DAIMLERCHRYSLER

XML in der objektorientierten Entwicklung

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

Gliederung

I. Design Time

Einsatz von XML im Bereich der Analyse und des Designs objektorientierter Anwendungen

II. Build Time

Einsatz der XML im Bereich der Realisierung objektorientierter Anwendungen

III. Run Time

Einsatz der XML im Bereich der Ausführung objektorientierter Anwendungen

Gliederung

Run Time

- XML-Schnittstellen und APIs
  - DOM
  - SAX
  - XML Data Binding (JSR-31, Castor, JAXB)

Build Time

- Code Generierung
- Dokumentation (Web-Publishing)

Design Time

- UML / XMI
- SVG (UML2 Diagram Interchange)
- XML Schema

Design Time

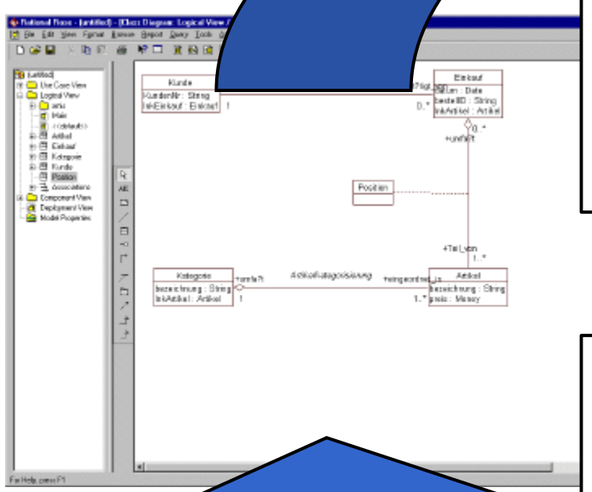
- Unified Modeling Language (UML) – in 10 Punkten
  - Graphische Notation
  - Vereinigt einige Vorgängeransätze (u.a. OMT, OOSE, Use Cases, ...)
  - Standard der Object Management Group (OMG)
  - Abbildung eines (technischen) Systems und seiner Umwelt
  - Bietet verschiedene Sichten auf das System (u.a. statisch, dynamisch, ...)
  - Durch den Anwender erweiterbare Notation (Profiles)
  - Konsistenzgarantierende Einschränkungen können mit formaler Sprache (Object Constraint Language (OCL)) beschreiben werden
  - Definiert keinen Entwicklungsprozeß
  - Definiert ein eigenes Metamodell und ist basierend auf diesem in die *Model Driven Architecture* der OMG eingebettet
  - ... inzwischen weit verbreitet

Design Time

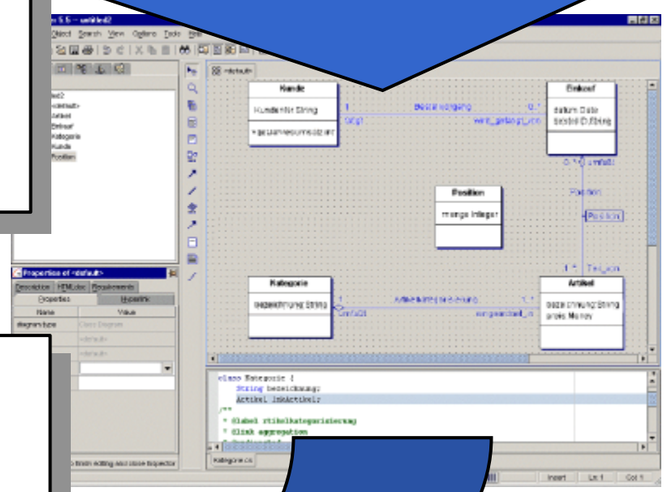
- XML Metadata Interchange (XMI) – in 10 Punkten
  - Format zur Darstellung von UML- und MOF-Modellen
    (Damit eine Möglichkeit UML in XML zu speichern!)
  - XML-basiert (Version 1.x: DTD, Version 2: W3Cs XML Schema)
  - Prozeß zur Erzeugung neuer XMI-Vokabulare aus MOF-basierten Metamodellen
  - Standard der Object Management Group (OMG)
  - Bestandteil der *Model Driven Architecture*
  - Modellaustausch zwischen CASE-Werkzeugen
  - Möglichkeit zur Online-Kopplung heterogener Werkzeuge
  - Langzeitspeicherung von Modellen
  - Weiterverarbeitung der XML-basierten Darstellung
  - ... inzwischen durch fast jedes UML-Werkzeug angeboten

Design Time

- UML-Modellaustausch mit XMI



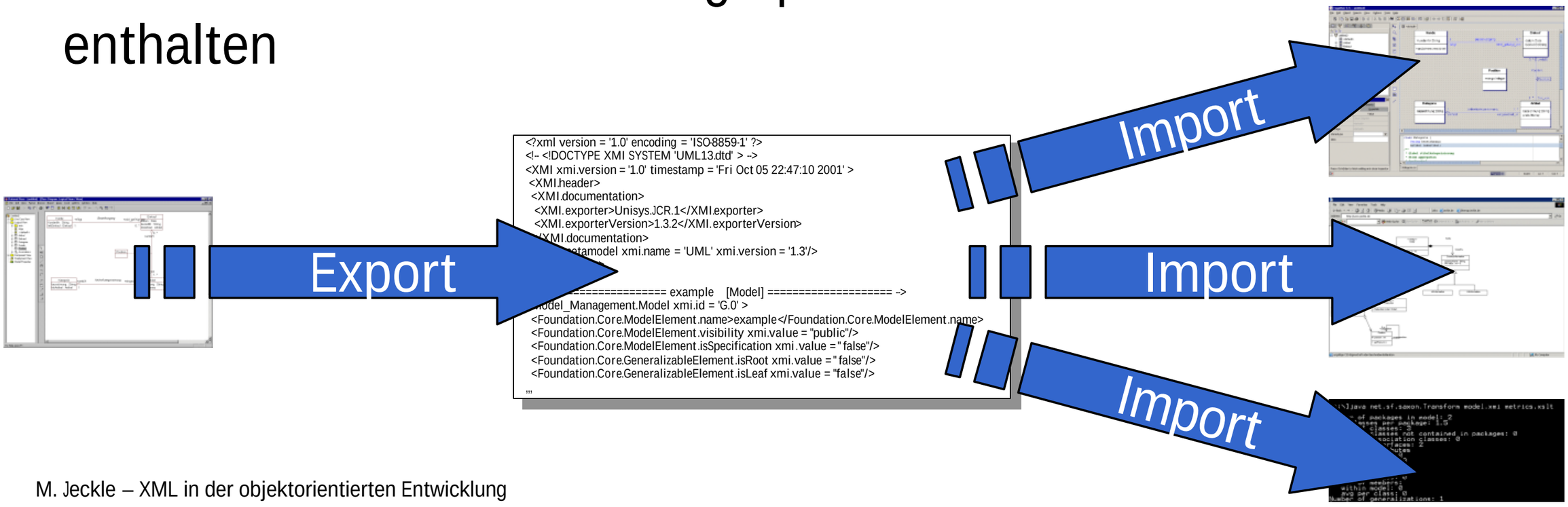
```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<!-- <!DOCTYPE XMI SYSTEM 'UML13.dtd' > -->
<XMI xmi.version = '1.0' timestamp = 'Fri Oct 05 22:47:10 2001' >
 <XMI.header>
  <XMI.documentation>
   <XMI.exporter>Unisys.JCR.1</XMI.exporter>
   <XMI.exporterVersion>1.3.2</XMI.exporterVersion>
  </XMI.documentation>
  <XMI.metamodel xmi.name = 'UML' xmi.version = '1.3'/>
 </XMI.header>
<XMI.content>
<!-- ==================== example    [Model] ==================== -->
<Model_Management.Model xmi.id = 'G.0' >
  <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"/>

,,,
```



```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<!DOCTYPE XMI SYSTEM 'UMLX13.dtd' >
<XMI xmi.version = '1.0'>
<XMI.header>
<XMI.documentation>
<XMI.exporter>Together</XMI.exporter>
<XMI.exporterVersion>5.0</XMI.exporterVersion>
</XMI.documentation>
<XMI.metamodel xmi.name = 'UML' xmi.version = '1.3'/>
</XMI.header>
<XMI.content>
<Model_Management.Model xmi.id = 'txmiid1' >
<Foundation.Core.ModelElement.name>example</Foundation.Core.ModelElement.name>
  <Foundation.Core.ModelElement.visibility xmi.value = 'private'/>
  <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>
```

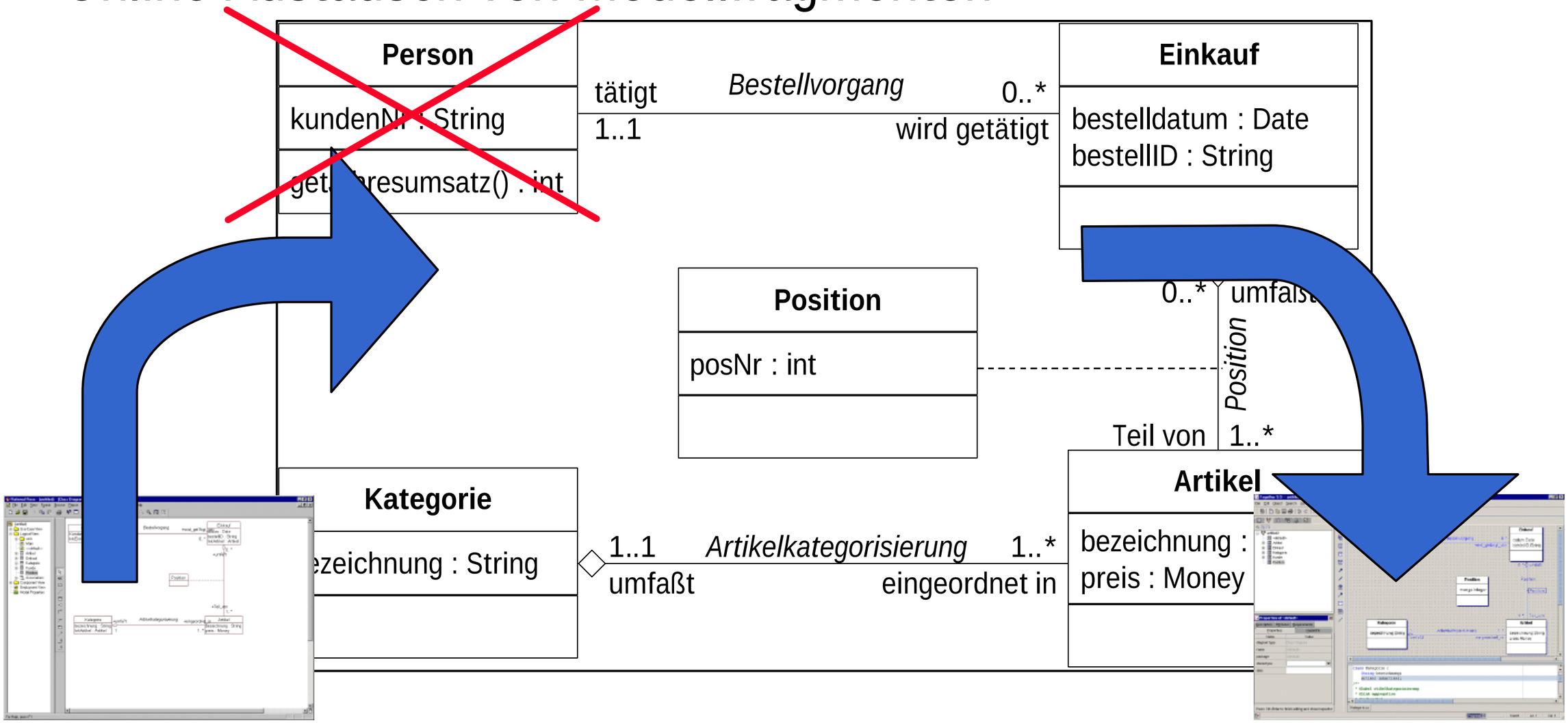
Design Time

- UML-Modellaustausch mit XMI
  - XMI v1.x enthält alle modellrelevanten Informationen (Klassen, Attribute, Operationen, Assoziationen, etc.) vollständig ohne auf werkzeugspezifische Vorgabewerte zurückzugreifen
  - XMI v2.x wird zusätzlich die graphische Modellinformation enthalten



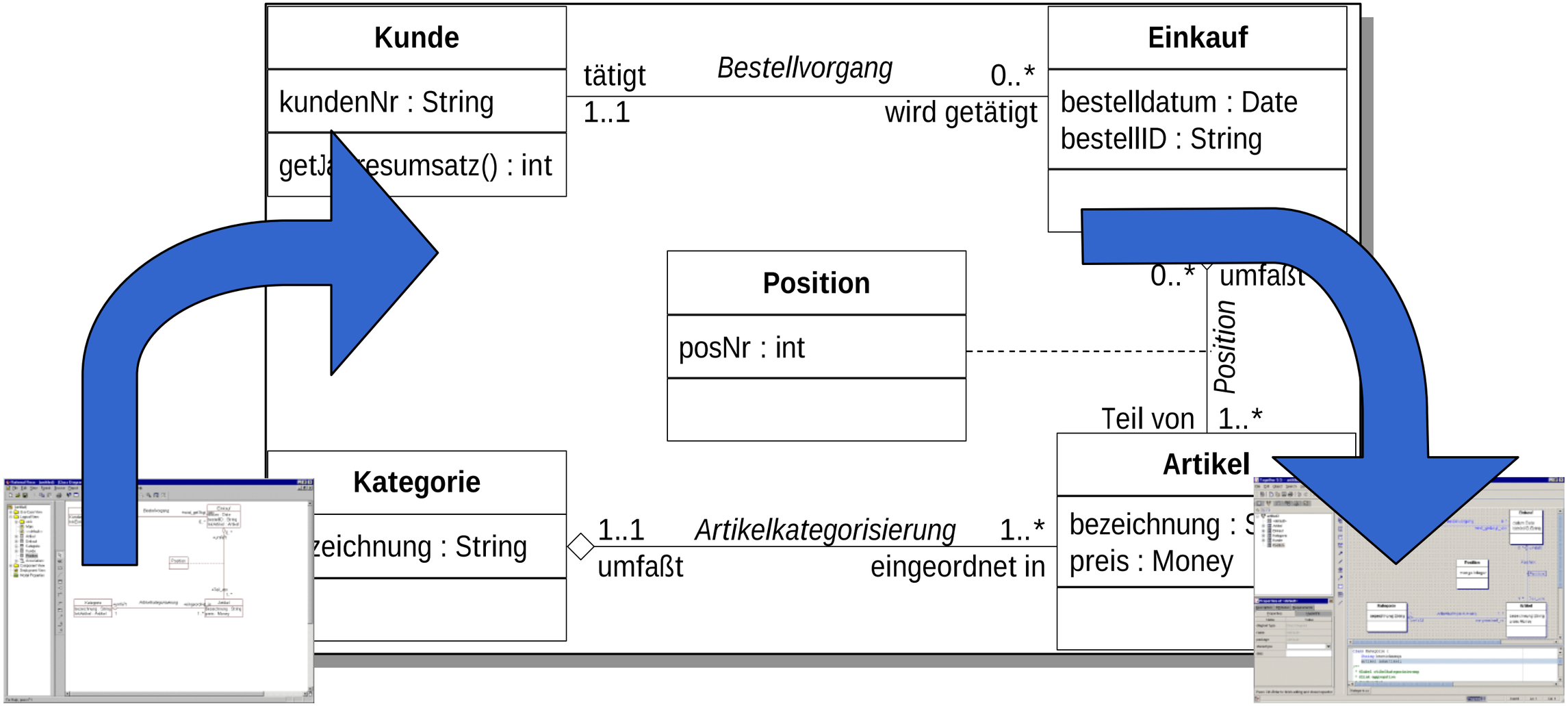
Design Time

- UML-Modellaustausch mit XMI
  - Online-Austausch von Modellfragmenten



Design Time

- UML-Modellaustausch mit XMI
  - Online-Austausch von Modellfragmenten



Design Time

- UML-Modellaustausch mit XMI
  - Online-Austausch von Modellfragmenten

```
<XMI>
  <XMI.header/>
  <XMI.difference>
    <XMI.replace>
      oundation.Core.Class xmi.id="PersonClass"/>
      ndation.Core.Class xmi.id="KundeClass">
        dation.Core.ModelElement.name>Kunde</Foundation.Core.Mo
        ation.Core.ModelElement.visibility xmi.value="public"/>
        ndation.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.Class.isActive xmi.value="false"/>
      </Foundation.Core.Class>
    </XMI.replace>
  </XMI.difference>
</XMI>
```

Design Time

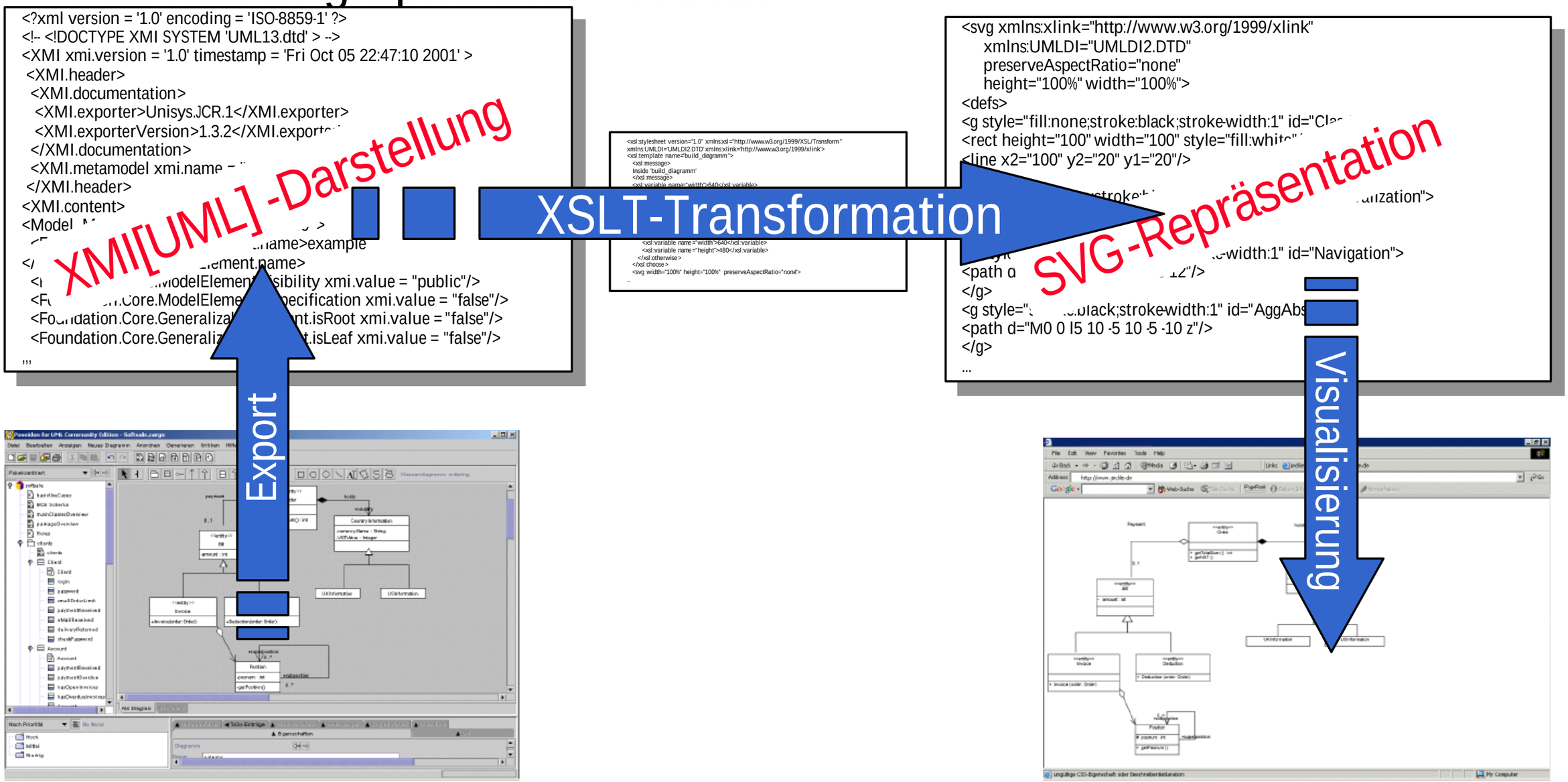
- UML-Modellaustausch mit XMI
  - Austausch graphischer Modelle
    - Integration der Präsentationsinformation in UML v2 und XMI v2
    - Nutzung des XML-Standardvektorformates *Scalable Vector Graphics (SVG)* zur werkzeugunabhängigen Darstellung

CASE-Werkzeug

SVG-Darstellung im Web-Browser

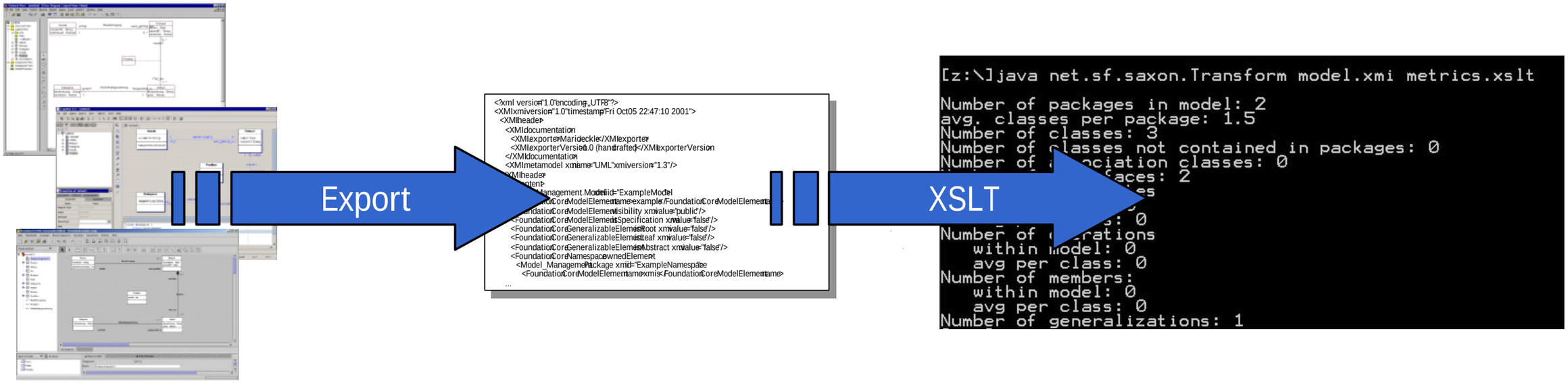
Design Time

- UML-Modellaustausch mit XMI
  - Austausch graphischer Modelle



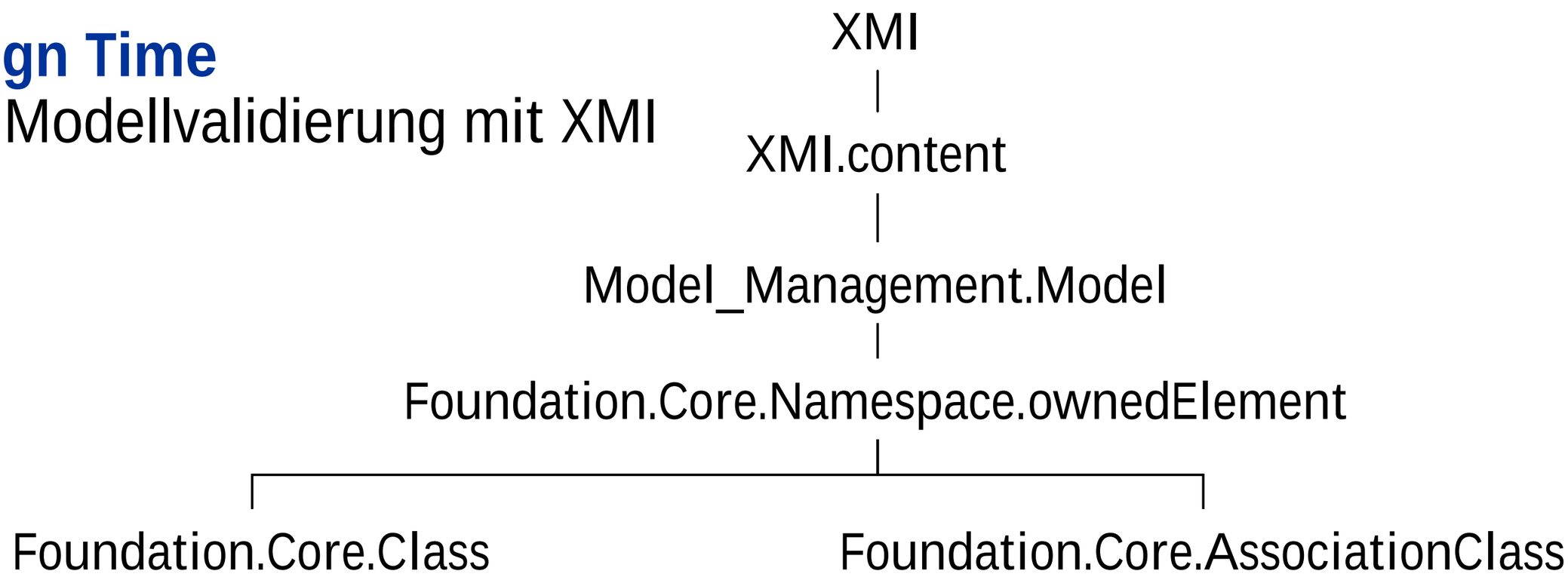
Design Time

- UML-Modellvalidierung mit XMI
  - Grundidee: Operation auf werkzeugneutraler UML-Darstellung
  - Anwendungsfelder:
    - Design Guidelines
    - Metriken
  - Realisierung: XSLT-Transformation auf XMI-Dateien
  - Beispielimplementierung → http://www.jeckle.de



Design Time

- UML-Modellvalidierung mit XMI

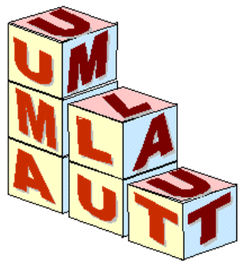


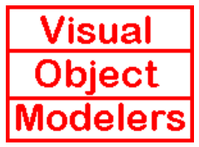

```
<xsl:template name="numberOfClasses">
  <xsl:variable name="noClasses" select="count(//Foundation.Core.Class)"/>
  Number of classes: <xsl:value-of select="$noClasses"/>
  Number of classes not contained in packages:
  <xsl:value-of select="count(/XMI/XMI.content/Model_Management.Model/Foundation.Core.Namespace.ownedElement/
  Foundation.Core.Class[@xmi.id])"/>
  Number of association classes: <xsl:value-of select="count(//Foundation.Core.AssociationClass)"/>
</xsl:template>
```

Design Time

- UML-Werkzeuge mit XMI-Unterstützung







Details: http://www.jeckle.de/umltools.html

Design Time

- XML-Schema – in 10 Punkten
  - Erlaubt die Erstellung regulärer kontextfreier Grammatiken für 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

Design Time

- XML-Schema – in 10 Punkten
  - Integriert die wichtigsten konkurrierenden Vorgängeransätze
  - Seit 2001-05-02 W3C Recommendation
  - Basis aller W3C-Standards der zweiten Generation
    (XPath v2.0, XSLT v2.0, XHTML v2.0, SOAP v1.2 / XMLP, ...)
  - Werkzeugunterstützung verfügbar
    (siehe u.a. auch: http://www.jeckle.de/xml/schema.html)
  - Erster Schritt der Schema-Bestrebungen, weitere werden folgen ...

Design Time

- XML-Schema – Mächtigkeit
  - Strukturell: Attribute und Elemente (wie in DTDs)
  - Namespace-Unterstützung
  - Wiederverwendungsunterstützung
  - Vererbung: Restriktion und Erweiterung
  - Schemafragmente: Typen, Attribut- und Elementgruppen
  - Substitution
  - Erweiterter Schlüsselmechanismus
  - NULL-Werte

Design Time

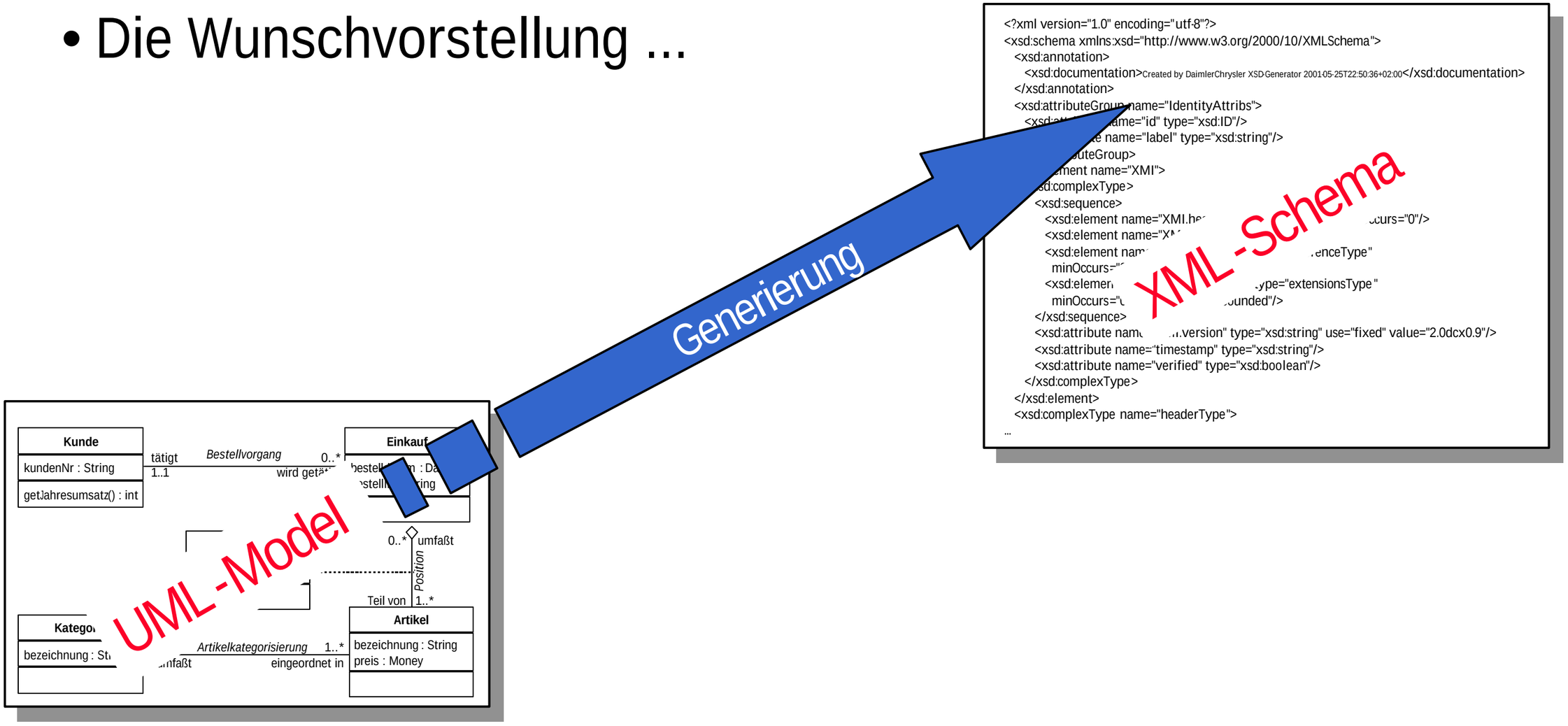
- XML-Schema – Mächtigkeit
  - Atomare Datentypen (int, float, boolean, ...)
  - Anwenderdefinierte
    - atomare skalare Datentypen (simpleType)
    - Einschränkung des Wertebereichs (Domänenrestriktion)
    - lexikalische Muster (reguläre Ausdrücke)
    - Aufzählungstypen
    - Mengentypen
    - komplexe Datentypen (complexType)

Design Time

- XML-Schema – im Kontext objektorientierter Entwicklung
  - XML-Schema vereinigt viele Merkmale verfügbarer OO-Sprachen
  - XML-Schema ist jedoch keine Datenmodellierungssprache
  - ... ebenso wenig eine Implementierungssprache!
  - Sinnvoll: Prozeßeinsatz von XML-Schema *gemeinsam* mit OO-Modellierungssprache
  - Einsatzszenario: XML-Schema zur Modellierung der textbasierten Ein- und Ausgabeformate, evtl. auch zur Persistenzmodellierung
  - Notwendig: Abgestimmtes Vorgehen um Redundanz zu vermeiden und Zeitaufwände zu minimieren
  - Standardisiert: Transformationsalgorithmus UML (v1.x) → DTD und XML-Schema

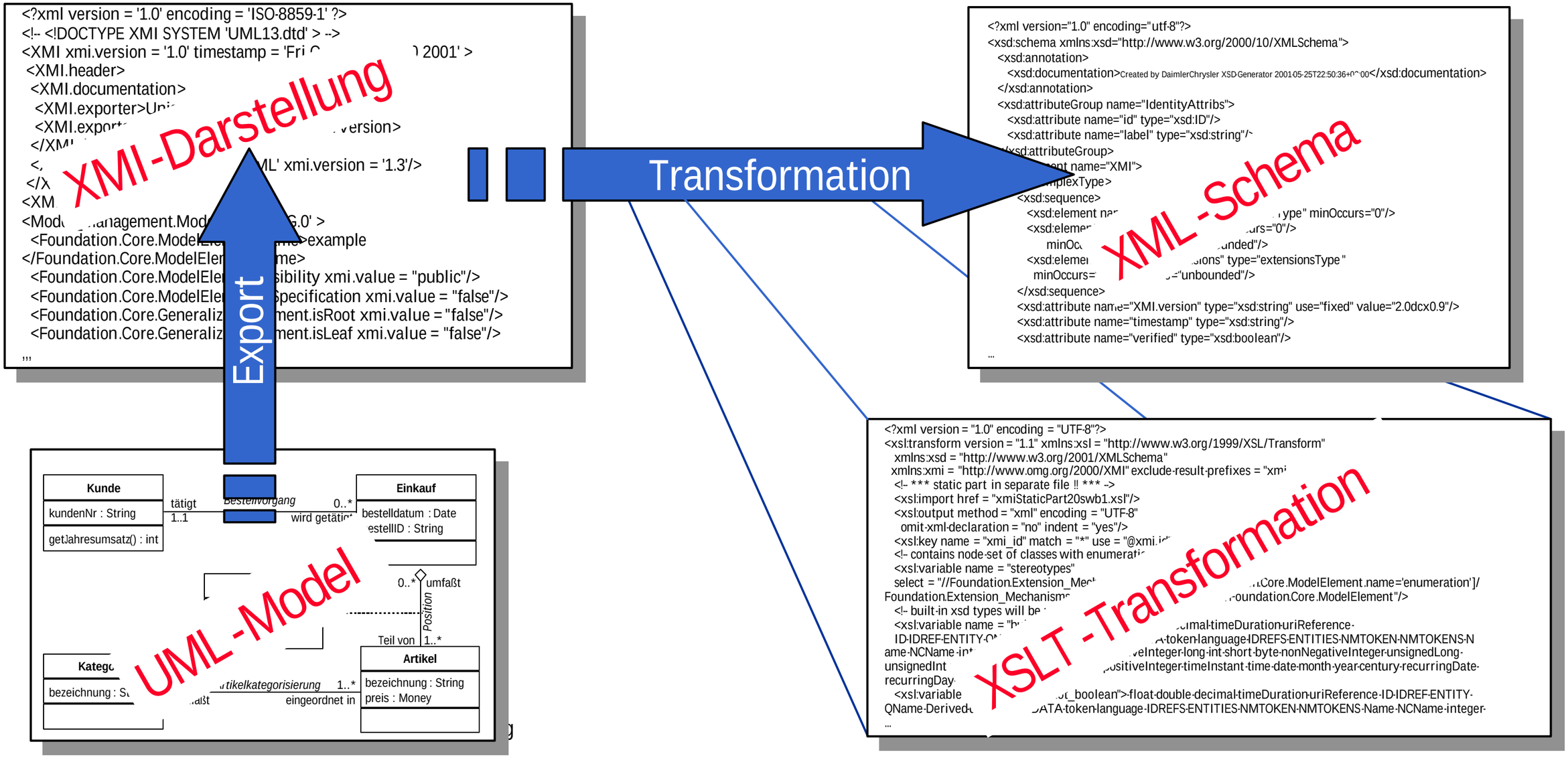
Design Time

- XML-Schema – im Kontext objektorientierter Entwicklung
  - Ansatz: Generierung der XML-Schemastrukturen aus OO-Modell
    - Die Wunschvorstellung ...



Design Time

- XML-Schema – im Kontext objektorientierter Entwicklung
  - Ansatz: Generierung der XML-Schemastrukturen aus OO-Modell



Build Time

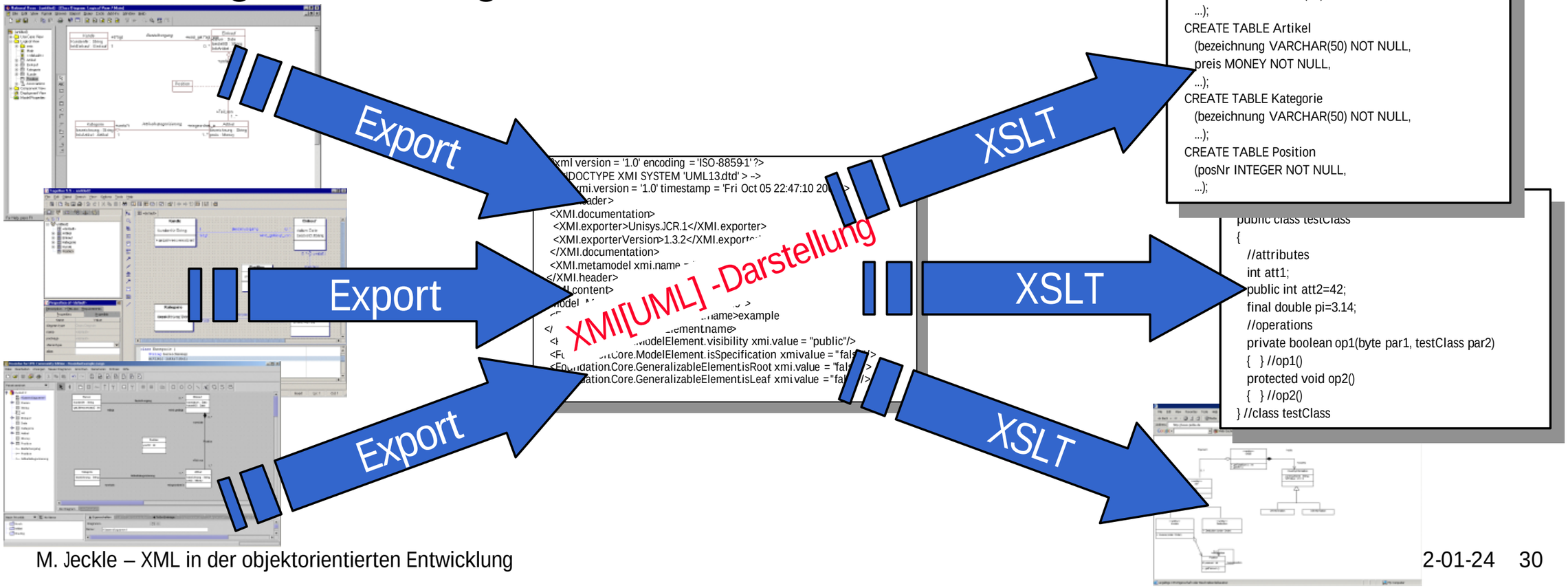
- Codegenerierung
  - Grundidee: Werkzeugunabhängige Quellcodeerzeugung aus objektorientierten Modellen
  - Konkreter: Allgemeingültige und konsistente (d.h. werkzeugtransparente) Quellcodeerzeugung aus Modellen derselben Modellierungssprache (z.B UML)
  - Hintergrund / Motivation:
    - Existierende und angebotene Codegenerierung unzureichend
    - Gewünschte Zielsprache nicht unterstützt

Build Time

- Codegenerierung
  - Anwendungsszenario:
    - Codegenerierung für programmiersprachliche Umsetzung
    - Datendefinitionsschemata
    - Prototypengenerierung
    - Testdatenerzeugung
  - Notwendig: Einheitliche Werkzeugschnittstelle
    - entweder als CASE-Tool API
    - oder als Exportformat

Build Time

- Codegenerierung
  - Lösungsmöglichkeit:
    - Verwendung von XMI zur werkzeugunabhängigen Modellrepräsentation
    - Codegenerierung mit XSLT



Build Time

- Codegenerierung
  - Beispiel: Java-Codegenerierung (Quellcode → http://www.jeckle.de)

XSLT-Transformation

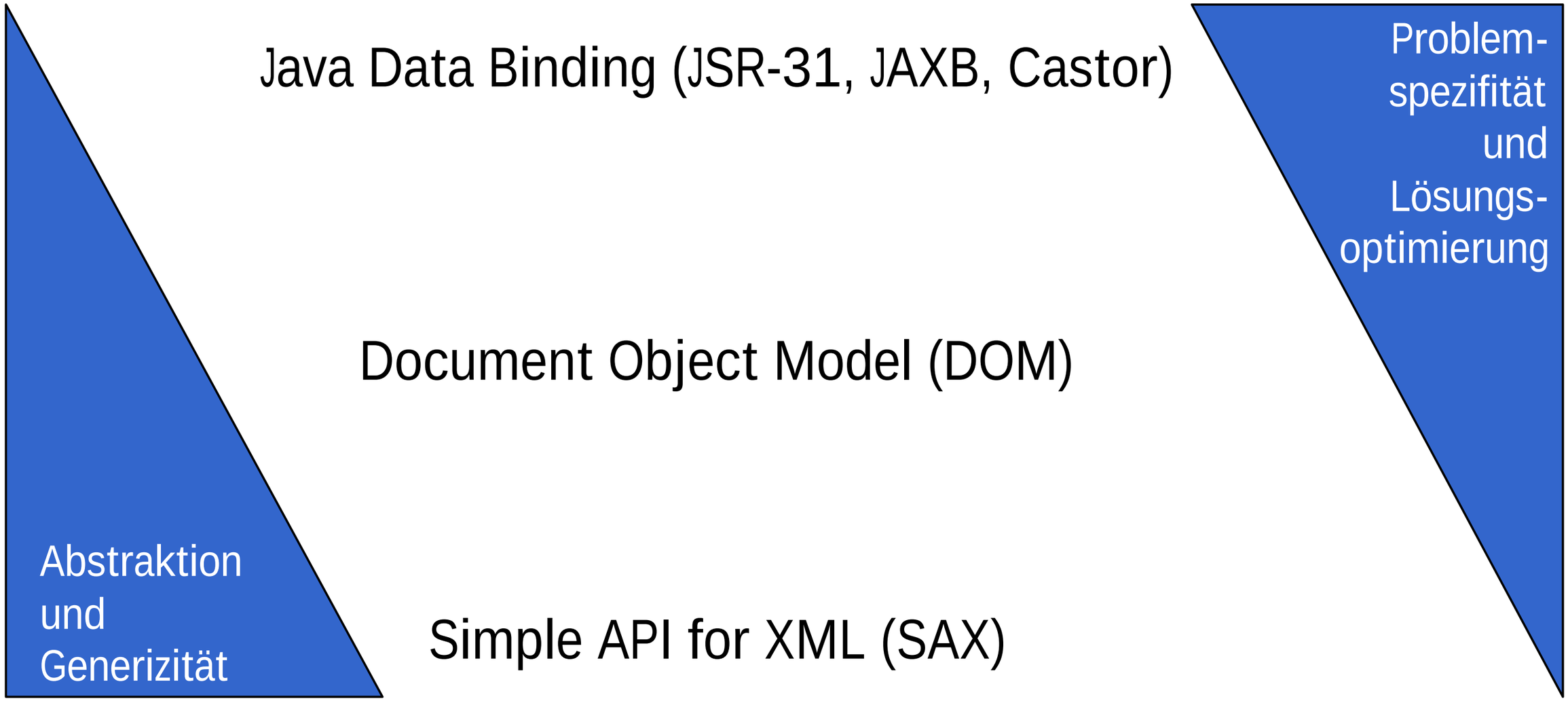
```
<xsl:template match="Foundation.Core.Class">
<xsl:value-of select="Foundation.Core.ModelElement.visibility/@xmi.value"/>
<xsl:if test="Foundation.Core.GeneralizableElement.isAbstract/@xmi.value='true'">
abstract
</xsl:if>
<xsl:if test="Foundation.Core.GeneralizableElement.isLeaf/@xmi.value='true'">
final
</xsl:if>
class <xsl:value-of select="Foundation.Core.ModelElement.name"/>
{
//attributes
<xsl:apply-templates select="descendant::Foundation.Core.Attribute"/>
//operations
<xsl:apply-templates select="descendant::Foundation.Core.Operation"/>
} //<xsl:value-of select="Foundation.Core.ModelElement.name"/>()
</xsl:template>
```

Ergebnisschablone

```
public abstract final class c1
{
//attributes

//operations

}  //class c1
```

Run Time

- Schnittstellen und APIs zur programmiersprachlichen XML-Verarbeitung



Run Time

- Schnittstellen und APIs zur programmiersprachlichen XML-Verarbeitung

Abstraktion und Generizität

Java Data Binding (JSR-31, JAXB, Castor)
- (aus DTD / XSD) generierte Speicherdarstellung
- Java-Speichersicht
- Generierte Zugriffsmethoden

Document Object Model (DOM)
- Abstrahierte Speicherdarstellung
- Baumartige Sichtweise
- Objektorientiere Schnittstelle

Simple API for XML (SAX)
- Lineare Verarbeitung
- Ereignisbasiert
- Pogrammierparadigmenneutral

Problem-spezifität und Lösungs-optimierung

Run Time

- Simple API for XML (SAX)
  - Entwickelt durch Mitglieder der Mailingliste xml-dev auf Initiative von David Megginson
  - Zusammenstellung sprachunabhängiger Schnittstellen
  - Leichtgewichtig:
    - einfach (in eigenen Programmen) einzusetzen
    - einfach (in beliebige Programmiersprachen) umzusetzen
  - Serieller linearer ereignisbasierter Parsingvorgang
  - Definiert und erfordert keine Speicherstrukturen
  - SAX ist eine Schnittstelle für Parser, kein Parser!
  - Laufzeitverhalten skaliert linear mit Dokumentgröße
  - JAXP und J2SE v1.4 enthält eine Implementierung des aktuellen Standes SAX2

Run Time

- Simple API for XML (SAX)

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

public class SAXExample extends DefaultHandler
{
   public void startDocument()
   {
     //...
   } //startDocument()

public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
{
   // ...
} //startElement()
public static void main (String args[]) throws Exception
   {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      SAXParser sp = spf.newSAXParser();
      sp.parse( args[0], new SAXExample() );
   } //main()
}//class SAXExample
```

Implementierung der Standardschnittstelle

Run Time

- Simple API for XML (SAX)

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

public class SAXExample extends DefaultHandler
{
   public void startDocument()
   {
     //...
   } //startDocument()

public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
{
   // ...
} //startElement()
public static void main (String args[]) throws Exception
   {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      SAXParser sp = spf.newSAXParser();
      sp.parse( args[0], new SAXExample() );
   } //main()
}//class SAXExample
```

SAX-Parserimplementierung
JAXP / Java 1.4

Run Time

- Simple API for XML (SAX)

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

public class SAXExample extends DefaultHandler
{
  public void startDocument()
  {
    //...
  } //startDocument()

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

Durch SAX in ihrer Signatur vorgegebene Callback-Methoden

Run Time

- Simple API for XML (SAX)

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

public class SAXExample extends DefaultHandler
{
  public void startDocument()
  {
    //...
  } //startDocument()

public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
{
  // ...
} //startElement()
public static void main (String args[]) throws Exception
  {
    SAXParserFactory spf = SAXParserFactory.newInstance();
    SAXParser sp = spf.newSAXParser();
    sp.parse( args[0], new SAXExample() );
  } //main()
}//class SAXExample
```

Erzeugung eines SAX-basierten Parsers und prüfen eines XML-Dokuments

Run Time

- Document Object Model (DOM)
  - W3C standardisierte API für HTML, XML, CSS, etc.
  - Definiert programmiersprachenunabhängige baumartige objektorientierte Speicherstruktur für XML mit Operationen zur Navigation und Manipulation
  - Speicheraufwand zur Verwaltung des gesamten XML-Dokuments im Hauptspeicher teilweise sehr hoch
  - Mit *JDOM* existiert eine für Java optimierte moderne Implementierung
  - Verfügbar: Diverse Implementierungen: Java 1.4, JDOM, DOM4J, ...

Run Time

- Document Object Model (DOM)

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMExample
{
  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();

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

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

    theRootElement.appendChild( aNewElement );

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

Implementierung der Standardschnittstellen

Run Time

- Document Object Model (DOM)

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMExample
{
  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();

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

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

    theRootElement.appendChild( aNewElement );

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

Erzeugung des Parsers und Einlese-vorgang

Run Time

- Document Object Model (DOM)

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMExample
{
  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();

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

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

    theRootElement.appendChild( aNewElement );

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

Extraktion eines Elements

Run Time

- Document Object Model (DOM)

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMExample
{
  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();

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

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

    theRootElement.appendChild( aNewElement );

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

Hinzufügen eines (neuen) Attributs

Run Time

- Document Object Model (DOM)

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMExample
{
  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();

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

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

    theRootElement.appendChild( aNewElement );

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

Erzeugung eines neuen Elements

Run Time

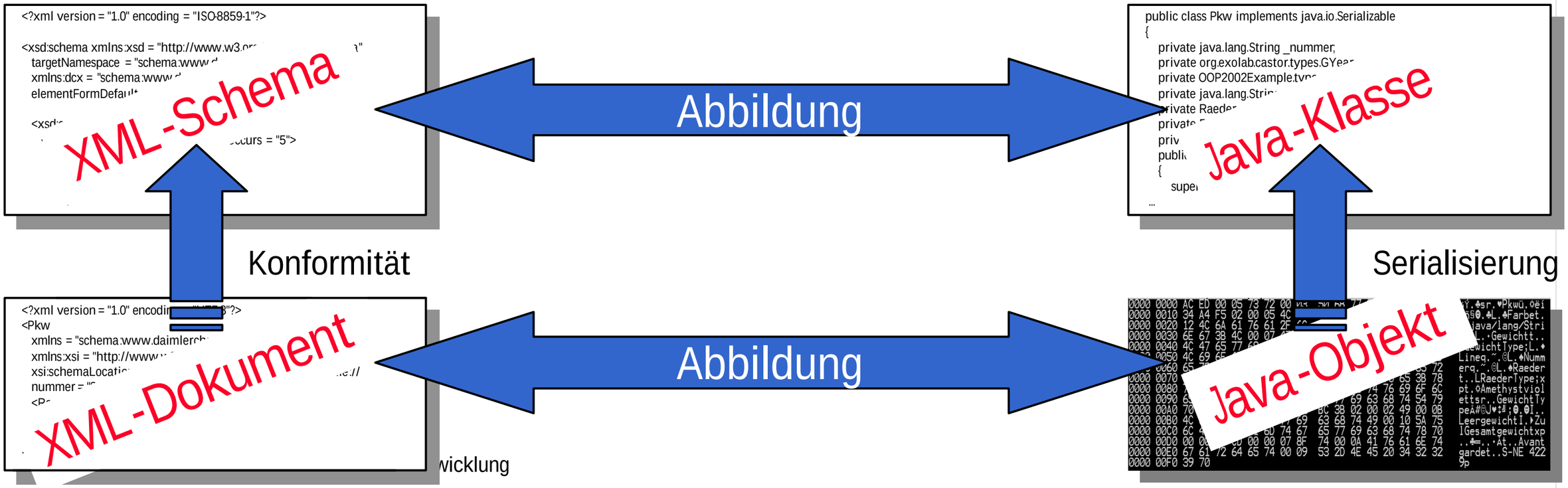
- Document Object Model (DOM)

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMExample
{
  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();

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

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

    theRootElement.appendChild( aNewElement );

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

Anhängen des erzeugten Elements als zusätzliches Kindelement

Run Time

- Java Data Binding
  - Grundidee: Verringerung des *Impedance Mismatch* zwischen Java und XML
  - Technisch: Generierung von Java-Klassendefinitionen aus Grammatik (DTD / XSD) und optional zusätzlich einer Abbildungsdefinition



Run Time

- Java Data Binding

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Pkw
  xmlns = "schema:www.daimlerchrysler.com"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "schema:www.daimlerchrysler.com file:/
  nummer = "S-NE 4229" line = "Avantgarde">
  <Raeder>
    <Rad/>
    <Rad/>
    <Rad/>
    <Rad/>
  </Raeder>
  <Farbe name = "Amethystviolett"/>
  <Gewicht>
    <Leergewicht>1485</Leergewicht>
    <ZulGesamtgewicht>1935</ZulGesamtgewicht>
  </Gewicht>
</Pkw>
```

Run Time

- Java Data Binding

```
import java.io.*;
import org.exolab.castor.xml.MarshalException;
import OOP2002Example.*;
public class example1
{
   public static void main(String[] argv)
   {
      try
      {
         Pkw myPkw = Pkw.unmarshal(new FileReader( argv[0] ));
      } //try
      catch (MarshalException e)
      {
         Exception originatingException = e.getException();
         if (originatingException != null)
         {
            System.out.println(originatingException);
         } //if
      } //catch
 ...
   } //main()
} //class example1
```

- Einlesen einer XML-Quelldatei und Erzeugung der Haupt-speicherobjekte

Run Time

- Java Data Binding

```
...
  //Accessing root's attributes
  System.out.println("Auto-Nummer: "+myPkw.getNummer() );
  System.out.println("PKW-Line: "+myPkw.getLine() );

  //Accessing root's child element
  System.out.println("Autofarbe: "+myPkw.getFarbe().getName() );

  Gewicht myGewicht = myPkw.getGewicht();
  System.out.println("Leergewicht: "+myGewicht.getLeergewicht() );
  System.out.println("Zulässiges Gesamtgewicht: "
    +myGewicht.getZulGesamtgewicht() );
...
```

- Lesender Zugriff auf Attribute und Elemente

Run Time

- Java Data Binding

```
...
  //setting an attribute
  myPkw.setNummer("UL-A 1234");

  //setting a element's content
   myPkw.getGewicht().setLeergewicht(1520);

  //creating an attribute
  myPkw.setBaujahr( new GYear(2002) );

   //creating an element
  //(somewhat complicated due to XML schema's structure)
  Rad myNewRad = new Rad();
  RadTypeItem myRadTypeItem = new RadTypeItem();
  myRadTypeItem.setRad ( myNewRad );
  myPkw.getRaeder().addRadTypeItem( myRadTypeItem );

   //creating an element (the normal way ;)
  myPkw.setHalter( "John Doe" );
...
```

- Schreibender Zugriff auf Attribute und Elemente
- Erzeugung von Elementen

Referenzen

XML Metadata Interchange

- http://www.omg.org

Document Object Model (DOM)

- http://www.w3.org/DOM

Simple API for XML (SAX)

- http://www.saxproject.org/

Java Data Binding

- http://jcp.org/jsr/detail/31.jsp
- http://castor.exolab.org/

Dieser Vortrag, der gezeigte Quellcode und weiterführende Informationen:

- http://www.jeckle.de