Web Service Description Language (WSDL)

References

1. Grid Computing, Joshy Joseph, Craig Fellenstein, Pearson Education
2. Web Services, Alonso, Springer

Web Service Description Language (WSDL)

- WSDL was originally created by IBM, Microsoft, and Ariba by merging three previous proposals:

1. Microsoft's SOAP Contract Language (SCL)
2. Microsoft's Service Description Language (SDL) together with
3. IBM's Network Accessible Service Specification Language (NASSL)

Web Service Description Language (WSDL)

Quick Overview of WSDL 1.1

- WSDL is an XML Infoset-based document, which provides a model and XML format for describing Web services. This enables services to be described, and enables the client to consume these services in a standard way without knowing much on the lower level protocol exchange binding including SOAP and HTTP.
- This high-level abstraction on the service limits human interaction and enables the automatic generation of proxies for Web services, and these proxies can be static or dynamic. WSDL allows the description of both document-oriented and RPC-oriented messages.

Web Service Description Language (WSDL)

- WSDL has a role and purpose similar to that of IDLs in conventional middleware platforms.
- In WSDL, specifications XML documents that describe Web services, and in particular describe service interfaces.
- Existing IDLs are tied to a concrete middleware platform, they are only concerned with the description of the service interface in terms of service name and signature (input and output parameters).
- The rest of the information is implicit, as the access mechanisms are identical for all services available on that middleware platform.

Web Service Description Language (WSDL)

- In the context of Web services, each service can be made accessible using different protocols, and it is therefore crucial that such information is provided as part of the service description. (SOAP supports bindings to different protocols)
- Another difference caused by the lack of a common middleware platform is the need for defining the location at which the service is available.
- In conventional middleware the service provider could simply implement an interface and register the implemented object with the middleware, which would take care of activating the object as needed. The location of object would be transparent and unknown to the client.

Web Service Description Language (WSDL)

- The absence of a centralized platform in Web services means that the client (or the Web services middleware at the client's site) should be able to identify the location at which the service is made available, so that is knows where to send SOAP messages. Therefore, service providers should have a standardized means of specifying this information
- The separation of interfaces and bindings to protocols, and the need for address information, also make the case for modular specifications. We can imagine that different services will want to implement the same interface, but provide the service at different addresses and be able to interact with different transport protocols.

Web Service Description Language (WSDL)

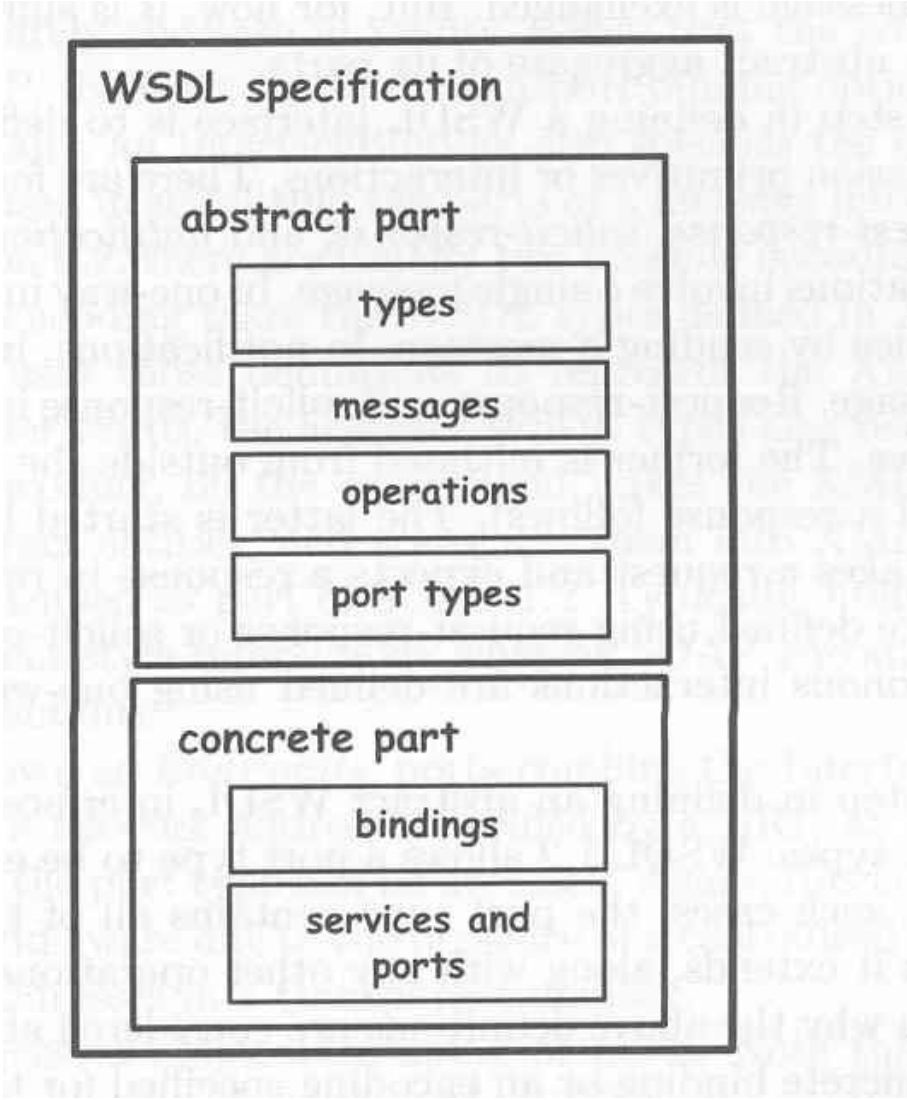
- Conventional IDLs are typically used to describe a single entry point to a service (a single RPC interaction). The invocation of a Web service typically involved exchange of several asynchronous messages between requesters and providers.
- To support these needs, WSDL includes a collection of different interaction paradigms, along with the ability to combine operations or groups of operations within an interface.

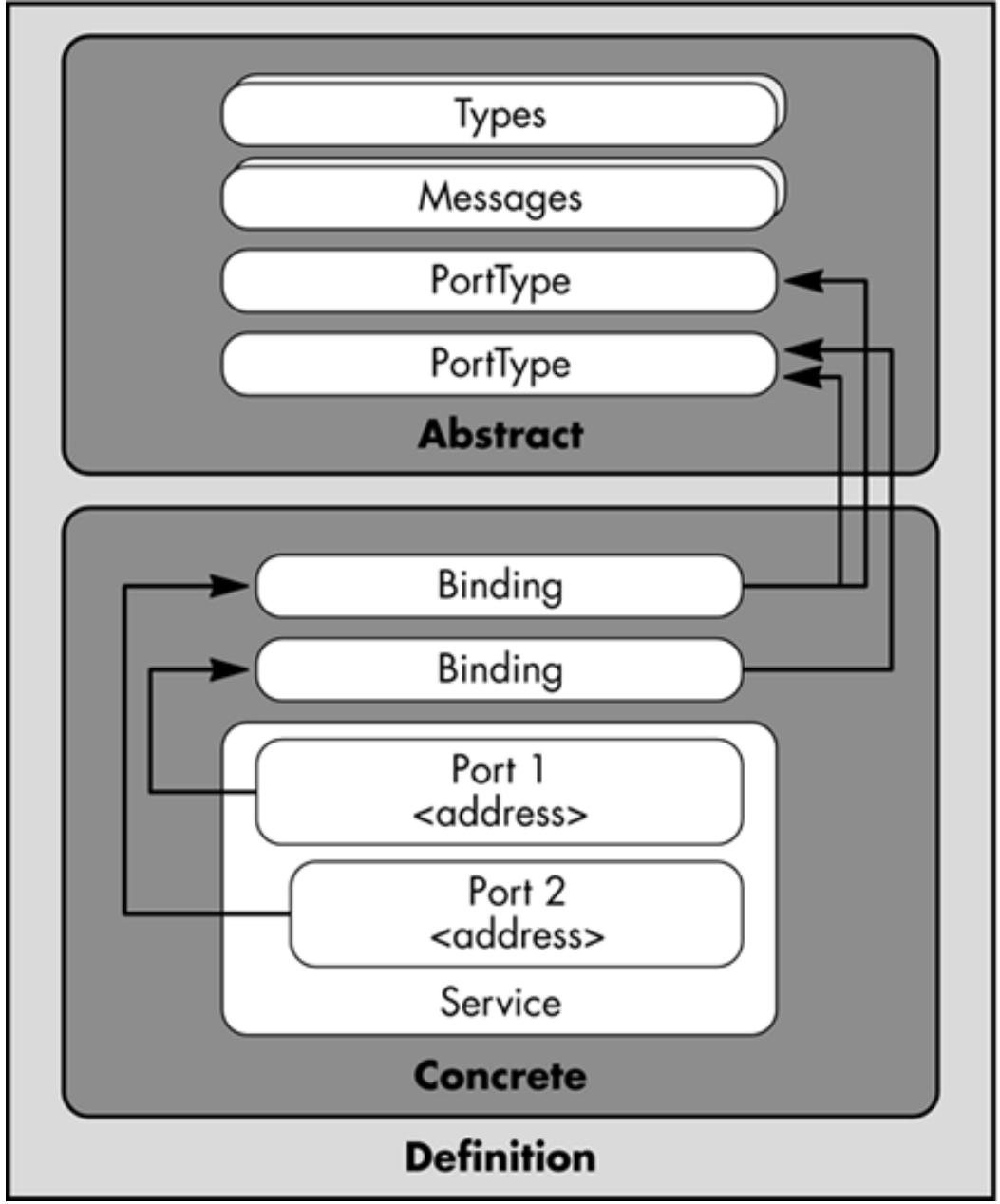
Structure of WSDL

- As shown in the next figure, a WSDL document can be divided into abstract definitions and agreed-upon definitions.
- The abstract section defines the SOAP messages in a platform-independent language and a neutral manner. The abstract definitions help to extend service definitions and enhance the reusability where the agreed-upon definition enables multiple protocol bindings such as HTTP and SMTP and end points of choice (e.g., service end points).

Structure of WSDL

- The abstract service definition components are Types, Messages, Operations, PortType, and Binding. Agreed-upon definition components are Service, Port, and Binding.
- These agreed-upon definitions specify the wire message serialization, transport selection, and other protocol implementation aspects.
- The WSDL binding elements are used to specify the agreed-upon grammar for input, output, and fault messages. There are specific binding extensions defined by WSDL for SOAP and HTTP.


WSDL specification
abstract part
types
messages
operations
port types
concrete part
bindings
services and ports



The WSDL 1.1 abstract and agreed-upon components

Structure of WSDL

- There can be two kinds of message encoding; that is, literal encoding and SOAP encoding.
- *Literal encoding* specifies that the messages will be constructed according to the XML schema "literal," and SOAP encoding occurs according to the SOAP-encoding rules "encoded," defined in the SOAP specification.

Web Service Description Language

- The other important concepts in WSDL are the message transfer format; messages can be transferred as "document" or "rpc" parameters.
- *Document* means that the message part is the body of the message, while RPC means that the parts are just parameters of the call, with specific signatures that are wrapped in an element.
- These transfer formats confirm to the SOAP semantic of the SOAP message exchange.

Structure of WSDL

- WSDL specifications are often characterized by an abstract part, conceptually analogous to conventional IDL, and a concrete part, that defines protocol binding and other information.
- The abstract part is made of port type definitions, which are analogous to interfaces in traditional middleware IDLs. Each port type is a logical collections of related operations.
- Each operation defines a simple exchange of messages. A message is a unit of communication with a Web service, representing the data exchanged in a single logical transmission. In case of Web services standards, these constructs are all defined in XML.

Structure of WSDL

- WSDL needs a type system so that the data being exchanged can be correctly interpreted at both ends of the communication. WSDL uses the same type system as XML schemas, although WSDL document can specify a different type system if necessary.
- In case of IDLs in conventional middleware, the type system is imposed by underlying platform.
- XML schemas have built-in basic data types and allow users to define more complex data types such as structures.
- So, the first step in defining a WSDL interface is to identify and define all data structures that will be exchanged as parts of messages between applications.

Structure of WSDL

- The second step is to define messages that build on such data structures. In WSDL, eah message is a types document divided into parts. Each part is characterized by a name and by a type, referring to a type typically defined in XML schema.
- For example, to invoke a procedure that takes one integer and one floating point number as parameters, we can define a message that has two parts. One will contain the integer and another will contain the floating point number. Integer and floating point numbers are part of standard XML Schema specification.
- We need to define a message that requires to exchange purchase orders, then data structures that represent product items and purchase orders have to be defined.
- The concrete relationship between the message and its parts is dependent on the interaction style of the operation in which the message is exchanged.

Structure of WSDL

- The third step in defining a WSDL interface is to define operations, also called transmission primitives or interactions. There are four basic operations:
- one-way, request-response, solicit-response, and notification.
- one-way and notification operation involves a single message. In one-way interaction, the client invokes a service by sending a message. In notifications, it is the service that sends the message

Structure of WSDL

- request-response: involve two messages, it initiated from outside the service, i.e. the service is invoked and a response follows.
- solicit-response: involve two messages, is stared by the service itself, the service makes a request and expects a response in return.
- Synchronous interactions are defined using request-response or solicit-response. Asynchronous interactions are defined using one-way and notification operations.

Structure of WSDL

- The final step in defining abstract WSDL interface is to group operations into port types. WSDL 1.2 allows a port type to be extended from other port types. In such cases, the port type contains all of the operations from the port types it extends, along with any other operations it defines.
- The reason why the above definitions are considered abstract is that there is neither a concrete binding or an encoding specified for these constructs nor a definition of a service that implements a set of port types.
- For example, in defining a real instance of a service, one has to define the exact set of port types that it implements, the transport bindings used in implementing such port types, and the address at which those implementations are made accessible to clients.
- The lack of these descriptions makes the port type definitions abstract.

Structure of WSDL

- The same port type can be implemented using multiple bindings and can be combined with other port type implementations in different combinations to form different services.
- Similarly, the above definition of messages is abstract since information about their XML encoding and protocol binding is absent. The same data type can be encoded into a message using different rules and the same message can be exchanged using different protocol bindings. The second part of a WSDL interface definition is to define a concrete service by specifying all of these aspects.

Structure of WSDL

- The concrete part of a WSDL interface is defined using the following three constructs:

InterfaceBindings

- the binding specifies the message encoding and protocol bindings for all operations and messages defined in a given port type. For instance, it could specify that an operation is an RPC-style or a document-style operation. It could also specify that the messages of an operation have to be communicated using SOAP protocol and HTTP transport bindings.
- WSDL allows communications protocols other than SOAP to be used as well.
- It also specifies the encoding rules that should be used in serializing the parts of a message into XML.

Structure of WSDL

- There are two usual encodings: literal and SOAP.
- Literal encoding takes the WSDL types defined in XML schemas and “literally” uses those definitions to represent the XML content of messages. The abstract WSDL types also become concrete types.
- SOAP encoding takes the XML schema definitions as abstract entities, and translate them into XML using SOAP encoding rules defined as part of SOAP 1.2
- Typically, literal encoding is used for document-style interactions, whereas SOAP encoding is used for RPC-style interaction.

Structure of WSDL

Ports

- Also known as Endpoints, ports combine the InterfaceBinding in formation with a network address (specified by a URI) at which the implementation of the port type can be accessed.
- Again, this is not needed in conventional middleware due to the presence of a centralized infrastructure that manages addresses in a transparent manner.

Structure of WSDL

Services

- Services are logical grouping of ports. Note that, at least in principle, this also means that a specific WSDL service could be available at different Web addresses (for example, different URIs corresponding to different server machines, may be one in Europe and one in Australia).
- It could also combine very different port types. In practice, however, it is likely that a WSDL service will group related ports, typically available at the same address.
- Another common grouping is the one in which the different ports represent different bindings for the same port type. This allows the same functionality to be accessible via multiple transport protocols and interaction styles.

Structure of WSDL

- As each of the above constructs are added to WSDL, the interface definition of a Web service becomes more concrete. With the binding information, users know what protocols to use, how to structure XML messages to interact with a service, and what to expect when contacting the service.
- WSDL 1.1 defines binding extensions for SOAP, HTTP, GET, POST and MIME.
- With the port information, users know the network address at which the functionality of a port type is implemented.
- Finally, with the service definition, users know all the ports that are implemented as a group.

A WSDL service specification

```
<?xml version="1.0"?>
<definitions name="Procurement"
   targetNamespace="http://example.com/procurement/definitions"
   xmlns:tns="http://example.com/procurement/definitions"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/" >

   <message name="OrderMsg">
      <part name="productName" type="xs:string"/>
      <part name="quantity" type="xs:integer"/>
   </message>

   <portType name="procurementPortType">
      <operation  name="orderGoods">
         <input message = "OrderMsg"/>
      </operation>
   </portType>

   <binding name="ProcurementSoapBinding" type="tns:procurementPortType">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="orderGoods">
         <soap:operation soapAction="http://example.com/orderGoods"/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
      </operation>
   </binding>

   <service name="ProcurementService">
      <port name="ProcurementPort" binding="tns:ProcurementSoapBinding">
         <soap:address location="http://example.com/procurement"/>
      </port>
   </service>

</definitions>
```

abstract part

messages

operation and port type

concrete part

binding

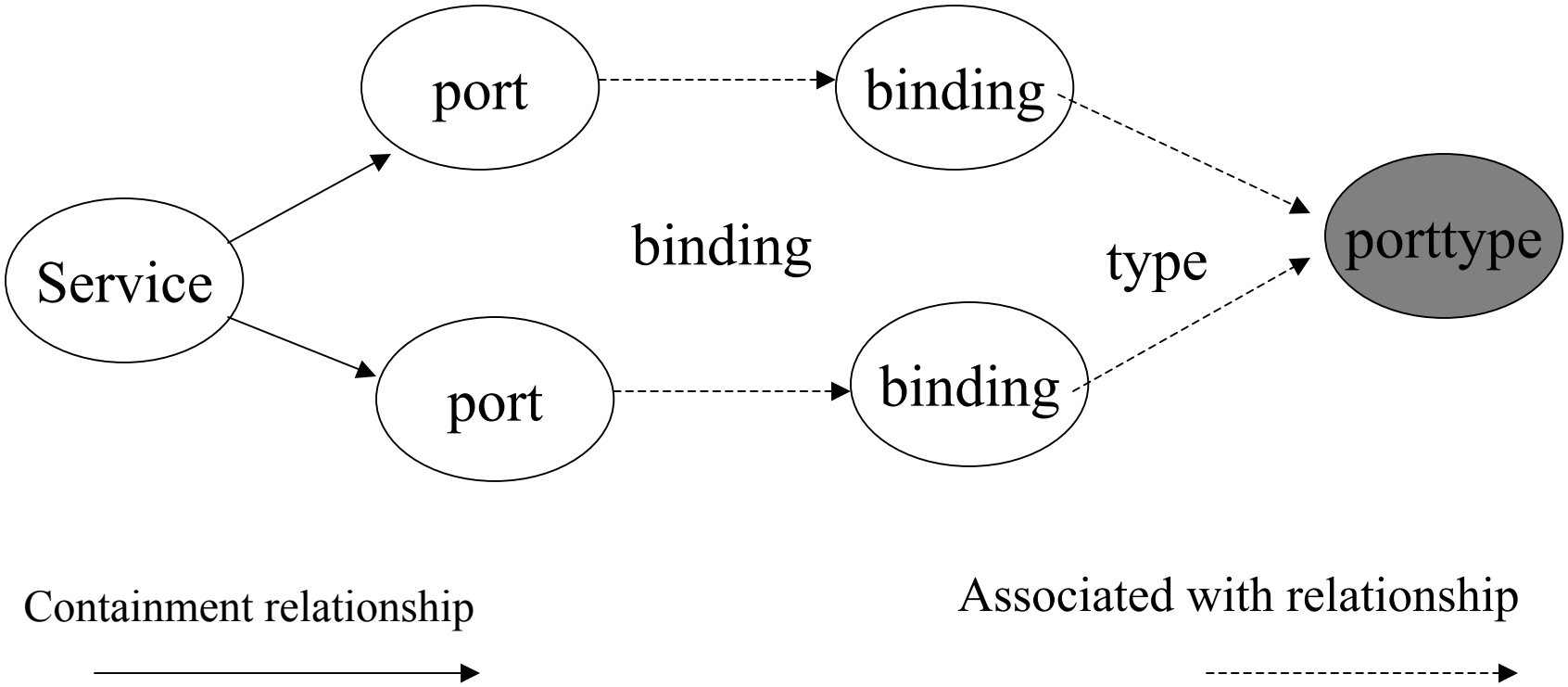
port and service

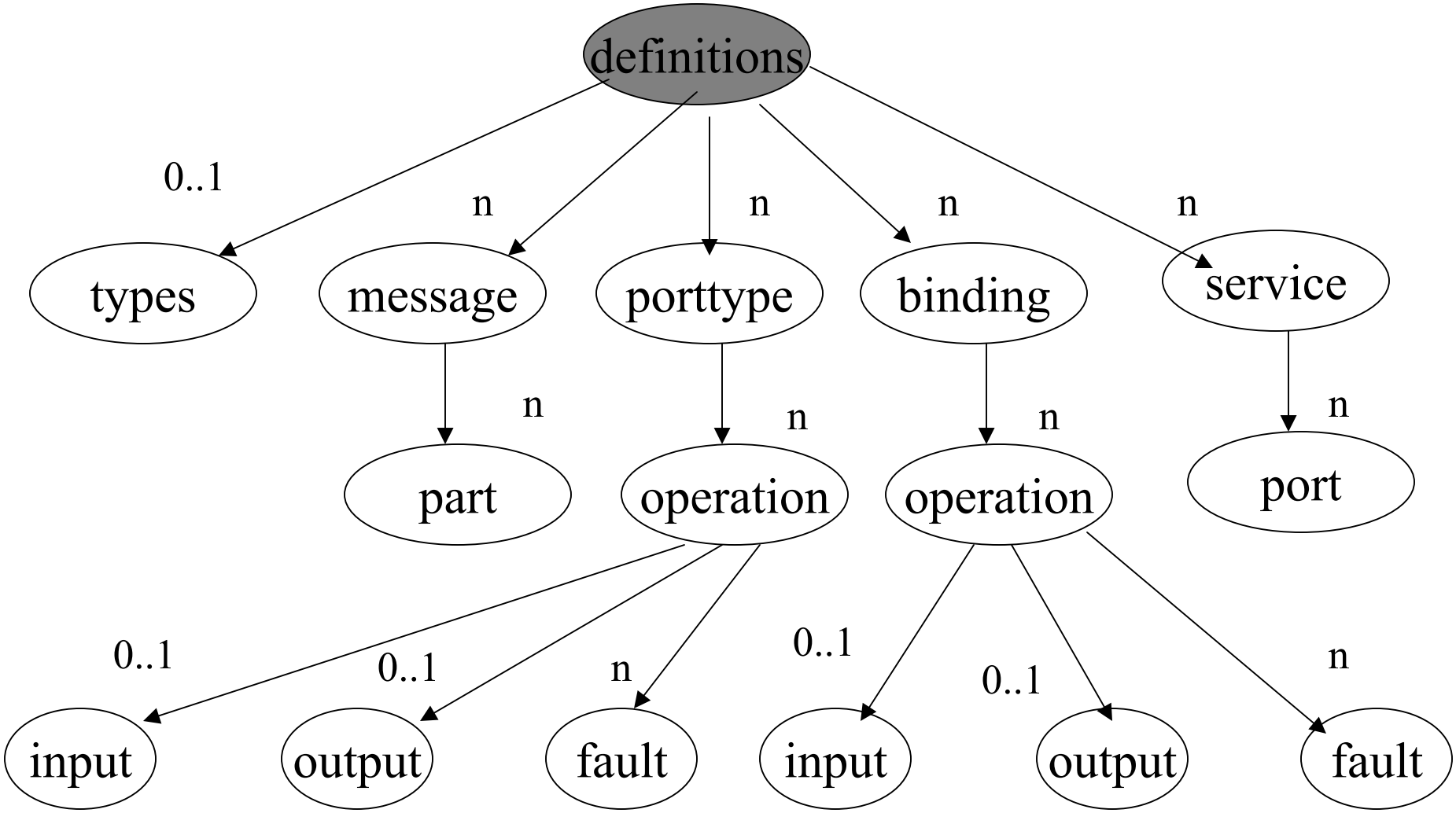
Structure of WSDL

- portType: is a named collection of operations.
- operation: A operation abstractly describes a service call. It may contain an input message, an output message and optionally several fault messages.
- message: A message is an abstract description of the data that is send. Messages consist of logical units called parts.
- part: Each part is associated with a data type
- Types: the types element provides a container for data type definitions.

Structure of WSDL

- binding: A binding provides concrete protocol and data format specifications for a particular port type.
- port: A port describes the network address of a service. In combination with a binding, it provides information about a single concrete service endpoint.
- service: A service is a collection of related ports. These ports may share the same type, but employ different bindings or network addresses. They may also be concrete service endpoint implementations of several port types.


port
binding
Service
binding
type
porttype
port
binding
Containment relationship
Associated with relationship



WSDL document containment structure

Service interface-related element

definitions element: provides the container for all other WSDL element

- Name: the name attribute can be used to provide some hint about the WSDL document content. It is not intended to be a documentation feature, though. Documentation is supported through the documentation element.
- targetNamespace: the target namespace attribute can be used to introduce the namespace to which the element declarations and type definitions contained in the WSDL document should be applied.

Service interface-related element

- targetNamespace: If WSDL definitions are separated into several documents, for example one containing the interface description and another one an associated implementation description, the WSDL import element in conjunction with the target namespace of the interface description imports the abstract service definition into a concrete service implementation document.
- xmlns: it tell the schema processor where to look for element declarations and type definitions used within the WSDL document.

Web Service Description Language

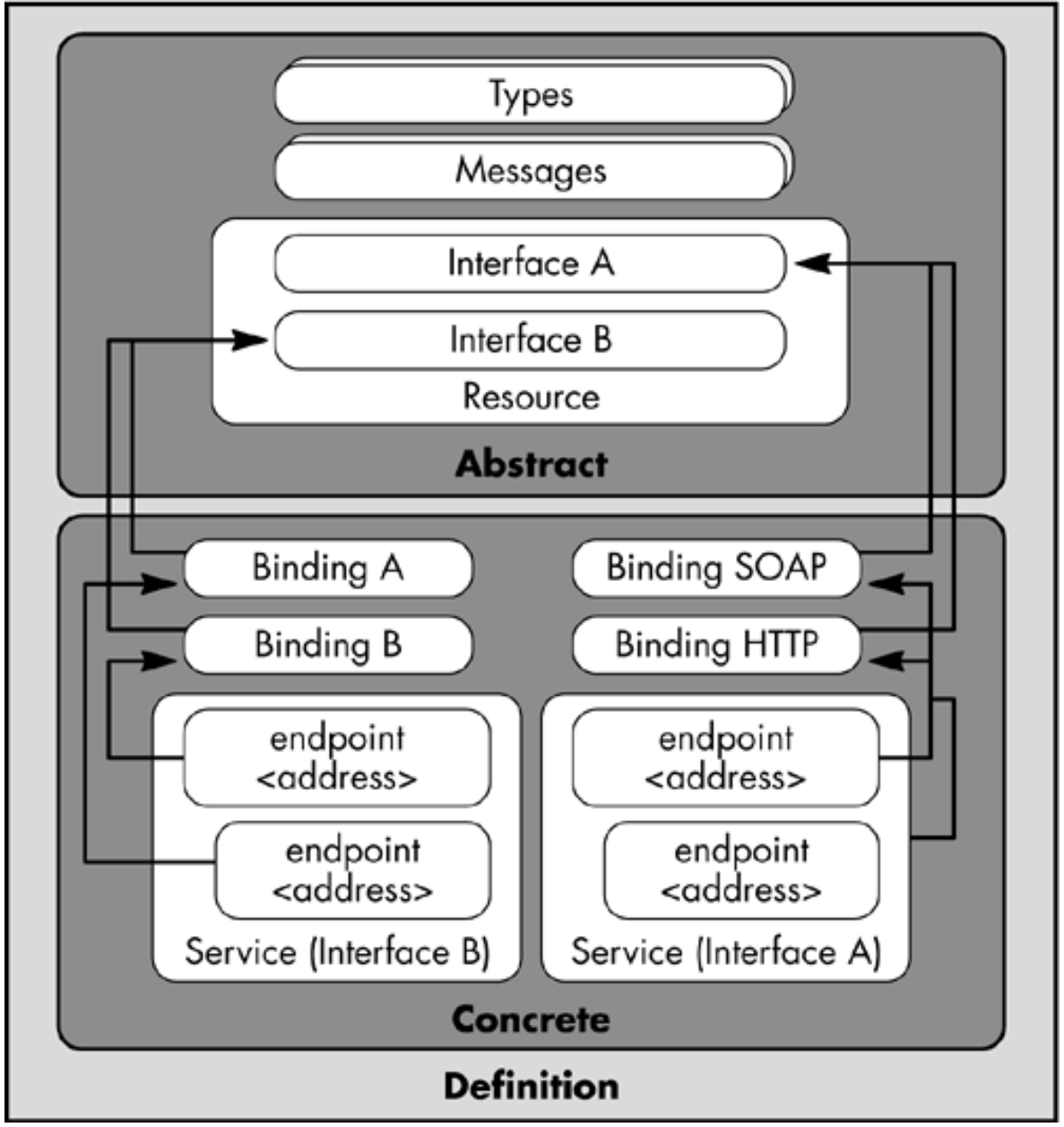
- There has been a number of limitations with this model including service semantic clarifications, extension mechanisms, and the support for new constructs in the SOAP 1.2 specification. WSDL 1.2 is trying to resolve these problems.

Quick Overview of WSDL 1.2

- Some of the major changes are introduced to align the WSDL 1.1 specification with the SOAP 1.2 specification, and the overall service-oriented architecture standards such as Web architecture and Web service architecture.
- One of the major semantic clarifications we can discuss is regarding the concept of the *service*. Let us explore the semantic details of this concept. We will start with the concept of a resource.

Quick Overview of WSDL 1.2

- A resource is anything that can be addressable by a URI. These resources can be software applications, hardware abstractions, or Web resources (e.g., HTML pages or forms).
- Every resource can be logically represented by a set of interfaces, such as manageability interfaces, operational interfaces, and other such interfaces. A service is then treated as a realization of a single interface of a resource.



The WSDL 1.2 abstract and agreed-upon components

Quick Overview of WSDL 1.2

- Note that a service can have any number of end points and bindings.
- But every service is a representation of a single interface. This interface may be a combination of multiple interfaces through derivation.
- This view is different from the WSDL 1.1's view, where a single service can be bound to multiple interfaces through bindings.
- This causes confusion on the semantic of a service.

Quick Overview of WSDL 1.2

- It is a fact that in WSDL 1.1, it is difficult to express service semantics through the description language.
- A binding may select to implement any interfaces of choice. This confusion is later clarified in WSDL 1.2 by having a service as a representation of a single interface of a resource.
- A service may then have multiple end points with different addresses and bindings, however, always bound to the same resource using the same interface.

Quick Overview of WSDL 1.2

- There is another construct introduced to clarify the resource that a service is implementing. This is done using a new construct called "targetResource."
- This "targetResource" is an attribute of a service element, with a URI value representing the resource URI.

A service element definition in WSDL 1.2

```
<definitions>
.....................................
  .....................................
<serviname="xs:NCName"
  interface="xs:QName"
  targetResource="xs:anyURI"? >
          <end-point />*
  </service>
</definitions>
```

Quick Overview of WSDL 1.2

- We can see the WSDL 1.2 description information elements in the following illustration. The concept of "Resource" is logical.
- As shown in the previous figure, the service developers have to define the abstract concepts of a service and attach a service to the target resource and the corresponding interface. The binding selection must then be a runtime activity.

Quick Overview of WSDL 1.2

- In addition to the clarification on the semantic of a service, WSDL 1.2 has introduced the concept of features, properties, and so on.
- These concepts are in line with the SOAP 1.2 counterparts and can be extended to other protocols of choice.
- By defining features and properties in WSDL, a reader (i.e., software agents or humans) can identify the following characteristics:

Quick Overview of WSDL 1.2

1. Some requirements on the interface/operations for its successful execution. For example, some operations need to provide a WS-Security assertion of type XML signature.
2. Some indications on the availability of some feature so that we can rely on them. For example, a service can state that a feature called "privacy policy" is available for verification.
3. General information on the values of the features and its properties.

Quick Overview of WSDL 1.2

- These features may oftentimes appear at the interface level or at the binding level.
- In addition to these extensions for features and modules, there are some major changes that have been introduced, as described below.

Quick Overview of WSDL 1.2

1. Operation overloading is removed from WSDL. Note that WSDL 1.1 supports the operation overloading facilities. Thus, we have to take care when designing WSDL, especially with interface inheritance.

Quick Overview of WSDL 1.2

2. As we have noticed, the "PortType" construct in WSDL 1.1 is now changed to "interface" to align with the existing distributed technologies and IDL.
   This interface supports the inheritance capability. It is worthy to note that this extensibility of interfaces is one of the core concepts in the grid service specification. The next example shows a sample interface hierarchy.

Quick Overview of WSDL 1.2

```
<Definitions .............>
.................................................
        <interface name="A">
                <operation name="add">
                </operation>
        </interface>

        <interface name="B" extends="A">
                <operation name="subtract">
                </operation>
        </interface>
.................................................
</definitions>
```

Quick Overview of WSDL 1.2

- The "ports" construct in WSDL 1.1 has changed to "end points" in WSDL 1.2.
- Another feature that is valuable is the "mechanisms" to include other schema languages to describe the types of the message. Most notable works in this area are inclusion of DTD and RELAX-NG.
- A number of MEPs has been defined to support complex message exchange scenarios, such as request-response, input, and output. WSDL 1.2 defined these MEPs as well-defined features with specific properties.

Implications of the WSDL Model

- Unlike IDLs in conventional middleware, a service can proactively initiate the interaction. In other words, it not only exposes operations that can be invoked, but also operations that invoke, meaning that a service can also behave like a client.
- This leads to a further blurring of the notion of clients and service providers, which is natural when the interaction tends to be more asynchronous in nature.
- In the Web services domain, it is quite common to model every interacting entity as a WSDL service.

Implications of the WSDL Model

- Another implication of the data structure specifications is that WSDL does not presume that exchanges will take place using a particular form of communication.
- WSDL can be used to describe two aspects of the a Web service. The first is the abstract interface of the service, without specifying the location or the protocol to be used. The second is the actual location of the service and the communication protocol to be used.
- The advantage of this separation is that WSDL specifications that describe abstract interfaces are reusable; different services could combine different interfaces using different bindings, and could make them available at different addresses.

Implications of the WSDL Model

- Reuse is also facilitated by the fact that WSDL documents may import other WSDL documents. This allows for modular WSDL specifications, where typically some documents define abstract interfaces, while other documents import those definitions and make them concrete by defining bindings and addresses.
- If the messages defined in WSDL are exchanged using SOAP, then the InterfaceBindings contain all the information needed to infer or automatically construct the SOAP messages.

Implications of the WSDL Model

- In this case, the interaction style, XML encoding, and transport bindings specified in WSDL translate to those used for constructing SOAP messages.
- For example, if the interaction style is RPC, then the input message of an operation in WSDL determines the procedure name and input parameters of the RPC call.
- The ability of both SOAP and WSDL to be able to use multiple transport bindings and XML encodings is a crucial aspect of the standardization efforts around Web services.
- SOAP is a generic envelope to wrap invocations that the applications may make using other tools. WSDL is a generic service description that may correspond to services actually built using other languages, e.g. stored procedures in databases that are automatically converted into Web services.

Implications of the WSDL Model

- In both the cases, the idea is to standardize interactions through an additional tier and therefore facilitate the development of tools that will translate existing interoperability mechanisms into Web services mechanisms.
- Eventually, middleware platforms may directly use SOAP or WSDL, but, at this stage and for probably quite a while, SOAP and WSDL will act as wrappers over existing protocols and description languages. Hence, the need to accommodate multiple communication protocols.

Implications of the WSDL Model

- Following this idea, the W3C working group has documented a number of different services that can be represented using WSDL. Some of the examples include simple service semantics, although the enforcement of these semantics and their actual meaning will necessarily be part of other specifications.
- These examples (which include s/as operations, multiple faults scenarios, versioning and the use of events) show that WSDL needs to evolve much more to adapt to changes in SOAP and in many other specifications that cover aspects such as security, delivery guarantees, transactional interaction, coordination etc.

Using WSDL

- To clarify the use of WSDL, the W3C Web Services Description Working Group has documented and published use cases that illustrate how a web service could be described in WSDL.
- The working group has identified three potential uses of WSDL descriptions:

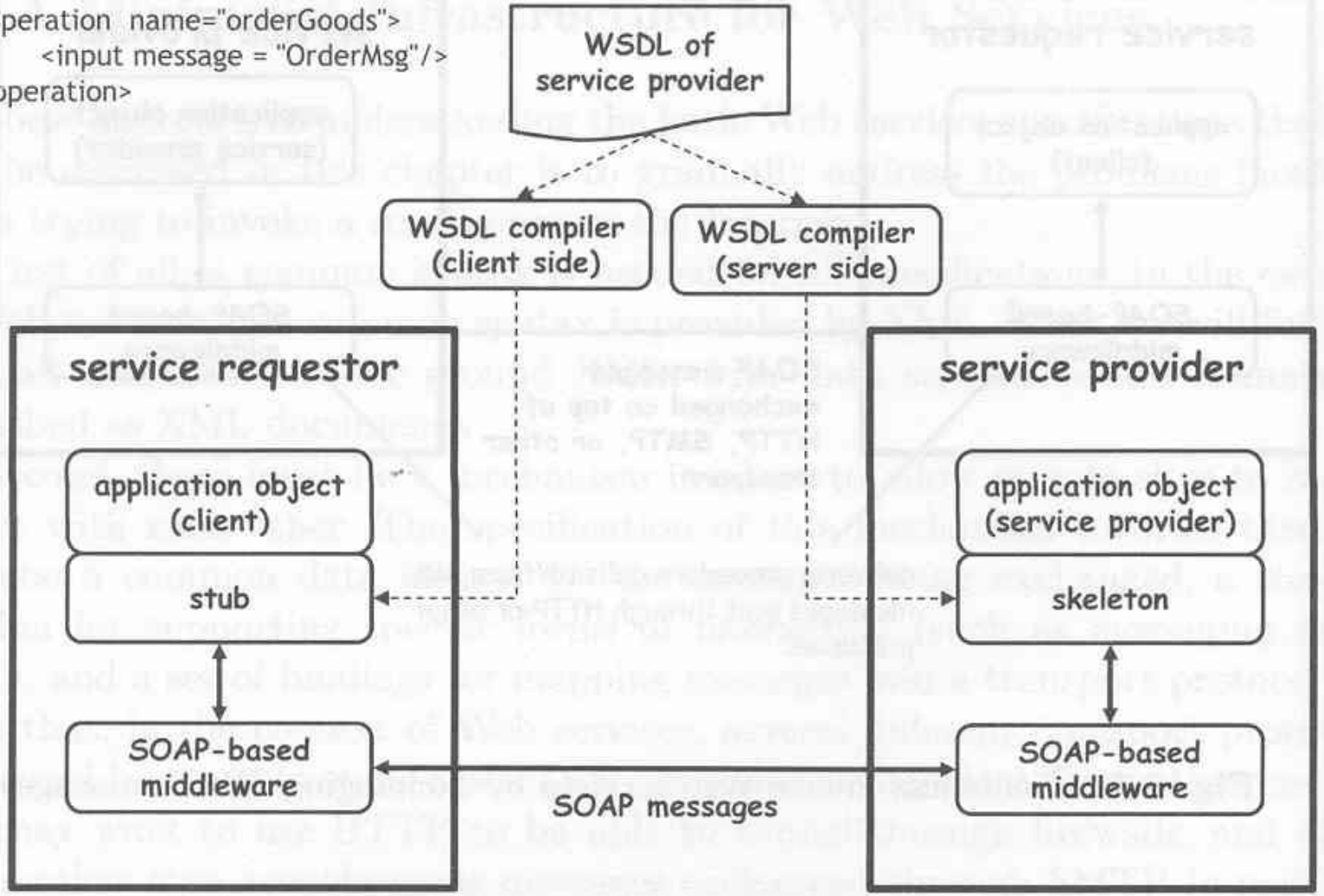
Using WSDL

- 1. The first use is as a traditional service description language. Thus, WSDL description can be seen as a contract that a Web service implements. This contract indicates how to interact with the service, what data needs to be sent, what data is to be expected in return, what operations are involved, and the format and protocol necessary to invoke the service.

Using WSDL

- 2. The second use is as input to stub compilers and other tools that, given a WSDL description, will generate the required stubs and additional information for developing both the service and the clients that invoke the service.
- As shown in the next figure, it describes this usage. As with IDLS in conventional middleware platforms, the use of automated tools for generating stubs facilitates development and ensures that the service is invoked properly.

```
<operation  name="orderGoods">
      <input message = "OrderMsg"/>
</operation>
```



Using WSDL

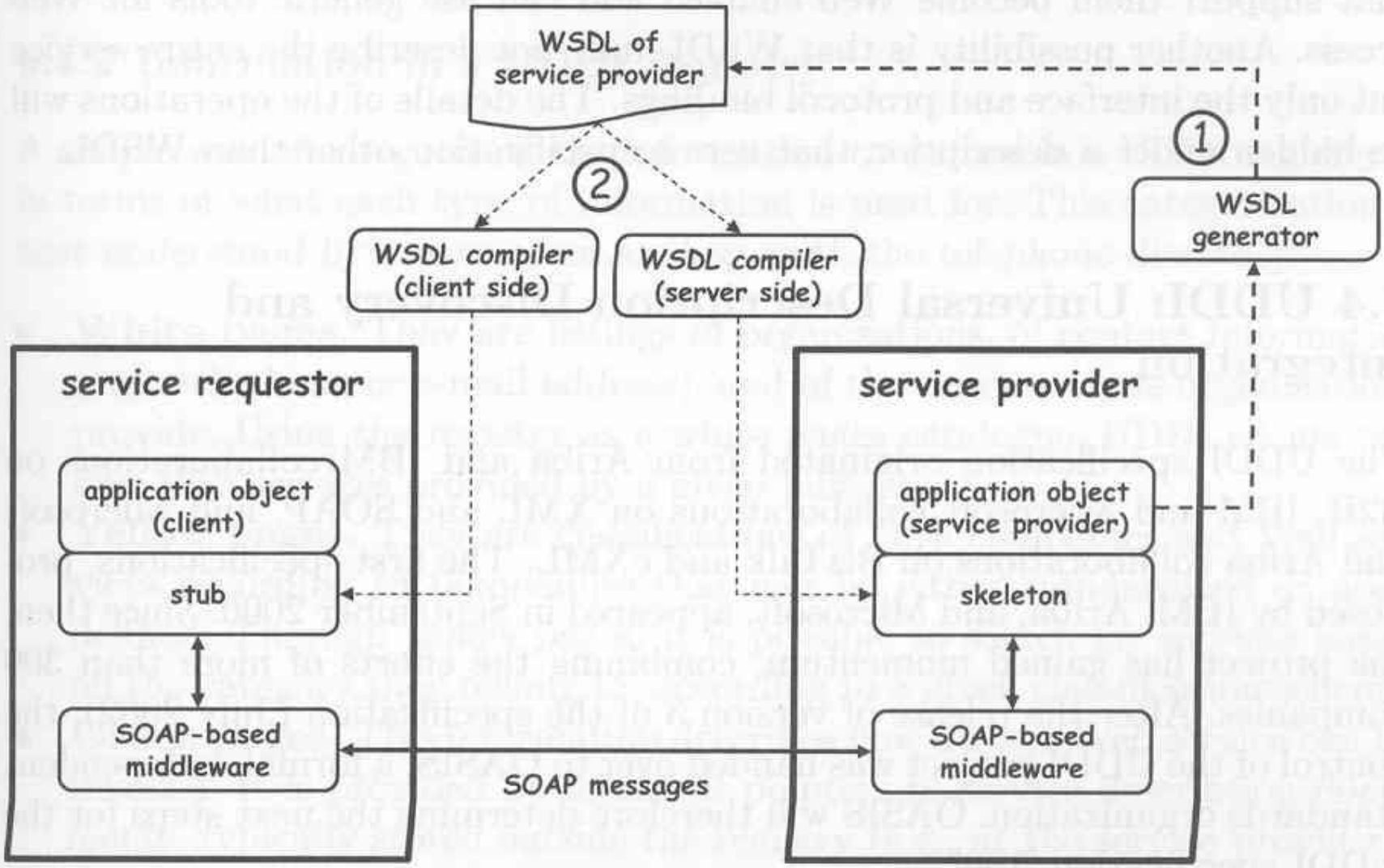
- 3. The third use proposed by the Working Group is rather vague, and it concerns semantics. The proposal is for a WSDL description to capture information that will eventually allow designers to reason about the semantics of a Web service. Nevertheless, it is important to keep in mind that semantics are outside the scope of current WSDL 1.2 specification and it is likely that they will remain outside the specification even in future versions.
- Currently, each Web service must use a separate specification to pin down the actual semantics or use some other mechanism outside the WSDL description to establish the necessary conventions.

Using WSDL

- WSDL should not be understood as being anything more than a form of advanced IDL for Web services, with the differences already mentioned in access mechanisms and groupings of ports. Of the three listed potential uses, the first two are identical to those of IDL languages. The advantage of WSDL over previous solutions is the reliance on XML and the potential for widespread acceptance, with can lead to a level of automation that was not possible before.
- Stub compilers and tools that help in developing interfaces have existed for a long time. However, they are tied to the underlying middleware platform and cannot be made generic. If WSDL is widely adopted, such tools will become pervasive and much more sophisticate than existing ones.

Using WSDL

- In most scenarios, the WSDL itself is automatically generated. Web services are a way of exposing internal operations. Therefore, in many cases, the application implementing the service is pre-existing.
- Based on the API of the application, a WSDL document can be automatically generated. From the WSDL, stubs and skeletons can also be derived.
- Commercial databases have started to provide functionality that automatically generates the WSDL description for a given stored procedure.
- Analogously, application servers provide facilities both for generating stubs from WSDL documents and for generating WSDL documents from classes (e.g. Java classes).



WSDL documents can be generated from APIs. Dashed lines represent compile-time activities. First, WSDL is generated. Next, stubs and skeletons are created.

WSDL and other Standards

- WSDL is a horizontal standard in the sense that it can be used in a wide variety of settings and does not contain any specifics of particular domains. It has been designed as a generic service description language that can be used for many services independently of what the service does. This is clearly not enough in the context of Web services.
- In addition to WSDL, a few competing B2B standards may coexist, e.g. EDI, used in manufacturing and SWIFT, used in the financial world.