Internetanwendungstechnik

Web Services

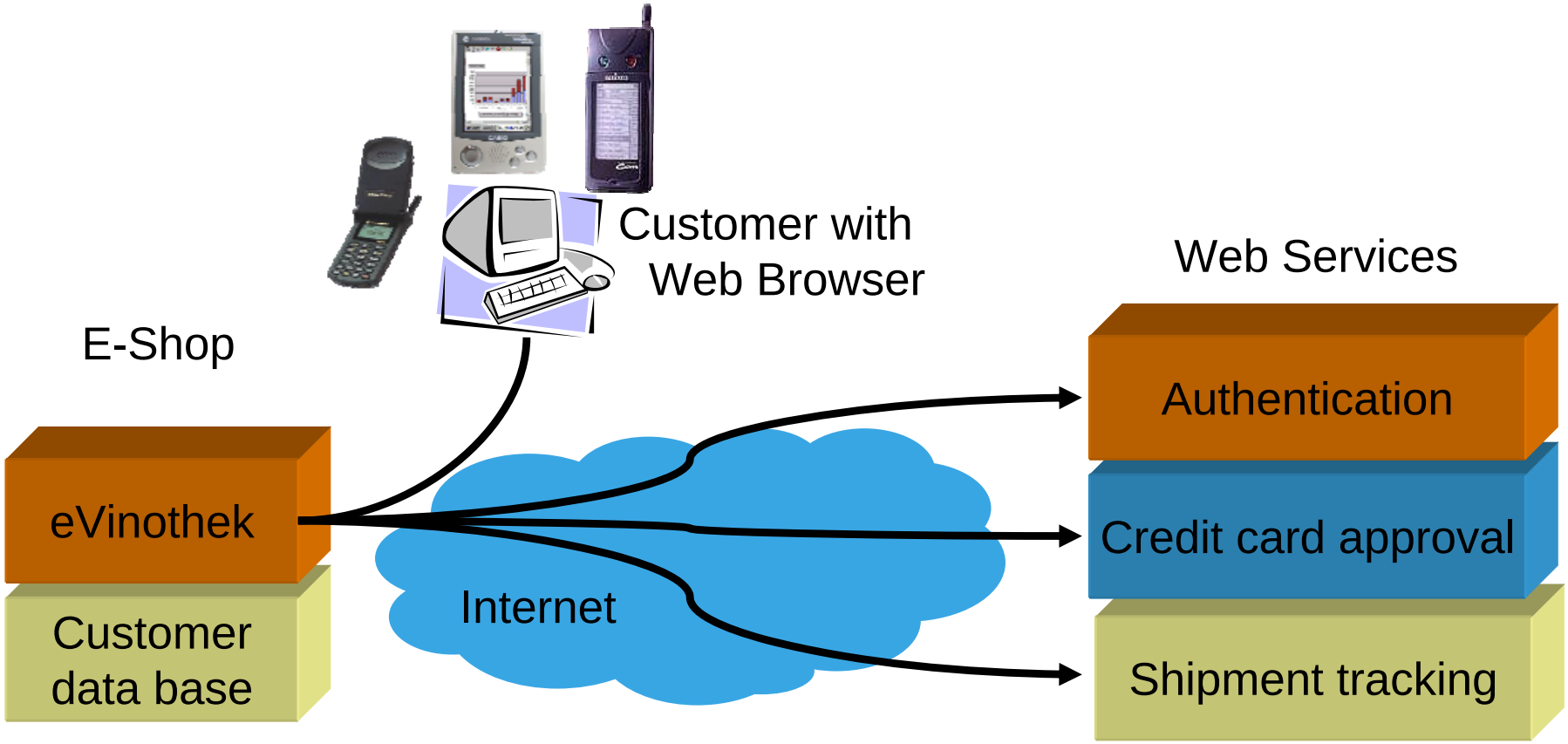
Gero Mühl

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Kommunikations- und Betriebssysteme (KBS)
Einsteinufer 17, Sekr. EN6, 10587 Berlin

Evolution of the Web

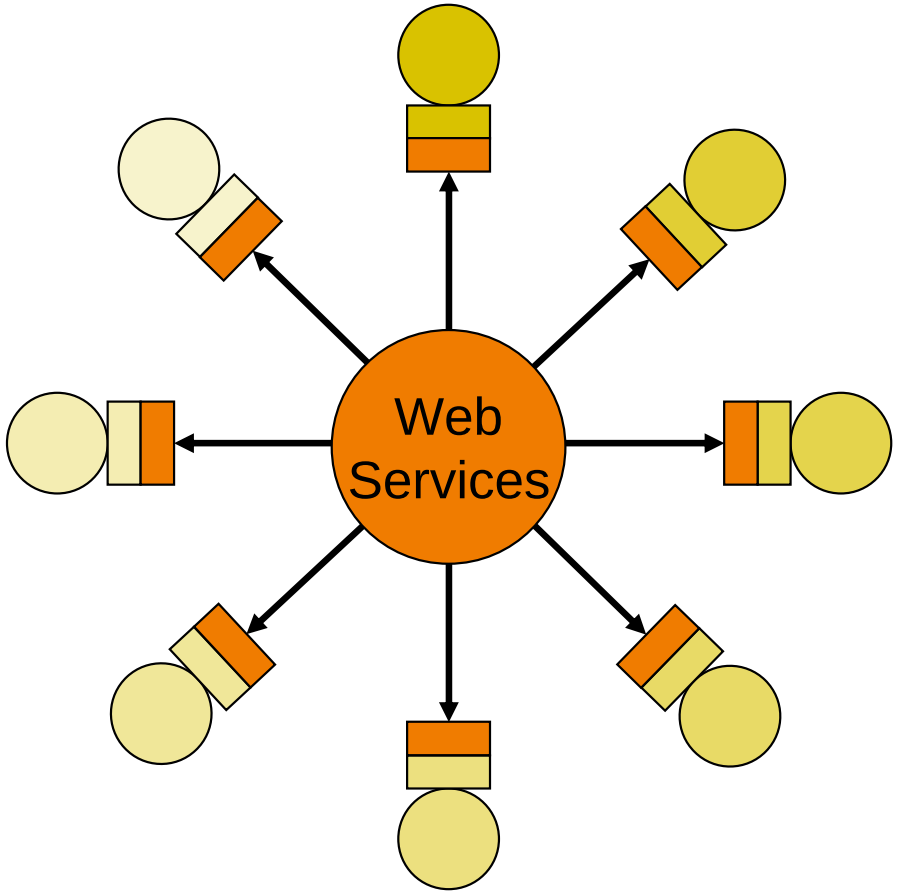
- Static web pages
  - Information browsing
- Dynamic web pages
  - Programmable web servers and clients
  - Applications: e-Shops, Train Schedules, …
- **Web Services**
  - Distributed computing for the Internet
  - Make services available on the Internet

Web Services Scenario



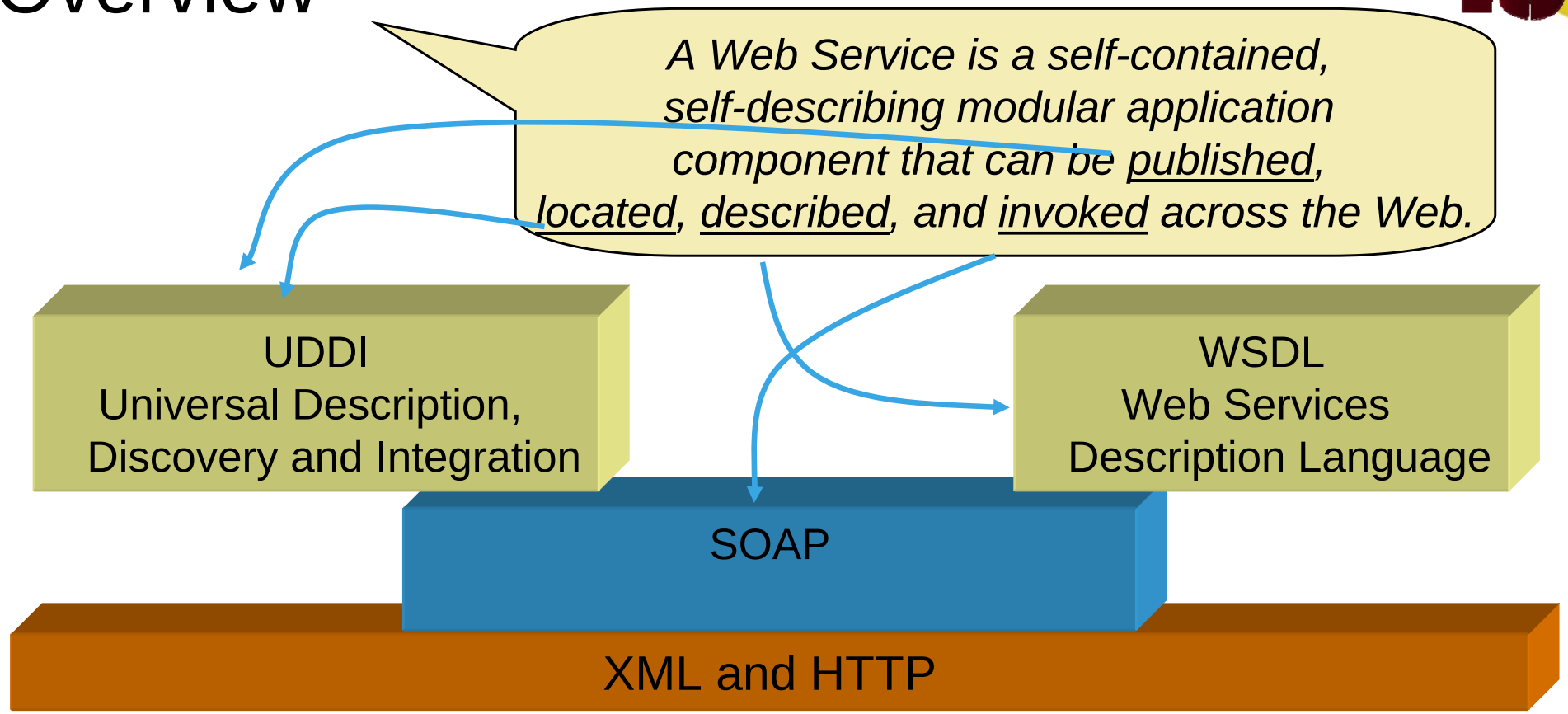
Internet = general purpose middleware

Web Services Scenario



Enterprise Application Integration (EAI)

Overview



- Examples
  - business oriented: credit authorization
  - consumer oriented: stock quotes
  - system oriented: user authentication
  - enterprise oriented: enterprise application integration (EAI)

Object Invocation Protocols Revisited

- CORBA: GIOP (General Inter ORB Protocol)
  - standardized message types and syntax
  - pretty complex framework, expensive products
- Java RMI: JRMP (Java Remote Method Protocol)
  - binary protocol
  - “requires” Java on both sides (→ platform restrictions)
- DCOM: ORPC
  - binary protocol
  - platform restrictions (mostly Windows)

→ all of them are usually blocked by firewalls ...

SOAP

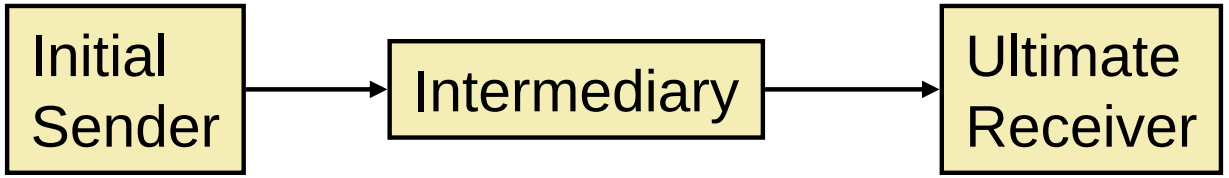
- Lightweight protocol for exchanging structured and typed information
- Essentially stateless, one-way message exchange between **SOAP Nodes**
- Applications can create more complex interaction patterns (request/response, request/multiple response, back-and-forth "conversation", etc.) on top of SOAP
- Applications responsible for message routing, security, reliability, etc.
- Design Goal: KISS (Keep It Simple Stupid)
  - simple to implement
  - stick to absolutely minimum of functionality
  - modular and extensible

SOAP

- SOAP is independent of any “underlying” transport protocol
- SOAP messages may be exchanged using a variety of transport protocols
- SOAP Bindings specify how SOAP messages may be passed using a certain protocol
  - SOAP HTTP Binding, SOAP Email Binding etc.
- A SOAP message is formally specified as an XML Infoset which provides an abstract description of its contents
- Infosets can have different on-the-wire representations, the most common is as an XML document

SOAP Processing Model

- SOAP nodes act on receiving a SOAP message
- SOAP messages are routed along a SOAP message path
  - from an initial SOAP sender
  - over SOAP intermediaries
  - to the ultimate SOAP receiver
- SOAP intermediaries are optionally
  - they act as both SOAP receivers and SOAP senders



Why XML?

- XML = eXtended Markup Language
- Text-based data presentation
  - hierarchical, self-describing, semistructured
- Platform-independent
- Supported by all major (and most minor) IT players
- Many tools and APIs (e.g. DOM, SAX) available
- XML documents can be transformed (e.g., into a HTML page or a plain text document) using XSLT

XML is

- ... an e**X**tensible **M**arkup **L**anguage
- ... a meta-language for defining other languages
- ... a semistructured data model
- … a self–describing exchange syntax
- ... the ASCII of the Web
- ... many good (and some bad) Computer Sciences ideas
- ...

XML (eXtensible Markup Language)

- Origins
  - HTML + SGML (ISO Standard, 1986, ~600pp)
- W3C standard (~26 pp): XML + DTD syntax
- XML = HTML – presentational tags
  + user-defined DTD
  = a meta-language for defining other languages
- XML = SGML – {complexity, document perspective}
  + simplicity
  + data exchange perspective
- DTD = Document Type Definition

HTML vs. XML

HTML tags:
Predefined, fixed set, presentation, generic document structure

```
<h1> Bibliography </h1>
<p> <i>  Foundations of DBs </i>, Abiteboul, Hull, Vianu
    <br> Addison-Wesley, 1995
<p> <i>  Logics for DBs and ISs </i>, Chomicki, Saake, eds.
    <br> Kluwer, 1998
```

XML tags:
Not predefined, extensible set, content-specific, „semantic“, (DTD-) specific

```
<biobliography>
  <book> <title> Foundations of DBs </title>
         <author> Abiteboul </author>
         <author> Hull </author>
         <author> Vianu </author>
         <publisher> Addison-Wesley </publisher>
         <year> 1998 </year>
  </book>
  <book> … <editor> Chomicki </editor> … </book>
  …
</bibliography>
```

XML as a *Self-Describing Data Exchange* Format

- Can be parsed easily
- Contains its own structure (→ parse tree) in the data
  - Allows the application programmer to (re)discover schema
  - What about the semantics?
- May also include an explicit schema description (e.g., a DTD)
  - meta-language: definition of a language with respect to which the document is valid
- allows separation of marked-up content from presentation
  → style sheets

XML – Different Perspectives

Document (SGML) Community

- data = linear text documents
- mark up (annotate) text pieces to describe
  - context,
  - structure,
  - semantics

  of the marked text

Database Community

- XML as a (prominent) example of the semistructured data model
- captures the whole spectrum from
  - highly structured,
  - regular data to
  - unstructured data

Many X-cellent Acronyms?

- XML (Extensible Markup Language)
- XML Namespaces
- XML DTDs, XML Schema
- RDF (Resource Description Framework)
- XSL (Extensible Style Sheet Language)
- XPath (=XSLT∩ XPointer), XLink
- XQL, XML-QL (XML Query Language), Quilt
- XMAS (XML Matching And Structuring language)
- eXcelon, ...

⇒ a family of technologies (extensions, tools, ...)

⇒ Generic standards and industry/community standards

XML Applications & Industry Initiatives

- Advertising: adXML place an ad onto an ad network or to a single vendor
- Literature: Gutenberg convert the world's great literature into XML
- Directories: dirXML Novell's Directory Services Markup Language (DSML)
- Web Servers: apacheXML parsers, XSL, web publishing
- Travel: openTravel information for airlines, hotels, and car rental places
- News: NewsML creation, transfer and delivery of news
- Human Resources: XML-HR standardization of HR/electronic recruiting XML definitions
- International Dvt: IDML improve the mgt. and exchange of info. for sustainable development
- Voice: VoxML markup language for voice applications
- Weather: OMF Weather Observation Markup Format (simulation)
- Geospatial: ANZMETA distributed national directory for and information
- Banking: MBA Mortgage Bankers Association of America → credit report, loan file, underwriting…
- Healthcare: HL7 DTDs for prescriptions, policies & procedures, clinical trials
- Math: MathML (Mathematical Markup Language)
- Surveys: DDI (Data Documentation Initiative) "codebooks" in the social and behavioral sciences

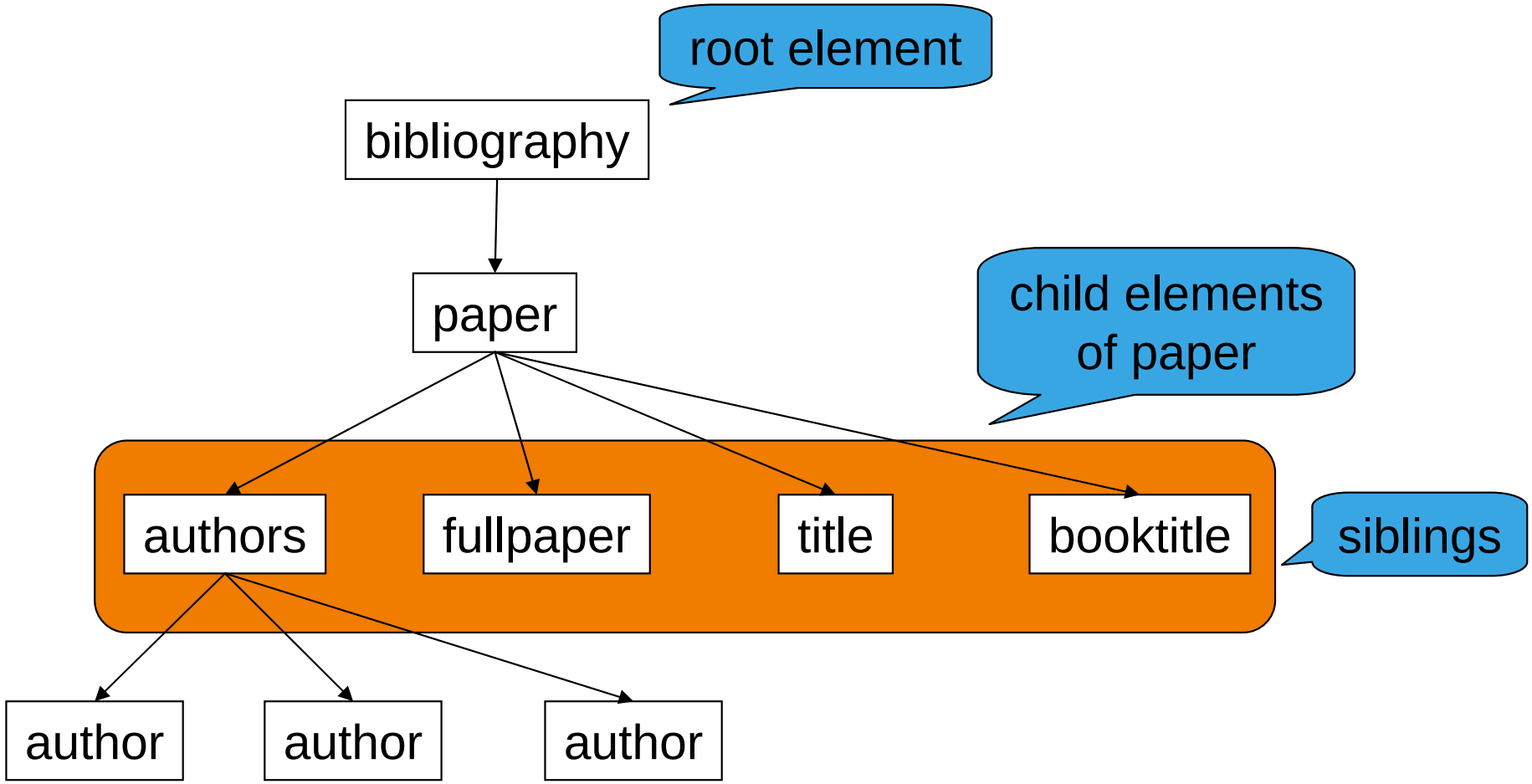
XML is Based on Markup

```
<bibliography>

  <paper ID= "object-fusion">
    <authors>
      <author>Y. Papakonstantinou</author>
      <author>S. Abiteboul</author>
      <author>H. Garcia-Molina</author>
    </authors>
    <fullPaper source="fusion"/>
    <title>Object Fusion in Mediator Systems</title>
    <booktitle>VLDB 96</booktitle>
  </paper>
<!-- comment -->
</bibliography>
```

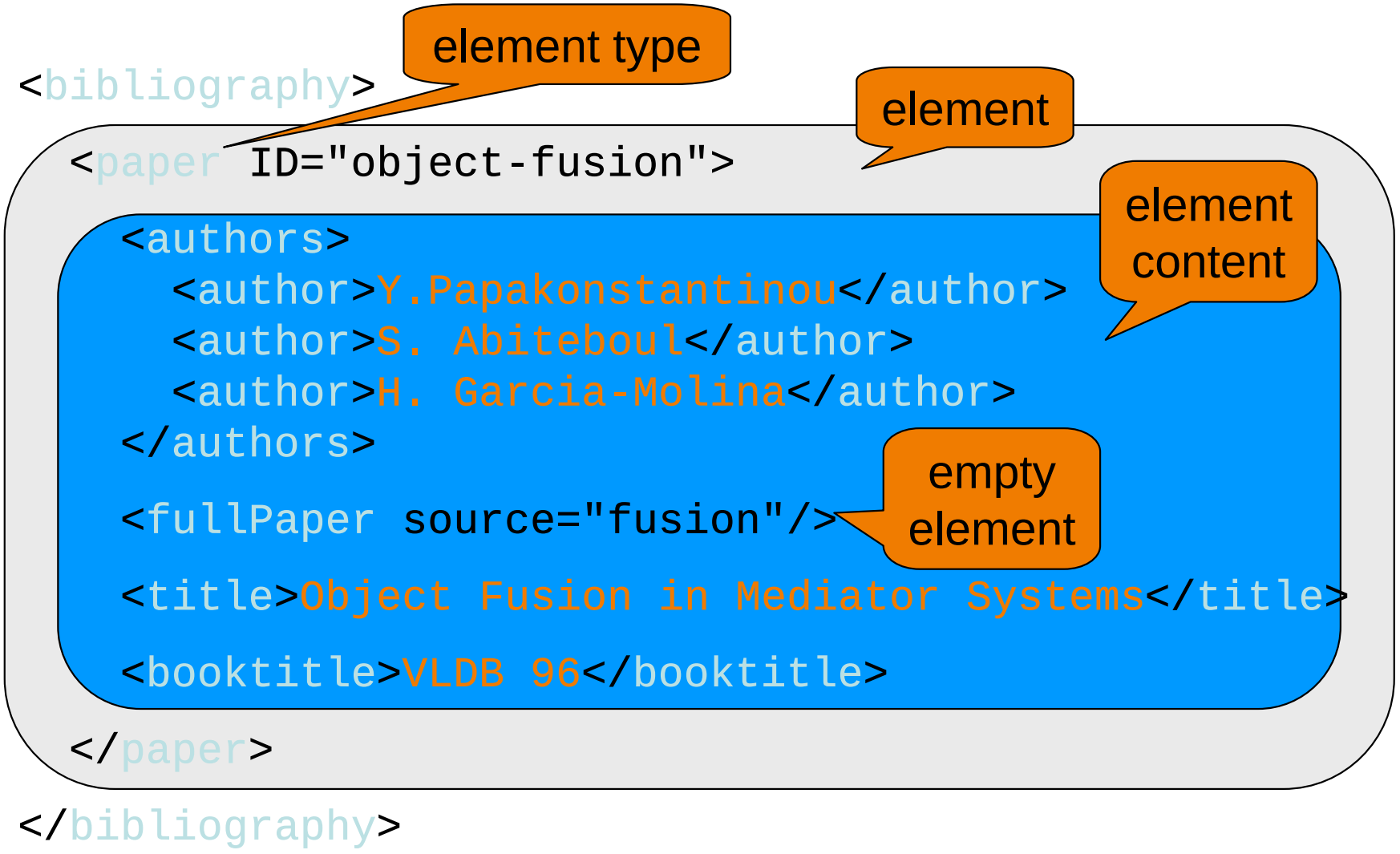
Markup indicates structure (and semantics?)

Decoupled from presentation

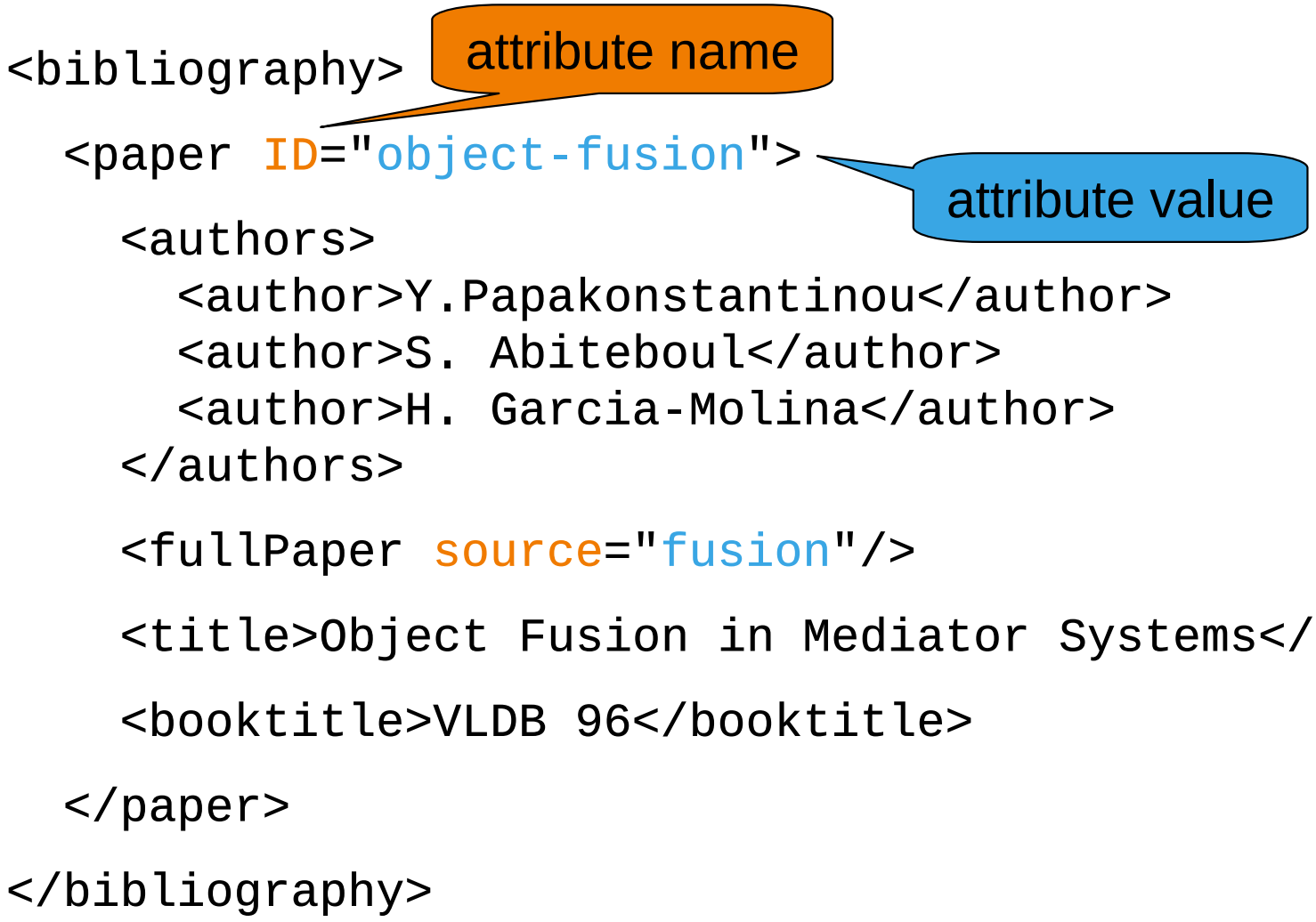
Logical Document Structure



Elements and their Content



Element Attributes




```
<bibliography>
  <paper ID="object-fusion">
    <authors>
      <author>Y.Papakonstantinou</author>
      <author>S. Abiteboul</author>
      <author>H. Garcia-Molina</author>
    </authors>
    <fullPaper source="fusion"/>
    <title>Object Fusion in Mediator Systems</title>
    <booktitle>VLDB 96</booktitle>
  </paper>
</bibliography>
```

Well-formed XML Documents

- begin with the XML declaration,
  e.g. `<?xml version="1.0"?>`
- have one unique root element
- all start tags must match end tags (case sensitive)
- all elements must be closed
- all elements must be properly nested
- all attribute values must be quoted
- XML Entities must be used for special characters,
  e.g. *<* for <

XML and XML DTDs

- Differentiate between
- tagging syntax for document instance
  (→ well-formed document)
- and
- DTD “schema”that defines a set of allowed documents
  (→ valid document)

XML DTDs as Extended Context Free Grammars

- XML DTD

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title,
                 booktitle)>
<!element authors author+>
```

- Grammar

bibliography → paper*
paper → authors fullPaper? title booktitle
authors → author+

Use of DTDs

- Internal DOCTYPE declaration

```
<?xml version="1.0"?>
<!DOCTYPE bibliography [
    <!ELEMENT bibliography paper*>
    …
]>
<bibliography>…</bibliography>
```

- External DOCTYPE declaration

```
<?xml version="1.0"?>
<!DOCTYPE bibliography SYSTEM "bib.dtd">
<bibliography>…</bibliography>
```

Document Type Definitions (DTDs)

Define and constrain element names and structure.

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title,
                 booktitle)>
<!element authors author+>
<!element author (#PCDATA)>
<!element fullPaper EMPTY>
<!element title (#PCDATA)>
<!element booktitle (#PCDATA)>

<!attlist fullPaper source ENTITY #REQUIRED>
<!attlist paper ID ID>
```

element type declaration

attribute list declaration

Element Declarations

sequence of 0 or more papers

authors followed by optional full paper, followed by title, followed by booktitle

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title,
                 booktitle)>
<!element authors author+>
<!element author (#PCDATA)>
<!attlist author age CDATA>

<!element fullPaper EMPTY>
<!element title (#PCDATA)>
<!element booktitle (#PCDATA)>
<!attlist fullPaper source ENTITY #REQUIRED>
<!attlist paper eid ID>
```

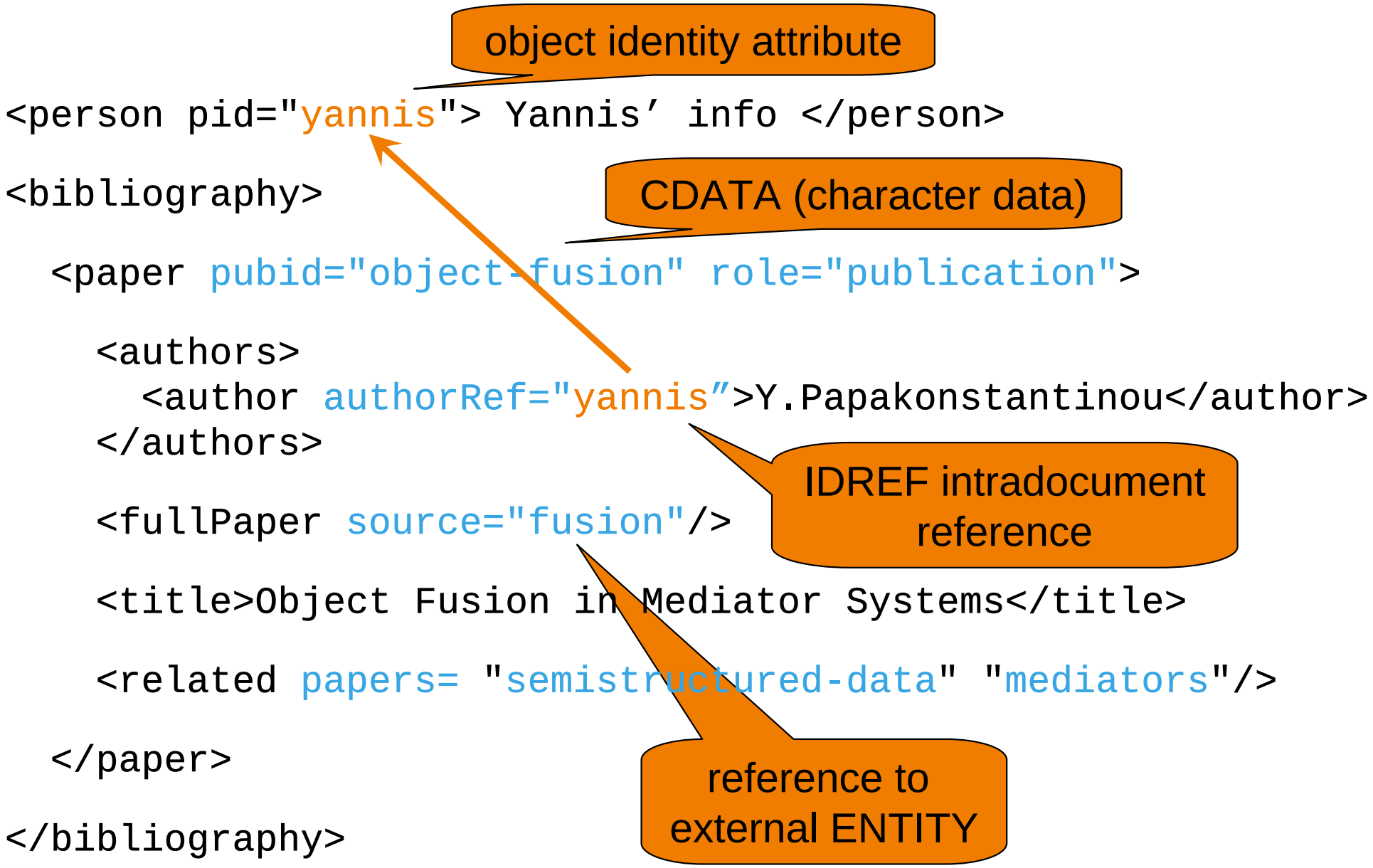
sequence of 1 or more authors

character content

Element Content Declarations

Declaration	*Meaning*
<!element abc>	Exactly one <element abc>
cardinality: R?	Zero or one instances of R
R*	Zero or more instances of R
R+	One or more instances of R
$R_1 \mid R_2 \mid R_3 \mid \ldots R_n$	One instance of R_1 or R_2 or R_3 or …
$R_1 , R_2 , R_3 , \ldots R_n$	Sequence of R's, order matters
#PCDATA	Character content
EMPTY	Empty element
(#PCDATA e*)*	Mixed Content
ANY	Anything goes

Attributes (XML use)



Attribute Declarations

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title,
                 booktitle)>
<!element authors author+>
<!element author (#PCDATA)>

<!element fullPaper EMPTY>
<!element title (#PCDATA)>
<!element booktitle (#PCDATA)>
<!attlist fullPaper source ENTITY #REQUIRED>
<!attlist person pid ID>
<!attlist author authorRef IDREF>
```

reference to an external entity

source (IDREF) and target (ID) declarations for intradocument “pointers”

Attribute Declarations

Type	*Meaning*
ID	**Token unique within the document**
IDREF	**Reference to an ID token**
IDREFS	**Reference to multiple ID tokens**
ENTITY	**External entity (image, video, …)**
ENTITIES	**External entities**
CDATA	**Character data**
NMTOKEN	**Name token**
NMTOKENS	**Name tokens**
NOTATION	**Data other than XML**
Enumeration	**Choices**
Conditional Sec	**INCLUDE & IGNORE declarations**

Attributes may be: REQUIRED, IMPLIED (optional)
can have: default values, which may be FIXED

Adding Structure and Semantics

- XML Document Type Definitions (DTDs)
  - define the structure of valid documents (i.e., *valid wrt. a DTD*)
  - ≈ database schema
- XML Schema
  - defines structure and data types
  - allows developers to build their own libraries of data types
- XML Namespaces
  - identify your vocabulary

From Docs to Data: XML Schema

- XML DTDs (part of the XML spec.)
  - flexible, semi-structured data model (nesting, ANY, ?, *, |, ...)
  - but document-oriented (SGML heritage)
  - no support for namespaces, data types, inheritance
    (e.g., type of *book.title* may be different from *poem.title*)
- XML Schema (W3C, May 2001)
  - schema definition language in XML
  - data-oriented: data types
  - extends capabilities of DTD

XML Schema

- Improvements over DTD
- Every XML Schema is a XML document → no special syntax
- Support for data types
  - many built-in data types
  - support for user-defined data types
- Inheritance and type substitution
- Null values
- Supports XML namespaces
- Extensible for future additions
- Better support for modularization and reuse

XML Schema (Simple Example)

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3schools.com"
    xmlns="http://www.w3schools.com"
    elementFormDefault="qualified">

  <xs:element name="note">

    <xs:complexType>

      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>

    </xs:complexType>

  </xs:element>

</xs:schema>
```

XML Schema – Elements

- Elements can have
  - arbitrary content (structured, unstructured, enumerated, empty, ...)
  - additional characteristics, such as name, type, default, final, minOccurs, maxOccurs, ref, ...
  - a restricted value range

- Examples

```
<element name="birthday" type="xsd:date"/>
<element name="firstname" type="xsd:token"
    minOccurs="1" maxOccurs="unbounded"/>
<element name="pi" type="xsd:double"
    fixed="3.141592" final="#all"/>
```

XML Schema – Attributes

- Attributes
  - carry additional information on elements
  - have additional characteristics, such as name, id, default, fixed, type, ref, use, ...

- Examples

```
<xsd:attribute name=”myAttribute1”/>

<xsd:attribute name=”myAttribute2” type=”xsd:decimal”/>

<xsd:attribute name=”myAttribute3”>
  <xsd:simpleType>
     <xsd:restriction base=”int”>
     <xsd:minInclusive value=”10”/>
     <xsd:maxInclusive value=”20”/>
     </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

XML Schema – Data types

- Simple data types
  - 44 built-in data types, e.g. date, int, float, string, ...
  - No attributes, no elements (= atomic types)
  - Value range may be restricted
- Complex data types
  - May contain several elements and attributes
  - Combination by operators: all, sequence, choice
  - May be defined inside and outside of an element

XML Schema: Example

```
<xsd:element name="choco">
 <xsd:complexType>

  <xsd:element name=“chocolate" maxOccurs="unbounded">
   <xsd:complexType>

    <xsd:attribute name=“number" type="xsd:positiveInteger"
        use="required"/>

    <xsd:sequence>
     <xsd:element name="taste" type="xsd:string"/>
     <xsd:element name="weight" type="xsd:string"/>
     <xsd:element name="best_before" type="xsd:date"/>
     <xsd:element name="price" type="xsd:float"/>
    </xsd:sequence>

   </xsd:complexType>
  </xsd:element>

  …
 </xsd:complexType>
</xsd:element>
```

Valid Document for the Example

```
<?xml version="1.0" encoding="UTF-8"?>

<choco>

  <chocolate number="1">
    <taste>noisette</taste>
    <weight>100g</weight>
    <best_before>2003-01-31</best_before>
    <price>0.79</price>
  </chocolate>

  <chocolate number="2">
    <taste>Lindor</taste>
    <weight>100g</weight>
    <best_before>2002-12-24</best_before>
    <price>1.50</price>
  </chocolate>

</choco>
```

DTD of Example

```
<!ELEMENT choco (chocolate*)>

<!ELEMENT chocolate (taste, weight, best_before, price)>

<!ATTLIST chocolate number CDATA #REQUIRED >

<!ELEMENT taste (#PCDATA)>

<!ELEMENT weight (#PCDATA)>

<!ELEMENT best_before (#PCDATA)>

<!ELEMENT price (#PCDATA)>
```

XML Schema: Derivation of new Types

```
<type name="personName">
  <element name="title" minOccurs="0"/>
  <element name="forename" minOccurs="0" maxOccurs="*"/>
  <element name="surname"/>
</type>

<type name="extendedName" source="personName"
    derivedBy="extension">
  <element name="generation" minOccurs="0"/>
</type>

<type name="simpleName" source="personName“
    derivedBy="restriction">
  <restrictions>
    <element name="title" maxOccurs="0"/>
    <element name="forename" minOccurs="1" maxOccurs="1"/>
  </restrictions>
</type>
```

Derivation of a new type by **extenxion**

Derivation of a new type by **restriction**

Identifying Vocabularies

- My element may not be your element
  - geometry context: `<element>line</element>`
  - chemistry context: `<element>oxygen</element>`
  - …

→ use XML namespaces to identify the vocabulary

XML Namespaces

- mechanism for globally unique tag names

```
<h:html xmlns:xdc="http://www.xml.com/books"
        xmlns:h="http://www.w3.org/HTML/1998/html4">
  <h:head><h:title>Book Review</h:title></h:head>
  ...
  <xdc:bookreview>
    <xdc:title>XML: A Primer</xdc:title>
    ...
  </xdc:bookreview>
</h:html>
```

- namespaces only identify the vocabulary
- additional mechanisms required for structure and meaning of tags

Processing XML

- Non-validating parser
  - checks that XML doc is well-formed
- Validating parser
  - checks that XML doc is also valid with respect to a given DTD

- Parsing an XML document yields tree/object representation
  - Document Object Model (DOM) API
- … or a stream of events (open/close tag, data):
  - Simple API for XML (SAX)

DOM Structure Model and API

- object-oriented approach to traverse the XML node tree
- hierarchy of Node objects
  - document, element, attribute, text, comment, ...
- language independent programming DOM API
  - get... first/last child, prev/next sibling, childNodes
  - insertBefore, replace
  - getElementsByTagName
  - ...
- requires that the whole document is parsed
- memory-intensive

Event-Based SAX API

- does not build a parse tree
- reports events when encountering begin/end tags
- for (partially) parsing large documents
- Pros
  - The whole file does not need to be loaded into memory
  - XML stream processing
  - Simple and fast
  - Allows you to ignore less interesting data
- Cons
  - Limited expressive power (query/update) when working on streams
  - Application needs to build (some) parse-tree when necessary

Transformation von XML Dokumenten

- Häufig müssen XML-Dokumente transformiert werden
  - Beispiel: Umwandlung nach HTML für Darstellung im Browser

- XSLT = eXtensible Style Language Transformations
  - Teil der XSL Spezifikation
  - Transformationssprache mit Elementen von funktionalen Programmiersprachen
  - template-basiert und deklarativ
  - benutzt XPath zur Navigation auf dem Dokumentbaum

XSLT: Beispiel

XML Dokument

```
…
<bibliography>
<book>
<title>Computer
Networks</title>
<author>A.
Tanenbaum</author>
</book>
<book>
<title>Distributed
Systems</title>
<author>G. Couloris</author>
</book>
</bibliography>
…
```

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl=http://… version=…>
<xsl:template match="/">

<html>
<body>
<table border="2">
<tr>
    <th> Titel </th>
    <th> Autor </th>
</tr>
<xsl:for-each select="bibliography/book>
<tr>
    <td> <xsl:value-of select="title"/></td>
    <td> <xsl:value-of select="author"/></td>
</tr>
<xsl:for-each>
```

Generiertes html

Titel	Autor
Computer Networks	A. Tanenbaum
Distributed Systems	G. Couloris

XPATH

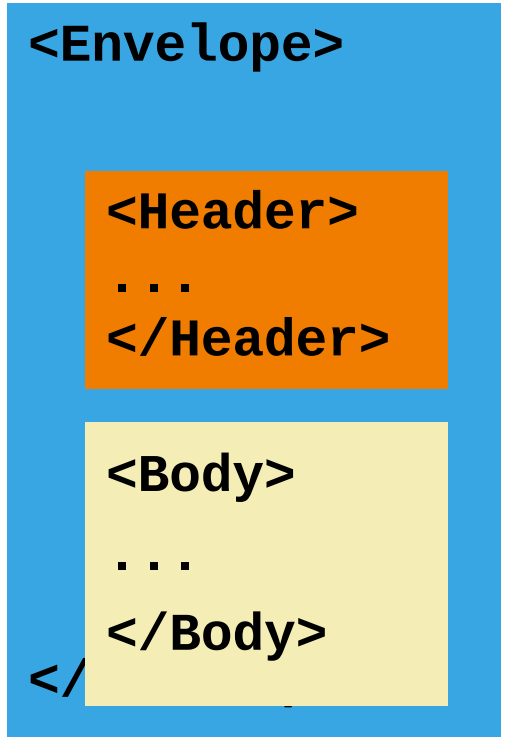
- XPath uses paths to refer to parts of an XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
   <cd country="USA">
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <price>10.90</price>
   </cd>
   <cd country="UK">
      <title>Hide your heart</title>
      <artist>Bonnie Tyler</artist>
      <price>9.90</price>
   </cd>
</catalog>
```

- Relative path: //cd[@country="UK"]
- Absolute path: /catalog/cd[price>10.80]

SOAP Messages

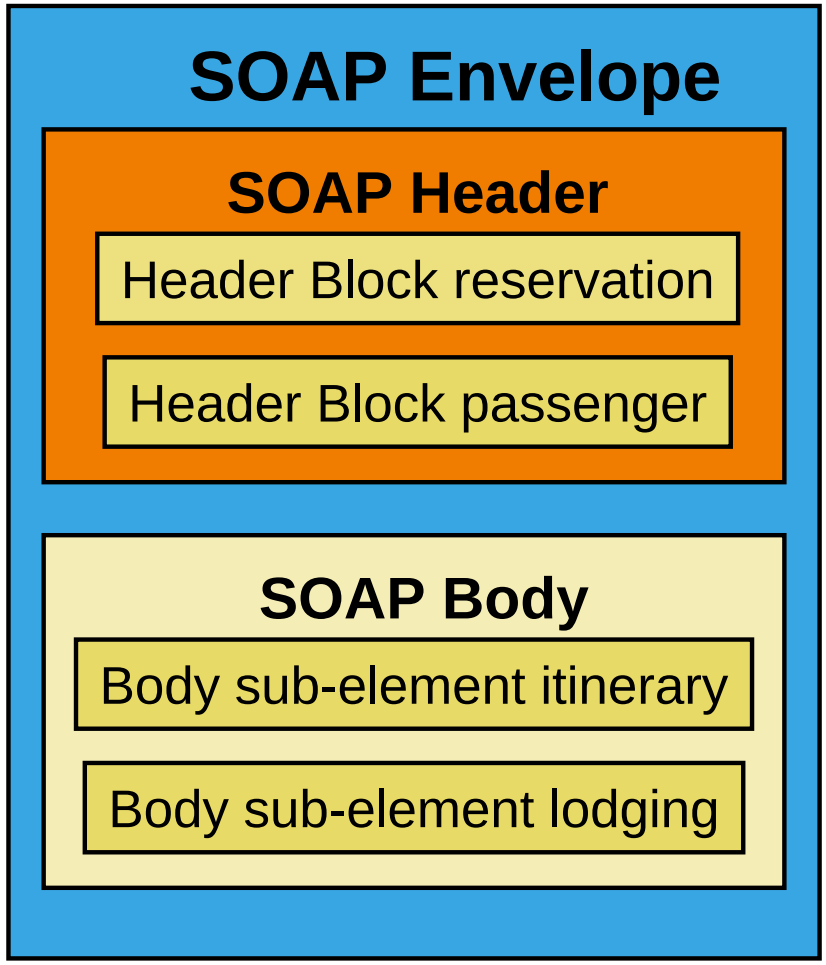
- XML documents having Envelope as root element (→ SOAP Envelope)
- The SOAP Envelope contains an optional Header element (→ SOAP Header) and a mandatory Body element (→ SOAP Body)
- SOAP Header contains additional processing and control information
  (e.g., for security, transactions, accounting, quality of service)
- SOAP Body contains the main application payload



SOAP Messages

- The information in the SOAP header is partitioned into the immediate child elements of the Header element which are called header blocks
- Header blocks can be inspected, modified, inserted, deleted, or forwarded by SOAP nodes on the message path
- The SOAP Body is targeted at the ultimate SOAP receiver
- What data is placed in a header block and what goes in the SOAP body is decided at the time of application design

SOAP Example: Travel Reservation



SOAP Example: Envelope

```
<?xml version='1.0'?>

<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope">

  <env:Header>
  ...
  </env:Header>

  <env:Body>
  ...
  </env:Body>

</env:Envelope>
```

SOAP Example: Header

```
<env:Header>

  <m:reservation xmlns:m="http://travel.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">

    <m:reference>uuid:093a2da1</m:reference>
    <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>

  </m:reservation>

  <n:passenger xmlns:n="http://mycompany.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">

    <n:name>Åke Jógvan Øyvind</n:name>

  </n:passenger>

</env:Header>
```

SOAP Roles (the `role` attribute)

- Header blocks can be targeted at SOAP nodes using the `role` attribute
- A targeted SOAP node is required to process a header block if it assumes the role identified by the value of the URI
- Header blocks without specified role are targeted at the ultimate receiver node
- By default, if a header block is processed, it must be removed from the outbound message; it may, however, be reinserted.

```
<q:oneBlock xmlns:q="http://example.com"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
  ...
</q:oneBlock>
```

SOAP Roles

Header block is targeted to …

- no node on the message path

  `env:role="…/soap-envelope/role/none"`

- the next node on the message paths

  `env:role="…/soap-envelope/role/next"`

- the ultimate receiver of the message

  `env:role="…/soap-envelope/role/ultimateReceiver"`

- every node assuming the application-defined role specified

  `env:role="http://example.com/Log"`

The `mustUnderstand` Attribute

- If set `mustUndestand` to `true`, the header block must be processed by a node that assumes the specified role.

```
<p:oneBlock xmlns:p="http://example.com"
    env:role="http://example.com/Log"
    env:mustUnderstand="true">
  ...
</p:oneBlock>
```

The `relay` Attribute

- When `relay` is set to `true`, a node can forward a header block targeted at itself without processing it.

```
<q:oneBlock xmlns:q="http://example.com"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
    env:relay="true">
  ...
</q:oneBlock>
```

SOAP Example: Body

```
<env:Body>

 <p:itinerary xmlns:p="http://travelcompany.org/reservation/travel">

  <p:departure>
   <p:departing>New York</p:departing>
   <p:arriving>Los Angeles</p:arriving>
   <p:departureDate>2001-12-14</p:departureDate>
  </p:departure>

  <p:return>
   <p:departing>Los Angeles</p:departing>
   <p:arriving>New York</p:arriving>
   <p:departureDate>2001-12-20</p:departureDate>
  </p:return>

 </p:itinerary>

 <q:lodging xmlns:q="http://travelcompany.org/reservation/hotels">
  <q:preference>none</q:preference>
 </q:lodging>

</env:Body>
```

SOAP Communication Styles

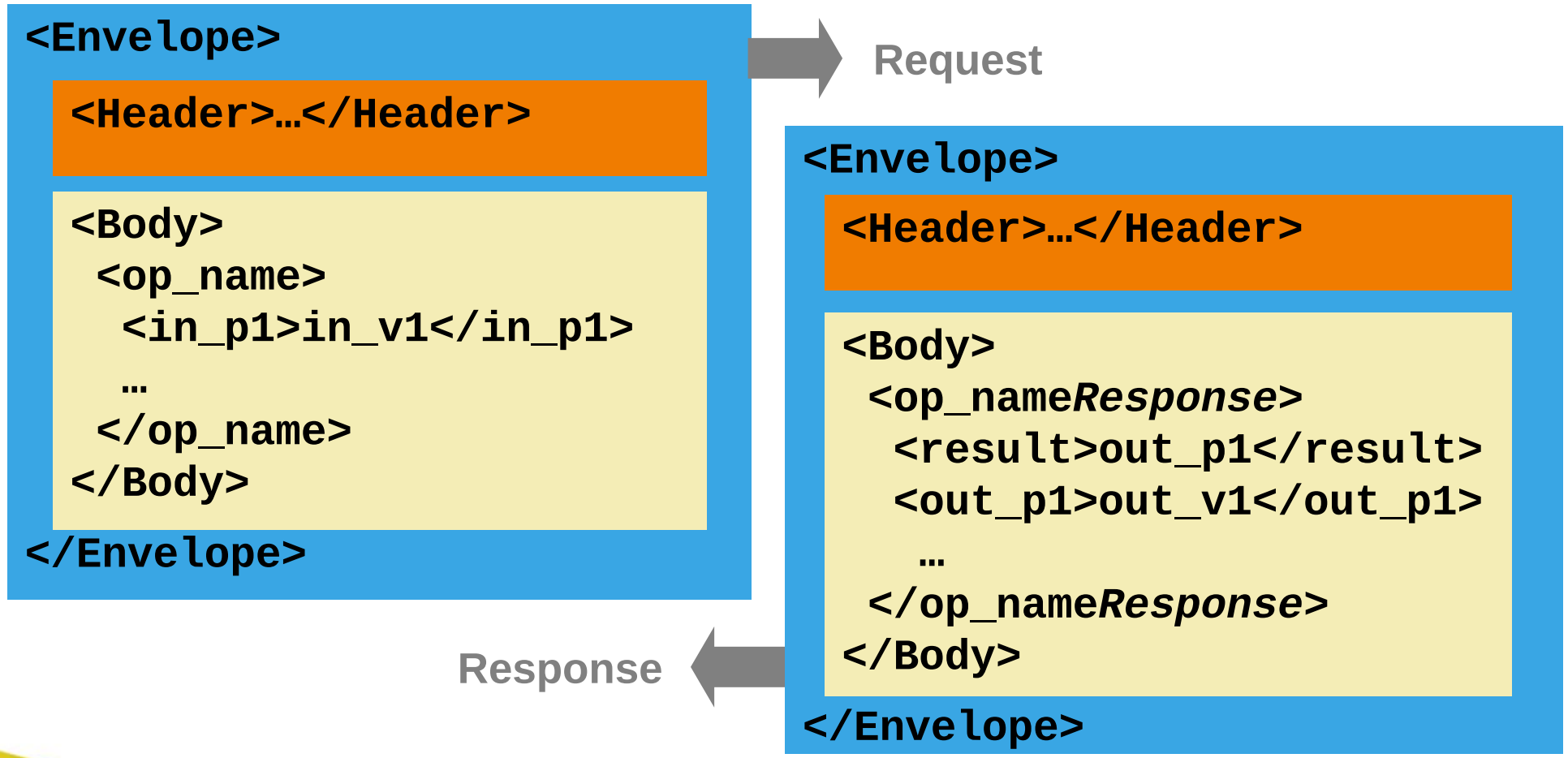
1. Document style
   - SOAP message contains arbitrary XML document

```
<Envelope>
    <Header>
    ...
    </Header>

    <Body>
    “arbitrary XML doc.”
    </Body>
</Envelope>
```

SOAP Communication Styles

2. RPC style

"RPC-structured" request and reply messages



SOAP HTTP Binding

- Two Message Exchange Patterns (MEPs) supported
- HTTP POST: SOAP Request/Response MEP
  - SOAP message in the bodies of the HTTP request *and* the HTTP response
- HTTP GET: SOAP Response MEP
  - No SOAP message in HTTP request
  - SOAP message in body of HTTP response
  - Used for information retrieval where the source is “untouched”
    → safe and idempotent operations according to HTTP RFC

SOAP HTTP GET

Resource identified by URI

```
GET /travelcompany.org/reservations?code=FT35ZBQ HTTP/1.1
Host: travelcompany.org
Accept: application/soap+xml
```

Request

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8“
Content-Length: nnnn

<?xml version='1.0' ?>

<env:Envelope>

  <env:Header>
    ...
  </env:Header>

  <env:Body>
    ...
  </env:Body>

</env:Envelope>
```

Response

SOAP HTTP POST

```
POST /Reservations HTTP/1.1
Host: travelcompany.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>

<env:Envelope>

  <env:Header>
    ...
  </env:Header>

  <env:Body>
    ...
  </env:Body>

</env:Envelope>
```

Request

SOAP over Email (*non-normative* W3C Note)

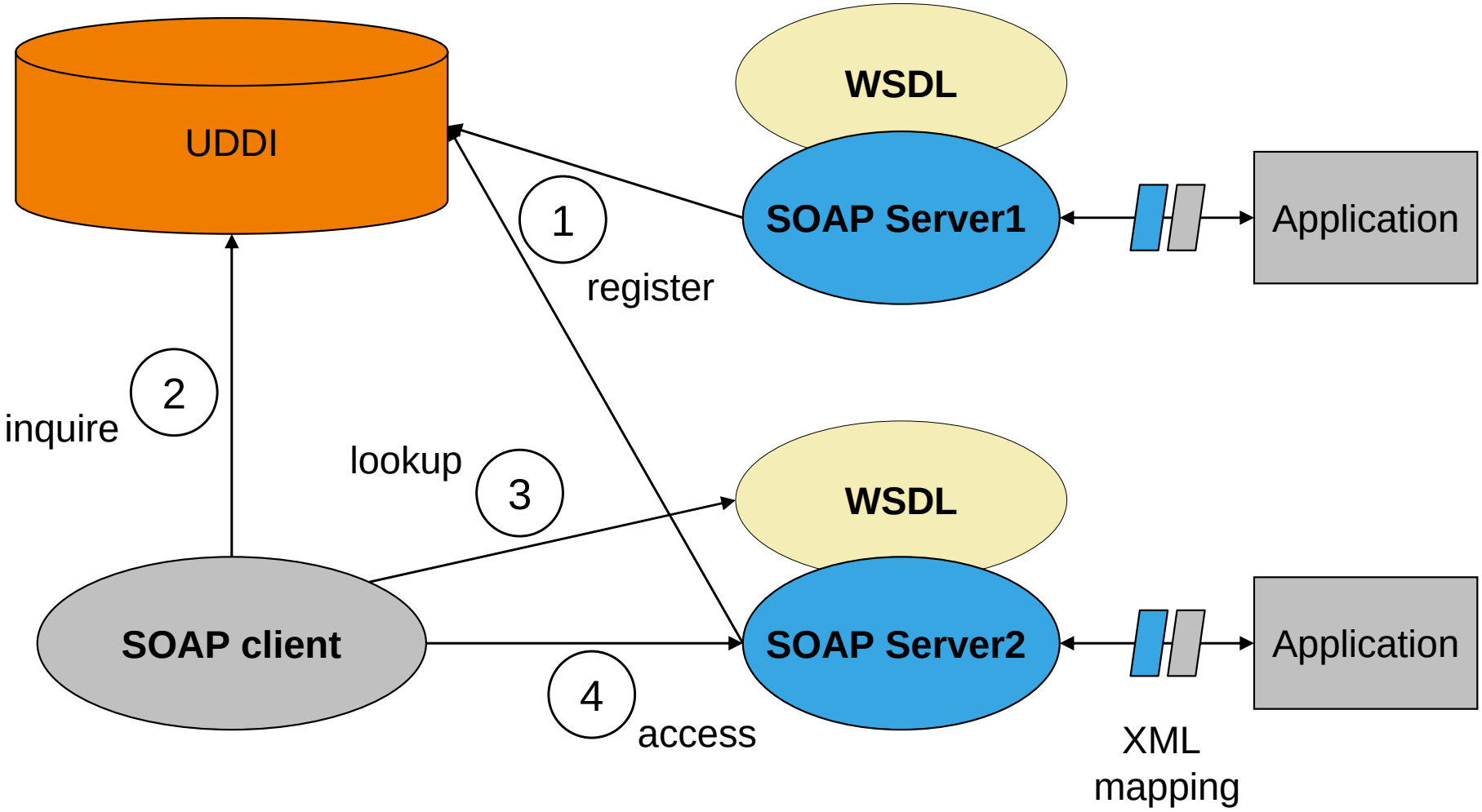
- Application developers can use the Internet email infrastructure to move SOAP messages as either email text or attachments

```
From: a.oyvind@mycompany.com
To: reservations@travelcompany.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id: <EE492E16A090090276D208424960C0C@mycompany.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>

<env:Envelope>
  ...
</env:Envelope>
```

Drawbacks of SOAP

- Security
  - virtually no security facility (firewalls are there for a reason …)
  - custom security measures on top of SOAP → loss of interoperability
  - custom security measures on top of HTTP → admin overhead
  - → Various standardization activities for security on-going
- Efficiency
  - performance loss of factor “3, 7, …” compared to RMI/IIOP (???)
  - mostly lost in coding and parsing payload
- Lack of infrastructure services
  - Transactions, Persistence, Management, Security, …

SOAP, UDDI and WSDL



Web Services Description Language (WSDL)

- XML schema for describing Web Services
  - Abstract service interface definition
    - corresponds to, e.g., CORBA IDL
  - Concrete service implementation definition
    - concrete endpoints and network addresses where Web Service can be invoked
- Core elements of service description
  - what does it do?
  - how can it be invoked?
  - where is it located?

WSDL

Concrete
Abstract

- Service: collection of ports
- Port: communication endpoint defined as a combination of a binding and a network address
- Binding: protocol and data format specification for a particular port type
- Port type: definition of a collection of operations
- Operation: definition of an action supported by a port
- Message: typed definition of data exchanged when executing a operation
- Types: collection of data type definitions using some type system (such as XSD)

WSDL Operation Types

- One-way: The endpoint receives a message.
- Request-response: The endpoint receives a message, and sends a correlated message.
- **Solicit-response**: The endpoint sends a message, and receives a correlated message.
- **Notification**: The endpoint sends a message.
- WSDL only defines bindings for the One-way and Request-response primitives!

WSDL Example: Document Structure

```
<?xml version="1.0"?>

<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
    xmlns:tns="http://example.com/stockquote.wsdl"
    xmlns:xsd1="http://example.com/stockquote.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>...</types>?

  <message>...</message>*

  <portType>...</portType>*

  <binding>...</binding>*

  <service>...</service>*

</definitions>
```

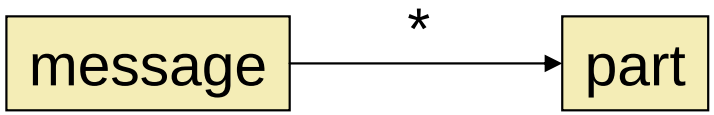
?	0-1 occurrences
*	0-*n* occurrences

WSDL Example: Type Definitions

```
<types>

  <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">

    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>

    <element name="TradePrice">
      <complexType>
        <all>
          <element name="price" type="float"/>
        </all>
      </complexType>
    </element>

  </schema>

</types>
```

Definition of reusable data types
Definition Reference

WSDL Example: Message Definitions

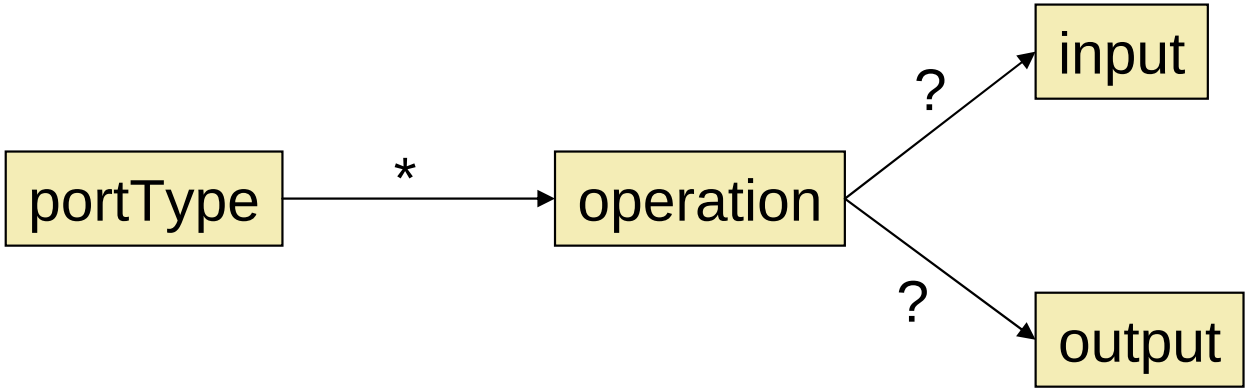



```
<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>
```

Definition of reusable message types

Definition Reference

WSDL Example: Port Type Definition




```
<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>
```

Definition of reusable port types
Definition Reference

WSDL Example: Binding Definition

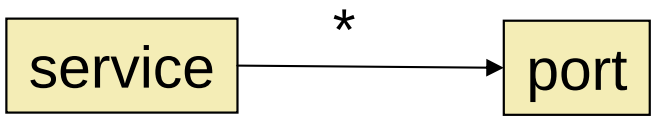
```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document“
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation
          soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use=“literal"/>
      </input>
      <output>
        <soap:body use=“literal"/>
      </output>
      </operation>
</binding>
```

Applied Protocol

Encoding for each operation

Definition Reference

WSDL Example: Service Definition




```
<service name="StockQuoteService">
  <documentation>
    My first service
  </documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
```

Definition Reference

Network address

UDDI

- UDDI = Universal Description, Discovery and Integration
- Repository of business data for the Web http://www.uddi.org
- A UDDI implementation is a Web Service registry providing mechanisms to advertise and discover Web services
- UDDI specification defines a SOAP API along with a WSDL description of the registry service
- Registry contains categorised information about businesses and the services that they offer
- Associates services with technical specifications usually given as WSDL documents

Types of UDDI Registries

- Public
  - A collection of UDDI servers hosted by major IT corporations
  - Anyone can obtain an account and register and search for services
- Protected
  - Groups of companies or other entities create their own UDDI server
  - Limited access for security, privacy, or technical performance
  - Part or all of their contents may be exported to the public directory
- Private
  - Hosted within companies; not part of the public-access business directory
  - Access restricted to internal networks or networks shared between a company and its trusted business partners
- The UDDI Project operates a global public registry called the UDDI Business Registry (UBR)

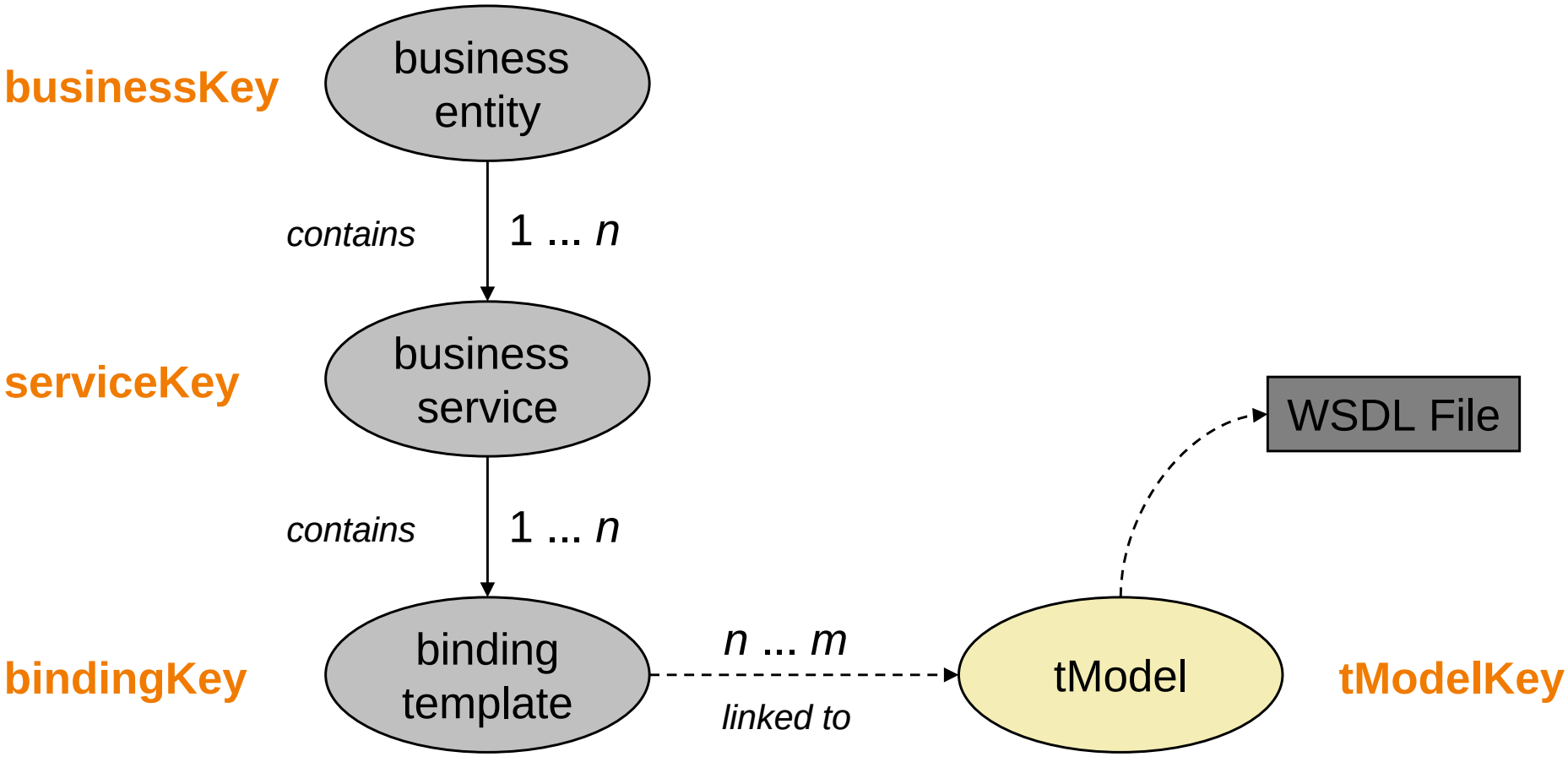
UDDI

White Pages	Yellow Pages	Green Pages
> Business name > Contact information > Human-readable description > Identifiers (taxID, etc.)	> Services and products index > Industry codes > Geographic index	> E-business rules > Service descriptions > Application invocation rules > Data binding
Who?	What?	How?

UDDI Information Model

- A UDDI information model is composed of instances of the following entity types:
  - businessEntity: Contains information about a business including its name, a short description, and some basic contact information.
  - businessService: Describes a collection of related Web services offered by a businessEntity.
  - bindingTemplate: Describes the technical information necessary to find/use a particular Web Service.
  - tModel: Describes a “technical model” representing a reusable concept, such as a Web Service type, a protocol used by Web Services.
  - publisherAssertion: Describes, in the view of one businessEntity, the relationship that the businessEntity has with another businessEntity

Relationship of UDDI Data Structures



CORBA vs. J2EE vs. Web Services

CORBA	J2EE	Web Services
Naming Service Trading Service	JNDI	UDDI
IDL	Java Interfaces	WSDL
GIOP	-	SOAP
IIOP	JRMP	SOAP/HTTP

Bibliography

- W3C. *Extensible Markup Language (XML) 1.1*. Technical Report, Feb. 2004. **http://www.w3.org/TR/xml11/**
- W3C. *Simple Object Access Protocol (SOAP) 1.2*, Technical Report, June 2003. **http://www.w3.org/TR/soap/**
- W3C. *Web Services Description Language (WSDL) 1.1*. Technical Report, Mar. 2001. **http://www.w3.org/TR/wsdl**
- W3C. *XML Information Set (Second Edition)*. Technical Report, Feb. 2004. **http://www.w3.org/TR/xml-infoset/**
- UDDI Spec Technical Committee. *UDDI Version 3.0.1: UDDI Spec Technical Committee Specification*. Technical Report, 2003. **http://uddi.org/pubs/uddi-v3.0.1-20031014.pdf**

Fragen?