Entwicklung und Nutzen von XML

Informatikseminar SS2004

Markus Kobligk
(wi4113)

I. Einleitung

- Auszeichnungen
- Auszeichnungssprachen

II. Entwicklung von XML

- Ansatzpunkte
- Realisierung
- Eigenschaften von XML

III. Nutzen von XML

IV. Zusammenfassung

Auszeichnungen?

- Informationen/Symbole
- Zusätzlich zu eigentlichem Text

Wozu Auszeichnungen?

- Bedeutungsgehalt erweitern
- Texte gliedern

Auszeichnungssprache?

- Menge von Symbolen, die im Text eines Dokuments platziert werden, um dessen Bedeutungsgehalt zu erweitern und Teile des Dokuments voneinander abzugrenzen

Generische Auszeichnungen

- Deskriptive Auszeichnungen
- *GenCode-Projekt*

Erste elektronische Formate (troff, TEX)

- Aufbereitung für Druck und Monitordarstellung
- Kein durchsuchen, keine Wiederverwendung, keine Strukturierung

Generalized Markup Language (GML)

- IBM-Projekt
- Ziel: Auszeichnung von Dokumenten für die Benutzung auf mehreren Informationssubsystemen

Standard Generalized Markup Language (SGML)

- ANSI
- Ziel: Standard-Textbeschreibungssprache auf Grundlage vom GML

Hypertext Markup Language (HTML)

- CERN
- SGML-Dokumenttyp
- Kompakt und effizient

Warum eine neue Sprache?

- Kritik an HTML
  - Nur ein einziger Dokumenttyp
  - Mischung von Inhalt und Layout
  - Keine problembezogene Auszeichnung
  - Problemspezifische Strukturierung nicht möglich
- Zunehmende Datenübertragung
- Problemspezifische Strukturierung von Daten
- Neue Anforderungen an Auszeichnungssprachen

Wunsch: Erweiterbare Auszeichnungssprache

- … mit der Mächtigkeit von SGML
- … und der Einfachheit und Akzeptanz von HTML

Einfache Sprache

- Vereinfachung von SGML
- XML als Untermenge von SGML
- Kompatibel zu SGML

XML ist …

- … keine Auszeichnungssprache
- … keine Programmiersprache
- … kein Ersatz für HTML
- … kein Allerheilmittel für das Web

XML ist eine Meta-Sprache zur Festlegung der Syntax von Auszeichnungssprachen

Grundsätze

- Anwendungsspezifische Auszeichnungen
- Trennung von Inhalt und Layout
- Bestmögliche Fehlerüberprüfung
- Eindeutige Strukturen
- Einfachheit

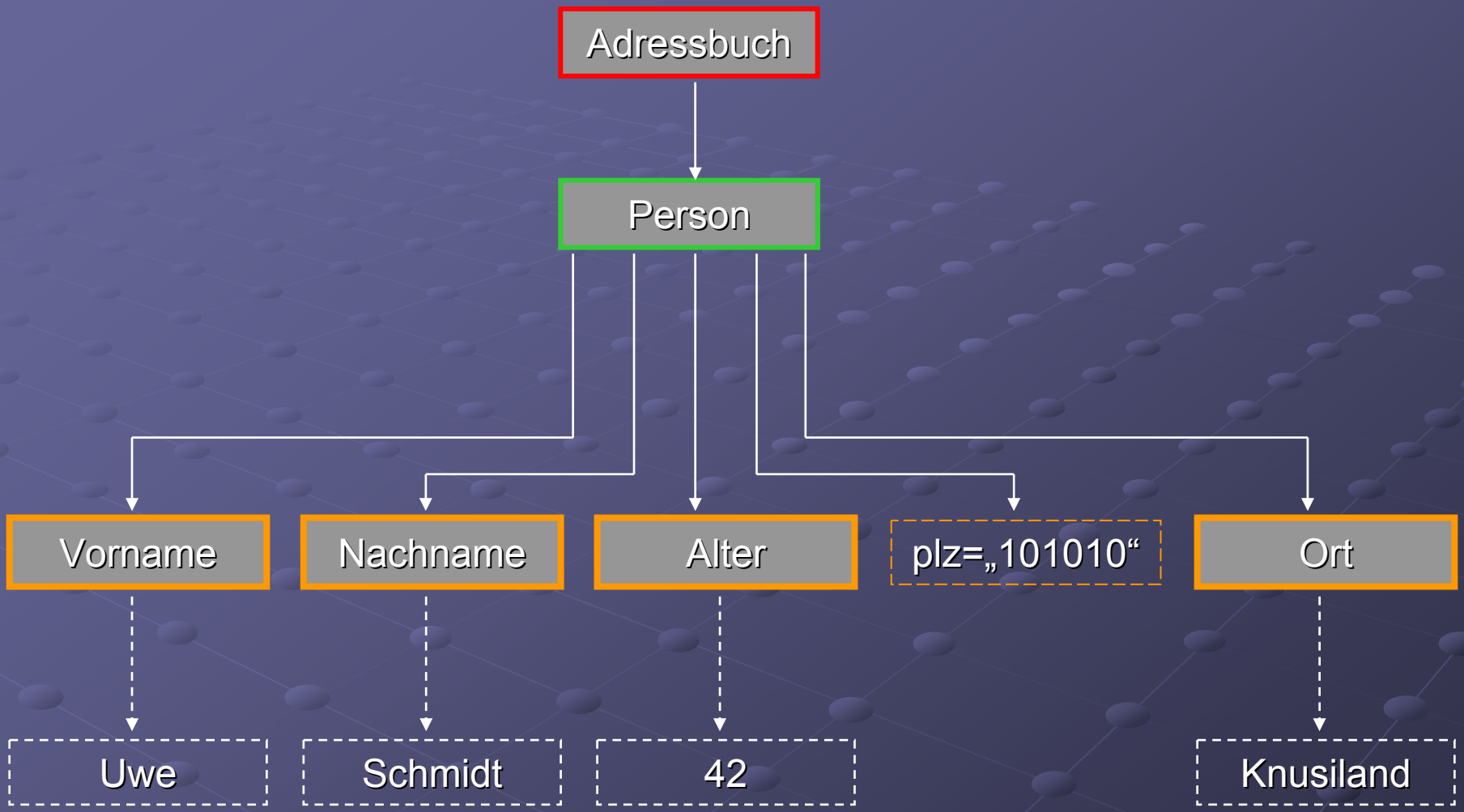
Eigenschaften

- Selbstdokumentierend
- Baumstruktur
- Internationalisierung
- Offener Standard
- Daten strukturiert

```
<?xml version="1.0" ?>
<Adressbuch>
        <Person>
                <Vorname>Uwe</Vorname>
                <Nachname>Schmidt</Nachname>
                <Alter>42</Alter>
                <Ort plz="10111">Knusiland</Ort>
        </Person>
</Adressbuch>
```

```
<?xml version="1.0" ?>
<Adressbuch>
    <Person>
        <Vorname>Uwe</Vorname>
        <Nachname>Schmidt</Nachname>
        <Alter>42</Alter>
        <Ort plz="10111">Knusiland</Ort>
    </Person>
</Adressbuch>
```

II. Baumstruktur von XML-Dokumenten II




Meine Auszeichnungssprache
Freeform XML
Dokumentenmodellierung
Relax NG
DTD
XML-Schema

Document Type Definition

- Deklaration erlaubter Elemente in Dokument (Vokabular der Sprache)
- Für jedes Element Inhaltsmodell und erlaubte Attribute
- Deklaration von Entitäten
- DTD interne Teilmenge oder extern in Datei
- Grammatik in EBNF

Warum DTDs?

- Einheitliche Festlegung der Syntax
- Wiederverwendung
- Internet-/Intranet-weite Nutzung

```
<!ELEMENT Adressbuch (Person)*>
<!ELEMENT Person (Vorname, Nachname, Ort, Alter)>
<!ELEMENT Vorname ANY>
<!ELEMENT Nachname ANY>
<!ELEMENT Alter ANY>
<!ELEMENT Ort ANY>
<!ATTLIST Ort plz CDATA #REQUIRED>
```

```
<?xml version="1.0" ?>
<!DOCTYPE Adressbuch SYSTEM "adressbuch.dtd">
<Adressbuch>
        <Person>
                <Vorname>Uwe</Vorname>
                <Nachname>Schmidt</Nachname>
                <Ort plz="101010">Knusiland</Ort>
                <Alter>42</Alter>
        </Person>
</Adressbuch>
```

Stärken

- Viele Dokumenttypen mit DTD definiert (HTML, XHTML)
- Alle SGML- und XML-Tools können DTDs verarbeiten
- Lange Erfahrung mit DTDs
- Handliche Größe
- Einfache Syntax

Schwächen

- Keine Unterstützung von Namensräumen
- Keine XML-Syntax (EBNF)
- Begrenztes Typ-Konzept

XML-Schema

- Vorlagen (Schemata) definieren Dokumentstruktur
- Schemata selbst XML

Vorteile

- Mächtigeres Typkonzept (Datum, Währung, Integer)
- Nutzung selbsterstellter Datentypen (Archtypes)
- Unterstützung von Namensräumen
- Vererbung deklarierter Objekte
- Schemata selbst wohlgeformtes XML

Nachteil

- Syntax sehr komplex

Vokabulare (Dialekte)

- Auszeichnungssprache für allg. Anwendungsgebiete
- Weltweite/internet-weite Nutzung
- Standardisiert
- Vokabulare für…

  Handel, Wissenschaft, Medizin, Jura, EDV, Astronomie, …

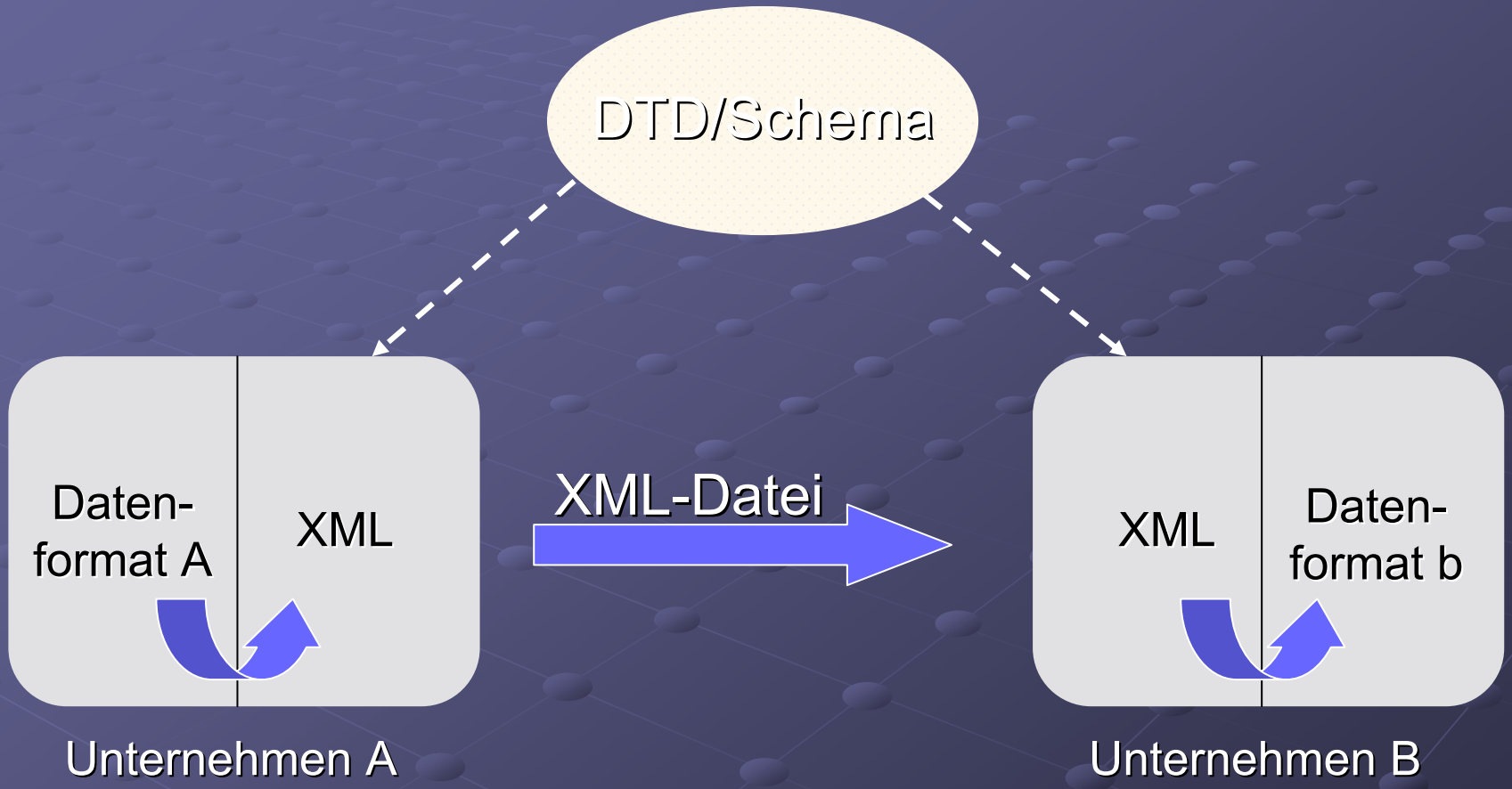
Beispiel: MathML

- Mit XML-DTD definierte Auszeichnungssprache
- Speziell für math. Ausdrücke und Formeln
- DTD öffentlich zugänglich/verwendbar
- Einheitliche Darstellung von math. Problemstellungen

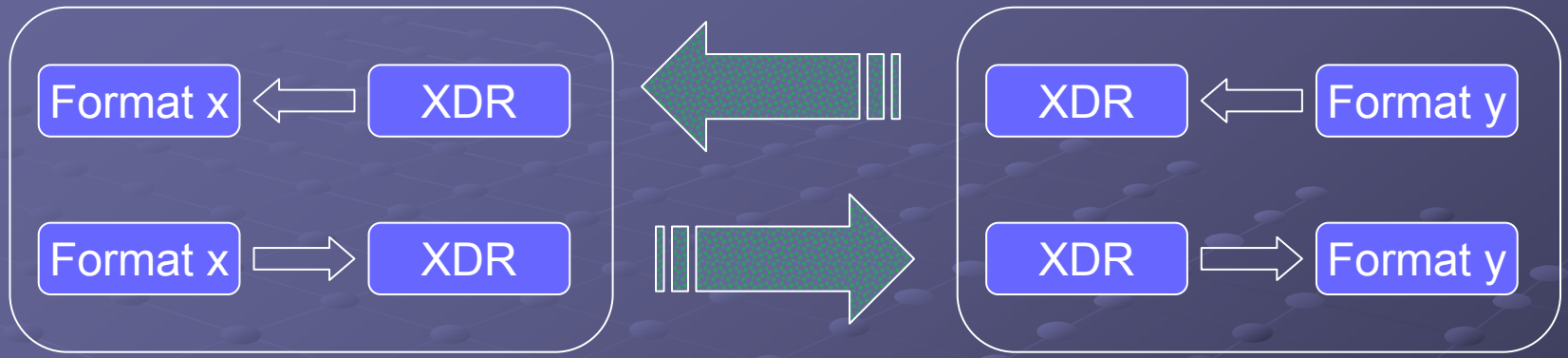
Warum ausgerechnet XML?

- Plattformunabhängig
- Internationalisierung (Unicode)
- Gut zu durchsuchen/verarbeiten
- Viele Tools vorhanden
- Entwicklung neuer Tools einfach
- Daten gut strukturierbar
- Möglichkeit der Meta-Daten-Übertragung

Beispiel: Datenaustausch zwischen Unternehmen



Beispiel: Remote Procedure Call



Kritik

- Zwar Maschinenunabhängigkeit durch XDR
- Insgesamt aber vier Umwandlungen

Remote Procedure Call mit XML

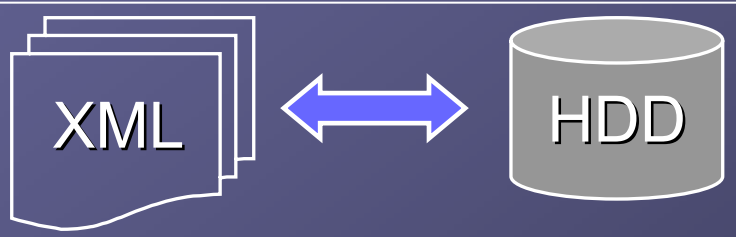
Vorteile

- Anwendungen erzeugen direkt XML
- Keine Umwandlung nötig
- XML leicht zu verarbeiten
- Tools/Schnittstellen vorhanden
- Entwicklung problemspezifischer Tools einfach

Implementierungen

- XML-RPC (Perl, Python, Java, Frontier, SKYRIX, C/C++, Lisp, PHP, MS .NET, Tcl, Delphi, Zope)
- SOAP (Simple Objects Access Protocol)

Vorteile



- XML leicht zu verarbeiten
- Schnittstellen für die meisten Programmiersprachen
- Umfang von Fehlerüberprüfungen und Ausnahmebehandlungen begrenzt durch Wohlgeformtheitsregeln

Nachteile

- Sehr speicherintensiv (keine Komprimierung)
- Ungeeignet zur Speicherung von Binärdaten
- Langsamer Zugriff (rel. DB)
- Kaum Optimierung für Zugriffsgeschwindigkeit (ID, IDREF zum schnelleren Finden von Knoten)
- Bei wenig Speicherplatz/Bandbreite nicht geeignet für große Datenmengen

XML-Prozessor

- Programm, das XML liest und verarbeitet
- Beispiele: Browser, XML-Editoren, Gültigkeitsprüfer

Bestandteile

- Parser (Syntaxanalyse → Tokenstrom)
- Event-Switcher (Events → Event-Handler)
- Regel-Prozessor (Aufbau von Baumstrukturen)
- Baum-Prozessor (Verarbeitung von Baumstrukturen)

Simple API for XML

- Event-orientiertes API
- Serielle Verarbeitung in einem einzigen Durchlauf
- Kein interne Baum-Repräsentation
- Verwendung von Callback-Routinen

Vorteile

- Dokumente nach Schlüsselworten durchsuchen
- Formatierte Inhalte in Reihenfolge des Erscheinens ausgeben
- Grundlage für komplexere APIs wie DOM

Nachteile

- Keine Neuordnung der Elemente im Dokument
- Keine Auflösung von Querverweisen
- Events werden vergessen

Document Object Model

- W3C-Empfehlung für standardisierte XML-API
- Baum-orientierte Verarbeitung
- Möglichkeit zur Manipulation des Dokuments

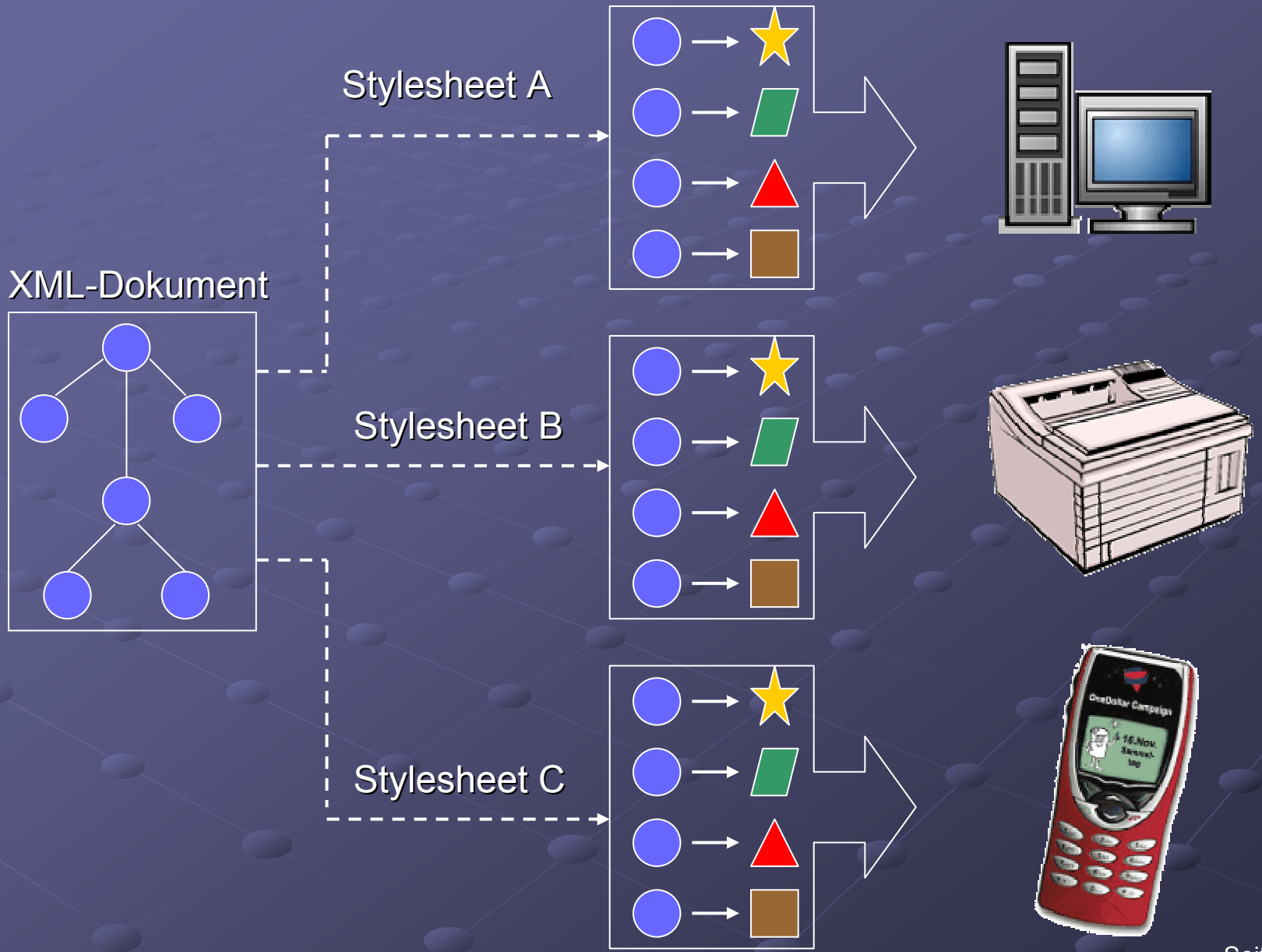
Vorteile

- Persistente Abbildung im Speicher als Baum
- Traversieren des Baumes

Nachteile

- Sehr Speicherintensiv bei großen Dokumenten
- Parsen dauert länger

III. Formatierung für unterschiedliche Ausgabemedien



eXtensible Stylesheet Language

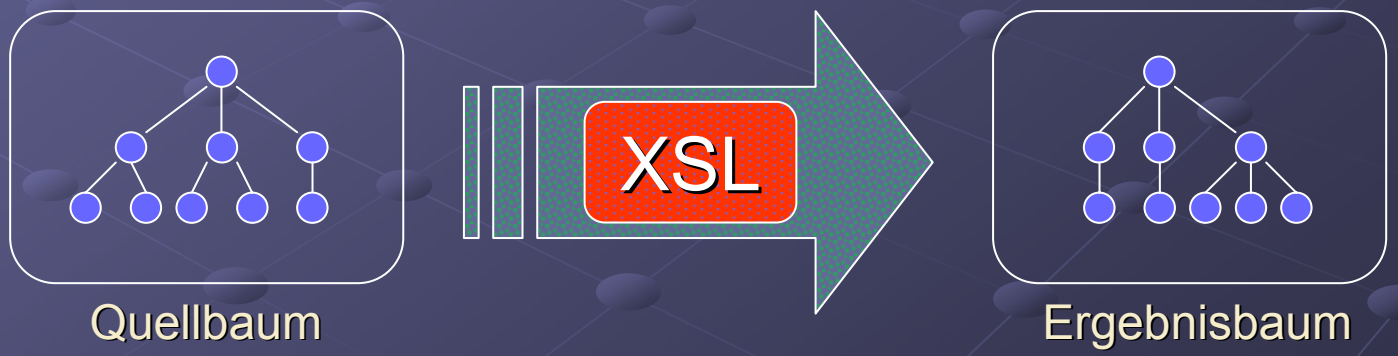
- Sprache zur Formatierung von XML-Dokumenten
- Formatierung durch XSL-Stylesheet
- Anwendung von Aktionen zu Regeln
- CSS möglich anstelle XSL-Stylesheet
- Teilsprachen: XSLT und XSL:FO

Vorteile:

- Wiederverwendung
- Formatierung für unterschiedlicher Ausgabemedien
- Trennung von Inhalt und Layout
- Formatänderungen zentral im Stylesheet
- Austausch von Stylesheets „on the fly“

XSL Transformations

- Bestandteil von XSL
- Transformationen mittels XSL-Stylesheet
- Assoziation von Mustern mit Templates im Stylesheet
- Mögliche Ergebnisse der Transformation:
  XML, HTML, XHTML, SVG, VRML, WAP, PDF, ...
- Transformation von Teilmengen möglich



XML Path Language (XPath)

- Sprache zum Zugriff auf XML-Dokumente und ausgewählte Teile davon
- Grundlage: Dokumente als Baumstruktur mit unterschiedlichen Knotentypen (Elementknoten, Attributknoten, etc.)
- Zentrales Sprachkonstrukt: Ausdrücke zur Ermittlung von Knotenmengen → Lokalisierungpfade
- Auswertung der Ausdrücke erfolgt bezogen auf den jeweiligen Kontext (-*Knoten*)
- Anwendung in XSLT und XPointer

- Problembezogene Auszeichnungssprachen
- XML als Datenaustauschformat
- XML als Format zur Datenspeicherung
- Flexibles Format zur Darstellung und Verarbeitung von Baumstrukturen
- Transformation in andere (XML-) Formate
- Möglichkeit der Meta-Daten Übertragung
- Überwindung von Inkompatibilitäten beim Datenaustausch in verteilten Systemen
- Gut lesbar von Mensch und Maschine
- Leicht zu durchsuchen und verarbeiten

- Dokumentklassenbildung mit DTDs/Schemata (Wiederverwendung)
- Erstellung von Software einfach und günstig
- Werkzeuge für XML vorhanden
- Schnittstellen für XML verfügbar
- Verwendung eines Dokumentes für die Ausgabe auf verschiedenen Medien
- Strukturierte Darstellung von Daten

XML dennoch keine Universallösung sämtlicher Probleme im Web !

Entwicklung
und
Nutzen von XML

<? Fragen ?>