Software Reuse and Component-Based SE

ITSE422

Lecture # 6 : Reuse and Composition in Service Computing

Outline

- Short review
- Concepts: Services, SOA, WebServices
- Services as reusable components
- Service engineering
- Software development with services

Main References

- Ian Sommerville, *Software Engineering*, 9th edition, chapter 19 (Service-oriented architecture)

Development with and without reuse
A funny Review:

- Problem and requirements: hungry person wants pizza.
- Solutions ?
  1. No reuse – do yourself everything from scratch
  2. Component-based development
  3. Service reuse
  4. Developing with services

Funny Review:
Case 1: Develop without reuse

Problem and requirements: hungry person wants pizza

Solution 1:
Prepare dough (mix and knead flour, yeast, salt and water)
Let dough rise
Flatten dough, shape
Make sauce (peel and chop tomatoes, mix with pepper, herbs)
Put sauce on dough
Put topping (ham, cheese)
Bake
Eat

Do everything from scratch:
Advantages: no limitations on design
Disadvantages: need skills, time, resources

Funny Review:
Case 2: Component-Based Development

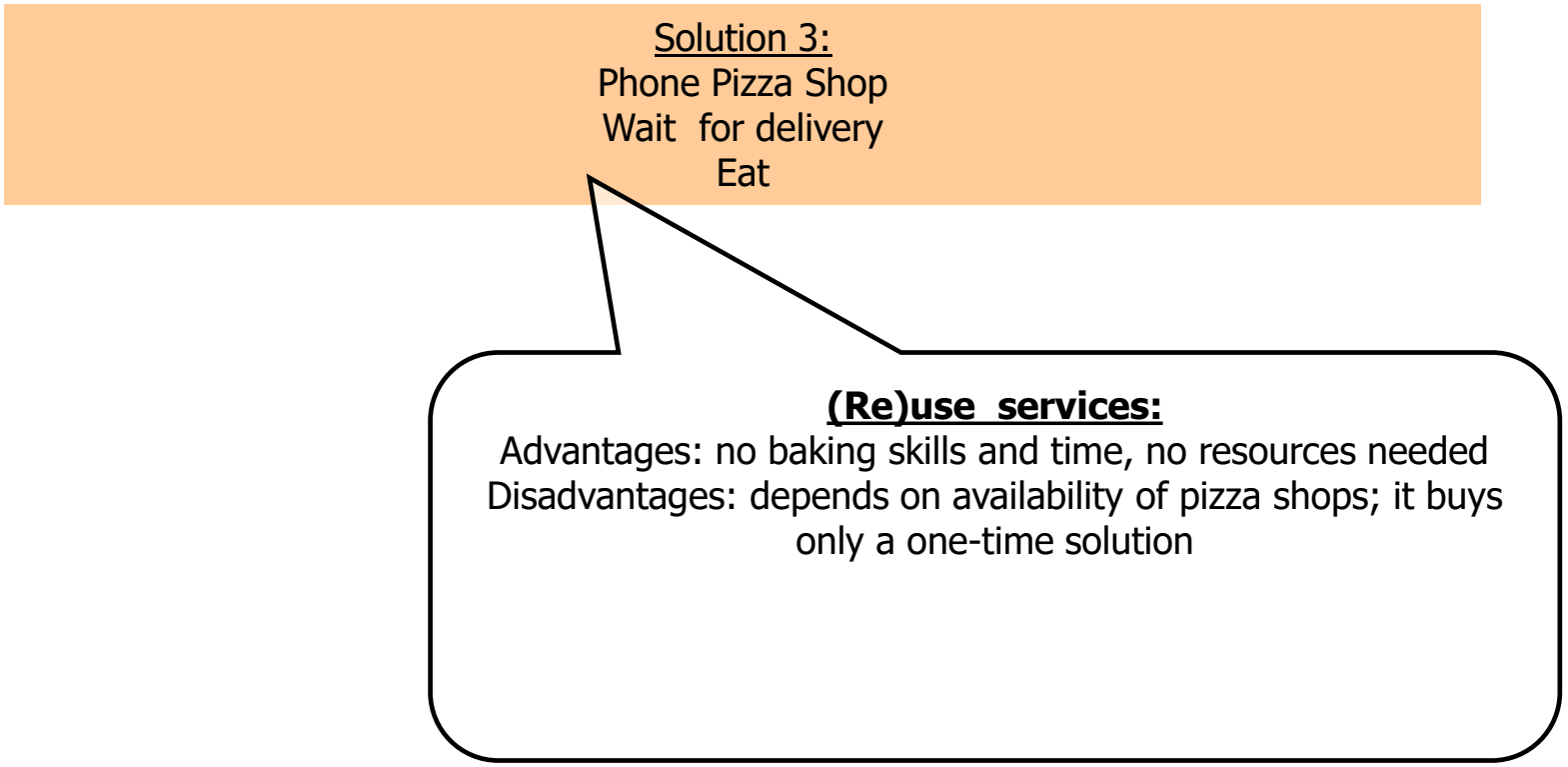
Problem and requirements: hungry person wants pizza

<u>Solution 2:</u>
Buy pizza dough
Buy ketchup
Put ketchup on dough
Put topping
Bake
Eat

<u>Build with COTS:</u>
Advantages: need less skills, need less time
Disadvantages: design limited by available components. Possible incompatibilities. Need resources

Funny Review:
Case 3: Use Services

Problem and requirements: hungry person wants pizza



Funny Review:
Folow-Up: Developing with services

- After successfully using pizza delivery services for its own use, our guy gets to the idea to start a new business, the Party Organizer Service:

Follow-up Solution:

Put together a party organizer service, by combining (composing and coordinating) external services:

- Pizza delivery service
- Taxi service
- Cleaning service

Developing with services: The Party Service Company has only a phone and an agenda, it does not do anything itself: it just coordinates the use of external services

Review: What are the "Software Entities" to compose and reuse?

- Functions
- Modules
- Objects
- Components
- Services
- …

1960
1970
1980
1990
2000
2010

1968: Douglas McIlroy: "*Mass Produced Software Components*"

1998: Clemens Szyperski: "*Component Software – Beyond Object Oriented Programming*"

Review: Objects-Components-Services

Entities for Reuse and Composition		
•Abstraction •Encapsulation		
Objects	**Components**	**Services**
•Location: same process •Inheritance •Polymorphism	•Location: different processes, same environment •Usually some runtime infrastructure needed •Provided and required interfaces •Are deployed at the consumer premises	•Location: different environments •More emphasis on interface/contract/service agreement •Mechanisms for dynamic discovery •Dynamically composable •Are deployed at the producer premises

Outline

- Short review
- Concepts: Services, SOA, WebServices
- Services as reusable components
- Service engineering
- Software development with services

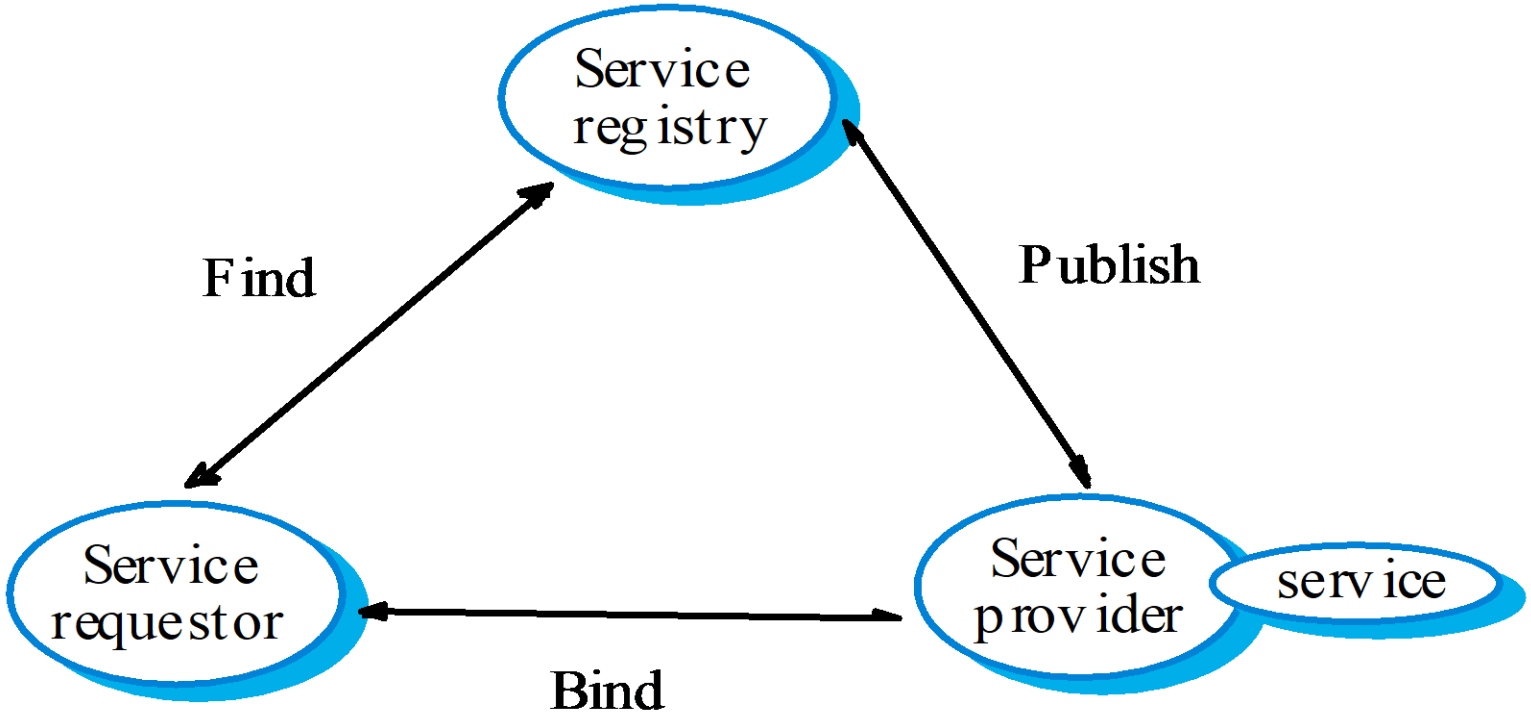
What is a service ?

- *An act or performance offered by one party to another. Although the process may be tied to a physical product, the performance is essentially intangible and does not normally result in ownership of any of the factors of production.*
- Service provision is therefore independent of the application using the service.

Service-oriented architectures

- A means of developing distributed systems where the components are stand-alone services
- Services may execute on different computers from different service providers
- Standard protocols have been developed to support service communication and information exchange
- Services are platform and implementation-language independent
- Registries enable the discovery of services

Characteristics of Primitive SOA



Benefits of services

- Provider independence.
- Public advertising of service availability.
- Potentially, run-time service binding.
- Opportunistic construction of new services through composition.
- Pay for use of services.
- Smaller, more compact applications.
- Reactive and adaptive applications.

Benefits of SOA

- Services can be provided locally or outsourced to external providers
- Services are language-independent
- Investment in legacy systems can be preserved
- Inter-organizational computing is facilitated through simplified information exchange

Services standards

- Services are based on agreed standards so can be provided on any platform and written in any programming language.
- Web services: one kind of services
- Thomas Erl: common misperceptions about SOA:
  - "An application that uses web services is service-oriented"
  - "SOA is just a marketing term used to re-brand web services"

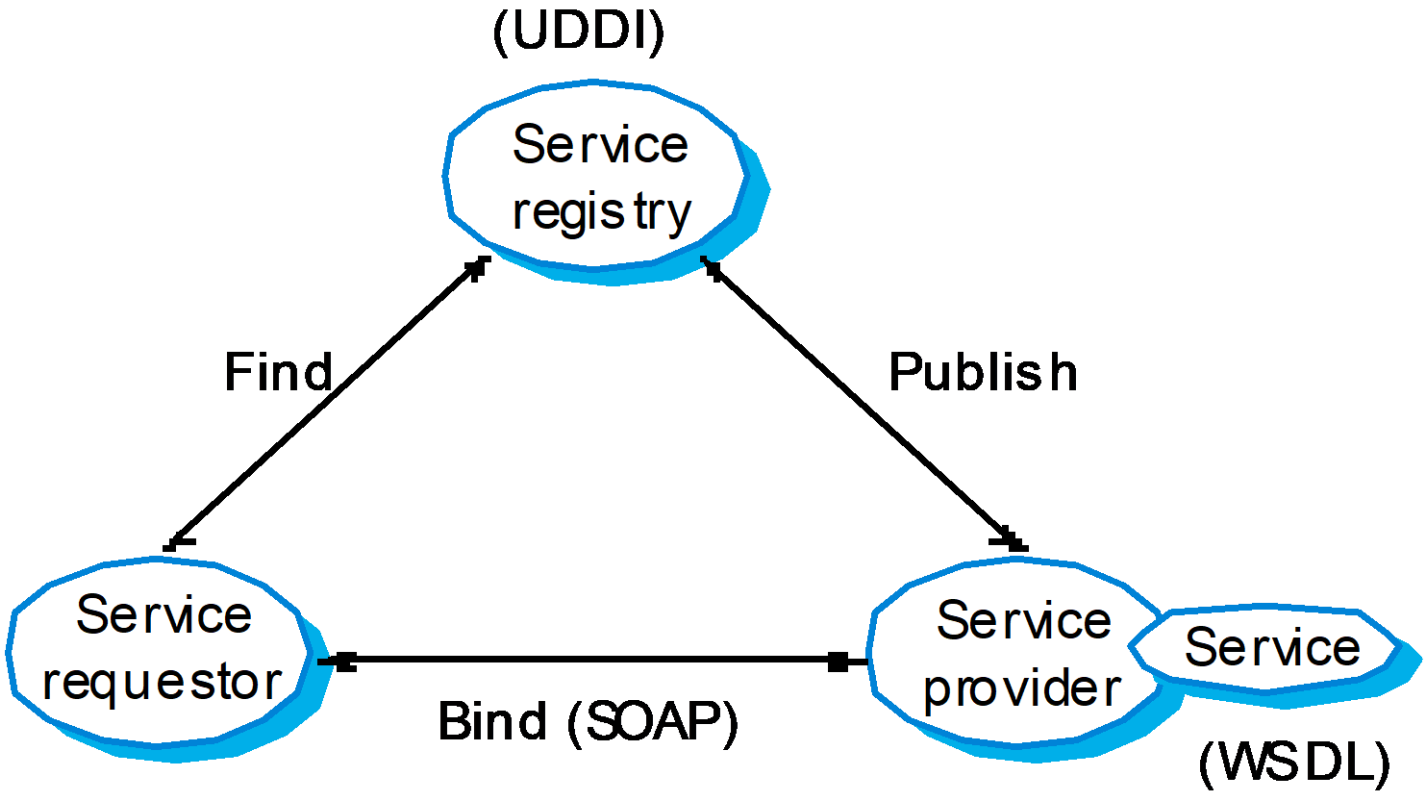
Web Services Definition by W3C

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

Web Services

- The Web Services initiative has been driven by standards from its beginning (vs Components where standardisation has been tried later and several different standards=component models are in use)
- Key standards
  - XML – Extensible Markup Language
  - SOAP - Simple Object Access Protocol;
  - WSDL - Web Services Description Language;
  - UDDI - Universal Description, Discovery and Integration.

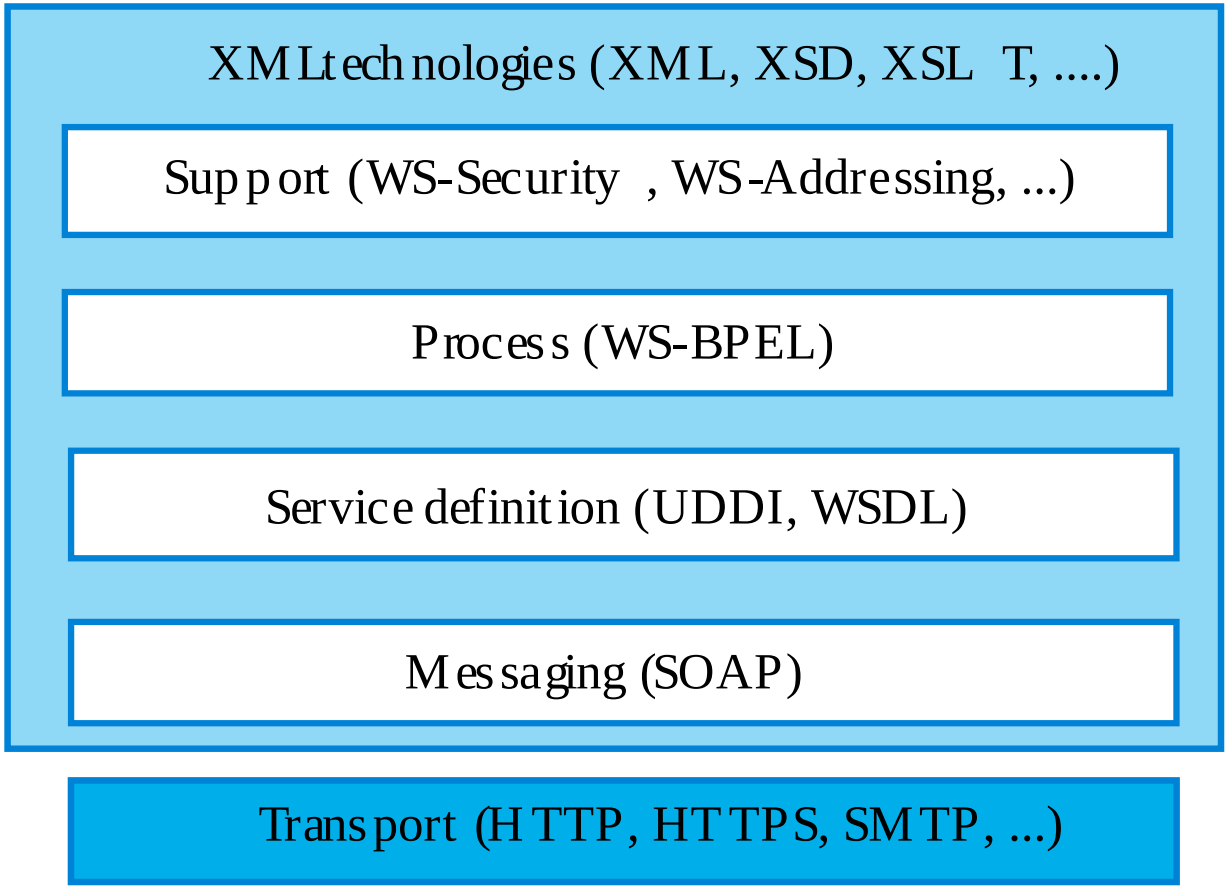
SOA based on Web Services



Key standards

- SOAP
  - A message layout standard that supports service communication
  - Defines a uniform way of passing XML-encoded data
  - Defines a way to bind HTTP as the underlying communication protocol
- WSDL (Web Service Definition Language)
  - This standard allows a service interface and its bindings to be defined
- UDDI
  - Defines the components of a service specification that may be used to discover the existence of a service
- WS-BPEL(Business Process Execution Language)
  - A standard for workflow languages used to define service composition

More Web Services Standards



Standards organizations that contribute to SOA / WS

- The World Wide Web Consortium W3C:
  - Goal: to further the evolution of the web, by providing fundamental standards that improve online business and information sharing
  - XML, XMLSchema, WSDL, SOAP
- Organisation for the Advancement of Structured Information Standards OASIS
  - Goal: to promote online trade and commerce via specialized Web services standards
  - UDDI, WS-BPEL, WS-Security
- Major vendors that contribute to SOA
  - Microsoft, IBM, BEA Systems, Sun, Oracle, Tibco, Hewlett-Packard, Canon

Outline

- Short review
- Concepts: Services, SOA, WebServices
- Services as reusable components
- Service engineering
- Software development with services

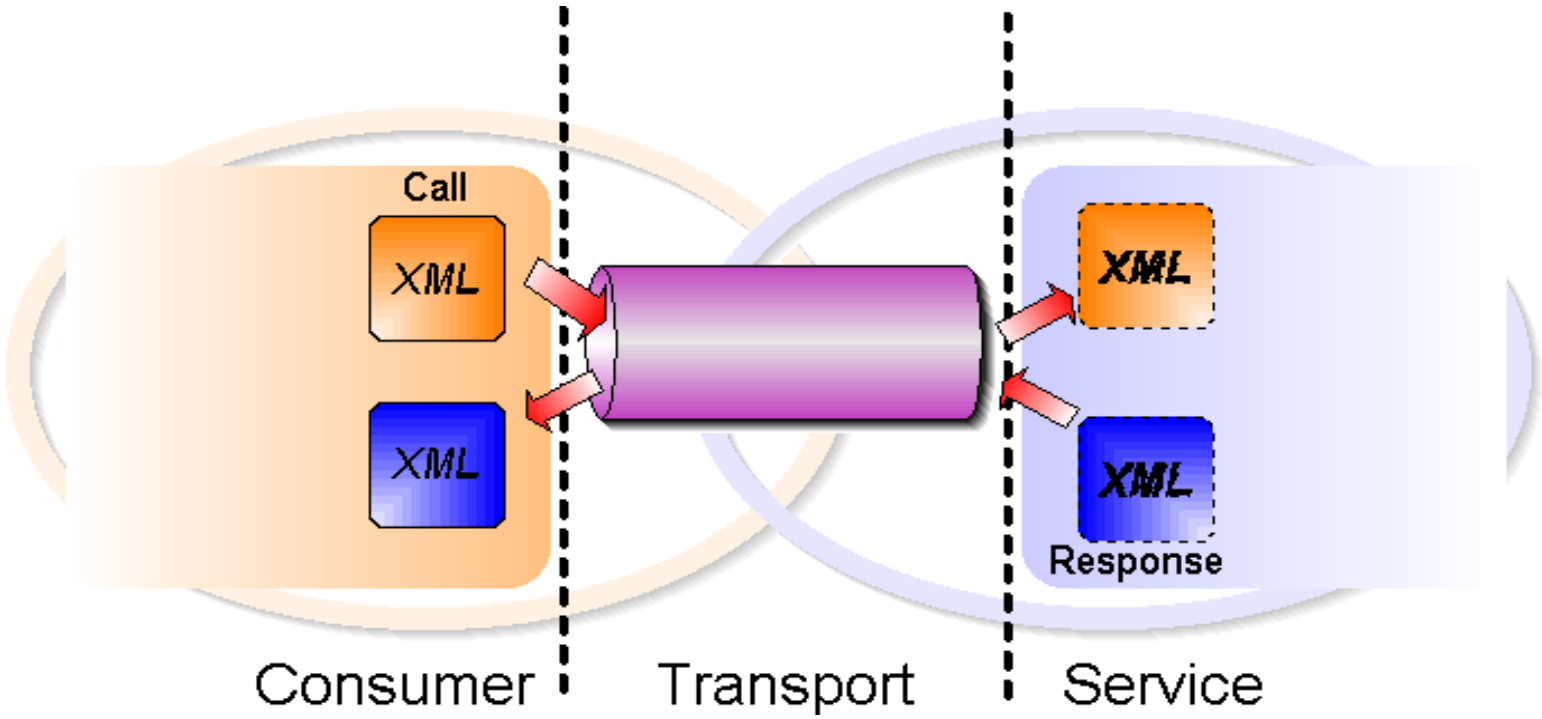
Services as reusable components

- A service can be defined as:
  - *A loosely-coupled, reusable software component that encapsulates discrete functionality which may be distributed and programmatically accessed. A web service is a service that is accessed using standard Internet and XML-based protocols*

Services as reusable components

- Abstraction and encapsulation: WSDL service description ('provides' interface) and service implementation
  - In order to use a service, a client needs only the WSDL
- A critical distinction between a service and a component as defined in CBSE is that services are independent
  - Services do not have a 'requires' interface
  - Services rely on message-based communication with messages expressed in XML (not on method calls)

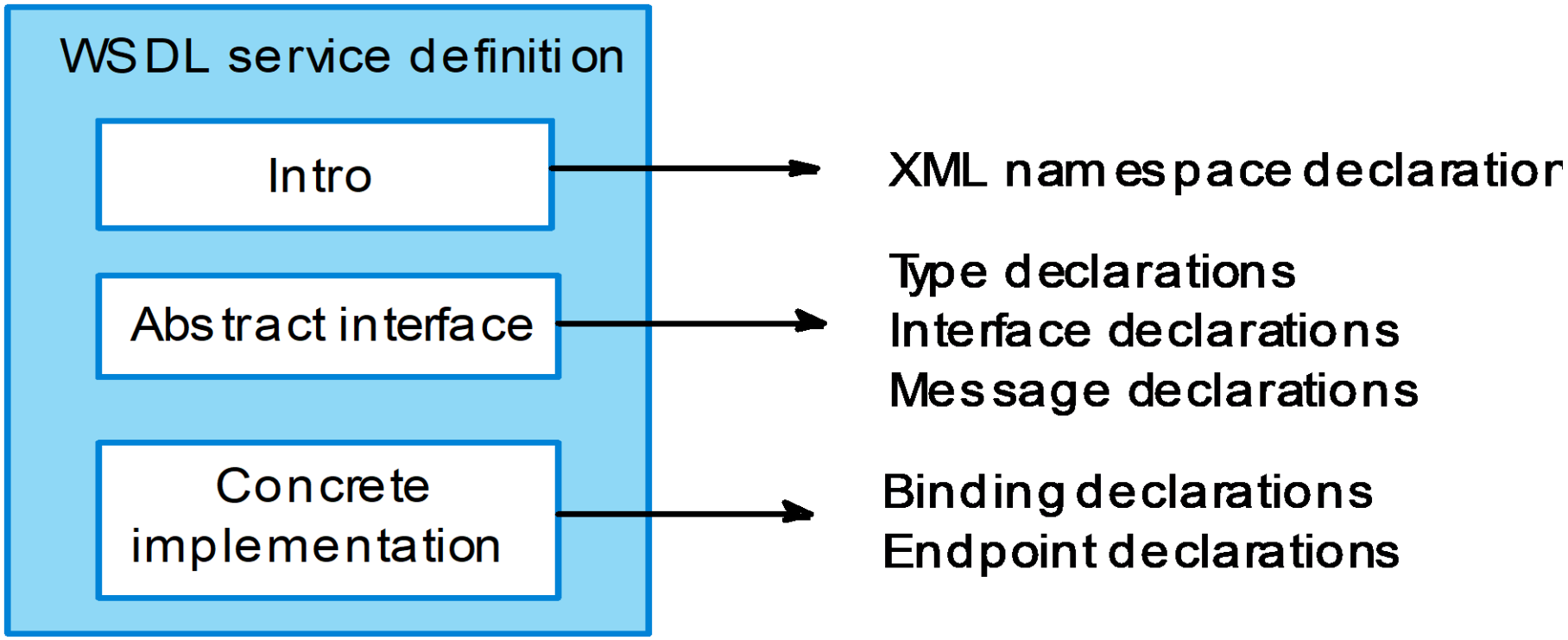
Message-based communication



Web service description language (WSDL)

- The service interface is defined in a service description expressed in WSDL. The WSDL specification defines
  - **What** operations the service supports and the format of the messages that are sent and received by the service
  - **How** the service is accessed - that is, the binding maps the abstract interface onto a concrete set of protocols
  - **Where** the service is located. This is usually expressed as a URI (Universal Resource Identifier)

Structure of a WSDL specification



Main elements of a WSDL document

- XML elements in its description:
  - Abstract interface:
    - <portType> (renamed <interface> in WSDL 2.0)
    - <operation>
    - <message>
    - <types>
  - Concrete implementation:
    - <binding>
    - <port> (renamed <endpoint> in WSDL 2.0)
    - <service>

WSDL document structure

```
<definitions>

    <portType>

    </portType>

    <message>

    </message>

    <types>

    </types>

    <binding>

    </binding>

</definitions>
```

<portType> Element

- This is probably the most important element
- Describes the web service
  - Operations that can be performed
  - Messages that are involved
  - Comparable to a function/method library in a programming language
  - A port is defined by associating a network address with a reusable binding

<message> Element

- Defines the data elements of an operation
- Each message can have one or more parts
- Each part is comparable to a function/method call in a programming language

Simplified Example

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

Types of operations

- One-way
  - Can receive a message, but will not return a message
  - Example use: receiving request to insert a new value in a database
- Request-response
  - Can receive a request and will return a response
  - Example use: receiving a request for a value from a database and sending it back in a response
- Solicit-response
  - Can send a request and will wait for a response
  - Example use: requesting a value from a database and having it sent back in the response
- Notification
  - Can send a message, but will not wait for a response
  - Example use: inserting a new value in a database

Example of one-way operation

```
<message name="newTermValues">
   <part name="term" type="xs:string"/>
   <part name="value" type="xs:string"/>
 </message>

 <portType name="glossaryTerms">
   <operation name="setTerm">
     <input name="newTerm" message="newTermValues"/>
   </operation>
 </portType >
```

<binding> Element

- One binding represents one possible transport technology the service can use to communicate
- A binding can apply to an entire interface or just a specific operation
- Has two attributes:
  - Name – Can be set to any value, which represents the name of the binding
  - Type – Points to the port (interface) for the binding
- When using SOAP, a <soap:binding> sub-element is used to set the style and transport values with elements:
  - Style – with value of either "rpc" or "document"
  - Transport – defines the SOAP protocol to use (like HTTP)

SOAP binding example

```
<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
  <soap:binding style="document"
  transport="http://schemas.xmlsoap.org/soap/http" />
    </operation>
</binding>
```

<port> (<endpoint>) Element

- A <port> or <endpoint> represents the physical address at which a service (interface) can be accessed with a specific protocol
- A <service> groups a set of related endpoints

Endpoint example

```
<service name="GlossaryTermsService">
      <port name="GlossaryTermsSoap1" binding="b1">
         <soap:address location="http://myserver.com/WebServices/GlossaryTerms1.asmx " />
      </port>
   </service>
```

IDL model for Web Services

- WSDL acts as an IDL for Web Services distributed programming model
- Definitions are processed by an IDL compiler to generate:
  - stubs for clients which look like local function calls
  - Dispatch routines for the server that invoke the developer's code
- Tools can generate WSDL descriptions from implementations (Bottom-up approach)
- Tools can generate implementation stubs from WSDL (Top-down approach)
- Technology examples:
  - Java: Axis: wsdl2java, java2wsdl
  - .NET: wsdl.exe

Outline

- Short review
- Concepts: Services, SOA, WebServices
- Services as reusable components
- Service engineering
- Software development with services

Service-oriented software engineering

- Existing approaches to software engineering have to evolve to reflect the service-oriented approach to software development
  - Service engineering. The development of dependable, reusable services
    - Software development for reuse
  - Software development with services. The development of dependable software where services are the fundamental components
    - Software development with reuse

Service engineering

- The process of developing services for reuse in service-oriented applications
- The service has to be designed as a reusable abstraction that can be used in different systems
- Involves
  - Service candidate identification
  - Service design
  - Service implementation

Service implementation and deployment

- Programming services using a standard programming language
  - Bottom-up approach: write class, generate WSDL
  - Top-down approach: have WSDL, generate class stub
- Services then have to be tested by creating input messages and checking that the output messages produced are as expected
- Deployment involves:
  - installing it on an application server
  - Optionally, publicising the service (using UDDI)

Outline

- Short review
- Concepts: Services, SOA, WebServices
- Services as reusable components
- Service engineering
- Software development with services

Next lecture!

- Questions?

Contemporary SOA

- Complex applications require more than the provider-requester-registry triangle
- Activities: a task performed by a set of interacting services
- Coordination: complex activities need context data and the subsequent need for this data to be managed at runtime
- Atomic transactions: being able to guarantee an outcome of an activity
- Business activities: manage complex long-running activities that can vary in scope and the participating services
- Orchestration: an organization- speciffic business workflow; an organization owns and controls logic behind composition
- Choreography: there is no single owner of the collaboration logic

Web Service Description Language (WSDL)

- An Interface Definition Language (IDL)
- An IDL is needed when languages differ
- Other example IDL's:
  - Corba IDL (Object-oriented syntax)
  - OSF's DCE (C like syntax)
  - DCOM IDL based on OSF's DCE and used by Microsoft's DCOM
  - Sun XDR (An IDL for RPC)