DAIMLERCHRYSLER

XML-Schemaerzeugung im Business

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

Gliederung

I XML-Schema

- Eine neue XML-Sprache zur Beschreibung von XML-Vokabularen
- Beschreibung von datenorientierten
  und business-(document-)orientierten Inhalten

II Herausforderungen im Umfeld Schemaerzeugung

- Integration in den (Entwicklungs-)Prozeß
- Technisch-strukturelle Fragestellungen

III Schemagenerierung in der Praxis

- Erzeugung von XML-Schemata aus bestehenden Modellen
- Der standardisierte Ansatz *XML Metadata Interchange*

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, XMLP, ...)
- Werkzeugunterstützung verfügbar
- Erster Schritt der Schema-Bestrebungen, weitere werden folgen ...

XML-Schema: – ein Beispieldokument

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>XML-Schemaerzeugung im Business</Titel>
  <Veranstaltung>
    <Name>5. Darmstädter XML-Kongress</Name>
    <Datum>2001-06-11</Datum>
  </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>
```

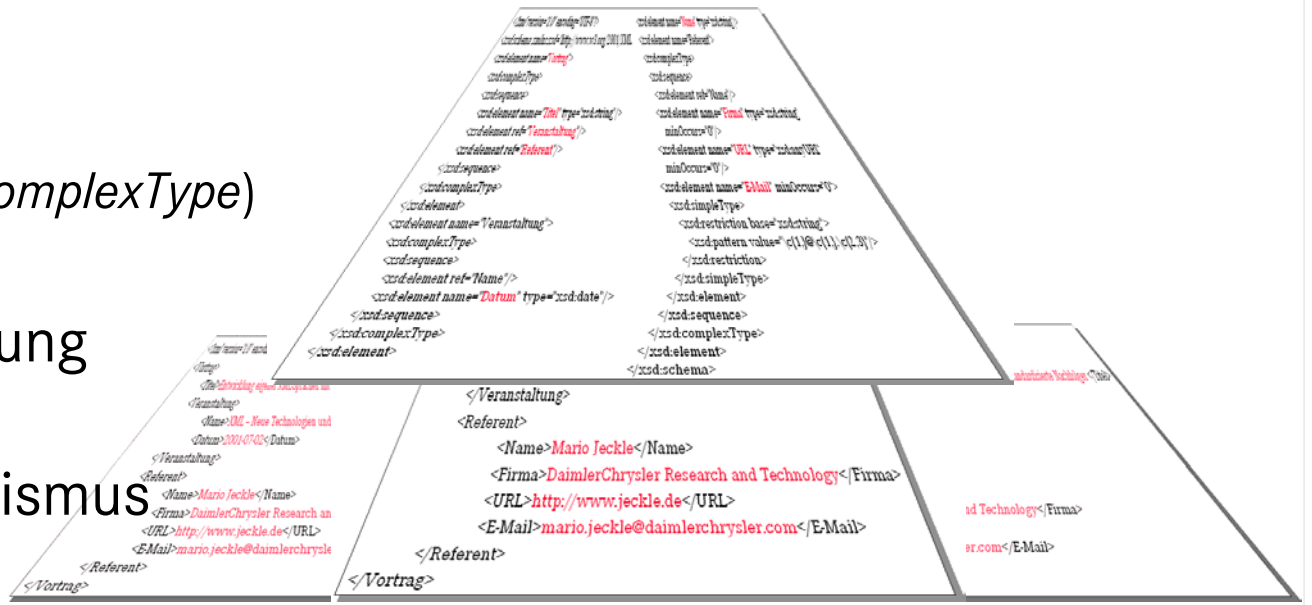
XML-Schema: – ... eine zugehörige Grammatik

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XML
  <xsd:element name="Vortrag">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:element ref="Veranstaltung"/>
        <xsd:element ref="Referent"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Veranstaltung">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Name"/>
        <xsd:element name="Datum" type="xsd:date"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Referent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Name"/>
        <xsd:element name="Firma" type="xsd:string"
          minOccurs="0"/>
        <xsd:element name="URL" type="xsd:anyURI"
          minOccurs="0"/>
        <xsd:element name="E-Mail" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="\c{1,}@\c{1,}.\c{2,3}"/>
            </xsd:restriction>
          </xsd:simpleTyp
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Vortrag>
  <Titel>XML-Schemaerzeugung im Business</Titel>
  <Veranstaltung>
    <Name>5. Darmstädter XML-Kongress</Name>
    <Datum>2001-06-11</Datum>
  </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>
```

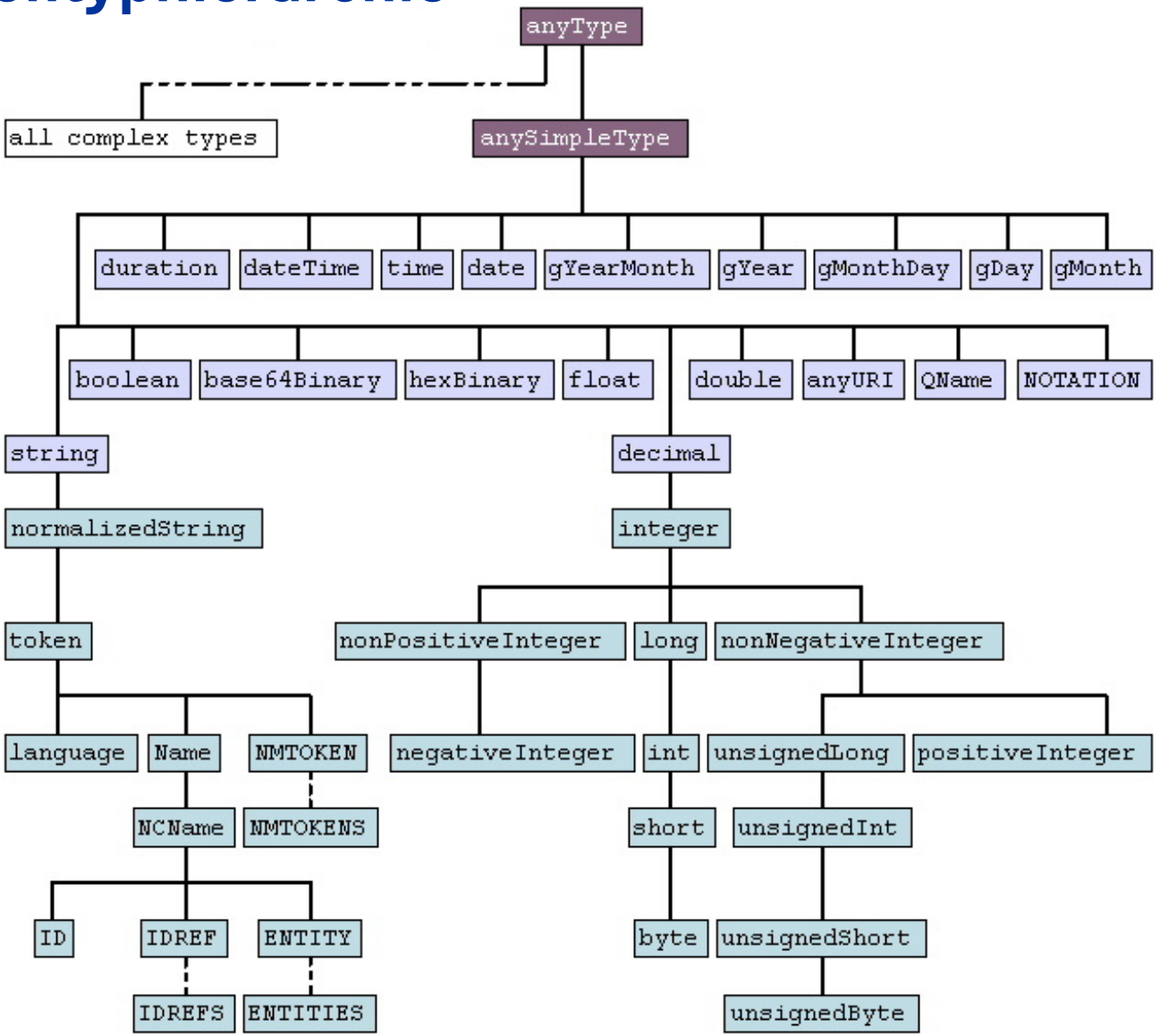
XML-Schema: Mächtigkeit

- Strukturell: Attribute und Elemente (wie in DTDs)
- Namespace-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



XML-Schema: Datentyphierarchie

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



Aus: XML-Schema Part 2: Datatypes,
W3C-Recommendation 2001-05-02, chap. 3

XML-Schema: Datentypen

Primitive Datentypen:

- string
- boolean
- decimal
- float
- double
- duration
- dateTime
- time
- date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- hexBinary
- base64Binary
- anyURI
- QName
- NOTATION

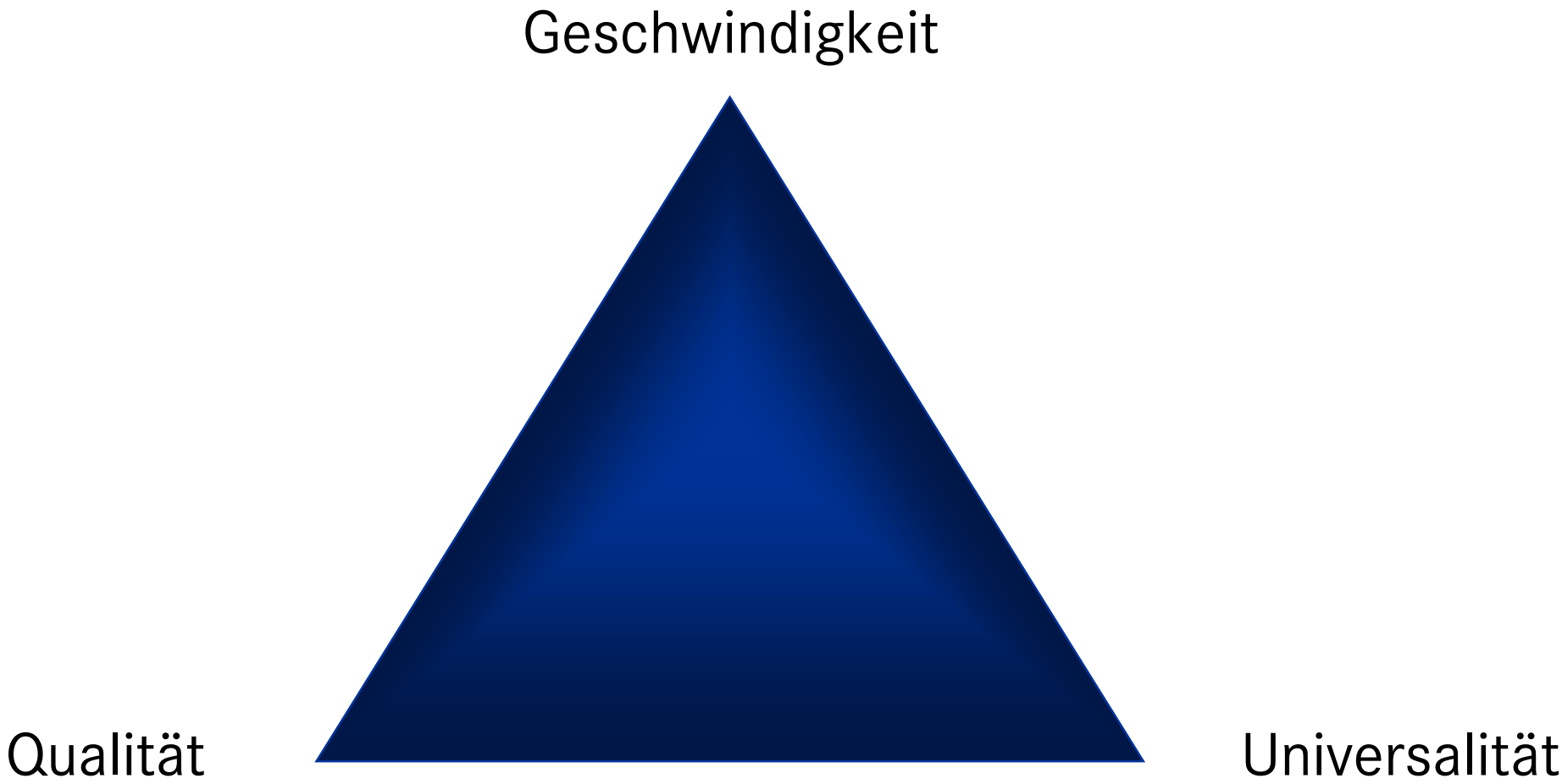
Abgeleitete Datentypen:

- normalizedString
- token
- language
- IDREFS
- ENTITIES
- NMTOKEN
- NMTOKENS
- Name
- NCName
- ID
- IDREF
- ENTITY
- integer
- nonPositiveInteger
- negativeInteger
- long
- int
- short
- byte
- nonNegativeInteger
- unsignedLong
- unsignedInt
- unsignedShort
- unsignedByte
- positiveInteger

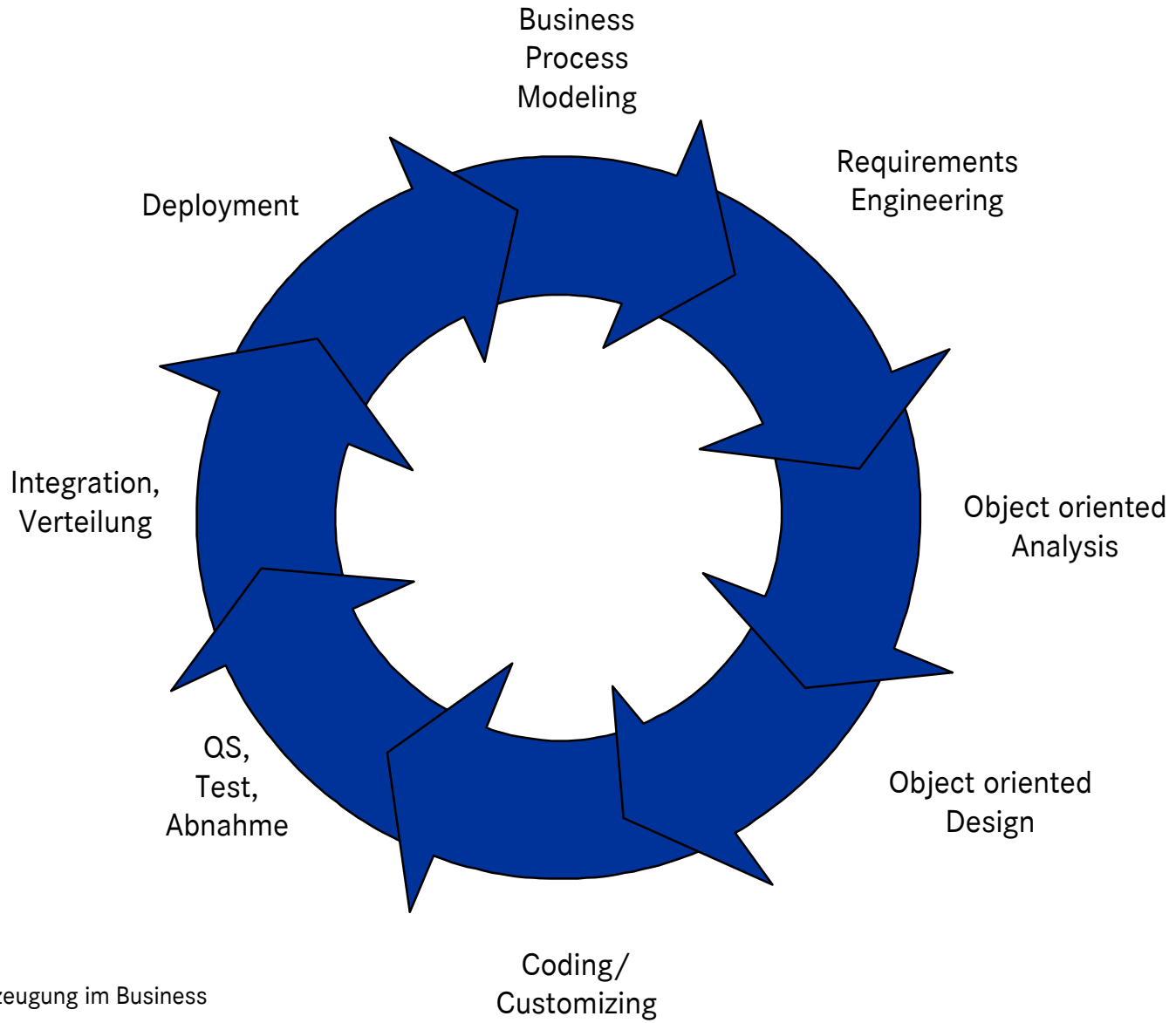
XML-Schemaerzeugung: Herausforderungen

- **Flexibilität**
  Zukunftssicherheit des entstehenden Vokabulars
  (Adaptionsfähigkeit und verbundener Aufwand an veränderte Anforderungen)
- **Geschwindigkeit**
  Schnelle Erzeugung von XML-Vokabularen für A2A-, B2B-, B2C-Anwendungen
  (XML-Sprachen als Basis der Daten- und Prozeßintregration)
- **Kohärenz**
  Synchronisation zwischen Daten- und Prozeßstrukturen und XML-Format
- **Korrektheit**
  XML-Format soll Daten- und Prozeßstrukturen möglichst unverändert wiederspiegeln
- **Stil**
  XML-Format soll *look-and-feel* der Applikation aufweisen
  (*Look*: Visuelle Sprachaspekte, *Feel*: Technisch strukturelle Aspekte)
- **Integration**
  Geringer zusätzlicher Aufwand durch XML-Vokabularerzeugung
- **Wiederverwendung**
  ... vorhandenen (Meta-)Wissens über Daten- und Prozeßstrukturen und Modellierung

XML-Schemaerzeugung: Das magische Dreieick



XML-Schemaerzeugung: Integration im Entwicklungsprozeß



XML-Schemaerzeugung: Integration im Entwicklungsprozeß

Deployment

Business Process Modeling

Requirements Engineering

Integration, Verteilung

Object oriented Analysis

QS, Test, Abnahme

Coding/ Customizing

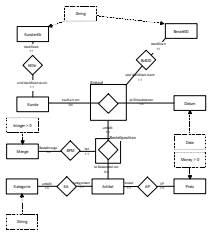
Object oriented Design

Einsatzfehlerprotokoll

Fehlerprotokoll

Datum	Beschreibung	reproduzierbar	gemeldet
2001-01-07	kein DB-Zugriff in Maske Stammdaten	Y	T. Müller
2001-01-09	Fehlerhafte Rundung bei Summenbildung	Y	R. Meier

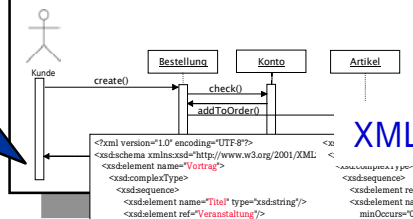
Informationsmodell



informale Formulierung

Ein *Bestellvorgang* setzt sich aus der *Bonitätsprüfung* des *Auftraggebers,* der *Ausbuchung* aus dem *Warenkatalog* sowie der *Rechnungserstellung* zusammen.

Dynamische Abläufe

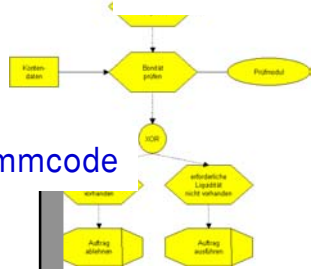


XML-Vokabular

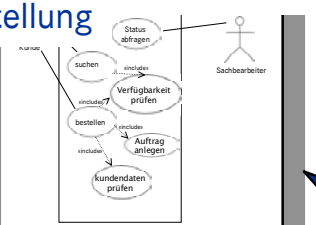
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Vortrag">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:element ref="Veranstaltung"/>
        <xsd:element ref="Referent"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Veranstaltung">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Name"/>
        <xsd:element name="Datum" type="xsd:date"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element ref="Name"/>
  <xsd:element name="Firma" type="xsd:string"
    minOccurs="0"/>
  <xsd:element name="URL" type="xsd:anyURI"
    minOccurs="0"/>
  <xsd:element name="E-Mail" minOccurs="0">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="\c{1,}@\c{1,}\.\c{2,3}"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

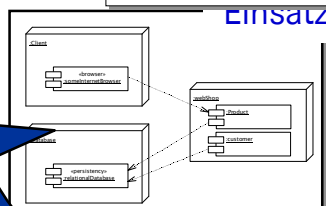
formalisierte Darstellung



Anwendungsfälle



Einsatzszenario



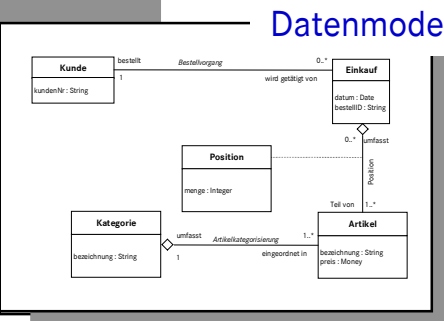
Darstellung

<> k2

<> a2.artike

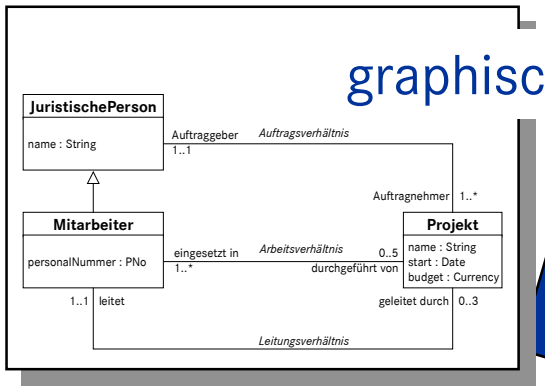
Programmcode

Datenmodell



XML-Schemaerzeugung: Von Datenmodellen zu XML-Vokabularen

graphisches UML-Model

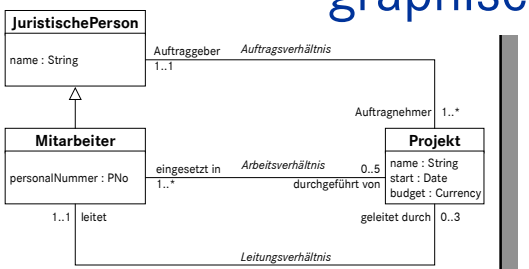


XML Schema (XSD)

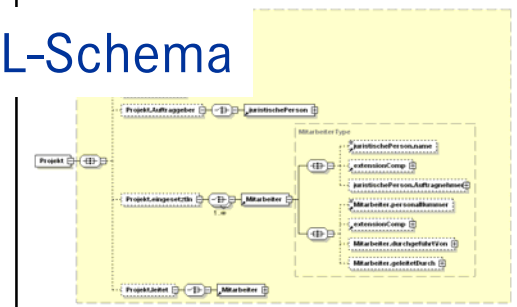
```
<?xml version="1.0" encoding="utf-8"?>
                                    ://www.w3.org/2000/10/XMLSchema">
                                    ed by DaimlerChrysler XSD-Generator 2001-05-25T22:50:36+02:00</xsd:documentation>
                                  IdentityAttribs">
  <xsd:attribute name="id" type="xsd:ID"/>
  <xsd:attribute name="label" type="xsd:string"/>
</xsd:attributeGroup>
<xsd:element name="XMI">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="XMI.header" type="headerType" minOccurs="0"/>
      <xsd:element name="XMI.content" minOccurs="0"/>
      <xsd:element name="XMI.difference" type="differenceType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="XMI.extensions" type="extensionsType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="XMI.version" type="xsd:string" use="fixed" value="2.0dcx0.9"/>
    <xsd:attribute name="timestamp" type="xsd:string"/>
    <xsd:attribute name="verified" type="xsd:boolean"/>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="headerType">
...
```

XML-Schemaerzeugung: Von Datenmodellen zu XML-Vokabularen

graphisches UML-Model



graphisches XML-Schema



Export

Visualisierung

XML Repräsentation

```
<?xml version="1.0" encoding="UTF-8"?>
<XMI xmi.version="1.0" timestamp="2001-05-25T22:48:0
 <XMI.header>
  <XMI.documentation>
   <XMI.exporter>some UML-Tool</XMI.exporter>
  </XMI.documentation>
  <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>
 </XMI.header>
<XMI.content>
<Model_Management.Model xmi.id="model:example" >
  <Foundation.Core.ModelElement.name>example</Foundation.Core.ModelElement.name>
  <Foundation.Core.ModelElement.visibility xmi.value="public"/>
  <Foundation.Core.ModelElement.isSpecification xmi.value="false"/>
  <Foundation.Core.GeneralizableElement.isRoot xmi.value="false"/>
  <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
  <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
  <Foundation.Core.Namespace.ownedElement>
    <Foundation.Core.Class xmi.id="class:juristischePerson" >
      <Foundation.Core.ModelElement.name>juristischePerson</Foundation.Core.ModelElement.name>
      <Foundation.Core.ModelElement.visibility xmi.value="public"/>
      <Foundation.Core.ModelElement.isSpecification xmi.value="false"/>
      <Foundation.Core.GeneralizableElement.isRoot xmi.value="true"/>
      <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
      <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
...
```

Schema-

Erzeugung

XML Schema (XSD)

```
<?xml version="1.0" encoding="utf-8"?>
                                    ://www.w3.org/2000/10/XMLSchema">
                                    ed by DaimlerChrysler XSD-Generator 2001-05-25T22:50:36+02:00</xsd:documentation>
                                    "IdentityAttribs">
  <xsd:attribute name="id" type="xsd:ID"/>
  <xsd:attribute name="label" type="xsd:string"/>
</xsd:attributeGroup>
<xsd:element name="XMI">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="XMI.header" type="headerType" minOccurs="0"/>
      <xsd:element name="XMI.content" minOccurs="0"/>
      <xsd:element name="XMI.difference" type="differenceType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="XMI.extensions" type="extensionsType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="XMI.version" type="xsd:string" use="fixed" value="2.0dcx0.9"/>
    <xsd:attribute name="timestamp" type="xsd:string"/>
    <xsd:attribute name="verified" type="xsd:boolean"/>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="headerType">
...
```

XML-Schemaerzeugung: Technisch-strukturelle Fragestellungen

Datentypen

- Abbildung der im Modell verwendeten (UML, CORBA, Business-spezifischen, ...) Datentypen auf die durch XML-Schema Part 2 angebotenen (Forderungen der Korrektheit und Kohärenz)

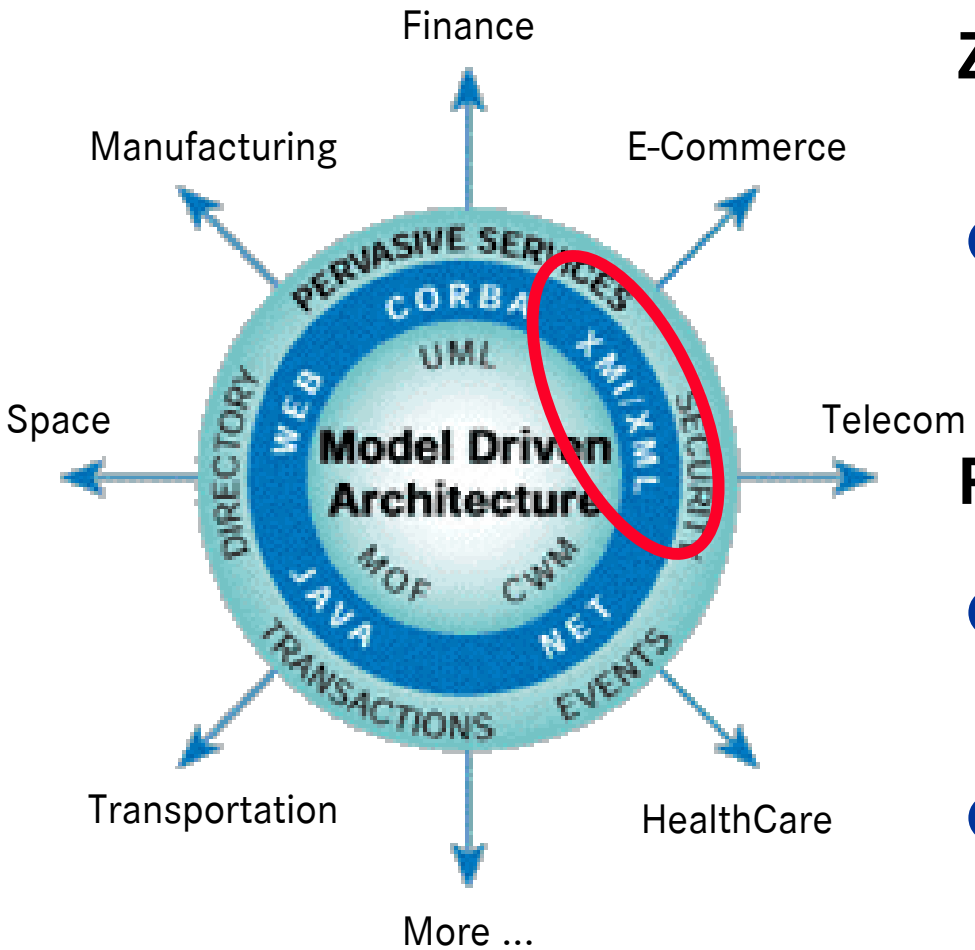
Strukturen

- Abbildung der modelliersten (Business-)Strukturen auf die XML-inhärente Baumsicht (Forderungen der Korrektheit, Kohärenz und des Stils)

Pragmatischer Aspekt

- Sicherheit, Reproduzierbarkeit und Zweckmäßigkeit der Umsetzung, insbesondere im Kontext existierender Systeme und Prozesse (Forderungen der Flexibilität, Geschwindigkeit, Integration und Wiederverwendung)

XML-Schemaerzeugung: Lösungsansatz XML Metadata Interchange



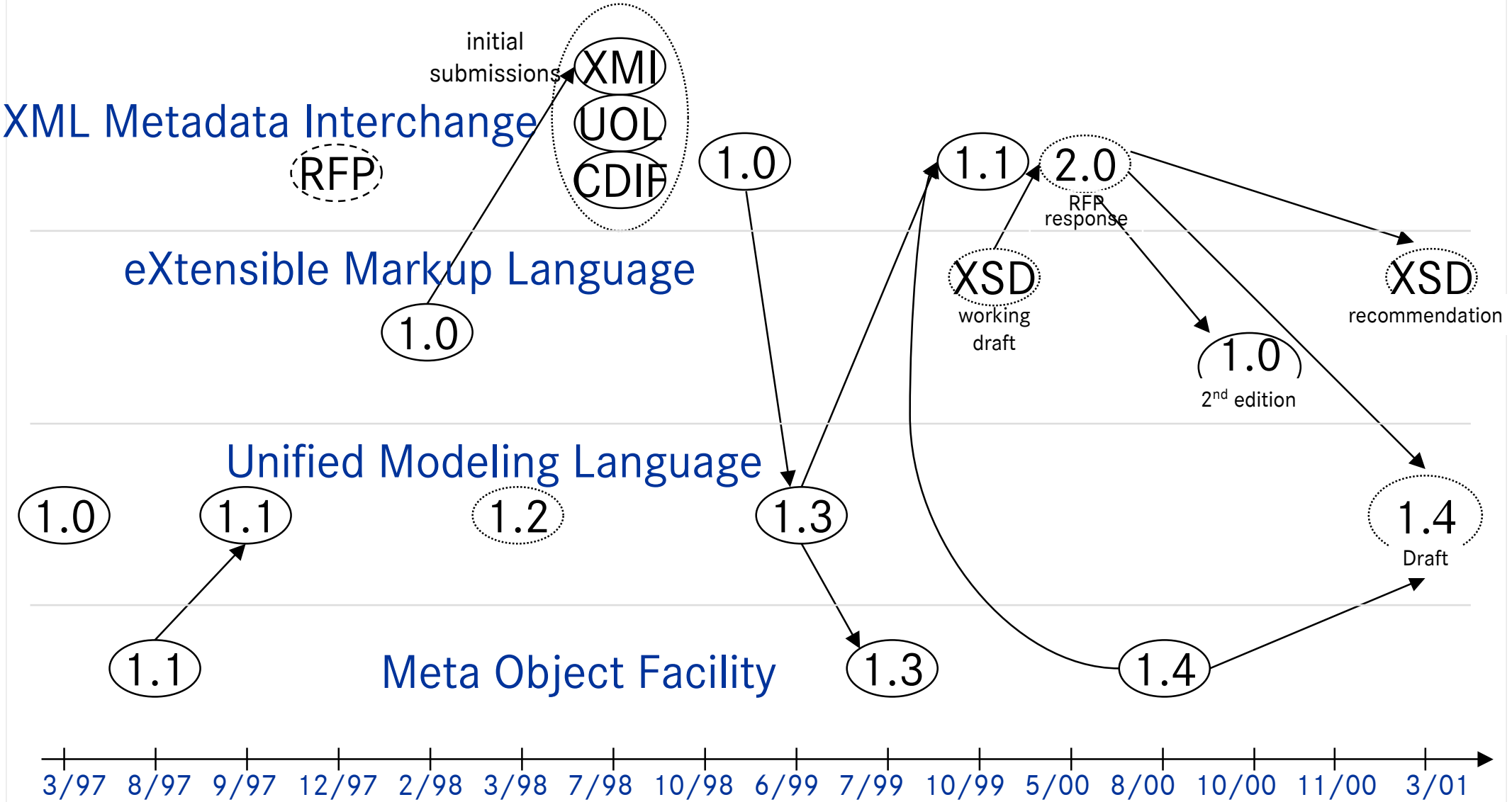
Zielsetzung:

- Metadatenaustausch zwischen Werkezeugen und Metadaten-Repositories

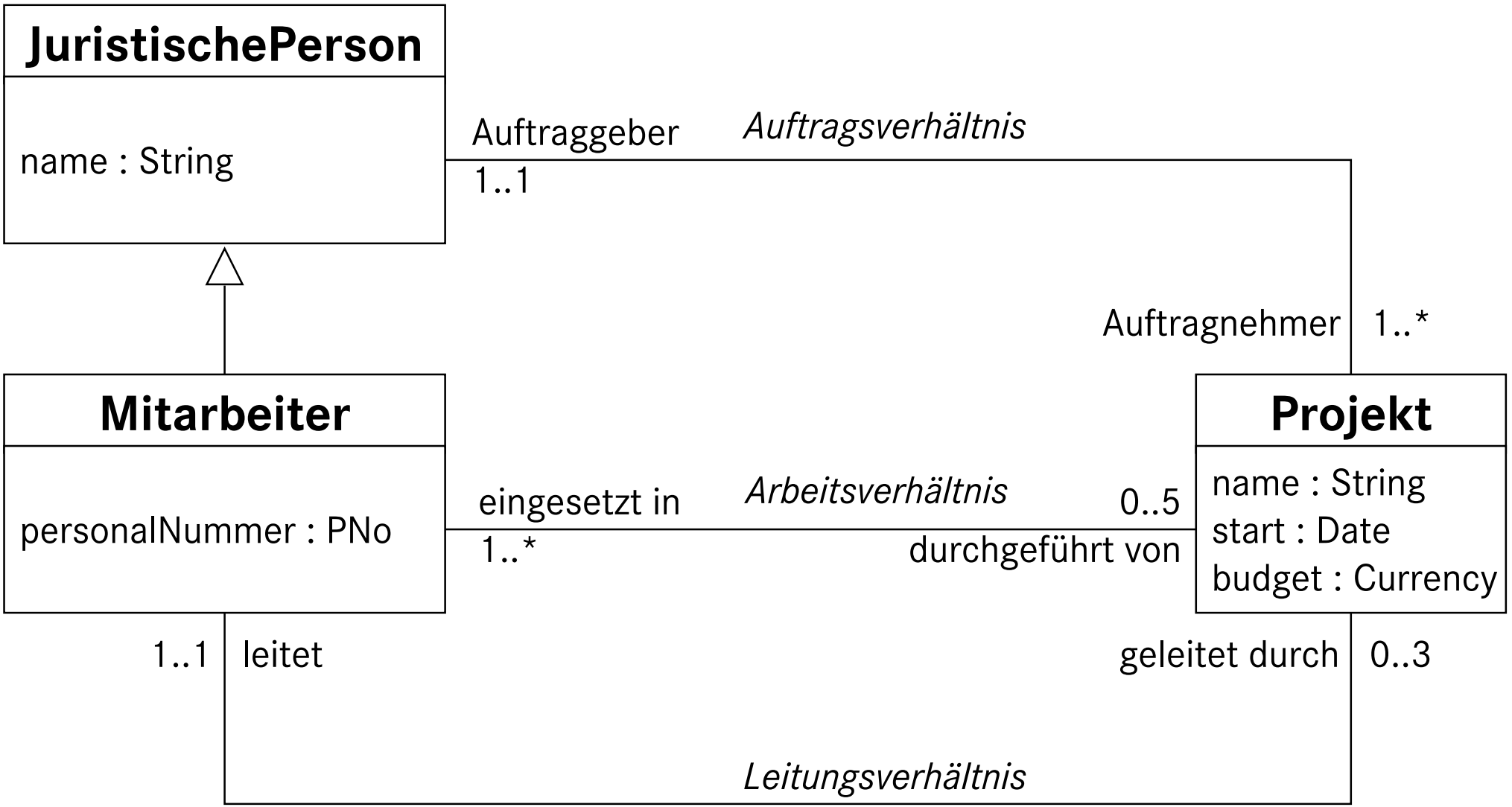
Realisierung:

- Document Type Definitions für UML und MOF
- DTD-Design-Prinzipien zur Erstellung eigener XML/XMI-Vokabulare

XML Metadata Interchange: Verhältnis zu anderen Standards



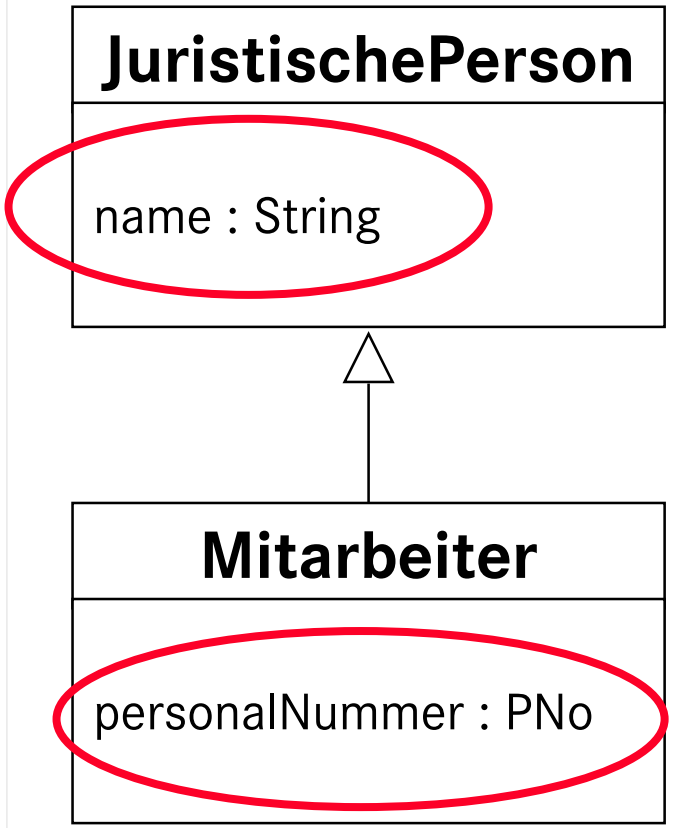
XML-Schemaerzeugung: ... am Beispiel



XML-Schemaerzeugung: Nutzbare UML-Konzepte

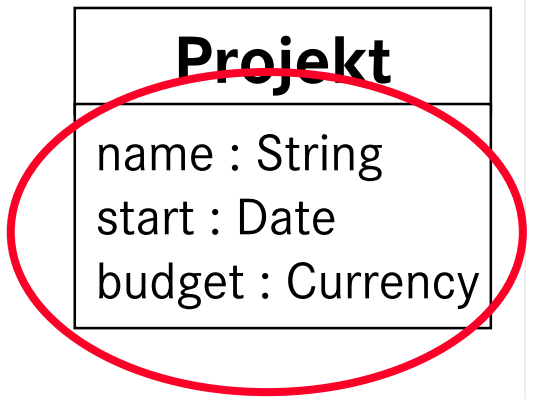
- Klassen
  - Vererbungsstrukturen
  - Assoziationsklassen
- Attribute
  - Konstanten
  - Vorgabewerte
  - Kardinalitäten (d.h. optionale und mengenwertige Attribute)
- Datentypen (skalare und komplxe)
  - vordefinierte (durch UML, MOF oder CORBA)
  - Anwender-definierte
- Assoziationen
  - Navigierbarkeit
  - Kardinalitäten

XML-Schemaerzeugung: Business-Regeln – Datentypen

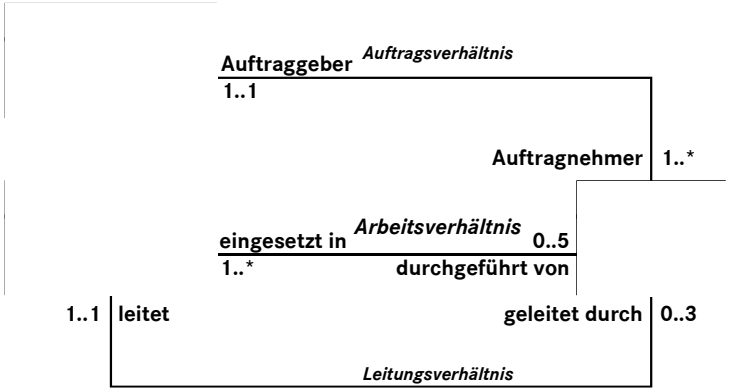


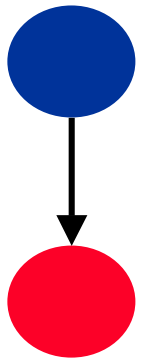
Erzeugung der „korrekten“ Datentypen

- Auswahl des best passenden, gemäß der modellierten Semantik (String, Date)
- Einschränkungen bestehender Datentypen (Currency)
- Erzeugung problemspezifischer eigener Datentypen (PNo)
- Strukturierung komplexer Datentypen (Vererbung)



XML-Schemaerzeugung: Business-Regeln – Strukturen

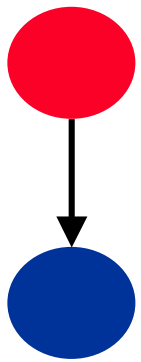




```
<JuristischePerson>
    <Projekt>
        ...
    </Projekt>
</JuristischePerson>
```

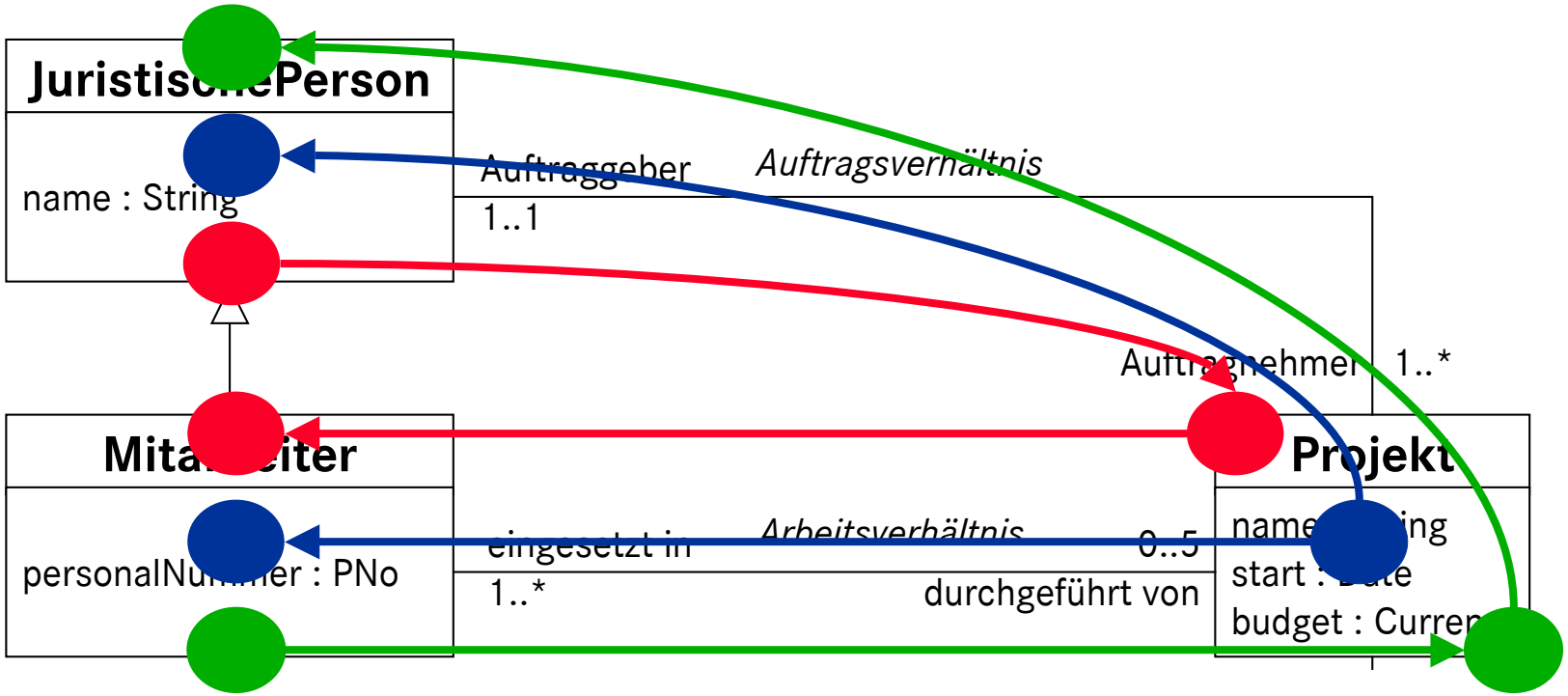
Erzeugung der „korrekten“ Strukturen

- Problem:
  Netz-artige Strukturen lassen sich im Allgemeinen nicht eindeutig in Baum-artige XML-Strukturen überführen
- Lösung:
  Einführung eines zusätzlichen Freiheitsgrades



```
<Projekt>
    <JuristischePerson>
        ...
    </JuristischePerson>
</Projekt>
```

XML-Schemaerzeugung: Business-Regeln – Strukturen



JuristischePerson
Projekt
Mitarbeiter

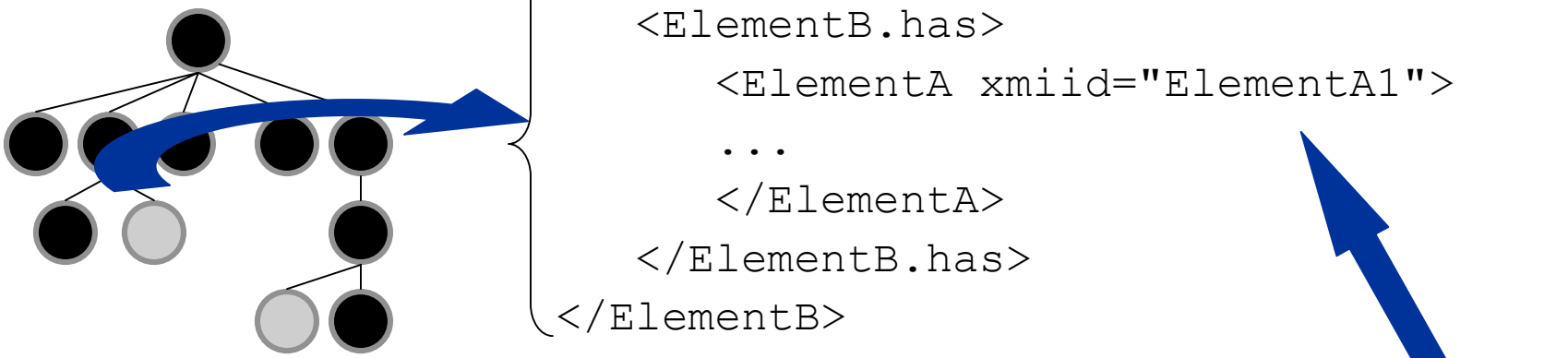
Projekt
JuristischePerson Mitarbeiter

Mitarbeiter
Projekt
JuristischePerson

XML-Schemaerzeugung: Redundanz und ihre Kontrolle

Grundidee:

- Alle Charakteristika (Attribute und Kindelemente) eines Elements werden als optional deklariert
- Jedes Element erhält zusätzlich ein identifizierendes Attribut, Referenzierung durch *simple XLink*



```
<ElementB>
    <ElementB.has>
        <ElementA xmiid="ElementA1">
        ...
        </ElementA>
    </ElementB.has>
</ElementB>

<ElementC>
    <ElementC.has>
        <ElementA xlink:type="simple"
                  xlink:href="ElementA1" . . .
```

Zusammenfassung

- Stabiler, dokumentierter und erprobter Algorithmus zur Erzeugung beliebiger XML-Vokabulare
- Flexibilität durch Wiederholbarkeit des Erzeugungsprozesses bei veränderten Anforderungen
- Geschwindigkeit im Entwicklungsprozeß durch automatische Erzeugung der benötigten XML-Vokabulare aus existierenden Entwicklungsdokumenten (UML Klassendiagramme)
- Kohärenz durch Rückgriff auf selbe Dokumentationsbasis und Modellbasis wie programmiersprachlicher Entwicklungsprozeß
- Korrektheit durch Berücksichtigung der im UML-Modell formulierten Zusammenhänge und Konsistenzbedingungen
- Einheitlicher Stil durch automatischen Generierungsprozeß, der sich am Business-Vokabular des UML-Diagramms orientiert
- Integration in beliebige Entwicklungsprozesse durch natürliche Plazierung nach dem objektorientierten Klassenentwurf
- Wiederverwendung sowohl des Klassenentwurfes, als auch dem Modellierungswissen

Referenzen

XML-Schema:

- Part 0: Primer: http://www.w3.org/TR/xmlschema-0/
- Part 1: Structures: http://www.w3.org/TR/xmlschema-1/
- Part 2: Datatypes: http://www.w3.org/TR/xmlschema-2/

XMI @ OMG:

http://www.omg.org/xml

XMI @ IBM:

http://www.software.ibm.com/ad/features/xmi.html

XMI @ XML.COM:

http://www.xml.com/xml/pub/n/New_XML-based_OMG_standard:_XMI

IBM's XMI-Toolkit:

http://www.alphaworks.ibm.com/tech/xmitoolkit

Dieser Vortag, seine Beispiele und vertiefende Hintergrundinformation:

http://www.jeckle.de