DAIMLERCHRYSLER

Tutorial: Die Extensible Markup Language (XML)

Mario Jeckle
DaimlerChrysler Forschungszentrum Ulm
mario.jeckle@daimlerchrysler.com
mario@jeckle.de
www.jeckle.de

Gliederung

- Dokumente und Daten ...
- XML-Standards und -Anwendungen der zweiten Generation ...
- Anwendungen der XML im praktischen Einsatz ...

Gliederung

- Dokumente und Daten ...
  - Einführung und Überblick
  - Strukturelle Grundkonzepte
  - Namensräume
  - Dokument-Typ-Definitionen (DTD)
  - XML-Schemasprachen
  - XML und das Web
- XML-Standards und -Anwendungen der zweiten Generation ...
- Anwendungen der XML im praktischen Einsatz ...

Gliederung

- Dokumente und Daten ...
- XML-Standards und -Anwendungen der zweiten Generation ...
  - (Dokument-)Verknüpfungen: XML Links
  - Die Lokatorsprache XPath
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - Transformation von XML-Dokumenten: XSL Transformations
  - Metadatenaustausch und Schemaerzeugung: XMI
  - Die Anfragesprache XQuery
- Anwendungen der XML im praktischen Einsatz ...

Gliederung

- Dokumente und Daten ...
- XML-Standards und -Anwendungen der zweiten Generation ...
- Anwendungen der XML im praktischen Einsatz ...
  - Die Simple API for XML (SAX)
  - Das Document Object Model (DOM)
  - Nahtlose Integration von XML und Hochsprache: XML Data Binding
  - Persistenz: XML und Datenbanken
  - XML-basierter Nachrichtenaustausch und entfernte Methodenaufrufe: XML Protokolle

Gliederung

- Dokumente und Daten ...
- **Einführung und Überblick**
- Strukturelle Grundkonzepte
- Namensräume
- Dokument-Typ-Definitionen (DTD)
- XML-Schema
- XML und das Web

Dokumente und Daten ... Einführung und Überblick

Das Datenformat [XML] erleichtert den Informationsaustausch zwischen vernetzten Computern

XML schickt sich an in die Fußstapfen von HTML zu treten

/2000, p. 200]

Streit um Programmiersprache XML
[F.A.Z., 2001-07-26]

[DER SPIEGE

Sinnliche Suchmaschine
[...] existierende Systeme wie [...] XML
[DER SPIEGEL, 2000-06-07]

Das XML-Format, [...] das richtige Werkzeug zur Herstellung eigener Webinhalte

L, 2000-06-22]

Alle Dokumente sind gleich
[SZ, 1999-02-16]

Vom 'zugänglichen' XML [...] im Web kaum noch was zu sehen [...]

[c't,18/2001, 188]

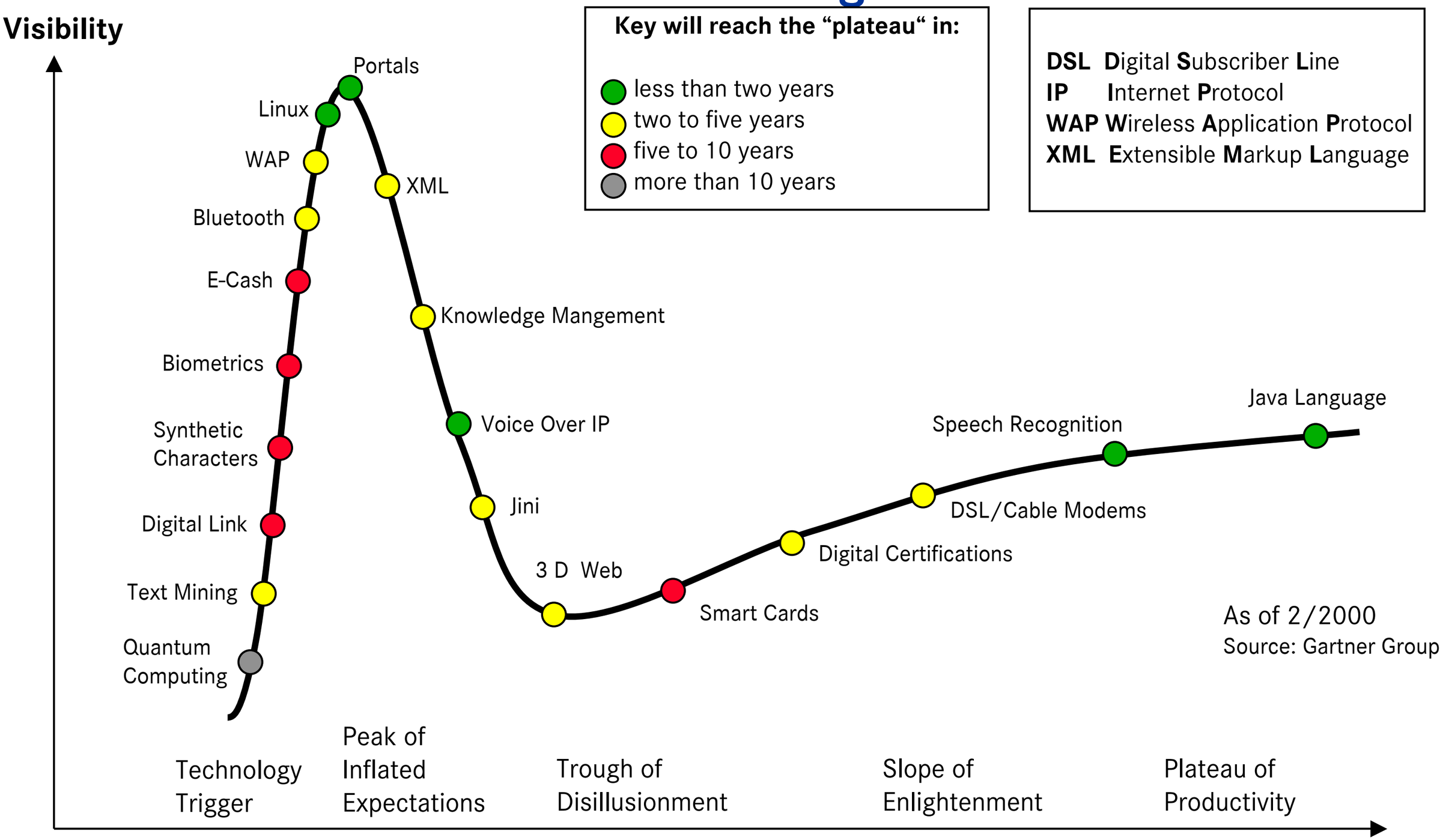
Nachfolger für ungeliebte Cookies
[...] Enge Verbindung von Java mit XML, [
Erweiterung des HTML-Standards
[DER SPIEGEL, 1999-10-0

Die Extended Markup Language für eCommerce
[F.A.Z.]

Sortieren statt Stottern
Programmiersprache HTML stößt an ihre Grenzen
XML ist kommender Code im Netz
[Süddeutsche Zeitung, 2000-01-11]

Ein digitales Esperanto für das Internet
[Die Welt, 2000-10-07]

Dokumente und Daten ... Einführung und Überblick

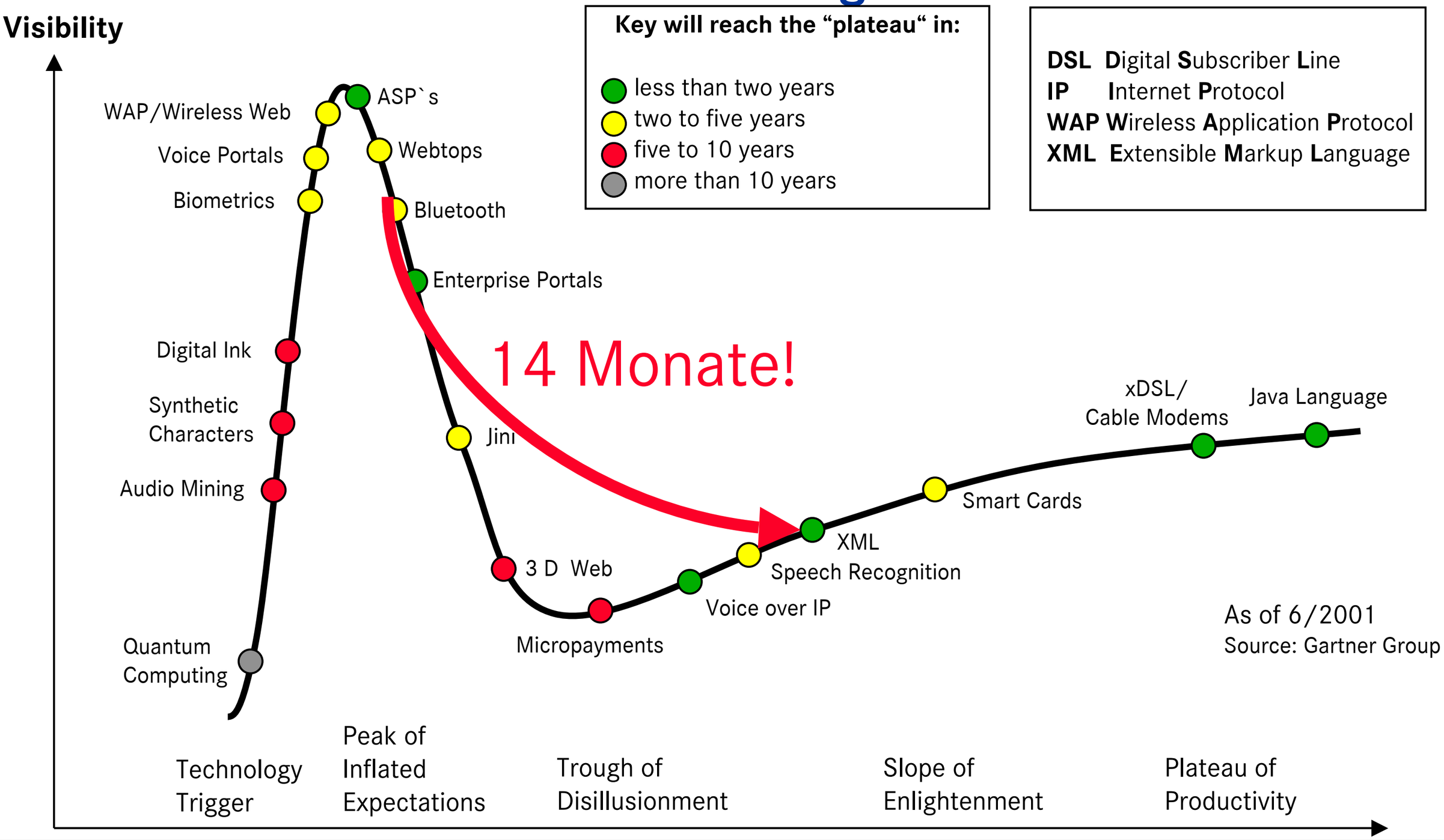


Key will reach the "plateau" in:

- less than two years
- two to five years
- five to 10 years
- more than 10 years

DSL Digital Subscriber Line
IP Internet Protocol
WAP Wireless Application Protocol
XML Extensible Markup Language

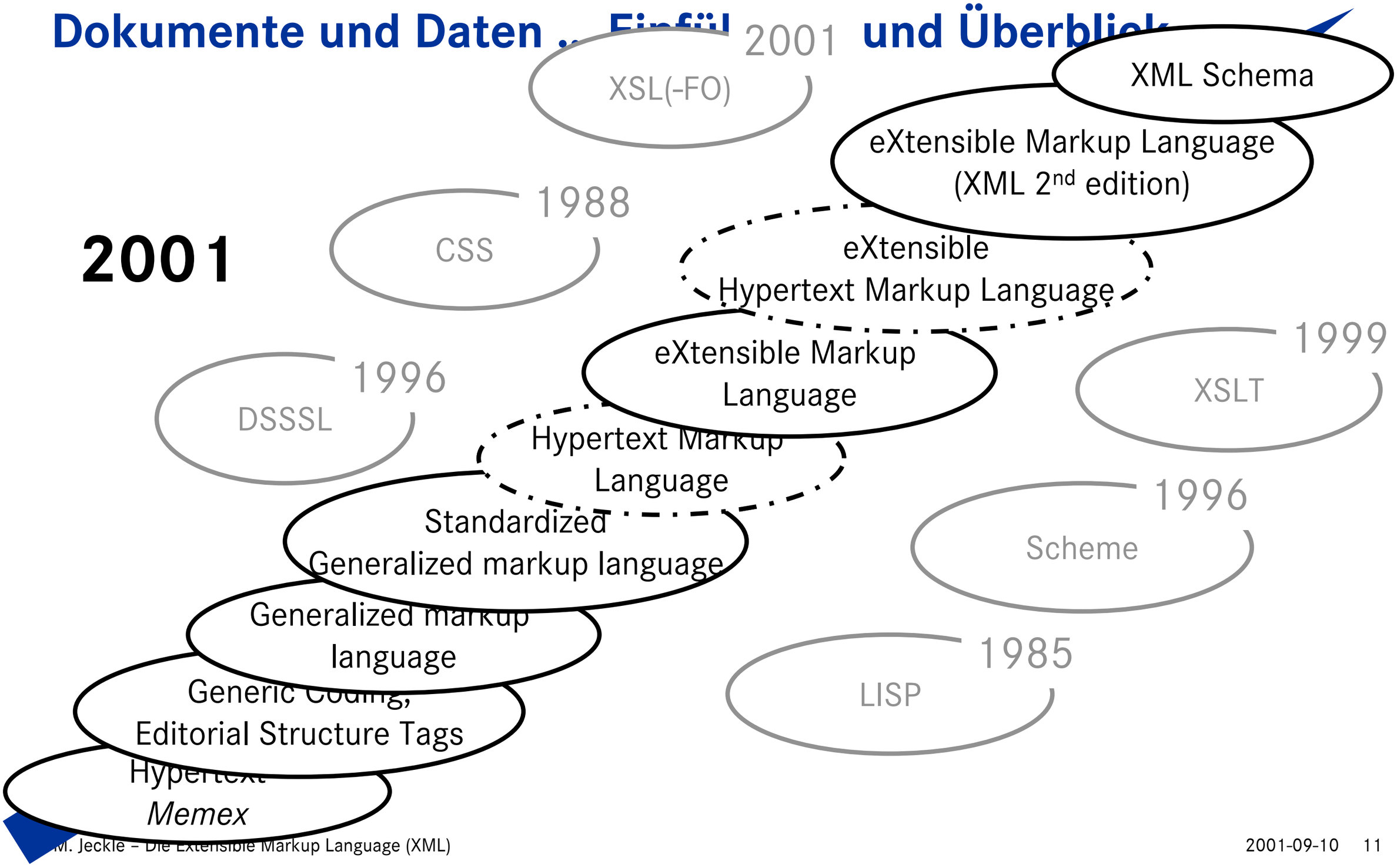
Dokumente und Daten ... Einführung und Überblick



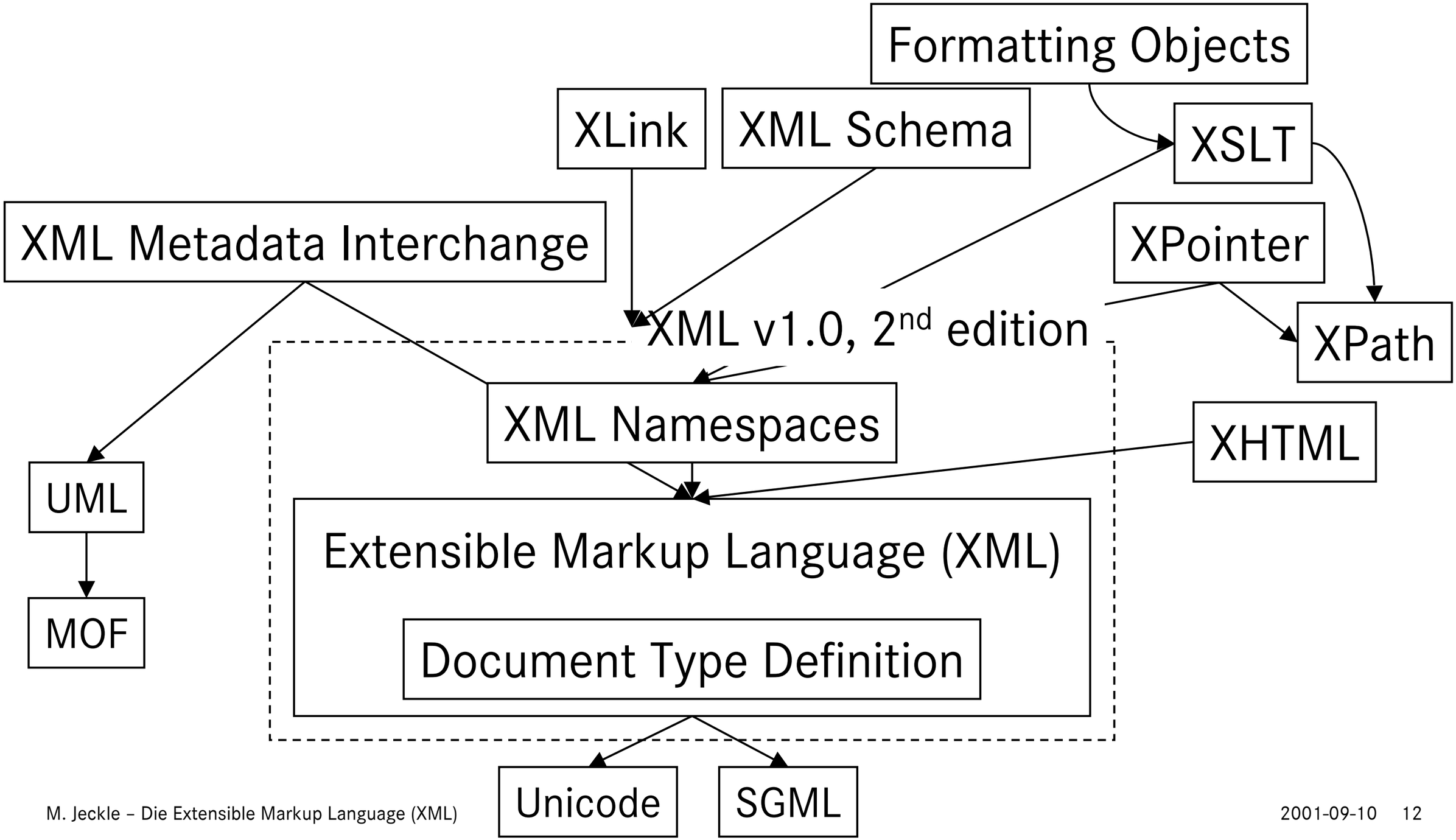
Dokumente und Daten ... Einführung und Überblick

- XML ist eine Methode, um strukturierte Daten in einer Textdatei darzustellen
- XML ist Text, aber nicht zum Lesen
- XML ist ausführlich, was aber kein Problem darstellt
- XML sieht fast aus wie HTML, ist aber kein HTML
- XML ist eine Familie von Techniken
- XML ist eine Untermenge des ISO-Standards SGML
- XML ist international und unterstützt beliebige Alphabete (Unicode)
- XML ist neu, aber nicht so neu
- XML ist lizenzfrei, plattform- und herstellerunabhängig, und gut unterstützt
- XML ist inzwischen weit verbreitet

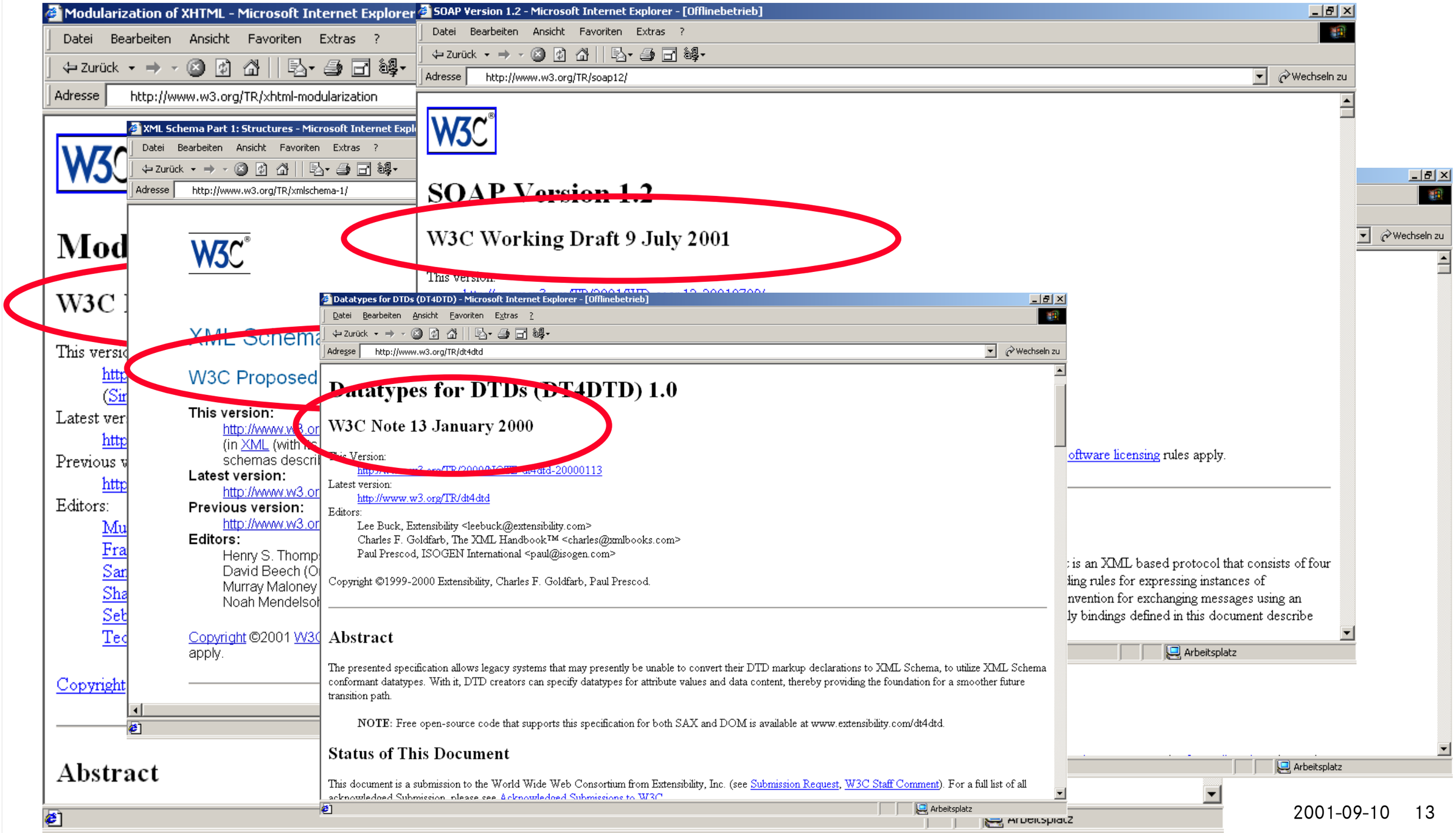
Dokumente und Daten .. Einfü und Überblick



Dokumente und Daten ... Einführung und Überblick

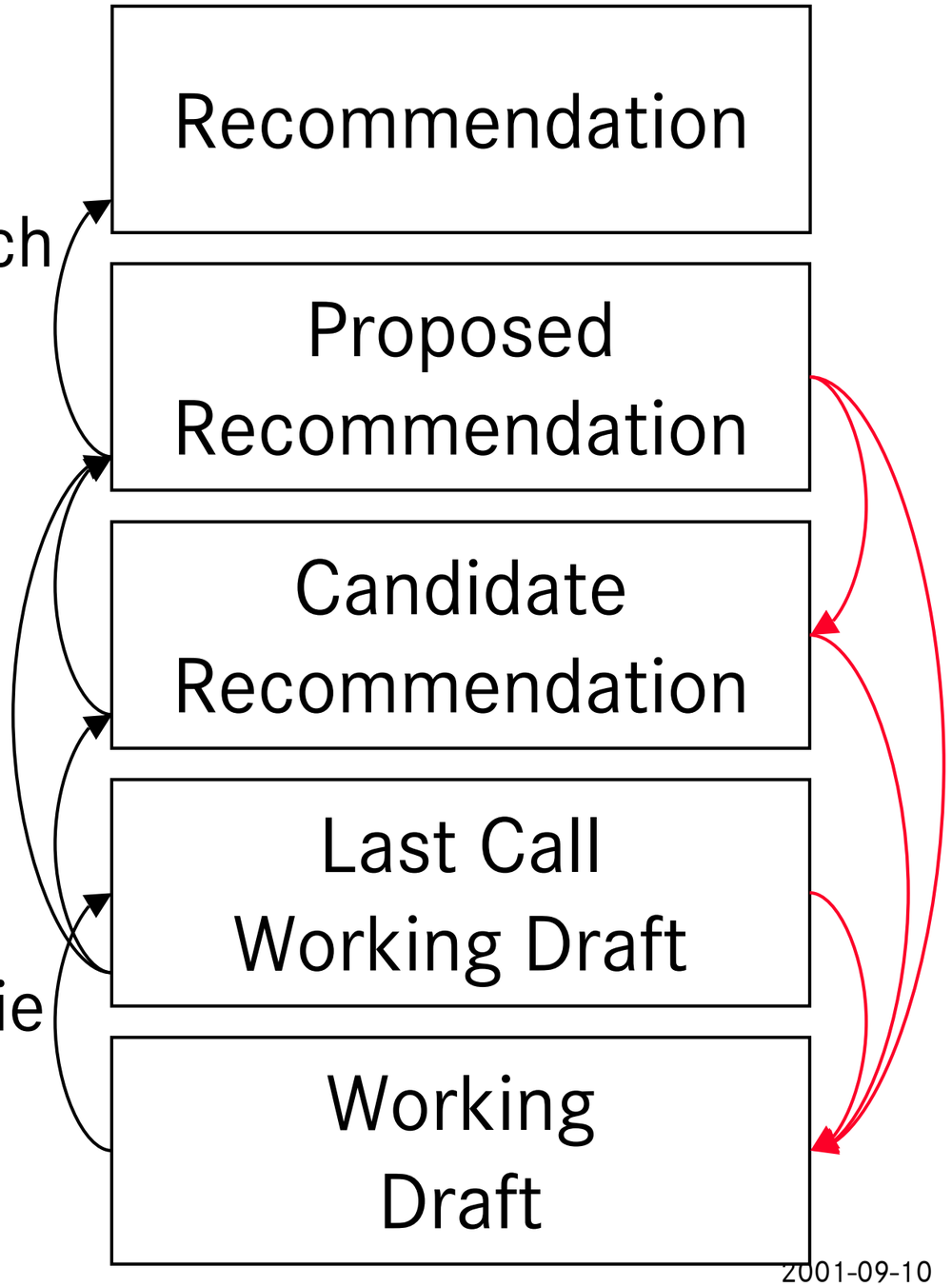


Dokumente und Daten ... Einführung und Überblick



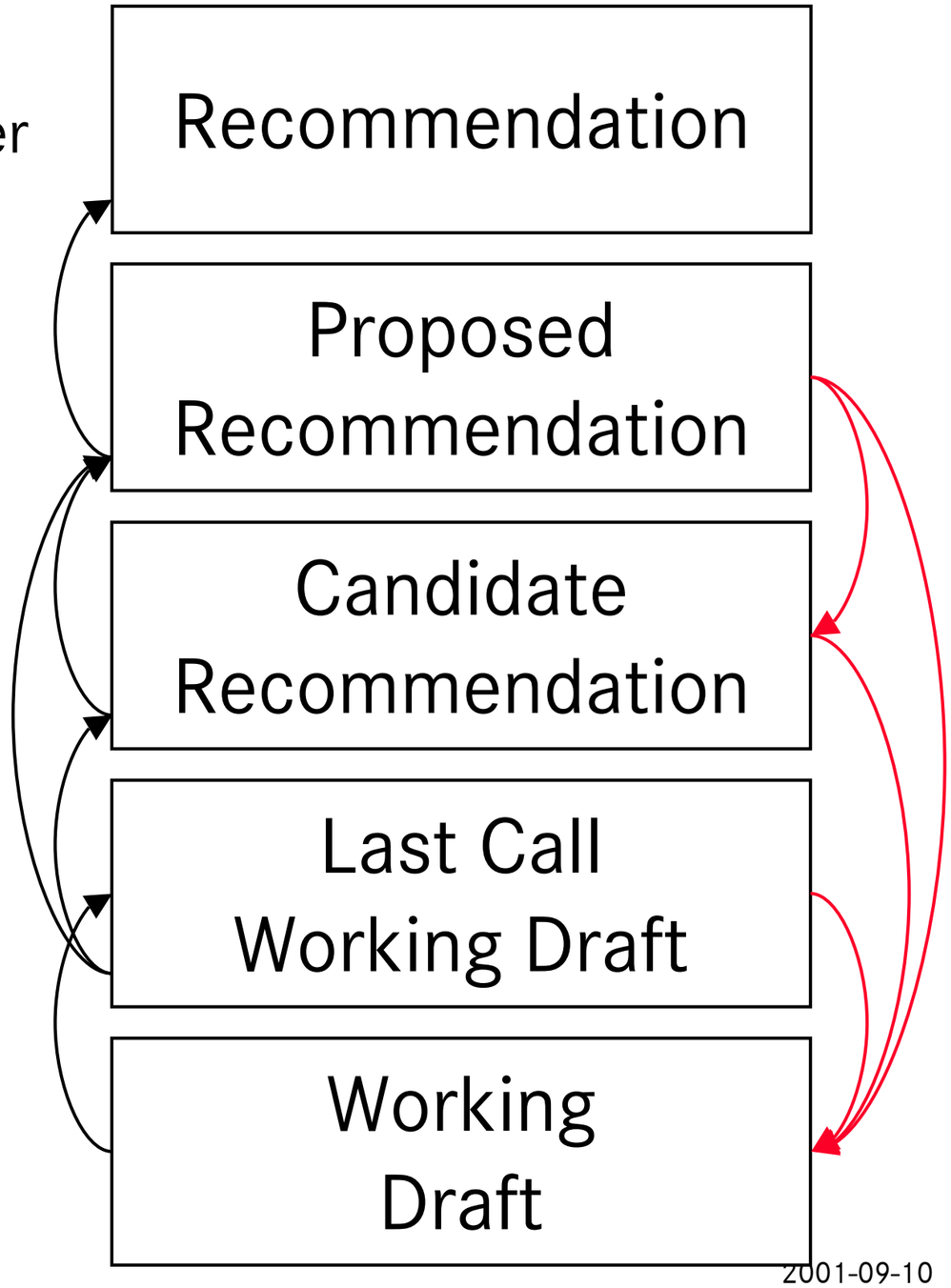
Dokumente und Daten ... Einführung und Überblick

- **Note:** Allgemeines datiertes Dokument zur Vorstellung einer Idee oder neuen Technik; es kann sich auch um einen formalen Kommentar handeln.
- **Working Draft:** Interimsstand einer W3C-Arbeitsgruppe zu einem Thema das durch das W3C weiter bearbeitet wird.
- **Last Call Working Draft:** Markiert die Erreichung der im Anforderungs-dokument festgelegten Ziele.



Dokumente und Daten ... Einführung und Überblick

- **Candidate Recommendation:** Direktor des W3C bestätigt die Erreichung der definierten Anforderungen, bzw. erklärt sich mit der Nichterreichung bestimmter Anforderungen einverstanden.
- **Proposed Recommendation (PR):** Bestandteile der PR sind umgesetzt. Idealerweise sollte die Arbeitsgruppe zwei interoperable Implementierungen jedes Spezifikationsbestandteils vorweisen können.
- **Recommendation:** Direktor erklärt sich mit dem Grade der Unterstützung durch das Advisory Committee einverstanden und erklärt PR zum Recommendation und damit zum offiziellen W3C-Standard.



Gliederung

- Dokumente und Daten ...
  - Einführung und Überblick
  - **Strukturelle Grundkonzepte**
  - Namensräume
  - Dokument-Typ-Definitionen (DTD)
  - XML-Schema
  - XML und das Web

Dokumente und Daten ... Strukturelle Grundkonzepte

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

- Ein **XML-Dokument** ist ein Datenstrom (der nicht zwingend als Datei vorliegen muß), welcher den Strukturierungsprinzipien der Extensible Markup Language genügt.

- Ein **XML-Prozessor** ist eine maschinelle Komponente (typischerweise: Software), die zum Lesen eines XML-Dokuments eingesetzt wird.

  Er erlaubt Zugriff auf den Inhalt und die Struktur des XML-Dokuments.

Dokumente und Daten ... Strukturelle Grundkonzepte

- Ein XML-Dokument enthält (informal):
  - (optional) die XML-Kopfzeile
  - Wurzelelement
  - Inhaltselemente
  - Attribute
  - Namensräume
  - textuelle Inhalte
  - Kommentare
  - Notationen
  - Entitäten
  - CDATA-Bereiche

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

- *Wohlgeformtes XML-Dokument*:

  Ein textartiges Objekt, dessen Inhalt folgenden Anforderungen genügt:
  - Das XML-Dokument nutzt eine DTD,
    oder enthält die Deklaration standalone="yes"
  - Zu jedem Start-Tag existiert ein Ende-Tag.
    Bei leeren Elementen können diese zu einem Tag zusammenfallen.
  - Korrekte Elementschachtelung,
    d.h. Elemente überlappen einander nicht.
  - Genau ein Wurzelelement.
  - Alle Attributwerte sind in einfachen oder doppelten Anführungszeichen.
  - Kein Element enthält zwei oder mehr Attribute desselben Namens.
  - Keine Kommentare oder Processing Instructions innerhalb von Tags.
  - Kommentare beginnen und enden mit genau zwei Bindestrichen.
  - Die Sonderzeichen < und & treten nicht innerhalb von
    Elementeninhalten oder Attributwerten auf.

Dokumente und Daten ... Strukturelle Grundkonzepte

- **XML Information Set** definiert die strukturellen Grundprimitive eines XML-Dokuments.
  Er bildet das *Metamodell der Extensible Markup Language*
- Ein XML-Dokument enthält (formal):
  - Document Information Item
  - Element Information Item
  - Attribute Information Item
  - Character Information Item
  - Namespace Information Item
  - Comment Information Item
  - ...

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

Document Information Item

character encoding scheme = ISO-8859-1
base URI =
standalone =
version = 1.0

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

Document Information Item

character encoding scheme = ISO-8859-1
base URI =
standalone =
version = 1.0

Element Information Item

namespace name =
local name = Vortrag
prefix =
base URI =

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

Document Information Item

character encoding scheme = ISO-8859-1
base URI =
standalone =
version = 1.0

Element Information Item

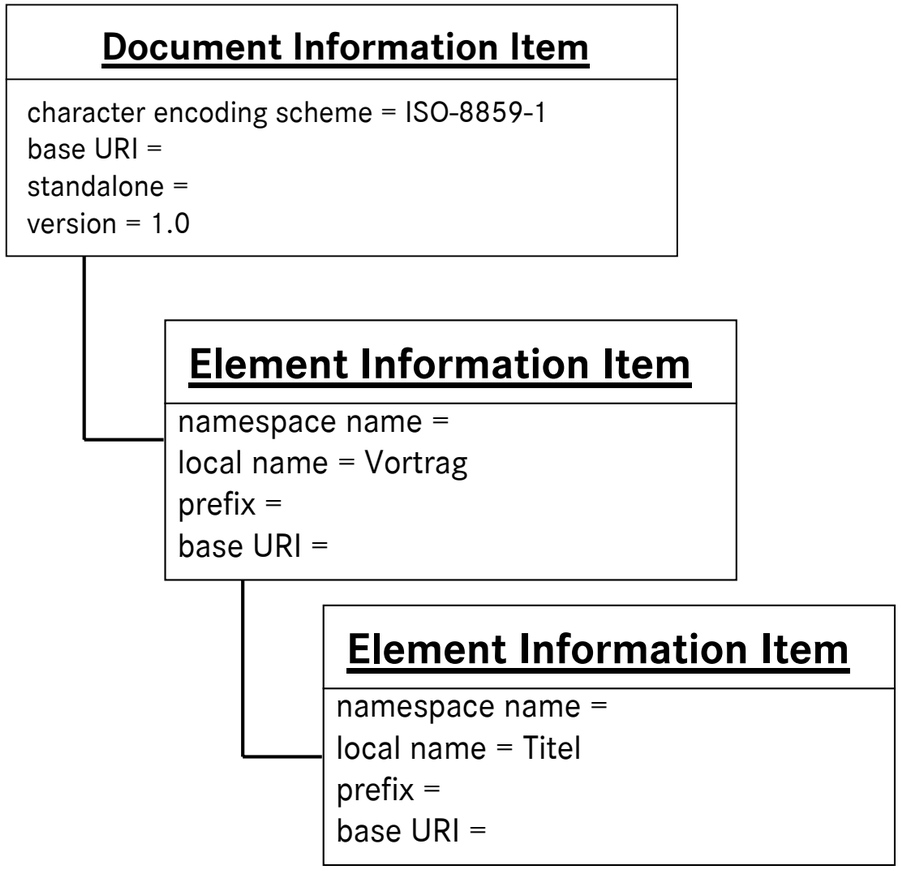
namespace name =
local name = Vortrag
prefix =
base URI =

Element Information Item

namespace name =
local name = Titel
prefix =
base URI =

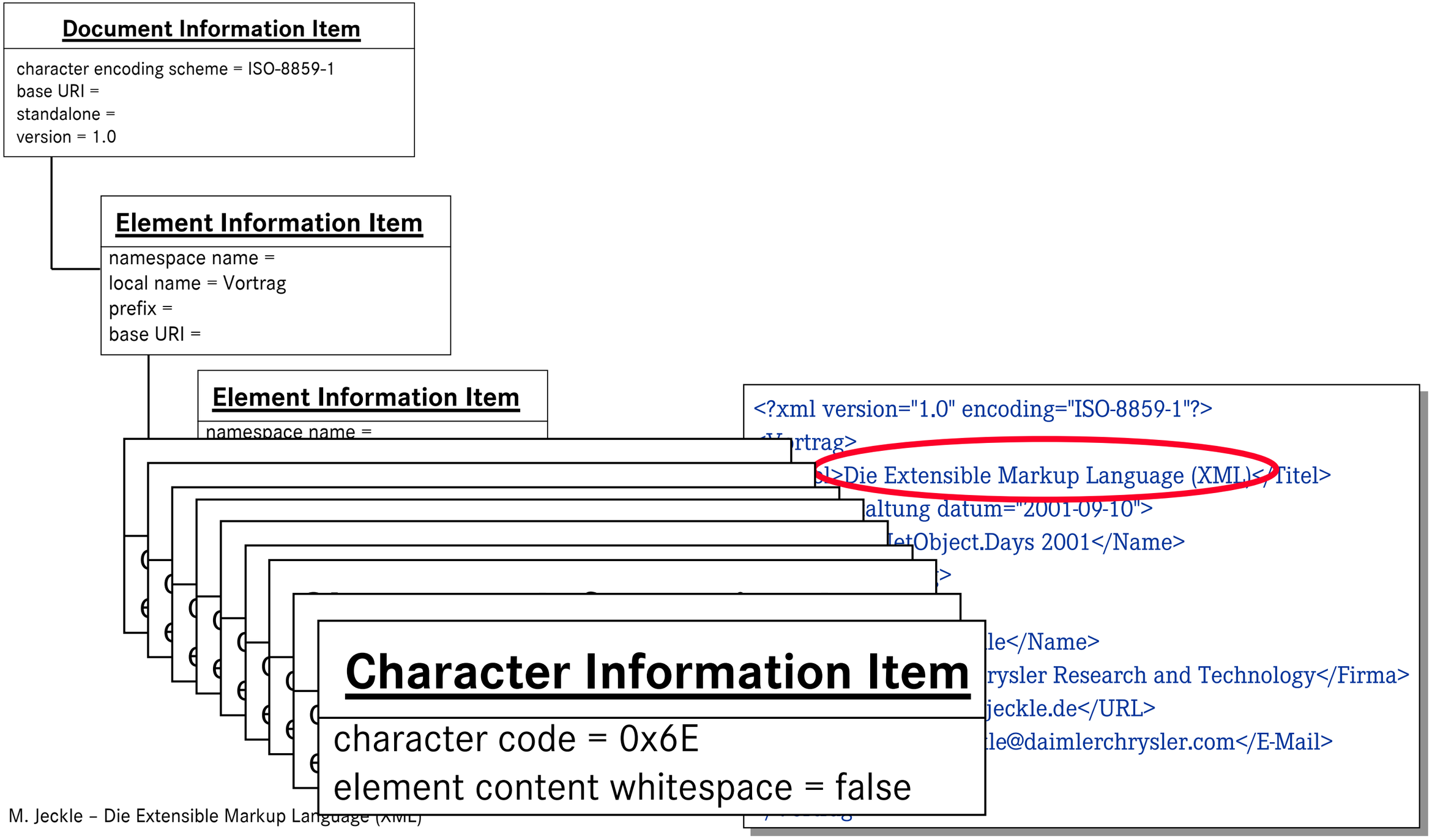
```
?xml version="1.0" encoding="ISO-8859-1"?>
Vortrag>
<Titel>Die Extensible Markup Language (XML)</Titel>
<Veranstaltung datum="2001-09-10">
  <Name>NetObject.Days 2001</Name>
</Veranstaltung>
<Referent>
  <Name>Mario Jeckle</Name>
  <Firma>DaimlerChrysler Research and Technology</Firma>
  <URL>http://www.jeckle.de</URL>
  <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
</Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

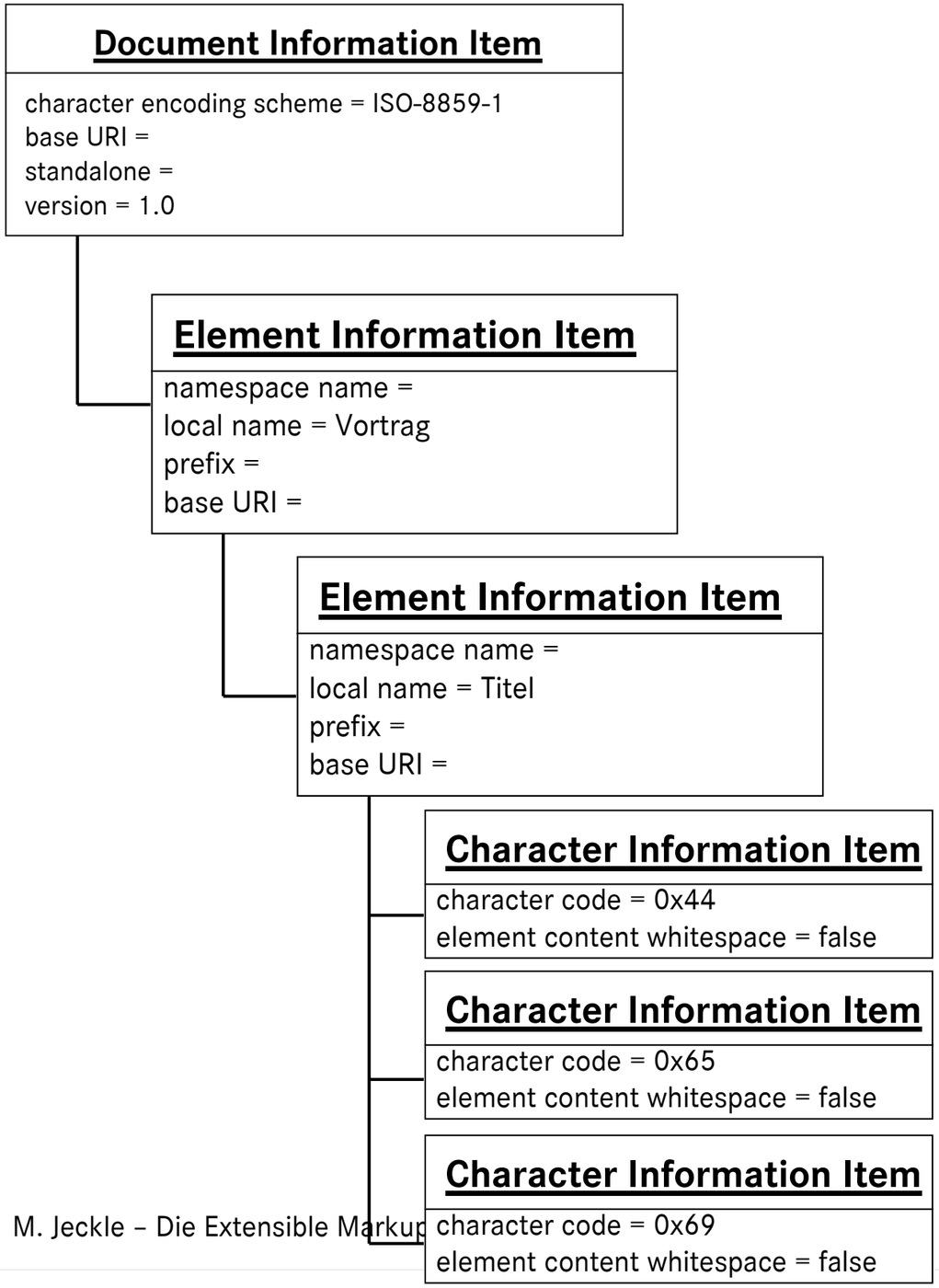



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte



Dokumente und Daten ... Strukturelle Grundkonzepte




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

Document Information Item

character encoding scheme = ISO-8859-1
base URI =
standalone =
version = 1.0

Element Information Item

namespace name =
local name = Vortrag
prefix =
base URI =

Element Information Item

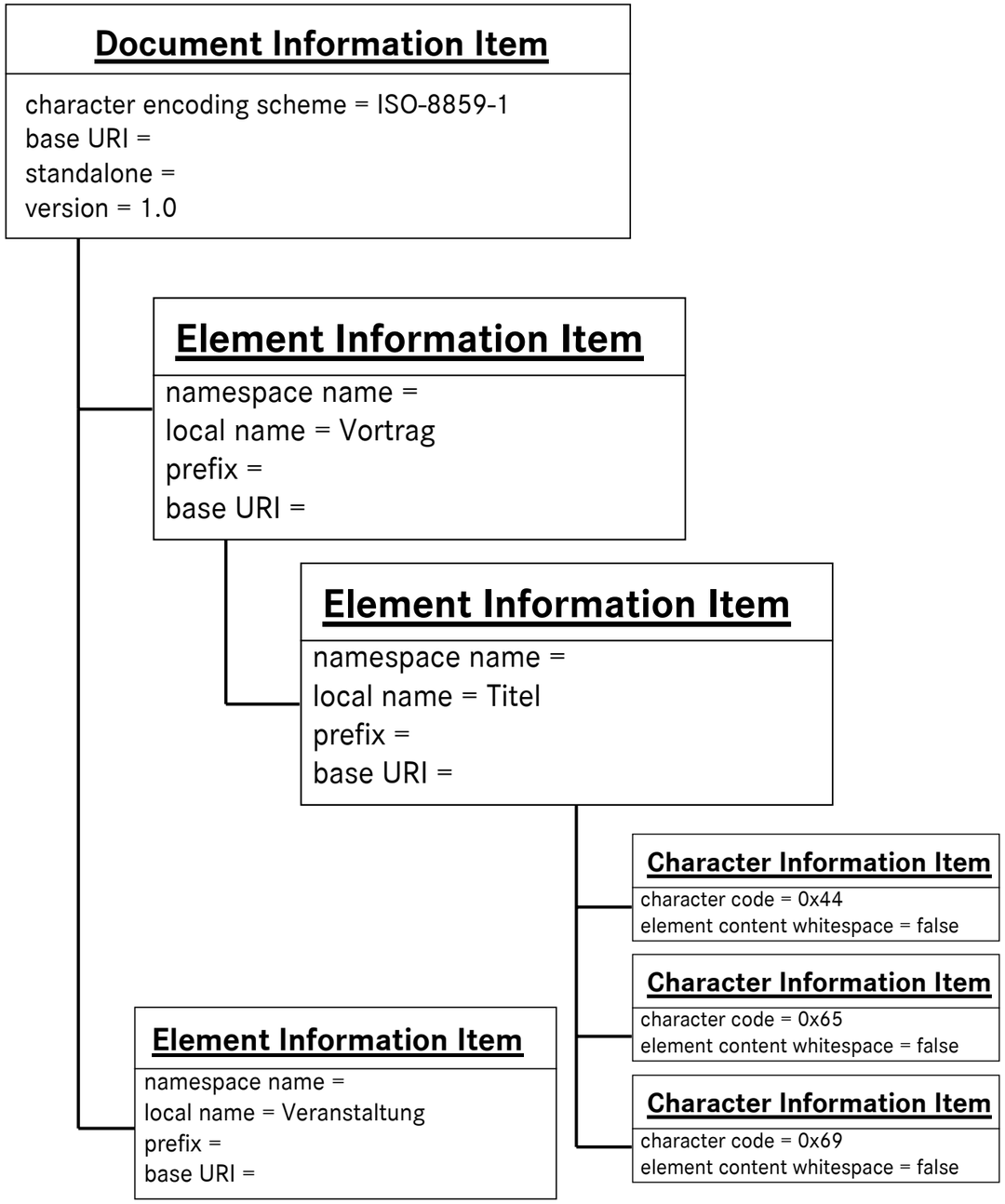
namespace name =
local name = Titel

Element Information Item

namespace name =
local name = Veranstaltung
prefix =
base URI =

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

Document Information Item

character encoding scheme = ISO-8859-1
base URI =
standalone =
version = 1.0

Element Information Item

namespace name =
local name = Vortrag
prefix =
base URI =

Element Information Item

namespace name =
local name = Titel
prefix =
base URI =

Character Information Item

Attribute Information Item

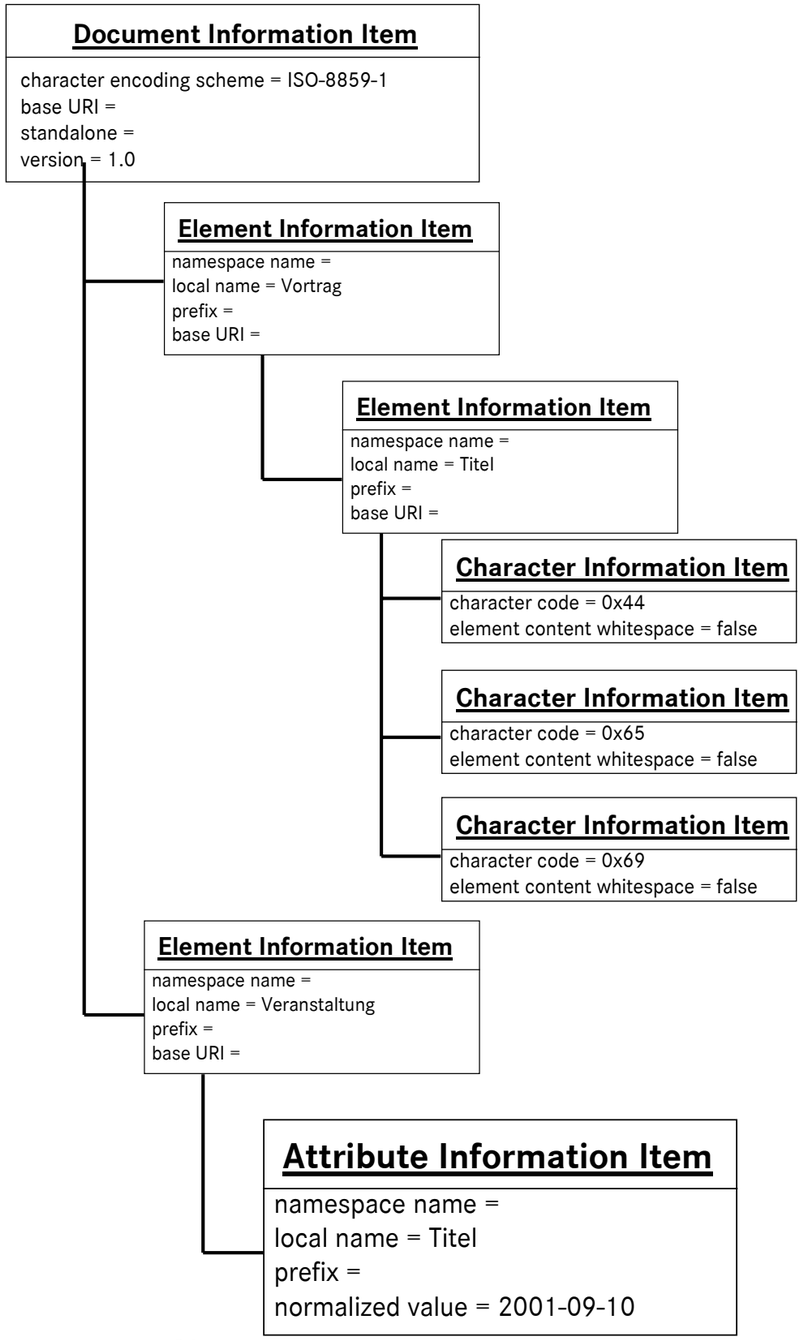
namespace name =
local name = datum
prefix =
normalized value = 2001-09-10

Elem

names
local n
prefix
base URI =

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  tel>Die Extensible Markup Language (XML)</Titel>
  eranstaltung datum="2001-09-10">
  Name>NetObject.Days 2001</Name>
  /eranstaltung>
  eferent>
  Name>Mario Jeckle</Name>
  Firma>DaimlerChrysler Research and Technology</Firma>
  URL>http://www.jeckle.de</URL>
  E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>Die Extensible Markup Language (XML)</Titel>
  <Veranstaltung datum="2001-09-10">
    <Name>NetObject.Days 2001</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

Dokumente und Daten ... Strukturelle Grundkonzepte

- Zusammenfassung: *XML Information Set*
  - Präzisiert die durch die XML Recommendation gegebene physische Repräsentation von XML-Daten semantisch
  - Abstraktes Modell zur Beschreibung logischer XML-Strukturen
  - Infoset ist im wesentlichen ein Baum
  - Baumknoten werden *Information Item* genannt
  - Liefert Begriffswelt für beliebige wohlgeformte XML-Dokumente
  - Wird durch andere W3C-Standards verwendet
  - Üblicherweise durch einen Parsingvorgang erzeugt

Gliederung

- Dokumente und Daten ...
  - Einführung und Überblick
  - Strukturelle Grundkonzepte
  - **Namensräume**
  - Dokument-Typ-Definitionen (DTD)
  - XML-Schema
  - XML und das Web

Strukturelle Grundkonzepte ... Namensräume

- Problem der (globalen) Eindeutigkeit von Attribut- und Element-bezeichnern in XML
- Lösungsmöglichkeit 1: (*klassisches* Vorgehen in EDI, STEP, ...)
  Standardisierung (d.h. normative Festlegung) der Bezeichner

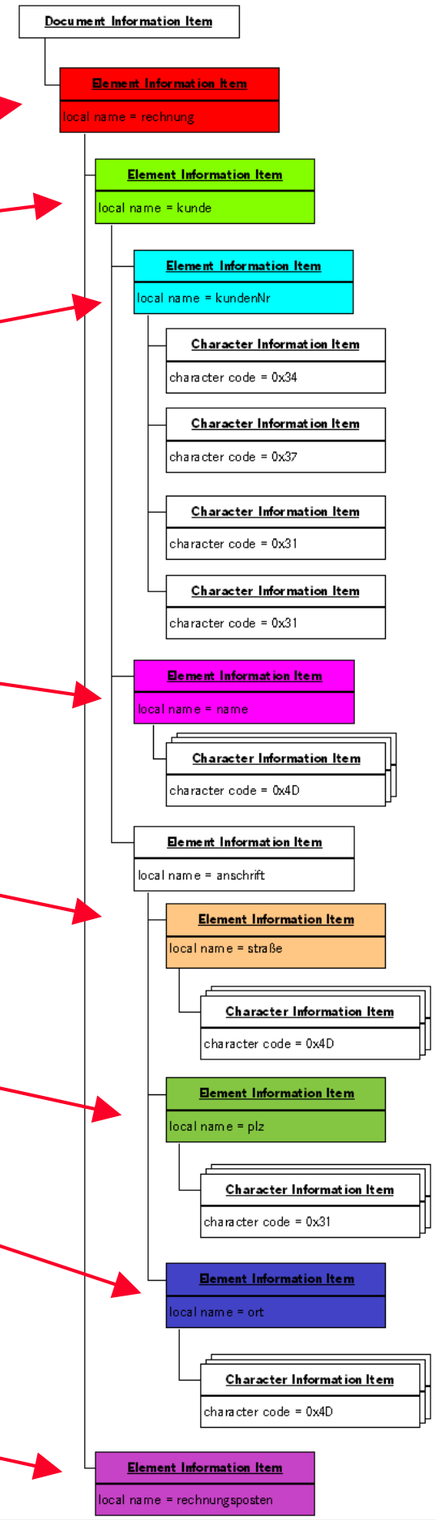
➡ Praktisch unmöglich, eine globale Einigung hinsichtlich Semantik und Inhalt zu erzielen

- Lösungsmöglichkeit 2: (ähnlich DNS)
  Prinzipiell freie Vergabe, bei zwingender zentraler Registrierung

➡ Nicht praktikabel hinsichtlich des abzuschätzenden Aufwandes sowie der abzusehenden (Rechts-)Streitigkeiten

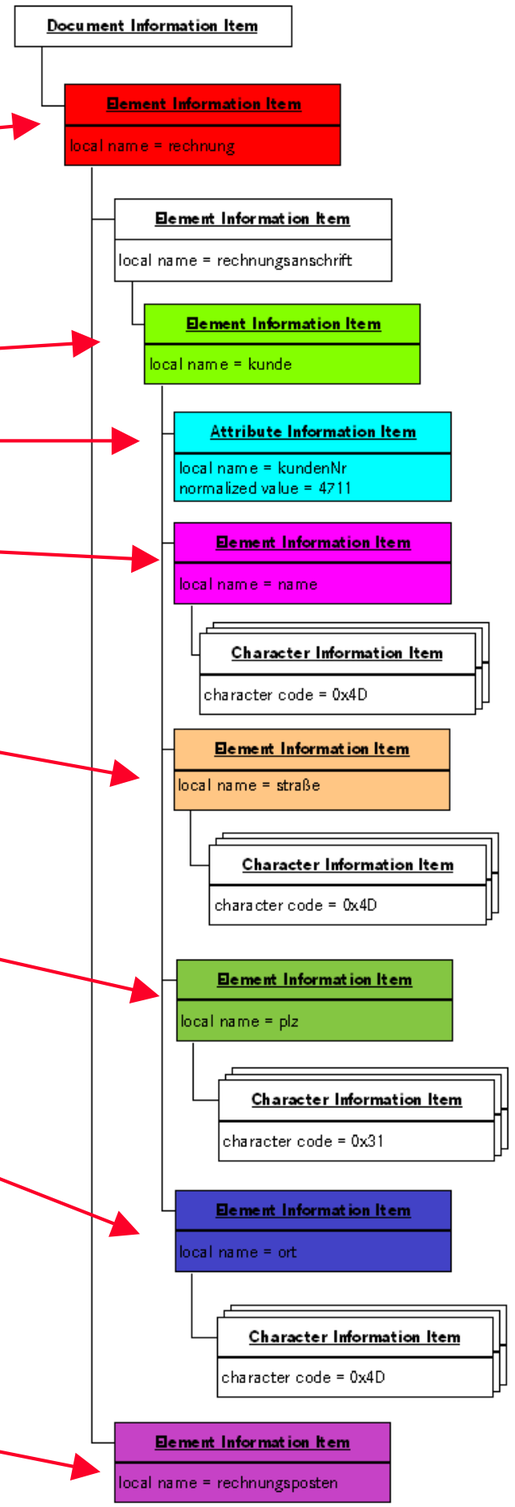
Strukturelle Grundkonzepte ... Namensräume

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rechnung>
  <kunde>
    <kundenNr>4711</kundenNr>
    <name>Max Mustermann</name>
    <anschrift>
      <straße>Musterplatz 1</straße>
      <plz>12345</plz>
      <ort>Musterstadt</ort>
    </anschrift>
  </kunde>
  <rechnungsposten>
...
</rechnung>
```



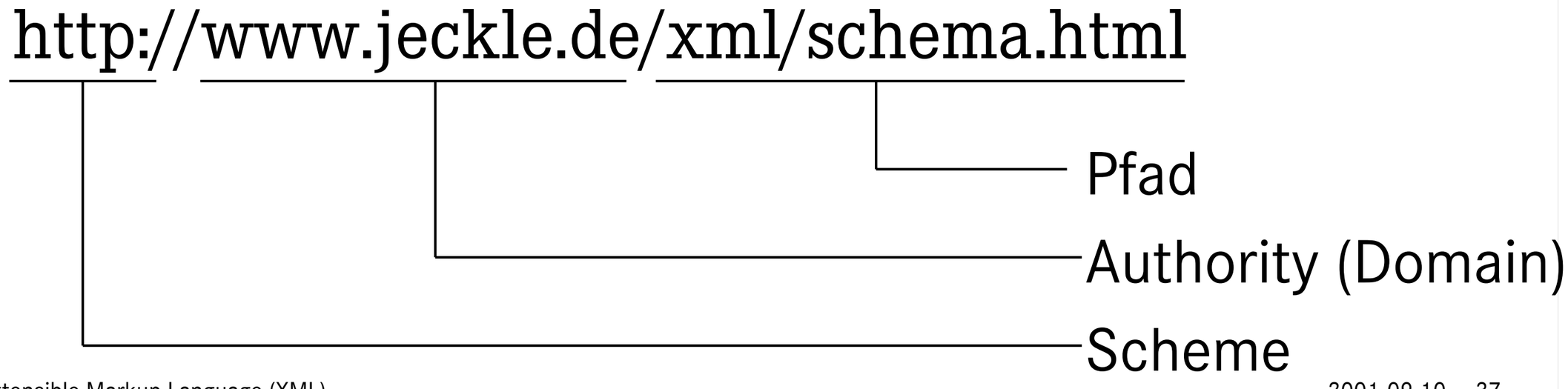
Strukturelle Grundkonzepte ... Namensräume

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rechnung>
  <rechnungsanschrift>
    <kunde kundenNr="4711">
      <name>Max Mustermann</name>
      <straße>Musterplatz 1</straße>
      <plz>12345</plz>
      <ort>Musterstadt</ort>
    </kunde>
  </rechnungsanschrift>
  <lieferanschrift>
  </lieferanschrift>
  <rechnungsposten>
...
</rechnung>
```



Strukturelle Grundkonzepte ... Namensräume

- Lösungsidee:
  (Wiederbe-)Nutzung der URI als Namenszusatz der Attribut- und Elementbezeichner
  - Zentral verwalteter Domain-Anteil garantiert generell Eindeutigkeit
  - Dezentral verwalteter Pfad-Anteil garantiert größtmögliche Flexibilität



Strukturelle Grundkonzepte ... Namensräume

- *Naive Lösung*: Erweiterung der Elementnamen

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<http://www.example.com/sales:rechnung>
  <http://www.example.com/sales:kunde>
    <http://www.example.com/sales:kundenNr>4711</http://www.example.com/sales:kundenNr>
    <http://www.example.com/sales:name>Max Mustermann</http://www.example.com/sales:name>
    <http://www.example.com/sales:anschrift>
      <http://www.example.com/sales:straße>Musterplatz 1</http://www.example.com/sales:straße>
      <http://www.example.com/sales:plz>12345</http://www.example.com/sales:plz>
      <http://www.example.com/sales:ort>Musterstadt</http://www.example.com/sales:ort>
    </http://www.example.com/sales:anschrift>
  </http://www.example.com/sales:kunde>
  <http://www.example.com/sales:rechnungsposten>
...
</http://www.example.com/sales:rechnung>
```

Strukturelle Grundkonzepte ... Namensräume

- *Naive Lösung*: Erweiterung der Elementnamen
  - Umständliche Notation,
    senkt Lesbarkeit dramatisch
  - Vergrößert Speicherplatzaufwand signifikant
    (im Beispiel rund 150%!)
  - Problematisch hinsichtlich Domain- oder Pfadänderungen
  - Verstoß gegen XML-Syntax;
    Pfadtrenner „/“ ist innerhalb Elementnamen nicht zugelassen!

Strukturelle Grundkonzepte ... Namensräume

- Standardisierte Lösung:
  - Zweistufiges Vorgehen:
    1. Bindung einer URI an ein frei wählbares Präfix durch ein standardisiertes Attribut

```
<rechnung xmlns:rch="http://www.examples.com/sales">
```

    2. Qualifizierung der Element- oder Attributnamen mit dem definierten Präfix

```
<rch:rechnung xmlns:rch="http://www.examples.com/sales">
  <rch:kunde>
    <rch:kundenNr>4711</rch:kundenNr>
  </rch:kunde>
...
```

Strukturelle Grundkonzepte ... Namensräume

- Zusammenfassung: *XML Namensräume*
  - Mächtiger, flexibler und einfach zu handhabender Mechanismus zur Gewährleistung der Namenseindeutigkeit von Attributen und Elementen
  - Grundvoraussetzung der XML-basierten *content syndication*
  - Abwärtskompatibel zu Namensraum-losen Dokumenten
  - Parser- und Werkzeugunterstützung verfügbar
  - Eingesetzt in verschiedenen XML-Standards und -Vokabularen
  - W3C-Recommendation seit 1999
  - Vorsicht mit *fragment URIs!*

Gliederung

- Dokumente und Daten ...
  - Einführung und Überblick
  - Strukturelle Grundkonzepte
  - Namensräume
  - **Dokument-Typ-Definitionen (DTD)**
  - XML-Schemasprachen
  - XML und das Web

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Dient zur Festlegung der strukturellen und inhaltlichen Merkmale eines Dokuments
- Eigene (einfache) Syntax
- Untermenge des SGML-Pendants
  - ... daher eher Dokumenten- denn Daten-orientiert
- Grammatik eines XML-Dokuments
- Steigerung der *Wohlgeformtheit* zur *Gültigkeit*
- Wird durch *validierende Parser* verarbeitet
- Werkzeugunterstützung verfügbar

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Verknüpfung zwischen Dokument und DTD

```
<!DOCTYPE ProjektVerwaltung
   SYSTEM "file:///C:/dtds/projektverwaltung.dtd">
<ProjektVerwaltung>

   ...
</ProjektVerwaltung>
```

- DOCTYPE-Deklaration referenziert wahlweise eine lokale Ressource (SYSTEM) oder eine allgemein bekannte (und potentiell zentral abgelegte) DTD (PUBLIC)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Elementdefinition
  - Elementname
  - *Inhaltsmodell* (in Klammern)

```
<!ELEMENT ProjektVerwaltung
   (Person, Projekt)>
```

...im XML-Dokument

```
...
<ProjektVerwaltung>
   <Person>...</Person>
   <Projekt>...</Projekt>
</ProjektVerwaltung>
...
```

...in InfoSet-Darstellung

Element Information Item
namespace name =
local name = ProjektVerwaltung
prefix =
base URI =

Element Information Item
namespace name =
local name = Person
prefix =
base URI =

Element Information Item
namespace name =
local name = Projekt
prefix =
base URI =

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Elementdefinition

```
<!ELEMENT ProjektVerwaltung  (Person+ , Projekt+ )>
<!ELEMENT Person (Vorname+, Nachname, Qualifikationsprofil?)>
```

- Vielfachheit:
  - + mindestens einmaliges Auftreten {1, 2, 3 ...}
  - * beliebiges Auftreten {0, 1, 2, ...}
  - ? höchstens einmaliges Auftreten {0, 1}
- Gruppierung durch Klammerung, Operatoren "," und "|"

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Elementdefinition

```
<!ELEMENT Projekt EMPTY>
<!ELEMENT Ueberschrift (#PCDATA)>
```

- Elemente ohne Kindelemente (*„leere Elemente“)* haben Inhaltsmodell EMPTY

- Unstrukturierter textueller Inhalt: PCDATA *parsed character data*

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Elementdefinition

```
<!ELEMENT Qualifikationsprofil
          (#PCDATA | Qualifikation | Leistungsstufe)*>
```

- *Gemischter Inhalt* („mixed content“), kombiniert unstrukturierten textuellen und durch markup strukturierten Inhalt

...im XML-Dokument

```
...
<Qualifikationsprofil>IT-Kompetenz verschiedene Betriebssysteme und
<Leistungsstufe>professionelle</Leistungsstufe>
  <Qualifikation>Programmierung</Qualifikation> verschiedener Programmiersprachen
  <Qualifikation>Entwickler</Qualifikation>von 1988-1990
  <Qualifikation>Projektleiterfunktion</Qualifikation> von 1990-93 im X42-Projekt
  in Abteilung AB&amp;amp;C </Qualifikationsprofil>
...
```

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Attributdefinition

```
<!ATTLIST Projekt

    ProjektNr ID #REQUIRED

    date CDATA #IMPLIED

    Projektleiter IDREF #REQUIRED

    Mitarbeiter IDREFS #REQUIRED>
```

- Projekt → zugehöriges Element
- ProjektNr ID #REQUIRED → zwingend anzugebender, Dokument-weit eindeutiger Schlüssel
- date CDATA #IMPLIED → Zeichenketten-artiger (*CDATA*), optional anzugebender Inhalt
- Projektleiter IDREF #REQUIRED → zwingend anzugebende Schlüsselreferenz
- Mitarbeiter IDREFS #REQUIRED → Menge von zwingend anzugebenden Schlüsselreferenzen

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Attributdefinition

```
<!ATTLIST ProjektVerwaltung

    version CDATA #FIXED "1.0">

<!ATTLIST Projekt

    budget CDATA "10000"

<!ATTLIST Person

    Gehaltsgruppe (1 | 1a | 2) "1a" >
```

konstante Belegung

Zeichenketten-artiger (*CDATA*), Inhalt mit Vorgabebelegung. Implizit optional; validierender Parser setzt bei fehlendem Wert automatisch Vorgabewert ein.

Aufzählungstyp mit Vorbelegung

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Entitäten

  ... bekannt aus HTML

```
<!ENTITY auml   "&#228;">
  <!– latin small letter a with diaeresis, U+00E4 ISOlat1 –>
```

  - Einfacher *handlicher* Textersetzungsmechanismus
  - in XML wegen frei wählbarer Codierung nicht mehr von herausgehobener Bedeutung
  - Bei unsachgemäßer Handhabung sehr fehlerträchtig

Dokumente und Daten ... Dokument-Typ-Definition (DTD)

- Zusammenfassung *Dokument-Typ-Definitionen*:
  - Grundlage der Erstellung eigener Vokabulare
  - Von SGML übernommene proprietäre (d.h. nicht XML-)Syntax
  - Dokument-orientiert
    - Rudimentäres (auf Text-Typen beschränktes) Typsystem
    - Nur einfache Strukturierungsmechanismen
    - Keine direkte Unterstützung der XML-Namensräume
  - Gemeinsam mit einem XML-Dokument Eingabe eines validierenden Parsers
  - Werkzeugunterstützung verfügbar

Gliederung

- Dokumente und Daten ...
  - Einführung und Überblick
  - Strukturelle Grundkonzepte
  - Namensräume
  - Dokument-Typ-Definitionen (DTD)
  - **XML-Schema**
  - XML und das Web

Dokumente und Daten ... XML-Schema

- DTDs im Daten-orientierten Umfeld ...
  - Streng hierarchische Sichtweise oftmals zu einschränkend. Gewünscht: Netzartig verknüpfte Elemente und Dokumente
  - Keine Namensraumunterstützung
  - Keine Validierbarkeit der Grammatik
  - Proprietäre Syntax der Grammatiksprache
  - Angebotenes Typsystem unzureichend
  - Angebotene Strukturprimitive unzureichend

Dokumente und Daten ... XML-Schema

- Erweiterungsforderungen ...
  - Strukturell ....
    - Namespace Integration (und damit XML 2nd edition)
    - Steuerung der Auftretensreihenfolge
      (Beschränkung und kontrollierte Freigabe)
    - Vererbung (Kopier- und Substitutionssemantik)
    - Wiederverwendungsunterstützung
    - Praktisch einsetzbarer Referenzierungsmechanismus
  - Inhaltlich ....
    - Primitive Datentypen (int, float, boolean, ...)
    - Binärstrukturen
    - Komplexe Datentypen (date, Elemente, Strukturen, ...)
    - Eigendefinierte lexikalische Datentypen
    - Konsistenzsichernde Einschränkungen

Dokumente und Daten ... XML-Schema

- Lösungsalternativen ...

Erweiterungen des bestehenden (SGML-/XML-)DTD-Mechanismus
- Data Types for DTD (DT4DTD)

Wissensbeschreibung
- Document Content Description for XML (DCD)

(*RDF basierte Weiterentwicklung von XML-Data*)

Inspiriert durch XML-API-Entwicklung
- Schema for Object oriented XML (SOX)

XML-Sprachen zur Inhaltsbeschreibung
- Document Definition Markup Language/XSchema (DDML)
- Schematron (XSL-basierte Auswertung der Dokumentstruktur)
- XML-Data/XML-Data Reduced (XDR) *(erster Ansatz noch vor Verabschiedung XML 1.0)*
- Document Structure Description (DSD)

Dokumente und Daten ... XML-Schema

- XML-Sprachen zur Inhaltsbeschreibung ...

  Zwei konkurrierende Philosophien:

  - **Regelbasiert**
    - Definiert Validierungsregeln
    - Bekannteste Implementierung: *Schematron*

  - **Reguläre kontextfreie Grammatik**
    - Definiert alle gültigen Dokumentstrukturen explizit
    - Bekannt aus Programmiersprachen
    - Implementierungen: *XSD, XML-Data, XDR, DSD, ...*
    - Stand 1998/9: Viele verschiedene (konkurrierende) Sprachen
      - Zunehmend kaum noch erkennbare inhaltliche Unterschiede
    - Ab Mai 1999: W3C Arbeitsgruppe entwickelt eigenen Ansatz
      - Trennung zwischen *Struktur* und *Datentypen*
      - Berücksichtigt (nahezu) alle bedeutenden Vorgänger (explizit: DCD, DDML, SOX, XDR, XML-Data)

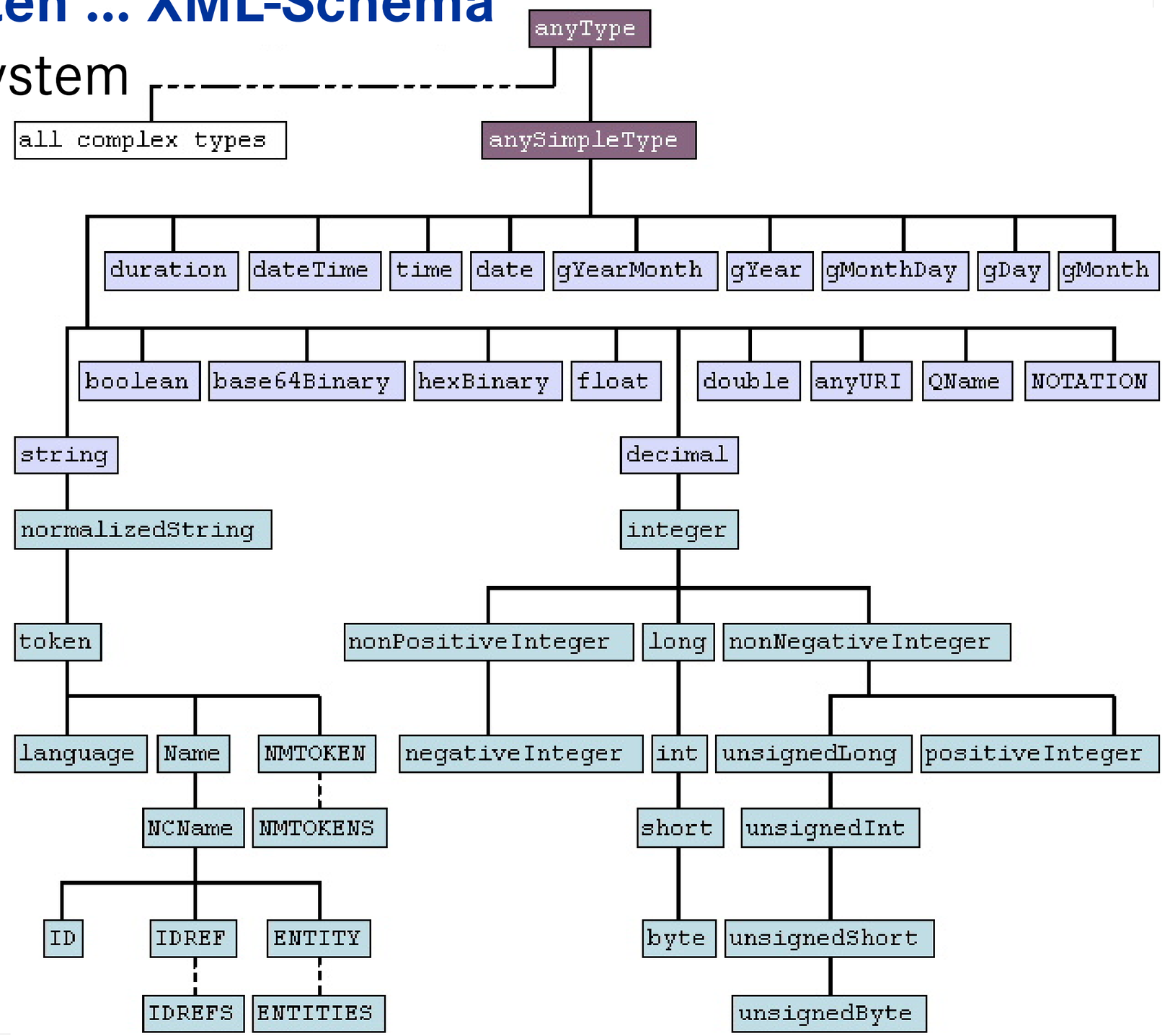
Dokumente und Daten ... XML-Schema

- Mächtigkeit von XML-Schema
  - Attribute und Elemente (wie in DTDs)
  - Namensraum-Unterstützung
  - Atomare Datentypen (int, float, boolean, ...)
  - Anwenderdefinierte
    - atomare Datentypen
      - Einschränkung des Wertebereichs (Domänenrestriktion)
    - lexikalische Muster (reguläre Ausdrücke)
    - Aufzählungstypen
    - Mengentypen
    - komplexe Datentypen (complexType)
  - Vererbung
    - Restriktion und Erweiterung
  - Substitution
  - Erweiterter Schlüsselmechanismus
  - NULL-Werte

Dokumente und Daten ... XML-Schema

- XML-Schema Typsystem

- Ur-typ
- Vordefinierter Primitivtyp
- Vordefinierter abgeleiteter Typ
- Typeinschränkung
- Aggregierter Typ



Dokumente und Daten ... XML-Schema

XML-Schema Typsystem: im Detail

ID	test, XYZ	XSD-Darstellung des DTD-Typen ID. Zugelassen sind alle Ausprägungen der Namespaceproduktion 4 (NCName). ID ist eine einschränkende Spezialisierung des Typs NCName
IDREF	test, XYZ	XSD-Darstellung des DTD-Typen IDREF. Zugelassen sind alle Ausprägungen der Namespaceproduktion 4 (NCName). IDREF ist eine einschränkende Spezialisierung des Typs NCName
IDREFS	test1 test2 test4, test3 test5	XSD-Darstellung des DTD-Typen IDREFS. Zugelassen sind Listen aus white space separierten Ausprägungen der Namespaceproduktion 4 (NCName). IDREFS ist eine nichtleere Aufzählung von IDREF-Ausprägungen
ENTITY		XSD-Darstellung des DTD-Typen ENTITY. Zugelassen sind alle Satzformen die der Produktion NCName der XML-Namensräume entsprechen und als ungeparste Entität definiert sind. ENTITY ist eine einschränkende Spezialisierung des Typs NCName
ENTITIES		XSD-Darstellung des DTD-Typen ENTITIES. Zugelassen sind Listen aus white space separierten Ausprägungen des Typs ENTITY. ENTITIES ist eine nichtleere Aufzählung von ENTITY-Ausprägungen
NOTATION		XSD-Darstellung des DTD-Typen NOTATION. Zur Verwendung dieses Typs in einem Schema muß eine Ableitung von NOTATION durch den Anwender definiert werden.
NMTOKEN	US, Deutschland	XSD-Darstellung des DTD-Typen NMTOKEN. Ausprägungen dieses Typs müssen konform zur Produktion 7 der XML-Spezifikation sein. NMTOKEN ist eine einschränkende Spezialisierung des Typs token
NMTOKENS	US UK Aus, Ger	XSD-Darstellung des DTD-Typen NMTOKENS. Zugelassen sind Listen aus white space separierten Ausprägungen des Typs NMTOKEN. NMTOKENS ist eine nichtleere Aufzählung von NMTOKEN-Ausprägungen

Dokumente und Daten ... XML-Schema

- XML-Schema Typsystem: im Detail

string	Hello
World	Jedes beliebige Unicode Symbol gemäß XML-Syntaxproduktion 2
normalizedString	&#20Hello&#20;World	Jedes beliebige Unicode Symbol außer Zeilenvorschub, Wagenrücklauf und Tabulatoren. normalizedString ist eine einschränkende Spezialisierung des Typs string
token	Hello World	Jeder normalizedString, unter Weglassung führender, abschließender und mehrfacher Leerzeichen (#x20), sowie Zeilenvorschüben (#xA) und Tabulatoren (#x9). token ist eine einschränkende Spezialisierung des Typs normalizedString
Name	aName, _helloWorld, :notThatGood	XML Name gemäß Syntaxproduktion 5. Name ist eine einschränkende Spezialisierung des Typs token
QName	xsd:element, xsl:template, foo	Durch Namensraumpräfix qualifizierter Name gemäß Produktion 6 der XML Namespace Recommendation
NCName	aName, _anotherName, X	Name, der keinen Doppelpunkt enthält (non colonized name), gemäß Produktion 4 der XML Namespace Recommendation

Dokumente und Daten ... XML-Schema

- XML-Schema Typsystem: im Detail

decimal	-1.23, 12678967.543233, +100000.00, 210	Wertebereich: $i*10^{-n}$, mit i, n aus integer, n>=0 Ein Prozessor muß mindestens 18 Dezimalstellen unterstützen
long	-1, 0, 12678967543233, +100000	Wertebereich: 2^{63} <= long <= 2^{63}-1 long ist eine einschränkende Spezialisierung des Typs integer
int	-1, 0, 126789675, +100000	Wertebereich: -2^{31} <= int <= 2^{31}-1 int ist eine einschränkende Spezialisierung des Typs long
byte	-1, 42	Wertebereich: -2^{7} <= byte <= 2^{7}-1 byte ist eine einschränkende Spezialisierung des Typs short
integer	-1, 0, 12678967543233, +100000	Wertebereich: entspricht der mathematischen Menge der ganzen Zahlen ($\mathbb{Z}$) integer ist eine einschränkende Spezialisierung des Typs decimal

Dokumente und Daten ... XML-Schema

- XML-Schema Typsystem: im Detail

nonNegativeInteger	1, 0, 12678967543233, +100000	Wertebereich: 0 <= nonNegativeInteger nonNegativeInteger ist eine einschränkende Spezialisierung des Typs integer
positiveInteger	1, 12678967543233, +100000	Wertebereich: entspricht der mathematischen Menge der natürlichen Zahlen ($\mathbb{N}$) positiveInteger ist eine einschränkende Spezialisierung des Typs nonNegativeInteger
negativeInteger	-1, -12678967543233, -100000	Wertebereich: {..., -2, -1}, die unendliche Menge der negativen Zahlen negativeInteger ist eine einschränkende Spezialisierung des Typs nonPositiveInteger
unsignedLong	0, 12678967543233, 100000	Wertebereich: 0 <= unsignedLong <= 2^{64}-1 unsignedLong ist eine einschränkende Spezialisierung des Typs nonNegativeInteger
unsignedInt	0, 1267896754, 100000	Wertebereich: 0 <= unsignedInt <= 2^{32}-1 unsignedInt ist eine einschränkende Spezialisierung des Typs unsignedLong
unsignedShort	0, 12678, 10000	Wertebereich: 0 <= unsignedShort <= 2^{16}-1 unsignedShort ist eine einschränkende Spezialisierung des Typs unsignedInt
unsignedByte	0, 126, 100	Wertebereich: 0 <= unsignedByte <= 2^{8}-1 unsignedByte ist eine einschränkende Spezialisierung des Typs unsignedShort

Dokumente und Daten ... XML-Schema

XML-Schema Typsystem: im Detail

time	13:20:00-05:00, 13:20:00.000	Uhrzeit, die täglich wiederkehrt, ausgedrückt im Format gemäß ISO 8601
date	2001-09-10, 20010910	Datumsformat: CCYY-MM-DD, gemäß ISO 8601
dateTime	2001-09-10T09:00:00.000+02:00	Zeitpunkt, ausgedrückt durch Datum und Uhrzeit; beide gemäß ISO 8601 codiert.
gDay	—05, —31	Darstellung eines wiederkehrenden Tages eines Monats gemäß ISO 8601
gMonth	–03–, –12–	Monatsformat: –MM– gemäß ISO 8601
gYear	1999, 2000	Darstellung von Jahren des gregorianischen Kalenders gemäß ISO 8601
gYearMonth	2001-09	Darstellung eines Monats eines bestimmten Jahres des gregorianischen Kalenders gemäß ISO 8601
duration	P1Y2M3DT10H30M12.3S Zeitraum von eim Jahr, zwei Monaten, drei Tagen, zehn Stunden, 30 Minuten und 12,3 Sekunden	Nach Größe (Signifikanz) geordnete Koordinate im sechs-dimensionalen Raum aus Jahr, Monat, Tag, Stunde, Minute und Sekunde. Formatdefinition laut ISO 8601

Dokumente und Daten ... XML-Schema

- XML-Schema Typsystem: im Detail

float double	-1E4, 1267.43233E12, 12.78e-2, 12, INF	32-Bit-Zahl mit einfacher Genauigkeit gemäß IEEE 754-1985. Wertebereich: $m * 2^e$, wobei m und e integer-Elemente mit $m <= 2^{24}$, und $-149 <= e < 104$ sind.
boolean	true, false, 1, 0	Unterstützung der klassischen zweiwertigen Logik
anyURI	http://www.jeckle.de foo:bar/baz	Jede gemäß IETF RFC 2396 bzw. IETF RFC 2732 gültige URI
language	en-GB, en, de-de	Sprachcodierung gemäß IETF RFC 1766 und XML Recommendation language identification. Die Identifikationsnamen werden durch ISO 639 sowie ISO 3166 definiert. language ist eine einschränkende Spezialisierung des Typs token
base64Binary	aGVsbG8gd29ybGQh	Base64-Darstellung eines beliebigen Binär-interpretierten Inhaltes gemäß IETF RFC 2045. Datenvolumen erhöht sich um Faktor 1,36
hexBinary	0FB7	Hexadezimale Darstellung beliebiger Binär-interpretierter Inhalte. Datenvolumen erhöht sich um Faktor 2
anyType	1, 2.3, aGVsb, 06b8f45, test&#20;für&#20;anyType &#0A; <sentence>the quick brown <animal>fox</animal>...	Allgemeinster Datentyp. Konzeptionell bildet er die Vereinigung aller angebotenen XSD-Typen.

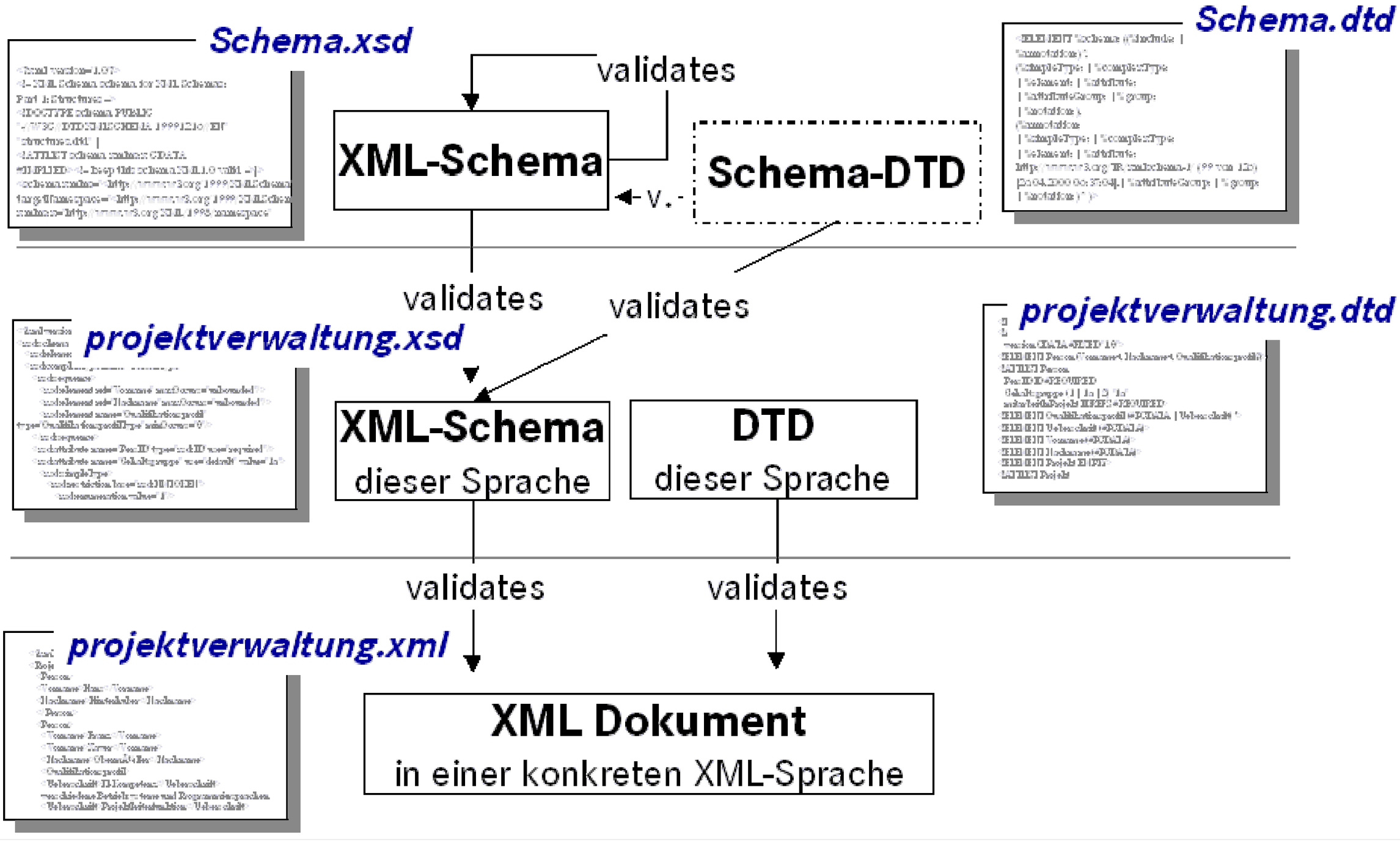
Dokumente und Daten ... XML-Schema

- Schemareferenz
  - Analog der SYSTEM-Variante der DOCTYPE-Deklaration

```
<?xml version="1.0"?>
<ProjektVerwaltung
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.jeckle.de/vorlesung/xml/examples/projektverwaltung.xsd">
...
</ProjektVerwaltung>
```

  - Deklaration des *schema instance* Namensraumes
  - Nutzung des Attributs schemaLocation zur Referenzierung
  - Zu jedem Dokument kann nur maximal ein Schema existieren

Dokumente und Daten ... XML-Schema



Dokumente und Daten ... XML-Schema

- Elementdefinition
  - einfachster Fall:

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

```
<xsd:element
  name="elementName"
  type="elementType"/>
```

  - Zeichenketten-artiges Inhaltsmodell:

```
<xsd:element
  name="Ueberschrift"
  type="xsd:string"/>
```

Dokumente und Daten ... XML-Schema

- Elementdefinition
  - Inhaltsmodelle:
    - explizite benannte Definition:

```
<xsd:complexType
  name="complexTypeName">
  ....
</xsd:complexType>
```

    - implizite anonyme Definition:

```
<xsd:element name="elementName">
  <xsd:complexType>
    ...
  </xsd:complexType>
</xsd:element>
```

Dokumente und Daten ... XML-Schema

- Elementdefinition
  - leeres Inhaltsmodell:

```
<!ELEMENT Projekt EMPTY>
```

    - implizite anonyme Definition:

```
<xsd:element name="Projekt">
  <xsd:complexType/>
</xsd:element>
```

    - Alternativ:

```
<xsd:complexType name="ctEmpty"/>

<xsd:element name="Projekt" type="ctEmpty"/>
```

Dokumente und Daten ... XML-Schema

- Elementdefinition

```
<!ELEMENT Person (Vorname+, Nachname, Qualifikationsprofil?)>
```

  - beliebige Inhaltsmodelle:

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Vorname" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="Nachname" type="xsd:string"/>
      <xsd:element name="Projekt" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:element>
```

  - minOccurs und maxOccurs erlaubt die feingranulare Auftretenshäufigkeitssteuerung.
    Reformulierung und Erweiterung von: ?, *, +

Dokumente und Daten ... XML-Schema

- Elementdefinition
  - beliebige Inhaltsmodelle:

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Vorname" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="Nachname" type="xsd:string"/>
      <xsd:element name="Projekt" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:element>
```

    - sequence erzwingt im XML-Dokument dieselbe Element-reihenfolge, die im Schema angegeben ist.
      Analog dem ","-Operator: (...,...,...)
    - all erlaubt das beliebig geordnete Auftreten der Elemente im XML-Dokument
      Analog: (...|...|...)*. Jedoch mit Häufigkeitskontrolle
    - choice erlaubt das auftreten genau eines Elements der Auswahl.
      Analog: (...|...|...)

Dokumente und Daten ... XML-Schema

- Elementdefinition, *mixed content*

```
<!ELEMENT Qualifikationsprofil
          (#PCDATA | Qualifikation | Leistungsstufe)*>
```

```
<xsd:element name = "Qualifikationsprofil">
  <xsd:complexType mixed = "true">
    <xsd:sequence>
      <xsd:element name = "Qualifikation" type = "xsd:string"
         minOccurs = "0" maxOccurs = "unbounded"/>
      <xsd:element name = "Leistungsstufe" type = "xsd:string"
         minOccurs = "0" maxOccurs = "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Dokumente und Daten ... XML-Schema

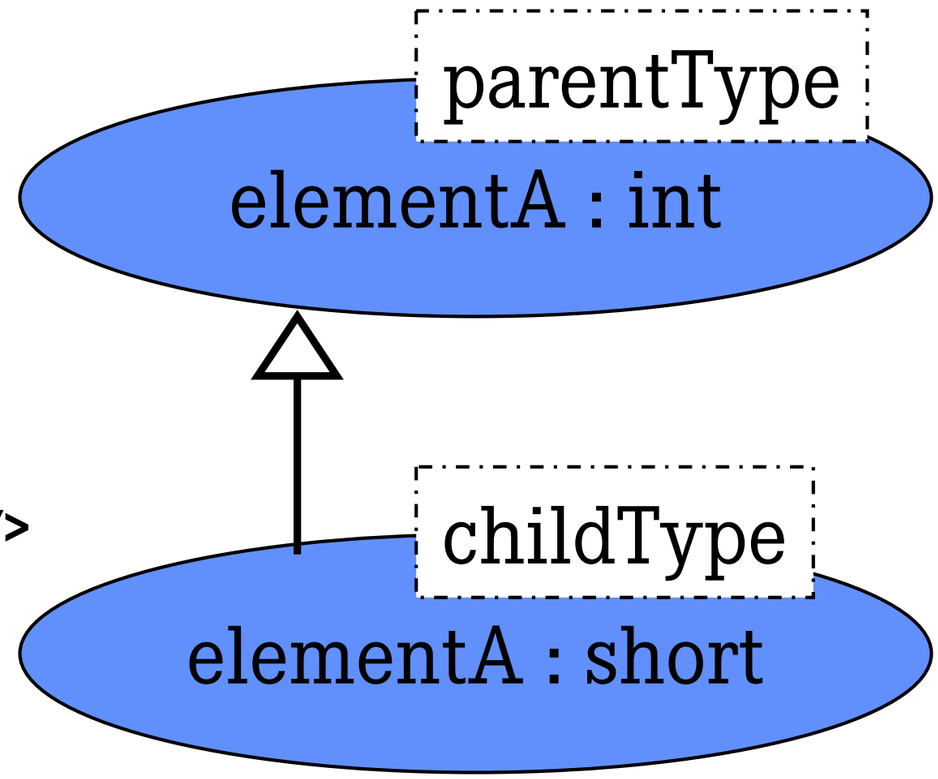
- Vererbung
  - Ableitung durch Einschränkung (*derivation by restriction*)
    Der erbende Subtyp gibt eine engere Definition des Supertypen
  - Ableitung durch Erweiterung (*derivation by extension*)
    Der erbende Subtyp erweitert die Definition des Supertypen
  - Ableitung durch Redefinition (*derivation by redefinition*)
    Der erbende Subtyp verändert die Typdefinition des Supertypen

Dokumente und Daten ... XML-Schema

- Vererbung: derivation by restriction

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="parentType">
  <xsd:sequence>
    <xsd:element name="elementA" type="xsd:int"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="childType">
<xsd:complexContent>
  <xsd:restriction base="parentType">
      <xsd:sequence>
        <xsd:element name="elementA" type="xsd:short"/>
      </xsd:sequence>
    </xsd:restriction>
</xsd:complexContent>
</xsd:complexType>
```

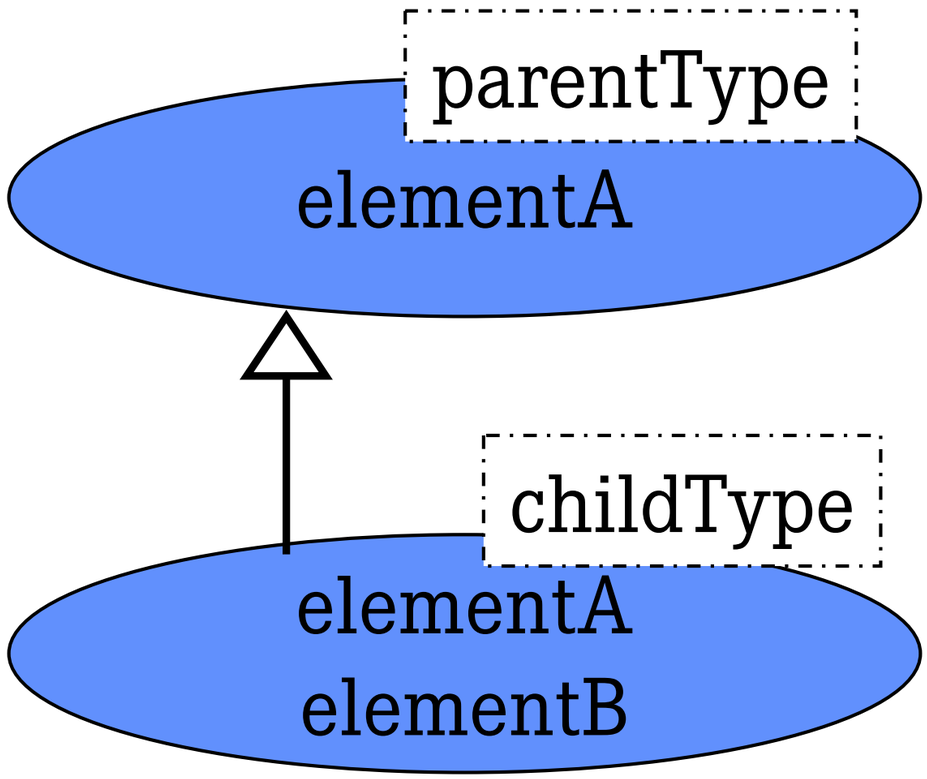
```
<xsd:element name="usage1" type="parentType"/>
<xsd:element name="usage2" type="childType"/>

</xsd:schema>
```



Dokumente und Daten ... XML-Schema

- Vererbung: derivation by extension

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="parentElement">
    <xsd:sequence>
      <xsd:element name="elementA"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="childElement">
    <xsd:complexContent>
      <xsd:extension base="parentElement">
        <xsd:sequence>
          <xsd:element name="elementB"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```



Dokumente und Daten ... XML-Schema

- Vererbung: derivation by redefinition

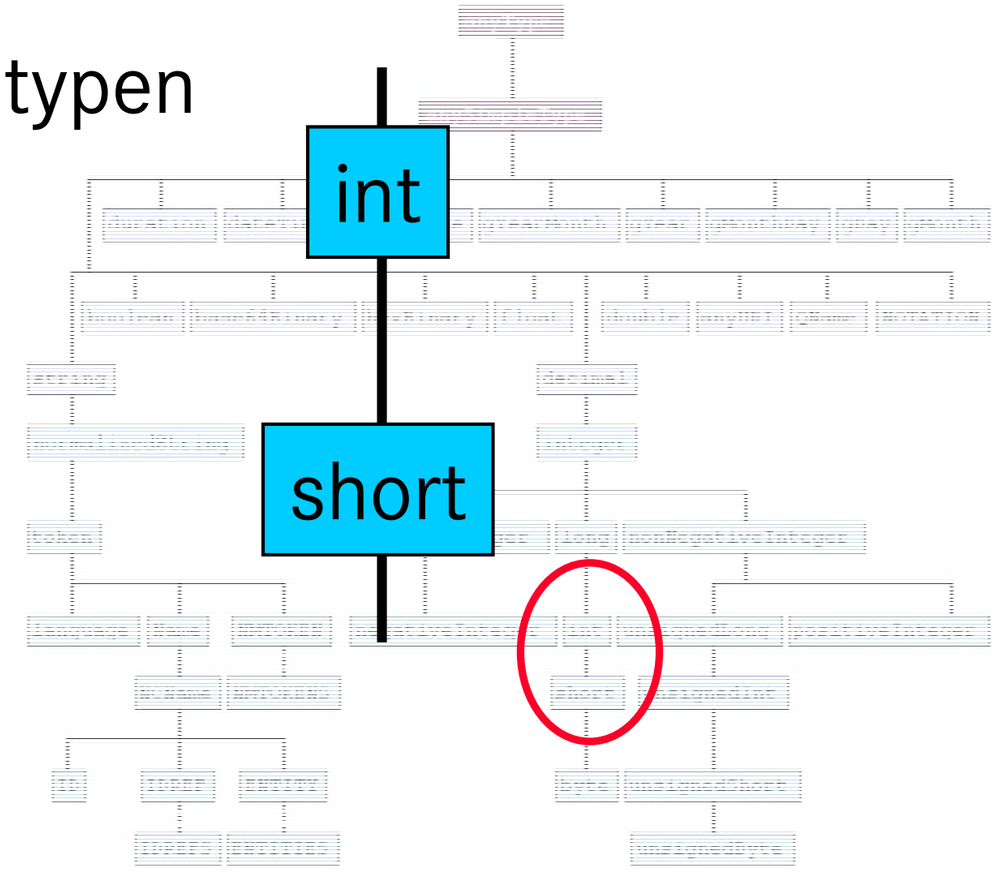
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="aType">
   <xsd:sequence>
     <xsd:element name="elementA"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="usage1" type="aType"/>
```

```
<xsd:complexType name="aType">
   <xsd:complexContent>
<xsd:extension base="aType">
      <xsd:sequence>
        <xsd:element name="elementB"/>
      </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="usage2" type="aType"/>
</xsd:schema>
```

Dokumente und Daten ... XML-Schema

- Erweiterung des Typsystems
  - Definition eigener einfacher Datentypen

```
<xsd:simpleType name="short">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="-32768"/>
    <xsd:maxInclusive value="32767"/>
  </xsd:restriction>
</xsd:simpleType>
```



Dokumente und Daten ... XML-Schema

- Erweiterung des Typsystems
  - Definition eigener einfacher Datentypen
    - Durch vollständige Aufzählung

```
<xsd:simpleType name="ampelfarben">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="rot"/> ...
</xsd:simpleType>
```

    - Durch Typeaggregation

```
<xsd:simpleType name="WarenkorbElemente">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>
```

    - Durch Typvereinigung

```
<xsd:simpleType name="termin">
  <xsd:union memberTypes="xsd:date NamenDerWochentage"/>
</xsd:simpleType>
```

Dokumente und Daten ... XML-Schema

- Erweiterung des Typsystems
  - Definition eigener einfacher Datentypen
    - Durch Beschränkung bestehender Typen
      - Längeneinschränkung

```
<xsd:simpleType name="Postleitzahl">
  <xsd:restriction base="xsd:string">
    <xsd:length value="5"/>
  </xsd:restriction>
```

      - ... sinngemäß Minimal- und Maximallänge
      - Lexikalische Einschränkung durch reguläre Ausdrücke

```
<xsd:simpleType name="gerAutoNummer">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\p{Lu}{1,3}-\p{Lu}{1,2} \p{Nd}{1,4}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Dokumente und Daten ... XML-Schema

- Abschließendes Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superfein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X-363-23"/>
  </artikel>
</bestellung>
```

Dokumente und Daten ... XML-Schema

```
<schema
  xmlns="http://www.w3.org/2001/XMLSchema">
  <element name = "bestellung">
    <complexType mixed= "false">
      <sequence>
        <element ref = "artikel"/>
      </sequence>
    </complexType>
  </element>
  <element name = "nummer" type = "string"/>
  <element name = "benennung" type = "string"/>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superlein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DHM</waehrung>
    </preis>
    <kunde nummer="X 363 23"/>
  </artikel>
</bestellung>
```

freie Elementdefinition
(strukturiertes Element)

Elementverwendung

freie Elementdefinition
(einfaches Element)

Dokumente und Daten ... XML-Schema

```
<element name = "preis">
  <complexType mixed = "false">
    <sequence>
      <element ref = "betrag"/>
      <element ref = "waehrung"/>
    </sequence>
  </complexType>
</element>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <bezeichnung>Wusch Superlein</bezeichnung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X 363 23"/>
  </artikel>
</bestellung>
```

freie Elementdefinition
(strukturiertes Element)

Elementverwendung

Dokumente und Daten ... XML-Schema

```
<element name = "artikel">
  <complexType mixed = "false">
    <sequence>
      <element ref = "nummer"/>
      <element ref = "benennung"/>
      <element ref = "preis"/>
      <element ref = "kunde"/>
    </sequence>
  </complexType>
</element>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superlein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X 363 23"/>
  </artikel>
</bestellung>
```

freie Elementdefinition
(strukturiertes Element)

Elementverwendungen

Dokumente und Daten ... XML-Schema

```
<simpleType name="currency">
    <restriction base="decimal">
        <precision value="8"/>
        <scale value="2" fixed="true"/>
        <minExclusive value="0"/>
    </simpleType>
    <element name = "betrag" type = "currency"/>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <bezeichnung>Wusch Superlein</bezeichnung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X 363 23"/>
  </artikel>
</bestellung>
```

Anwenderdefinierter Datentyp
(abgeleitet von decimal)

Typverwendung

Dokumente und Daten ... XML-Schema

```
<simpleType name="currencyName">
    <restriction base="string">
      <enumeration value="DEM"/>
      <enumeration value="USD"/>
    </restriction>
  </simpleType>

<element name = "waehrung"
            type = "currencyName"/>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <bezeichnung>Wusch Superlein</bezeichnung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DHM</waehrung>
    </preis>
    <kunde nummer="X 363 23"/>
  </artikel>
</bestellung>
```

Anwenderdefinierter Aufzählungstyp

Typverwendung

Dokumente und Daten ... XML-Schema

```
<simpleType name="kundenNummer">
    <restriction base="string">
      <pattern value="\p{Lu}-\d{3}-\d{2}"/>
    </restriction>
  </simpleType>
  <element name = "kunde">
    <complexType mixed = "false">
      <attribute name = "nummer" use = "required"
            type = " kundenNummer "/>
    </complexType>
  </element>
</schema>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superlein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X 363 23"/>
  </artikel>
</bestellung>
```

Anwenderdefinierter Datentyp
(lexikale Definition)

Typverwendung

Dokumente und Daten ... XML-Schema

- Zusammenfassung: *W3C's XML Schema*
  - Grammatik für beliebige XML-Vokabulare
  - Part 1 beschreibt Strukturen und Inhaltseinschränkungen
  - Part 2 definiert Datentypdefinition für Schema Part 1 und weitere XML-Vokabulare
  - Signifikante Erweiterung der DTD-Mächtigkeit, wird diese langfristig ersetzen
  - Ist eine XML-Sprache
  - Integriert die wichtigsten konkurrierenden Vorgängeransätze
  - Seit 2001-05-02 W3C Recommendation
  - Basis aller W3C-Standards der 2. Generation (XPath v2.0, XSLT v2.0, XHTML v2.0, SOAP 1.2, ...)
  - Werkzeugunterstützung verfügbar
  - Erster Schritt der Schema-Bestrebungen, weitere werden folgen ...

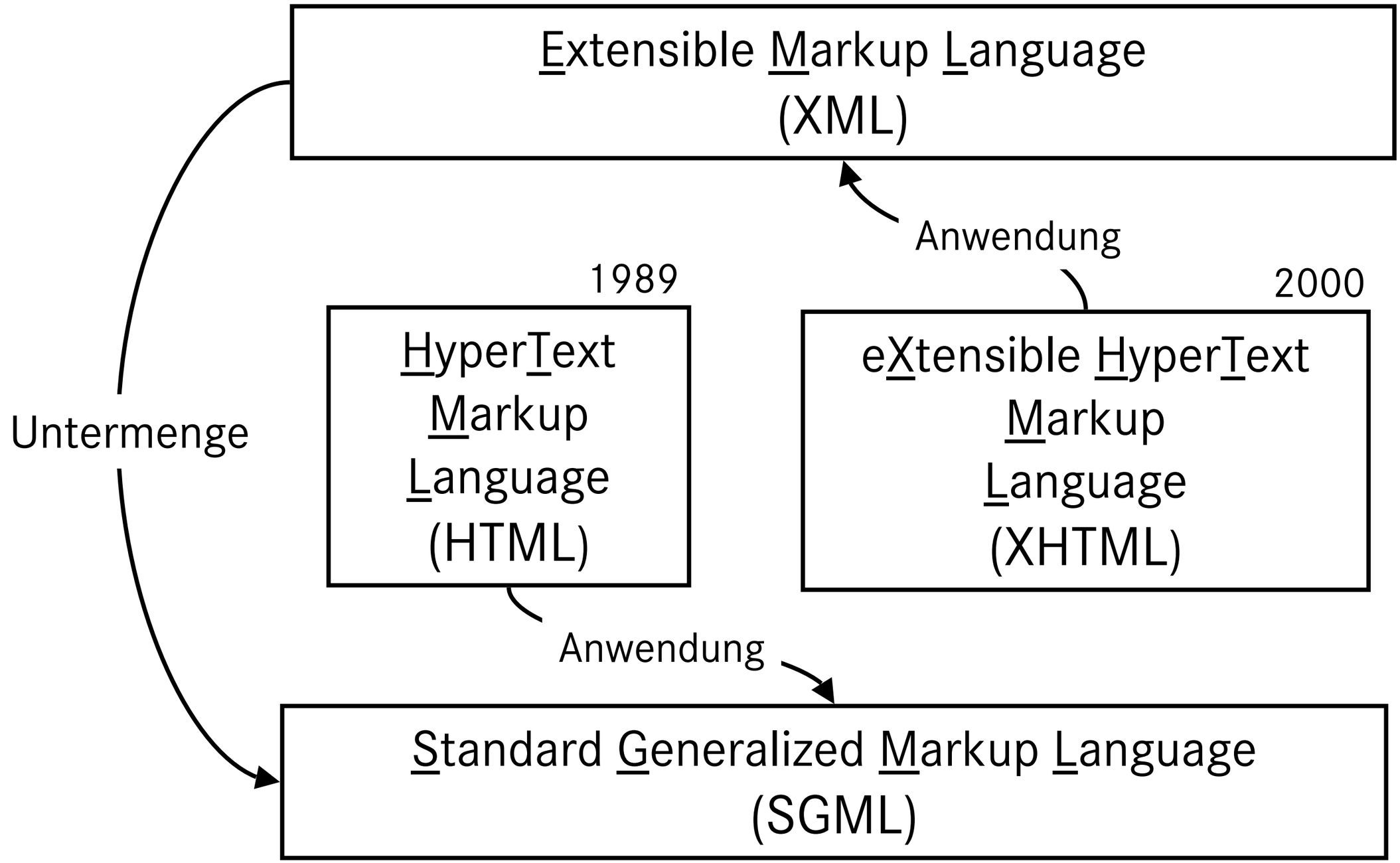
Gliederung

- Dokumente und Daten ...
  - Einführung und Überblick
  - Strukturelle Grundkonzepte
  - Namensräume
  - Dokument-Typ-Definitionen (DTD)
  - XML-Schema
  - **XML und das Web**

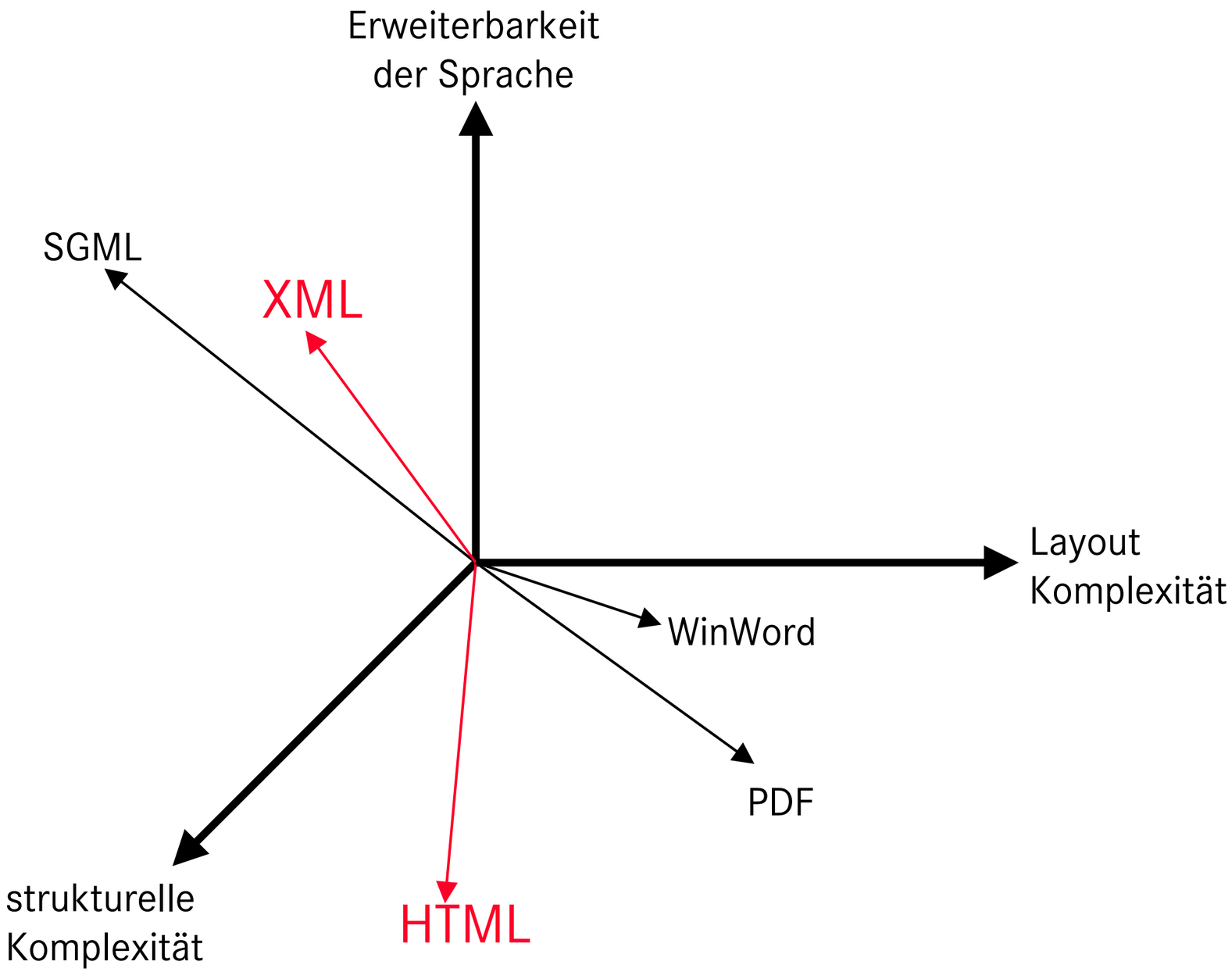
Dokumente und Daten ... XML und das Web

- HTML im Zeitalter von XML
  - von HTML zu XML
  - Aktuelle Entwicklungen
  - was geht heute schon?
- XML im Web-Browser?!
- HTML und eigene präsentationsorientierte Markupsprachen

Dokumente und Daten ... XML und das Web



Dokumente und Daten ... XML und das Web



Dokumente und Daten ... XML und das Web


```
<html>
<head><title>jeckle.de</title></head>
<body bgcolor="#003366" topmargin="0" leftmargin="0"  marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="0" cellpadding="0" height="65%" bgcolor="#f0f0f0">
  <tr>
    <td valign="bottom"> </td>
    <td colspan="2" valign="bottom" align="center">
      <table width="800" border="0" cellspacing="0" cellpadding="0" align="center" height="315">
        <tr bgcolor="#f0f0f0">
          <td height="25"> </td>        <td height="25"> </td>
          <td height="25"> </td>        <td height="25"> </td>
...
```

Tim Berners-Lee

Beginn Fettdruck (*bold*)

Ende Fettdruck

- HTML legt (hauptsächlich) das Präsentationsverhalten fest
- XML definiert die syntaktische Struktur der Information

Dokumente und Daten ... XML und das Web

- Keine überlappenden Elemente

```
<p>here is an emphasized <em>paragraph</em>.</p>
```

```
<p>here is an emphasized <em>paragraph.</p></em>
```

- Korrekte Terminierung

```
<p>here is a paragraph.</p>
<p>here is another paragraph.</p>
```

```
<p>here is a paragraph.
<p>here is another paragraph.
```

Dokumente und Daten ... XML und das Web

- Attributdarstellung

table rows=“3“

table rows=3

- Attributminimierung

<dl compact="compact“>

<dl compact>

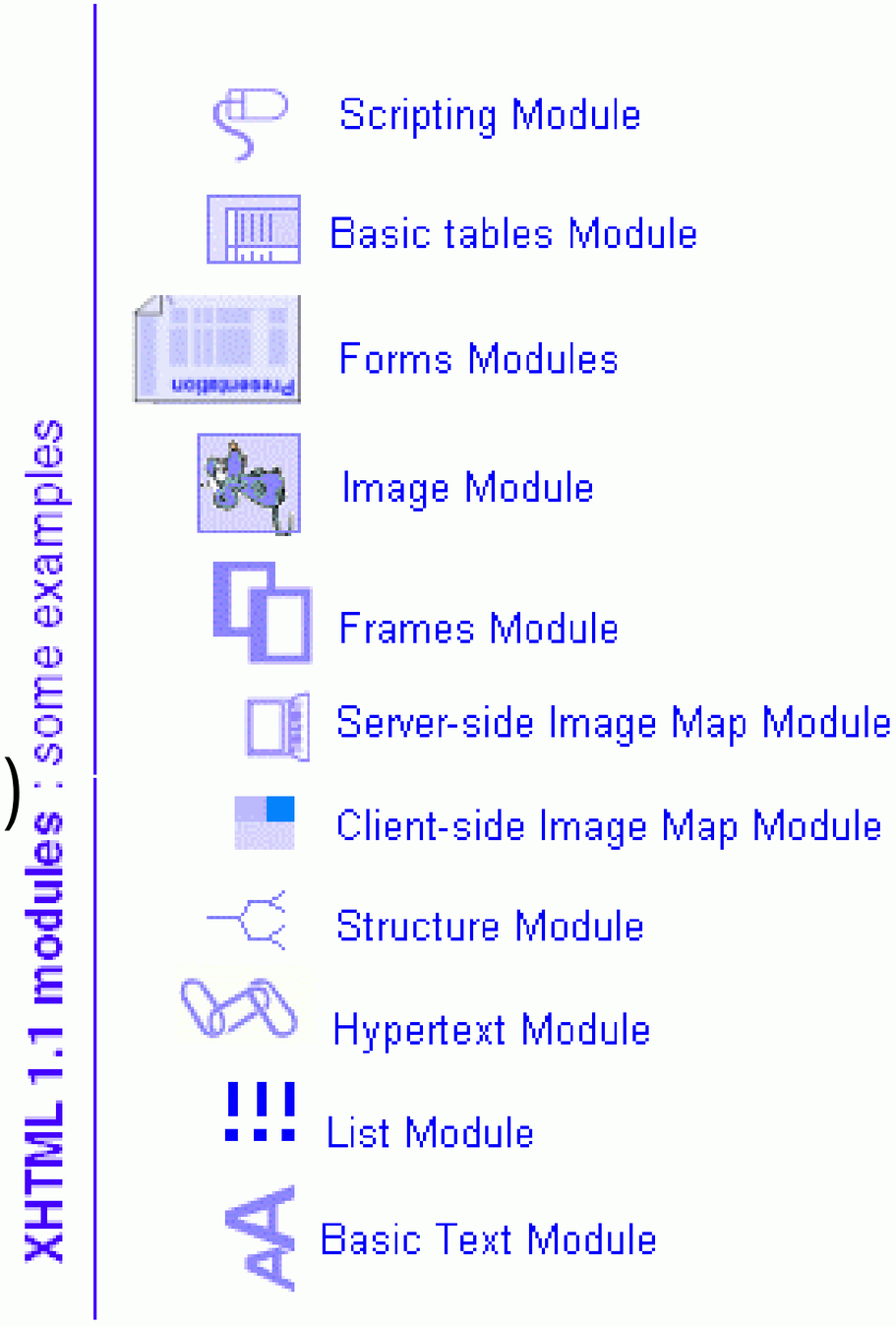
- Leere Elemente

<hr/>

<hr>

Dokumente und Daten ... XML und das Web

- Modulares XHTML ist die funktionale Dekomposition von XHTML.
- Separierung der in XHTML adressierten verschiedenen Problembereiche, um diese
  - getrennt einsetzen zu können (WAP!)
    erweitern zu können, ohne XHTML-Standard zu verletzen



Dokumente und Daten ... XML und das Web

C:\mcjdt\www\jeckle.de\life\vorlesung\xml\examples\projektverwaltung2.xml - Microsoft Internet Explorer - [Offlinebetrieb]

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück

Adresse http://www.jeckle.de/vorlesung/xml/examples/projektverwaltung2.xml Wechseln zu

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <ProjektVerwaltung xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.jeckle.de/vorlesung/xml/examples/projektverwaltung.xsd">
  - <Person PersID="Pers01" mitarbeitInProjekt="Prj01">
      <Vorname>Hans</Vorname>
      <Nachname>Hinterhuber</Nachname>
    </Person>
  - <Person PersID="Pers02" mitarbeitInProjekt="Prj02">
      <Vorname>Franz</Vorname>
      <Vorname>Xaver</Vorname>
      <Nachname>Obermüller</Nachname>
    - <Qualifikationsprofil>
        IT-Kompetenz verschiedene Betriebssysteme und
        <Leistungsstufe>professionelle</Leistungsstufe>
        <Qualifikation>Programmierung</Qualifikation>
        verschiedener Programmiersprachen
        <Qualifikation>Projektleiterfunktion</Qualifikation>
        von 1990-93 im X42-Projekt in Abteilung AB&C
      </Qualifikationsprofil>
    </Person>
- <Person PersID="Pers03" mitarbeitInProjekt="Prj02">
    <Vorname>Fritz</Vorname>
    <Nachname>Meier</Nachname>
  </Person>
  <Projekt ID="Prj01" Projektleiter="Pers01" Mitarbeiter="Pers01" />
  <Projekt ID="Prj02" Projektleiter="Pers02" Mitarbeiter="Pers03" />
</ProjektVerwaltung>
```

Fertig Arbeitsplatz

Dokumente und Daten ... XML und das Web

- Einbettung von XHTML-Modulen in eigene XML-Vokabulare
  - Namensräume!
  - XML-Schema!

XML-Dokument:

```
<Qualifikationsprofil>
  <u>IT-Kompetenz</u><em>verschiedene</em> Betriebssysteme und
  <Leistungsstufe>professionelle</Leistungsstufe>
  <em><Qualifikation>Programmierung</Qualifikation></em>
  verschiedener Programmiersprachen
  <em><u><Qualifikation>Entwickler</Qualifikation></u></em> von 1988-1990
  <u><Qualifikation>Projektleiterfunktion</Qualifikation></u>
  von <b>1990-93</b> im X42-Projekt in Abteilung AB&amp;amp;C
</Qualifikationsprofil>
```

Dokumente und Daten ... XML und das Web

- Einbettung von XHTML-Modulen in eigene XML-Vokabulare
  - Namensräume!
  - XML-Schema!

XML-Schema:

```
<xsd:complexType name="QualifikationsprofilType" mixed="true">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="Qualifikation"/>
    <xsd:element ref="Leistungsstufe"/>
    <xsd:any namespace="http://www.w3.org/1999/xhtml"
        minOccurs="0" maxOccurs="unbounded"/>
  </xsd:choice>
</xsd:complexType>
```

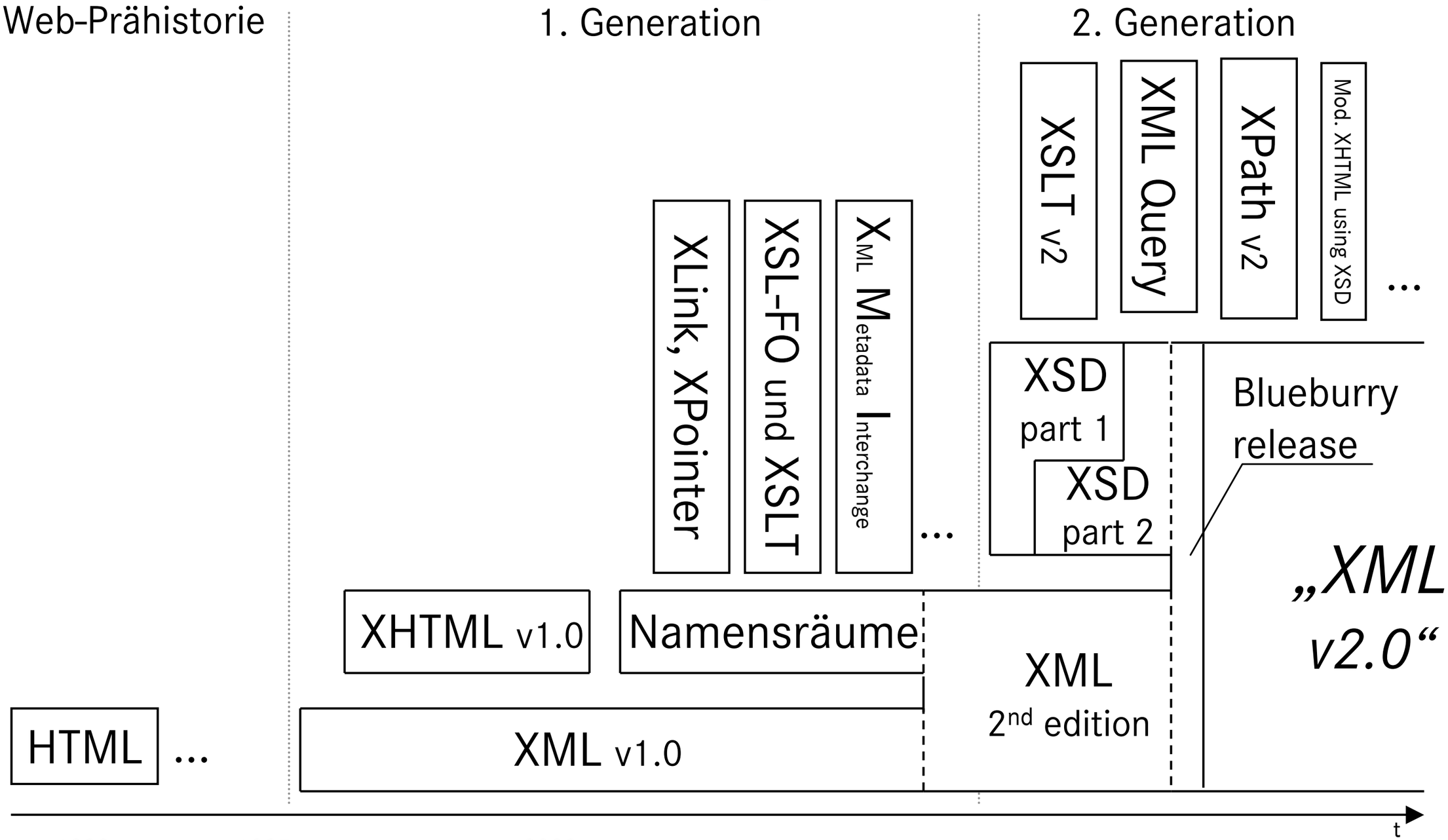
Dokumente und Daten ... XML und das Web

- Zusammenfassung: XML und das Web
  - HTML hat sich von einer Anwendung der SGML, zu XHTML, einer Anwendung von XML, weiterentwickelt
  - Reformulierung XHTML v1.0 erschließt HTML v4.01 die XML-Sprach- und Werkzeugwelt
  - Durch Namensräume und XML-Schema kann HTML-Semantik in eigene XML-Sprachen übernommen werden
  - Web-Browser-Darstellung von nativem XML ist möglich und wird verschiedentlich unterstützt. Die Hauptintention liegt jedoch nicht in der Präsentation; hierfür existieren Transformationssprachen (->XSL(T))

Gliederung

- **XML-Standards und -Anwendungen der zweiten Generation ...**
  - (Dokument-)Verknüpfungen: XML Links
  - Die Lokatorsprache XPath
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - Transformation von XML-Dokumenten: XSL Transformations
  - Die Anfragesprache XQuery

XML-Standards und -Anwendungen der zweiten Generation

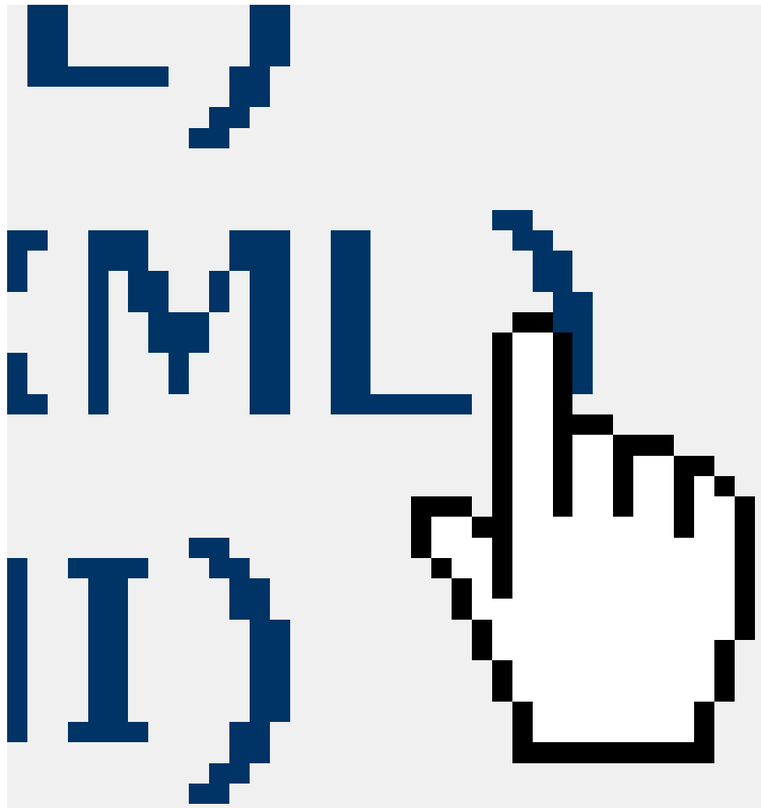


Gliederung

- XML-Standards und -Anwendungen der zweiten Generation ...
- ➡ **(Dokument-)Verknüpfungen: XML Links**
  - Die Lokatorsprache XPath
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - Transformation von XML-Dokumenten: XSL Transformations
  - Metadatenaustausch und Schemaerzeugung: XMI
  - Die Anfragesprache XQuery

(Dokument-)Verknüpfungen: XML Links

- HTML-Hyperlinks sind allgegenwärtig
- 404-en auch :-(
- Links sind nur unidirektional
- Verweisen nur auf ein einziges Ziel
- Sind Dokument-intern abgelegt



(Dokument-)Verknüpfungen: XML Links

- Verallgemeinerung des HTML-Hyperlinkmechanismus auf beliebige XML-Dokumente
- Erweiterung der klassischen Möglichkeiten
- XLink definiert hierzu eine Menge von XML-Attributen, die in beliebige eigene Sprachen eingebaut werden können
- XLink-Prozessoren *verstehen* nur diese Attribute und lassen andere Sprachbestandteile unverarbeitet

(Dokument-)Verknüpfungen: XML Links

- *Simpler*-XLink rekonstruiert Mächtigkeit der HTML href

```
<myElementA xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="http://www.jeckle.de">
  ...
</myElementA>
```

- Vorgehen:
  - Definition des XLink-Namensraums
  - Verwendung der XLink-Attribute in beliebigen eigenen XML-Vokabularen

(Dokument-)Verknüpfungen: XML Links

- Bereits HTML bietet zwei Hyperlink-Varianten:
  - Offensichtlich: `href`
  - Transklutorische Links: Das `img`-Element
    `img` verweist auf beliebige, durch URI bezeichnete, Ressourcen.
    Allerdings sind als referenzierte Datentypen nur Bildformate zugelassen
- Verallgemeinerung in XLink für beliebige Ressourcen:

```
<myElementB
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:href="http://www.w3.org/Icons/w3c_home.gif"
  xlink:actuate="onLoad"
  xlink:show="embed">
  ...</myElementB>
```

Download Java(TM) 2 SDK, Standard Edition, Beta 2 1.4.0 - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: http://java.sun.com/Download5 What's Related

Case Studies

FTP download

If you are behind a firewall and cannot access our FTP servers, please try the HTTP download below.

Alternate FTP mirror sites:

FTP download from US West I

FTP download from US West II

FTP download from US West III

FTP download from US West IV

FTP download from US Central I

FTP download from US Central II

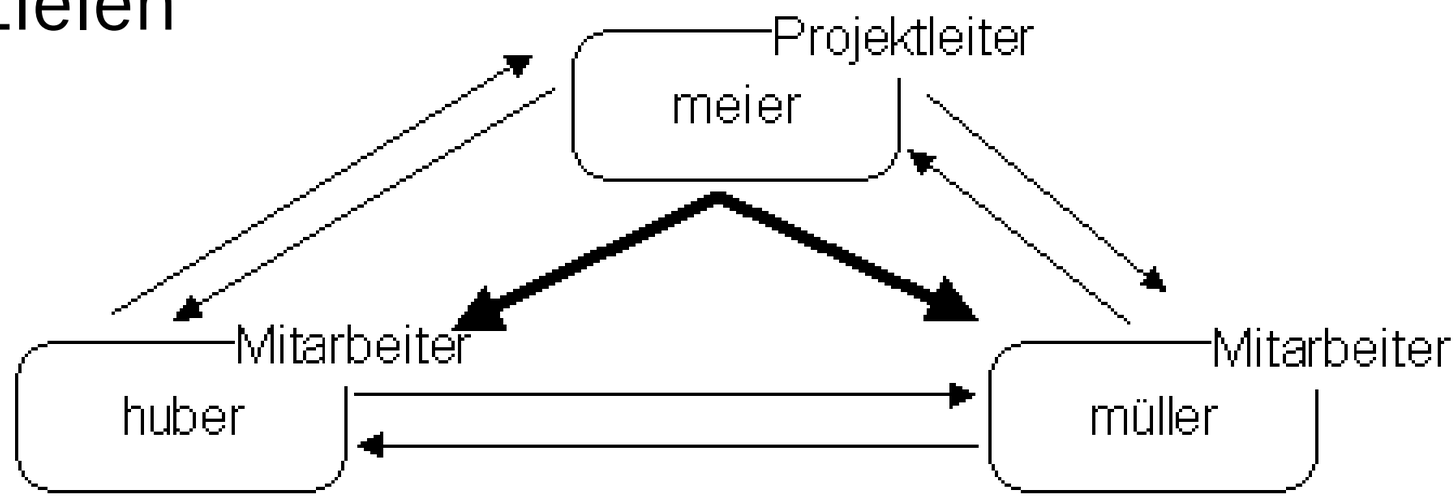
FTP download from US East I

FTP download from US East II

Document: Done

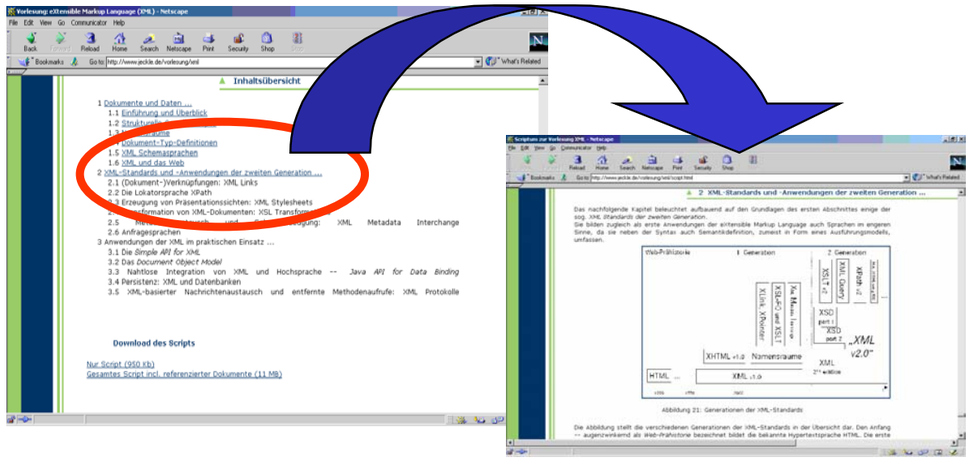
(Dokument-)Verknüpfungen: XML Links

- XLinks zu mehreren Zielen

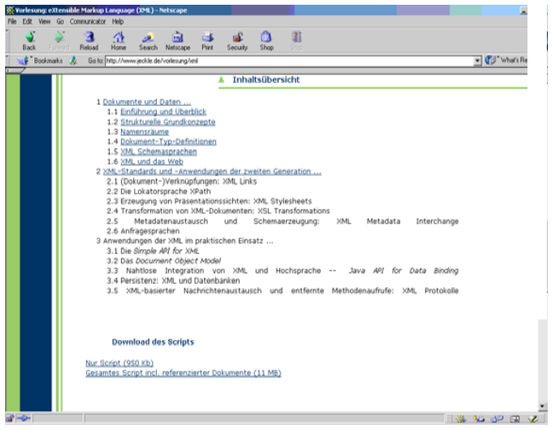


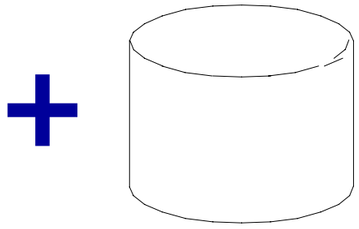

```
<Projekt xmlns:xlink="http://ww
xlink:type="extended">
  <Person xlink:type="locator"
  xlink:href="urn:meier"
  label="ID001"
  xlink:title="Projektleiter"
  xlink:role="http://www.example.com/pl" />
  <Person xlink:type="locator"
  xlink:href="urn:huber"
  label="ID002"
  xlink:title="Mitarbeiter"
  xlink:role="http://www.example.com/emp" />
  <Person xlink:type="locator"
    xlink:href="urn:müller"
    label="ID003"
    xlink:title="Mitarbeiter"
    xlink:role="http://www.example.com/emp" />
  <leader xlink:type="arc" xlink:from="ID001"
    xlink:to="ID002"/>
  <leader xlink:type="arc" xlink:from="ID001"
    xlink:to="ID003"/> </Projekt>
```

(Dokument-)Verknüpfungen: XML Links



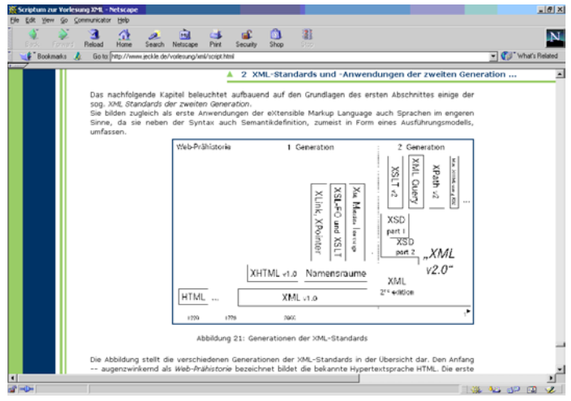
Interner Link:





Externe Linkdatenbank

Externer Link:



(Dokument-)Verknüpfungen: XML Links

- Zusammenfassung: *XML Links (XLink)*
  - Führen die Möglichkeit einfacher und komplexer Verweise in die XML-Sprachwelt ein
  - Basiskonzept zur Adressierung und Identifikation ist die URI
  - Orientieren sich konzeptionell an erprobten Lösungen wie: HTML, HyTime, TEI, Dexter, FRESS, OHS, Microcosm, Intermedia
  - Verweise können in XML-Dokumente eingebettet werden, oder in einer separierten *Link Datenbank* versammelt werden

Gliederung

- XML-Standards und -Anwendungen der zweiten Generation ...
  - (Dokument-)Verknüpfungen: XML Links
  - ➡ **Die Lokatorsprache XPath**
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - Transformation von XML-Dokumenten: XSL Transformations
  - Metadatenaustausch und Schemaerzeugung: XMI
  - Die Anfragesprache XQuery

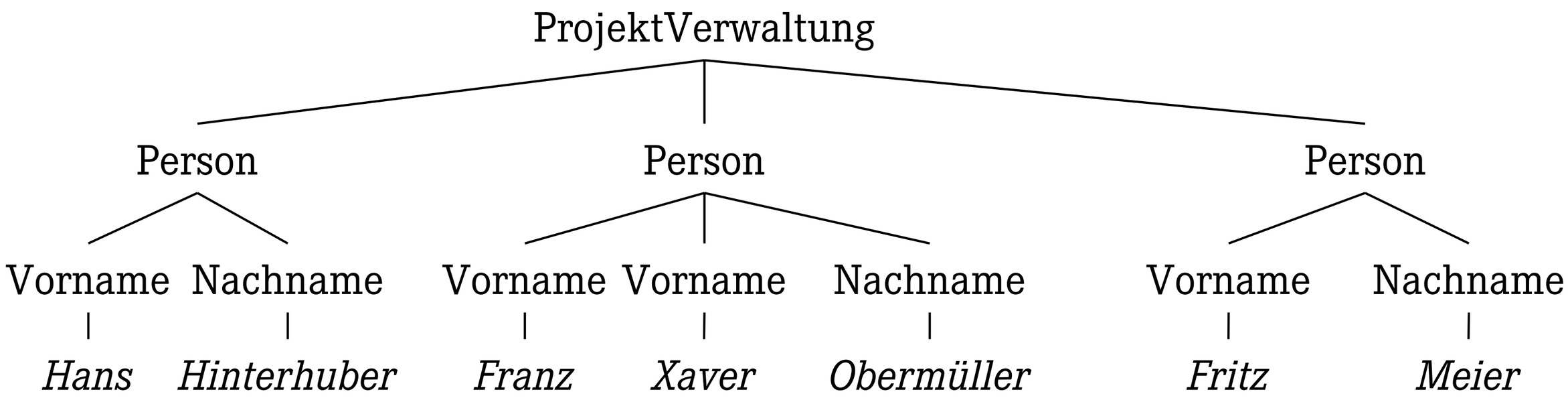
Die Lokatorsprache XPath

- Eigentlich „nur“ Schnittmenge zwischen XSLT und XPointer
- Technisch: Lokatorsprache zur Adressierung beliebiger Knotenmengen eines XML-Dokuments
- Grundlegende Möglichkeiten zur Manipulation von Zahlen, Zeichenketten und Boole‘schen Werten
- Keine XML-Sprache, wohl aber W3C-Recommendation und XML flankierender Standard
- Anfragen werden immer, sofern nicht explizit anders angegeben, relativ zum Kontextknoten ausgewertet

Die Lokatorsprache XPath

- Lokalisierungspfade
  - Hierarchische Struktur wie aus dem InfoSet bekannt
  - Top-down Navigation entlang benannter (typisierter) Knoten

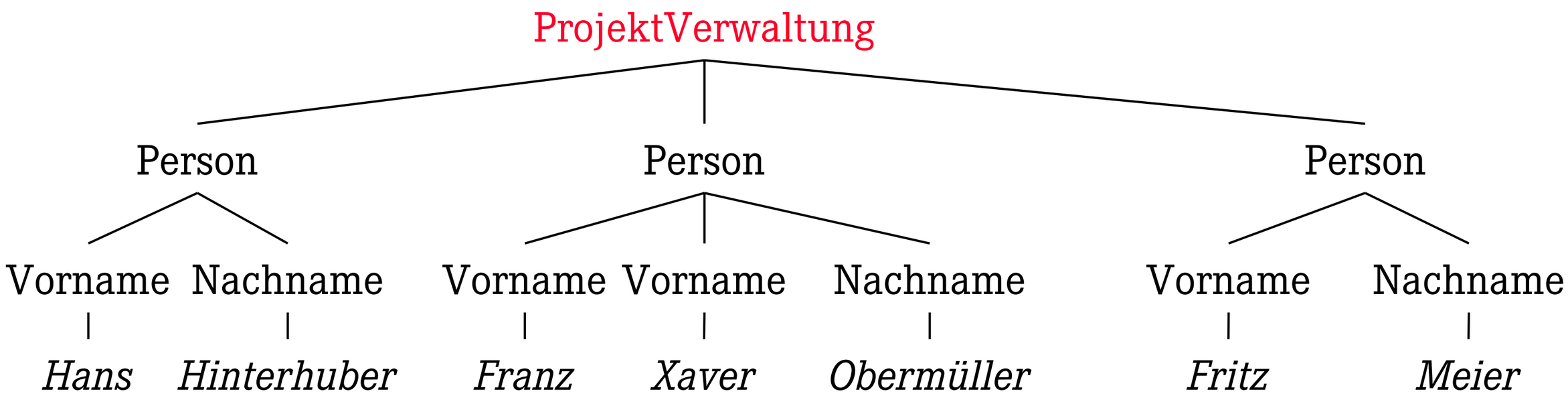
/ProjektVerwaltung/Person/Vorname



Die Lokatorsprache XPath

- Lokalisierungspfade
  - Hierarchische Struktur wie aus dem InfoSet bekannt
  - Top-down Navigation entlang benannter (typisierter) Knoten

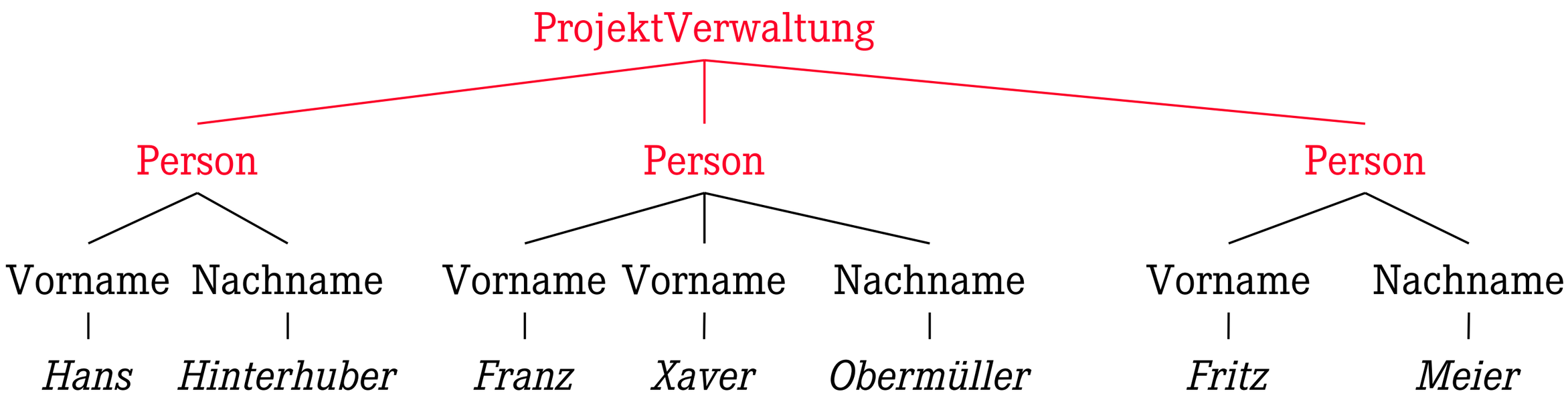
/ProjektVerwaltung/Person/Vorname



Die Lokatorsprache XPath

- Lokalisierungspfade
  - Hierarchische Struktur wie aus dem InfoSet bekannt
  - Top-down Navigation entlang benannter (typisierter) Knoten

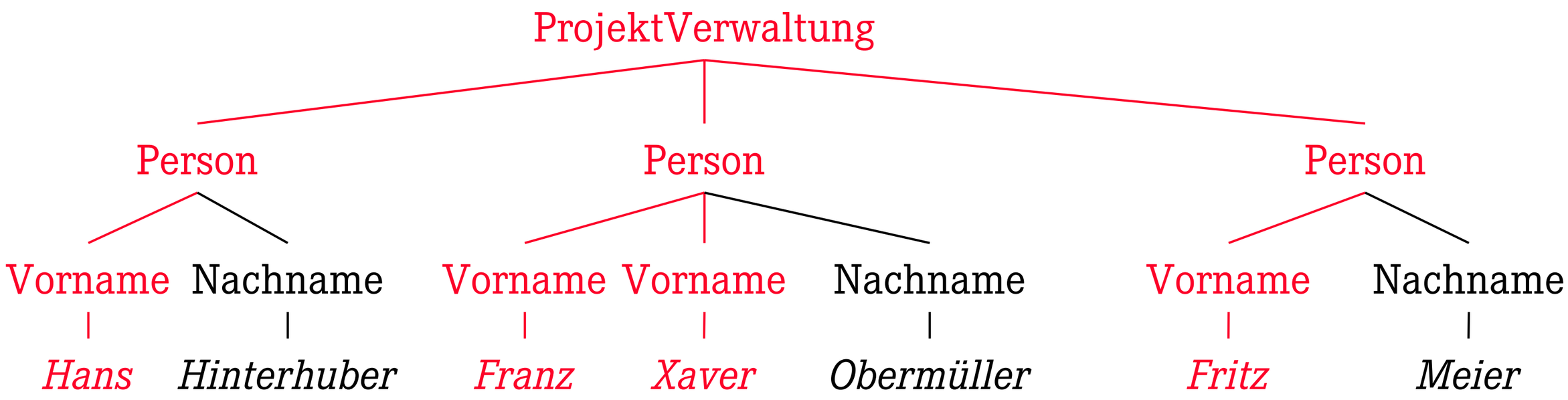
/ProjektVerwaltung/Person/Vorname



Die Lokatorsprache XPath

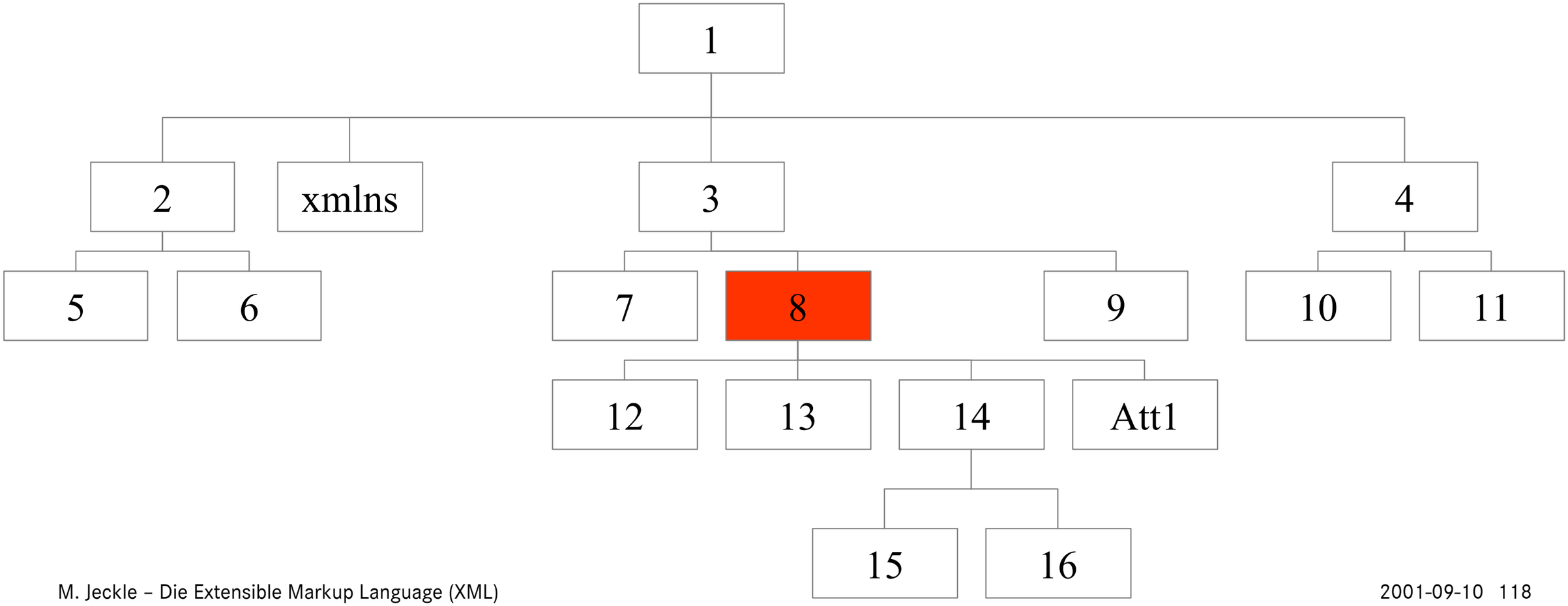
- Lokalisierungspfade
  - Hierarchische Struktur wie aus dem InfoSet bekannt
  - Top-down Navigation entlang benannter (typisierter) Knoten

/ProjektVerwaltung/Person/Vorname



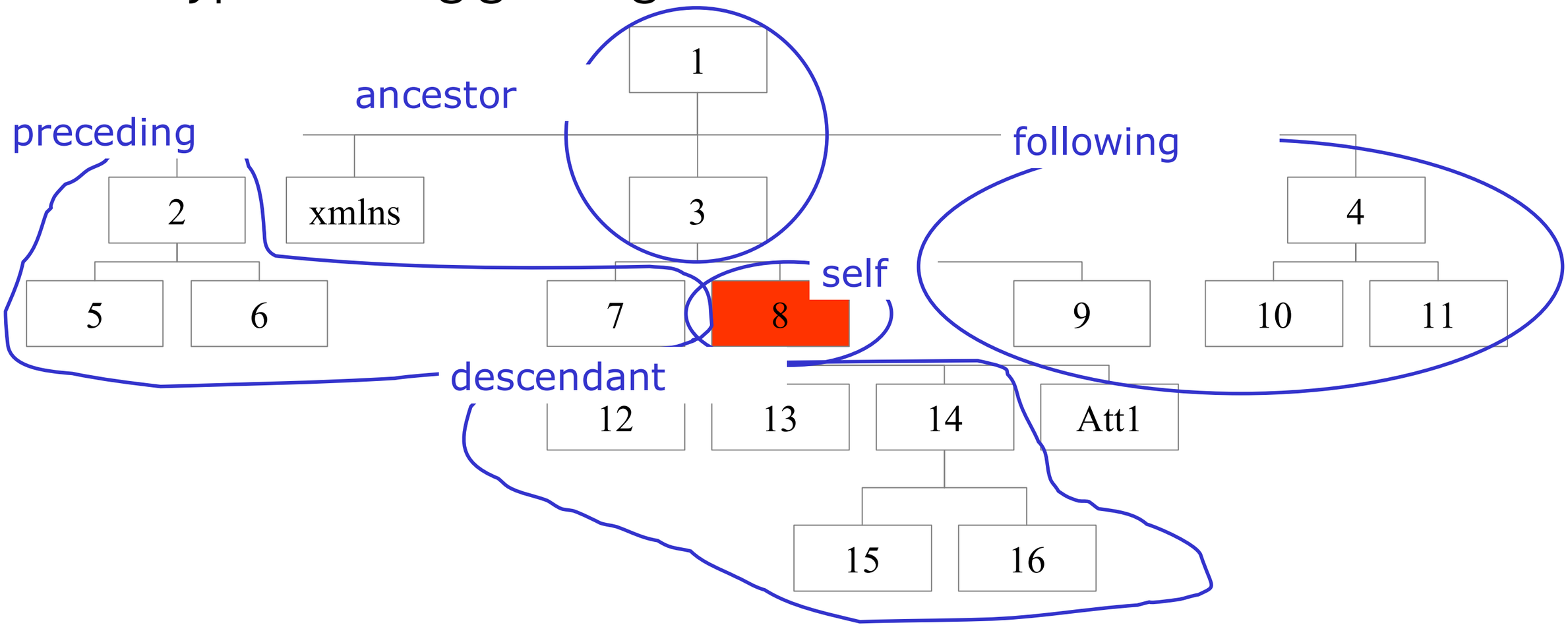
Die Lokatorsprache XPath

- Achsen
  - Ziel: Durchbrechen der rein hierarchisch-absteigenden Traversierung und Einbeziehung *benachbarter* Knoten



Die Lokatorsprache XPath

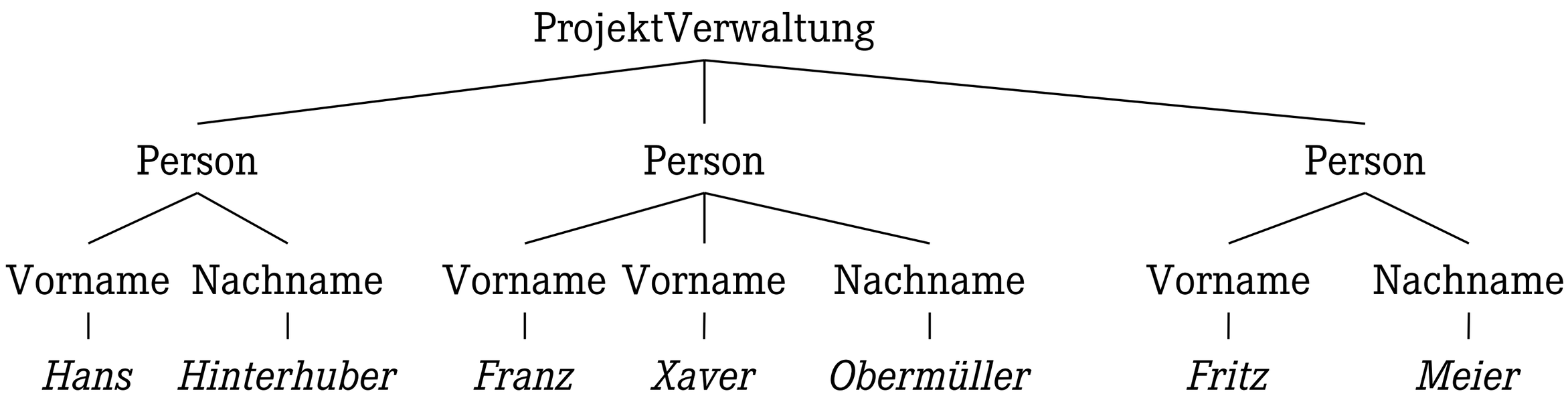
- Achsen
  - Insgesamt existieren 13 verschiedene Achsen zum typunabhängigen Zugriff auf Knoten verschiedenster Position



Die Lokatorsprache XPath

- XPath als Anfragesprache
  - Nutzung der Prädikate

//Person[count(Vorname) > 2]/Nachname

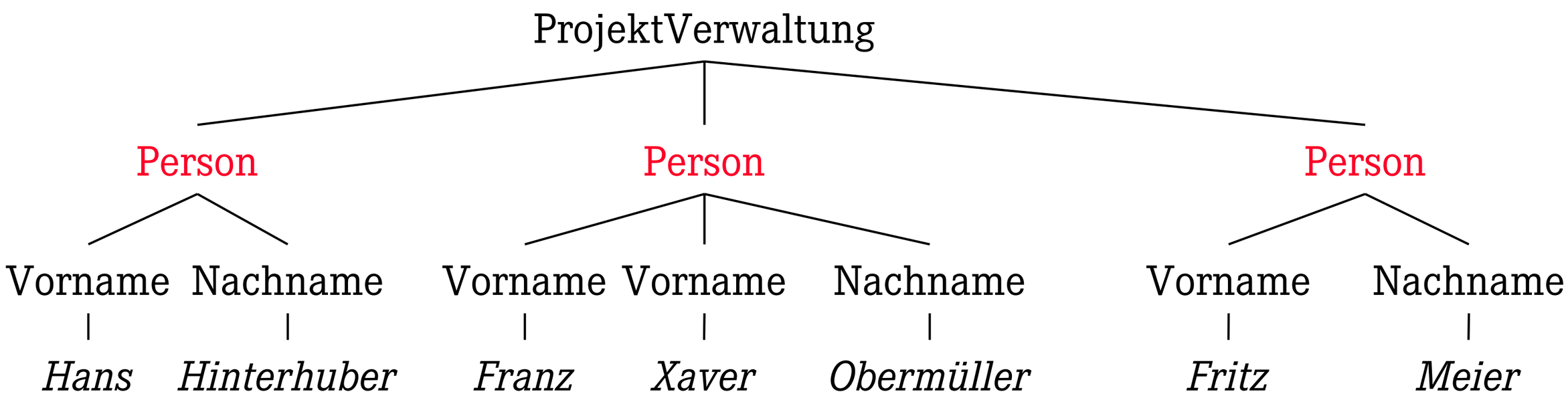


Die Lokatorsprache XPath

- XPath als Anfragesprache
  - Nutzung der Prädikate

//Person[count(Vorname) = 2]/Nachname

- Auswahl aller Personen an beliebiger Hierachieposition

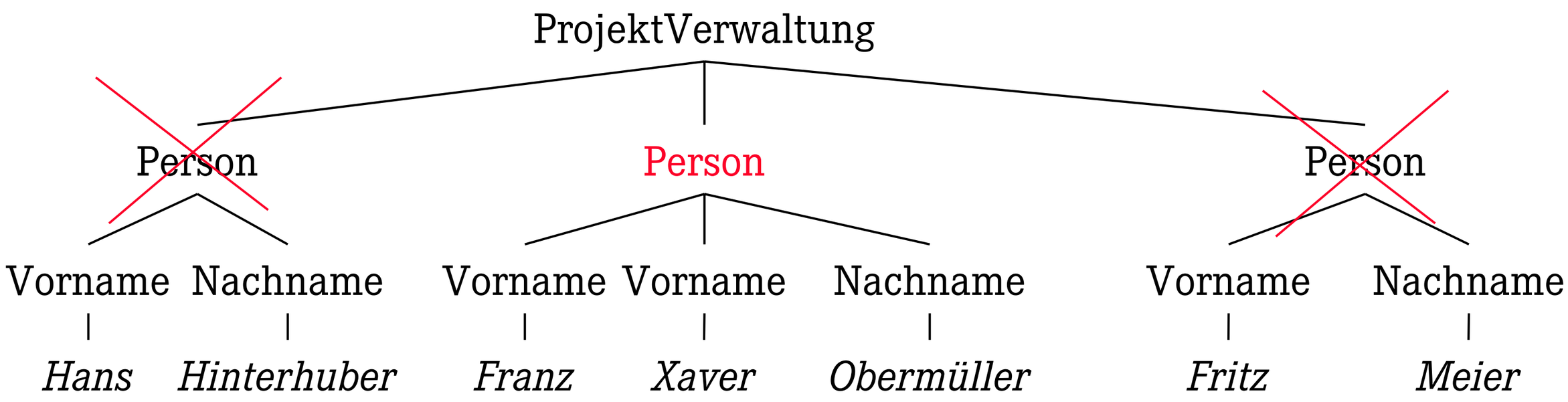


Die Lokatorsprache XPath

- XPath als Anfragesprache
  - Nutzung der Prädikate

//Person[count(Vorname) = 2]/Nachname

  - Verminderung der Ergebnismenge

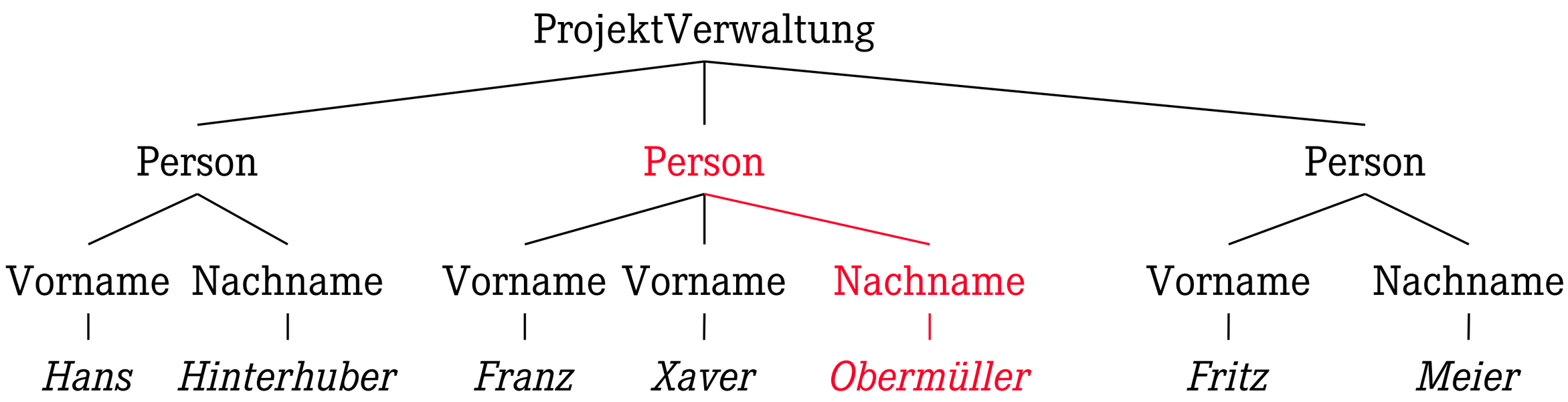


Die Lokatorsprache XPath

- XPath als Anfragesprache
  - Nutzung der Prädikate

//Person[count(Vorname) = 2]/Nachname

  - (Weiter-)Selektion



Die Lokatorsprache XPath

- Zusammenfassung: *XPath*
  - Mächtige Lokatorsprache mit navigierendem Zugriff
  - Erlaubt die Auswahl beliebiger Knoten(-mengen)
  - Syntax ist String-basiert, nicht XML;
    findet jedoch in den XML-Sprachen XPointer
    und XSLT Verwendung
  - (Teilweise mit etwas Mühe) als Anfragesprache geeignet.
    => XQuery stellt hier eine (bessere) Alternative dar, die
    eine SQL-artige (algebraische) Anfrage auf beliebige XML-
    Dokumente definieren wird

Gliederung

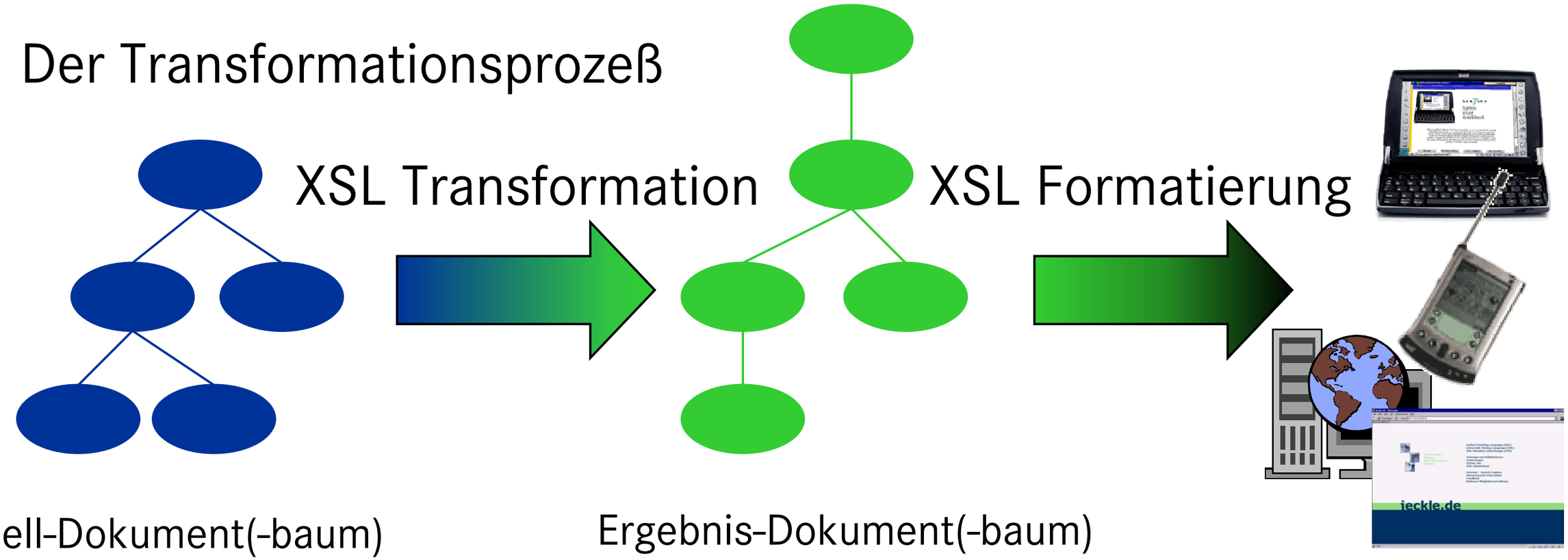
- XML-Standards und -Anwendungen der zweiten Generation ...
  - (Dokument-)Verknüpfungen: XML Links
  - Die Lokatorsprache XPath
  - **Erzeugung von Präsentationssichten: XML Stylesheets**
  - Transformation von XML-Dokumenten: XSL Transformations
  - Metadatenaustausch und Schemaerzeugung: XMI
  - Die Anfragesprache XQuery

Erzeugung von Präsentationssichten: XML Stylesheets

- Aufgabe: Erzeugung von (beliebigen) Präsentationsdarstellungen aus XML-Dokumenten
- Definiert Vokabular zur Spezifikation von Formatierungs-semantik und Transformationssprache (XSLT) zur Umwandlung von XML-Dokumenten (siehe nächstes Kapitel)
- XSL-Formatierunganweisungen sind Medien-neutral und können für verschiedene Endgeräte unterschiedlich umgesetzt werden
- XSL erweitert die von HTML bekannten *Cascading Style Sheets*

Erzeugung von Präsentationssichten: XML Stylesheets

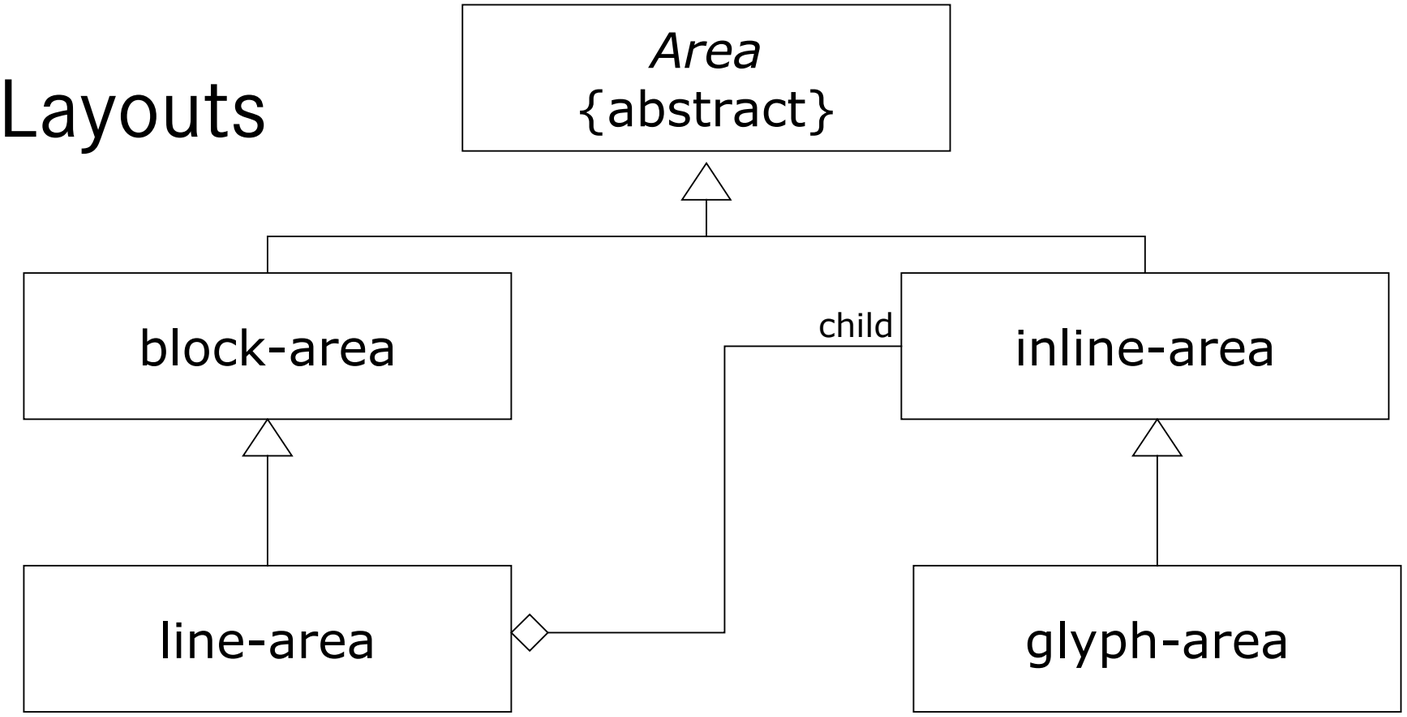
- Der Transformationsprozeß



- XSLT-Prozessor formt XML-strukturierte Eingabe um; Ergebnis: XML-Dokument mit Präsentationsinformation in XML (XSL:FO)
- XSL:FO-Prozessor erzeugt beliebiges (nicht XML-)Ergebnisformat

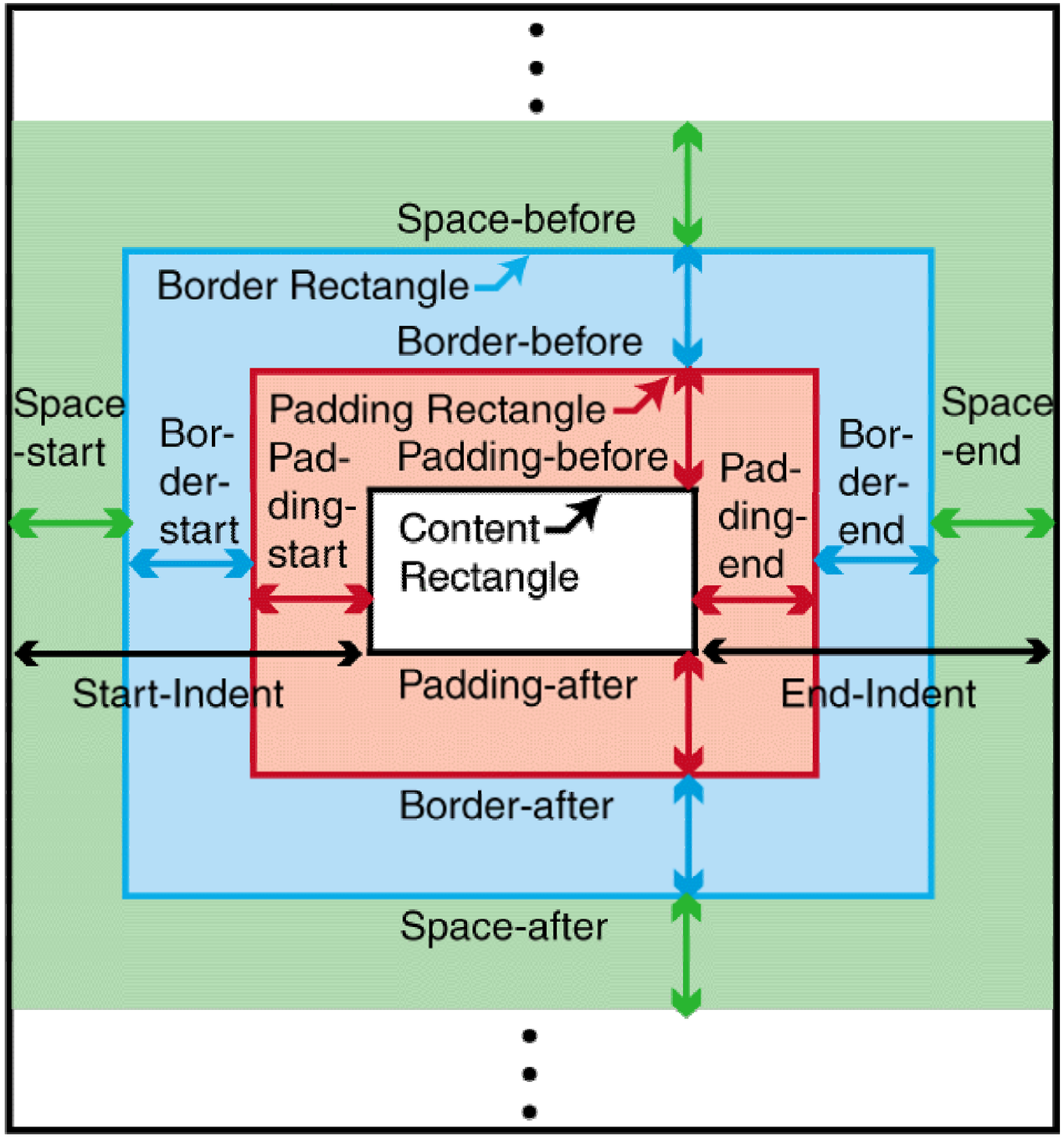
Erzeugung von Präsentationssichten: XML Stylesheets

- Die Grundbausteine des Layouts
  - **block-area:** Allgemeinstes Element Rechtwinkliger Layout-Bereich zur Aufnahme von Absätzen, Überschriften etc.

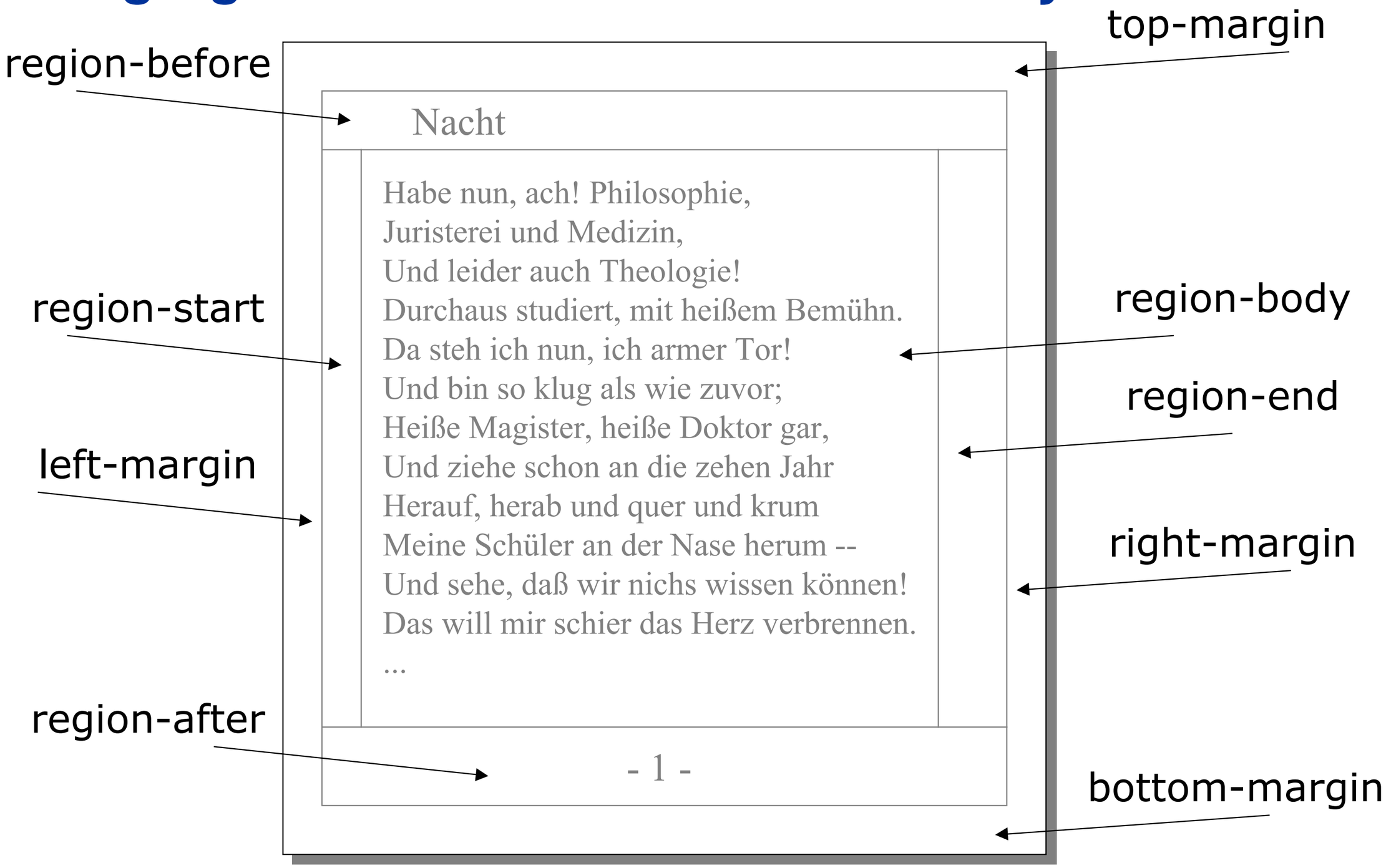


  - **inline-area:** Graphische Hervorhebung einzelner Textbereiche
  - **line-area:** Textbereiche ohne umgebenden Rand
  - **glyph-area:** Einzelner Buchstabe

Erzeugung von Präsentationssichten: XML Stylesheets



Erzeugung von Präsentationssichten: XML Stylesheets

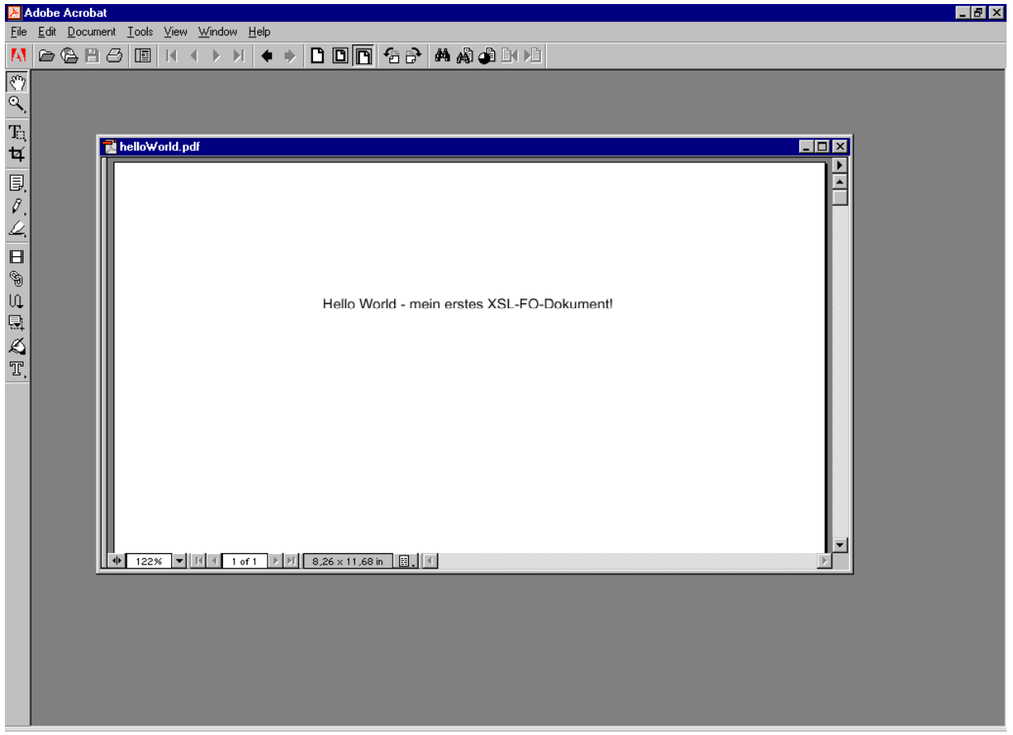


Erzeugung von Präsentationssichten: XML Stylesheets

- Durchgängiges Beispielszenario

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple"
        page-height="29.7cm"
        page-width="21cm"
        margin-top="1cm"
        margin-bottom="2cm"
        margin-left="2.5cm"
        margin-right="2.5cm">
      <fo:region-body margin-top="3cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
...
</fo:root>
```

XSL:FO-Prozessor



Quelldokument (XML-FO)

Zieldokument (PDF)

Erzeugung von Präsentationssichten: XML Stylesheets

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple"
        page-height="29.7cm"
        page-width="21cm"
        margin-top="1cm"
        margin-bottom="2cm"
        margin-left="2.5cm"
        margin-right="2.5cm">
      <fo:region-body margin-top="3cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
</fo:root>
```

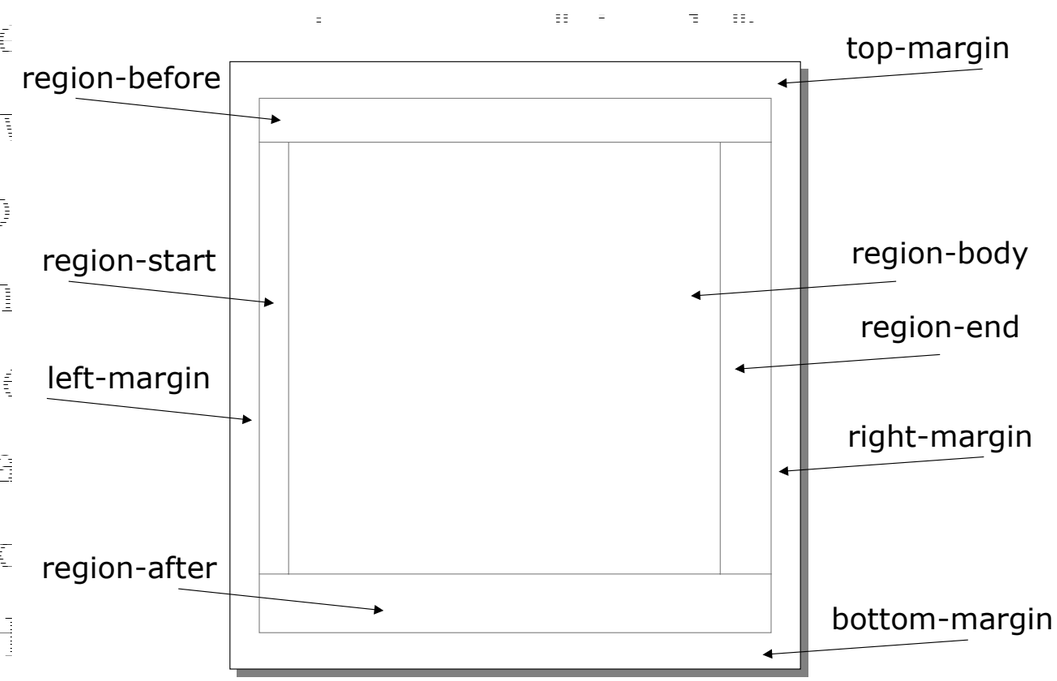
```
<fo:page-sequence master-name="simple">
    <fo:flow flow-name="xsl-region-body">
      <fo:block font-size="12pt"
        font-family="sans-serif"
        line-height="15pt"
        space-after="3pt"
        text-align="center">
          Hello World - mein erstes XSL-FO-Dokument!
      </fo:block>
    </fo:flow>
</fo:page-sequence>
```

Erzeugung von Präsentationssichten: XML Stylesheets

Global-gültige Seiteneinstellungen

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://ww
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple"
        page-height="29.7cm"
        page-width="21cm"
        margin-top="1cm"
        margin-bottom="2cm"
        margin-left="2.5cm"
        margin-right="2.5cm">
      <fo:region-body margin-top="3cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
</fo:root>
```

```
<fo:page-se
  <fo:flo
    <fo:b
      fon
      line
      spa
      tex
      H
    </fo:block>
  </fo:flow>
</fo:page-sequence>
```



Erzeugung von Präsentationssichten: XML Stylesheets

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple"
        page-height="29.7cm"                                  aster-name="simple">
        page-width="                                          le="xsl-region-body">
        margin-top="1cm"
        margin-bottom="2cm"
        margin-left="2.5cm"
        margin-right="2.5cm">
      <fo:region-body margin-top="3cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
</fo:root>
```

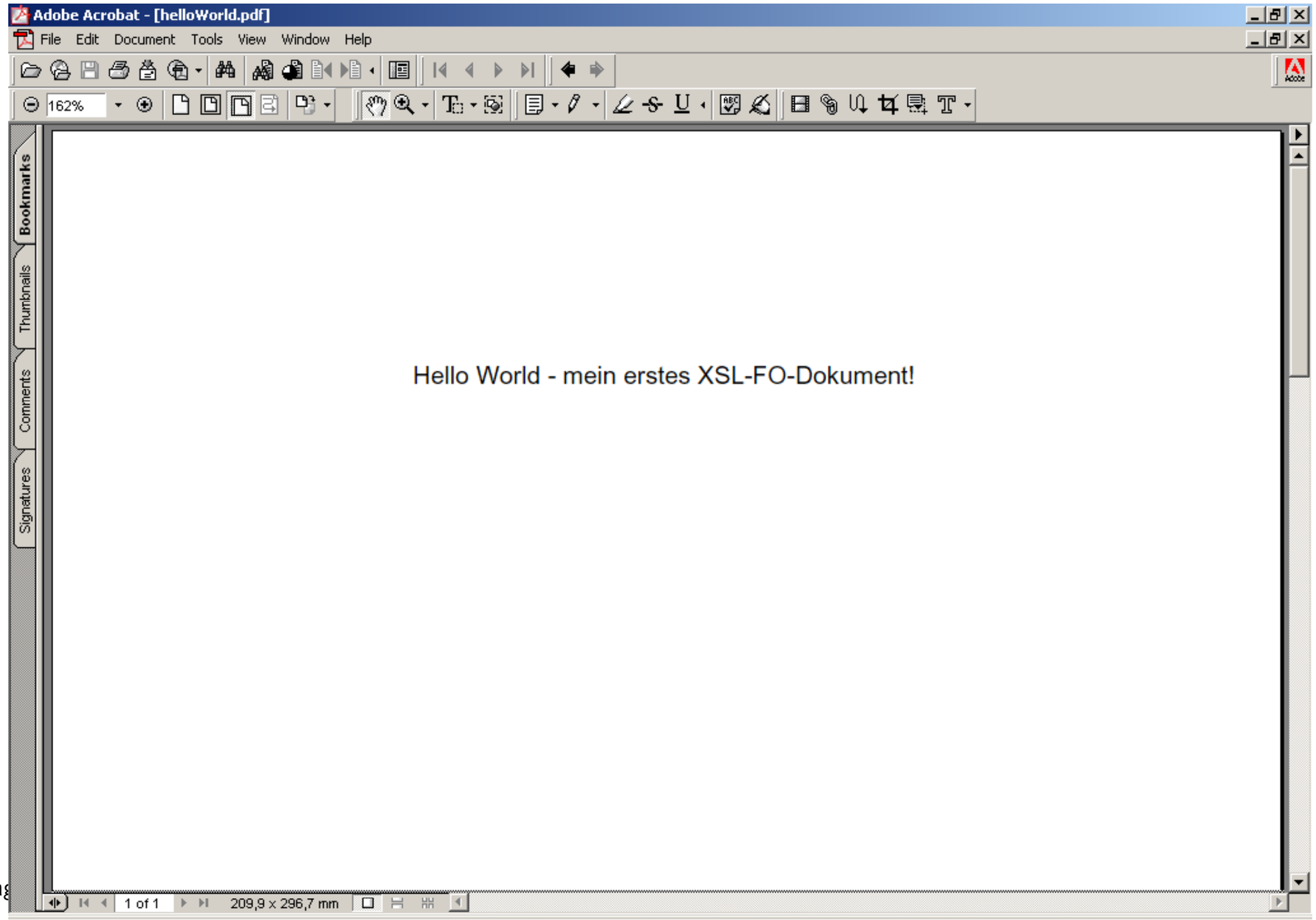
Inhaltsbereich (*block region*)

```
<fo:block font-size="12pt"
  font-family="sans-serif"
  line-height="15pt"
  space-after="3pt"
  text-align="center">
    Hello World - mein erstes XSL-FO-Dokument!
</fo:block>
```

```
  </fo:flow>
</fo:page-sequence>
```

Erzeugung von Präsentationssichten: XML Stylesheets

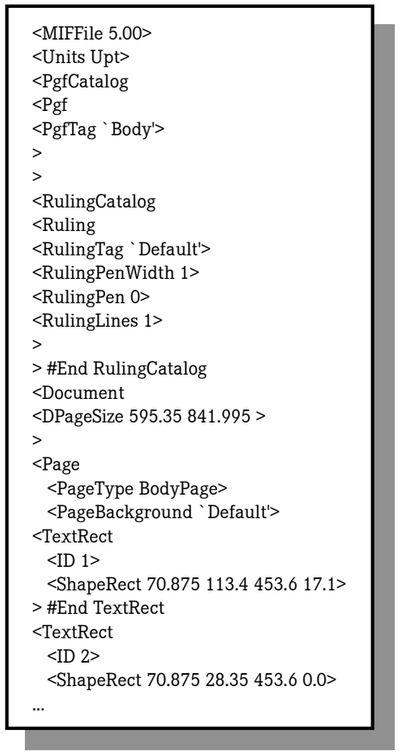
- Das Ergebnis



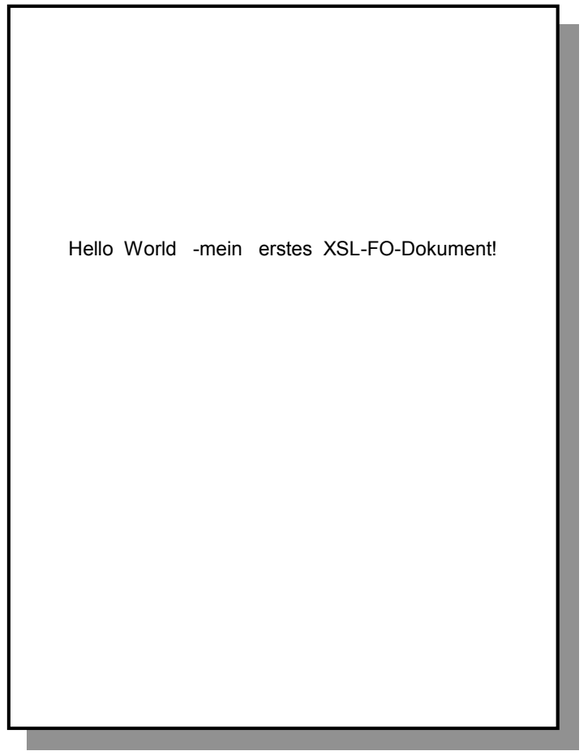
Erzeugung von Präsentationssichten: XML Stylesheets

- weitere mögliche Ergebnisse

PCL-Seitenbeschreibungssprache



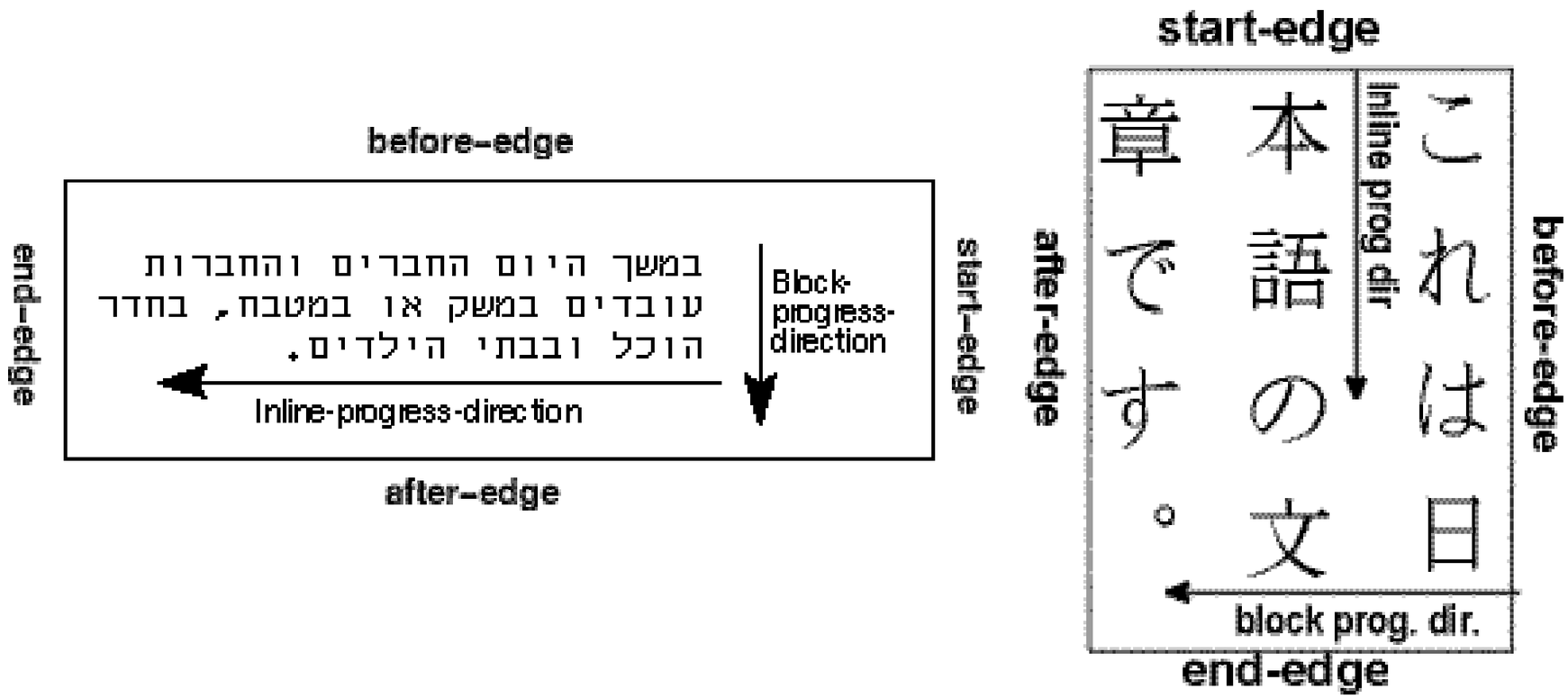
MIF-Format



ASCII

Erzeugung von Präsentationssichten: XML Stylesheets

- Internationalisierung von Dokumenten



Erzeugung von Präsentationssichten: XML Stylesheets

- Zusammenfassung: XSL:FO
  - Standardisiertes mächtiges Vokabular zur Beschreibung verschiedenster Layouts
  - Transformation (mittels XSL:FO-Prozessor) in (nahezu) beliebiges Zielformat
  - Erste Prozessoren und Rendering Engines verfügbar
  - XSL:FO-Vokabular *unhandlich*, daher in der Praxis besser durch automatisierte Anreichung bestehender XML-Dokumente handhabbar

Gliederung

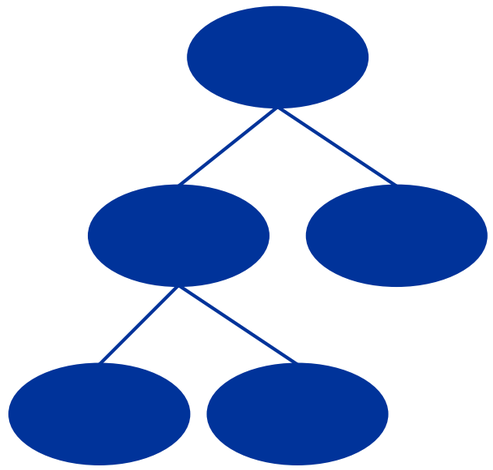
- XML-Standards und -Anwendungen der zweiten Generation ...
  - (Dokument-)Verknüpfungen: XML Links
  - Die Lokatorsprache XPath
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - ➡ **Transformation von XML-Dokumenten: XSLT**
  - Metadatenaustausch und Schemaerzeugung: XMI
  - Die Anfragesprache XQuery

Transformation von XML-Dokumenten: XSL Transformations

- XML-Sprache zur Erzeugung beliebiger Unicode-Streams aus XML-formatierten Eingaben
- Im Zusammenhang mit XSL:FO entwickelt
- Orientiert an LISP und DSSSL
- Setzt auf der Lokatorsprache *XPath* auf
- (Turing-)vollständige funktionale Programmiersprache

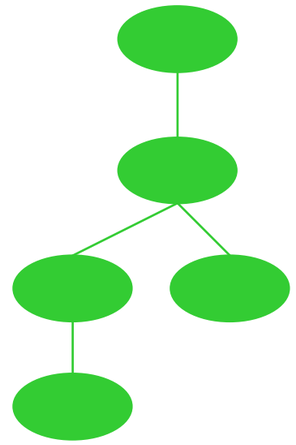
Transformation von XML-Dokumenten: XSL Transformations

- Architektur

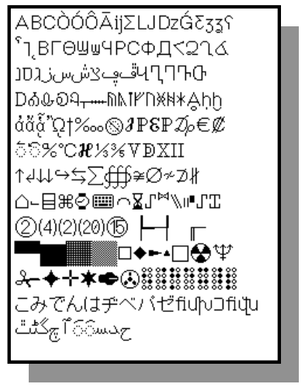


Quell-Dokument(-baum)

XSL Transformation



Ergebnis-Dokument(-baum)



Ergebnis-Unicode-Strom

- Anwendungsgebiete
  - Formattransformationen im e-business
  - Gewinnung von Präsentationsformaten (XSL:FO, (X)HTML)

Transformation von XML-Dokumenten: XSL Transformations

- Struktur einer XSLT-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="...">
    ...
  </xsl:template>
</xsl:transform
```

- Kein klassischer Kontrollfluß (d.h. kein main-Programm)
- Deklarative Schablonen (*templates*) beschreiben zu transformierende Sachverhalte

Transformation von XML-Dokumenten: XSL Transformations

- Struktur der Transformationsschablonen

Lokalisierungspfad

```
<xsl:template match="      ">
```

Ersetzungsmuster

```
</xsl:template>
```

- Beispiel

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

Transformation von XML-Dokumenten: XSL Transformations

- Transformationsregel

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

- Ausführung

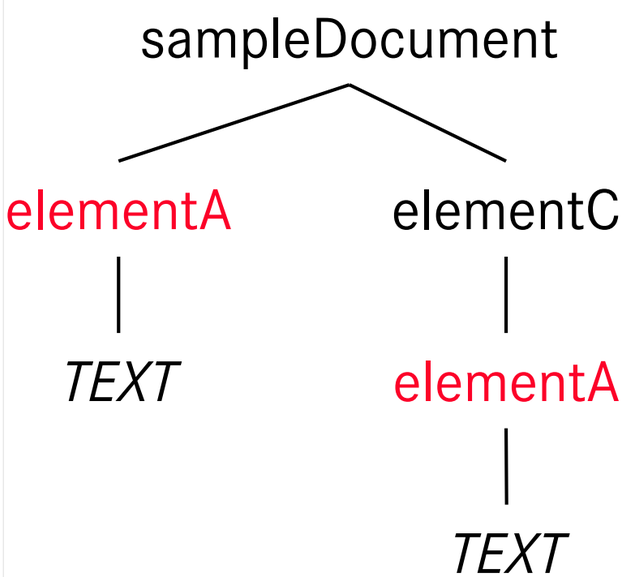
```
<?xml version="1.0" encoding="UTF-8"?>
<sampleDocument>
  <elementA>the quick brown fox ...</elementA>
  <elementC>
    <elementA>... over the lazy dog</elementA>
  </elementC>
</sampleDocument>
```

Transformation von XML-Dokumenten: XSL Transformations

- Transformationsregel

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

- Ausführung



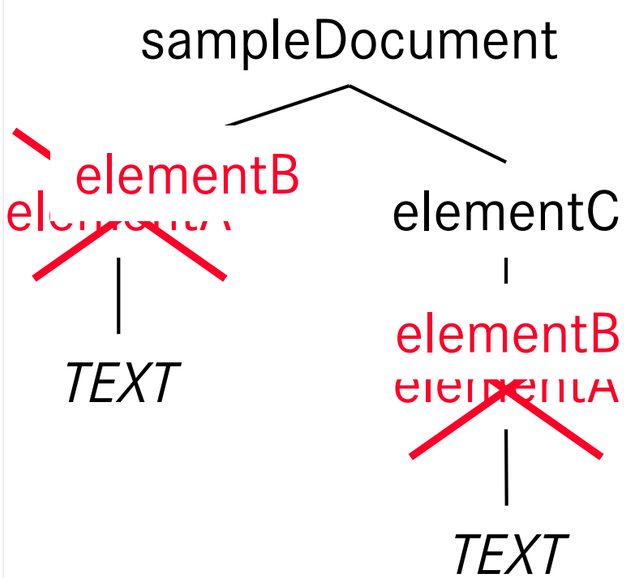

```
<?xml version="1.0" encoding="UTF-8"?>
<sampleDocument>
  <elementA>the quick brown fox ...</elementA>
  <elementC>
    <elementA>... over the lazy dog</elementA>
  </elementC>
</sampleDocument>
```

Transformation von XML-Dokumenten: XSL Transformations

- Transformationsregel

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

- Ausführung



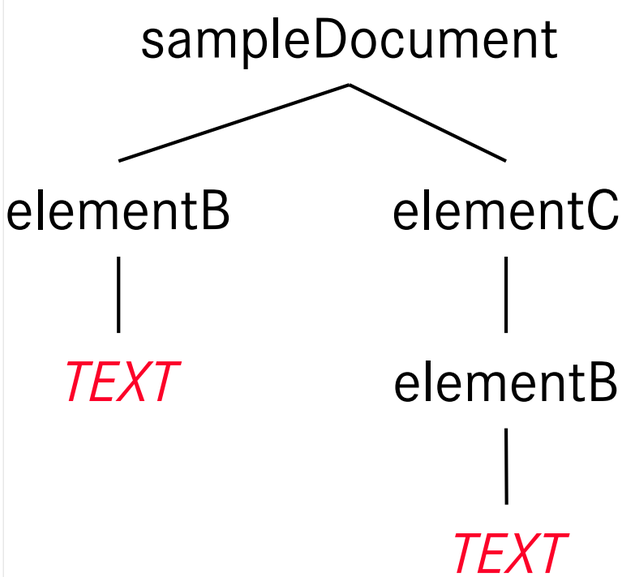

```
<?xml version="1.0" encoding="UTF-8"?>
<sampleDocument>
  <elementB>the quick brown fox ...</elementB>
  <elementC>
    <elementB>... over the lazy dog</elementB>
  </elementC>
</sampleDocument>
```

Transformation von XML-Dokumenten: XSL Transformations

- Transformationsregel

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

- Ausführung




```
<?xml version="1.0" encoding="UTF-8"?>
<sampleDocument>
  <elementB>the quick brown fox ...</elementB>
  <elementC>
    <elementB>... over the lazy dog</elementB>
  </elementC>
</sampleDocument>
```

Transformation von XML-Dokumenten: XSL Transformations

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superfein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X-363-23"/>
  </artikel>
</bestellung>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<order>
  <customer>
    <custID>X-363-23</custID>
  </customer>
  <itemlist>
    <item>
      <itemNO>4711</itemNO>
      <identification>Wusch Superfein</identification>
      <price currency="EUR">100.20...</price>
    </item>
  </itemlist>
</order>
```

Transformation von XML-Dokumenten: XSL Transformations

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exte

<xsl:output method="xml" encoding="ISO-8859-1" omit-xml-d

<xsl:template match="bestellung">
  <order>
    <xsl:apply-templates select="//kunde"/>
  </order>
</xsl:template>

<xsl:template match="kunde">
  <customer>
    <custID><xsl:value-of select="@nummer"/></custID>
  </customer>

  <itemlist>
  <xsl:apply-templates select="//artikel"/>
  </itemlist>
</xsl:template>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superfein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X-363-23"/>
  </artikel>
</bestellung>
```

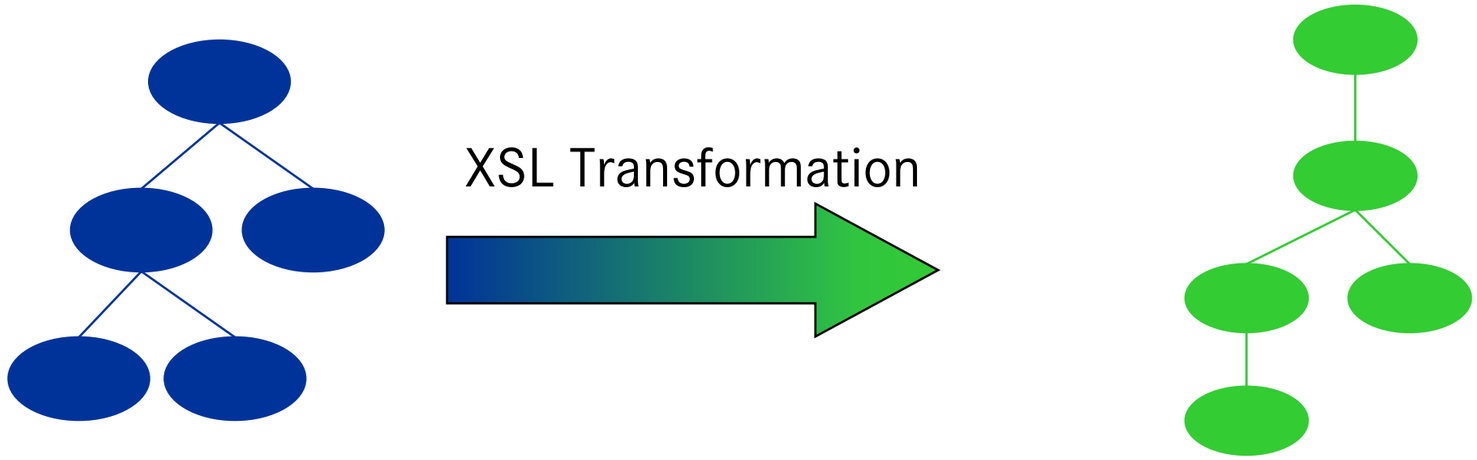
Transformation von XML-Dokumenten: XSL Transformations

```
<xsl:template match="artikel">
  <item>
    <itemNo>
      <xsl:value-of select="nummer"/>
    </itemNo>
    <identification>
      <xsl:value-of select="benennung"/>
    </identification>
    <price currency="EUR">
      <xsl:value-of select="number(./preis/betrag) * 0.511291881"/>
    </price>
  </item>
</xsl:template>

</xsl:transform>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bestellung>
  <artikel>
    <nummer>4711</nummer>
    <benennung>Wusch Superfein</benennung>
    <preis>
      <betrag>195.99</betrag>
      <waehrung>DEM</waehrung>
    </preis>
    <kunde nummer="X-363-23"/>
  </artikel>
</bestellung>
```

Transformation von XML-Dokumenten: XSL Transformations

- Erzeugung von (X)HTML-Darstellungen

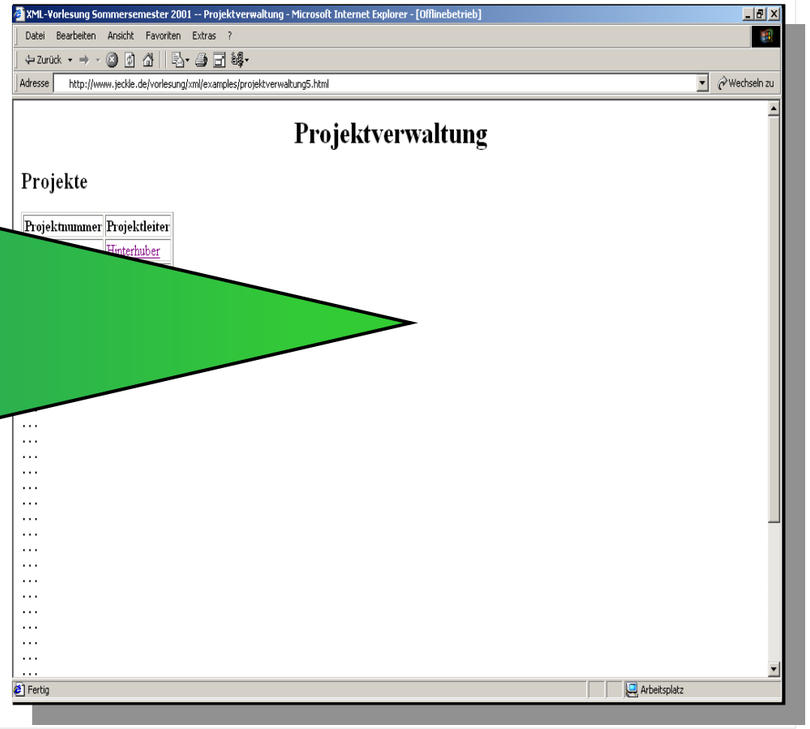


Quell-Dokument(-baum)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="projektverwaltung.xsl"?>

<ProjektVerwaltung
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.jeckle.de/vorlesung/xml/examples/projektverwaltung.xsd">
  <Person PersID="Pers01" mitarbeitInProjekt="Prj01">
  <Vorname>Hans</Vorname>
  <Nachname>Hinterhuber</Nachname>
  </Person>
  <Person PersID="Pers02" mitarbeitInProjekt="Prj02">
  <Vorname>Franz</Vorname>
  <Vorname>Xaver</Vorname>
  <Nachname>Obermüller</Nachname>
  <Qualifikationsprofil>
    <u>IT-Kompetenz</u><em>verschieden
    <Leistungsstufe>professionelle</Leistu
    <em><Qualifikation>Programmierung</
    verschiedener Programmiersprachen
    <em><u><Qualifikation>Entwickler</Qualifikation></u></em> von 1988-1990
    <u><Qualifikation>Projektleiterfunktion</Qualifikation></u>
    von <b>1990-93</b> im X42-Projekt in Abteilung AB&amp;C
  </Qualifikationsprofil>
  </Person>
  <Person PersID="Pers03" mitarbeitInProjekt="Prj02">
  <Vorname>Fritz</Vorname>
  <Nachname>Meier</Nachname>
  </Person>
  <Projekt id="Prj01" Projektleiter="Pers01" Mitarbeiter="Pers01"/>
  <Projekt id="Prj02" Projektleiter="Pers02" Mitarbeiter="Pers03"/>
</ProjektVerwaltung>
```

Ergebnis-Dokument(-baum)

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

<xsl:output method="xml" encoding="ISO-8859-1" omit-xml-declaration="no"  indent="yes"
    xmlns="http://www.w3.org/1999/xhtml" />

<xsl:template match="/">
  <html>
    <head>
      <title>XML-Vorlesung Sommersemester 2001 – Projektverwaltung</title>
    </head>
</xsl:template>

<xsl:template match="ProjektVerwaltung">
  <center><h1>Projektverwaltung</h1></center>

  <h2>Projekte</h2>
  <table border="1">
    <tr>
      <td><b>Projektnummer</b></td>
      <td><b>Projektleiter</b></td>
    </tr>
    <xsl:apply-templates select="Projekt"/>
  </table>
```



XML-Vorlesung Sommersemester 2001 -- Projektverwaltung - Microsoft Internet Explorer - [Offlinebetrieb]

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück

Adresse http://www.jeckle.de/vorlesung/xml/examples/projektverwaltung5.html Wechseln zu

Projektverwaltung

Projekte

Projektnummer	Projektleiter
Prj01	Hinterhuber
Prj02	Obermüller

. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .

Transformation von XML-Dokumenten: XSL Transformations

- Zusammenfassung: XSL Transformations (XSLT)
  - Geeignet zur Erzeugung beliebiger Darstellungen aus XML-Eingaben
  - Theoretisch unbegrenzte Mächtigkeit
  - In prozeduralen/objektorientierten Code überführbar
  - Verabschiedeter W3C-Standard
  - Leistungsfähige Open Source-Implementierungen verfügbar

Gliederung

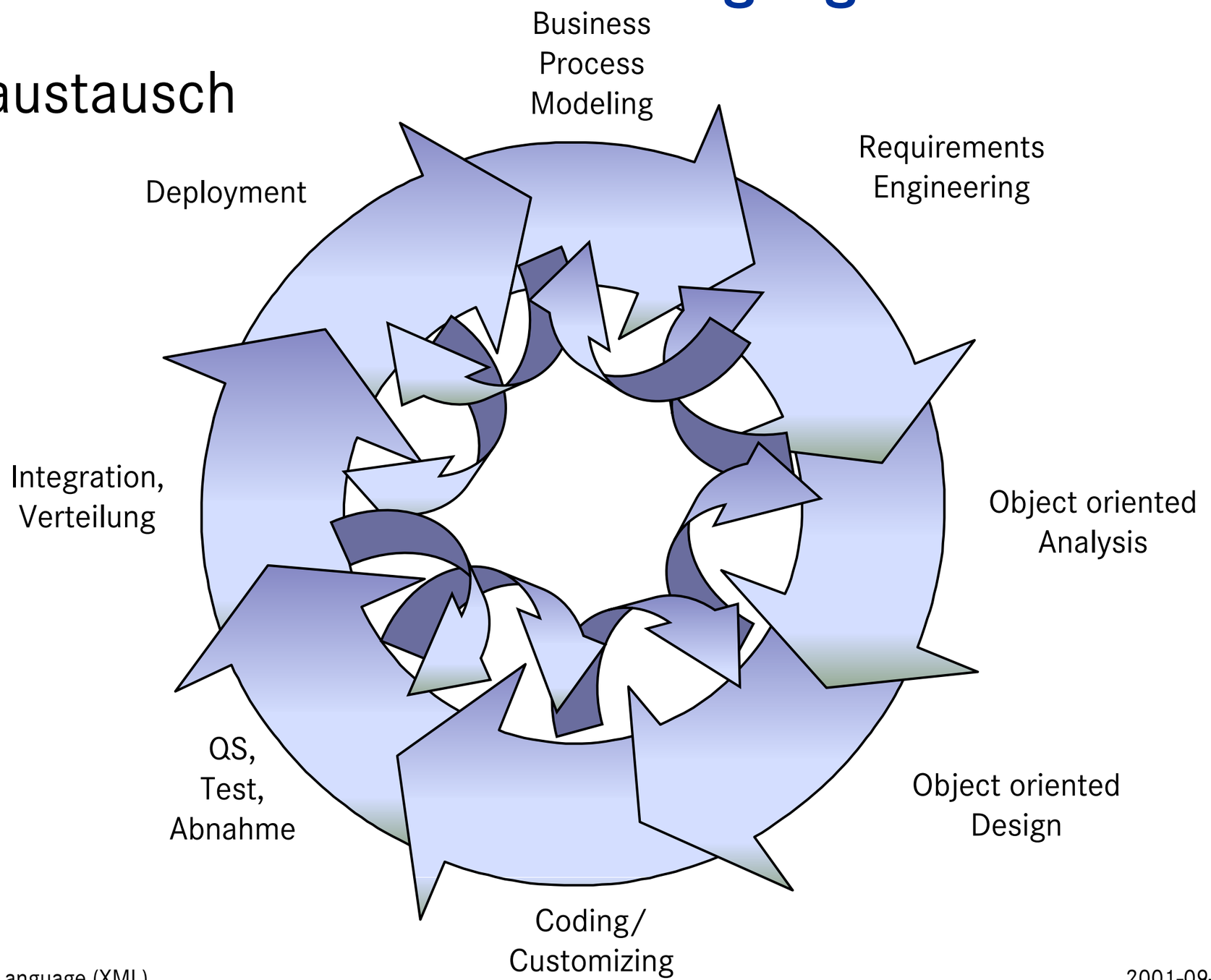
- XML-Standards und -Anwendungen der zweiten Generation ...
  - (Dokument-)Verknüpfungen: XML Links
  - Die Lokatorsprache XPath
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - Transformation von XML-Dokumenten: XSL Transformations
  - ➡ **Metadatenaustausch und Schemaerzeugung: XMI**
  - Die Anfragesprache XQuery

Metadatenaustausch und Schemaerzeugung: XMI

- Metadatenaustausch
  - XML-Darstellung für objektorientierte (Meta-)Modelle
  - Offene herstellerunabhängige Schnittstelle für CASE-Tools
  - Neutraler Ansatzpunkt für verschiedene Spezialwerkzeuge
- DTD-/Schemaerzeugung
  - Zunächst: Generierung der normativen DTDs aus den Metamodellen der UML und MOF
  - Prinzipiell: Möglichkeit der Erzeugung eigener XML-Vokabulare aus UML-Klassendiagrammen

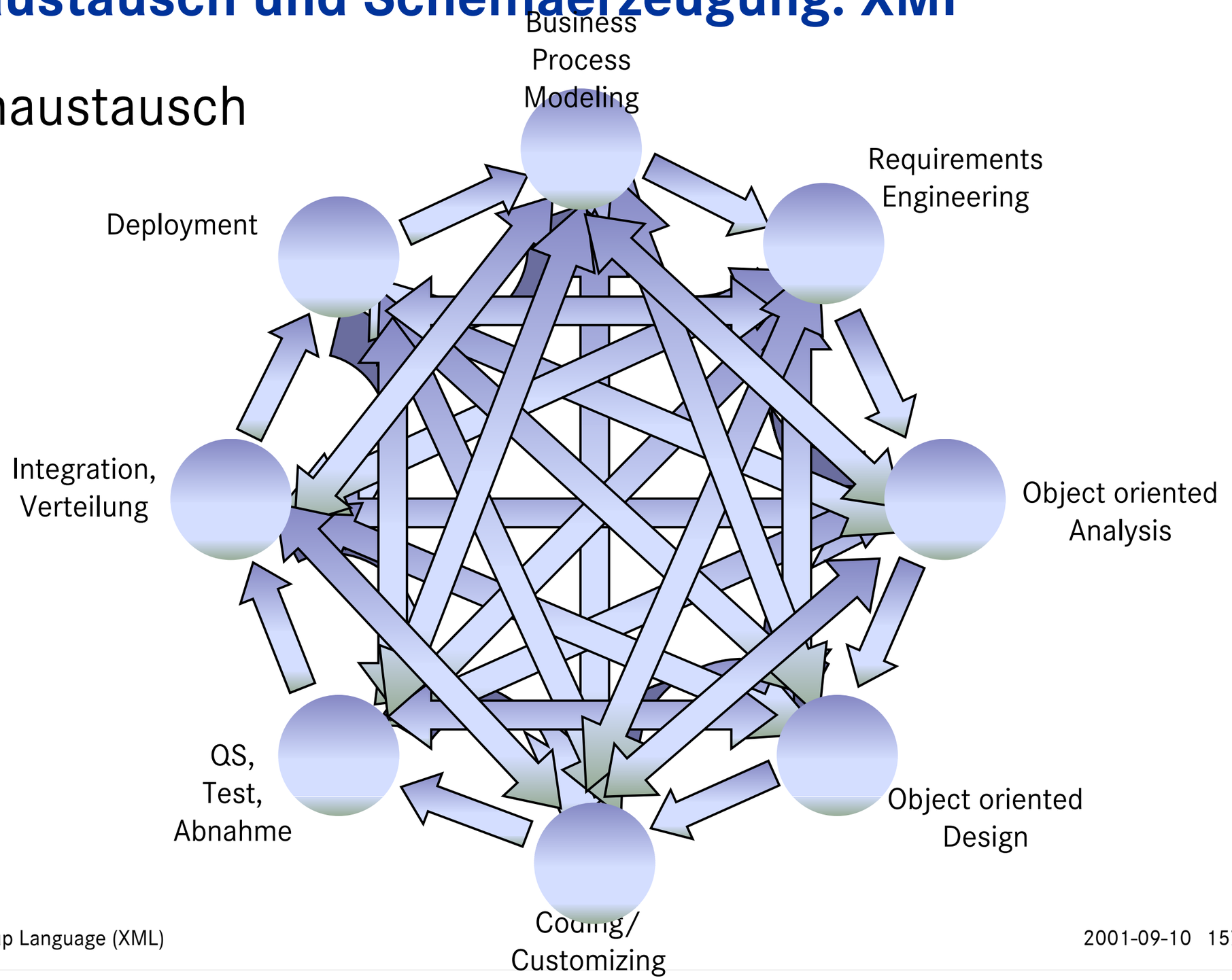
Metadatenaustausch und Schemaerzeugung: XMI

- Metadatenaustausch

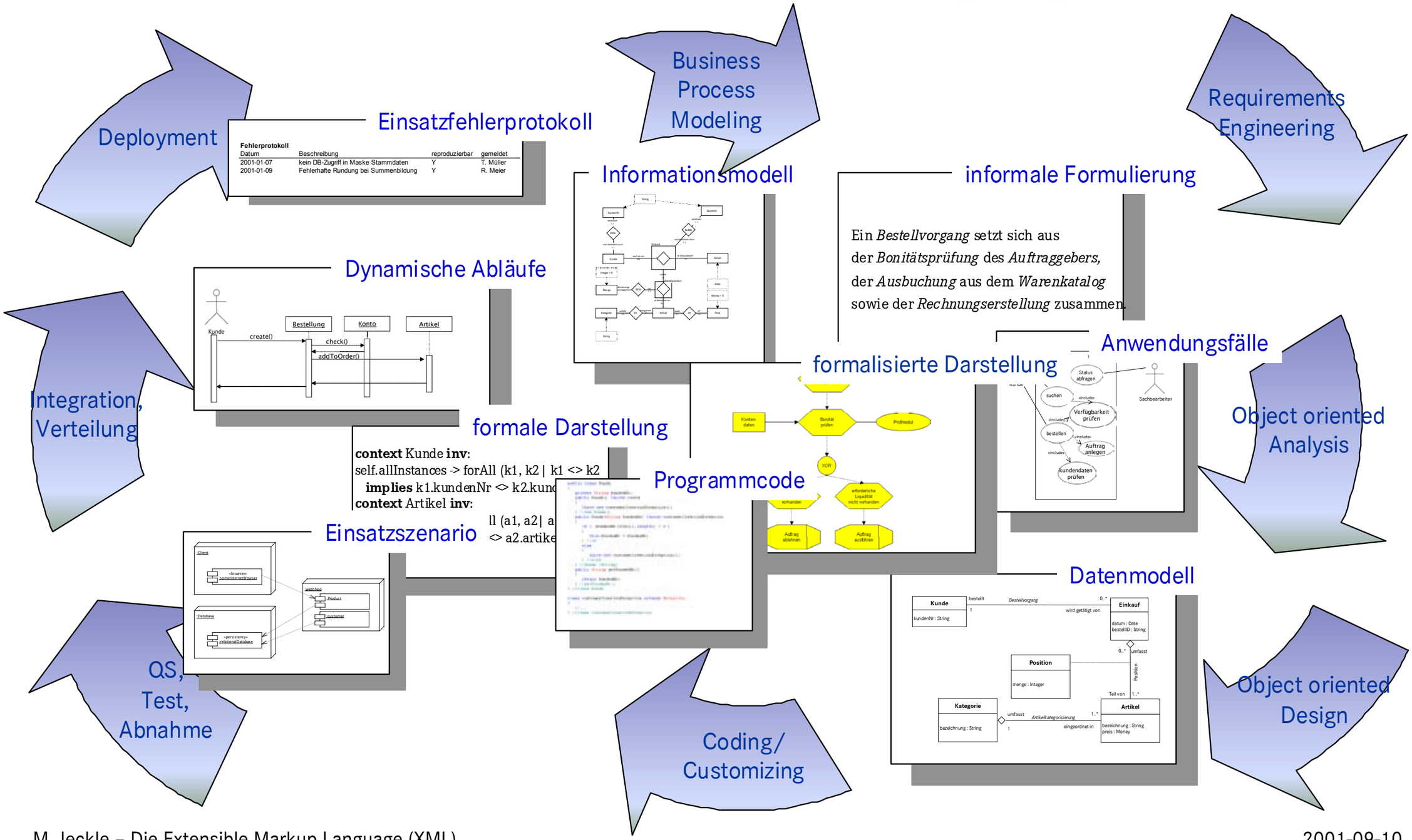


Metadatenaustausch und Schemaerzeugung: XMI

- Metadatenaustausch



Metadatenaustausch und Schemaerzeugung: XMI

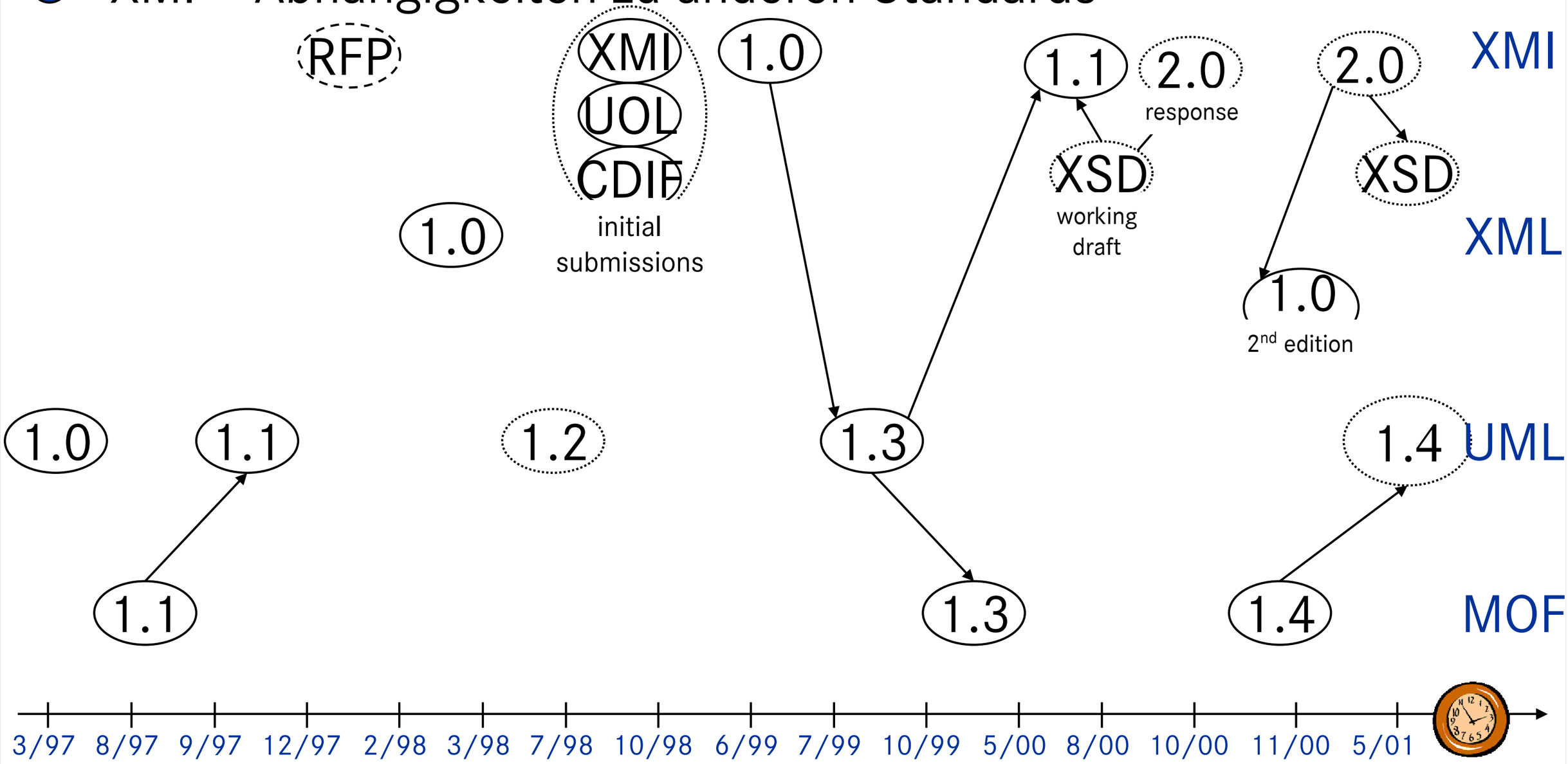


Metadatenaustausch und Schemaerzeugung: XMI

- Verabschiedeter OMG-Standard
  - erarbeitet durch: Unisys, IBM, DSTC, Oracle, Platinum, Fujitsu, Softeam, Recerca Informatica, DaimlerChrysler
  - unterstützt durch: Genesis, Inline, Rational, Select, Sprint, Cayenne, Sybase, Xerox, Boeing, Ardent, MCI Systemhouse, Aviatis, ICONIX, Integrated Systems, Verilog, Nihon Unisys, NTT, Telefonica I+D, NCR, Universitat Politecnica de Catalunya,

Metadatenaustausch und Schemaerzeugung: XMI

- XMI – Abhängigkeiten zu anderen Standards

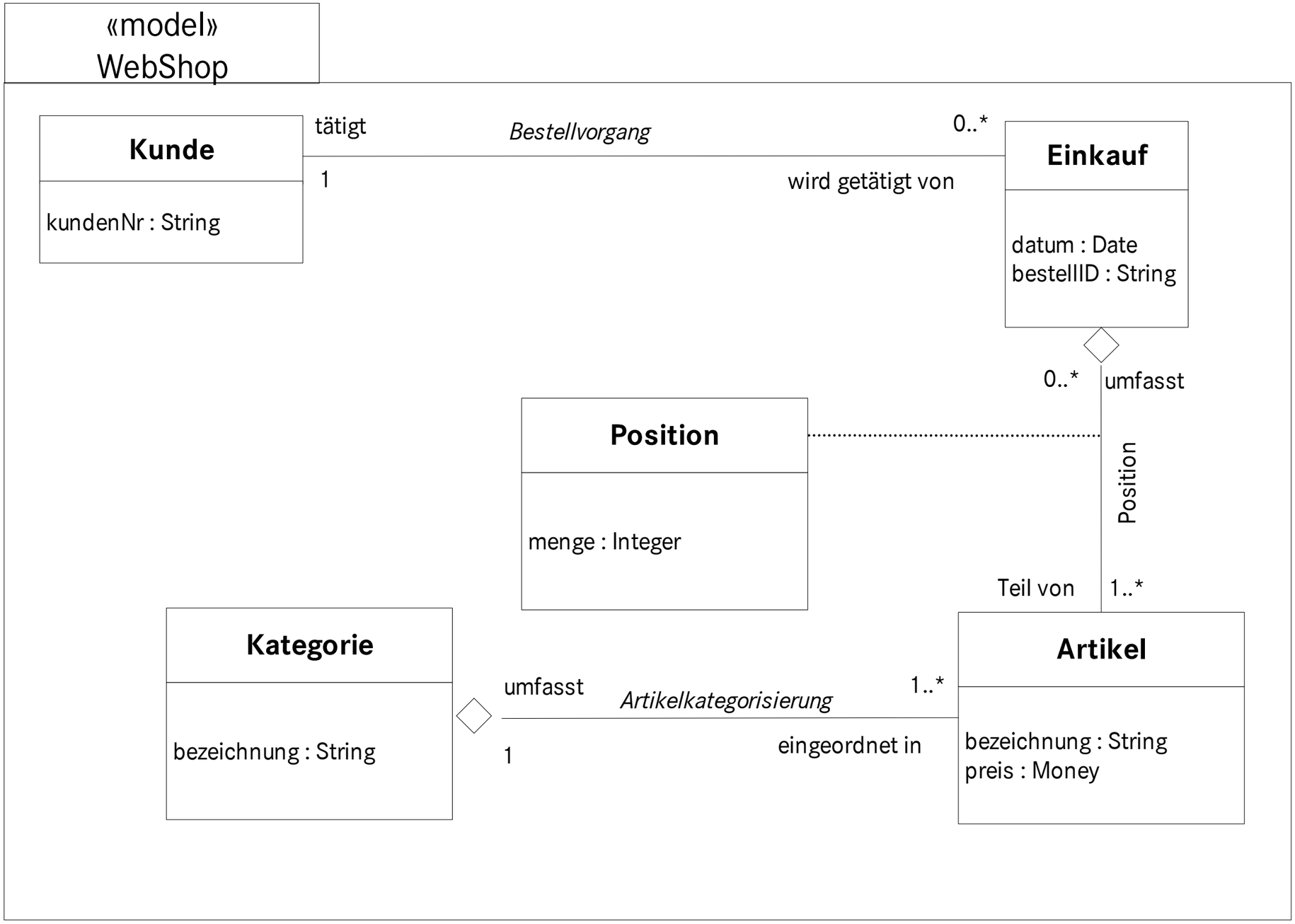


Metadatenaustausch und Schemaerzeugung: XMI

- Austausch von UML-Modellen
  - Document Type Definition (DTD) zur Darstellung der Modellinformation existiert (XMI[UML])
  - UML-Werkzeuge mit Umsetzung dieser Schnittstelle für Im- und Export sind verfügbar
  - Encoding aller Modellinformation incl. graphisch nicht dargestellter Vorgabewerte (Format ist *self-contained*)
  - Keine Darstellung der Präsentations- und Layoutinformation (nicht Bestandteil des UML-Metamodells (erst mit UML v2.0))
  - Nutzung von XML-Namespaces (ab XMI v1.1)
  - Unterstützung der W3C XML Schema Description (ab XMI v2.0)

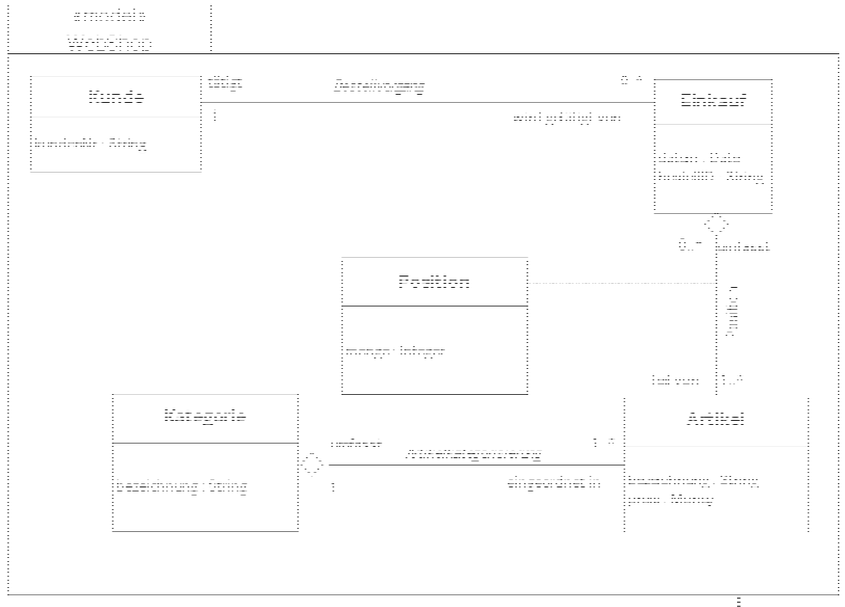
Metadatenaustausch und Schemaerzeugung: XMI

- Beispiel



Metadatenaustausch und Schemaerzeugung: XMI

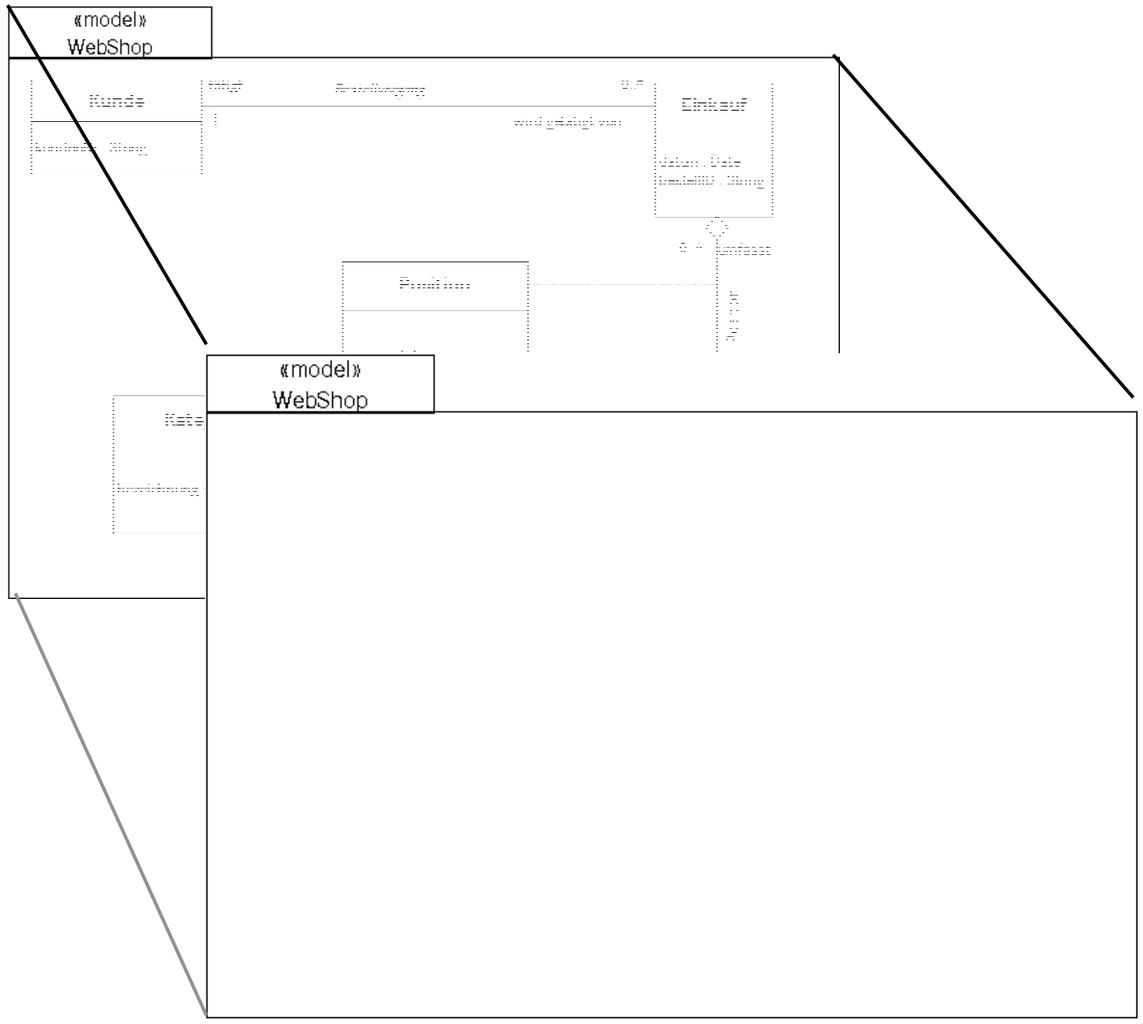
- Beispiel




```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<XMI xmi.version = '1.1'
      xmlns:UML='//org.omg/UML/1.3'>
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>Mario Jeckle</XMI.exporter>
      <XMI.exporterVersion>1.0 ;)</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name = 'UML'
       xmi.version = '1.3'/>
  </XMI.header>
```

Metadatenaustausch und Schemaerzeugung: XMI

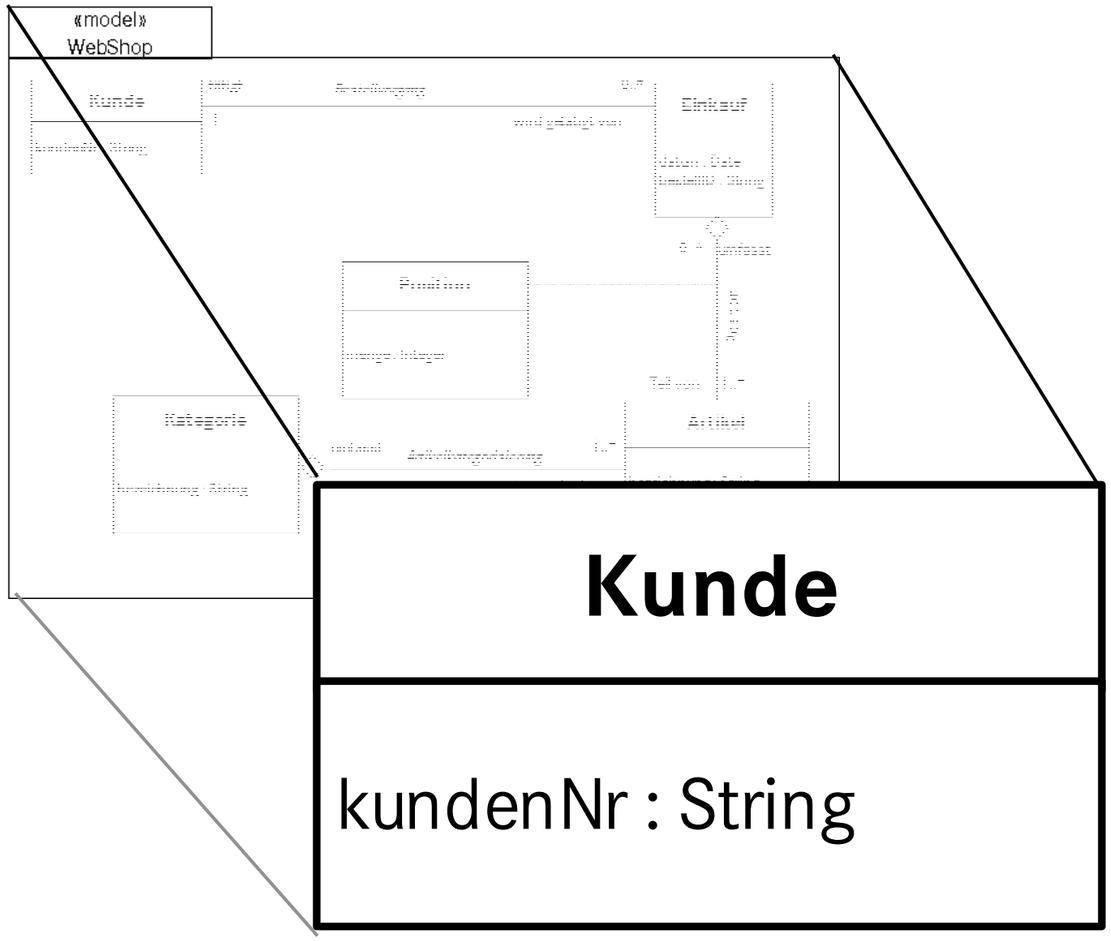
- Beispiel




```
<UML:Model
  xmi.id='Model:WebShop'
  name='WebShop'
  visibility='public'
  specification='false'
  isRoot='false'
  isLeaf = 'false'
  isAbstract = 'false' >
```

Metadatenaustausch und Schemaerzeugung: XMI

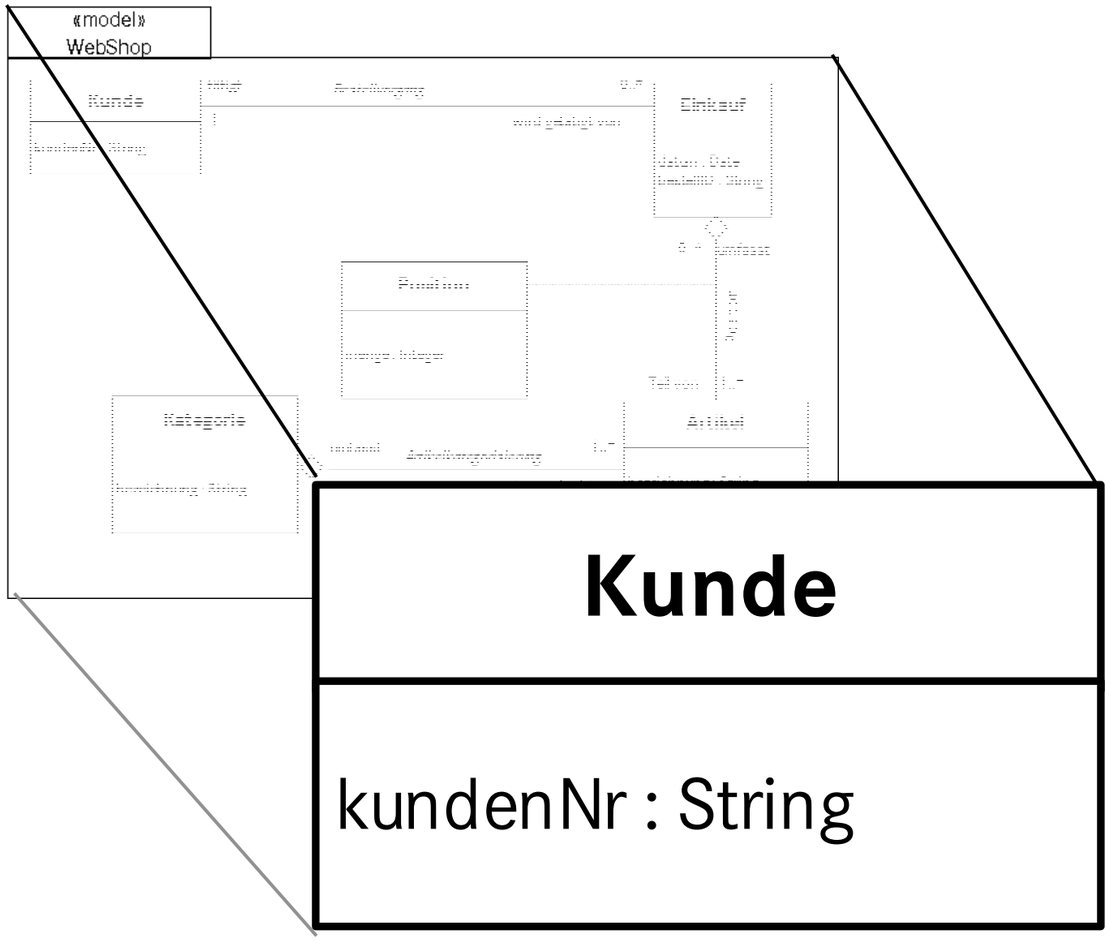
- Beispiel




```
<UML:Namespace.ownedElement>
    <UML:Class xmi.id = 'Class:Kunde'
      name = 'Kunde' visibility = 'public'
      specification = 'false'
      isRoot = 'true' isLeaf = 'true'
      isAbstract = 'false' isActive = 'false'
      namespace = 'Model:WebShop' >
```

Metadatenaustausch und Schemaerzeugung: XMI

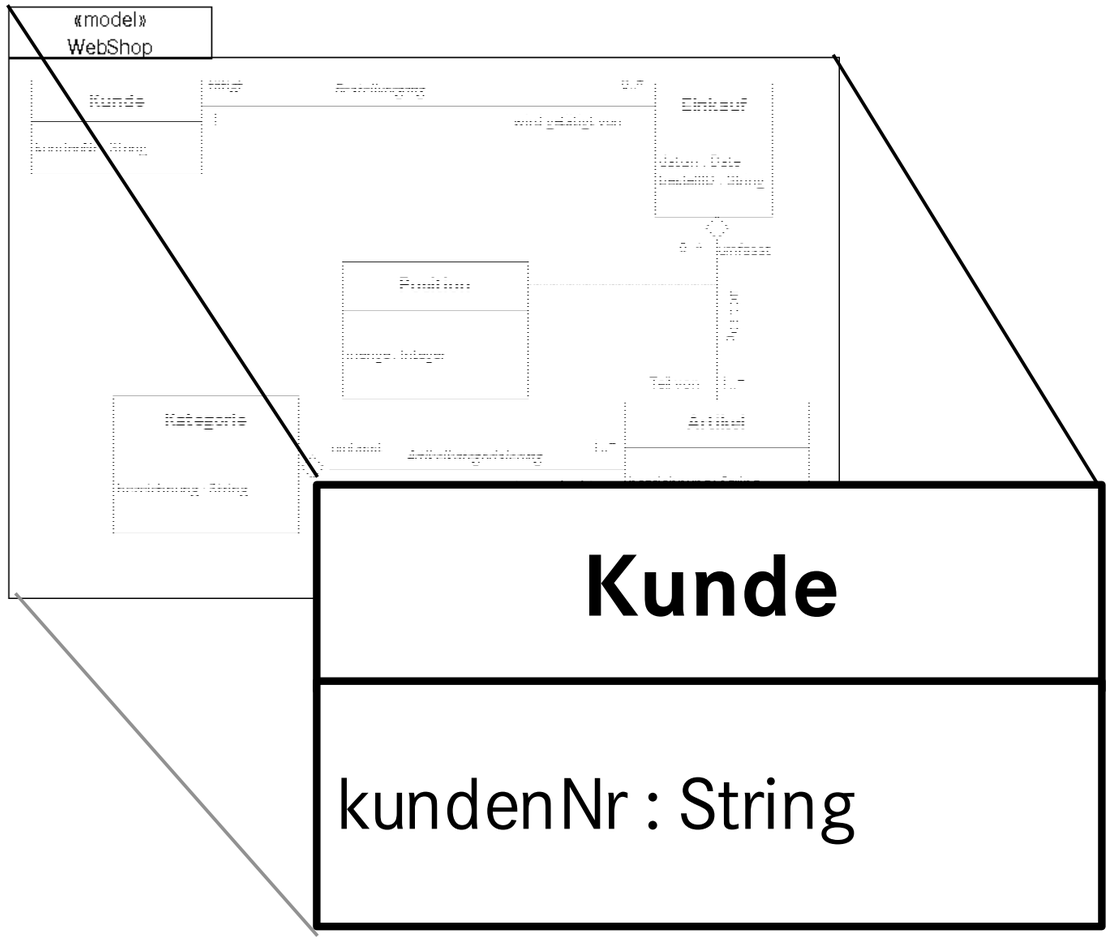
- Beispiel




```
<UML:Classifier.feature>

  <UML:Attribute

  xmi.id = 'Class:Kunde:Attribute:kundenNr'

  name = 'kundenNr' visibility = 'private'

  specification = 'false'   default

  ownerScope = 'instance'

  changeability = 'changeable'

  targetScope = 'instance'

  type = 'DataType:String' >
```

Metadatenaustausch und Schemaerzeugung: XMI

- Beispiel

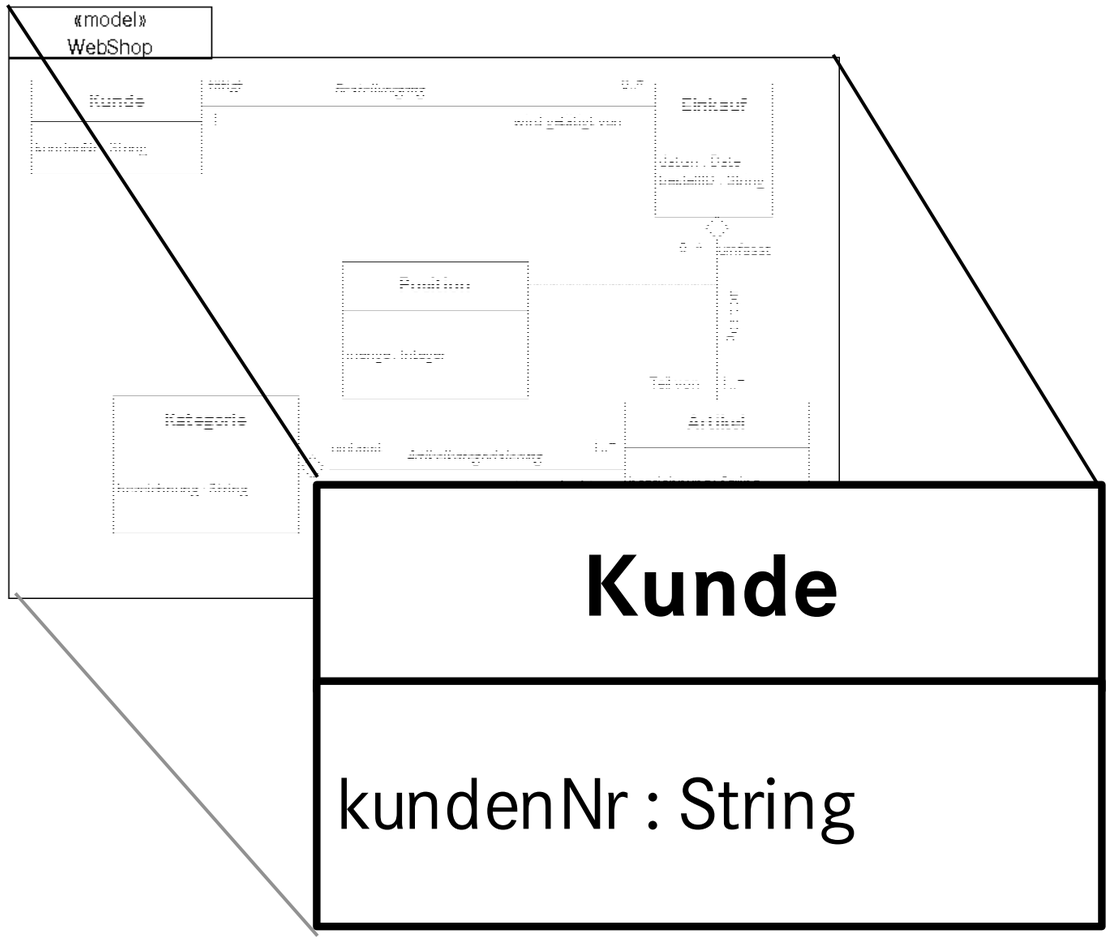



```
<UML:StructuralFeature.multiplicity>
  <UML:Multiplicity >
    <UML:Multiplicity.range>
      <UML:MultiplicityRange
        lower = '1' upper = '1' />
    </UML:Multiplicity.range>
  </UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
```

default

Metadatenaustausch und Schemaerzeugung: XMI

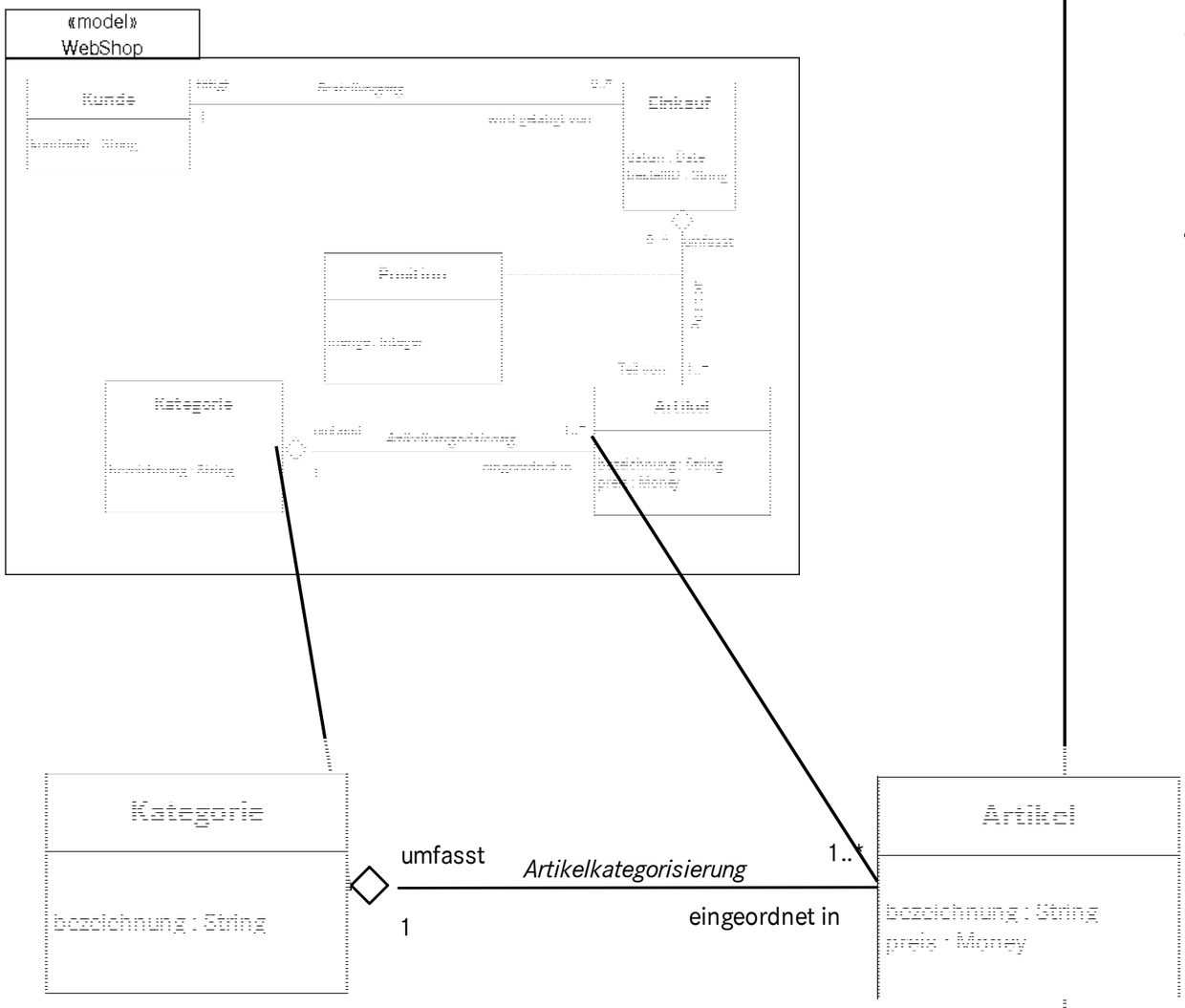
- Beispiel




```
<UML:DataType
   xmi.id = 'DataType:String'
   name = 'String'
   visibility = 'public'
   specification = 'false'
   isRoot = 'false'
   isLeaf = 'false'
   isAbstract = 'false' />
```

Metadatenaustausch und Schemaerzeugung: XMI

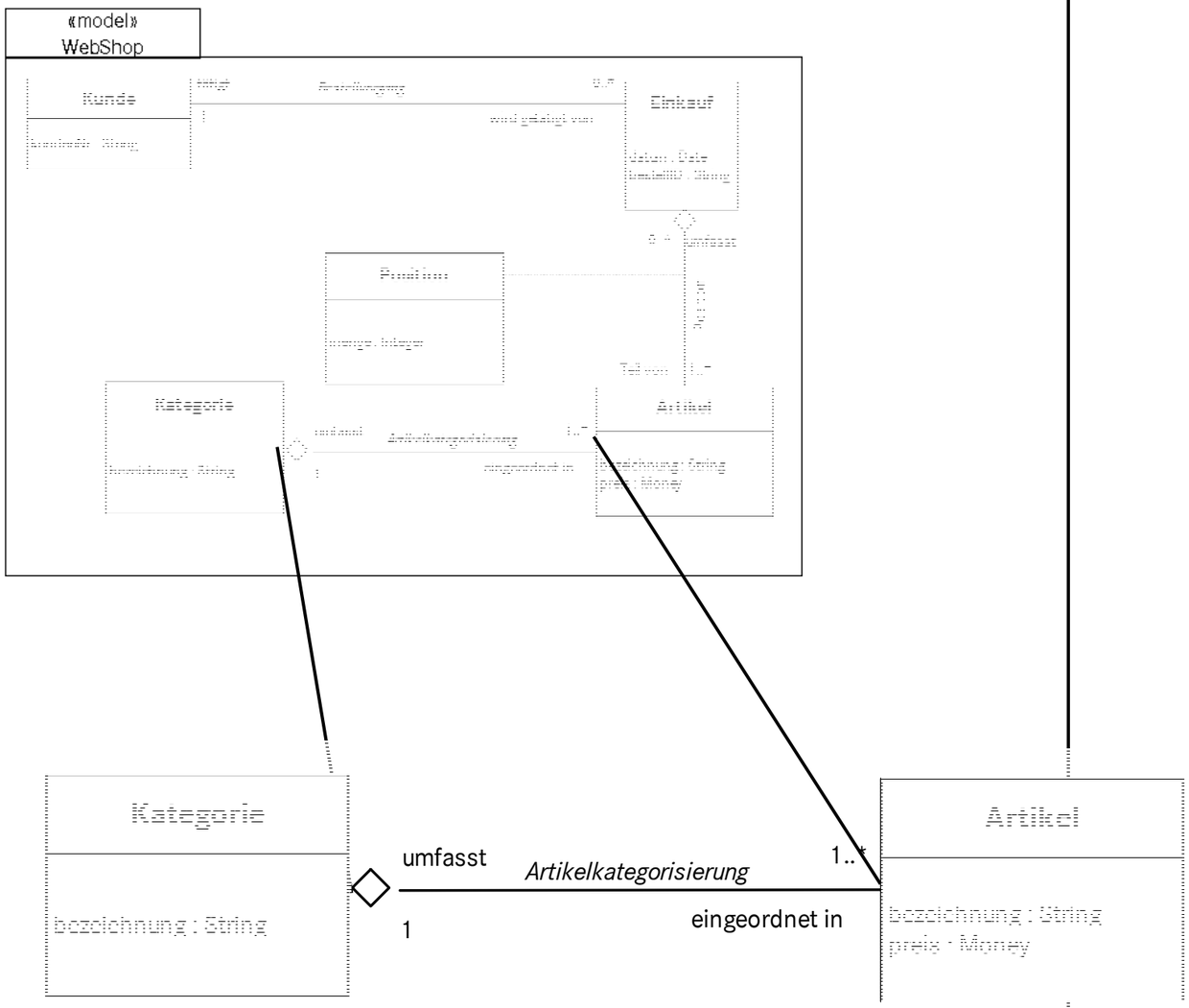
- Beispiel




```
<UML:Association
    xmi.id='Association:Artikelkategorisierung'
    name='Artikelkategorisierung'
    visibility='public'
    specification='false'   default
    isRoot='false'
    isLeaf='false'
    isAbstract='false' >
```

Metadatenaustausch und Schemaerzeugung: XMI

- Beispiel



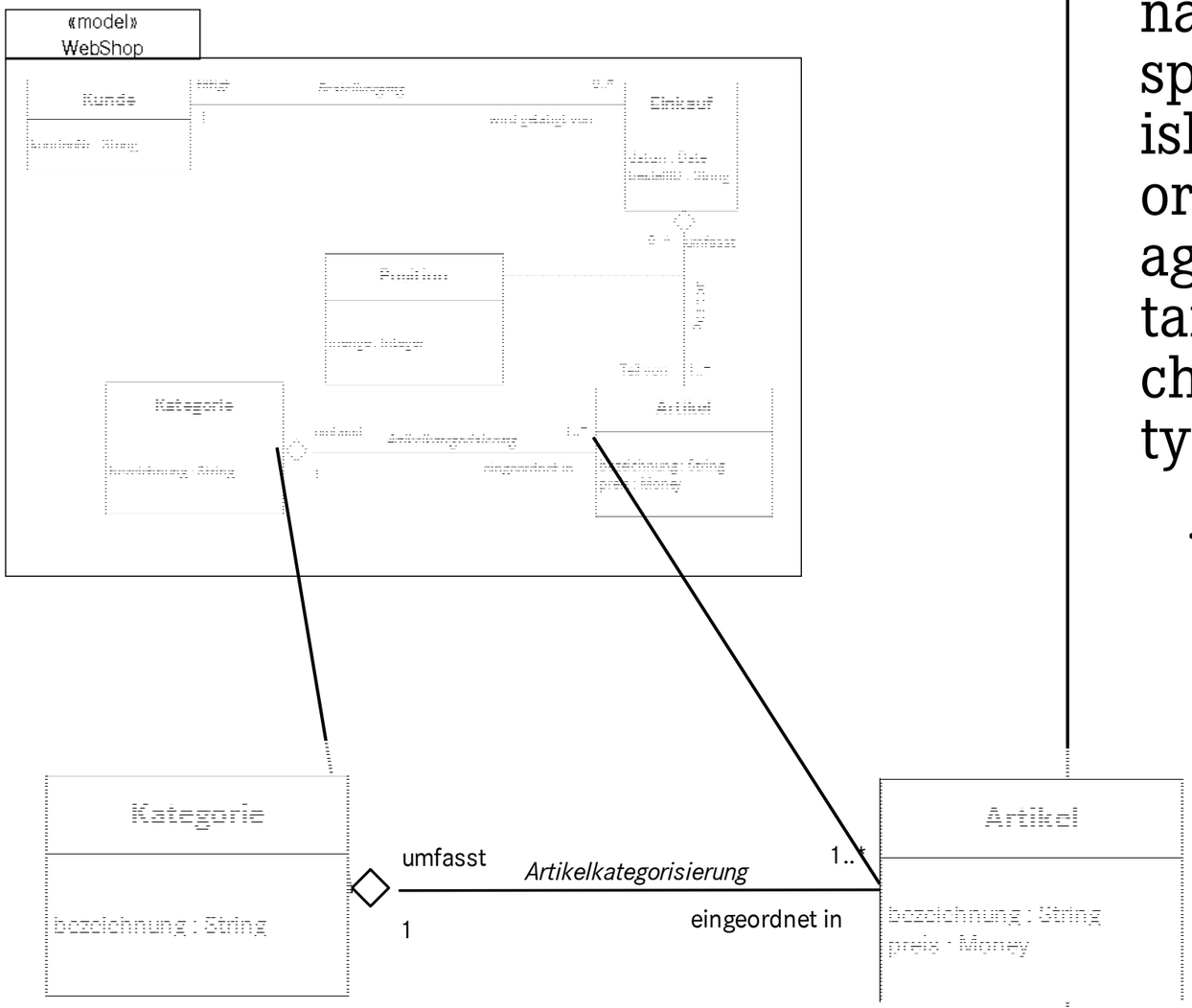

```
<UML:AssociationEnd
    xmi.id='Association:Artikelkategorisierung:Role:eingeordnet_in'
    name = 'eingeordnet_in'
    visibility = 'private'
    specification = 'false'
    isNavigable = 'true'
    ordering = 'unordered'
    aggregation = 'none'
    targetScope = 'instance'
    changeability = 'changeable'
    type = 'Class:Artikel' >

  <UML:AssociationEnd.multiplicity>
   <UML:Multiplicity >
    <UML:Multiplicity.range>
     <UML:MultiplicityRange
      lower = '1' upper = '-1' />
    </UML:Multiplicity.range>
   </UML:Multiplicity>
  </UML:AssociationEnd.multiplicity>
</UML:AssociationEnd>
```

default

Achtung!

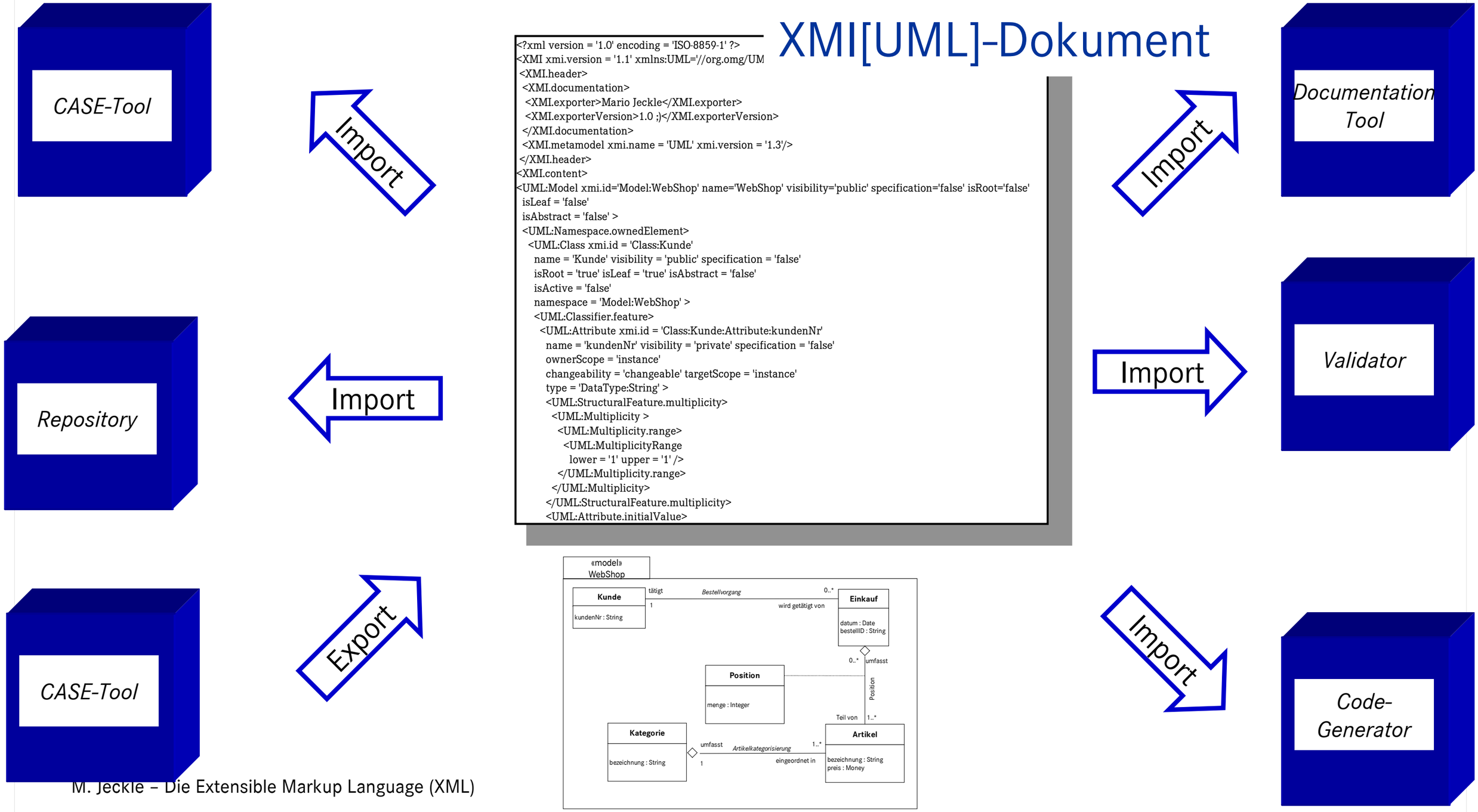
Metadatenaustausch und Schemaerzeugung: XMI

- Beispiel




```
<UML:AssociationEnd
  xmi.id = 'Association:Artikelkategorisierung:Role:umfaßt'
  name = 'umfaßt' visibility = 'private'
  specification = 'false'
  isNavigable = 'true'
  ordering = 'unordered'
  aggregation='aggregate'
  targetScope = 'instance'
  changeability = 'changeable'
  type = 'Class:Kategorie' >

    <UML:AssociationEnd.multiplicity>
      <UML:Multiplicity >
        <UML:Multiplicity.range>
          <UML:MultiplicityRange
            lower = '1' upper = '1' />
        </UML:Multiplicity.range>
      </UML:Multiplicity>
    </UML:AssociationEnd.multiplicity>
  </UML:AssociationEnd>
```

Metadatenaustausch und Schemaerzeugung: XMI

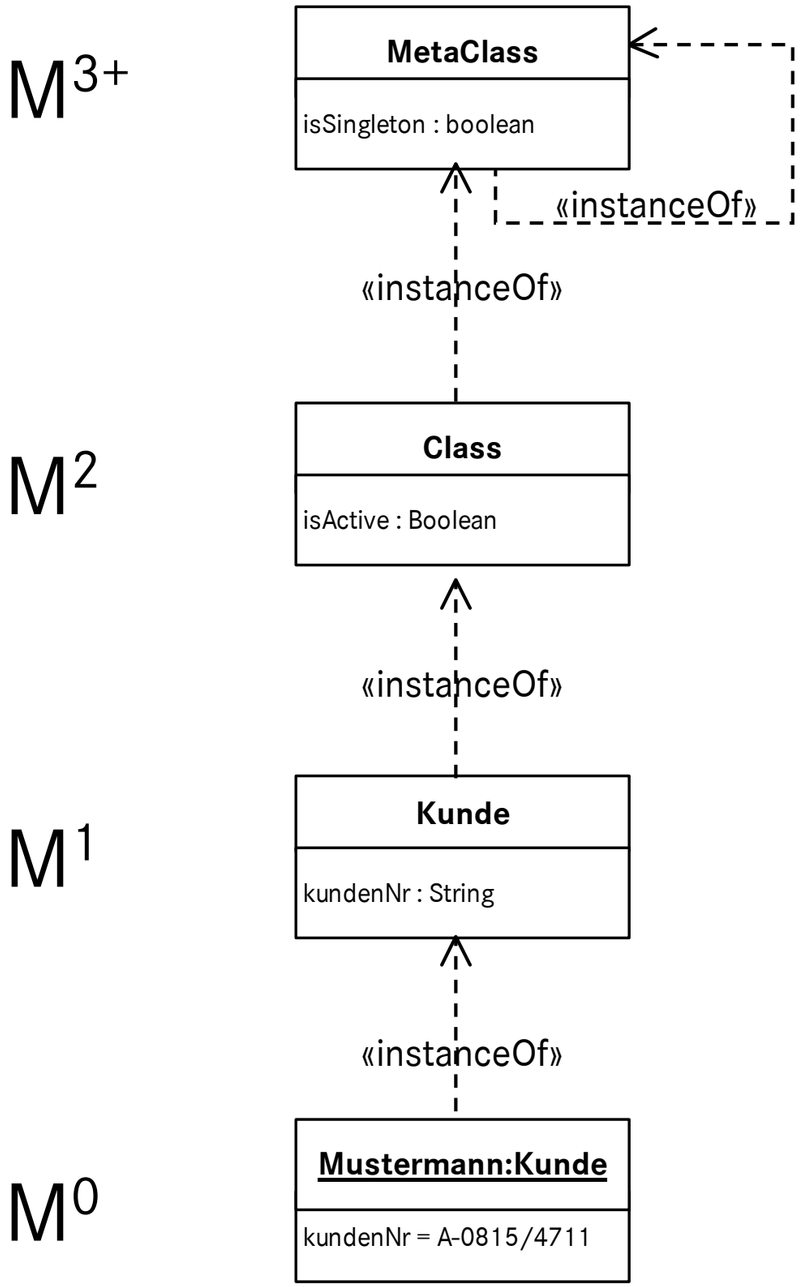



```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<XMI xmi.version = '1.1' xmlns:UML='//org.omg/UM
 <XMI.header>
  <XMI.documentation>
   <XMI.exporter>Mario Jeckle</XMI.exporter>
   <XMI.exporterVersion>1.0 ;)</XMI.exporterVersion>
  </XMI.documentation>
  <XMI.metamodel xmi.name = 'UML' xmi.version = '1.3'/>
 </XMI.header>
<XMI.content>
<UML:Model xmi.id='Model:WebShop' name='WebShop' visibility='public' specification='false' isRoot='false'
 isLeaf = 'false'
 isAbstract = 'false' >
 <UML:Namespace.ownedElement>
   <UML:Class xmi.id = 'Class:Kunde'
     name = 'Kunde' visibility = 'public' specification = 'false'
     isRoot = 'true' isLeaf = 'true' isAbstract = 'false'
     isActive = 'false'
     namespace = 'Model:WebShop' >
     <UML:Classifier.feature>
      <UML:Attribute xmi.id = 'Class:Kunde:Attribute:kundenNr'
        name = 'kundenNr' visibility = 'private' specification = 'false'
        ownerScope = 'instance'
        changeability = 'changeable' targetScope = 'instance'
        type = 'DataType:String' >
        <UML:StructuralFeature.multiplicity>
         <UML:Multiplicity >
           <UML:Multiplicity.range>
             <UML:MultiplicityRange
               lower = '1' upper = '1' />
           </UML:Multiplicity.range>
         </UML:Multiplicity>
        </UML:StructuralFeature.multiplicity>
        <UML:Attribute.initialValue>
```

Metadatenaustausch und Schemaerzeugung: XMI

- Verfügbare Werkzeuge mit XMI-Unterstützung

Metadatenaustausch und Schemaerzeugung: XMI



Meta-Metaklassenebene
Meta-Metamodell

Metaklassenebene
Metamodell

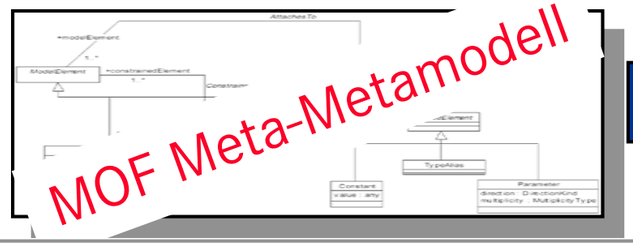
Modellebene

Ausprägungs- oder
Instanzenebene

Metadatenaustausch und Schemaerzeugung: XMI

- Erzeugung eigener XML-Vokabulare

M^{3+}

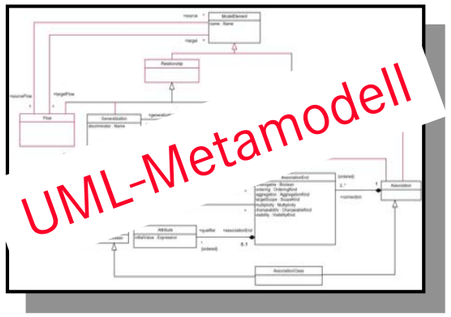


XML DTD Production

```
<!ELEMENT XMI (XMI.header?, XMI.content?, XMI.difference*,
        XMI.extensions*) >
<!ATTLIST XMI
        xmi.version CDATA #FIXED "1.1"
        timestamp CDATA #IMPLIED
        verified (true | false)
<!ELEMENT XMI.he
                                  odel*, XMI.metamodel*,
                                  ort*) >
<!ELEMENT XM
                                  DATA | XMI.owner | XMI.contact |
                                  scription | XMI.shortDescription |
                                  .exporter | XMI.exporterVersion |
                                  XMI.notice)* >
...
```

XMI[MOF]-DTD

M^2



encoding

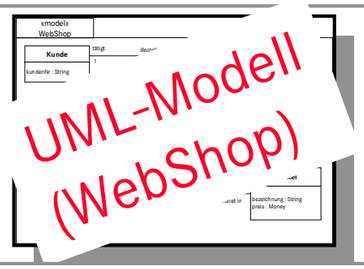
XMI[MOF]-Dokument

XML DTD Production

```
<!ELEMENT XMI (XMI.header?, XMI.content?                    ce*,
        XMI.extensions*) >
<!ATTLIST XMI
        xmi.version CD
        timesta
        verified
...
```

XMI[UML]-DTD

M^1

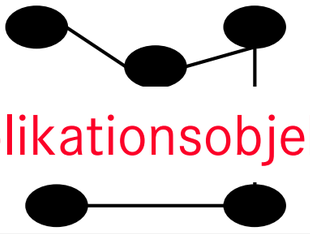


XML DTD Production

```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<XMI xmi.version = '1.1'
      xmlns:UML='//org.omg/UML/1.3'>
  <XMI.header>
 ....
```

XMI[WS]-DTD

M^0



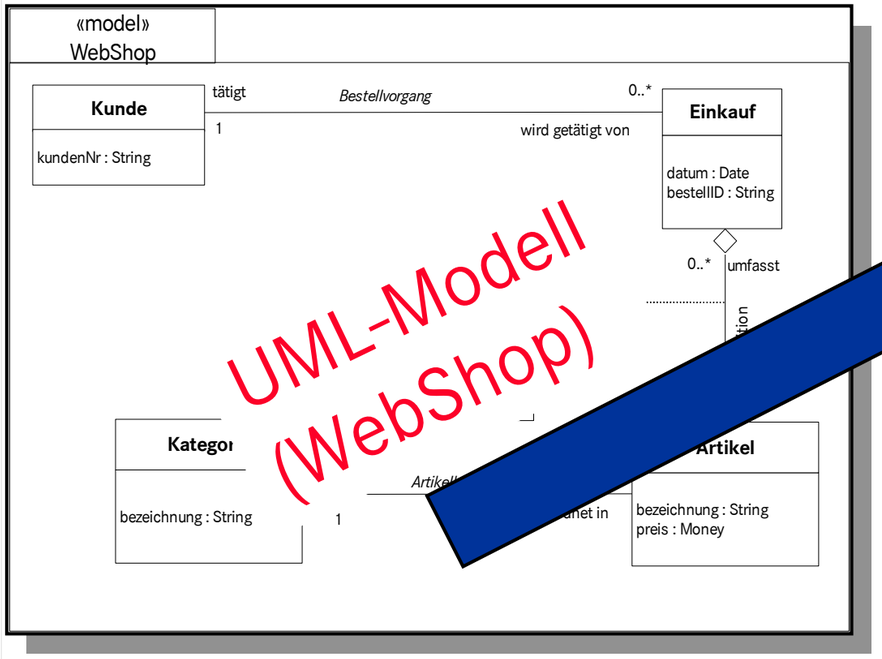
Applikationsobjekte

encoding

XMI[WS]-Dokument

Metadatenaustausch und Schemaerzeugung: XMI

- Erzeugung eigener XML-Vokabulare




```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<XMI xmi.version = '1.1' xmlns:UML='//org.omg/UML/1.3'>
 <XMI.header>
  <XMI.documentation>
   <XMI.exporter>Mario Jeckle
   <XMI.exporterVersion>1.0 ;
  </XMI.documentation>
  <XMI.metamodel xmi.name =
 </XMI.header>
<XMI.content>
<UML:Model                    WebShop' name='WebShop' visibility='public' specification='false' isRoot='false'
 isL
                = 'false' >
  UML:Namespace.ownedElement>
   <UML:Class xmi.id = 'Class:Kunde'
    name = 'Kunde' visibility = 'public' specification = 'false'
    isRoot = 'true' isLeaf = 'true' isAbstract = 'false'
```

WebShop-DTD

Metadatenaustausch und Schemaerzeugung: XMI

- Erzeugung eigener XML-Vokabulare
  - Orientierung an netzartiger Struktur des UML-Modells
  - Modifizierung der Minimal-Multiplizität
    zur Vermeidung endlos zirkulärer Strukturen
  - Unterscheidung zwischen definierendem und referenzierendem Informationsauftreten,
    zur Vermeidung redundanter Information
  - Unterstützung von Sichten durch Teilmodellgraphen

Metadatenaustausch und Schemaerzeugung: XMI

- UML-Aufzählungsdatentypen:

  <!ATTLIST *T* (*choices*) #IMPLIED>

- Alle weiteren Datentypen:

  <!ATTLIST *T* CDATA #IMPLIED>

- Assoziationen mit Maxmimal-Multiplizität größer 1:

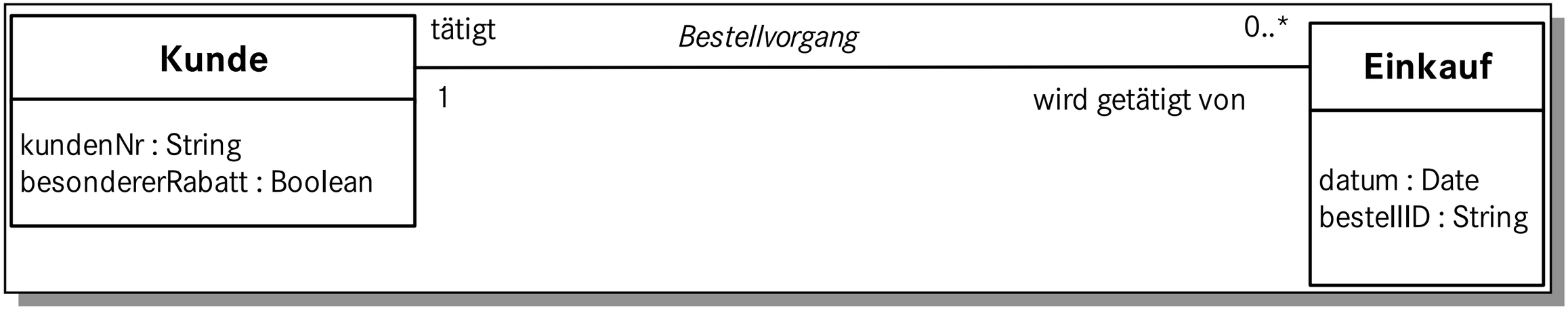
  <!ATTLIST *T* IDREFS #IMPLIED>
  <!ELEMENT *T* (... *foreignRole.referencedElement** ...)>

- Assoziationen mit Maximal-Multiplizität gleich 1:

  <!ATTLIST *T* IDREF #IMPLIED>
  <!ELEMENT *T* (... *referencedElement?* ...)>

Metadatenaustausch und Schemaerzeugung: XMI

- Erzeugung eigener XML-Vokabulare

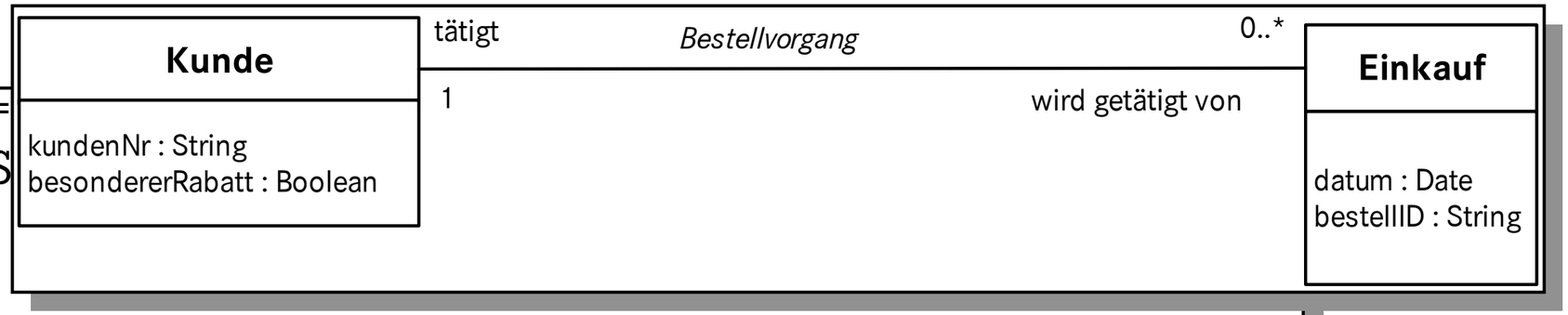



```
<!ELEMENT Kunde  (wird_getätigt_von.Einkauf* )>
<!ATTLIST Kunde  xmi.id            ID      #IMPLIED
                 xmi.idref         IDREF   #IMPLIED
                 kundenNr          CDATA   #IMPLIED
                 besondererRabatt  (true | false )  #IMPLIED>
<!ELEMENT wird_getätigt_von.Einkauf  (Einkauf )>
```

```
<!ELEMENT Einkauf  (tätigt.Kunde? )>
<!ATTLIST Einkauf  xmi.id          ID      #IMPLIED
                   xmi.idref       IDREF   #IMPLIED
                   tätigt.Kunde IDREFS  #IMPLIED
                   datum           CDATA   #IMPLIED
                   bestellID       CDATA   #IMPLIED >
<!ELEMENT tätigt.Kunde  (Kunde )>
```

Metadatenaustausch und Schemaerzeugung: XMI

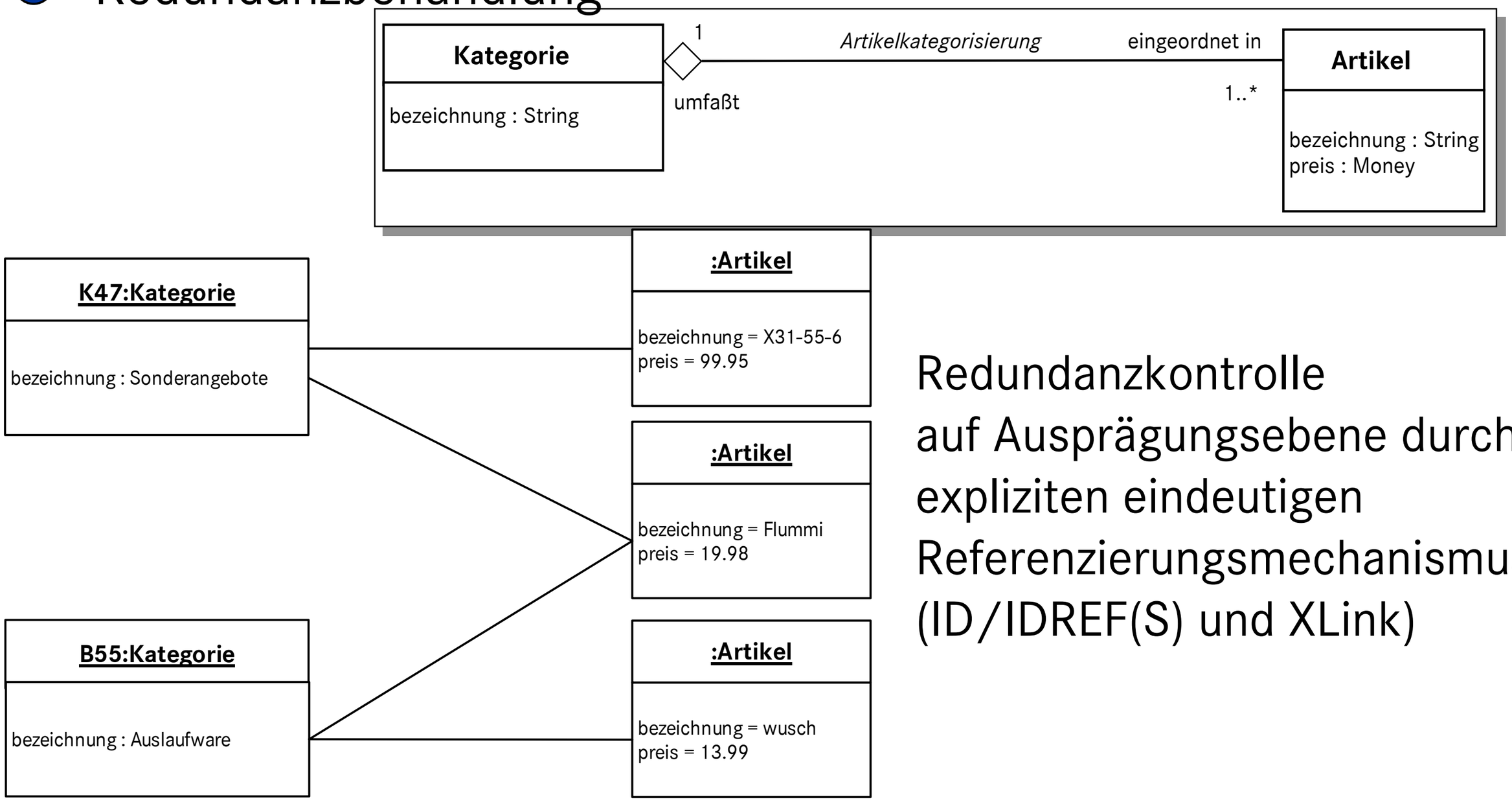
- Speicherung von Applikationsobjekten




```
<?xml version = "1.0" encoding=
<!DOCTYPE XMI SYSTEM "webS
<XMI xmi.version = "1.1">
  <XMI.header>
    <XMI.metamodel xmi.name = "webShop" xmi.version = "1.0"/>
  </XMI.header>
  <XMI.content>
    <webShop>
      <Kunde xmi.id = "Kunde:XMIID1" kundenNr = "A-0815/4711" besondererRabatt = "false">
        <wird_getätigt_von.Einkauf>
          <Einkauf xmi.id = "Einkauf:XMIID2" datum = "2001-09-10" bestellID = "AX46287"/>
        </wird_getätigt_von.Einkauf>
      </Kunde>
    </webShop>
  </XMI.content>
</XMI>
```

Metadatenaustausch und Schemaerzeugung: XMI

- Redundanzbehandlung



Redundanzkontrolle auf Ausprägungsebene durch expliziten eindeutigen Referenzierungsmechanismus (ID/IDREF(S) und XLink)

Metadatenaustausch und Schemaerzeugung: XMI

- Redundanzbehandlung

```
...
  <webShop>
    <Kategorie bezeichnung = "Sonderangebote">
      <eingeordnet_in.Artikel>
        <Artikel bezeichnung = "X31-55-6" preis = "99.95"/>
      </eingeordnet_in.Artikel>
      <eingeordnet_in.Artikel>
        <Artikel bezeichnung = "Flummi" preis = "19.98" xmi.id = "A1"/>
      </eingeordnet_in.Artikel>
    </Kategorie>
    <Kategorie bezeichnung = "Auslaufware">
      <eingeordnet_in.Artikel>
        <Artikel xmi.idref = "A1"/>
      </eingeordnet_ in.Artikel>
      <eingeordnet_in.Artikel>
        <Artikel bezeichnung = "wusch" preis = "13.99"/>
      </eingeordnet_in.Artikel>
    </Kategorie>
  </webShop>
...
```

Metadatenaustausch und Schemaerzeugung: XMI

- Anwendungsgebiete
  - (Meta-)Modellaustausch
  - XML-Spracherzeugung
  - Langzeitspeicherung von Modelldaten
  - Dokumentationsgenerierung (XSLT)
  - Modellvalidierung (Qualitätssicherung; Metriken, etc.)
  - Codegenerierung
  - Prototypengenerierung
  - Versionsverwaltung mit textbasierten Standardwerkzeugen
  - ...

Metadatenaustausch und Schemaerzeugung: XMI

- Zusammenfassung: *XML Metadata Interchange*
  - XML-basierte Darstellung von objektorientierten Meta-Modellen
  - DTDs für UML und MOF durch die OMG standardisiert
  - Standardschnittstelle für Modellaustausch in heterogenen Werkzeuglandschaften
  - Einfache Möglichkeit, eigene XML-Vokabulare im Rahmen eines Modell-zentrierten Entwicklungsprozesses zu erzeugen
  - Werzeugunterstützung verfügbar

Gliederung

- XML-Standards und -Anwendungen der zweiten Generation ...
  - (Dokument-)Verknüpfungen: XML Links
  - Die Lokatorsprache XPath
  - Erzeugung von Präsentationssichten: XML Stylesheets
  - Transformation von XML-Dokumenten: XSL Transformations
  - Metadatenaustausch und Schemaerzeugung: XMI
  - ➡ **Die Anfragesprache XQuery**

Die Anfragesprache XQuery

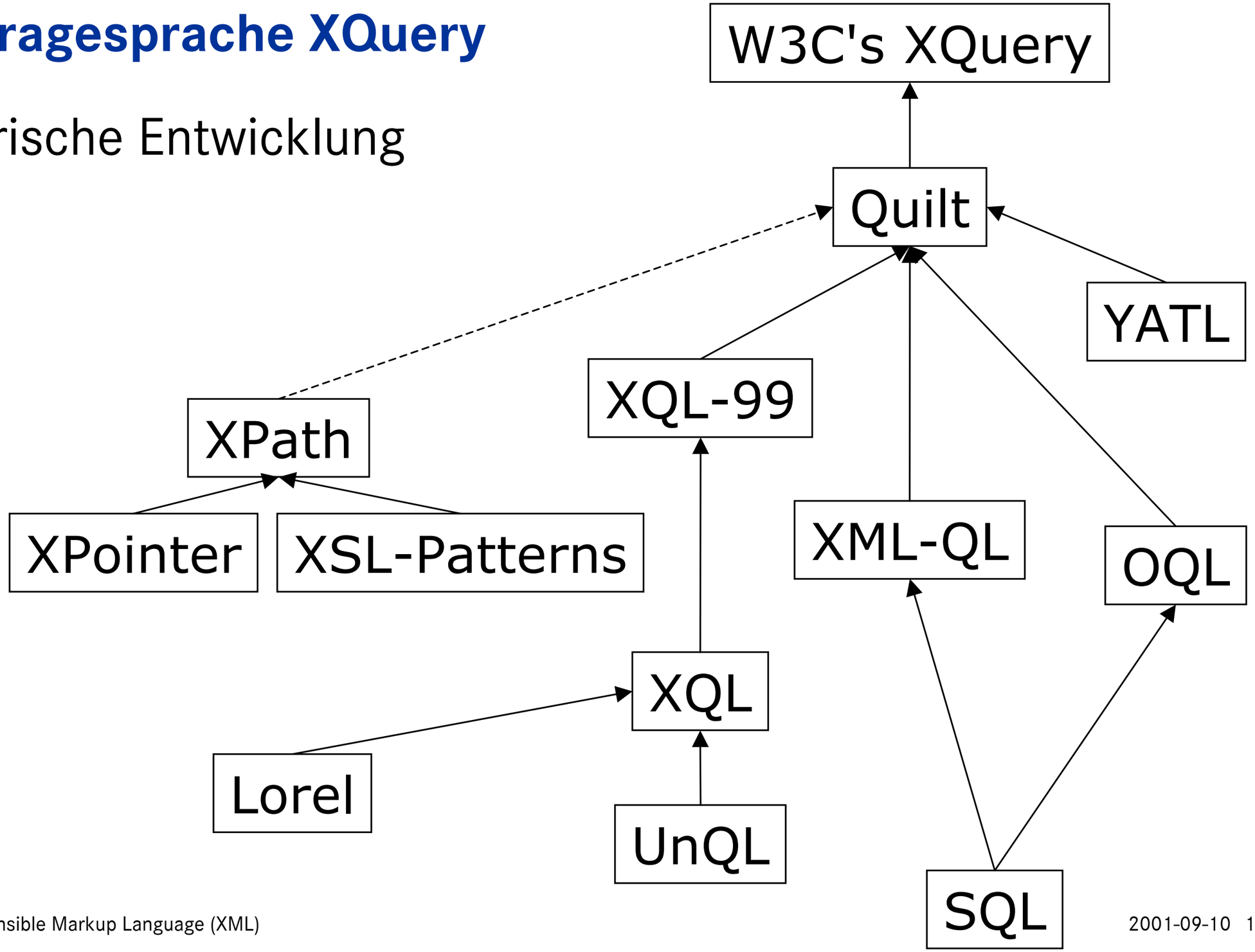
- Low-Level APIs sind für Anfragen und Auswertungen denkbar schlecht geeignet ...
- Aktivität der XML Query Working Group
- Definiert Datenmodell für XML-Dokumente, basierend auf XML InfoSet und Namensräumen
- Anfrageoperatoren für das Datenmodell
- Anfragesprache basierend auf den definierten Operatoren
- Anfragen können sich auf einzelne Dokumente oder eine Reihe von XML-Dokumenten beziehen
- Anfrageergebnis ist ein komplettes Dokument, ein beliebiger Teilbaum oder ein neu konstruiertes Dokument

Die Anfragesprache XQuery

- Anforderungen an eine Anfragesprache
  - **Abgeschlossenheit**: Ein- und Ausgabeformat ist XML
  - **Präzise Semantik**: Formale denotationelle Semantik
  - **Optimierbarkeit**: Anfrageoptimierung (syntaktisch: Umbau) vor ihrer Ausführung
  - **Adäquatheit**: Unterstützung der verschiedenen XML-Primitive
  - **Operationen**: Selektion, Extraktion, Reduktion, Verbund, Restrukturierung und Kombination
  - **Schema-Freiheit**: Anfragen müssen auch ohne Vorliegen eines Schemas ausführbar sein, ist es vorhanden, soll es in die Auswertung einbezogen werden

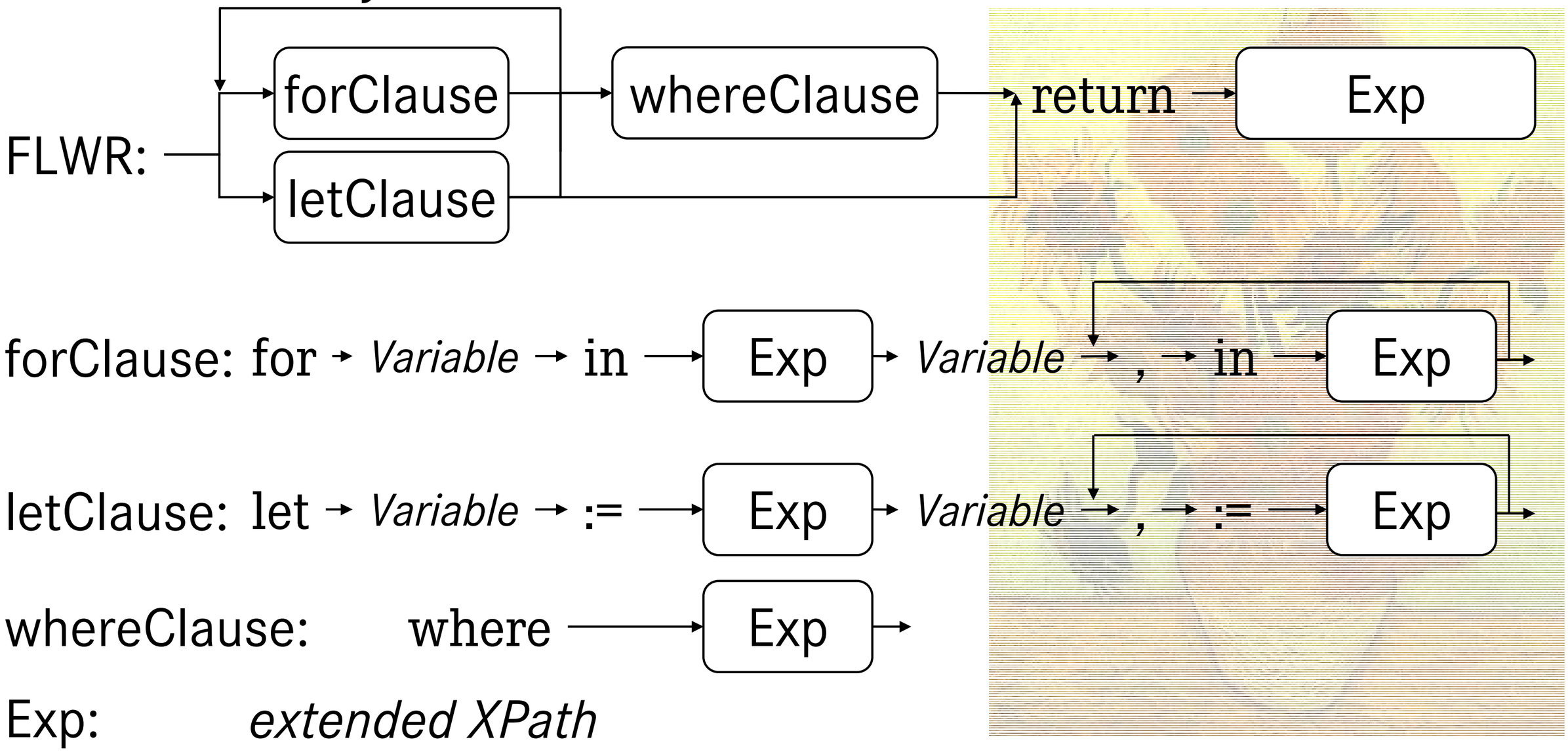
Die Anfragesprache XQuery

- Historische Entwicklung



Die Anfragesprache XQuery

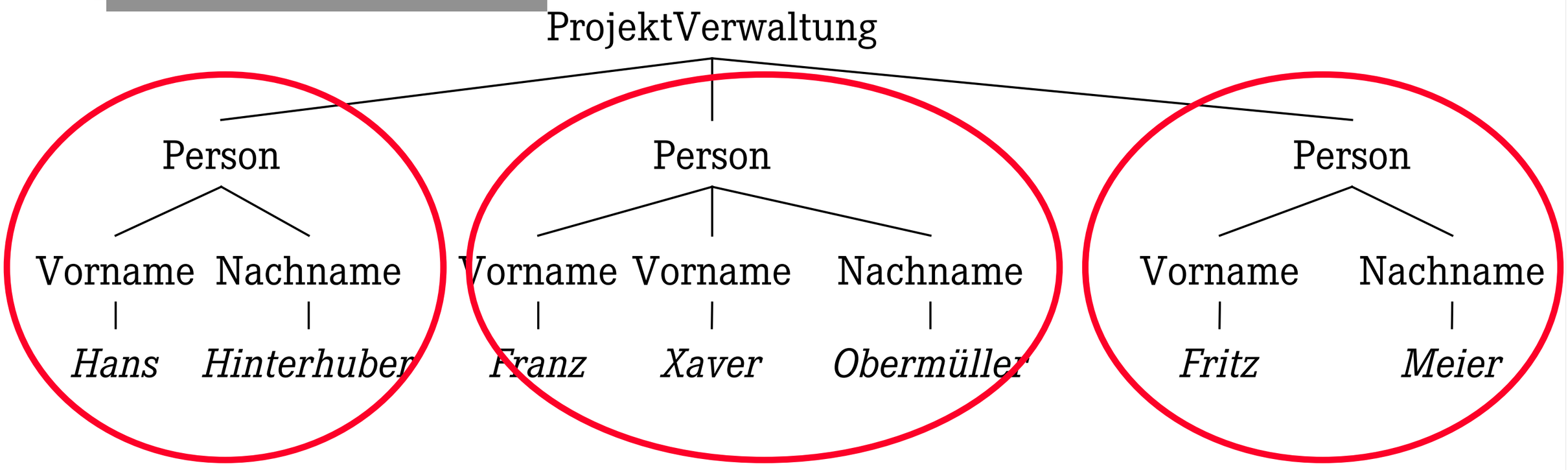
- Die FLWR-Syntax



Die Anfragesprache XQuery

- Die FLWR-Syntax im Beispiel

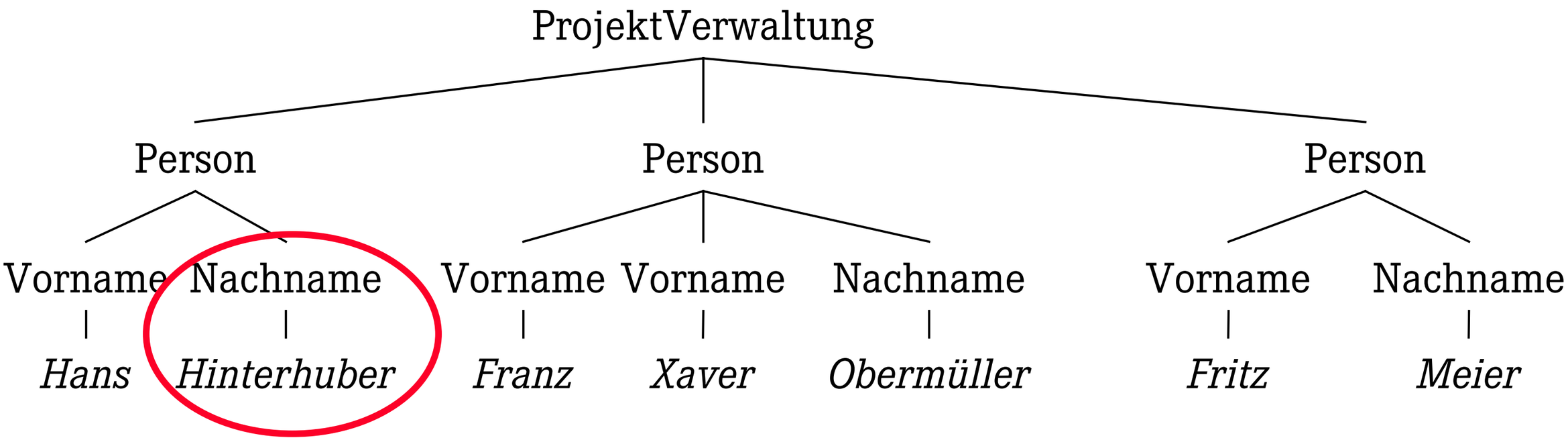
```
FOR $x IN //Person
RETURN $x
```



Die Anfragesprache XQuery

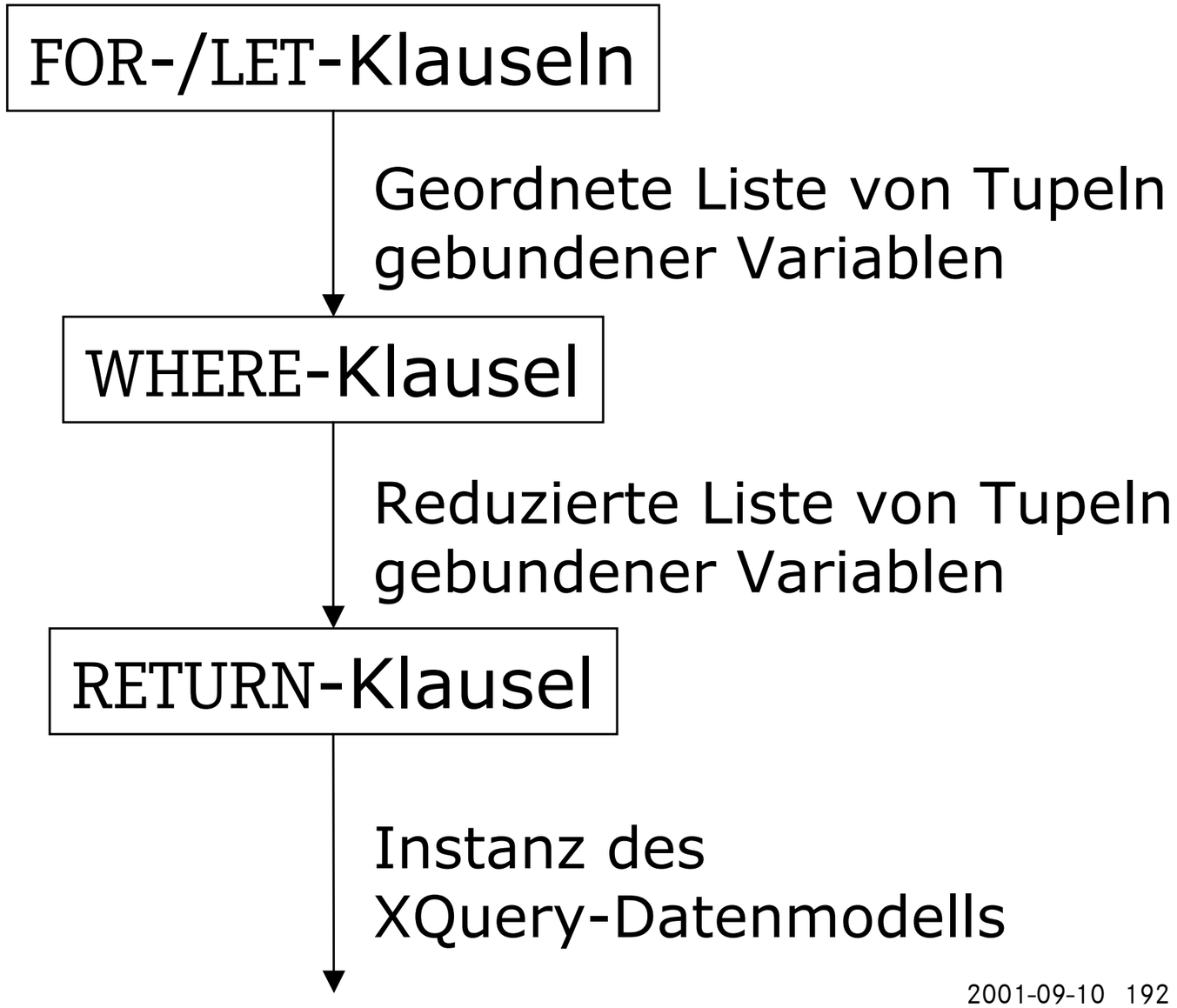
- Die FLWR-Syntax im Beispiel

```
FOR $y IN //Person
WHERE $y/Vorname = 'Hans'
RETURN $y/Nachname
```



Die Anfragesprache XQuery

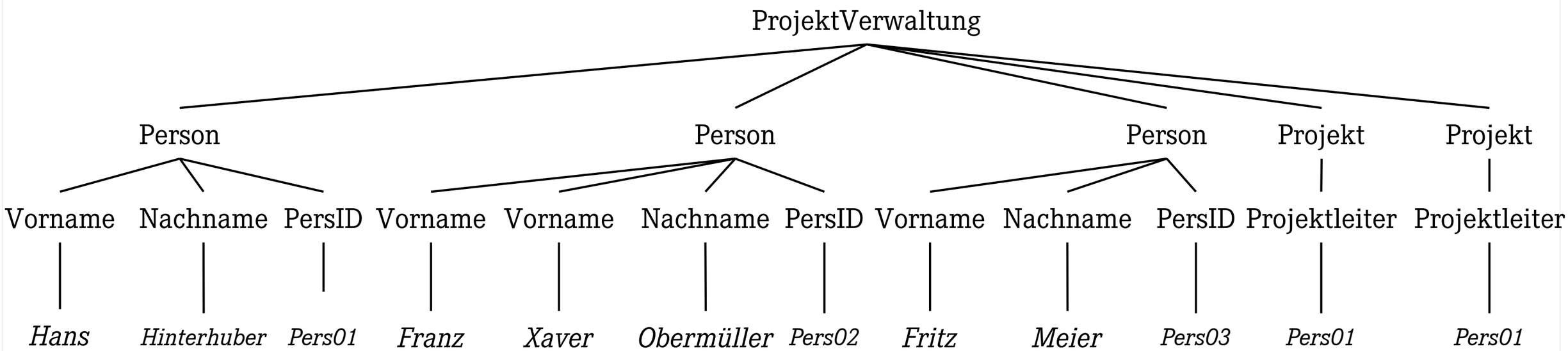
- FLWR-Auswertungsreihenfolge



Die Anfragesprache XQuery

- Die FLWR-Syntax im Beispiel

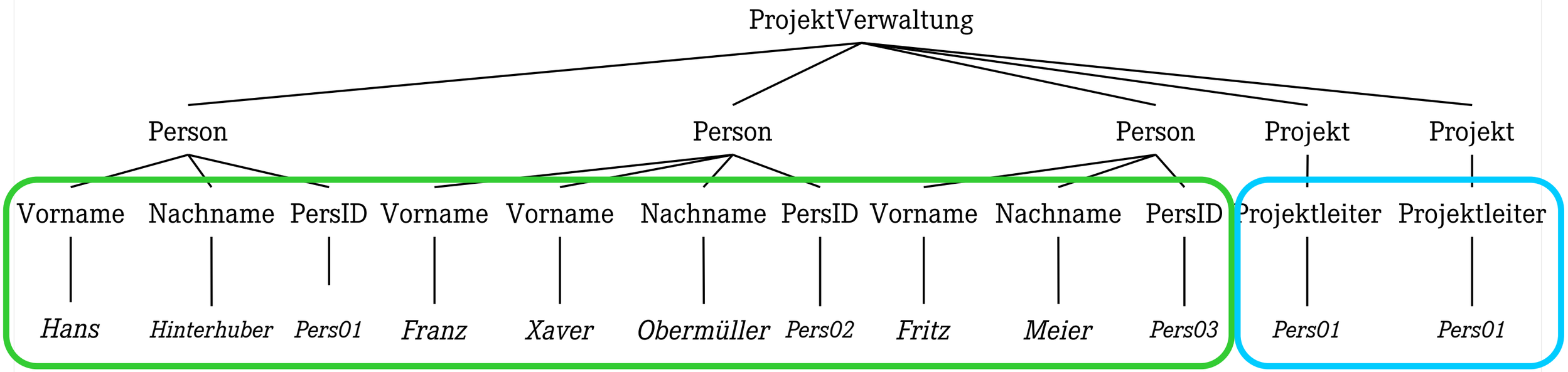
```
FOR $personen IN //Person
LET $projekte := //Projekt
WHERE $projekte/@Projektleiter = $personen/@PersID
RETURN $personen/Vorname
```



Die Anfragesprache XQuery

- Die FLWR-Syntax im Beispiel

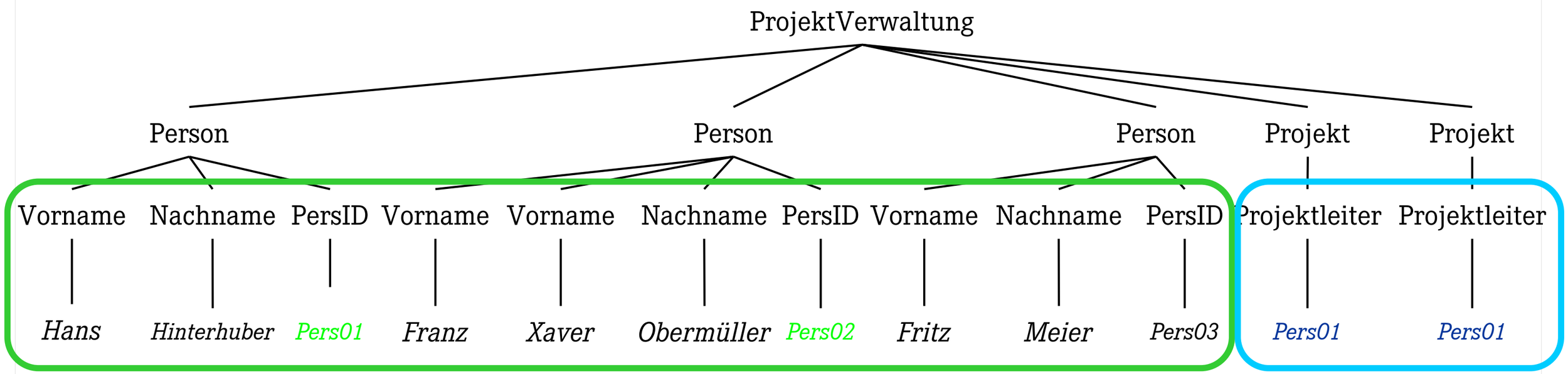
```
FOR $personen IN //Person
LET $projekte := //Projekt
WHERE $projekte/@Projektleiter = $personen/@PersID
RETURN $personen/Vorname
```



Die Anfragesprache XQuery

- Die FLWR-Syntax im Beispiel

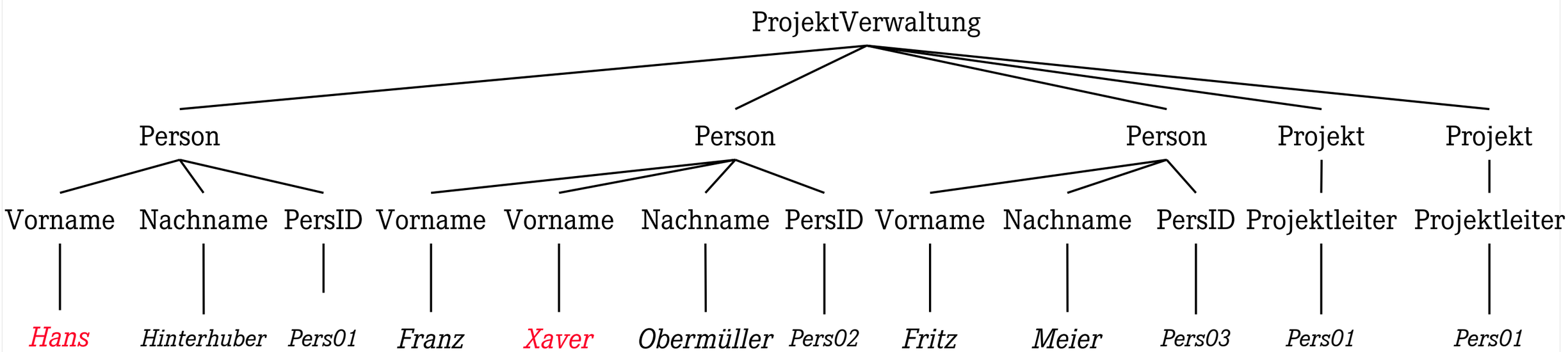
```
FOR $personen IN //Person
LET $projekte := //Projekt
WHERE $projekte/@Projektleiter = $personen/@PersID
RETURN $personen/Vorname
```



Die Anfragesprache XQuery

- Die FLWR-Syntax im Beispiel

```
FOR $personen IN //Person
LET $projekte := //Projekt
WHERE $projekte/@Projektleiter = $personen/@PersID
RETURN $personen/Vorname
```



Die Anfragesprache XQuery

- Zusammenfassung: *XQuery*
  - Eigenständige neue Anfragesprache auf XML-Dokumente
  - Syntax an SQL angelehnt
  - Deutlich leichter zu handhaben als XPath
  - Im allgemeinen effizienter als eigener Code
  - Zur Ausführungszeit extern automatisiert optimierbar
  - Erste Prototypen verfügbar

Gliederung

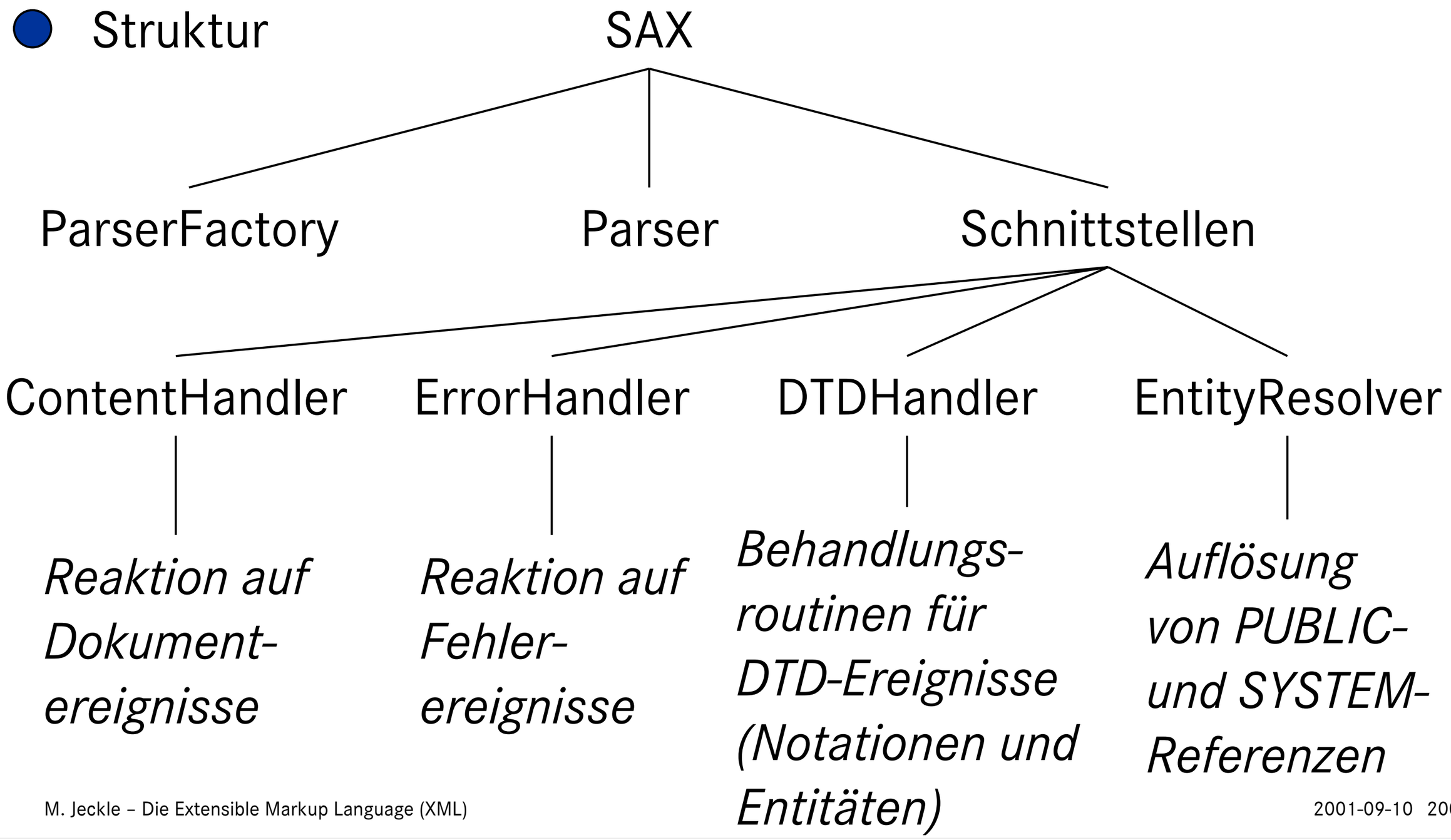
- Anwendungen der XML im praktischen Einsatz ...
- **Die Simple API for XML (SAX)**
  - Das Document Object Model (DOM)
  - Nahtlose Integration von XML und Hochsprache: XML Data Binding
  - Persistenz: XML und Datenbanken
  - XML-basierter Nachrichtenaustausch und entfernte Methodenaufrufe: XML Protokolle

Die Simple API for XML (SAX)

- Ziel: Einfache generische low-level Schnittstelle zur XML-Verarbeitung in Programmiersprachen
- Initiiert durch David Megginson
  Entwickelt durch die *xml-dev*-Mailingliste
- Leichtgewichtiger Ansatz (Speicherplatzbedarf, Laufzeit)
- Stellt keine konkrete Implementierung zur Verfügung, sondern Sammlung abstrakter Schnittstellen

Die Simple API for XML (SAX)

- Struktur

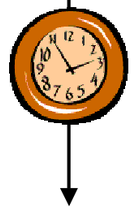


Die Simple API for XML (SAX)

- Nutzung der SAX-API

```
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;

public class SAXExample1 extends DefaultHandler
{
   public void startDocument()
   {
      System.out.println("document started");
   } //startDocument()

   public void endDocument()
   {
      System.out.println("document ended");
   } //endDocument()

   public static void main (String args[]) throws Exception
   {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      SAXParser sp = spf.newSAXParser();
      sp.parse( args[0], new SAXExample1() );
   } //main(args[])
}//class SAXExample1
```

```
<?xml version="1.0"?>
<root>
  ...
</root>
```

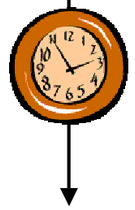


Die Simple API for XML (SAX)

- Nutzung der SAX-API

```
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;

public class SAXExample1 extends DefaultHandler
{
  public void startDocument()
  {
    System.out.println("document started");
  } //startDocument()

  public void endDocument()
  {
    System.out.println("document ended");
  } //endDocument()

  public static void main (String args[]) throws Exception
  {
    SAXParserFactory spf = SAXParserFactory.newInstance();
    SAXParser sp = spf.newSAXParser();
    sp.parse( args[0], new SAXExample1() );
  } //main(args[])
}//class SAXExample1
```

```
<?xml version="1.0"?>
<root>
  ...
</root>
```

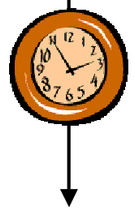


Die Simple API for XML (SAX)

- Nutzung der SAX-API

```
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;

public class SAXExample1 extends DefaultHandler
{
  public void startDocument()
  {
    System.out.println("document started");
  } //startDocument()

  public void endDocument()
  {
    System.out.println("document ended");
  } //endDocument()

  public static void main (String args[]) throws Exception
  {
    SAXParserFactory spf = SAXParserFactory.newInstance();
    SAXParser sp = spf.newSAXParser();
    sp.parse( args[0], new SAXExample1() );
  } //main(args[])
}//class SAXExample1
```

```
<?xml version="1.0"?>
<root>
  ...
</root>
```

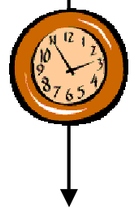


Die Simple API for XML (SAX)

- Nutzung der SAX-API

```
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;

public class SAXExample1 extends DefaultHandler
{
   public void startDocument()
   {
      System.out.println("document started");
   } //startDocument()

   public void endDocument()
   {
      System.out.println("document ended");
   } //endDocument()

   public static void main (String args[]) throws Exception
   {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      SAXParser sp = spf.newSAXParser();
      sp.parse( args[0], new SAXExample1() );
   } //main(args[])
}//class SAXExample1
```

```
<?xml version="1.0"?>
<root>
  ...
</root>
```



Die Simple API for XML (SAX)

- SAX-Ereignisse
  - startDocument()
  - processingInstruction(String target, String data)
  - startElement(String namespaceURI, String localName, String qName, Attributes atts)
  - characters(char[] ch, int start, int length)
  - ignorableWhitespace(char[] ch, int start, int length)
  - endElement(String namespaceURI, String localName, String qName)
  - startPrefixMapping(String prefix, String uri)
  - endPrefixMapping(String prefix)
  - endDocument()

Die Simple API for XML (SAX)

- Abschließendes Beispiel

```
<?xml version="1.0"?>
<?myPI this is a test?>
  <html
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:svg="http://www.w3.org/2000/svg">
  <head>
    <title>
      Testpage
    </title>
  </head>
  <body>
    <p>
      This is a
      <a href="http://www.jeckle.de">
        link
      </a>
    <svg:svg width="4cm" height="8cm">
      <svg:ellipse cx="2cm" cy="4cm" rx="2cm" ry="1cm"/>
    </svg:svg>
    </p>
  </body>
</html>
```

startDocument
processingInstruction
startElement
startPrefixMapping
startPrefixMapping
startElement
startElement
characters
endElement
endElement
startElement
startElement
characters
startElement
characters
endElement
startElement
startElement/endElement
endElement
endElement
endElement
endElement
endDocument

Die Simple API for XML (SAX)

- Einsatzgebiete und -empfehlungen
  - Auswertungen und einfache Transformationen auf Dokumente
  - Schnelle Verarbeitung
  - Kontextfreie Operationen
  - Umgebungen mit wenig verfügbarem Hauptspeicher

Die Simple API for XML (SAX)

- Zusammenfassung: *Simple API for Java*
  - Paradigmen- und ausführungsmodellunabhänige herstellerneutrale Schnittstellen-Sammlung
  - Lineare ereignisbasierte Verarbeitung von XML-Dokumenten
  - Nur lesender Zugriff, keine Modifikationen des Eingangs-dokuments möglich
  - Implementierungen für verschiedene Programmiersprachen verfügbar (z.B. Java, C/C++, Perl, Eiffel, SmallTalk)
  - Basis der Konstruktion höherer Schnittstellen (z.B. DOM)

Gliederung

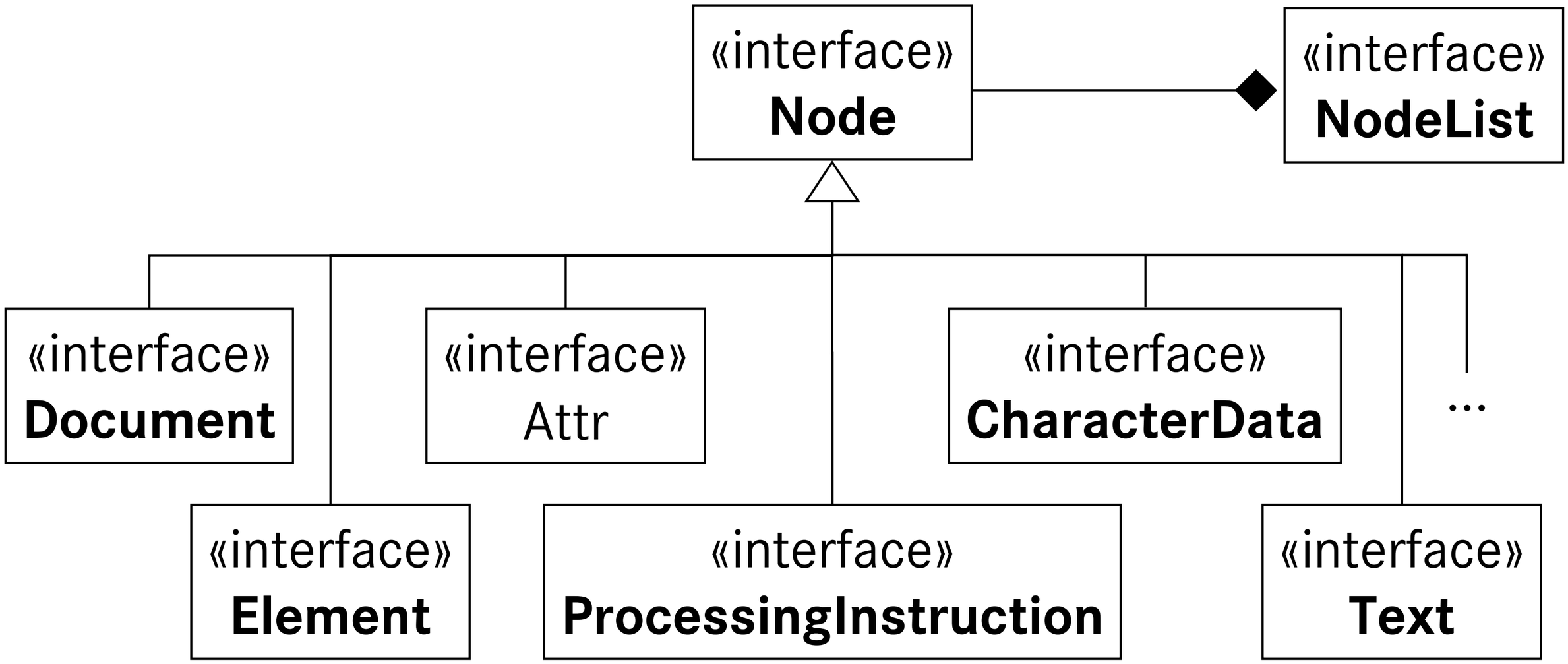
- Anwendungen der XML im praktischen Einsatz ...
  - Die Simple API for XML (SAX)
  - **Das Document Object Model (DOM)**
  - Nahtlose Integration von XML und Hochsprache: XML Data Binding
  - Persistenz: XML und Datenbanken
  - XML-basierter Nachrichtenaustausch und entfernte Methodenaufrufe: XML Protokolle

Das Document Object Model (DOM)

- Programmiersprachenunabhängige generische Speicherstruktur für XML-Dokumente
- W3C-Standard
- Ursprünglich für HTML entwickelt und von Web-Browsern umgesetzt
- Implementierungen in verschiedenen Sprachen verfügbar
- Bietet lesenden und schreibenden Zugriff, d.h. Modifikationen am Eingangsdokument möglich

Das Document Object Model (DOM)

- Das Objektmodell



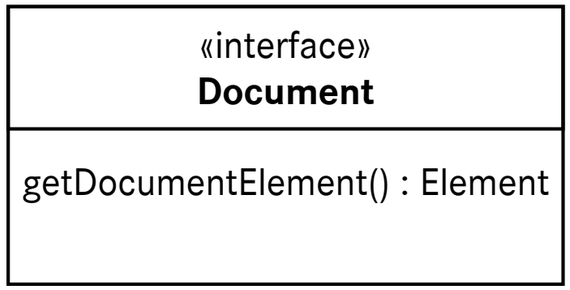
Das Document Object Model (DOM)

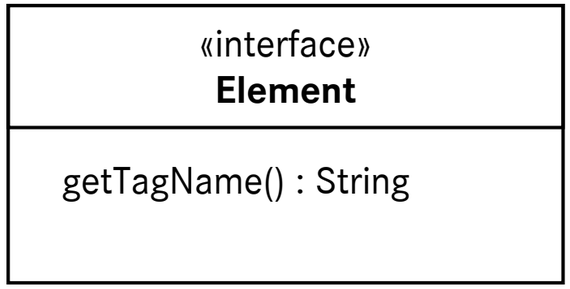
- Beispiel
  (lesender Zugriff)

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample2
{
  public static void main(String[] args) throws Exception
  {
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document document = builder.parse( args[0] );

    Element theRootElement = document.getDocumentElement();

    System.out.println("root element's name: "+ theRootElement.getTagName());

    System.out.print("the element has");
    if (!theRootElement.hasAttributes())
      System.out.print(" no");
    System.out.println(" attributes");
  } //main()
} //class DOMExample2
```

Das Document Object Model (DOM)

- Beispiel
  (lesender Zugriff)

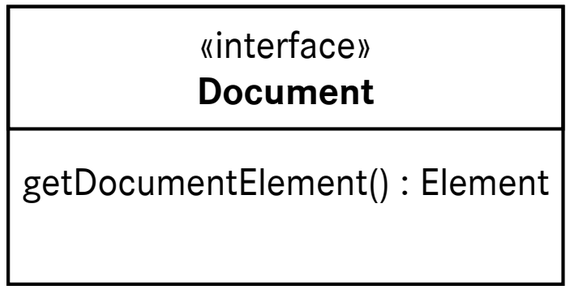


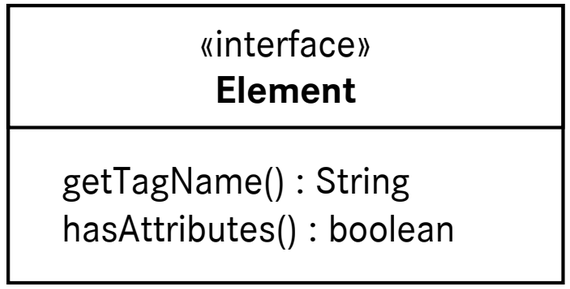



```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample2
{
  public static void main(String[] args) throws Exception
  {
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document document = builder.parse( args[0] );

    Element theRootElement = document.getDocumentElement();

    System.out.println("root element's name: "+ theRootElement.getTagName());

    System.out.print("the element has");
    if (!theRootElement.hasAttributes())
      System.out.print(" no");
    System.out.println(" attributes");
  } //main()
} //class DOMExample2
```

Das Document Object Model (DOM)

- Beispiel
  (lesender Zugriff)






```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample2
{
  public static void main(String[] args) throws Exception
  {
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document document = builder.parse( args[0] );

    Element theRootElement = document.getDocumentElement();

    System.out.println("root element's name: "+ theRootElement.getTagName());

    System.out.print("the element has");
    if (!theRootElement.hasAttributes())
      System.out.print(" no");
    System.out.println(" attributes");
  } //main()
} //class DOMExample2
```

Das Document Object Model (DOM)

- Beispiel
  (Modifikation am Dokument)

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample3
{
   public static void main(String[] args) throws Exception
   {


      ...

      Element theRootElement = document.getDocumentElement();

      theRootElement.setAttribute("myFirstNewAttribute","01");

      Element aNewElement = document.createElement("myNewElement");

      theRootElement.appendChild( aNewElement );

      System.out.print( theRootElement );
   } //main()
} //class DOMExample3
```

Das Document Object Model (DOM)

- Beispiel
  (Modifikation am Dokument)

«interface» **Document**
getDocumentElement() : Element

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample3
{
   public static void main(String[] args) throws Exception
   {

      ...

      Element theRootElement = document.getDocumentElement();

      theRootElement.setAttribute("myFirstNewAttribute","01");

      Element aNewElement = document.createElement("myNewElement");

      theRootElement.appendChild( aNewElement );

      System.out.print( theRootElement );
   } //main()
} //class DOMExample3
```

Das Document Object Model (DOM)

- Beispiel
  (Modifikation am Dokument)

«interface» **Document**
getDocumentElement() : Element

«interface» **Element**
setAttribute(DOMString n, DOMString v)

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample3
{
   public static void main(String[] args) throws Exception
   {

      ...

      Element theRootElement = document.getDocumentElement();

      theRootElement.setAttribute("myFirstNewAttribute","01");

      Element aNewElement = document.createElement("myNewElement");

      theRootElement.appendChild( aNewElement );

      System.out.print( theRootElement );
   } //main()
} //class DOMExample3
```

Das Document Object Model (DOM)

- Beispiel
  (Modifikation am Dokument)

«interface» **Document**
getDocumentElement() : Element createElement(DOMString name) : Element

«interface» **Element**
setAttribute(DOMString n, DOMString v)

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample3
{
   public static void main(String[] args) throws Exception
   {

      ...

      Element theRootElement = document.getDocumentElement();

      theRootElement.setAttribute("myFirstNewAttribute","01");

      Element aNewElement = document.createElement("myNewElement");

      theRootElement.appendChild( aNewElement );

      System.out.print( theRootElement );
   } //main()
} //class DOMExample3
```

Das Document Object Model (DOM)

- Beispiel
  (Modifikation am Dokument)

«interface» **Document**
getDocumentElement() : Element createElement(DOMString name) : Element

«interface» **Element**
setAttribute(DOMString n, DOMString v) appendChild(Element newEl)

```
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMExample3
{
   public static void main(String[] args) throws Exception
   {

      ...

      Element theRootElement = document.getDocumentElement();

      theRootElement.setAttribute("myFirstNewAttribute","01");

      Element aNewElement = document.createElement("myNewElement");

      theRootElement.appendChild( aNewElement );

      System.out.print( theRootElement );
   } //main()
} //class DOMExample3
```

Das Document Object Model (DOM)

- Zusammenfassung: *Document Object Model (DOM)*
  - Generisches programmiersprachenunabhängiges Speichermodell
  - Angelehnt an InfoSet
  - Möglichkeit zur Traversierung und Modifkation von XML-Dokumenten
  - Eigenes (programmiersprachenneutrales) Typsystem
  - Implementierungen für viele kommerzielle Programmiersprachen verfügbar

Gliederung

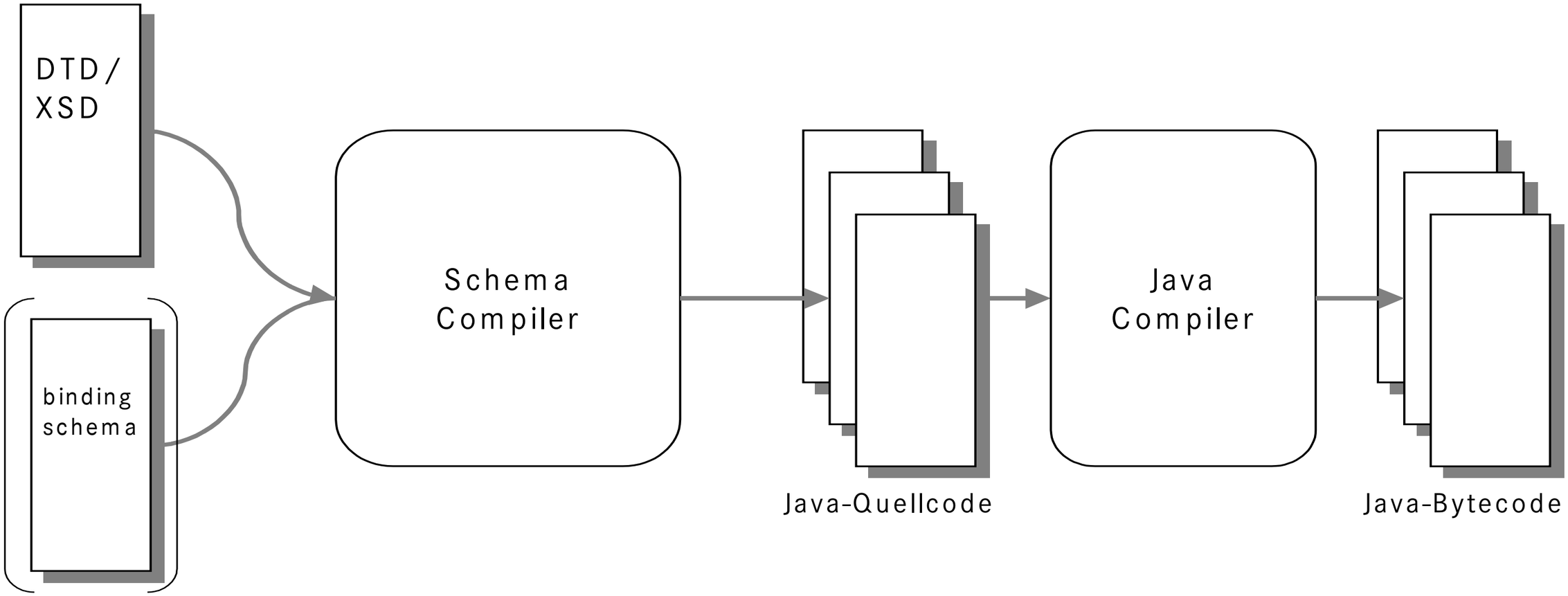
- Anwendungen der XML im praktischen Einsatz ...
  - Die Simple API for XML (SAX)
  - Das Document Object Model (DOM)
  - **Nahtlose Integration von XML und Hochsprache: XML Data Binding**
  - Persistenz: XML und Datenbanken
  - XML-basierter Nachrichtenaustausch und entfernte Methodenaufrufe: XML Protokolle

Nahtlose Integration von XML und Hochsprache

- Verschiedene (bisher nicht standardisierte) Ansätze am Markt verfügbar
- Gemeinsame Grundidee: Erzeugung von Datenstrukturen aus XML-Strukturen.
  Zumeist: Java-Quellecode aus DTDs oder XSD-Dokumenten
- Erhoffte Vorteile:
  - Problemnäher als DOM
  - Nicht signifikant schlechteres Laufzeitverhalten als SAX
  - XML transparent als *natürliches* Serialisierungsformat für Objekte

Nahtlose Integration von XML und Hochsprache

- Ansatz: JAXB – Java Architecture for XML Binding
  (ehemals: Project Adelade)



Nahtlose Integration von XML und Hochsprache

- Beispiel

```
<!ELEMENT tree (node+)>
<!ELEMENT node (node*)>
<!ATTLIST node  information CDATA  #IMPLIED >
```

Tree

- Node : List
- pred_node : Predicate

+ getNode() :List
+ deleteNode()
+ emptyNode()

+ validateThis()
+ validate(v : Validator)
+ marshal(m : Marshaller)
+ unmarshal(u : Unmarshaller)
+ unmarshal(i : InputStream) : Tree
+ unmarshal(x : XMLScanner) : Tree
+ unmarshal(x : XMLScanner, d : Dispatcher) : Tree
+ equals(o : Object) : boolean
+ hashCode() : int
+ toString() : String
+ newDispatcher() : Dispatcher

Node

- _Information : String
- _Node : List
- pred_Node : Predicate

+ getInformation() : String
+ setInformation(i : String)
+ getNode() : List
+ deleteNode()
+ emptyNode()

+ validateThis()
+ validate(v : Validator)
+ marshal(m : Marshaller)
+ unmarshal(u : Unmarshaller)
+ unmarshal(i : InputStream) : Tree
+ unmarshal(x : XMLScanner) : Tree
+ unmarshal(x : XMLScanner, d : Dispatcher) : Tree
+ equals(o : Object) : boolean
+ hashCode() : int
+ toString() : String
+ newDispatcher() : Dispatcher

Nahtlose Integration von XML und Hochsprache

```
import java.io.*;
import java.util.List;

public class JAXBExample1
{
   public static void main(String args[]) throws Exception
   {
      Tree myTree = new Tree();

      List nodes = myTree.getNode();

      Node myNode = new Node();
      myNode.setInformation("42");

      nodes.add(myNode);

      myTree.validate();
      FileOutputStream treeFile = new FileOutputStream( new File("someTree.xml") );
      myTree.marshal(treeFile);
   } // main()
} //class JAXBExample1
```

Nahtlose Integration von XML und Hochsprache

- Zusammenfassung: *Java Architecture for XML Binding*
  - Aktuell verfügbare Beta setzt noch auf DTDs auf
  - Erzeugung von *spezialisierten* Klassen (je DTD-ELEMENT eine)
  - Transparentes Lesen und Schreiben der XML-Dokumente durch automatisch generierte und implementierte unmarshal- bzw. marshal-Methode
  - Navigation und Traversierung durch vordefinierte Methoden
  - Auch als Ausgangspunkt von XML-unabhängiger Implementierung verwendbar

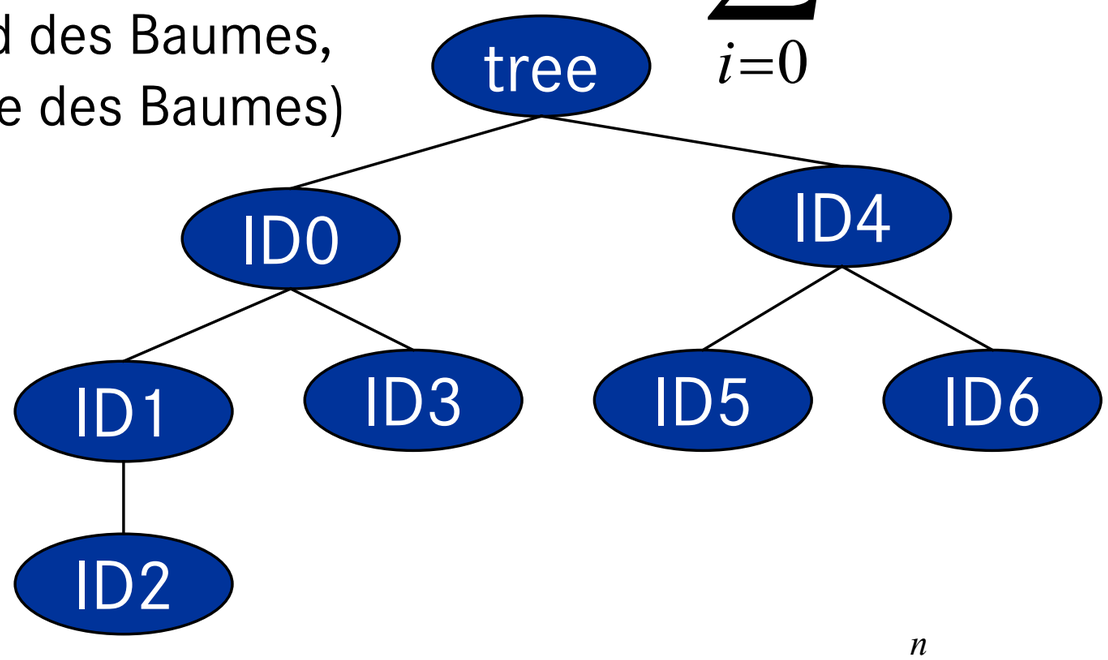
Vergleich von XML-Schnittstellen

- Testmethodik: Verwendete Dokumente

```
<?xml version="1.0" encoding="utf-8"?>
<tree>
  <node>
    <nodeInfo>ID0</nodeInfo>
    <node>
      <nodeInfo>ID1</nodeInfo>
      <node>
        <nodeInfo>ID2</nodeInfo>
      </node>
      <node>
        <nodeInfo>ID3</nodeInfo>
      </node>
    </node>
    <node>
      <nodeInfo>ID4</nodeInfo>
      <node>
        <nodeInfo>ID5</nodeInfo>
      </node>
      <node>
        <nodeInfo>ID6</nodeInfo>
      </node>
    </node>
  </node>
</tree>
```

Anzahl Baumelemente: $\sum_{i=0}^{n} i^m$

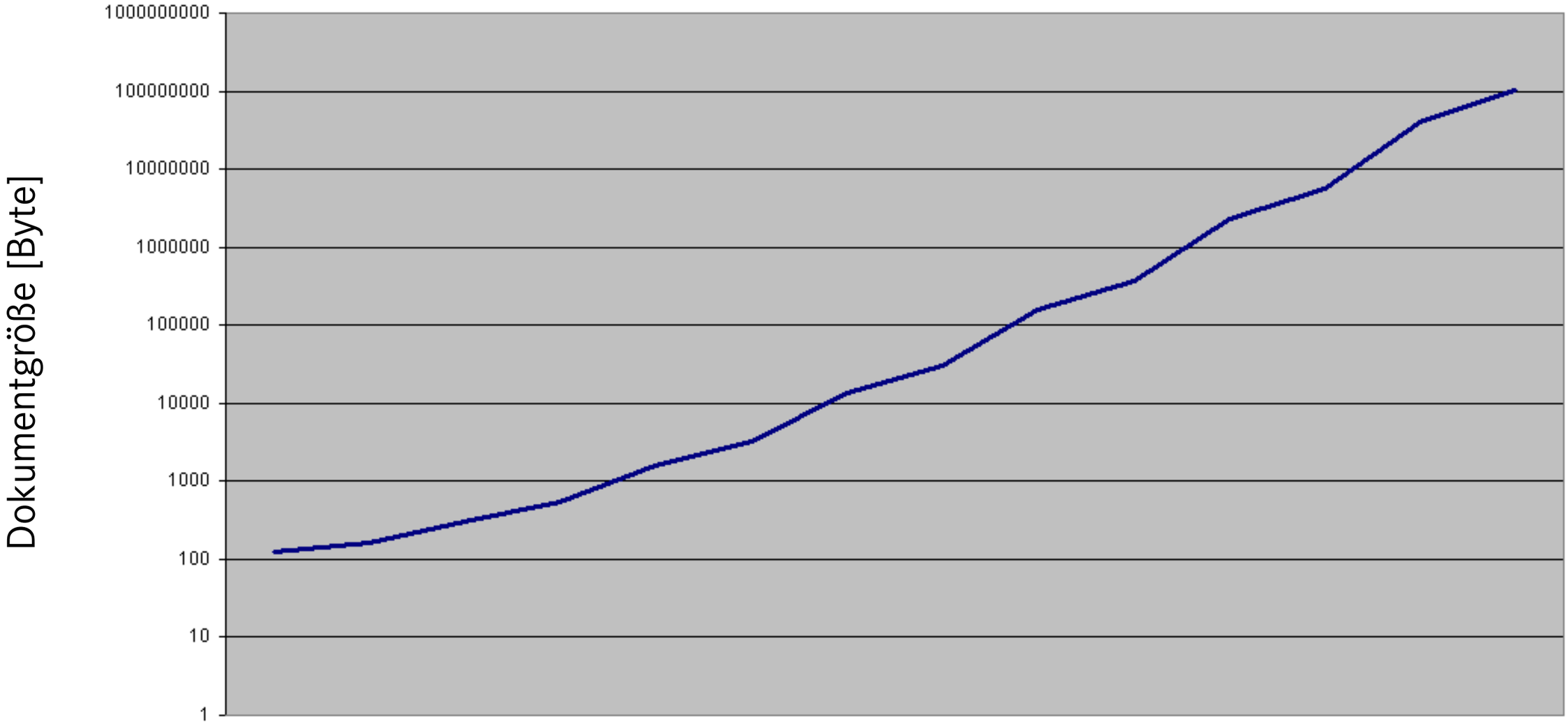
(*n* Grad des Baumes,
m Höhe des Baumes)



Dokumentgröße: $51 + 36\sum_{i=0}^{n} i^m + \sum_{j=1}^{\sum_{i=0}^{n} i^m} \lfloor \log(j+1) \rfloor$

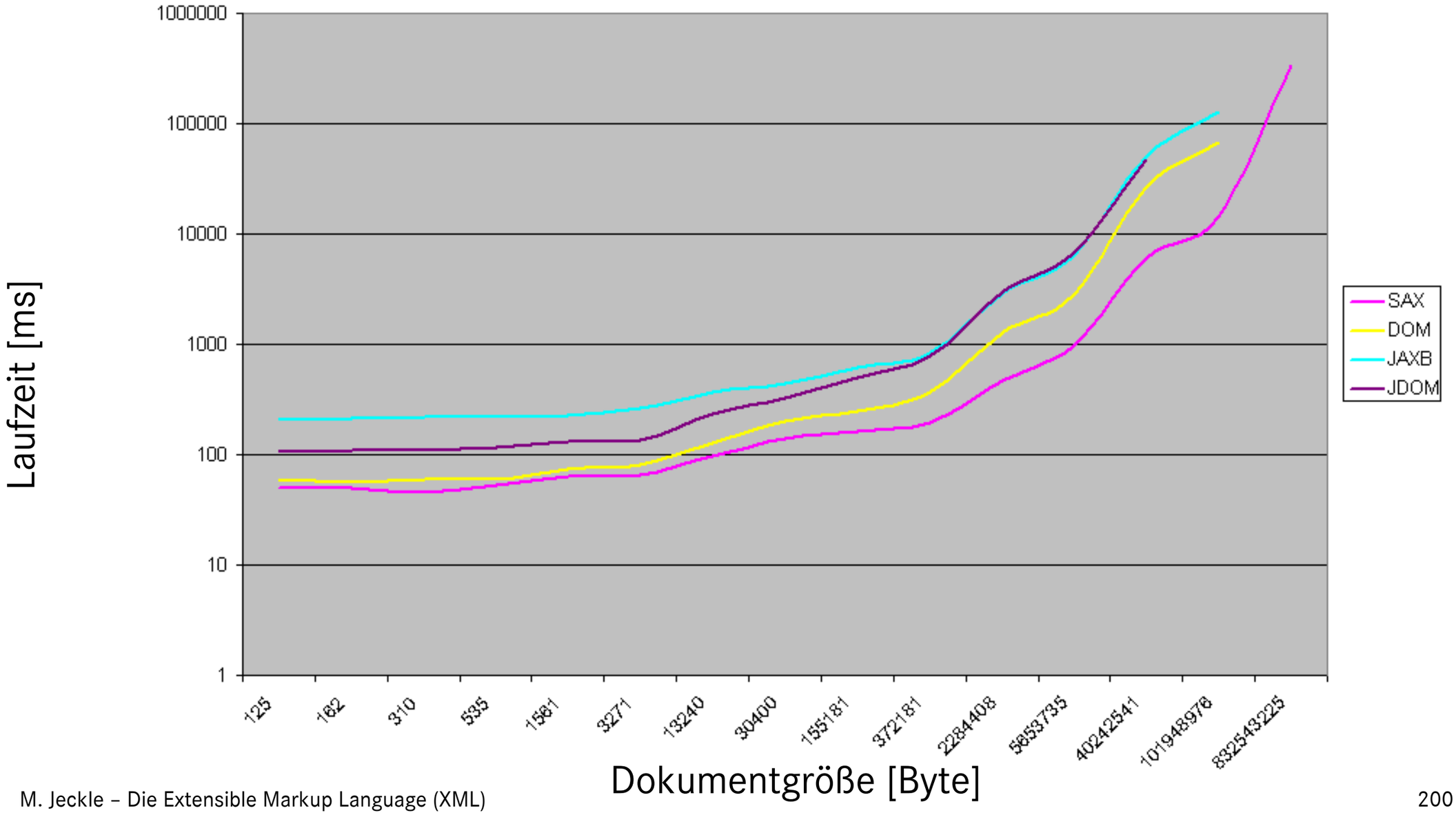
Vergleich von XML-Schnittstellen

- Testmethodik: Verwendete Dokumentgrößen



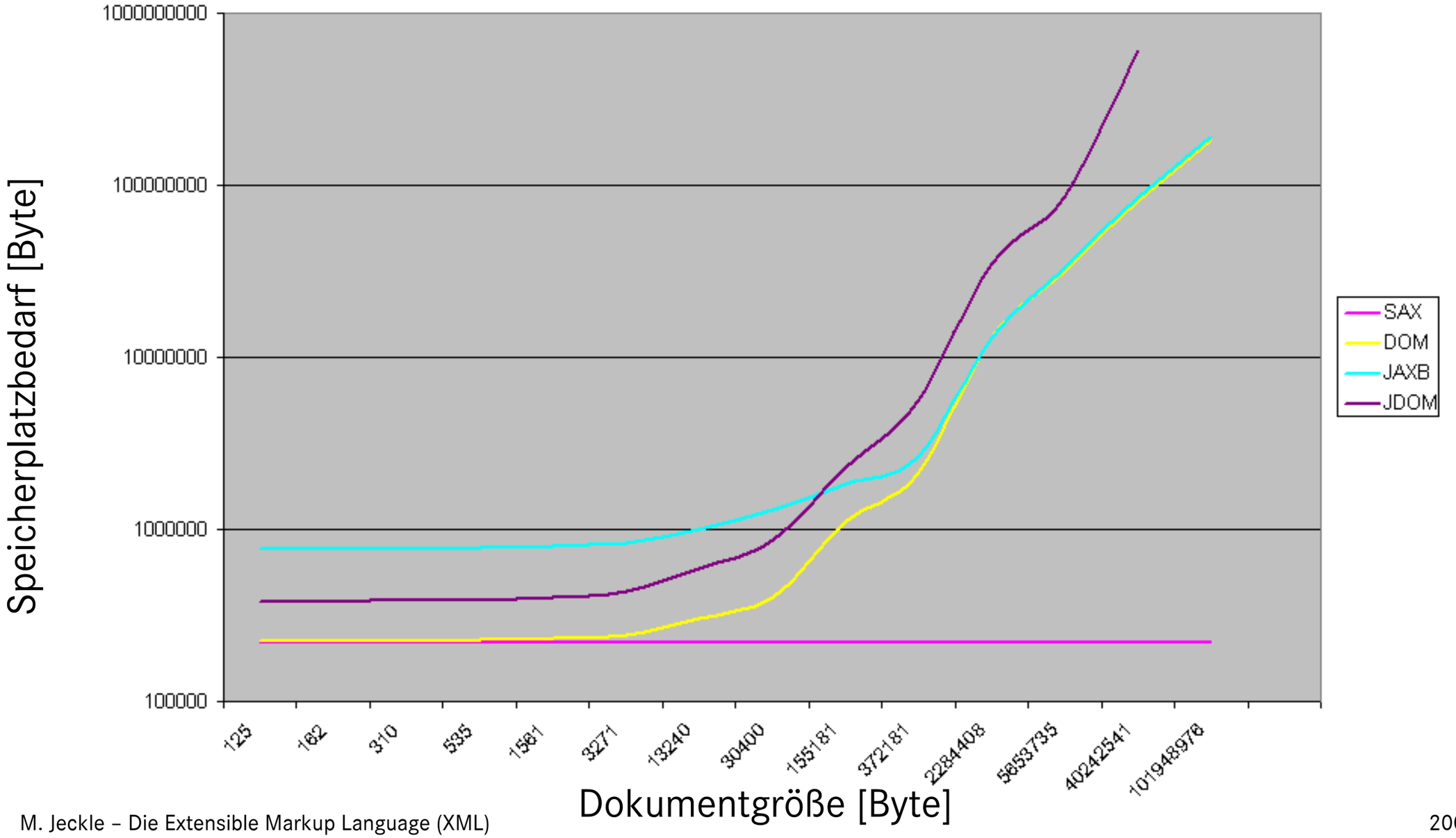
Vergleich von XML-Schnittstellen

- Parser-Laufzeiten



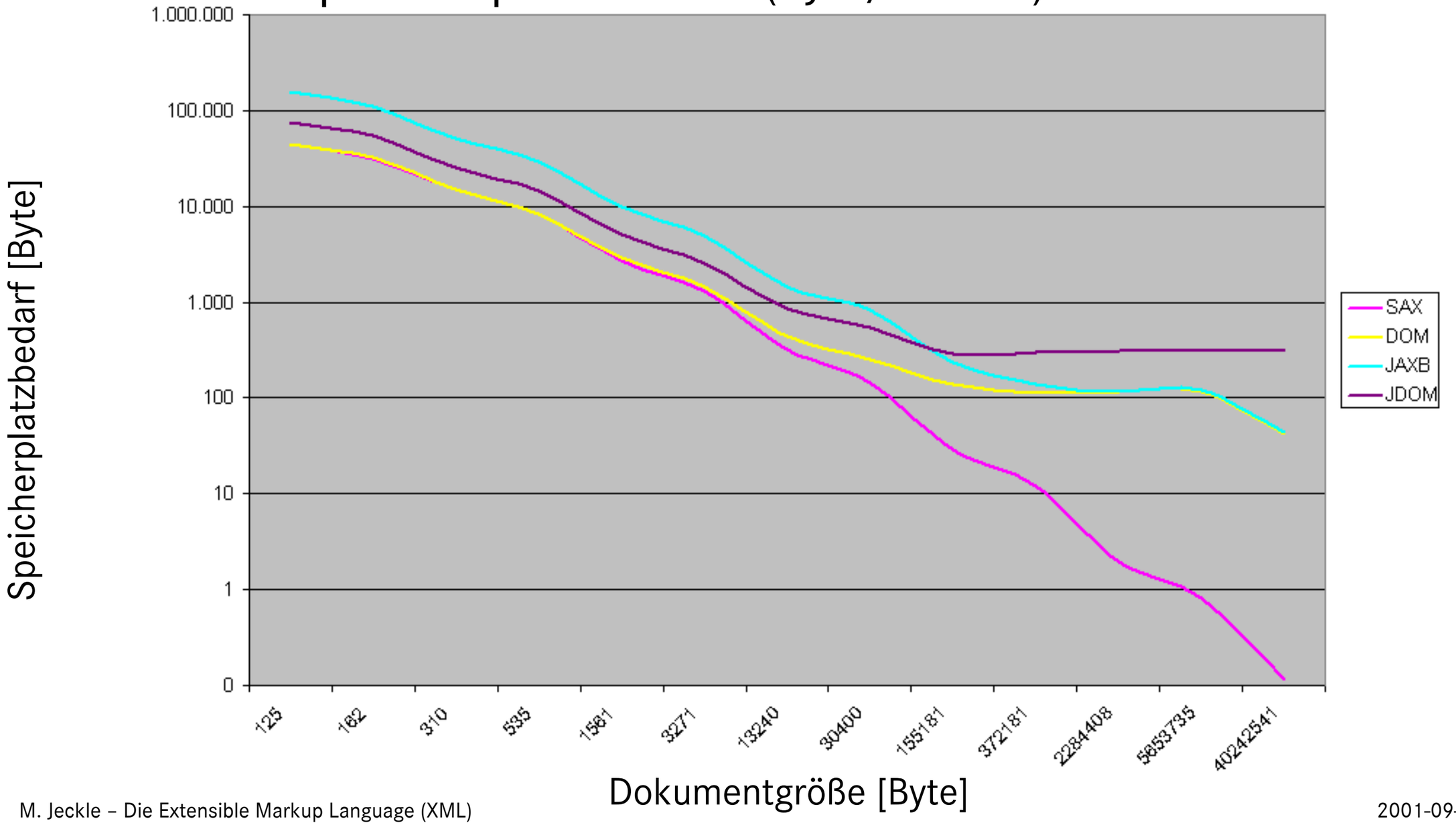
Vergleich von XML-Schnittstellen

- Parser-Speicherplatzbedarf



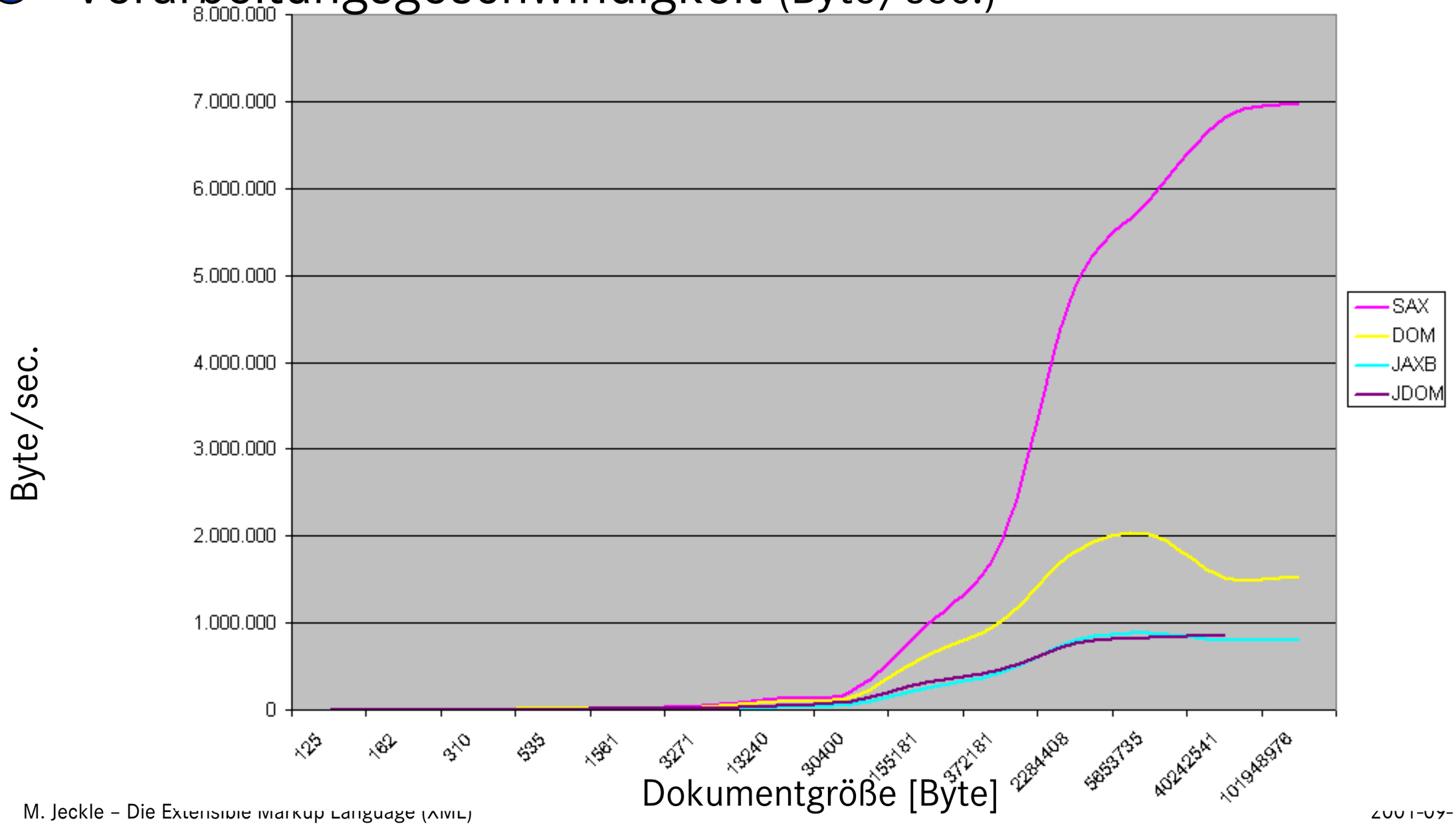
Vergleich von XML-Schnittstellen

- Relativer Speicherplatzbedarf (Byte/Knoten)



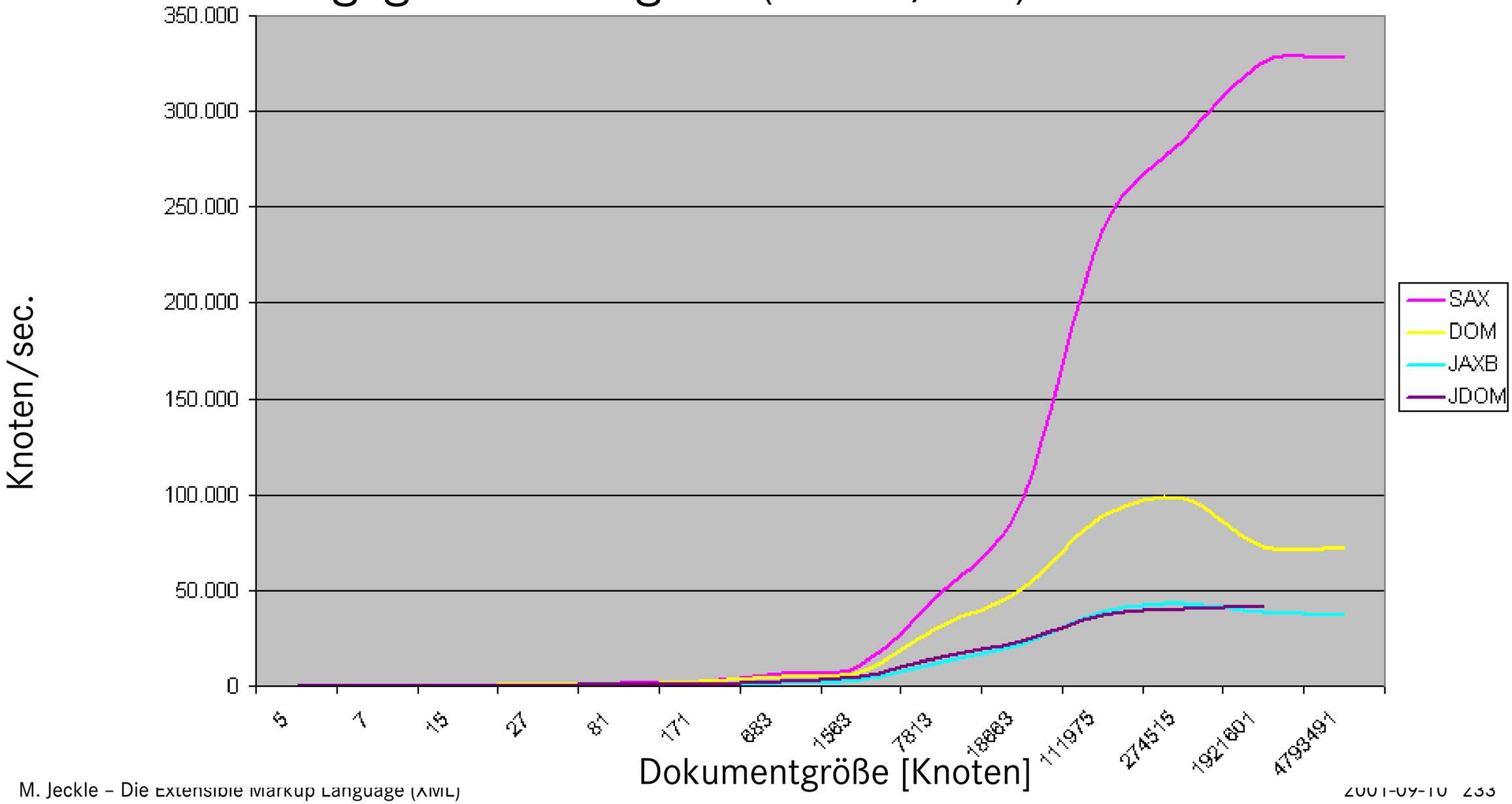
Vergleich von XML-Schnittstellen

- Verarbeitungsgeschwindigkeit (Byte/sec.)



Vergleich von XML-Schnittstellen

- Verarbeitungsgeschwindigkeit (Knoten/sec.)



Gliederung

- Anwendungen der XML im praktischen Einsatz ...
  - Die Simple API for XML (SAX)
  - Das Document Object Model (DOM)
  - Nahtlose Integration von XML und Hochsprache: XML Data Binding
  - **Persistenz: XML und Datenbanken**
  - XML-basierter Nachrichtenaustausch und entfernte Methodenaufrufe: XML Protokolle

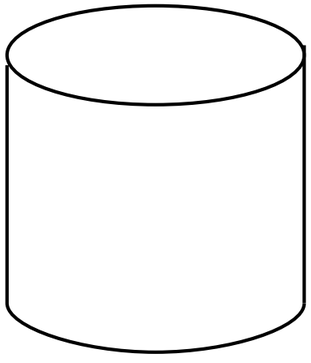
Persistenz: XML und Datenbanken

- XML scheint strukturinhärent eher eine Baum-artige denn eine mengenorientierte Speicherung zu bedingen
- Geschwindigkeitsgewinn beim Zugriff durch Indexierung
- Einschränkende Sichten auf XML-Daten
- Aggregation und Integration
- Erweiterung post-relationaler Datenbankmanagementsysteme
- Erweiterung semi-strukturierter DBMS
- „Native“ XML-DBMS

Persistenz: XML und Datenbanken

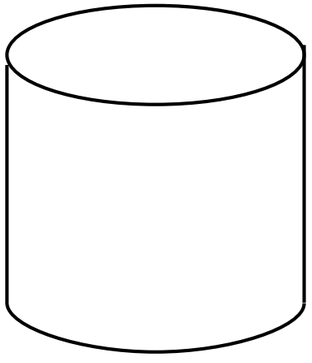
```
<?xml version="1.0" encoding="UTF-8"?>
<Vortrag>
  <Titel>Die Extensible Markup Language</Titel>
  <Veranstaltung datum="2001-07-13">
    <Name>EADS Hamburg</Name>
  </Veranstaltung>
  <Referent>
    <Name>Mario Jeckle</Name>
    <Firma>DaimlerChrysler Research and Technology</Firma>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Referent>
</Vortrag>
```

elektronifizierte (Business-)Dokumente



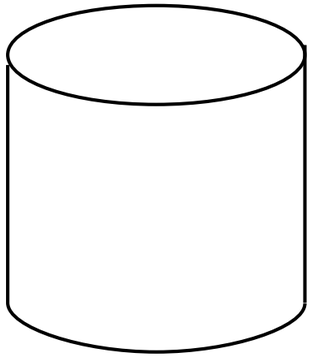
- Bestellungen
- Aufträge
- ...

Elektronische Dokumente



- Nachrichten
- Katalogdaten
- ...

Daten

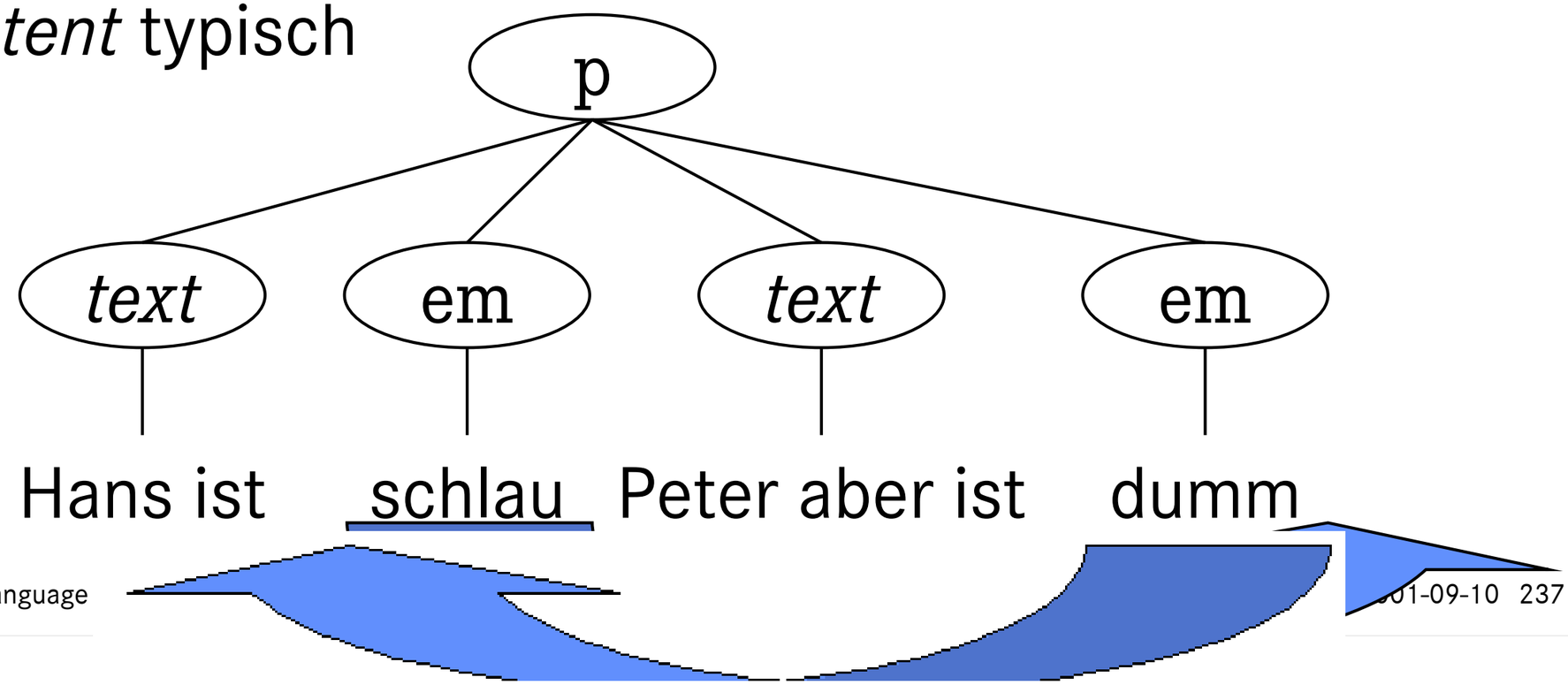


- Produktionsdaten
- Meßwerte
- ...

Persistenz: XML und Datenbanken

- Dokumente
  - unterschiedlich strukturiert
  - Reihenfolge signifikant
  - Volltextsuche sinnvoll
  - *mixed content* typisch

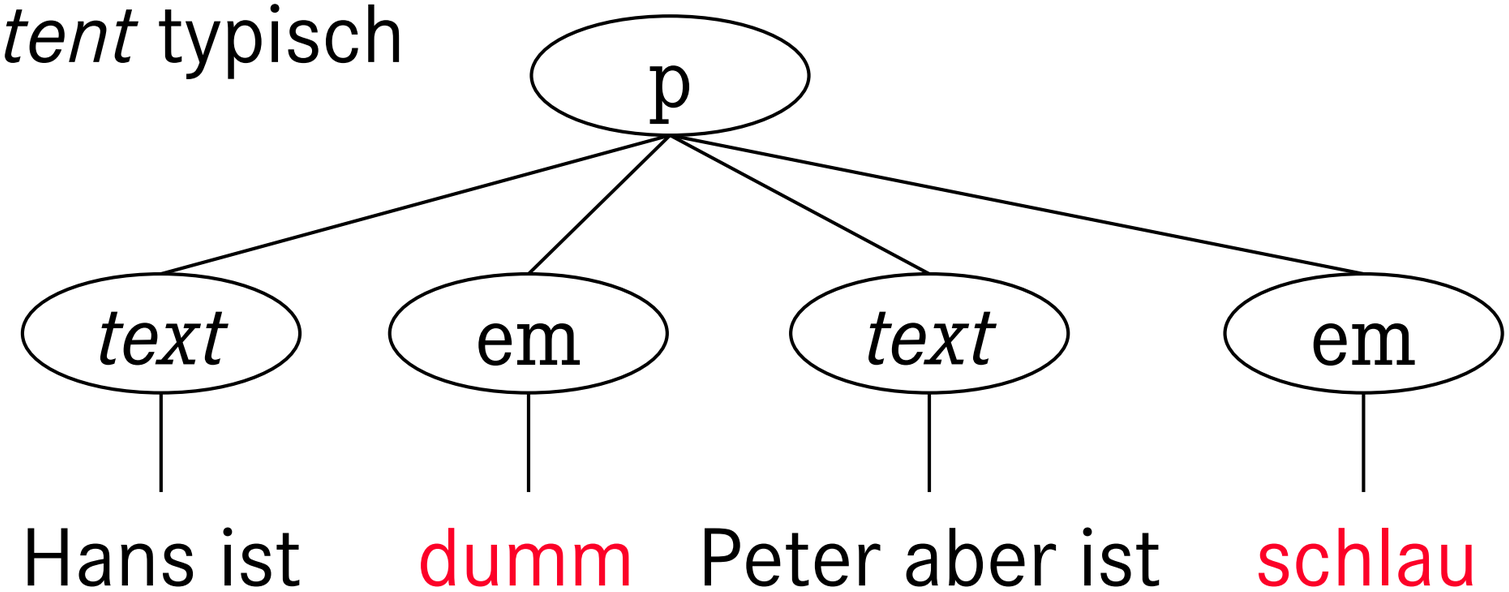
```
<p>Hans ist <em>schlau</em>,
Peter aber ist <em>dumm</em></p>
```



Persistenz: XML und Datenbanken

- Dokumente
  - unterschiedlich strukturiert
  - Reihenfolge signifikant
  - Volltextsuche sinnvoll
  - *mixed content* typisch

```
<p>Hans ist <em>schlau</em>,
Peter aber ist <em>dumm</em></p>
```



Persistenz: XML und Datenbanken

- Dokumente
  - unterschiedlich strukturiert
  - Reihenfolge signifikant
  - Volltextsuche sinnvoll
  - *mixed content* typisch
- Daten
  - zumeist keine feste Ordnung
  - einheitliche klar definierte Strukturierung
  - sind typisiert

Persistenz: XML und Datenbanken

- Abbildung von XML auf relationale DB-Strukturen
  - XML-Dokument ist Inhalt eines Feldes
  - Aufspaltung des XML-Dokuments auf mehrere Tabellen
  - Uninterpretiert als BLOB oder CLOB
  - Unterstützung durch DB-interne Funktionalität

Persistenz: XML und Datenbanken

Emp

empNo	ename	job
7369	Smith	clerk
7499	Allen	salesman
7566	Jones	manager
7654	Martin	salesman
7698	Blake	manager
7782	Clark	manager

```
<Emp>
  <employee>
    <empNo>7369</empNo>
    <ename>Smith</ename>
    <job>clerk</job>
  </employee>
  <employee>
    <empNo>7499</empNo>
    <ename>Allen</ename>
    <job>salesman</job>
  </employee>
  <emplyoee>
    <empNo>7566</empNo>
...
```

Persistenz: XML und Datenbanken

Emp

empNo	ename	job
7369	Smith	clerk
7499	Allen	salesman
7566	Jones	manager
7654	Martin	salesman
7698	Blake	manager
7782	Clark	manager

```
<Emp>
  <employee>
    <empNo>7369</empNo>
    <ename>Smith</ename>
    <job>clerk</job>
  </employee>
  <employee>
    <empNo>7499</empNo>
    <ename>Allen</ename>
    <job>salesman</job>
  </employee>
  <emplyoee>
    <empNo>7566</empNo>
...
```

Persistenz: XML und Datenbanken

Emp

empNo	ename	job
7369	Smith	clerk
7499	Allen	salesman
7566	Jones	manager
7654	Martin	salesman
7698	Blake	manager
7782	Clark	manager

```
<Emp>
  <employee>
    <empNo>7369</empNo>
    <ename>Smith</ename>
    <job>clerk</job>
  </employee>
  <employee>
    <empNo>7499</empNo>
    <ename>Allen</ename>
    <job>salesman</job>
  </employee>
  <emplyoee>
    <empNo>7566</empNo>
...
```

Persistenz: XML und Datenbanken

Emp

empNo	ename	job
7369	Smith	clerk
7499	Allen	salesman
7566	Jones	manager
7654	Martin	salesman
7698	Blake	manager
7782	Clark	manager

```
<Emp>
  <employee>
    <empNo>7369</empNo>
    <ename>Smith</ename>
    <job>clerk</job>
  </employee>
  <employee>
    <empNo>7499</empNo>
    <ename>Allen</ename>
    <job>salesman</job>
  </employee>
  <emplyoee>
    <empNo>7566</empNo>
...
```

Persistenz: XML und Datenbanken

Emp

empNo	ename	job
7369	Smith	clerk
7499	Allen	salesman
7566	Jones	manager
7654	Martin	salesman
7698	Blake	manager
7782	Clark	manager

```
<Emp>
  <employee>
    <empNo>7369</empNo>
    <ename>Smith</ename>
    <job>clerk</job>
  </employee>
  <employee>
    <empNo>7499</empNo>
    <ename>Allen</ename>
    <job>salesman</job>
  </employee>
  <emplyoee>
    <empNo>7566</empNo>
...
```

Persistenz: XML und Datenbanken

- Konvertierung jeder Tabelle (SQL-Anfrageergebnis) nach XML ist vergleichsweise einfach
- Kann außerhalb des Datenbankkerns geschehen
- Transformation von XML-Eingaben in relationale Strukturen mit unter aufwendig
  - nicht immer allgemein lösbar (evtl. Definition von Regeln notwendig)
  - relationale DB-Struktur muß explizit manuell erzeugt werden
  - XML-Daten müssen Struktur der Tabelle entsprechen
  - nicht alle XML-Konstruke intuitiv abbildbar
  - Überführung Baumstruktur in mengenorientierte Sicht

Persistenz: XML und Datenbanken

- Gesamtes XML-Dokument wird in einer Spalte einer Tabelle oder als externe Datei abgelegt

```
create table myTable (
   empNo NUMBER(4) NOT NULL,
   ename VARCHAR(10) NOT NULL,
   job VARCHAR(10) NOT NULL,
   cv BLOB,
   ...)
```

- Zugriff durch geeignete interpretierende Module
- Zugriffsoptimierung nicht durch DBMS gewährleistet

Persistenz: XML und Datenbanken

- „Native“ Speicherung der XML-Daten
- XML-Spezifische Anfragesprachen (z.B. QUILT, XQuery, XPath)
- Ausgabeformat: XML
- Struktur- und wertebasierte Anfragen
- Unterscheidung zwischen Daten- und Dokumentsicht
- Erlaubt und berücksichtigt schematische Beschreibung der Daten

Persistenz: XML und Datenbanken

- Zusammenfassung: *XML und Datenbanken*
  - Klassische (R)DBMS-Hersteller erweitern ihre Produkte
  - „Native“XML-DBMS entstehen
  - „Native“ Speicherungsform für dokumentenorientierte Zugriffe zumeist performanter und besser geeigner
  - Klassische (R)DBMS besitzen (noch) Entwicklungsvorsprung hinsichtlich Infrastrukturdiensten
    (z.B. Backup/Recovery, Föderierbarkeit, Skalierbarkeit)

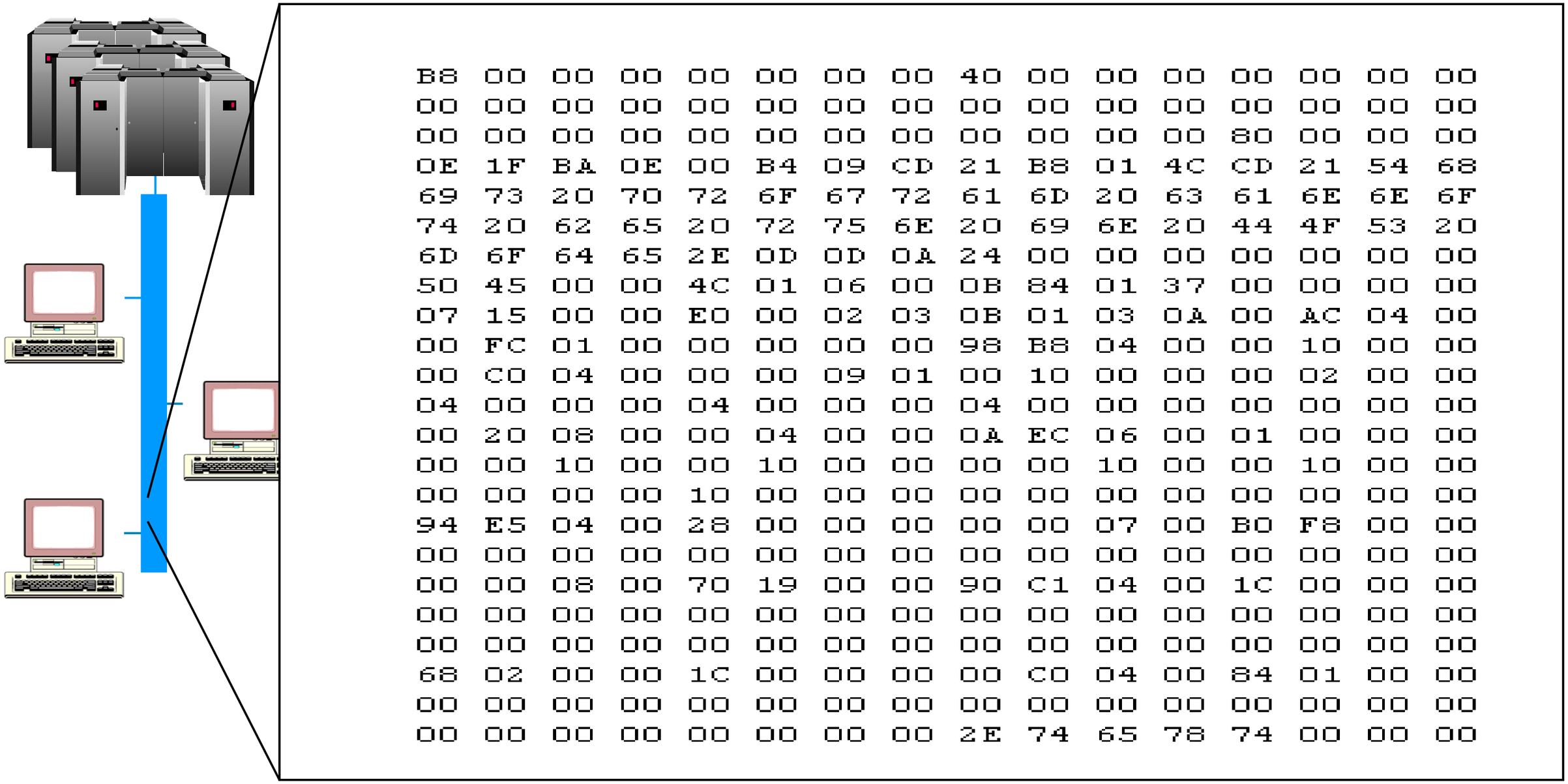
Gliederung

- Anwendungen der XML im praktischen Einsatz ...
  - Die Simple API for XML (SAX)
  - Das Document Object Model (DOM)
  - Nahtlose Integration von XML und Hochsprache: XML Data Binding
  - Persistenz: XML und Datenbanken
  - **XML-basierter Nachrichtenaustausch und entfernte Methodenaufrufe: XML Protokolle**

XML-basierter Nachrichtenaustausch und RPCs

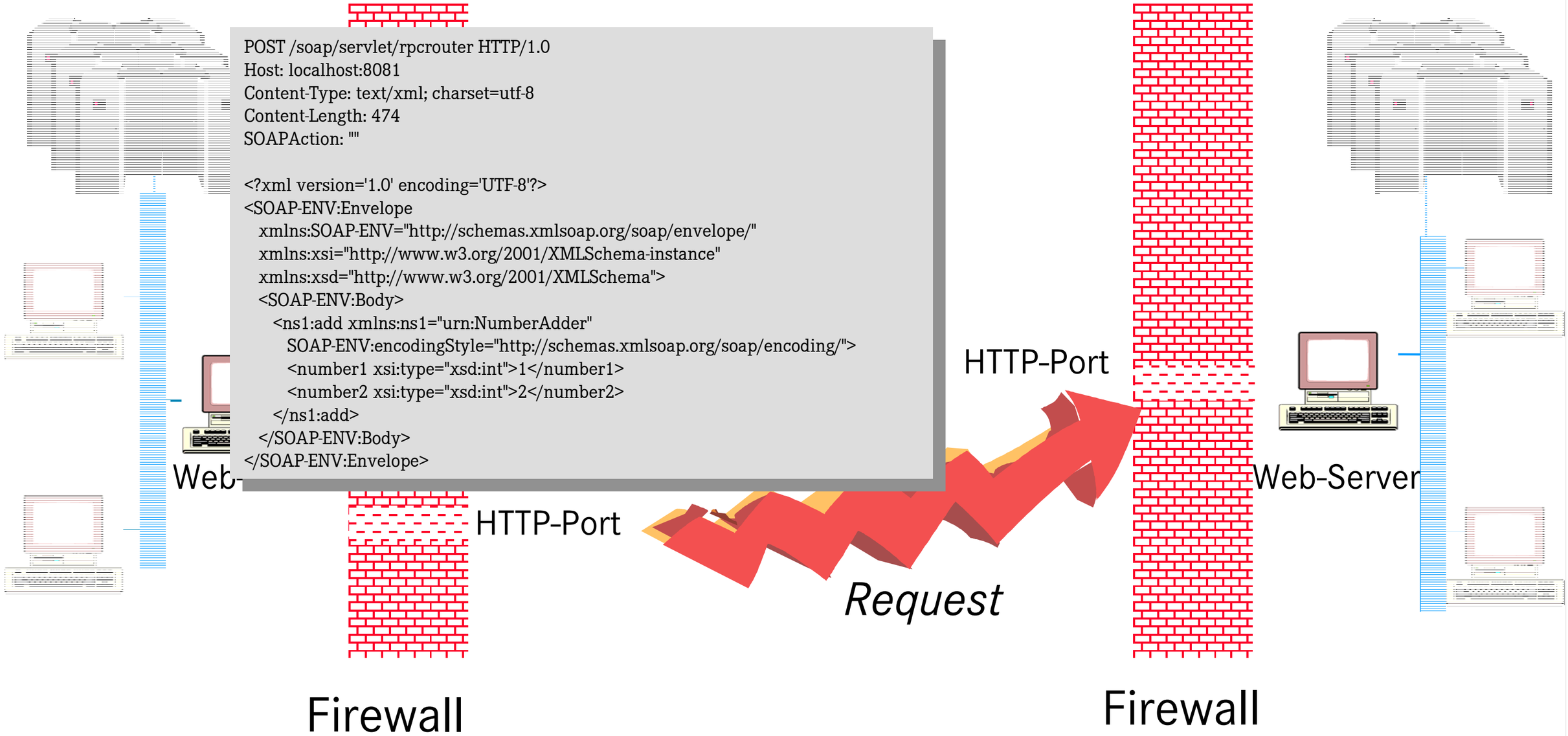
- Interoperabilitäts-Architekturen oftmals unpraktikabel für Unternehmens-übergreifende Integrationsaufgaben
- Implementierungs- und Wartungsaufwand
- Skalierbarkeit
- Sicherheitsaspekte im Internet! (z.B. Firewalls)
- Zunehmend leichtgewichtiger offener Kommunikations-mechanismus gewünscht
- Web Services

XML-basierter Nachrichtenaustausch und RPCs

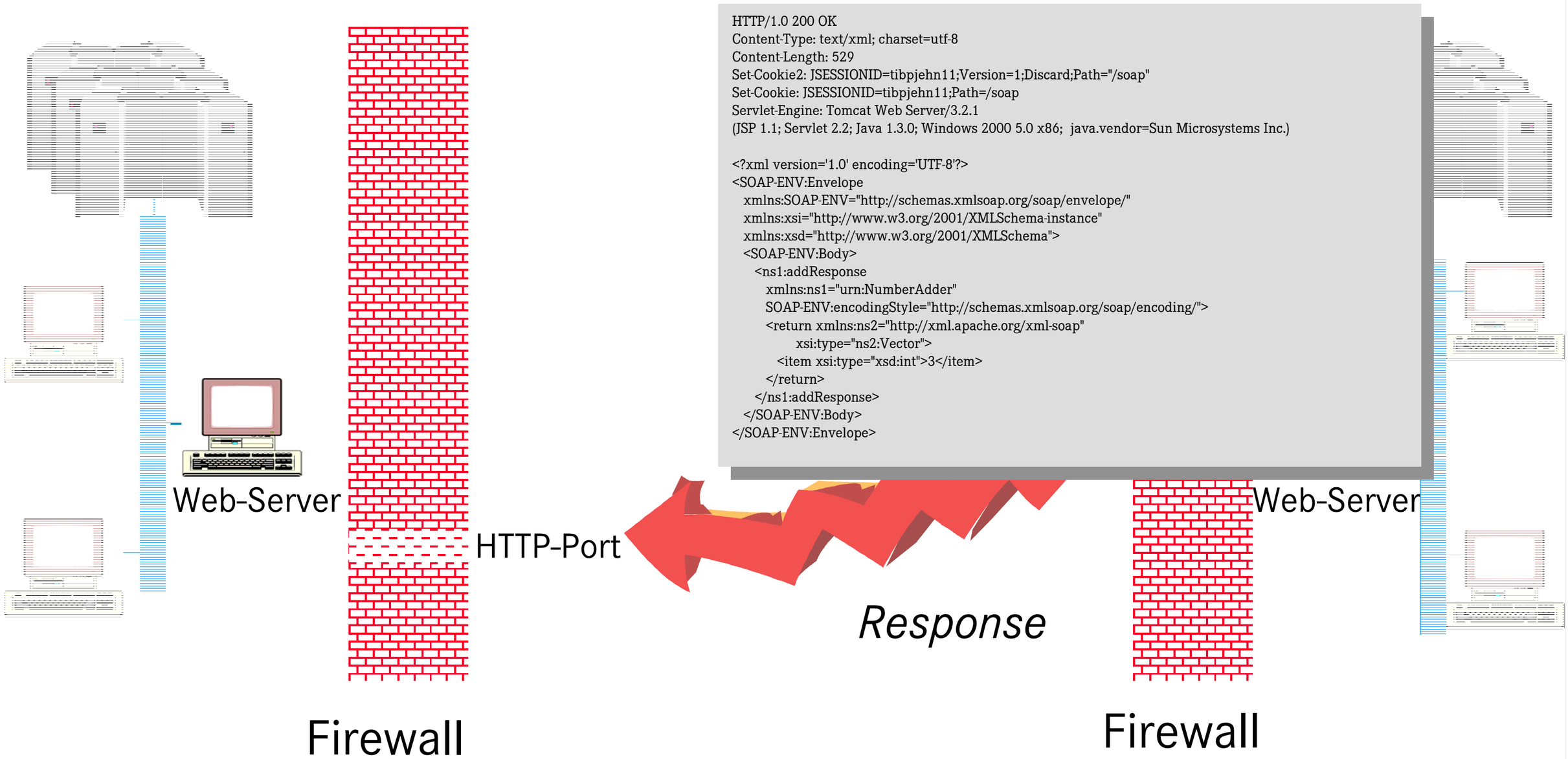


XML-basierter Nachrichtenaustausch und RPCs

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: localhost:8081
Content-Type: text/xml; charset=utf-8
Content-Length: 474
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:add xmlns:ns1="urn:NumberAdder"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <number1 xsi:type="xsd:int">1</number1>
      <number2 xsi:type="xsd:int">2</number2>
    </ns1:add>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

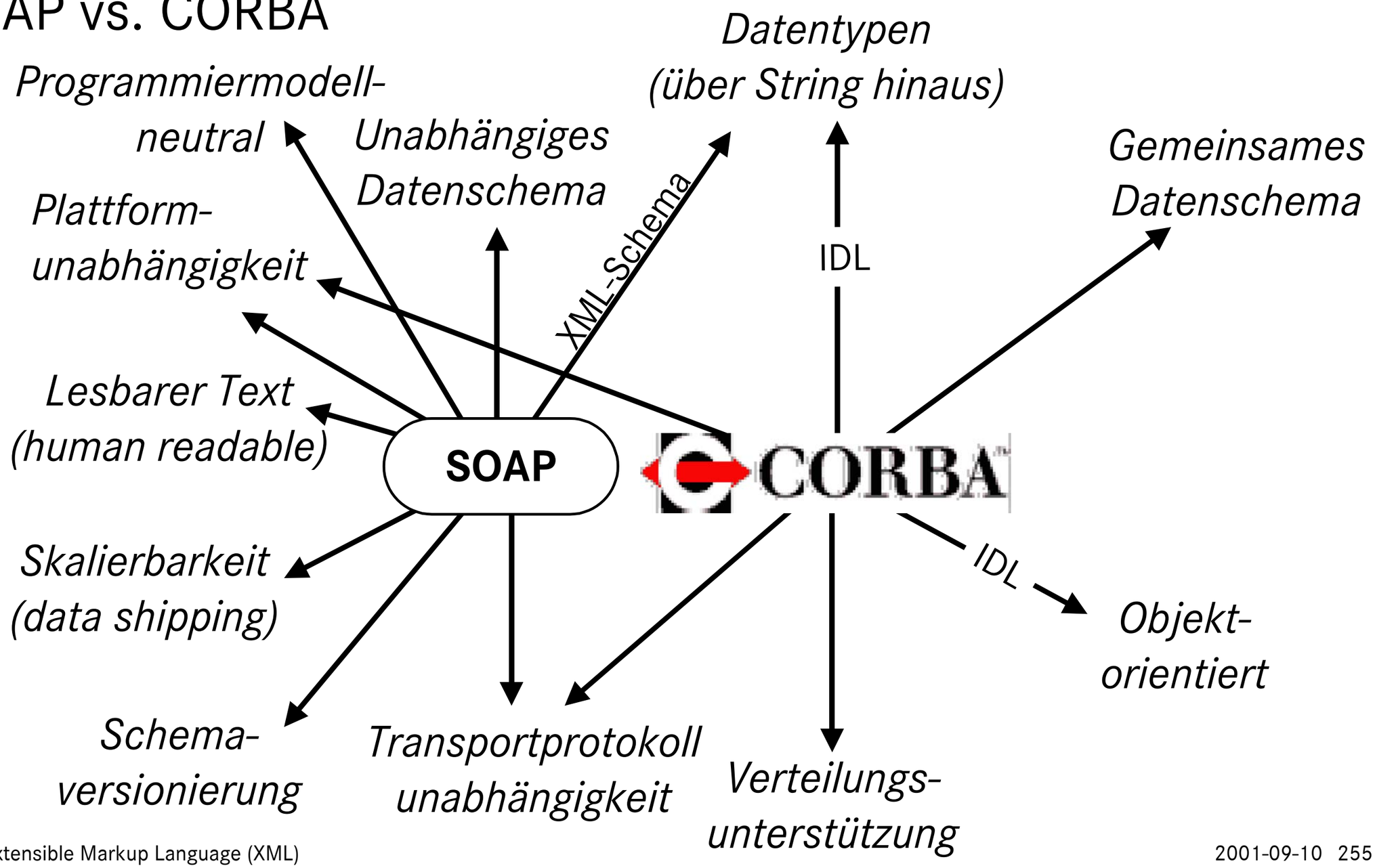


XML-basierter Nachrichtenaustausch und RPCs



XML-basierter Nachrichtenaustausch und RPCs

- SOAP vs. CORBA



XML-basierter Nachrichtenaustausch und RPCs

Kriterium	*CORBA*	*SOAP*
Plattformunabhängigkeit: Realisierung von Client und Server in (nahezu) beliebiger Programmiersprache auf beliebigem Betriebssystem	✔	✔
Lesbarer Inhalt: Ausgetauschte Dateninhalte mit Low-End Werkzeugen (wie Texteditoren) verarbeitbar	⊘	✔
Datentypen: Verarbeitung von programmiersprachlichen Datentypen Definition eigener Inhaltstypen	✔	✔

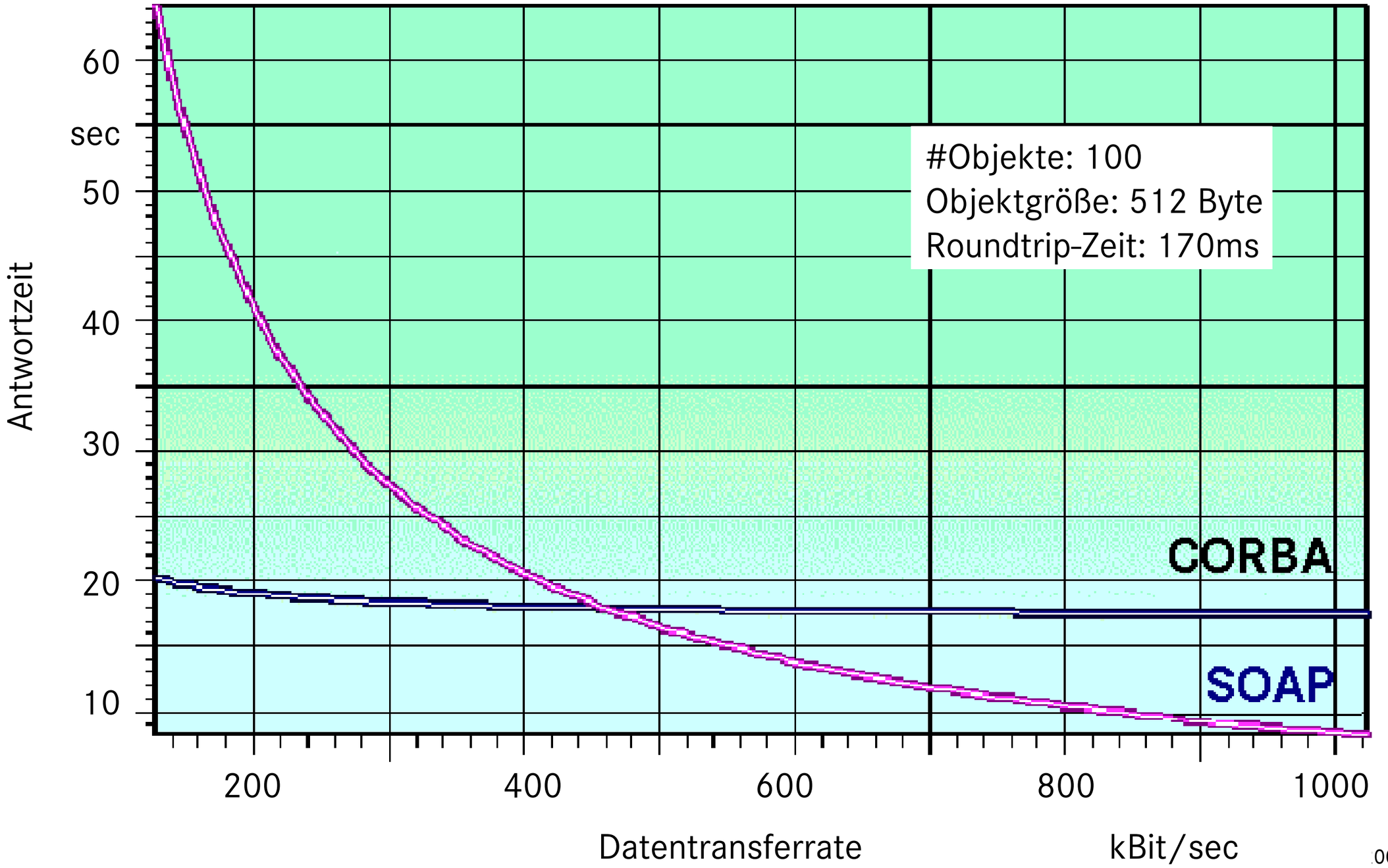
XML-basierter Nachrichtenaustausch und RPCs

Kriterium	*CORBA*	*SOAP*
Verarbeitung großer Datenmengen: Können Datenbestände größer als der zur Verfügung stehende Hauptspeicher verarbeitet werden	⊘	✓
Schemaversionierung: Verarbeitung von Instanzen verschiedener Schemaversionen durch die Applikation	⊘	✓
Verteilungsunterstützung:		
• Verteilungstransparenz	✓	⊘
• Lookup-Funktionalität		
• Authentifizierung und Sicherheit		
• Transaktionsunterstützung		

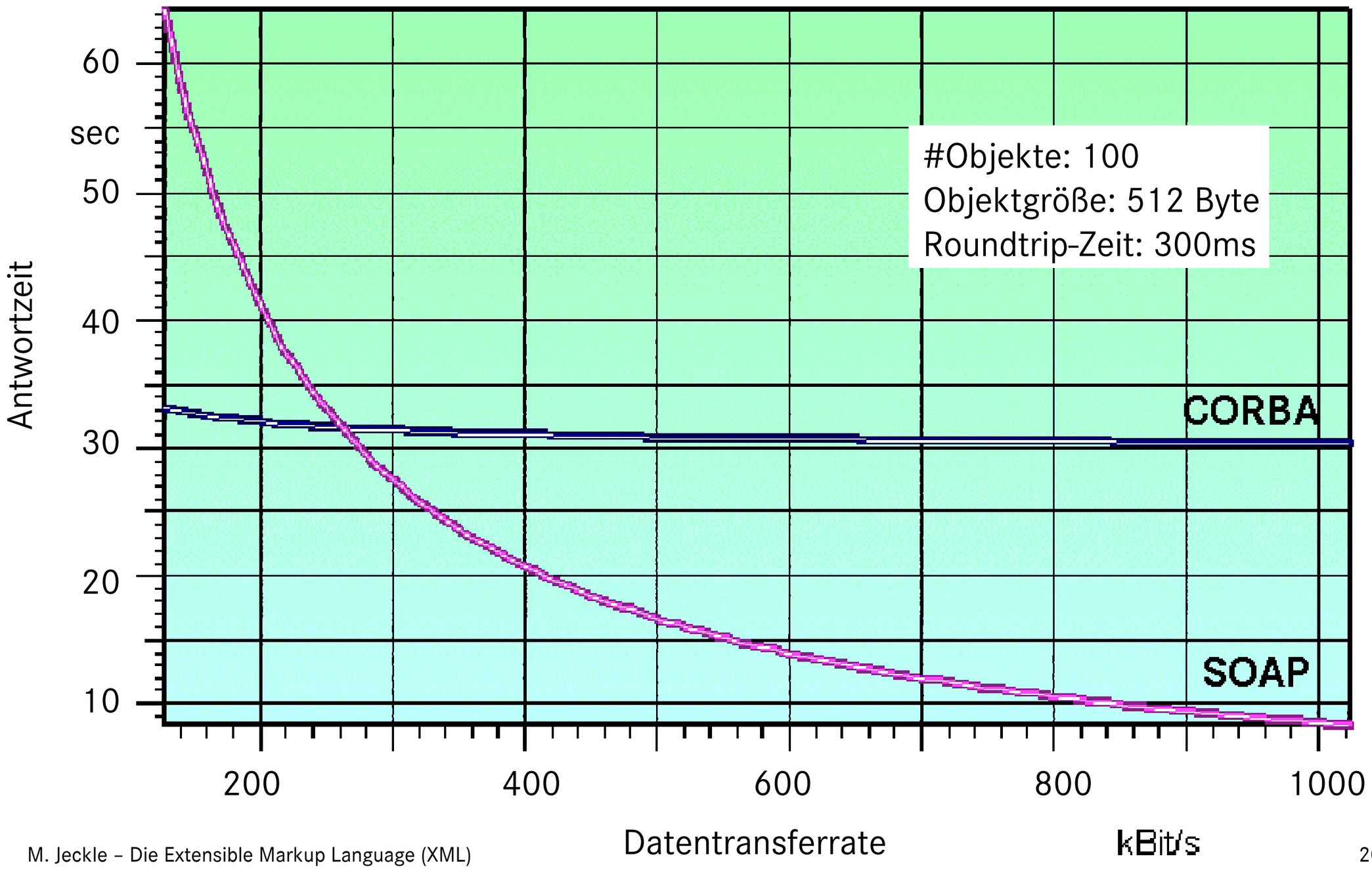
XML-basierter Nachrichtenaustausch und RPCs

Kriterium	*CORBA*	*SOAP*
Nachrichtenübermittlung Informationstransfer im Stile von Datagrammen (keine RPC-Semantik)	✓	✓
Programmiermodellunabhängigkeit Einsetzbarkeit in verschiedenen Programmiersprachen; Paradigmenneutralität	✓	✓
Transportprotokollunabhängigkeit Transparente Verwendung verschiedener (Transport- oder Sitzungs-)Protokolle	✓	✓

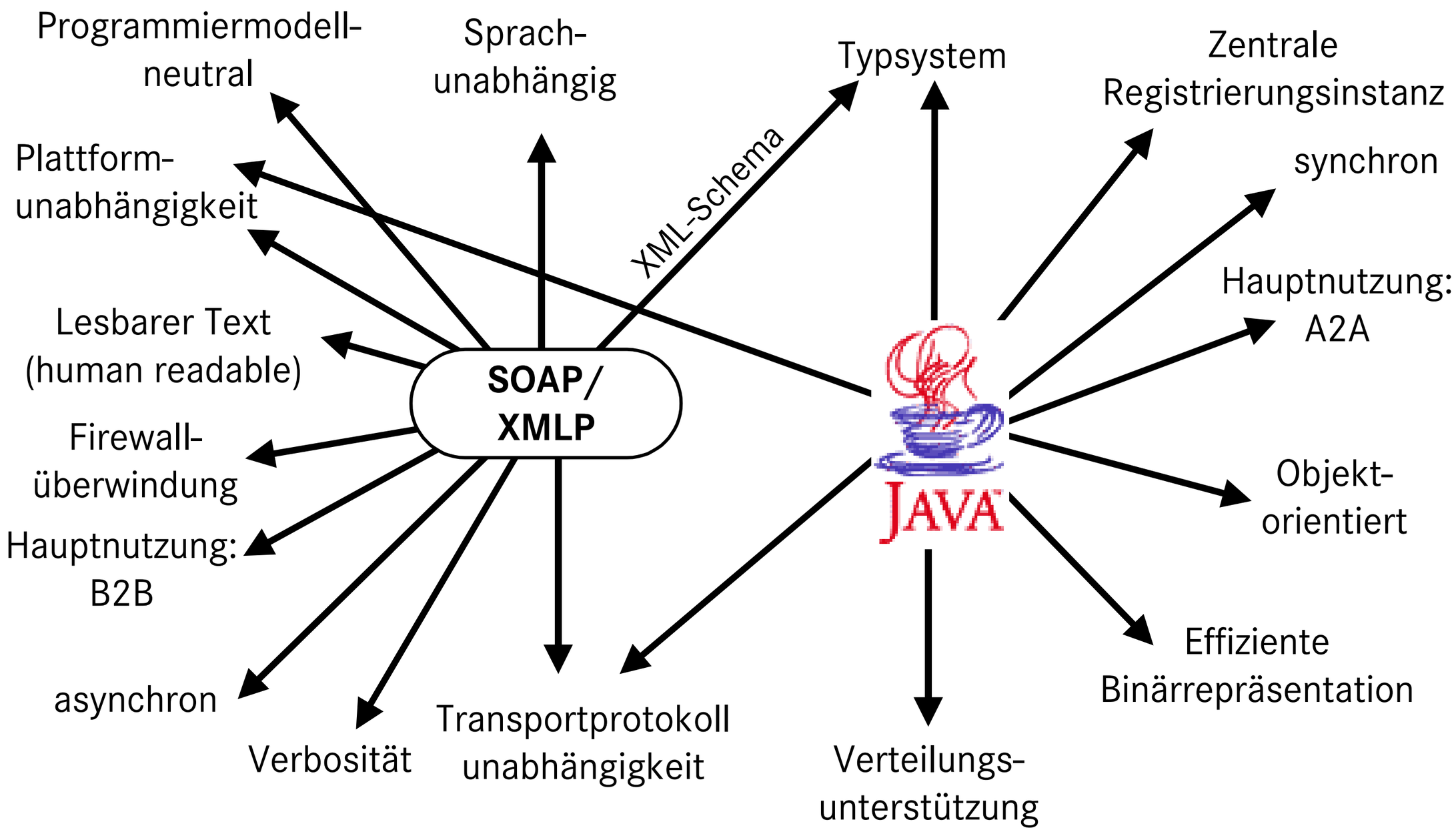
XML-basierter Nachrichtenaustausch und RPCs



XML-basierter Nachrichtenaustausch und RPCs

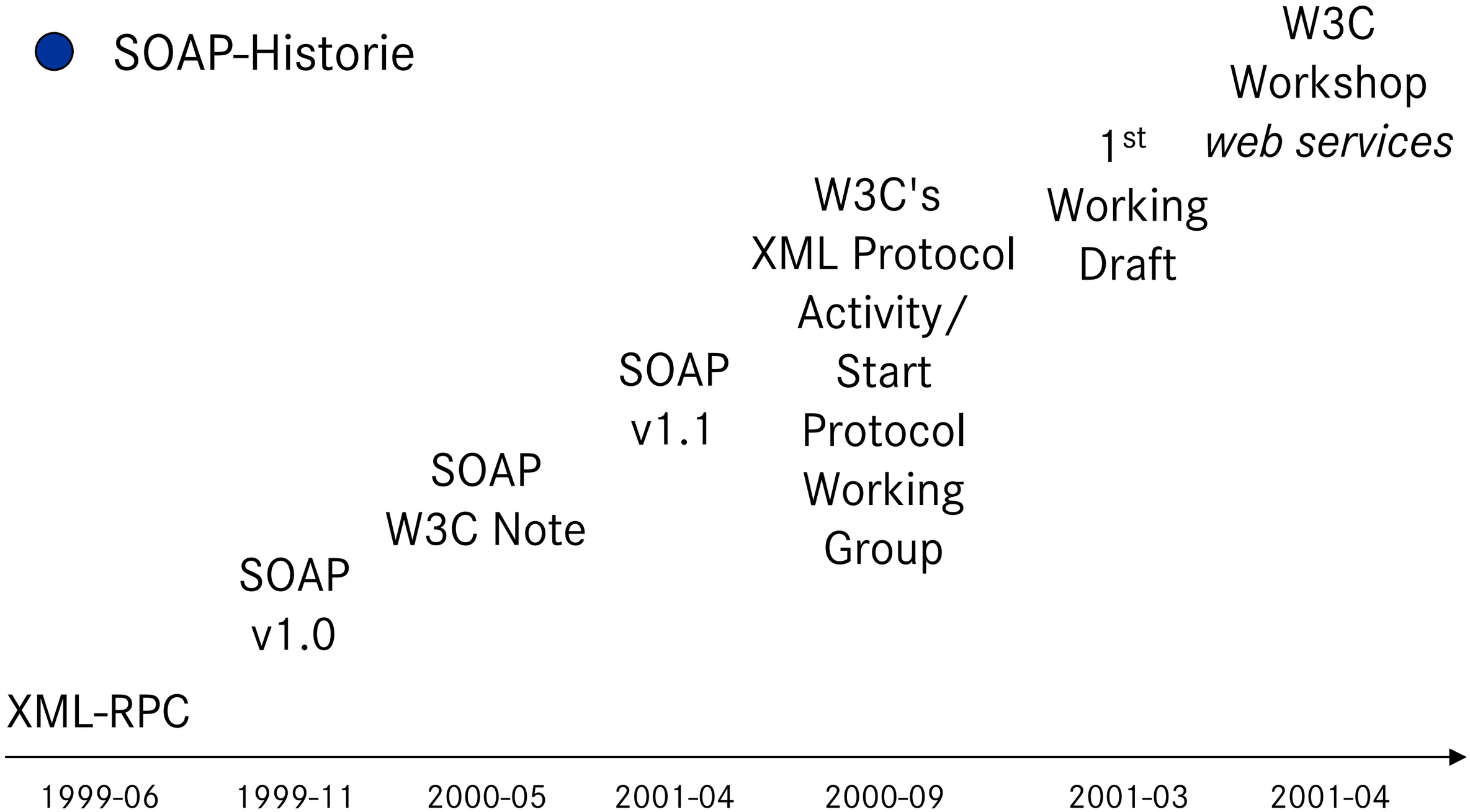


XML-basierter Nachrichtenaustausch und RPCs



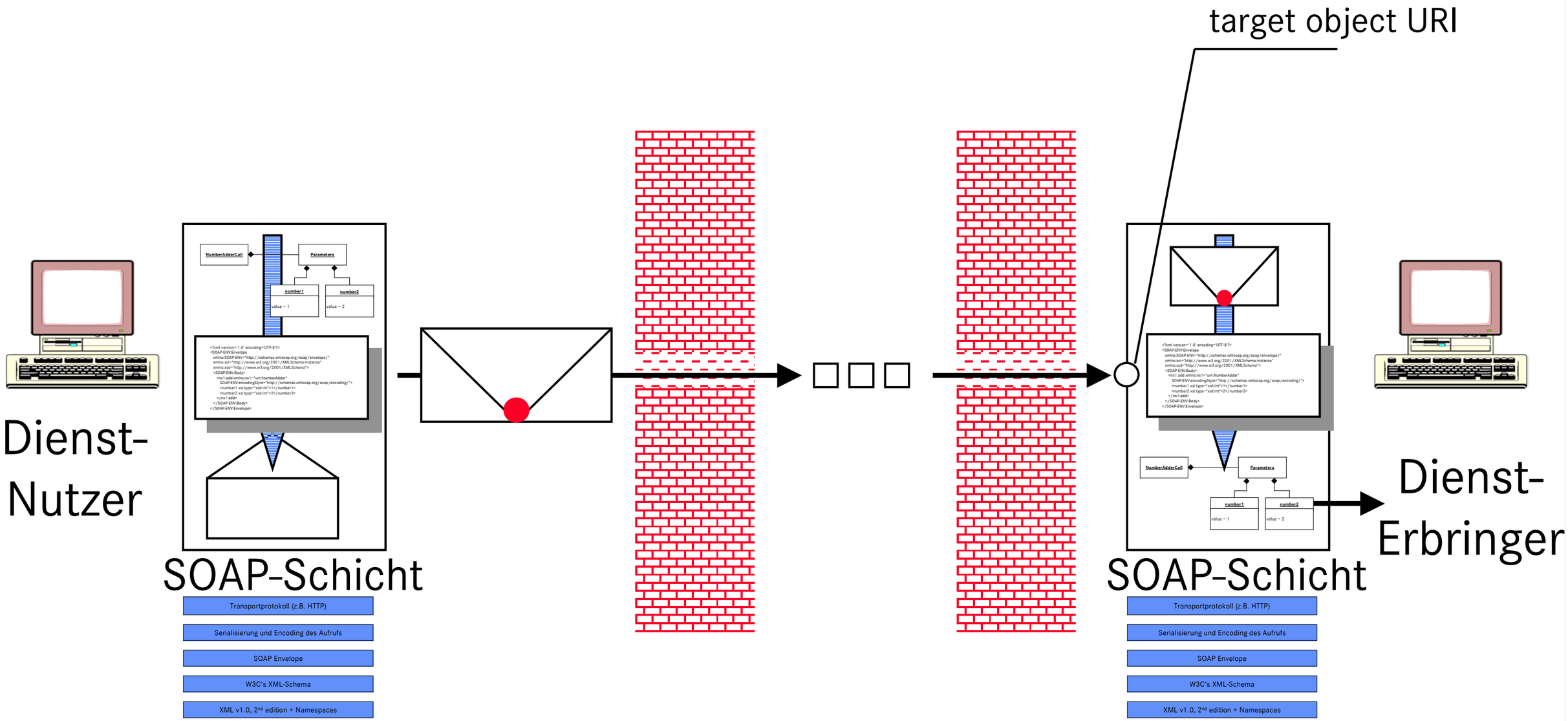
XML-basierter Nachrichtenaustausch und RPCs

- SOAP-Historie

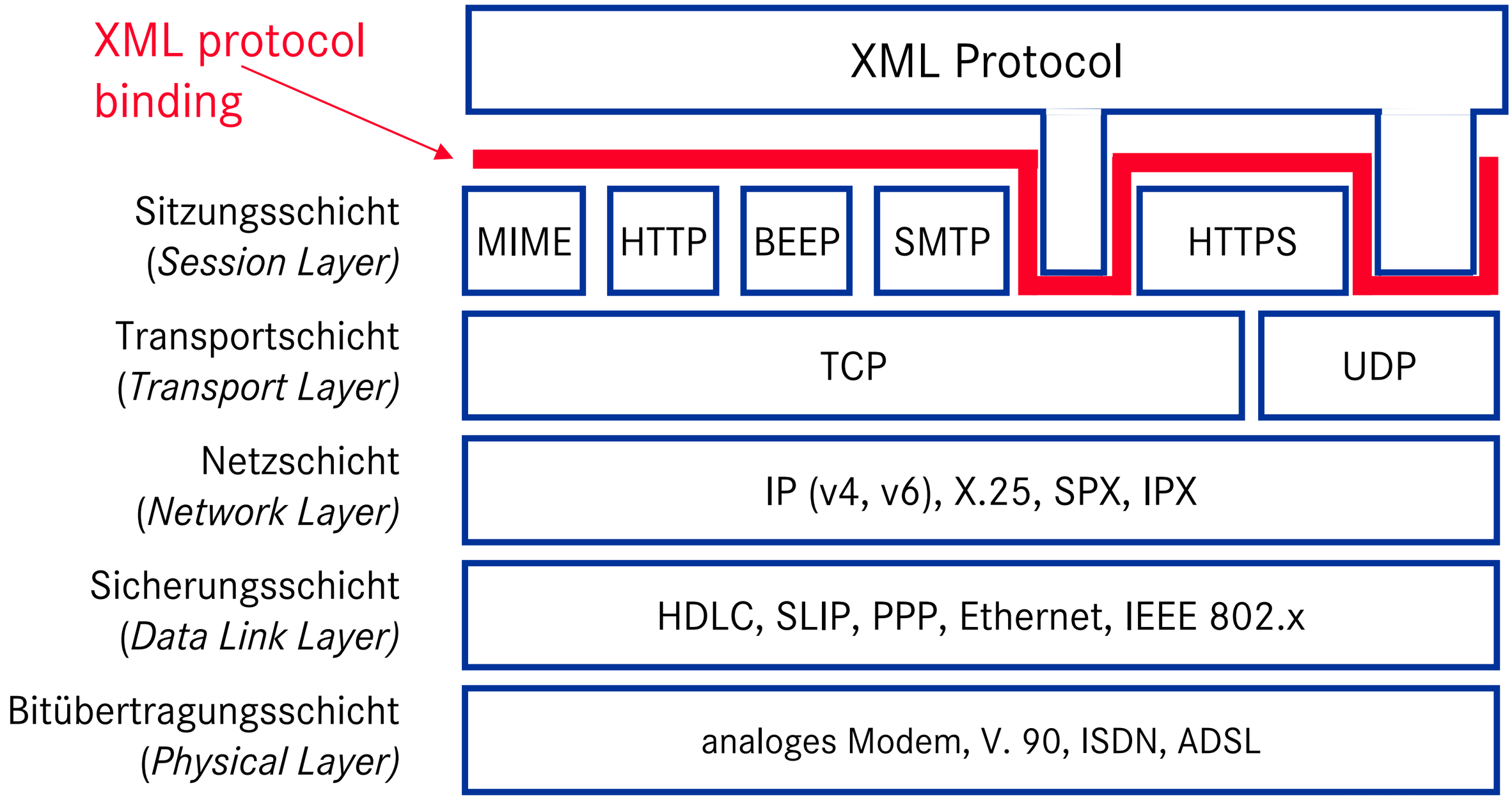


XML-basierter Nachrichtenaustausch und RPCs

- Ablauf eines SOAP-Aufrufs

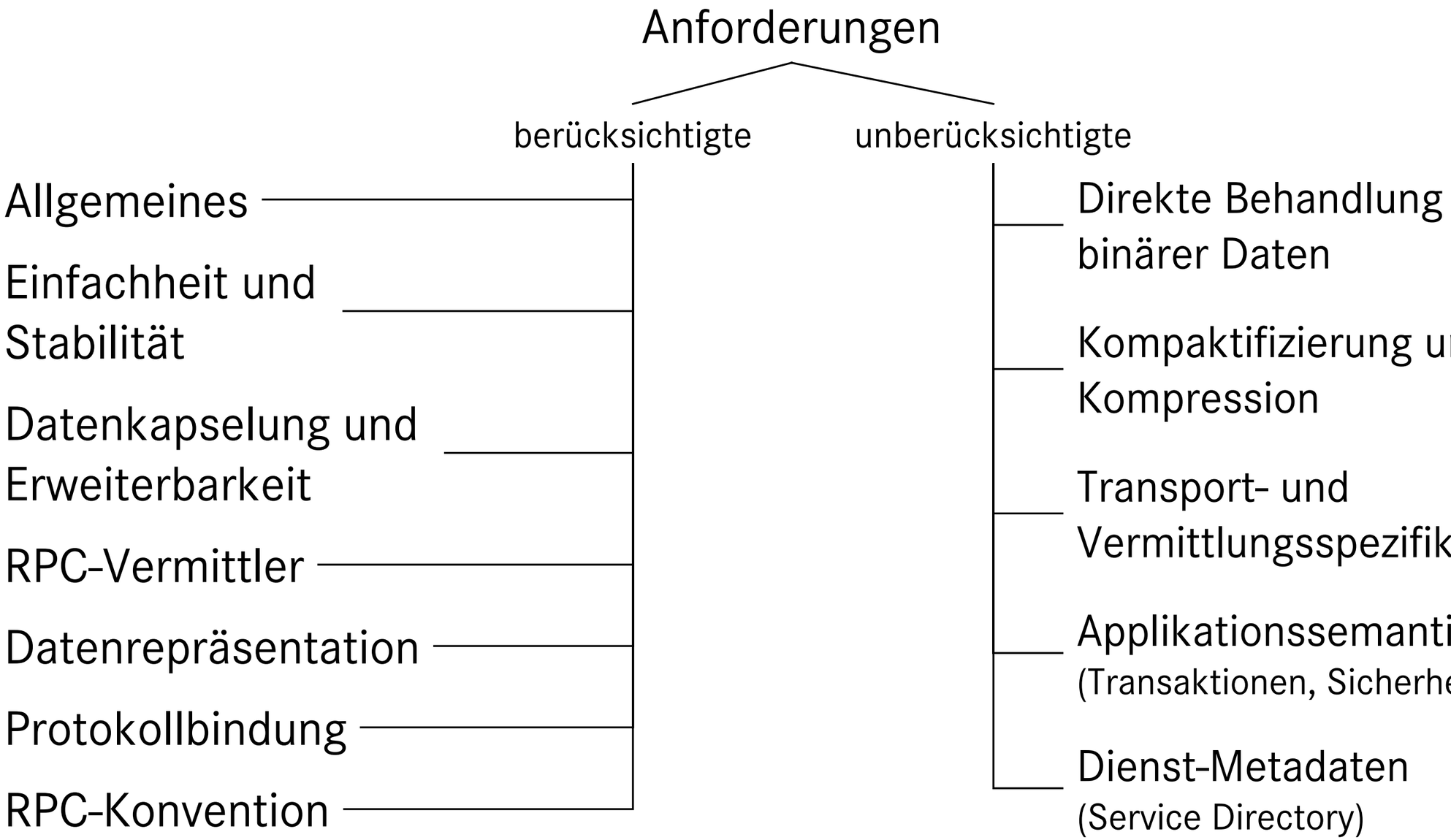


XML-basierter Nachrichtenaustausch und RPCs



XML-basierter Nachrichtenaustausch und RPCs

- Ausblick SOAP v1.2



XML-basierter Nachrichtenaustausch und RPCs

Ausblick SOAP v1.2

- Keine Festlegung auf
  - Programmiermodell (Sprache oder Paradigma)
  - Kommunikationsmechanismus (*protocol binding*)
  - Kommunikationsszenarien (*one-way, request-response, publish-subscribe, ...*)
- Leichtgewichtige Spezifikation
  - minimale Anzahl zwingender Eigenschaften
  - Orthogonalität optionaler Eigenschaften
- Kommunikation zwischen Partnern ohne vorheriges Wissen übereinander

XML-basierter Nachrichtenaustausch und RPCs

Ausblick SOAP v1.2

- XML-Schemabeschreibung eines Umschlages (*envelope*) zur Kapselung der transportierten Daten
- Definition eines Verarbeitungsmodells für den Umschlag; Berücksichtigung möglicher Fehlersituationen
- Einbettungsszenarien für Applikationsinformation in den Umschlag:
  - Umschlag enthält Applikationsinformation physisch
  - Applikationsinformation wird extern gehalten und durch den Inhalt des Umschlages referenziert
  - Umschlag enthält weitere Umschläge
  - Referenz auf externe Umschläge innerhalb eines Umschlages

XML-basierter Nachrichtenaustausch und RPCs

Ausblick SOAP v1.2

- Einsetzbarkeit über Organisationsgrenzen hinweg
- Modularität und Erweiterbarkeit durch Schichtenbildung als Basis einer langlebigen und zukunftssicheren Spezifikation
- Formulierung der Spezifikation so, daß Konformitätstests einfach durchzuführen sind; somit für Implementierer einfache Möglichkeit zur Gewährleistung der Konformität
- Dezentrale unabgestimmte Erweiterungsmöglichkeit
- Einfache Einsetzbarkeit in Systemen die bereits XML (insbesondere XML-Schema und XML-Namensräume) unterstützen
- Einsetzbarkeit auf Endgeräten mit beschränkten Ressourcen

XML-basierter Nachrichtenaustausch und RPCs

Ausblick SOAP v1.2

- Vermittler (*intermediaries*) können verschiedene Aufgaben auf dem Weg der XMLP-Nachricht zwischen Sender und dem Empfänger erfüllen:
  - Proxy (Empfang obwohl nicht endgültiger Empfänger)
  - Cache (Zwischenspeicherung zur Geschwindigkeitssteigerung)
  - Store-and-forward (Zwischenspeicherung gesamter XMLP-Nachrichten vor Weitersendung)
  - Protokollumsetzer (Gateways)
- Transportvermittler (*transport intermediary*)
  Keine Verarbeitung der übermittelten Nachricht, sondern Operation als Teil der Bindung an das Transportprotokoll
- Verarbeitender Vermittler (*processing intermediary*)
  Vollständige XMLP-Prozessoren mit der Möglichkeit die XMLP-Nachricht zu verarbeiten

XML-basierter Nachrichtenaustausch und RPCs

Ausblick SOAP v1.2

- Orthogonalität zwischen Datendarstellung (*data representation*) und Dateneinbettung (*data encapsulation*) in eine XMLP-Nachricht
- Unterstützung der XML-Schematypen *simple-types* und *complex-types* für die Datenrepräsentation
- Datenrepräsentation muß Instanzen von nicht XML-Schema-basierenden Modellen (z.B.: Objektgraphen, gerichtete Graphen) ausdrücken können
- Referenzierung von nicht-serialisierten Daten durch Uniform Resource Identifier (URI)
- Darstellung geschachtelter Arrays

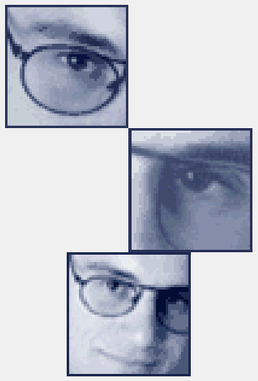
XML-basierter Nachrichtenaustausch und RPCs

- Zusammenfassung: *Nachrichtenaustausch und RPCs*
  - Versendung XML encodierter Nachrichten zur Überwindung von Firewall-Grenzen
  - Protokollunabhängige RPCs mit dem SOAP-Ansatz
  - Standardisierung von SOAP
  - Unterstützung durch Toolkits und verfügbare Lösungen
  - Kein Client-seitiges Customizing erforderlich
  - Basis für Web Services

jeckle.de - Netscape

File Edit View Go Communicator Help

Bookmarks Go to: http://www.jeckle.de



Mario Jeckle ...
Dialog ...
über diese Seiten ...
suchen ...
SiteMap

Unified Modeling Language (UML)
eXtensible Markup Language (XML)
XML Metadata Interchange (XMI)

Vorträge und Publikationen
Vorlesungen
GOOAL.net
XML-Arbeitskreis

Internet > Search Engines
Mersennesche Primzahlen
Feedback
Rotkreuz Mitgliederverwaltung

jeckle.de

Document: Done