Software Service Engineering

Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Section 4

HTTP EXTENSION: WEBDAV

Introduction

- **Distributed Authoring and Versioning Protocol for the World Wide Web (WebDAV)** – An extension to the HTTP/1.1 protocol that allows clients to perform remote Web content authoring operations.
- IETF Standard: RFC 4918

Terminology

- **Collection** – A resource that contains a set of termed member URIs, which identify member resources
- **Member URI** – A URI which is a member of the set of URIs contained by a collection
- **Property** – A name-value pair that contains descriptive information about a resource
  - **Live Property** – Semantics and syntax enforced by the server: E.g. "getcontentlength" live property: length of the entity returned calculated by the server
  - **Dead Property** – Semantics and syntax are not enforced by the server:
    Server only records the value - client is responsible for maintaining the value

E.g. Creating Collections

- **Request:**

  MKCOL /martin/contacts/ HTTP/1.1

  Host: www.example.com

- **Response:**

  HTTP/1.1 201 Created

<<**DEMO**>>
Try this at home!

Distributed Authoring Methods

- New and refined HTTP-Methods:
- PROPFIND – retrieve Properties for a Resource (URI)
- PROPPATCH – set and/or remove Properties on a URI
- MKCOL – create a new collection
- GET, HEAD for Collections – as defined in RFC 2068
- POST for Collections – semantics as defined
- DELETE – removes URI (all or none semantics)
- PUT – replaces Get Response Entity
  - Properties defined on the URI may be recomputed
  - Put without a parent collection must fail
- COPY – create a Duplicate of Source-URI in the Destination-URI
- MOVE – move Resource to Destination-URI
- LOCK – take out a Lock of any Access Type on a given Resource (URI)
  - Method describes only those Semantics that are specific to the LOCK
  - But independent of the Access Type of the Lock being requested
  - Shared or Exclusive Lock, e.g.
    <D:locktype><D:write/></D:locktype><D:lockscope><D:exclusive/></D:lockscope>
- UNLOCK – remove the Lock identified by the Lock Token for a URI

E.g. – Property Retrieval I

- **Request:**
  PROPFIND /martin/contacts HTTP/1.1
  Host: www.example.com
  Content-type: text/xml; charset="utf-8"
  Content-Length: …

```
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
    <D:prop xmlns:R="http://www.example.com/contactschema/" />
    <D:allprop />
</D:propfind>
```

E.g. – Property Retrieval II

- **Response:**
  HTTP/1.1 207 Multi-Status
  Content-Type: text/xml; charset="utf-8"
  Content-Length: xxxx

```
<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.example.com/martin/contacts</D:href>
    <D:propstat>
      <D:prop xmlns:R="http://www.example.com/contactschema/">
        <R:description>Contacts of Martin Gaedke</R:description>
        <D:creationdate>1997-12-01T17:42:21-08:00</D:creationdate>
        <D:resourcetype><D:collection/></D:resourcetype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

PART III

SOAP-SERVICES

Introduction

- So far: How to discover Web Services
- Now: How to describe and use Web Services

- Interaction with Web Services
  - Messages & Encoding
  - Message Exchange Patterns (MEP)
  - Interaction Rules
  - Interaction Semantics
- Description of Web Services
  - Remember: Endpoint Description (ABC)

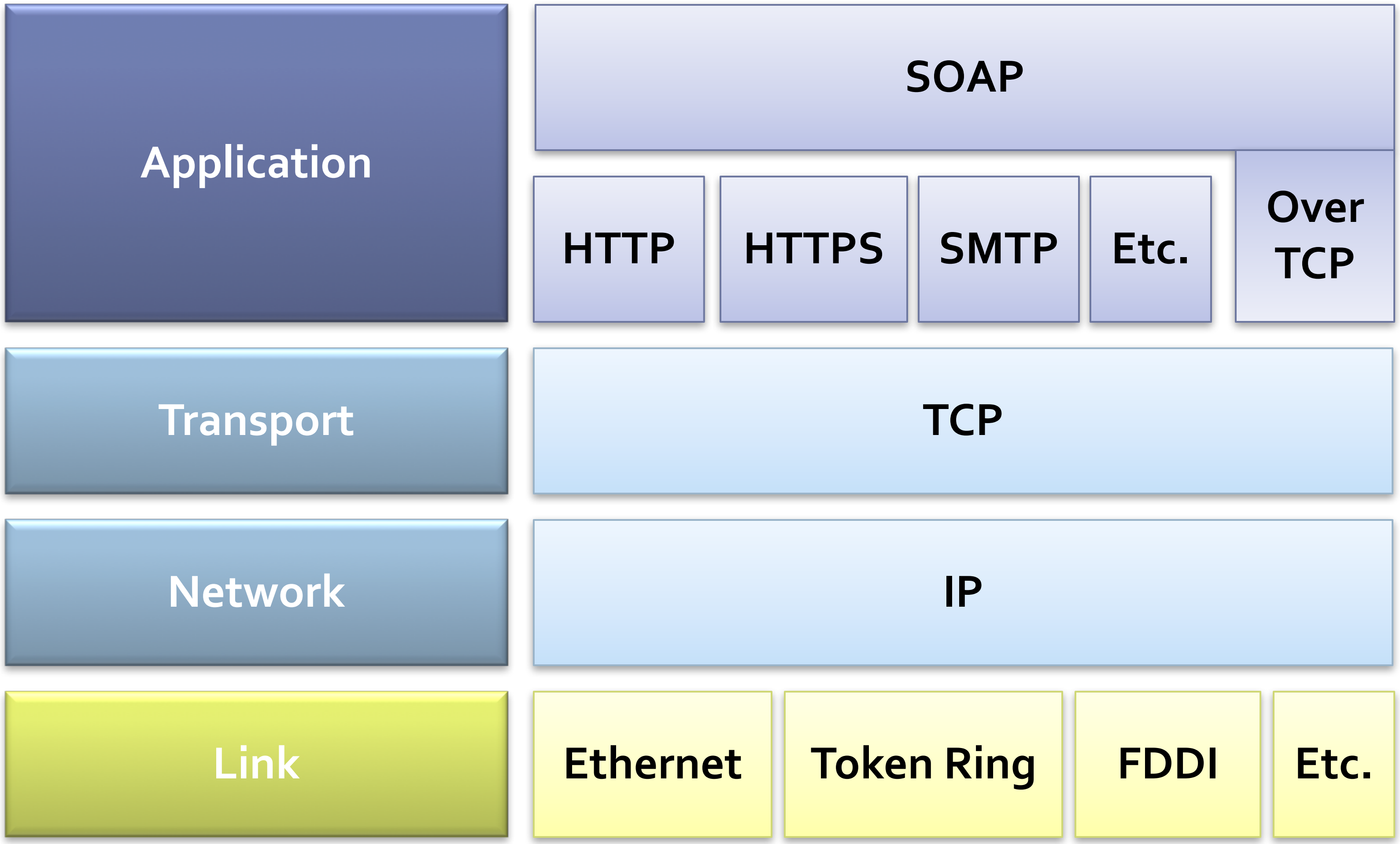
Chapter 1

SOAP

SOAP

- **SOAP** provides a simple and lightweight mechanism of structured and typed data exchange between peers (communication partners) in a decentralized distributed system
  - SOAP Version 1.2 speaks only of SOAP!
  - SOAP does not define any application semantics, e.g. programming model
- SOAP Version 1.2
  W3C Recommendation 27 April 2007
  - Parto - Tutorial: http://www.w3.org/TR/soap12-parto/
  - Part1: Defines Messaging Framework
  - Part2: Adjuncts (SOAP Data Model, SOAP Encoding, SOAP RPC, Message Patterns, Bindings)

SOAP & Internet Protocol Stack



Development of SOAP (1)

- Collaboration of Dave Winer (Mr. RSS) and Microsoft
  - First developments: XML-RPC
  - XML-RPC is still in use here and there (e.g. on some blog sites)
  - With the help of Don Box (founder of DevelopMentor), who later moved to Microsoft; first Version of Simple Object Access Protocol (SOAP) Version 0.9
  - 1999 SOAP Version 1.0

- **Idea of XML-RPC**
  - Transfer the RPC call over HTTP
  - Data is encoded in XML (some of the allowed data types are pre-defined – no schema, though)
- **Example of XML-RPC**

```
<?xml version="1.0"?>
<methodCall>
    <methodName>
        Dienst.add
    </methodName>
    <params>
        <param>
            <value><int>23</int> </value>
        </param>
        <param>
            <value><int>12</int> </value>
        </param>
     </params>
</methodCall>
```

Development of SOAP (2)

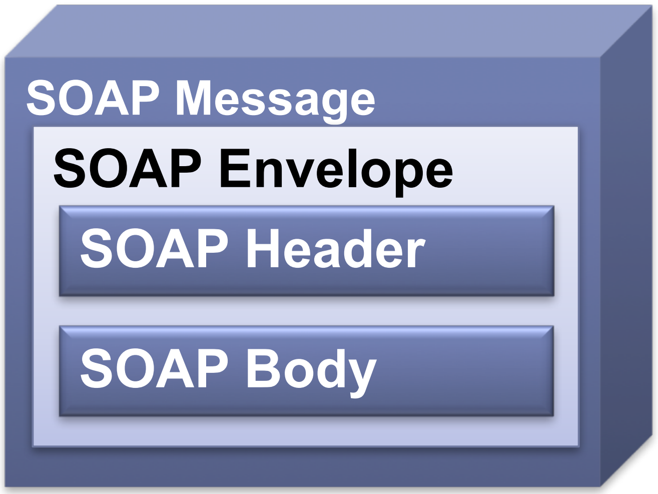
- Potential of SOAP is recognized
  - 2000 IBM joins in
  - Submission of specification at W3C
  - SOAP 1.1 from IBM, Microsoft, DevelopMentor (Don Box) and UserLand Software (Dave Winer)
- W3C working group expands the topic
  - 2007 SOAP 1.2 becomes a W3C Recommendation

SOAP Framework

- **SOAP Envelope**
  - Defines message content, names, namespaces
- **SOAP Encoding Rules**
  - Define serialization mechanism for application-specific data types.
- **SOAP Message Exchange Patterns (MEP)**
  - Define established interaction scenarios, e.g. request-response, fire and forget
- **RPC representation**
  - Definition of Remote Procedure Call and Response representation conventions

SOAP Envelope

- **SOAP Envelope consists of**
  - **Header** (Optional)
  - **Body**
  - And defines Namespace
- SOAP Header
  - Extension mechanism for associating data, which doesn't belong to payload to a SOAP message
  - Serves for transfer of control information such as routing, security, etc. – and especially to share what the receiver MUST understand (**mustUnderstand** attribute)
  - XML Elements in the Header are called Header Blocks
- SOAP Body
  - Application data (so-called **payload**) exchanged between an (initial) SOAP Sender and an (ultimate) SOAP Receiver in an end-to-end manner
  - XML elements in the Body are called Body sub-elements

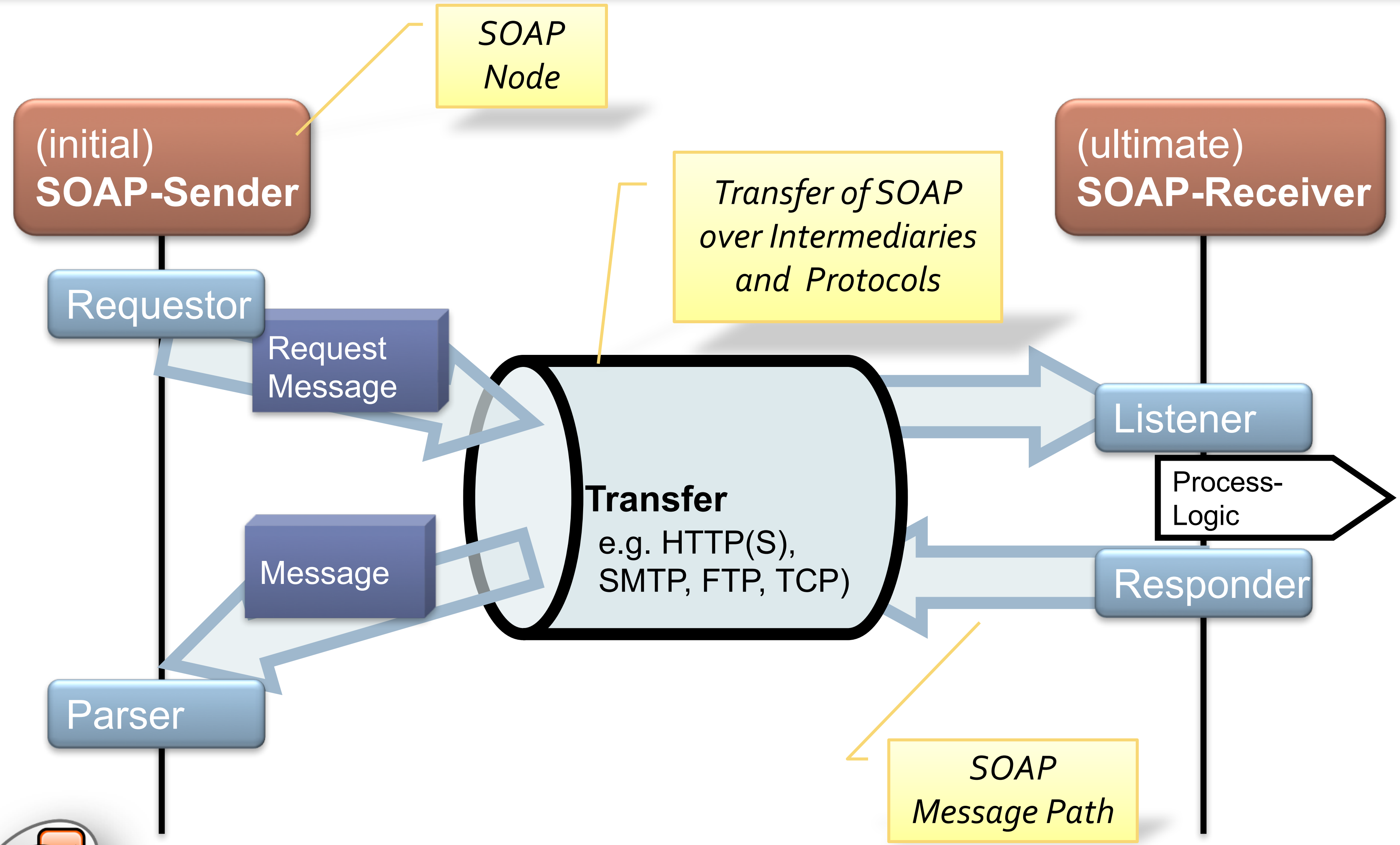


```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap=
"http://www.w3.org/2003/05/soap-envelope">
        <soap:Header>
        </soap:Header>
        <soap:Body>
        </soap:Body>
</soap:Envelope>
```

SOAP in Action – How?

- To work with SOAP one requires:
  - A client, which constructs and sends SOAP requests
  - A server, which understands SOAP requests, implements/executes the required functionality, creates a response in the form of a SOAP response and sends it back to the Client
  - Possibly, SOAP intermediaries, which route the messages
- This scenario is independent of the underlying technology, programming languages, platform
  - It is ultimately realized by a Web Application

SOAP in Action



Message Exchange Patterns

- Template, devoid of application semantics, that describes a generic pattern for the exchange of messages between agents.
- Describes the relationships (e.g., temporal, causal, sequential, etc.) of multiple messages exchanged in conformance with the pattern, as well as the normal and abnormal termination of any message exchange conforming to the pattern.
- SOAP patterns (more patterns are defined in WSDL spec):
  - Request-Response Message Exchange Pattern
  - Response Message Exchange Pattern

SOAP with HTTP Binding

- SOAP-Request via HTTP-POST-Request

```
POST /WebCalculator/Calculator.asmx HTTP/1.1
Content-Type: text/xml
...
~~SOAPAction: "http://tempuri.org/Add"~~
Content-Length: 386

<?xml version="1.0"?>
<soap:Envelope ...>
  ...
</soap:Envelope>
```

SOAPAction is not used anymore (SOAP1.2)

SOAP-Envelope Example (1)

- SOAP-Schema XML document

```
<?xml version="1.0"?>
<soap:Envelope ...>
  <soap:Header ...>
      <hb1:headerblock1 soap:mustUnderstand/>
      <hb2:headerblock2/>...
  </soap:Header>
  <soap:Body>
    <MyQuery xmlns="http://tempuri.org/">
      <n1>12</n1>
      <n2>10</n2>
    </MyQuery >
  </soap:Body>
</soap:Envelope>
```

SOAP-Envelope Example (2)

- Data structures serialized in XML

```
<soap:Envelope ...>
 <soap:Body>
  <MyQueryResult xmlns="http://tempuri.org/">
   <result>
    <Description>Plastic Novelties Ltd</Description>
    <Price>129</Price>
    <Ticker>PLAS</Ticker>
   </result>
  </MyQueryResult>
 </soap:Body>
</soap:Envelope>
```

SOAP Example (1)

- SOAP-Request through HTTP-POST

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn

<env:Envelope
   xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <m:symbol>DIS</m:symbol>
       </m:GetLastTradePrice>
   </env:Body>
</env:Envelope>
```

SOAP Example (2.1)

- SOAP-Response over HTTP

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

<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <m:GetLastTradePriceResponse xmlns:m="Some-URI">
       <Price>34.5</Price>
      </m:GetLastTradePriceResponse>
   </soap:Body>
</soap:Envelope>
```

SOAP Example (2.2)

- SOAP-Fault (error message) over HTTP

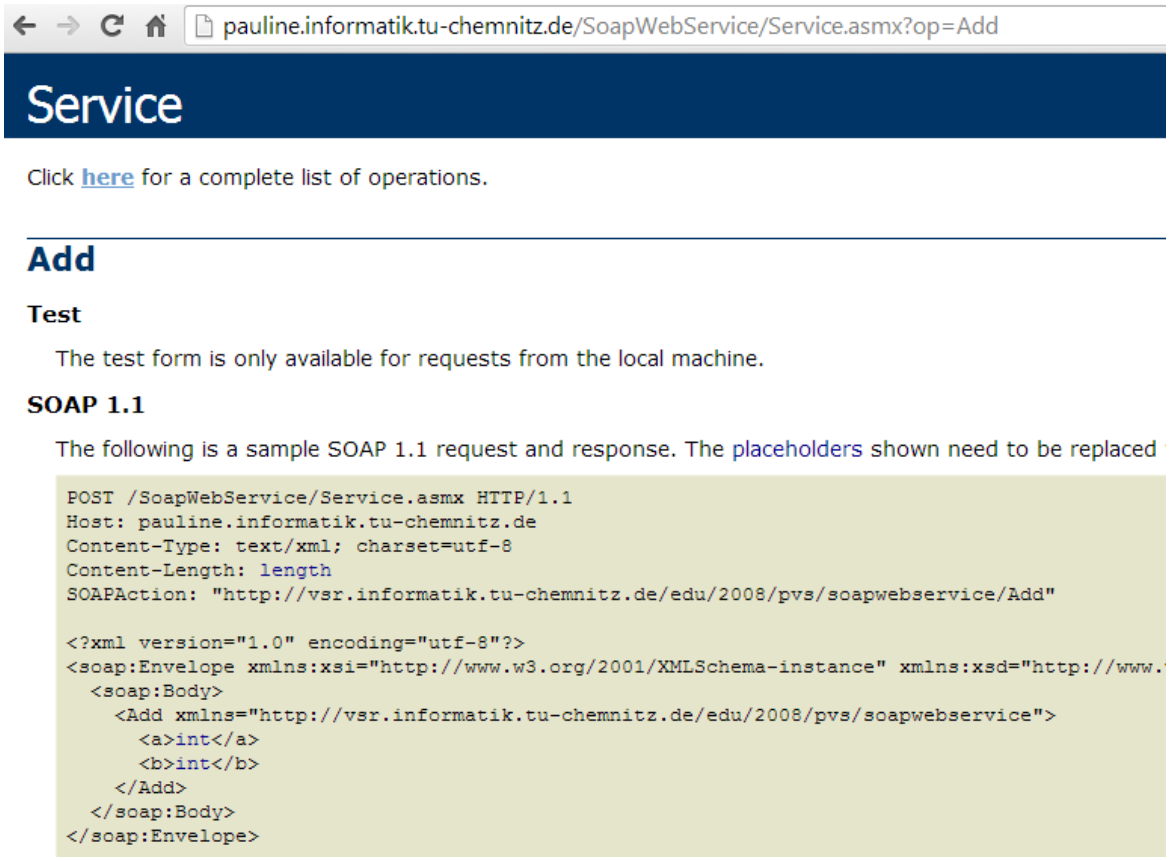
```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <soap:Fault>
         <faultcode>SOAP: MustUnderstand</faultcode>
         <faultstring>SOAP Must Under Error</faultstring>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

SOAP - Final Considerations

- SOAP security
  - End-to-end security – what does it mean in SOAP context?
  - HTTPS can be used in the HTTP context, does it enable end-to-end security?
- The strength of SOAP lies in its extensibility (similarly to HTTP)
  - Which leads to:
    - a variety of protocols that build up on SOAP
    - a lot of extensions, which systematically expand SOAP's capabilities
    - a variety of further bindings that enable using other protocols to transfer SOAP messages
- SOAP (as well as HTTP) is one of the pillars of Web Services

SOAP - Demo

http://pauline.informatik.tu-chemnitz.de/SoapWebService/Service.asmx


pauline.informatik.tu-chemnitz.de/SoapWebService/Service.asmx?op=Add

Service

Click here for a complete list of operations.

Add

Test

The test form is only available for requests from the local machine.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced

```
POST /SoapWebService/Service.asmx HTTP/1.1
Host: pauline.informatik.tu-chemnitz.de
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://vsr.informatik.tu-chemnitz.de/edu/2008/pvs/soapwebservice/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.
  <soap:Body>
    <Add xmlns="http://vsr.informatik.tu-chemnitz.de/edu/2008/pvs/soapwebservice">
      <a>int</a>
      <b>int</b>
    </Add>
  </soap:Body>
</soap:Envelope>
```