# Modellgetriebene Softwareentwicklung

Vorbereitung der Zwischenpräsentationen,
Grundlagen Modelltransformationen

**Dipl.-Inf. Stefan Kühne**
Betriebliche Informationssysteme
Institut für Informatik
Universität Leipzig
kuehne@informatik.uni-leipzig.de

# Vorbereitung der Zwischenpräsentationen

# Agenda

- Intension
- Inhalte
- Anforderungen
- Termine
- Fragen

## Intension

- **Zwischenergebnisse präsentieren**
- **Anregungen sammeln**
- **Üben**
  - zu Präsentieren
  - sich an Zeitvorgaben zu halten
  - auf Rückfragen zu reagieren
  - sich an eine Vorlage zu halten
- **Synchronsation/Vergleich mit anderen Studenten**
- **Arbeitsstand einschätzen**
- **…**

# Inhalte

- **Thema**
  - Gegenstand
  - Problemstellung
  - Motivation
  - Zielsetzung
- **Gliederung**
- **Arbeitsstand**
- **Zeitplan**

## Anforderungen

- **Präsentation der Inhalte**
  - Auf ca. 5 Folien
  - Unter Einhaltung der Vorlage
  - In max. 10min

- **Folien zur Verfügung stellen**
  - An Betreuer
  - Per E-Mail
  - 1 Tag vor Präsentation

## Termine

- **04.12.2009**
  1. Jörg Hartmann: Adaption domänenspezifischer Modellierungswerkzeuge am Beispiel bflow* Toolbox
  2. Ziad Sehili: Definition von textuellen und graphischen Modellierungswerkzeugen
  3. Markus Hütter: Vergleich Microsoft DSL Tools vs. Eclipse GMF
  4. Diep Phan: Modellbasierte Werkzeugintegration
  5. Stefan Mertins: Synchronisierungsstrategien
- **11.12.2009**
  1. Stanley Hillner: Bridging Microsoft Oslo und Eclipse EMF
  2. Christian Böhme: Evaluation des Bridging Musters
  3. Michael Siebauer: Vergleich von Modell-Repositories
  4. Torsten Grigull: Berechnung domänenspezifischer Modelldifferenzen
  5. Nico Korn: Mergingstrategien zwischen Modellen

# Grundlagen Modelltransformationen

# Vorkenntnisse

- Modell
- Modellierungssprache
- Abstrakte Syntax
- Konkrete Syntax
- Metamodell
- Metametamodell
- Metamodellierungssprache
- Technikraum

# Agenda

- Beispiele
- Definition
- Eigenschaften
- Klassifikation
- Werkzeuge

# Anwendungsbeispiele

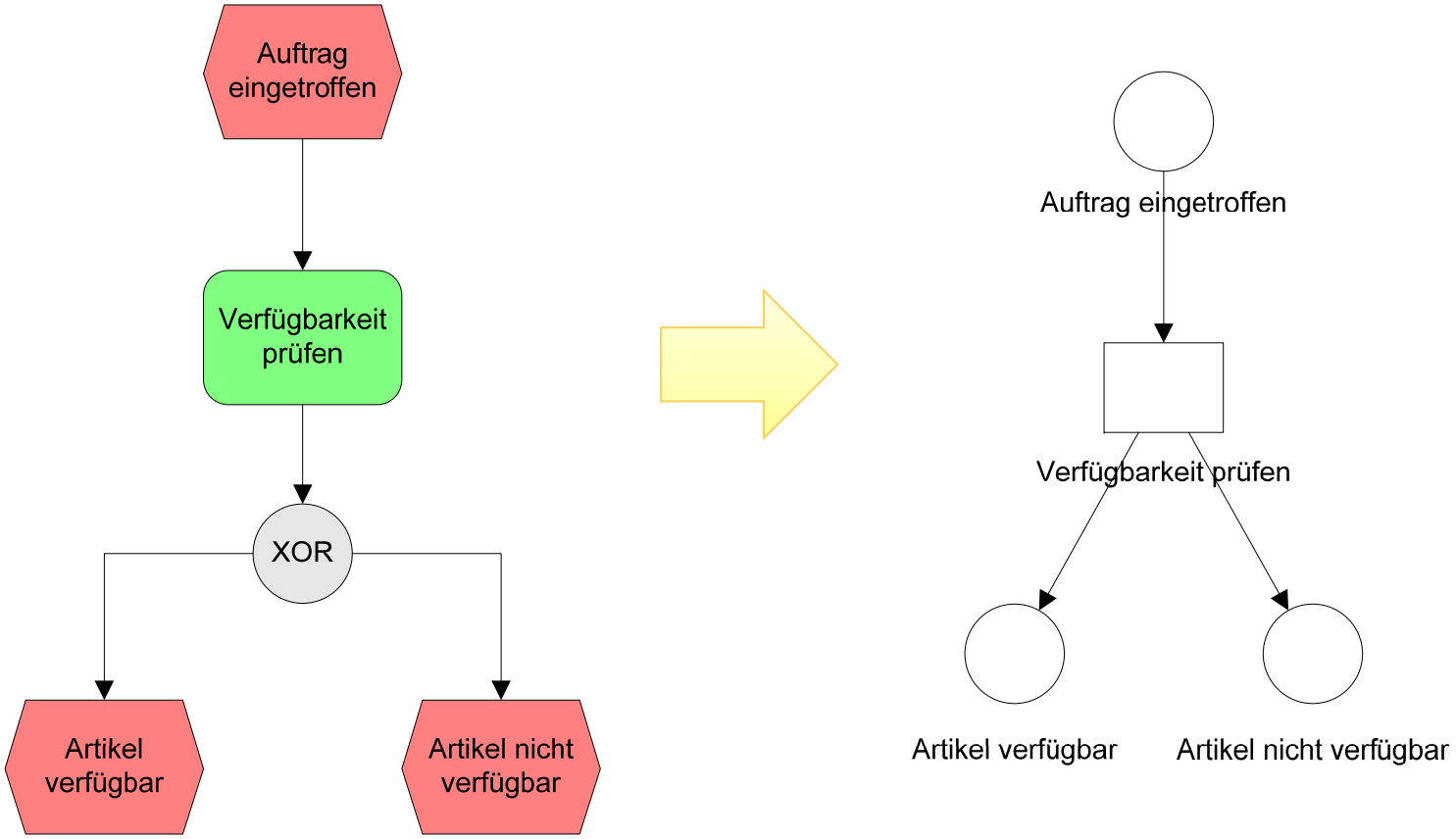
## Beispiel 1

```
entity Customer {
    name = String;
    address = Address;
}

entity Address {
    street = String;
    zip = String;
    city = String;
}
```

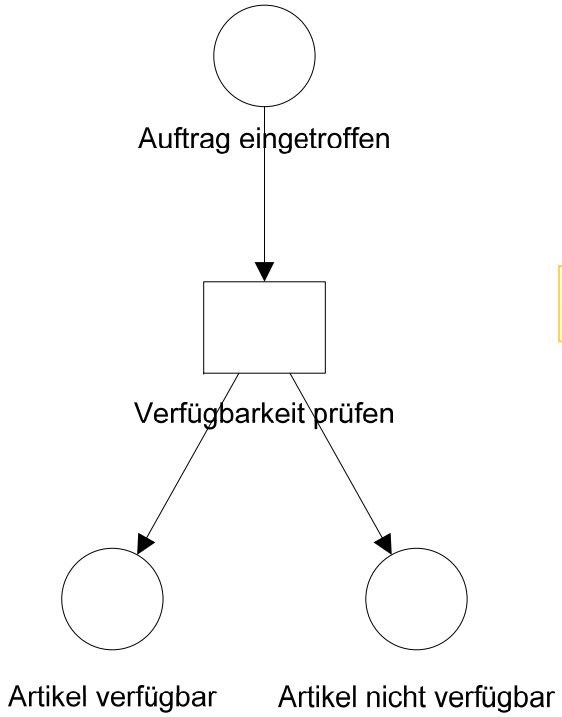


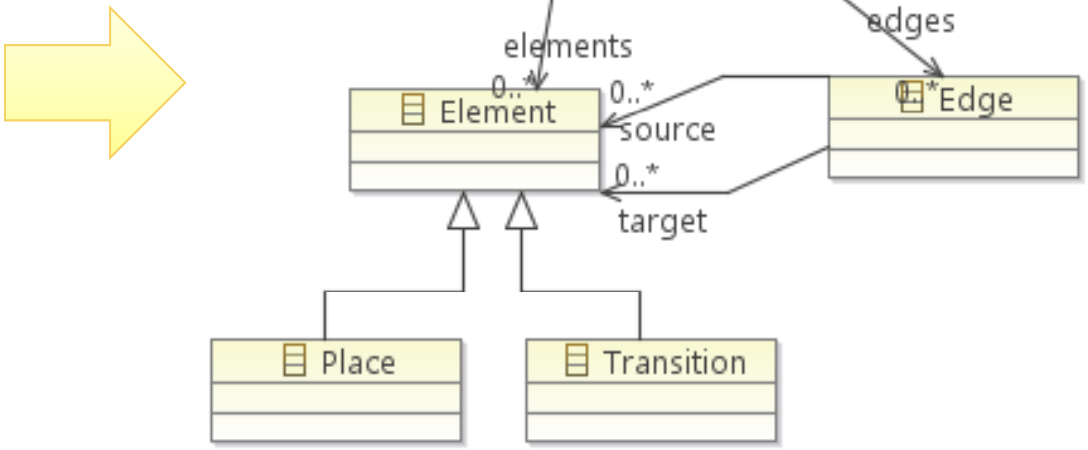
```
public class Customer
    implements Serializable {

    protected String name;
    protected Address address;

    public String getName() {
        return name;
    }
    public void setName(String n) {
        this.name = name;
    }
    public Address getAddress() {
        return address;
    }
```

# Beispiel 2

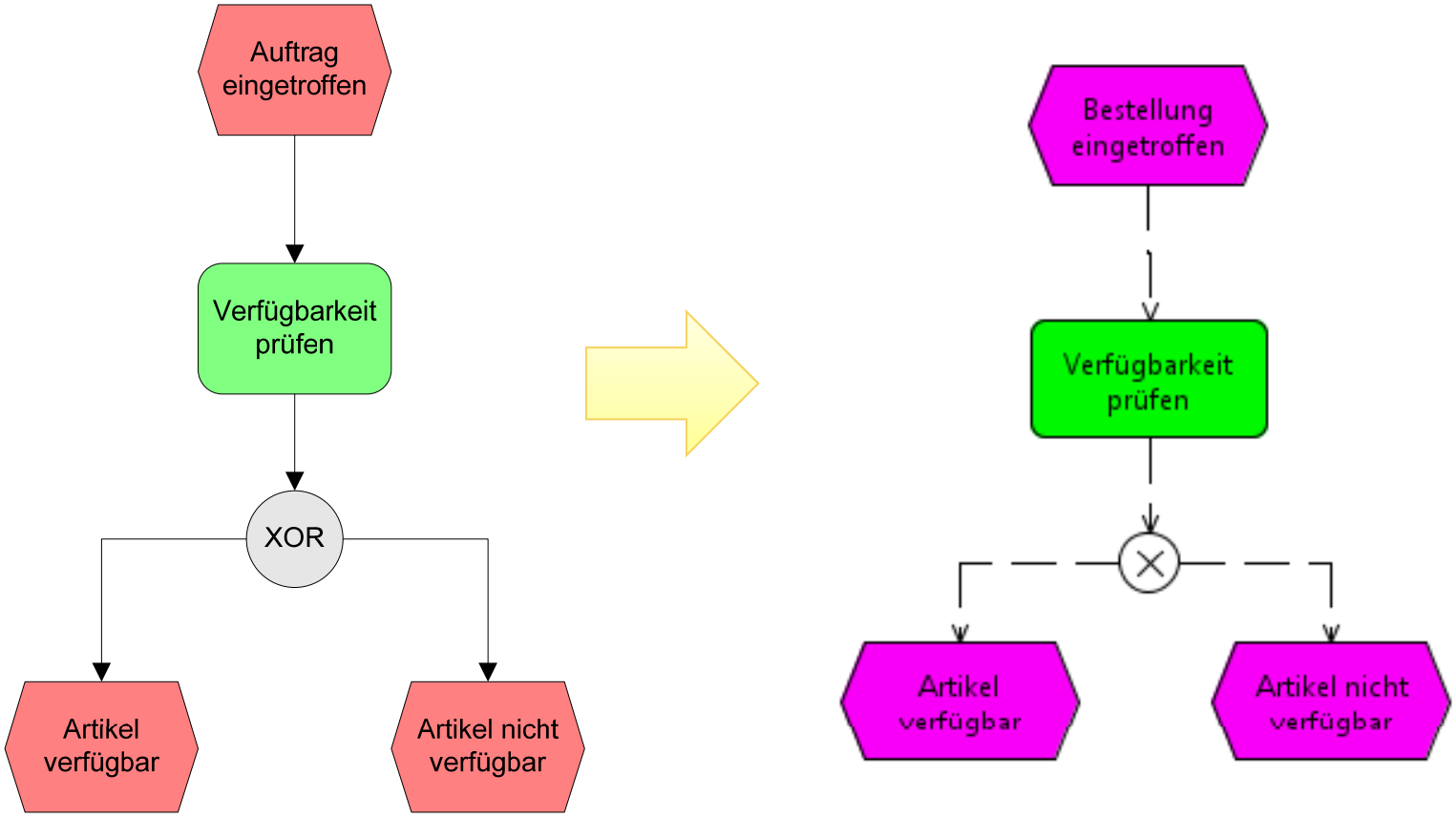

Auftrag eingetroffen
Verfügbarkeit prüfen
XOR
Artikel verfügbar
Artikel nicht verfügbar
Auftrag eingetroffen
Verfügbarkeit prüfen
Artikel verfügbar
Artikel nicht verfügbar

# Beispiel 3


Auftrag eingetroffen
Verfügbarkeit prüfen
Artikel verfügbar
Artikel nicht verfügbar


PetriNet
elements
0..*
edges
0..*
Element
source
0..*
Edge
0..*
target
Place
Transition

# Beispiel 4


Auftrag eingetroffen
Verfügbarkeit prüfen
XOR
Artikel verfügbar
Artikel nicht verfügbar
Bestellung eingetroffen
Verfügbarkeit prüfen
Artikel verfügbar
Artikel nicht verfügbar

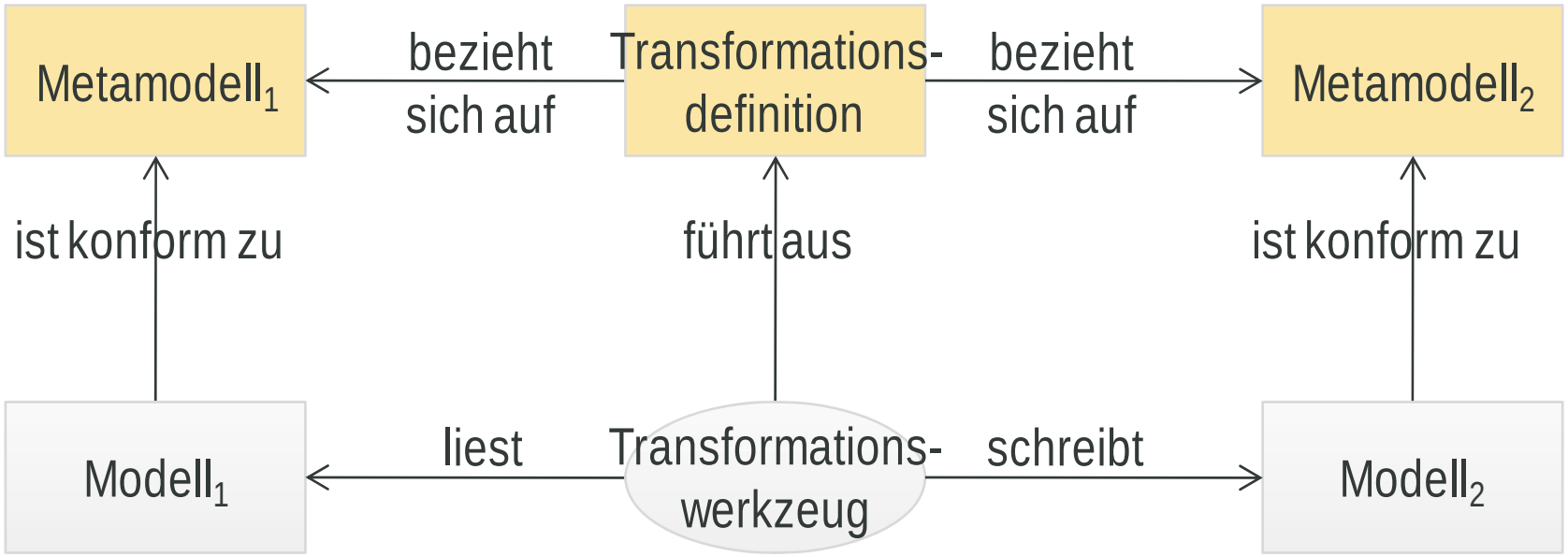
# Definition

# Modell-zu-Modelltransformation

**“A transformation function, or transformation for short, is a function in the mathematical sense of the term, that is a set of pairs with the constraint that a value [...] in the domain maps to at most one value in the range. To be more precise a transformation (function) is a set of transformation instances.”**

**[Favre u. Nguyen 2005, S. 68]**

# Modell-zu-Modelltransformation



# Transformationsbeispiele

# XSLT

```
<xsl:for-each select="//*[name()='elements'][@xmi:type='epc:Event']">
    <xsl:variable name="EmptydefRef">
        <xsl:value-of select="@defRef"/>
    </xsl:variable>
    <definition>
        <xsl:if test="$EmptydefRef="">
            <xsl:attribute name="defId">
                <xsl:value-of select="$AnzahlDefId"/>
            </xsl:attribute>
            <saxon:assign name="AnzahlDefId" select="$AnzahlDefId+1"/>
        </xsl:if>
        <xsl:if test="$EmptydefRef!="">
            <xsl:attribute name="defId">
                <xsl:value-of select="$EmptydefRef"/>
            </xsl:attribute>
        </xsl:if>
    </definition>
</xsl:for-each>
```

# XTend

```
List[aris::MT_EEPC] eepcModels(aris::Group group):
	group.eAllContents.typeSelect(aris::MT_EEPC);

create epc::Epc createEpcModel(aris::MT_EEPC model,
		String relativeWorkflowPath):
	elements.addAll(model.containSymbols.createElement(relativeWorkflowPath)) ->
	connections.addAll(model.lines(relativeWorkflowPath)) ->
connections.typeSelect(InformationArc).adjustStartEventMessageFlow();

create epc::Event createElement(aris::ST_EV element,
		String relativeWorkflowPath):
	this.setName(element.object.name);

create epc::XOR createElement(aris::ST_OPR_XOR_1 element,
		String relativeWorkflowPath):
	this.setName(element.object.name.norm());
```

```
rule A
        transform a : A
        to b : B extends C2D {

        guard : true

        doit();

}

rule B
        transform a : A
        to b : B, c : C {

        guard {
                return false;
        }

}
```

# ATL

```
rule Member2Female {
        from
                s : Families!Member (s.isFemale())
        to
                t : Persons!Female (
                        fullName <- s.firstName + ' ' + s.familyName
                )
}
```

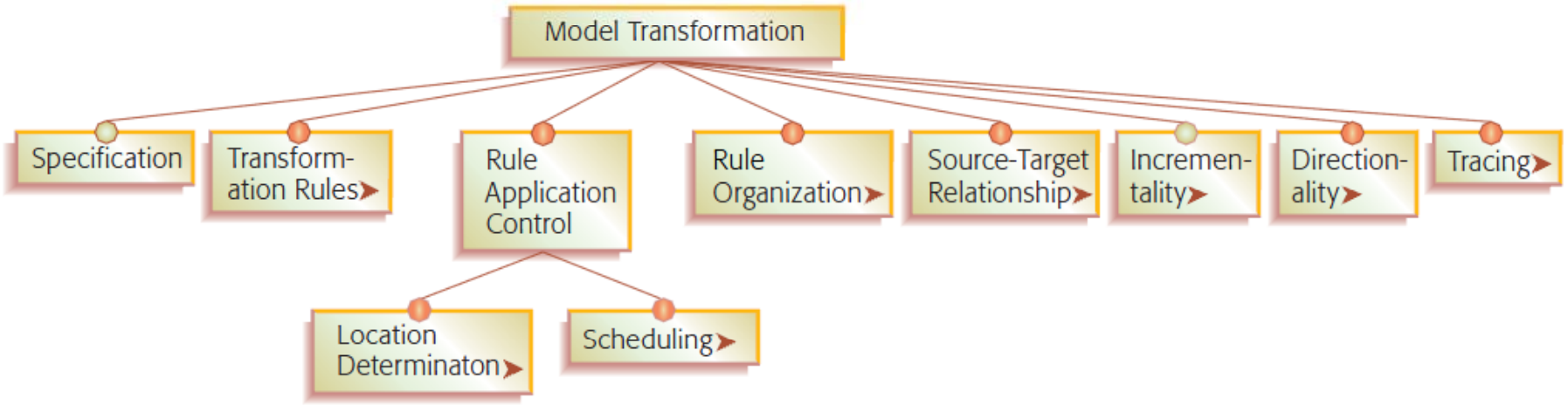
# XPand

```
«IMPORT "http://www.eclipse.org/gmf/2008/GenModel"»
«IMPORT "http://www.eclipse.org/emf/2002/Ecore"»
«IMPORT "http://www.eclipse.org/emf/2002/GenModel"»
«EXTENSION xpt::diagram::parts::Common»
«EXTENSION xpt::diagram::Helper»

«DEFINE visualIDConstant FOR gmfgen::GenCommonBase-»
    «EXPAND xpt::Common::generatedMemberComment»
    public static final int VISUAL_ID = «visualID»;
«ENDDEFINE»

«DEFINE modelIDConstant FOR gmfgen::GenDiagram-»
    «EXPAND xpt::Common::generatedMemberComment»
    public static String MODEL_ID = "«editorGen.modelID»";
    «EXPAND xpt::Common::nonNLS»
«ENDDEFINE»
```

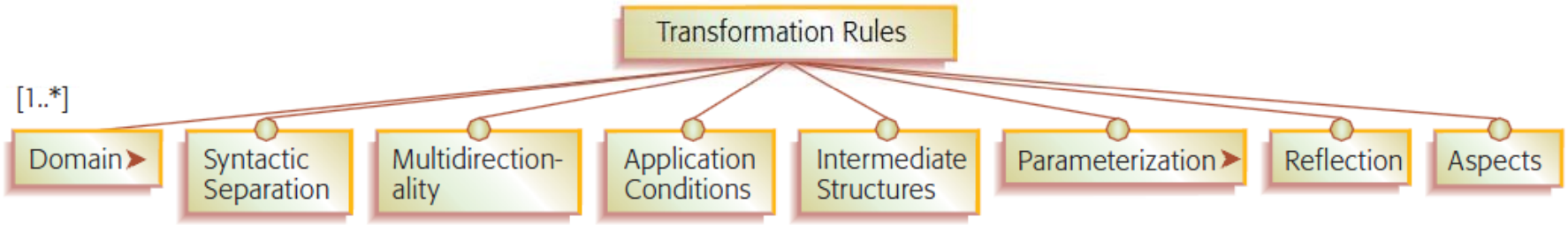
```
[% for (i in Sequence{1..10}) { %]
i is [%=i%]
[% } %]
```

# Eigenschaften

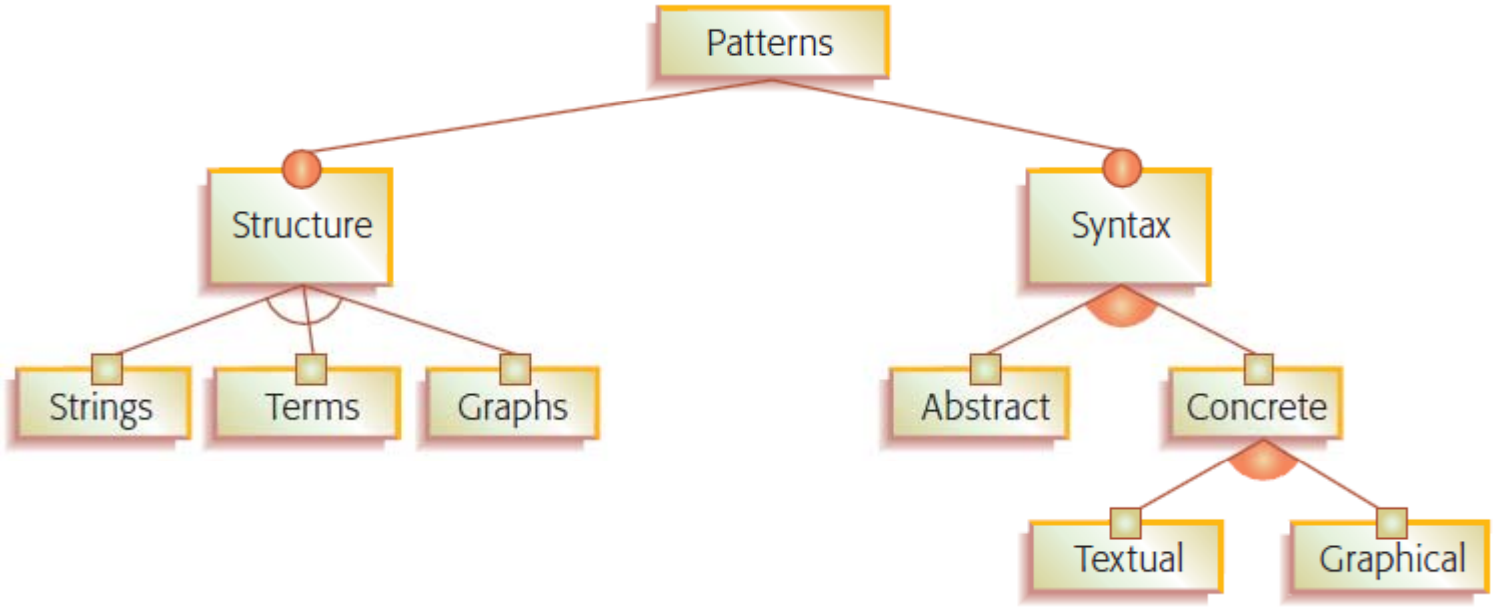
# Modelltransformationen [CzHe06]



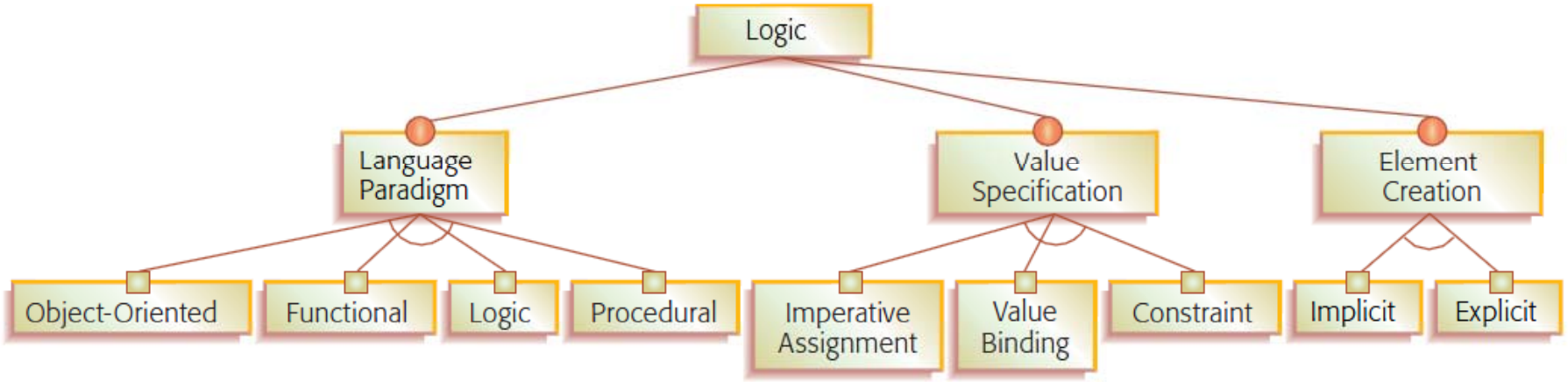
# Transformationsregeln [CzHe06]



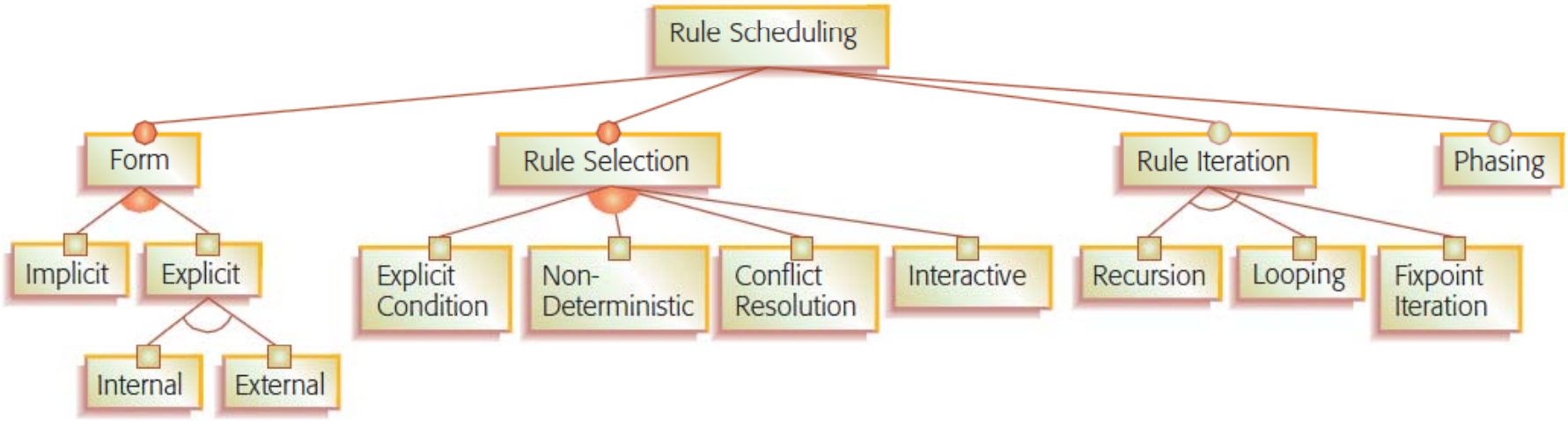
# Muster in Transformationsregeln [CzHe06]



# Logik von Transformationsregeln [CzHe06]



# Ausführungsreihenfolge von Transformationsregeln [CzHe06]



# Erfolgskriterien von Transformationen und Werkzeugen [MeGo05]

- **Korrektheit**
  - Syntaktisch
    - **Konform**
    - **Vollständig**
  - Semantisch
    - **Verhaltenseigenschaften**
    - **Terminierung**
    - **Konfluenz**
- **Anpassung, Wiederverwendung**
- **Test, Validierung**
- **Verarbeitung unvollständiger Modelle**

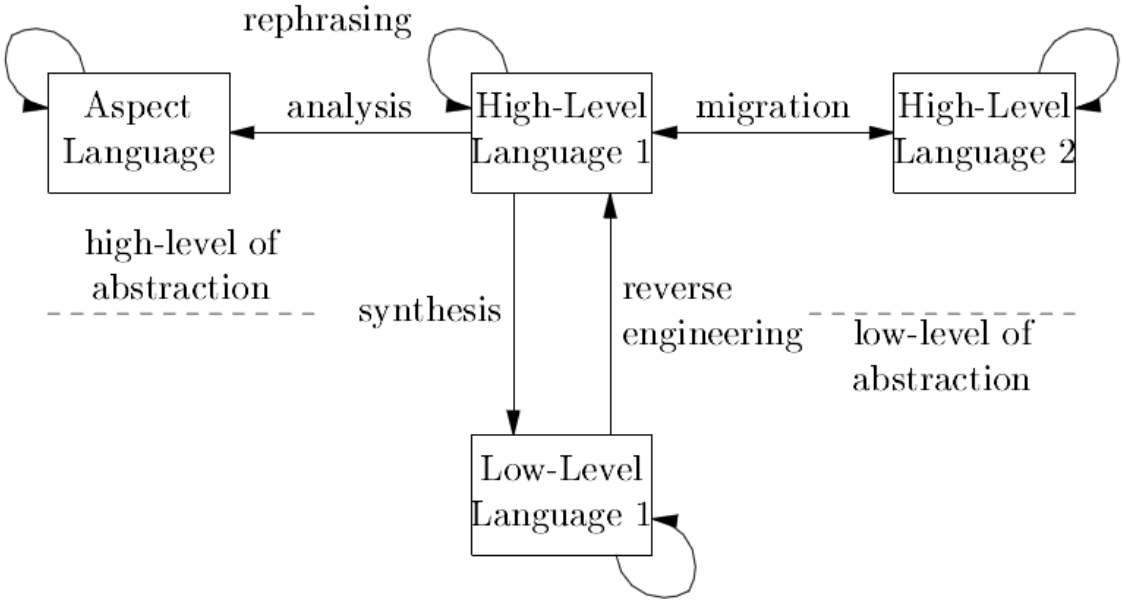
# Klassifikation

# Abstraktionsgrad [Viss01]

- **Horizontale Transformation vs. vertikale Transformationen**

- **Horizontale Transformationen**
  - Migration
  - Umformulierung
  - Analyse

- **Vertikale Transformationen**
  - Abstraktion
  - Verfeinerung



## Ein- und Ausgabe

- **Modell-zu-Modell vs. Modell-zu-Text**
- **Modell-zu-Modelltransformationen**
  - Bezug zum Ausgabe-Metamodell
- **Modell-zu-Text-Transformationen**
  - Kein Bezug zum Ausgabe-Metamodell
  - Erzeugung textueller Artefakte

## Sprache

- **Endogen vs. Exogen**
- **Endogene Transformation**
  - Sprache von Ein- und Ausgabemodellen ist identisch
- **Exogene Transformationen**
  - Sprache von Ein- und Ausgabemodellen ist unterschiedlich

## Technikraum

- **Homogen vs. Heterogen**
- **Homogene Transformation**
  - Technikraum von Ein- und Ausgabemodellen ist identisch
- **Heterogene Transformation**
  - Technikraum von Ein- und Ausgabemodellen ist unterschiedlich

# Transformationswerkzeuge

## Auswahl

- **Imperativ vs. Deklarativ**

- **Deklarativ**
  - Template-basiert (z.B. EGL, XPand, XSLT)
    - **Frame-basiert [CzEi00] (z.B. XVCL, XFramer, ANGIE)**
  - Graphbasiert [MGVK05]
    - **Allgemeine: AGG, PROGRES**
    - **Reengineering: Fujaba, Varlet**
    - **Modelltransformation: GReAT, MOLA**
    - **Modell-Checking: Viatra, Groove**
  - Relational (z.B. IBM MTF)
  - Logische Programmierung (z.B. Prolog)
  - Funktional (z.B. XTend)

- **Hybrid**
  - ATL
  - ETL

# Model Management Frameworks

- **openArchitectureWare**
  - oAW-Expression
  - XTend
  - XPand
  - Check

- **Epsilon**
  - EOL
  - ETL
  - EGL
  - EVL
  - EML

## Literatur


[CzEi00]
Czarnecki, K. & Eisenecker, U. W.: Generative Programming : Methods, Tools, and Applications. Addison-Wesley, 2000

[CzHe06]
Czarnecki, K. & Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal, 2006, 45, S. 621-645

[MeGo05]
Mens, T. & van Gorp, P.: A Taxonomy of Model Transformation Proc. Int'l Workshop on Graph and Model Transformation (GraMoT), Institute of Cybernetics at Tallinn Technical University, 2005, S. 7-23

[MGVK05]
Mens, T.; van Gorp, P.; Varró, D. & Karsai, G.: Applying a Model Transformation Taxonomy to Graph Transformation Technology Institute of Cybernetics at Tallinn Technical University, 2005, 24-39

[Viss01]
Visser, E.: A Survey of Rewriting Strategies in Program Transformation Systems. Electronic Notes in Theoretical Computer Science, 2001, 57