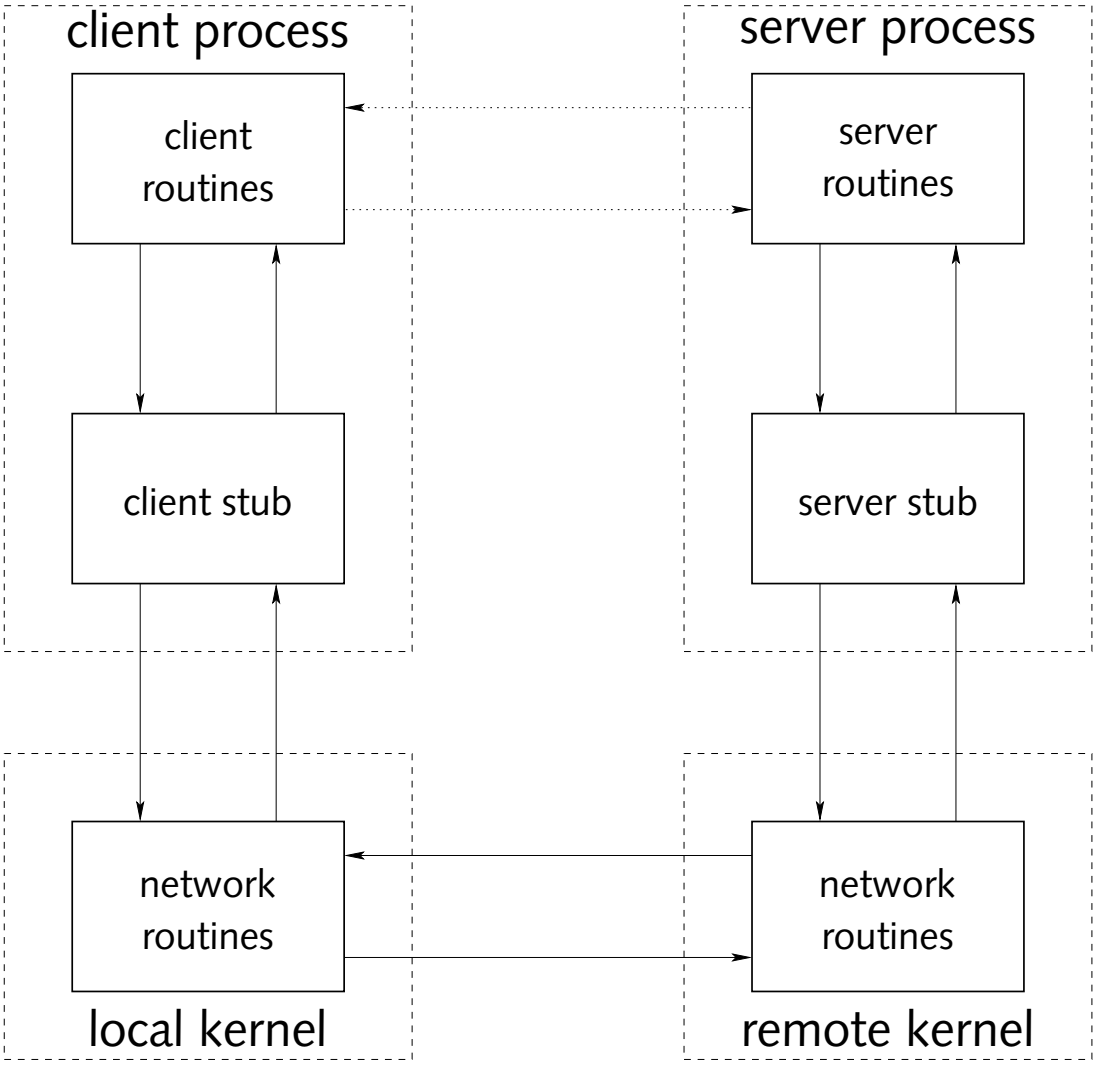
XML-RPC, SOAP und Web Services

Jörn Clausen
joern@TechFak.Uni-Bielefeld.DE

Übersicht

- Was ist RPC?
- Was hat XML mit RPC zu tun?
- Was sind XML-RPC und SOAP?
- Was sind Web Services?
- Wird das die Welt retten?

Remote Procedure Calls



Remote Procedure Calls, cont.

- Ziel: Illusion einer lokalen Funktion/Prozedur
- Probleme:
  - Parameterübergabe: call-by-value vs. call-by-reference
  - Bindung: (host,port), Folklore, directory service
  - Transport-Protokoll: verbindungslos vs. -orientiert, ...
  - Fehlerbehandlung: timeouts, „Verschwinden“ eines Teilnehmers
  - Aufruf-Semantik: idempotente Prozeduren
  - Darstellung der Daten: Kodierung, Größe, endianess
  - Performanz: (De-)Kodieren der Daten, Netzwerk, ...
  - Sicherheit: die üblichen ...

XML-basierte RPC-Lösungen

- gelöste und ungelöste Probleme:
  - Transport-Protokoll: häufig HTTP, aber auch andere Protokolle
  - Bindung: URIs, meistens URLs
  - Darstellung der Daten: XML
  - Binärdaten
  - Sicherheit: die übliche Sorglosigkeit . . .
- XML-RPC: Transport-Protokoll HTTP, grundlegende Datentypen
- SOAP: Protokoll-unabhängig, flexibleres Typ-Konzept (W3C Schemas)

XML-RPC

- entstanden 1998/99 im SOAP-Umfeld
- Spezifikation: 5 Seiten
- Implementierungen in vielen gängigen Sprachen
- Transport-Protokoll: HTTP
  - request-response-Schema paßt zu RPC
  - umfangreiche Code-Basis
- Datentypen: Integer, Double, String, Boolean, Base64, Array, Struct
- DateTime.iso8601: problematisch wegen Zeitzone und Kalender
- Fault: Struct mit Schlüsseln `faultCode` und `faultString`

einfache Datentypen

Integer	`<int>1234</int>`	4 Byte, Vorzeichen
Double	`<double>-12.345678</double>`	
String	`<string>Laurel &amp; Hardy</string>`	XML-Kodierung
Boolean	`<boolean>1</boolean>`	0 oder 1
Datum	`<dateTime.iso8601>20020604T14:54:34Z`	
Base64	`<base64>WE1M</base64>`	

komplexe Datentypen: Arrays

```
<array>
  <data>
    <value>
      <string>eenie</string>
    </value>
    <value>
      <string>meenie</string>
    </value>
    <value>
      <string>minie</string>
    </value>
    <value>
      <string>moe</string>
    </value>
  </data>
</array>
```

komplexe Datentypen: Structs

```
<struct>
  <member>
    <name>Italy</name>
    <value>
      <string>Trapattoni</string>
    </value>
  </member>
  <member>
    <name>England</name>
    <value>
      <string>Mortensen</string>
    </value>
  </member>
  <member>
    <name>Germany</name>
    <value>
      <string>Völler</string>
    </value>
  </member>
</struct>
```

alles auf einmal

```
<struct>
  <member>
    <name>Germany</name>
    <value>
      <array>
        <data>
          <value>
            <string>Kahn</string>
          </value>
          <value>
            <string>Ziege</string>
          </value>
        </data>
      </array>
    </value>
  </member>
  <member>
    <name>England</name>
    <value>
      <array>
        <data>
          <value>
            <string>Seaman</string>
          </value>
          <value>
            <string>Beckham</string>
          </value>
        </data>
      </array>
    </value>
  </member>
</struct>
```

Request

```
POST /dictionary
User-Agent: SupaDupa XML-RPC Client/0.1
Host: hobel.TechFak.Uni-Bielefeld.DE
Content-Type: text/xml
Content-Length: 201

<?xml version="1.0"?>
<methodCall>
  <methodName>translator.german2english</methodName>
  <params>
    <param>
      <value><string>Schmusedecke</string></value>
    </param>
  </params>
</methodCall>
```

Response

```
HTTP/1.1 200 OK
Connection: Close
Content-Length: 160
Content-Type: text/xml
Server: Langundbreit Dictionary Server/1.0

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>security blanket</string></value>
    </param>
  </params>
</methodResponse>
```

XML-RPC-Client mit Perl

```
#!/vol/perl-5.6/bin/perl

use RPC::XML::Client;

my $client = RPC::XML::Client->
           new('http://babelfish.org/dictionary');
my $response = $client->
           send_request('translator.german2english',
           'Schmusedecke');
print $response->value,"\n";
```

XML-RPC-Server mit Perl

```
#!/vol/perl-5.6/bin/perl

use RPC::XML::Server;

my $server = RPC::XML::Server->new(host => 'babelfish.org',
                                   port => '80');
my $translator = RPC::XML::Procedure->
           new({ name => 'translator.german2english',
                 code => \&g2e,
                 signature => [ 'string string' ]});
$server->add_method($translator);

$server->server_loop;
```

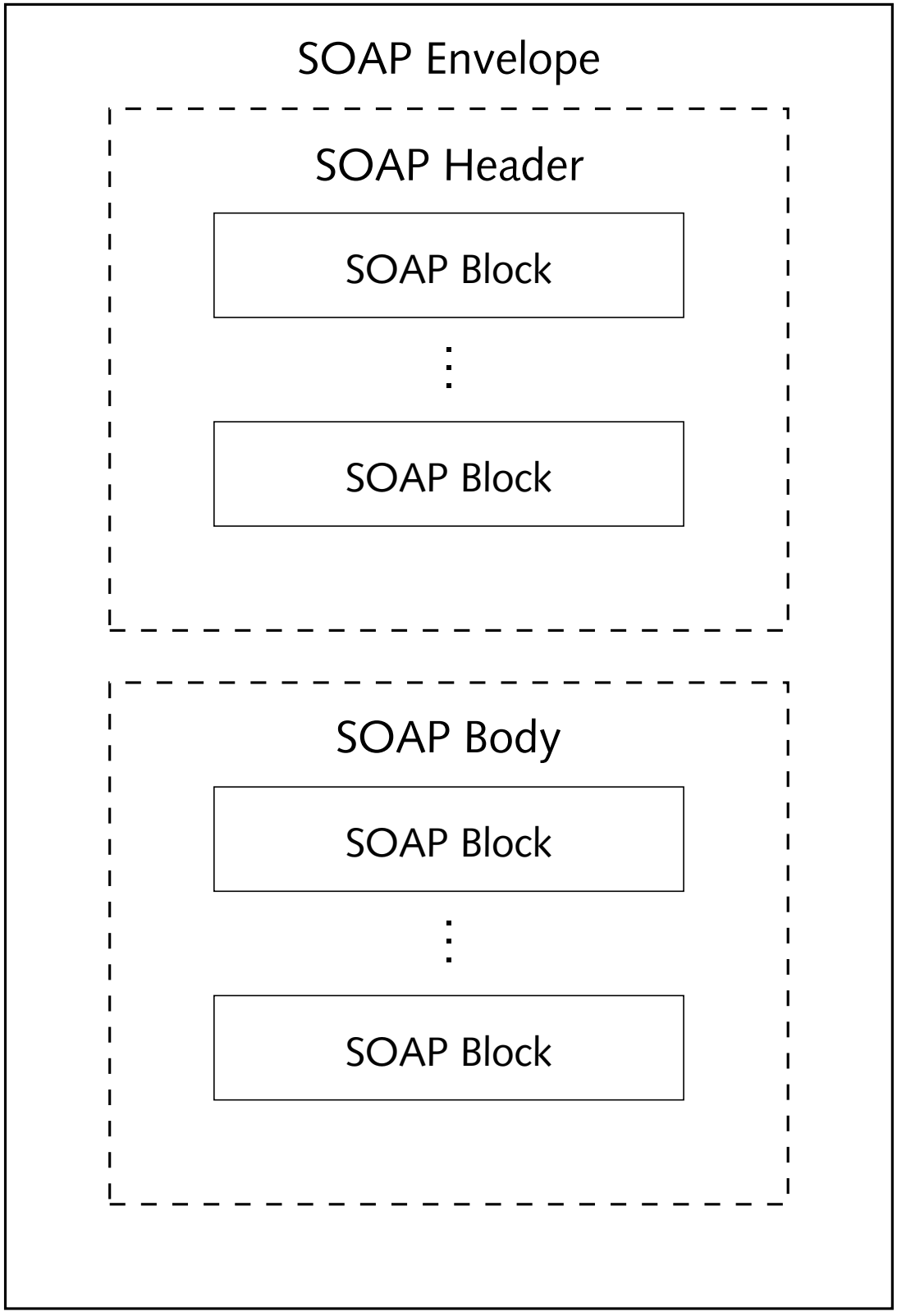
ein echter XML-RPC-Dienst

```
#!/vol/perl-5.6/bin/perl

use RPC::XML::Client;

my $cli = RPC::XML::Client->new('http://covers.wiw.org/RPC.php');
my $resp = $cli->send_request('covers.Covered','Bob Dylan');

format STDOUT =
@<<<<<<<<<<<<<<<<<<<<<<<<<   @<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$artist                      $song
.

foreach $node (@{$resp->value}) {
    $artist = $node->{artist};
    $song = $node->{song};
    write;
}
```

SOAP

- Working Draft des W3C
- früher „Simple Object Access Protocol“, heute kein Akronym mehr
- Messaging-Protokoll, RPC nur ein Spezialfall
- nicht an bestimmtes Transport-Protokoll gebunden
- Spezifikationen für HTTP, EMail, Jabber, BEEP
- bessere Typisierung, z.B. durch W3C XML Schema

SOAP Message



Request per SOAP

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2001/12/soap-envelope">
  <env:Header>
    <c:customer xmlns:c="http://micropayment.com/cust_account">
      <c:name>Joe User</c:name>
      <c:account>12345-6789</c:account>
    </c:customer>
  </env:Header>
  <env:Body>
    <t:translate xmlns:t="http://babelfish.org/translatorservice">
      <t:from>german</t:from>
      <t:to>english</t:to>
      <t:phrase>Schmusedecke</t:phrase>
      <t:phrase>großer Kürbis</t:phrase>
    </t:translate>
  </env:Body>
</env:Envelope>
```

Typisierung

```
<xs:element name="translate"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="phrase" type="xs:string"
                  minOccurs="1" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- wieder grundlegende Datentypen: Integer, String, Array, Struct, ...
- „Rollennamen“ für Parameter

Verarbeitung von SOAP Messages

- Message kann von mehreren SOAP-Prozessoren verarbeitet werden
- initial sender, intermediary, ultimate receiver
- intermediaries können header lesen und verändern

```
<env:Header>
  <c:customer xmlns:c="http://micropayment.com/cust_account"
              env:actor="http://babelfish.org/cashier">
    <c:name>Joe User</c:name>
    <c:account>12345-6789</c:account>
  </c:customer>
</env:Header>
```

Web Services

- verschiedene Definitionen, noch mehr Hype:
  - Übertragung und Verarbeitung von XML
  - Übertragung von XML per HTTP
  - Übertragung von SOAP Messages per HTTP
  - HTTP + XML + SOAP + WSDL + UDDI
- .NET (Microsoft), ONE (Sun), ebXML, UN/CEFACT, . . .
- Ziele:
  - abgeschlossene, interoperable Dienste
  - alles in XML
  - Auffindbarkeit (discoverabilty)

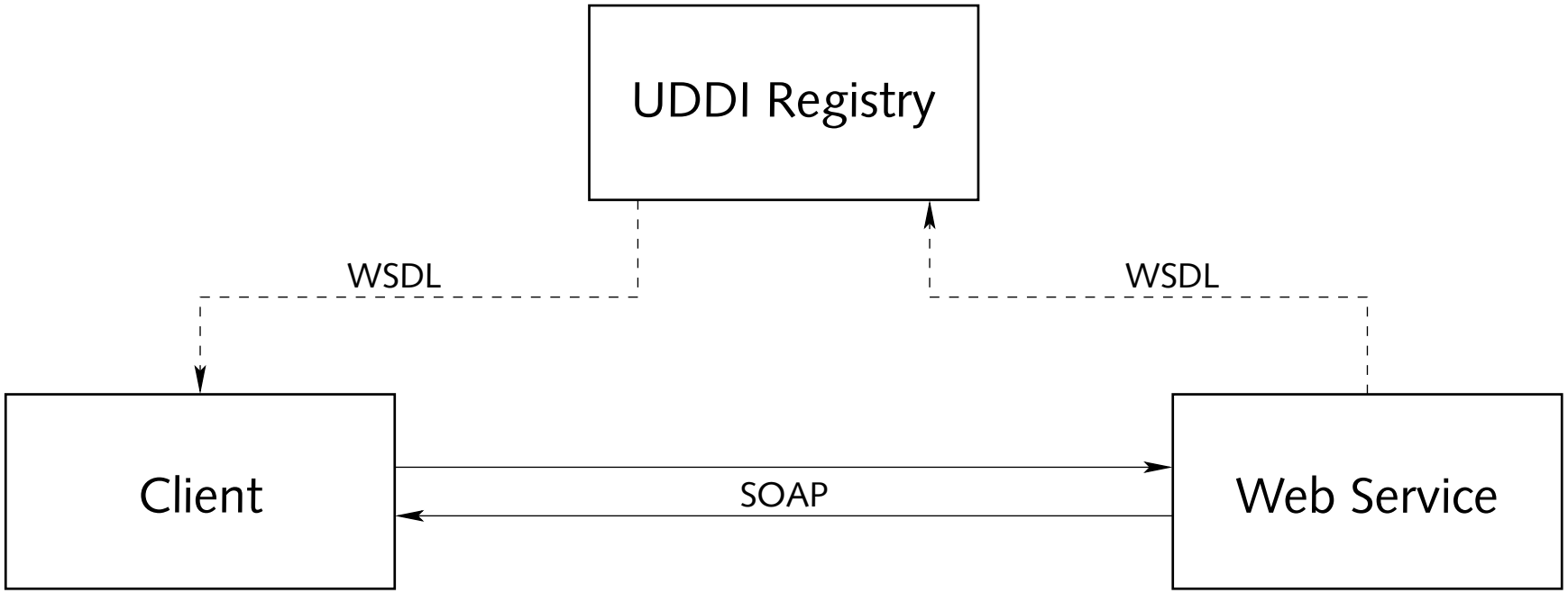
UDDI

- Universal Description, Discovery and Integration
- Verzeichnisdienst(e) für Web Services:
  - white pages: Adressen, Ansprechpartner
  - yellow pages: Geschäftsbereiche, Dienstleistungen
  - green pages: technische Spezifikationen der Dienste
- `http://www.uddi.org`

WSDL

- Web Services Description Language
- beschreibt einen Web Service
  - Datentypen
  - Methoden/Prozeduren
  - Bindung
- WSDL-Beschreibung kann per UDDI abgefragt werden
- automatische Erzeugung von Code-Fragmenten

Vision Web Services



Diskussion

- XML zur Datenrepräsentation?
  - $+$ leicht zu verstehen/debuggen
  - $-$ Binärdaten
  - $\pm$ Aufblähung der Daten
- HTTP als Transport-Protokoll?
  - $+$ große Codebasis
  - $-$ nicht allgemein genug?
  - $\pm$ Proxies und Firewalls
- Alles schon mal da gewesen?
- Interoperabilität? Bei den Mitspielern??
- Wird das die Welt retten?