TU Dresden

Fakultät Informatik

Institut für Softwaretechnologie II

Methodik zur Codegenerierung

http://www-st.inf.tu-dresden.de/UMLToolset

Hochschullehrer	**Prof. Dr. rer. nat. habil. H. Hußmann**
Betreuer	**Frau Dr. B. Demuth**
Referent	**Sven Obermaier**

Agenda

♦ = Schwerpunkt

1 Thema der Diplomarbeit (♦)

2 Codegenerierungsansätze (♦ ♦ ♦)

- Implementierungsentwürfe
- Beschreibungssprachen

3 Abbildung der UML auf SQL (♦ ♦)

4 XSLT (XSL Transformations) (♦ ♦ ♦ ♦ ♦)

- Einordnung & Eigenschaften
- Das kleine Einmaleins

5 Ausblick (♦)

6 Diskussion (♦ ♦ ♦)

1 Thema der Diplomarbeit

Ziele

1 Thema der Diplomarbeit

ZIELE:

1 Analyse von Codegenerierungsansätzen

2 Generierung von Skripten aus Modellen

3 Prototypisches Beispiel: UML 1.3 → SQL-92

4 Entwicklung eines Musterkatalogs für die Codegenerierung mit XSLT*

*Forderung aus laufender Arbeit entstanden

5 (MDM-Architektur, DTD2J, Handbücher)

2 Codegenerierungsansätze

Anforderungen
Implementierungsentwürfe
Beschreibungssprachen

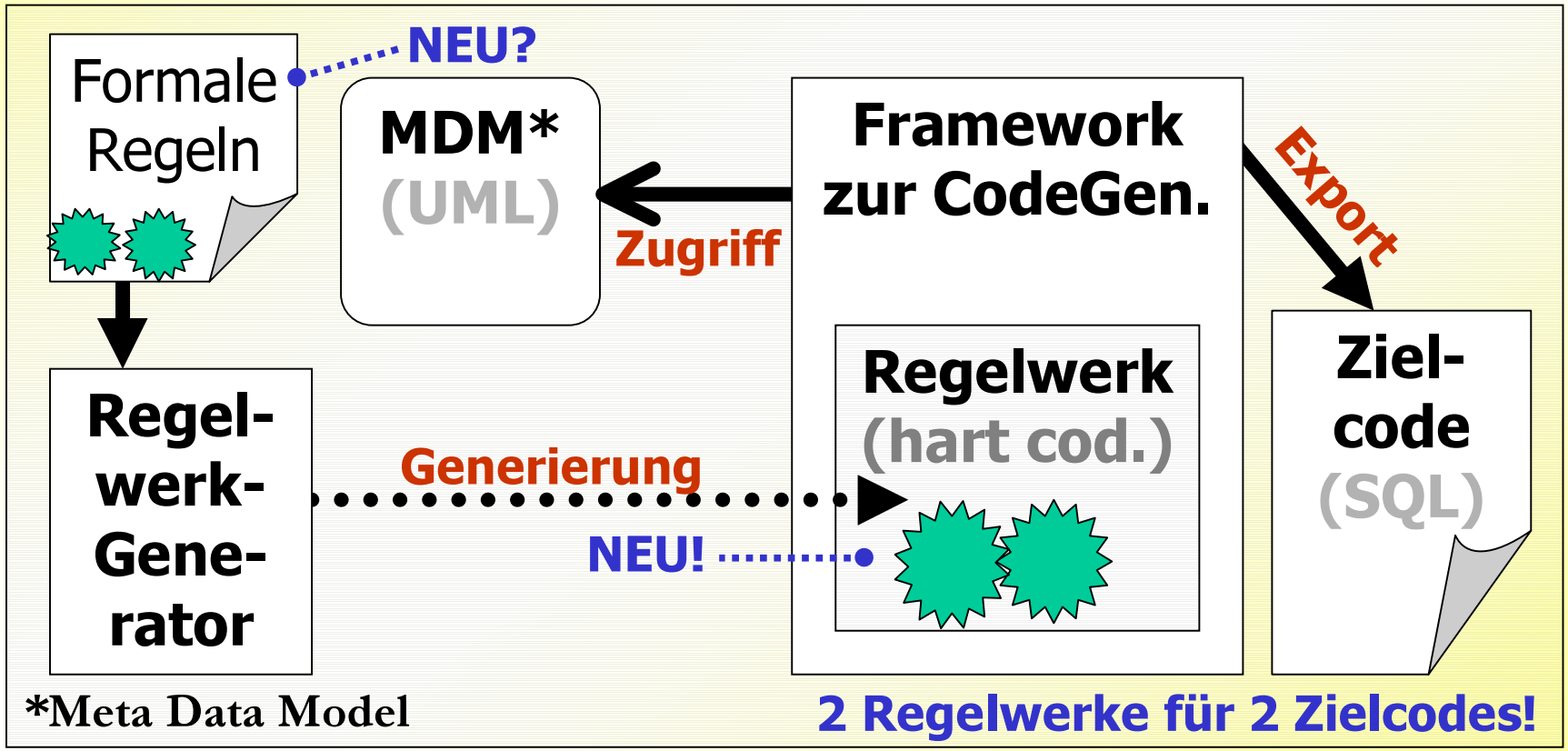
2 Codegenerierungsansätze
Anforderungen

- **Anforderungen an das Framework**
  - **Basierend auf *externen* Abbildungsvorschriften**
  - **Beschreibung der Abbildungsvorschriften**

    unabhängig vom Quell- und Zielcode !
  - **Framework *unabhängig vom Meta-Modell !***
  - **Framework zur Codegenerierung *erweiterbar***

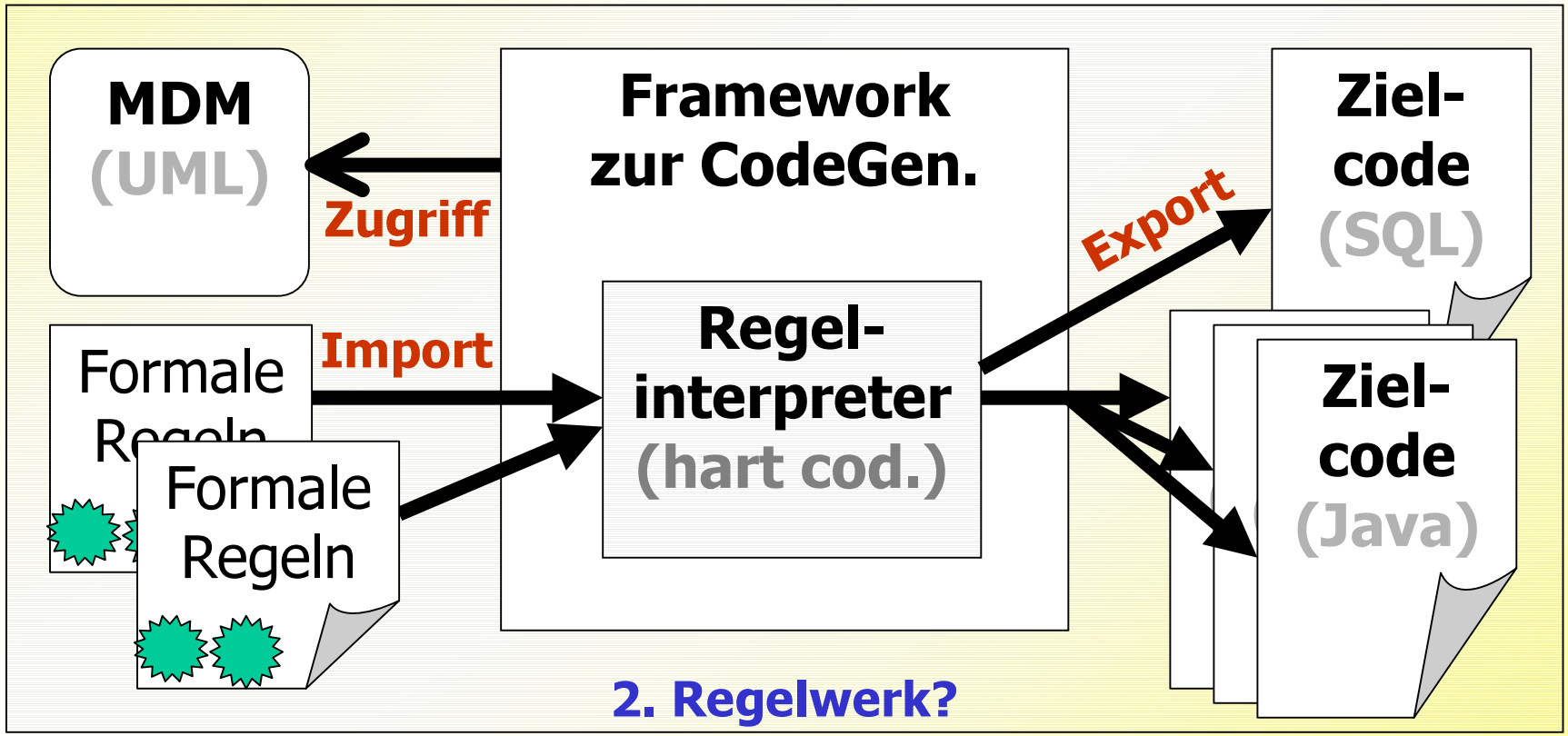
2 Codegenerierungsansätze
Implementierungsentwürfe (1)

- **Compiler-Ansatz** (vereinfachtes Schema):



2 Codegenerierungsansätze
Implementierungsentwürfe (2)

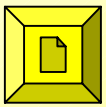
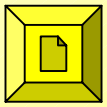
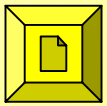
- Interpreter-Ansatz (vereinfachtes Schema):



Als Regelwerk-Generator für Comiler-Ansatz nutzbar!

2 Codegenerierungsansätze Beschreibungssprachen

- Allgemein möglich:
  - „Hart codiert“ ;-)
  - Attributierte Grammatiken
  - Regelbasierte Systeme
- Untersuchte Beschreibungssprachen (regelbasiert):
  - OCL (Object Constraint Language)
  - Meine.DTD (Eigenentwicklung basierend auf XML (Extensible Markup Language))
  - XSLT (Extensible Stylesheet Language Transformations)

3 Abbildung der UML auf SQL

Überblick

Beispiel Assoziation

Schwierigkeiten

3 Abbildung der UML auf SQL
Überblick

- Abbildung trivial?
  - Klasse → Relation
  - Assoziation → Fremdschlüsselbeziehung
  - Generalisierung → Universalrelation, horizontale oder vertikale Zerlegung
- Tücke im Detail!
  - OCL-Ausdrücke
  - Interfaces
  - Wahrung der in UML beschriebenen Modellkonsistenz

3 Abbildung der UML auf SQL
Beispiel Assoziation

- Assoziationstypen (1:1, 1:N, N:M) und Subtypen (Assoz.-end: 0..1, 1, 0..N, 1..N, ...)
- Assoziationsklassen
- Aggregationen
- Spezielle Multiplizitäten (0,1,2 | 0..3,5..12)
- Qualifizierer
- n-ary-Assoziationen
- Xor-Assoziationen

3 Abbildung der UML auf SQL
Schwierigkeiten

- Verlorengegangene Informationen
  - Sichtbarkeiten (public, protected, private)
  - Zugehörigkeit Operationen zu Klassen
  - Navigierbarkeit von Assoziationen
- Package-Struktur
- Lange Zeichenketten (SQL-92 vs. Realität)
- Interfaces
- ...

4 XSLT (XSL Transformations)

Einordnung & Eigenschaften

Das kleine Einmaleins an Beispielen

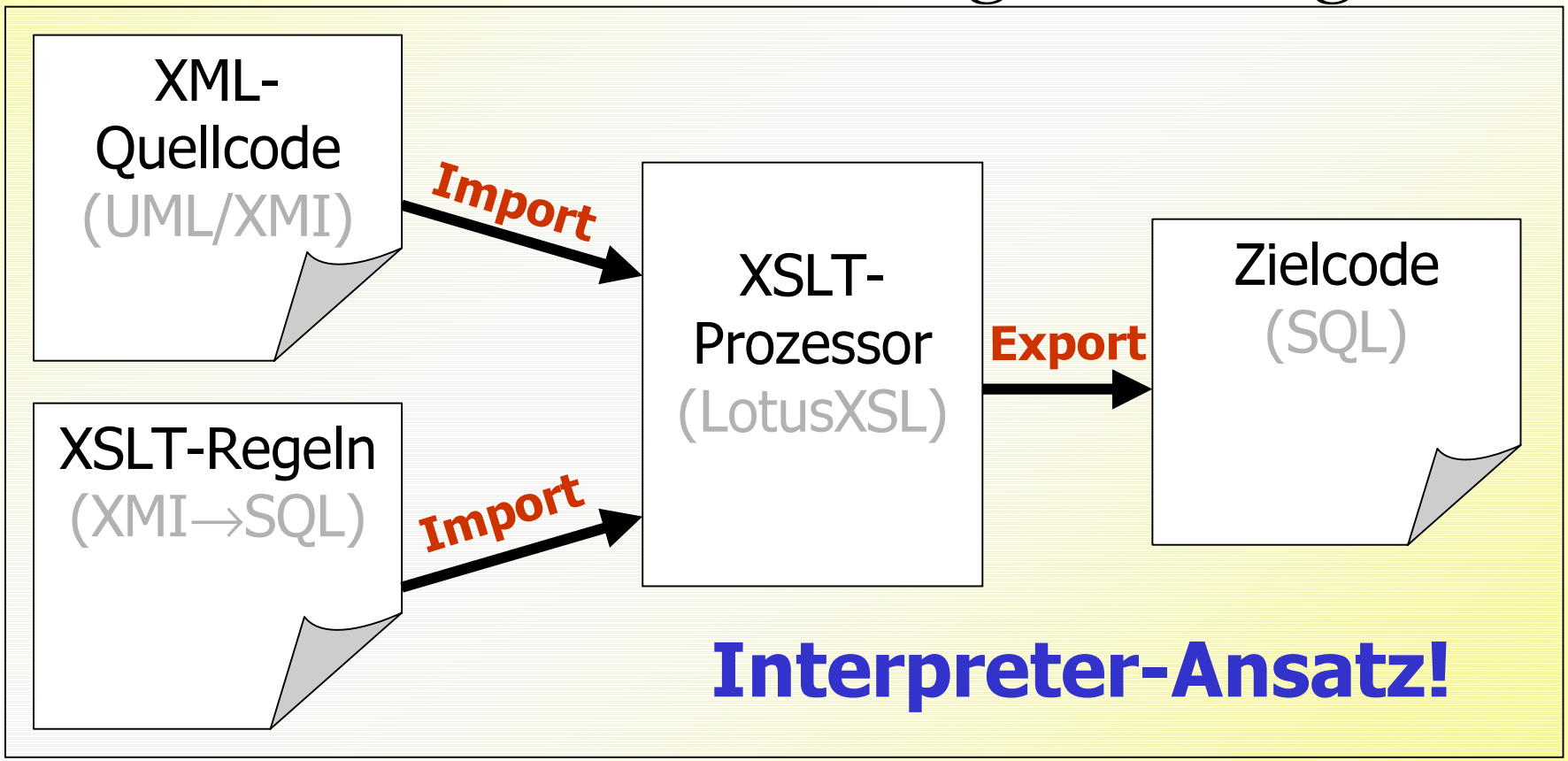
(Sprachbeschreibung)

4 XSLT
Einordnung & Eigenschaften (1)

- Extensible Stylesheet Language Transformations (W3)
  - Sprache zur Beschreibung von Transformationen auf DOM-Bäumen
  - Regelsystem - basierend auf XML
  - Bestandteil von XSL
  - Anwendung des XPath-Konzeptes
- 8.10.1999: Proposed Recommendation
  - http://www.w3.org/TR/1999/PR-xslt-19991008

4 XSLT
Einordnung & Eigenschaften (2)

- **Funktionsweise einer Codegenerierung:**



4 XSLT

Das kleine Einmaleins (1)

- Alle XSLT-Elementnamen beginnen gleich

```
<xsl:... Parameter*>
<!-- ... -->
</xsl:...>
```

- Wurzelelement xsl:stylesheet

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

4 XSLT

Das kleine Einmaleins (2)

- Templates (Regeln)

```
<xsl:template
  match="Foundation.Core.Class"
  name="class_element"
  priority="0.5"
  mode="create_table">...
```

- Template-Sprung

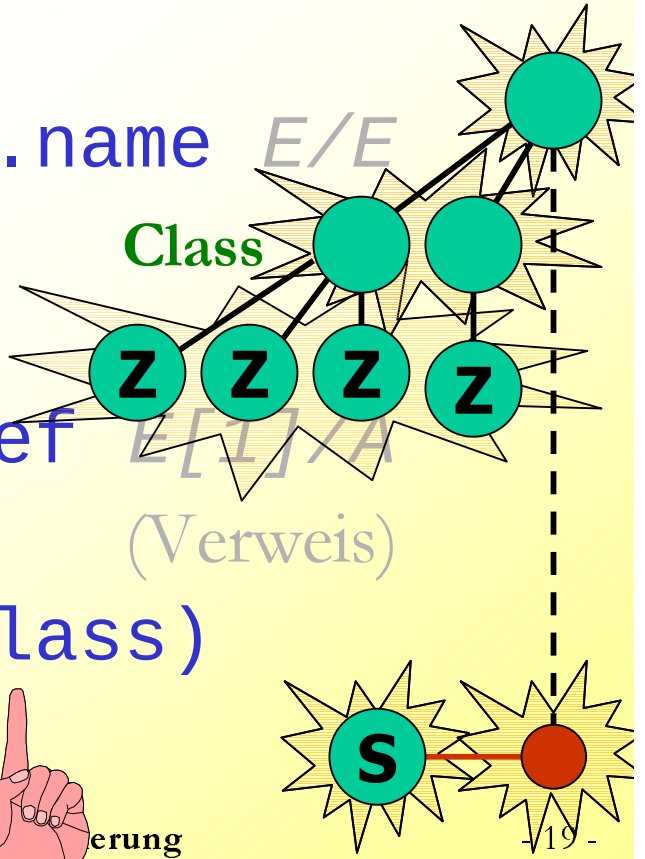
```
<xsl:apply-templates
  select="Foundation.Core.Class"
  mode="create_table"/>
```

4 XSLT

Das kleine Einmaleins (3)

- **Knoten- und Attributadressierung mit *Elementen*, *Funktionen*, *Attributen* (XPath)**

```
Foundation.Core.Class E
F.C.Class/F.C.ModelElement.name E/E
@id A
/XMI/@xmi.version Root/E/A
associationEnd[1]/@xmi.idref E[1]/A
../idref(@xmi.id) E/F(A)
from-children(id(@idref)/Class)
  F(F(A)/E)
```



4 XSLT

Das kleine Einmaleins (4)

- Text

```
<xsl:text>Hallo Zuhörer!</xsl:text>
<xsl:element name="BR"/>   Outp.:<BR/>
```

- Werte von Attributen/Elementen

```
<xsl:value-of select="@name"/>
```

- Prozedurale Möglichkeiten

```
<xsl:if test="number(@zahl)=1">...
<xsl:choose>
  <xsl:when test="position()=last()">...
  <xsl:otherwise>...
<xsl:for-each select="Class">...
```

4 XSLT

Das kleine Einmaleins (5)

- Stringverarbeitung
  - Funktionen: `number`, `string`, `translate`, `substring-after`, `concat`, `normalize`, ...
- Außerdem:
  - Vergleiche auf String, Zahl, Boolean
  - Weitere Funktionen: `div`, `quo`, `mod`, `count`, `contains`, `function-available`, `not`, ...
  - Variablen
  - Sortieren (order, case-ordered, lang, data-type)
  - Funktionsaufrufe des XSLT-Prozessors
  - ...

4 XSLT

Das kleine Einmaleins (6)

- **Bsp.-Regel: UML-Class→SQL-Relation*:**

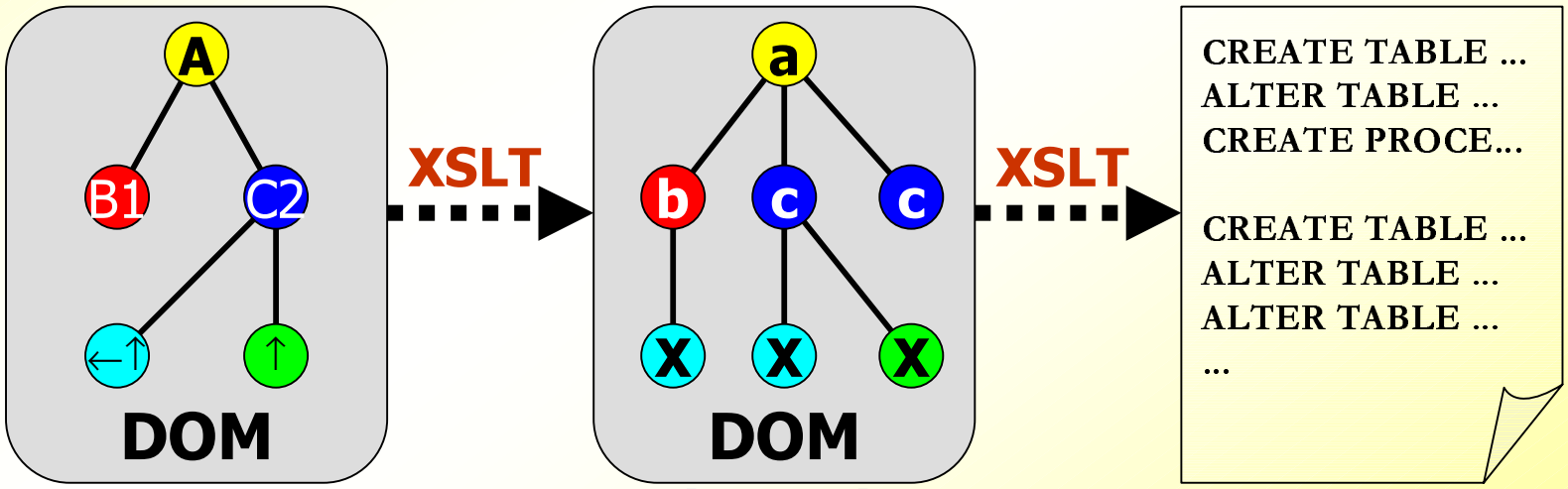
```
<xsl:template match="Class">
 <xsl:if test="not(@isAbstract)">
  <xsl:text>CREATE TABLE </xsl:text>
  <xsl:value-of select="@name">
  <xsl:text> (OID INTEGER NOT NULL</xsl:text>
  <xsl:apply-templates
                  select="features/attribute"/>
  <xsl:text>);</xsl:text>
 </xsl:if>
</xsl:template>
```

*Nicht UML-konform!

5 Ausblick

5 Ausblick (1)

- Umzusetzende Technologie
  (Beispiel UML-SQL):
  UML-Modell → SQL-Modell → SQL-Code



Warum ein zweites DOM?
Damit Quellcode unabhängiger vom Zielcode!
Übersichtlichkeit! Leistungsfähigkeit!

5 Ausblick (2)

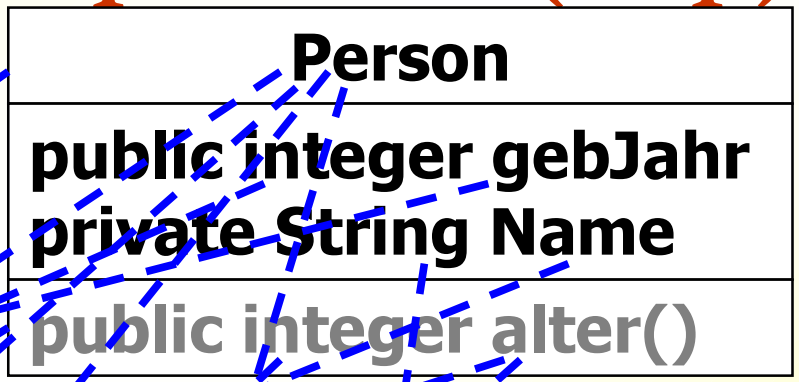
- XSLT-Handbuch (Kurzmanual)
- !! Musterkatalog zur CodeGen. mit XSLT !!
  - Allgemeine Vorschriften (Muster)
    - 1:1-Abbildung, 1-Schritt- und Mehr-Schritt-Abbildungen
  - Am Beispiel SQL
- Editor für Regelbearbeitung (nur theoretisch?)
  - Übersichtliche grafische Regel-Visualisierung
  - Unabhängig von Quell- und Zielsprache
  - Unterstützung von DTD's (Drag-And-Drop)
  - Import von Regelmengen
- (MDM, DTD2J, ...)

6 Diskussion

http://www-st.inf.tu-dresden.de/UMLToolset

2 Codegenerierungsansätze Beschreibungssprachen (Bsp)

- Beispiel (Prinzip):
  - UML (Quelle):
  - SQL (Ziel):




```
CREATE TABLE Person (OID INTEGER NOT NULL,
gebJahr INTEGER NULL, Name VARCHAR(255)
NULL);

CREATE PROCEDURE profPerson_alter AS
DEFINE @alter INTEGER
/*...*/
GO;

ALTER TABLE Person ADD CONSTRAINT
PK_Person PRIMARY KEY (OID);
```

2 Codegenerierungsansätze Beschreibungssprachen (OCL)

- **OCL-Beispiel** (nur CREATE TABLE)*:

```
context Class::codeGen():String
  pre : self.isAbstract = false
  post: result = "CREATE TABLE “ + self.name +
        " (OID INTEGER“ + "NOT NULL“ + self.Attribute
        ->collect(codeGen)->toString+");\n“

context Attribute::codeGen():String
  pre : self.type = “integer"
  post: result = ", “ + self.name + " INTEGER NULL"

context Attribute::codeGen():String
  pre : self.type = “String"
  post: result =", “ + self.name +" VARCHAR(255) NULL"
```

*Nicht UML-konform!

2 Codegenerierungsansätze Beschreibungssprachen (DTD)

- Meine.DTD-Beispiel (UML → SQL)*:

```
<Rule name="rule_Class"><ForObject name="Class"/>
  <Guard><Check type="not equal">
      <Param><Attribute name="isAbstract"/></Param>
      <Param>true</Param>
  </Check></Guard>
  <Action>CREATE TABLE <Attribute name="name"/> (OID
    INTEGER NOT NULL<Child name="Attribute">
      <Operation rules="rule_Attr1 rule_Attr2"/>
    </Child>;<CR/></Action>
</Rule>

<Rule name="rule_Attr1"><ForObject name="Attribute"/>
  <Guard><Check type="equal">
      <Param><Attribute name="type"/></Param>
      <Param>integer</Param>
  </Check></Guard>
  <Action>, <Attribute name="name"/> INTEGER
    NULL</Action>
</Rule>
```

*Nicht UML-konform!

2 Codegenerierungsansätze Beschreibungssprachen (XSLT)

- **XSLT-Beispiel (UML → SQL):**

- **Kein Beispiel, da später mehr zu XSLT!**