Module 4

Web Services, RSS, Mashups

Recap

- Module 1: Why XML?
  - communication data, meta-data, documents
  - more flexibility
- Module 2: What is XML?
  - Basics: Namespaces, DTDs
- Module 3: What is XML (ctd.)?
  - XML Schema
- Module 4: XML for communication?
  - Web Services, RSS / ATOM, Mashups

Agenda

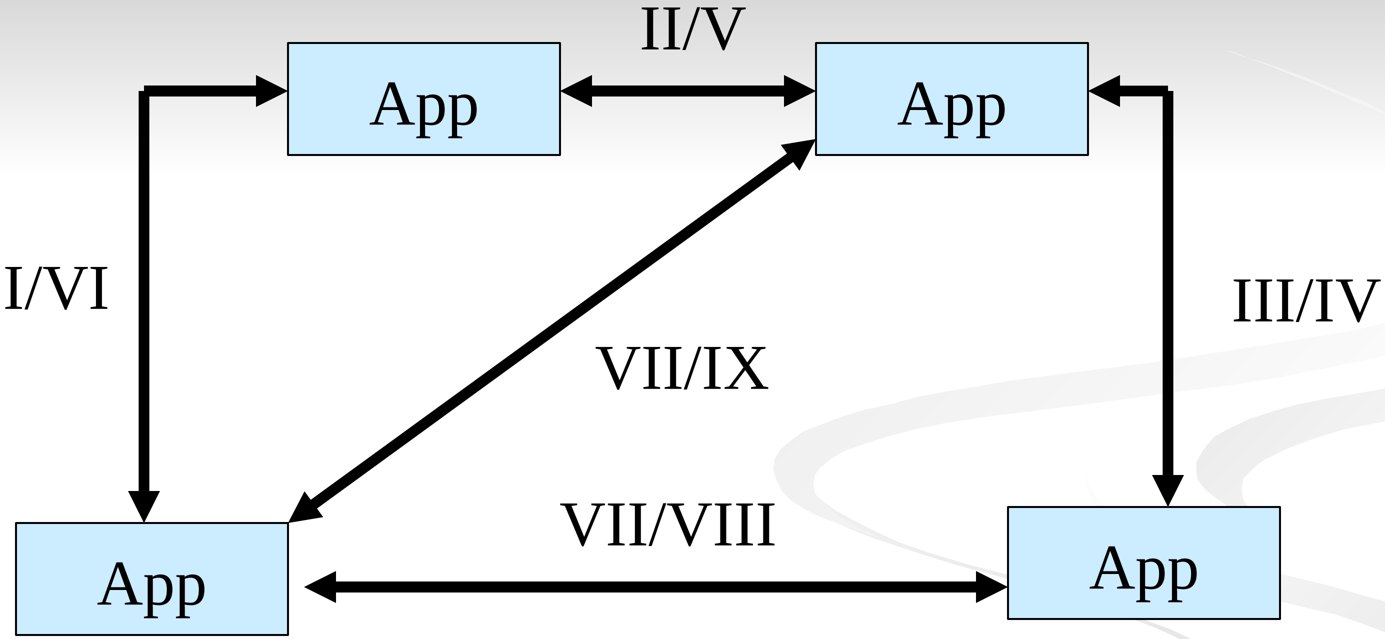
- Web Services
  - Definition
  - SOAP
  - WSDL
  - UDDI
- RSS / ATOM
- MashUps
  - (Demo)

Why Web Services?

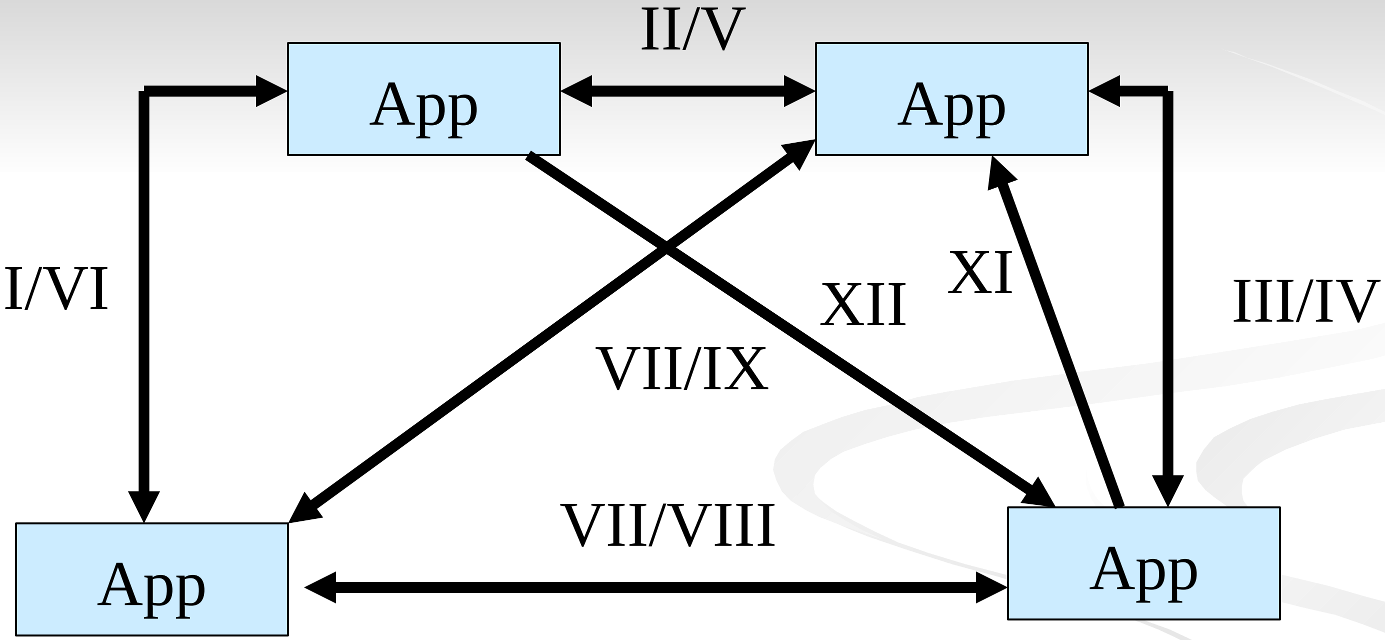
- Automization of Processes
  - Enterprise Application Integration (EAI)
  - Workflow Management
- Data Integration
  - Enterprise Information Integration (EII) (Connectivity, Global Data Model)
  - Portals

Integration, Integration, Integration

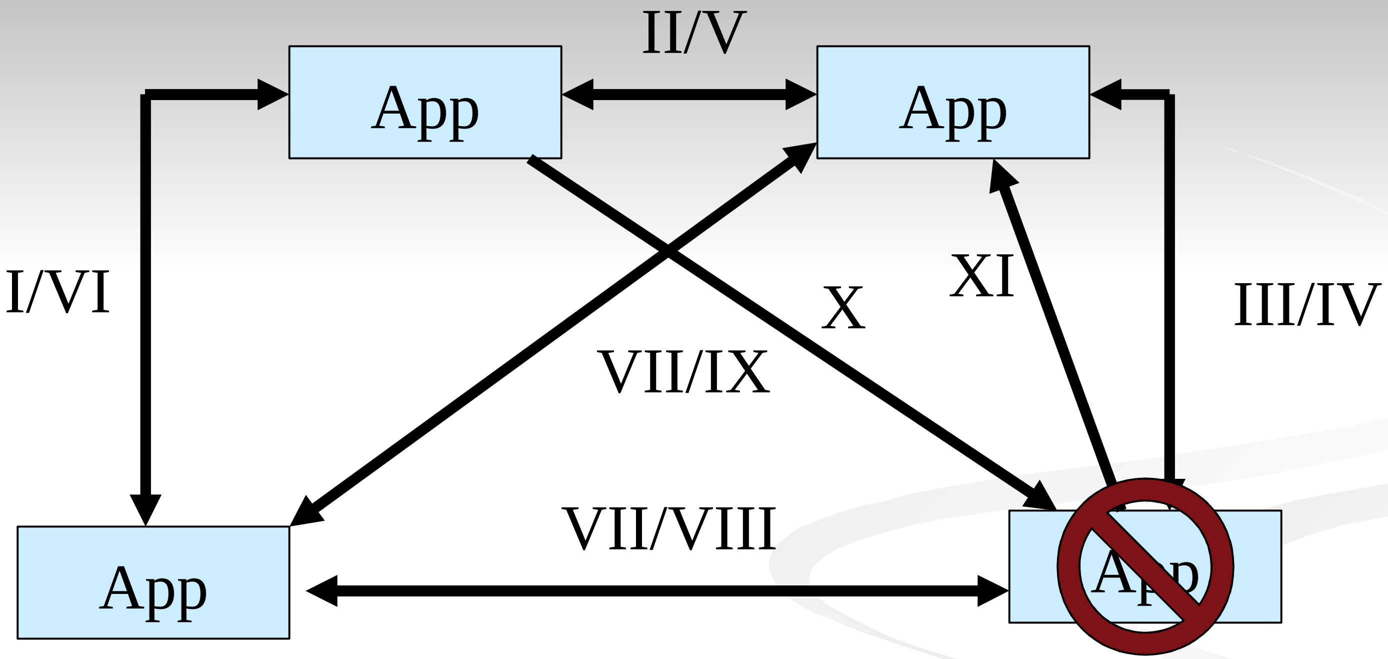
Application Integration



Application Integration



Application Integration

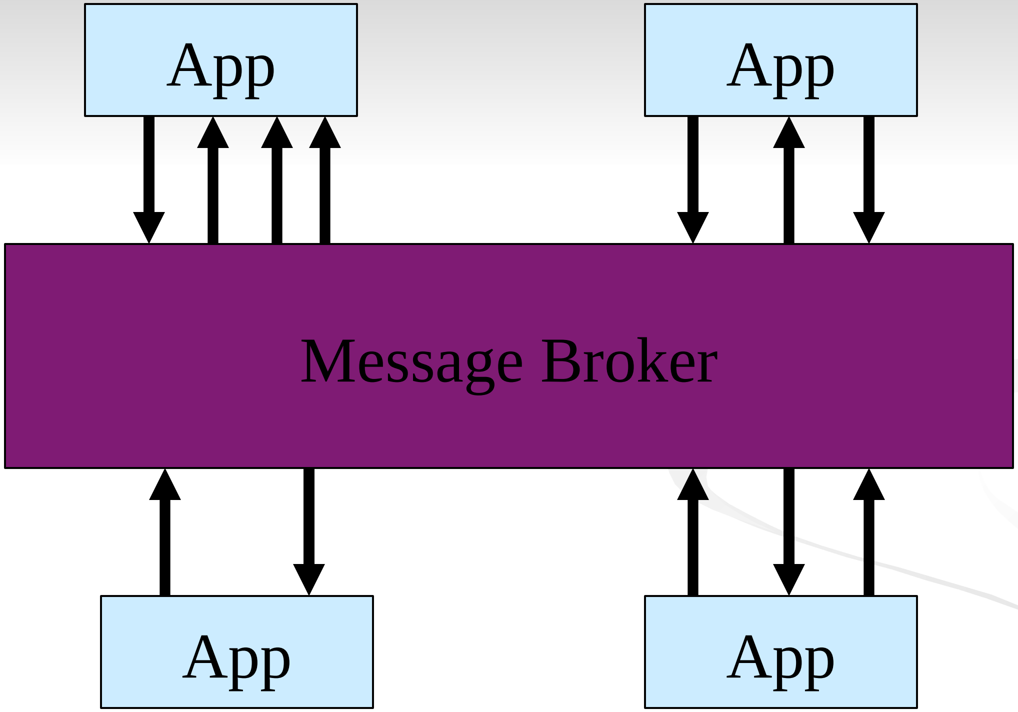


- What impact do delays have?
- Who is affected by a change in one interface?
- How can this process be optimized?
- What about humans? How to exploit a Grid of machines?

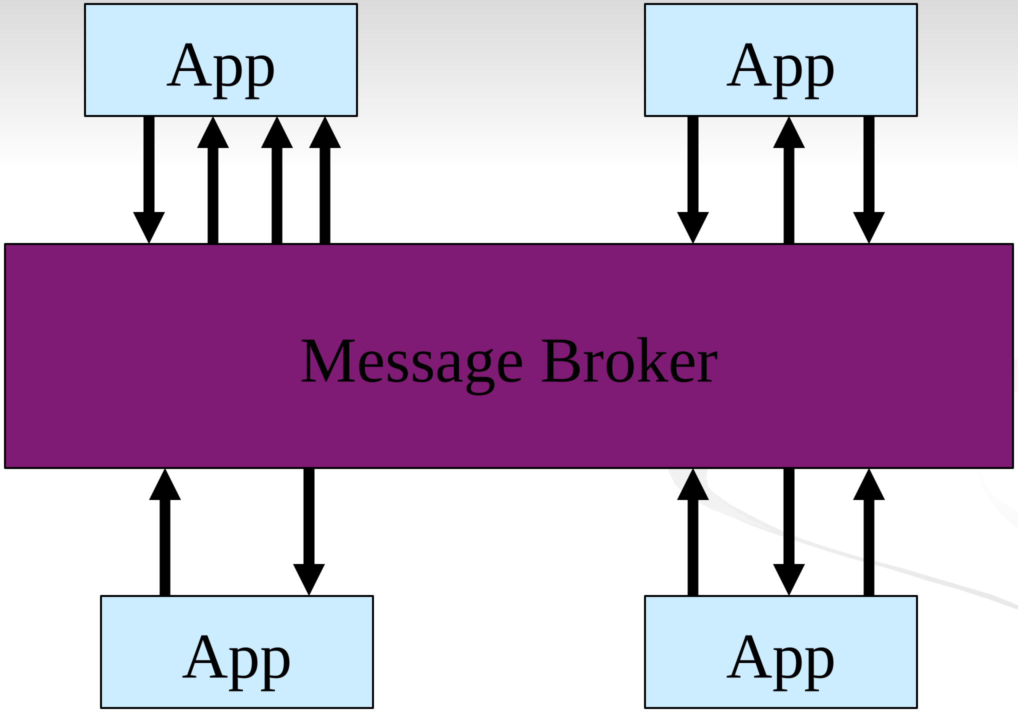
Issues: Distributed Computing

- Platform Dependency
- Management and Optimization
  - What do 99,99% availability mean?
  - How can I guarantee 3 seconds response time?
  - Who owns the Log? Who owns the context?
  - Load Balancing, Caching, Replication?
- Change a Process: All or Nothing
  - Whole system fails when a component is upgraded
  - Versioning, Schema Evolution while process runs

Loose Coupling of Apps

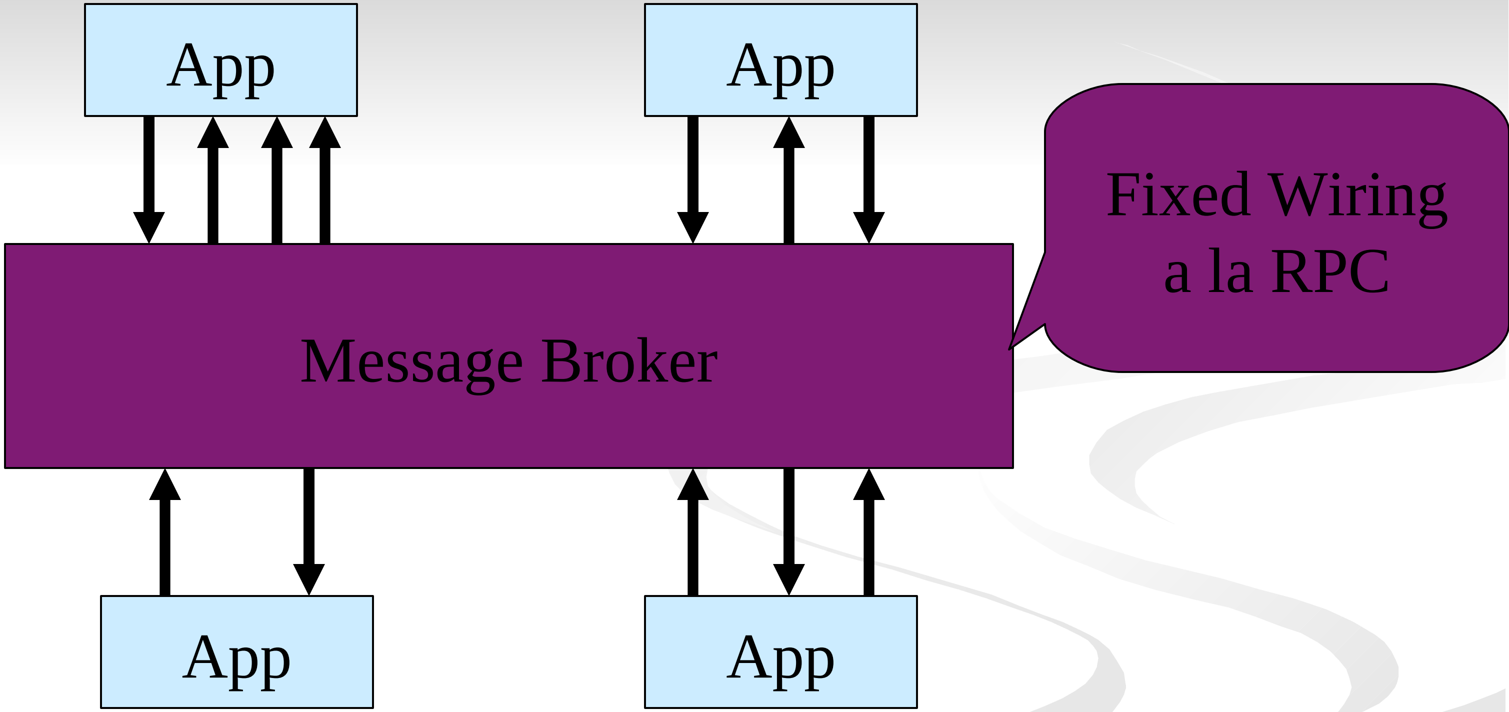


Loose Coupling of Apps (CORBA)



Good old CORBA!!!

Loose Coupling of Apps (CORBA)

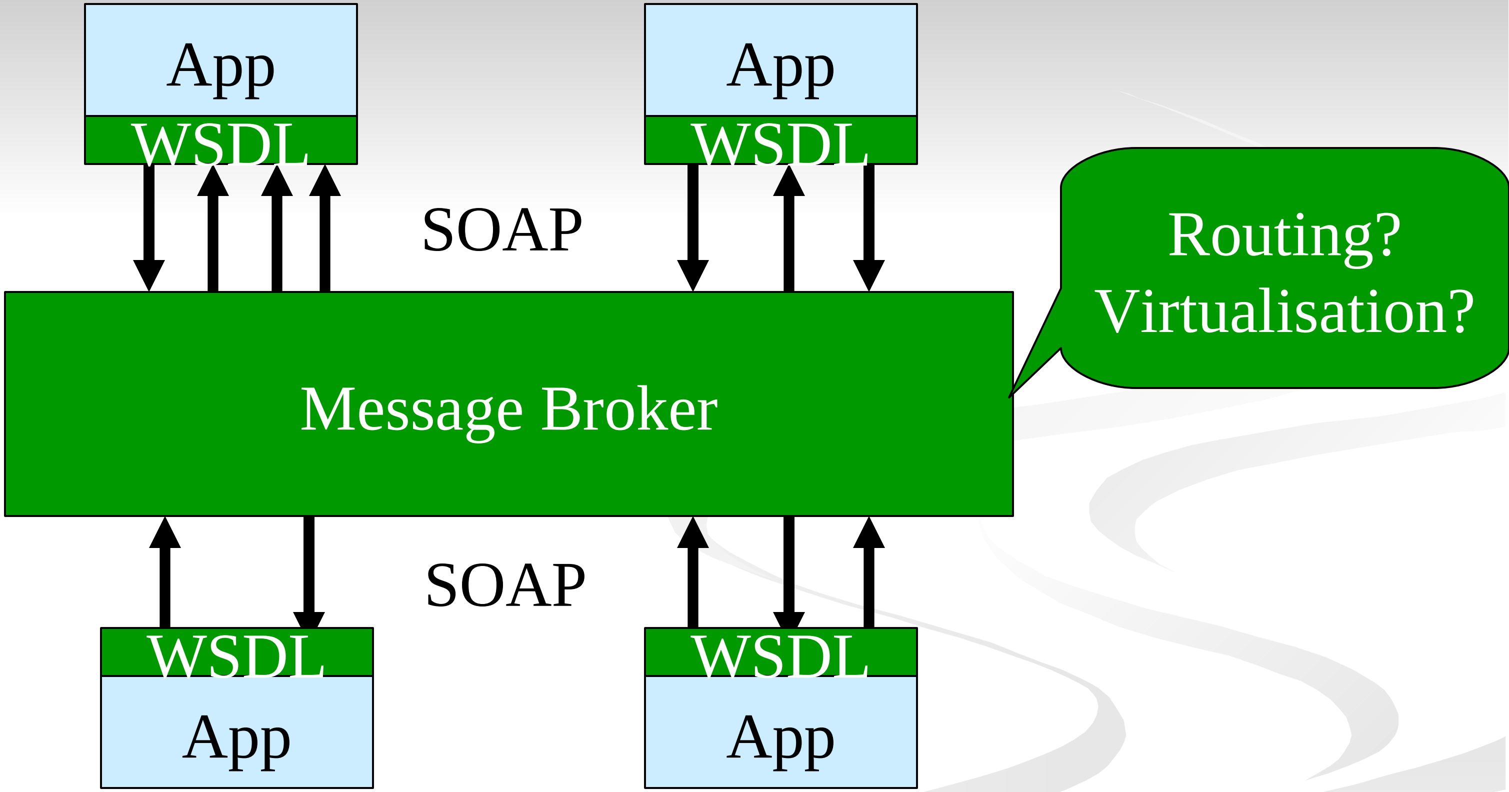


Good old CORBA!!!

Evaluation of CORBA

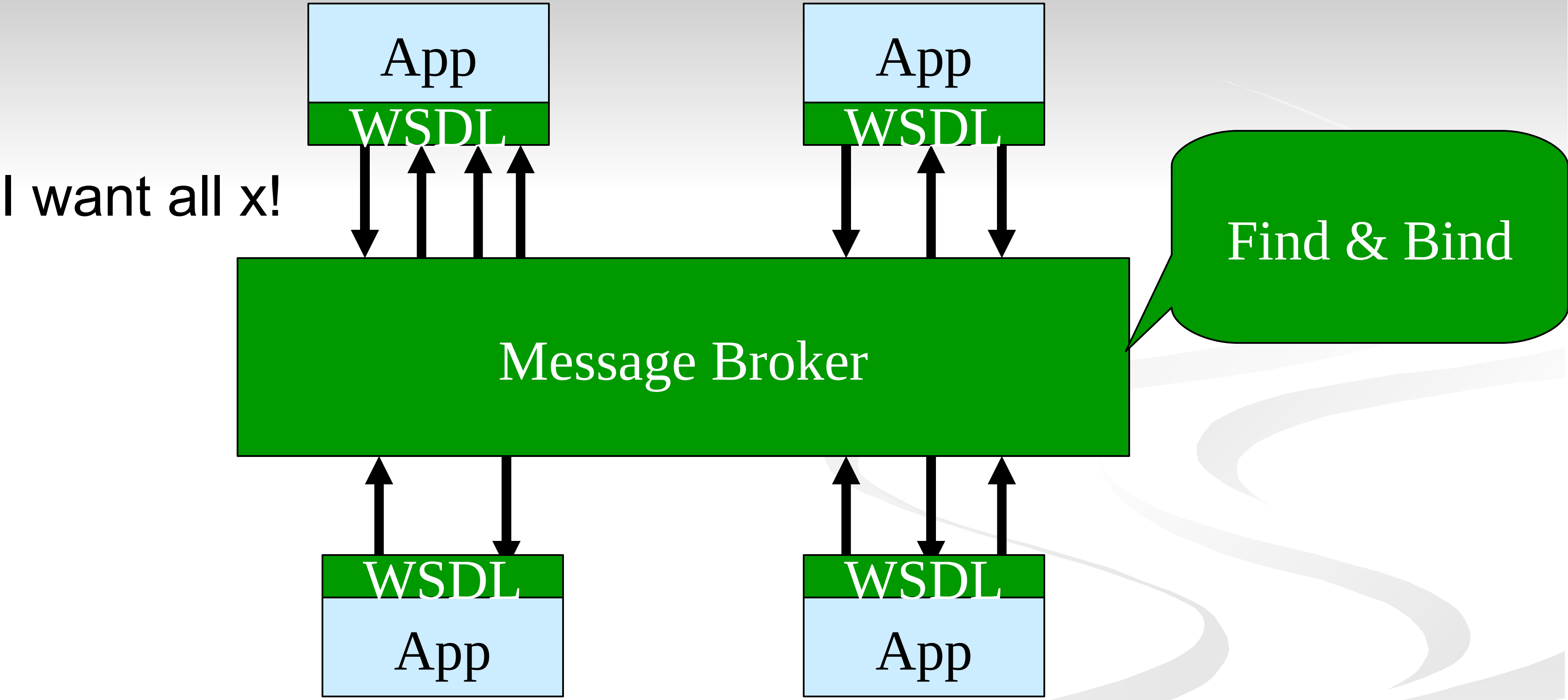
- **Platform Independence (okay)**
- Management and Optimization (poor)
- Dynamic Change of a Process (poor)

Loose Coupling of Apps (Web Services)



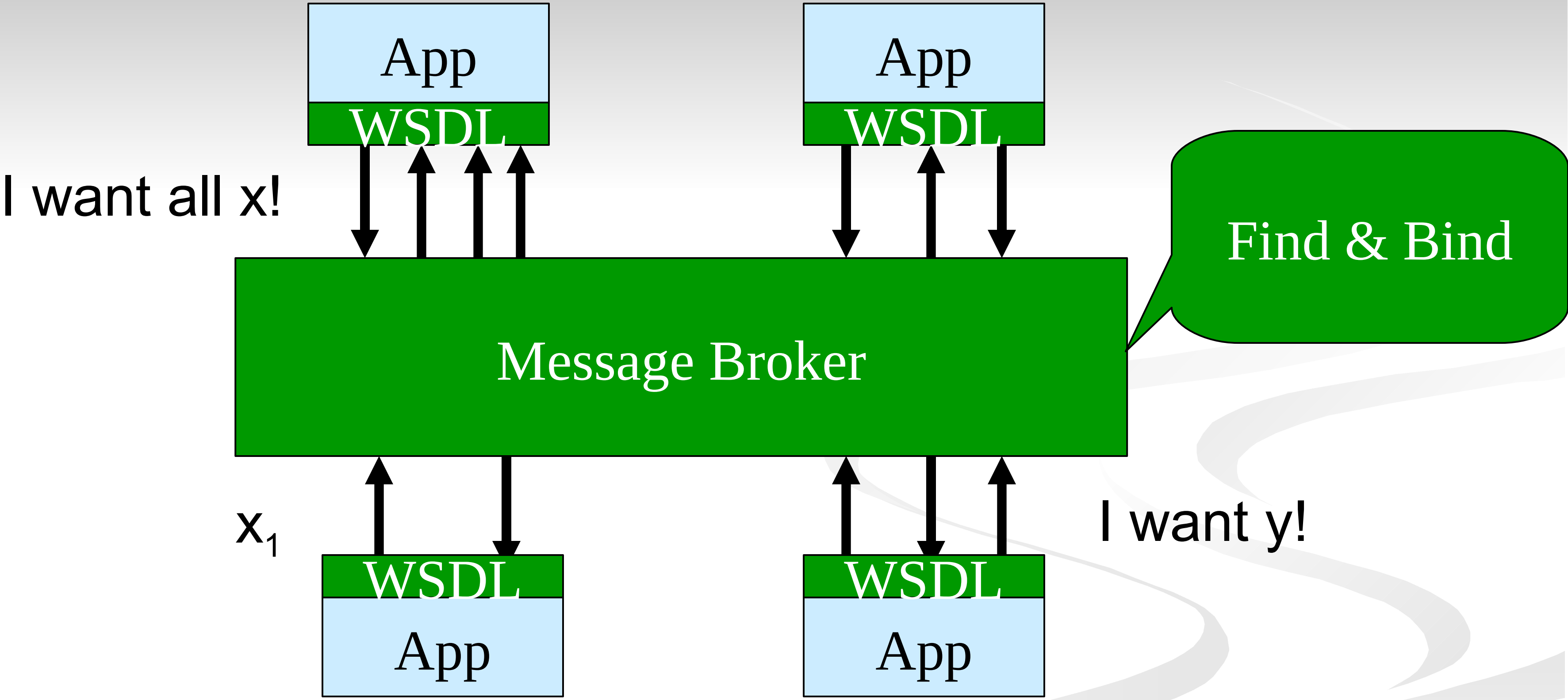
Web Services

Virtualisation of Apps



Web Services

Virtualisation of Apps



App

WSDL

App

WSDL

I want all x!

Find & Bind

Message Broker

x_1

WSDL

App

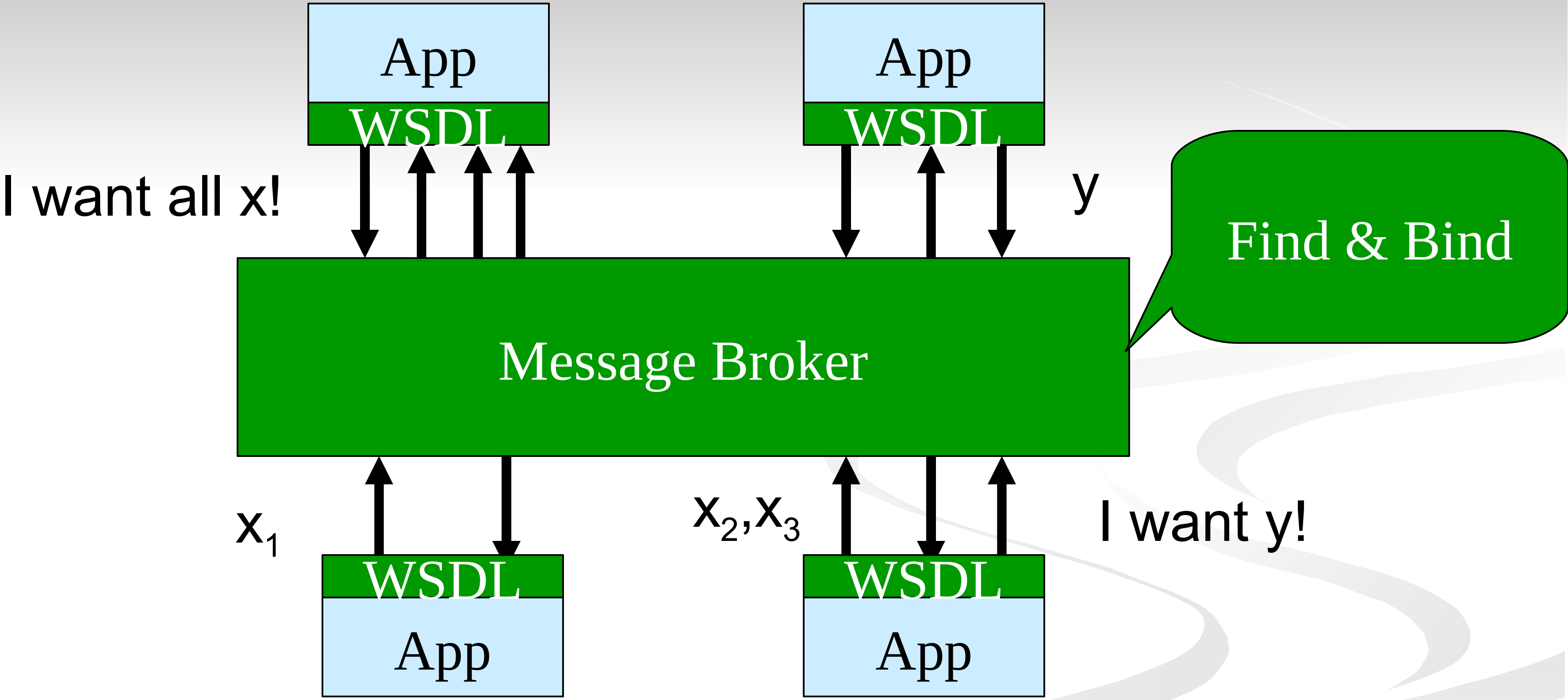
WSDL

App

I want y!

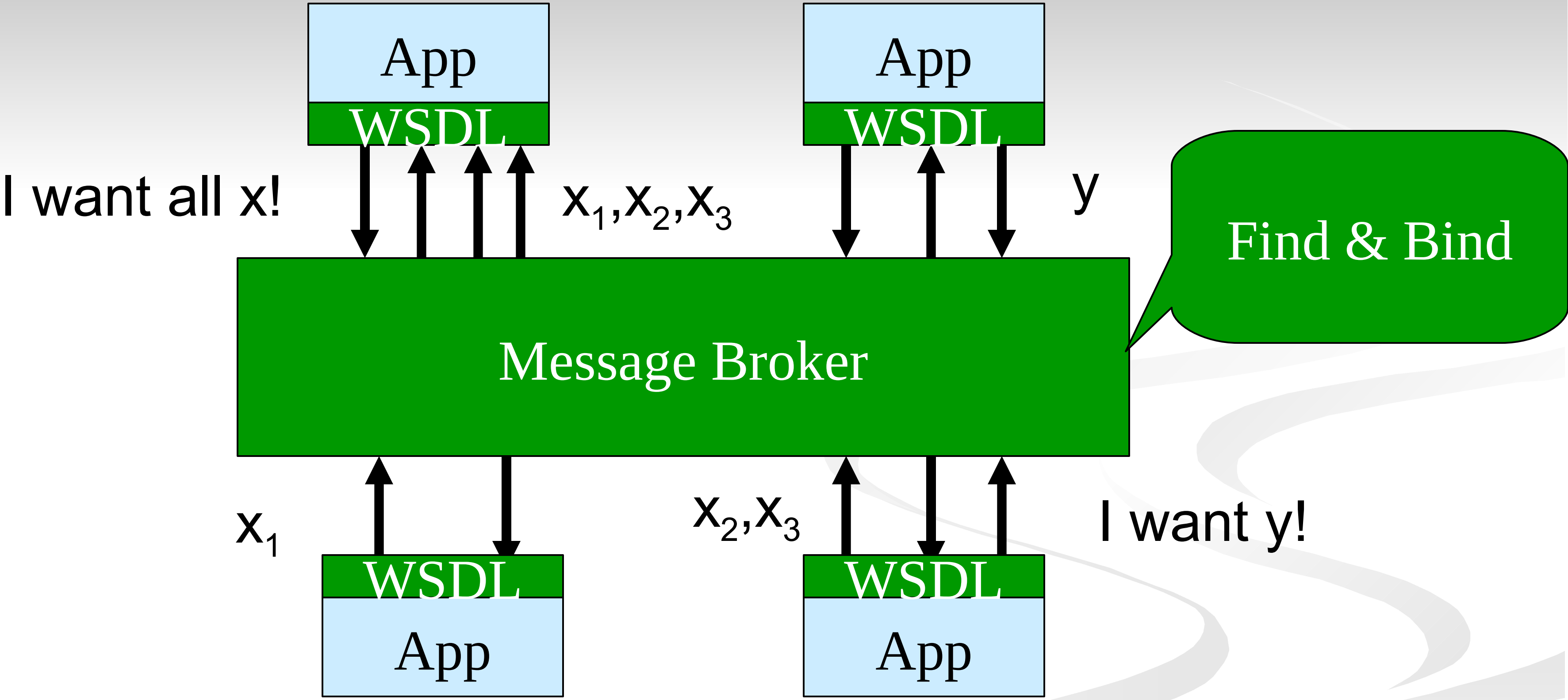
Web Services

Virtualisation of Apps



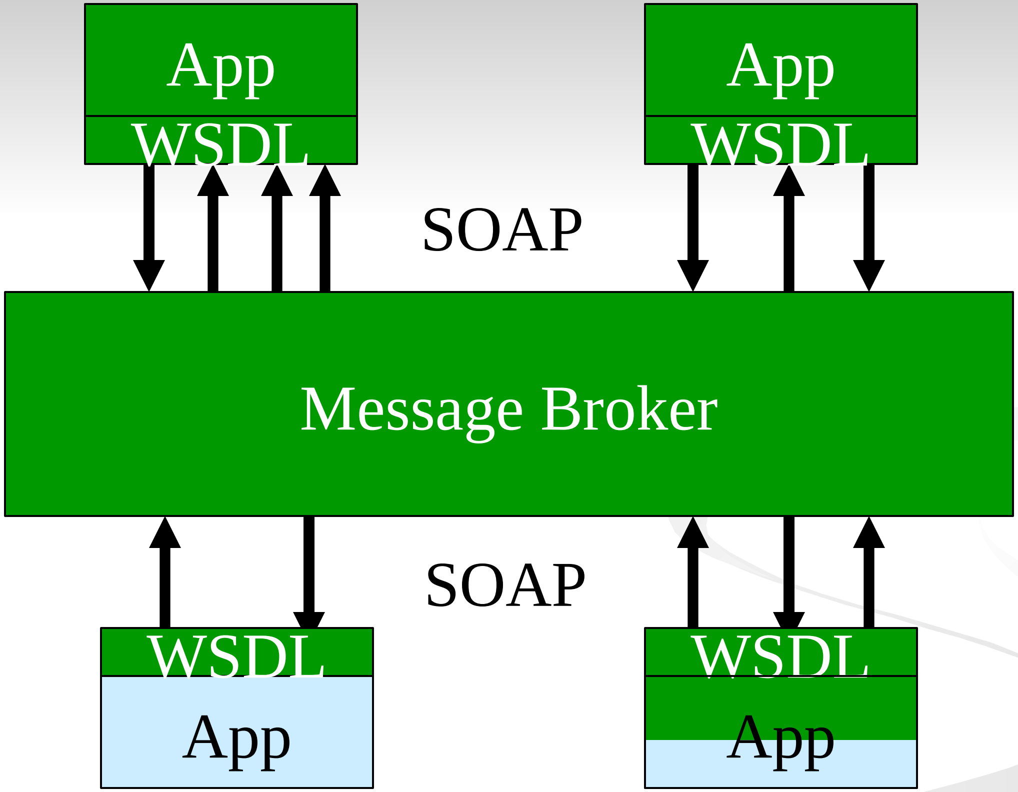
Web Services

Virtualisation of Apps



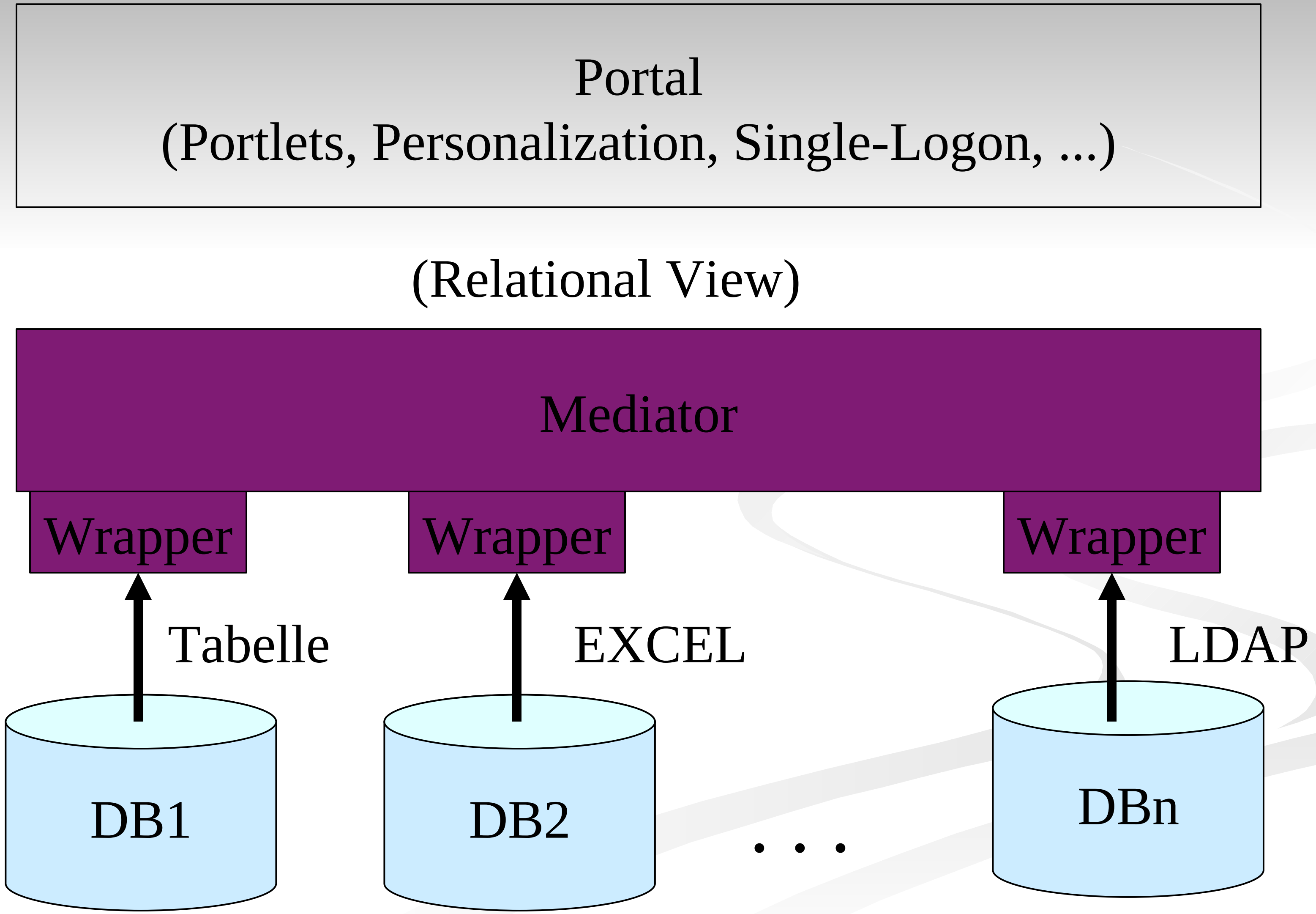
Web Services

How deep does XML go?

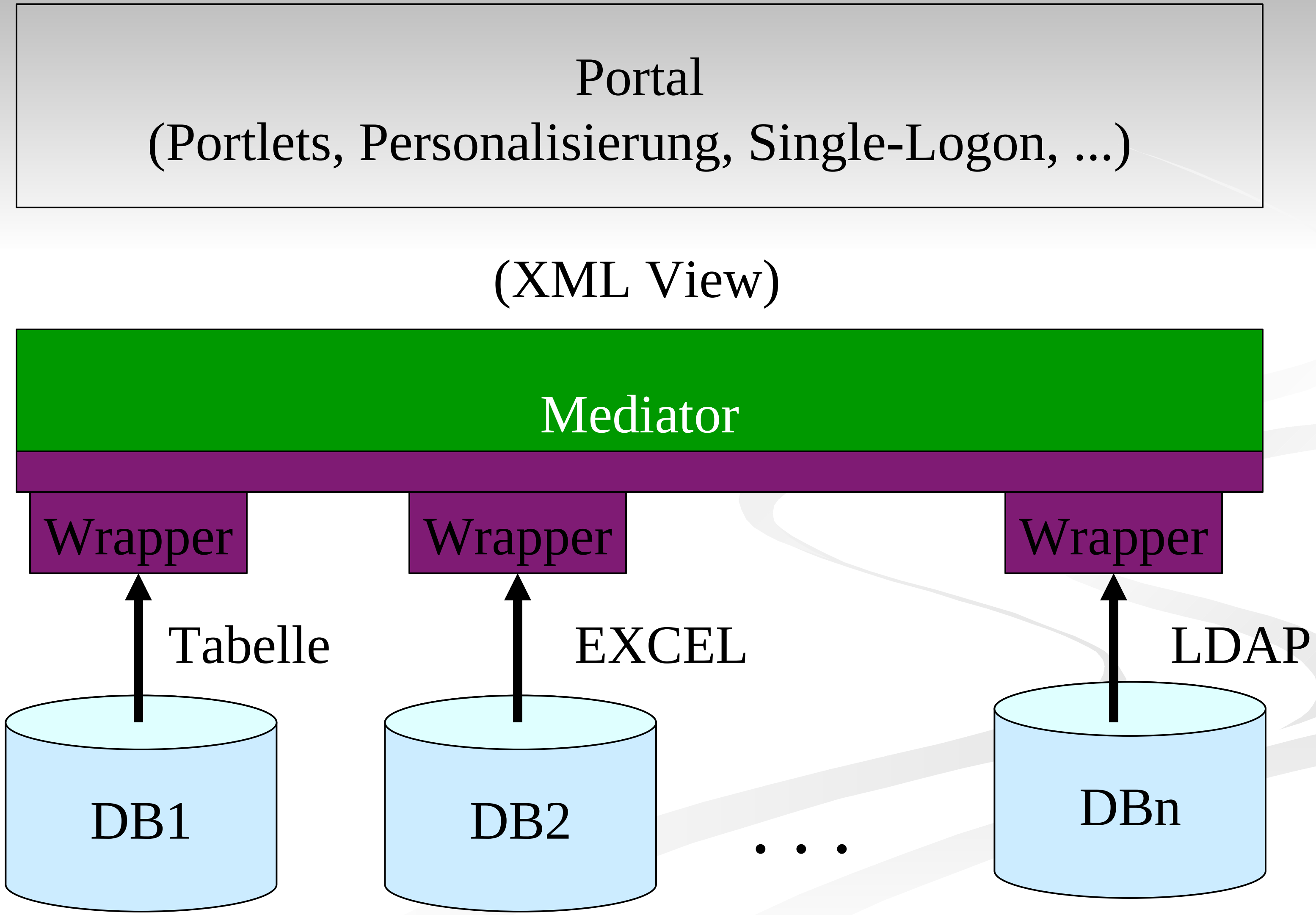


Web Services

Information Integration



Information Integration



Portal

(Portlets, Personalisierung, Single-Logon, ...)

(XML View)

Mediator

Wrapper

Wrapper

Wrapper

Tabelle

EXCEL

LDAP

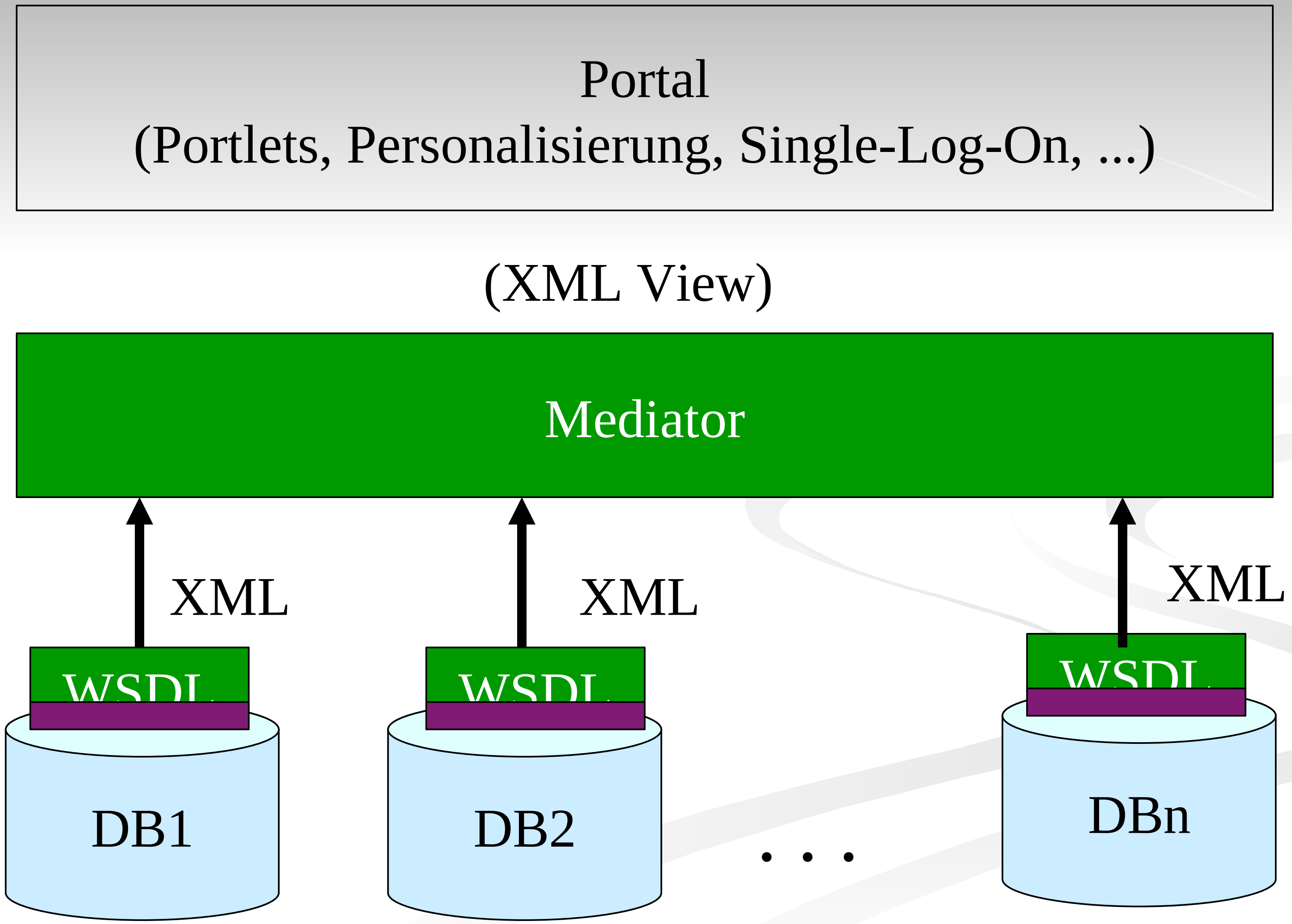
DB1

DB2

. . .

DBn

Information Integration



Portal

(Portlets, Personalisierung, Single-Log-On, ...)

(XML View)

Mediator

XML

XML

XML

WSDL

WSDL

WSDL

DB1

DB2

. . .

DBn

Information Integration

- Is the data model „XML“ or relational?
  - Advantage: Power of XML
  - Problem: XML for Business Intelligence???
- Are data sources wrapped as XML?
  - Many variants conceivable
  - Impact performance and project cost

Summary

- Why should I care about Web Services?
  - Potentially best technology for Integration and Management of large IT Infrastructures
  - Great Model for Outsourcing
  - Because everybody does it (Connectivity, Standards, Tools, Research)
- Why should I ignore Web Services?
  - „still“ an old hat
  - Not mature (cost, robustness, performance)
  - Technology Djungle

Agenda

- Web Services
  - **Definition**
  - SOAP
  - WSDL
  - UDDI
- RSS / ATOM
- MashUps
  - (Demo)

What is a Web Service?

- The short answer...
  - „A class on the Web“
- The long answer...
  - (see next 50 slides)

Defintion

- A service is a software component, has a purpose
- A service has a unique Id in its environment
  - Id is a URI -> Web Service
- Services communicate via messages
  - XML messages, SOAP, HTTP -> Web Service
  - Operations (Actions) are the Interface of a Service
  - Correlation/Conversations for complex tasks
- Compose complex services from basic services
- A service is autonomous; implem. is encapsulated
- A service may have a persistent state:
  - Transactions: CD okay, AI limited
  - Compensation, Options, restricted Abort

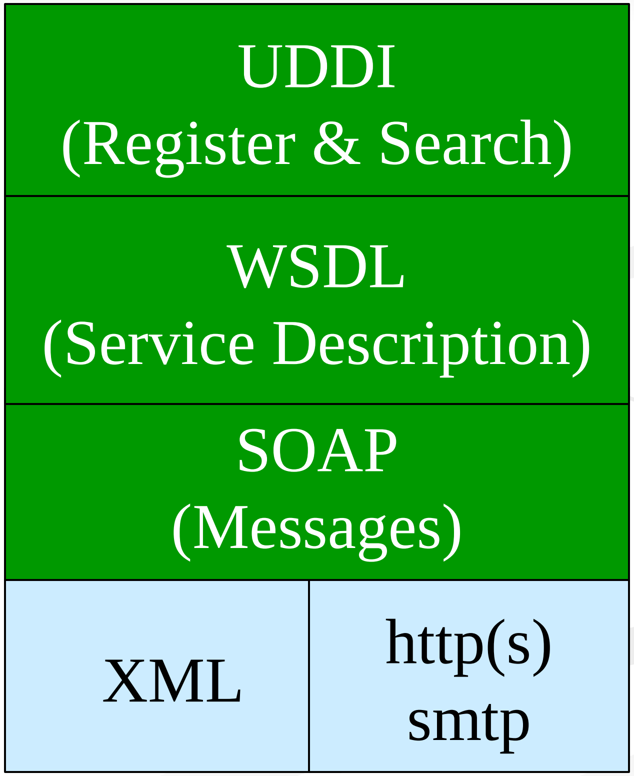
Definition of W3C

- A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.

SOA vs. OO

Web Services	**Object-Oriented**
Service (URI)	Class (Name)
Instance (Conversation) URI (ID of Conversation)	Object Object ID
Operation	Method
Message	Method Call
?	Inheritance, Substituability
Autonomy, Asyn., SLA	?

Web Services Stack



UDDI
(Register & Search)

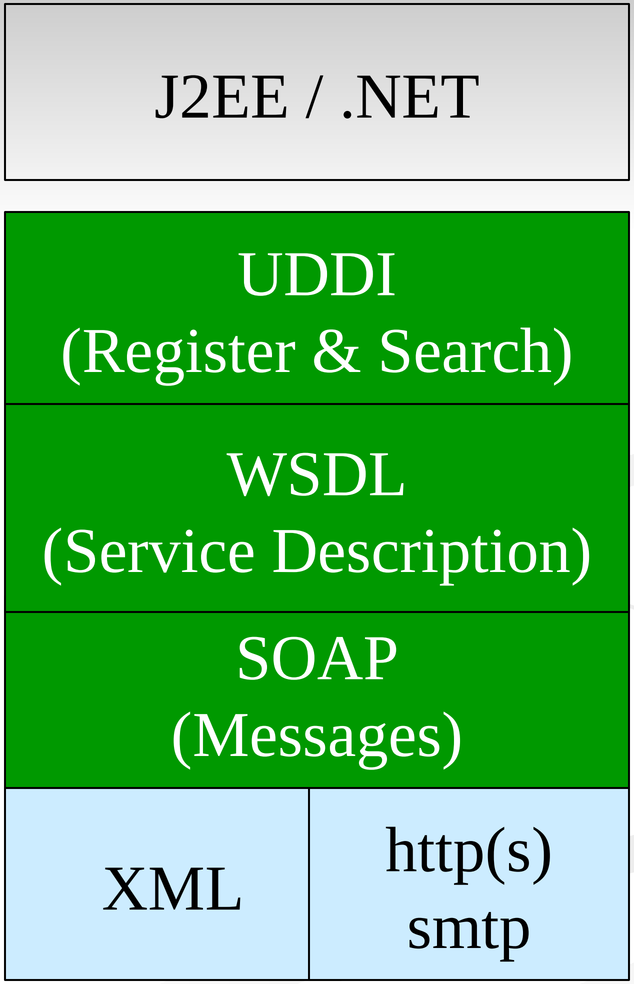
WSDL
(Service Description)

SOAP
(Messages)

XML

http(s)
smtp

Web Services Stack



J2EE / .NET

UDDI
(Register & Search)

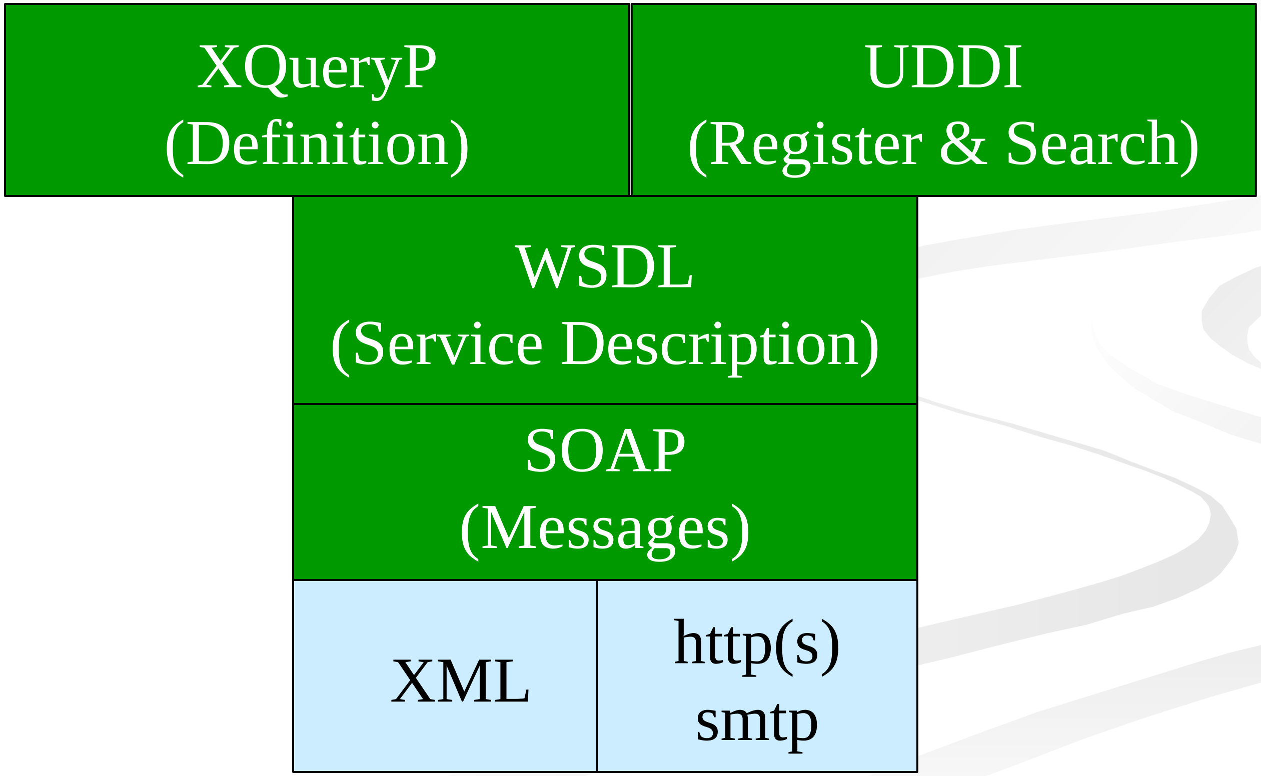
WSDL
(Service Description)

SOAP
(Messages)

XML

http(s)
smtp

Web Services Stack



XQueryP
(Definition)

UDDI
(Register & Search)

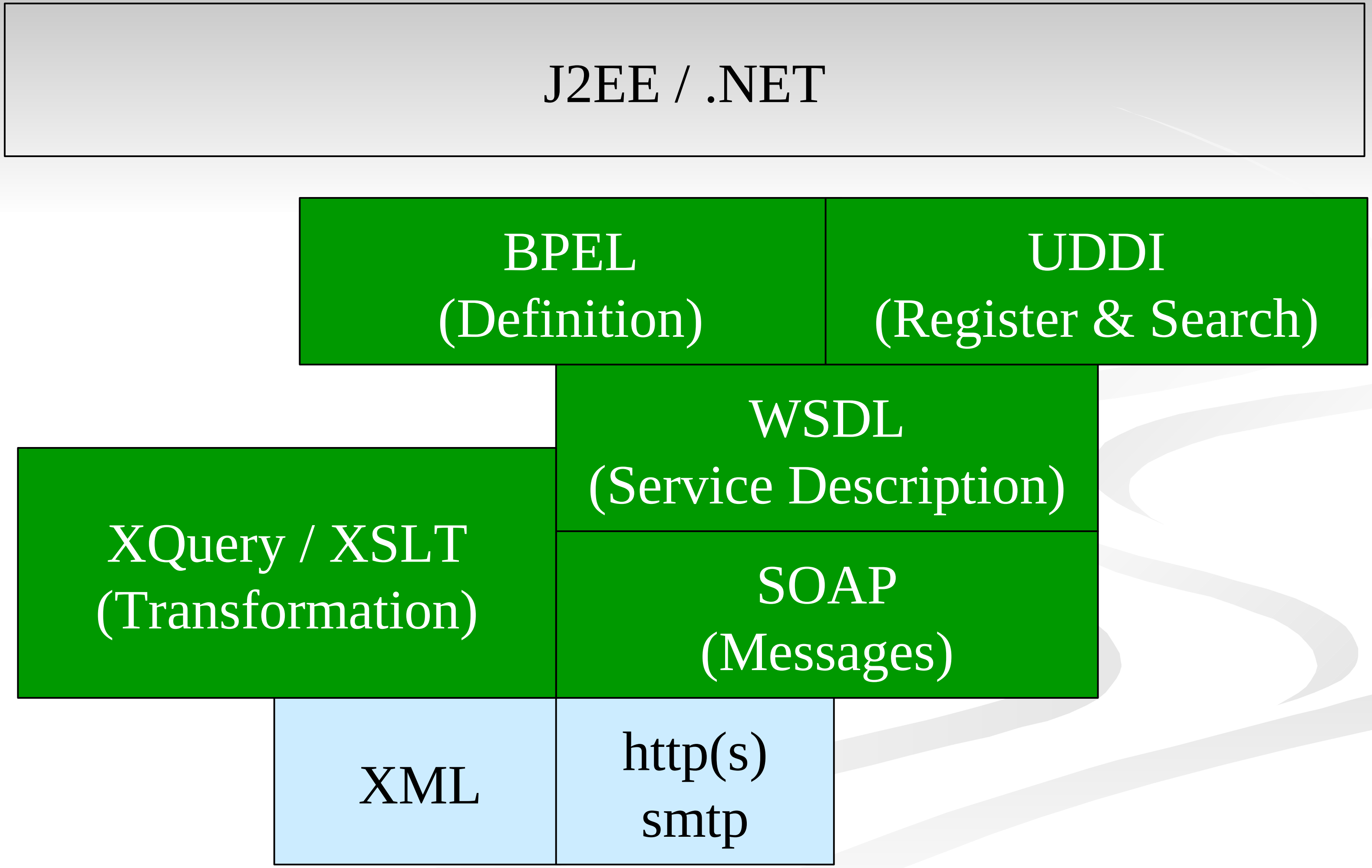
WSDL
(Service Description)

SOAP
(Messages)

XML

http(s)
smtp

Technology Stack



J2EE / .NET

BPEL
(Definition)

UDDI
(Register & Search)

WSDL
(Service Description)

XQuery / XSLT
(Transformation)

SOAP
(Messages)

XML

http(s)
smtp

Other Technologies (W3C, IBM & Microsoft)

- WS-Security, WS-Trust, WS-Privacy, WS-Federation, WS-Authorization, WS-Addressing
  - Signatures and Encryption of SOAP Messages
  - Security Tokens (e.g., for Kerberos)
  - Rules for privacy
  - Authorisation and federations of Web services
- WS-Policy
  - Properties / Restrictions for the use of Web services
- WSLA - Service Level Agreements
- WS-Transactions

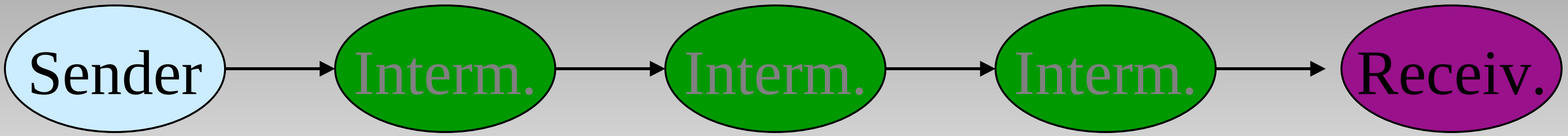
Agenda

- Web Services
  - Definition
  - **SOAP**
  - WSDL
  - UDDI
- RSS / ATOM
- MashUps
  - (Demo)

SOAP

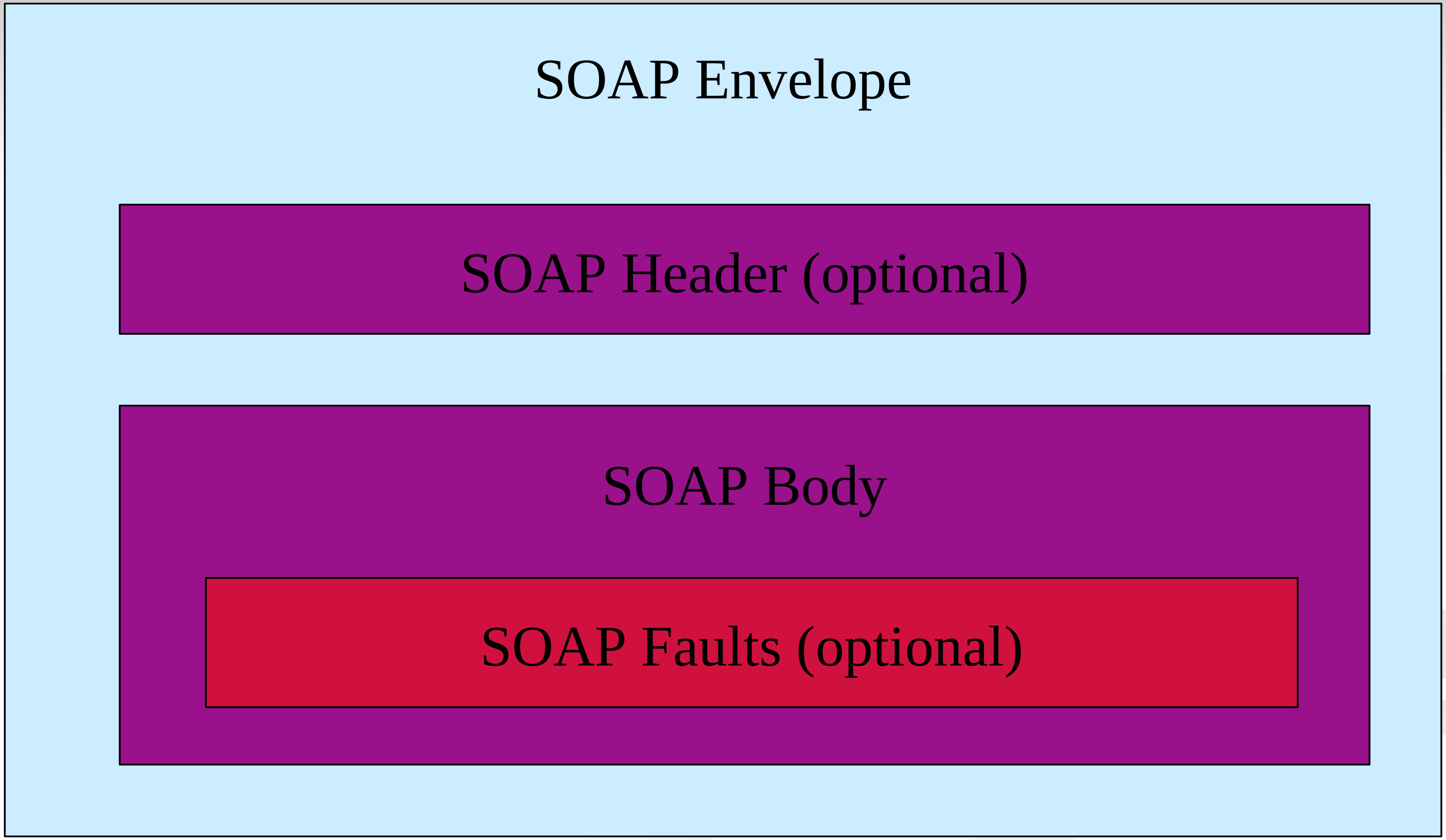
- SOAP = Simple Object Access Protocol
- W3C Standard; current version 1.2
- Communication between applications (e.g. RPC, Streams of Sensor Data)
- Defines Layout (Type) of Messages
- Use in Internet and through Firewalls
- Platform- and PL independent
- Based on XML
- Simple and extensible
- Basis for further standards (Encryption, ...)

Scenario



- Send Message from Sender to Receiver
- Pass Message through Intermediaries
  - Logging, Authorization, ..., application-specific
- Role of SOAP
  - Define Layout of Messages
  - Define Roles of Nodes
  - Enable alternative Protocol Bindings
- Enable Communication Patterns (e.g., RPC)

Structure of a Message



SOAP Envelope

SOAP Header (optional)

SOAP Body

SOAP Faults (optional)

SOAP Envelope

```
<?xml version=‚1.0‘ ?>
<env:Envelope
  xmlns:env=„http://.../soap-envelope“>
   <env:Header> ... </env:Header>
   <env:Body>  ...
       <env:Fault> ... </env:Fault>
       ...
    </env:Body>
</env:Envelope>
```

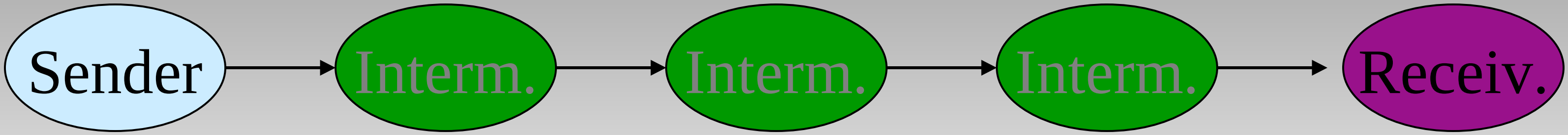
SOAP Header

- Info for Receiver and Intermediaries
- Structured in Blocks (sub-elements)
- Each block specifies
  - Who should read it
    (default: only receiver)
  - Who needs to understand (default: nobody)
- All elements must have qualified names
- Headers are optional

Example: Flight Reservation

```
<?xml version=‚1.0‘ ?>
<env:Envelope xmlns env = „...“>
<env:header>
 <m:reservation xmlns:m = „...“
     env:role = „http://.../next“
     env:mustUnderstand = „true“ >
   4711-31415
 </m:reservation>
  ... andere Blöcke im Header ...
</env:header>
 ...
```

Roles



- Predefined Roles (URIs in SOAP N.Space)
  - „next“ - everybody (Intermediaries and Receiver)
  - „none“ - nobody
  - „ultimateReceiver“ - Receiver only
- User-defined Roles (new URIs)
  - Application-dependent Matching possible

Processing Model

1. Parse Message
2. Check Block in Header
   - Does the role fit to me?
     (N.B. Blocks can involve several roles!!!)
   - Do I understand the block?
     - jes -> action
     - no -> if (mustUnderstand) then Error
       else Ignore
3. Further Actions (Relaying, Processing ...)

SOAP Body

- Only relevant for the receiver
- Exactly one body per message
- Content free and application defined
- All elements must be qualified

```
...
<env:body>
  <f:destination xmlns:f =
   „...“>Paris</f:destination>
  <f:origin>München<f:origin>
  <f:number>LH285</f:number>
</env:body>
```

Remote Procedure Call

- Specify operation + parameters in body
- Example: foo(5, Wutz)

```
<env:body>
 <op:call  xmlns:op = „...“  name = „foo“>
    <op:param>5</op:param>
    <op:param>Wutz</op:param>
 </op:call>
</env:body>
```

- Return result in „result“ element of body

Errors

- Element in Body with Sub-elements
  - code: defines code of error (mandatory)
  - reason: human-readible text (mandatory)
  - detail: further details (optional)
  - node: URI of node raising error (optional)
  - role: URI of role of node (optional)

Example: Error in RPC Call

```
<env:body>
 <env:Fault>
  <env:Code> <env:Value>env:Sender</env:Value>

   <env:Subcode>rpc:BadArguments</env:Subcode>
   </env:Code>
   <reason>Too many parameters:got 3, expected 2
   </reason>
 </env:Fault>
</env:body>
```

Predefined Error Codes

- env:VersionMismatch
  Envelope has no or wrong name space
- env:MustUnderstand
  Did not understand block in header
- env:DataEncodingUnknown
  Encodierung of message not supported
- env:Sender
  Wrong call of service (e.g., RPC)
- env:Receiver
  Local error at receiver (e.g. OutOfStock)

Protocol Binding

- Specify how message should be delivered
  - E.g., HTTP or SMTP protocol
- Specify how to serialize the message
  - E.g., pure XML, compressed XML, encrypted
- Specify Req/Resp Pattern (z.B. HTTP)
- Every hop has its own binding

SOAP 1.2 (XML Protocol)

- Extension of SOAP
  - Simpler error handling
  - „misunderstood“ Elements
  - Additional Roles (none, anonymous)
  - „response“ Element for RPC
  - etc.
- XML Protocol
  - Abstractes Model
  - Many, diverse application scenarios

Summary SOAP

- Important Building Block
  - Defines type / layout of messages
  - Defines error handling
  - Defines transmission through Intermediaries
  - Defines protocol bindings (http, smtp, beep, ...)
- Supports many Scenarios
  - More than just RPC
  - Wrapping of non XML Messages

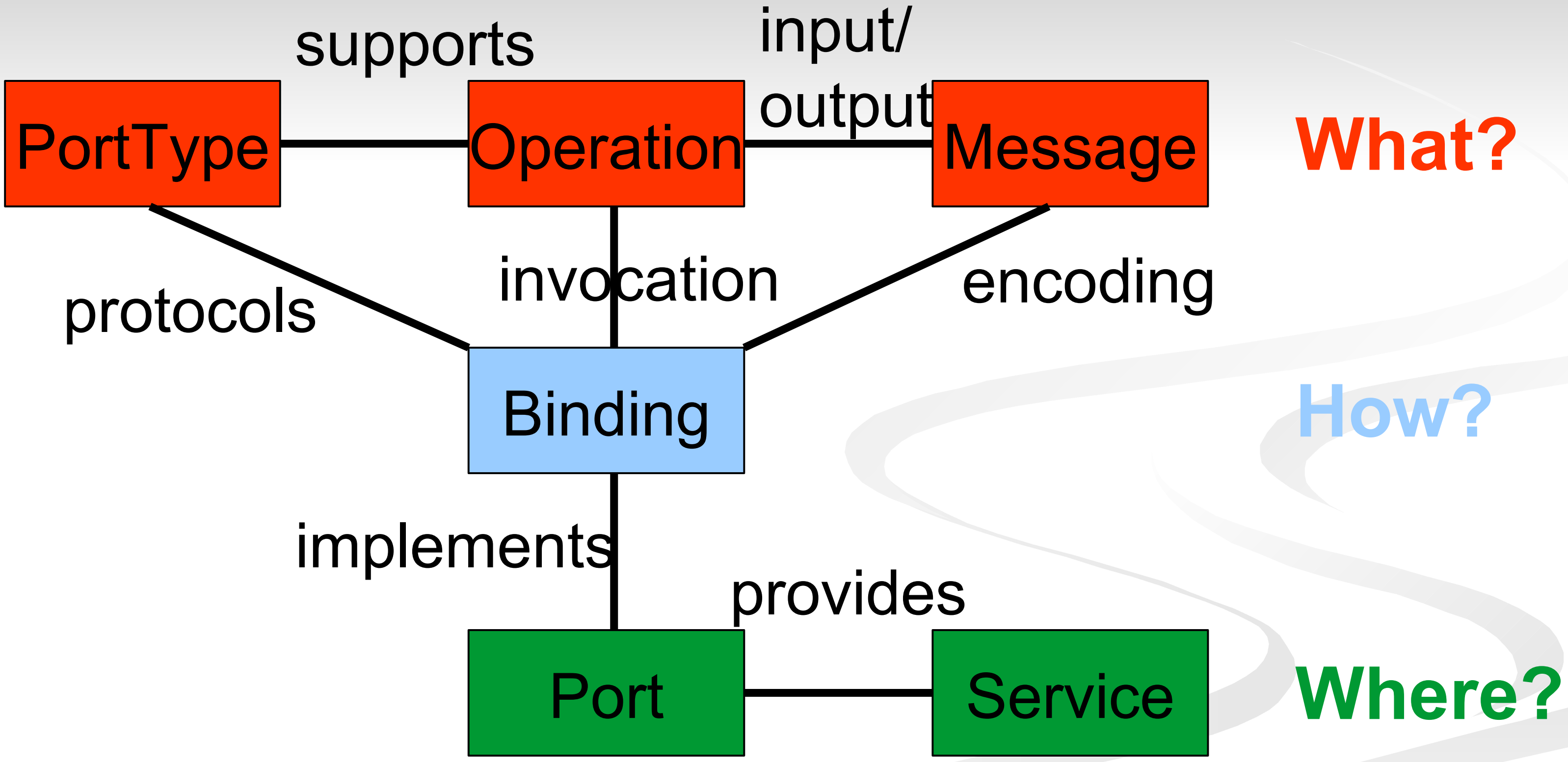
Agenda

- Web Services
  - Definition
  - SOAP
  - **WSDL**
  - UDDI
- RSS / ATOM
- MashUps
  - (Demo)

WSDL

- Web Service Description Language
- Describes the Interface of a Web Service
- Call of a Web Service done via SOAP
- Allows the registration of services
  - Basis for UDDI
- Syntax is XML

WSDL Overview



Components of Description

```
<wsdl:definitions  xmlns:wsdl =
   „http://w3.org/...“>
  <wsdl:documentation ... />
  <wsdl:types>  Schema Imports  </wsdl:types>
  <wsdl:message>  Messages  </wsdl:message>
  <wsdl:portType>  Operations </wsdl:portType>
  <wsdl:serviceType>OSets </wsdl:serviceType>
  <wsdl:binding> Protocols </wsdl:binding>
  <wsdl:service> Servicedefinition </wsdl:service>
</wsdl:definitions>
```

Types and Parameters

```
<wsdl:types> ?
    <wsdl:documentation .... /> ?
    <xsd:schema .... /> *
</wsdl:types>

<wsdl:message name="ncname"> *
    <wsdl:documentation .... /> ?
    <part name="ncname" element="qname"?
            type="qname"?/> *
</wsdl:message>
```

Operations

```
<wsdl:portType name="ncname"> *
    <wsdl:documentation .... /> ?
    <wsdl:operation name="ncname"> *
      <wsdl:documentation .... /> ?
      <wsdl:input message="qname"> ?
        <wsdl:documentation .... /> ?
      </wsdl:input>
      <wsdl:output message="qname"> ?
        <wsdl:documentation .... /> ?
      </wsdl:output>
      <wsdl:fault name="ncname" message="qname"> *
        <wsdl:documentation .... /> ?
      </wsdl:fault>
    </wsdl:operation>
</wsdl:portType>
```

Example: Addition

```
<message name=“addRequest">
   <part name="term1" type="xs:double"/>
   <part name="term2" type="xs:double"/>
</message>

<message name=“addResponse">
  <part name="value" type="xs:double"/>
</message>

<portType name=“arithmetics">
 <operation name=“add">
    <input message=“addRequest"/>
    <output message=“addResponse"/>
 </operation>
</portType>
```

N.B.: „wsdl:“ Qualifizierung fehlt

Bindings

```
<wsdl:binding name="ncname" type="qname"> *
    <wsdl:documentation .... /> ?
    <-- binding details --> *
    <wsdl:operation name="ncname"> *
        <wsdl:documentation .... /> ?
        <-- binding details --> *
        <wsdl:input> ?
           <wsdl:documentation .... /> ?
           <-- binding details -->
        </wsdl:input>
       desgleichen für Output und Fehler
  </wsdl:operation>
</wsdl:binding>
```

Servicetypes and Services

```
<wsdl:serviceType name="ncname"> *
  <wsdl:portType name="qname"/> +
</wsdl:serviceType>

<wsdl:service name="ncname" serviceType="qname"> *
    <wsdl:documentation .... /> ?
    <wsdl:port name="ncname" binding="qname"> *
       <wsdl:documentation .... /> ?
       <-- address details -->
    </wsdl:port>
</wsdl:service>
```

WSDL Summary

- What WSDL can do:
  - Describes the types of messages (in + out)
  - Describes protocols used in bindings
  - Describes the *static* Interface of a service
- What WSDL can ***not*** do:
  - Describe dynamic aspects (Choreographie)
  - Describe SLA, Transactions, Cost (WS Policy)
  - Describe properties for optimization (Side-effects, Compensation, ...)

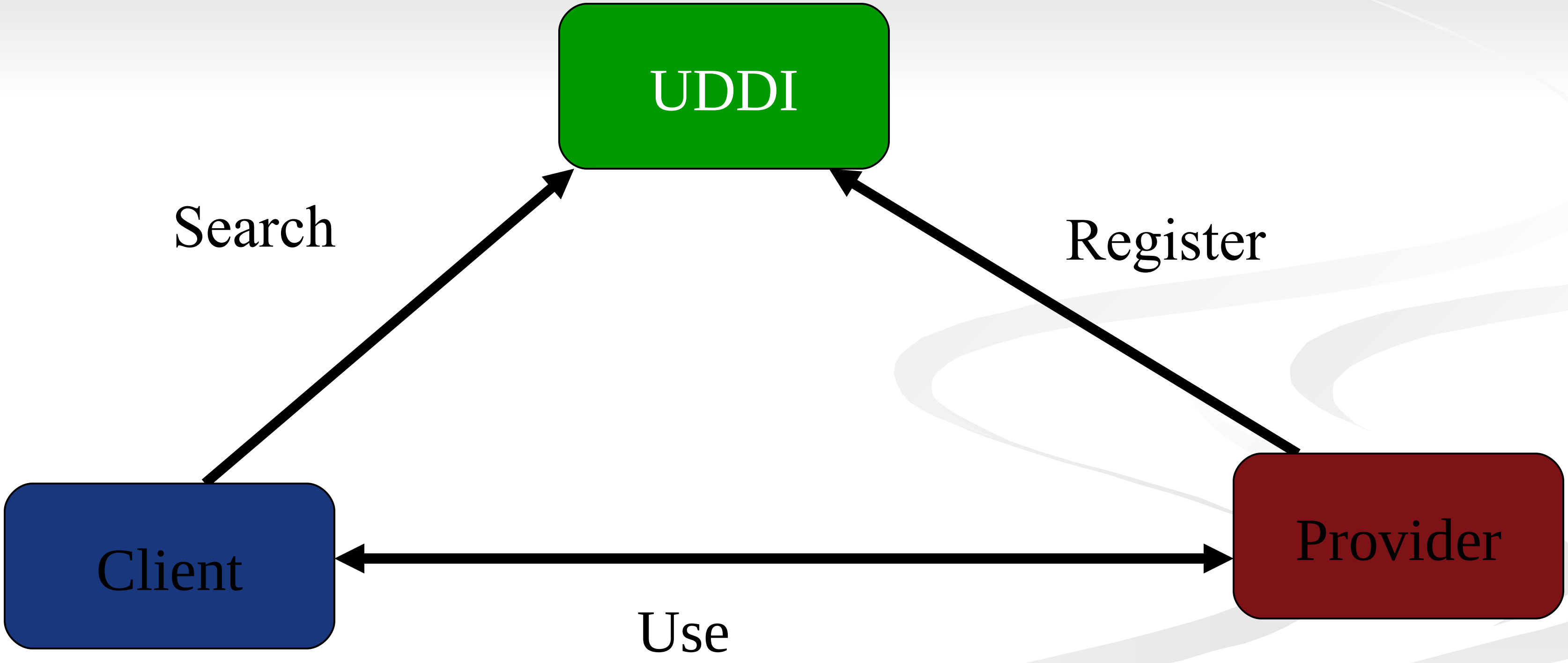
Agenda

- Web Services
  - Definition
  - SOAP
  - WSDL
  - **UDDI**
- RSS / ATOM
- MashUps
  - (Demo)

UDDI

- Universal Description Discovery Integration
- Directory which stores WSDL
- „Jini“ for the Web, „yellow pages“
- Communicates via SOAP Messages
- Organized in white, yellow and green pages
  - white, yellow pages: Informationen about Providers
  - green pages: WSDL of Services
- IBM and Microsoft have public UDDI Server
- Today, typically used in Intranet

UDDI



UDDI Summary

- What UDDI can do:
  - Store Meta data of Services
    (good for Intranet and Extranet)
  - Unified Interface for Register & Search
    (first step towards „Virtualisation“)
- What UDDI can *not* do:
  - Guarantees about providers (TÜV)
    (needed in Internet)
  - Extensible Data Model
    (no user-defined meta-data)
  - Does not compensate for WSDL‘s weaknesses

Agenda

- Web Services
  - Definition
  - SOAP
  - WSDL
  - UDDI
- **RSS / ATOM**
- MashUps
  - (Demo)

Is „Pull“ the winner?

- Most of our interaction with the Web and Databases is „Pull“
  - Browse the web to find the pictures of my friend‘s vacation on some remote island
  - Query the database to find out which books were the the top sellers in Zurich around Christmas
  - Invoke a web service to compute π up to ten billion digits

Is this the whole story?

Examples of „Push“

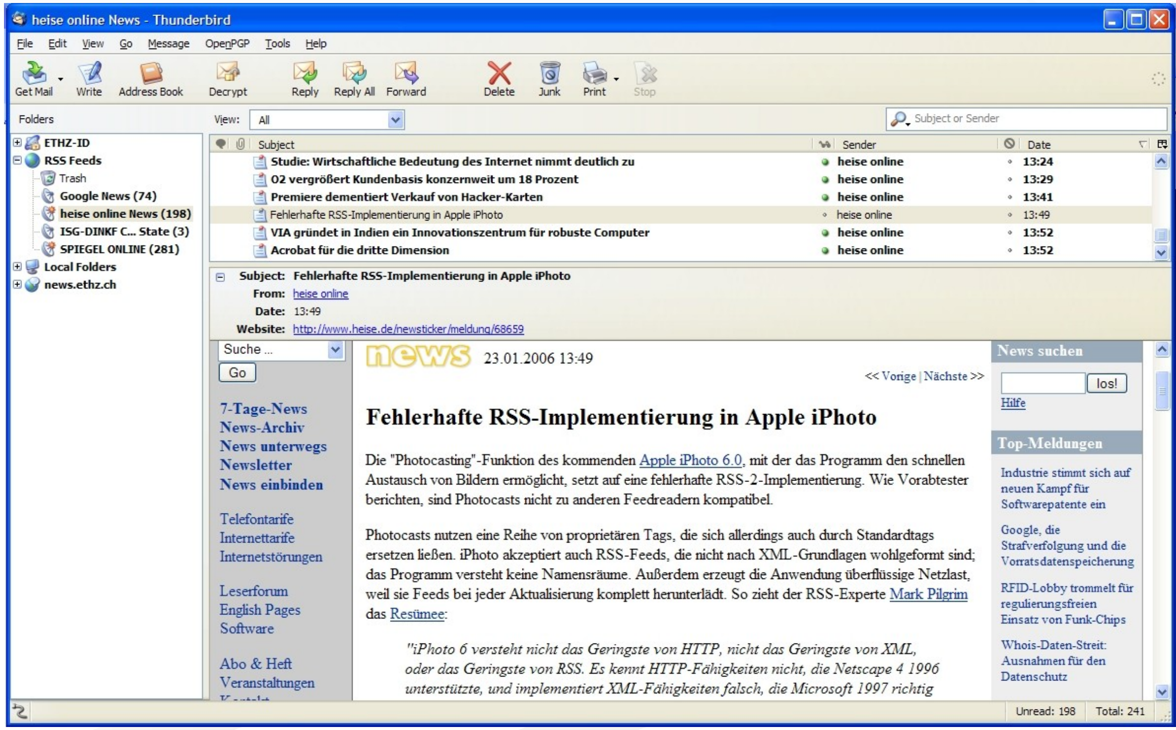
- E-Mail communication:
  - Who uses a Blackberry?
  - Who sends more than a 100 SMS/month?
- Event notification
  - Information about offers (apartments, cars, jobs)
  - News, stock tickers
- Sensor data
  - Monitor temperature in a building

Factors favoring „Push“

- Long-standing interest
- Very low or very high update rate
  - 1 update per week
  - 100 updates per second
- Large number of independant sources
  - Watching 100.000 news sites all over the world is impossible, but watching an RSS feed via Google News is certainly possible
- Scalability: many users want the same thing
  - E.g., this lecture, TV, …

RSS

- Content syndication:
  - News tickers
  - Blogs
  - Alerts
- Simple XML format
- Lightweight
- Still some get it wrong ☺



Subject	Sender	Date
Studie: Wirtschaftliche Bedeutung des Internet nimmt deutlich zu	heise online	13:24
O2 vergrößert Kundenbasis konzernweit um 18 Prozent	heise online	13:29
Premiere dementiert Verkauf von Hacker-Karten	heise online	13:41
Fehlerhafte RSS-Implementierung in Apple iPhoto	heise online	13:49
VIA gründet in Indien ein Innovationszentrum für robuste Computer	heise online	13:52
Acrobat für die dritte Dimension	heise online	13:52

Subject: Fehlerhafte RSS-Implementierung in Apple iPhoto
From: heise online
Date: 13:49
Website: http://www.heise.de/newsticker/meldung/68659

23.01.2006 13:49

Fehlerhafte RSS-Implementierung in Apple iPhoto

Die "Photocasting"-Funktion des kommenden Apple iPhoto 6.0, mit der das Programm den schnellen Austausch von Bildern ermöglicht, setzt auf eine fehlerhafte RSS-2-Implementierung. Wie Vorabtester berichten, sind Photocasts nicht zu anderen Feedreadern kompatibel.

Photocasts nutzen eine Reihe von proprietären Tags, die sich allerdings auch durch Standardtags ersetzen ließen. iPhoto akzeptiert auch RSS-Feeds, die nicht nach XML-Grundlagen wohlgeformt sind; das Programm versteht keine Namensräume. Außerdem erzeugt die Anwendung überflüssige Netzlast, weil sie Feeds bei jeder Aktualisierung komplett herunterlädt. So zieht der RSS-Experte Mark Pilgrim das Resümee:

> *"iPhoto 6 versteht nicht das Geringste von HTTP, nicht das Geringste von XML, oder das Geringste von RSS. Es kennt HTTP-Fähigkeiten nicht, die Netscape 4 1996 unterstützte, und implementiert XML-Fähigkeiten falsch, die Microsoft 1997 richtig*

Industrie stimmt sich auf neuen Kampf für Softwarepatente ein

Google, die Strafverfolgung und die Vorratsdatenspeicherung

RFID-Lobby trommelt für regulierungsfreien Einsatz von Funk-Chips

Whois-Daten-Streit: Ausnahmen für den Datenschutz

RSS 2.0

- Simple Message Format for Data Push
  - <channel>
  - <item>

    ...

    <cal:startTime>...</cal:startTime>

    </item>…
  - </channel>

RSS Items and Types

Items
Item+ **Title: Text** **Link: URL** **Description: Text** GUID? : Unique ID Author*: Name Category*: Tag Comments?: URL Enclosure?: Blob Source?: URL

“Types”	
Author	Just Text
Date	Sat, 07 Sep 2002 9:42:31 GMT
URL	<link> http://www.nytimes.com/2002/09/07/movies/07FEST.html </link>
Text	<description> Some of chatter at Venice was about the stars. </description>
Comment	<comments>http://a.org/comments/123 </comments>
Blob	<enclosure url="http://a.org/mp3/cscms.mp3" length="1069871“type="audio/mpeg"/>
Tag	<category domain="google.com">entertainment</category>

RSS 2.0 example

```
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <channel>
      <title>D-INFK Events</title>
      <description>Events of the Department of Computer Science, ETH Zurich</description>
      <link>http://www.inf.ethz.ch/news/events/</link>
      <docs>http://www.inf.ethz.ch/rss</docs>
      …
      <pubDate>Tue, 17 Jan 2006 11:06:04 GMT</pubDate>
       <image> <url>http://www.inf.ethz.ch/rss/inf-logo.png</url>
                 <title>Department of Computer Science</title>
                 <link>http://www.inf.ethz.ch/</link>
                 <width>140</width> <height>35</height>
       </image>
       <item rdf:about="http://www.inf.ethz.ch/news/events/details/index?id=593">
                 <title>Establishing trust in electronic business correspondence</title>
      <link>http://www.inf.ethz.ch/news/events/details/index?id=593</link>
                 <category>ZISC Colloquium</category>
                 <description>Tuesday, 17 January 2006 17:15, by: Dr. Ralf Hauser Privasphere
                       AG</description>
<dc:date>2006-01-17</dc:date>
      <guid>http://www.inf.ethz.ch/news/events/details/index?id=593</guid> </item>
```

RSS Notes

- Format wars: RSS 0.91, 1.0, 2.0, Atom
- All major news sites use it now
- Blogs would not work without it
- Currently targeted to human-machine communication
- Might be a good candidate for push-style machine-machine communication, too

- Ironically, RSS currently uses
  - push-only as interaction model
  - pull at the communication level

Atom

- Direct Competition to RSS 2.0
  - <feed>
  - <entry>

      ...
      <cal:startTime>...</cal:startTime>

    </entry>…
  - </feed>

Atom/RSS Types

“Types”	Atom	RSS
Author	<name>George Matesky</name> <email>geo@herald.com</email> <uri>www.matesky.net</uri>	Just Text
Date	2002-09-07T09:42:31Z	Sat, 07 Sep 2002 9:42:31 GMT
URL	<link rel="alternate" type="text/html“ ref="http://example.org/2005/04/02/atom" />	<link> http://www.nytimes.com/2002/09/07/movies/07FEST.html </link>
Text	<content type="xhtml" xml:lang="en"> <div xmlns="http://www.w3.org/1999/xhtml"> <p>Some of the chatter at <i>Venice </i> was about the stars</p> </div> </content>	<description> Some of chatter at Venice was about the stars.</description>
Comment	<link rel="comments" href=“http://a.org/comments/123"/>.	<comments>http://a.org/comments/123 </comments>
Blob	<link rel="enclosure" type="audio/mpeg" length="1337" href="http://a.org/mp3/cscms.mp3"/>	<enclosure url="http://a.org/mp3/cscms.mp3" length="1069871“type="audio/mpeg"/>
Tag	<category term="entertainment" scheme="google.com"/>	<category domain="google.com">entertainment</category>

Atom/RSS Items

Atom	RSS
Entry+	Item+
Title: Text	**Title: Text**
ID: Unique ID	**Link: URL**
Updated: Date	**Description: Text**
Content? : Text	GUID? : Unique ID
Link?: URL	Author*: Name
Link*: Blob	Category*: Tag
Link*: Comments	Comments?: URL
Published?: Date	Enclosure?: Blob
Author*: Author	Source?: URL
Contributor*: Author	
Category*: Tag	
Link?: EditURL(rel=sService.edit)	

Agenda

- Web Services
  - Definition
  - SOAP
  - WSDL
  - UDDI
- RSS / ATOM
- **MashUps**
  - **(Demo)**

Mashups

- Compose a new Web Service / Page from existing Web Services
  - VERY simple concept (that is the beauty)
- Examples
  - Restaurant Guide + Google Maps
  - http://www.gangstaweb.com
  - http://www.programmableweb.com
- Demo - MXQuery Engine
  - http://www.mxquery.org

Summary

- Integration, Integration, Integration
  - Applications and/or Data
- XML is strong because
  - more forgiving if formats change
  - serialization of data
- Push: XML is strong because
  - RSS and Atom are incidently XML
  - XML is good for text / documents with some structure
  - XML is good for integration
- Mashups: XML is strong because
  - can be processed by machines (WS) and humans (XHTML)
- Next: Programming for XML (XPath, ..., XQueryP)