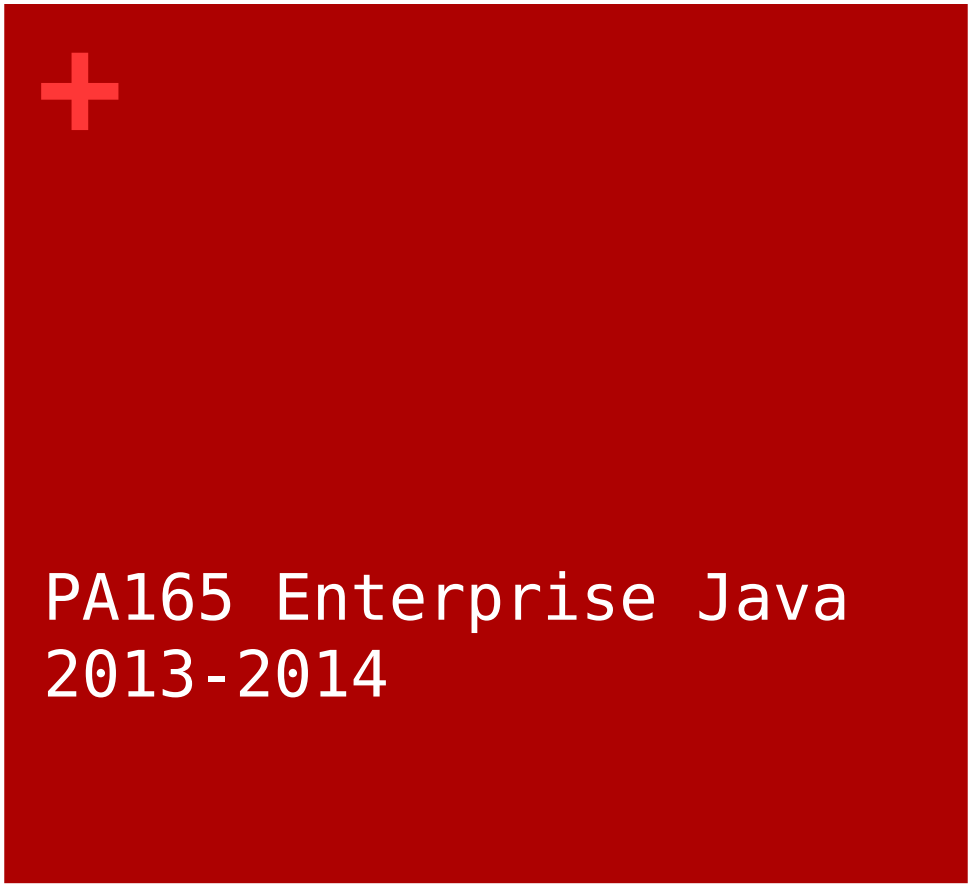


PA165 Enterprise Java
2013-2014

SOAP and WS* Webservices

Bruno Rossi & Juha Rikkilä

SOAP, in general terms

- Originally acronym for **Simple Object Access Protocol**, now a common name
- A **communication protocol**, designed to communicate via Internet
- Extends **HTTP for XML messaging**
- Provides data transport for Web services (SOAP, WSDL, UDDI)
- Exchanges **complete documents** or **call a remote procedure**
- Is used for **broadcasting a message**
- Is platform and **language independent**
- Is the XML way of defining what information gets sent and how

https://kore.fi.muni.cz/wiki/index.php/PA165/WebServices_(English)

XML (Extensible Markup Language)

- created to structure, store, and transport data by defining a set of rules for encoding documents
- a format that is both human-readable and machine-readable
- defined in the XML 1.0 Specification produced by the W3C
- there are two current versions of XML.
  - XML 1.0, currently in its fifth edition, and still recommended for general use
  - XML 1.1, not very widely implemented and is recommended for use only by those who need its unique features

XML

- emphasizes simplicity, generality, and usability over the Internet
- textual data format with Unicode support for the languages of the world
- focuses on documents, but is widely used for the representation of arbitrary data structures, for example in web services.
- many APIs for processing XML data
- several schema systems
- hundreds of XML-based languages (e.g. RSS, Atom, SOAP, and XHTML)
- the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork

XML, markup and content

- **markup**
  - either begin with the character < and end with a >,
  - or they begin with the character & and end with a ;
- strings of characters that are not markup are content
- **tag**, a markup construct that begins with < and ends with >
  - start-tags; for example: <section>
  - end-tags; for example: </section>
  - empty-element tags; for example: <line-break />
- **element**, begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag.

Schema and validation

- well-formed, and may be valid.
  - Document contains a reference to DTD,
  - DTD declares elements and attributes, and specifies the grammatical rules
- XML processors
  - re validating or non-validating
  - If error discovered it is reported, but processing may continue normally
- schema languages constrain
  - the set of elements in a document,
  - attributes that are applied to them,
  - the order in which they appear,
  - the allowable parent/child relationships

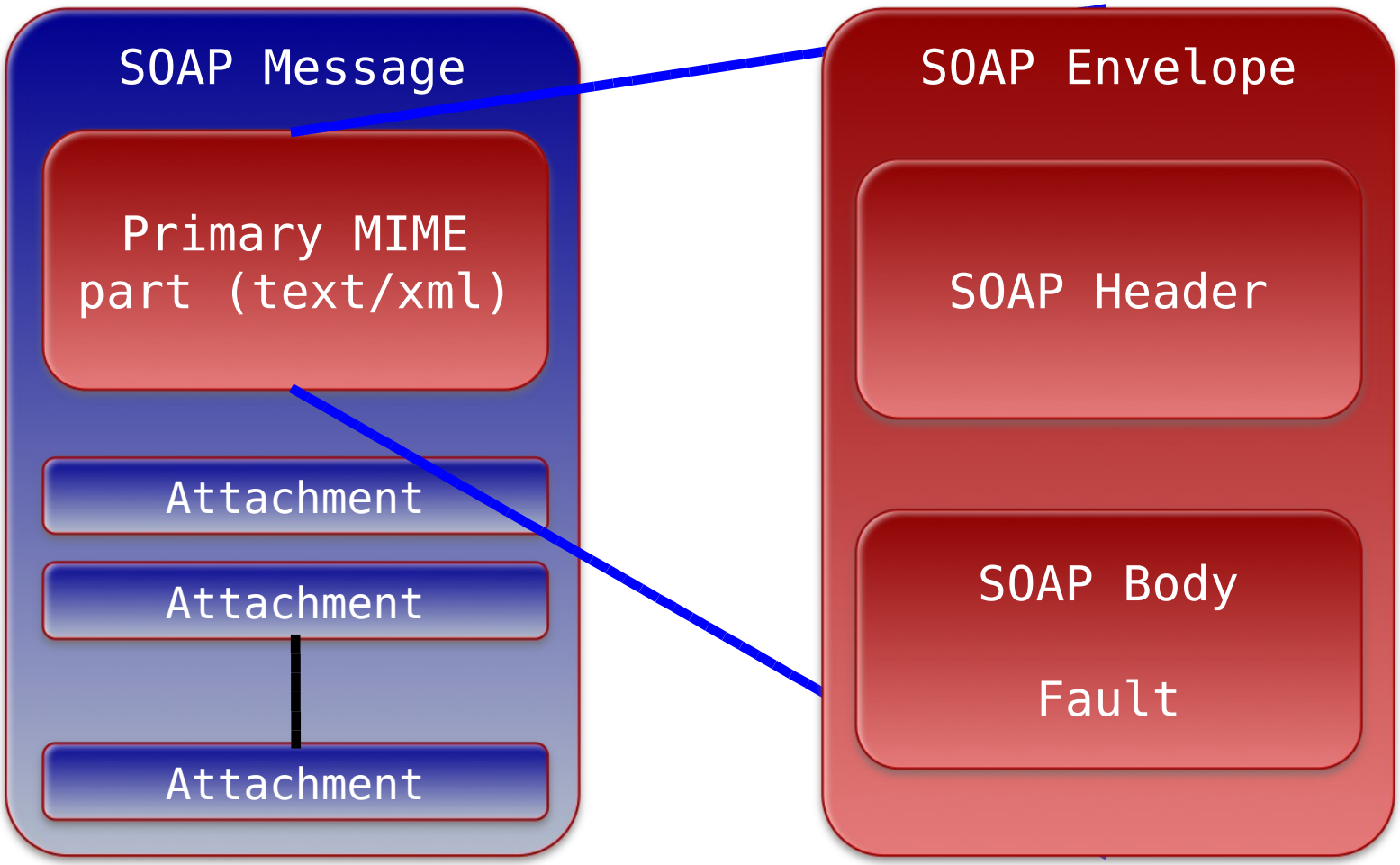
XML Schema

- schema language, described by the W3C
  - (successor of DTD = Document Type Definition)
  - XML schema is more powerful than DTDs
- often referred to as XSD (XML Schema Definition)
- XSDs use an XML-based format, so XML tools can be used process them.

XML Messaging

- SOAP 1.1 defined:
  - An **XML envelope for XML messaging**:
    - Headers + body.
  - An **HTTP binding for SOAP messaging**:
    - SOAP is “transport independent”.
  - A **convention for doing RPC**,
  - An **XML serialization format for structured data**.
- SOAP Attachments adds:
  - How to carry and reference data attachments using in a MIME envelope and a SOAP envelope.

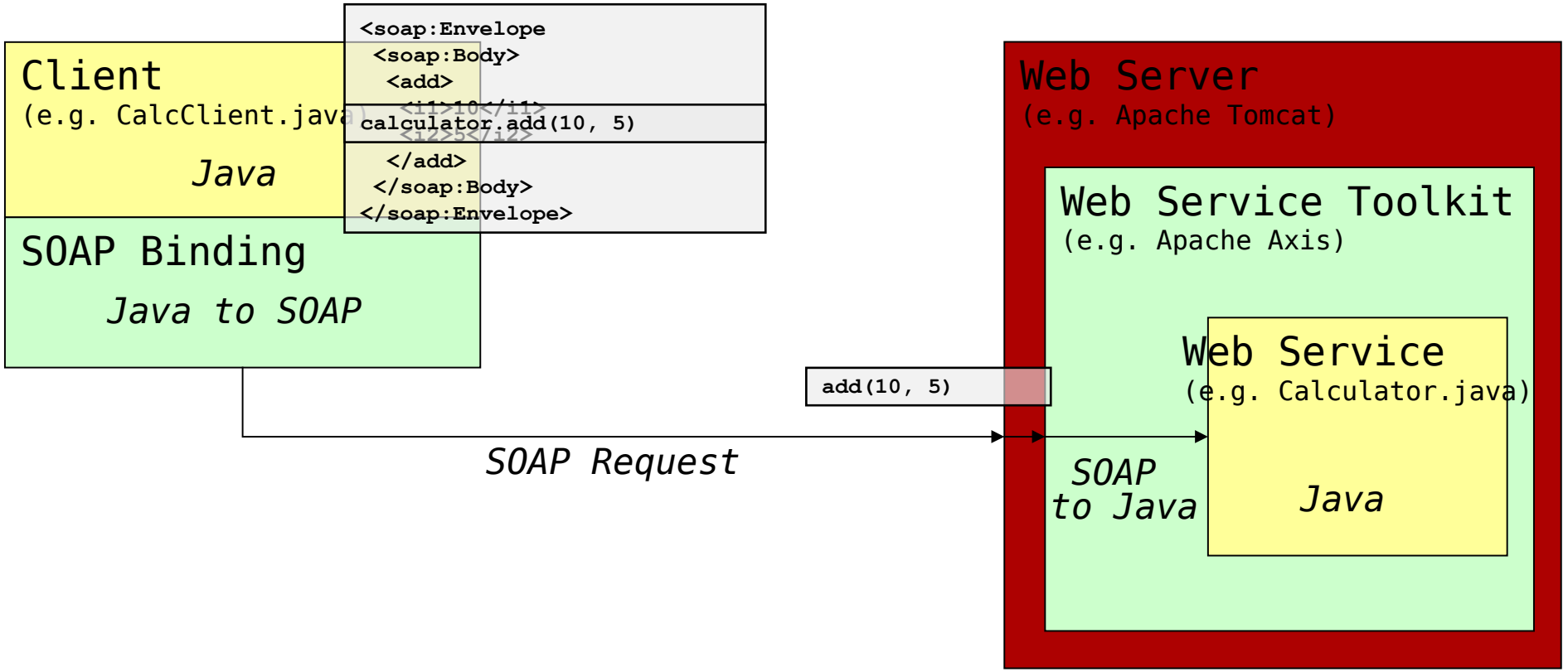
SOAP Message



SOAP Message Envelope

- Encoding information
- Header
  - Optional
  - Contains context knowledge
    - Security
    - Transaction
- Body
  - Methods and parameters
  - Contains application data

SOAP in practice, an animated example (Java)



A SOAP Request


```
POST /temp HTTP/1.1
Host: www.somewhere.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
SOAPAction: "http://www…../temp"

<?xml version="1.0"?>
……..
```

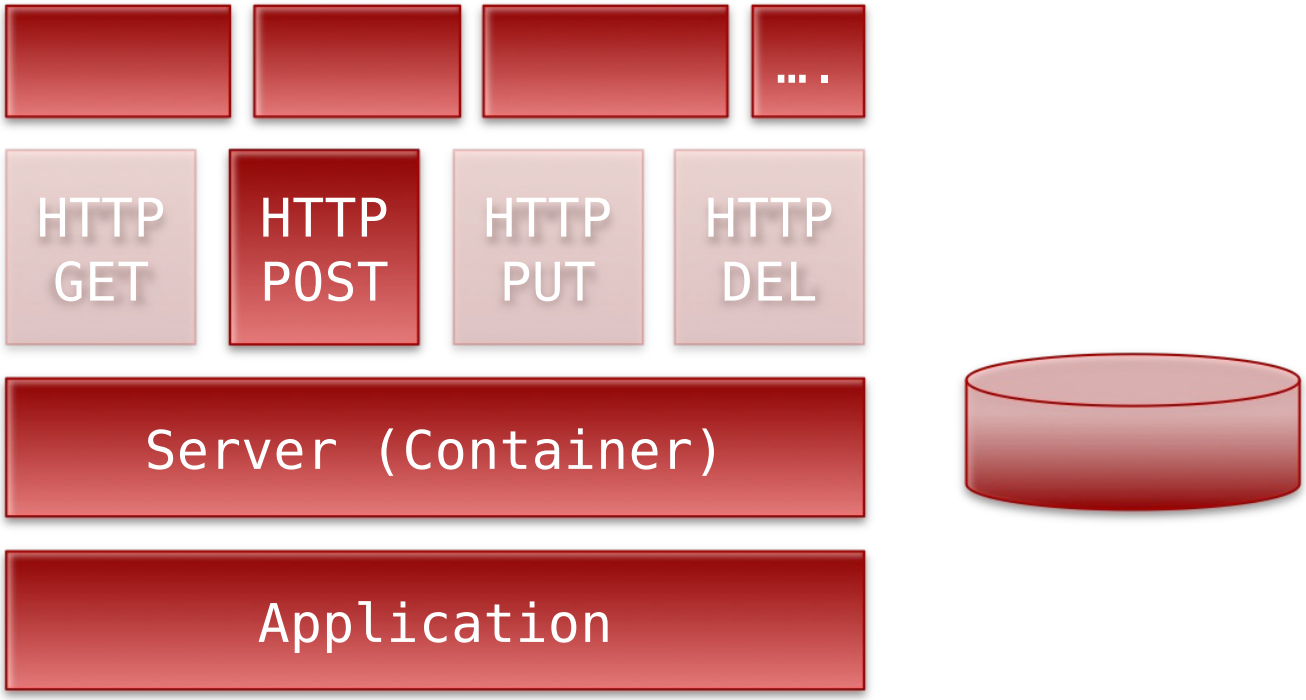
HTTP headers and the blank line

an XML document

"The SOAPAction HTTP request header field can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request."
Note: in SOAP 1.2, the SOAPAction header has been replaced with the "action" attribute on the application/soap+xml media type (Content-Type: application/soap+xml; charset=utf-8). But it works almost exactly the same way as SOAPAction.

SOAP

- Uses only POST over HTTP
- Container parsing and interpreting



XML message structure

```
<?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-encodi
ng">

    <soap:Header>
        ...
    </soap:Header>

    <soap:Body>
        ...
        <soap:Fault>
            ...
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

Version number

soap-encoding !?!

SOAP encoding

- When SOAP specification was written for the first time, XMLSchema was not available, so a common way to describe messages was defined.
- Now SOAP encoding defines it's own namespace as http://schemas.xmlsoap.org/soap/encoding/ and a set of rules to follow.
- Rules of expressing application-defined data types in XML
- Based on W3C XML Schema
- Simple values
  - Built-in types from XML Schema, Part 2 (simple types, enumerations, arrays of bytes)
- Compound values
  - Structures, arrays, complex types

http://www.tutorialspoint.com/soap/soap_encoding.htm

SOAP, “closer to the bit space”

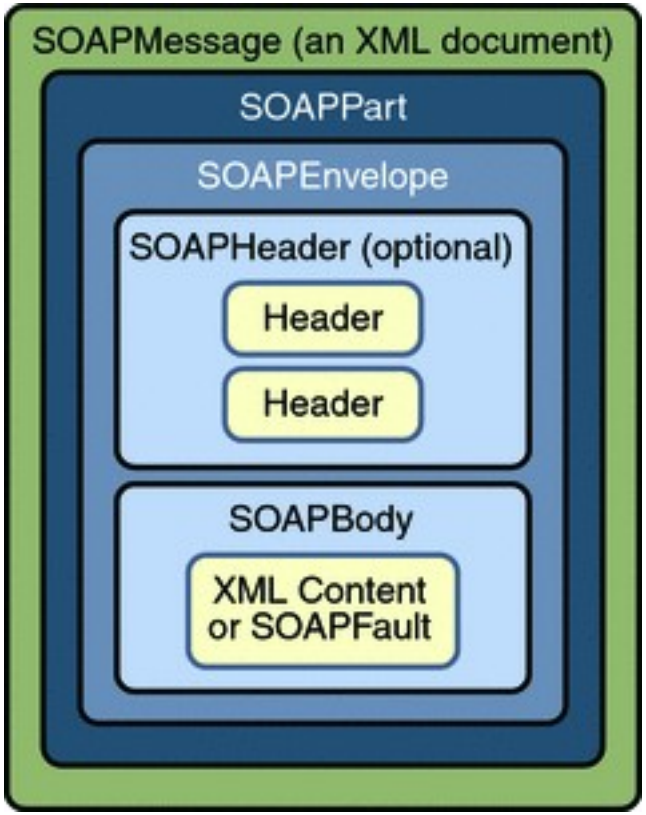
- Summing up:
- SOAP, originally defined as **Simple Object Access Protocol**, is a protocol specification for exchanging structured information in the implementation of **Web Services** in computer networks. It relies on Extensible Markup Language (XML) for its message format, and usually relies on other **Application Layer** protocols, most notably **Hypertext Transfer Protocol (HTTP)** and Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.
- SOAP can form the **foundation layer of a web services protocol stack**, providing a basic messaging framework upon which web services can be built.

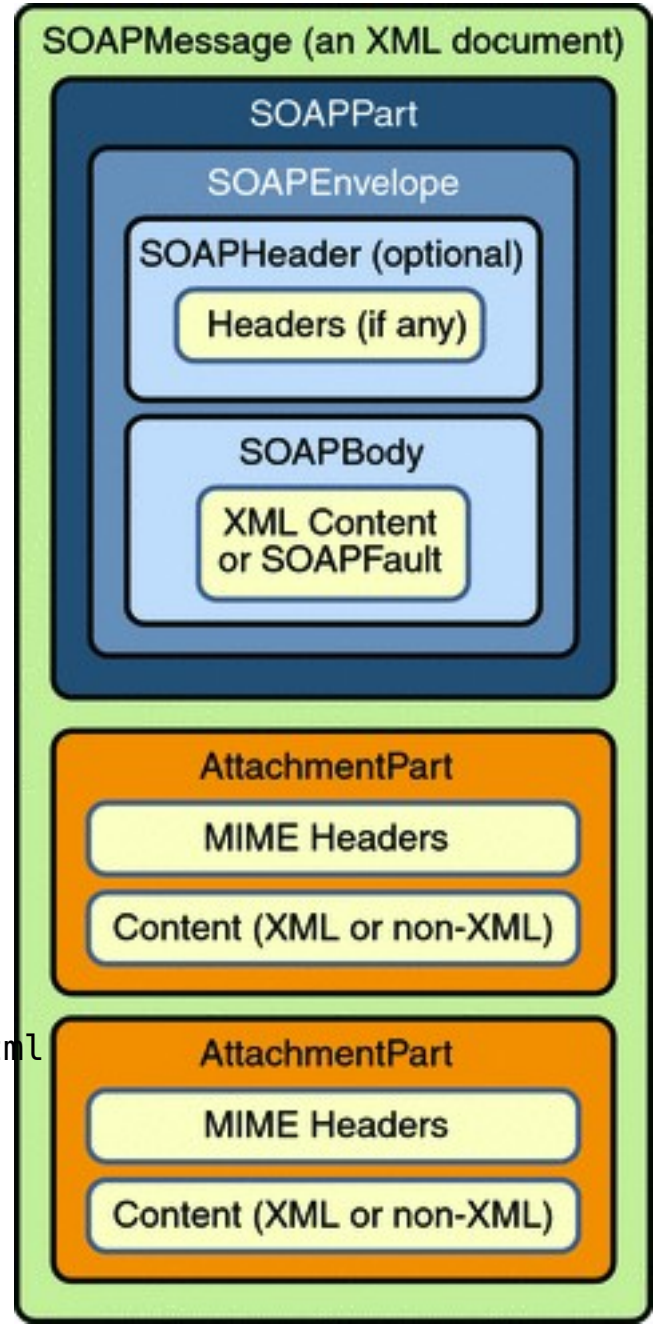
SOAP with Attachments,
SOAP with Attachments API for Java (SAAJ)

- **SOAP with Attachments (SwA)** or MIME for Web Services refers to the method of using Web Services to send and receive files using a combination of SOAP and MIME, primarily over HTTP.
- Note that SwA is not a new specification, but rather a **mechanism for using the existing SOAP and MIME facilities to perfect the transmission of files using Web Services invocations**.
- The **SOAP with Attachments API for Java** or **SAAJ** provides a standard way to send XML documents over the Internet from the Java platform.
- **SAAJ** enables developers to produce and consume messages conforming to the SOAP 1.1 specification and SOAP with Attachments note.
- Developers can also use it to **write SOAP messaging applications directly instead of using JAX-RPC (obsolete) or JAX-WS**
- This will mean working with the lower details → so more control but more possibilities for mistakes

The SOAP with Attachments API Version 1.3

- The essential object for using **SAAJ** is a **SOAPMessage** object created by a call to the **createMessage()** method of **MessageFactory**.
- **SOAPMessage** object contains a complete SOAP message, either a SOAP-formatted XML document or a MIME multipart message whose first section is an XML document.
- XML is contained in a **SOAPPart**, all **SOAPMessages** contain a single **SOAPPart**, which in turn contains a **SOAPEnvelope** corresponding to the **root element of the document**.
- Inside the **Envelope** element, a SOAP message is required to have a **Body** element and **may have one Header element**. SAAJ provides the **SOAPHeader** and **SOAPBody** objects to enable the programmer to manipulate the content of these elements. SAAJ just provides the mechanism, actually creating the contents of the SOAPBody and SOAPHeader is up to the programmer.
- A **SOAPMessage object may have zero, one or many additional AttachmentPart objects with any MIME content type such as an XML document, plain text or an image**. Attachments are added using the **AttachmentPart** class, which **requires a data source**, typically an **InputStream**, and a **MIME content type**.





SOAP with Attachments API for Java
The Java EE 5 Tutorial
http://docs.oracle.com/javaee/5/tutorial/doc/bnbhf.html

Creating a SOAP Conection

```
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPConnection;
import javax.xml.soap.MessageFactory;
…….

public SimpleSAAJ {
    public static void main(String args[]) {
       try {
           //Create a SOAPConnection
           SOAPConnectionFactory factory =
                    SOAPConnectionFactory.newInstance();

           SOAPConnection connection =
                    factory.createConnection();

           ..................
           // Close the SOAPConnection
           connection.close();

       } catch (Exception e) {
           System.out.println(e.getMessage());
       }
  }
}
```

Creating a SOAP Message

```
……
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPBody;
import java.net.URL;
……

            //Create a SOAPMessage
            SOAPMessageFactory messageFactory =
MessageFactory.newInstance();
            SOAPMessage message = messageFactory.createMessage();
            SOAPPart  soapPart = message.getSOAPPart();
            SOAPEnvelope envelope = soapPart.getEnvelope();
            SOAPHeader header = envelope.getHeader();
            SOAPBody body = envelope.getBody();
            header.detachNode();
```

Populate a SOAP Message

```
//Create a SOAPBodyElement
 Name bodyName = envelope.createName("GetLastTradePrice"
                 "m", "http://wombat.ztrade.com");
SOAPBodyElement bodyElement = body.addBodyElement(bodyName);

//Insert Content
  Name name = envelope.createName("symbol");
  SOAPElement symbol = bodyElement.addChildElement(name);
  symbol.addTextNode("SUNW");
```

This will produce the SOAP envelope:

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="http://wombat.ztrade.com">
      <symbol>SUNW</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

That you can send with

```
java.net.URL endpoint = new URL("http://wombat.ztrade.com/quotes");
SOAPMessage response = connection.call(message, endpoint);
```

Some Remarks

- SOAP is not *“what it used to be”*, the name remained, but the content has changed
- **SOAP term** is often used as **synonym** for **WS* web service architecture**, though it is one element of it
- **SOAP is not just one element of WS***, it is used in other context as well, even parallel with ReST web services.
- **SOAP is often hidden from the developer**, build into tools in such a way that developer does not have to deal with it at a detailed level.

The Standardization Process

OASIS Organization for the Advancement of Structured Information Standards

- a **global consortium** that **drives** the development, convergence, and adoption of e-business and web service standards

- the work categories: Web Services, e-Commerce, Security, Law & Government, Supply Chain, Computing Management, Application Focus, Document-Centric, XML Processing, Conformance/Interoperability, and Industry Domains

- Standards: BCM, CAM, CAP, CIQ, DSS, DocBook, DITA, DSML, ebXML, EDXL, EML, KMIP, OpenDocument, SAML, SDD, SOAP-over-UDP, SPML, UBL, UDDI,WSDM, XRI, XDI, WS-BPEL, WSRF, WSS, XACML

OASIS
Advancing open standards for the information society

W3C World Wide Web Consortium

- the **main international standards organization for the World Wide Web**
- the **consortium** is made up of **member organizations working together in the development of standards for the World Wide Web**.

the World Wide Web Consortium (W3C) has 390 Members (2 December 2013).

- Standards: CGI, CSS, DOM, GRDDL, HTML, MathML, OWL, P3P, RDF, SISR, SKOS, SMIL, SOAP, SPARQL, SRGS, SSML, SVG, VoiceXML, XHTML, XHTML+Voice, XML, XML Events, XML Information Set, XML Schema, Xpath, Xquery, XSL-FO, XSLT, WCAG, WSDL, XForms

WS-I
(Web Services Interoperability Organization)

- the (WS-I) is an open industry organization
  - chartered to establish Best Practices for Web services interoperability,
  - for selected groups of Web services standards, across platforms, operating systems and programming languages.
- WS-I comprises of
  - Web services leaders from a wide range of companies
  - standards development organizations
- WS-I committees and working groups create Profiles and supporting Testing Tools based on Best Practices for selected sets of Web services standards. The Profiles and Testing Tools are available for use by the Web Services community to aid in developing and deploying interoperable Web services.

WSIT (Web Services Interoperability Technology)

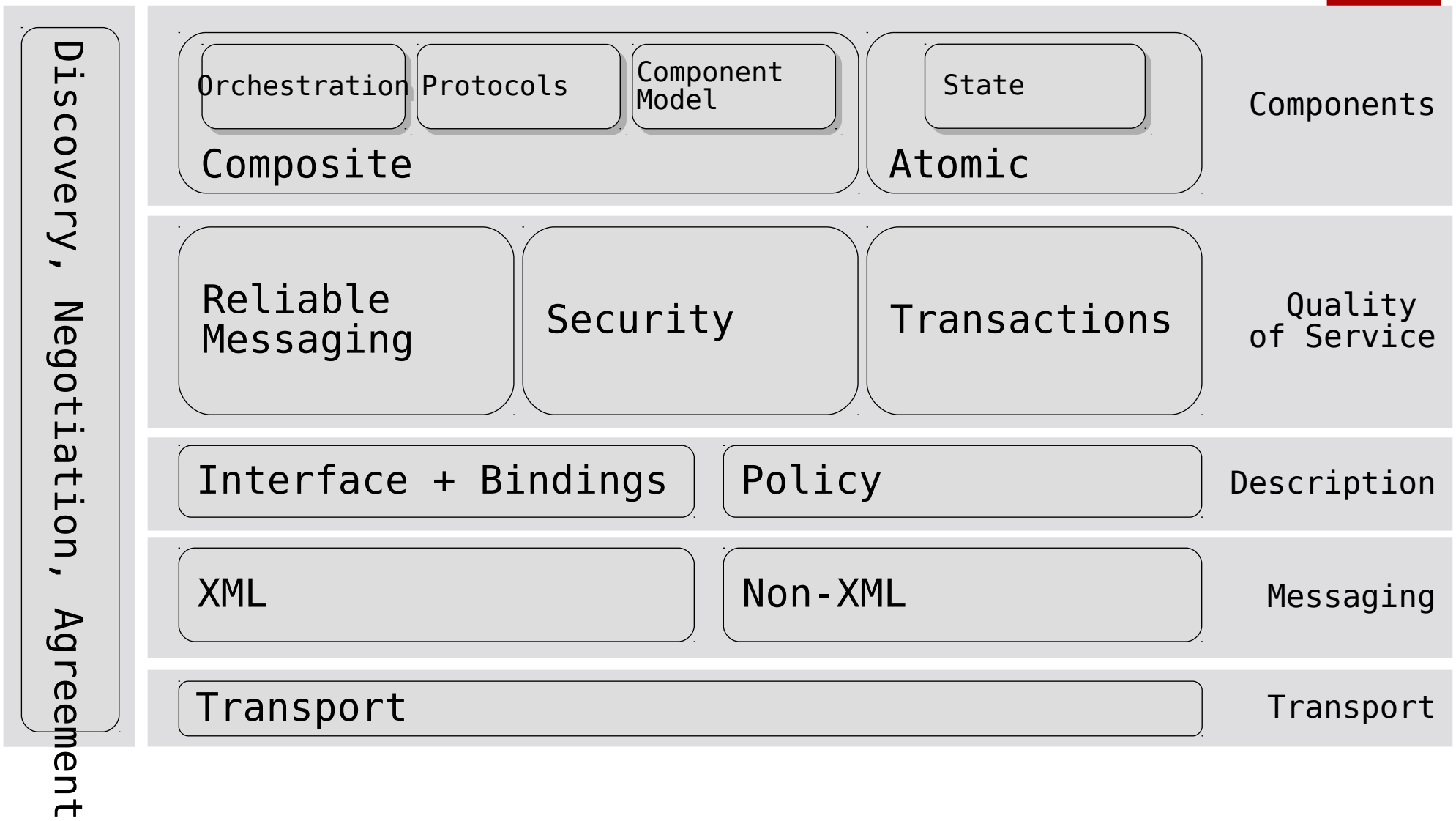
- An open-source project to develop the next-generation of Web service technologies.
- It consists of Java programming language APIs that enable advanced WS-* features to be used in a way that is compatible with Microsoft's .NET.
- The interoperability between different products is accomplished by implementing a number of Web Services specifications, like JAX-WS that provides interoperability between Java Web Services and Microsoft Windows Communication Foundation
- WSIT is a series of extensions to the basic SOAP protocol, and so uses JAX-WS and JAXB (Java API for XML Binding)

WSIT implements the WS-I specifications

- Metadata
  - WS-MetadataExchange
  - WS-Transfer
  - WS-Policy
- Security
  - WS-Security
  - WS-SecureConversation
  - WS-Trust
  - WS-SecurityPolicy
- Messaging
  - WS-ReliableMessaging
  - WS-RMPolicy
- Transactions
  - WS-Coordination
  - WS-AtomicTransaction

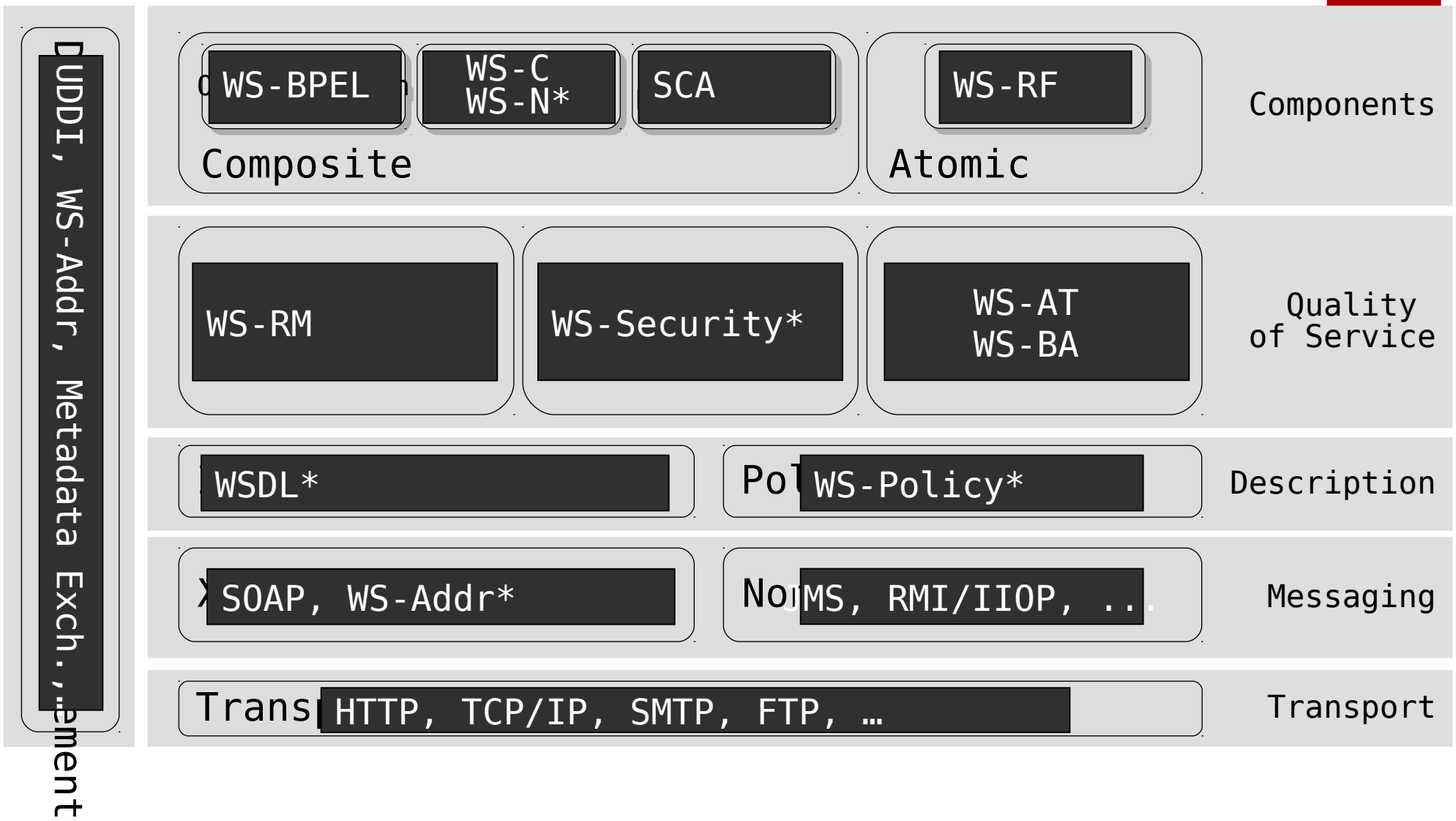
Web Services Standards for SOA

The Web Services Platform Architecture



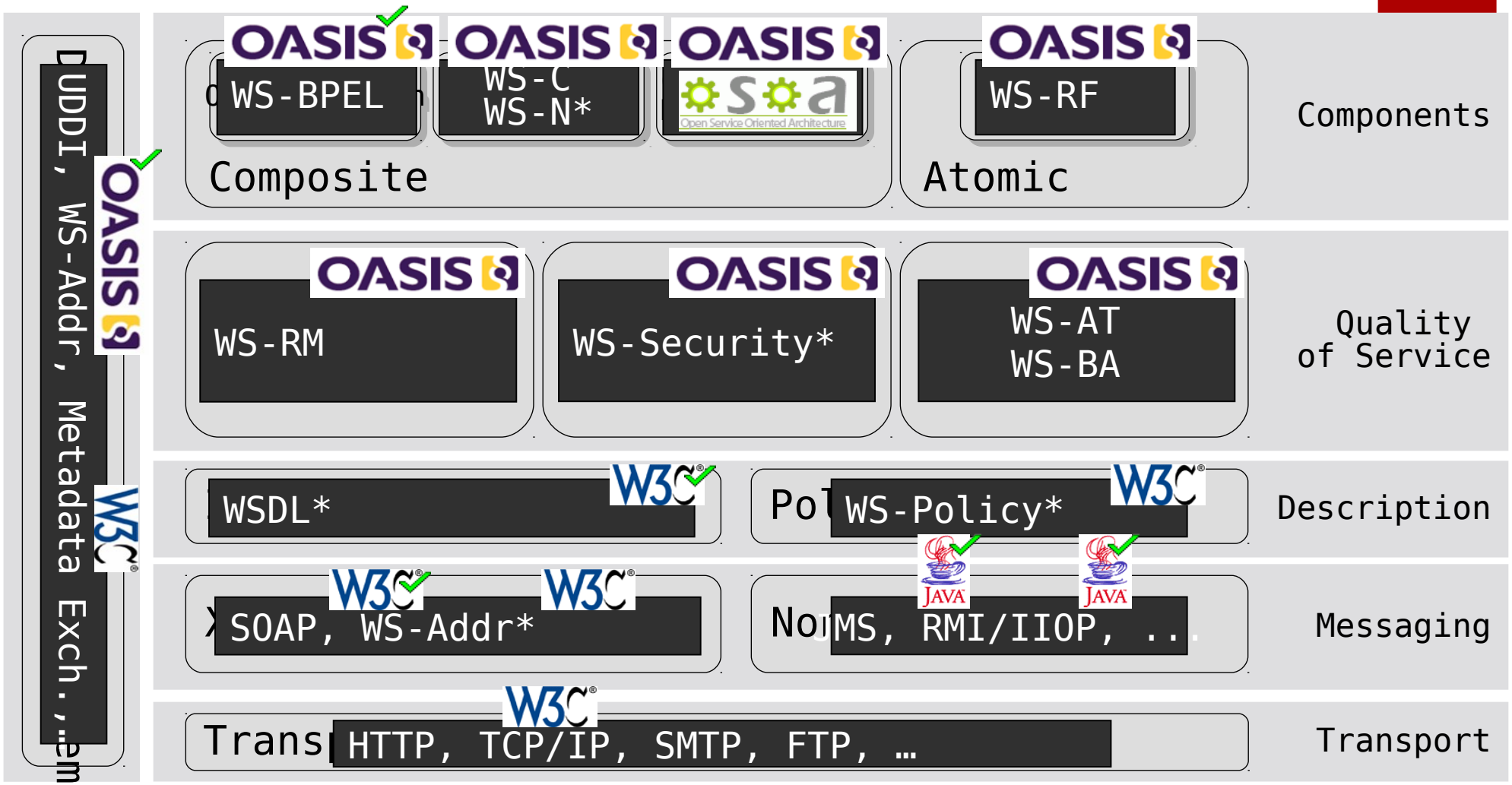
Web Services Standards for SOA

The Web Services Platform Architecture



Web Services Standards for SOA

The Web Services Platform Architecture



W3C Definition of Web Services

A Web service is a software system designed to support **interoperable machine-to-machine interaction over a network**. It has an interface described in a **machine processable format** (specifically **WSDL**). Other systems interact with the Web service in a manner prescribed by its description using **SOAP messages**, typically conveyed using **HTTP with an XML** serialization in conjunction with **other Web-related standards**.

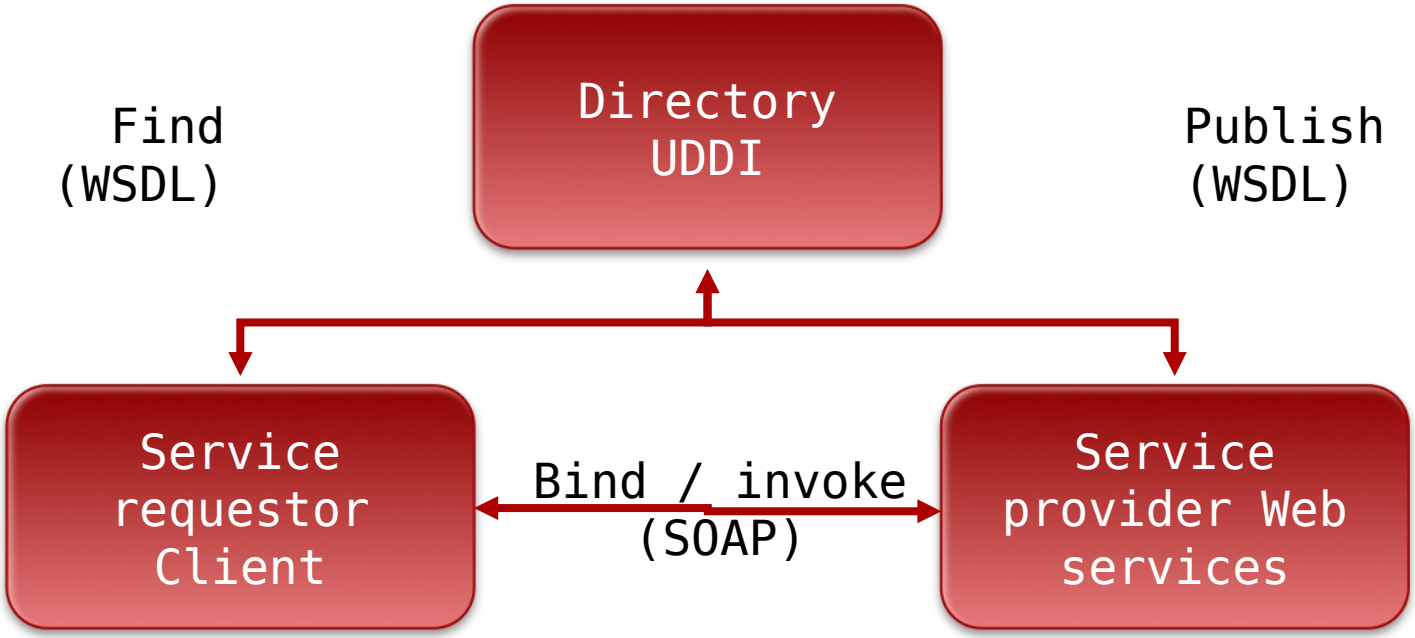
WS* Web Services

- A Web consisting of Services
- Application-to-Application, Machine-to Machine communication
- Standard protocols
  - SOAP (Simple Object Access Protocol)
  - WSDL (Web Services Description Language)
  - UDDI (Universal Description, Discovery, and Integration)

Web Services

- Acronyms:
  - UDDI
  - WSDL
  - SOAP
  - HTTP, SMTP, FTP
  - Programming
  - Schema
  - XML
- Practical Examples:
  - Phone Book
  - Contract
  - Envelope
  - Mail person
  - Speech
  - Vocabulary
  - Alphabet

Use of web services



SOAP, WSDL, UDDI, and XML in all of them

UDDI (Universal Description, Discovery and Integration)

- is a **platform-independent, Extensible Markup Language (XML)-based registry** by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications.

- is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet.

UDDI

- UDDI servers act as a directory of available services and service providers. **SOAP can be used to query UDDI to find the locations of WSDL definitions of services**, or the search can be performed through a user interface at design or development time.
- Data structure specification describes what kind of data is stored in UDDI.
- The programmer's API specification contains how a UDDI registry can be accessed.
- The replication specification contains descriptions of how registries replicate information among themselves.

UDDI

- UDDI registries contains information about businesses and the Services these businesses offer.
  - Public registries
  - Private registries
- Three basic functions
  - publish : how to register a web service
  - search: how to find a specific web service
  - binding: how to connect to a web service

Public Registries

- IBM Registration: https://uddi.ibm.com/ubr/registry.html
  - inquiryURL= https://uddi.ibm.com/ubr/inquiryapi
  - publishURL= https://uddi.ibm.com/ubr/publishapi
- HP Registration: http://uddi.hp.com
  - inquiryURL = http://uddi.hp.com/ubr/inquire
  - publishURL = https://uddi.hp.com/ubr/publish
- Microsoft Registration: http://uddi.rte.microsoft.com
  - inquiryURL=http://uddi.rte.microsoft.com/inquire
  - publishURL=https://uddi.rte.microsoft.com/publish
- SAP Registration: http://udditest.sap.com
  - inquiryURL=http://uddi.sap.com/UDDI/api/inquiry/
  - publishURL=https://uddi.sap.com/UDDI/api/publish/

Public Registries *(well, it used to be...)*

- IBM Registration: https://uddi.ibm.com/ubr/registry.html
  - inquiryURL= https://uddi.ibm.com/ubr/inquiryapi
  - publishURL = https://uddi.ibm.com/ubr/publish
- Microsoft Registration: http://uddi.rte.microsoft.com
  - inquiryURL=http://uddi.rte.microsoft.com/inquire
  - publishURL=https://uddi.rte.microsoft.com/publish
- SAP Registration: http://uddirest.sap.com
  - inquiryURL=http://uddi.sap.com/UDDI/api/inquiry/
  - publishURL=https://uddi.sap.com/UDDI/api/publish/

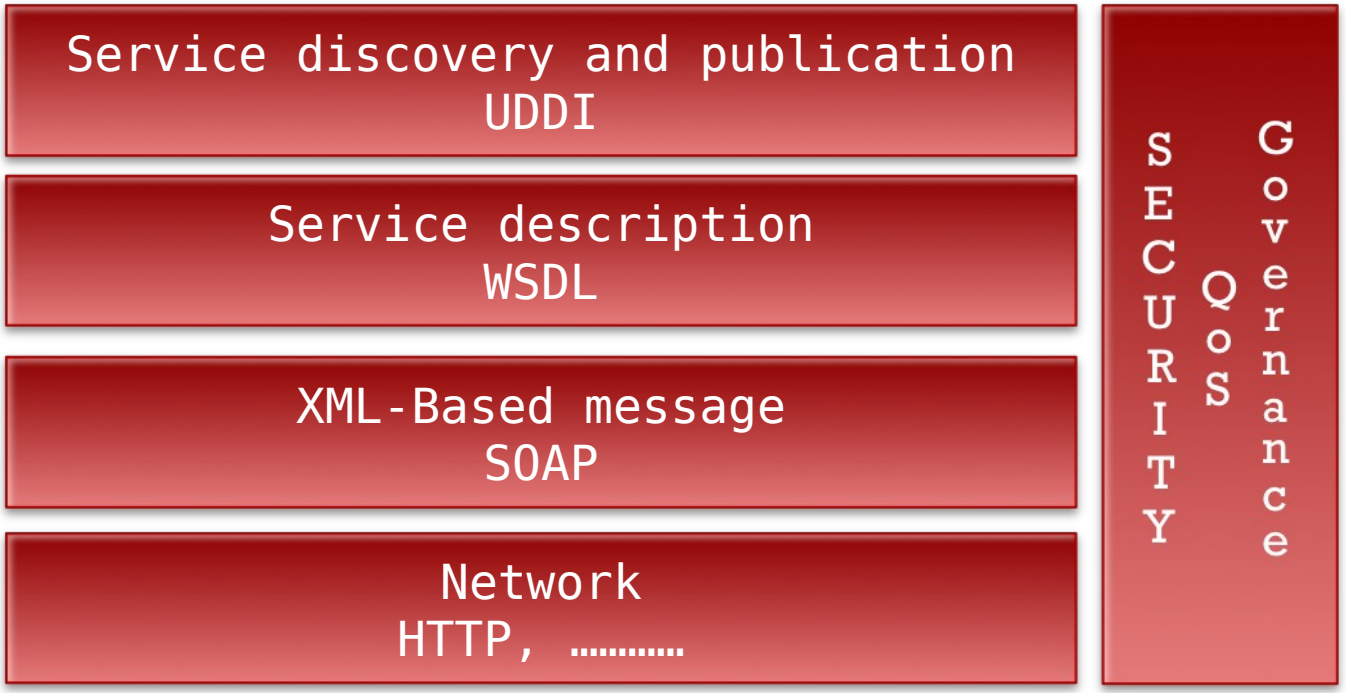
UDDI **has not been adopted as widely as its designers had hoped**. IBM, Microsoft, and SAP announced they were **closing** their public UDDI nodes in January 2006.

The OASIS UDDI Specification Technical Committee voted to complete its work in late 2007 and has been **closed**.

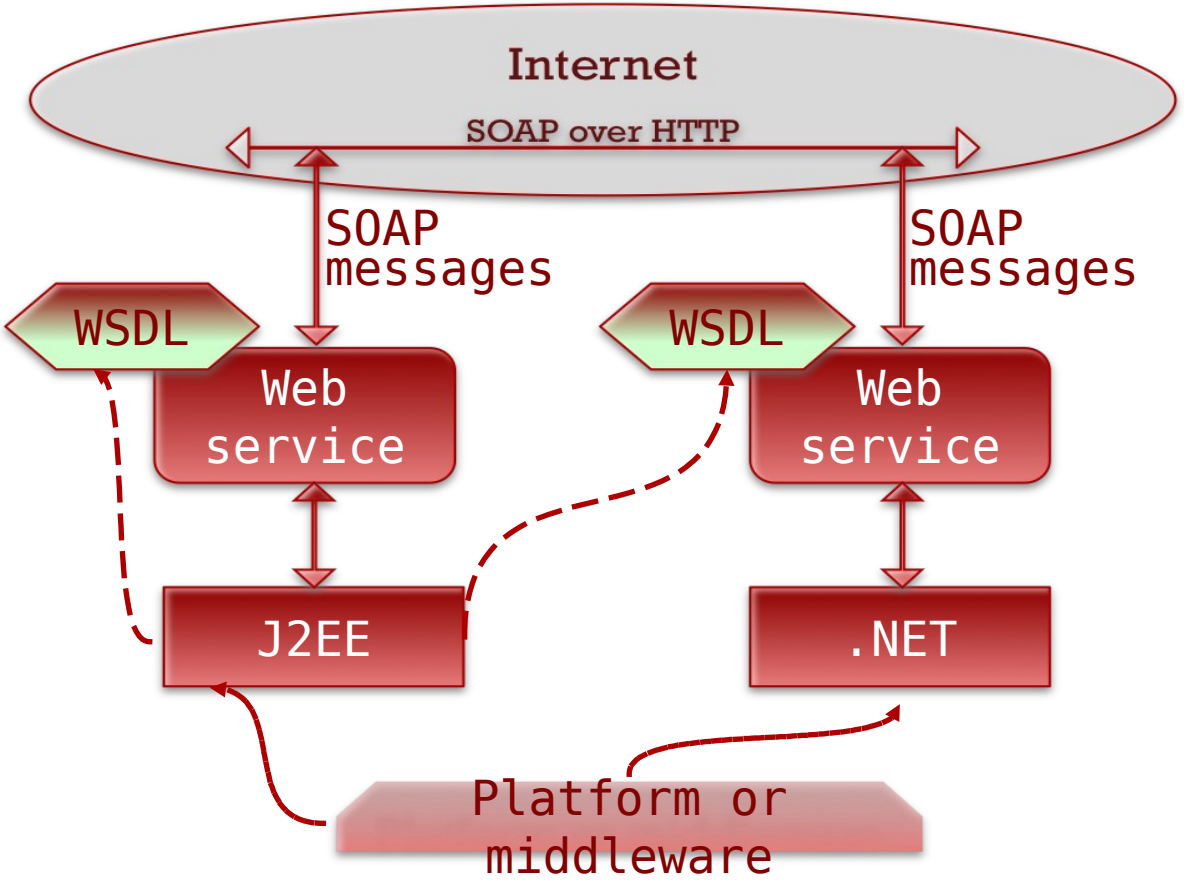
In September 2010, Microsoft announced they were **removing** UDDI services from the Windows Server operating system.

UDDI systems **are most commonly found inside companies**, where they are used to dynamically bind client systems to implementations. However, much of the search metadata permitted in UDDI is not used for this relatively simple role.

Enabling technologies



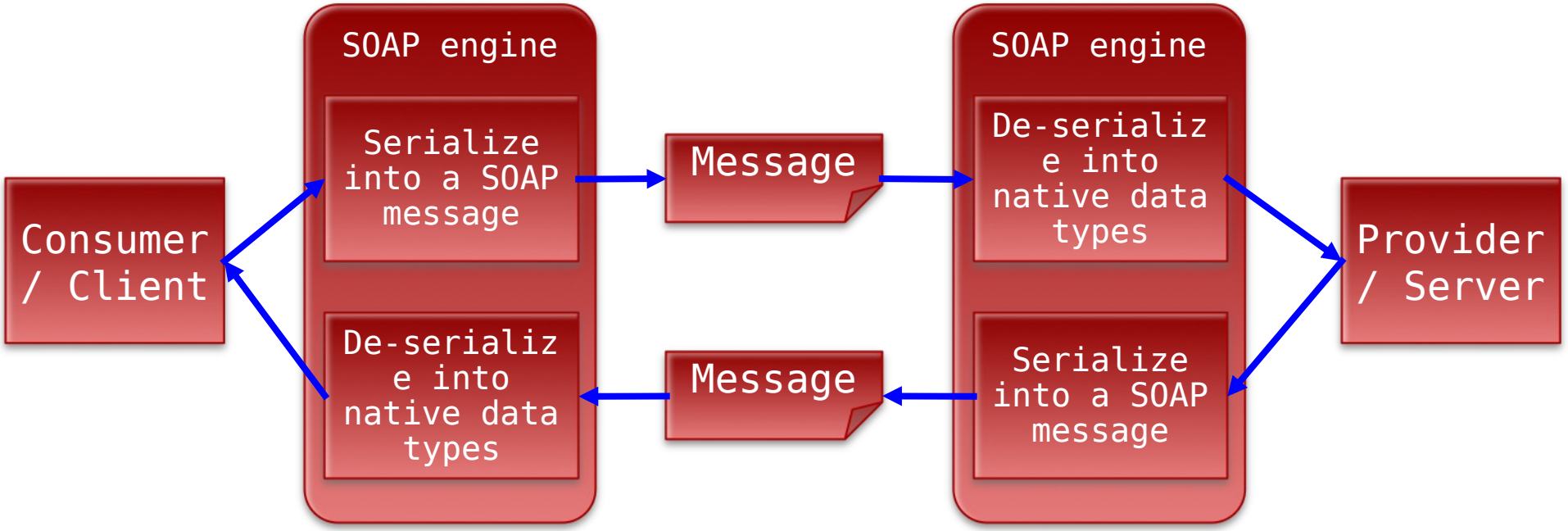
WS*



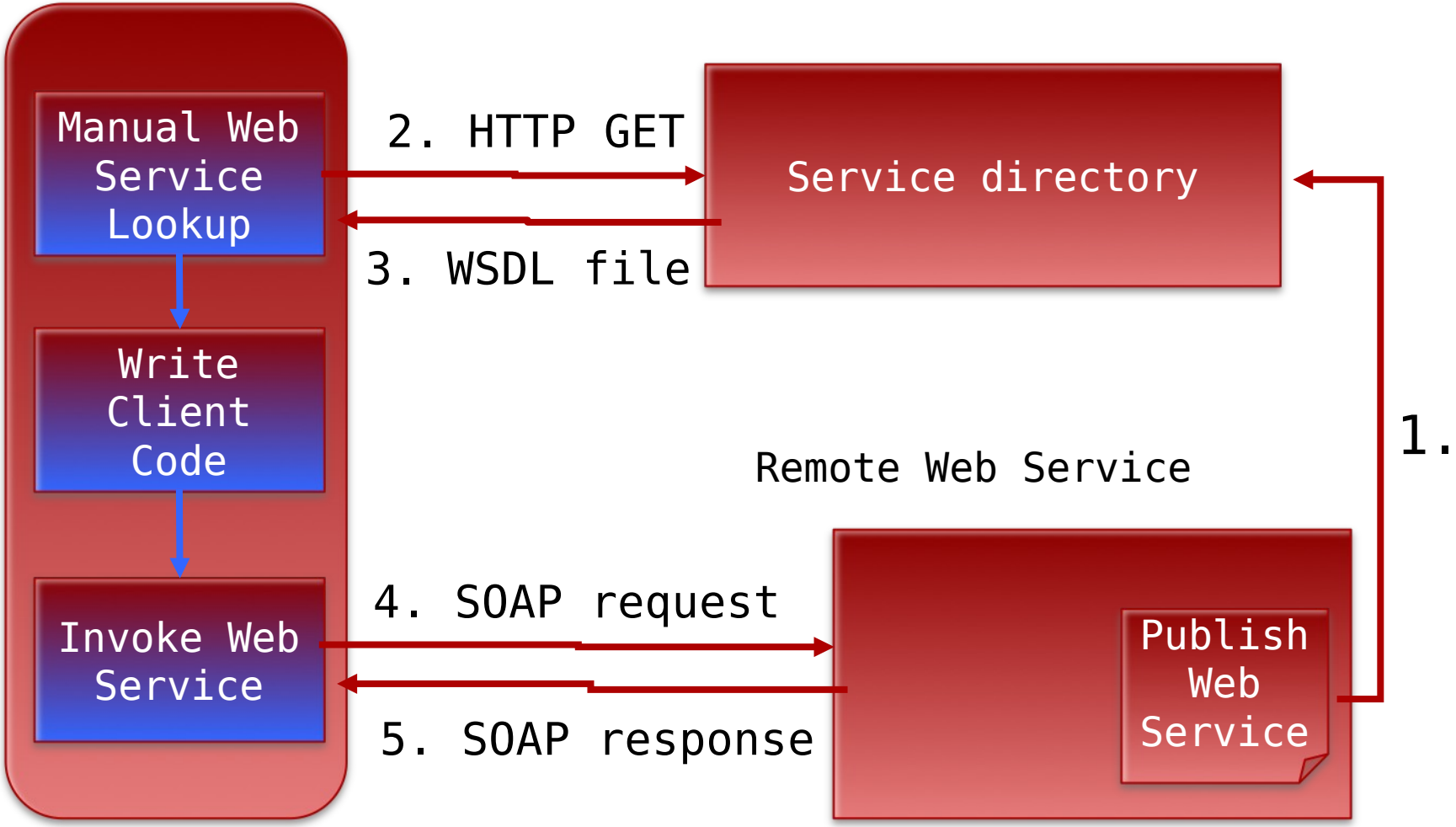
- clear specifications of the service interface and the data types in use
- communication protocol independent (platform, programming language)
- interoperability.

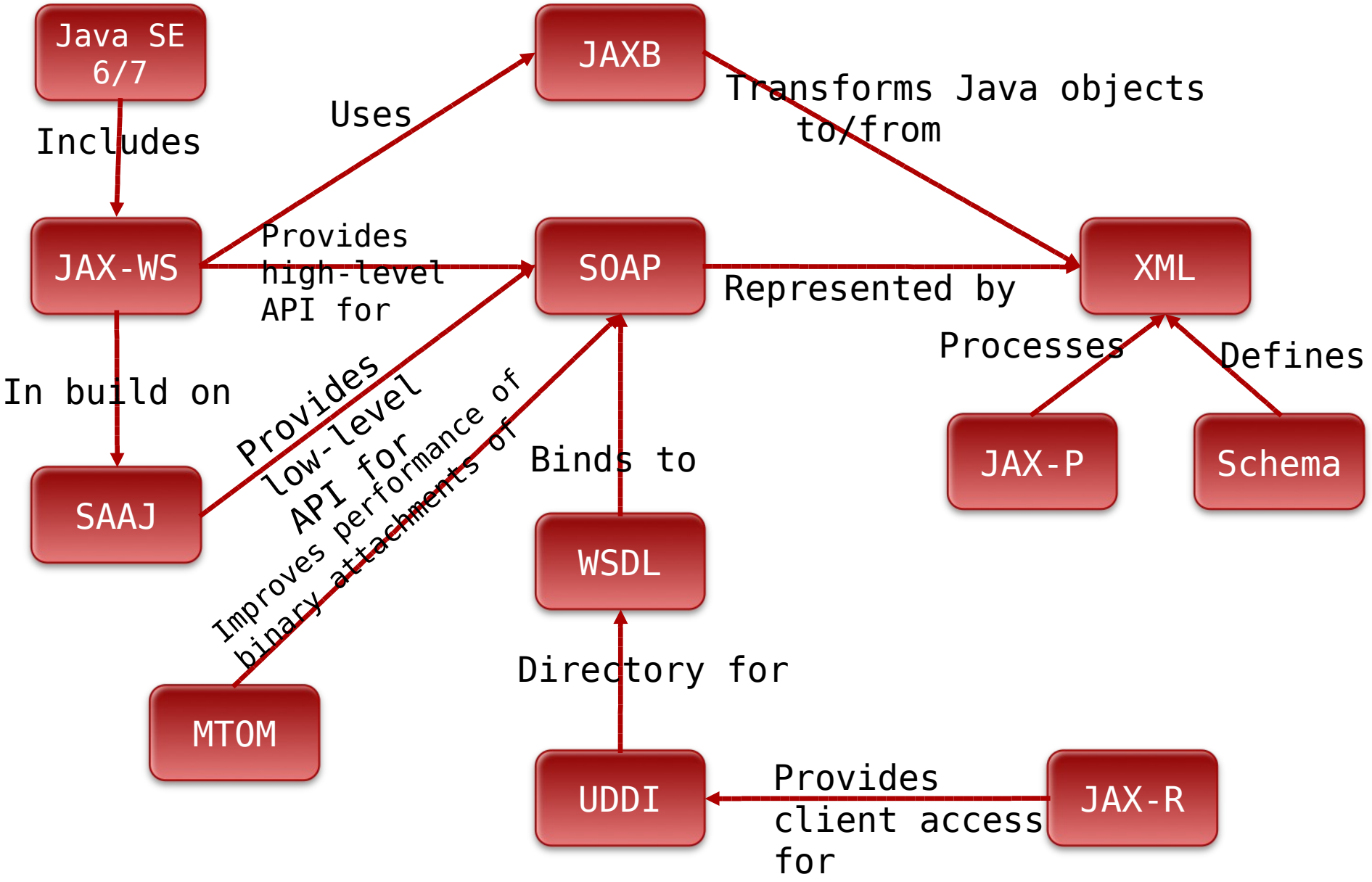
SOAP engines

A **SOAP engine** is a framework used in servers and clients that facilitates:
1.Serializing objects from a programming language into SOAP messages
2.De-serializing SOAP messages into objects in a programming language, i.e. creating appropriate data types and populating these with the message content.



Simple Web Service Invocation




Java SE 6/7
Includes
JAX-WS
In build on
SAAJ
MTOM
Uses
JAXB
Provides high-level API for
SOAP
Provides low-level API for
Improves performance of binary attachments of
Binds to
WSDL
Directory for
UDDI
Transforms Java objects to/from
XML
Represented by
Processes
Defines
JAX-P
Schema
Provides client access for
JAX-R

An example (1/7)
Implementing a simple web service with Java

1. Create the “service endpoint interface”
   - Interface for web service

1. Create the “service implementation”
   - Class that implements the service

1. Create the “service publisher”

- Java supports web services in core Java
  - JAX-WS (Java API for XML-Web Services)

- In full production mode, one would use a Java application server such as Tomcat, Glassfish, etc.

An example (2/7)
Service Endpoint Interface

```
package example.echo;   // echo server
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

@WebService      // This is a Service Endpoint Interface (SEI)
@SOAPBinding(style = Style.RPC) // Needed for the WSDL
public interface EchoServer {
    @WebMethod    // This method is a service operation
        String EchoMessage(String strMsg); }
```

An example (3/7)

Service Implementation

```
package example.echo;
import javax.jws.WebService;
/**
   * The @WebService property endpointInterface links this class
   * to example.echo.EchoServer.
*/
@WebService(endpointInterface =
"example.echo.EchoServer")
public class EchoServerImpl implements EchoServer {
      public String EchoMessage(String Msg) {
            String capitalizedMsg;
            System.out.println("Server: EchoMessage() invoked...");
            System.out.println("Server: Message > " + Msg);
            capitalizedMsg = Msg.toUpperCase();
            return(capitalizedMsg);
      }
```

An example (4/7)
Service Publisher

```
package example.echo;
import javax.xml.ws.Endpoint;

public class EchoServerPublisher {
      public static void main(String[ ] args) {
      // 1st argument is the publicaNon URL
      // 2nd argument is an SIB instance
      Endpoint.publish("http://localhost:9876/es", new
      EchoServerImpl());
      }
}
```

An example (5/7)
Deploying and testing

1. Compile the Java code
2. Run the publisher
   - java example.echo.EchoServerPublisher
1. Testing the web service with a browser
   - URL: http://localhost:9876/es?wsdl

An Example (6/7)
WSDL for echo service

```
<definitions targetNamespace=”http://echo.example/" name="EchoServerImplService">
<types/>
<message name="EchoMessage”> <part name="arg0" type="xsd:string"/> </message>
<message name="EchoMessageResponse”><part name="return"
type="xsd:string"/></message>

<portType name="EchoServer">
      <operation name="EchoMessage">
            <input message="tns:EchoMessage"/>
            <output message="tns:EchoMessageResponse"/>
      </operation>
</portType>

<binding name="EchoServerImplPortBinding" type="tns:EchoServer">
      <soap:binding transport=”http://schemas.xmlsoap.org/soap/http" style="rpc”/>
      <operation name="EchoMessage">
      <soap:operation soapAction=""/>
      <input> <soap:body use="literal" namespace="http://echo.example/"/> </input>
      <output> <soap:body use="literal" namespace="http://echo.example/"/> </output>
      inding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
      </operation>
</binding>

<service name="EchoServerImplService">
<port name="EchoServerImplPort" binding="tns:EchoServerImplPortBinding”>
      <soap:address location=”http://localhost:9876/es"/>
</port>
</service>
</definitions>
```

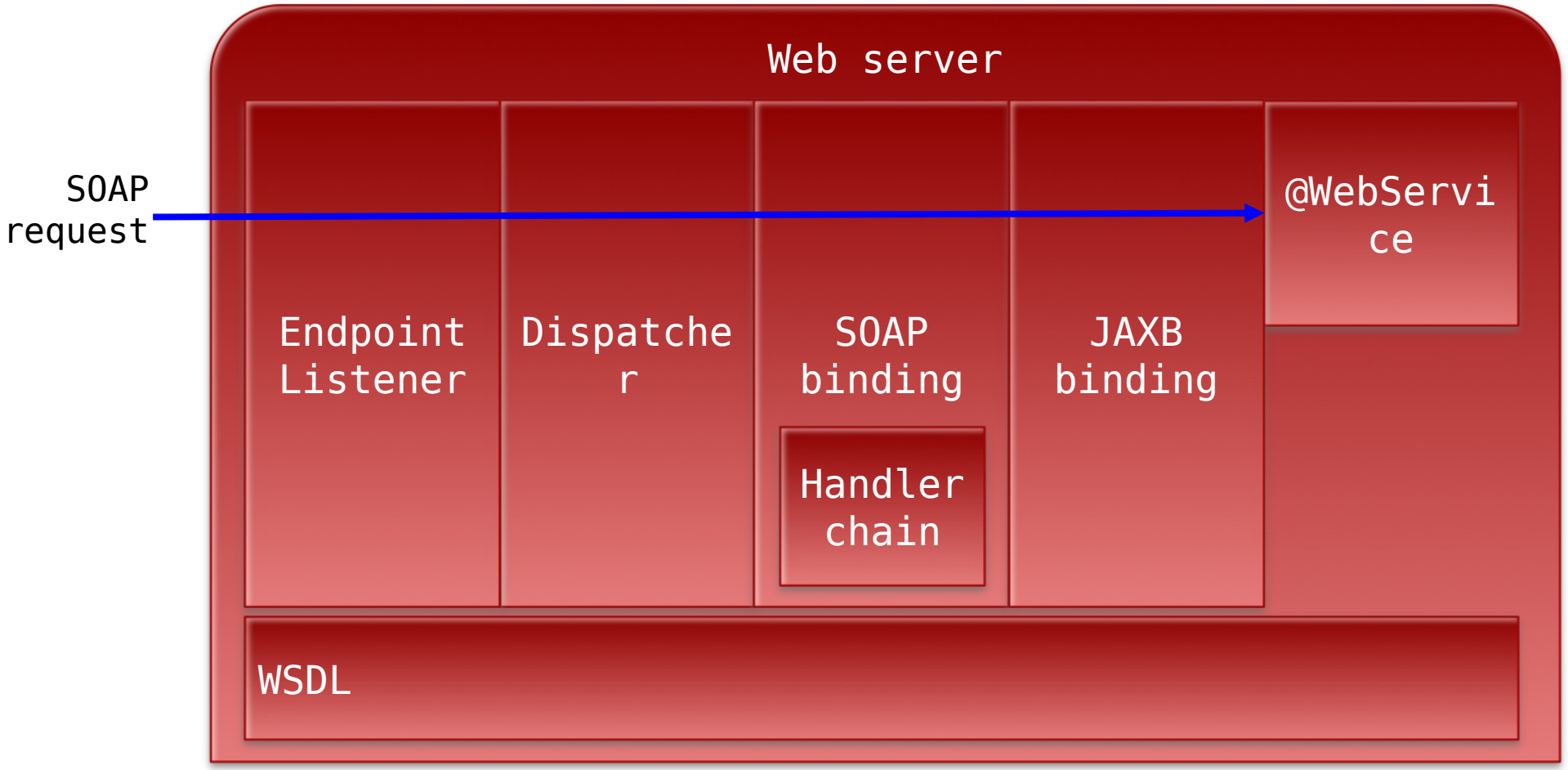
An Example (7/7)
EchoClient

```
package example.echo;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import java.net.URL;

class EchoClient {
      public static void main(String argv[ ]) throws Exception {
            if (argv.length < 1) {
            System.out.println("Usage: java EchoClient \"MESSAGE\"");System.exit(1);}

            String strMsg = argv[0];
            URL url = new URL(”http://localhost:9876/es?wsdl");
                  // Qualified name of the service:
                  // 1st arg is the service URI
                  // 2nd is the service name published in the WSDL
            QName qname = new
            QName(“http://echo.example/”,“EchoServerImplService");
            Service service = Service.create(url, qname);
                  // Extract the endpoint interface, the service "port".
            EchoServer eif = service.getPort(EchoServer.class);
            System.out.println(eif.EchoMessage(strMsg));
      }
}
```

Server side


Web server
SOAP request
Endpoint Listener
Dispatcher
SOAP binding
Handler chain
JAXB binding
@WebService
WSDL

Developing a Web Service

Server code

@WebService
POJO class
Servlet-based

WSDL

Service contract

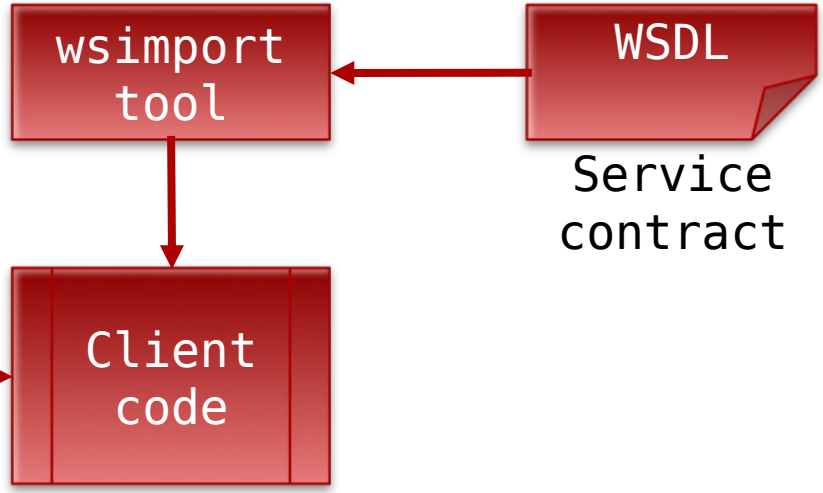
war file (or ear)

Deployment

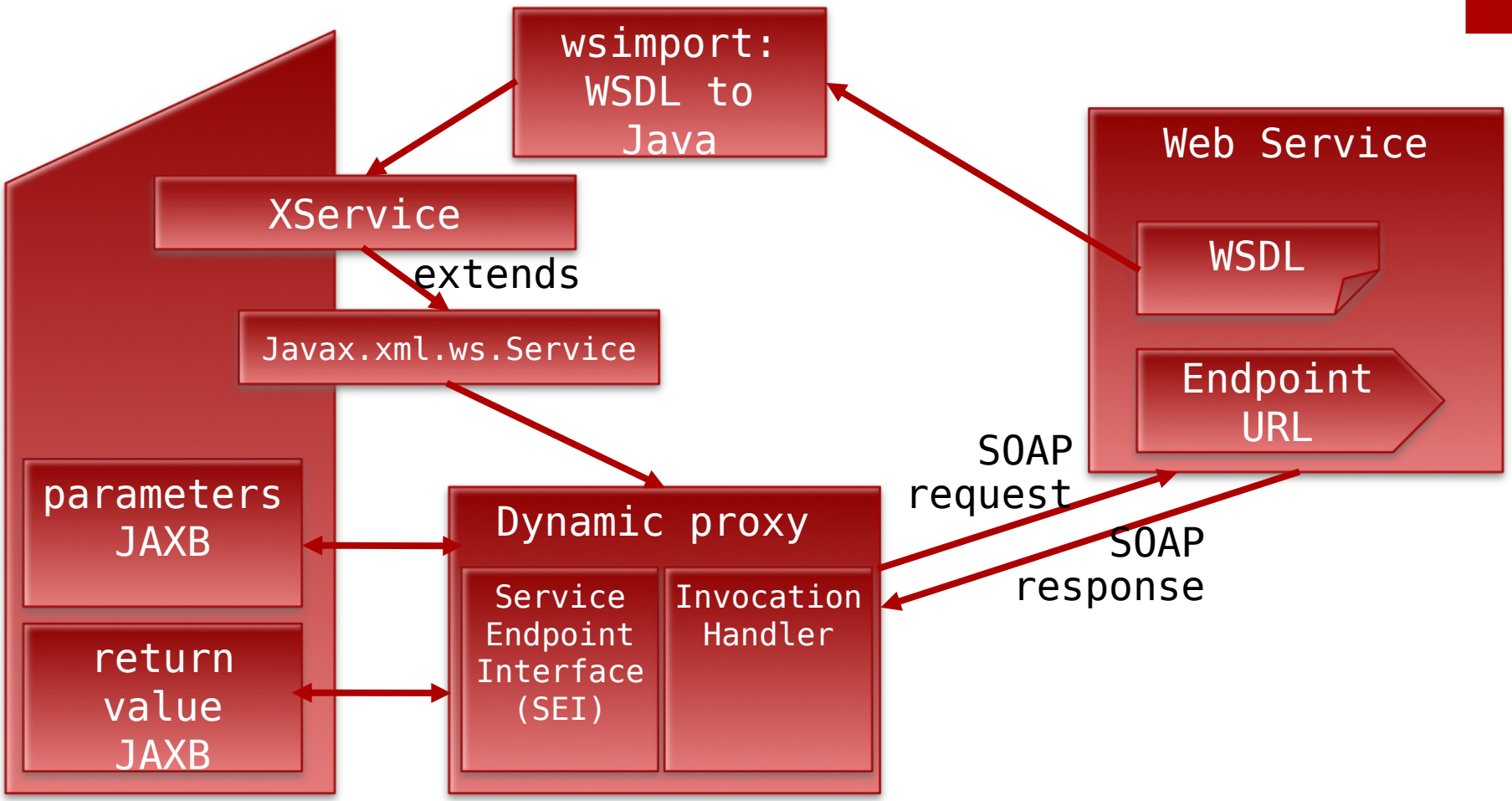
JAXB & JAX-WS files

Client-side programming

You develop Client which uses proxy to call Web Service



Client side


wsimport:
WSDL to
Java
XService
extends
Javax.xml.ws.Service
parameters
JAXB
return
value
JAXB
Dynamic proxy
Service
Endpoint
Interface
(SEI)
Invocation
Handler
Web Service
WSDL
Endpoint
URL
SOAP
request
SOAP
response

WSDL

- The **Web Service Description Language** is a **technical description of a Web Service**
- It mentions **all interfaces available**, with the relevant information for the invocation (parameters, return type...)
- It is possible to generate
  - the client code for accessing the Web Service
  - A WSDL file from Java source code
  - A Java source code skeleton from **WSDL** file

Web Service Example

A Web service AddFunction with operation addInt is known through its WSDL:

```
<wsdl:message name="addIntResponse">
      <wsdl:part name="addIntReturn" type="xsd:int" />
 </wsdl:message>
<wsdl:message name="addIntRequest">
       <wsdl:part name="a" type="xsd:int" />
       <wsdl:part name="b" type="xsd:int" />
 </wsdl:message>
<wsdl:portType name="AddFunction">
    <wsdl:operation name="addInt" parameterOrder="a b">
        <wsdl:input message="impl:addIntRequest" name="addIntRequest" />
        <wsdl:output message="impl:addIntResponse" name="addIntResponse" />
    </wsdl:operation>
 </wsdl:portType>
```

```
// possible implementation of WS using
the wsimport tool:
// AddFunction.jws
public class AddFunction {
  int addInt(int a, int b){
    return(a+b);
  }
}
```

Generating a WSDL file from a Java class

```
public class WeightConverter {
    public double kgtopounds (double kg){
        return kg*2.20462262;
    }
    public double poundstokg (double pounds)
    {
        return pounds/2.20462262;
    }
}
```

javac –cp . WeightConverter.java
java2wsdl –cp . –tn weightconverter –stn weightconverter –cn WeightConverter

`-cp = classpath; -tn target namespace; -stn schema target namespace; -cn class name`

Generating the service code skeleton from the WSDL file

wsdl2java -ss -sd -uri WeightConverter.wsdl

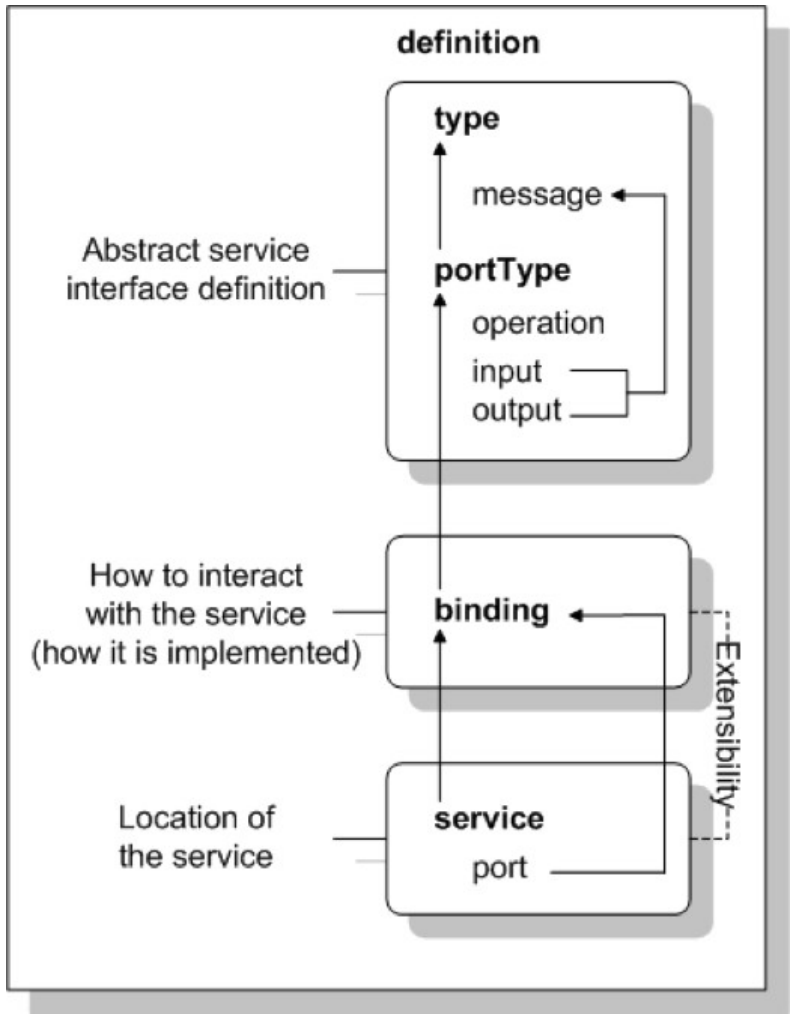
-ss = server side; -sd = service descriptor

- A src directory is created with the source code for our server side files
- A resources directory is created with the WSDL file for the service and a service descriptor (services.xml) file
- A build.xml file is created in the current directory, which will be used to create the ws deployment file

Using WSDL

- As extended IDL: **WSDL allows tools to generate compatible client and server stubs**:
  - Tool support for top-down, bottom-up and “meet in the middle” development.
- Allows industries to define standardised service interfaces.
- Allows advertisement of service descriptions, enables dynamic discovery and binding of compatible services:
  - Used in conjunction with UDDI registry
- Provides a normalised description of heterogeneous applications.

WSDL elements



- **<types>**, the data types of input and output data, used by the web service
- **<message>**, messages to be exchanged, used by the web service
- **<portType>**, the operations input and output exposed by the web service
- **<binding>**, the coupling and protocols used by the web service
- **<port>** service location and binding

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://tempuri.org/AreaService/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="AreaService"
targetNamespace="http://tempuri.org/AreaService/">

 <wsdl:types>
       <xsd:schema targetNamespace="http://tempuri.org/AreaService/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
             <xsd:element name="area" type="xsd:float"/>
             <xsd:element name="parameters" type="tns:dimensions"/>
             <xsd:complexType name="dimensions">
                  <xsd:sequence>
                      <xsd:element name="width"
            type="xsd:float"></xsd:element>
                      <xsd:element name="height"
            type="xsd:float"></xsd:element>
                  </xsd:sequence>
             </xsd:complexType>
       </xsd:schema>
  </wsdl:types>

  <wsdl:message name="CalculateRectAreaResponse">
       <wsdl:part element="tns:area" name="area"/>
  </wsdl:message>
  <wsdl:message name="CalculateRectAreaRequest">
       <wsdl:part element="tns:parameters" name="parameters"/>
  </wsdl:message>
```

```
    <wsdl:portType name="AreaService">
        <wsdl:operation name="CalculateRectArea">
            <wsdl:input message="tns:CalculateRectAreaRequest"/>
            <wsdl:output message="tns:CalculateRectAreaResponse"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="AreaServiceSOAP" type="tns:AreaService">
        <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="CalculateRectArea">
            <soap:operation
        soapAction="http://tempuri.org/AreaService/NewOperation"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
     </wsdl:binding>

     <wsdl:service name="AreaService">
        <wsdl:port binding="tns:AreaServiceSOAP" name="AreaServiceSOAP">
            <soap:address location="http://tempuri.org"/>
        </wsdl:port>
     </wsdl:service>
</wsdl:definitions>
```

Summary

- WS* standards are unevenly taken into use
  - Service orientation is well accepted
  - Several competing solutions, most notably WS* vs REST that are merging to complement each other
  - Successful and accepted standardization in technical interfaces
  - Business interfaces not proceeding
- Many technical complexities still remains
- Emergence of new new solutions is frequent