

XML Vorlesung ETHZ, Sommersemester 2006

XML Schemasprachen

Erik Wilde

20.6.2006

<http://dret.net/lectures/xml-ss06/>

20.6.2006

XML Vorlesung ETHZ SS 2006

1

Übersicht

- DTDs und XML Schema
 - Einschränkungen und Nachteile
- Schemasprachen als Konzept
 - Grammatiken und Anderes
- Beispiel für andere Schemasprachen
 - regelbasierte Validierung (Schematron)
- Schemasprachen selbstgemacht

20.6.2006

XML Vorlesung ETHZ SS 2006

2

Wozu überhaupt Schemasprachen?

- Einschränkung der akzeptierten Dokumente
 - definiertes Standardvokabular für Einschränkungen
 - Verwendung von Standardsoftware zur Validierung
- Schemasprachen können nie alles
 - DTDs unterstützen kaum Datentypen
 - dass ein Attributwert eine positive Zahl sein muss
 - XML Schema unterstützt keine Co-Constraints
 - dass ein Attributwert kleiner sein muss als ein anderer
 - Schematron kann nicht mit externen Daten arbeiten
 - dass ein Attributwert in einer Datenbank existieren muss

20.6.2006

XML Vorlesung ETHZ SS 2006

3

Schemasprachen sparen Geld

- Schemas schränken XML deklarativ ein
 - werden von Software interpretiert und angewendet
 - häufig verwendete Software-Komponenten
 - tendenziell weit verbreitet und ausgereift
- keine Eigenentwicklung von Software nötig
 - falls Software für die Schemasprache existiert
 - reduziert Fehlermöglichkeiten
 - einfacher Update (neuen Validator installieren)
- besserer "Schutz" der Applikation
 - weniger Fehlerbehandlung in der Applikation

20.6.2006

XML Vorlesung ETHZ SS 2006

4

XML DTDs

- die XML Spezifikation definiert zwei Syntaxen
 - die Syntax von XML Dokumenten
 - die Syntax von XML DTDs
 - und wie sie den Aufbau von Dokumenten beeinflussen
- DTDs definieren Grammatiken
 - Aufbauregeln für XML Dokumente
 - definieren unendlich grosse Mengen an Dokumenten
 - mit wenigen (trivialen) Ausnahmen
- DTDs müssen von Parsern interpretiert werden
 - Entity-Definitionen sind in der DTD

20.6.2006

XML Vorlesung ETHZ SS 2006

5

Inhalt einer DTD

- Elementtyp Definitionen
 - Element und sein Content Model
- Attributlisten Definitionen
 - Mengen von Attributen für Elemente
 - der Typ von Attributen (u.U. Aufzählungsliste)
 - diverse Qualifier für Attribute
 - optional, verlangt, Default, fixer Wert
- Entity Definitionen
 - Interpretation erzeugt neuen Text im Dokument

20.6.2006

XML Vorlesung ETHZ SS 2006

6

Nachteile der DTDs

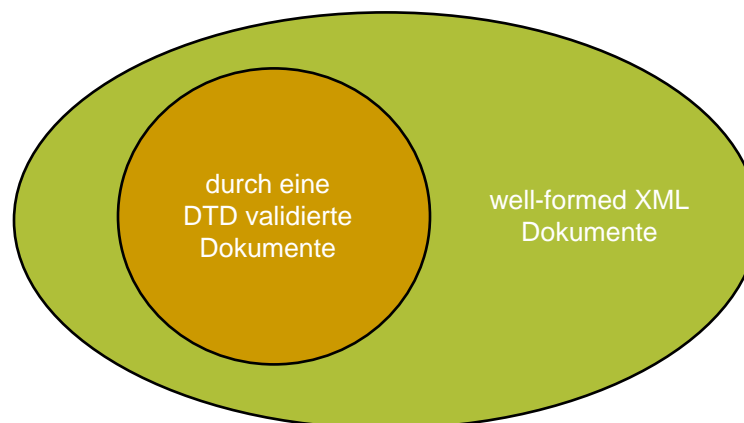
- keine Beziehungen zwischen Elementtypen
 - keine Typ-Hierarchie der Elemente
 - zusammenhangsloses Nebeneinander
- keine Unterstützung von Wiederverwendung
 - verbreitetes Parameter Entity Design Pattern
- keine anwendungsorientierten Datentypen
- keine Unterstützung für XML Namespaces
 - "DTDs and Namespaces don't mix"
- keine XML Syntax
 - kann nicht mit XML Tools verarbeitet werden

20.6.2006

XML Vorlesung ETHZ SS 2006

7

Schemasprachen als Konzept



20.6.2006

XML Vorlesung ETHZ SS 2006

8

XML Schema

- Kompromiss aus vielen Vorschlägen
 - komplett neue Abstraktionsebene
 - umfangreiche Bibliothek an eingebauten Datentypen
- Modellierungshierarchien
 - Typenhierarchien für Simple und Complex Types
 - basierend auf einem konzeptionellen *Ur-Type*
 - unterschiedliche Ableitungsmethoden
 - Simple: Restriction, List und Union
 - Complex: Restriction und Extension
- Ziel war eine möglichst mächtige Sprache
 - nicht unbedingt der beste Ansatz (monolithisch)

20.6.2006

XML Vorlesung ETHZ SS 2006

9

XML Schema Eigenheiten

- keine Unterstützung von Entities
 - falls benötigt, in einer minimal-DTD definiert
- nur Namespace-Compliant XML erlaubt
- Identity Constraints als ID/IDREF++
 - basieren auf Elementnamen (keine Typen!)
 - Bruch im Design von XML Schema
- ist definiert auf Infosets
 - Validierung ergänzt das Infoset
 - Zugriff nicht durch einen Standard definiert
 - Abhängigkeit von der spezifischen Implementierung

20.6.2006

XML Vorlesung ETHZ SS 2006

10

Nachteile von XML Schema

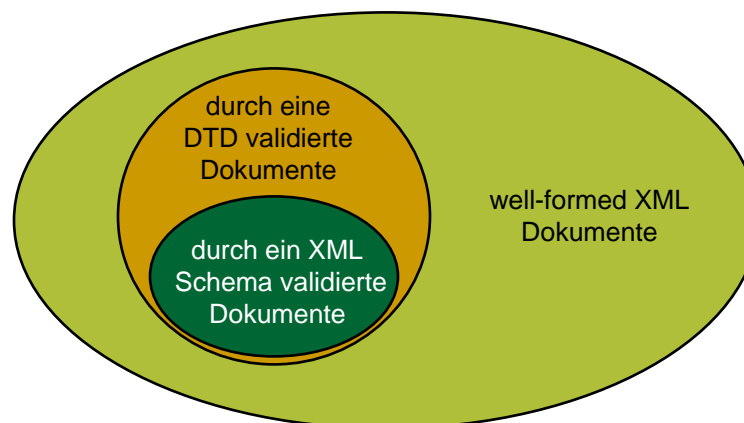
- hohe Komplexität
- momentan nur mangelhafte Implementierungen
 - zum Schreiben von XML Schema selber
 - einzig IBM's SQC implementiert XML Schema komplett
 - zum Validieren von XML Dokumenten
 - keine komplette Implementierung bekannt
- keine Co-Constraints
 - minimal unterstützt durch Identity Constraints
- kein explizites Root Element
- Zeichen in *mixed Content* haben keinen Typ
 - keine Einschränkungen für *mixed Content* möglich

20.6.2006

XML Vorlesung ETHZ SS 2006

11

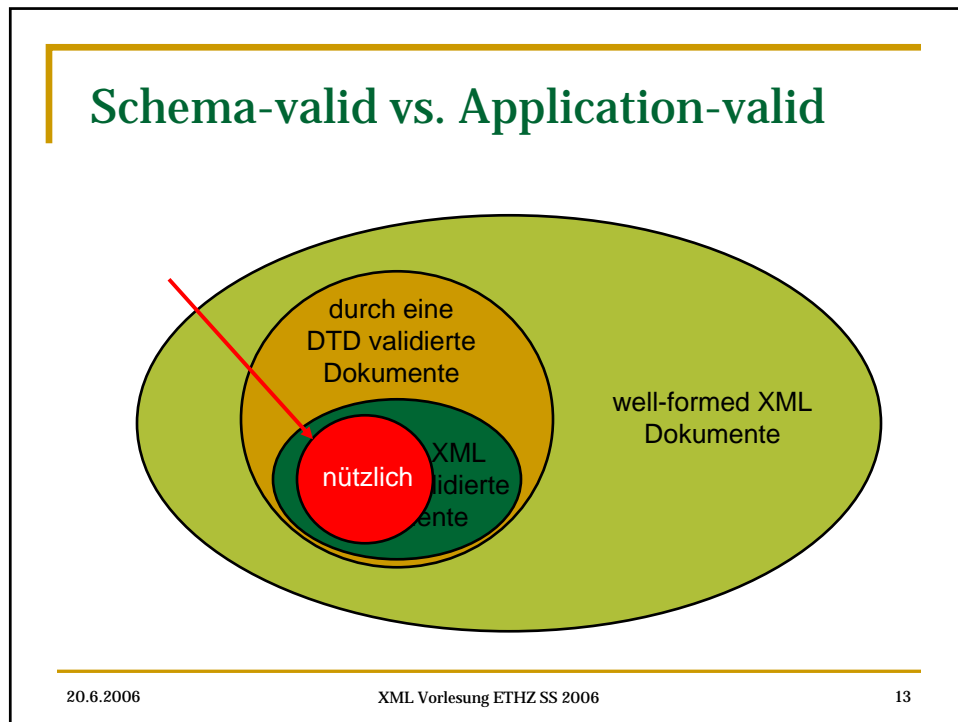
XML Schema als DTD++



20.6.2006

XML Vorlesung ETHZ SS 2006

12



Verbindung Instanz/Schema

- verschiedene Mechanismen sind denkbar
 - irgendwie verankert in der XML Syntax
- DTDs benutzen spezielle Konstrukte
 - Document Type Declaration: `<!DOCTYPE . . .`
- XML Schema benutzt Namespaces
 - es gibt einen Instance Namespace
 - und dieser definiert ein entsprechendes Attribut
- RELAX NG definiert keine Verbindung
- XML ermöglicht weitere Varianten
 - Kommentare, Processing Instructions, Namespaces

20.6.2006 XML Vorlesung ETHZ SS 2006 14

RELAX NG

- alternative Grammatik-basierte Schemasprache
 - RELAX, entwickelt von Murata Makoto
 - TREX, entwickelt von James Clark
- basierend auf bekanntem Formalismus
 - DTDs benutzen einen ad hoc Formalismus
 - XML Schema benutzt einen ad hoc Formalismus
 - RELAX NG benutzt *Hedge Automata*
- keine Typen (\Rightarrow keine Typhierarchien)
- keine eigene Typbibliothek
 - verwendet häufig die XML Schema Simple Types

20.6.2006

XML Vorlesung ETHZ SS 2006

15

RELAX NG Prinzipien

- das Dokument bleibt unverändert
 - keine Default-Werte im Schema
 - keine Entities
- nur Beziehungen Instanz-Schema
 - DTDs und XML Schema: Instanz-Instanz
 - ID/IDREF(S) bzw. Identity Constraints
 - XML Schema: Schema-Schema
 - Typhierarchie
- Gleichbehandlung von Attributen und Elementen
 - Content Model enthalten beide
 - ermöglicht viele nützliche Anwendungen

20.6.2006

XML Vorlesung ETHZ SS 2006

16

DTD Beispiel

```
<! ELEMENT document (heading, chapter)>
```

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

```
<! ELEMENT chapter (heading, para+)>
```

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

XML Schema Beispiel

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:element name="document">  
    <xs:complexType>  
      <xs:sequence>  
        <xs:element ref="heading"/>  
        <xs:element ref="chapter"/>  
      </xs:sequence>  
    </xs:complexType>  
  </xs:element>  
  <xs:element name="heading" type="xs:string"/>  
  <xs:element name="chapter">  
    <xs:complexType>  
      <xs:sequence>  
        <xs:element ref="heading"/>  
        <xs:element name="para" type="xs:string" maxOccurs="unbounded"/>  
      </xs:sequence>  
    </xs:complexType>  
  </xs:element>  
</xs:schema>
```

RELAX NG Beispiel

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start><ref name="document"/></start>
  <define name="document">
    <element name="document">
      <ref name="heading"/>
      <ref name="chapter"/>
    </element>
  </define>
  <define name="heading">
    <element name="heading"><text/></element>
  </define>
  <define name="chapter">
    <element name="chapter">
      <ref name="heading"/>
      <oneOrMore>
        <element name="para"><text/></element>
      </oneOrMore>
    </element>
  </define>
</grammar>
```

RELAX NG Beispiel (Compact Syntax)

```
start = document

document = element document { heading, chapter }

heading = element heading { text }

chapter = element chapter {
  heading,
  element para { text }+
}
```

RELAX NG Eigenheiten

- keine eingebauten Simple Types
 - oft werden die XML Schema Simple Types verwendet
- keine eigenständige Typenebene
 - Patterns definieren die Grammatik
- nicht-deterministische "Content Models"
 - umstritten und problematisch in der Interpretation
- Gleichbehandlung von Attributen und Elementen
 - Attribute werden Teil des "Content Model"
- keine Einschränkung der xs:all Gruppe
 - freie Kombination mit anderen Gruppen

20.6.2006

XML Vorlesung ETHZ SS 2006

21

Grenzen von Schemasprachen

- was kennen Schemasprachen?
 - das Schema
 - die Instanz
 - u.U. in die Schemasprache eingebaute Dinge
 - Typbibliotheken, Funktionen, ...
- was kennen sie nicht?
 - applikationsspezifische Anforderungen
 - Daten ausserhalb von Schema oder Instanz
- Schemasprachen können sehr einfach sein
 - `(/order/@shipdate > today() + 2) & (/order/@cid = dblookup(customer))`

20.6.2006

XML Vorlesung ETHZ SS 2006

22

Applikationsanforderungen

- Schemasprachen sind nie komplett
 - Code ist immer notwendig
 - das Ziel ist so wenig Code wie möglich
- zwei prinzipielle Wege sind möglich
 - Schemasprachen kombinieren
 - modularer Ansatz
 - Validierung als Kombination verschiedener Tools
 - Schemasprachen erweitern oder definieren
 - sollte auch wieder auf XML Tools basieren
 - beide Wege können kombiniert werden
- gute Planung spart Implementierungsaufwand

20.6.2006

XML Vorlesung ETHZ SS 2006

23

Klassifizierung von Schemasprachen

- Grammatik-basierte Sprachen
 - sehr weit verbreitet in der XML-Welt (DTD)
 - sinnvolles Konzept für XML
 - DTD, XML Schema, RELAX NG
- regelbasierte Sprachen
 - definieren Regeln, die eingehalten werden müssen
 - eher isolierte Bedingungen
 - oftmals gute Ergänzung zu Grammatiken
- andere Sprachen
 - momentan noch stark in Entwicklung

20.6.2006

XML Vorlesung ETHZ SS 2006

24

Schematron

- benutzt *Tree Patterns*
 - Identifikation von Teilen (mittels XPath)
 - Tests basierend auf dem selektierten Context
- sehr einfache Abbildung auf XSLT
 - beliebige XPath Ausdrücke als Assertions
 - keine spezielle Software notwendig
- zwei Arten von Constraints
 - Assertions müssen gelten (true ergeben)
 - andernfalls wird eine Aktion ausgelöst
 - Reports informieren über Tatsachen
 - werden ausgegeben, falls sie false ergeben

20.6.2006

XML Vorlesung ETHZ SS 2006

25

Schematron Beispiel

```
<house>
  <wall /><wall /><wall /><wall />
  <window /><window />
  <roof />
</house>
```

```
<rule context="house">
  <assert test="count(wall) = 4">A house should have
    four walls</assert>
  <report test="not(roof)">This house does not have a
    roof</report>
  <report test="window">This house has
    windows</report>
</rule>
```

20.6.2006

XML Vorlesung ETHZ SS 2006

26

Schematron Eigenschaften

- Rules definieren den Context von Assertions
 - Abbildung auf XSLT Templates
 - enthalten Assertions und Reports
 - Abstract Rules ermöglichen Modularität
 - Gruppierung einer Menge von Tests
 - Referenzierung aus anderen Rules
- Patterns fassen Rules zusammen
- Phases referenzieren Patterns
 - Anwendung des Schemas zu verschiedenen Zeiten
 - Parametrisierung der Validierung

20.6.2006

XML Vorlesung ETHZ SS 2006

27

Schematron Assertions

- Bedingungen, die gelten müssen
 - Assertions müssen gelten (sonst Fehlermeldung)
 - Reports dürfen nicht gelten (sonst Fehlermeldung)
- Assertions enthalten eine Meldung
 - `<name/>` gibt den Kontext aus
 - `<emph>` für Hervorhebungen ist erlaubt

```
<assert test="count(topicref[@TID = ../.. /@TID])">  
  description must mention the described topic</assert>
```

- offene Frage: Kontext der Auswertung?

20.6.2006

XML Vorlesung ETHZ SS 2006

28

Schematron Rules

- Kontext für Assertions
 - definiert mit einem XSLT Pattern
 - Untermenge von XPath
 - Elemente, Attribute, Prädikate, Pfade, Unions
 - Knoten, für die Assertions ausgewertet werden

```
<rule context="desc">
  <assert ... </assert>
</rule>
```

- offene Frage: Gruppierung von Rules?

20.6.2006

XML Vorlesung ETHZ SS 2006

29

Schematron Patterns

- Patterns sind Gruppen von Rules
 - inhaltliche Zusammenfassung von Rules

```
<pattern name="description checking">
  <rule ... </rule>
</pattern>
```

- Patterns sind Kinder des Schemas selber
 - identifiziert über den Schematron Namespace Name

```
<schema xmlns="http://www.ascc.net/xml/schematron">
  <pattern ... </pattern>
</schema>
```

20.6.2006

XML Vorlesung ETHZ SS 2006

30

Schematron Phases

- Gruppierung von Patterns in Phasen
 - Ebenfalls Kinder des schema Elements
 - Referenzierung der Patterns durch den Namen
- Verwendung für verschiedene Passes
 - einfacher Check als erste Phase
 - kompletter Check als zweite Phase
- Phasen können ausgewählt werden
 - per Default sind alle Phasen aktiv
 - beim Aufruf können Phasen ausgewählt werden

20.6.2006

XML Vorlesung ETHZ SS 2006

31

Schematron Design Guidelines

- Zugriff auf externe Dokumente ist möglich
 - mit Hilfe der document () Funktion aus XSLT
- Abwägung zwischen Rules und Assertions
 - komplizierter Kontext und einfache Assertions
 - einfacher Kontext und komplizierte Assertions
- kompletter Ersatz von DTD oder XML Schema
 - meistens sehr aufwendig
 - Änderungen sind kompliziert und fehleranfällig
- Ergänzung bestehender Grammatiken
 - um Co-Constraints und andere Mechanismen

20.6.2006

XML Vorlesung ETHZ SS 2006

32

Schemasprachen selbstgemacht

- wechselnde Anforderungen einer Applikation
 - z.B. verschiedene Anwendungskontexte
 - z.B. Anbindung an unterschiedliche Infrastruktur
- Schemasprachen können einfach sein
 - einfaches Schema (für die Schemasprache)
 - Implementierung mit XML Tools
 - Abbildung auf bestehende Software-Komponenten
- Aufwand vs. Ertrag
 - einfache Parametrisierung einer XML-Anwendung
 - bessere Wiederverwendbarkeit
 - Selbstschutz der Entwickler (Encapsulation)

20.6.2006

XML Vorlesung ETHZ SS 2006

33

Zusammenfassung

- Schemasprachen als Programmierung
 - deklarativ vs. prozedural
 - u.U. eigene Schemasprachen benutzen
- wachsende Zahl an Schemasprachen
 - XML wird weiterhin komplexer
 - Toolset an klar fokussierten Tools
 - modularer Aufbau einer Anwendung
- Validierung als Pipeline
 - Kombination verschiedener Schemasprachen

20.6.2006

XML Vorlesung ETHZ SS 2006

34