Web Service
An Introduction

Lin Zuoquan

Information Science Department

Peking University

lz@is.pku.edu.cn

http://www.is.pku.edu.cn/~lz/teaching/stm/saswws.html

Courtesy some graphic slides from online

Outline

- Web Services
- WS Standards
- WS Composition
- WS Development
- WS Application

Web Services

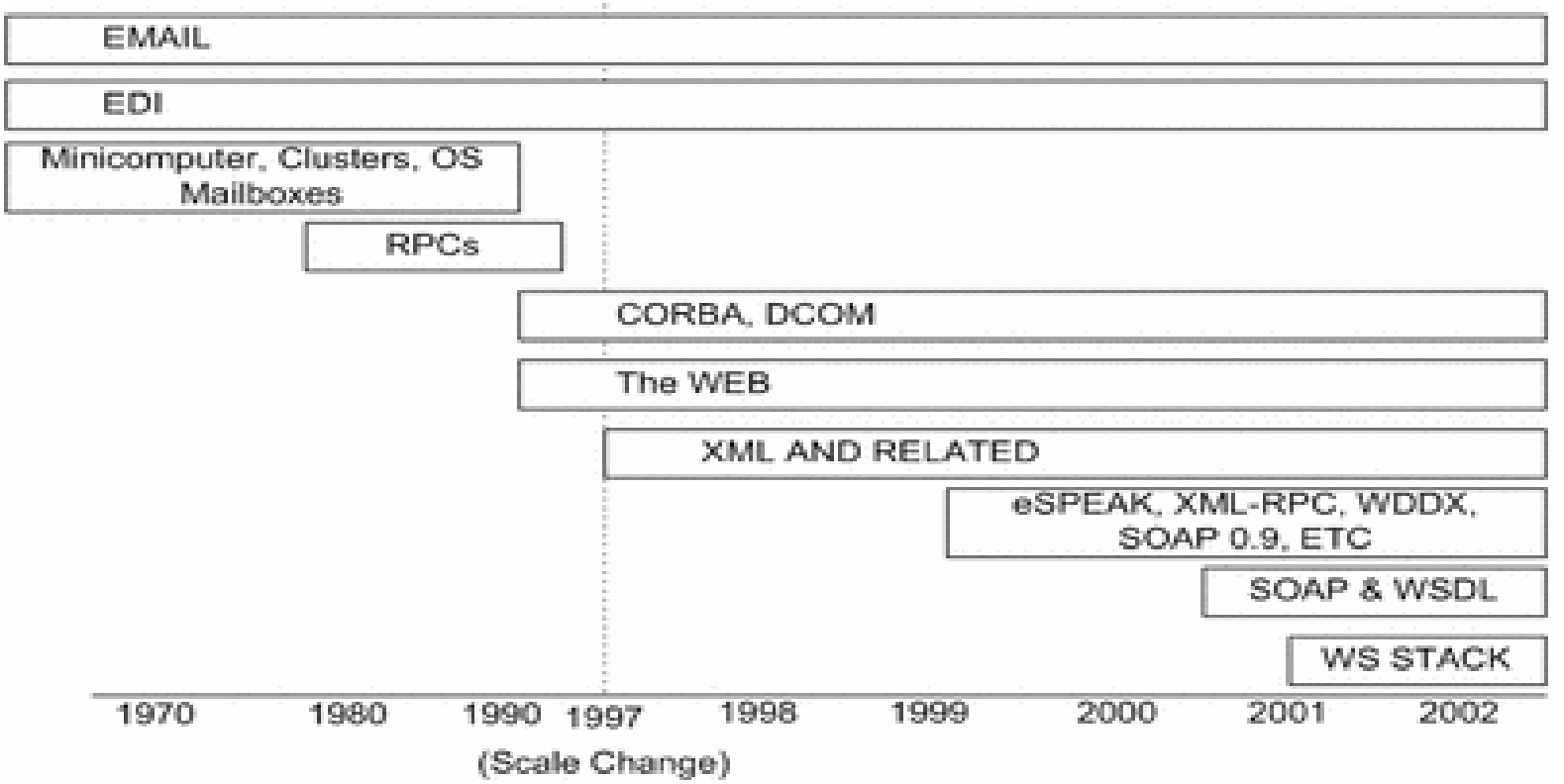
- Where WS From
- What're WS
- Why WS
- How WS Work

Where WS From

Distributed computing

- EDI (Electronic Data Interchange)
- DCE (Distributed Computing Environment, RPC)
- CORBA (Common Object Request Broker Architecture)/DCOM (Distributed Component Object Model)
- JAVA (RMI)
- The Web
- XML related
- WS Stacks

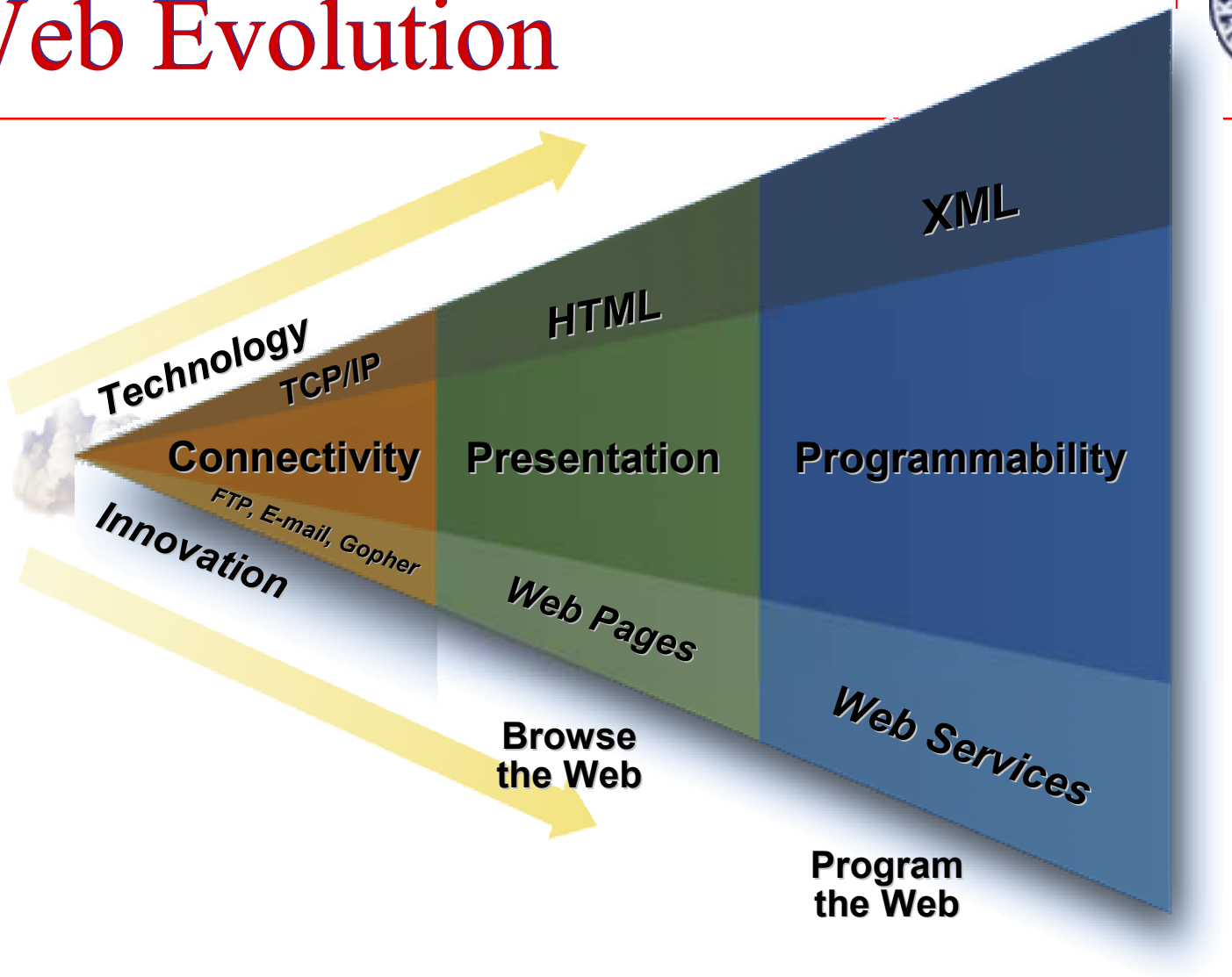
History


EMAIL
EDI
Minicomputer, Clusters, OS Mailboxes
RPCs
CORBA, DCOM
The WEB
XML AND RELATED
eSPEAK, XML-RPC, WDDX, SOAP 0.9, ETC
SOAP & WSDL
WS STACK
1970
1980
1990
1997
1998
1999
2000
2001
2002
(Scale Change)

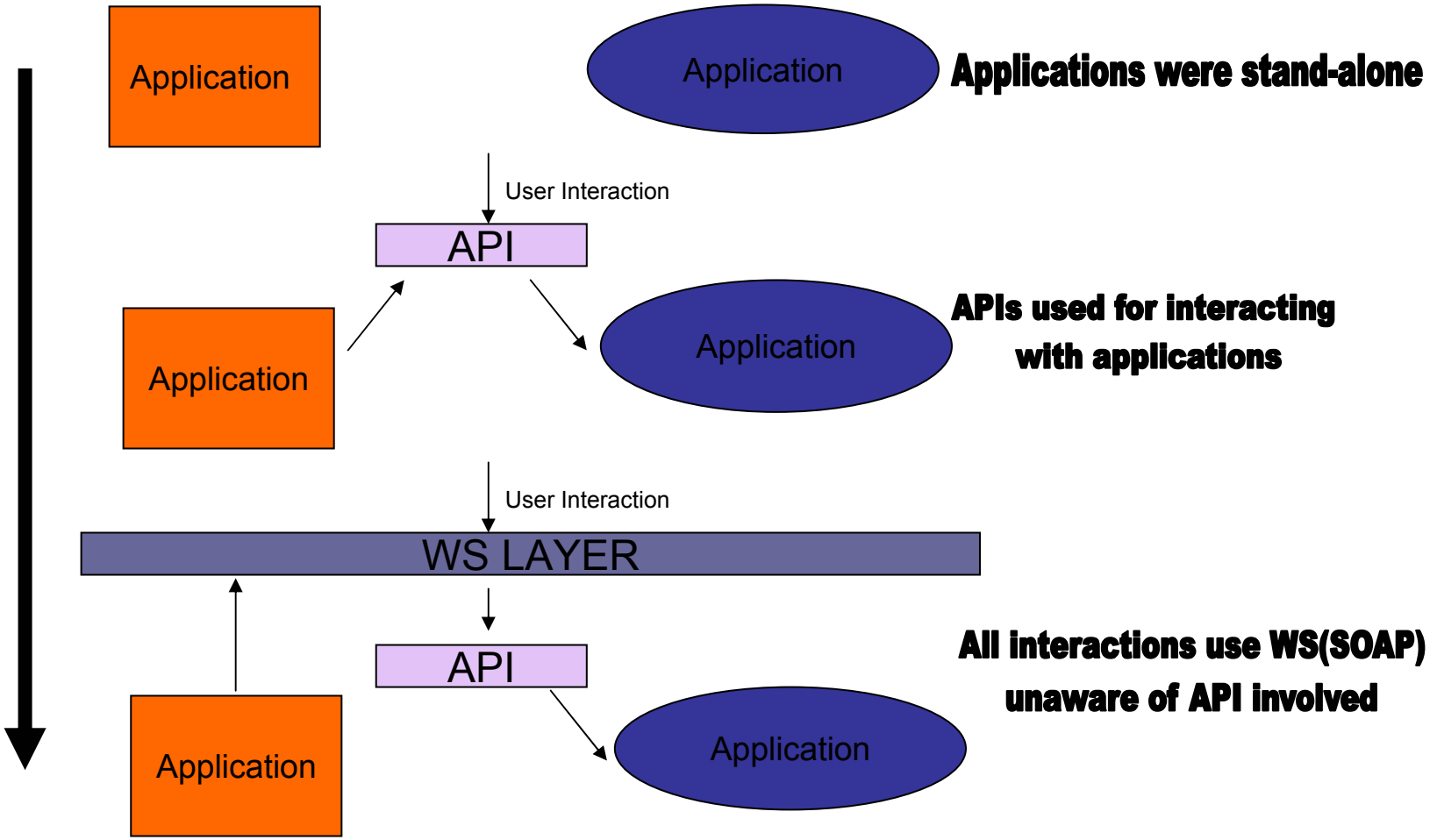
The Web

- The Web
  - information sharing
  - B2C/B2B
  - search, mail, chat, RPC and what not?
- What is missing
  - automated interactions
  - distributed computing platform

Web Evolution



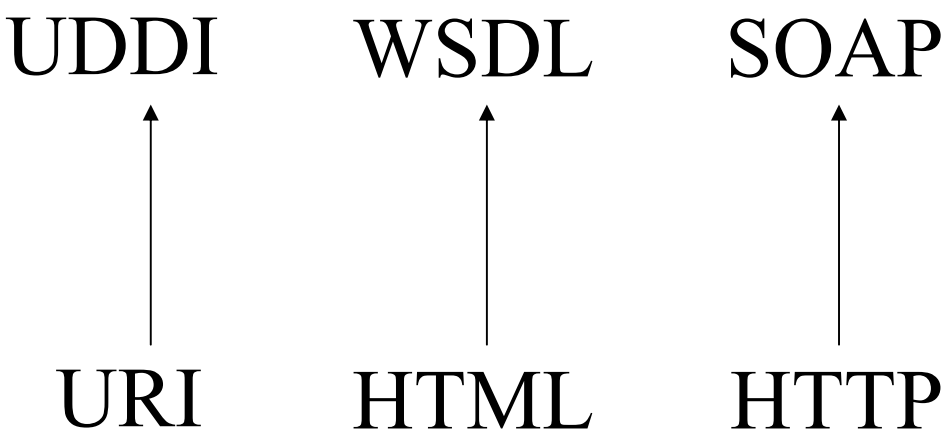
Evolution (contd)



Evolution (contd)

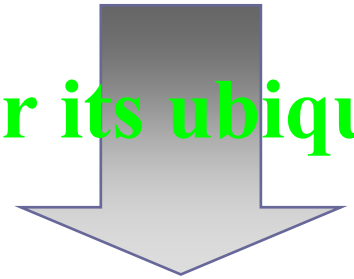
- Ongoing Web
  - automating interactions
  - unifying distributed computing platform
- Goals
  - enable internet scale dynamic binding
  - efficiently support both open web and more constrained environments
  - universal availability and interoperability

State of the Art



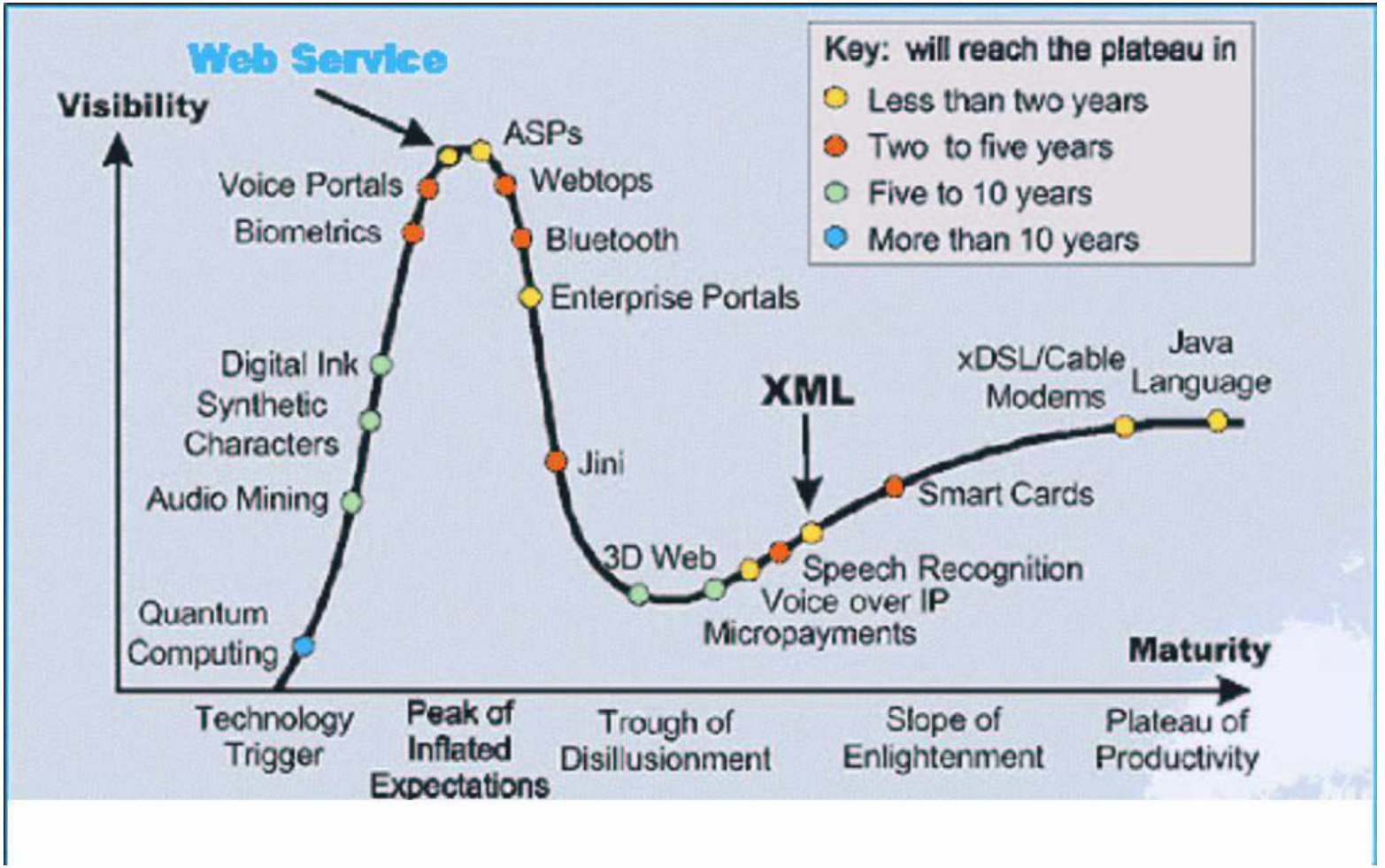
The Network

Web services must be network-accessible



- HTTP is the standard network protocol for internet-available Web Services
- Others network protocols are SMTP, FTP and for intranet domain: MQSeries, CORBA

Good Future (optimism)



What're WS

- Definition from W3C

  "Web Service is a software application identified by a ***URI***, whose interfaces and bindings are capable of being ***defined***, ***described***, and ***discovered*** by XML artifacts and which supports direct ***interactions*** with other software applications using XML-based messages via internet-based protocols."

WS Definition(contd)

- Definition from IBM
  - Web services are self contained, self-describing, modular applications that can be published, located, and invoked across the Web. (Tidwell, IBM, 2001)
  - Web service applications are encapsulated, loosely coupled Web “components” that can bind dynamically to each other. (Curbera, IBM, 2001)
  - A Web service is an interface that describes a collection of operations that are network-accessibile through standardized XML messaging. (Kreger, IBM, 2001)

WS Definition(contd)

- Web services = Web of Services (software)
- A2A (applications to applications/agents to agents) over the web
- Every component that
  - works in a network
  - is modular
  - is self-descriptive
  - provides services independent of platform and application
  - conforms to an open set of standards
  - follows a common structure for description and invocation

Why WS

- Interoperability
  - XML based technology (OS agnostic)
  - any WS can interact with any other WS
  - platform/language independent (Web protocols)
- Ubiquity
  - any device which supports HTTP + XML can host and access WS
- Effortless
  - easily understood (entry) + free toolkits
- Industry Support.
  - major vendors support surrounding technology

WSs are good by providing ….

INTEGRATION

- In many organizations the data and logic of one application are basically useless to other applications. So the application and its data are isolated from other applications and there is an operational "GAP"
- WS are better at sharing data and functions. They reduce the "GAP" the isolated systems can talk to each other
- WS allow applications to be integrated more rapidly, easily and less expensively because integration is based on messages centered more on service semantics than on network protocol semantics

WSs are good by providing ….

ACCESS

- WS are especially good at providing access through different interfaces
- A WS can have a dedicated client application, but it can also be readily accessed through browsers, wireless devices, voice-activated interfaces, etc.
- Adding new access methods is much simpler than with a traditional application

WSs are good by providing ….

FLIEXIBILITY

- One of the most important innovations in WS is "machine-to-machine communications"
- This means that a WS can ask another WS to do something, and that WS can ask another WS to do something, and so on
- In the future, many WS will really just be aggregations of other WS
- Create loosely coupled applications
  - changing implementation of a function does not require change in invoking function
- Create encapsulated applications
  - implementation is hidden from outside

WSs are good by providing ….

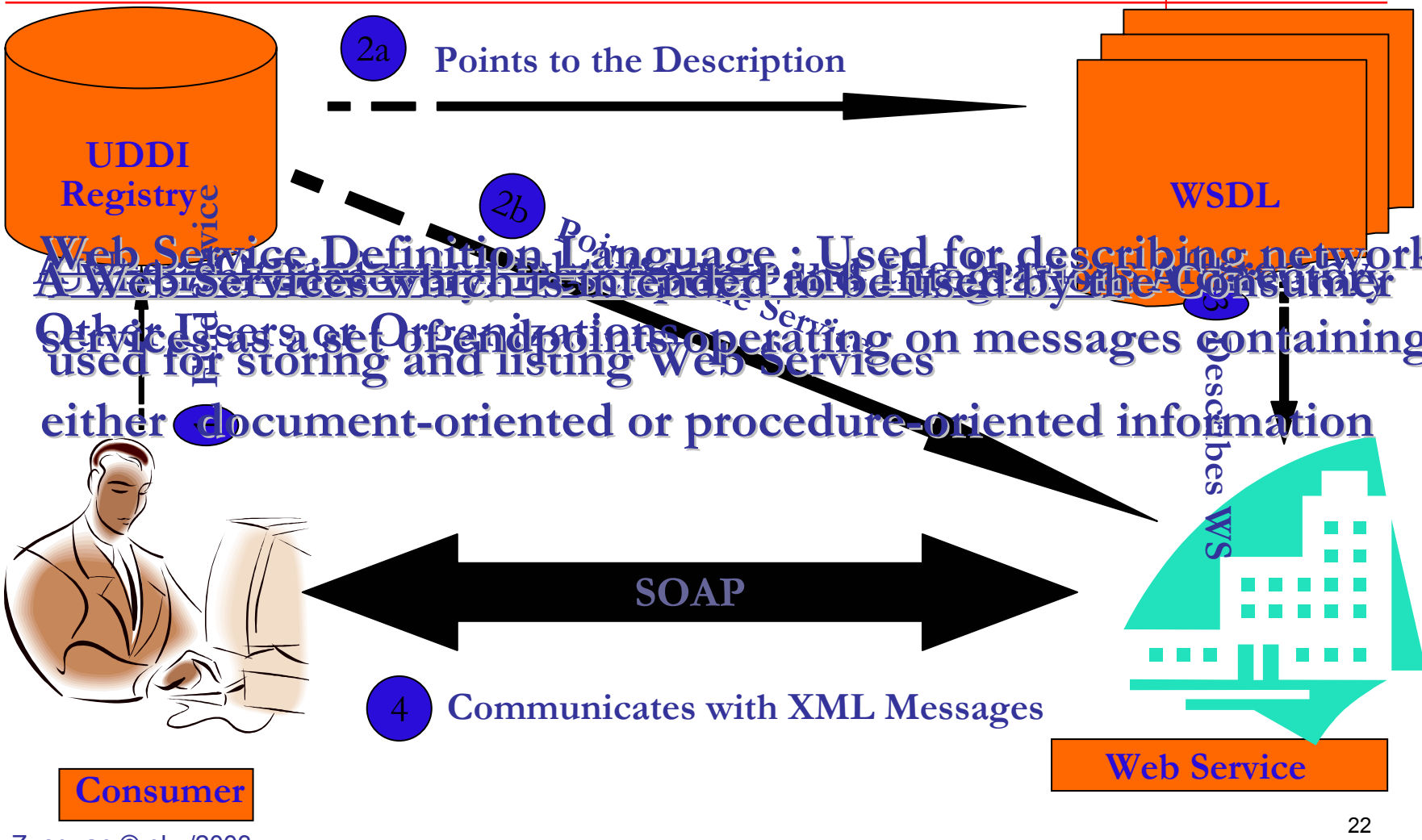
REUSABILITY

- Allow reuse of software components
  - reusable datas
- Address middleware issues over web
  - without much complexity
  - FYI: Middleware is a high level software layer that provides a standardized interface to collection of distributed objects
  - FYI: What's wrong with CORBA/RMI/COM ?
    - don't talk to each other (some implementations do)
    - tightly coupled with application
    - different data representation, vendors and dialects
    - RMI: Java-to-Java only technology
    - CORBA: uses proprietary IIOP (typically blocked by Firewall) and interoperability of ORB implementations is not guaranteed
    - COM/DCOM: Microsoft only solution

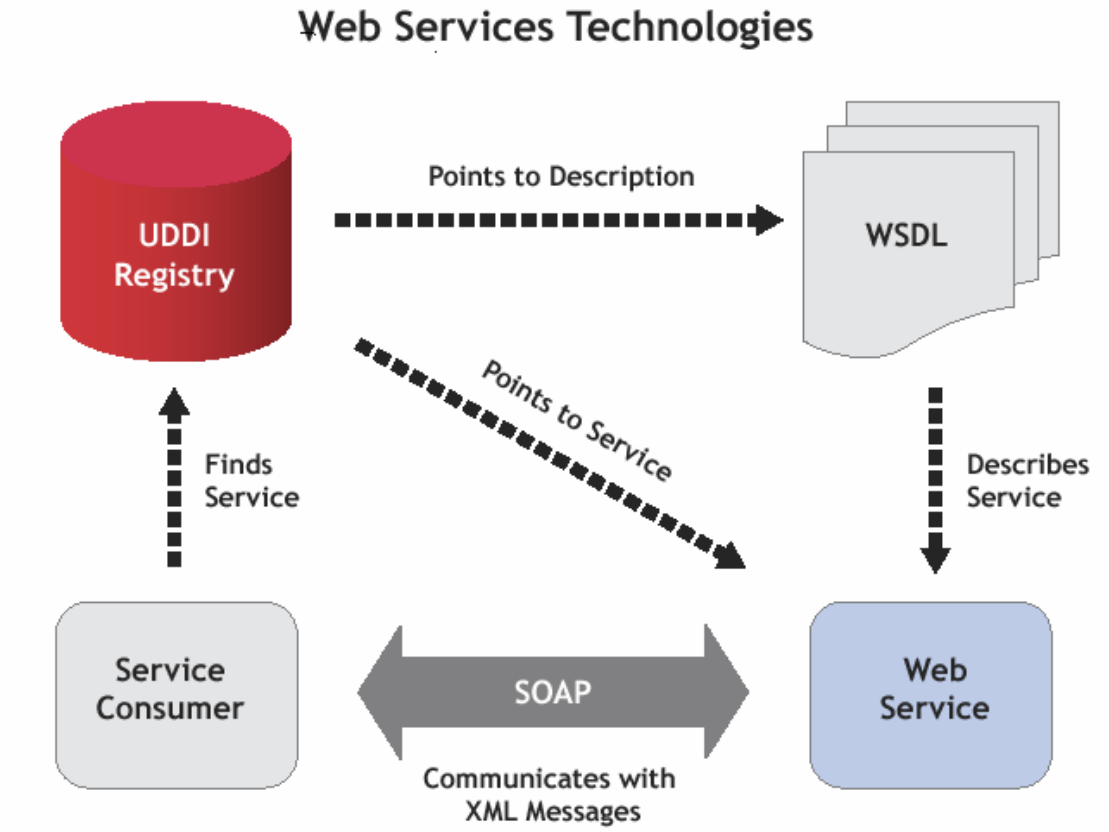
WS vs CORBA

Feature	CORBA	Web Services
Data Model	Object Model	SOAP Message exchange model
Client Server Coupling	Tight Coupling	Loose Coupling
Parameter Passing	Pass by reference/value	Pass by value only
Type Checking	1.Static + Runtime type checking (Regular) 2. Runtime type checking only (DII)	RunTime type checking only
State	Stateful	1. Stateless, Uncorrelated (Web Services) 2. Stateful (Web Process)
Firewall Traversal	Work in Progress	Uses HTTP port 80
Service Discovery	CORBA naming/trading Service	UDDI
Communication Mode	1-way, 2-way sync 2-way async	2-way sync (Web Services) 1-way, 2-way sync, 2-way async (Web Process)

How WS work

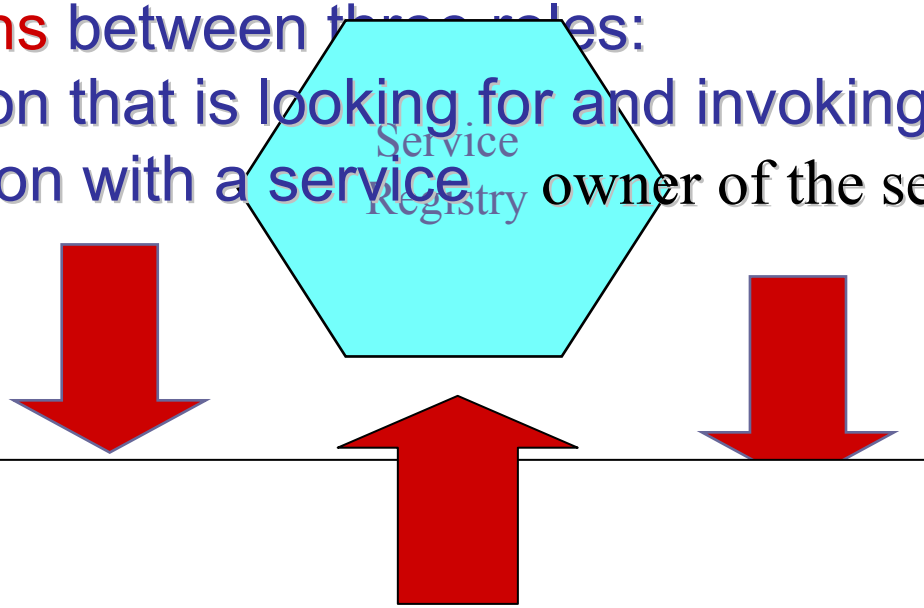

2a
Points to the Description
UDDI
Registry
WSDL
2b
Web Service Definition Language : Used for describing network
A WebServices which is intended to be used by the Consumer
Other Users or Organizations operating on messages containing
services as a set of endpoints
used for storing and listing Web Services
either document-oriented or procedure-oriented information
SOAP
4
Communicates with XML Messages
Consumer
Web Service

WS Model


Web Services Technologies
UDDI Registry
Points to Description
WSDL
Points to Service
Finds Service
Describes Service
Service Consumer
SOAP
Web Service
Communicates with XML Messages

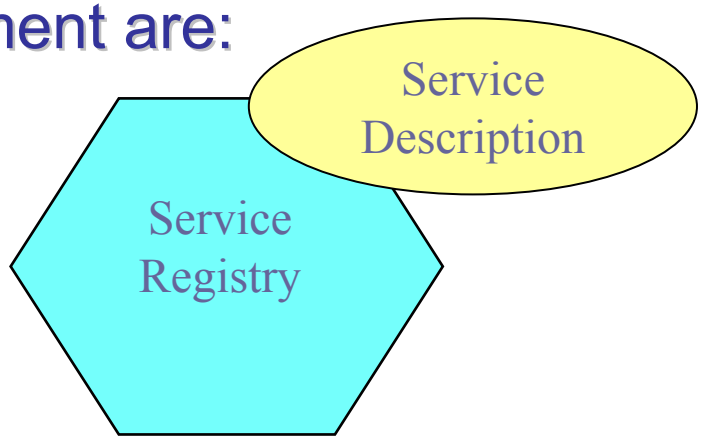
Model(contd)

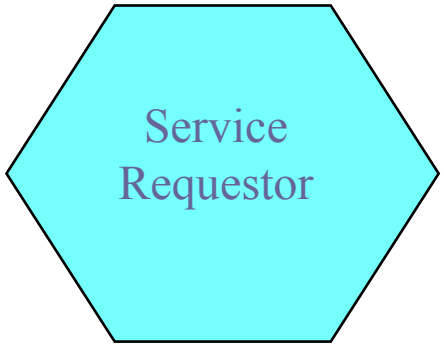
- The Web service architecture is based upon the interactions between three roles:
  application that is looking for and invoking an interaction with a service owner of the services

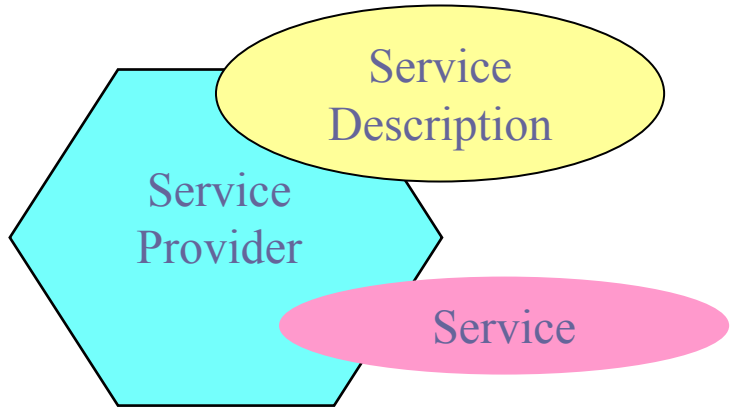


Model(contd)

- Others element are:

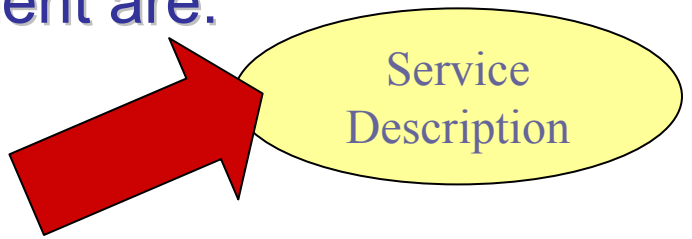




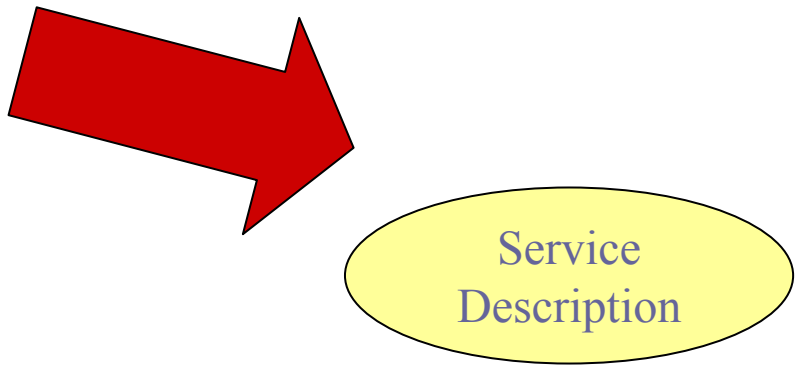


Model(contd)

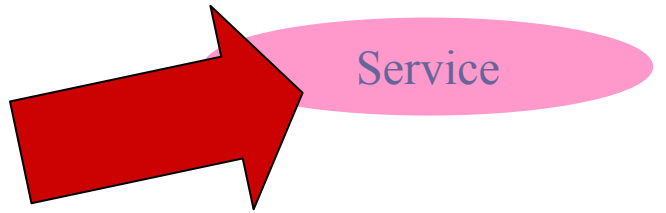
- Others element are:



Contains details of the interface and implementation of the service

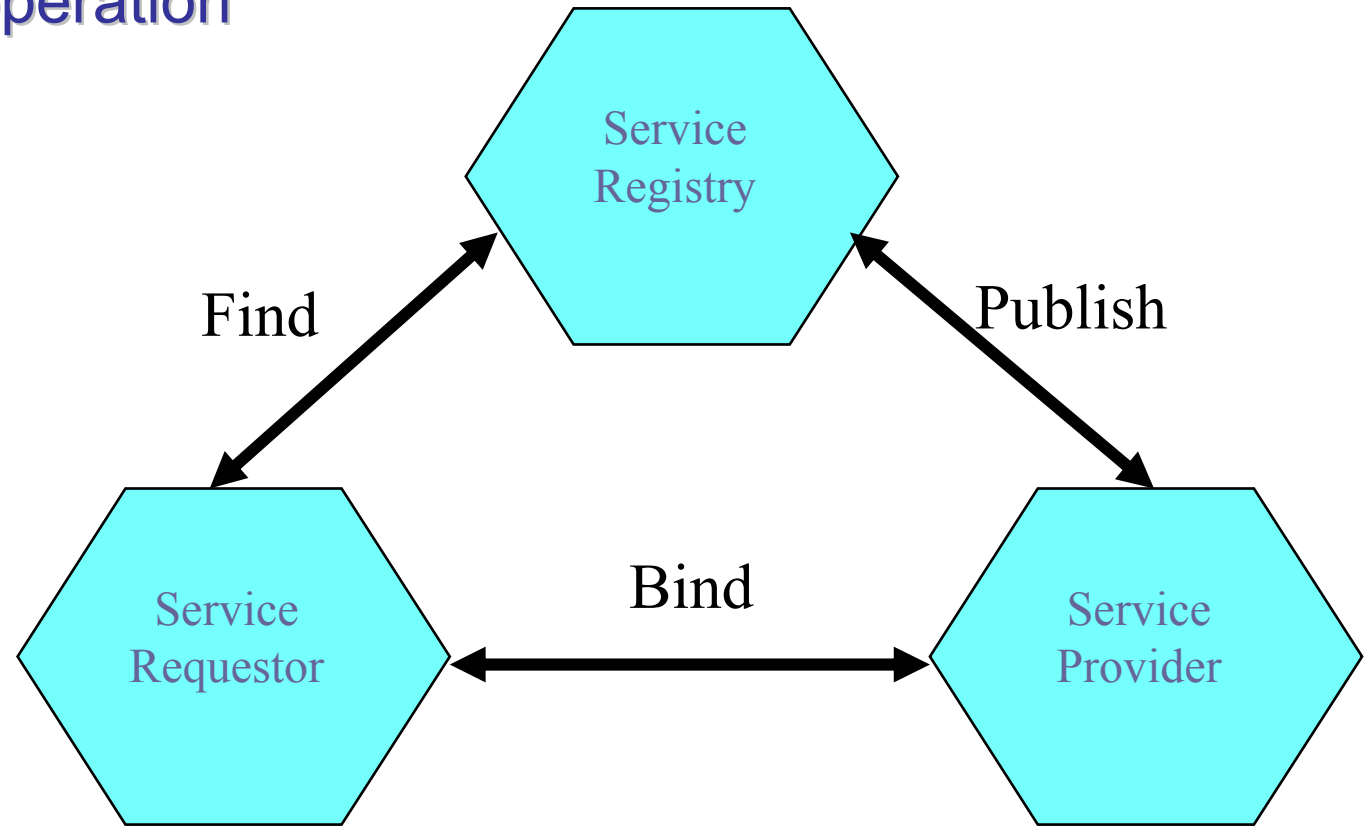


It is a software module deployed on the network-accessible platforms provided by the service provider



Model(contd)

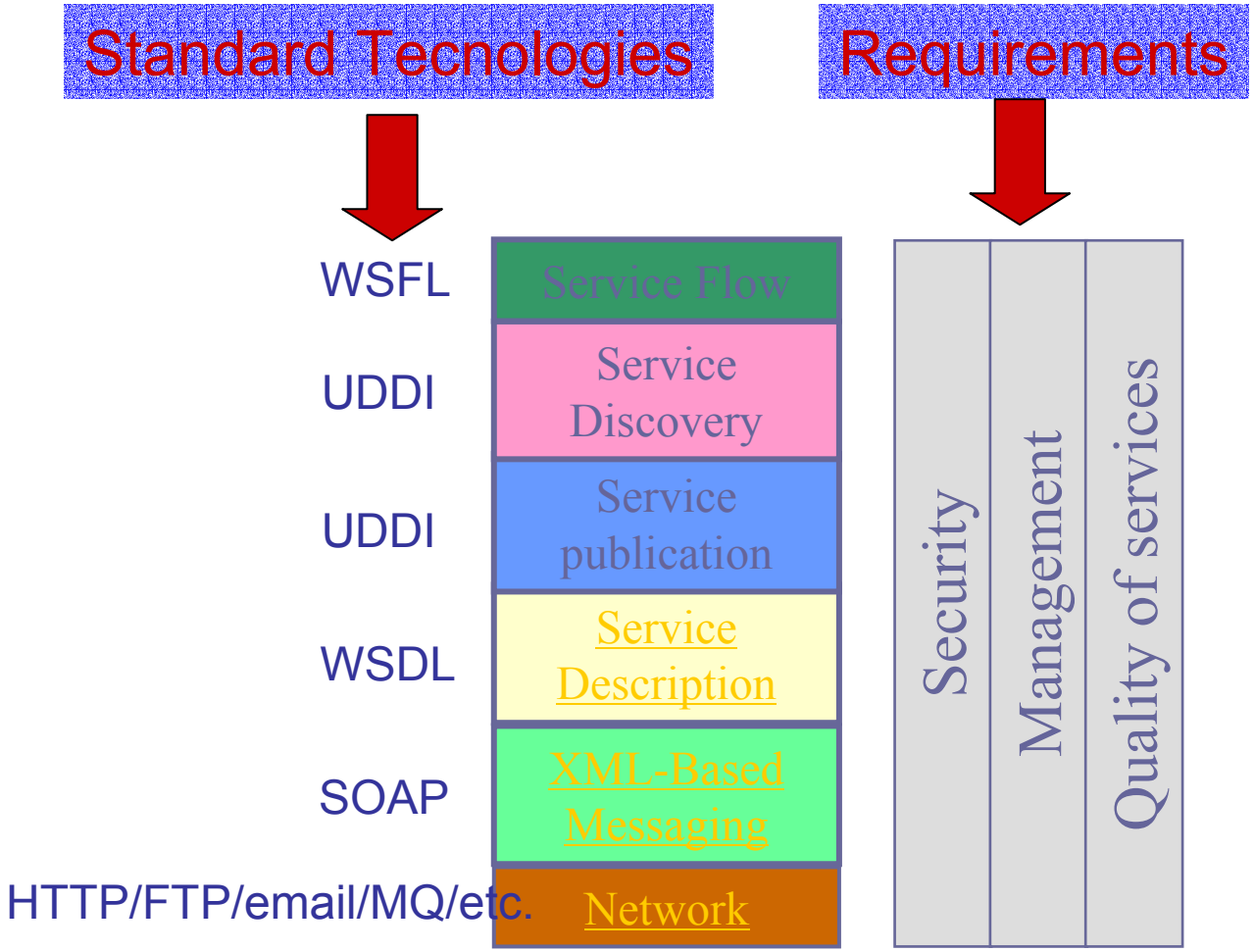
- The interactions involve the **publish**, **find** and **bind** operation



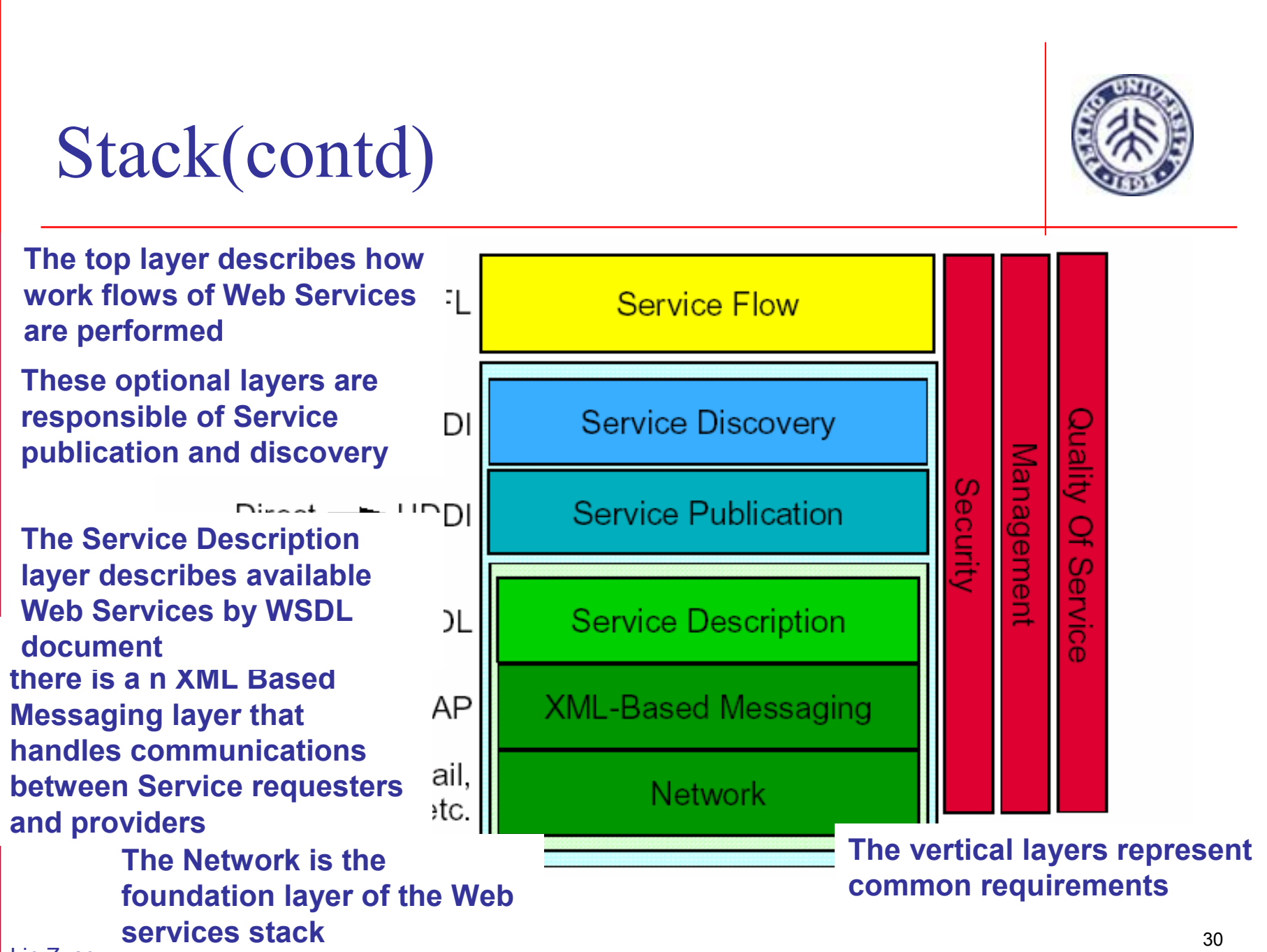
Model(contd)

- Components
  - Service Providers
  - Service Brokers
  - Service Requestors
- Operations
  - Publish / Unpublish
  - Find
  - Bind

WS Stack


Standard Tecnologies
Requirements
WSFL
UDDI
UDDI
WSDL
SOAP
HTTP/FTP/email/MQ/etc.
Service Flow
Service Discovery
Service publication
Service Description
XML-Based Messaging
Network
Security
Management
Quality of services

Stack(contd)



The top layer describes how work flows of Web Services are performed

These optional layers are responsible of Service publication and discovery

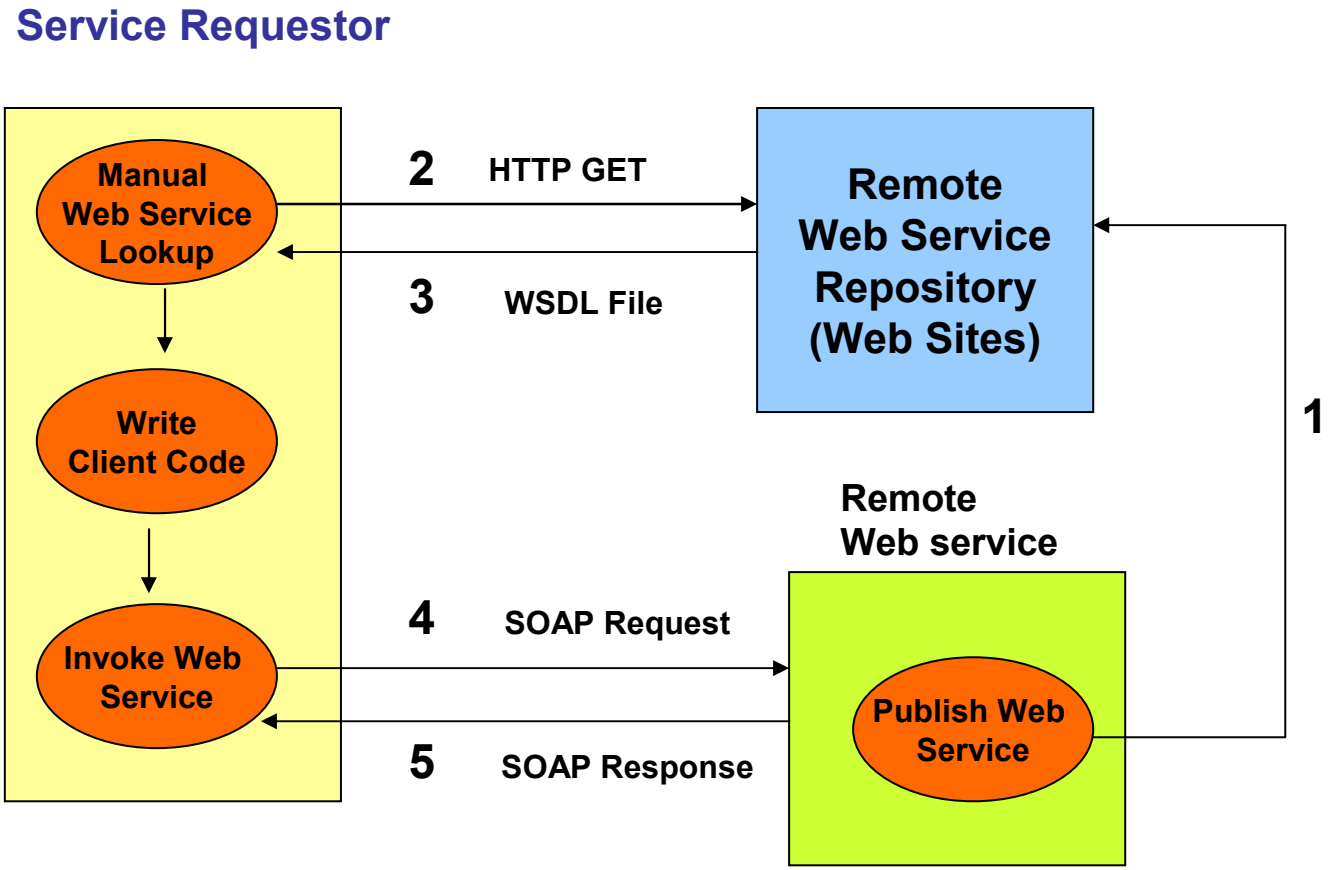
The Service Description layer describes available Web Services by WSDL document

there is a n XML Based Messaging layer that handles communications between Service requesters and providers

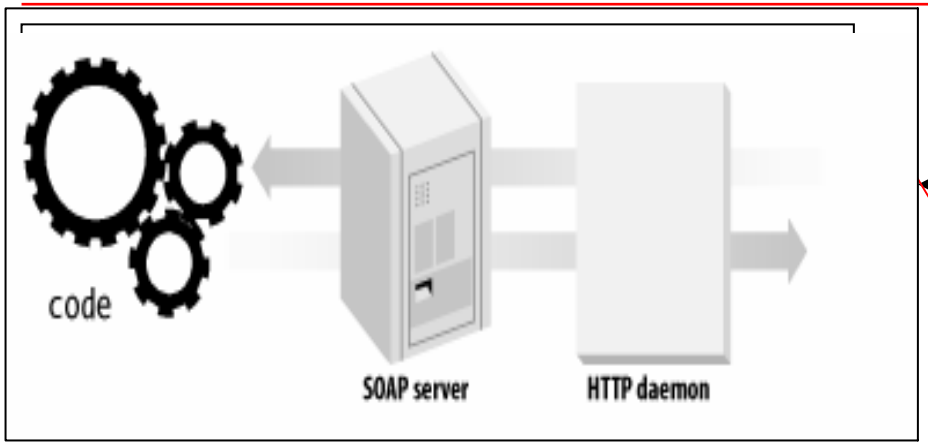
The Network is the foundation layer of the Web services stack

The vertical layers represent common requirements

Simple WS Invocation



Behind the scene



SOAP Messages

(http transport)

Requestor

SOAP Client

Endpoint

Web Service Provider

- Components required
  - software which needs to be exposed as a web service
  - SOAP Server (Apache Axis, SOAP::Lite, etc.)
  - HTTP Server (if HTTP is used as the transport level protocol)
  - SOAP Client (Apache Axis, SOAP::Lite etc.)
- Deployment: registering with SOAP server with details about the code to execute when requested
- Invocation: requesting for some service. Could be done with any SOAP client

WS Standards

- SOAP
- WSDL
- UDDI

The Standards

With the following enabling technologies, it is possible to build standards-based systems that are truly interoperable

- **SOAP**: Simple Object Access Protocol
  - SOAP is the **XML**-based protocol for sending requests and responses to and from web services (**for web services invocation)**
- **WSDL**: Web Services Description Language (new: DAML-S)
  - WSDL is an **XML**-based format for specifying the interface to a web service (**for web services description**)
- **UDDI**: Universal Description, Discovery and Integration
  - UDDI registry stores descriptions about companies and the services they offer in a common **XML** format (**for web services publishing and discovery**)

SOAP in brief

- SOAP is **based on XML**
  - SOAP = HTTP + XML
  - XMLP (XML Protocol)
- SOAP is a **communication protocol**
- SOAP is for communication **between applications**
  - one-way, request/response, multicast, etc.
- SOAP is designed to communicate **via Internet**
  - RPC over Web (SMTP, FTP, TCP/IP, HTTP)
- SOAP is a format for **sending messages**
- SOAP is **platform/language independent**
- SOAP is **simple and extensible**
- SOAP allows you to **get around firewalls**
- SOAP will be developed as a **W3C standard**

SOAP why

- It is important for application development to allow Internet communication between programs.
- Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM/CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.
- A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.
- SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP RPC

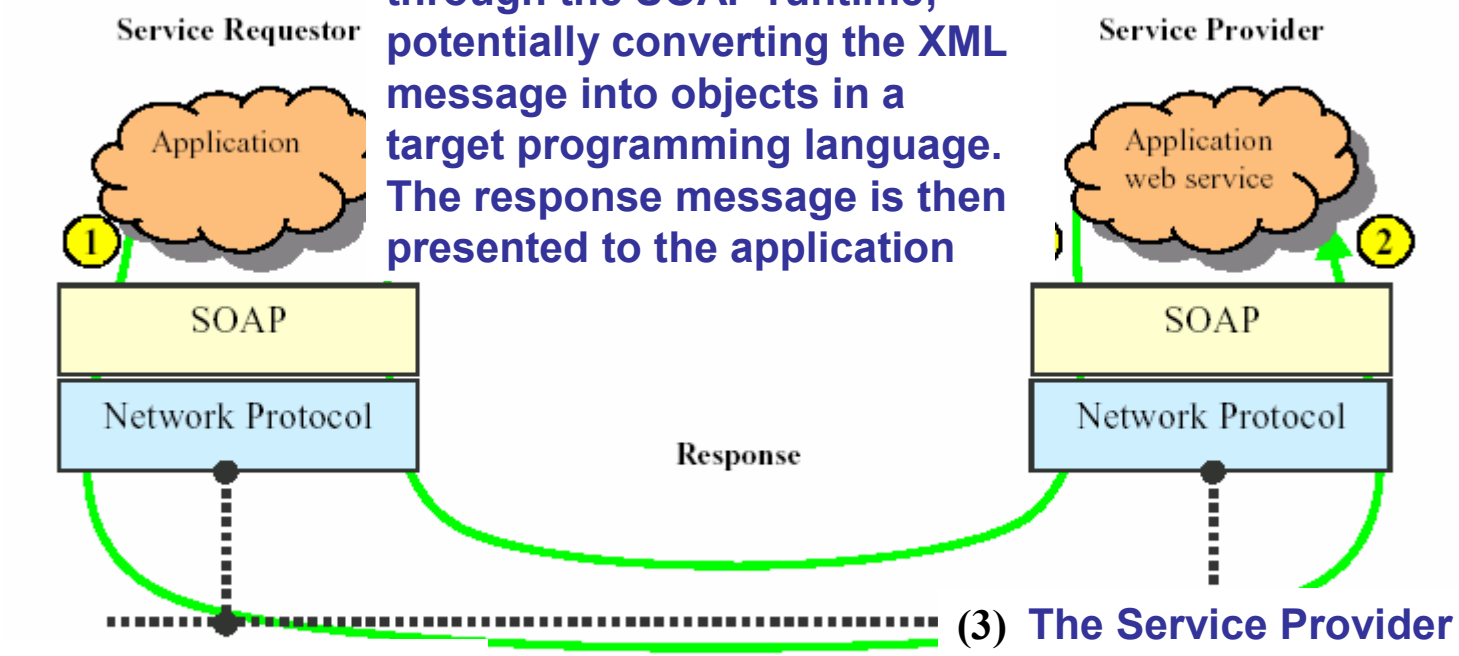
- Define an RPC protocol
  - How will types be transported (in XML) and how application represents them
  - RPC parts (object id, operation name, parameters)

Assumes a type system based on XML-schema

XML Messaging

(2) The SOAP server routes the message to the Service Provider converting XML lata into system-specific)bject

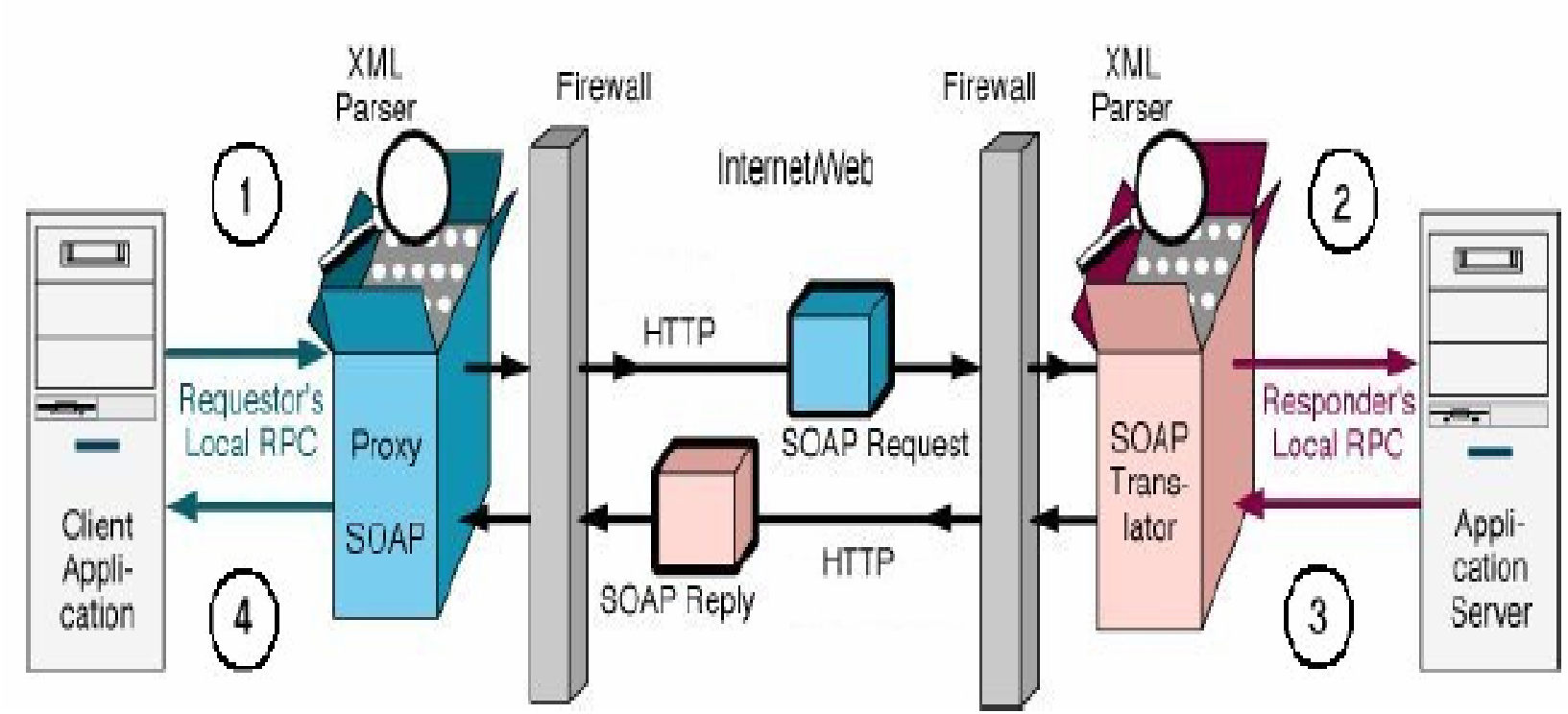
(4) The response is routed through the SOAP runtime; potentially converting the XML message into objects in a target programming language. The response message is then presented to the application



(1) A Service Requestor create SOAP message and then passes it to the network infrastructure responsible of the delivery

(3) The Service Provider processes the request and formulate a response. The SOAP server sends the response to the Requestor over the network

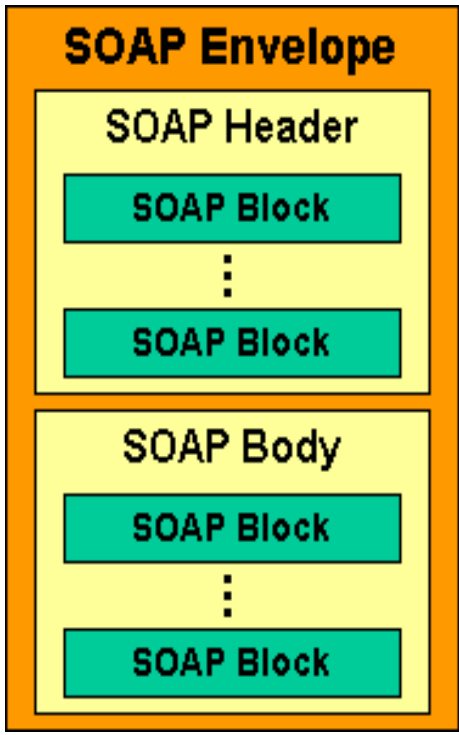
Messaging(contd)



SOAP blocks

A SOAP message is an ordinary XML document containing the following elements:

- A required **Envelope** element that identifies the XML document as a SOAP message
- An optional **Header** element that contains header information
- A required **Body** element that contains call and response information
- An optional Fault element that provides information about errors that occurred while processing the message



All the elements above are declared in the default namespace for the SOAP envelope:
http://www.w3.org/2001/12/soap-envelope
and the default namespace for SOAP encoding and data types is:
http://www.w3.org/2001/12/soap-encoding

SOAP skeleton

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
... ...
</soap:Header>
<soap:Body>
... ...
   <soap:Fault>
    ... ...
   </soap:Fault>
</soap:Body>
</soap:Envelope>
```

Skeleton(contd)

```
<SOAP:Envelope
   xmlns:SOAP='http://schemas.xmlsoap.org/soap/envelope/'
   SOAP:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
   xmlns:v='http://www.topxml.com/soapworkshop/'>

   <SOAP:Header>
         <v:From SOAP:mustUnderstand='1'>
               cdix@soapworkshop.com
         </v:From>
   </SOAP:Header>

   <SOAP:Body>
         <v:DoCreditCheck>
               <ssn>123-456-7890</ssn>
         </v:DoCreditCheck>
   </SOAP:Body>

</SOAP:Envelope>
```

Header

Body

SOAP message sample

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn

SOAPAction: "www.stockquoteserver.com/services/getquote.htm"

<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="www.stockquoteserver.com/services/getquote">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HTTP Header

Action

Envelope

Body

Real Data

SOAP doGoogleSearch

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV= http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <SOAP-ENV:Body>
        <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <key xsi:type="xsd:string">00000000000000000000000000000000</key>
            <q xsi:type="xsd:string">my query</q>
            <start xsi:type="xsd:int">0</start>
            <maxResults xsi:type="xsd:int">10</maxResults>
            <filter xsi:type="xsd:boolean">true</filter>
            <restrict xsi:type="xsd:string"/>
            <safeSearch xsi:type="xsd:boolean">false</safeSearch>
            <lr xsi:type="xsd:string"/>
            <ie xsi:type="xsd:string">latin1</ie>
            <oe xsi:type="xsd:string">latin1</oe>
        </ns1:doGoogleSearch>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example doGoogleSearchResult

```
<SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/" ……..
    <SOAP-ENV:Body>
    <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
            ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <return xsi:type="ns1:GoogleSearchResult">
            <documentFiltering xsi:type="xsd:boolean">false</documentFiltering>
            <estimatedTotalResultsCount
                    xsi:type="xsd:int">3</estimatedTotalResultsCount>
            <directoryCategories xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
                    xsi:type="ns2:Array" ns2:arrayType="ns1:DirectoryCategory[0]"/>
            <searchTime xsi:type="xsd:double">0.194871</searchTime>
            <resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
                    xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[3]">
            <item xsi:type="ns1:ResultElement">
            <cachedSize xsi:type="xsd:string">12k</cachedSize>
            <directoryCategory
    xsi:type="ns1:DirectoryCategory">Category</directoryCategory>
            <relatedInformationPresent
                    xsi:type="xsd:boolean">true</relatedInformationPresent>
            <directoryTitle xsi:type="xsd:string"/>
            <summary xsi:type="xsd:string"/>
            <URL xsi:type="xsd:string">http://hci.stanford.edu/cs147/example/shrdlu/</URL>
            <title xsi:type="xsd:string">&lt;b&gt;SHRDLU&lt;/b&gt;</title>
            </item>
```

WSDL in brief

- WSDL is an XML vacabulary
- WSDL is used to describe WS
- WSDL is also used to locate WS

WSDL 1.1 is not yet a W3C standard (but soon)

WS Description

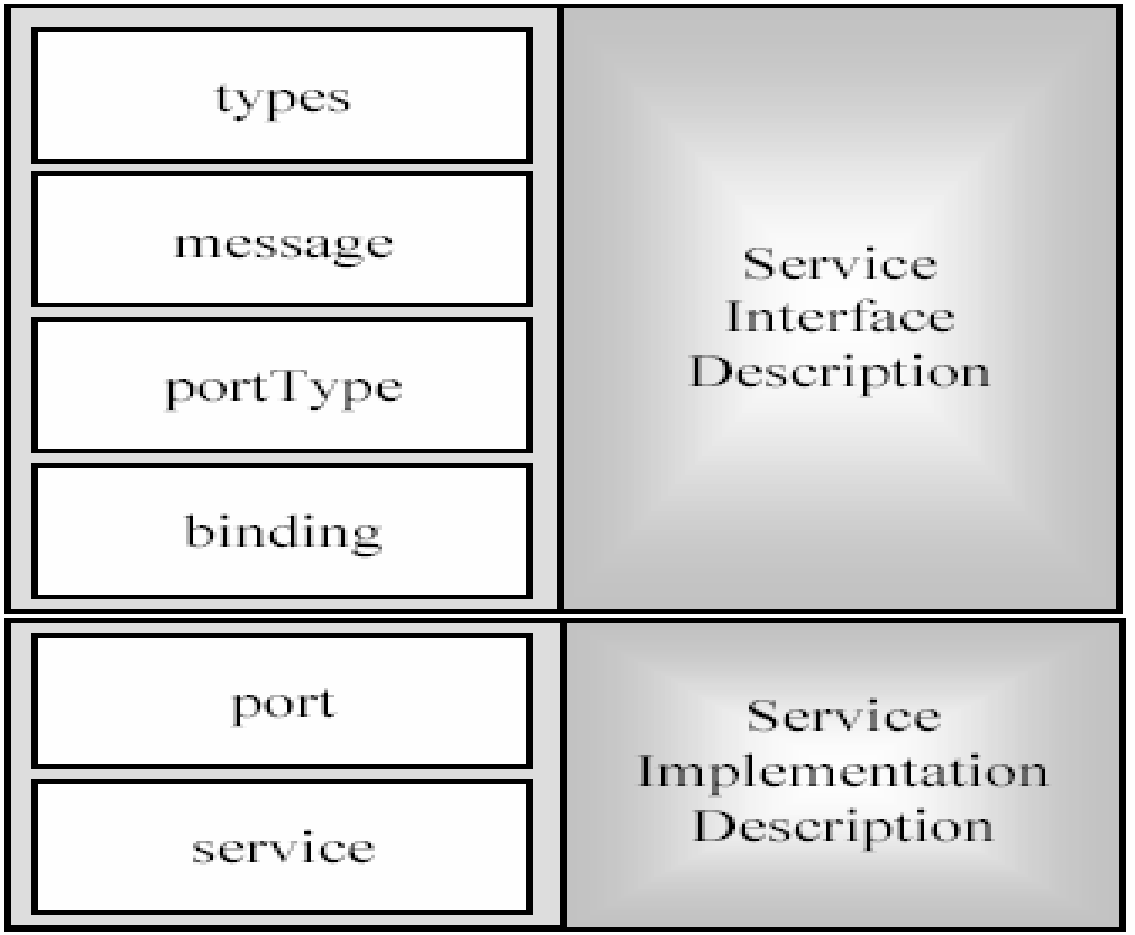
- Why describe Web services?
  - A service requestor needs to analyze a service for his requirements
  - A Web service needs to provide the following information
    - the operations it supports
    - the transport and messaging protocols on which it supports those operations
    - the network endpoint of the web service
- Languages such as WSDL, DAML-S, RDF can be used for describing Web services
  - WSDL – describes the syntactic information of a service
  - DAML-S and RDF – describe the syntactic as well as the semantic information

WSDL Structure

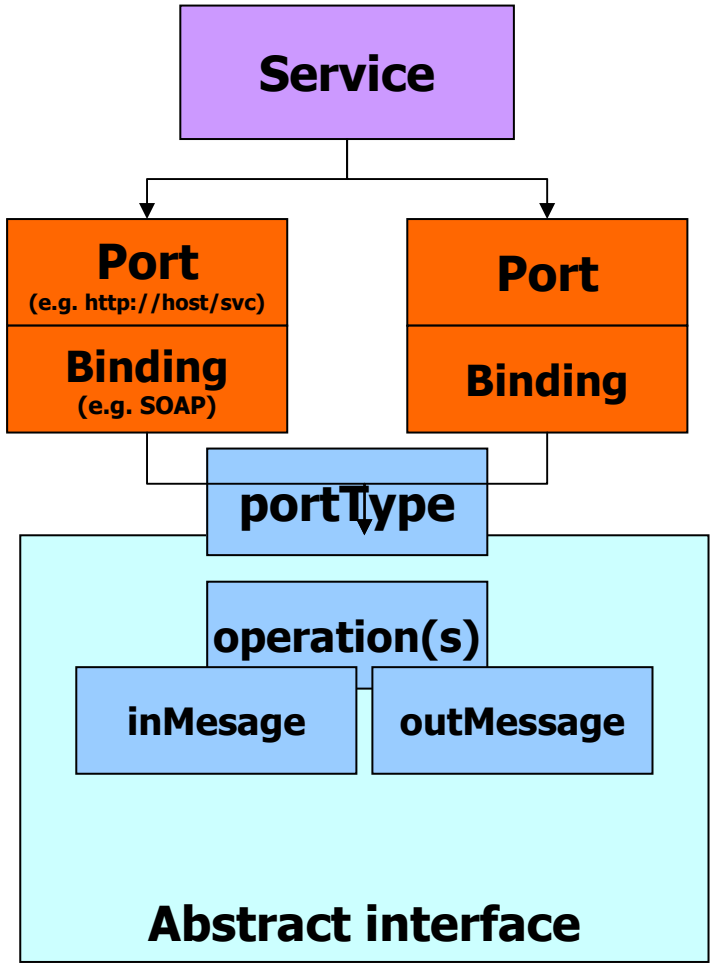
- Service
  collection of endpoints
  endpoints= port + binding
- Port type
  set of operations supported by endpoints
  (ref. a function library/a module/a class)
- Operations
  abstract description of action
- Binding
  maps abstract specification to specific protocol
- Ports
  network address/URI that implements service
- Message
  typed definition of data communicated

Type (data type definition/XSD) is not necessary for primitive data type

Structure(contd)



Structure(contd)



WSDL File

```
<definitions>

    <types>
          definition of types..
    </types>

    <message>
          definition of messages...
    </message>

    <portType>
          <operation> ..... </operation>
          <operation> ..... </operation>
    </portType>


    <binding>
          definition of binding....
    </binding>

    <service>
          <port>....</port>
          <port>....</port>
    </service>

</definitions>
```

Abstract Description

Concrete Description

Grammar

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?
                  xmlns:prefix="uri"*>

  <import namespace="uri" location="uri"/>*

  <wsdl:documentation ... />?

  <wsdl:types>*
    <wsdl:documentation ... />?
    <xsd:schema ... />*
    <-- extensibility element -->*
  </wsdl:types>

  <wsdl:message name="nmtoken">*
    <wsdl:documentation ... />?
    <part name="nmtoken" element="qname"? type="qname"?/>*
  </wsdl:message>
```

Grammar(contd)

```
<wsdl:portType name="nmtoken">*
  <wsdl:documentation ... />?
  <wsdl:operation name="nmtoken">*
    <wsdl:documentation ... />?
    <wsdl:input name="nmtoken"? message="qname">?
      <wsdl:documentation ... />?
    </wsdl:input>
    <wsdl:output name="nmtoken"? message="qname">?
      <wsdl:documentation ... />?
    </wsdl:output>
    <wsdl:fault name="nmtoken" message="qname">*
      <wsdl:documentation ... />?
    </wsdl:fault>
  </wsdl:operation>
</wsdl:portType>
```

Grammar(contd)

```
<wsdl:binding name="nmtoken" type="qname">*
  <wsdl:documentation ... />?
  <-- extensibility element -->*
  <wsdl:operation name=nmtoken">*
    <wsdl:documentation ... />?
    <-- extensibility element -->*
    <wsdl:input name="nmtoken"?>?
      <wsdl:documentation ... />?
      <-- extensibility element -->*
    </wsdl:input>
    <wsdl:output name="nmtoken"?>?
      <wsdl:documentation ... />?
      <-- extensibility element -->*
    </wsdl:output>
    <wsdl:fault name="nmtoken">*
      <wsdl:documentation ... />?
      <-- extensibility element -->*
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
```

Grammar(contd)

```
<wsdl:service name="nmtoken">*
    <wsdl:documentation ... />?
    <wsdl:port name="nmtoken" binding="qname">*
      <wsdl:documentation ... />?
      <-- extensibility element -->
    </wsdl:port>
    <-- extensibility element -->*
  </wsdl:service>

  <-- extensibility element -->*

</wsdl:definitions>
```

WSDL Abstract interface

- Messages exchanged in an interaction
- Components:
  - Vocabulary (XSD for type definition)
  - Message: abstract, typed data definition sent to and from services
  - Interaction

Vocabulary

```
<wsdl:types>
  <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:GoogleSearch">
  <xsd:complexType name="GoogleSearchResult">
    <xsd:all>
      <xsd:element name="documentFiltering" type="xsd:boolean"/>
      <xsd:element name="searchComments" type="xsd:string"/>
      <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
      <xsd:element name="estimateIsExact" type="xsd:boolean"/>
      <xsd:element name="resultElements" type="typens:ResultElementArray"/>
      <xsd:element name="searchQuery" type="xsd:string"/>
      <xsd:element name="startIndex" type="xsd:int"/>
      <xsd:element name="endIndex" type="xsd:int"/>
      <xsd:element name="searchTips" type="xsd:string"/>
      <xsd:element name="directoryCategories"
                   type="typens:DirectoryCategoryArray"/>
      <xsd:element name="searchTime" type="xsd:double"/>
    </xsd:all>
</xsd:complexType>
```

Message

```
<message name="doGoogleSearch">
        <part name="key" type="xsd:string"/>
        <part name="q" type="xsd:string"/>
        <part name="start" type="xsd:int"/>
        <part name="maxResults" type="xsd:int"/>
        <part name="filter" type="xsd:boolean"/>
        <part name="restrict" type="xsd:string"/>
        <part name="safeSearch" type="xsd:boolean"/>
        <part name="lr" type="xsd:string"/>
        <part name="ie" type="xsd:string"/>
        <part name="oe" type="xsd:string"/>
</message>
<message name="doGoogleSearchResponse">
        <part name="return" type="typens:GoogleSearchResult"/>
</message>
```

Interaction

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
    <soap:binding style="rpc“
    transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="doGetCachedPage">
        <soap:operation soapAction="urn:GoogleSearchAction"/>
        <input>
            <soap:body use="encoded"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:GoogleSearch"/>
        </input>
        <output>
            <soap:body use="encoded"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:GoogleSearch"/>
        </output>
    </operation>
```

UDDI in brief

- UDDI is a directory for storing information about WS
- UDDI is a directory of WS interfaces described by WSDL
- UDDI communicates via SOAP
  - Standards-based specifications for service description and discovery
  - Shared operation of a business registry on the web

UDDI Why

Broader B2B

A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols

Smarter Search

A flower shop in Australia wants to be "plugged in" to every marketplace in the world, but doesn't know how

Easier Aggregation

A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.

Describe Services

Discover Services

Integrate Them Together

UDDI How

1. 
SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

2. 
Businesses populate the registry with descriptions of the services they support

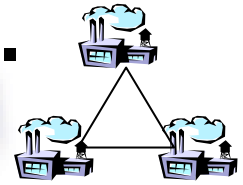
UDDI Business Registry

Business Registrations

Segrvice Type Reistrations

3. UBR assigns a programmatically unique identifier to each service and business registration

4. 
Marketplaces, search engines, and business apps query the registry to discover services at other companies

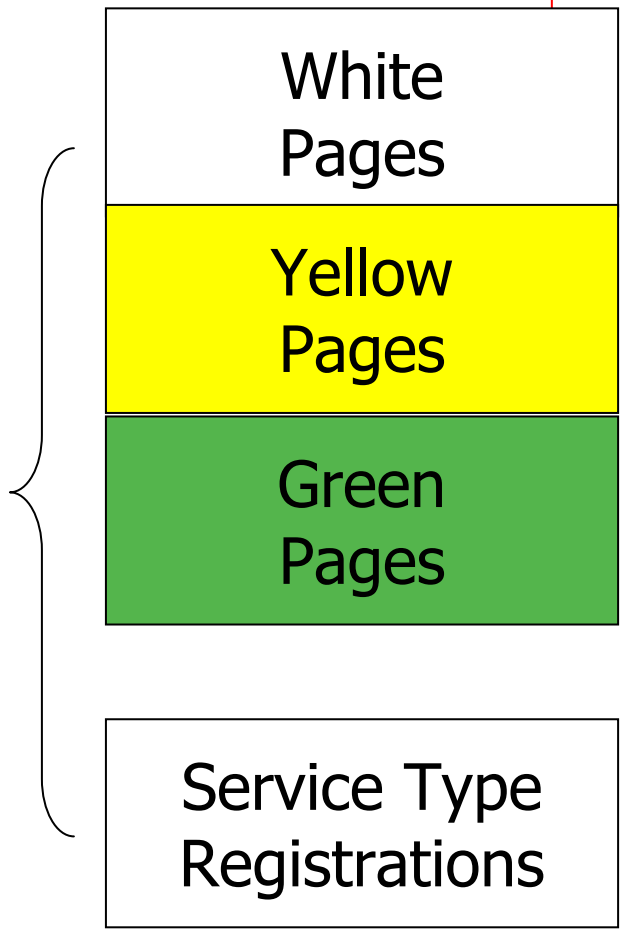
5. 
Business uses this data to facilitate easier integration with each other over the Web

UUDI Classification

- UDDI classifies businesses and services according to standard taxonomies
- Why Classification ?
  - Searches based on keywords alone, could return a large set of hits for a particular search
  - Classification of services and businesses allows to perform better searches

Registry Data

- Businesses register public information about themselves
- Standards bodies, Programmers, Businesses register information about their Service Types



White Pages

- Business Name
- Text Description
  - list of multi-language text strings
- Contact info
  - names, phone numbers, fax numbers, web sites…
- Known Identifiers
  - list of identifiers that a business may be known by - DUNS, Thomas, other

Yellow Pages

- Business categories
  - 3 standard taxonomies in V1
    - Industry: NAICS (Industry codes - US Govt.)
    - Product/Services: UN/SPSC (ECMA)
    - Location: Geographical taxonomy
  - Implemented as name-value pairs to allow any valid taxonomy identifier to be attached to the business white page

Green Pages

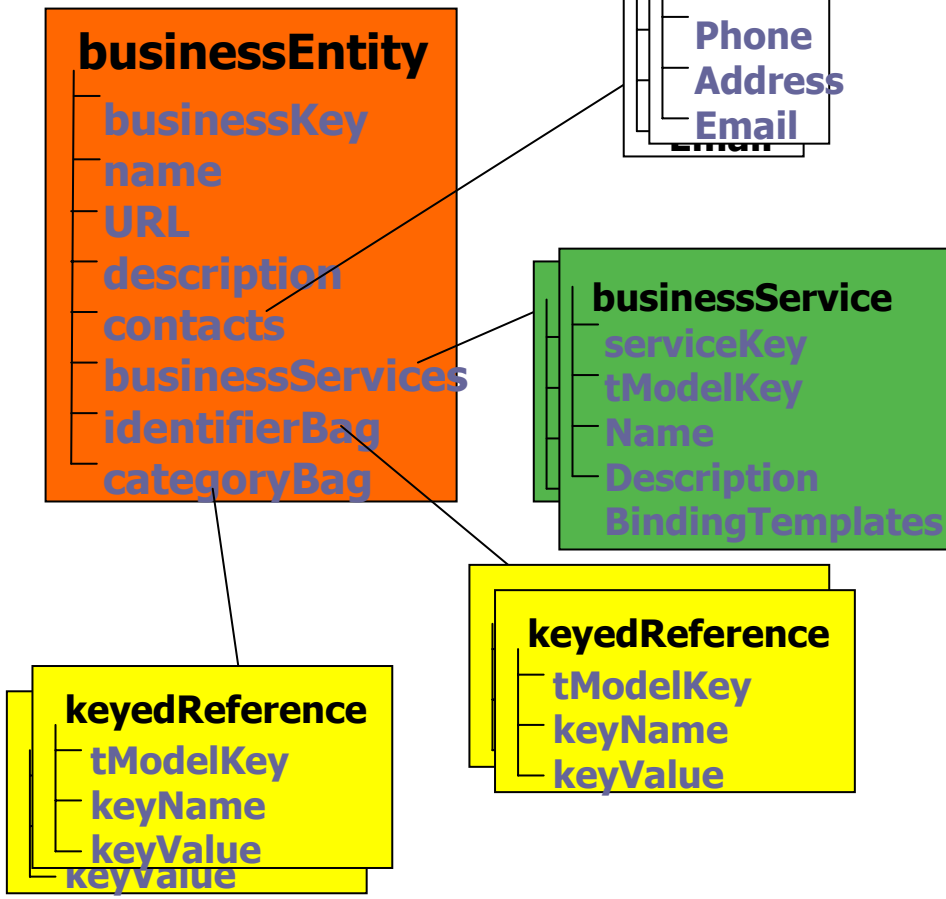
- New set of information businesses use to describe how to “do e-commerce” with them
  - Nested model
    - Business processes
    - Service descriptions
    - Binding information
  - Programming/platform/implementation agnostic
  - Services can also be categorized

Service Type Registration

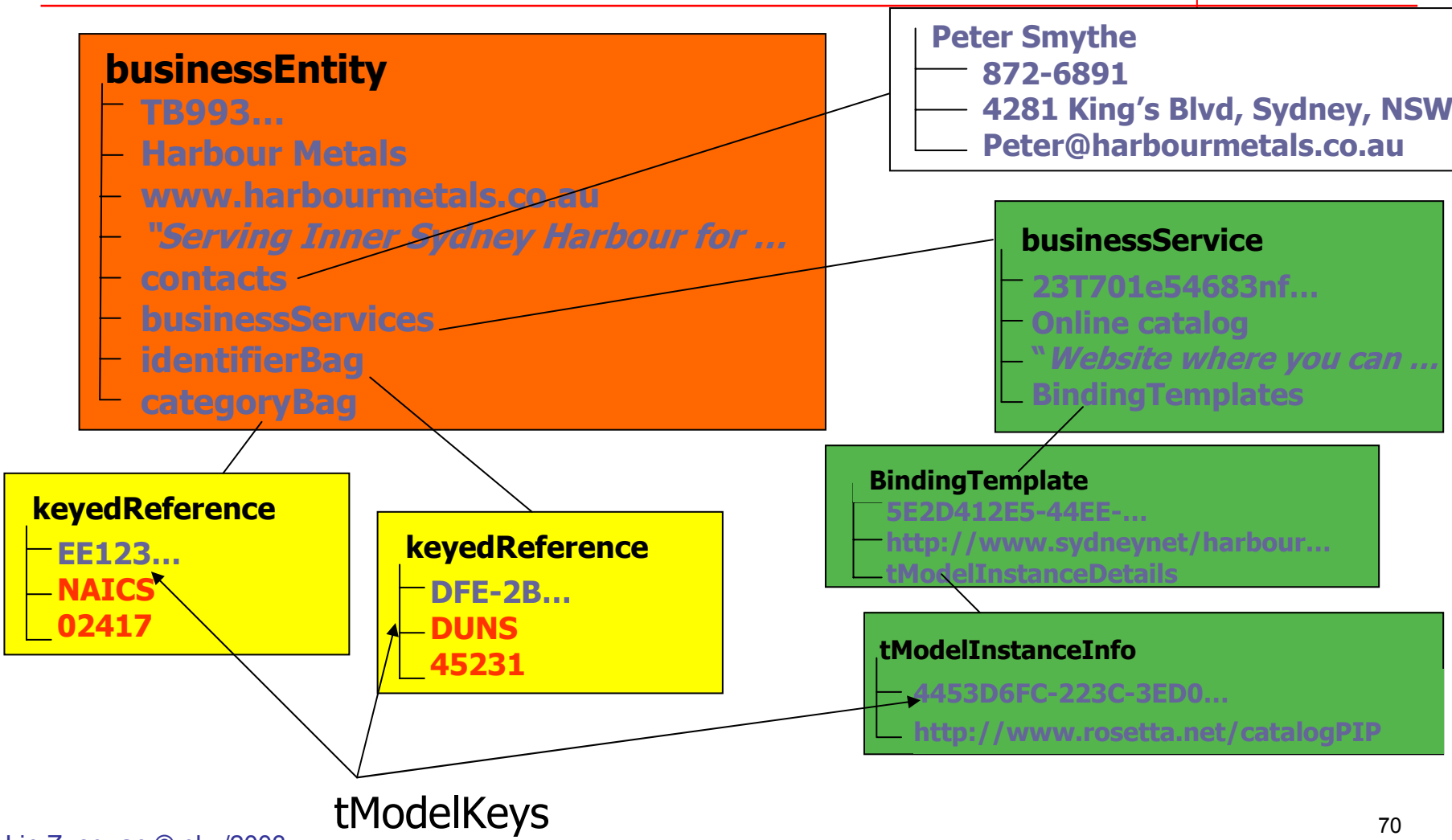
- Pointer to the namespace where service type is described
  - What programmers read to understand how to use the service
- Identifier for who published the service
- Identifier for the service type registration
  - called a tModelKey
  - Used as a signature by web sites that implement those services

Business Registration

- XML document
- Created by end-user company (or on their behalf)
- Can have multiple service listings
- Can have multiple taxonomy listings



Example of a Registration



UDDI at Work

1. Harbour Metals creates online website with local ASP

SydneyNet.com

UDDI Registry

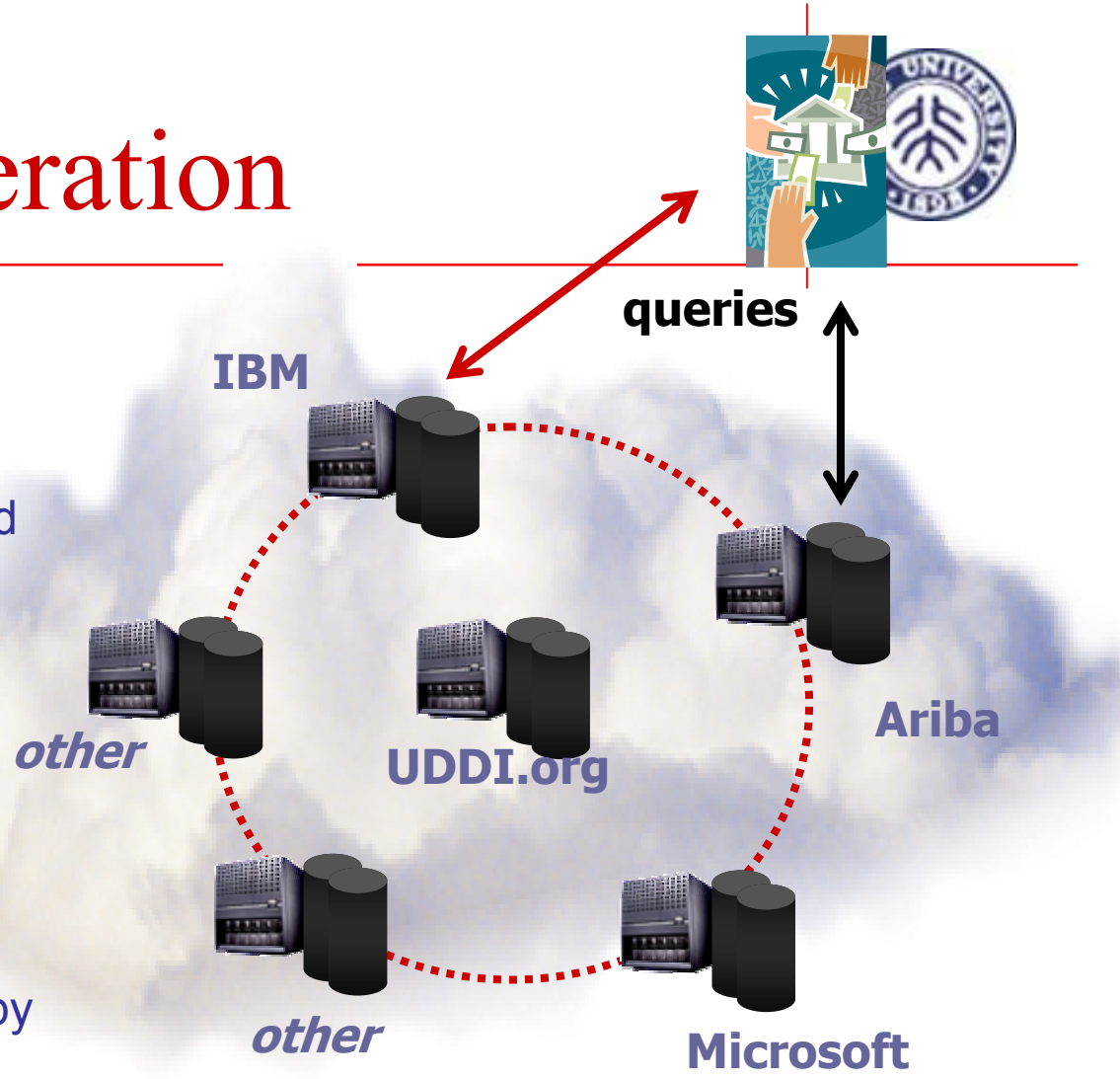
2. ASP registers Harbour Metals with UBR

3. Marketplaces and search engines query UBR, cache Harbour Metals data, and bind to its services

4. Consumers and businesses discover Harbour Metals and do business with it

Registry Operation

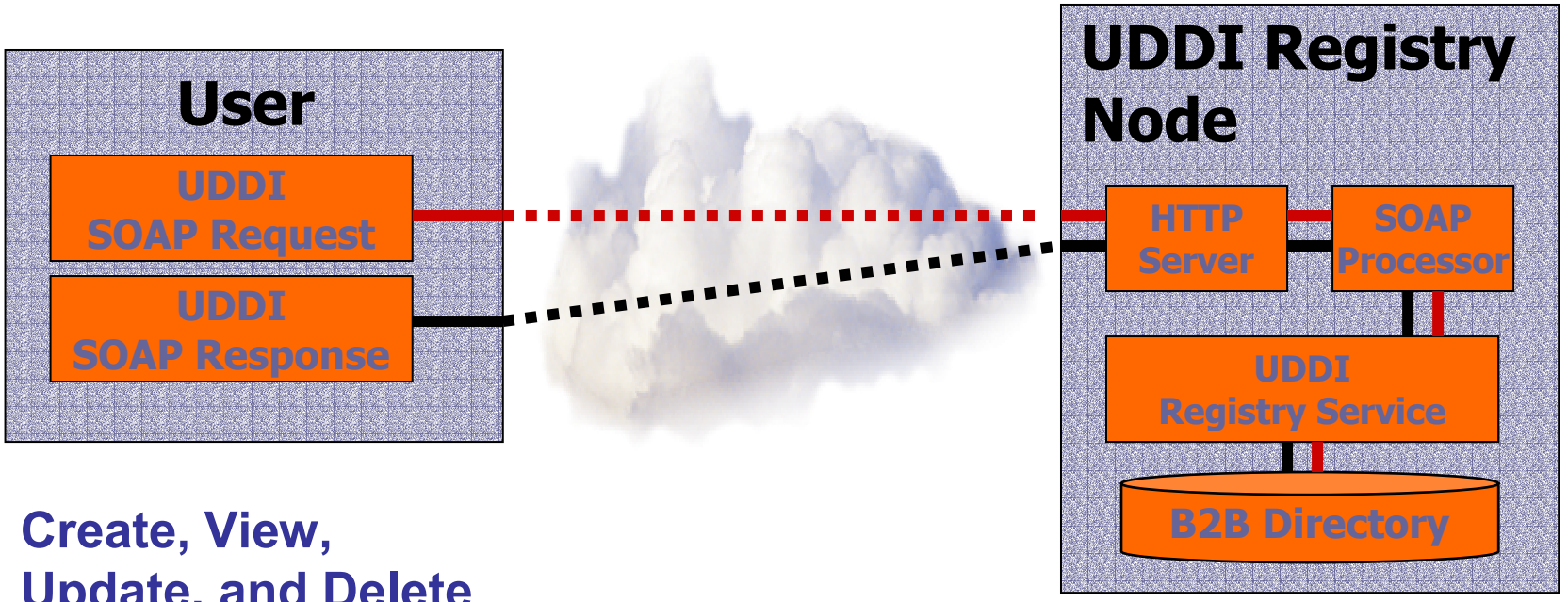
- Peer nodes (websites)
- Companies register with any node
- Registrations replicated on a daily basis
- Complete set of "registered" records available at all nodes
- Common set of SOAP APIs supported by all nodes
- Compliance enforced by business contract



Why a DNS-like Model?

- Enforces cross-platform compatibility across competitor platforms
- Demonstration of trust and openness
- Avoids tacit endorsement of any one vendor's platform
- *May migrate to a third party*

UDDI and SOAP



Create, View, Update, and Delete registrations

Implementation-neutral

UUDI APIs

- Publication API - Authenticated set of operations that allow organizations to publish businesses, services, service type specifications
- Inquiry API - Non authenticated public set of operations that allows users to extract information out of the UDDI registry.

Registry APIs
SOAP Messages

- Inquiry API
  - Find things
    - find_business
    - find_service
    - find_binding
    - find_tModel
  - Get Details about things
    - get_businessDetail
    - get_serviceDetail
    - get_bindingDetail
    - get_tModelDetail
- Publishers API
  - Save things
    - save_business
    - save_service
    - save_binding
    - save_tModel
  - Delete things
    - delete_business
    - delete_service
    - delete_binding
    - delete_tModel
  - security…
    - get_authToken
    - discard_authToken

WS Composition

Composition

- Definition

  Technique of composing the uncontionalities of relatively simpler services to produce a ‘meaningful’ arbitrarily complex application

Composition
Classification

- Proactive Composition & Reactive Composition
  - Proactive: offline composition of available services
    - When: services are stable and always running
    - Example: ticket reservation service
  - Reactive: dynamically creating a composite service.
    - When: composite service not often used and service processes not stable.
    - Example: tour manager where the itinerary is not predefined

Classification(contd)

- Mandatory & Optional-Composite Services
  - Mandatory: all subcomponents must participate to yield a result
    - Example: service that calculates the averages of stock values for a company.
  - Optional: subcomponents are not obligated to participate for a successful execution.
    - Example: services that include a subcomponent that is an optimizer.

Some Issues on WSC

- Service Discovery
- Service Coordination and Management
- Uniform Information Exchange Infrastructure
- Fault Tolerance and Scalability
- Adaptiveness
- Reliability & Transactions
- Security
- Accountability
- Testing

Languages

- WSFL (Web Services Flow Language)
- BPML (Business Process Modeling Language)
- XLANG
- ebXML BPSS (Business Process Specification Schema)
- BPEL4WS (Business Process Execution Language for Web Services)

WS Development

- WS Development Lifecycle
- WS Provider
- WS Requestor

WS Development Lifecycle

The development lifecycle can have four phase:

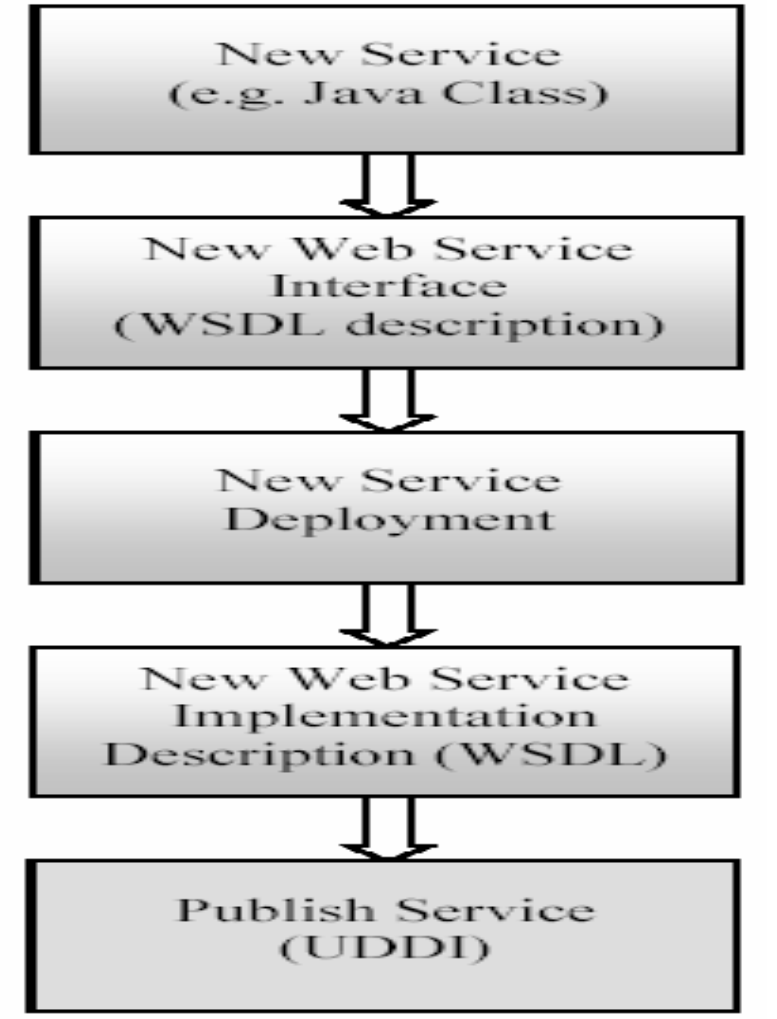
- **BUILD :**
  - development and testing of the web service implementation
  - definition of the services interface description
  - definition of the service implementation description
- **DEPLOY** :
  - publication of the services interface and service implementation
- **RUN** :
  - The web service is available for invocation
  - The web service is fully deployed, operational and network-accessible from the services provider
  - Now the service requestor can perform the find and bind operations
- **MANAGE** :
  - Covers ongoing management and administration of the web services application

WS Provider

Service Provider Methods

	New Service Interface	*Existing Service Interface*
New Web Service	Green-field	Top-down
Existing Application/Service	Bottom-up	Meet-in-the-middle

Green-field Scenario

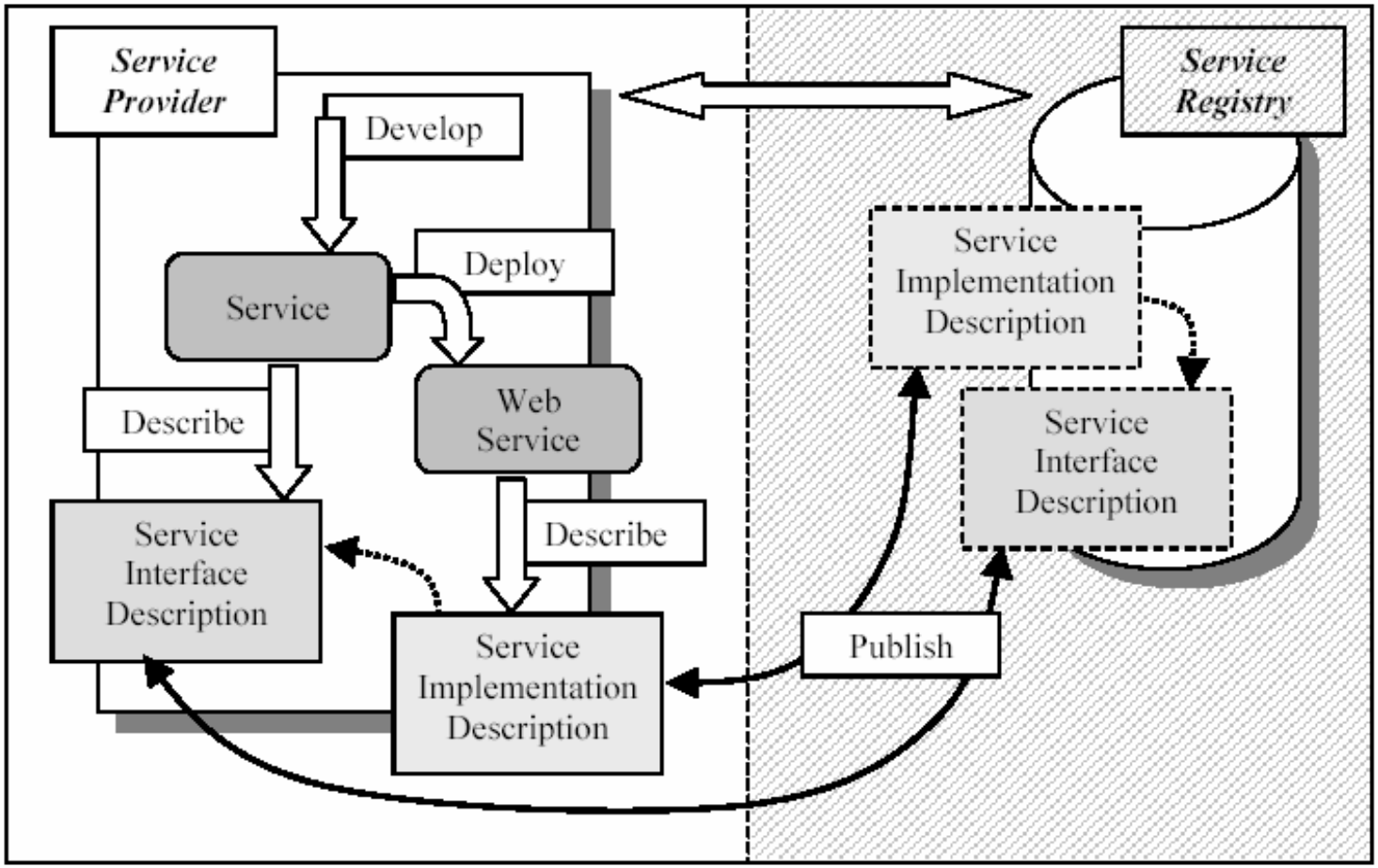


Green-field(contd)

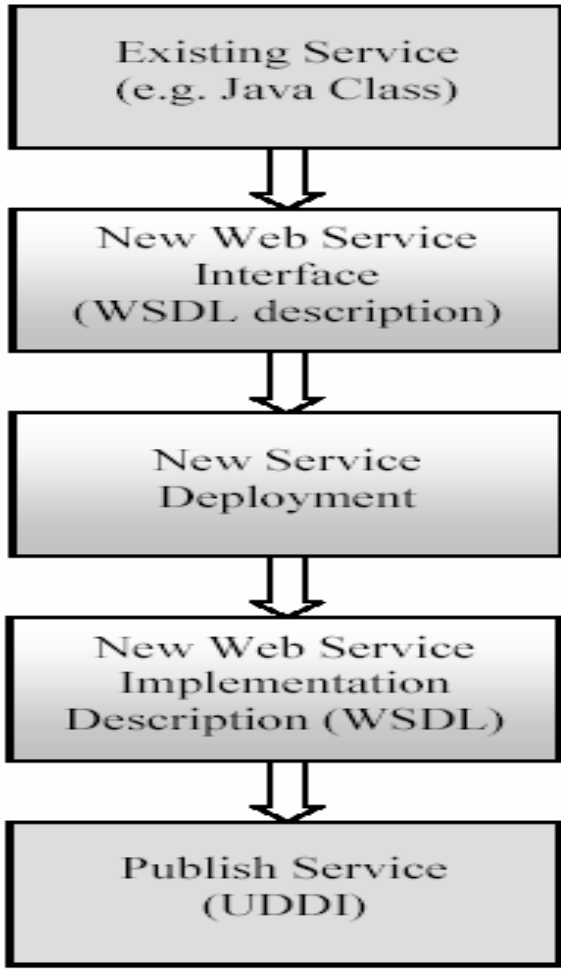
- Build Phase
  1. Development of the new web service
  2. Defining a new service interface
- Deploy Phase
  1. Publishing of the service interface
  2. Deployment of the web service
  3. Creation of the service implementation description
  4. Publishing the service implementation description
- Run Phase

  Running the web service

Green-field(contd)



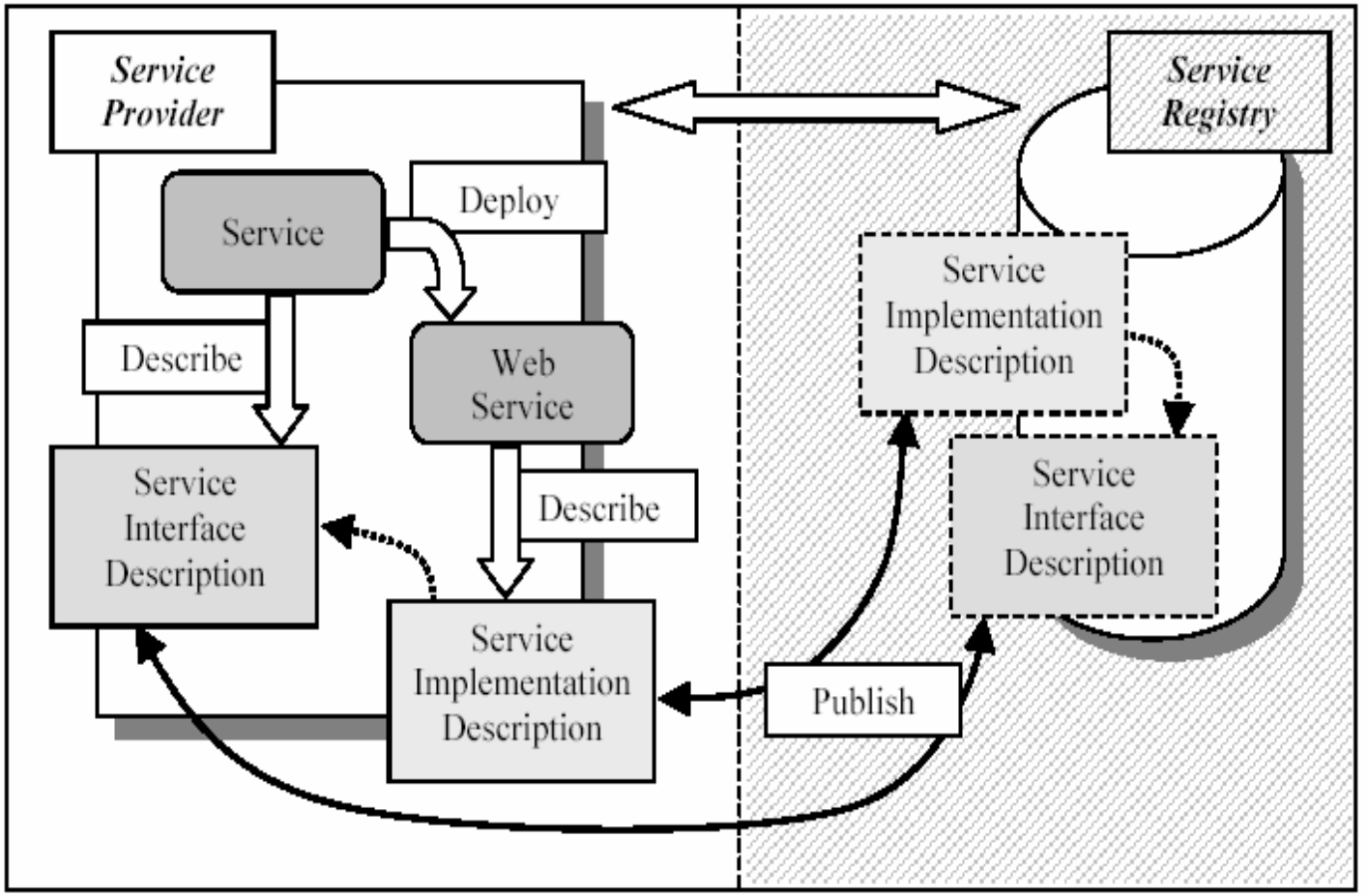
Bottom-up Scenario



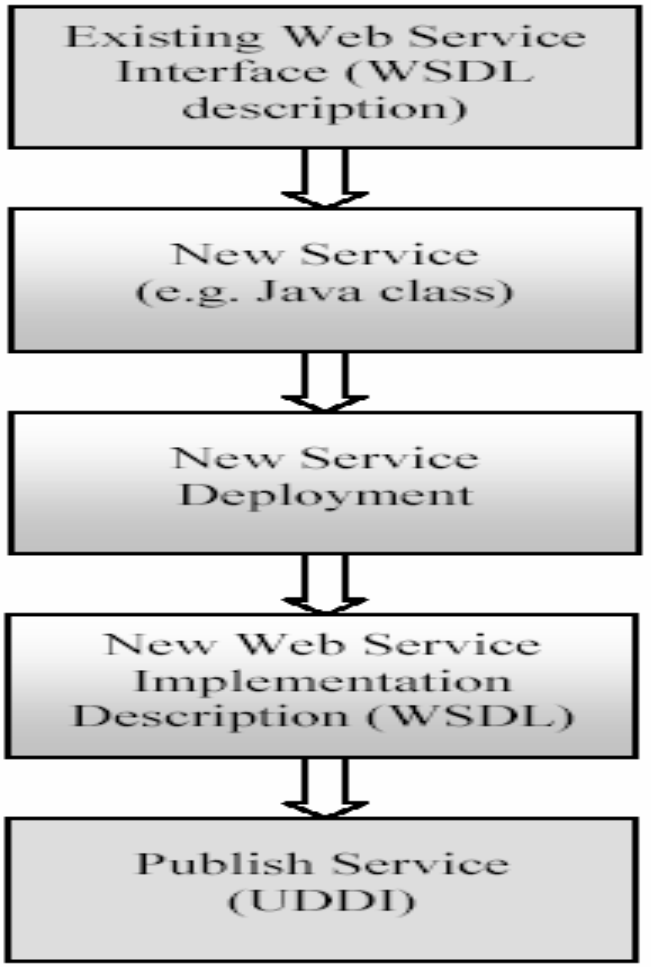
Bottom-up(contd)

- Build Phase
  Creation of the web service interface
- Deploy Phase
  1. Deployment of the web service
  2. Creation of the service implementation description
  3. Publishing the service interface description
  4. Publishing the service implementation description
- Run Phase
  Running the web service

Bottom-up(contd)



Top-down Scenario

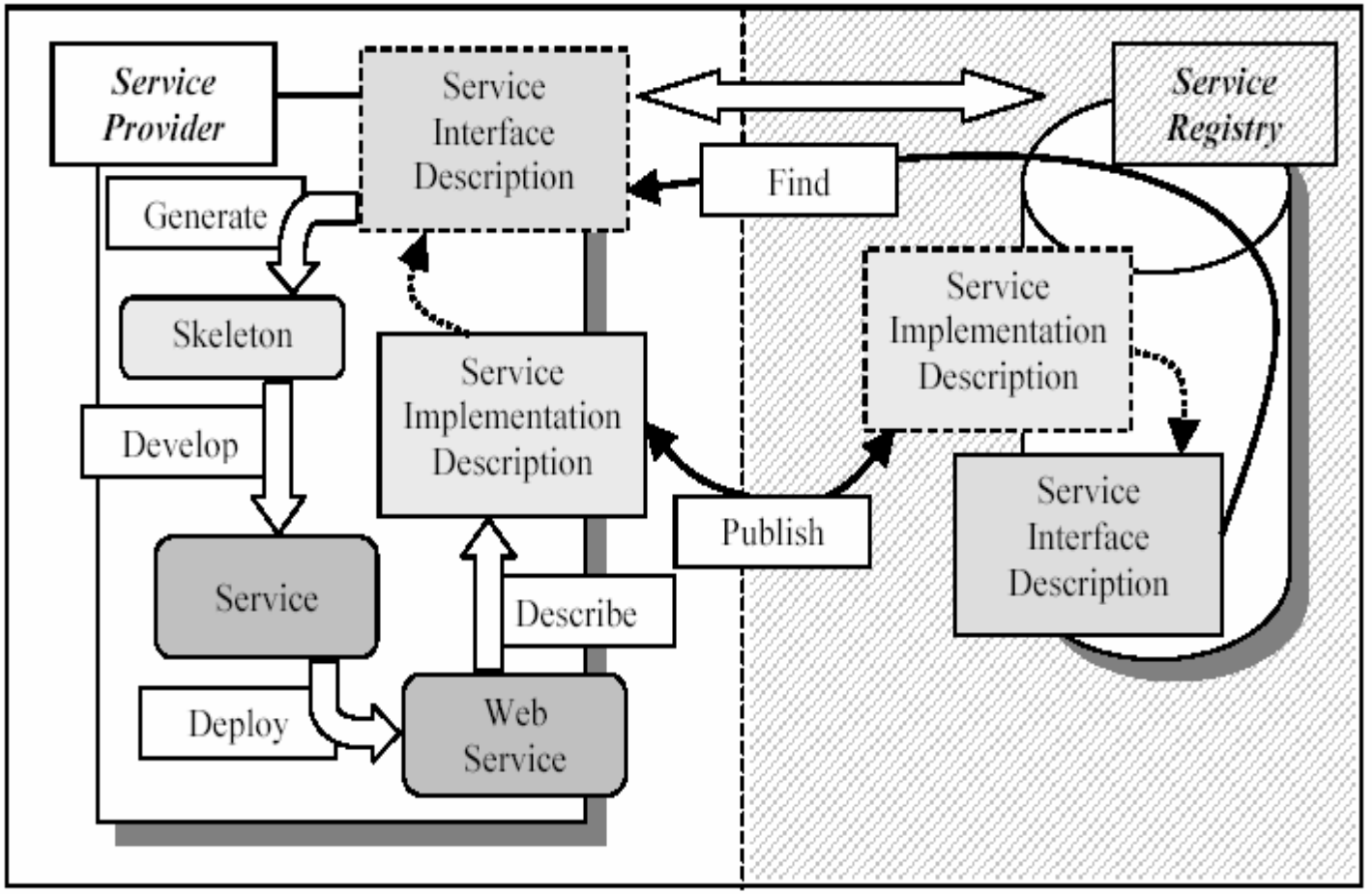


Top-down(contd)

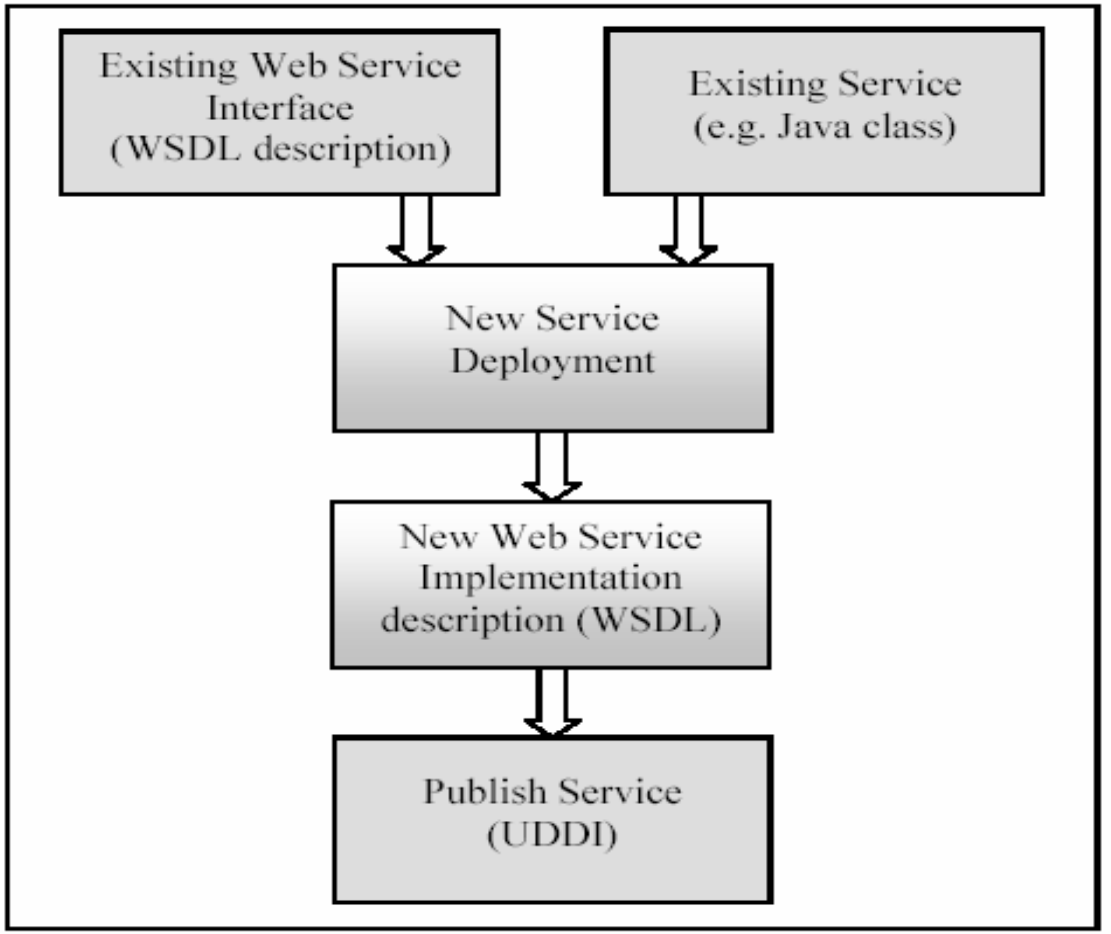
- Build Phase
  1. Finding the service interface
  2. Creation of a service implementation template
  3. Development of the new web service
- Deploy Phase
  1. Deployment of the web service
  2. Creation of the service Implementation description
  3. Publishing the service Implementation description
- Run Phase

  Running the web service

Top-down(contd)



Meet-in-the-middle Scenario

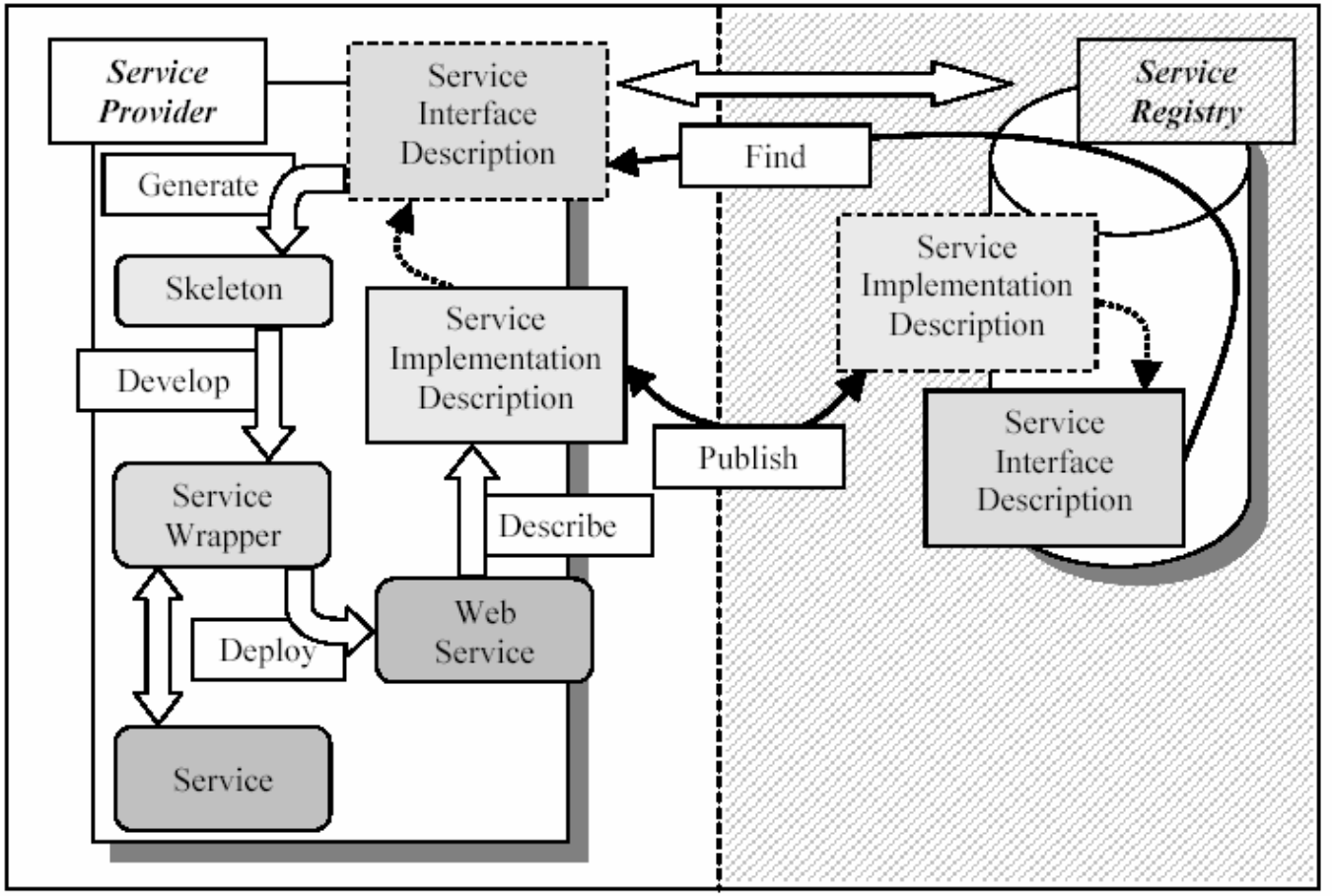


Meet-in-the-middle(contd)

- Build Phase
  1. Finding the service interface
  2. Creation of a service implementation template
  3. Development of the service wrapper
- Deploy Phase
  1. Deployment of the web service
  2. Creation of the service implementation description
  3. Publishing the service implementation description
- Run Phase

  Running the web service

Meet-in-the-middle(contd)



WS Requestor

- Static Binding

  there is only one service implementation that can be used at runtime

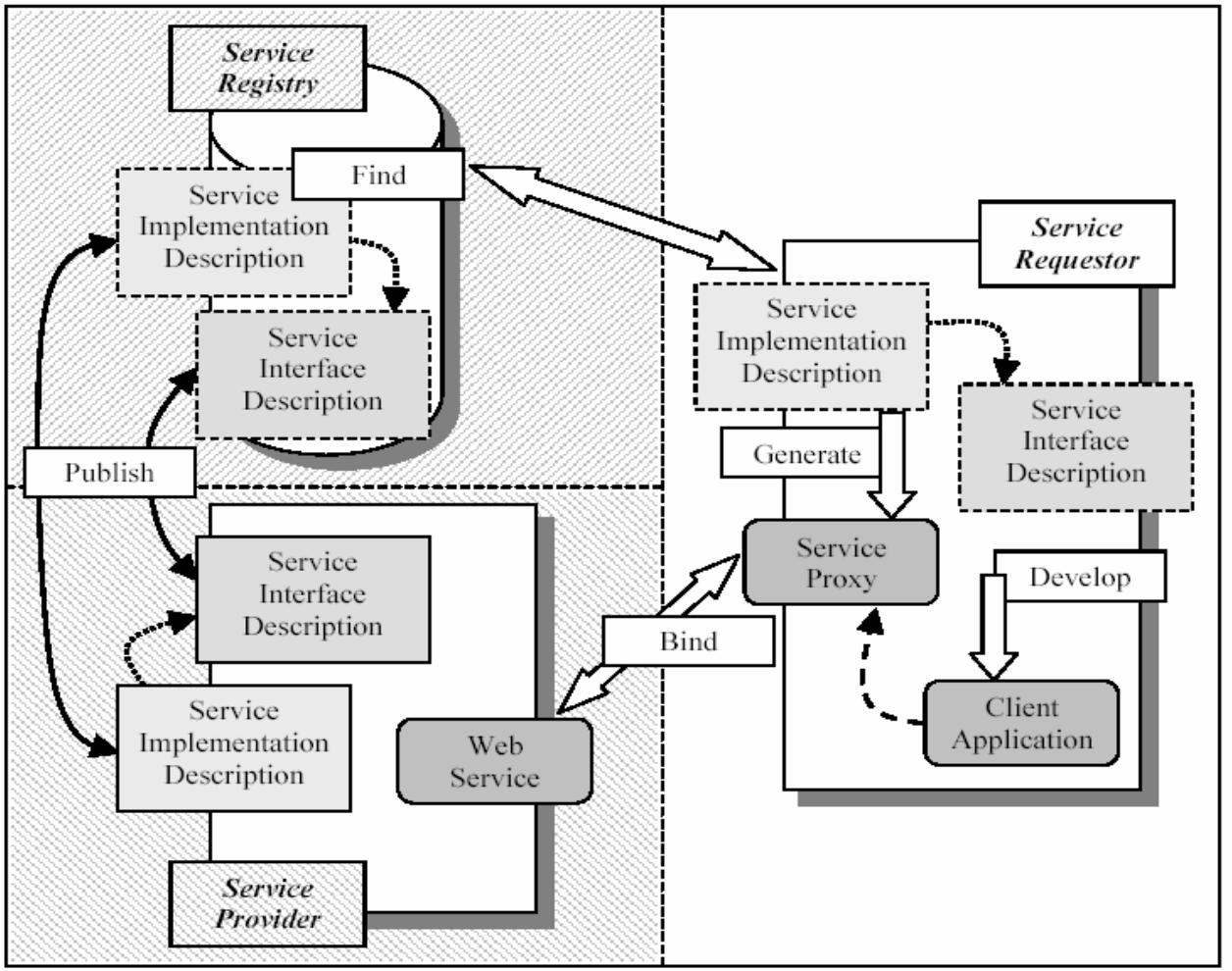
- Dynamic Binding

  a service requestor wants to use a specific type of web service, but its implementation is not known until runtime or it can change at runtime

Static Binding

- Build Phase
  1. Finding the service implementation description
  2. Creation of the service proxy
  3. Development of the client application
- Deploy Phase
  Deployment of the service proxy and client application
- Run Phase
  Invocation of the web service

Static binding(contd)

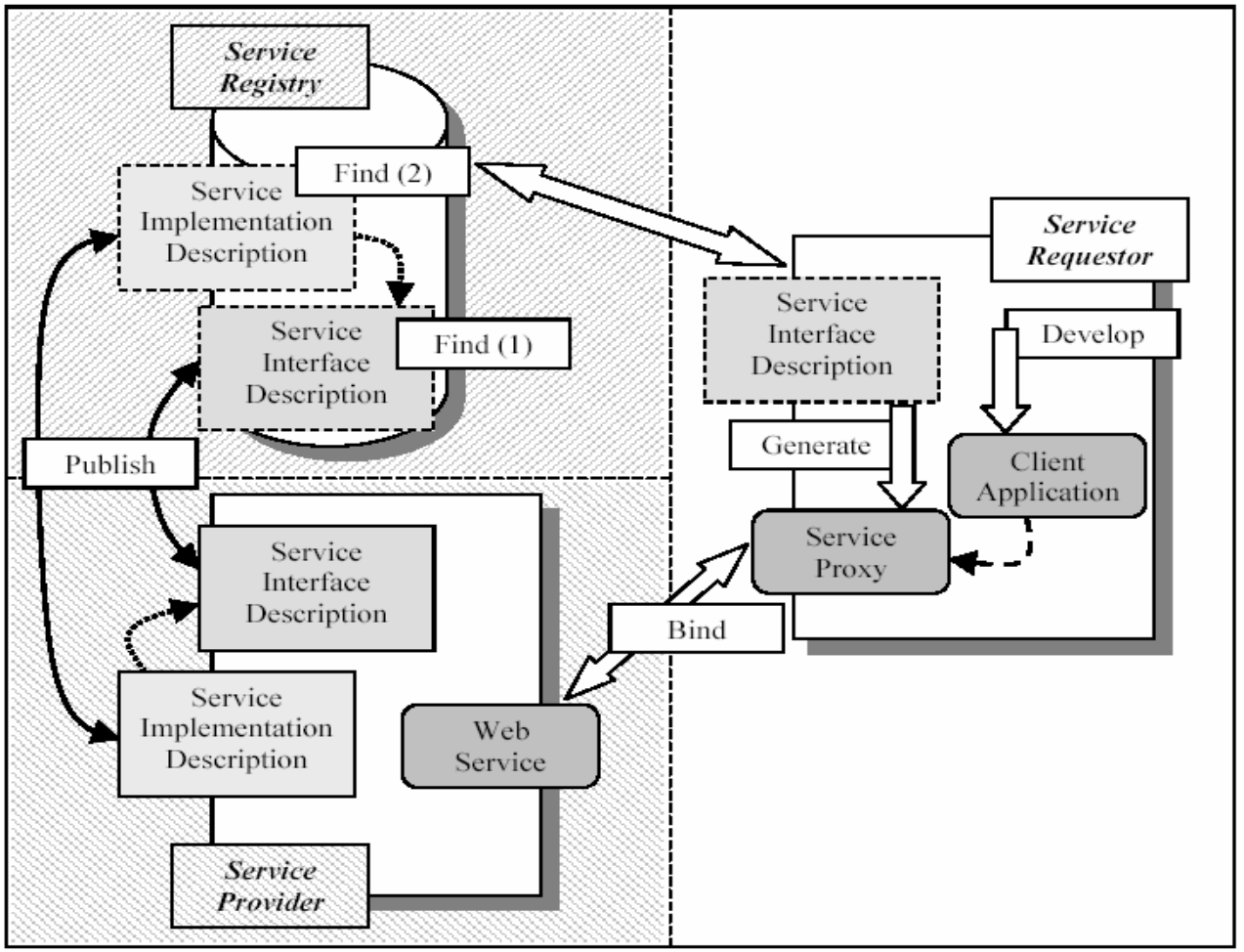


Dynamic Binding

- Build Phase
  1. Finding the service interface description
  2. Creation of a generic service proxy
  3. Development of the client application
- Deploy Phase

  Deployment of the service proxy and client application
- Run Phase
  1. finding the service implementation description
  2. Invocation of the web service

Dynamic binding(contd)



WS Applications

- State of the art
- Some examples
- WS Tools

State of the Art

- Interoperability by minimizing the requirements for shared understanding
- Interoperability of legacy applications
- Reduction of complexity by encapsulation
- Just-in-time integration
- Ubiquity
- Low barrier to entry
- Industry Support

Some Examples
Airport Weather Service

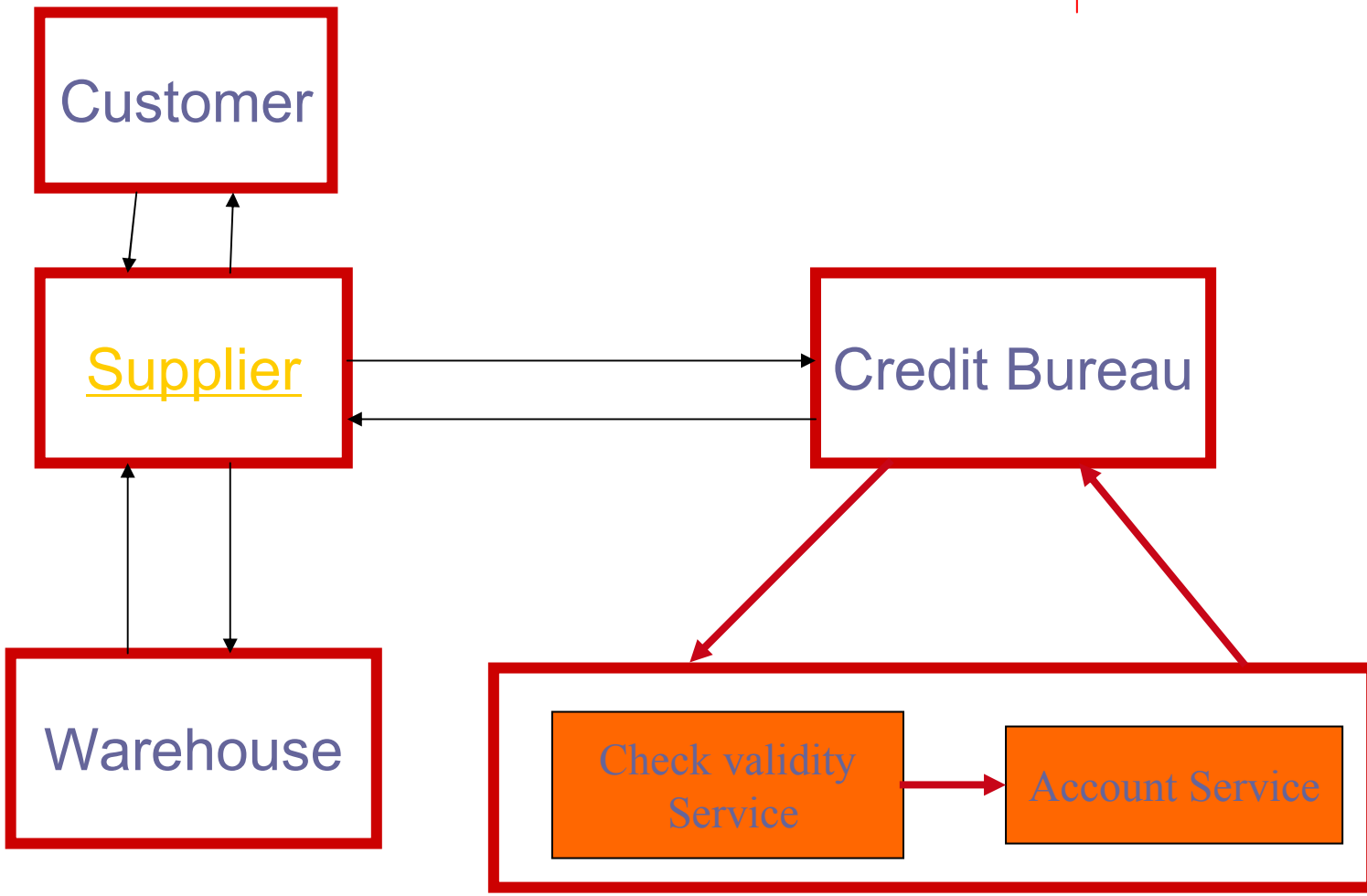
Interface:

- getLocation()
- getWind()
- getTemperature()
- getSky()
- getPressure()
- getHumidity()
- getSummary()

More informations:

- Web Interface:
  http://live.capescience.com/AirportWeather/index.html

Example(contd)
B2B



WS Tools

- Microsoft SOAP Toolkit 2.0
- Microsoft Visual Studio .NET and .NET Framework
- Apache SOAP Toolkit and AXIS
- IBM Web Services Toolkit
- SUN JavaTM 2 Platform, Enterprise Edition (J2EE)

Tool(contd)
Web Service ToolKit

- http://alphaworks.ibm.com
- Components
  - A bundle of other separately available tools, e.g.
    - The Apache SOAP Client API and server package
    - IBM UDDI4J, which is an implementation of the "UDDI Programmer's API Version 1.0 Specification"
  - The "Embedded IBM WebSphere" Application Server which is a "lite" version of the corresponding commercial product for test purposes
  - A private UDDI Version 1.0 Registry implementation requiring the IBM DB2 Database
  - The UDDI4B Web Services Browser that allows discovery of the businesses and services
    published at the IBM UDDI Test Registry
  - Tools for generating on the one hand WSDL documents from existing applications, or on
    the other hand client and server application code from a WSDL document (see below).
  - A variety of Web Services documentation
  - Some Web Services demo applications