Software Architecture

Distributed Architecture

Types of Distributed Architecture Style

- Client-server
- Multitier
- Proxy
- Dispatcher (Load Balancer)
- P2P
- Broker
- Service-oriented architecture
- Microservice architecture

Service-Oriented Architecture (SOA) Style

Eunmi Choi
Kookmin University

브로커는 필요하면 찾아가서 내가 이런 요청이 있는데 찾아주세요!
함
브로커가 찾음

Service-Oriented Architecture (SOA) Style

서버들이 다 연결되어서 새로운 서비스를 창출해냄. (Contribution?이 핵심)

- Synapsis
  - A Service-Oriented Architecture (SOA) starts with a businesses process.

클라이언트와 회사 : C2B 비즈니스,
회사는 다른 서버들을 연결해서 B2B 비즈니스

    - a service is a business functionality that is well-defined, self-contained, independent from other services, and published
    - used via a standard programming interface.

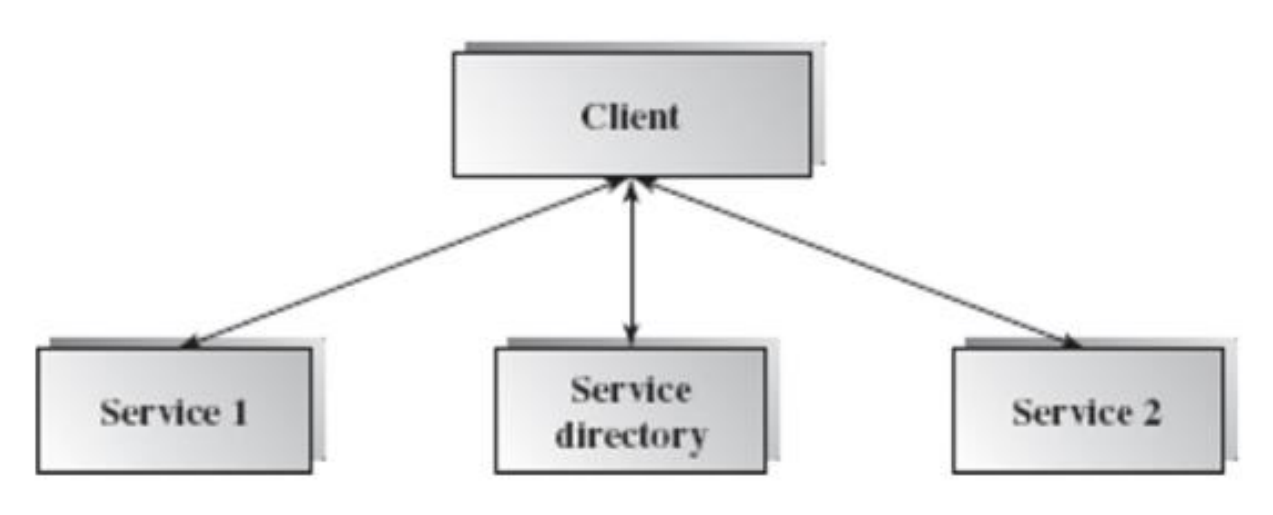
  - Loose coupling of service orientation provides great flexibility for enterprises to make use of all available service resourses regardless of platform and technology restrictions.

~ 7:35

Service-Oriented Architecture (SOA) Style

- A client can find a service via a service directory and then accesses it in a service request-response mode.

클라이언트가 다른 종류의 서비스를 사용



Service-Oriented Architecture (SOA) Style

- Structure
  - A typical service-oriented application makes use of many available services using some flow control language
    - e.g., BPEL for web services
  - Orchestration languages allow for specifying the sequence and logical order of the business executions based on the business logic

서비스가 작으면 의미노노. 헤비한 어플리케이션을 돌리는 경우가 많음.

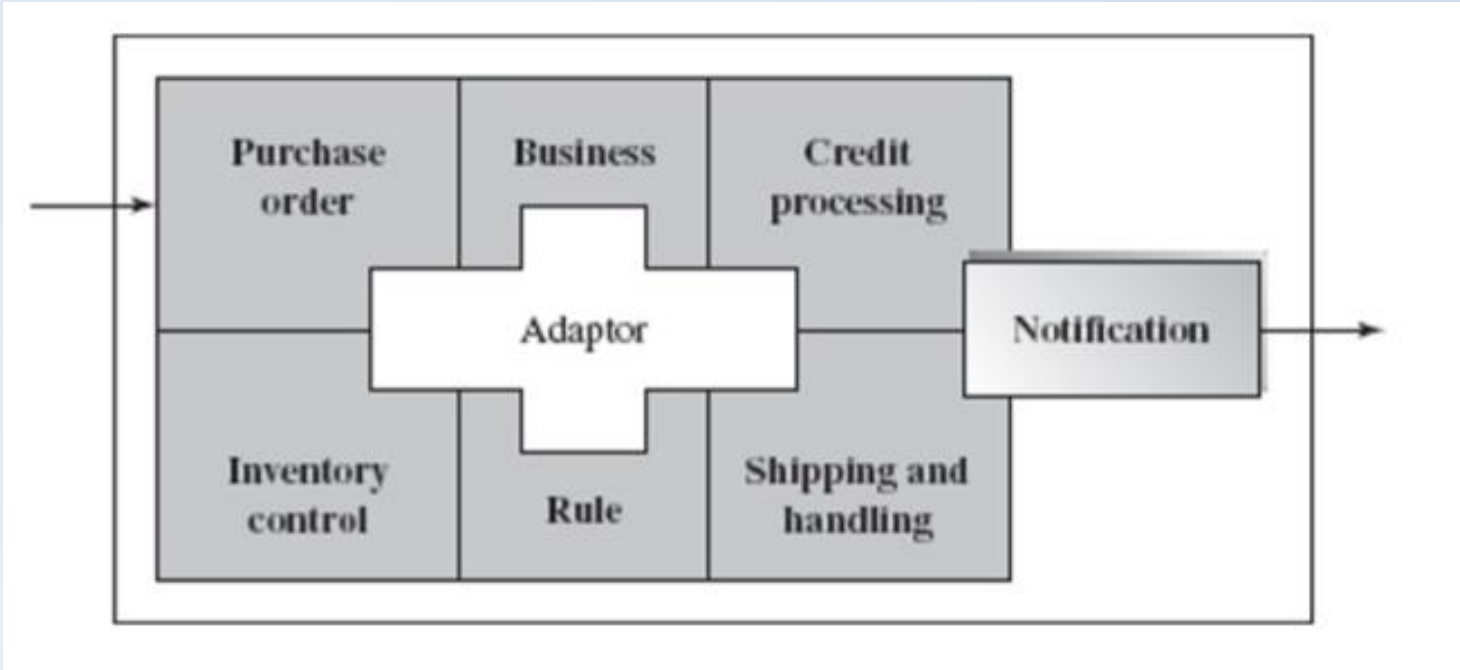
B2B는 work flow가 있을 수 있음.

다른 종류의 instrument 들이 특성을 가지고 전체를 합쳐 컴포지션을 해서 운영하는 형태
이걸 기반으로 비지니스 익스큐션이 일어남.

Service-Oriented Architecture (SOA) Style

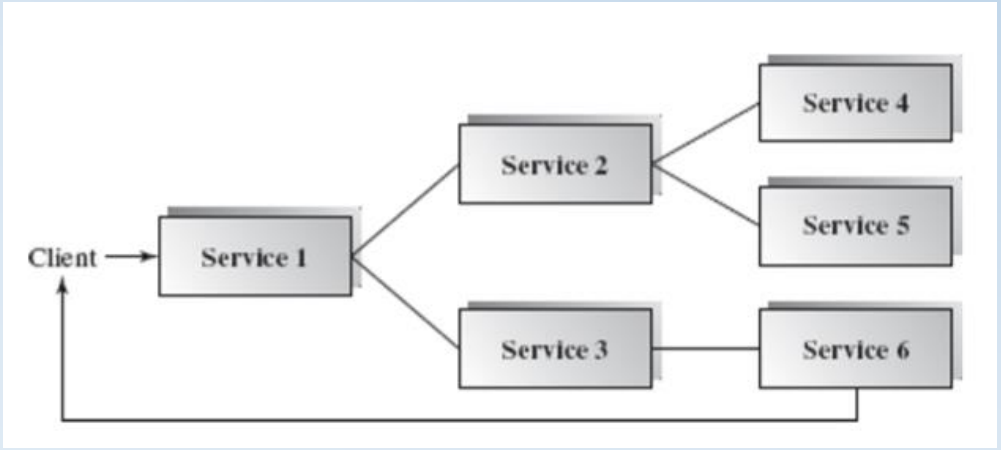
- Service composition

~9:50

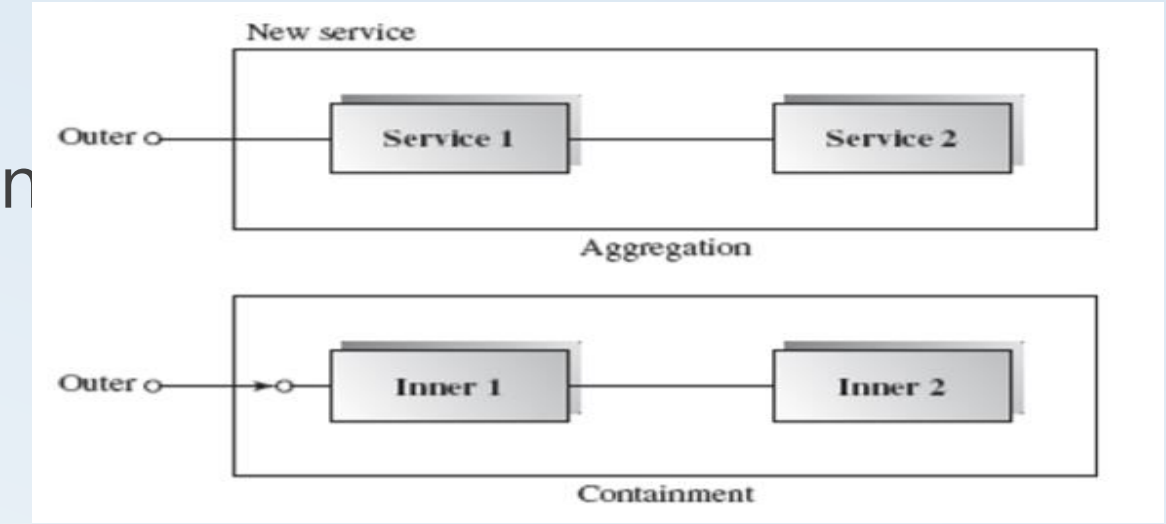


Service-Oriented Architecture (SOA) Style

- Service reuse



- Service composition



SOA: Advantages

- Advantages of SOA:
  - **Loosely-coupled connections** :
    - Loose-coupling is the key attribute of service-oriented architecture.
    - Each service component is independent due to the stateless service feature.
    - The implementation of a service will not affect its application as long as the exposed interface is not changed. This makes SOA software much easier to evolve and update.

서버쪽은 독립적이고 상태를 지속시키지 x. request만 처리하면 됨.

임플리멘테이션은 어플리케이션에 영향을 그렇게 많이 주지 않음..

  - **Interoperability :** 상호 호환적.
    - Technically, any client or service can access other services regardless of their platform, technology, vendors, or language implementations.

다 독립적. 어떤 ~ㄹ를 쓰든간에 스탠다드만 하면 쓸 수 있음.

  - **Reusability :**
    - Any service can be reused by any other service. Because clients of a service need only to know its public interfaces, service composition and integration become much easier. This makes SOA-based business application development much more efficient in terms of time and cost.
  - **Scalability :**
    - Loosely-coupled services are easy to scale. The coarse-grained, document-oriented, and asynchronous service features enhance the scalability attribute.

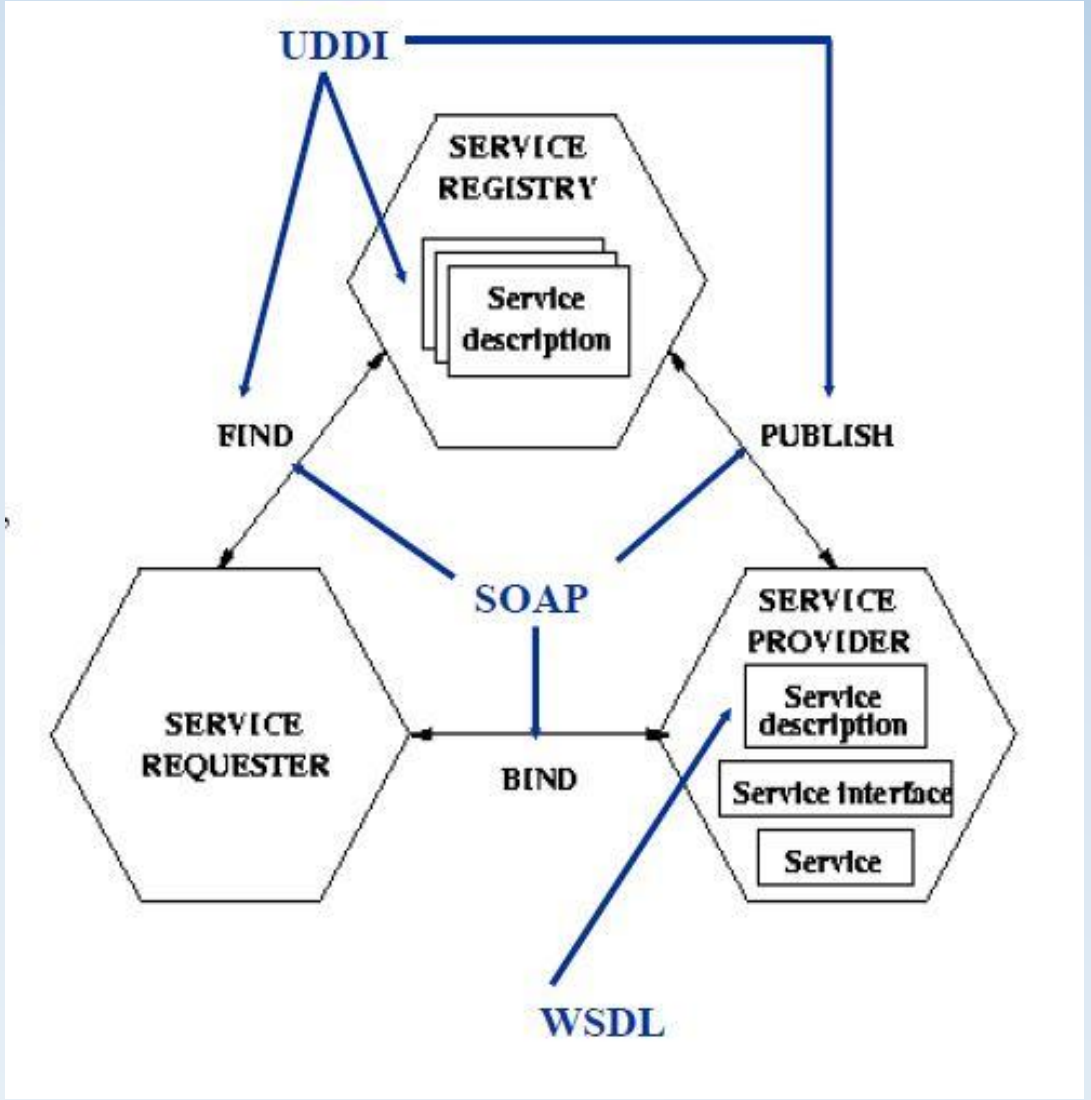
XML-based Web Service

SOA Implementation in Web Services

- A web service is a service that communicates with other services or clients via standard protocols and technologies such as SOAP, XML, and HTTP.
- A web service is a message-oriented service that can deliver document-oriented messages as well as RPC messages.
  - Because an XML-based message is semi-structured, it makes a web service architecture universally accessible and flexible.

Web Service Architecture

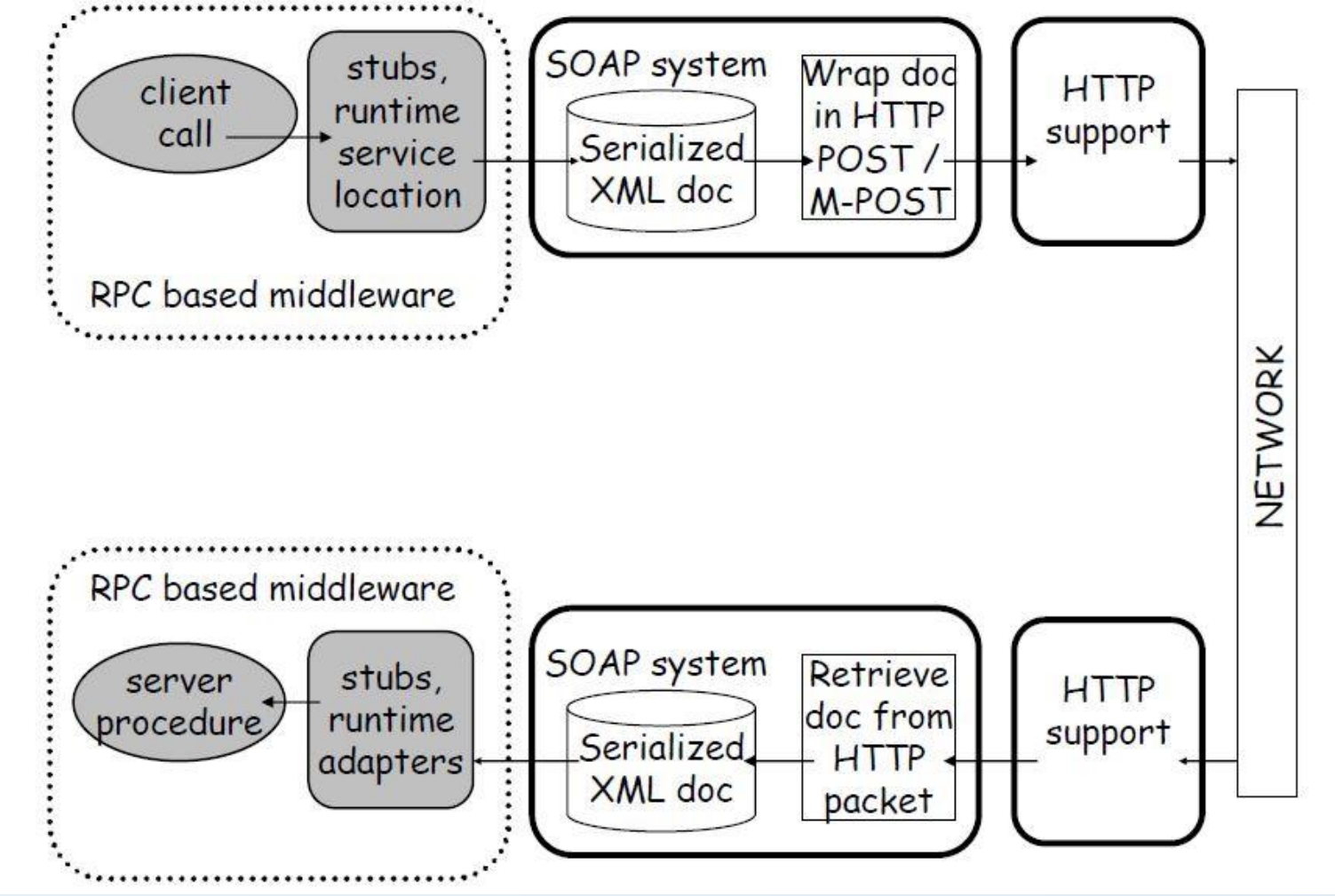
- The architecture has a remarkable client/server flavor based on:
  - SOAP (Simple Object Access Protocol)
  - UDDI (Universal Description and Discovery Protocol)
  - WSDL (Web Services Description Language)



Web Service

- 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.

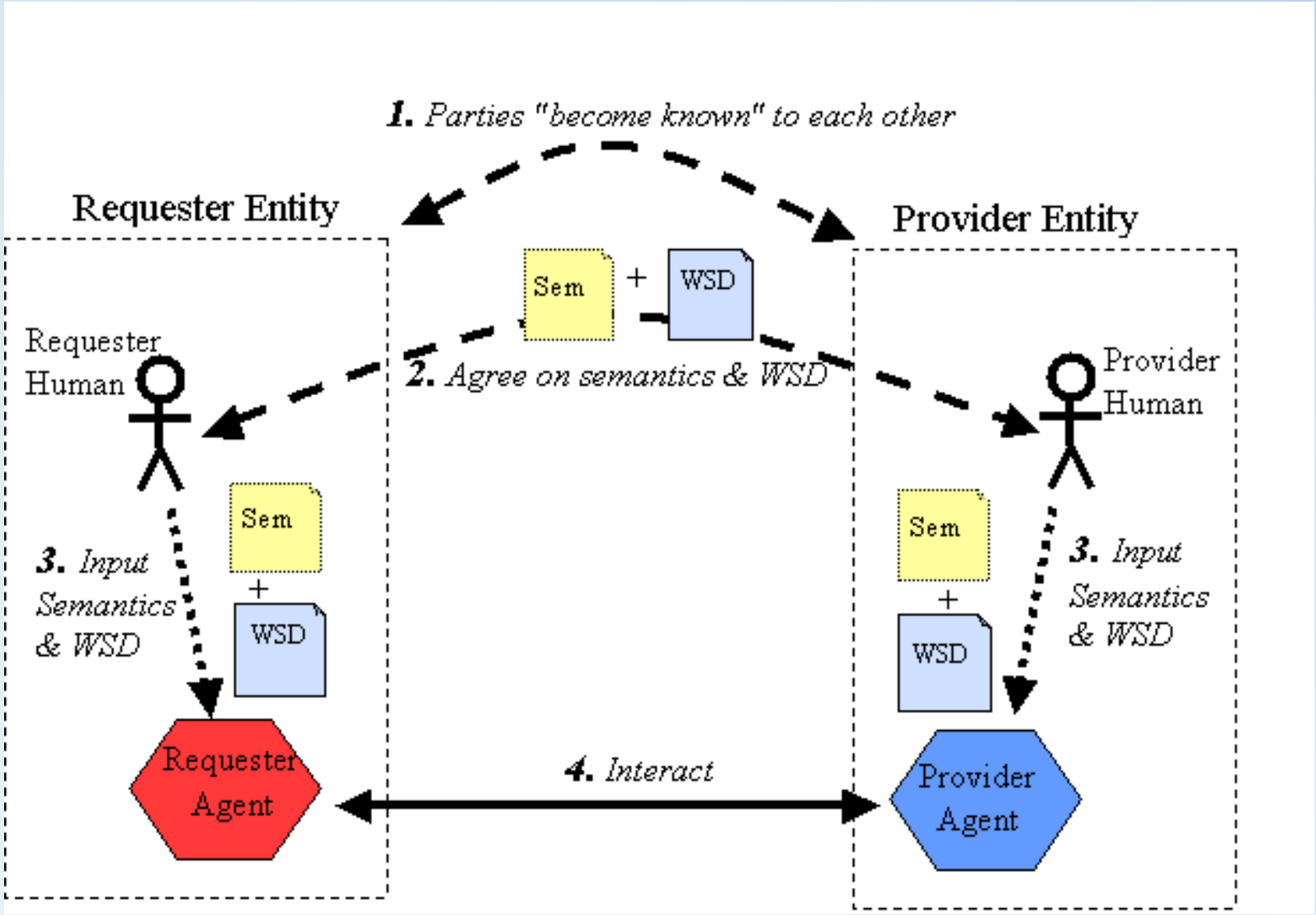
Conversion RPC - SOAP



WS Semantics

- WS semantics: the shared expectation about the behavior of the service.
  - "contract" between the requester entity and the provider entity regarding the purpose and consequences of the interaction.
  - a legal agreement or an informal (non-legal) agreement

The General Process of Engaging a Web Service



SOAP (Cont'd)

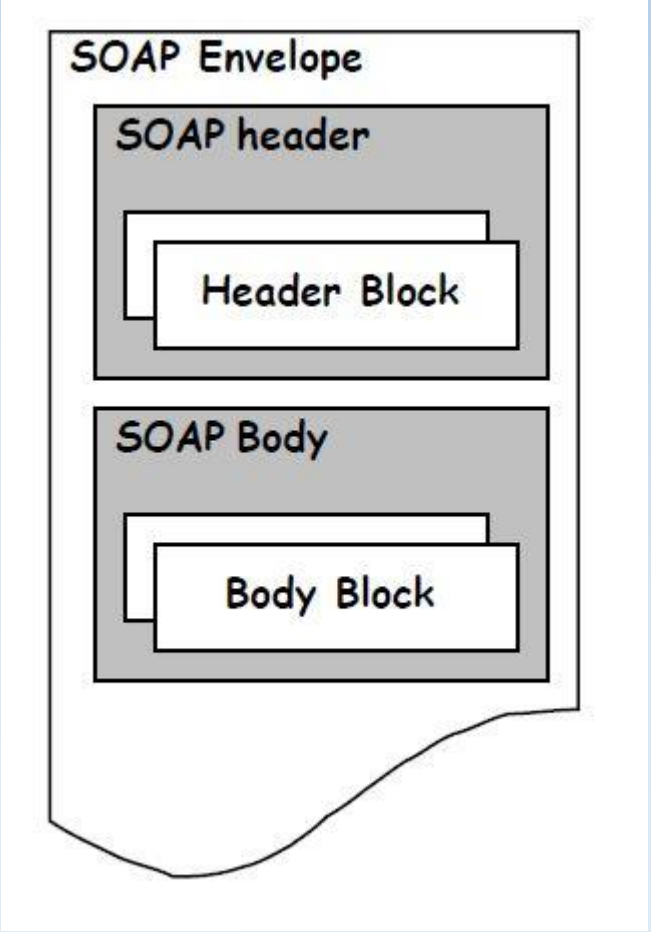
- Two expansions of the term that reflect these different ways in which the technology can be interpreted.
- Service Oriented Architecture Protocol:
  - In the general case, a SOAP message represents the information needed to invoke a service or reflect the results of a service invocation, and contains the information specified in the service interface definition.
- Simple Object Access Protocol:
  - When using the optional SOAP RPC Representation, a SOAP message represents a method invocation on a remote object, and the serialization of in the argument list of that method that must be moved from the local environment to the remote environment.

SOAP message

- The SOAP header may include security, source and destination location, and other elements that can be used by an intermediate service.
- The body contains the main part of the SOAP message; that is, the part intended for the final recipient of the SOAP message.
- A SOAP message is an XML-based document that is independent of any platform and thus can be transported by many protocols, such as HTTP or SMTP.

a SOAP message requesting

```
<SOAP-ENV: Envelope xmlns:SOAP-ENV=
      "http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
       <from>a_client</from>
       <to>a_target</to>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
       <m:GetQuote xmlns:m="myURI">
       <name>My Life and Times</name>
       </m: GetQuote>
 </SOAP-ENV:Body>
</SOAP-Envelope>
```



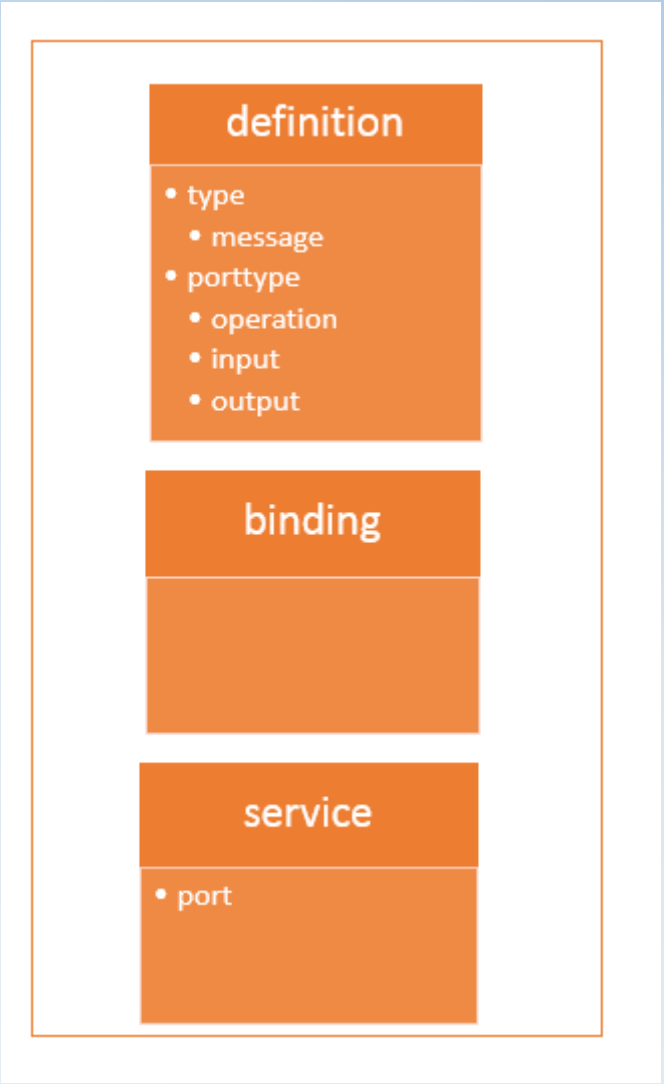
Web Service Description Language (WSDL)

- Web services are self-describing
- A WSDL document is an interface document that describes the interface of a web service.
  - To access a web service, a client has to know the endpoint of a web service directly or indirectly via a web service registry repository where the Web service has registered.
  - The interface information in the WSDL document helps build programmatic calls to the web service.
- a WSDL document for an online stock price search web service, where getStock is an operation of this web service declared in WSDL.
  - <operation name="getStock" …

WSDL Example

```
<!-- WSDL definition structure -->
<definitions
                name="Guru99Service"
        targetNamespace=http://example.org/math/
        xmlns=http://schemas.xmlsoap.org/wsdl/>
        <!-- abstract definitions -->
                <types> ...
                        <message> ...
                        <portType> ...

   <!-- concrete definitions -->
                <binding> ...
                <service> ...
</definition>
```



https://www.guru99.com/wsdl-web-services-description-language.html

WSDL Example

Example Analysis

- **Definitions** – HelloService
- **Type** – Using built-in data types and they are defined in XMLSchema.
- **Message** –
  - sayHelloRequest – firstName parameter
  - sayHelloresponse – greeting return value
- **Port Type** – sayHello operation that consists of a request and a response service.
- **Binding** – Direction to use the SOAP HTTP transport protocol.
- **Service** – Service available at http://www.examples.com/SayHello/
- **Port** – Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

https://www.tutorialspoint.com/wsdl/wsdl_example.htm

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.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://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

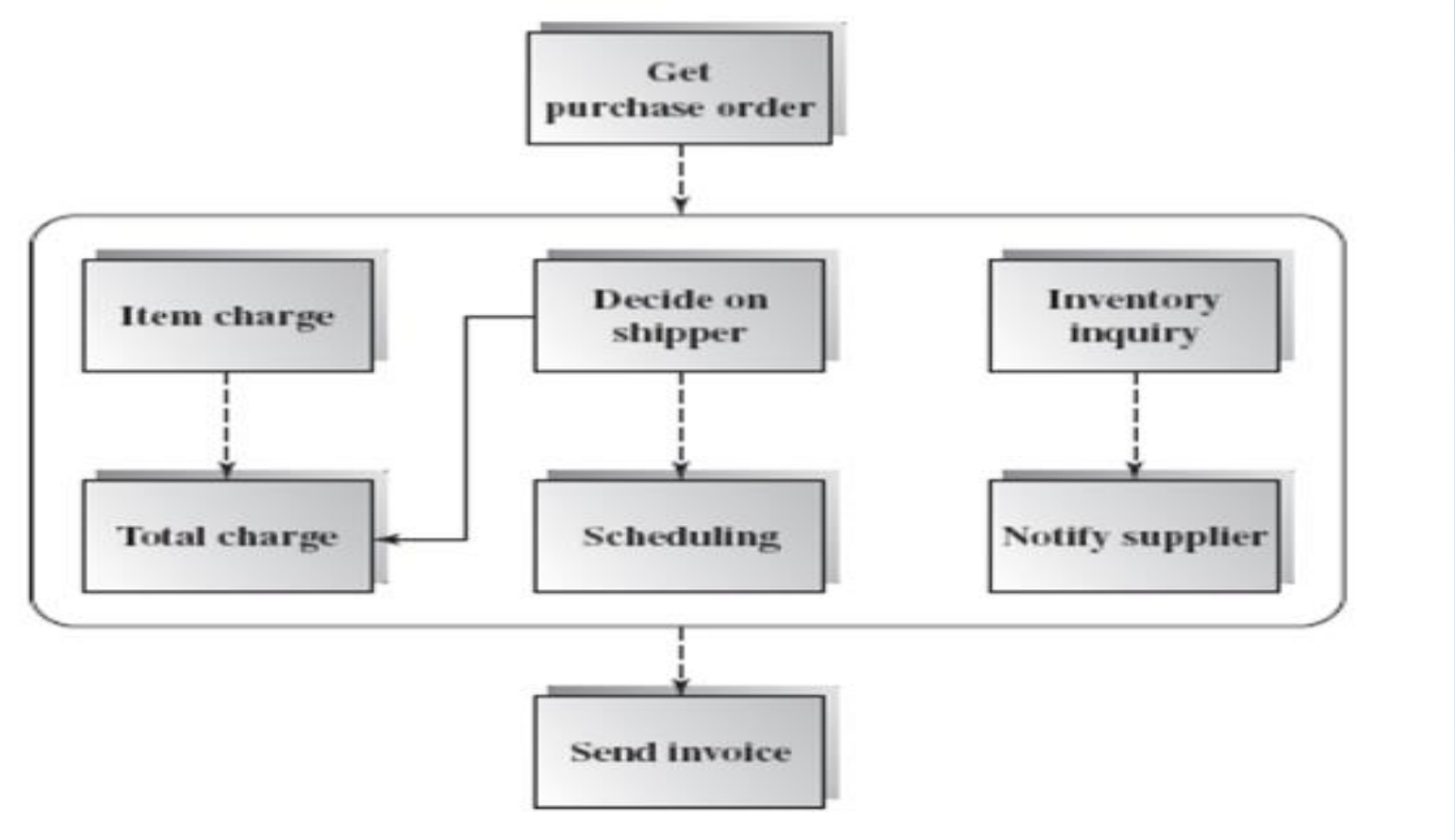
Business Process Execution Language (BPEL)

- An XML-based description language for organizing existing web services to work together.
- BPEL describes the web services that participate in a process, the workflow of these web services, and interactions between them.
- For example, the following BPEL entry describes one of the services in the purchase order process:
  - It receives a purchase order by an operation called purchaseOrder with a purchaseOrder port type which specifies its input and output messages from a participating web service partner link purchase.
  - After the purchase order is received, the BPEL spawns the flow control into two concurrent flows: makeInvoice and scheduleShipping.

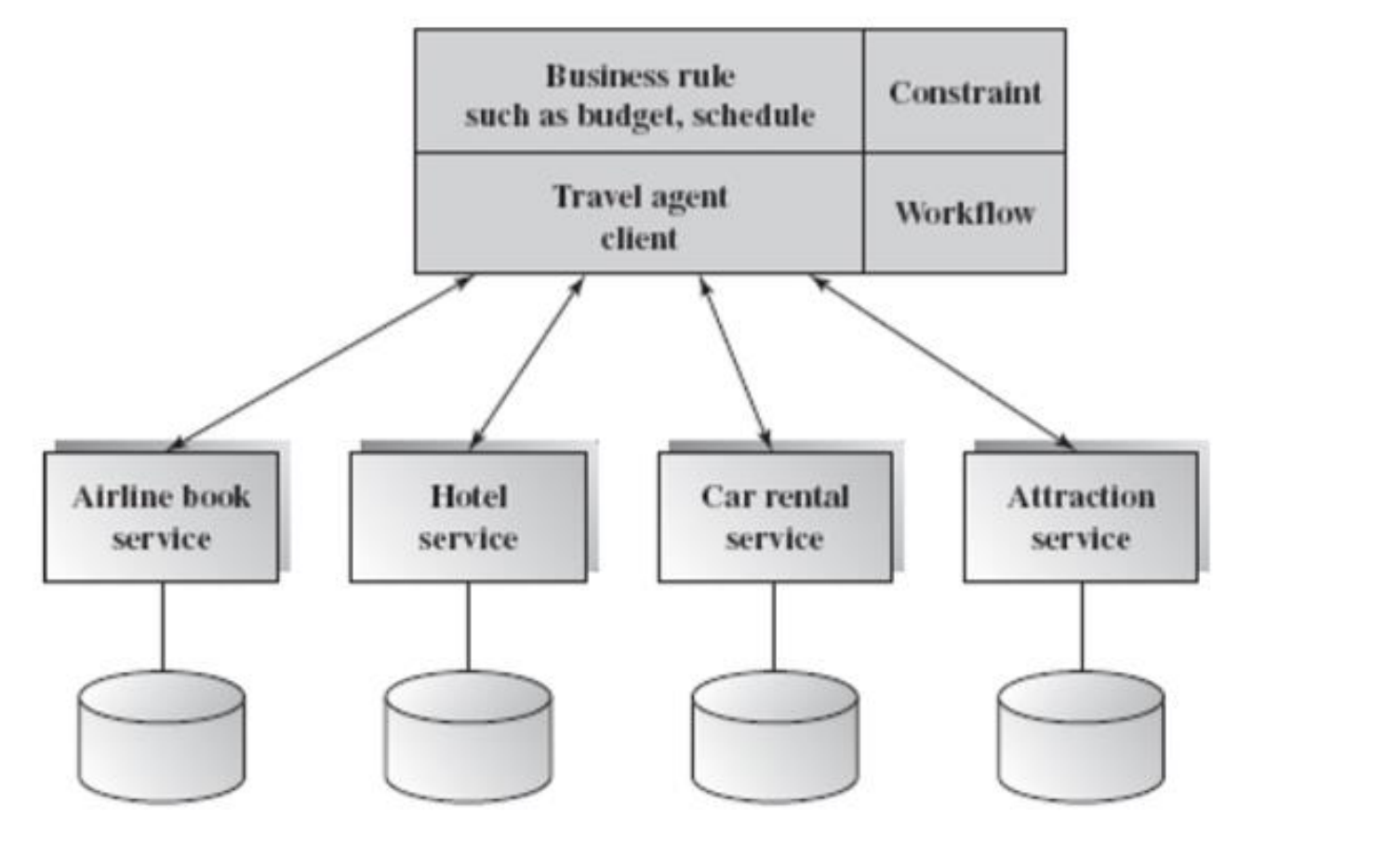
BPEL Example

```
<sequence>
  <receive partnerLink="purchase"
        portType="lns:purchaseOrderPT"
        operation="PurchaseOrder"
        variable="PO">
  </receive>
  <flow>
     <invoke partnerLink="invoiceInk"
          portType="lns:invoicePT"
          operation="makeInvoice"
      />
    <invoke partnerLink="shippingInk"
          portType="lns:shipping"
          operation="scheduleShipping"
      />
 </flow>
</sequence>
```

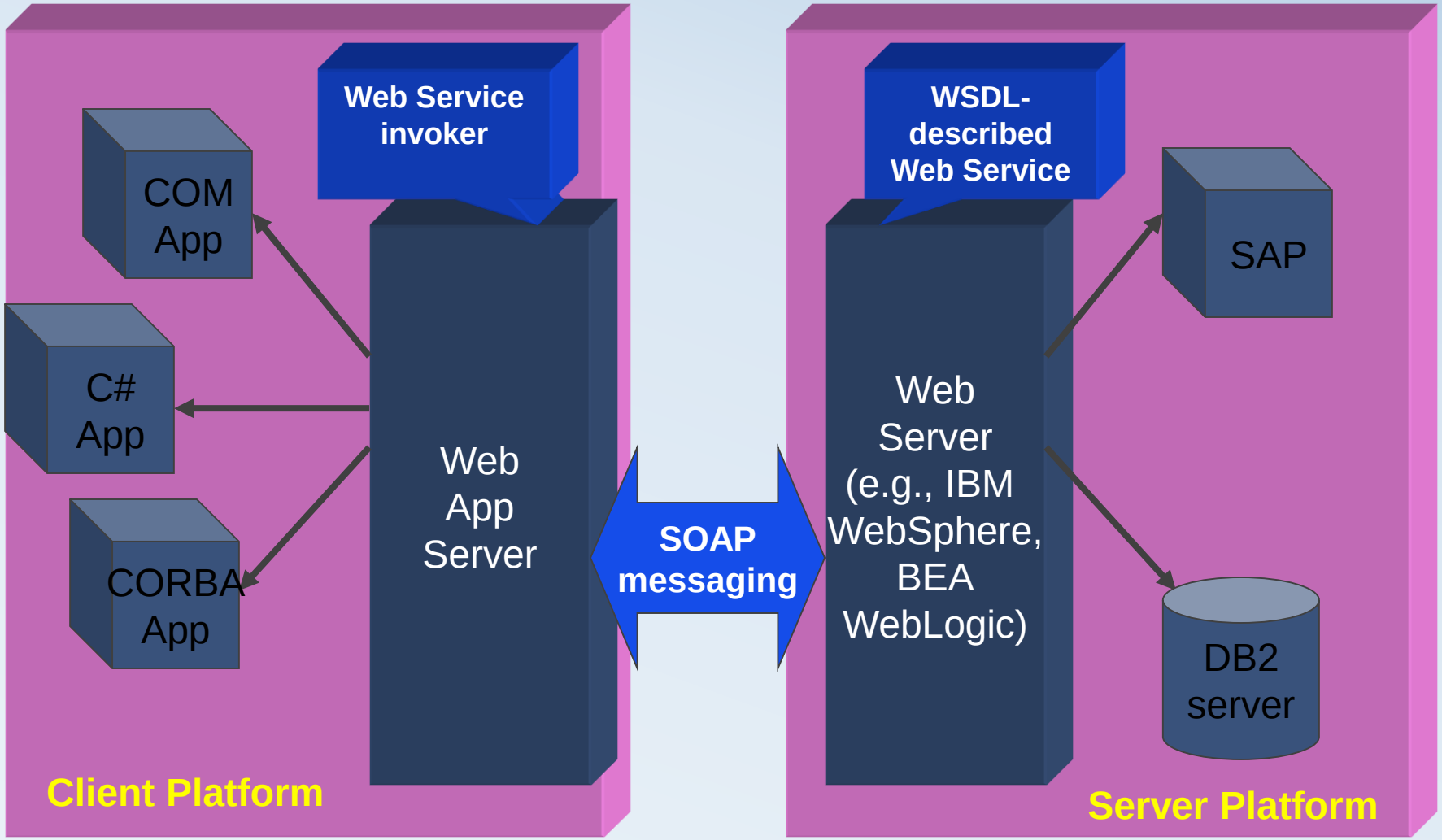
Web service compositions by BPEL



Web service business process model



Web Services are often Front Ends



Summary of Distributed Architecture

- Multi-tier architecture distributes and separates data and processing duties over different tiers so that each tier has its own responsibilities.
  - It reduces message traffic on the network and increases system reliability.
- The client-server architecture is widely used in current enterprise business and industry. Web server, data server, and application server are all examples of server tiers.
- A broker, in the broker architecture, has the responsibility of brokering messages between remote components or other brokers so that a complex enterprise system may involve into multiple brokers, clients, and servers.
- The CORBA, a primordial broker architecture implementation, is also a component-oriented architecture.
- Service-oriented architectures are widely used in B2B enterprise business applications.
  - Each service is a building-block component, such as web or grid service, which can be reused by other service components or other business process applications.