

Integration von Integritätsbedingungen bei der XML-Schemaevolution

Hannes Grunert





Inhalt

I. Allgemeines

II. Integritätsbedingungen

i. Wertebereichsintegrität

ii. Schlüsselintegrität

iii. Strukturelle Integrität

iv. Logische Konsistenz

III. Transformationsprozess

IV. Testszenario



I. Allgemeines

- **Thema der Arbeit:** Integration von Integritätsbedingungen bei der XML-Schemaevolution
- **Art:** Konzeption und Implementation
- **Gutachter:** PD Dr.-Ing. habil. Meike Klettke
- **Zweitgutachter:** Prof. Dr.-Ing. habil. Peter Forbrig
- **Betreuer:** Dipl. Inf. Thomas Nösinger
- **Abgabedatum:** 29.04.2013

Aufgabenstellung

- Erweiterung von CodeX¹ um Integritätsbedingungen
 - Vergleich mit anderen Systemen
 - Integration und Speicherung auf Modellebene
 - Visualisierung im Editor
 - Analyse, Kostenabschätzung, Bewertung und Korrektur von Integritätsbedingungen
 - Schemaevolution nach Anpassung des Schemas an einem Beispiel zeigen
 - Testszenarien entwickeln

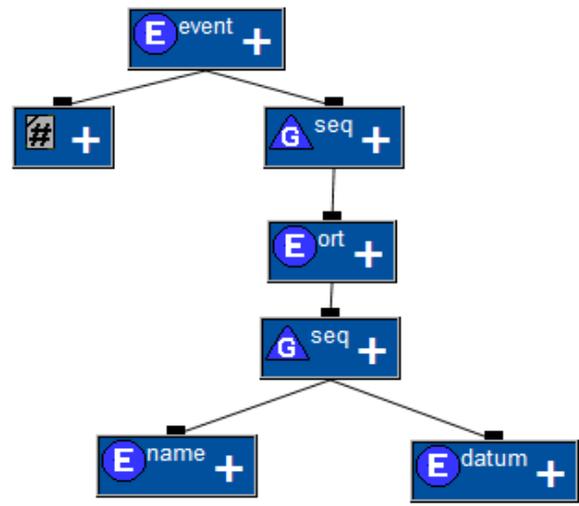
1: CodeX - Conceptual Design and Evolution of XML-Schema; siehe nächste Folie

XML-Schemaevolution

- Veränderung eines XML-Schemas mit anschließender Überprüfung der dazugehörigen XML-Dateien auf ihre Gültigkeit
- Forschungsschwerpunkt am Lehrstuhl DBIS
- Konzept: Schemaevolution mittels konzeptionellen Modell durchführen (EMX – Entity Model for XML-Schema)
- Werkzeug: Forschungsprototyp CodeX (Conceptual Design and Evolution of XML-Schema)
 - Realisierung mittels Google Web Toolkit (GWT) als Webanwendung
 - Client-Server-Architektur

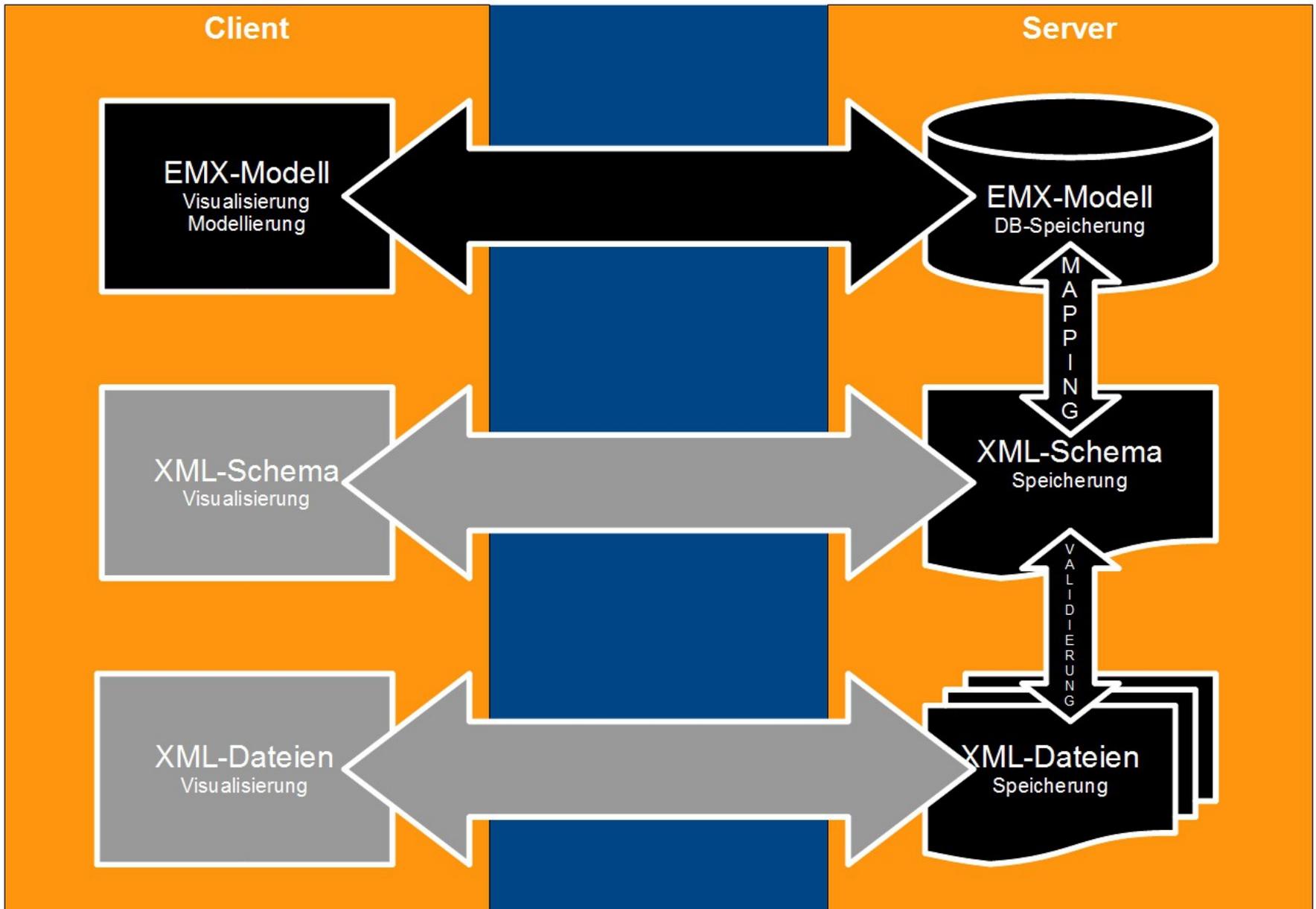
- 2
- noName.emx
- noName2.emx
- emx.xsd
- drei

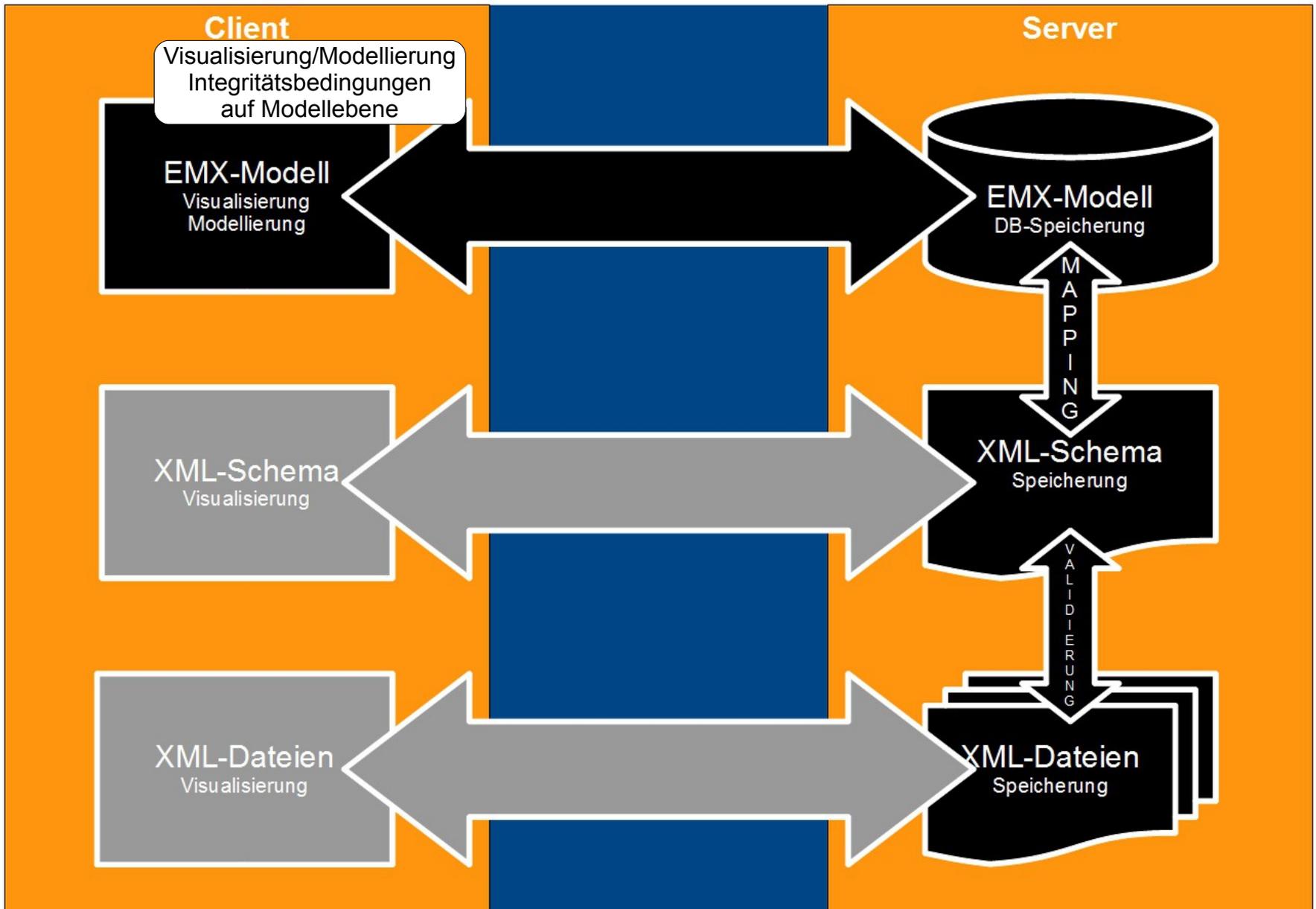
2/noName.e... [Save] [Close]

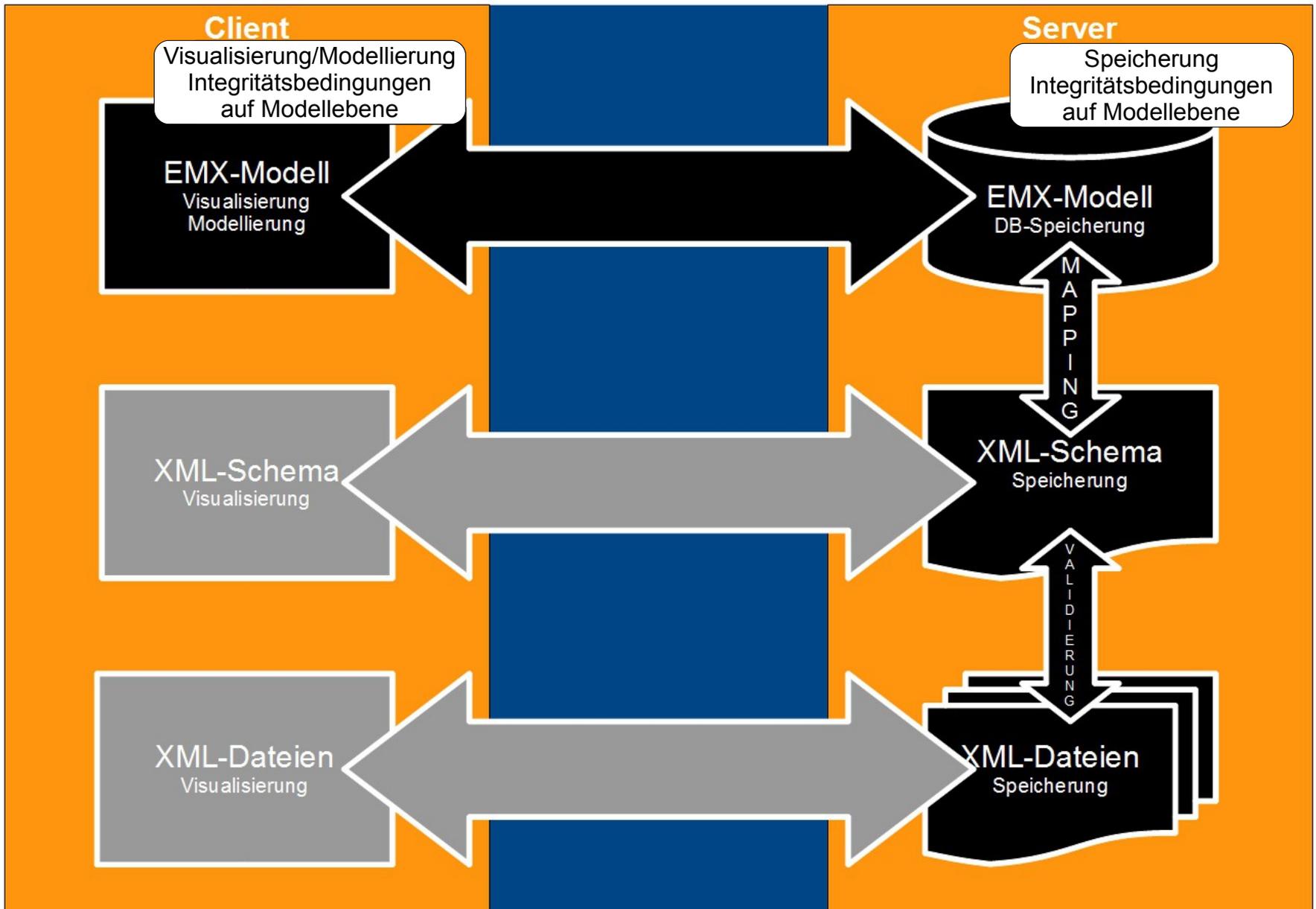


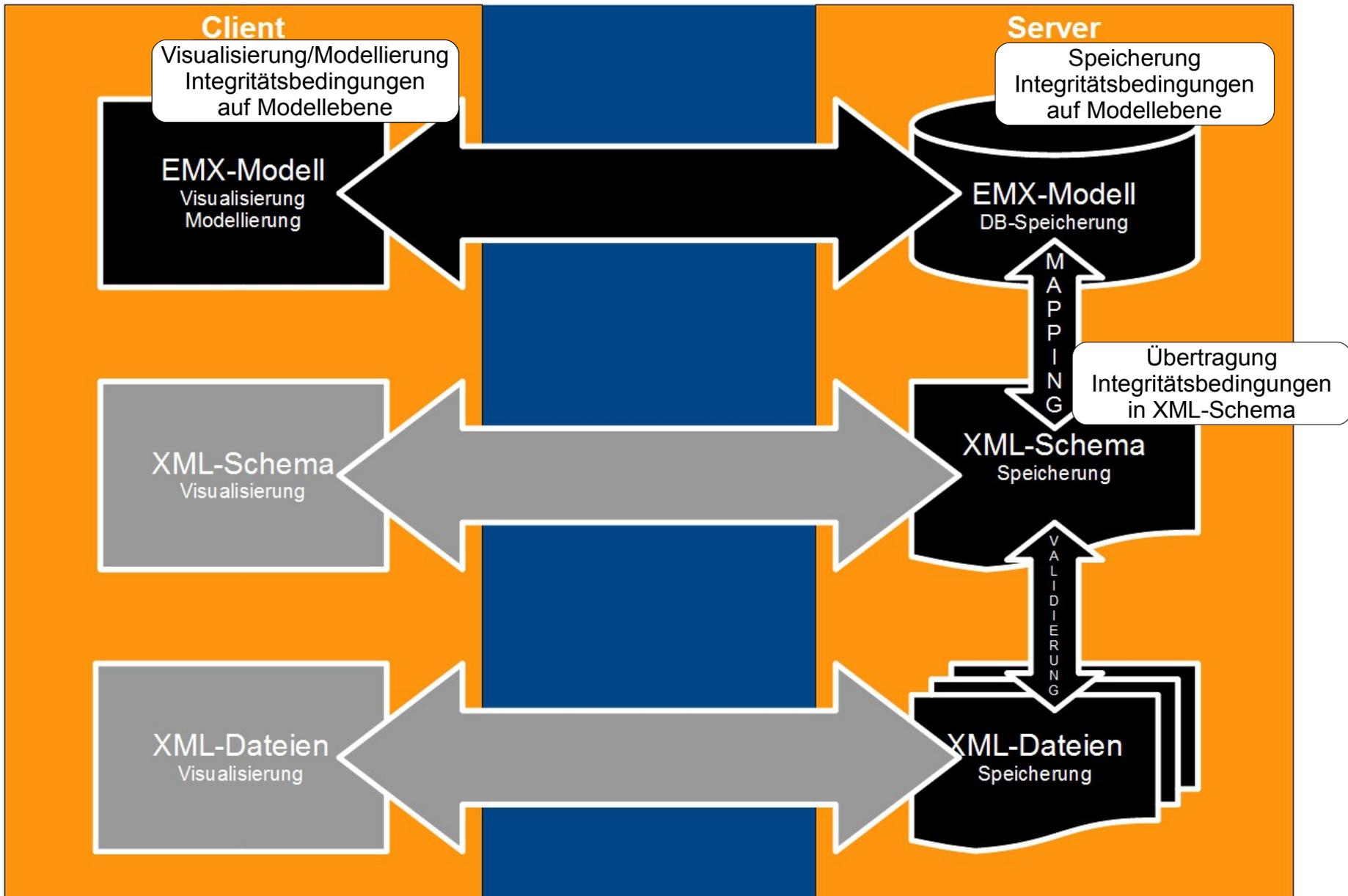
- Selection
- Annotation
- Element
- AttributeGroup
- Modul
- Group
- Constraint
- Connection
- SimpleType

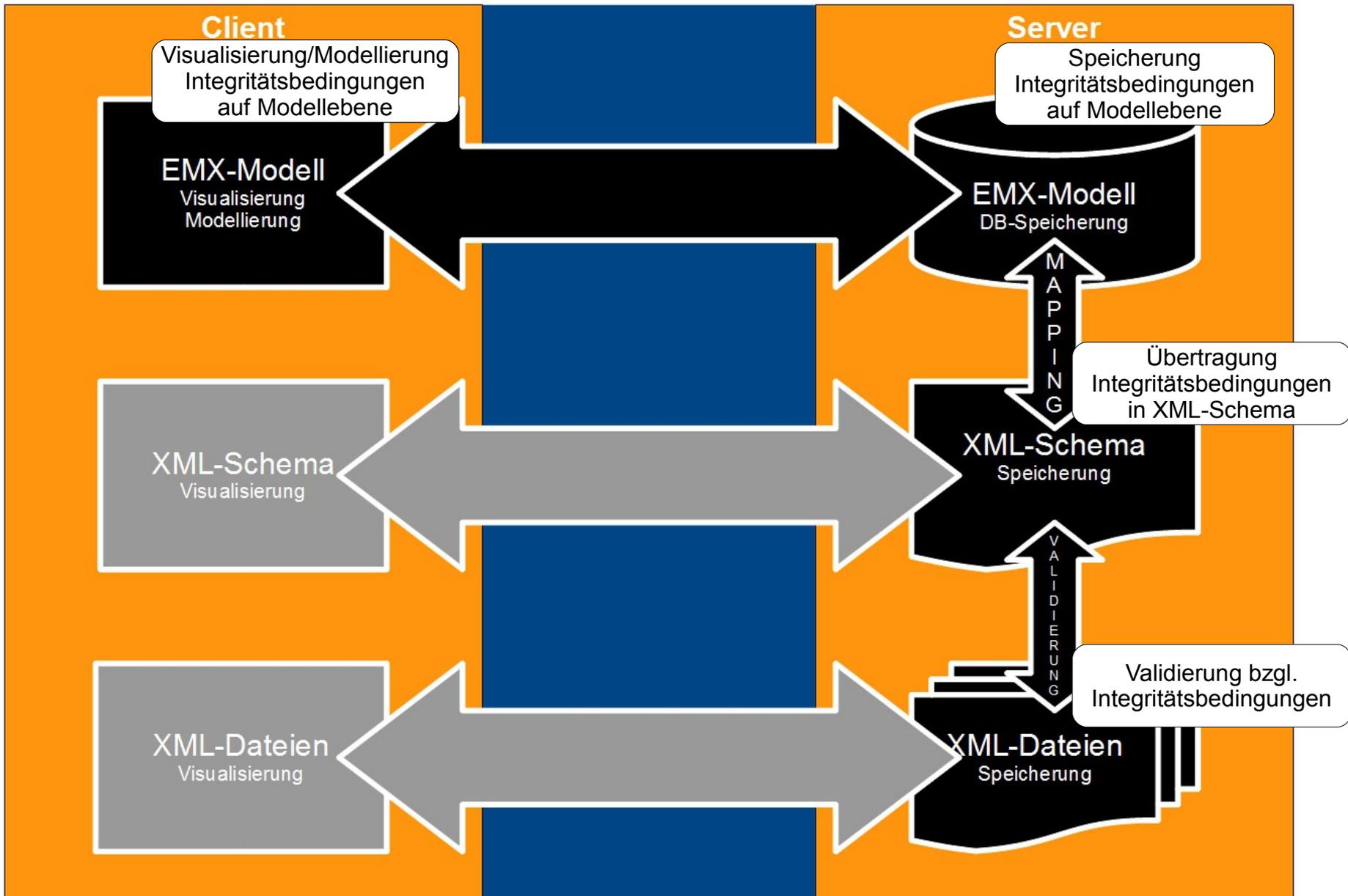
17:54:30 NewEMXEditorDialog EID: 2103175430
17:54:27 NewEMXEditorDialog EID: 2103175427
17:53:44 Entity 15 added.
17:53:44 Entity 14 added.
17:53:44 Entity 13 added.













Werkzeuge

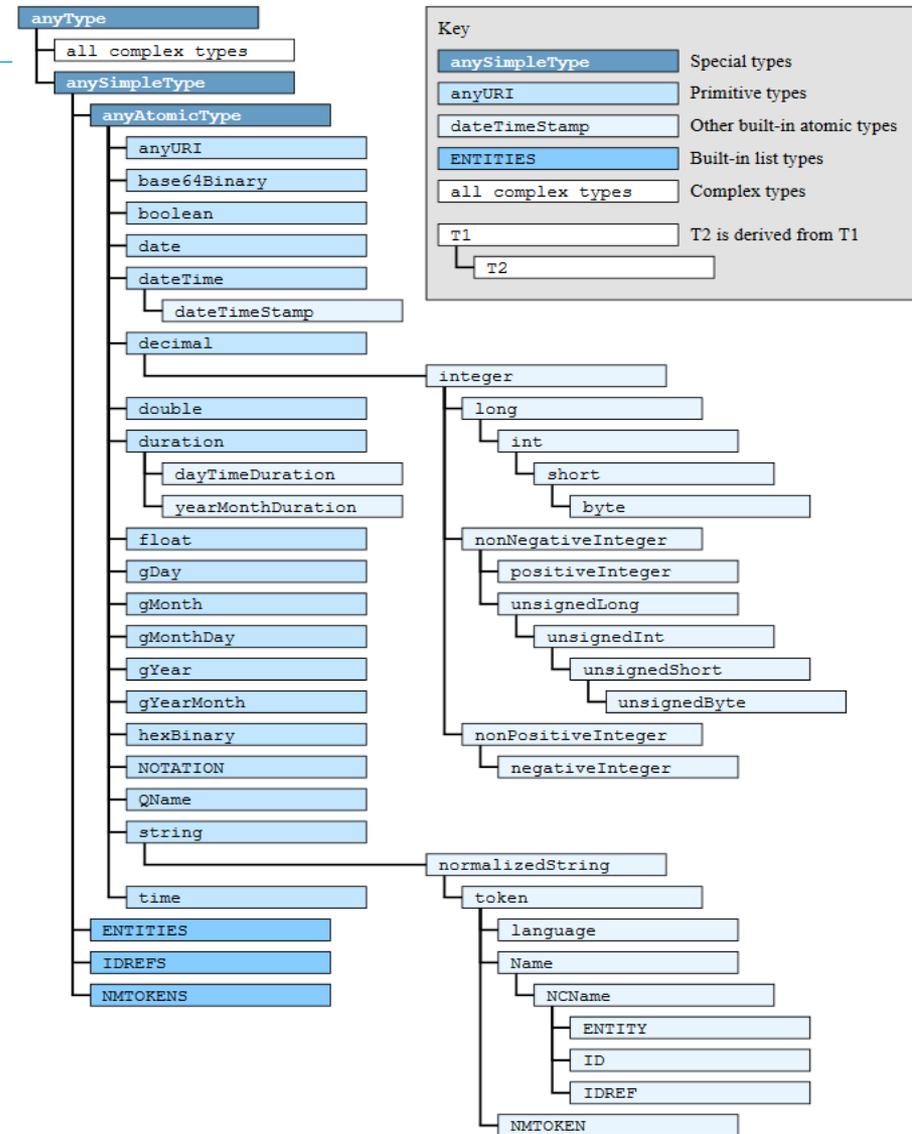
- Programmierung vorwiegend in Java
- Werkzeuge:
 - Google Web Toolkit (Java → JavaScript)
 - MySQL-Datenbank (Zugriff via JDBC)
 - Apache Tomcat (Server)
 - Apache Xerces, Apache Xalan (XML-Verarbeitung)
 - Eclipse (Entwicklungsumgebung)

II. Arten von Integritätsbedingungen

- Wertebereichsintegrität
- Schlüsselintegrität
 - Schlüssel
 - Fremdschlüssel
- Strukturelle Integrität
- Logische Konsistenzbedingungen

Wertebereichsintegrität

- Prüfung, ob der eingegebene Attributwert aus dem Wertebereich des Attributes stammt
- Vordefinierte Datentypen in XML-Schema
 - String
 - Boolean
 - Integer
 - ...
- Benutzerdefinierte Datentypen



Wertebereichsintegrität

- Überprüfung
 - Auf Dokumentenebene:
 - Wird durch Parser bereits geprüft
 - Auf Schema-Ebene:
 - Angaben zu fixed-, default-, ... Attributen können bereits ungültige Werte enthalten
 - Fehler werden i. A. erst beim Validieren der Dokumente entdeckt
 - Forderung: Fehlerhafte Eingaben bereits bei der Modellierung entdecken bzw. verhindern

Wertebereichsintegrität

- Dialogfenster in CodeX gestatten Anwender Definition von XML-Elementen und Attributen
- Angabe von Werten über UI-Komponenten
 - Checkboxes
 - Comboboxen ▼
 - Textfeldern
- Bei Änderung der Angaben automatische Überprüfung
- XML-Datentypen Teil der Java-Bibliothek, aber nicht clientseitig in GWT nutzbar
 - Nachbildung der Datentypen notwendig

Wertebereichsintegrität

- Beispiel: Datentyp Language
- Übergeordneter Datentyp muss erfüllt sein (Token)
- Pattern zusätzlich erfüllen (nach Vorgabe des W3C)

```
public boolean checkDomainIntegrity(String input) {  
    XmlToken t = new XmlToken();  
    if (t.checkDomainIntegrity(input)) {  
        if (input.matches("[a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*")) {  
            return true;  
        }  
    }  
    return false;  
}
```

Wertebereichsintegrität

- Integration in Dialogfenster

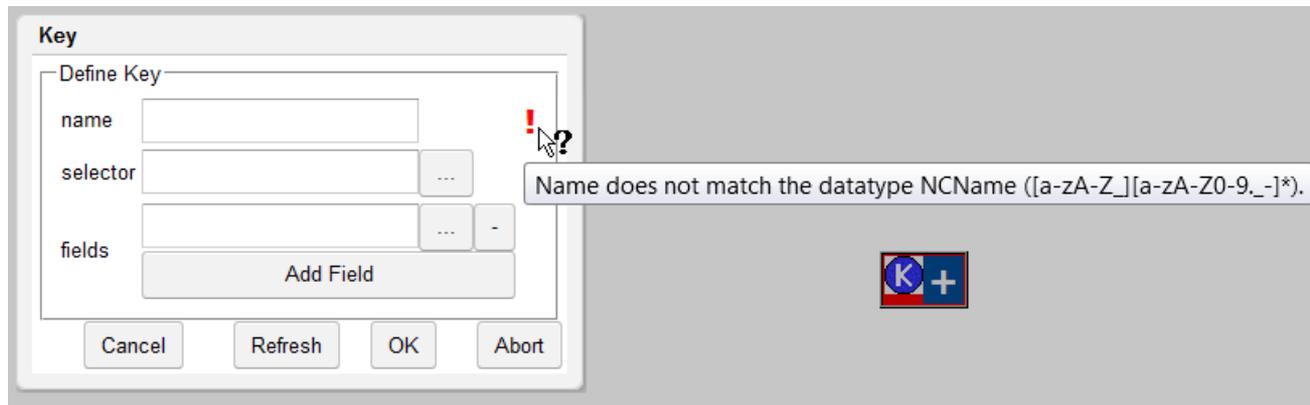
```

textBoxName = new TextBox();
textBoxName.addValueChangeListener(new ValueChangeListener<String>() {
    public void onValueChange(ValueChangeEvent<String> event) {
        String value = event.getValue();
        XmlNCName tester = new XmlNCName();
        nameValid = tester.checkDomainIntegrity(value);
        if(nameValid) {
            nameError.setEnabled(false);
            entity.setHeaderCorrect(true);
        }
        else{
            nameError.setToolTipText("Name does not match the datatype NCName
                                     ([a-zA-Z_][a-zA-Z0-9._-]*)");
            nameError.setEnabled(true);
            entity.setHeaderCorrect(false);
        }
    }
});

```

Wertebereichsintegrität

- Visualisierung



- Rotes Ausrufezeichen um Fehler zu kennzeichnen
- Tooltip um Grund für Fehler zu erklären

Schlüsselintegrität

- Schlüssel-Fremdschlüssel-Beziehung ähnlich wie in relationalen Datenbanken
- Drei Möglichkeiten zur Realisierung in XML-Schema
 - id/idref
 - key/keyref
 - unique
- Sicherstellung Schlüsseleigenschaft durch id, key oder unique
- Referenzierbarkeit mittels idref bzw. keyref

Schlüsselintegrität

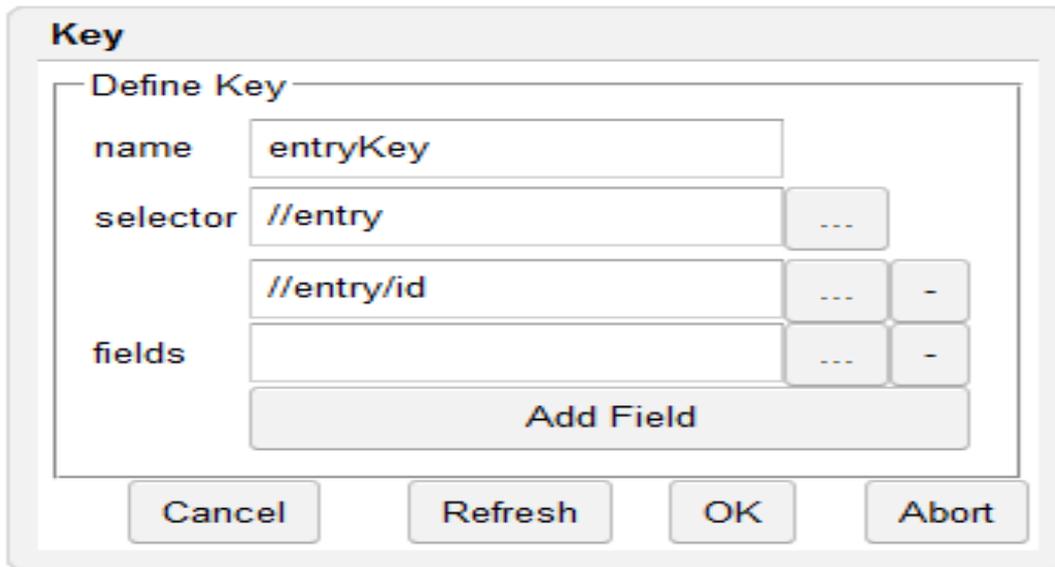
- id
 - Eindeutige Identifizierung eines Elementes in einem Dokument („künstlicher, globaler Schlüssel“)
 - Angabe über Attribut „id“
- key/unique
 - „selector“-Attribut zur Bestimmung einer Elementmenge
 - „field“-Attribute um Elemente/Attribute des Schlüssels festzulegen
 - XPath zur Bestimmung von „selector“ und „field“
- idref/keyref
 - Referenzierung von Elementen, die mit id bzw. key/unique ausgezeichnet wurden

Schlüsselintegrität

- Überprüfung auf Dokument-Ebene
 - Wird durch Parser bereits geprüft
 - Aufgrund unzureichender Fehlermeldung eigene Implementierung
- Formulierung auf Modellebene
 - Dialogfenster zur Unterstützung des Anwenders
 - Festlegung des Typs (id, key...)
 - Unterstützung für selector- und field-Angaben
- Visualisierung von keyref-Fremdschlüsseln durch Verbindungslinien im Modell

Schlüsselintegrität

- Dialogfenster für Definition eines Keys



The image shows a dialog window titled "Key" with a "Define Key" section. It contains the following fields and controls:

- name:** A text input field containing "entryKey".
- selector:** A text input field containing "//entry" with a "..." button to its right.
- fields:** A table with two columns. The first column contains the text "//entry/id". The second column contains a "-" sign. There are "..." and "-" buttons to the right of the first row, and another "..." and "-" button to the right of the second row.
- Add Field:** A button located below the fields table.
- Buttons:** "Cancel", "Refresh", "OK", and "Abort" buttons are located at the bottom of the dialog.

Strukturelle Integrität

- Schema-Komponenten lassen sich nicht beliebig miteinander verbinden
- Vorgaben durch das W3C
- Übernahme der Strukturvorgaben in das Modell
- Fehlerhafte Nutzereingaben abfangen, ggf. korrigieren
- Einhaltung der Struktur in XML-Dokumenten erfolgt beim Parsen

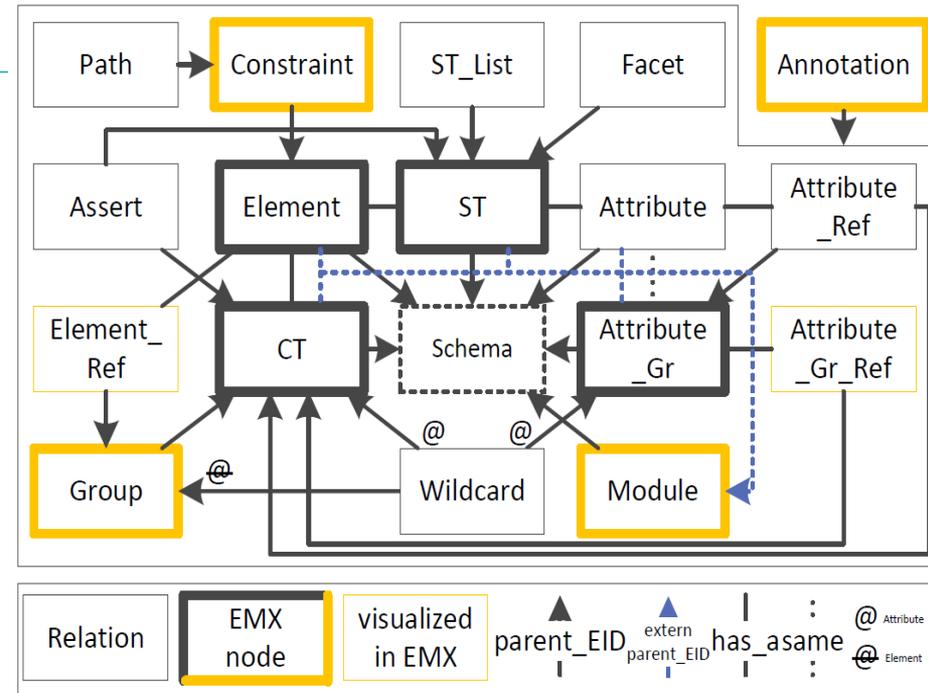


Abb.: EMX-Modell (aus[TN13]¹)

1: Thomas Nösinger (GvD 2013) - A Conceptual Model for the XML Schema Evolution - Overview: Storing, Base-Model-Mapping and Visualization

Strukturelle Integrität

- Annotationen können zu Elementen gehören
- Annotationen können nicht selbst Annotationen enthalten

```

public void createConnection() {
    ...
    if(firstClicked instanceof EmxElementModel) {
        if(secondClicked instanceof EmxAnnotationModel) {
            addAndDrawValidConnection(firstClicked, secondClicked);
            return;
        }
        ...
    }
    ...
    if(firstClicked instanceof EmxAnnotationModel) {
        if(!secondClicked instanceof EmxAnnotationModel) {
            addAndDrawValidConnection(secondClicked, firstClicked);
            return;
        }
    }
    ...
}

```

Logische Konsistenz

- Integritätsbedingungen, die nicht (oder nur sehr umständlich) durch Struktur, Wertebereich und Schlüsselbedingungen beschrieben werden können.
- In XML-Schema 1.0 existierten keine Möglichkeiten um komplexere Integritätsbedingungen zu formulieren
- Forschung/Industrie entwickelte Ansätze um XML-Schema zu erweitern bzw. Entwicklung von eigenen Schemasprachen
 - Schematron
 - RelaxNG
 - SchemaPath
 - Object Constraint Language (in UML-Modellen)
 - ...

Logische Konsistenz

- Mit v. 1.1 wurde XML-Schema um einige Features erweitert:
 - Neue Datentypen
 - Bedingte Typisierung
 - Schemaweite Attribute
 - Flexibleres Schema durch `<openContent>` und `<defaultOpenContent>`
 - Versionsverwaltung
 - **Logische Konsistenzbedingungen** (Assertions)
- Recommendation seit April 2012



Logische Konsistenz

- Assertions lassen sich für einfache und komplexe Typen definieren
- Vereinfachen z.B. die Definition von Datentypen und den Wertevergleich zwischen verschiedenen Elementen
- Formulierung mittels XPath-Functions (vom W3C)
- Geringe Unterstützung der neuen Version in aktuellen Systemen
- Überprüfung in Dokumenten mittels Apache Xerces-J (Beta) oder Saxon EE

Logische Konsistenz

- “Früher”:

```
<simpleType name='triple'>
  <restriction base='integer'>
    <pattern value=
      ' ([0369] | [147] [0369]* [258] | (([258] | [147] [0369]* [147])
      ([0369] | [258] [0369]* [147]) * ([147] | [258]
      [0369]* [258])) *' />
    </restriction>
  </simpleType>
```

- Assertion für einfache Typen:

```
<simpleType name='triple'>
  <restriction base='integer'>
    <assertion test='$value mod 3 eq 0' />
  </restriction>
</simpleType>
```

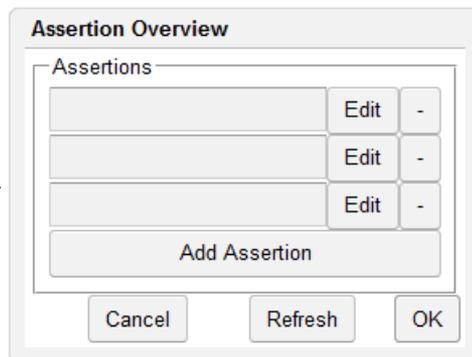
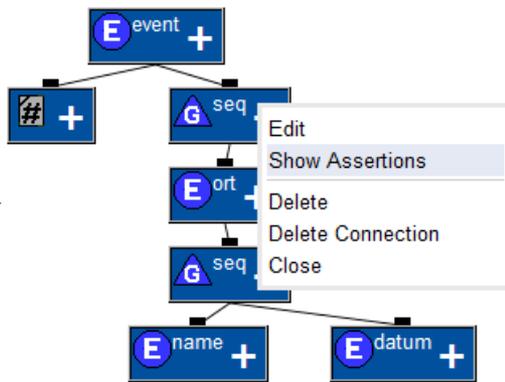
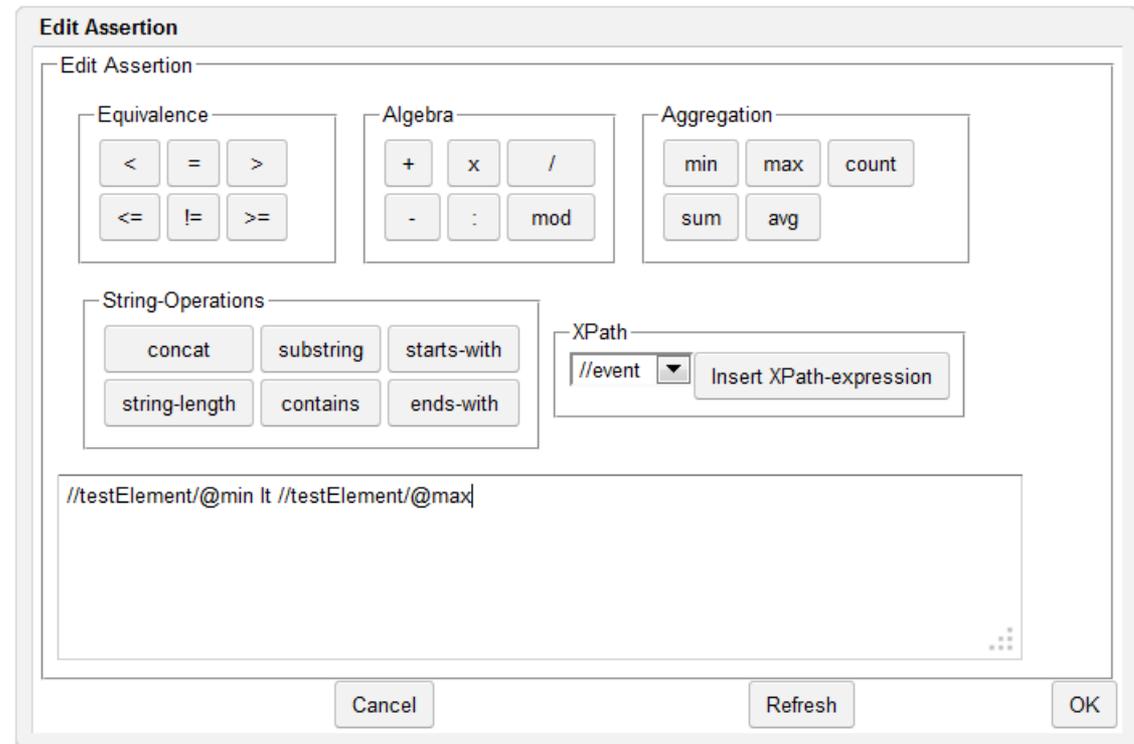
Logische Konsistenz

- Assertion für komplexe Typen
- Datum Publikation muss vor Korrektur („updated“) liegen

```
<xs:complexType name="entryType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="id" type="atom:idType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="author" type="atom:personType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="contributor" type="atom:personType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="title" type="atom:textType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="content" type="atom:contentType" minOccurs="0" maxOccurs="1"/>
    [...]
    <xs:element name="published" type="atom:dateTimeType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="updated" type="atom:dateTimeType" minOccurs="1" maxOccurs="1"/>
  </xs:choice>
  <xs:assert test="published <= updated"/>
</xs:complexType>
```

Logische Konsistenz

- Unterstützung bei der Formulierung durch Toolbox

Logische Konsistenz

- Übersetzungsprozess
 - Manipulation des DOM
 - Übernahme der ausformulierten Bedingung in das test-Attribut einer Assertion
 - Einfügen unterhalb des einfachen bzw. komplexen Typs
- Beispiel
 - CT „entryType“
 - Bedingung „published le updated“

```
<xs:complexType name="entryType">
```

```
...
```

```
  <xs:assert test="published le updated"/>
```

```
</xs:complexType>
```

III. Transformationsprozess

- Neben der Evolution des Schemas müssen auch ggf. die XML-Dokumente angepasst werden
- Evolutions-Engine in der Neuimplementation von CodeX noch nicht programmiert
- Bisherige Engine umfasste Wertebereich und Struktur
- In Masterarbeit wird gezeigt, wie sich Integritätsbedingungen bzgl. Schlüssel und logischer Konsistenzbedingungen auf
 - insert-
 - update- und
 - delete-Operationenauf Attributen und Elementen zurückführen lassen.



Transformationsprozess

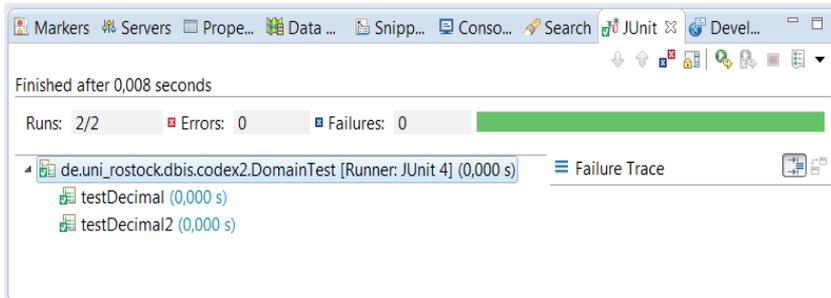
- Bisheriger Transformationsprozess:
 - Logging der Evolutionsschritte
 - Ermittlung der minimalen notwendigen Änderungen an XML-Dokumenten
 - Erzeugung eines XSLT-Skripts zur Adaption
 - Anwendung des Skripts auf die Dokumente
- Neuimplementation von CodeX verwendet neue Sprache (ELaX) um Evolutionsschritte darzustellen
 - Mapping von ELaX zu XSLT-Skript bzw. DOM-Operationen fehlt zur Zeit

IV. Testszenario

- Beispielschema: Atom (ähnlich zu RSS)
- Insbesondere Tests zum Wertebereich und zur logischen Konsistenz
- Wertebereich:
 - Austesten aller Datentypen mit JUnit-Tests
 - Beispielwerte vom W3C
 - Randwerte
 - Manuelle Prüfung der Visualisierung im Editor
- Logische Konsistenz:
 - Erweiterung von komplexen Typen um Assertions
 - Überprüfung von Dokumenten mittels XML-Parser

TestszENARIO

```
<xs:complexType name="entryType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="id" type="atom:idType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="author" type="atom:personType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="contributor" type="atom:personType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="title" type="atom:textType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="content" type="atom:contentType" minOccurs="0" maxOccurs="1"/>
    [...]
    <xs:element name="published" type="atom:dateTimeType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="updated" type="atom:dateTimeType" minOccurs="1" maxOccurs="1"/>
  </xs:choice>
  <xs:assert test="published le updated"/>
</xs:complexType>
```



Information

2 errors occurred while validating the XML-file:

- Assertion evaluation ('@max ge @min') for element 'test' with type '#anonymous' did not succeed.
- Assertion evaluation ('@max ge @min') for element 'test' with type '#anonymous' did not succeed.

OK



Danke für Ihre Aufmerksamkeit.

Fragen?

Anmerkungen?

Feedback?

Quellen (Auszug)

- **CodeX**: Thomas Noesinger (GvD 2013) - A Conceptual Model for the XML Schema Evolution - Overview: Storing, Base-Model-Mapping and Visualization
- **XML-Schema v.1.1 Recommendation**: <http://www.w3.org/TR/2011/CR-xmlschema11-1-20110721/>
- **XPath-Functions Recommendation**:
<http://www.w3.org/TR/xpath-functions/>
- **Überblick Neuerungen XML-Schema v.1.1**:
<http://www.oio.de/public/xml/w3c-xml-schema-1-1-ueberblick.htm>
- **System-Überblick**: J an Deffke 2012 - XML-Schema Evolution: Evolution in der Praxis (Bachelorarbeit)
- **Kostenanalyse Schemaevolution**: Hannes Grunert 2011 - XML-Schema Evolution: Kategorisierung und Bewertung (Bachelorarbeit)
- **XSLT-Übersetzung**: Hannes Grunert 2012 - XML-Schema Evolution: Integration XSLT und XUpdate in CodeX (Projektarbeit)