Module Two: Service Message Exchange SOAP and Service Description WSDL

Our textbook for this module:

- **Ethan Cerami, Web Services Essentials, Publisher： O’Reilly， ISBN：9780596002244，**
  - Chapter 3 - SOAP
  - Chapter 6 - WSDL
- **Liang-Jie Zhang, Services Computing, Publisher: Springer， ISBN：9783540382812 -** You can find an online version of this book for free through our library webpage.
  - Chapter 3.1
  - Chapter 3.2 (without 3.2.5)

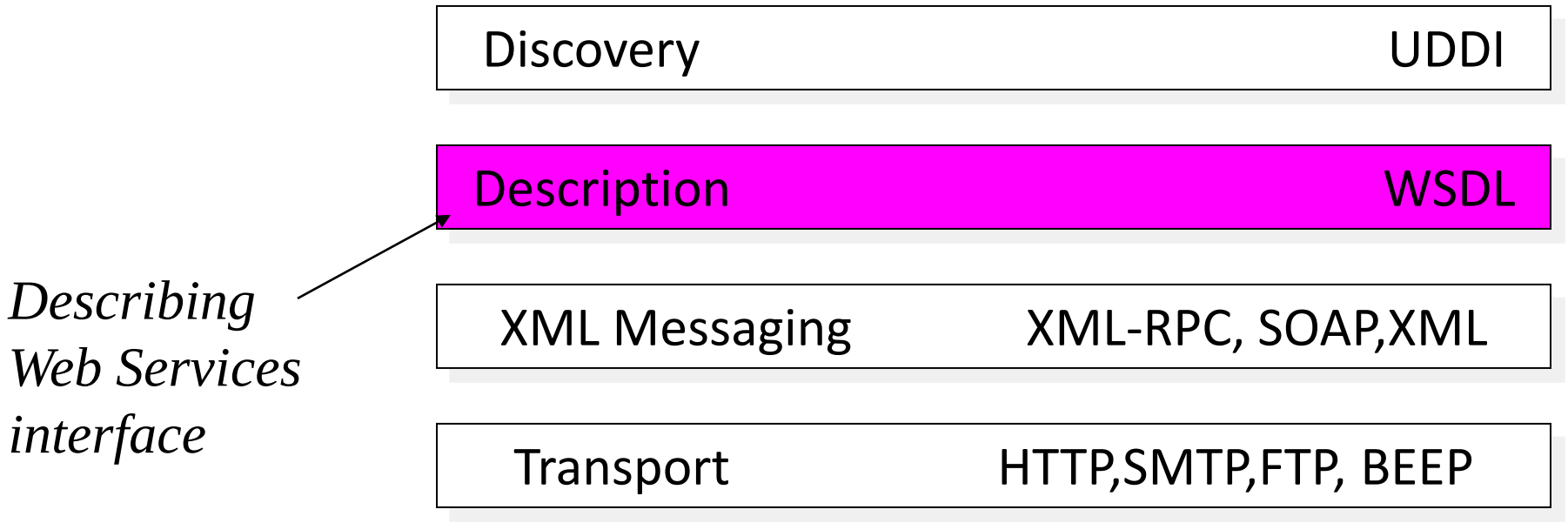
Module 2 Learning Outcomes

- Understand the basics of the SOAP protocol
- Understand the details about the SOAP XML Message specification
- Understand the SOAP encoding rules
- Understand the basics of WSDL

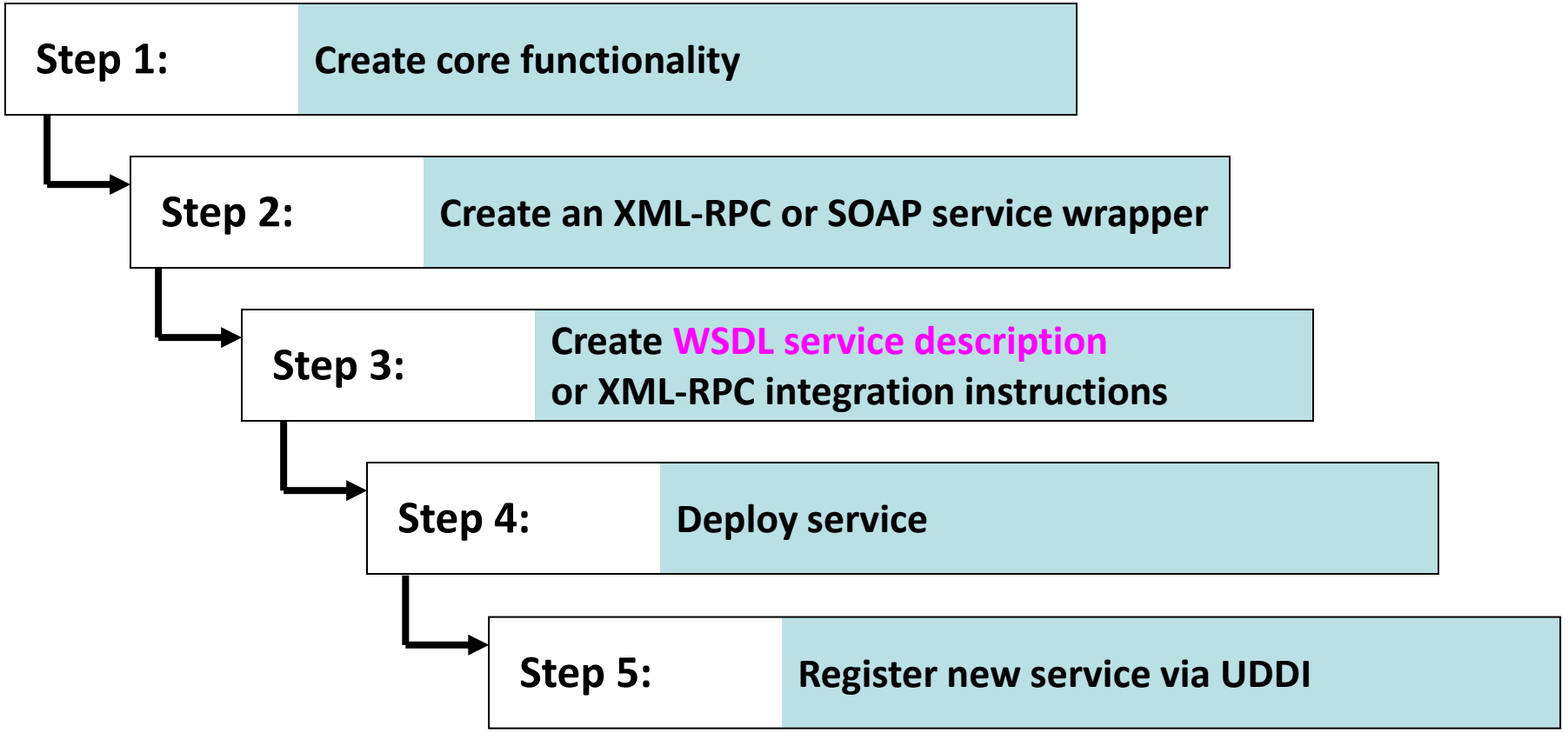
- Let’s continue where we left off on Monday

Service Description
WSDL

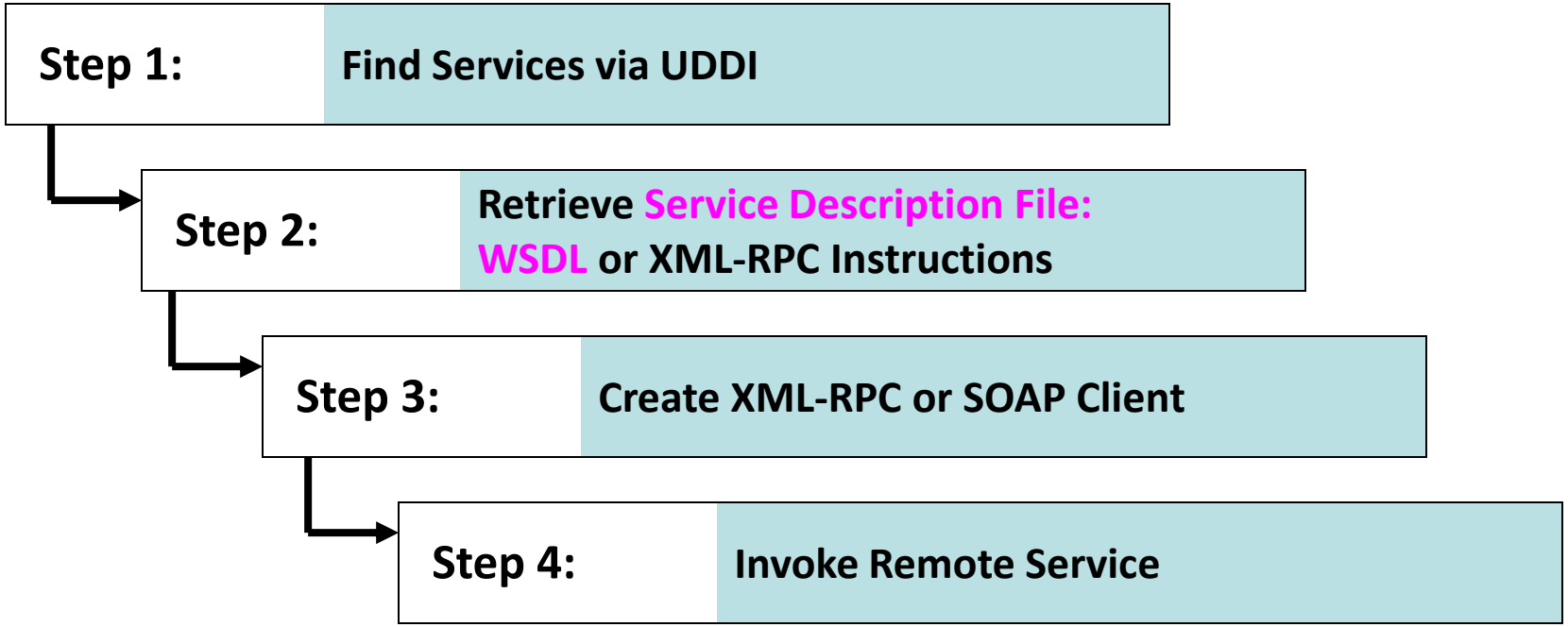
Web Service Protocol Stack



Using the Protocols Together – service provider perspective



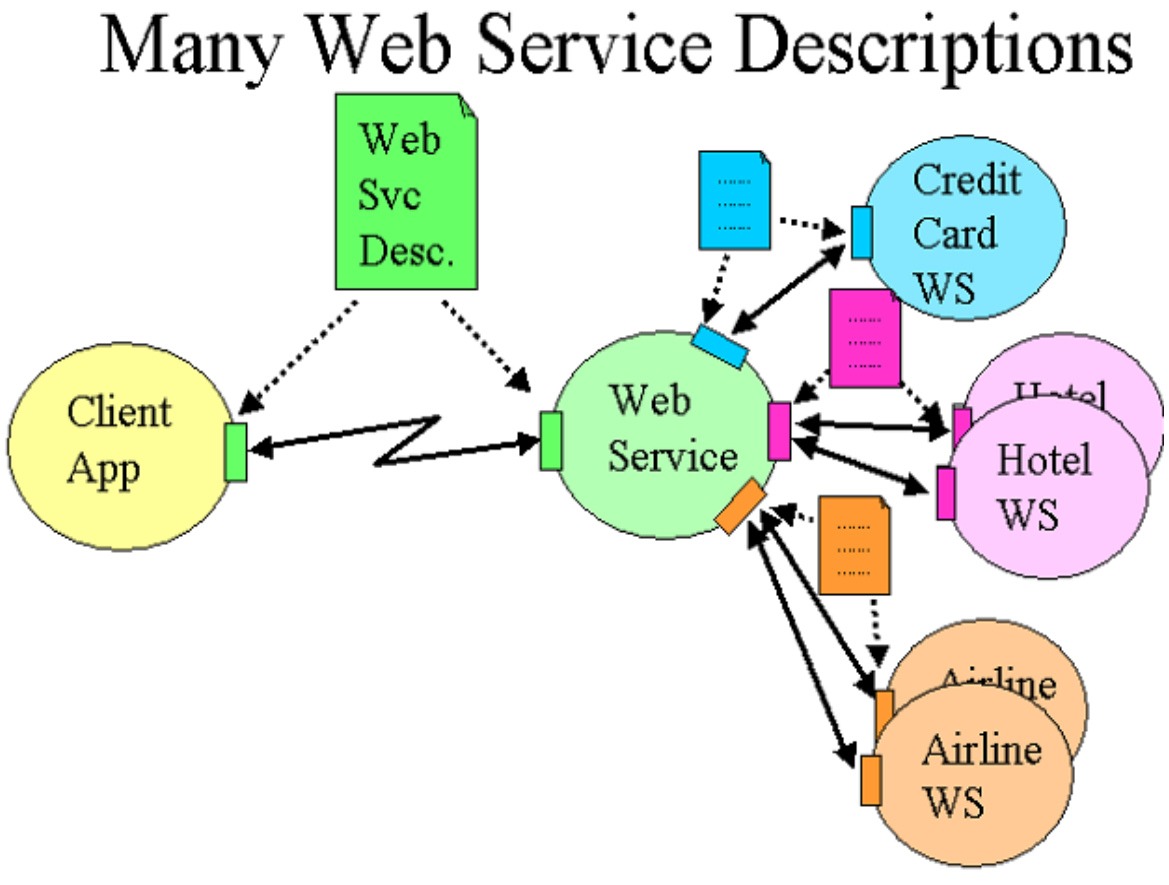
Using the Protocols Together – service request perspective



A client program reads a WSDL document to understand what a Web service can do; then it uses SOAP to actually invoke the functions listed in the WSDL document.

WSDL Essentials

- For services to interact, they must be aware of each other



Service Description

- WSDL provides a notation to answer the following three questions:
  - What is the service about?
  - Where does it reside?
  - How can it be invoked? (i.e., what, where, and how)

What is WSDL

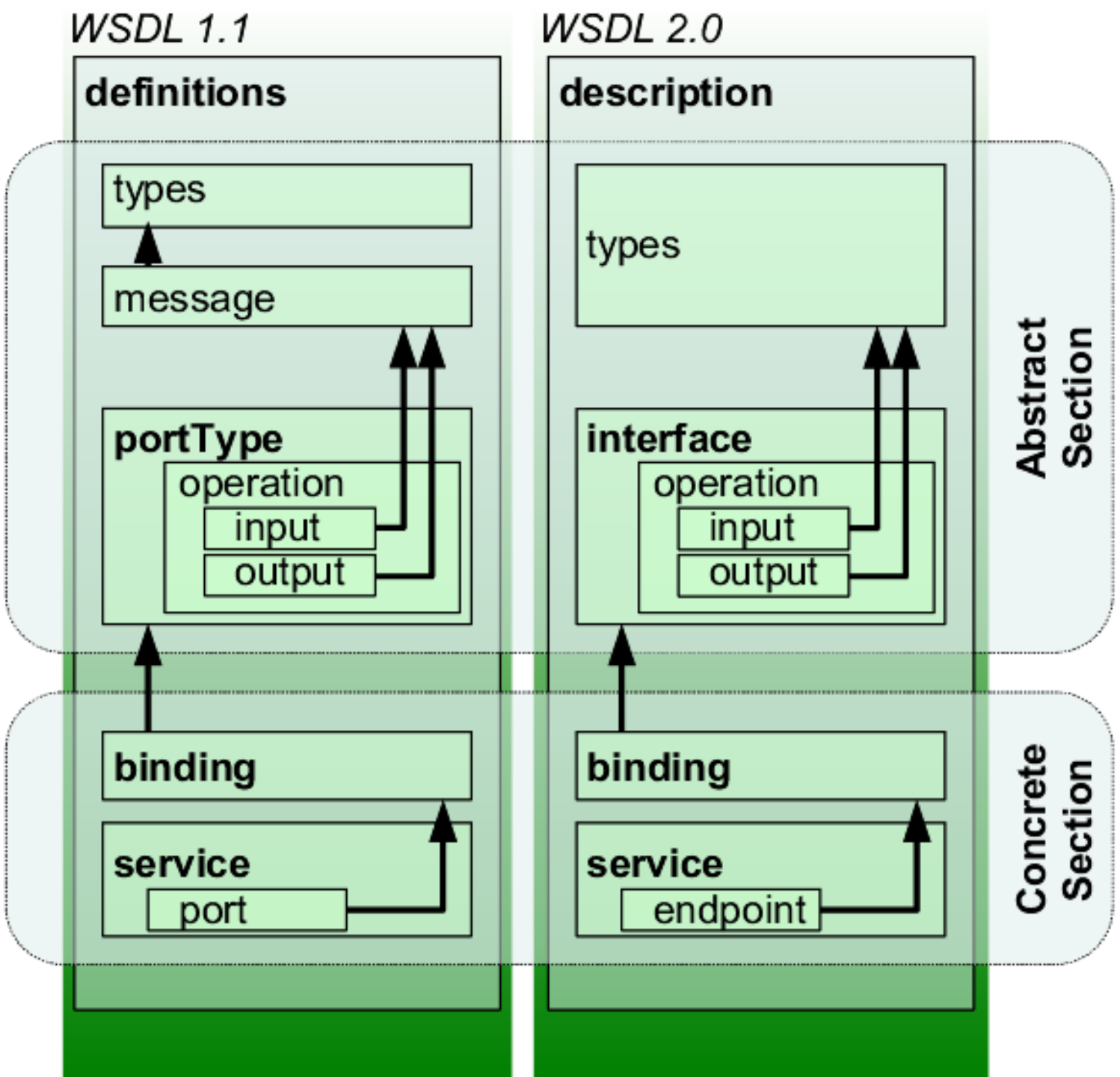
- WSDL is a specification defining how to describe web services in a common XML grammar.
- WSDL describes four critical pieces of data:
  - Interface information describing all publicly available functions
  - Data type information for all message requests and message responses
  - Binding information about the transport protocol to be used
  - Address information for locating the specified service

WSDL Benefits

- Platform- and language-independent
- Using WSDL, a client can locate a web service and invoke any of its publicly available functions.
- With WSDL-aware tools, you can also automate this process, enabling applications to easily integrate new services with little or no manual code.
- WSDL provides a common language for describing services and a platform for automatically integrating those services.

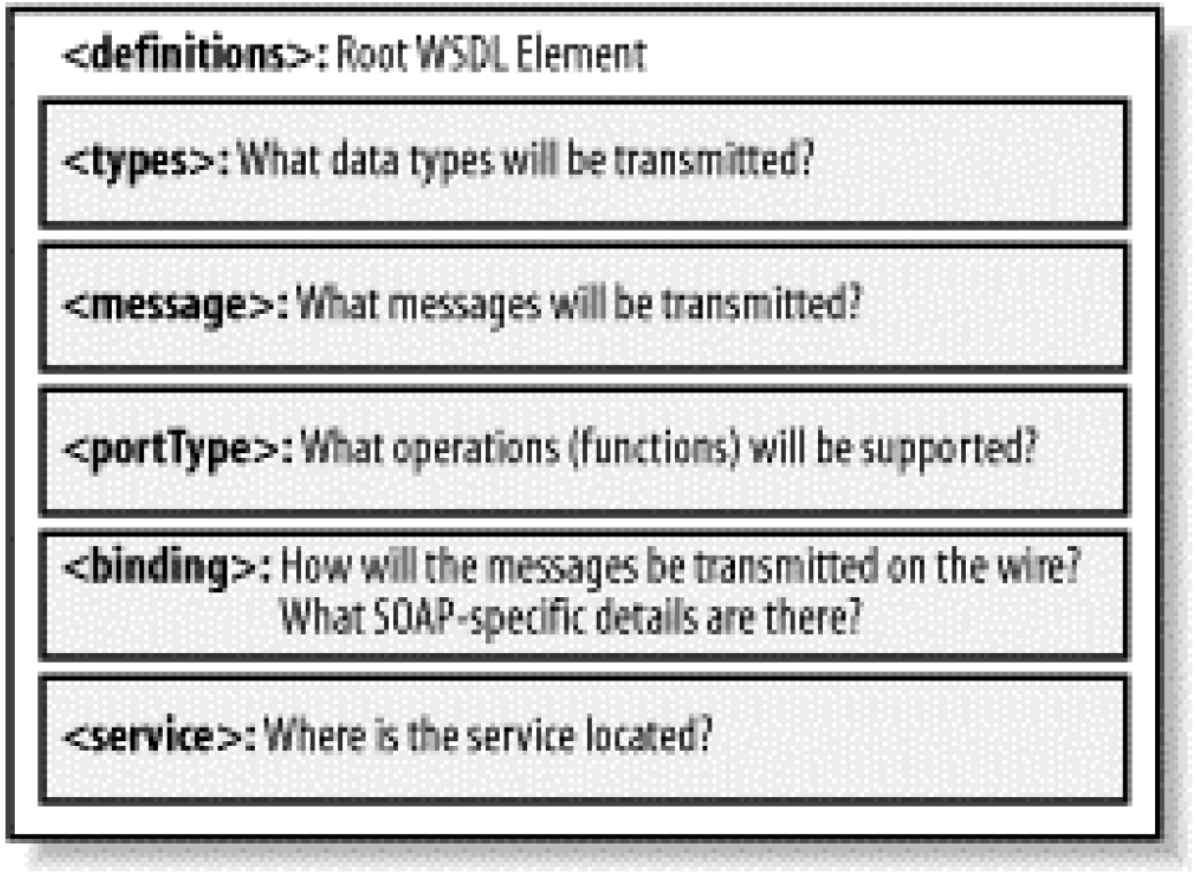
WSDL

- An XML-based interface description language
- Used for describing the functionality offered by a web service
- WSDL describes services as collections of network endpoints or ports



WSDL Specification major Elements

- definitions
- types
- message
- portType
- binding
- service



WSDL Specification utility Elements

- documentation
  - The documentation element is used to provide human-readable documentation and can be included inside any other WSDL element
- import
  - The import element is used to import other WSDL documents or XML Schemas.
  - This enables more modular WSDL documents. For example, two WSDL documents can import the same basic elements and yet include their own service elements to make the same service available at two physical addresses.

Example - HelloService.wsdl

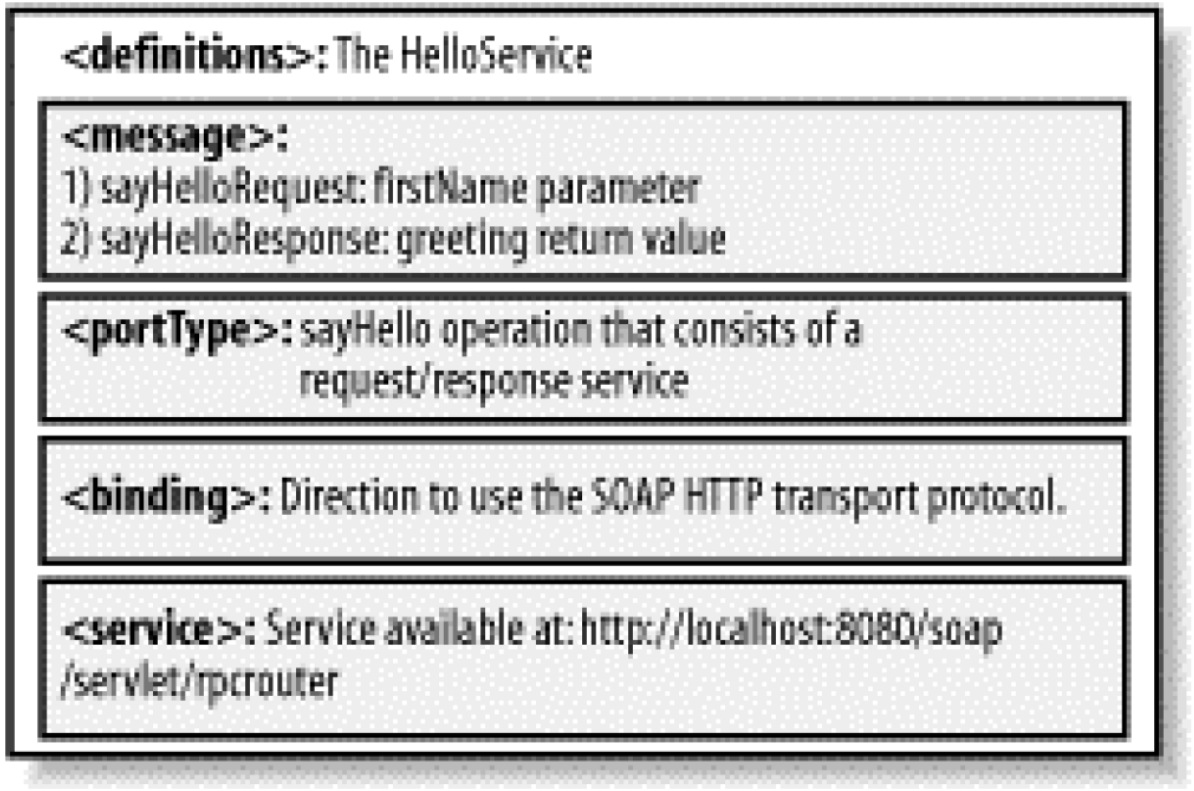
- The service provides a single publicly available function, called sayHello.
  - The function expects a single string parameter
  - The function returns a single string greeting.
  - For example, if you pass the parameter world, the service returns the greeting: "Hello, world!"

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

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>

   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>
```

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
   <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </input>
      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </output>
   </operation>
</binding>
```

```
   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://localhost:8080/soap/servlet/rpcrouter"/>
      </port>
   </service>
</definitions>
```

Example - HelloService.wsdl

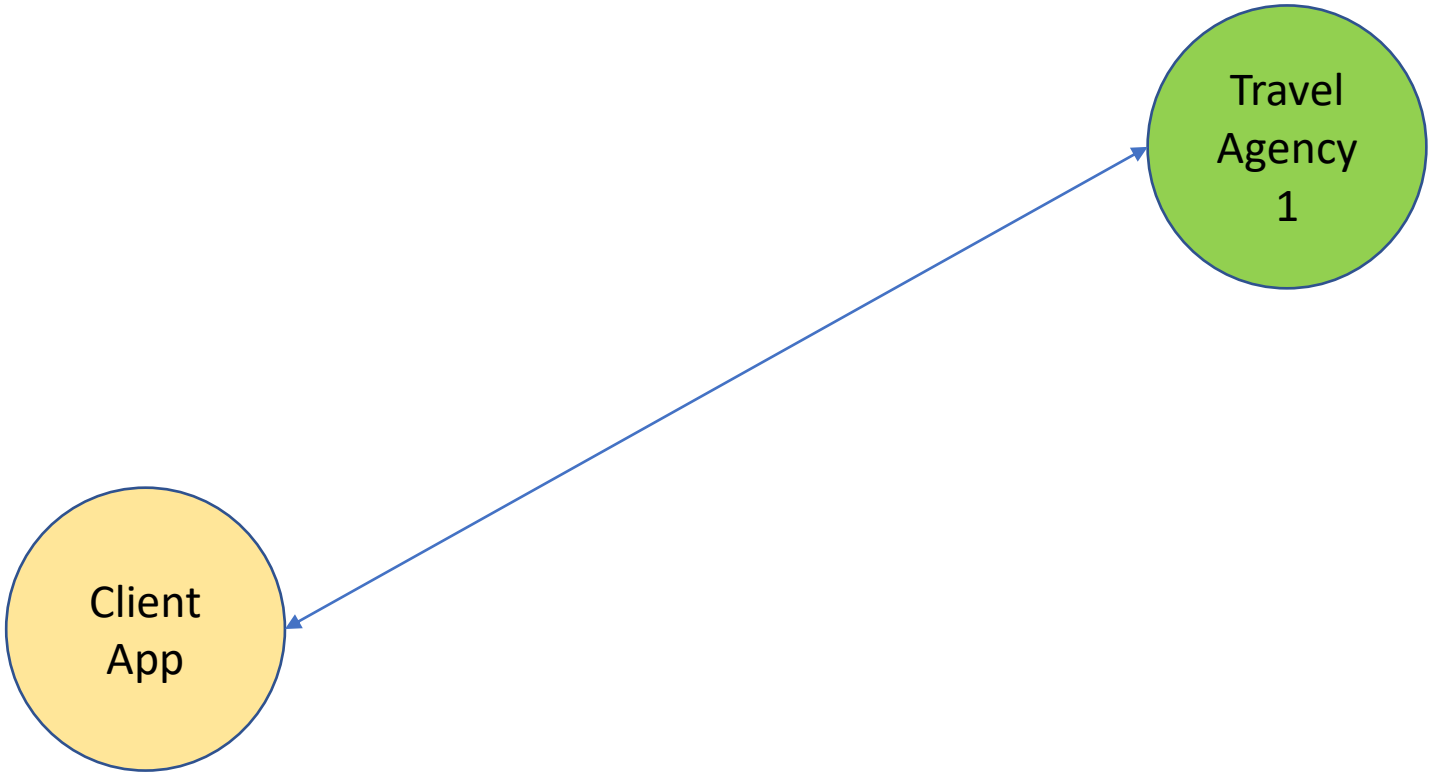


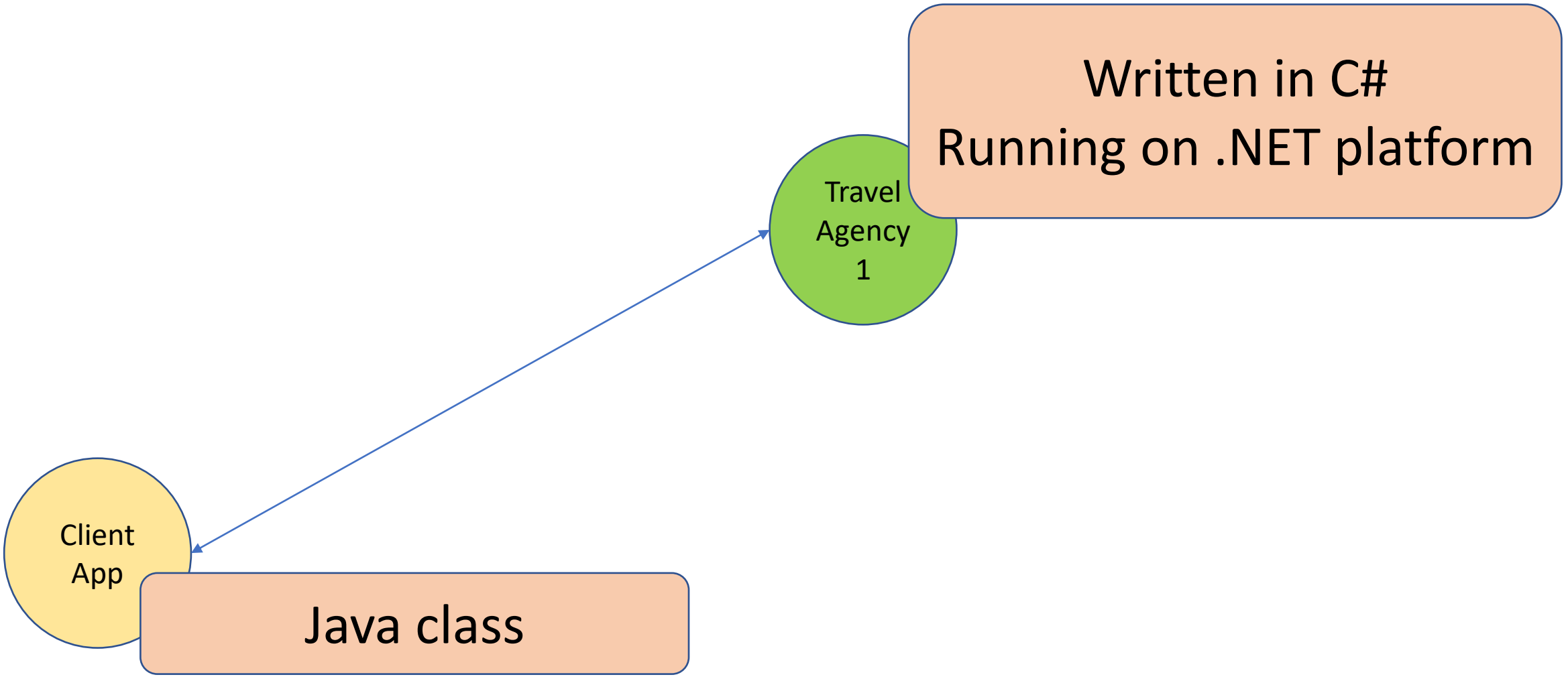
WSDL and SOAP

- WSDL is often used in combination with SOAP and XML schema to define a Web service over the Internet.
  - A client program reads a WSDL document to understand what it can do; data types used are embedded in the WSDL file in the form of XML schema.
  - The client then uses SOAP to actually invoke the functions listed in the WSDL document.

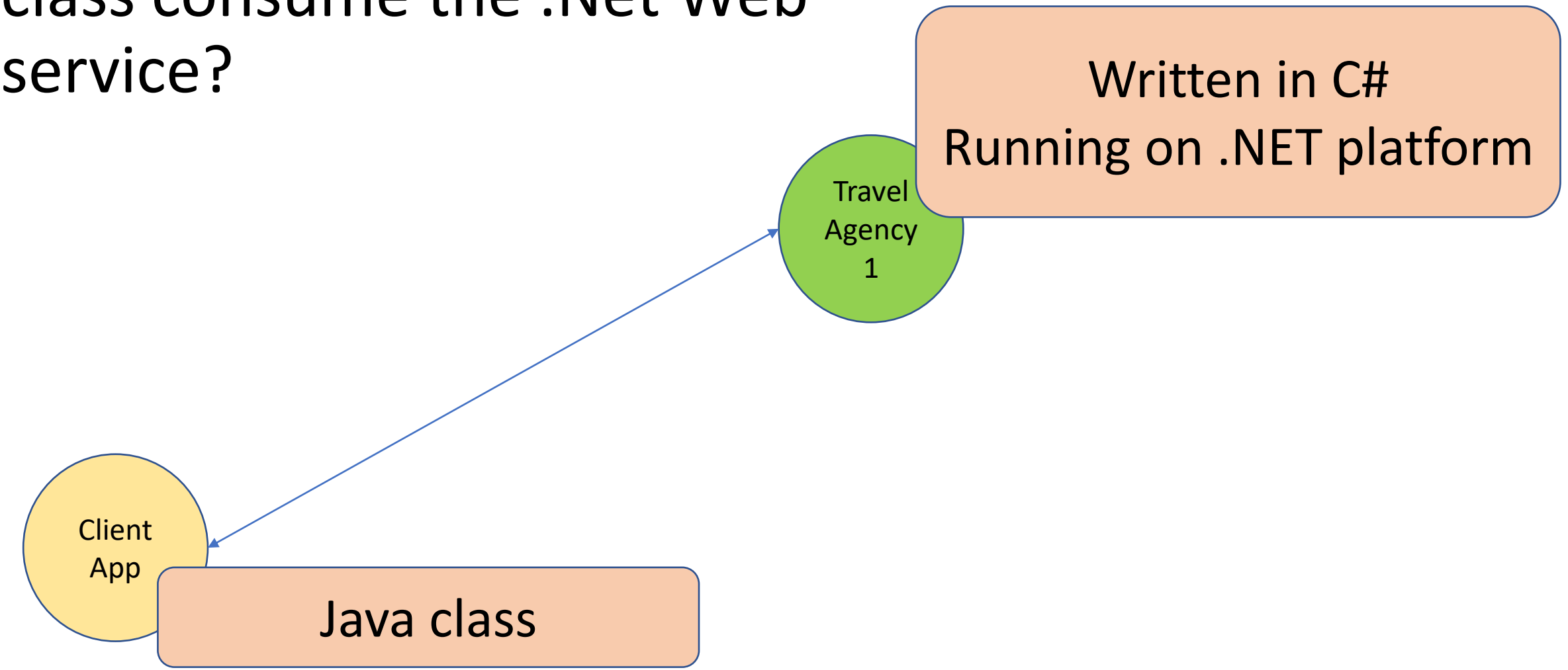
Interoperability

- Ability of services to connect and communicate with one another


Travel
Agency
1
Client
App


Written in C#
Running on .NET platform
Travel Agency 1
Client App
Java class

Interoperability - can a Java class consume the .Net Web service?



Interface - Web Services Interoperability

- Services interoperate based on a formal definition (WSDL) that is independent of the underlying platform and programming language

Interface - Web Services Interoperability

- Services interoperate based on a formal definition (WSDL) that is independent of the underlying platform and programming language
- The interface definition hides the implementation of the language-specific service

Interface - Web Services Interoperability

- Services interoperate based on a formal definition (WSDL) that is independent of the underlying platform and programming language
- The interface definition hides the implementation of the language-specific service
- SOA-based services can function independently of development technologies and platforms (Java, .NET, etc.)

WSDL file is what binds everything together

- WSDL file is written in XML
  - XML can be read by any programming language
  - Both .Net and Java have corresponding commands that have the ability to work with XML
  - If the client application was written in .Net – it would understand the XML file
  - If the client app was written in Java – it could also interpret the WSDL file

WSDL file is what binds everything together

- WSDL file is written in XML
  - XML can be read by any programming language
  - Both .Net and Java have corresponding commands that have the ability to work with XML
  - If the client application was written in .Net – it would understand the XML file
  - If the client app was written in Java – it could also interpret the WSDL file
- Web services allow multiple applications built on various programming languages to talk to each other
  - We can have a .Net web application talking to a Java application via a Web service

Web Services Interoperability

- Universal accessibility
  - Standard interface description
  - Standard communication protocols
- Can be implemented in different programming languages
- Can be implemented on different platforms

Web Services Interoperability benefits

business to business

- Facilitate B2B collaboration
  - Each organization exposes its business applications as services on the Internet and makes them accessible via standard programming interfaces

Web Services Interoperability benefits

- Facilitate B2B collaboration
  - Each organization exposes its business applications as services on the Internet and makes them accessible via standard programming interfaces
- Facilitate distributed computing and resource sharing over the Internet
  - Cross-language and cross-platform

Web Services Interoperability benefits

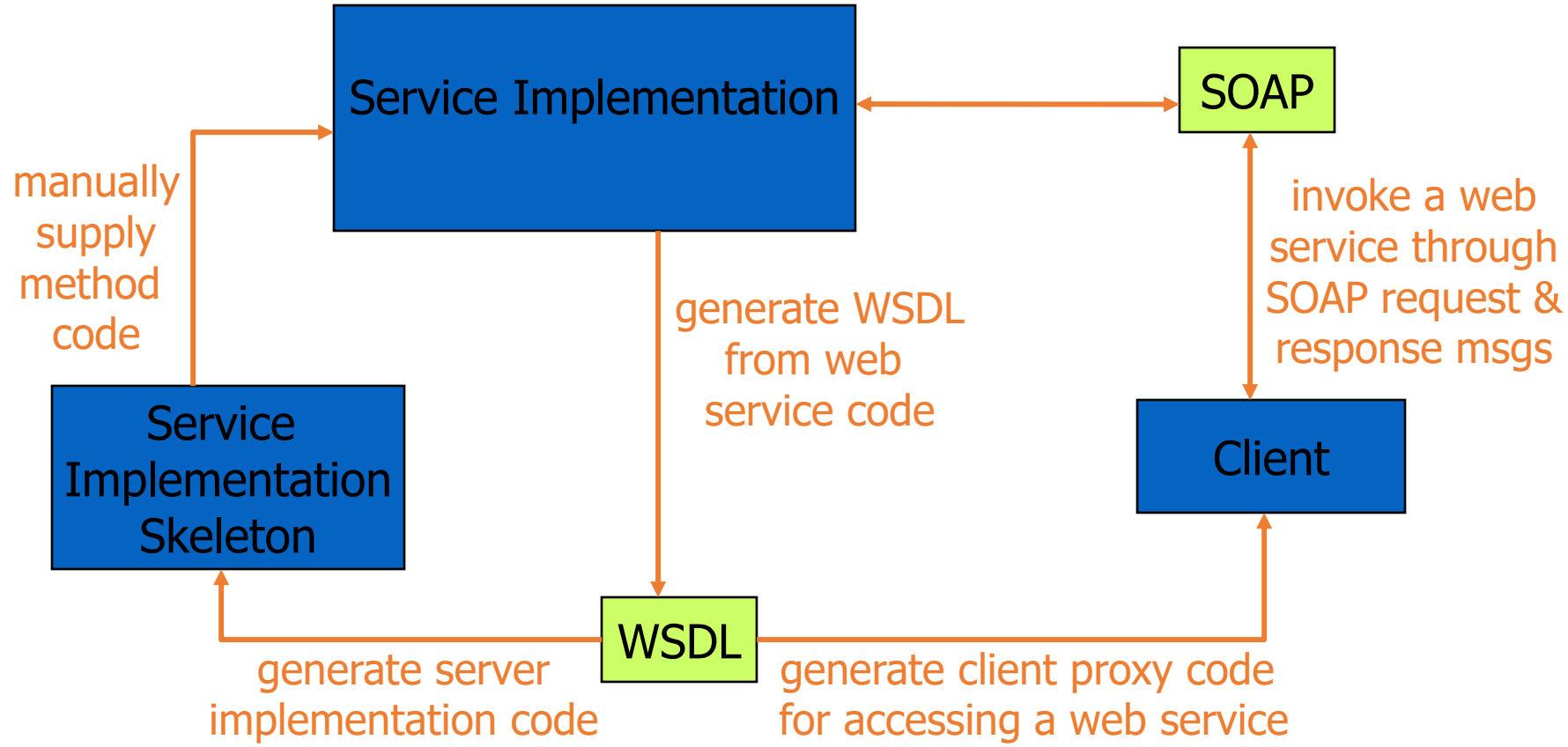
- Facilitate B2B collaboration
  - Each organization exposes its business applications as services on the Internet and makes them accessible via standard programming interfaces
- Facilitate distributed computing and resource sharing over the Internet
  - Cross-language and cross-platform
- Cost effective way to quickly develop and deploy Web applications
  - Integrate other independently published Web service components into new business processes

just call the function

Application Design

- Web service is defined in WSDL
- Top-down
  - WSDL is created (or found) first before its implementation
- Bottom-up
  - WSDL gets generated from existing JEE components
- Middle-ground

Web Services Toolkits

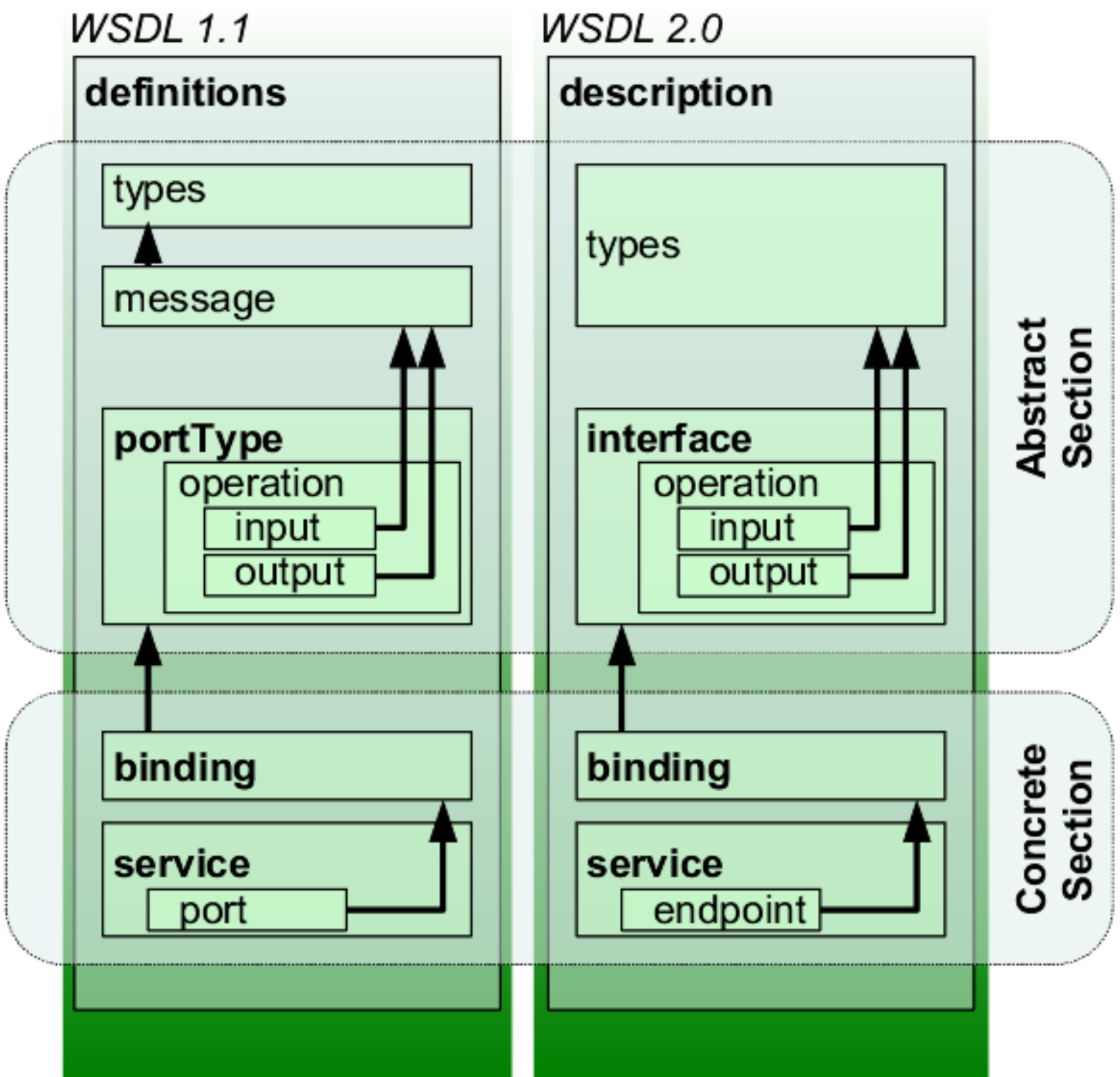


Service reusability

- Service reuse is often mentioned as an important aspect of SOA
- The aim is to create services that can be reused across a business
- Does WSDL help with reusability?

WSDL

- An XML-based interface description language
- Used for describing the functionality offered by a web service
- WSDL describes services as collections of network endpoints or ports



What is WSDL?

- Web service is described as
  - A set of communication endpoints (ports)
- Endpoint is made of two parts
  - Abstract definitions of operations and messages
  - Concrete binding to networking protocol (and corresponding endpoint address) and message encoding
- These portions are often defined in two or more files
  - Concrete file imports the abstract one
- Why this separation?
  - Enhance reusability

Authoring Style Recommendation

- To enhance **Reusability** and **maintainability**
- Maintain WSDL document in 3 separate parts
  - Data type definitions
  - Abstract definitions
  - Specific service bindings
- Use "*import*" element to import necessary part of WSDL document

Example7A: http://example.com/stockquote/stockquote.xsd

```
<?xml version="1.0"?>
<schema targetNamespace="http://example.com/stockquote/schemas"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
  <element name="TradePriceRequest">
    <complexType>
      <all>
        <element name="tickerSymbol" type="string"/>
      </all>
    </complexType>
  </element>
  <element name="TradePrice">
    <complexType>
      <all>
        <element name="price" type="float"/>
      </all>
    </complexType>
  </element>
</schema>
```

Example7B:http://example.com/stockquote/stockquote.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote/definitions"
        xmlns:tns="http://example.com/stockquote/definitions"
        xmlns:xsd1="http://example.com/stockquote/schemas"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
  <import namespace="http://example.com/stockquote/schemas"
         location="http://example.com/stockquote/stockquote.xsd"/>
   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
```

Abstract part

```
  <message name="GetLastTradePriceOutput">
     <part name="body"
  element="xsd1:TradePrice"/>
  </message>
  <portType name="StockQuotePortType">
     <operation name="GetLastTradePrice">
        <input
  message="tns:GetLastTradePriceInput"/>
        <output
  message="tns:GetLastTradePriceOutput"/>
     </operation>
  </portType>
</definitions>
```

- Here we just declare the expected elements of a message, but we do not really define here how the actual SOAP message matching this definition looks like

Example7C: http://example.com/stockquote/stockquoteservice.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote/service"
        xmlns:tns="http://example.com/stockquote/service"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:defs="http://example.com/stockquote/definitions"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
  <import namespace="http://example.com/stockquote/definitions"
         location="http://example.com/stockquote/stockquote.wsdl"/>
   <binding name="StockQuoteSoapBinding“ type="defs:StockQuotePortType">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetLastTradePrice">
```

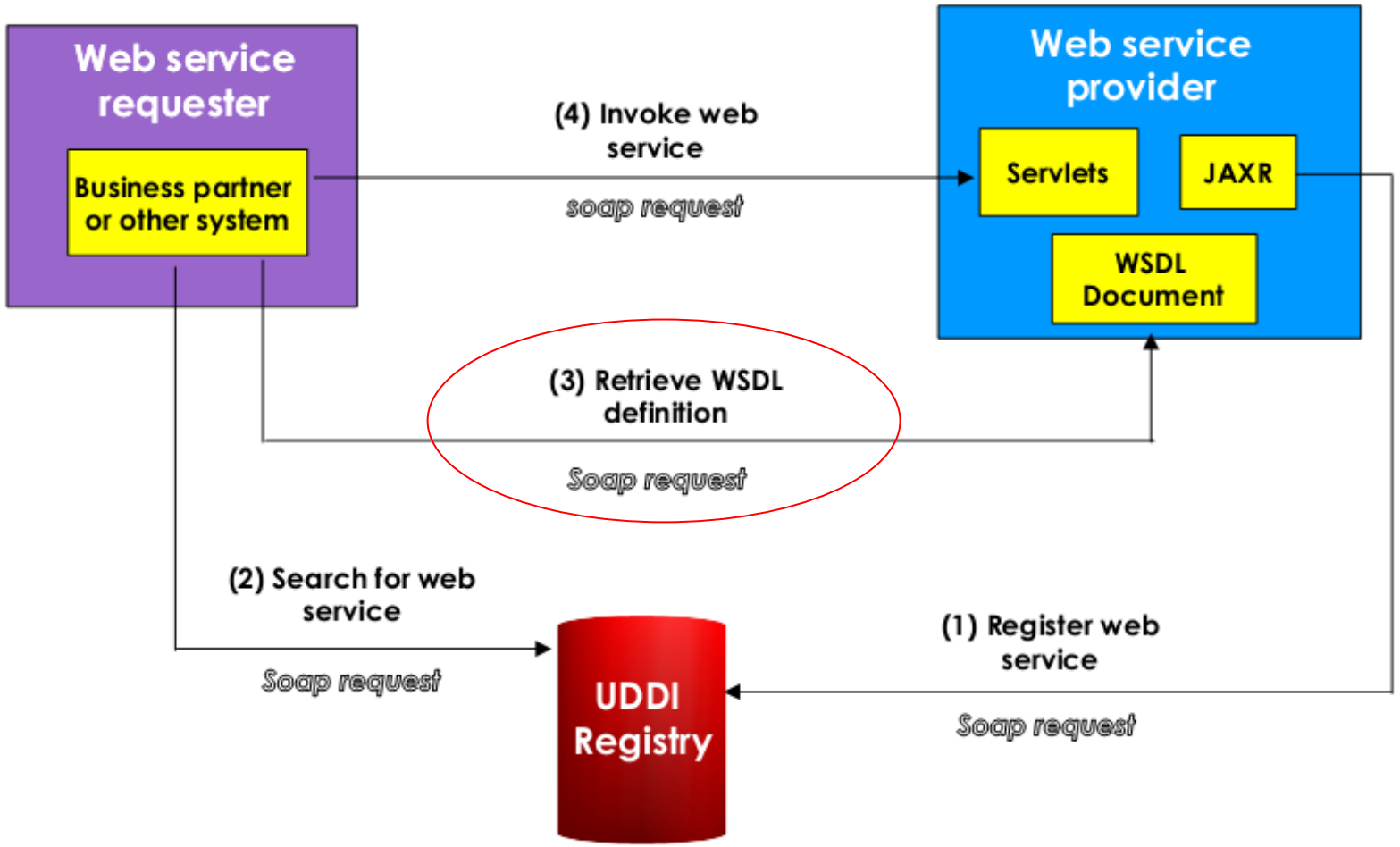
Concrete part – specific service bindings

```
 <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input><soap:body use="literal"/> </input>
    <output><soap:body use="literal"/></output>
  </operation>
</binding>
<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
```

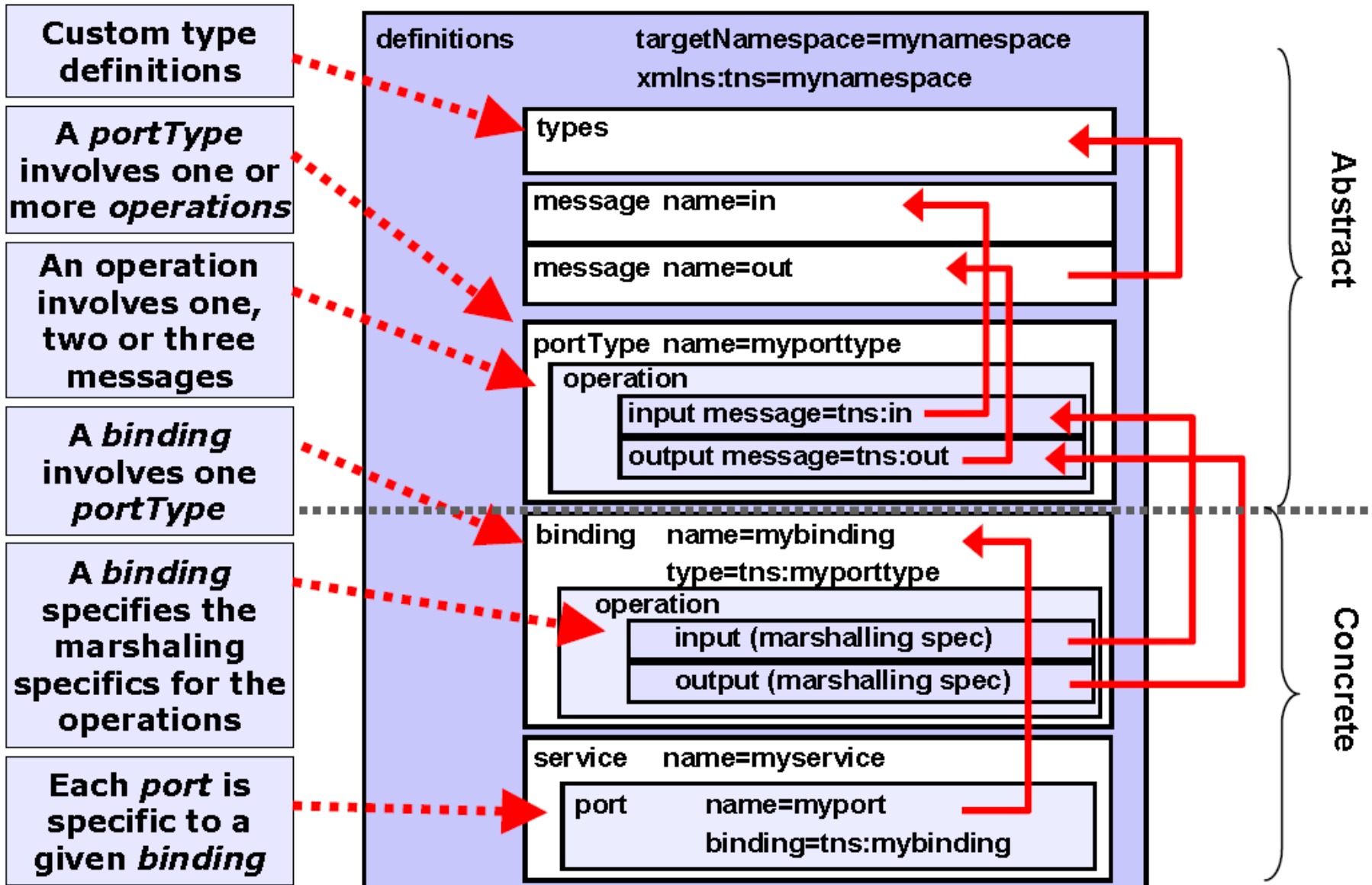
SOAP Binding

When to use What?

Where is WSDL Used?



WSDL elements



Example7C: http://example.com/stockquote/stockquoteservice.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote/service"
        xmlns:tns="http://example.com/stockquote/service"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:defs="http://example.com/stockquote/definitions"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
   <import namespace="http://example.com/stockquote/definitions"
         location="http://example.com/stockquote/stockquote.wsdl"/>
    <binding name="StockQuoteSoapBinding“ type="defs:StockQuotePortType">
      <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
        <input><soap:body use="literal"/> </input>
        <output><soap:body use="literal"/></output>
      </operation>
    </binding>
```

WSDL SOAP Binding style and mode

- **There are four (+1) combinations of WSDL SOAP Binding style and mode:**
  - rpc/encoded
  - rpc/literal
  - ~~document/encoded~~
  - document/literal
  - document/literal/wrapped

rpc/encoded

WSDL segment

```
<message name="myMethodRequrest">
    <part name="x" type="xsd:int">
    <part name="y" type="xsd:float">
</message>
<message name="empty"/>
<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequrest"/>
        <output message="empty"/>
    </operation>
</portType/>
```

SOAP segment

```
<soap:envelop>
    < soap:body >
        <myMethod>
            <x xsi:type="xsd:int">5</x>
            <y xsi:type="xsd:float">5.0</y>
        </ myMethod >
    < soap:body />
</soap:envelop >
```

rpc / literal

WSDL segment

```
<message name="myMethodRequrest">
    <part name="x" type="xsd:int">
    <part name="y" type="xsd:float">
</message>
<message name="empty"/>
<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequrest"/>
        <output message="empty"/>
     </operation>
</portType/>
```

SOAP segment

```
<soap:envelop>
    < soap:body >
        <myMethod>
            <x>5</x>
            <y>5.0</y>
        </ myMethod >
     < soap:body />
</soap:envelop >
```

document/literal

WSDL segment

```
<types>
    <schema>
        <element  name="xElement" type="xsd:int"/>
        <element  name="yElement" type="xsd:float"/>
    </schema>
</types>
<message name="myMethodRequrest">
    <part name="x" type="xElement">
    <part name="y" type="yElement">
</message>
<message name="empty"/>
<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequrest"/>
        <output message="empty"/>
     </operation>
</portType/>
```

SOAP segment

```
<soap:envelop>
    < soap:body >
        < xElement >5</ xElement >
        < yElement >5.0</yElement >
    < soap:body />
</soap:envelop >
```

document/literal/wrapped

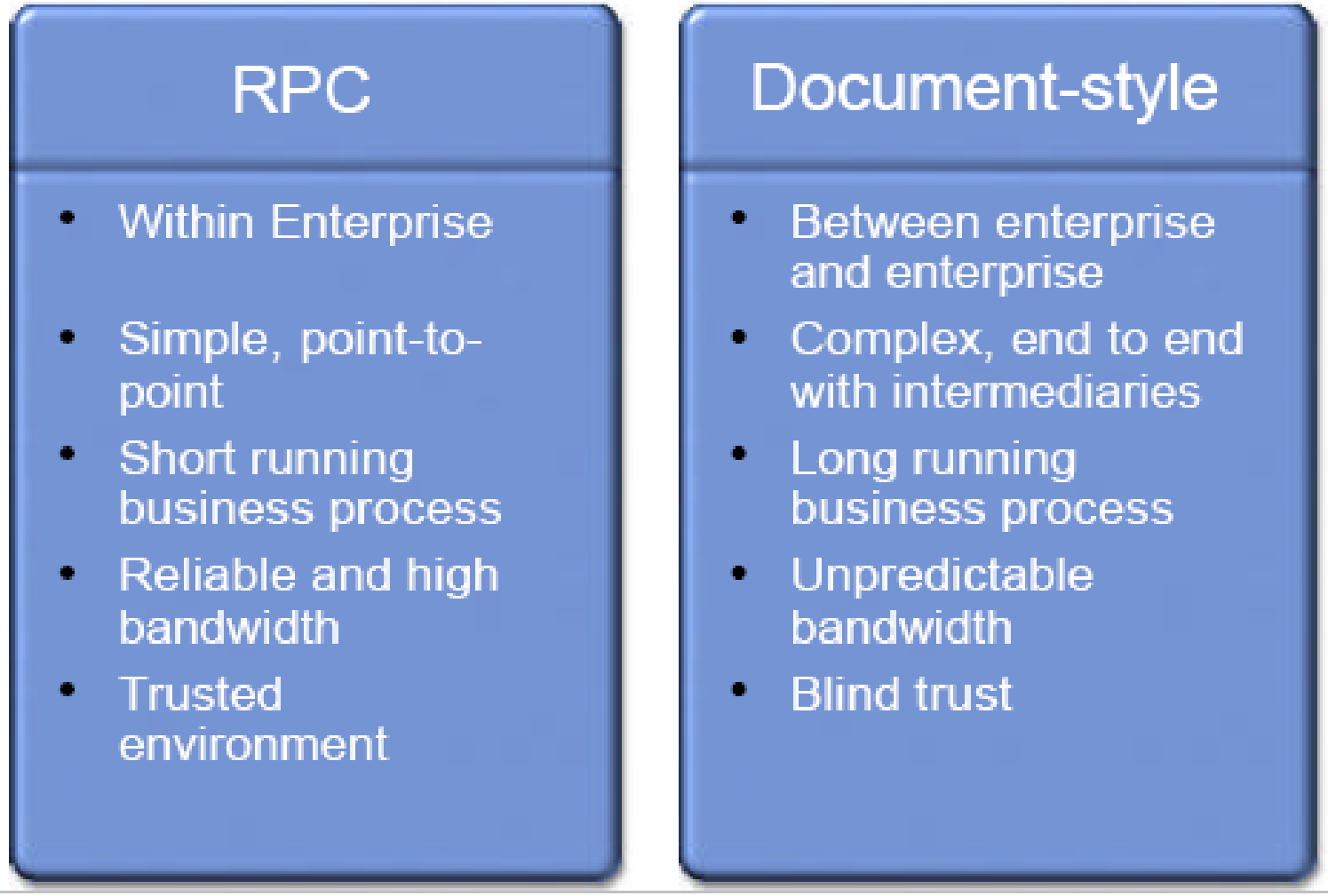
WSDL segment

```
<types>
    <schema>
        <xs:element name=" myMethodRequrest ">
            <xs:complexType>
                <xs:sequence>
                    <xs:element type="xs:int" name=" xElement " />
                    <xs:element type="xs:float" name=" yElement " />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </schema>
</types>
<message name="myMethodRequ
    <part name="part1" type="myI
</message>
<message name="empty"/>
<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequrestMessage"/>
```

SOAP segment

```
<soap:envelop>
    < soap:body >
        < myMethodRequrest >
            < xElement >5</ xElement >
            < yElement >5.0</yElement >
        < /myMethodRequrest >
    < soap:body />
</soap:envelop >
```

When to use Which model?



Module 2 Summary

- SOAP technology
- WSDL technology