3-schichtige Informationssystem-Architektur

- *plattformunabhängig*
- *beliebige Endgeräte*
- *Client als Applikation & Applet*

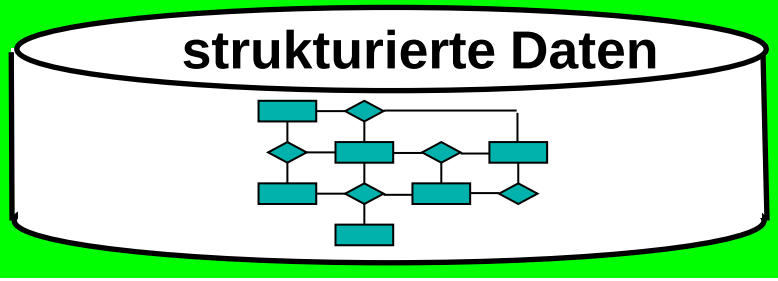
XML über SOAP

Standard

- *plattformunabhängig*
- *objektorientierte Architektur*
- *multiuserfähig (Threads)*

WebServer: WebServices

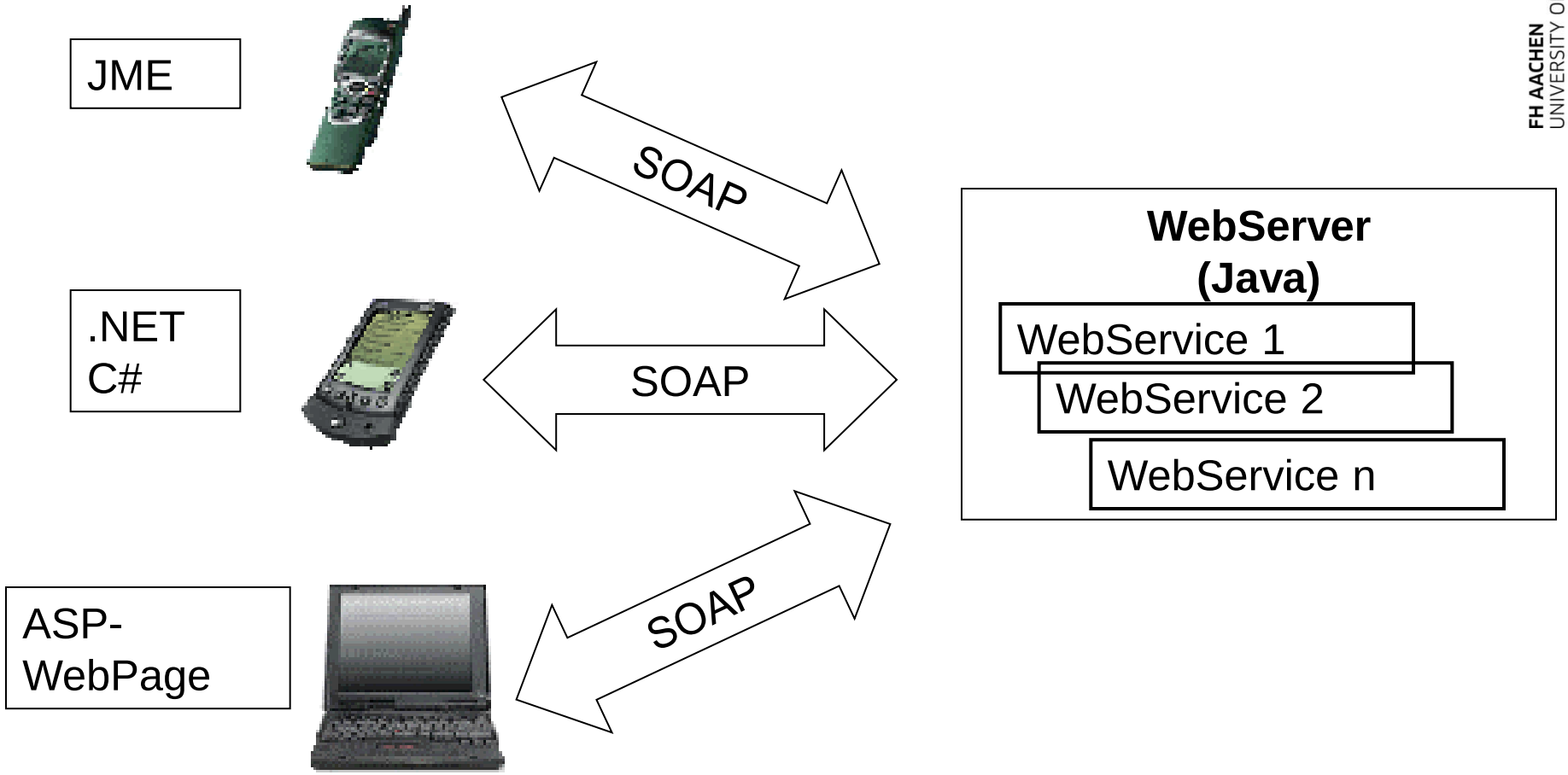
hier einfach Dateisystem des PCs bzw. DB-Anschluss



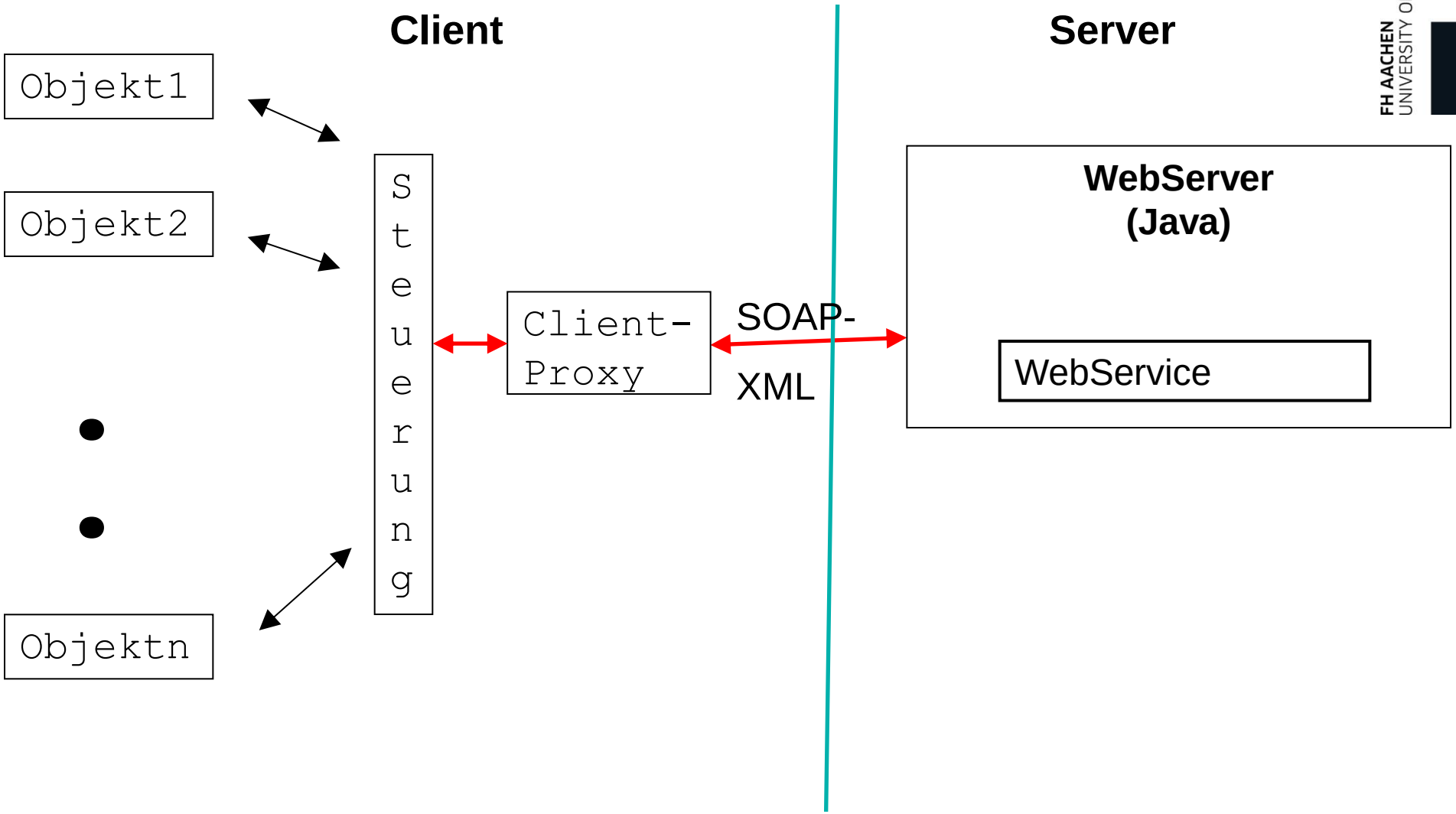
Warum WebServices?

- **Nachrichtenversand standardisiert**
  - XML (textuell)
  - SOAP
- **Interoperabilität zwischen**
  - Hardware
  - Betriebssystemen
  - Programmiersprachen
- **Zugriff über http (Port 80)**
  - keine Firewallproblematik
  - Schnittstellenbeschreibungen über Browser abrufbar
- **standardisiertes Auffinden von WebServices in UDDI**

WebService-Kommunikation



Vergleich zu bisheriger Architektur



Änderungen

bisher	jetzt
Dienstanbieter Java-Objekt	**WebService** läuft im WebServer
Server-Orb Socket-Protokoll -> Dienstaufruf Ergebnis -> Socket-Protokoll	**WebServer (automatisch)** SOAP-Nachricht -> Dienstaufruf Ergebnis -> SOAP-Nachricht
Client-Orb Dienstaufruf -> Socket-Protokoll Socket-Protokoll -> Ergebnis	**Client-Proxy (wird erzeugt)** Dienstaufruf -> SOAP-Nachricht SOAP-Nachricht -> Ergebnis
Socket-Verbindung auf Java bezogen	**SOAP-Verbindung** plattform-/sprachen-unabhängig

WSDL

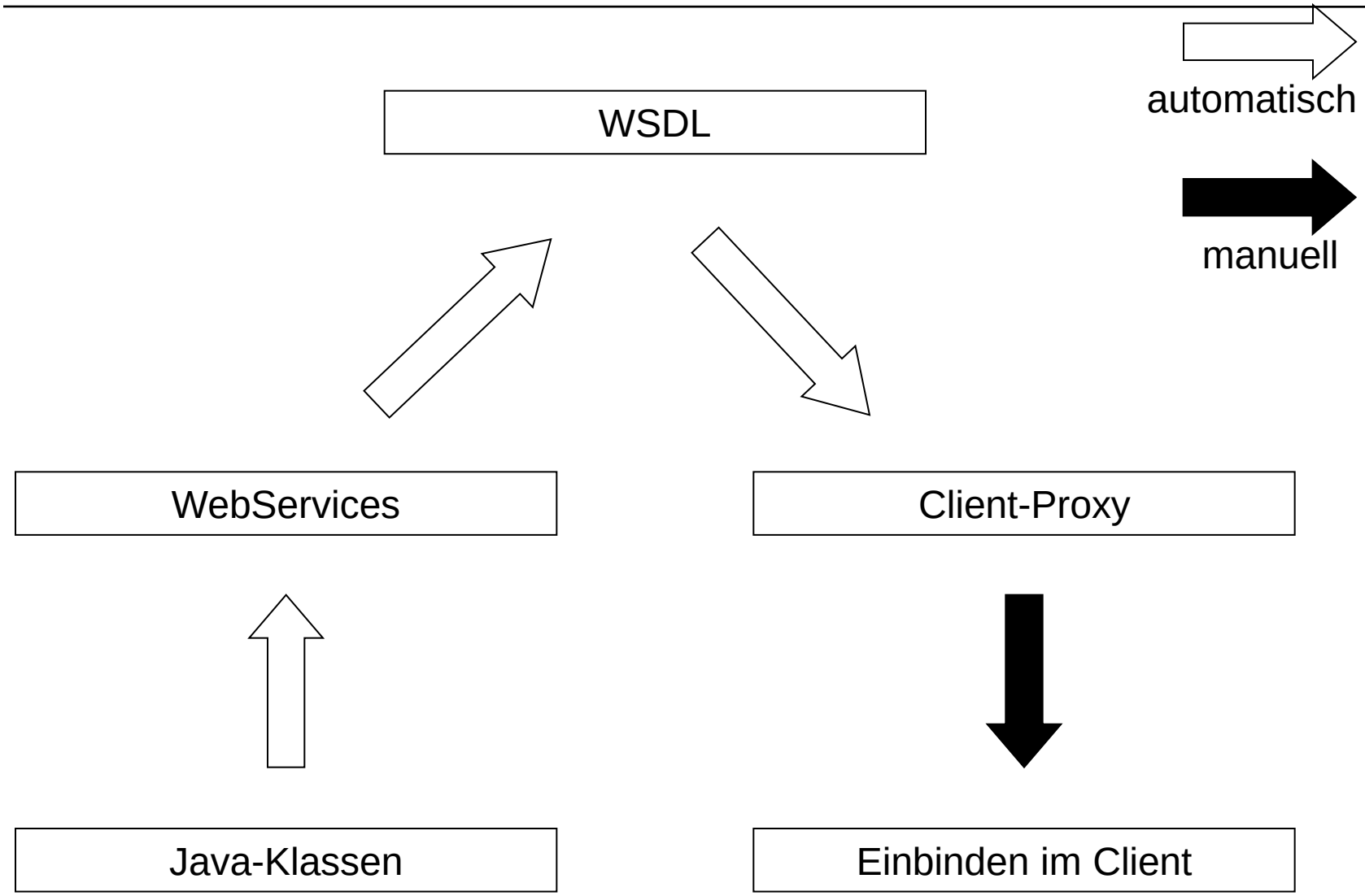
- **bisher: Schnittstelle als Java-Interface oder –Klasse**
- **jetzt: Plattformunabhängigkeit**
  - -> plattformunabhängige Schnittstellendefinitionen
  - -> eigene Sprache für Schnittstellendefinitionen
  - -> Web Service Description Language WSDL
- **analog zu Interface Definition Language IDL in CORBA**

WebService-Entwicklung

- **Contract first (Top-Down)**
  - CORBA-Ansatz
  - zuerst WSDL definieren
  - daraus Server-Stub und Client-Proxy erzeugen
  - **Problem:** WSDL-Definition von Hand sehr schwierig s.o.

- **Code first (Bottom-Up)**
  - zuerst Implementierung des Services
  - daraus automatische Erzeugung der WSDL
  - **Problem:** achte auf Interoperabilität der verwendeten Datentypen

Hier: Mischen der beiden Ansätze



Aufbau eines WSDL-Dokuments

- **XML-Dokument**
- **anhand des Dienstes**

```
String hello(String name);
```

- **WSDL-Dokument automatisch erzeugt**
- **pro Dienst: je eine *Input-* und *Output-Message***
- **Zusammenfassung von mehreren Diensten in *Porttypes***
- **pro Porttype: ein *Binding***
- **Binding legt fest:**
  - wie SOAP-Message erstellt
  - und Parameter kodiert
- **Lokalisierung des Service durch *URL* im Port**

Definitions

```
<wsdl:definitions targetNamespace="http://service"
      xmlns:apachesoap="http://xml.apache.org/xml-soap"
      xmlns:impl="http://service"
      xmlns:intf="http://service"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

- **Root-Element definitions**
- **Definition der Name-Spaces definitions**

Types

```
<wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace= .. >
   <element name="hello">
    <complexType>
     <sequence>
      <element name="name" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="helloResponse">
    <complexType>
     <sequence>
      <element name="helloReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>
```

Dienst

Parameter

Return-Typ

- **XML-Schemata für komplexe Datentypdefinitionen**

Message

```
<wsdl:message name="helloRequest">
  <wsdl:part element="impl:hello"name="parameters"/>
</wsdl:message>
<wsdl:message name="helloResponse">
  <wsdl:part element="impl:helloResponse„
                                    name="parameters"/>
</wsdl:message>
```

- **Message-Beschreibungen**
- **nicht festgelegt, ob Parameter oder Return-Wert**
- **Messages bisher noch unabhängig von Methoden**

PortType

```
<wsdl:portType name="HelloWorld">
  <wsdl:operation name="hello">
    <wsdl:input message="impl:helloRequest"
                                name="helloRequest"/>
    <wsdl:output message="impl:helloResponse"
                                name="helloResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

- **Zusammenfassung der Messages zu Schnittstelle**
- **input: Client -> Server, output: Server -> Client**
- **Fault-Messages bei Auftreten eines Fehlers**

Binding

```
<wsdl:binding name="HelloWorldSoapBinding"
type="impl:HelloWorld">
  <wsdlsoap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="hello">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="helloRequest">
          <wsdlsoap:body use="literal"/>
       </wsdl:input>
       <wsdl:output name="helloResponse">
          <wsdlsoap:body use="literal"/>
       </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

- **Beschreibung des Kommunikationsprotokolls für PortType-Element**

Service

```
<wsdl:service name="HelloWorldService">
  <wsdl:port binding="impl:HelloWorldSoapBinding"
                                        name="HelloWorld">
    <wsdlsoap:address location="http://localhost:
      8080/HelloWorldService/services/HelloWorld"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>  //Gesamt-Abschluss-Statement
```

- **Zusammenfassung von Port-Elementen**
- **kann mehrere Binding-Elemente beinhalten**
- **beinhaltet Kommunikationsendpunkte als URL**