PS Distributed Systems

Patricia Derler

02.04.2008

Schedule

	Mo	Tue	Wed	Thu	Fr	Sat	Sun
31.3. – 6.4.			2.4.				
7.4. – 13.4.			9.4.				
14.4. – 20.4.							
21.4. – 27.4.			23.4.				
28.4. – 4.5.							
5.5. – 11.5.			7.5.				
12.5. – 18.5.							
19.5. – 25.5.			21.5.				
26.5. – 1.6.							
2.6. – 8.6.			4.6.				
9.6. – 15.6.							
16.6. – 22.6.			18.6.				
23.6. – 29.6.							

Web Services

Definition:

> *"A web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols."*

(Web Services Architecture document, W3C)

Anwendungsgebiete

- Frei verfügbare Dienste
  - Daten werden angeboten, damit sie in möglichst viele bestehende Services integriert werden können.
  - Beispiele: Veranstaltungs-, Tourismus-, Wetterdaten.
- Gebührenpflichtige Dienste
  - Zukauf von Ressourcen (Speicher, Rechenleistung, ...).
  - Zukauf von Daten: GIS-Daten.
- Enterprise Application Integration (EAI)
  - Verteilung der Business-Logik auf mehrere Standorte.
- B2B Datenaustausch
  - Ablöse bestehender Standards (EDIFACT, ...)

Relevante Standards

- XML: eXtensibleMarkupLanguage
  - Strukturierte Darstellung von Daten,
  - Metasprache zur Definition von Sprachen,
  - Anwendung: UDDI, WSDL, SOAP.
- XML-Schema
  - Definition der Grammatik von XML-Sprachen.
- SOAP: Simple ObjectAccess Protocol
  - Standardisierte Darstellung von Daten,
  - Darstellung von Methodenaufrufen und Parametern.
- WSDL: Web Service Description Language
  - Beschreibungssprache für Web-Services.

WS-I Basic Profile

- WS-I (= Web Services Interoperability Organization)
  - Vereinigung von Anbietern und Benutzern von Web-Services-Plattformen: Microsoft, Sun, IBM, BEA, …
  - Aufgaben:
    - Definition von „Profilen“,
    - Erstellung von Bespielszenarien und –code für Web-Services,
    - Erstellung von Werkzeugen für Konformitätstests.
- WS-I Basic Profile 1.0, 1.1
  - Spezifikationen (WSDL, SOAP, …) sind sehr umfassend und oft nicht eindeutig.
  - Basic Profile schränkt Spezifikationen ein.
  - Ziel: Interoperabilität zwischen allen Herstellen.

XML

- Metasprache zur Definition anderer Sprachen
- XML-Sprachen beschreiben die Struktur von Dokumenten und Daten.

```
<?xml version="1.0" encoding="UTF-8" ?>
<person category="business">
  <name>Mustermann</name>
  <age>30</age>
  <address>
    <street>Jakob-Haringer-Str. 2</street>
    <place>Salzburg</place>
    <zip>5020</zip>
  </address>
</person>
```

Namensräume

- Aufgabe: Gewährleistung der Eindeutigkeit von Tags und Attributen.
- Default-Namenraum: `<myTag xmlns="URI"...>`
- Deklaration eines Namenraums: `<myTag xmlns:myNS="URI"...>`
- Verwendung eines Namenraums: `<myNS:tag>...</myNS:tag>`

```
<person category="business"
 xmlns="http://myCompany/person"
 xmlns:addr="http://myCompany/address">
  <name>Mustermann</name>
  <age>30</age>
  <addr:address>
    <addr:street>Jakob-Jaringer-Str. 2</addr:street>
    <addr:place>Salzburg</addr:place>
    <addr:zip>5020</addr:zip>
  </addr:address>
</person>
```

Eigenschaften und Verarbeitung

- Eigenschafen eines XML-Dokuments
  - Wohlgeformtheit (*well-formedness):*
    - Dokument entspricht den Regeln der XML-Spezifikation.
      - Ein Wurzelelement
      - Start- und Endtags
      - Werte der einfachen Typen im korrekten Wertebereich
  - Validität (*validity).*
    - Dokumentstruktur entspricht einer vorgegebenen Beschreibung (DTD oder XML-Schema).

- Arten von Parsern:
  - DOM: Parser generiert eine baumartige Repräsentation.
  - SAX: Parser generiert bei Abarbeitung Ereignisse.

Beschreibung von XML-Dokumenten: DTD

- Möglichkeit 1: DocumentType Definition (DTD)

Person.dtd

```
<!ELEMENT person(name, age?, address+)>
<!ELEMENT age(#PCDATA)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT address(street, place?, zip)>
<!ELEMENT place(#PCDATA)>
<!ELEMENT street(#PCDATA)>
<!ELEMENT zip(#PCDATA)>
<!ATTLIST person category CDATA #REQUIRED>
```

Person.xml

```
<!DOCTYPE person SYSTEM "person.dtd">
<person category="business"> ... </person>
```

Nachteile:

Es kann lediglich festgelegt werden, dass Elemente andere Elemente, Text oder nichts enthalten dürfen.

Der Datentyp von Blättern kann nicht definiert werden.

Beschreibung von XML-Dokumenten: Schema

- XML-Schema: XML-Sprache zur Beschreibung von XML-Sprachen.
- (*.xsd)

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://myCompany/person"
 xmlns:tns="http://myCompany/person">
  <element name="person" type="tns:personType"/>
  <complexType name="personType">
    ...
  </complexType>
  <complexType name="addressType">
    ...
</complexType>
</schema>
```

XML-Schema: Typen

- Ein XML-Schema enthält die Definition von
  - einfachen und
  - komplexen Typen

- Einfache Typen:
  - Definition eines Elements mit einem einfachen Typ:

```
<element name="street" type="string">
```

  - Die XML-Schema definition des W3C definiert 44 Standardtypen (*"built-intypes"*)

```
string, short, int, long, double, float, date, time, unsignedInt,
   decimal, base64Binary, …
```

Komplexe Typen: Sequenzen

- Komplexe Typen sind aus anderen (einfachen und komplexen) Typen zusammengesetzt.

- Sequenzen:
  - Fixe Anordnung von Elementen mit verschiedenem Typ.
  - Multiplizität der Elemente kann definiert werden.

```
<complexType name="personType">
  <sequence>
    <element name="name" type="string"/>
    <element name="age" type="unsignedShort"
          minOccurs="0" maxOccurs="1"/>
    <element name="address" type="tns:addressType"
          maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

Komplexe Typen: all-Elemente/Attribute

- All-Elemente:
  - Anordnung von Elementen mit verschiedenem Typ, wobei Reihenfolge nicht vorgegeben wird.
  - Multiplizität: Elemente können höchstens einmal vorkommen.

```
<complexType name="addressType">
  <all>
    <element name="street" type="string"/>
    <element name="place" type="string" minOccurs="0"/>
    <element name="zip" type="unsignedShort"/>
  </all>
</complexType>
```

- Attribute:

```
<complexType name="personType">
  ...
  <attribute name="category" type="string" use="required"/>
</complexType>
```

Vererbung

- Erweiterung (*extension): Hinzufügen von Elementen zum Basistyp.*

```
<complexType name="studentType">
  <complexContent>
    <extension base="personType">
      <element name="id" type="StudentID">
    </extension>
  </complexContent>
</complexType>
```

- Einschränkung (*restriction): Modifikation bzw. Weglassen von Elementen des Basistyps.*

```
<simpleType name="ZipCode">
  <restriction base="int">
    minInclusive value="1000"
    maxExclusive value="10000"
  </restriction>
<simpleType>
```

Verbindung Schema/Schema-Instanz

- Durch globales Element wird Wurzelelement eines XML-Dokuments definiert.

```
<element name="person" type="tns:personType"/>
  <complexType name="personType">...</complexType>
```

- xsi:schemaLocation referenziert das Schema-Dokument im XML-Dokument.

```
<pns:person category="business"
    xmlns:pns="http://myCompany/person"
    xmlns:xsi= ="http://www.w3.org/2001/XMLSchema-instanc2"
    xsi:schemaLocation="http://myCompany/person person.xsd">
  <name>Huber</name>
  ...
</pns:person>
```

Web Service Protocol Stack

Und noch viele mehr,
z.b. WS-Policy, WS-Security,
WS-Addressing, WS-Routing,
WS-Coordination, WS-Transaction

Codierung der Nachrichten mit XML,
damit Gesprächspartner die Nachricht
lesen können

SOAP: Simple Object Access Protocol

- Merkmale
  - SOAP ist eine XML-Sprache mit einem XML-Schema.
  - SOAP-Nachrichten werden über Transportprotokolle übertragen (*tunneling): HTTP, SMTP, TCP/IP, …*
  - SOAP ist sprach- und plattformunabhängig.
  - SOAP ist unabhängig von Messaging-Protokoll:
    - synchron/asynchron,
    - unidirektional (one-way) bzw. bidirektional (request/response).
  - SOAP ist das Basisprotokoll für Web-Services.

Struktur

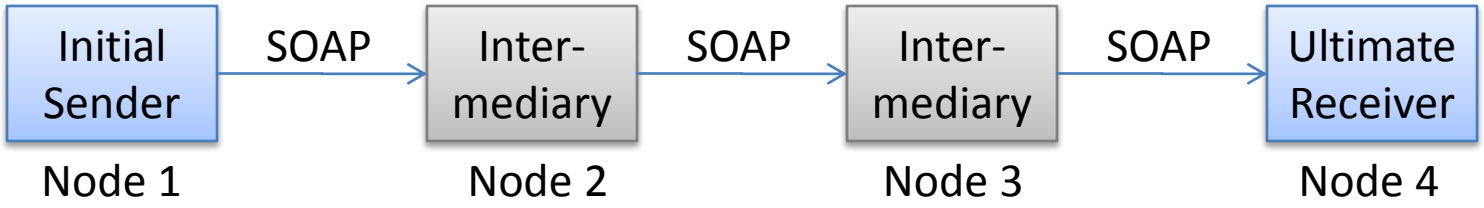
```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelop
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelop/">

    <soap:Header>
        ...
    </soap:Header>

    <soap:Body>
        ...
    </soap:Body>

</soap:Envelop>
```

Header (optional): Infos über die Nachricht
Security-Tokens, Transaktions-Informationen, Routing-Anweisungen.

Body: Nachricht im XML-Format

Header

- MessagePath



- Rollenzuordnung durch das *actor-Attribut*

```
<soap:Header>
 <ns:myMessage soap:actor="http://myRole" soap:mustUnderstand="1">
    ...
  </ns:myMessage>
</soap:Header>
```

- "Identifiziert" sich ein Knoten mit einer bestimmten Rolle, muss er die Nachricht verarbeiten.
- Zwischenknoten dürfen den Header verändern (Elemente löschen), aber nicht den Body.
- Anwendungsbeispiel: prüfen von digitalen Signaturen

Body

- wohl-geformtes XML-Dokument
- enthält Daten oder Parameter eines entfernten Methodenaufrufs.

4 Nachrichten-Modi*:*

messaging style / Encoding	Document	RPC
Literal	Document/Literal	RPC/Literal
Encoded	Document/Encoded	RPC/Encoded

Nachrichtenart

- Document/Literal:
  - Zur Übertragung von Daten.
  - Body enthält ein Fragment eines XML-Dokuments.

```
<soap:Body>
  <ns:person>
    <ns:name>Huber</ns:name>
    ...
  </ns:person>
</soap:Body>
```

- RPC/Literal:
  - Zur Darstellung entfernter Methodenaufrufe
  - Body enthält Methodennamen und Methodenparameter bzw. rückgabewerte und evtl. Fehlermeldungen

```
<soap:Body>
  <ns:getAge>
    <ns:name>Huber</ns:name>
  </ns:getAge>
</soap:Body>
```

```
<soap:Body>
  <ns:getAgeResponse>
    <result>29</result>
  </ns:getAge>
</soap:Body>
```

Nachrichtenart

- RPC/Encoded und Document/Encoded
  - Darstellung entfernter Methodenaufrufe
  - Abbildung von Datentypen auf XML-Schema.
  - Ermöglicht Repräsentierung von Objektgraphen.
  - Interoperabilitätsprobleme wegen vielfältiger Darstellungsmöglichkeiten; nicht in WS-I Basic Profile enthalten

```
<soap:Body>
  <ns:getGrades soap:encodingStyle="…/soap/encoding">
  <ns:id xsi:type="xsd:string>Mustermann</ns:id>
  </ns:getGrades>
</soap:Body>
```

```
<soap:Body xmlns:enc="…/soap/encoding">
  <ns:getGradesResponse soap:encodingStyle="…/soap/encoding">
    <enc:Array enc:arrayType="xsd:short[2]">
      <enc:short>1</enc:short>
      <enc:short>2</enc:short>
    </enc:Array>
  </ns:getGradesResponse>
</soap:Body>
```

SOAP Faults

- Fehler-Nachrichten (*soapfaults)* werden an den Vorgängerknoten geschickt.
- Fehlercodes:
  - soap:Client: Falsche Parameter.
  - soap:Server: Fehler auf Serverseite.
  - soap:Mustunderstand: Unbekanntes obligatorisches Header-Element.
  - soap:VersionMismatch: Falsche SOAP-Version.

```
<soap:Body>
  <soap:Fault>
    <faultcode>soap:Client</faultcode>
    <faultString>Invalid ID</faultString>
    <faultActor>http://myActor</faultActor>
    <detail>XML document fragment</detail>
  </soap:Fault>
</soap:Body>
```

Transportprotokoll: SOAP über HTTP

- SOAP ist unabhängig vom Transportprotokoll
- Am häufigsten wird aber HTTP verwendet
  - Vorteil: Selten Probleme mit Firewalls

```
POST /URLHTTP/1.1
Host: host-address
Content-Type: text/xml
Content-Length: nnn
SOAPAction: "URL/getAge"

<?xml version="1.0 …>
<soap:Envelop>
  <soap:Body>
    <ns:getAge>
      <ns:name>Muster</ns:name>
    </ns:getAge>
  </soap:Body>
</soap:Envelop>
```

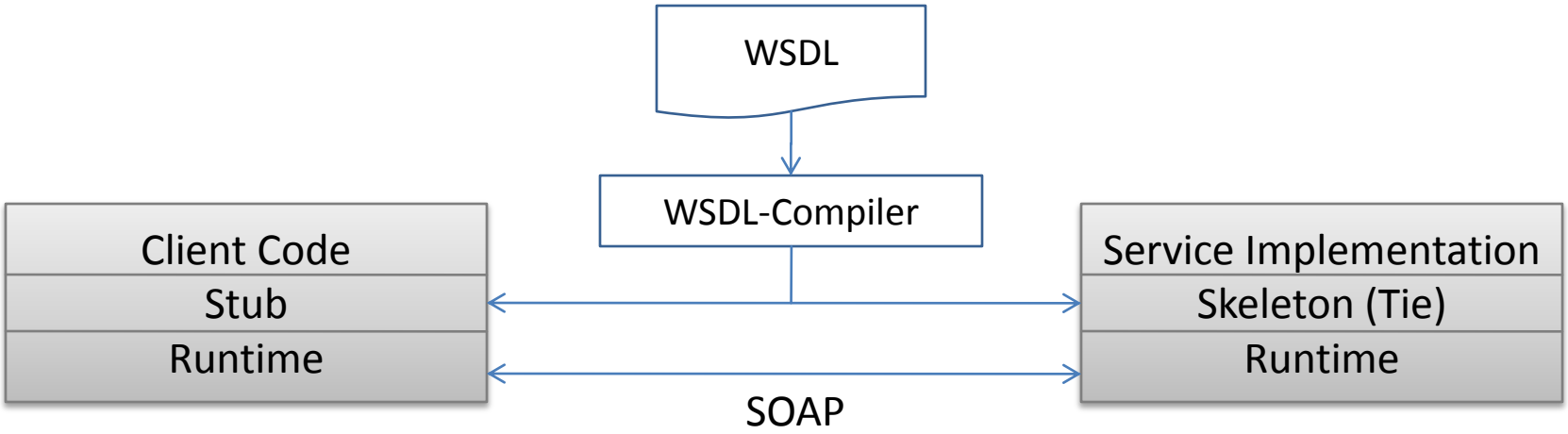
HTTP-Request

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnn

<?xml version="1.0 …>
<soap:Envelop>
  <soap:Body>
    <ns:getAgeResponse>
      <result>30</result>
    </ns:getAge>
   </soap:Body>
</soap:Envelop>
```

HTTP-Response

WSDL

- Ein WSDL-Dokument definiert für ein Web-Service:
  - das Interface (Methoden und Parameter),
  - das Nachrichten-Format (Document/Literal, RPC/Literal, …),
  - das zu verwendende Transportprotokoll (HTTP, SMTP, TCP/IP, …),
  - die Adresse (URL).
- Anwendung: Generierung von Tie- (Skeleton-)/Stub-Code:



WSDL

```
<definitions name=„MyWebService“
   targetNamespace=„http://MyCompany/MyWS“
   xmlns:tns=„http://MyCompany/MyWS“
   xmlns=„http://schmas.xmlsoap.org/wsdl“>

  <types> …</types>

  <message> …</message>

  <portType> …</portType>

  <binding> …</binding>

  <service> …</service>

</definitions>
```

Abstrakter Teil
- verwendet für die Interface-Beschreibung

Konkreter Teil
- enthält Deployment-Informationen

WSDL: types

- Definition von benutzerdefinierten einfachen und komplexen Typen.
- Typen werden für die Definition von Nachrichten verwendet

```
<types>
  <xsd:schema targetNamespace="http://MyCompany/MyWS">
    <xsd:complexType name="ArrayOfString">
      <xsd:sequence>
        <xsd:element name="string"type="xsd:string"
                     minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
</types>
```

WSDL: message

- Inhalt einer SOAP-Nachricht
- Für jede eingehende und jede ausgehende Nachricht wird eine message-Element

```
<message name="GetGradesRequest">
  <part name="studentID" type="xsd:string"/>
  <part name="year" type="xsd:int"/>
</message>

<message name="GetGradesResponse">
  <part name="grades" type="tns:ArrayOfInt"/>
</message>
```

WSDL: portType

- Interface eines Web-Services = Folge von Operationen (operation)
- Jede Operation besteht aus
  - einer ausgehenden Nachricht,
  - einer eingehenden Nachricht (optional) und
  - einer Fehlernachricht (optional):

```
<portTypename="StudentPort">
  <operation>
    <input name="id" message="tns:GetGradesRequest"/>
    <output name="result" message="tns:GetGradesResponse"/>
    <fault name="InvalidParams" message="tns:InvalidParams"/>
  </operation>
</portType>
```

WSDL: binding

- Binding definiert, wie die Interface-Methoden und -Parameter auf SOAP abgebildet und übertragen werden:
  - Nachrichtenart (style): RPC oder document,
  - Darstellungsform (encoding): literal oder encoded,
  - Transportprotokoll: HTTP, SMTP, …

```
<binding name="Student_Binding" type="tns:Student">
  <soapbind:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetGrades">
    <soapbind:operation soapAction="http://.../Get"/>
    <input name="GetGradesRequest">
      <soapbind:body use="encoded" namespace=".../MyWS"/>
    </input>
    <outputname="GetGradesResponse"> ... </output>
  </operation>
</binding>
```

WSDL: service

- Jedem service-Element können ein oder mehrere Ports (port) zugeordnet sein.
- Ein Port ordnet einer Bindung (binding) eine Internet-Adresse zu.

```
<service name="StudentService">
  <port name="StudentPort" binding="tns:Service_Binding">
    <soapbind:address location="http://.../Service"/>
  </port>
</service>
```

References

- W3C, The World Wide Web Consortium, http://www.w3.org/
- J. Heinzelreiter, Web Services, http://staff.fh-hagenberg.at//jheinzel/PRG5/Docs/Web-Services-Grundlagen.pdf