SOAP

Jasmien De Ridder and
Tania Van Denhouwe

Content

- Introduction
- Structure and semantics
- Processing model
- SOAP and HTTP
- Comparison (RPC vs. Message-based)
- SOAP and REST
- Error handling
- Conclusion

Introduction of SOAP

- What is SOAP?
- What is it used for?
- How has it evolved over time?

What is SOAP?

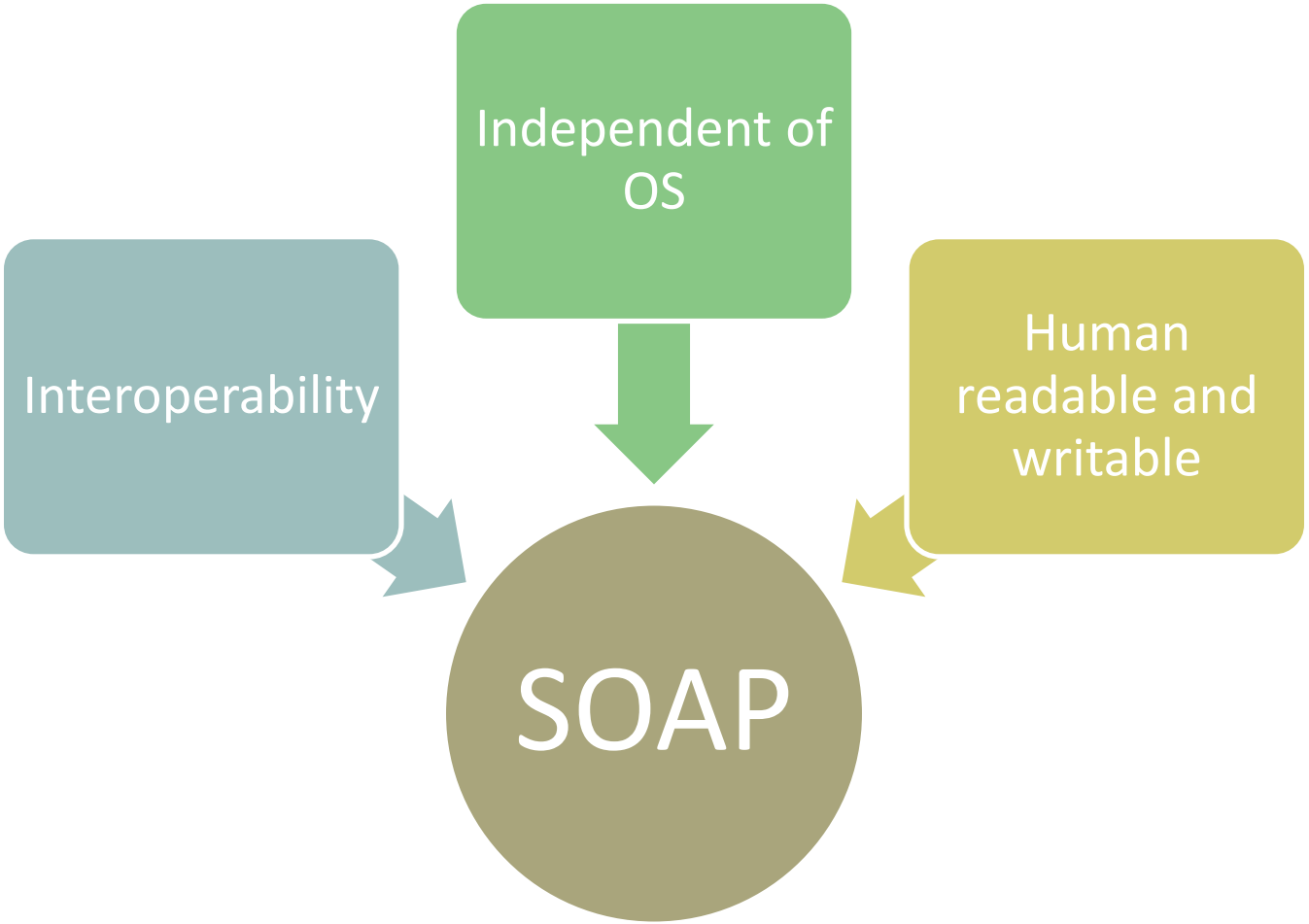
What is SOAP?

- Simple Object Access Protocol
  →not transportation

- Lightweight wire protocol
  (exchange of information)

- XML-messages over HTTP(S)/SMTP/FTP

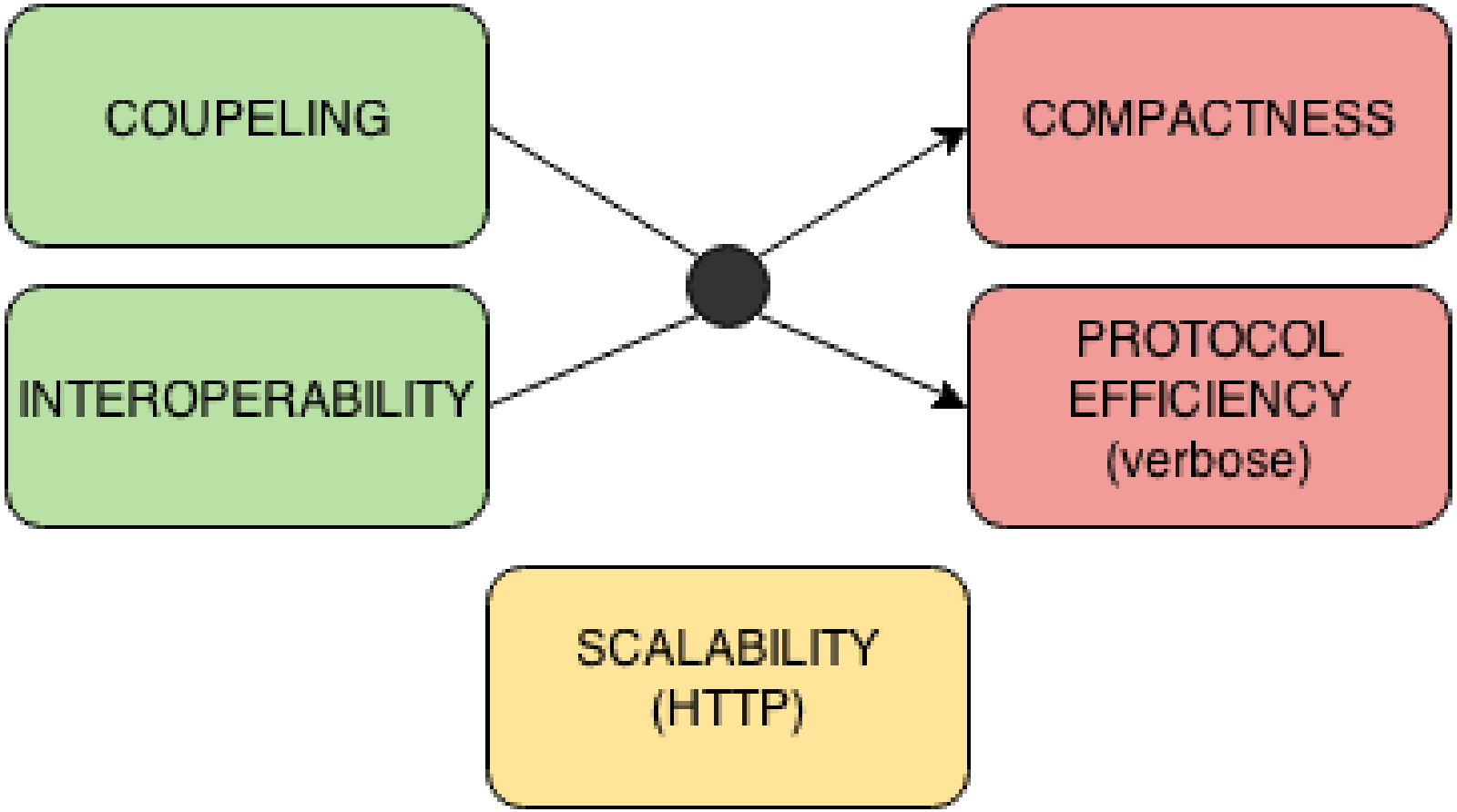
Why use SOAP?


Independent of OS
Interoperability
Human readable and writable
SOAP

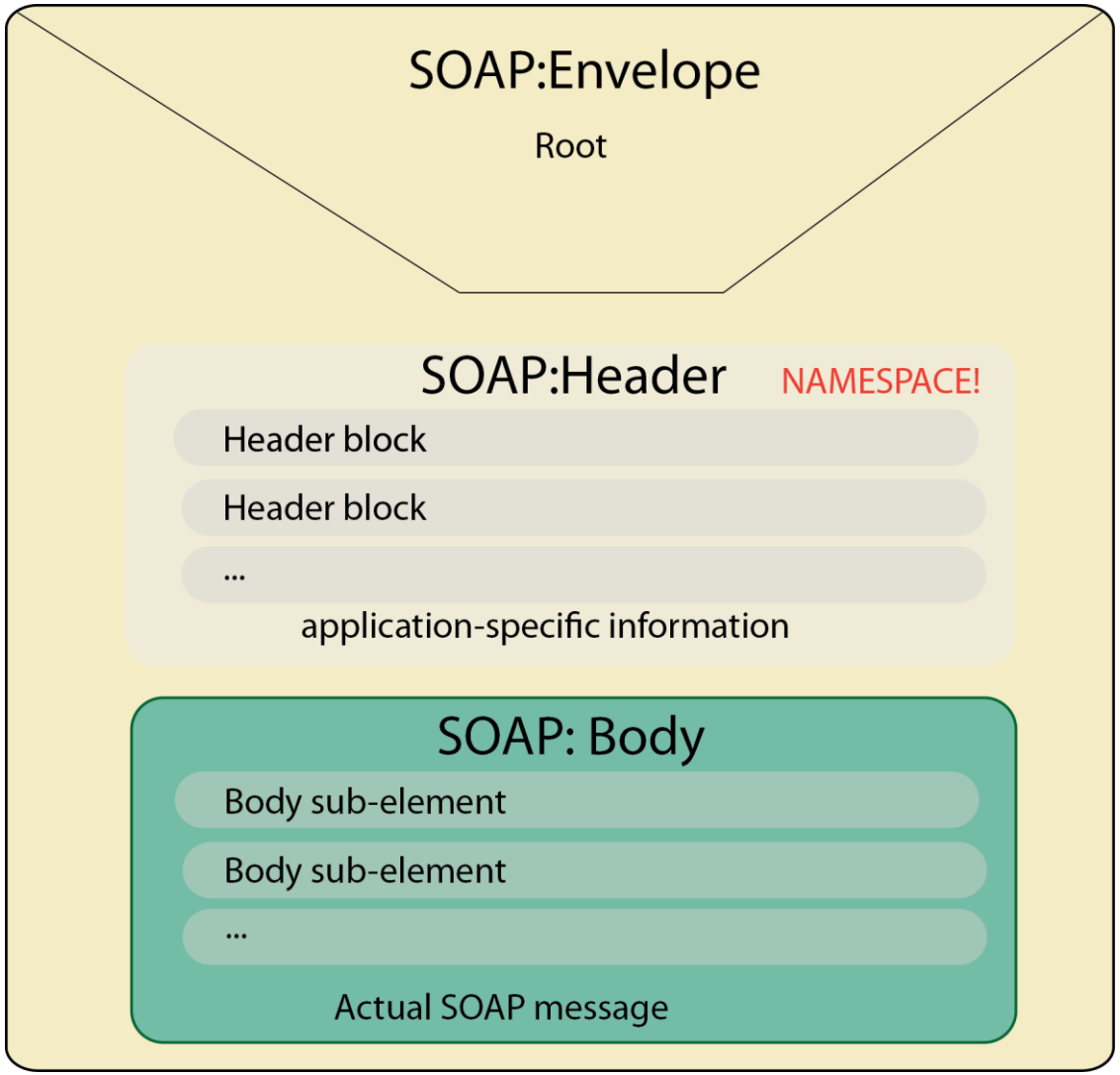
History

- Simple Object Access Protocol (Microsoft 1998)
  - XML-RPC
- SOAP 1.1
  - Modular design
- SOAP 1.2 W3C recommandation (2003)
  - XML Protocol Working Group (2000)

Design criteria



Structure of SOAP-message



Semantics: envelope

- Root element
- Identify as SOAP message
- Namespace name → SOAP version
- Contains
  - (Header)
  - Body

```
<soap:Envelope

xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
>
        <soap:Header>  <!-- optional -->
                <!-- header blocks go here... -->
        </soap:Header>

        <soap:Body>
                <!-- payload or Fault element goes here... -->
        </soap:Body>
</soap:Envelope>
```

Semantics: header

- Optional but before body
- Generic container
  - Control information
    - Negotiation of behaviour
    - Contextual information
  - Any number of elements (header blocks)
    - Individual nodes
  - Any namespace
- Global SOAP attribute
  - mustUnderstand→understand header before processing

Semantcis:Body

- Mandatory
- Message payload
- Generic container
  - Any number of elements
  - Any namespace
  - Data to send
  - Fault element (errors)

Example: transfer funds

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <soap:Body>

        <x:TransferFunds
          xmlns:x="urn:examples-org:banking">

            <from>22-56677</from>
            <to> 56-878895</to>
            <amount> 55.55</amount>

        </x:TransferFunds>

    </soap:Body>

</soap:Envelope>
```

START of SOAP message
Namespace = envelope namespace

STARTof BODY (no header)

START of TransferFunds
Namespace= banking

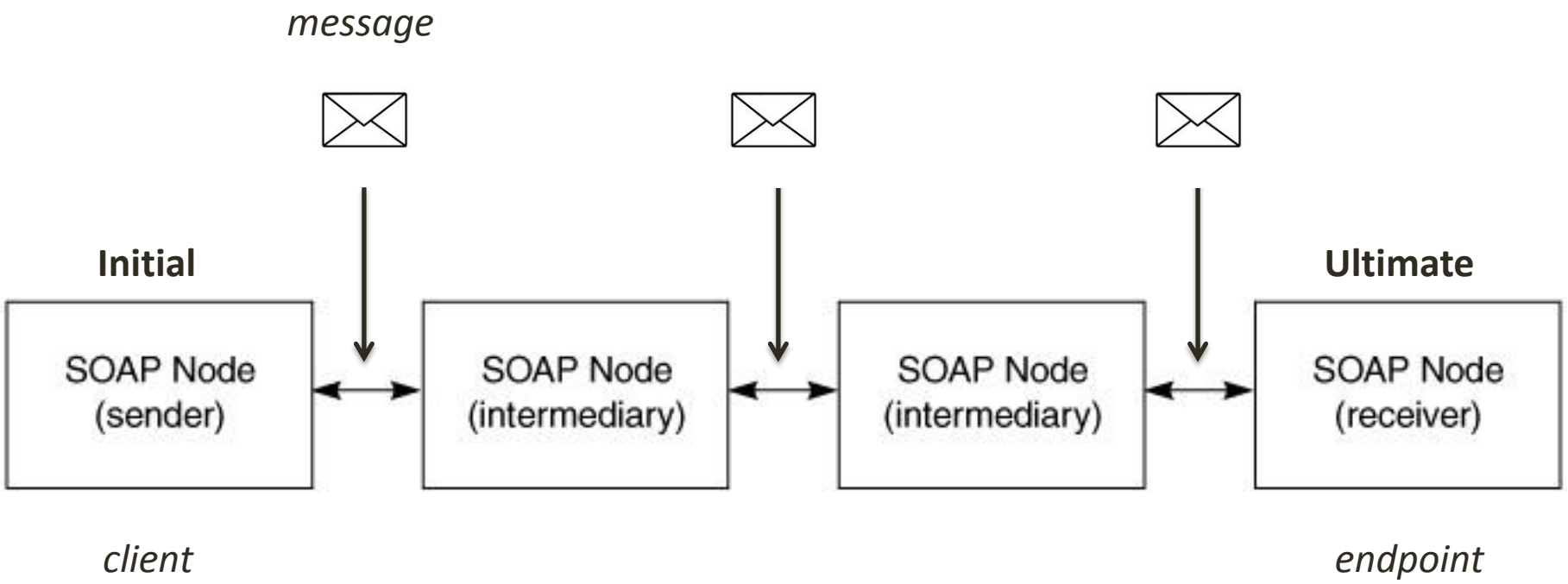
Transfer from account 22-56677
To account 56-878895
The amount 55,55

END of TransferFunds

END of BODY

END of SOAP message

Processing Model



SOAP Node

- Processing the message
  1. Verify if it is a SOAP message
  2. Identify and process header blocks targeted at the node
  3. If the node is an intermediary, forward the request.
  4. If not, process the SOAP body.

- Roles
  - Next
  - None
  - Ultimate Receiver

SOAP and HTTP

- SOAP is most widely used with HTTP
  - Works well with internet infrastructure
  - SOAP model tunnels fine in the HTTP request/response model

HTTP SOAP GET request

```
GET /travelcompany.example.org/reservations?code=FT35ZBQ  HTTP/1.1
Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml
```

HTTP SOAP GET response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
       env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-30T16:25:00.000-05:00</m:dateAndTime>
  </m:reservation>
 </env:Header>
 <env:Body>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:x="http://travelcompany.example.org/vocab#"
      env:encodingStyle="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <x:ReservationRequest
 rdf:about="http://travelcompany.example.org/reservations?code=FT35ZBQ">
      <x:passenger>Åke Jógvan Øyvind</x:passenger>
      <x:outbound>
       <x:TravelRequest>
        <x:to>LAX</x:to>
        <x:from>LGA</x:from>
        <x:date>2001-12-14</x:date>
       </x:TravelRequest>
      </x:outbound>
      <x:return>
       <x:TravelRequest>
        <x:to>JFK</x:to>
        <x:from>LAX</x:from>
        <x:date>2001-12-20</x:date>
       </x:TravelRequest>
      </x:return>
    </x:ReservationRequest>
  </rdf:RDF>
 </env:Body>
</env:Envelope>
```

HTTP SOAP POST request

```
POST /InStock HTTP/1.1
    Host: www.example.org
    Content-Type: application/soap+xml; charset=utf-8
    Content-Length: nnn

    <?xml version="1.0"?>
    <soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

      <soap:Body xmlns:m="http://www.example.org/stock">
            <m:GetStockPrice>
                  <m:StockName>IBM</m:StockName>
            </m:GetStockPrice>
      </soap:Body>

    </soap:Envelope>
```

HTTP SOAP POST response

```
HTTP/1.1 200 OK
    Content-Type: application/soap+xml; charset=utf-8
    Content-Length: nnn

    <?xml version="1.0"?>
    <soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

      <soap:Body xmlns:m="http://www.example.org/stock">
            <m:GetStockPriceResponse>
             <m:Price>34.5</m:Price>
            </m:GetStockPriceResponse>
      </soap:Body>

    </soap:Envelope>
```

SOAP a messaging protocol

- Diffuse barriers of heterogeneity
  - Simplicity
  - Flexibility
  - Firewall friendliness
  - Platform neutrality
  - XML message based (reuse)

  →Standardize communication over the web

- SOAP-message= SOAP-XML Document

SOAP-message

- Network application protocol
  - Transfer over HTTP/…
  - Services with WSDL interfaces

- SOAP-encapsulates
  - Format
  - Not delivery
    - Where
    - How

SOAP messaging to RPC (1)

- SOAP-fundamentally
  - Stateless
  - One-way
  - Messaging paradigm
  - BUT more complex interaction patterns possible
    - Underlying layers/ protocols
    - Application specific information
    - One-way exchanges
- Simple mechanism
  - Packaging
  - Encoding in modules
  - No routing

SOAP messaging to RPC (2)

- message
  - Body contains only XML
- RPC (remote procedure call)
  - Body contains XML representation of method call
    - XML ≠ encoded into XML

XML encoding	XML-Shema
At runtime	No ambiguity
Error prone	Literally defines format

Message exchange

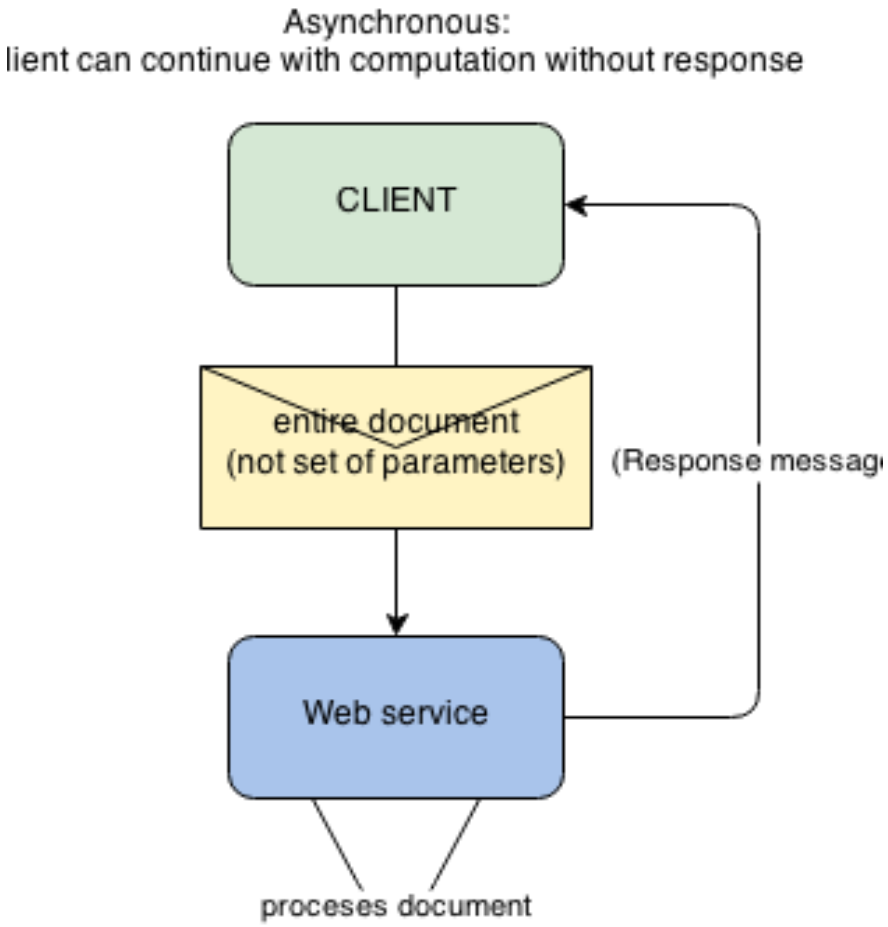
- Request-response pattern
  - 2 response possibilities (like HTTP)
    - Requested information
    - Fault message
  - Mostly used for RPC

- Other usage scenarios
  - XML-based content exchange
  - Conversational message exchanges

- SOAP 1.2:
  - SMTP, TFP
    → Global exchange protocol
  - Intermediary nodes

SOAP as a message (message based)

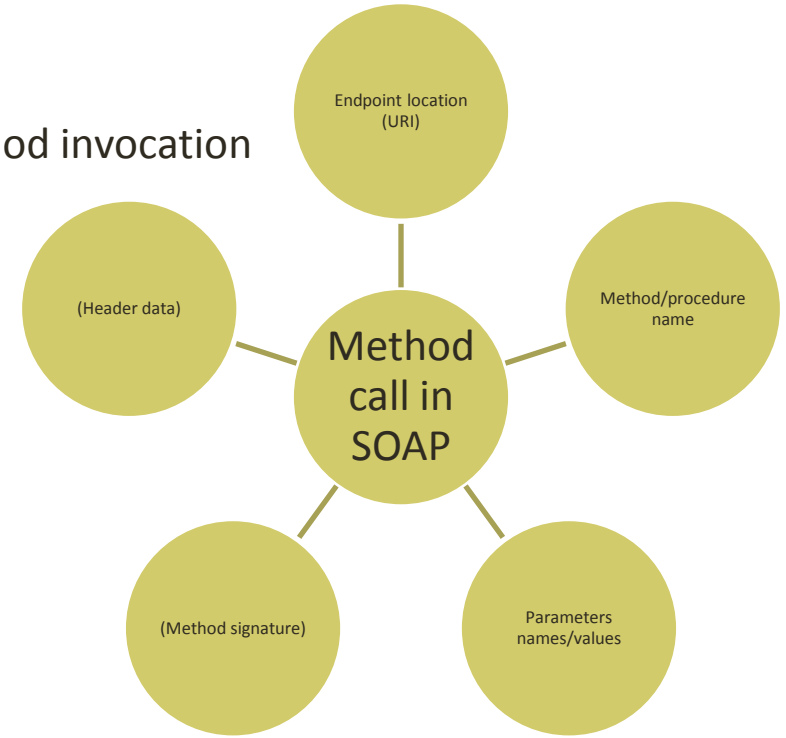
- Message driven
- Asynchronous
- ≠RPC
  - No serialization/deserialization
  - Assumes well formedness



SOAP as RPC (1)

- SOAP >> objects

- Encapsulate & exchange RPC calls
  - Mapping: RPC → SOAP
    - At runtime: SOAP message ←→ method invocation
    - No redesign



VIA: IDL files, type libraries, WSDL files,…

SOAP as RPC (2)

- RPC and http binding:
  - Associate request with response
  - Useful when client communicates with multiple providers
- Method signature → request/response →XML
- Invocation: STRUCT (method name)
  - in
  - In/out
- Response: STRUCT (convention: name+ Response)
  - Return value (SOAP 1.2 rcp:result)
  - Out
  - In/out

SOAP as RPC:Example

double add(ref double x, double y)

- Invocation:
  struct add{double x; double y}

- Response:
  struct addResponse{double result; double x}

→XML:

```
<add>
    <x> value </x>
    <y>value</y>
</add>
```

```
<addResponse>
    <result> 77 </result>
    <x>      33  </x>
<addResponse>
```

Soap encoding rules → XML Schema definitions

SOAP and REST

- RESTful APIs do not require XML-based web services
- answer to the same question:
  - How to access Web services
- SOAP ←→ REST
  - REST came after SOAP
  - REST: Truly simple accessing method
  - SOAP: sometimes easier
  - SOAP: more heavy weight

SOAP VS. REST

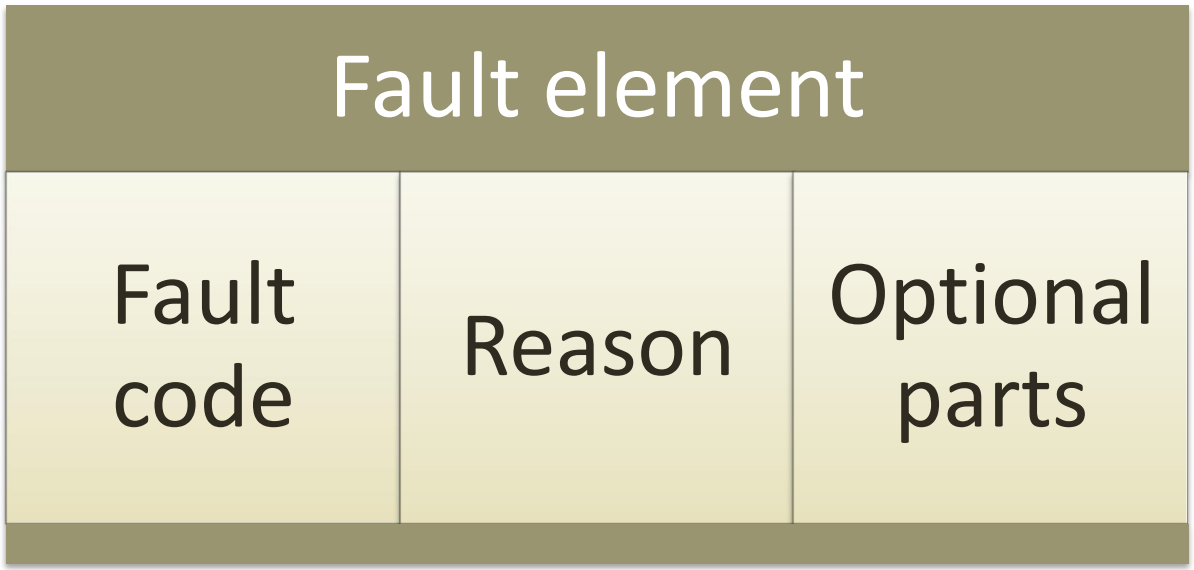
SOAP	REST
Transport, language and platform independent (not only HTTP)	No expensive tools required
Works well in distributed enterprise environments (not only point-to-point)	Smaller learning curve
Provides extensibility in form of WS* standards	Efficient (smaller message format)
Automation when used with certain language products	Fast
	Closer to web technologies in design filosophy

Error handling

- How does an error message looks like?
- How does SOAP treat errors?
- What kind of errors can it handle?

Structure of error message

- SOAP Fault element
  - holds errors and status information
  - only one fault element per message



Example error message

```
<?xml version="1.0"?>
<env:Envelope
    xmlns:env=http://www.w3.org/2003/05/soap-
    envelope>
    <env:Body>
        <env:Fault>
        …
        </env:Fault>
    </env:Body>
</env:Envelope>
```

Example error message(2)

- Fault code and subcodes

```
<env:Code>
    <env:Value>env:Sender</env:Value>
        <env:Subcode>
            <env:Value>rpc:BadArguments</env:Value>
        </env:Subcode>
</env:Code>
```

Example error message(3)

- Reason
  - Human-readable description of fault

```
<env:Reason>
    <env:Text xml:lang=en-US>Processing error<env:Text>
</env:Reason>
```

Example error message(4)

- <env:Node>
  - Actor causing the fault
- <env:Role>
  - Role being played by actor at the time
- <env:Detail>
  - Application-specific info

SOAP conclusion

ADVANTAGES	DISADVANTAGES
Simplicity (XML)	Stateless
Interoperability	Slower due to verbose XML format
Independent of operating system	WSDL dependence
Firewall-friendly	Serialization by value and not by reference
Use of open standard	Not all languages offer appropriate support
Widely accepted	

References

- M. P. Papazoglou; Web Services: Principles and Technology; chapter 4, pages 119-145.
- http://www.w3.org/TR/2007/REC-soap12-part1-20070427/
- http://msdn.microsoft.com/en-us/library/ms995800.aspx
- David A Chappell, Tyler Jewell; Java Web Services; chapter 4, section 2.