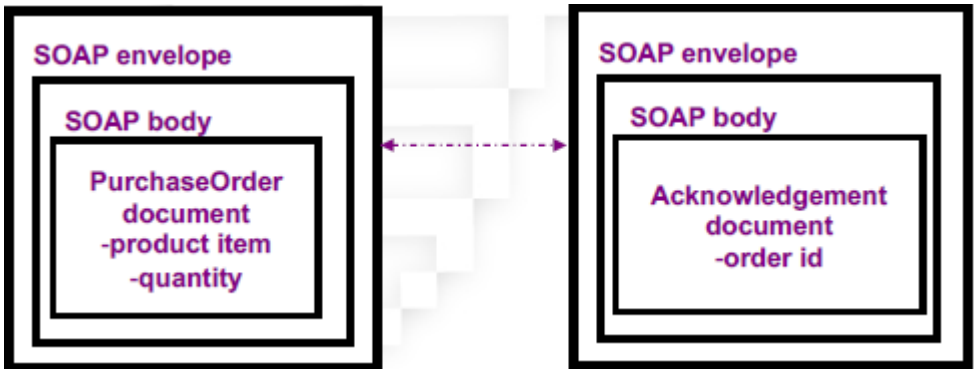
Example of RPC-style SOAP response message

```
<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>
```

5.2.2.2 Message-style

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.



An example:

```
<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>
```

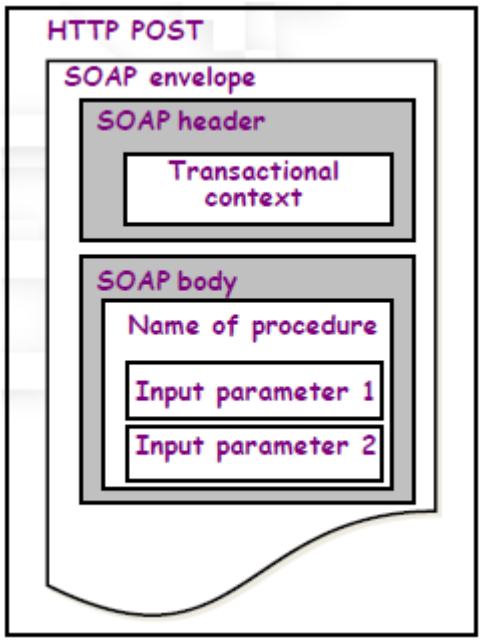
Example of document-style SOAP body

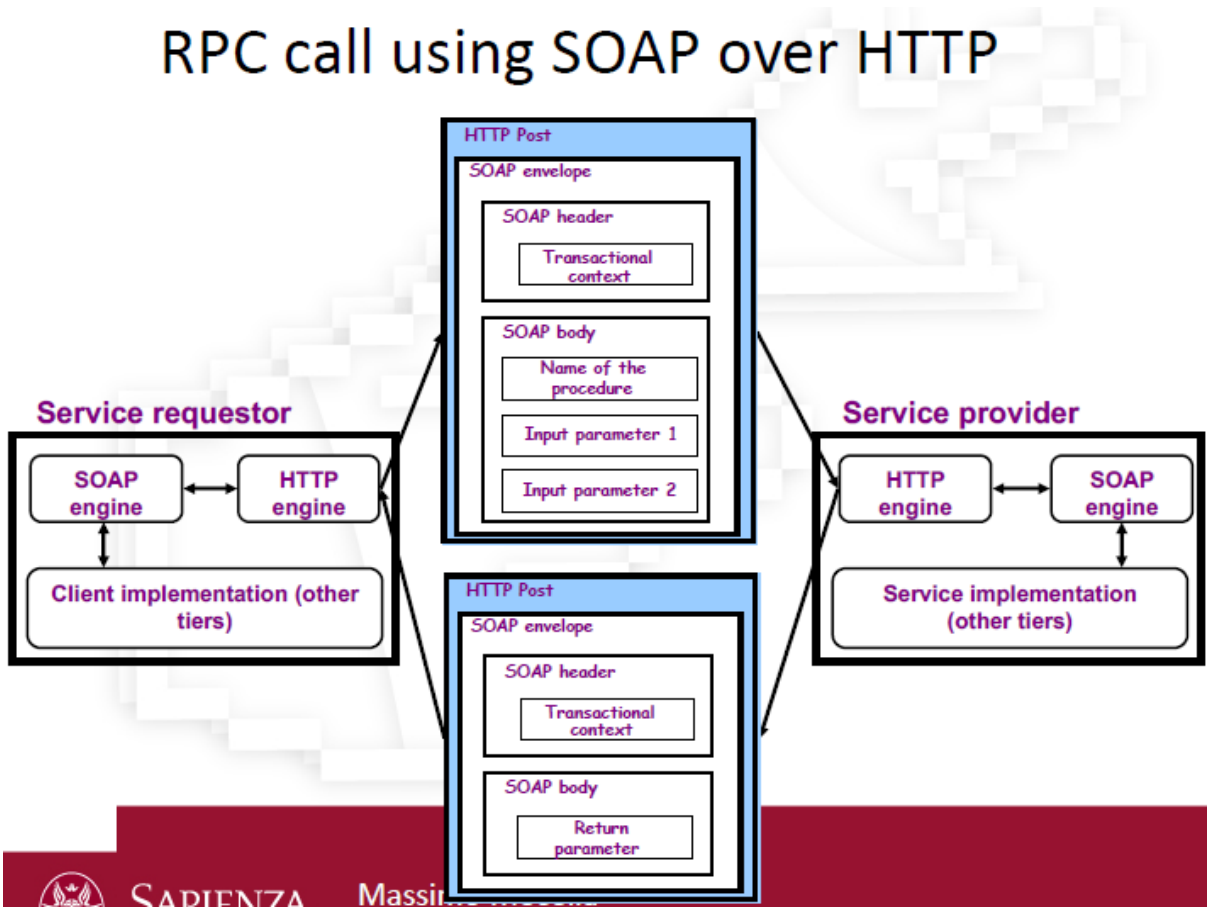
5.2.3 SOAP and HTTP

Typical bind for SOAP is HTTP, it is a description of how a SOAP message has to be.

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.

Uses the same error and status code as HTTP.





5.2.4 Advantages and disadvantages

Advantages:

- Simplicity, portability, firewall friendliness, open standards, interoperability, universal acceptance.

Disadvantages:

- Too reliance on HTTP, statelessness, serialization by value and not by reference.

5.3 Service Description & WSDL

Web services must be defined in a consistent manner to be disvoeed and used by ohter services and apps.

5.3.1 Service Description

Service description reduces the aount of required common understanding and custom programming and inegration: describe the operations, specifies the format and transport protocol, describe the payload data.

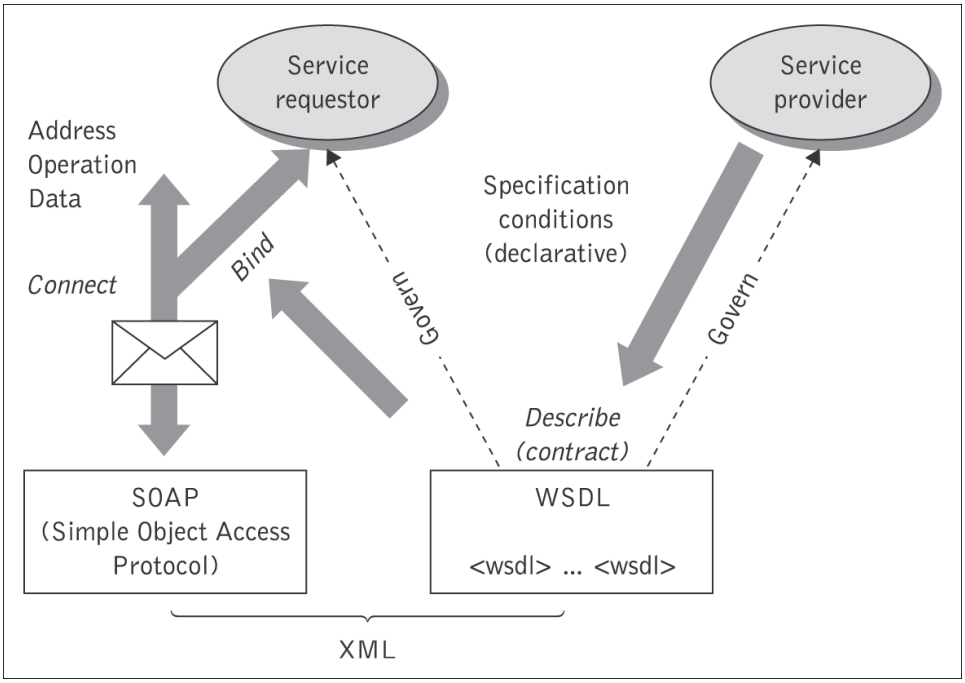
SD + SOAP isolates all technical details.

5.3.2 WSDL

The WSDL is the XML-based service representation language (platform and language independent) used to describe the details of the complete interfaces.

WSDL represent a *contract* between the service requestor and the service provider.

WSDL is used to describe: what a service does, where it resides, how to invoke it



5.3.2.1 Structure of WSDL documents

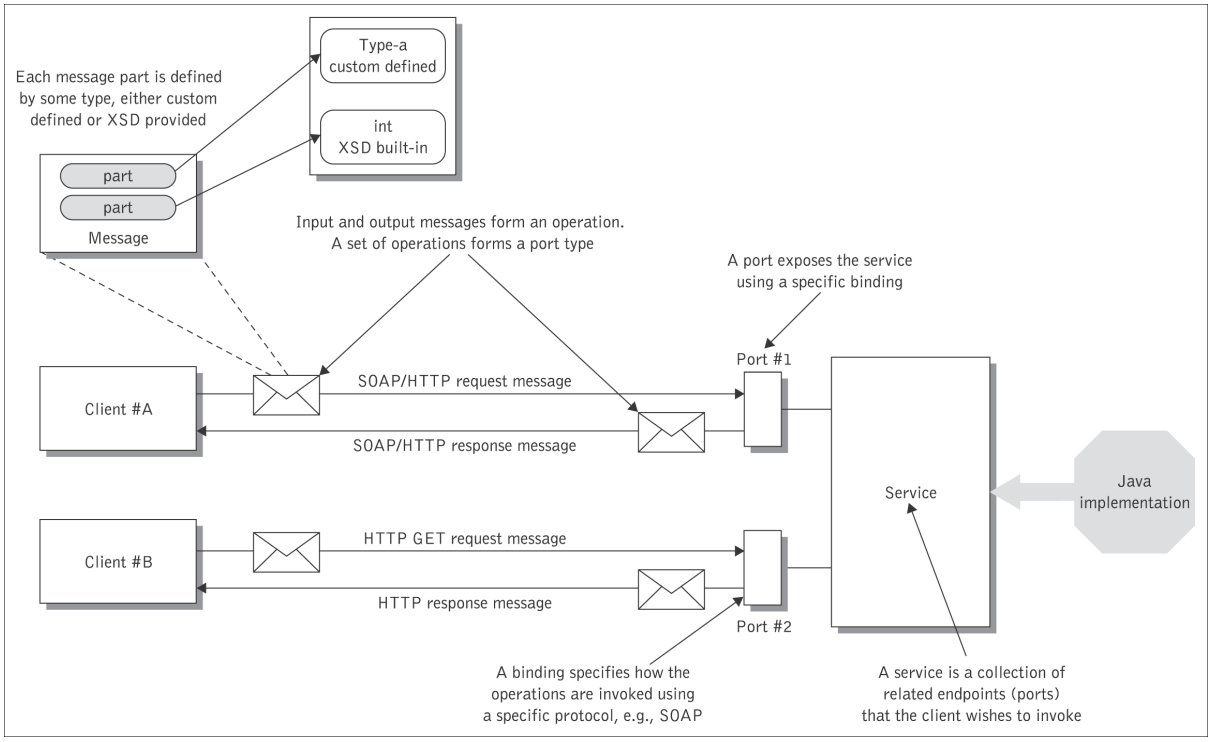
2 sections:

1) service-interface definition: describes the interface structure (all the operation supported, parameters, types)

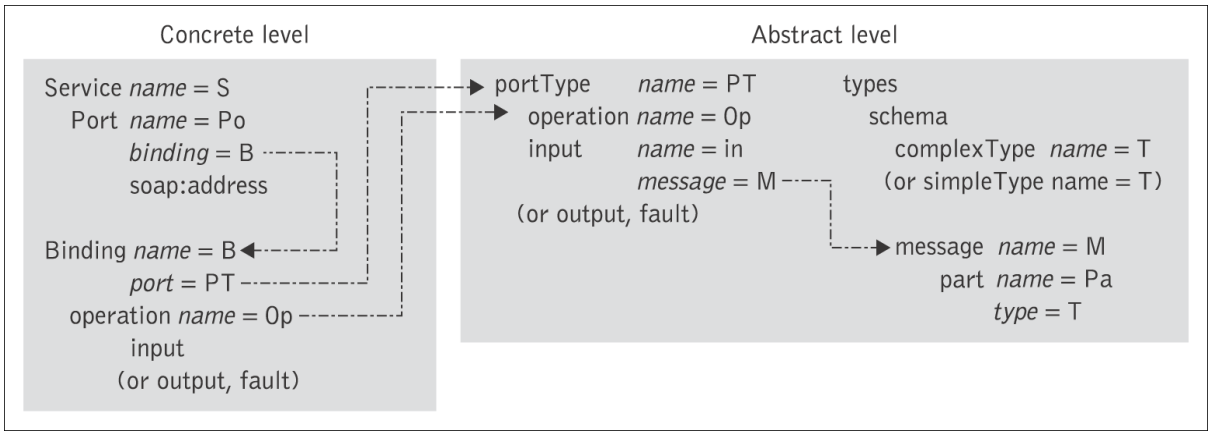
2) service implementation part: binds the interface to a network address, to a protocol and to data structures

This make it reusable; these two part contain sufficient information to describe how to invoke and interact with the web service.

5.3.2.2 Elements of WSDL

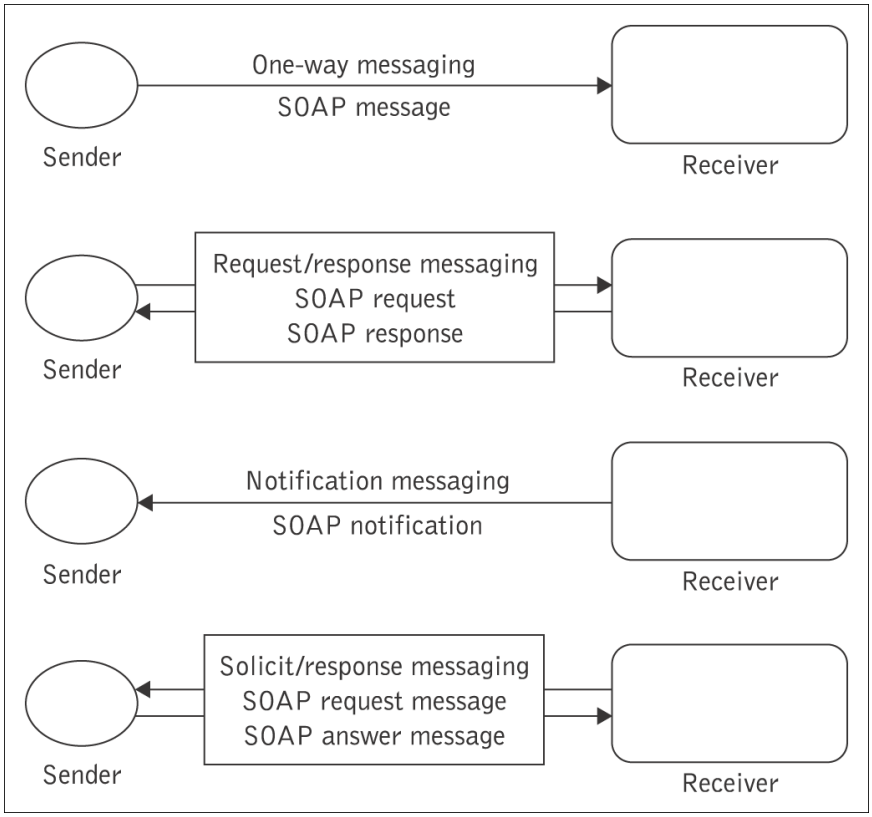


5.3.2.3 Connecting the abstract and concrete levels of a Web Service

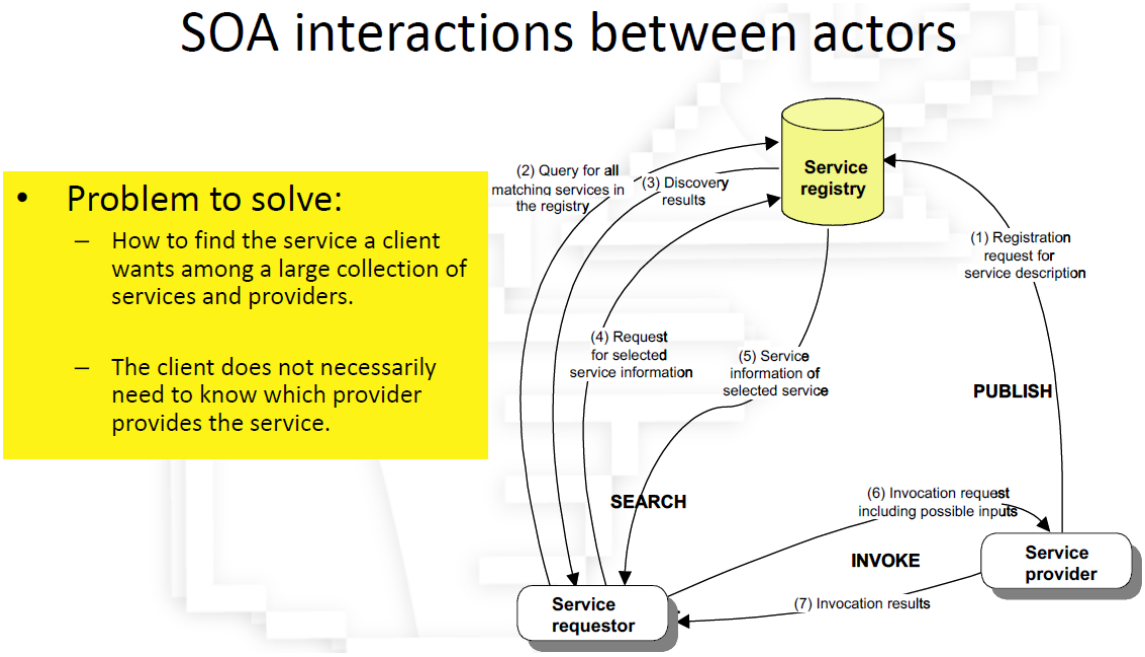


5.3.2.4 WSDL Message Exchange Patterns

WSDL interfaces support four common types of operations that represent possible combinations of input and output messages.



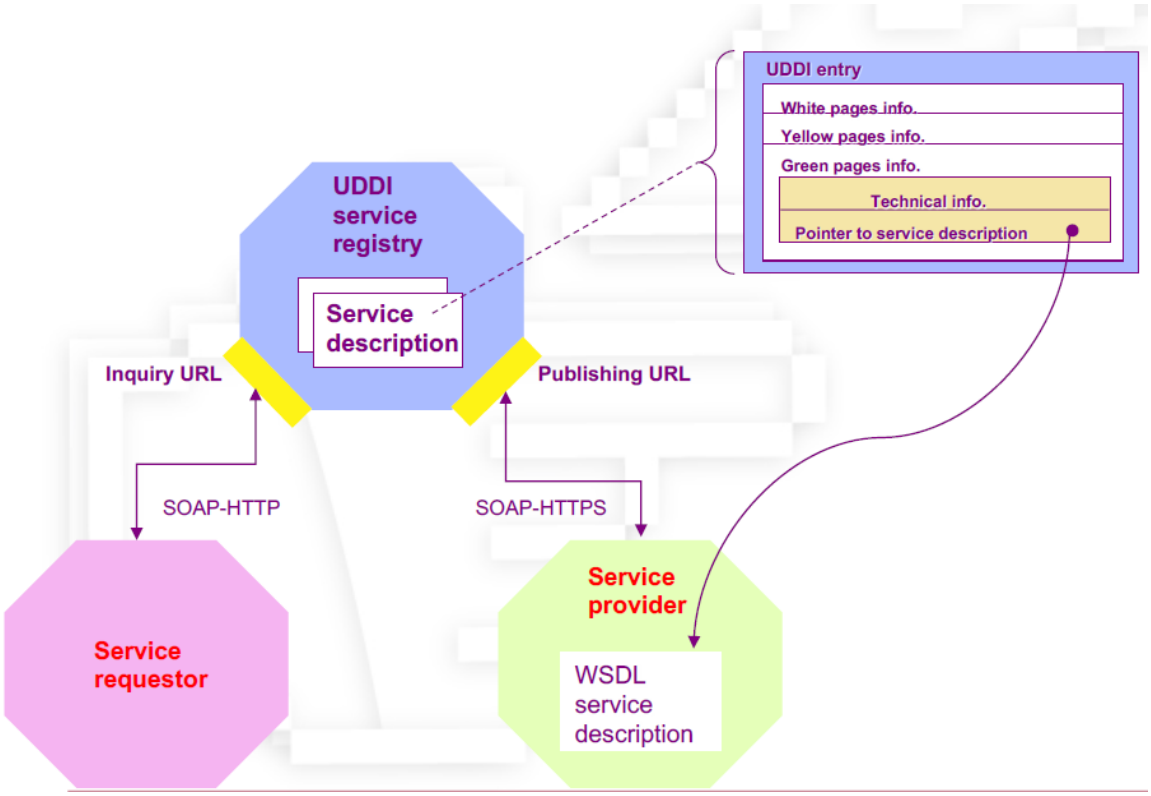
5.4 Registry and UDDI



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 to: describe its business and its services, discover other businesses, integrate (interoperate) with other businesses.
• UDDI consists of three components:
– “white pages” (address, contact, and other key points of contact);
– “yellow pages” classification info.
– “green pages”, the technical capabilities and information about services.

5.4.1 UDDI and WSDL



5.4.2 Web Services' Pitfalls

- Performance issues
- lack of appropriate support
- lack of expressing business semantics
- achieving a widespread agreement and harmonization on a wide range of existing and emerging standards.

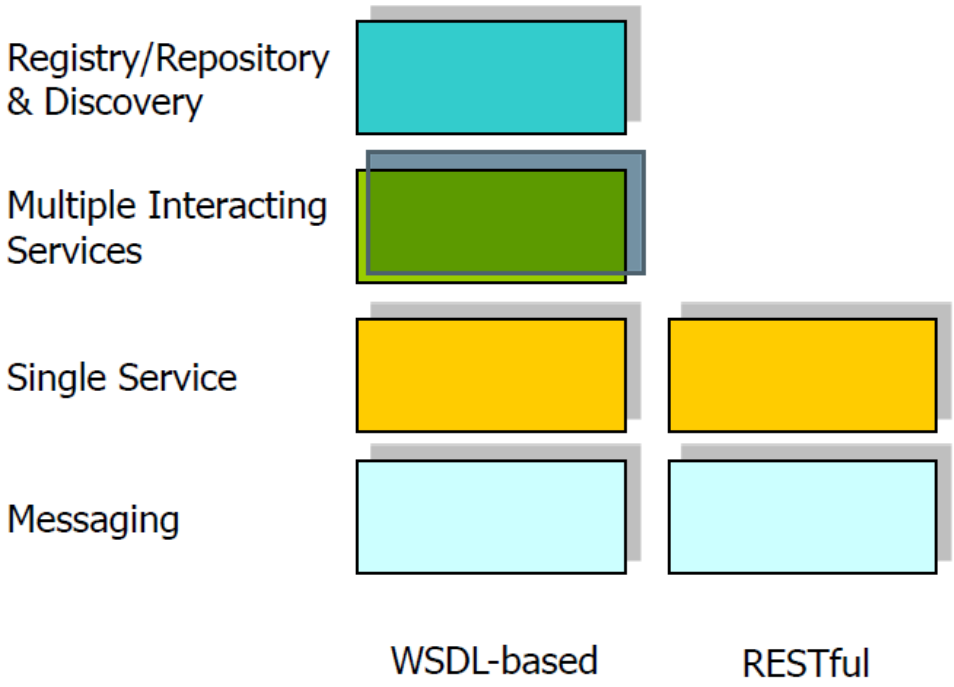
5.4.3 Services Mash-up

Combined Data from more sources into a single integrated tool

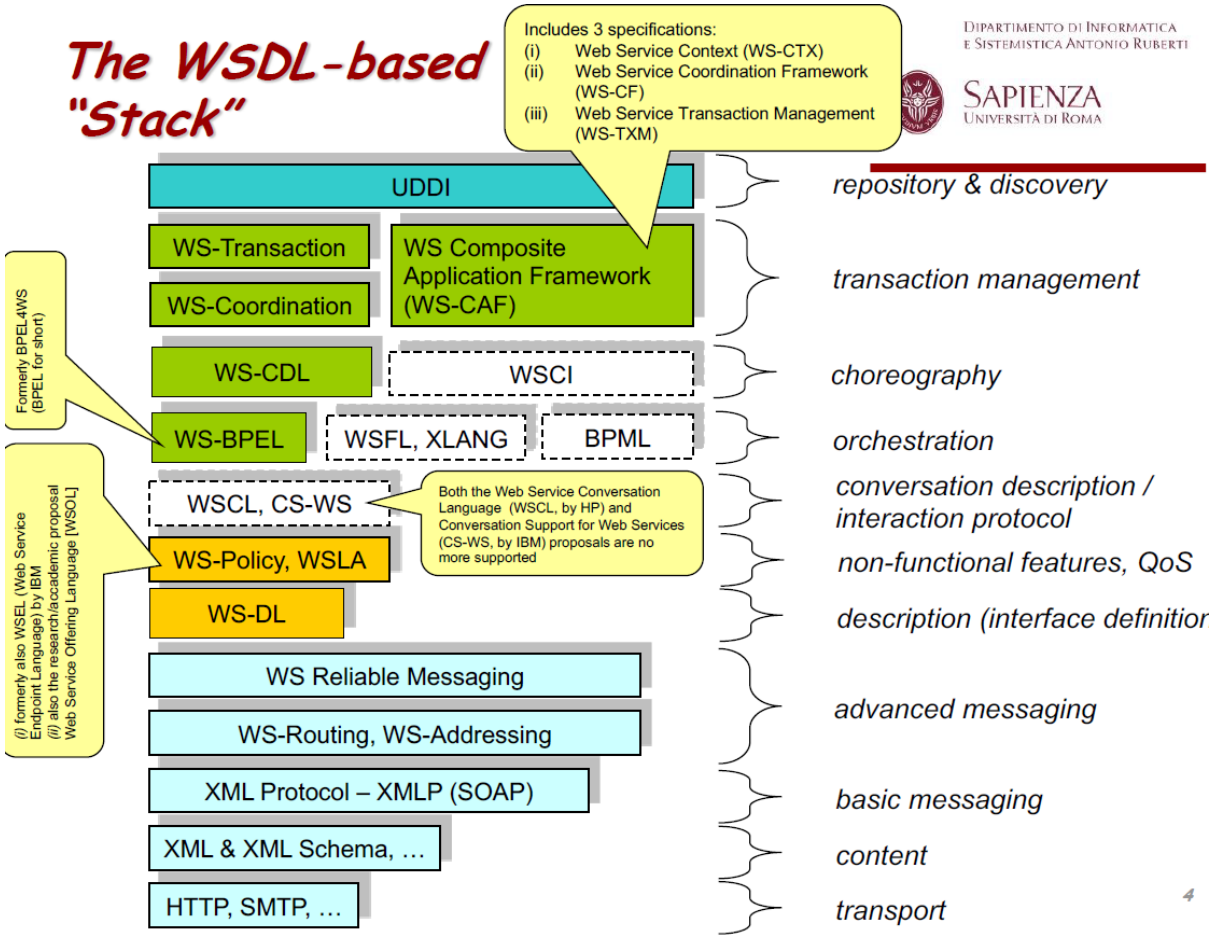
Bottom-up, developers-driven approach

Based on various technologies: Web services, SOAP, RESTful, Atom/RSS

5.4.4 The stacks of service technologies



5.4.5 The WSDL-based "Stack"



5.5 RESTful

REST is the architectural style that explains the quality attribute of WWW seen as an open, distributed and decentralized hypermedia application; it includes the design constraints which have been followed to define the HTTP protocol. As the Web became widespread, TCP/IP port 80 started to be left open by default on most internet firewalls, making it possible to use the HTTP protocol as a universal mean for tunnelling messages in business integration scenarios.

Refers to simple app interfaces over HTTP without e.g. SOAP

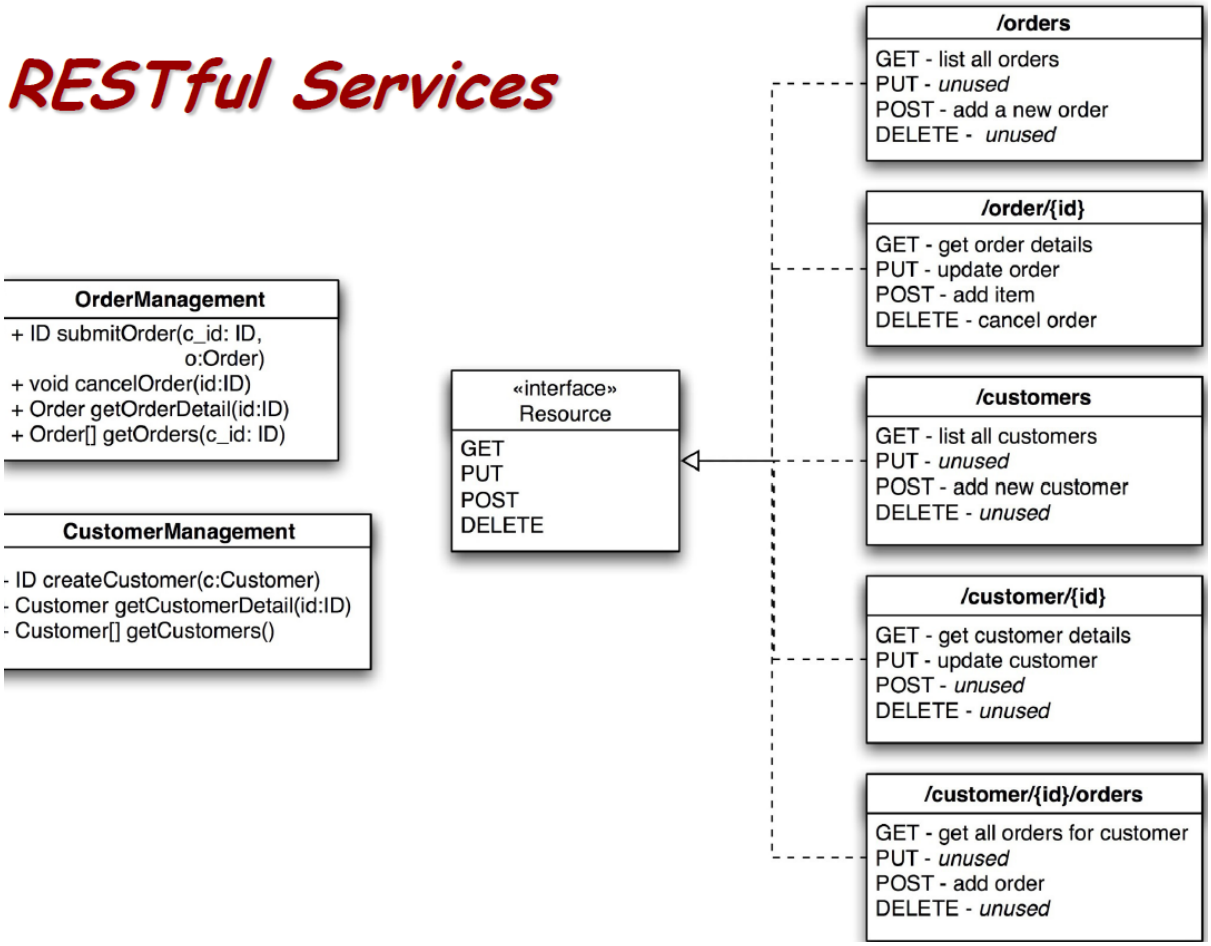
require an architectural style to make sense of them (the REST one).

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

Rest is a kind of RPC but the methods have been defined in advance



REST is incompatible with "end-point" RPC

- REST does the former
- End-point RPC does the latter

5.5.1 REST principles

5.5.1.1 Design constraints

- Addressability: all resources should be given a unique and stable identifier. These identifiers are globally meaningful, so no central authority is involved.

- Uniform Interface: All resources interact through a uniform interface that provides a small, generic and functionally sufficient set of methods to support interactions.
- Stateless interactions: Services don't establish any permanent session; requests to a resource are independent from each other.
- Self-describing messages (metadata): Services interact by exchanging request and response messages, which contain both the data and the corresponding meta-data. Messages also should contain explicit meta-data about the representation so that srvices do not need to assume any kind of out-of-band agreement on how the representation should be parsed, processed and understood.
- Hypermedia: is about embedding references to related resources inside resource representations or in the corresponding meta-data. Clients can thus discover ids of related resources when processing representations and choose to follow the link as they navigate the graph built out of relationships between resources.

See examples on the slides SE_05

5.5.1.2 Maturity of RESTful Services

Only services that are fully mature can be called RESTful

Levels:

0 - HTTP as a tunnel: simply exchange XML documents over HTTP POST request and responses, effectively following some kind of XML-RPC protocol; even if such services are not making use of SOAP messages, they are not really making full use of the HTTP protocol according to the REST constraints either. Since all messages go to the same endpoint URL, a service can distinguish between different operations only by parsing such information out of the XML.

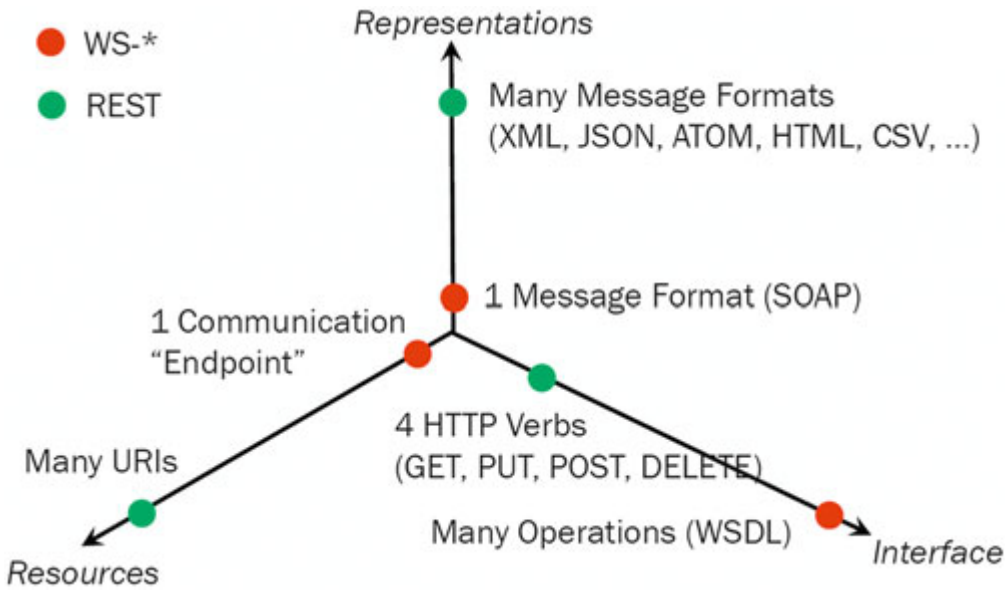
1 - Resources: Services make use of multiple identifiers to distinguish different resources. Each interaction is addressed to a specific resource, which can however still be misused to identify different operations or methods to be performed on the payload.

2 - HTTP verbs: Not only clients can perform a method on a resource, in addition to POSTing to it, but also do so in compliance with the semantics of such methods.

Since we can assume that the HTTP methods are used according to their standard semantics, we can use their proprieties to optimize the system.

3 - Hypermedia: fully mature: in addition to exposing multiple addressable resource which share the same uniform interface also make use of hypermedia to model relationship between resources. This is achieved by embedding so-called hypermedia controls; Hypermedia controls will be typed according to the semantics of the relationship and contain all information necessary for a client to formulate a request to a related resource; Key to achieve this level of maturity is the choice of media types which support hypermedia controls.

5.5.1.3 REST vs. WS-*



Cotinue on the additional document SE_05_RESTful if necessary.

5.6 e-Services

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;

e-Services and Web services

e-Service is the provision of a service via the Internet (the prefix “e” standing for “electronic”);

Web service is a software component available on the Web; 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.