T-110.5140 Network Application Frameworks and XML

Web Services and WSDL

15.2.2010

Tancred Lindholm

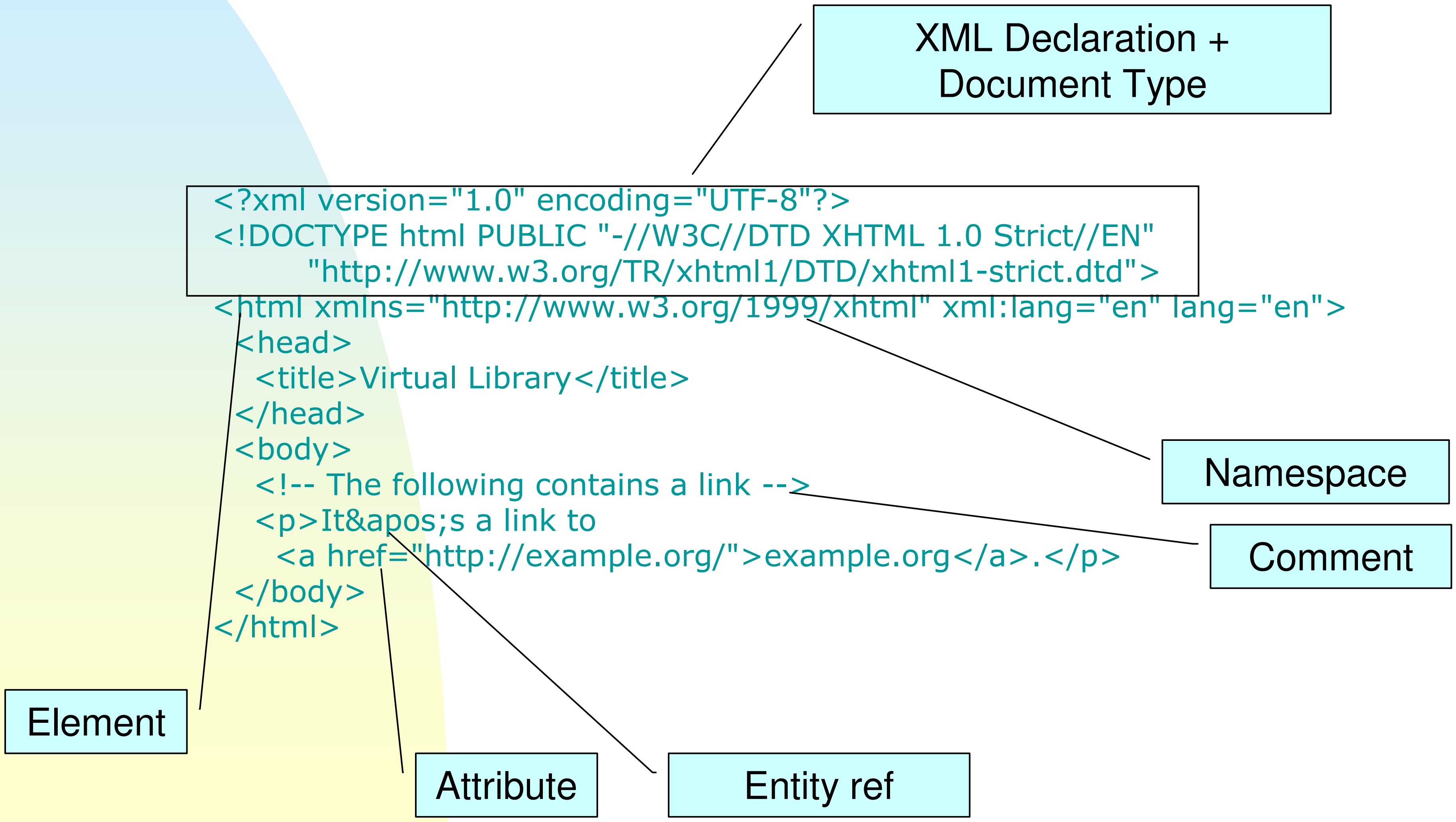
Based on slides by Sasu Tarkoma and Pekka Nikander

Contents

XML

- XML (eXtensible Markup Language) is a framework for defining markup languages
- Standardized by W3C
- Idea: to separate syntax from semantics, custom markup, internationalization, platform independence
- XML Document: prolog, elements, attributes, entity references, comments
- Validated using DTD or schema
- Two things: well-formedness and validity

XML Document Example



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <!-- The following contains a link -->
    <p>It&apos;s a link to
      <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

Related standards

- Namespaces
  - Modular document definition, multiple inheritance, collision avoidance
- XPath (XQuery)
  - Navigation and query of parts of the document
  - E.g. /body/*[10] selects 11th child of <body>
- XSLT
  - Extensible Stylesheet Language Transformation
  - Transformation of documents

Namespaces

- Many documents can have identical element names that denote different things
  - E.g. to diffrentiate <number> of credit card and telephone <number>, put these in different name spaces
  - ◆A **qualified name** is a name subject to namespace interpretation
- In general, a namespace is just a tag
  - ◆An arbitrary string
  - ◆Defined to be a URI
  - ◆Case-sensitive
- A common practice to store a schema / WSDL into the place referenced by the URI
  - ◆Semantics depends on the specific platform
  - ◆Some XML validators use these

Example namespace

```
<x xmlns:edi='http://ecommerce.example.org/schema'>
  <!-- the "edi" prefix is bound to
      http://ecommerce.example.org/schema for the "x"
      element and contents -->
</x>

<x xmlns:edi='http://ecommerce.example.org/schema'>
  <!-- the 'taxClass' attribute's namespace is
  http://ecommerce.example.org/schema -->
  <lineItem edi:taxClass="exempt">Baby food</lineItem>
</x>
```

About Schemas

- XML language for describing and constraining the content of XML documents
- A W3C Recommendation
- Used to specify
  - The allowed structure of an XML document
  - The allowed data types contained in XML documents
  - E.g. a <book> has a <title> and 1 or more <chapter>s
- XML Schema documents are XML documents

XML Schema Example

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="House" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="Town" type="xs:string" />
        <xs:element name="County" type="xs:string" minOccurs="0" />
        <xs:element name="PostCode" type="xs:string" />
        <xs:element name="Country">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="FR" />
              <xs:enumeration value="DE" />
              <xs:enumeration value="ES" />
              <xs:enumeration value="UK" />
              <xs:enumeration value="US" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="utf-8"?>
<Address>
  <Recipient>Mr. Walter C. Brown</Recipient>
  <House>49</House>
  <Street>Featherstone Street</Street>
  <Town>LONDON</Town>
  <PostCode>EC1Y 8SY</PostCode>
  <Country>UK</Country>
</Address>
```

Example source: Wikipedia

Web Services

- Let's make machine-callable services using web principles
- A central role is played by the description of the service's interface
- Implementation less important, avoid implementation-specifics
- Business aspects considered
  - Use across organizations
  - Multiple competing implementations

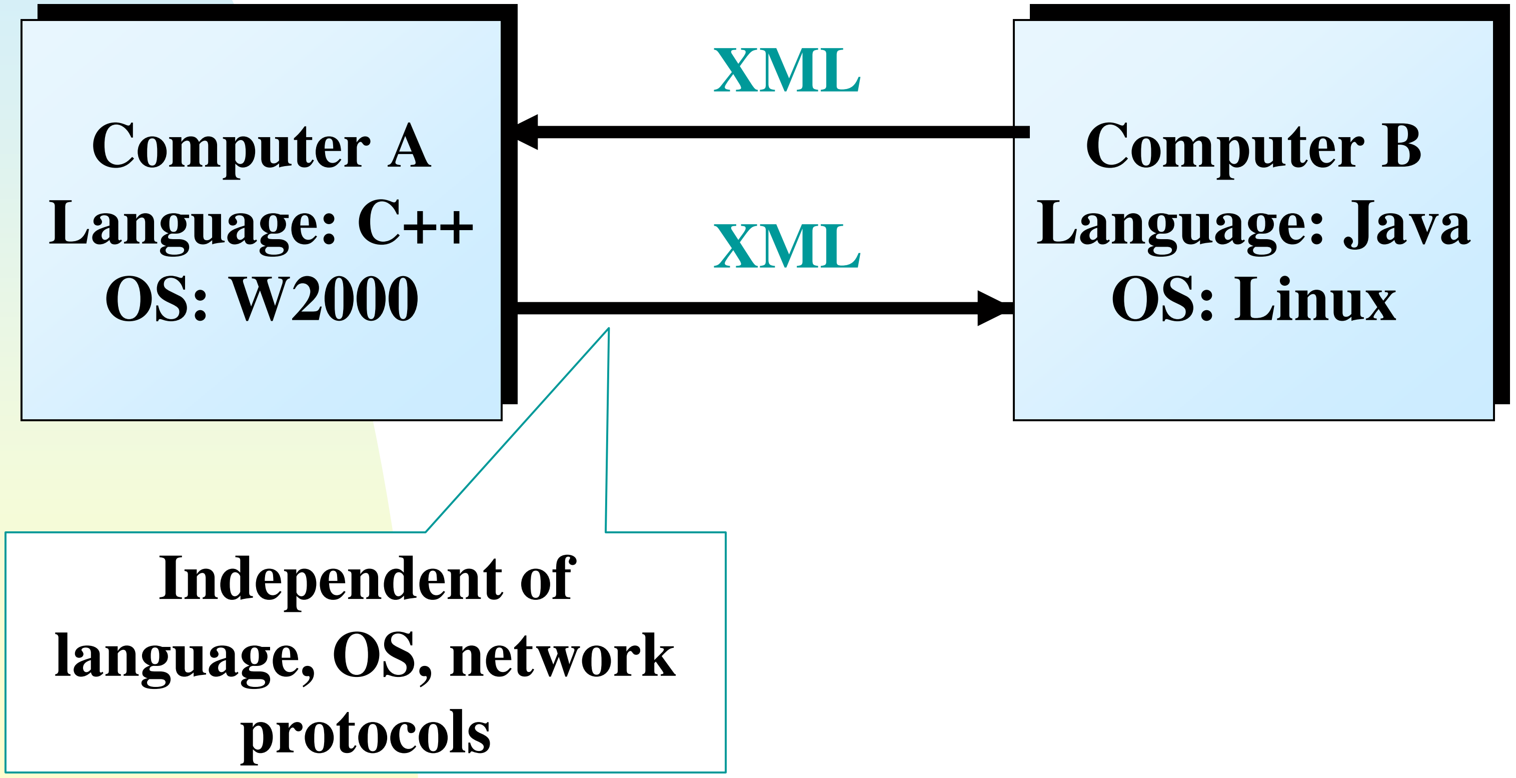
Driving forces I

- Universal data access / representation
  - Independent of OS, programming language, network protocol, …
- Move from human-centric to application-centric web
  - Applications converse with each other and use machine-related information published on the web
  - Application-areas: package tracking, card verification, shopping bots, single sign-on, calendar, email, ...

Driving forces II

- Making Web a programming interface
  - We have had servlets, CGI, CORBA for years
  - Idea is to standardise languages and protocols to have better integration
- Make service composition possible
  - Faster project throughput
  - Better utilization of global resources
  - Cope with heterogeneity
- Deferred binding
  - Discovery / broker, interpret, compose, execute
  - Many levels of deference

A Basic Web Service



Additional properties

- A web service should be self-describing
  - Interface is published with the implementation
  - Minimum interface is human readable description
  - The interface can also be written in a common XML grammar (WSDL)
- A web-service should be discoverable
  - The web service is published
  - It has a life cycle
  - Interested parties can find it
- Not mandatory but desirable properties

Standardization

- W3C Web Services Activity
  - XML Protocol Working Group
    - SOAP
  - Web Services Addressing Working Group
    - How to address WS entities
  - Web Services Choreography Working Group
    - Processes involving several WS, coordination
  - Web Services Description Working Group
    - WSDL
- OASIS
  - UDDI (Universal Description, Discovery and Integration)
- WS-I (Web Service Interoperability Org.)
  - Best Practices on how to use WS* standards

Web Service Architecture

- The three major roles in web services
  - Service provider
    - Provider of the WS
  - Service Requestor
    - Any consumer / client
  - Service Registry
    - logically centralized directory of services
- A protocol stack is needed to support these roles

Web Services Protocol Stack

- Message Exchange
  - Responsible for transporting messages
  - HTTP, BEEP
- XML Messaging
  - Responsible for encoding messages in common XML format
  - XML-RPC, SOAP
- Service Description
  - Responsible for describing an interface to a specific web service
  - WSDL
- Service discovery
  - Responsible for service discovery and search
  - UDDI

WS Protocol Stack

Discovery: UDDI

Description: WSDL

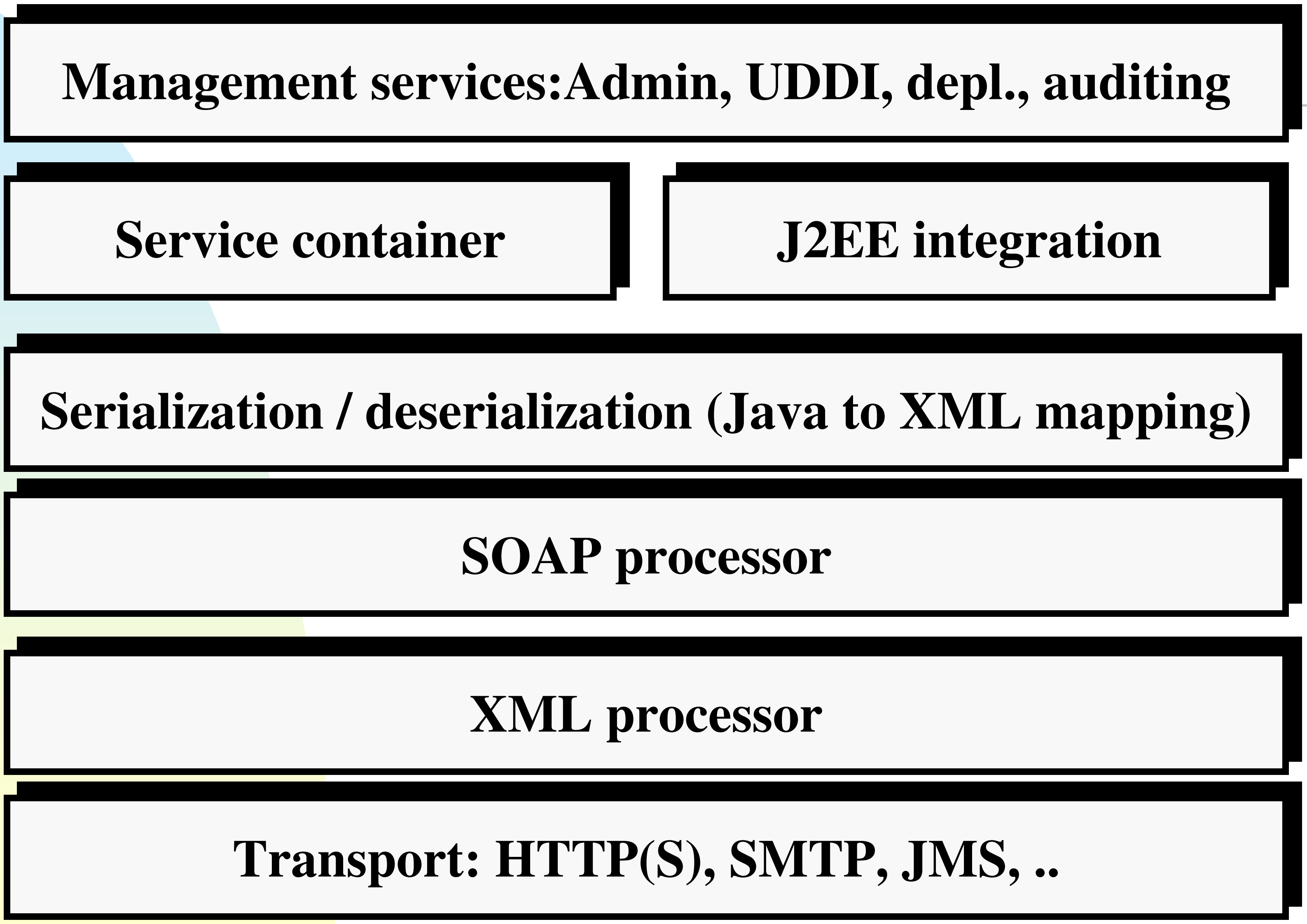
XML Messaging: SOAP, XML-RPC, XML

Transport: HTTP, FTP, BEEP, SMTP, JMS

Main components today

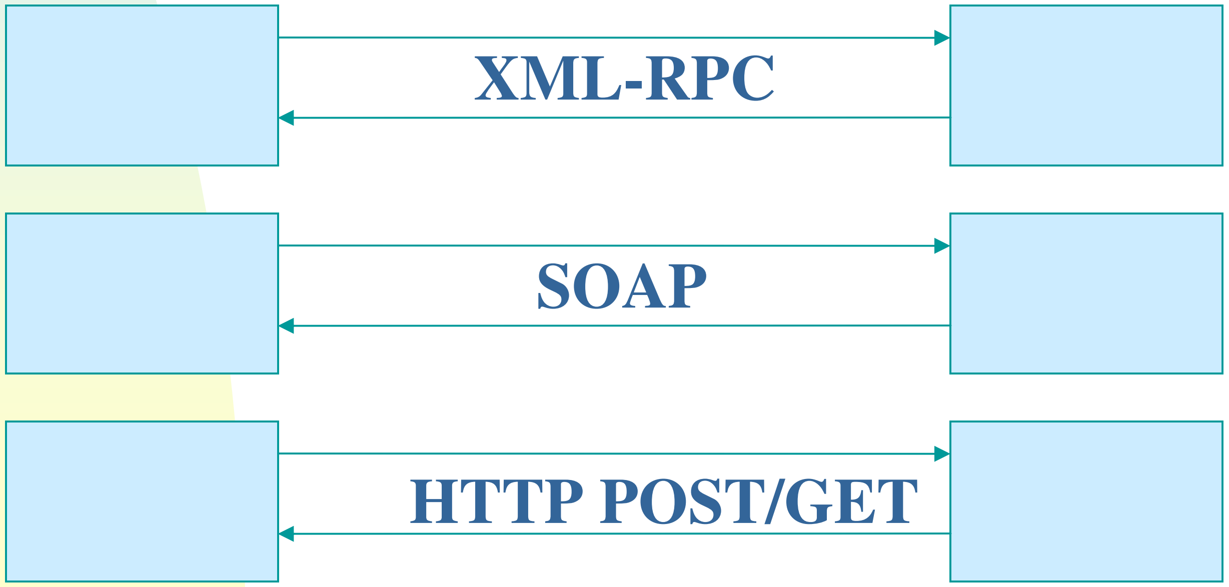
- XML data representation
  - XML Schema Definitions (xsd) for types
  - XML Namespaces for unambiguity
- SOAP
  - Basic transport (XML messaging)
  - Sync / async communication and RPC
- WSDL
  - Description of (SOAP) services
- UDDI
  - Universal Description Discovery and Integration
  - Service registry

Example WS layering


Management services:Admin, UDDI, depl., auditing
Service container
J2EE integration
Serialization / deserialization (Java to XML mapping)
SOAP processor
XML processor
Transport: HTTP(S), SMTP, JMS, ..

XML Messaging

- Several alternatives for XML messaging
  - SOAP
  - XML Remote Procedure calls (XML-RPC)
  - Regular XML transported over HTTP



SOAP Version 1.2

- protocol for exchanging structured (XML) and typed information between peers
- A SOAP message is formally specified as an XML Infoset ("abstract XML")
- Infosets can have different on-the-wire representations, one common example of which is as an XML 1.0 document.
- A stateless, one-way message exchange paradigm
- Applications can create more complex interaction patterns
  - request/response, request/multiple responses

How it could work

- 1. A standard body creates a WSDL interface definition
- 2. A service programmer implements a service according to the WSDL definition
- 3. A client programmer implements a client according to the WSDL definition
- 4. A service provider deploys the service and publishes a WSDL implementation definition, and registers it into UDDI
- 5. A client program pulls WSDL from UDDI, checks conformance, and uses SOAP for access

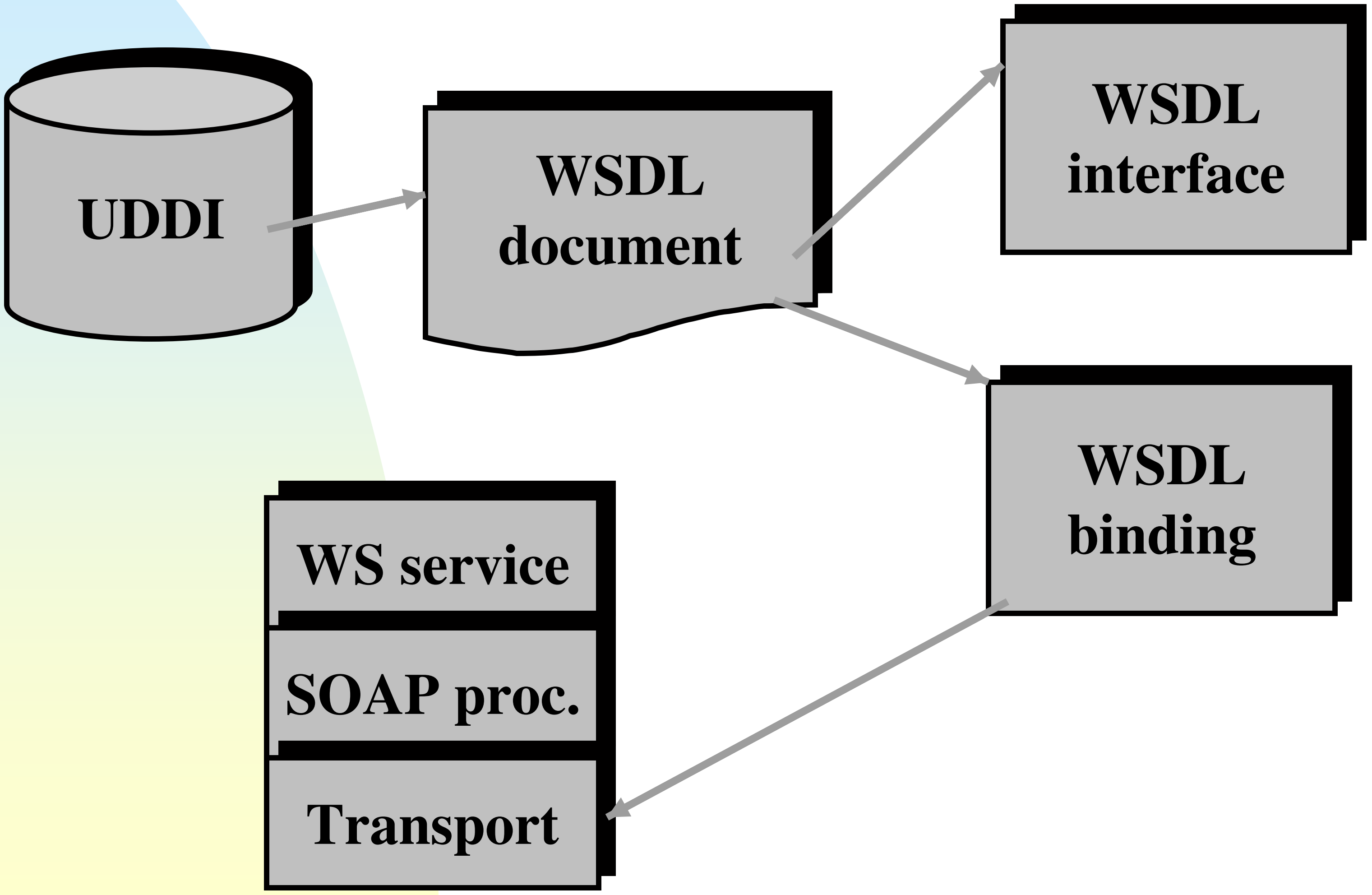
2. Creating server application

- Pull WSDL definition from somewhere (UDDI)
  - Only use high-level WSDL, no bindings yet
- Generate platform specific skeleton code using automated tools
- Write the actual program code

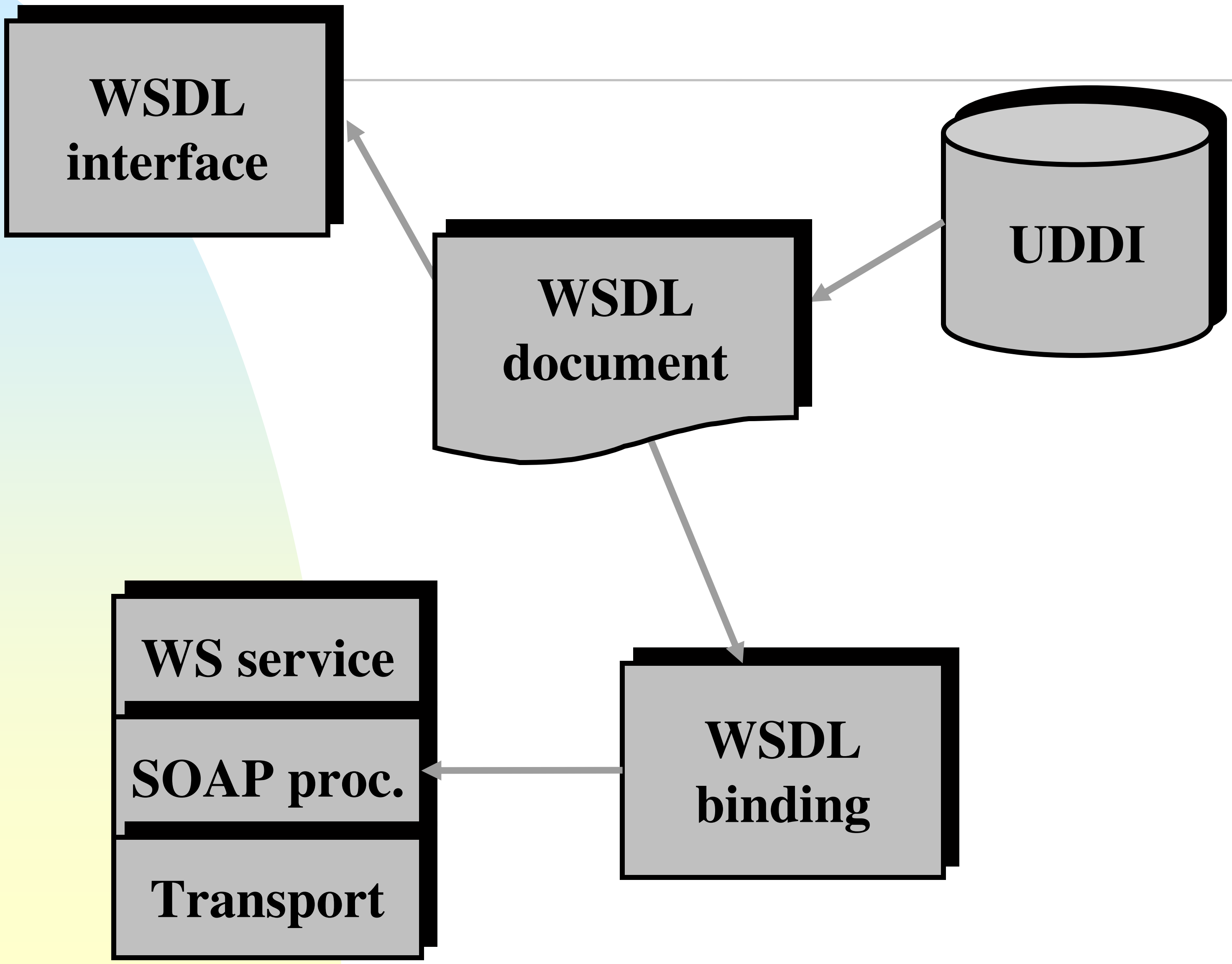
3. Creating client application

- Pull WSDL definition from somewhere (UDDI)
  - Use only high-level WSDL, no bindings yet
- Generate platform specific stub code using automated tools
- Write the actual program code

4. Deploying the service



5. Using a service



About invocation mechanisms

- WS does not define invocation/execution mechanism
- Alternatives
  - Microsoft .NET framework
  - Java-based framework
    - JAVA API for WSDL (JWSDL)
    - JAX-RPC
    - Java API for XML registries (JAXR)
    - Apache Axis
    - ...

What is WSDL?

- WSDL: Web Service Description Language
- An XML language used to describe and locate web services
  - location of web service
  - methods that are available
  - data type information
- Commonly used to describe SOAP-based services
- W3C standard (work in progress)
  - Initial input: WSDL 1.1 as W3C Note
  - Current version 2.0 (Recommendation)
  - Some differences between 1.1 and 2.0

WSDL Document Elements (v. 2.0)

- **<types>** - data type definitions
- **<interface>** - A set of abstract operations
- **<binding>** - Concrete protocol and data format specifications for the operations and messages defined by a particular interface. Endpoint type.
- **<endpoint>** - An address for binding. Endpoint instance.
- **<service>** - A set of endpoints

WSDL Overview

<definitions>: ROOT WSDL element

<types>: The data types that are used

<interface>: The supported operations

<binding>: The binding to concrete protocols

<service>: Reference to actual location

A WSDL Document

- A WSDL document contains two parts
- Abstract part
  - Interfaces, types
- Concrete part
  - Binding to concrete protocol and encoding
- May be published separately
  - Interfaces agreed between many companies
  - Each company published their own implementation in UDDI and import the abstract interface.

The main structure (v. 2.0)

```
<definitions namespace = "http://..">
          <types>XML Schema types</types>
           <interface>a set of operations</interface>
           <binding>Communication protocols</binding>
           <service>A list of binding and endpoints</service>
</definitions>
```

Types

- <types> define data types used in interface declaration
- For platform neutrality, WSDL uses XML Schema syntax to define data
  - XML Schema must be supported by any vendor of WSDL conformant products
  - Other kinds of type definitions also possible
    - Possible interoperability issues
  - If the service uses only XML Schema built-in simple types, such as strings and integers, the types element is not required

WSDL Interfaces

- The <interface> element is the most important WSDL element
- The operations that can be performed
- An <endpoint> defines the connection point to a web service, an instance of <interface>
- It can be compared to a function library (or a module, or a class) in a programming language

Message Exchange Patterns

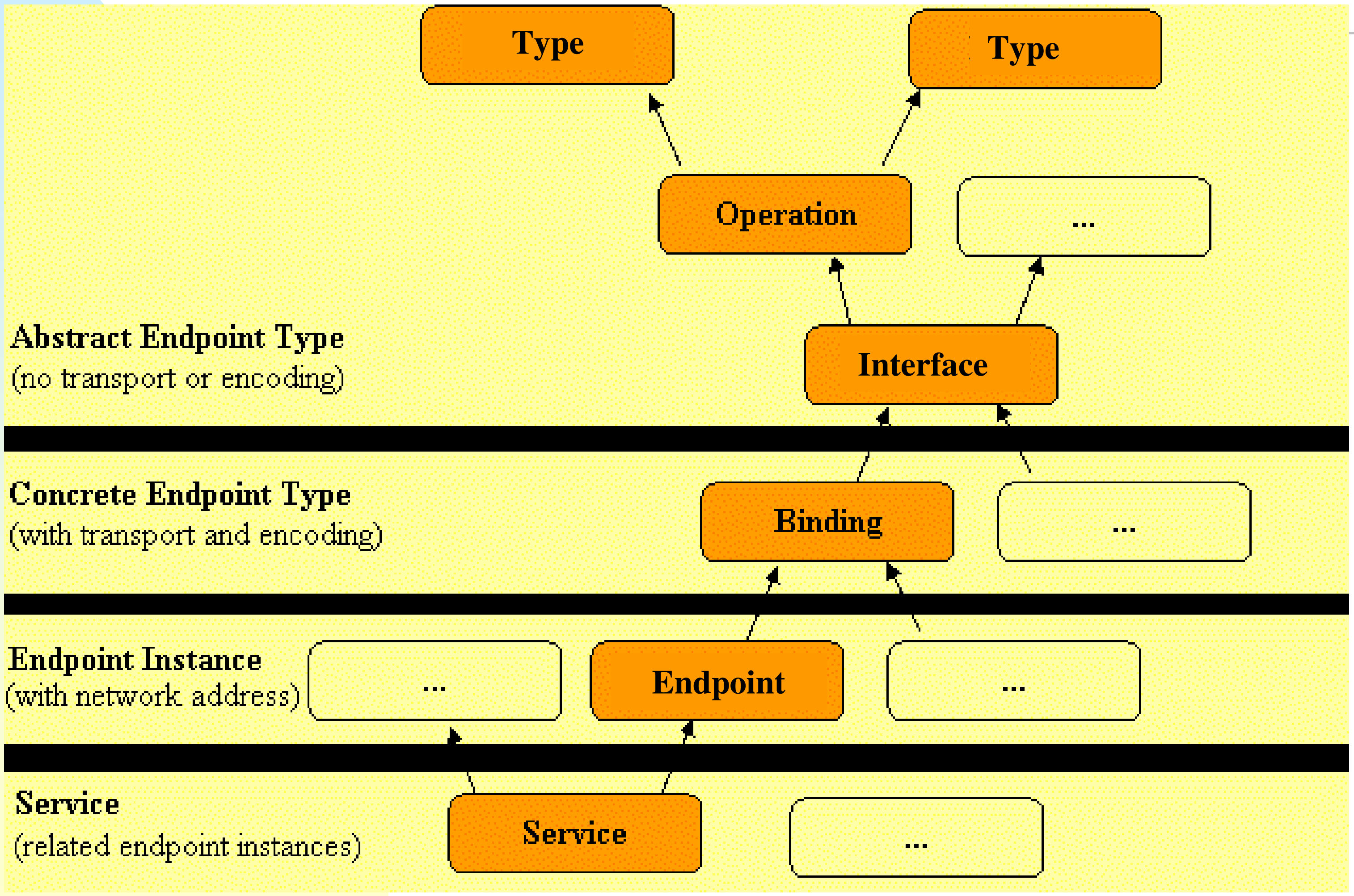
- In-Only: The operation can receive a message but will not return a response
- In-Out: The operation can receive a request and will return a response
- Out-In: The endpoint sends an output message and then receives an input message.
- Out-Only: The endpoint can send a message but will not wait for a response
- Robust- variants: faults can occur

Example

```
<types>
   <xsd:schema
         xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
     <xsd:element name="TradePriceRequest" type="xsd:string"/>
     <xsd:element name="TradePrice" type="xsd:float"/>
</types>

<interface name="StockQuote">
   <operation name="GetLastTradePrice"
                  pattern="http://www.w3.org/ns/wsdl/in-out ">
     <input message="In" element="TradePriceRequest" />
     <output message="Out" element="TradePrice "/>
   </operation>
</interface>
```

Putting it together



Original source: http://msdn.microsoft.com/

WSDL 2.0

- 3 specifications
- Part 1: Core
  - Abstract interfaces, independent of protocol and encoding
- Part 2: Message Exchange Patterns
  - Predefined types of interactions
- Part 3: Bindings
  - SOAP and HTTP/MIME binding
- Lots of changes from 1.1

Changes from 1.1 to 2.0

- Adding further semantics to the description language.
- Removal of message constructs.
- No support for operator overloading.
- PortTypes renamed to interfaces. Support for interface inheritance is achieved by using the extends attribute in the interface element.
- Ports renamed to endpoints.
- Abstracted message patterns

Implementations

- Microsoft .NET
  - SOAP 1.1, WSDL 1.1
- Java API for XML-based RPC (JAX-RPC)
  - SOAP 1.1, WSDL 1.1
- Java API for XML Registries (JAXR)
- Apache Axis
  - Tools for automatically creating WSDL to/from Java
  - SOAP 1.1 and 1.2, WSDL 1.1, JAX-RPC 1.0
  - Axis2 supports WSDL 2.0 component model
- IBM Websphere, BEA Web Logic, SAP NetWeaver, …

Uses of WSDL documents

- Description of service interfaces
  - Compile-time
    - Developer uses WSDL before service deployment
  - Run-time
    - Client downloads WSDL description and uses the info it provides to execute the service
- As a side-effect
  - Developers can use WSDL to speed up the development of code
  - WSDL→Java code
  - Java interfaces → WSDL

WS Criticism

- Quite heavy, lots of specs
- More suited for large well-structured organizations, rather than fast innovation?
- Is there anything new here?
- Are all abstraction layers really needed?
- REST

Summary

- Web Services: let's make machine-callable services using web principles
- XML as universal syntax
- Web Services Stack
- Is all this really necessary?

Extra: WSDL 1.1 Messages

- The <message> element defines the data elements of an operation
  - the name of the message
  - contains zero or more message part elements
- The parts can be compared to the parameters of a function call in a traditional programming language
- All parts are typed by an XML Schema type
- V.2.0 changes: message element is not used, replaced by patterns