Repräsentation von Struktur

Übersicht

- Einführung
- Document Type Definitions
- XML Schema
- Weitere Strukturbeschreibungssprachen

Einführung

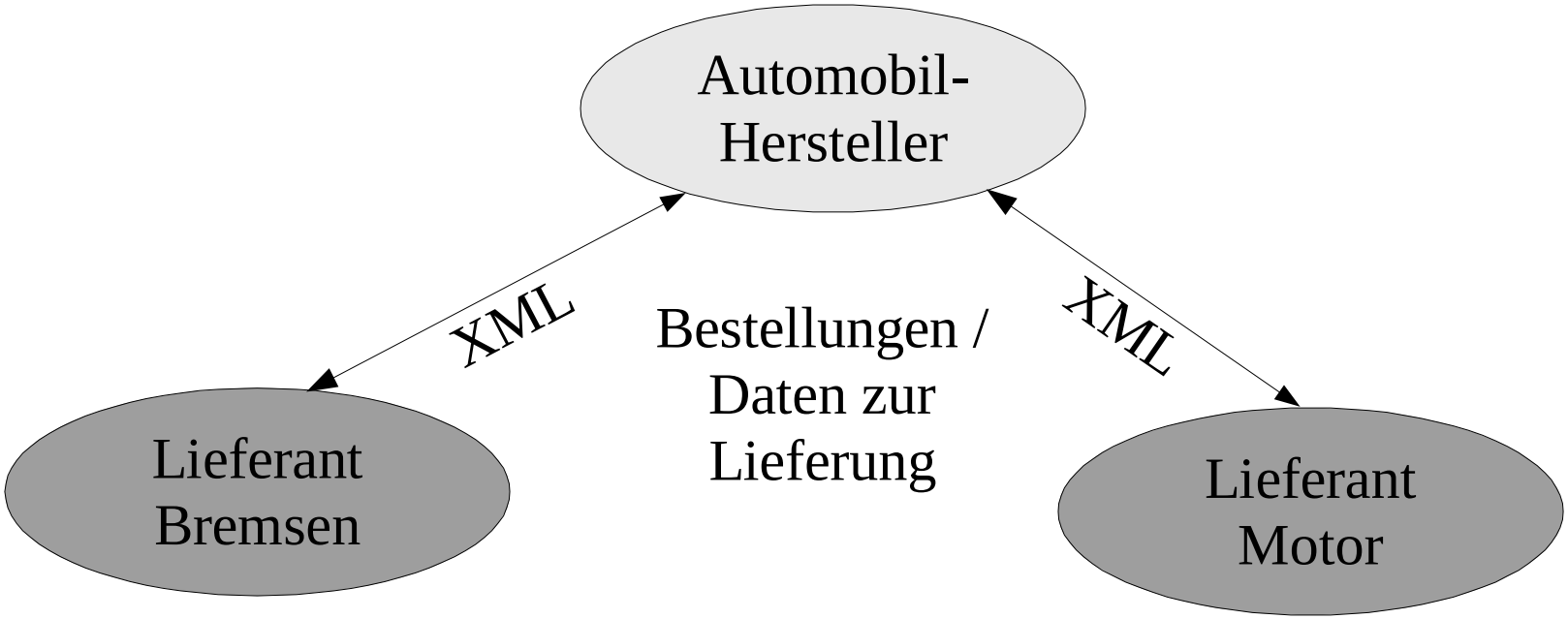
- Was machen Strukturbeschreibungen ?
  - Ziel: Definition einer Menge von erwünschten XML Dokumenten
  - Kriterien dazu
    - Wohlgeformtheit
      - Nur ein Wurzelelement
      - Jedes öffnende Tag wird auf der gleichen Ebene wieder geschlossen
    - Gültigkeit
      - Dokument hält eine beschriebene Struktur ein
      - Thema dieses Vortrags

Einführung

- Mögliche Festlegungen für die Struktur
  - Reihenfolge der Elemente
    - Schachtelung und Inhalt (Typ)
  - Attribute
    - Namen und Typen
- Zusätzliche Features
  - Referenzen, Textersetzung ...

Einführung

- Nutzen der Festlegung von Struktur
  - Beispiel: Datenaustausch im B2B-Umfeld



Einführung

- Nutzen der Festlegung von Struktur
  - Formaler “Vertrag” über Schnittstelle
  - Überprüfung der Struktur nicht in Applikation sondern extern
    - Zeit- und Kostenersparnis
  - Branchenspezifische Strukturbeschreibungen möglich
    - z.B. in den USA: Steuerformulare

Einführung

- Beispiel:

  - Bestellung besteht aus:
    - Teilespezifischen Informationen
      - Teilnummer
      - Stückzahl
      - Lieferdatum
    - Zielort (Werk oder Zwischenlager)
    - Anmerkung (optional)

Einführung

- Beispiel einer Bestellung

```
<order date=”2002-12-01”>
   <part pnumber=”4711”>
      <quantity>10</quantity>
      <duedate>2002-12-03</duedate>
   </part>
   <part> .... </part>
   <destination>
      <interimstorage>KL</interimstorage>
   </destination>
   <note>
      Please don't deliver broken parts
    </note>
</order>
```

Document Type Definitions

- In XML 1.0 spezifiziert
- Ursprung: SGML
  - Syntax kein XML
  - Dokumentzentriert
- Ersetzungen in Dokumenten möglich
- Weit verbreitet aber: Nicht mehr “state of the art”

Document Type Definitions

- Definition von Elementen
  - Syntax: <!ELEMENT name (Inhaltsmodell) >
  - Inhaltsmodell
    - #PCDATA – Element enthält nur Text
    - element1, element2, element3 – Sequenz aus Elementen
    - element1 | element2 | element3 – Auswahl aus Elementen
    - Multiplizitäten durch Anhängen von Zeichen an Elementnamen
      - keines: 1 Vorkommen
      - ? : 0-1 Vorkommen
      - \+ : 1-n Vorkommen
      - * : 0-n Vorkommen

Document Type Definitions

- Elementdeklarationen

```
<!ELEMENT order(part+, destination, note?) >
<!ATTLIST order
            date CDATA #REQUIRED >

<!ELEMENT part (quantity, duedate)>
<!ATTLIST part
            pnumber CDATA #REQUIRED >

<!ELEMENT destination (factory | interimstorage)>

<!ELEMENT factory (#PCDATA)>
<!ELEMENT interimstorage (#PCDATA)>
 ....
```

Document Type Definitions

- Definition von Attributen
  - Syntax:
    <!ATTLIST elementName
    attrName1 attrTyp1 attrVorgabe1 attrStandardwert1
    attrName2 attrTyp2 attrVorgabe2 attrStandardwert2
    .... >
    - attrTyp: Text, Wertaufzählungen und Verweise möglich
    - attrVorgabe: #REQUIRED, #IMPLIED, #FIXED, Stdwert
    - attrStandardwert

Document Type Definitions

- Attributdeklarationen

```
<!ELEMENT order(part+, destination, note?) >
<!ATTLIST order
            date CDATA #REQUIRED >

<!ELEMENT part (quantity, duedate)>
<!ATTLIST part
            pnumber CDATA #REQUIRED >

<!ELEMENT destination (factory | interimstorage)>

<!ELEMENT factory (#PCDATA)>
<!ELEMENT hall (#PCDATA)>
 ....
```

Document Type Definitions

- Weitergehende Definitionen: Entities
  - Ersetzung in XML Dokumenten und DTDs
  - Ähnlich C-Präcompiler Makros
  - Vorgehen
    - 1. Definiere “Makro” im DTD: <!ENTITY name “inhalt”>
    - 2. Referenziere “Makro”:
      - Im XML Dokument: “&name;”
      - Im DTD: “%name;”

Document Type Definitions

- Grenzen von DTDs
  - Sehr schwache Typisierung
    - Quasi nur “String” möglich
  - Anzahl der DTDs pro Dokument auf eins beschränkt
    - Umgehung möglich aber unsauber
  - Fehlende Unterstützung von Abstraktionskonzepten (z.B. Vererbung)
  - Fehlende Wirksamkeitsgarantie von DTDs
    - Regeln sind überschreibbar

XML Schema

- Designierter Nachfolger von DTDs
  - Seit dem 02.05.01 W3C Recommendation
- Syntax ist XML-konform
- Räumt mit vielen Problemen von DTDs auf
  - Starke Typisierung
  - Unterstützt nachträgliche Erweiterungen
- Aber: Hohe Komplexität => Hier nur grob vorgestellt

XML Schema

- Wie sieht ein XML Schema Dokument aus ?

```
<xsd:element name=”order”>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name=”part” type=”partType” minoccurs=”1”
                                                  maxoccurs=”unbounded”/>
         <xsd:element name=”destination” type=”destType”/>
         <xsd:element name=”note” type=”noteType” minoccurs=”0”/>
      </xsd:sequence>
      <xsd:attribute name=”date” use=”required” type=”xsd:date”>
   </xsd:complexType>
</xsd:element>
<xsd:complexType name=”partType”> .... </xsd:complexType>
  ....
```

XML Schema

- Elementdeklarationen

```
<xsd:element name=”order”>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name=”part” type=”partType” minoccurs=”1”
                                                  maxoccurs=”unbounded”/>
         <xsd:element name=”destination” type=”destType”/>
         <xsd:element name=”note” type=”noteType” minoccurs=”0” />
      </xsd:sequence>
      <xsd:attribute name=”date” use=”required” type=”xsd:date”>
   </xsd:complexType>
</xsd:element>
<xsd:complexType name=”partType”> .... </xsd:complexType>
  ....
```

XML Schema

- Typdeklarationen

```
<xsd:element name=”order”>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name=”part” type=”partType” minoccurs=”1”
                                                  maxoccurs=”unbounded”/>
         <xsd:element name=”destination” type=”destType”/>
         <xsd:element name=”note” type=”noteType” minoccurs=”0” />
      </xsd:sequence>
      <xsd:attribute name=”date” use=”required” type=”xsd:date”>
   </xsd:complexType>
</xsd:element>
<xsd:complexType name=”partType”> .... </xsd:complexType>
  ....
```

XML Schema

- Attributdeklarationen

```
<xsd:element name=”order”>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name=”part” type=”partType” minoccurs=”1”
                                                  maxoccurs=”unbounded”/>
         <xsd:element name=”destination” type=”destType”/>
         <xsd:element name=”note” type=”noteType” minoccurs=”0” />
      </xsd:sequence>
      <xsd:attribute name=”date” use=”required” type=”xsd:date”>
   </xsd:complexType>
</xsd:element>
<xsd:complexType name=”partType”> .... </xsd:complexType>
  ....
```

XML Schema

- Weitergehende Definitionen
  - Erweitern von Typen:
    - <xsd:extension>
  - Einschränken von Typen:
    - <xsd:restriction>
  - Umfassende Möglichkeiten zum Einsatz von Referenzen (ID/IDRef, Key/Keyref, ...)
  - NULL Werte aus DB-Welt abbildbar

Weitere Strukturbeschreibungssprachen

- Neben DTDs und XML Schema existiert noch eine Reihe weiterer Sprachen mit z.T. sehr unterschiedlichen Ansätzen
  - Datatypes for DTDs
  - XML Data Reduced (XDR)
  - The Schematron
  - SOX
  - TREX
  - ...

Datatypes for DTDs

- Erweiterung von DTDs um Datentypen
  - Beseitung eines Hauptmangels
- Integration:
  - Definition zweier Spezialattribute
    - e-dtype: Elementdatentypen
    - a-dtype: Attributdatentypen
  - Kopplung an extern definierte Datentypen
    - z.B. in XML Schema
  - Begünstigt Migration DTD -> XML Schema

Datatypes for DTDs

- Beispiel:

```
<!ELEMENT order(part+, destination, note?) >
<!ATTLIST order
        a-dtype CDATA #FIXED “date dateTime”>

<!NOTATION dateTime SYSTEM “urn:schemas-microsoft-com:datatypes/dateTime”>

<!ELEMENT part (quantity, duedate)>
<!ATTLIST part
        pnumber CDATA #REQUIRED >

<!ELEMENT destination (factory | interimstorage)>

<!ELEMENT factory (#PCDATA)>
<!ELEMENT hall (#PCDATA)>
```

XML Data Reduced

- Alternative zu XML Schema
- Ursprung: Firmenkonsortium (u.A. Microsoft)
- In der ursprünglichen .NET Initiative implementiert
  - Internet Explorer 5.0
  - Biztalk Server
- Inzwischen: Migration von .NET auf XML Schema

XML Data Reduced

- Beispiel:

```
<xdr:ElementType name=”order” order=”seq”>
   <xdr:element type=”part” minoccurs=”1” maxoccurs=”*” />
   <xdr:element type=”destination” />
   <xdr:element type=”note” minoccurs=”0”/>
</xdr:ElementType>

<xdr:ElementType name=”note” dt:type=”dt:string”/>

<xdr:ElementType name=”part”>
  ....
</xdr:ElementType>

.....
```

The Schematron

- Völlig anderer Ansatz
  - Bisher: Grammatikbasiert
  - Hier: Mustererkennung auf Bäumen
    - Nutzt aus, dass XML Dokumente als Baum darstellbar sind
    - Denkbare Regel: “Jedes <part> Element muss Kind des <order> Elements sein”
  - Kein Ersatz sondern Ergänzung

The Schematron

- Beispiel: “Prüfe ob Element <part> Kind von Element <order> ist”

```
<schema>
   <pattern name=”Element”>
      <rule context=”part”>
         <assert test=”parent::order”>
         <name/>
         Element muss Kind von Order sein
      </rule>
   </pattern>
</schema>
```

Zusammenfassung und Ausblick

- Strukturbeschreibungen machen Sinn !
- Menge von Sprachen verfügbar
  - Standard derzeit: XML Schema
- Keine bahnbrechenden Neuentwicklungen absehbar

Fragen ?