

Das Umfeld von XML



Erik Wilde
TIK – ETH Zürich
Sommersemester 2001

Übersicht

- *XML Information Set* als abstrakte Sicht
 - *Canonical XML* als Anwendung von XML Infoset
 - digitale Signaturen mit XML (*XML Signature*)
- *XML Namespaces* für Schema-Kombination
- XML APIs
 - *Document Object Model (DOM)*
 - *Simple API for XML (SAX)*
- XML und Hypermedia (*XLink* und *XPointer*)
- *Mathematical Markup Language (MathML)*
- allgemeine Metadaten-Formate
 - *Platform for Internet Content Selection (PICS)*
 - *Resource Description Framework (RDF)*

XML Information Set (XML Infoset)

- XML definiert Syntax für strukturierte Daten
 - eigentlich wichtig ist die Struktur
 - die Syntax ist Nebensache und irritiert manchmal
- XML Infoset definierte eine abstrakte Sicht
 - Baum aus *Information Items*
 - Elemente, Attribute, Text, Kommentare, ...
 - keine Schnittstelle, sondern nur ein Modell
- OSI taucht immer wieder auf...
 - XML Infoset als ASN.1 abstrakte Syntax
 - XML als (eine mögliche...) Codierung des Baumes
 - ASN.1 XER (kein Standard!) macht genau das

Canonical XML

- XML ist sehr syntax-zentriert
 - Vergleich von XML-Dokumenten ist schwierig
 - Whitespace, Attribute order, Entities, Namespaces, ...
- Canonical XML legt eine Normalisierung fest
 - mache ein XML Infoset aus dem Dokument
 - generiere aus diesem Baum XML nach festen Regeln
- bessere Basis für Vergleiche als reines XML
 - allerdings aufwendige Operation
 - noch nicht unterstützt von der meisten Software
- interessant auch für digitale Signaturen von XML

Digitale Signaturen (DSig)

- PICS ermöglicht Aussagen über Ressourcen
 - Problem der Zuverlässigkeit der Aussagen
- *PICS Signed Labels (DSig)*
 - Verbindung von Label und Ressource
 - Verbindung von Label und Rating Service
- Ressource: nicht sinnvoll für *generic labels*
- Implementierung durch Kryptographie
 - Hashfunktionen (Fingerprints) der Ressource
 - *Public Key* Verfahren für Signatur eines Labels
 - Problem: Zertifizierung der Schlüssel?
- Weitergehende Standards in Entwicklung

XML Namespaces

- XML ermöglicht anwendungsabhängige Namen
 - Namenskonflikte sind nahezu vorprogrammiert
 - Problem der "friedlichen Koexistenz" von Namensräumen
- Lösung auf einfache Art: XML Namespaces
 - Identifikation eines Namensraumes durch eine URI
 - Verbindung eines lokalen Präfix mit der globalen URI
 - Verwendung des Präfix im XML-Dokument
- jeder kann eigene Namespaces definieren
 - keine formale Bedeutung der Namespace URI
 - meistens Verweis auf DTD oder Standard-Dokument
- Namespace Support wichtig für XML Parser

Beispiel XML Namespaces

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:output method="html" indent="yes" />

  <xsl:template match="/">
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="kurs">
    <html>
      <head>
        <title>
          <xsl:value-of select="titel"/>
        </title>
      </head>
```

Anwendungen für Namespaces

- Kombination verschiedener DTDs
 - potentielle Namenskonflikte (XML hat lokale Namen)
- eigene DTD, die fremde DTDs wiederverwendet
 - z.B. "Kurs-DTD" zusammen mit HTML oder MathML
- Trend zur DTD-Modularisierung (XHTML 1.1)
- XSLT zum Namespace-Filtern sehr einfach
 - Node-Tests können Namespaces erkennen
- alle XML-Standards unterstützen Namespaces

Identifizier vs. Links

- Identifizier (URIs bzw. URLs)
 - Identifikation unabhängig von Verbindungen
 - Identifizier existieren auch ohne Referenzen

`ETH`

- Links
 - enthalten (u.U. mehrere) Identifizier
 - zusätzliche Semantik (z.B. Link-Typ)
 - HTML definiert sehr einfache Links (→ XLink)

XML Linking

- *XLink* als Verallgemeinerung von HTML Links
- *XLink simple links* mit einer Ressource
 - Zeiger von einem Punkt auf einen anderen Punkt
- *XLink extending links* mit Ressourcen-Liste
 - Verallgemeinerung des 1:1 Konzeptes
 - erlaubt die Typisierung von Links
 - erlaubt externe Links
- *XPointer* für Zeiger in XML Dokumente
 - HTML muss Ziel im Dokument definieren
 - XPointer erlaubt Pointer, ohne das Ziel-Dokument zu ändern
 - strukturelle Spezifikation der Ressource

Motivation XLink

- XML kennt keine Hyperlinks
- HTML definiert ein festes Element
 - <A> stellt Hyperlinks dar
 - <A> ist immer innerhalb eines Dokuments
 - <A> Links sind immer unidirektional
 - <A> Links haben immer genau ein Ende (1:1)
- XLink integriert Hyperlinks in XML
 - fest definierter Mechanismus zur Einbettung
 - nicht Bestandteil des XML Standards selber
 - Browser müssen demnach XML/XLink unterstützen

XLink

- XLinks ausserhalb von Dokumenten
- Links nicht als Bestandteil einer Ressource
 - Management von Links (Gültigkeit überprüfen)
 - Links in read-only Dokumenten anbringen
 - Links u.U. wichtiger als die Ressourcen selber
 - Links auf beliebig vielen Ressourcen
- Auffinden von externen Links
 - als Bestandteil der Ressource (der HTML-Ansatz)
 - als *Linkbase* referenziert (verallgemeinert)
 - Konsequenz: nicht alle existierenden Links werden gefunden
- ermöglicht neue Dienste (Verteilen von Links)

XLink in einem XML Dokument

- Einbettung über Attribute
 - fest definierte Attributnamen
 - Elementnamen sind frei wählbar
- für Links notwendige Information
 - *locator* (identifiziert eine Ressource)
 - *link semantics* (Bedeutung eines Links)
 - *local resource semantics*
 - *remote resource semantics*
- Verhalten beim Folgen eines Links
 - **SHOW** für die Darstellung des Inhalts
 - **ACTUATE** für den Zeitpunkt des Folgens

Verwendung von XLinks

- momentan noch sehr junger W3C Standard
 - Entwicklung bereits seit 1998
 - viele Probleme der internen Abstimmung im W3C
- kann jedoch als Beispiel gelten für
 - Trend in Hinsicht auf definierte XML Semantiken
 - Trend in Hinsicht auf Browserentwicklung
- parallel zum Thema Content Management
 - alter WWW-Slogan: "Content is King"
 - weiterhin gültig, denn ohne Content keine Links!
 - aber Link Management wird wichtig werden
- Entwicklungen voraussehen & berücksichtigen

Links als neuer Ressourcotyp

- heute Konzentration auf Content
 - Content Management Systems sind "in"
 - zunehmende Integration mit Datenhaltung
- kommende Browsergeneration
 - unterstützt XLink/XPointer und XSL-FO
 - unterstützt damit neues Hypermedia-Modell
- Link Management als strategische Aufgabe
 - spezialisierte Link Management Systeme
 - Integration in Content Management
 - Wiederbenutzbarkeit von Information
 - weniger Redundanz, mehr Konsistenz

APIs: Zugriff auf XML Dokumente

- XML-Dokumente sind strukturierten Daten
 - Problem der definierten Interfaces
 - Zugriff oder Speicherung nicht unbedingt als XML
- *XML Information Set*: abstrakte Sicht
 - Standards beziehen sich meist auf XML Infoset
 - syntax-unabhängige abstrakte Sicht von XML
 - jedoch: nur ein Modell, keine definierte Schnittstelle!
- Programmierschnittstellen für XML-Dokumente
 - *Document Object Model (DOM)*
 - *Simple API for XML (SAX)*
 - DOM und SAX sind keine direkten Konkurrenten!

Document Object Model (DOM)

- entstanden für portables JavaScript
- Programmierinterface für Dokumente
- unabhängig vom Dokumentenmodell
 - *core* definiert einen strukturellen Kern
 - *HTML* definiert HTML-spezifische Aspekte
 - *XML* erlaubt den Zugriff auf XML-Dokumente
- unabhängig von der Programmiersprache
 - Definition in *Interface Definition Language (IDL)*
 - *language bindings* für diverse Sprachen
 - bisher *ECMAScript* und *Java*

DOM Dokumentenmodell

- das DOM Modell beschreibt
 - logische Struktur eines Dokumentes
 - Zugriff auf Teile eines Dokumentes
 - Manipulation eines Dokumentes
- *core* definiert allgemeine Eigenschaften
 - 12 Objekttypen (u.U. abgeleitete Typen)
- HTML und XML bilden separate Einheiten
 - Abbildung von HTML Elementen auf Objekte
 - u.U. Methoden (z.B. *InsertCell* in Tabellenzeilen)

DOM vs. Simple API for XML (SAX)

- Standardisierungsweg
 - DOM wird vom W3C entwickelt (definierter Weg)
 - SAX wird von einer informellen Gruppe definiert
- DOM bietet eine Datenstruktur
 - komplexer und leistungsfähiger
- SAX bietet eine Sequenz von Events
 - einfacher, aber limitiert (praktisch kein Kontext)
 - gut geeignet zum Serialisieren (z.B. Export/Import)
- viele Produkte unterstützen DOM und SAX
 - SAX zum Lesen von Daten und Erzeugen von DOM
 - DOM traversieren und SAX Events erzeugen

Simple API for XML (SAX)

- eventbasiertes Interface zu XML
 - gut geeignet für grosse Dokumente
 - gut geeignet für sequentielle Bearbeitung
 - weniger geeignet für komplexe Bearbeitung
- Parser generiert Ereignisse für
 - Start und Ende von Elementen
 - Attribute
 - Kommentare, Zeichenketten, Entities
 - und alles andere, was im Dokument ist...
- z.B. Apache's Xerces: SAX1/2, DOM1/2
 - DOM basiert meist auf SAX-Primitiven

Mathematical Markup Language

- MathML ist eine XML-Anwendung
- HTML unterstützt keine Formeln
 - Darstellung durch Bilder
 - Darstellung durch HTML-Layout-Steuerung
- MathML definiert zwei Notationen
 - Presentation Elements
 - für die Struktur der Darstellung (Präsentation)
 - Konzepte wie Klammern, Matrizen, Operatoren
 - Content Elements
 - für die mathematische Struktur
 - Konzepte wie Addition, Exponentiation, Sinus
 - beschränkter Vorrat an definierten Konzepten

MathML Inhalt

- keine manuelle Erzeugung
 - XML verlangt volles Markup
 - komplexe Darstellung einfacher Formeln
 - Tools (Formeleditoren), u.U. Import und Export
- *Presentation Element* für Zahlen
 - Zahl als Operand in einer Formel
 - Layout-Steuerung (Font und Farbe)
- *Content Element* für Zahlen
 - Zahl als Repräsentation eines Konzepts
 - Typ der Zahl (ganzzahlig oder Fließpunkt)
 - Basis der Zahl (dezimal, binär, oktal)

Beispiel einer MathML Formel

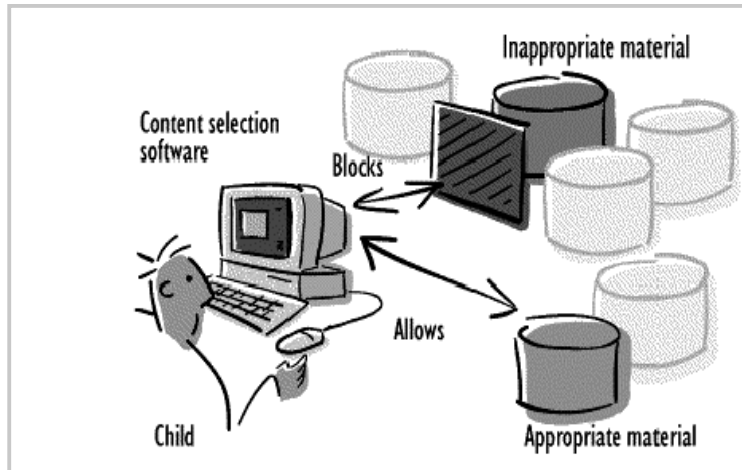
$$(a + b)^2$$

```
<msup>
  <mfenced>
    <mrow>
      <mi>a</mi>
      <mo>+</mo>
      <mi>b</mi>
    </mrow>
  </mfenced>
  <mn>2</mn>
</msup>
<apply>
  <power/>
  <apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
  </apply>
  <cn>2</cn>
</apply>
```

Platform for Internet Content Selection

- gedacht als Plattform für die Selbstkontrolle
- drei verschiedene Arten, Labels zu verwenden
 - in einem HTML-Dokument
 - mit einem HTML-Dokument (in HTTP Headers)
 - Sammlungen von Labels unabhängig vom Server
- *Rating Systems* und *Rating Services*
 - Rating System: Menge an möglichen Labels
 - Rating Service: erzeugt und verteilt Labels
- PICS definiert nur den Austausch von Labels
 - keine Definition der Erzeugung von Labels
 - keine Definition der Software zur Auswertung

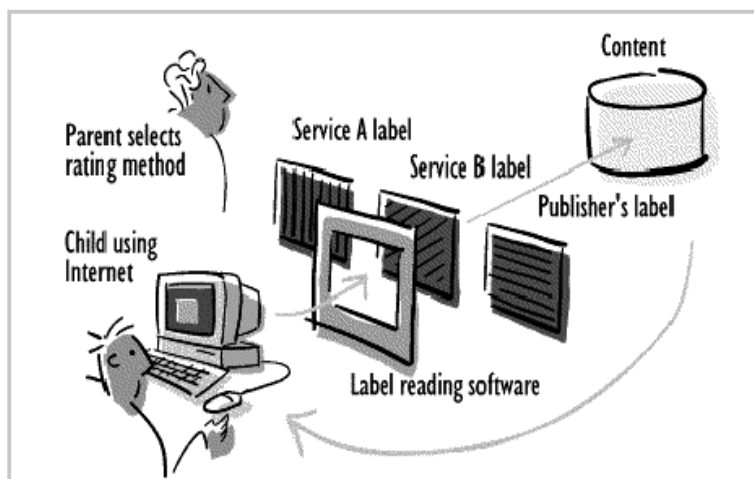
Content Selection (PICS)



WWW (SS2001) - Das Umfeld von XML

25

Unterschiedliche PICS Labels



WWW (SS2001) - Das Umfeld von XML

26

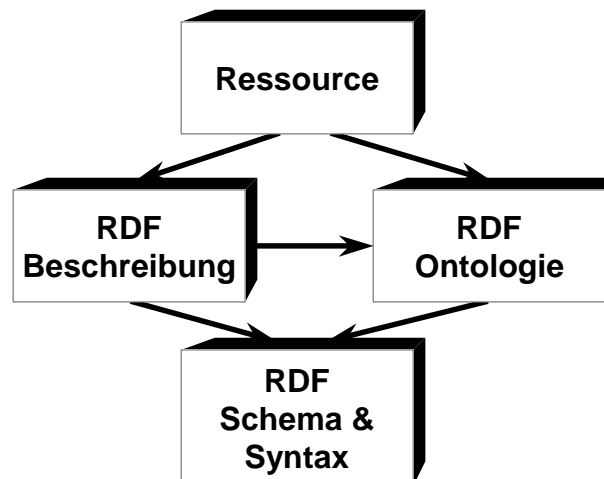
PICS Architektur

- *PICS Services and Ratings*
 - beschreibt *rating services*
 - wichtigster Teil: *rating system*
 - Dimensionen und Wertebereiche
- *PICS Label Distribution*
 - eingebettet im Dokument (HTML **<META>** Element)
 - zusammen mit dem Dokument (HTTP Header für Anforderung eines Ratings und Label)
 - *label bureau*: spezialisierter HTTP-Server
- *PICSRules*
 - Definition von Profilen (Service, Label, Verhalten)
 - ermöglicht den Austausch von Profilen

Resource Description Framework

- Entwicklung aus verschiedenen Bereichen
 - XML als Beschreibung strukturierter Dokumente
 - XML *Web Collections* als Microsofts Vorschlag
 - XML *Meta Content Framework (MCF)* von Netscape
- RDF Schemas ähnlich den PICS Rating Systems
 - allgemeiner, keine Festlegung von konkreten Werten
 - orientiert an Datenbanken und mehr noch Knowledge Representation Languages (wie z.B. KIF)
- erst im Anfangs-Stadium, Entwicklung noch unklar (Unterstützung der Search Engines)

RDF Metadata



WWW (SS2001) - Das Umfeld von XML

29

Zusammenfassung

- XML selber ist ein sehr einfaches Format
 - nicht mehr als eine Syntax für strukturierte Daten
 - "ASCII für das 21. Jahrhundert"
- XML lebt von der Akzeptanz und vom Umfeld
 - XML als anerkannter Standard
 - viele verfügbare Tools, laufend mehr
 - Ziel: Unix Toolset auf XML Basis
- momentan noch vieles im Fluss
 - ausser XML 1.0 selber nur wenig stabile Standards
 - erst am Anfang einer langen Entwicklung

WWW (SS2001) - Das Umfeld von XML

30