Web Services: Principles & Technology

Chapter 4
SOAP: Simple Object Access Protocol

Mike P. Papazoglou
mikep@uvt.nl

Universiteit van Tilburg

Topics

- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

Inter-Application Communication

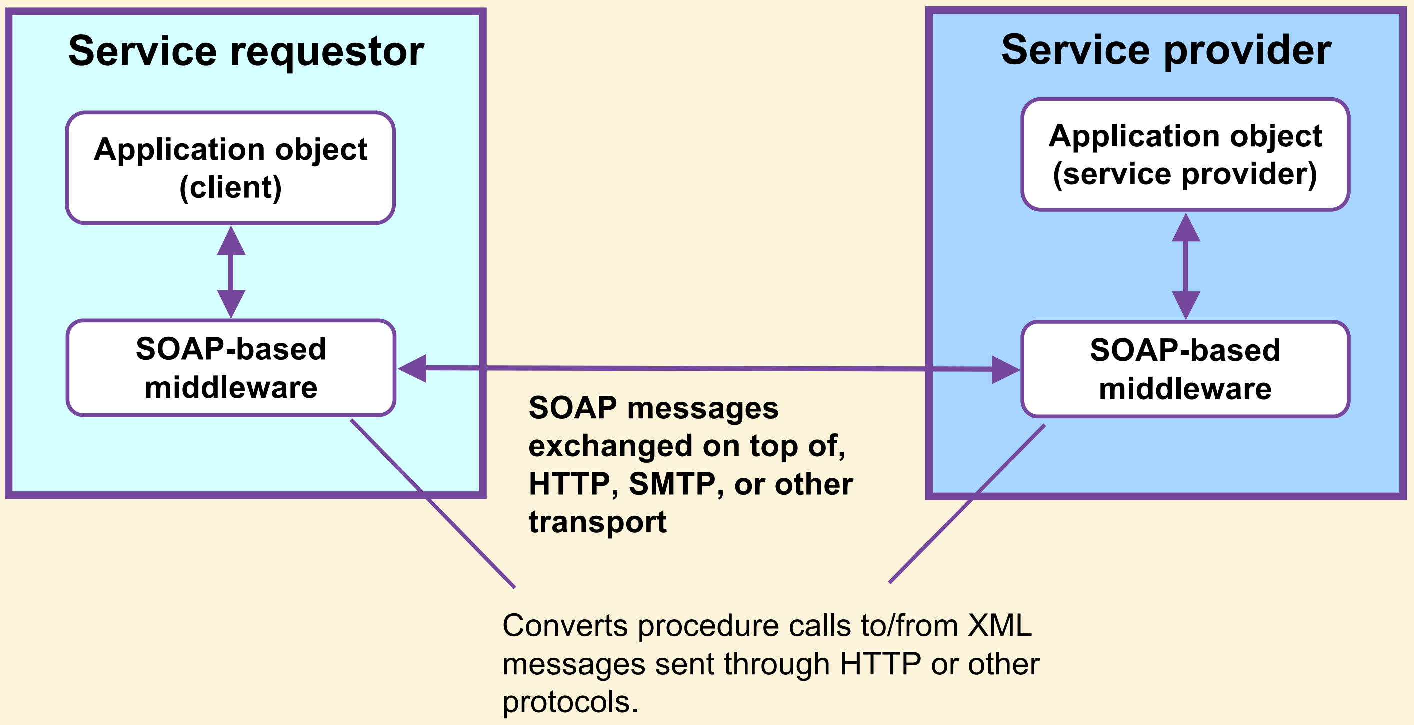
- Conventional distributed applications use distributed communication technologies, e.g., CORBA, DCOM, Java/RMI, based on object RPC (ORPC) protocols that attempted to marry object orientation and network protocols.
  – ORPC request is an identifier or symbolic name that the server could use to locate the target object inside the server process.
- Weaknesses
  – Both ends of the communication link would need to be implemented under the same distributed object model (Java/RMI or CORBA/IIOP)
  – Difficulty of getting these protocols to work over firewalls or proxy servers, e.g, most firewalls are configured to allow hypertext transfer protocol (HTTP) to pass across, but not IIOP.
- To address the problem of overcoming proprietary systems running on heterogeneous infrastructures, Web services rely on SOAP, an XML-based communication protocol for exchanging messages between computers regardless of their operating systems, programming environment or object model framework.

Topics

- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

What is SOAP?

- SOAP is the standard messaging protocol used by Web services. SOAP's primary application is inter application communication. SOAP codifies the use of XML as an encoding scheme for request and response parameters using HTTP as a means for transport.

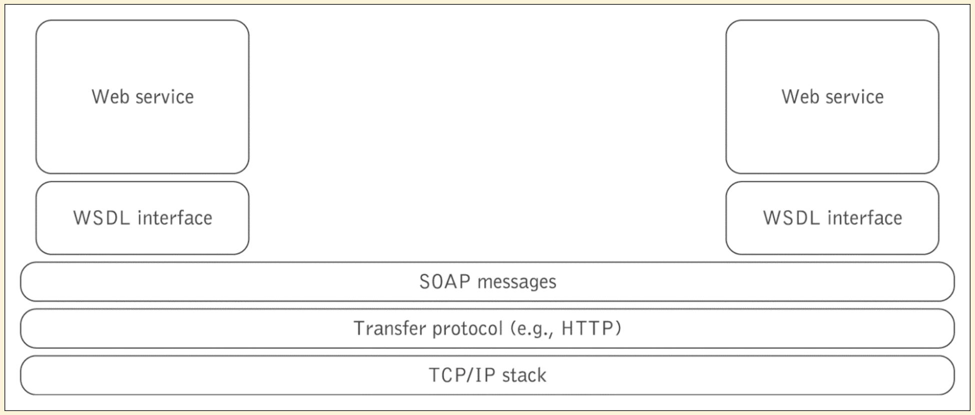

Service requestor
Application object (client)
SOAP-based middleware
Service provider
Application object (service provider)
SOAP-based middleware
SOAP messages exchanged on top of, HTTP, SMTP, or other transport
Converts procedure calls to/from XML messages sent through HTTP or other protocols.

What is SOAP? (continued)

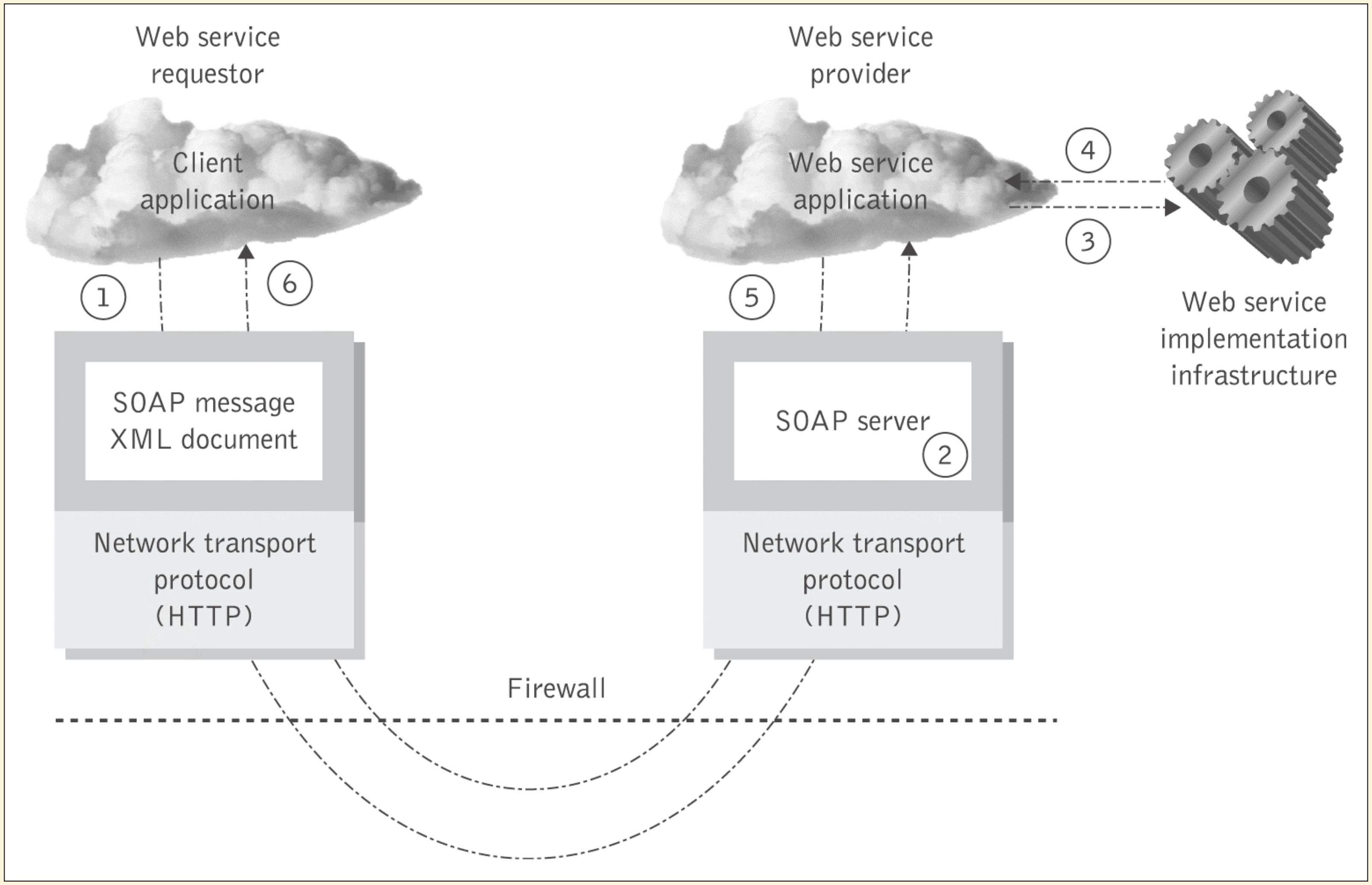
- SOAP covers the following four main areas:
  - **A message format** for one-way communication describing how a message can be packed into an XML document.
  - A **description** of how a SOAP message should be transported using HTTP (for Web-based interaction) or SMTP (for e-mail-based interaction).
  - A **set of rules** that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message.
  - A **set of conventions** on how to turn an RPC call into a SOAP message and back.

SOAP as a lightweight protocol

- SOAP is a lightweight protocol that allows applications to pass messages and data back and forth between disparate systems in a distributed environment enabling remote method invocation.
- By lightweight we mean that the SOAP protocol possesses only two fundamental properties. It can:
  - send and receive HTTP (or other) transport protocol packets, and
  - process XML messages.
- This can be contrasted with the heavyweight protocols such as ORPC protocols.



Distributed messaging using SOAP

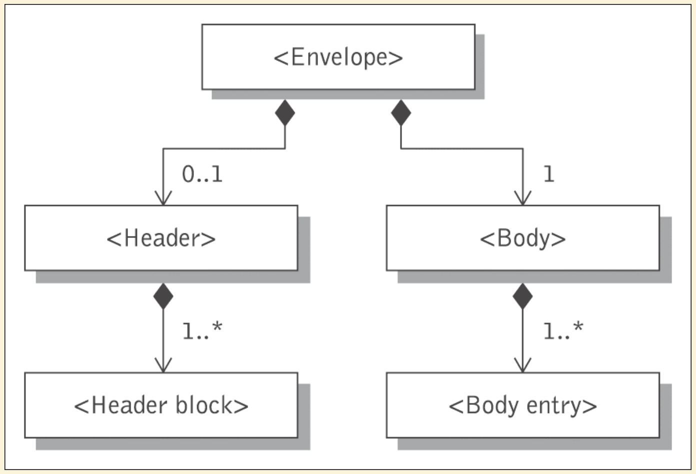

Web service requestor
Client application
Web service provider
Web service application
4
3
Web service implementation infrastructure
1
6
5
SOAP message XML document
SOAP server
2
Network transport protocol (HTTP)
Network transport protocol (HTTP)
Firewall

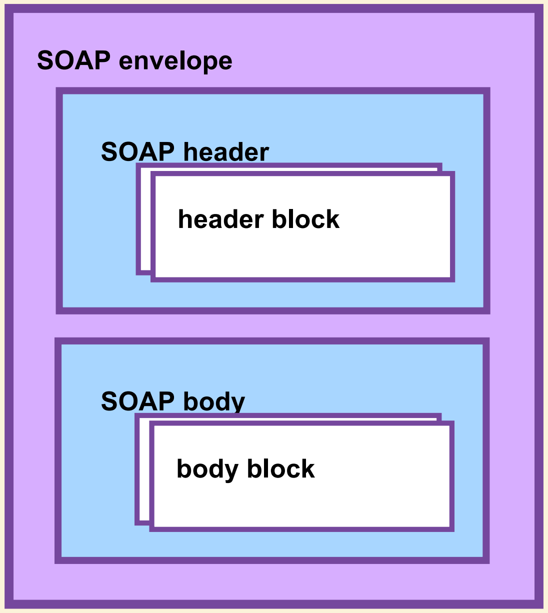
Topics

- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

SOAP messages

- SOAP is based on **message exchanges.**
- Messages are seen as **envelopes** where the application encloses the data to be sent.
- A SOAP message consists of an <Envelope> element containing an optional <Header> and a mandatory <Body> element.
- The contents of these elements are application defined and not a part of the SOAP specification.
- A SOAP <Header> contains blocks of information relevant to how the message is to be processed. This helps pass information in SOAP messages that is not for the application but for the SOAP engine
- The SOAP <Body> is where the main end-to-end information conveyed in a SOAP message must be carried.





SOAP envelope and header

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         ……
</env:Envelope>
```

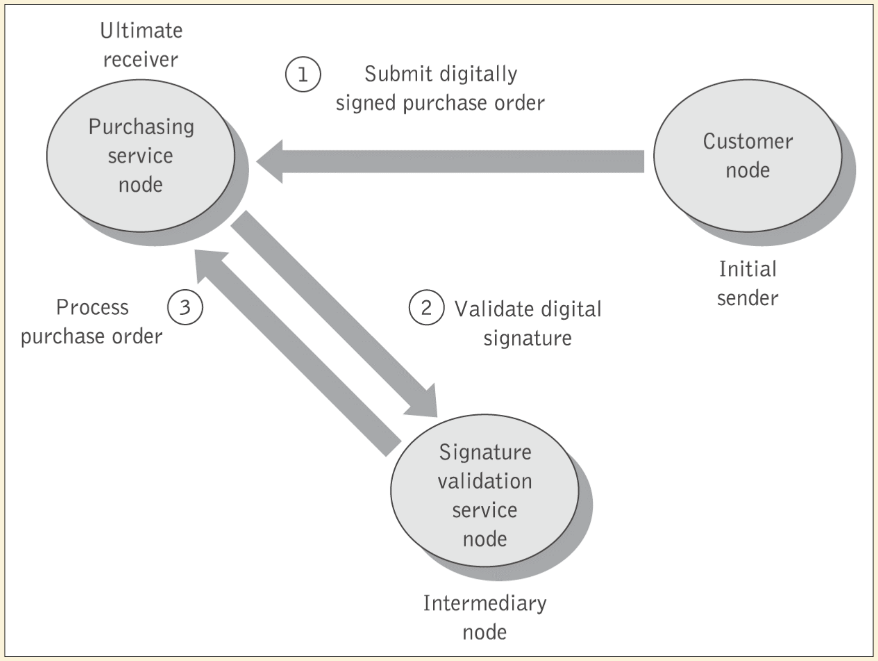
Example of SOAP envelope

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
      …
    <env:Header>
       <tx:transaction-id
           xmlns:tx="http://www.transaction.com/transaction"
                        env:mustUnderstand="true">
                512
            </tx:transaction-id>
       <notary:token xmlns:notary="http://www.notarization-services.com/token"
              env:mustUnderstand="true">
                 GRAAL-5YF3
       </notary:token>
    </env:Header>
             ……………
</env:Envelope>
```

Example of SOAP header

SOAP Intermediaries

- SOAP headers have been designed in anticipation of participation of other SOAP processing nodes – called **SOAP *intermediaries*** – along a **message's path** from an initial SOAP sender to an ultimate SOAP receiver.
- A SOAP message travels along the message path from a sender to a receiver.
- All SOAP messages start with an initial sender, which creates the SOAP message, and end with an ultimate receiver.



Example of SOAP header with message routing

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
   xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:order
     xmlns:m="http://www.plastics_supply.com/purchase-order"
      env:role="http://www.w3.org/2003/05/soap-
envelope/role/next"
           env:mustUnderstand="true">
    <m:order-no >uuid:0411a2daa</m:order-no>
    <m:date>2004-11-8</m:date>
  </m:order>
  <n:customer xmlns:n="http://www.supply.com/customers"
      env:role="http://www.w3.org/2003/05/soap-
envelope/role/next"
           env:mustUnderstand="true">
      <n:name> Marvin Sanders </n:name>
  </n:customer >
 </env:Header>
 <env:Body>
     <-- Payload element goes here -->
 </env:Body>
</env:Envelope>
```

The “next” role is one that all SOAP nodes are required to support;

By tagging each header in this message with the “next” role, this message is saying that all SOAP nodes (intermediaries and the ultimate receiver) MUST understand and be able to process the m:order header and the n:customer header.

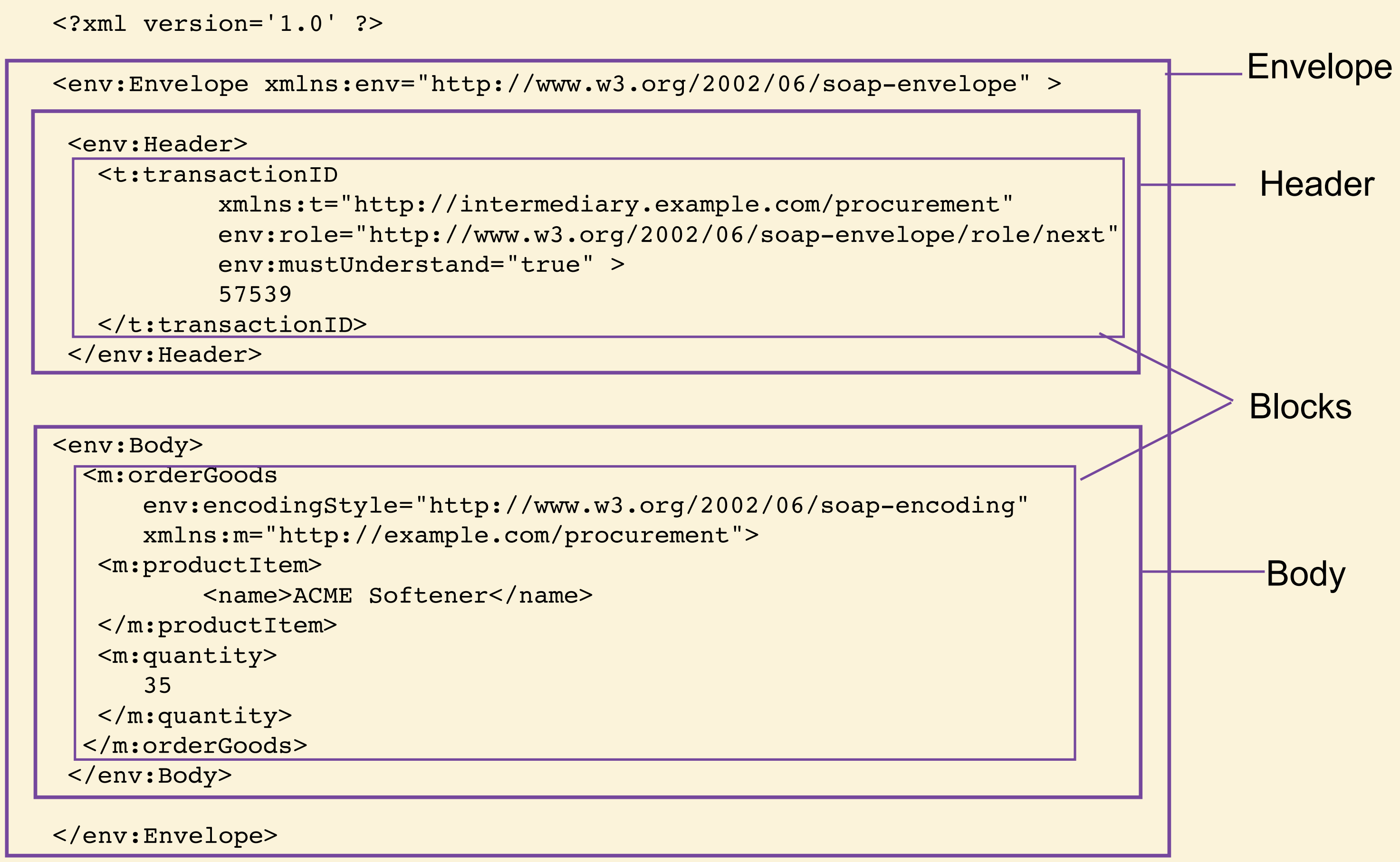
The SOAP Body

- The SOAP body is the area of the SOAP message, where the application specific XML data (payload) being exchanged in the message is placed.

- The **<Body> element** must be present and is an immediate child of the envelope. It may contain a number of child elements, called body entries, but it may also be empty. The **<Body> element** contains either of the following:
  - **Application-specific data:** is the information that is exchanged with a Web service. The SOAP <Body> is where the method call information and its related arguments are encoded. It is where the response to a method call is placed, and where error information can be stored.
  - **fault message:** is used only when an error occurs.

- A SOAP message may carry either application-specific data or a fault, but not both.

Example SOAP Message



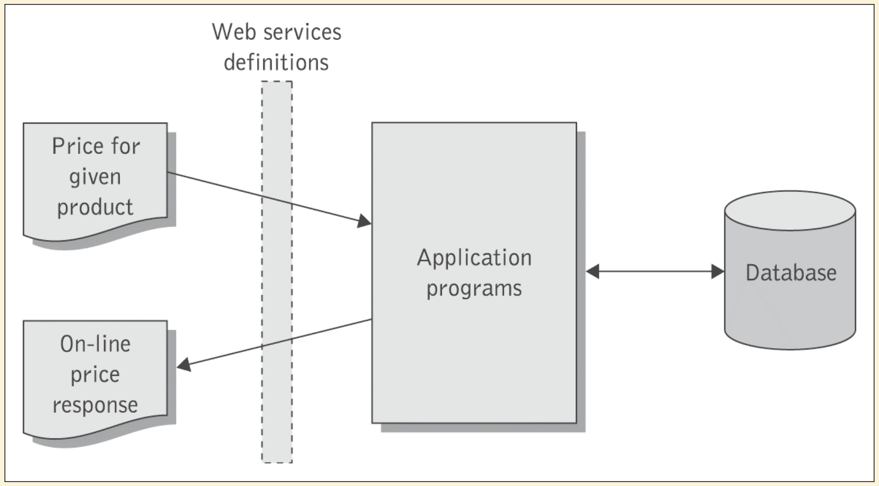
```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
 <env:Header>
   <t:transactionID
           xmlns:t="http://intermediary.example.com/procurement"
           env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
           env:mustUnderstand="true" >
           57539
   </t:transactionID>
 </env:Header>
<env:Body>
  <m:orderGoods
     env:encodingStyle="http://www.w3.org/2002/06/soap-encoding"
     xmlns:m="http://example.com/procurement">
   <m:productItem>
          <name>ACME Softener</name>
   </m:productItem>
   <m:quantity>
     35
   </m:quantity>
  </m:orderGoods>
 </env:Body>
</env:Envelope>
```

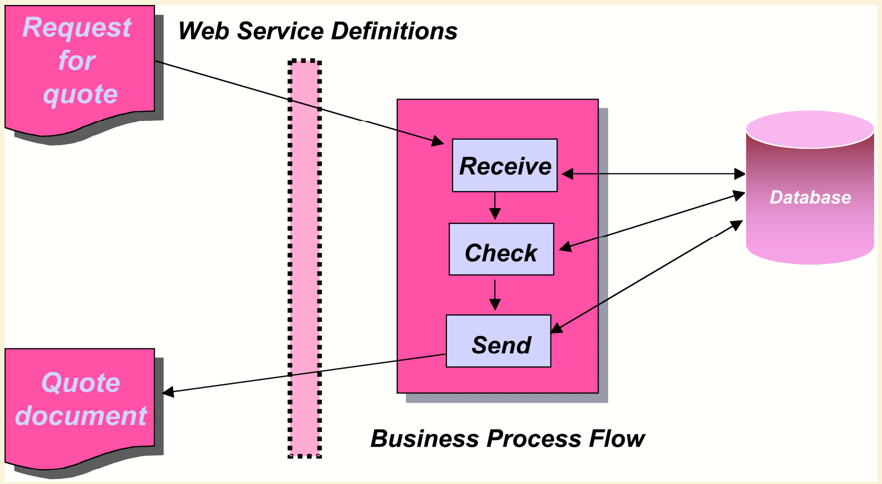
Topics

- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

The SOAP Communication Model

- SOAP supports two possible communication styles:
  - remote procedure call (RPC) and
  - document (or message).

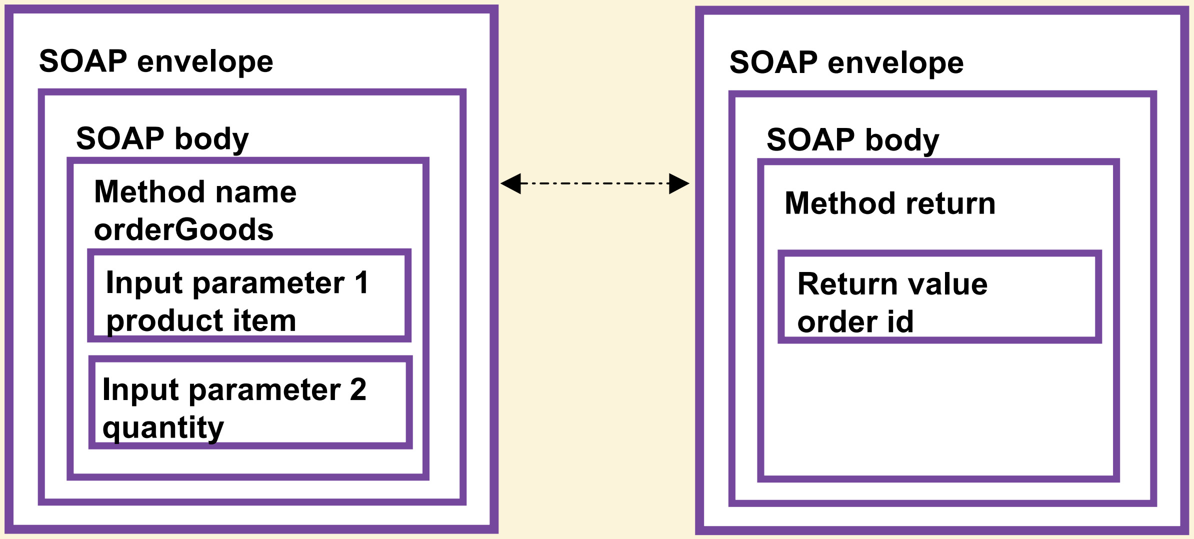




RPC-style interaction **Document-style interaction**

RPC-style SOAP Services

- A remote procedure call (RPC)-style Web service appears as a remote object to a client application. The interaction between a client and an RPC-style Web service centers around a service-specific interface. Clients express their request as a method call with a set of arguments, which returns a response containing a return value.



RPC-style web services

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
            <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
                env:mustUnderstand='1'>
               512
            </tx:Transaction-id>
    </env:Header>
    <env:Body>
            <m:GetProductPrice>
               <product-id>  450R6OP  </product-id >
            </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

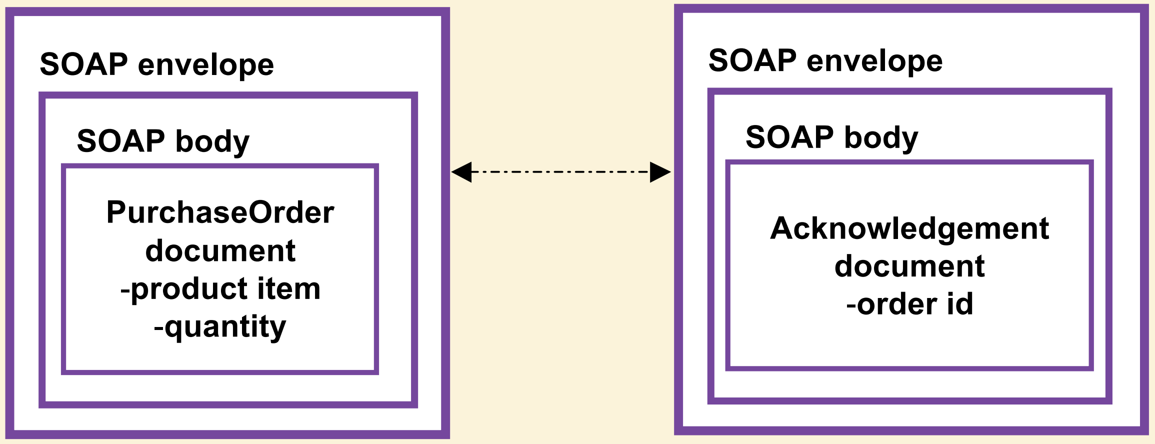
Example of RPC-style SOAP body

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
            <--! – Optional context information -->
    </env:Header>
    <env:Body>
            <m:GetProductPriceResponse>
               <product-price>   134.32  </product-price>
            </m:GetProductPriceResponse>
    </env:Body>
</env:Envelope>
```

Example of RPC-style SOAP response message

Document (Message)-style SOAP Services

- In the document-style of messaging, the SOAP <Body> contains an XML document fragment. The <Body> element reflects no explicit XML structure.
- The SOAP run-time environment accepts the SOAP <Body> element as it stands and hands it over to the application it is destined for unchanged. There may or may not be a response associated with this message.



Example of document-style SOAP body

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">

    <env:Header>
            <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
                env:mustUnderstand='1'>
               512
    </env:Header>
    <env:Body>
            <po:PurchaseOrder oderDate="2004-12-02"
           xmlns:m="http://www.plastics_supply.com/POs">
           <po:from>
             <po:accountName>   RightPlastics   </po:accountName>
                  <po:accountNumber>   PSC-0343-02  </po:accountNumber>
           </po:from>
           <po:to>
             <po:supplierName>  Plastic Supplies Inc.  </po:supplierName>
             <po:supplierAddress>  Yara Valley Melbourne  </po:supplierAddress>
           </po:to>
           <po:product>
               <po:product-name> injection molder </po:product-name>
               <po:product-model> G-100T </po:product-model>
               <po:quantity> 2 </po:quantity>
           </po:product>
            </ po:PurchaseOrder >
    </env:Body>
</env:Envelope>
```

Example of document-style SOAP body

Topics

- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

SOAP Fault element

- SOAP provides a model for handling faults arise.
- It distinguishes between the conditions that result in a fault, and the ability to signal that fault to the originator of the faulty message or another node. The SOAP <Body> is the place where fault information is placed.

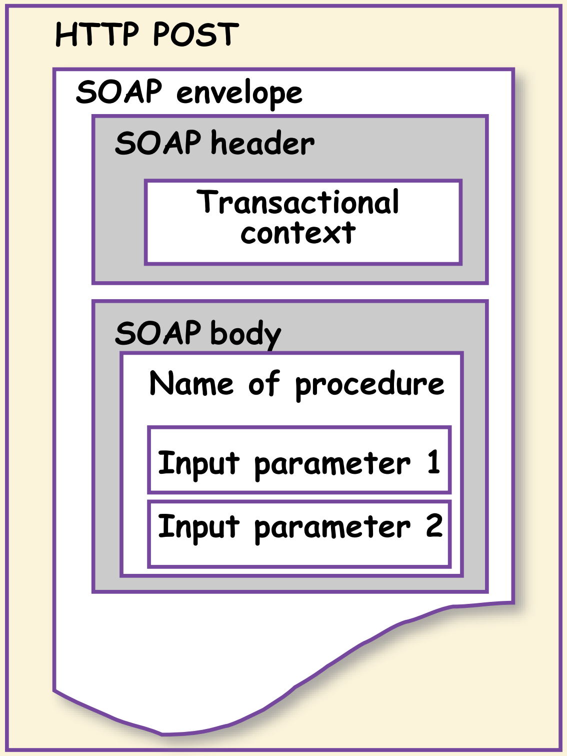
```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
               <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
                   env:mustUnderstand='1'>
               512
               </tx:Transaction-id>
    </env:Header>
    <env:Body>
       <env:Fault>
          <env:Code>
             <env:Value>env:Sender</env:Value>
             <env:Subcode>
                <env:Value> m:InvalidPurchaseOrder </env:Value>
             </env:Subcode>
          </env:Code>
          <env:Reason>
             <env:Text xml:lang="en-UK"> Specified product did not exist </env:Text>
          </env:Reason>
          <env:Detail>
            <err:myFaultDetails
               xmlns:err="http://www.plastics_supply.com/faults">
              <err:message> Product number contains invalid characters </err:message>
              <err:errorcode> 129 </err:errorcode>
            </err:myFaultDetails>
          </env:Detail>
       </env:Fault>
    </env:Body>
</env:Envelope>
```

Topics

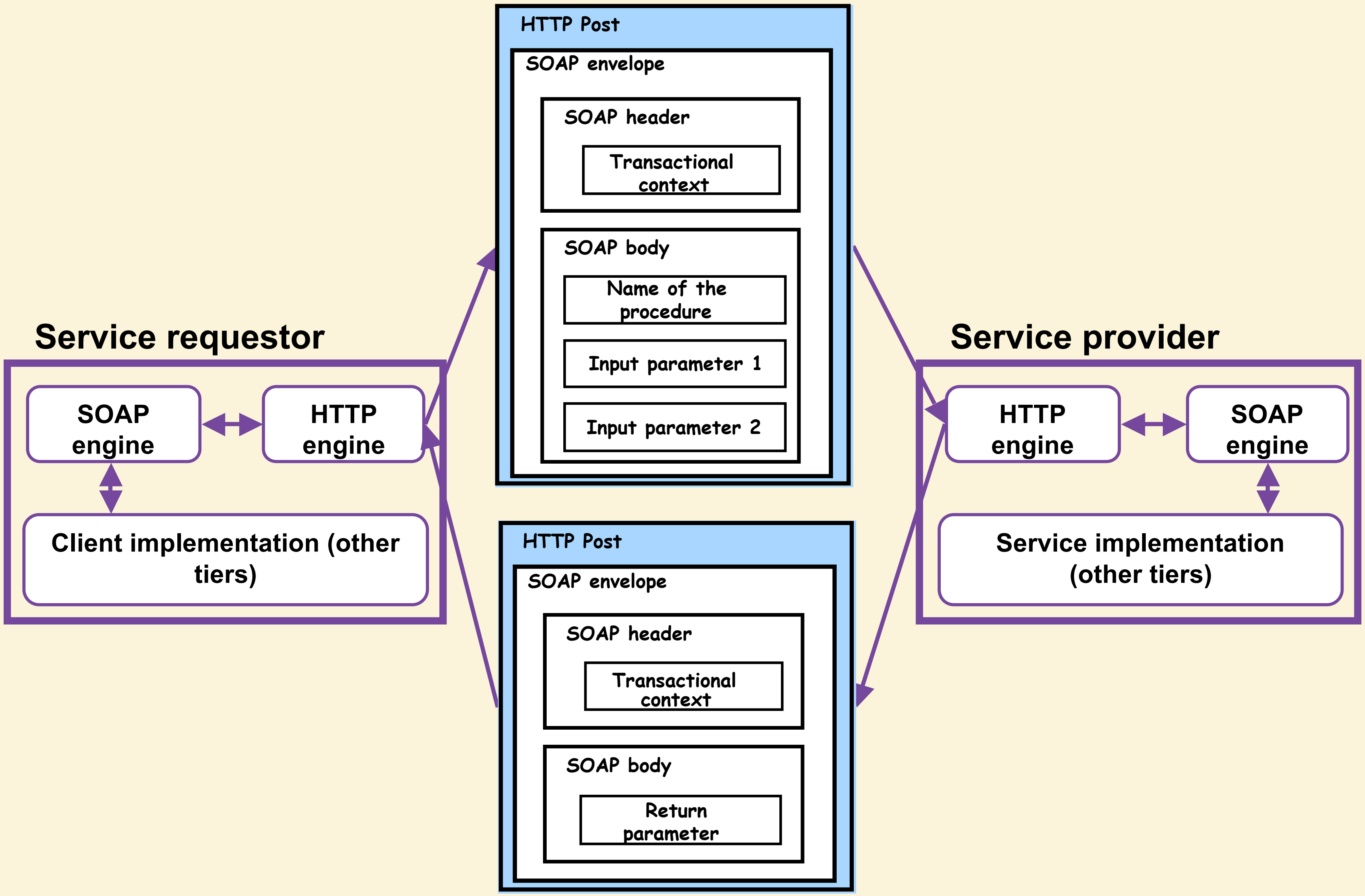
- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

SOAP and HTTP

- A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent using that transport protocol.
- The typical binding for SOAP is HTTP.
- SOAP can use GET or POST. With GET, the request is not a SOAP message but the response is a SOAP message, with POST both request and response are SOAP messages (in version 1.2, version 1.1 mainly considers the use of POST).
- SOAP uses the same error and status codes as those used in HTTP so that HTTP responses can be directly interpreted by a SOAP module.



RPC call using SOAP over HTTP



Topics

- *Inter application communication*
- *SOAP as a messaging protocol*
- *Structure of a SOAP message*
- *SOAP communication model*
- *SOAP fault message*
- *SOAP over HTTP*
- *Advantages and disadvantages of SOAP*

Advantages and disadvantages of SOAP

- Advantages of SOAP are:
  - Simplicity ⟵ **Try not to laugh too hard at this assertion!!!**
  - Portability
  - Firewall friendliness
  - Use of open standards
  - Interoperability
  - Universal acceptance.
- Disadvantages of SOAP are:
  - Too much reliance on HTTP
  - Statelessness
  - Serialization by value and not by reference.