



**Forschungszentrum
Informatik**



**Universität
Karlsruhe (TH)**

Datenaustausch



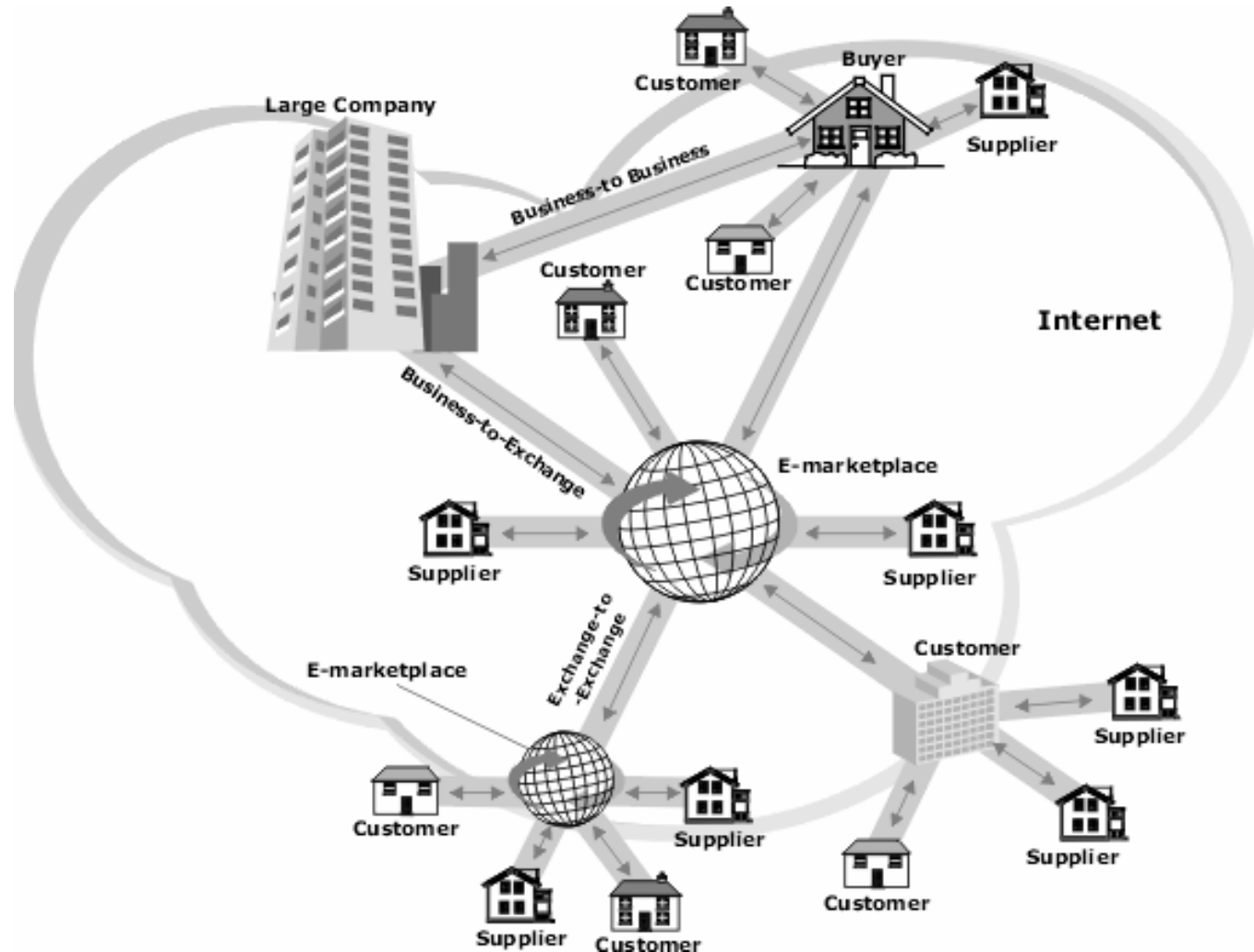
Andreas Schmidt



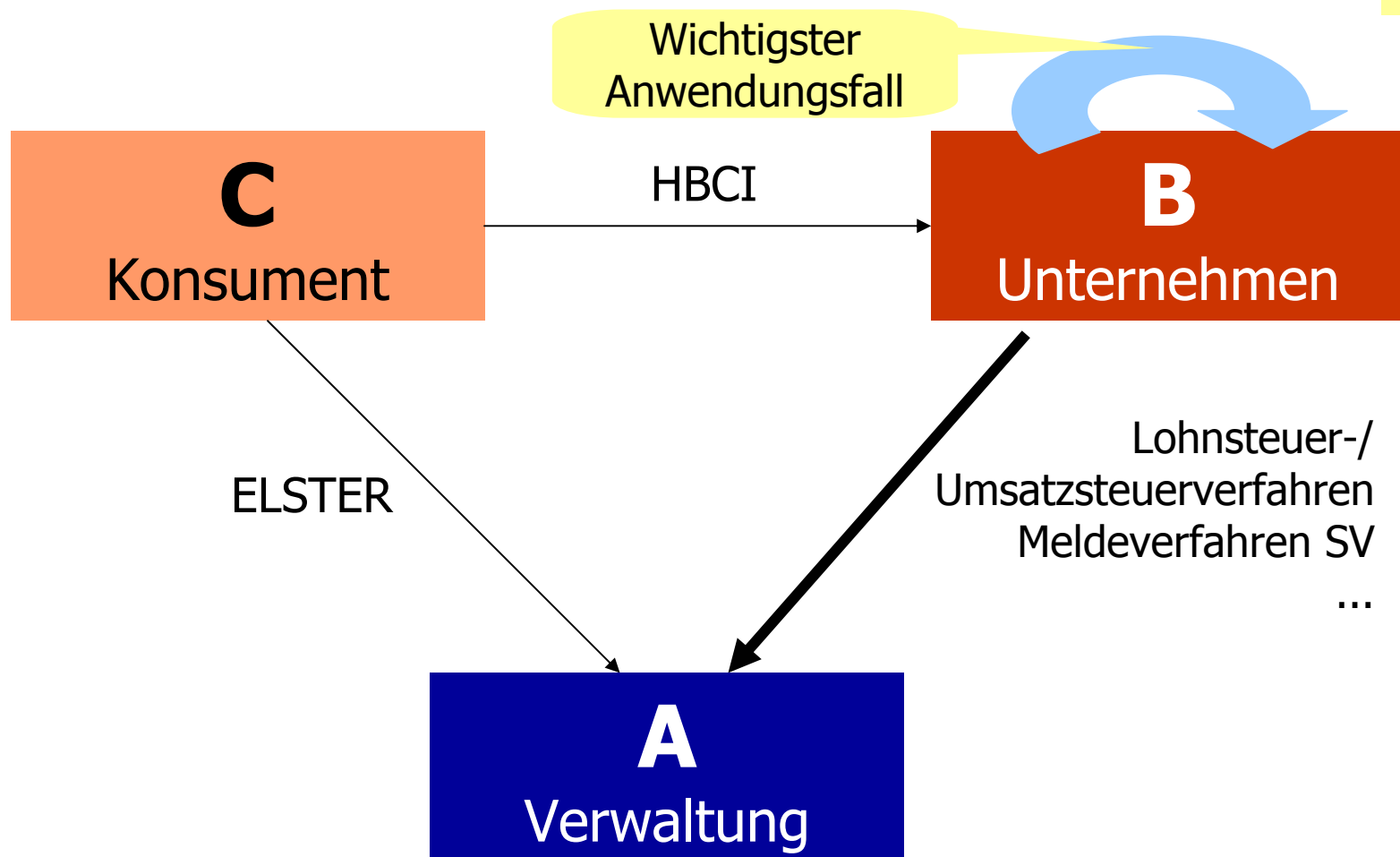
- Datenaustausch
 - ▶ Anforderungen und Probleme
 - ▶ Traditionelle EDI-Lösungen
 - ▶ XML-basierter Datenaustausch
- Validierung als wichtigstes (technisches) Teilproblem
 - ▶ XML Schema
 - ▶ Andere Validierungstechniken für XML

- bisher: Informationsintegration aus Konsumentensicht (B2C)
 - ▶ für eine spezielle (Portal-) Anwendungen werden Informationen aus Informationsquellen extrahiert
 - ▶ Asymmetrie: es existiert eine **zentrale** Stelle, von der aus alles koordiniert wird
 - Ausgangs- und Endpunkt aller Interaktionen
 - ▶ Primäres Interaktionsmuster: Anfrage-Ergebnis
- jetzt: allgemeinerer Ansatz
 - ▶ Symmetrie: **keine vorherrschende Interaktionsrichtung**
 - ▶ weiterhin: Autonomie der technischen Systeme
 - aber teilweiser Verzicht auf organisatorische Autonomie

Datenaustausch: Akteure und Szenarien



Beispiele nach Akteuren



Datenaustausch: Grundlagen



Datenaustausch - Informationsaspekt

- Kern: Datenaustauschformat(e) / -standard(s)
 - ▶ Größtes Problem ist die Heterogenität der ausgetauschten Inhalte
 - ▶ Entwicklung eines gemeinsamen Vokabulars für eine ganze Branche
 - ▶ Muß in einem gemeinsamen Datenmodell und Schema ausgedrückt werden

- Entwicklung des gemeinsamen Vokabulars/Schemas
 - ▶ Modellierungsaufgabe wie bei virtueller Integration
 - ▶ Akzeptanz ist essentiell für den Erfolg
 - ▶ Sehr aufwendig und langwierig

Datenaustausch – Kommunikationsaspekt

- Ebenso wichtig: Festlegen von Übertragungsprotokollen
 - ▶ Technische Übertragungsprotokolle
 - ▶ Aber auch: abstrakte Kommunikationsprotokolle
=> orientieren sich an den Geschäftsprozessen
- Modellierung von übergreifenden Geschäftsprozessen
 - ▶ Bestellung und Rechnungsabwicklung
 - ▶ ...

Was müssen die Partner tun?



■ Informationsebene

- ▶ Bereitstellung der Daten im gewünschten Schema
- ▶ Verarbeitung der ausgetauschten Daten und Einbringung in das eigene System

■ Anwendungs-/Prozeßebene (meist komplexer!)

- ▶ Einbindung der Datenaustauschschritte in die eigenen Geschäftsprozesse
- ▶ Abstimmung der Anwendungslogik

Datenaustausch: Nachrichten

- Komplexere Datenaustauschszzenarien meistens nachrichtenbasiert
 - ▶ oft ist die Interaktion zwischen den Partnern komplexer als nur einfacher Informationsaustausch
 - ▶ zusätzliche Semantik erforderlich (welchen Zweck hat die Übermittlung)
 - Katalogdaten
 - Bestellungen
 - Rechnungen
 - ...
 - ▶ Vorstufe zur dienstbasierten Integration

Datenaustausch vs. Informationsintegrationsansatz

- Generell: drei getrennte Welten
 - ▶ **Datenaustausch:** pragmatischer, aus der Praxis kommender Ansatz zum Austausch von Daten zwischen unabhängigen, aber kooperierenden Einrichtungen
=> Schnittstellen nach außen
 - ▶ **Data Warehousing (materialisierte Integration):** unternehmensinterner Aufbau einer integrierten Datenbasis zur Verbesserung der Auswertung, zur Wissensgewinnung etc., dabei Entkopplung der Auswertung vom operativen Geschäft => Integration im Innern
 - ▶ **Virtuelle Informationsintegration:** Forschungsansatz, der sich auf die Übersetzung von Anfragen und Ergebnissen zur Laufzeit kümmert
=> Integration von externen Informationsquellen

Datenaustausch vs. Informationsintegrationsansatz (2)

- Zusammenwachsen der unterschiedlichen Problembereiche und Lösungen
 - ▶ Enterprise Application Integration
 - ▶ Web-Service-basierte Lösungen
=> dienstorientierte Integration

- Deshalb Schwerpunkt dieser Vorlesung auf den Beitrag des Datenaustauschszenarios
 - ▶ Festlegung von Datenformaten
 - ▶ Validierung von außen empfangener Daten

Klassisches EDI



Austauschformate: EDI

■ Electronic Data Interchange

- ▶ einigermaßen standardisierter nachrichtenbasierter Datenaustausch
- ▶ unterteilt in unterschiedlichen Nachrichtengruppen, Nachrichtentypen, Segmente, Attribute
- ▶ Definition von speziellen Nachrichtentypen, z.B.
 - Angebot
 - Bestellung
 - Rechnung
 - Bezahlung

■ Unterschiedliche Ausprägungen

- ▶ ANSI X.12
- ▶ UN/EDIFACT
(**E**lectronic **D**ata **I**nterchange for **A**dministration, **C**ommerce and **T**ransport)

Beispiel: Rechnung

Absender:

Klick and Bau
Haid-und-Neu-Straße 10-14
76131 Karlsruhe

Empfänger:

Huber GmbH
Obstgasse 2
81549 München

Rechnungsnr. 200301 vom 08.01.2003

Ihre Bestellung O0010001 vom 23.12.2002

1 Bohrmaschine Bosch PSB 500 RE	43,06
Summe netto	43,06
MwSt 16%	6,89
Summe brutto	49,95

Beispiel: Rechnung (2)

UNA:+,?'

Syntaxinformationen

UNB+UNOA:2+KLICKANDBAU+HUBER+20030108:1001+200301081010'

UNH+INVOIC0001+INVOIC:D:93A:UN'

Sender und Empfänger (IDs), Datum

BGM+380+2003001+9'

Nachrichtentyp Rechnung

DTM+3+20030108+102'

Rechnung, Rechnungsnummer, Original
Rechnungsdatum

RFF+ON+00010001'

DTM+4+20021223:102'

Referenz auf Bestellnummer
Bestelldatum

NAD+SE++Klick and Bau++Haid-und-Neu-Str.10-14+Karlsruhe++76131'

NAD+BY++Huber GmbH++Obstgasse 2+München++81549'

Adresse des Rechnungsempfängers

Adresse des Rechnungsstellers

Beispiel: Rechnung (3)

LIN+1++4711.001'	Rechnungsposition 1 mit Artikelnr. 4711.001
IMD+F+++::Bohrmaschine Bosch PSB 500 RE'	Beschreibung
QTY+47:1:PCE'	Berechnete Anzahl 1 (PCE=Stück)
MOA+43,06:750'	Nettopreis der Position
PRI+AAA:43,06'	Einzelpreis der Position
UNS+S'	Summensektion
MOA+79:43,06'	Netto-Gesamtsumme
MOA+124:6,89'	(Mehrwert-)steuer
MOA+128:49,95'	Brutto-Gesamtsumme
TAX+7+VAT+++:::16+S'	Steuerart = MwSt, Satz 16%
UNT+18+INVOIC0001'	Abschluß der Rechnung, #Segmente=18
UNZ+1+200301081010'	Abschluß der Nachricht

- EDIFACT-Lösungen haben eine große praktische Bedeutung, vor allem bei
 - ▶ hohen Transaktionsvolumina
 - ▶ stabilen Beziehungen zwischen den Partnern

- Z.B.
 - ▶ Einsatz in der Automobilbranche (Odette)
 - ▶ Bankenwesen (SWIFT)
 - ▶ Kommunikation im deutschen Gesundheitswesen (§§301,302 SGB V)
 - Krankenkassen, Krankenhäuser, Arbeitgeber, Sonstige Leistungserbringer
 - Mit massiven Einführungsproblemen

Probleme mit X12/EDIFACT

- Für dynamische Umgebungen schlecht geeignet
 - ▶ Zentralisierte Vergabe von Identifikationsnummern
 - ▶ Mangelnde Erweiterbarkeit der Nachrichten
- Fehlende standardisierte und maschinenverarbeitbare Nachrichtenbeschreibungssprache
 - ▶ Keine standardisierten und nachrichtentypunabhängigen Validierungswerkzeuge
 - ▶ Oft ungenaue Spezifikationen
- Schwer zu erlernen
 - ▶ deshalb häufige Probleme mit inkompatiblen Implementierungen
 - ▶ Und: teuer
- Branchenspezifische Lösungen
 - ▶ Keine universelle EDIFACT-Lösung (Odette, SWIFT) wie ursprünglich anvisiert
- Für kleinere Unternehmen oder Ad-Hoc-Geschäftsbeziehungen nicht geeignet

Datenaustausch mit XML



Datenaustausch mit XML

- Definition einer DTD bzw. eines XML Schemas

- Vorteile
 - ▶ Verfügbarkeit generischer Werkzeuge
 - Generierung
 - Validierung
 - Transformation
 - Import
 - Präsentation
 - ▶ selbstbeschreibendes Dateiformat

Datenaustausch mit XML: Probleme



- aber: Effizienz?
 - ▶ Hoher syntaktischer Overhead
 - Preis für die bessere Lesbarkeit
 - ▶ Kompressionstechniken können die Übertragungsmenge reduzieren

- DTDs sind ein stark dokumentenorientierter Ansatz für die Schemadefinition
 - ▶ kontextfreie Grammatiken für Elementstruktur (»Inhaltsmodelle«)
 - ▶ Attributdefinitionen
 - rudimentäre Typen (CDATA, NMTOKEN, ID, ...)
 - optional?
 - Standardwert
 - ▶ Einfacher Schlüsselmechanismus (über ID/IDREF-Attribute)

DTD: Beispiel

```
<!ELEMENT produkte      (produkt*) >
<!ELEMENT produkt      (bezeichnung, hersteller, preis) >
<!ELEMENT preis        #PCDATA>
<!ELEMENT bezeichnung  #PCDATA>
<!ELEMENT hersteller   EMPTY>
<!ELEMENT produzenten  (produzent*)>
<!ELEMENT produzent    (name)>
<!ELEMENT name         #PCDATA>

<!ATTLIST produzent    id      ID>
<!ATTLIST hersteller   id      IDREF>
```

- Typisierung von Attribut- und Elementinhalten
 - ▶ Keine Typisierung von Elementinhalten (Text)
 - ▶ Nur wenig Typen für Attribute
- Strukturelle Beschreibungsmöglichkeiten
 - ▶ Gemeinsamkeiten zwischen unterschiedlichen Elementen
- Unflexibles Schlüssel-/Fremdschlüsselkonzept
 - ▶ ID/IDREF, keine wertbasierten Verknüpfungen
- Eingeschränkte Wiederverwendbarkeit
 - ▶ Mängel bei Dokumentation, Erweiterbarkeit, keine Namensräume
- Eigene Syntax
 - ▶ Verarbeitung von Schemata erschwert

XML Schema



Warum XML Schema?

■ Klassische Datenbanksicht

- ▶ Garantien für die Anwendung über die Struktur der Daten
- ▶ Sicherung der Konsistenz der Datenbasis
- ▶ Freiräume für die physikalische Schicht (Optimierung!)
- ▶ Verzicht auf Schema verlagert die Komplexität in die Anwendung und die Speicherung!

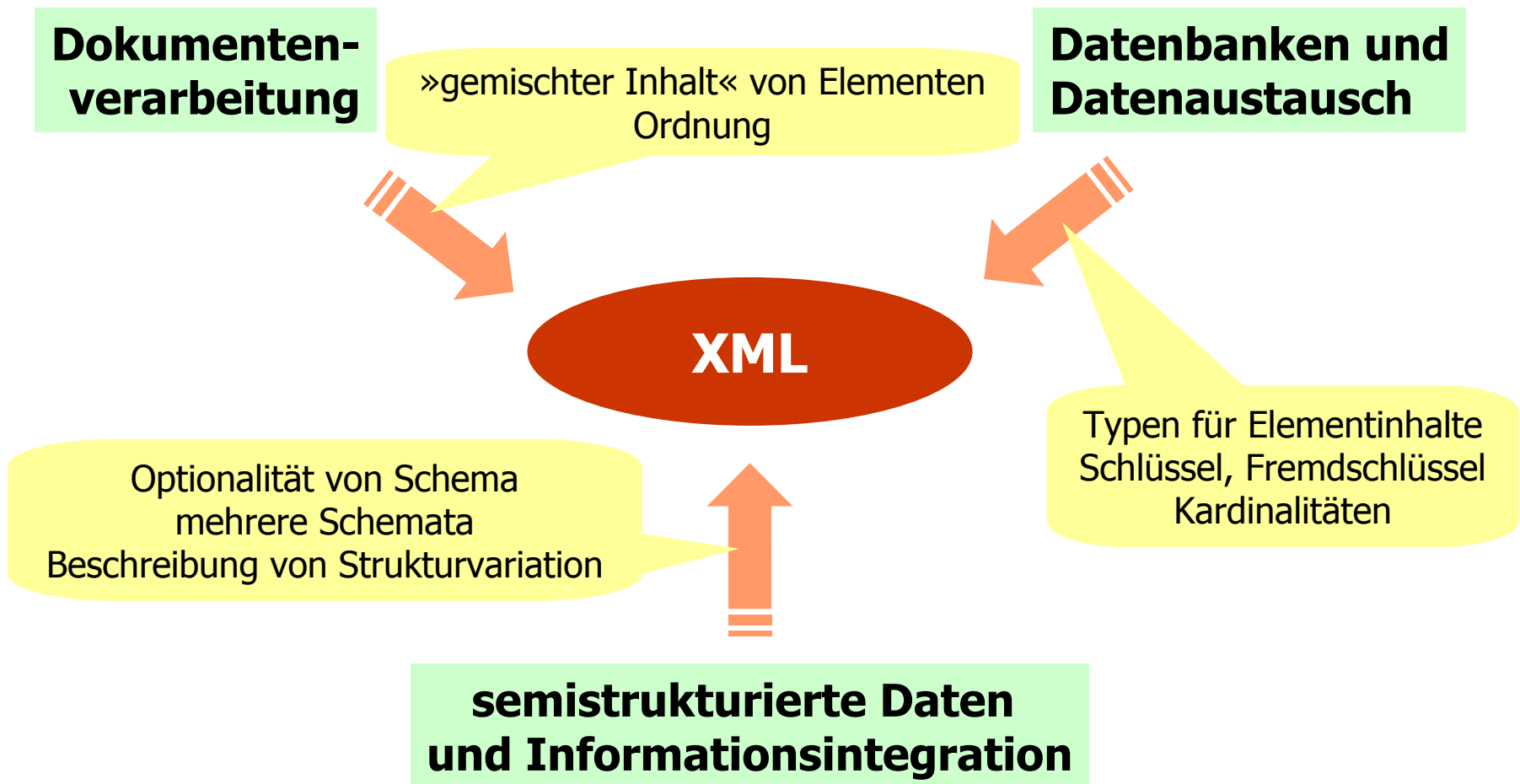
■ Datenaustausch

- ▶ generische Fehler- und Konsistenzprüfung der übermittelten Daten

■ Integration

- ▶ Beschreibungssprache für a posteriori Schemata und sonstige Metainformationen über Struktur und Konsistenz

Schema für XML und die drei Welten

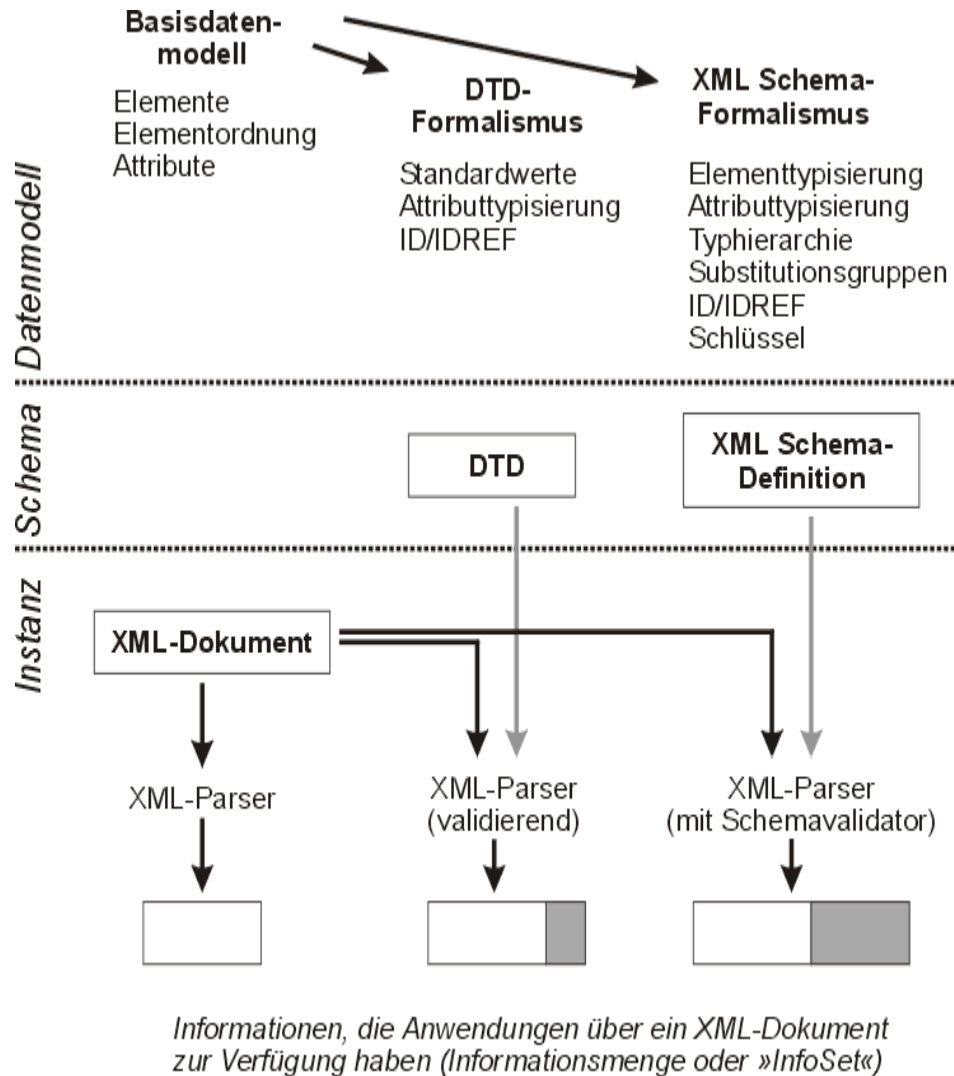


- offizielle XML-Schemadefinitionssprache des W3C
- Seit Mai 2001 Empfehlung (recommendation)
- soll langfristig DTDs ganz ablösen
- Inkorporation der erweiterten Möglichkeiten (Typen, Schlüssel) in andere Standards wie XSL-T oder XPath
 - ▶ XPath 2.0, XSL-T 2.0 und XQuery: November 2003
Last Call Working Draft

XML Schema: Validierung

- Validierungskonzept ähnlich wie bei DTDs
 - ▶ Schemadefinition weiterhin optional
 - ▶ Verknüpfung mit einer Schemadefinition kann flexibel erfolgen
 - direkter Verweis im Instanzdokument
 - Angabe bei der eigentlichen Validierung
- XML Information Set als repräsentationsunabhängige Sicht auf ein XML Dokument
 - ▶ Validierung vergrößert die Informationsmenge, die Anwendungen über ein Dokument zur Verfügung steht
 - ▶ das Vorhandensein einer Schemadefinition entscheidet also über das Datenmodell!

XML Schema: Auswirkung der Validierung



Validierung verändert das Datenmodell und die Informationsmenge, die über ein Dokument verfügbar ist.

XML Schema: Bausteine

Wiederverwendung

Modularisierung, Erweiterbarkeit

Konsistenzregeln

Schlüssel, Fremdschlüssel

Struktur

für komplexe Elemente

Datentypen

für Elementinhalte und Attribute

Typkonzept in XML Schema

■ DTD

- ▶ Elementdeklarationen in DTDs
- ▶ Elemente als Instanzen in Instanzdokumenten

■ XML Schema

- ▶ Typen als weitere Abstraktionsebene
- ▶ Elementdeklarationen als Instanzen eines Typs
- ▶ Elemente als Instanzen einer Elementdeklaration

Typen in XML Schema

- Einfache Typen
 - ▶ Atomare Typen
 - ▶ Listen- und Vereinigungstypen
 - ▶ Bildung neuer Typen

- Komplexe Typen (Strukturteil!)
 - ▶ all, sequence, choice, any
 - ▶ Ableitung von neuen komplexen Typen

- String, boolean
- Numerische Typen
 - ▶ Beschränkter Zahlenvorrat
 - float, double, long, int, byte, unsigned int, ...
 - ▶ Unbeschränkter Zahlenvorrat
 - number, integer
- Datum/Zeit
 - ▶ Zeitpunkte (dateTime, date, gYearMonth, gYear, ...)
 - ▶ Wiederkehrende Daten (time, gMonthDay, gMonth, ...)
 - ▶ Zeitdauer (duration)
- Binärdaten
- XML-spezifische Typen

Ableitung neuer atomarer Typen

- Einschränkung existierender atomarer Typen
 - ▶ Datentypen sind Eigenschaften (facets) zugeordnet, die eingeschränkt werden können, z.B.
 - ▶ Aufzählung von möglichen Werte `<xsd:enumeration>`
 - ▶ Strings:
 - Zeichenanzahl
 - reguläre Ausdrücke als Muster `<xsd:pattern>`
 - ▶ Numerische Typen
 - `minInclusive`, `maxInclusive`

Beispiel für atomare Typen

```
<simpleType name="währung">  
  <restriction base="decimal">  
    <minInclusive value="0" />  
    <fractionDigits value="2" />  
  </restriction>  
</simpleType>
```

```
<simpleType name="kategorieTyp">  
  <restriction base="string">  
    <enumeration value="Bohrmaschinen" />  
    <enumeration value="Akku-Schrauber" />  
    ...  
  </restriction>  
</simpleType>
```

Listen- und Vereinigungstypen

■ Listentyp

- ▶ durch Leerraum getrennte Einzelwerte eines bestimmten Typs
- ▶ z.B. Liste von Zahlen „1 2 4 838“

■ Vereinigungstypen

- ▶ Vereinigung der Wertebereiche von zwei verschiedenen Typen
- ▶ geeignet für das Hinzufügen von Sonderelementen zu numerischen Wertebereichen o.ä.

Struktur: Konstruktoren

- sequence
 - ▶ geordnete Folge von Elementen
- all
 - ▶ beliebige Reihenfolge
 - ▶ AND-Konnektor von SGML kehrt zurück!
 - ▶ Einschränkungen hinsichtlich der Unterstruktur
- choice
 - ▶ entweder - oder
- any
 - ▶ namensraumbezogene Erweiterung um beliebige Elemente

Struktur: Kardinalitäten

- In DTDs war nur möglich
 - ▶ optional ja/nein ?, *
 - ▶ wiederholbar ja/nein +, *

- XML-Schema unterstützt nun auch allgemeine Kardinalitäten
 - ▶ minOccurs, maxOccurs

Beispiel für komplexe Typen

```
<complexType name="produkttyp">
  <sequence>
    <element name="bezeichnung" type="string" minOccurs="1" />
    <element name="beschreibung" type="string"
      minOccurs="0" maxOccurs="1" />
    <element name="hersteller" type="string" minOccurs="1" />
    <element name="preis" type="währung" minOccurs="1" />
    <element name="kategorie" type="kategorieTyp"
      minOccurs="1" maxOccurs="3" />
  </sequence>
</complexType>
```

Ableitung komplexer Typen



■ Restriktion

- ▶ Verschärfung von Kardinalitäten
- ▶ Typisierung von noch nicht typisierten Elementen

■ Erweiterung

- ▶ Hinzufügen weiterer Elemente ans Ende der Unterelementliste

Zusammenfassung Typen: Typhierarchie

- strenge Typhierarchie
 - ▶ keine Mehrfachableitung
 - ▶ Einschränkung der Ableitungsmöglichkeit (finale Typen)
 - ▶ Möglichkeit der Definition abstrakter Typen
- **Einfache Typen** dienen der Typisierung von Element- und Attributinhalt
- **Komplexe Typen** definieren die Unterstruktur von Elementen

Strukturierungsmittel auf Elementebene

- lokale vs. globale Elementdeklarationen
 - ▶ in DTDs sind alle Elementdeklarationen global, d.h. in jedem Kontext verwendbar
 - ▶ in XML-Schema sind auch lokale Deklarationen innerhalb eines bestimmten Elementkontextes möglich
 - ▶ gleich benannte Elemente in unterschiedlichen Kontexten können unterschiedliche Strukturvorgaben haben!
- Substitutionsgruppen
 - ▶ wo ein bestimmtes Element (Kopfelement) angegeben ist, können alle Mitglieder der entsprechende Substitutionsgruppe eingesetzt werden
- Elementgruppen als Ersatz für Parameterentitäten

- **Substituierbarkeit auf Typebene**
 - ▶ die Ableitungsbeziehung (sowohl Restriktion als auch Extension) hat zur Folge, daß die abgeleiteten Typen überall dort verwendet werden dürfen, wo die Obertypen auch verwendet werden dürfen
- **Substituierbarkeit auf Elementebene**
 - ▶ zusätzlich kann für Elemente durch Substitutionsgruppen eine Substituierbarkeit begründet werden
 - ▶ Typ des Kopfes und der Mitglieder einer Substitutionsgruppe müssen in einer Ableitungsbeziehung zueinander stehen!

Konsistenzregeln



- Nullwerte
 - ▶ Unterschied zwischen `xsd:nil="true"` und leerem Element
- Eindeutigkeit (unique)
- Schlüsselbedingung (key)
- Fremdschlüsselbedingung (keyref)

■ select

- ▶ definiert die Menge, für deren Elemente ein Schlüssel (bzw. eine Eindeutigkeitsbedingung) definiert werden soll (*scope*)

■ field

- ▶ definiert die Felder, die zum Schlüssel gehören
- ▶ nur die Kind-Achse von XPath ist erlaubt
 - keine relativen Schlüssel durch Hinzunahme »übergeordneter« Werte möglich

Schlüsselkonzept: Beispiel

```
<key name="A">
```

```
  <selector xpath="//hersteller" />
```

```
  <field xpath="name" />
```

```
</key>
```

```
<keyref name="B" refer="A">
```

```
  <selector xpath="//produkte" />
```

```
  <field xpath="hersteller" />
```

```
</keyref>
```

```
<herstellerliste>
```

```
  <hersteller>
```

```
    <name>Bosch</name>
```

```
    ...
```

```
  </hersteller>
```

```
</herstellerliste>
```

```
<produkte>
```

```
  <produkt>
```

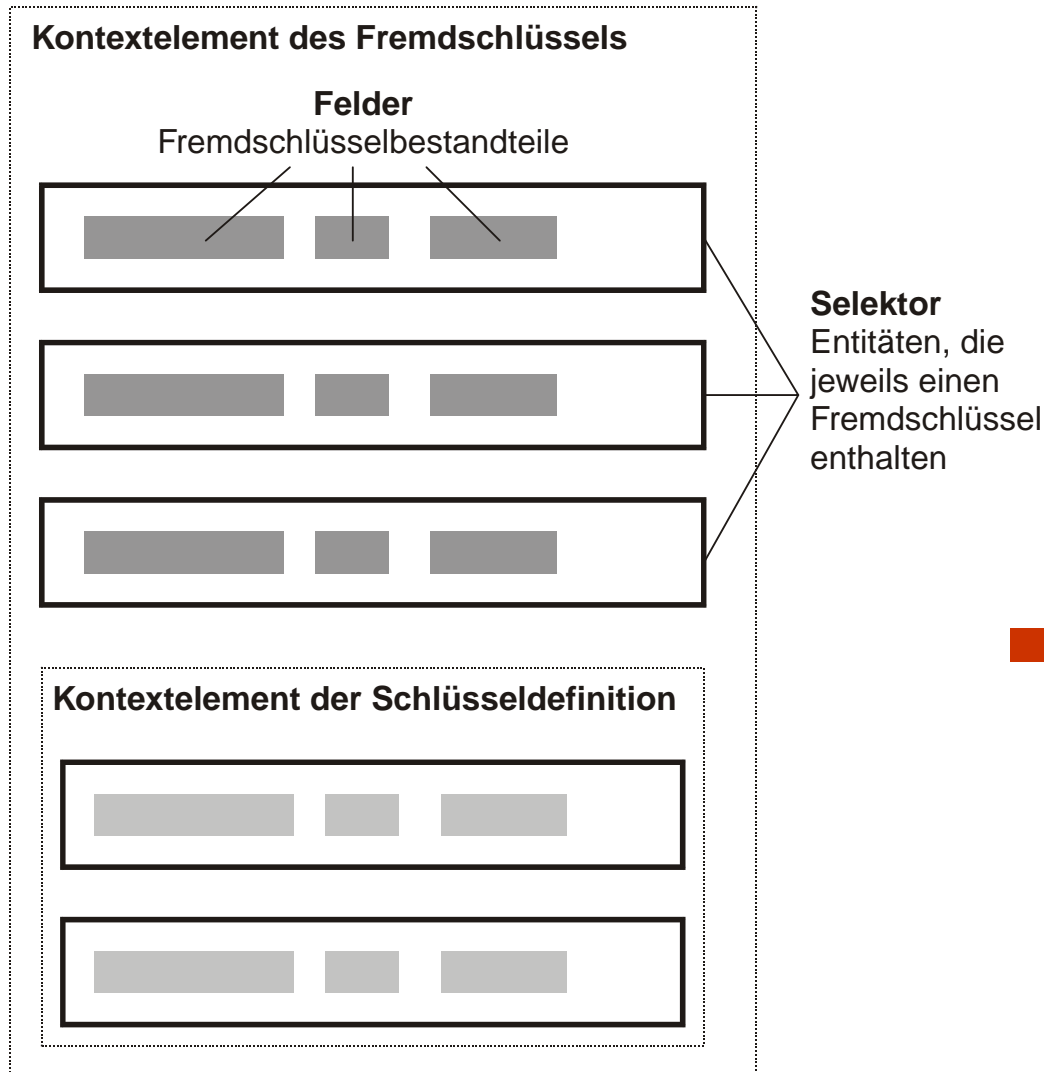
```
    <name>...</name>
```

```
    <hersteller>Bosch</hersteller>
```

```
  </produkt>
```

```
</produkte>
```

Schlüsselkonzept: Kontextelement



- Rel. Datenbanksicht
 - ▶ Kontext für Fremdschlüssel ist Datenbasis
 - ▶ Schlüsselkontext ist die Relation

Schlüssel und Datenaustausch

- Das Schlüsselkonzept von XML Schema ist nur eingeschränkt für die Datenaustauschproblematik zu gebrauchen:
 - ▶ Schlüssel sind auf das Dokument als Einheit beschränkt
 - ▶ Validierung von Problemen wie "Artikelnummer in der Bestellung muß einen im Katalog enthaltenen Artikel referenzieren"
 - kann nur bei versuchtem Einbringen in eine Datenbasis kontrolliert werden
 - Keine Validierungsmöglichkeit bei Ankunft der Bestellung
 - ▶ Vorschläge, hierfür XLink zu benutzen
 - Aber: kaum Werkzeuge, eher ein Exot

Wiederverwendung

- Schemadokumentation
 - ▶ `<xsd:documentation>`
 - ▶ menschenlesbare Dokumentation
- Erweiterung
 - ▶ `<xsd:appinfo>` weiterverarbeitbare Zusatzinformation
 - ▶ beliebige Elemente und Attribute aus anderen Namensräume können in die Schemadefinition eingebaut werden
- Modularisierung
 - ▶ `<xsd:include>` Einbinden von Schemafragmenten
 - ▶ `<xsd:redefine>` Einbinden und Umdefinieren
 - ▶ `<xsd:import>` Einbinden eines fremden Namensraumes

Bewertung von XML Schema

- sehr mächtiger Schemaformalismus für XML
 - ▶ fügt flexible Typisierung von Elementinhalten hinzu
 - ▶ erhöht Wiederverwendung

- aber: recht komplexer Standard
 - ▶ insbesondere konkurrierende Ausdrucksmittel
 - ▶ nicht mehr so einfach zugänglich
 - sowohl für Implementierungen
 - als auch für Anwendung

Andere Schemaansätze für XML



Andere Schemaansätze

■ RELAX NG

- ▶ Entstanden aus TREX und RELAX
- ▶ Übernahme des Datentypenteils von XML-Schema
- ▶ Vereinfachung des Strukturteils
- ▶ Außerdem: kompakte Notation
- ▶ inzwischen ISO-Standard

■ Schematron

- ▶ basiert auf Zusicherungen
- ▶ kann Fehlerberichte einfach erzeugen
- ▶ soll von der ISO standardisiert werden
- ▶ versteht sich als orthogonal zu anderen Schemaansätzen

- Regelparadigma statt Grammatikorientierung
 - ▶ Strukturen werden nicht »erlaubt« durch das Angeben von Produktionsregeln einer Grammatik,
 - ▶ sondern beschränkt durch Validierungsregeln, die erfüllt sein müssen
- Grundelement von Schematron ist daher die Zusicherung (assertion)
 - ▶ als Argument ein XPath-Test
 - ▶ Verknüpfung mit Berichtstexten für Fehlerberichte
- Kann vollständig mittels XSL-T realisiert werden
 - ▶ einfach und kompakte Validierungssprache

Schematron: Beispiel

```
<schema>
  <pattern name="Produkte überprüfen">
    <rule context="produkt">
      <assert test="bezeichnung">Produktbezeichnung fehlt</assert>
      <assert test="//herstellerliste/hersteller[name=current()/hersteller]">
        Hersteller nicht in der Herstellerliste</assert>
      <assert test="floor(lagerbestand) = number(lagerbestand)">
        Anzahl nicht ganzzahlig</assert>
    </rule>
  </pattern>
</schema>
```

- Es lassen sich auch positive Prüfergebnisse ausgeben
 - ▶ report-Element
- Schematron will andere Schemaansätze nicht ersetzen, sondern ergänzen
 - ▶ Einbettung in <appinfo> von XML Schema
 - ▶ <xsd:annotation>
 - <xsd:appinfo>
 - <sch:pattern name="...">
 - <sch:rule context="d:Demo">
 - </xsd:appinfo>
 - </xsd:annotation>
 - ▶ Analoge Einbettung in RELAX NG Schemata

Schematron: Bewertung

- Wachsende Beliebtheit von Schematron
 - ▶ Zahlreiche Validatoren sind bereits verfügbar
- Sehr einfach
- Hervorragend geeignet für Validierungsberichte
- Mögliche Perspektive
 - ▶ Formulierung der Grobstruktur mit XML Schema unter Verzicht zu starker Einschränkungen
 - ▶ Formulierung zusätzlicher Einschränkungen über Schematron

Entwurf von XML Schemata

- XML-Schema ist keine Modellierungssprache (konzeptuelle Ebene), sondern eine Schemadefinitionssprache (logische Ebene)
- Deshalb: idealerweise Modellierung mit konzeptuellen Modellen wie z.B. ER oder UML
 - ▶ dann Übersetzung in XML Schemata
 - ▶ konkrete Übersetzung stellt eine grundsätzliche Entwurfsentscheidung dar

Fazit



Zusammenfassung

- XML hat sich im Bereich des Datenaustausches für neue Projekte als die Technologie der Wahl herauskristalliert
 - ▶ standardisierte Spezifikations Sprachen für Schemata
 - ▶ generische Werkzeuge (davon viele frei verfügbar) zu ihrer Verarbeitung
 - insbesondere Kombination mit Transformation
 - ▶ (leichte Integration/Migration in Web Service-Infrastrukturen)

- aber: XML ist nicht die Lösung aller Probleme
 - ▶ Verdrängt auf absehbare Zeit klassische EDI-Lösungen nicht
 - ▶ Die Verwendung von XML macht nicht die Systeme von sich aus interoperabel!
 - ▶ Insbesondere die Entwicklung der Vokabulare ist schwierig
=> Ontologieproblematik

Weiteres Programm



- Dienstorientierte Sicht auf die Integrationsproblematik
 - ▶ nächste Vorlesung:
Dienstorientierte Integration mit Web Services
- Ontologien als Werkzeug für die Informationsintegration
 - ▶ konzeptuelle Modelle
 - ▶ Logische Schlußfolgerungen

- Wassili Kazakos, Andreas Schmidt, Peter Tomczyk:
Datenbanken und XML, Springer, März 2002
 - ▶ Kapitel 4
- W3C <http://www.w3c.org>
- RELAX-NG
 - ▶ <http://www.relaxng.org/>
- Schematron
 - ▶ <http://www.ascc.net/xml/resource/schematron/>