ADBIS | 2003
September 3-6, 2003, Dresden, Germany

Rule-Based Generation of XML DTDs from UML Class Diagrams

Thomas Kudrass, Tobias Krumbein

{kudrass | tkrumbe}@imn.htwk-leipzig.de

Overview

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Motivation

- Growing Importance of XML
  - standard representation
  - different data expressed in XML
- Different Applications
  - data exchange / data representation / data storage
- Different Requirements
- Classification of XML Documents
  - document-centric, semistructured, data-centric
- Heterogeneity of XML not always desired
  - definition of a schema to restrict structure and content (e.g., for XML databases)
- Lack of Design Tools for Schemas
- Definition mostly intuitive
  - e.g., by tree-based graphical tools (XML Spy, ...)
- Semantics of Data not guaranteed
  - Different views on data

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Motivation

- Complexity ←→ correct schema
- Redundancy cannot be avoided
- Later changes of the data model expensive
- Growing usage of XML → requires efficient management and re-use → XML databases
- Up to now no complete design methodology for XML databases
- Target of our work
  - Development of an XML DB design
  - XML schema-independent modelling (conceptual model)
  - automatic generation of a DTD as XML DB schema

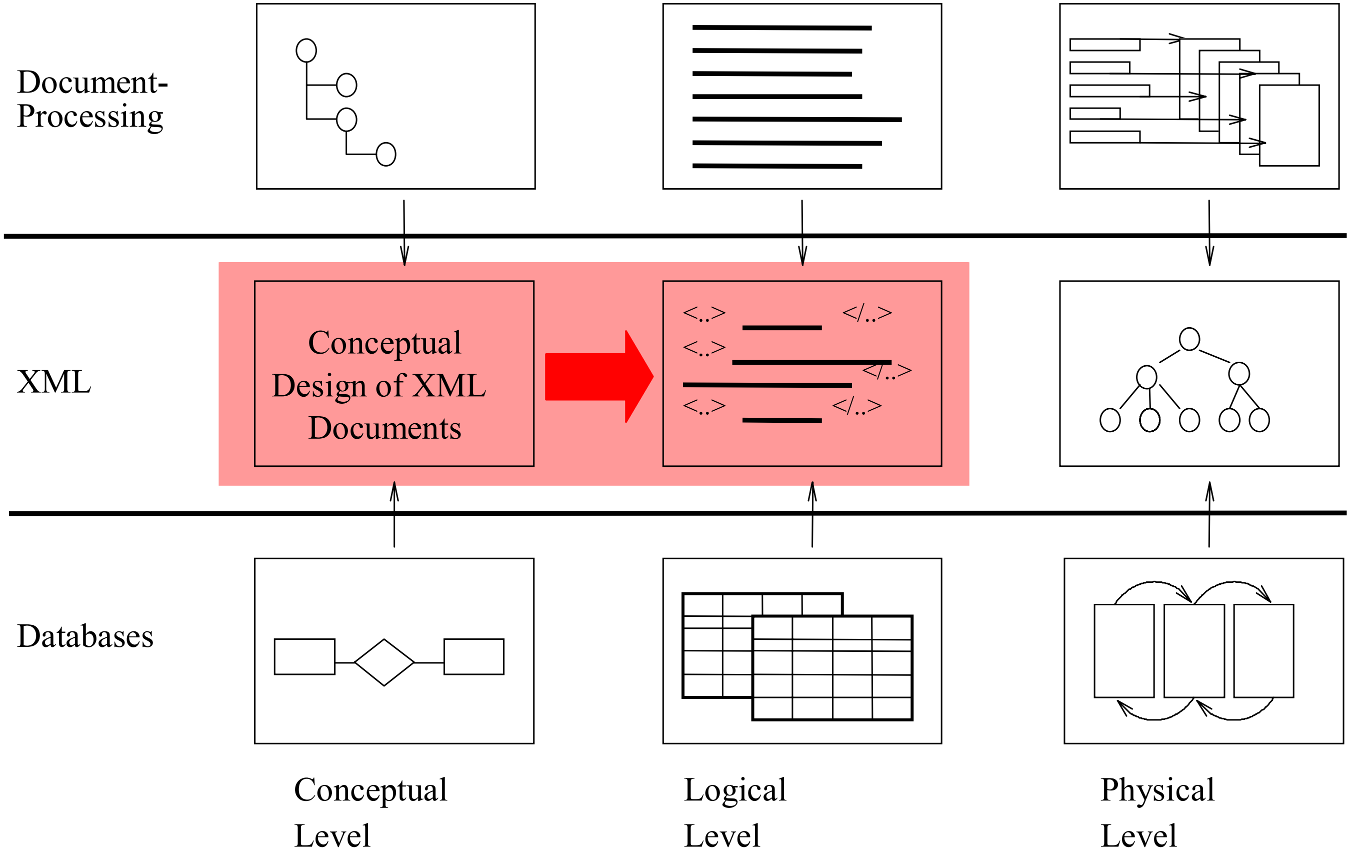
Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

XML DB Design

- Three-Level Design Process
  - Conceptual Model
    - modeling of the information requirements
  - Logical Schema (DTD as XML schema)
    - storage and representation of the data and the data model
  - Physical Level
    - indexes and access paths

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

XML DB Design



Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

XML DB Design

- Three-Level Design Process
  - Conceptual Model
    - modeling of the information requirements
  - Logical Schema (DTD as XML schema)
    - storage and representation of the data and the data model
  - Physical Level
    - indexes and access paths
- Classification of XML documents
  - Document-centric (unstructured, irregular)
  - Semistructured (data- und document-centric components)
  - Data-centric (structured, regular)
- No uniform XML DB design for all document types

XML DB Design

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

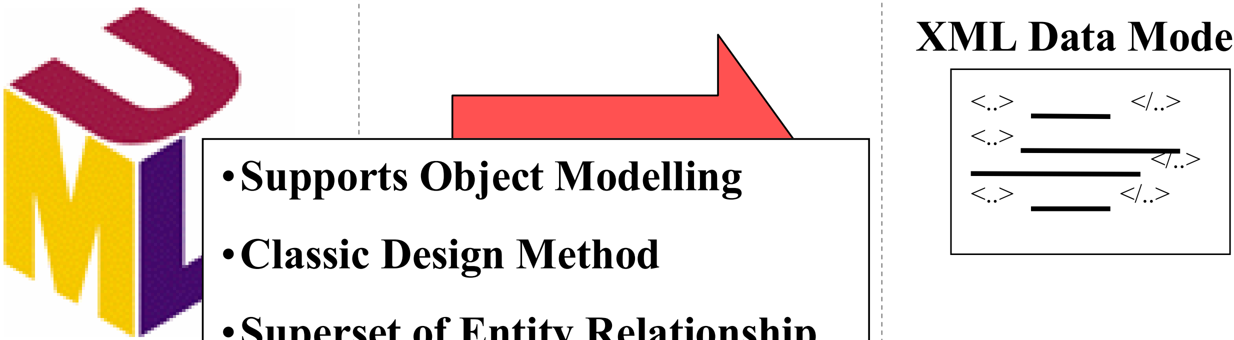
	document-centric	semistructured	data-centric
conceptual level	modeling of stucture and content	modeling of structure and content	modeling of structure
	tree-based XML editors		ER, UML, ORM
logical level	document model / updates to structure / content	data model / document model queries / updates to structure / content	data model queries / updates to content
	SGML, XML XPath, DOM, XQuery	OEM, XML Lorel, XQuery	relational DM object-oriented DM XML SQL, OQL, XQuery
physical level	storage of structure at value level	storage of structure at schema and value level	storage of structure at schema level
	files full-text index structure index	generic storage of graphs DOM	relational DB object-relational DB object-oriented DB

UML to XML Data Model

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Conceptual Level

Logical Level

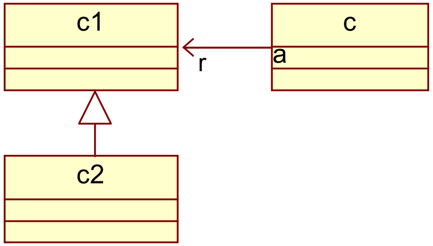


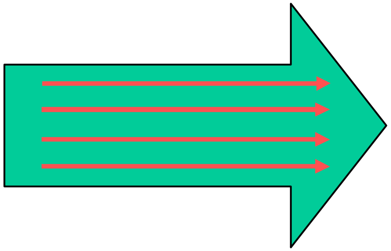
- **Supports Object Modelling**
- **Classic Design Method**
- **Superset of Entity Relationship Model**
- **Standardized Exchange Format with XMI**
- **CASE tool independent Modelling possible**

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML to XML Data Model

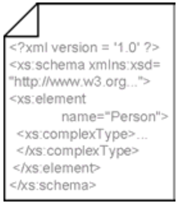
Conceptual Level





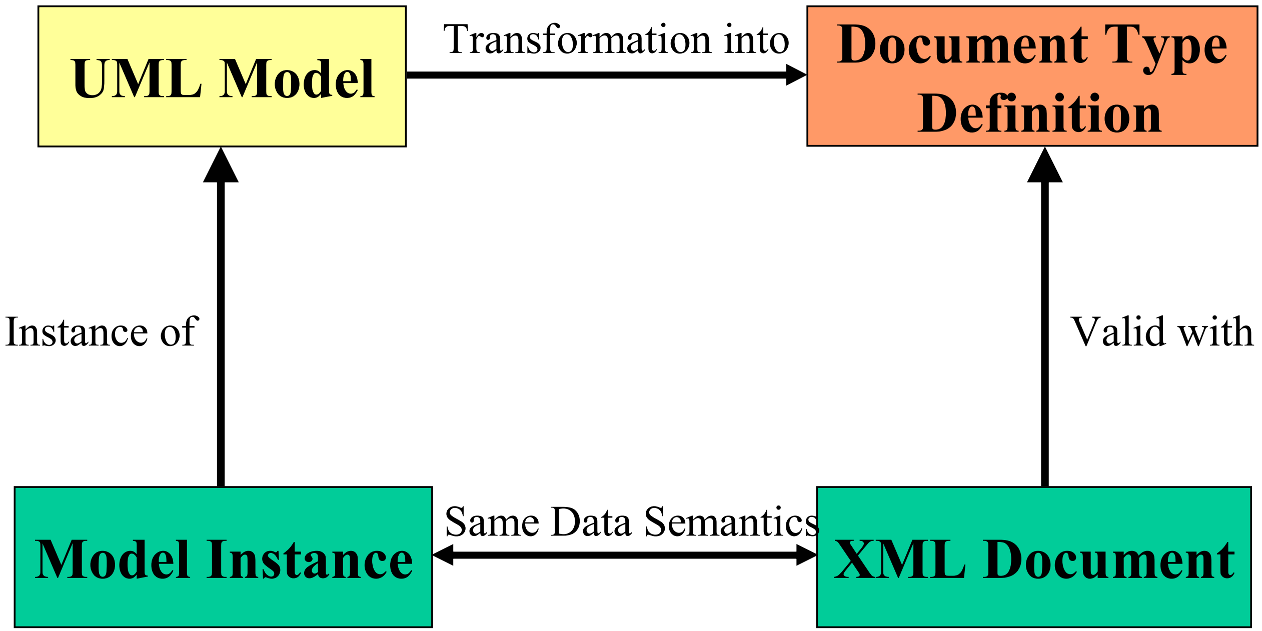
Logical Level

DTD



```
<?xml version = '1.0' ?>
<xs:schema xmlns:xsd=
"http://www.w3.org...">
<xs:element
            name="Person">
  <xs:complexType>...
  </xs:complexType>
 </xs:element>
</xs:schema>
```

UML to DTD


UML Model

Transformation into

Document Type Definition

Instance of

Valid with

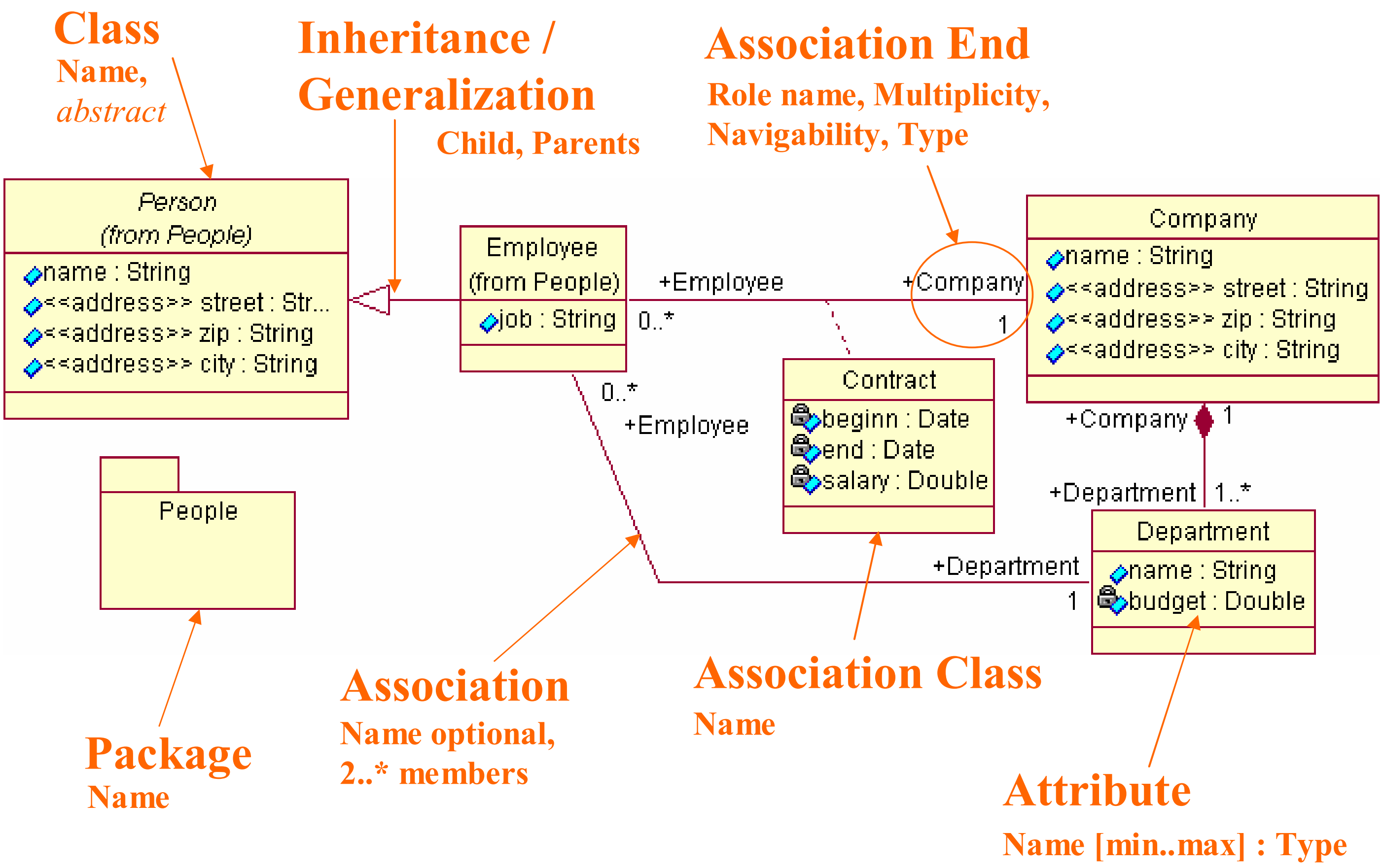
Model Instance

Same Data Semantics

XML Document

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML Class Diagram



Class
Name,
abstract

Inheritance / Generalization
Child, Parents

Association End
Role name, Multiplicity, Navigability, Type

Package
Name

Association
Name optional, 2..* members

Association Class
Name

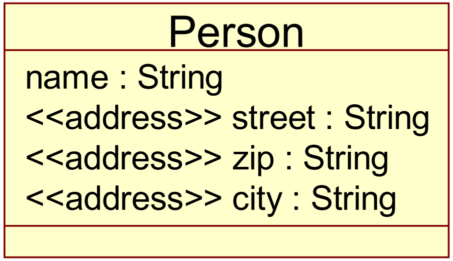
Attribute
Name [min..max] : Type

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML Classes to DTD

- Class
  - Properties represented as attributes
  - Can take part at associations
  - instances = objects
    - unique
- XML element definition
  - can possess properties as attributes
  - can have subelements
  - instances = elements
    - Uniqueness by ID attribute

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook



Person

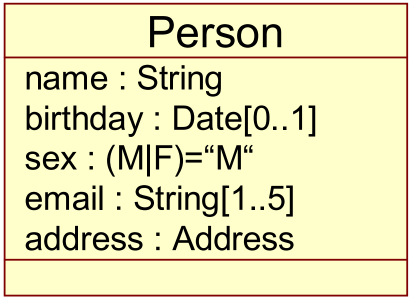
name : String
<<address>> street : String
<<address>> zip : String
<<address>> city : String

```
<!ELEMENT Person EMPTY>
<!ATTLIST Person
  id ID #REQUIRED
  name CDATA #REQUIRED
  address.street CDATA #REQUIRED
  address.zip CDATA #REQUIRED
  address.city CDATA #REQUIRED>
```

UML Attributes to DTD

UML Attribute	XML Attribute	XML Element
primitive / complex datatypes	primitive datatype	primitive / complex datatype
any multiplicity	[0..1] and [1..1]	any multiplicity
property-string	not supported	property-string as attributes
default value	default value	not supported
initial value	only fixed value	not supported
value list	enumeration supported	not supported
scope of definition	local scope	global scope
access properties	not supported	not supported

UML Attributes to DTD



Person

name : String
birthday : Date[0..1]
sex : (M|F)=“M“
email : String[1..5]
address : Address

- Transformation as XML attributes

```
<!ELEMENT Person EMPTY>
<!ATTLIST Person
  name CDATA #REQUIRED
  birthday CDATA #IMPLIED
  sex (M|F) “M“
  email NMTOKENS #REQUIRED
  address CDATA #REQUIRED>
```

- Transformation as XML elements

```
<!ELEMENT Person(name,birthday?,sex,(email,
       email?,email?,email?,email?),address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthday (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT address (Address)>
```

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML Associations to DTD

- Hierarchy
  - Use of element – subelement relationship
  - Only 1:n relationship possible, otherwise redundancy
  - Subelement always bound to the life span of the parent element
  - Problemes with recursions in model
  - Only applicable for read-only data
- Association Element
  - Global element representing the association
  - IDREF references to class elements of the association
  - No definition of multiplicity
  - No type integrity

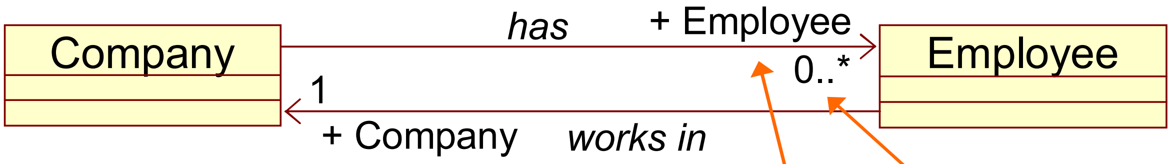
UML Associations to DTD

- References with ID/IDREF
  - Class element contains subelement with IDREF Attribute to reference the related class elements
  - Any multiplicity can be represented
  - No type integrity, only use of naming conventions
  - Only unidirectional associations can be represented
  - No guarantee for mutual references in bidirectional associations
- XLink und XPointer
  - Predefined Attributes
  - Simple / Extended XLinks available
  - Links among multiple documents possible
  - No type integrity, check depends on XML processor

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML Associations to DTD

- References with ID/IDREF



```
<!ELEMENT Employee (Ref_Employee.Company)>
<!ATTLIST Employee
  id ID #REQUIRED >
     <!ELEMENT Ref_Employee.Company EMPTY>
     <!ATTLIST Ref_Employee.Company
       Company IDREF #REQUIRED>

<!ELEMENT Company (Ref_Company.Employee*)>
<!ATTLIST Company
 id ID #REQUIRED >
     <!ELEMENT Ref_Company.Employee EMPTY>
     <!ATTLIST Ref_Company.Employee
       Employee IDREF #REQUIRED>
```

Role Name

Multiplicity

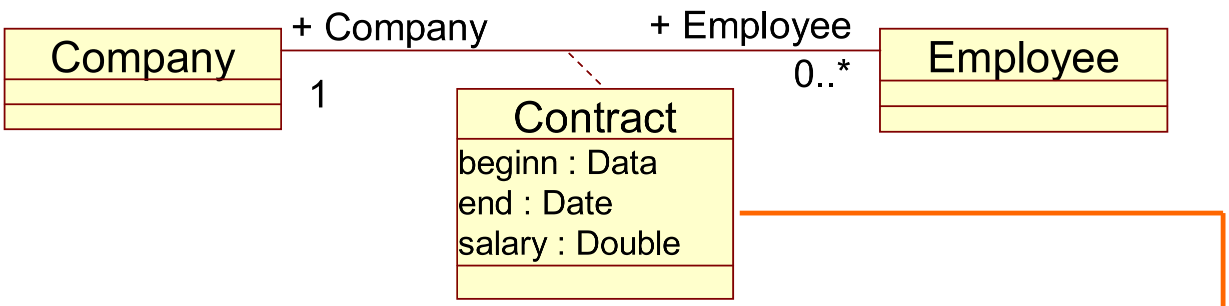
Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Association Classes and N-ary Associations

- Same mapping alternatives as for associations
  - Hierarchical relationship – association attributes add to the subelement
  - Association Element – contain the association attributes
  - References with ID/IDREF - association attributes add to the reference element (are stored twice)
  - XLink and XPointer – contain the association attributes
  - resolution into a Class and two binary association
- N-ary associations
  - Association Element and XLink can only represent a N-ary association
  - resolution into a Class and N binary association

Association Classes to DTD

- References with ID/IDREF

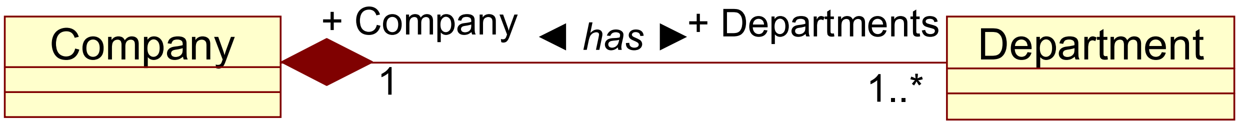


```
<!ELEMENT Employee (Ref_Employee.Company)>
<!ATTLIST Employee id ID #REQUIRED >
 <!ELEMENT Ref_Employee.Company EMPTY>
 <!ATTLIST Ref_Employee.Company Contract IDREF #REQUIRED>

<!ELEMENT Company (Ref_Company.Employee*)>
<!ATTLIST Company … >
 <!ELEMENT Ref_Company.Employee EMPTY>
 <!ATTLIST Ref_Company.Employee Contract IDREF #REQUIRED>

<!ELEMENT Contract (Ref_Contract.Employee,
Ref_Contract.Company)>
<!ATTLIST Contract … >
 <!ELEMENT Ref_Contract.Employee EMPTY>
 <!ATTLIST Ref_Contract.Employee Employee IDREF #REQUIRED>
 <!ELEMENT Ref_Contract.Company EMPTY>
 <!ATTLIST Ref_Contract.Company Company IDREF #REQUIRED>
```

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML Composition to DTD

- Exclusive part-whole relationship
- Lifespan of parts are bound to the whole
- Hierachical relationship provides suitable semantics



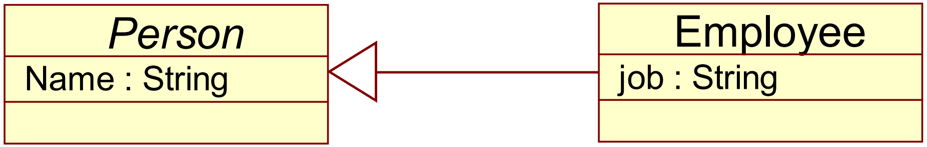
```
<!ELEMENT Company (Ref_Company.Department+)
<!ATTLIST Company ... >
    <!ELEMENT Ref_Company.Department (Department)>

<!ELEMENT Department (...)
<!ATTLIST Department ... >
```

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

UML Generalization to DTD

- Parameter Entities
  - defined for attributes and subelements of superclasses
  - subclass inherits attributes using parameter entities
  - single inheritance only
- Embedded Elements
  - superclass embedded into the subclass element
  - superclass element substituted by a choice list that contains the superclass element and all its subclass elements
  - multiple inheritance possible

UML Generalization to DTD



Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

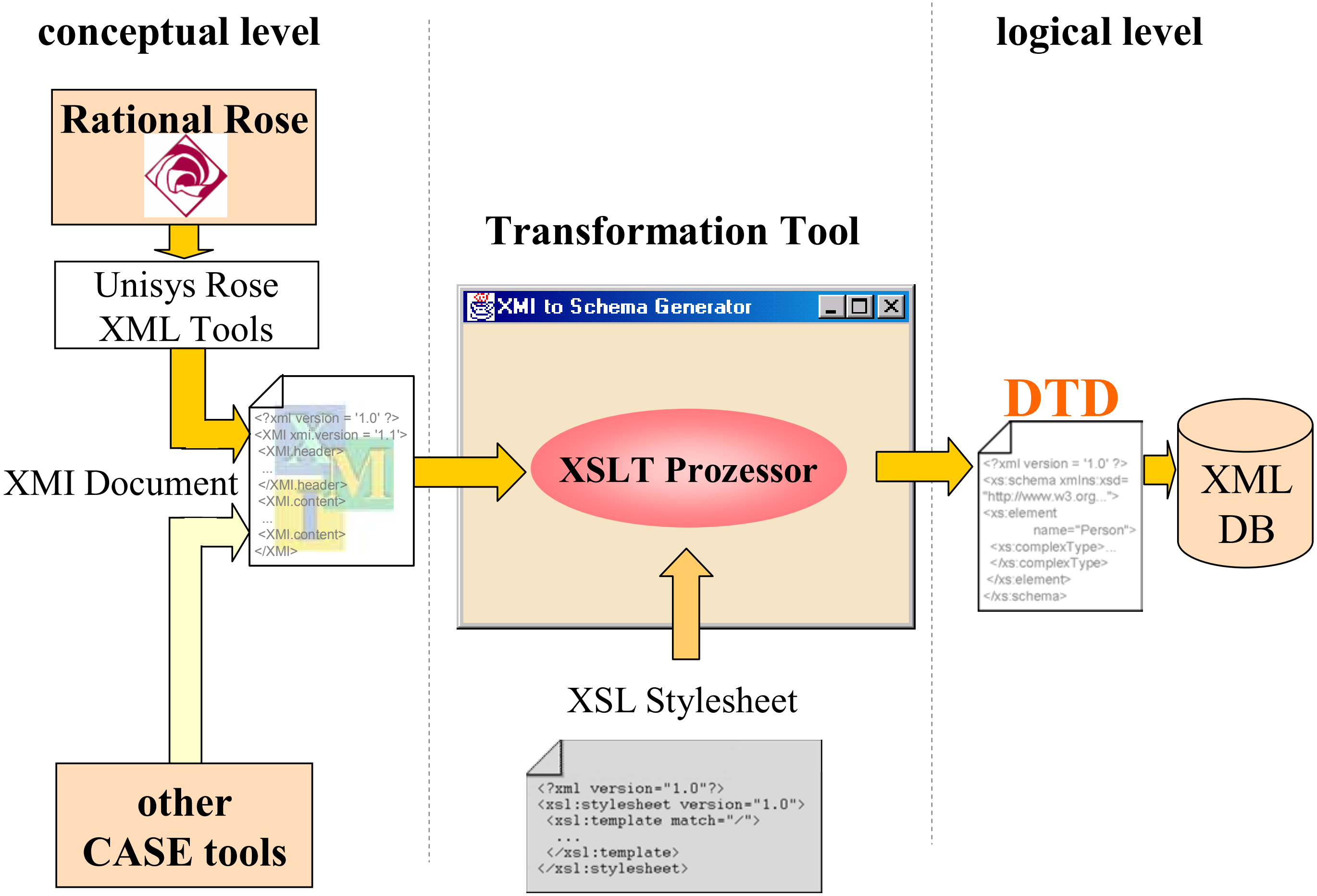
- **Parameter Entities**

```
<!ENTITY % Person “EMPTY”>
<!ENTITY % PersonAttList “
  id ID #REQUIRED
  Name CDATA #REQUIRED”>

<!ELEMENT Person (%Person;)>
<!ATTLIST Person %PersonAttList;>

<!ELEMENT Employee (%Person;)>
<!ATTLIST Employee
  %PersonAttList;
  Job CDATA #REQUIRED >
```

- **Embedded Elements**

```
<Person id=“p1” Name=“Paul”/>

<Employee id=e1” job=“Programmer”>
    <Person id=“p2” Name=“Evi”/>
</Employee>
```

UML to XML Data Model



```
<?xml version = '1.0' ?>
<XMI xmi.version = '1.1'>
 <XMI.header>
  ...
 </XMI.header>
 <XMI.content>
  ...
 <XMI.content>
</XMI>
```

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0">
 <xsl:template match="/">
  ...
 </xsl:template>
</xsl:stylesheet>
```

```
<?xml version = '1.0' ?>
<xs:schema xmlns:xsd=
"http://www.w3.org...">
<xs:element
            name="Person">
  <xs:complexType>...
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Evaluation of the DTD Transformation

Advantages	-widely accepted standard
Drawbacks	-weak data typing -no type integrity -global element definitions only -no object-oriented constructs (e.g., generalization) -no XML Syntax

Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Conclusions

- Separate conceptual and logical level
- Advantages of the XML DB Design
  - better quality of database / DB schema
  - preserve data semantics
  - early detection and elimination of errors
  - cost savings by conceptual modeling
  - quick changes possible
  - design can be verified vs. the requirements
  - better readibility and understanding by graphical representation

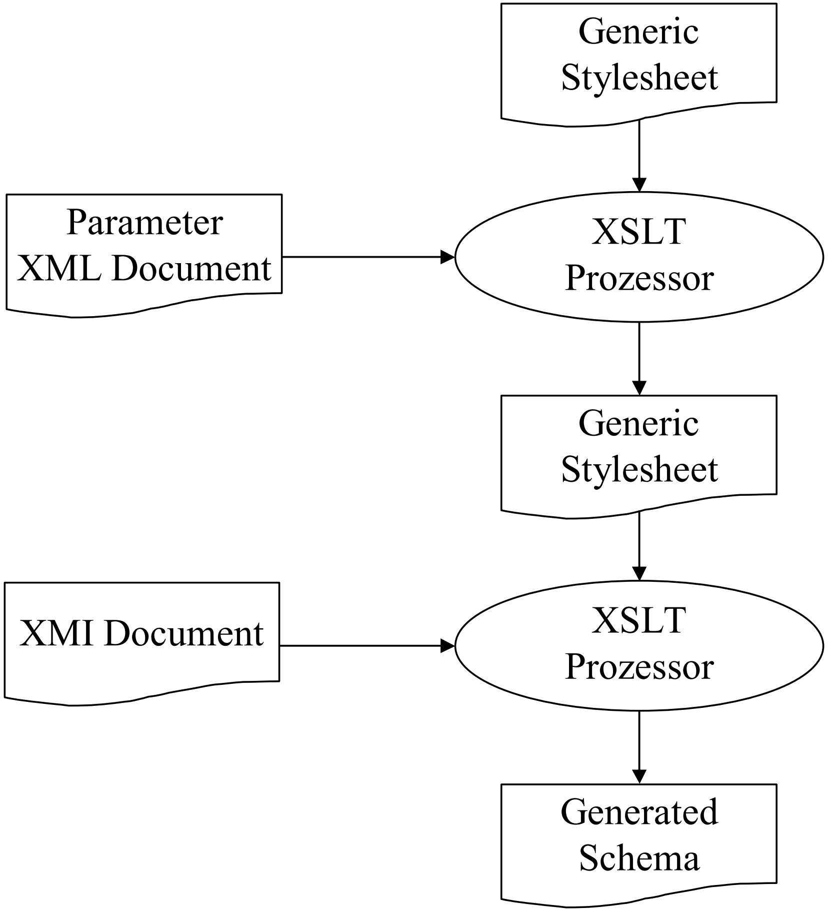
Conclusions

- Problems
  - UML is object-oriented, XML tree-based
  - Classes and associations in UML
    - XML has no concept for associations (embedding of child elements corresponds with aggregation)
  - No 1:1-Mapping of UML to XML structures
    - Mapping of UML to XML ambiguous (several possibilities)
    - Choice of mapping alternatives are **Design Decisions**
  - Transformation not lossless but loss of information can be determined

Outlook

- Use of name spaces to avoid naming conflicts
- Evaluate OCL Constraints
- Transformations to XML Schema and Tamino Schema Definition
- Better Implementation with XSLT and XPath 2.0 possible ?
- Adaptation of the XSL-Stylesheets to the new version of Unisys Rose XML Tools - no 100 % XMI-Standard
- Dynamic Transformation
  - Multi-level XSLT-Transformation

Outlook



Motivation
XMLDB Design
UML → DM
UML → DTD
Design Process
Conclusions
Outlook

Thank you very much for your attention