Universität Rostock

Traditio et Innovatio

Bachelorarbeit
XML-Schema-Evolution: Kategorisierung und Bewertung

Hannes Grunert

© 2011 **UNIVERSITÄT ROSTOCK** | FAKULTÄT FÜR INFORMATIK UND ELEKTROTECHNIK

Einführendes Beispiel

Angestrebter Abschluss	Studiengang	Fachsemester
Bachelor	Informatik	9
Master	Informatik	1

```
<studium>
  <erststudium>
    <abschluss>Bachelor</abschluss>
    <studiengang>Informatik</studiengang>
    <semester>9</semester>
  </erststudium>
  <zweitstudium>
    <abschluss>Master</abschluss>
    <studiengang>Informatik</studiengang>
    <semester>1</semester>
  </zweitstudium>
</studium>
```

Einführendes Beispiel – XML-Schema (Ausschnitt)

```
<xsd: element name="studium">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="erststudium" minOccur="0" maxOccur="1"/>
      <xsd:element ref="zweitstudium" minOccur="0" maxOccur="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd: element>
```

Evolution

```
<xsd: element name="studium">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="erststudium" minOccur="0" maxOccur="1"/>
      <xsd:element ref="zweitstudium" minOccur="0" maxOccur="unbounded"/>
    </xsd:choice>
  </xsd:complexType>
</xsd: element>
```

Einführendes Beispiel

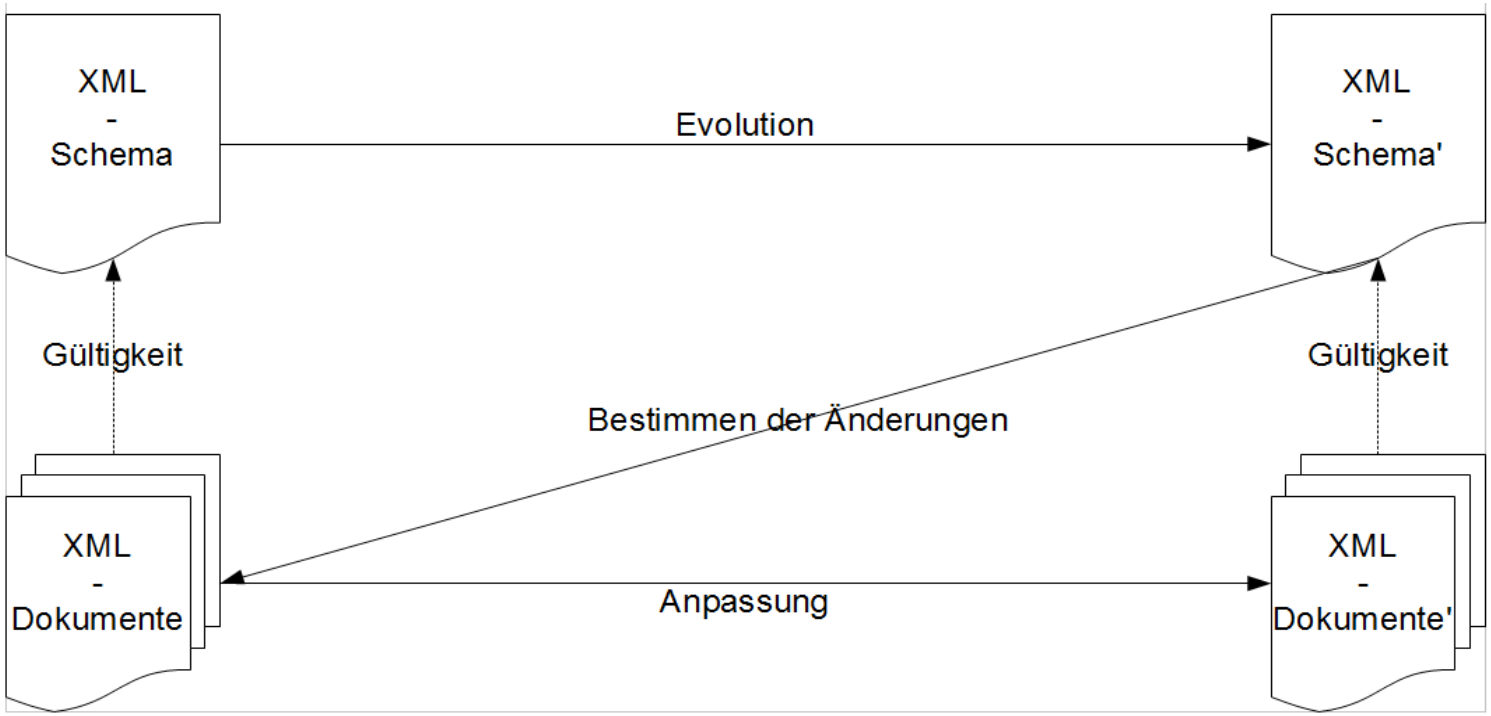
Angestrebter Abschluss	Studiengang	Fachsemester
Bachelor	Informatik	9

```
<studium>
  <erststudium>
    <abschluss>Bachelor</abschluss>
    <studiengang>Informatik</studiengang>
    <semester>9</semester>
  </erststudium>
  <!-- Eintrag durch Evolution entfernt -->
</studium>
```

Inhalt

XML-Schema-Evolution

- Neue Version eines XML-Schemas
- Überprüfen der Gültigkeit von Instanzen
- Ggf. Anpassung der Instanzen



Kategorisierungskriterien Schema-Evolution

- Was wird geändert? (SimpleType, ComplexType, Attribute)
- Wie wird es verändert? (Löschen, Hinzufügen, Update)
- **Wird es verändert?**
- **Änderung der Informationskapazität**
  - erweiternd
  - reduzierend
  - erhaltend
  - verändernd

Beispiel – Schema (Ausschnitt)

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Name"/> <!-- nicht-optionaler simpleType -->
      <xsd:element ref="Gesinnung"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Evolution

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:attribute name="Name" use=”required” type="xsd:string"/>
    <xsd:sequence>
      <xsd:element ref="Gesinnung"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Beispiel - Instanz

```
<Person>
  <Name>”Thomas”</Name>
  <Gesinnung>rechtschaffen neutral</Gesinnung>
</Person>
```

Transformation

```
<Person name="Thomas">
  <Gesinnung>rechtschaffen neutral</Gesinnung>
</Person>
```

Beispiel - Kategorisierung

- Was wird geändert?

  → Complex Type

- Wie wird es verändert?

  → Umwandlung eines Elementes in ein Attribut

- **Wird es verändert?**

  → **Ja**

- **Änderung der Informationskapazität**

  → **erhaltend**

Kategorisierungen in anderen Arbeiten

- Betrachtete Kategorisierungen erfüllen nicht alle Kriterien oder beziehen sich nicht auf XML-Schema als Schemasprache
- Menge der Evolutionsschritte von Christian Will am umfangreichsten

Kategorisierung	Sprache	Vollständigkeit	Komponente	Art	Kapazität	Auswirkung
XEM	DTD	+	+	+	-	+
Andre Zeitz	DTD	+	+	+	+	+
Tan, Goh	XML-Schema	-	+	+	-	-
Guerrini, Mesiti, Rossi	XML-Schema	-	+	+	-	+
Universität Leipzig	XML-Schema	-	-	+	+	(+)
Moro, Malaika, Lim	XML-Schema	-	(+)	+	-	-
Cicchetti, Di Ruscio et al.	Meta-Modelle	-	-	+	-	+
Universität Prag	XML-Schema	-	+	+	-	(+)
Christian Will	XML-Schema	+	+	+	-	(+)

\+ : Kriterium betrachtet
(+) : Kriterium teilweise betrachtet
\- : Kriterium nicht betrachtet

Kategorisierung nach Will ([Wil06])

- Umfangreichste Sammlung von Evolutionsschritten
  - Bezieht Beziehungen, Identitätsdefinitionen und Anmerkungen mit ein
- 1. Stufe: Was wird geändert?
- 2. Stufe: Wie wird es geändert
- Kurze Anmerkung, dass auch nach Auswirkungen auf Instanzen kategorisiert werden kann

Eigene Kategorisierung

- 1. Stufe: für Bewertung wichtig, ob die Änderung(en)
  - Keine Auswirkung auf Instanzen
  - Möglicherweise Auswirkung auf (einige) Instanzen
  - Auf jeden Fall Auswirkung auf Instanzen hat
- 2. Stufe: Art der Änderung
  - Änderung der Informationskapazität
    - Abschätzen, wie viele Informationen verloren gehen und neu eingeben werden müssen
  - Update-Operationen teurer, da Hintereinanderausführung von Insert und Delete
- 3. Stufe: Was wird geändert?

Eigene Kategorisierung

- Verfeinerung einzelner Evolutionsschritte notwendig
- Beispiel: Änderungen des Typs eines komplexen Elementes
  - Typ entweder *sequence*, *choice* oder *all*
  - → 6 Fälle:
    - sequence → all
    - sequence → choice
    - choice → all
    - choice → sequence
    - all → sequence
    - all → choice
  - Unterschiede in Auswirkung auf Instanzen und Informationskapazität

Kosten

- Anzahl betroffener Dokumente (davon wird zunächst abstrahiert)
- „Umfang" der Komponente, die durch Insert-, Delete- und Update-Operationen verändert wird
  - Start- und End-Tags
  - Attribute
  - Werte von Elementen
- Anzahl der betroffenen Instanzen der betroffenen Komponente
  - Abschätzung, wie oft Komponente in einer Instanz vorkommt

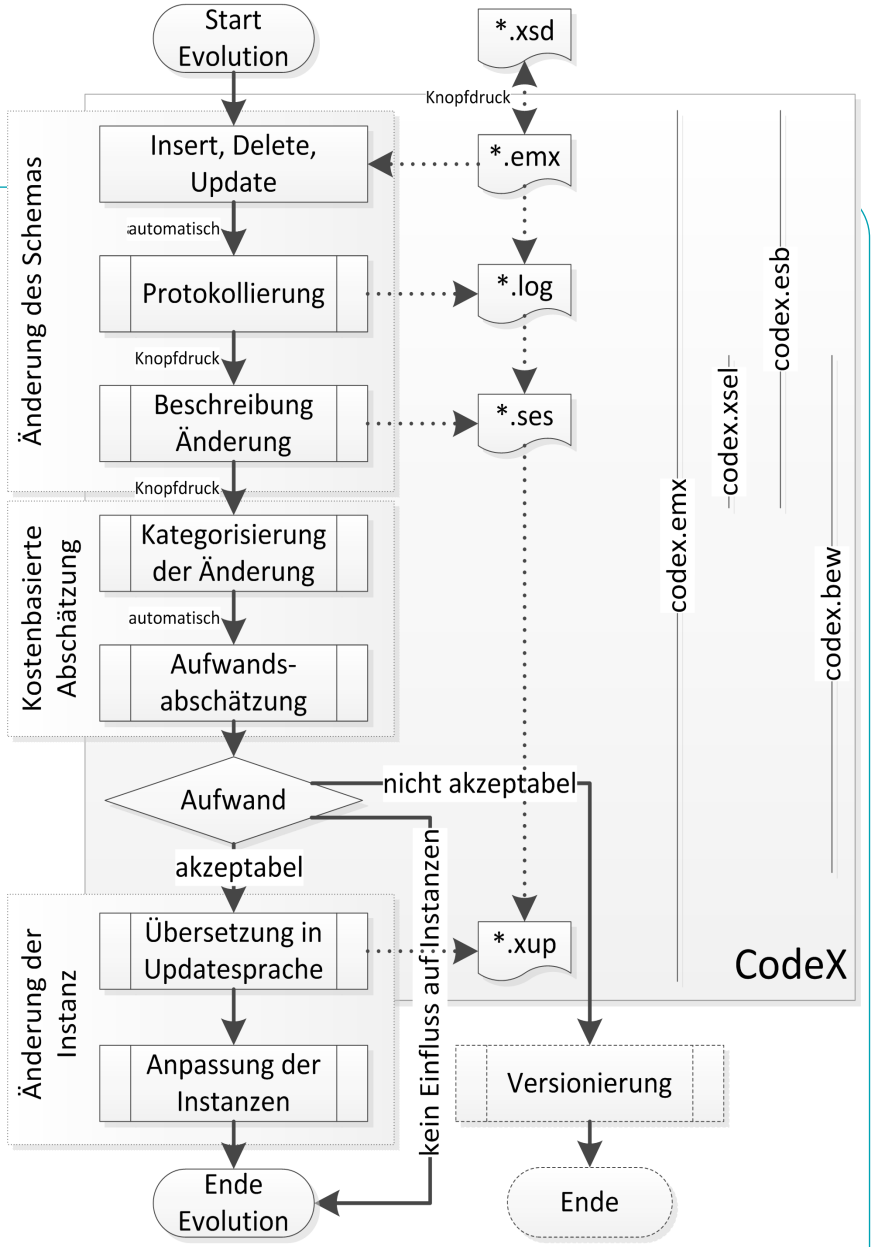
Pseudocode Entscheidung

Eingabe: XML-Schema xs, Liste Evolutionsschritte list, maximale Kosten max_cost
Ausgabe: Evolution/Versionierung

```
current_cost := 0;
for (schritt in list)
    current_cost += calcCost(schritt, xs);
    if(current_cost > max_cost)
        return Versionierung;
return Evolution;
```

praktische Umsetzung

- CodeX
  - Plugin für Eclipse
  - Grafischer Editor für XML, entstand im Rahmen mehrerer studentischer Arbeiten an der Universität Rostock
  - Programmiersprache: Java
  - Eclipse RCP

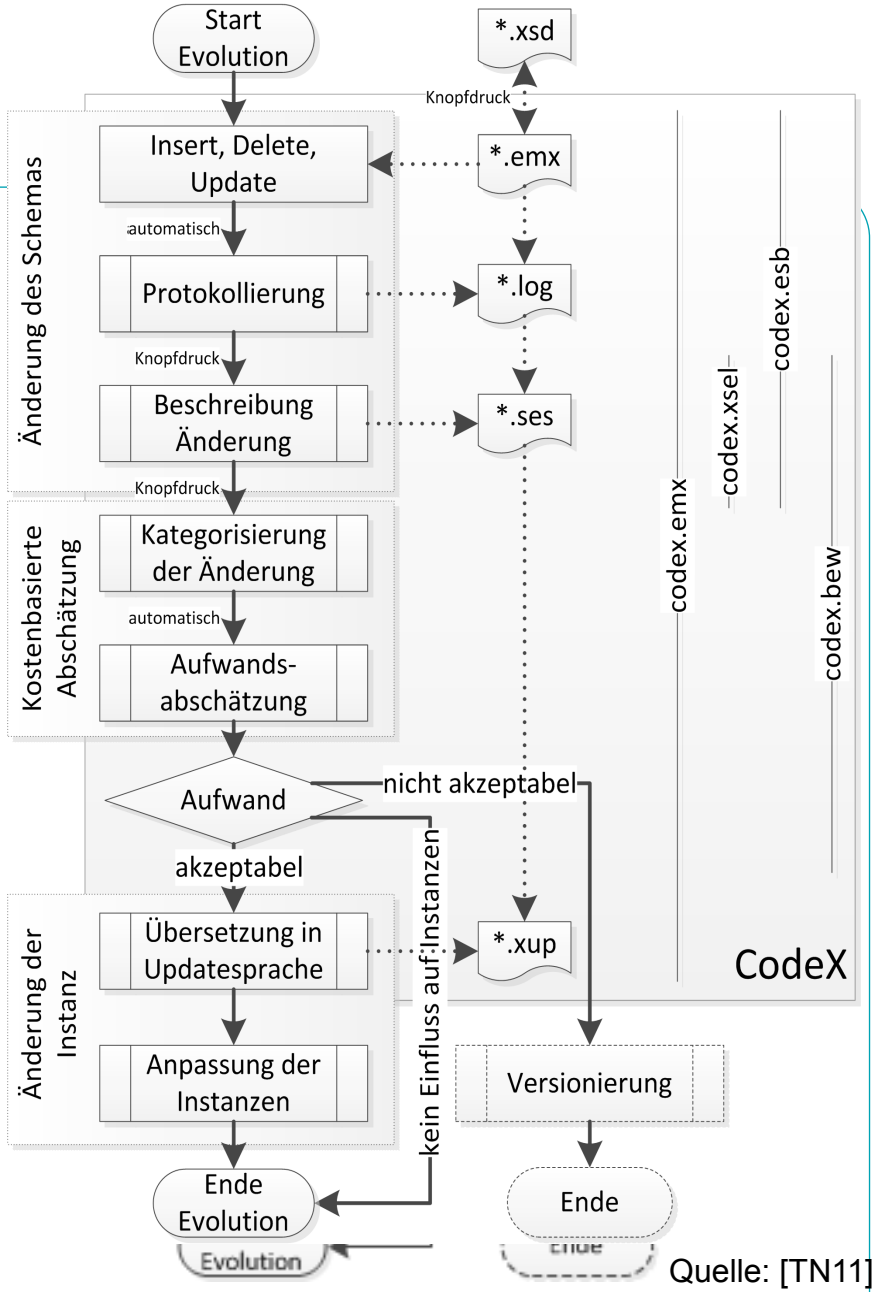


Quelle: [TN11]

praktische Umsetzung (2)

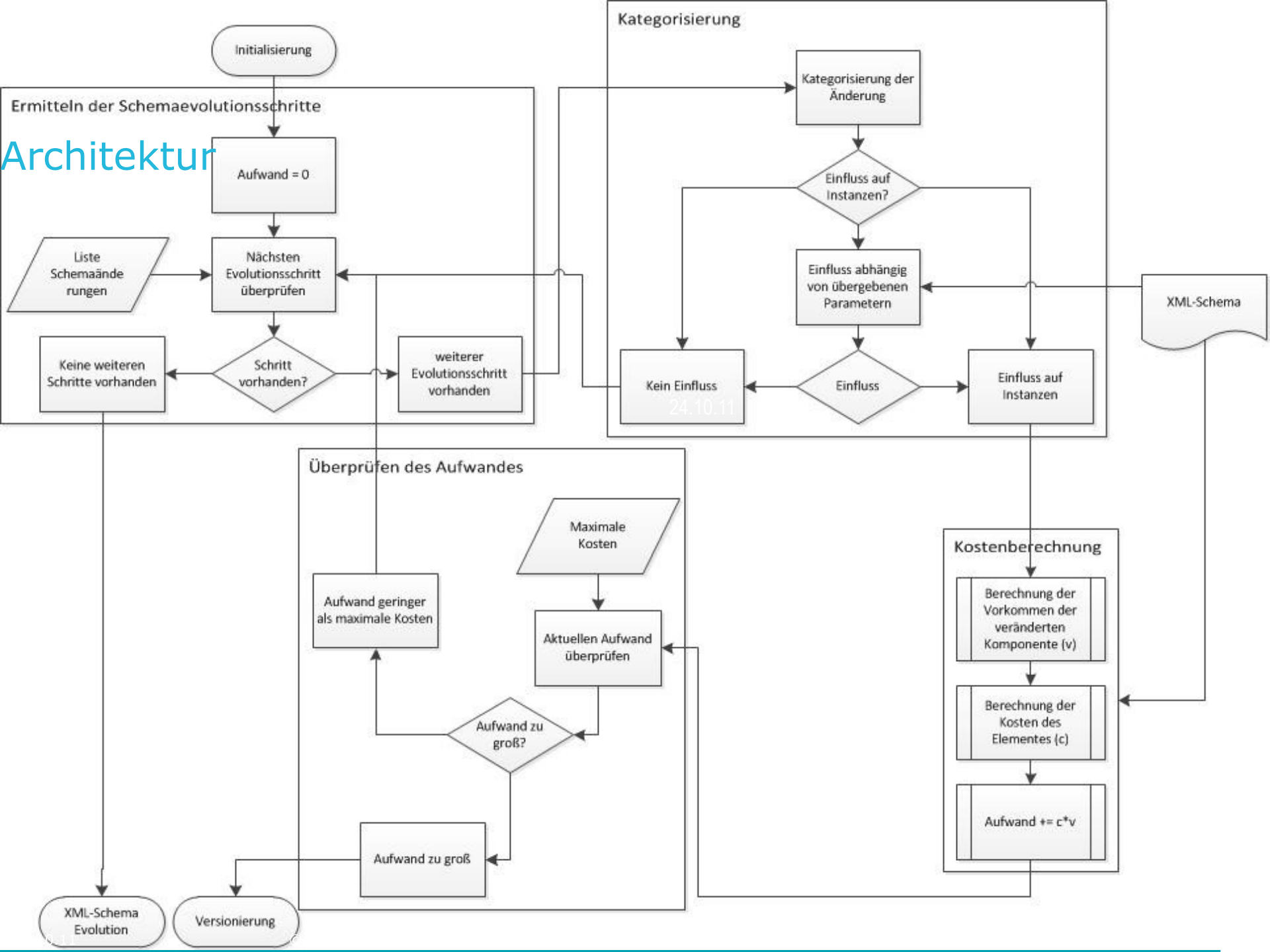
- codex.bew
  - Eingabe: Datei mit Schema-evolutionsschritten (*.ses)
  - Berechnung der Kosten aller Schritte
  - Ausgabe
    - Versionierung

      -oder-
    - Schemaevolution



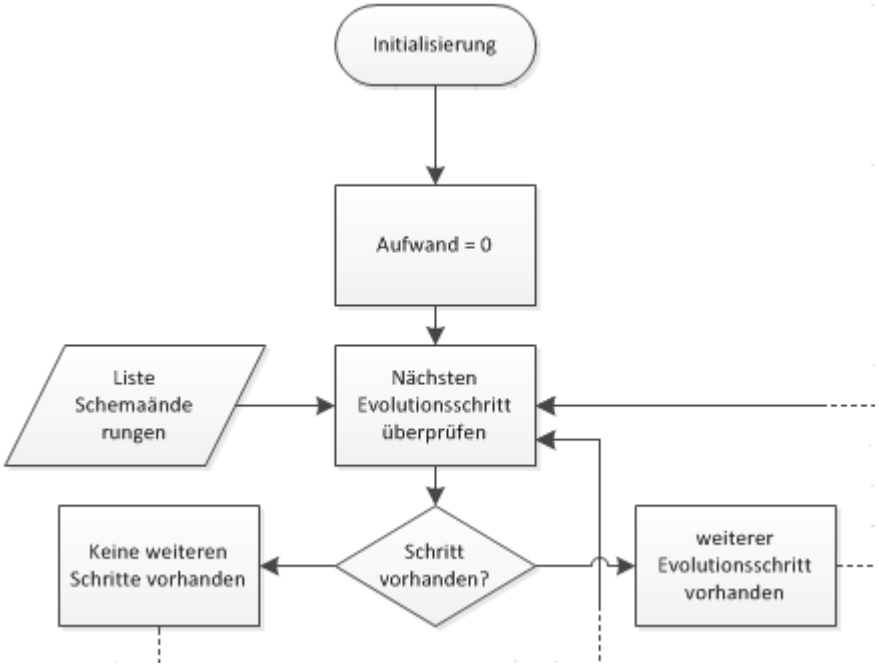
Quelle: [TN11]

Architektur



Architektur – Ermittlung der Evolutionsschritte

- Evolutionsschritte liegen in Datei vor (CodeX: *.ses-File)
- Charakterisierung durch Namen und ggf. Parameter
- Jeder Schritt wird als Objekt in einer Liste abgespeichert



- Beim Start und nach Abarbeitung eines Evolutionsschrittes, der die maximalen Kosten nicht überschreitet, wird Liste überprüft.
  - Liste leer → Evolution wird durchgeführt
  - Sonst → nächsten Evolutionsschritt abarbeiten

Architektur – Ermittlung der Evolutionsschritte(2)

```
public static boolean calcAllCostWithLimit(Collection<ChangeObject>
evolutionSteps, int limit){

    boolean isHigh = false;

    double sumOfAllSteps = 0;

    for(Iterator<ChangeObject> iterator = evolutionSteps.iterator();
    iterator.hasNext();){

        …

    }
```

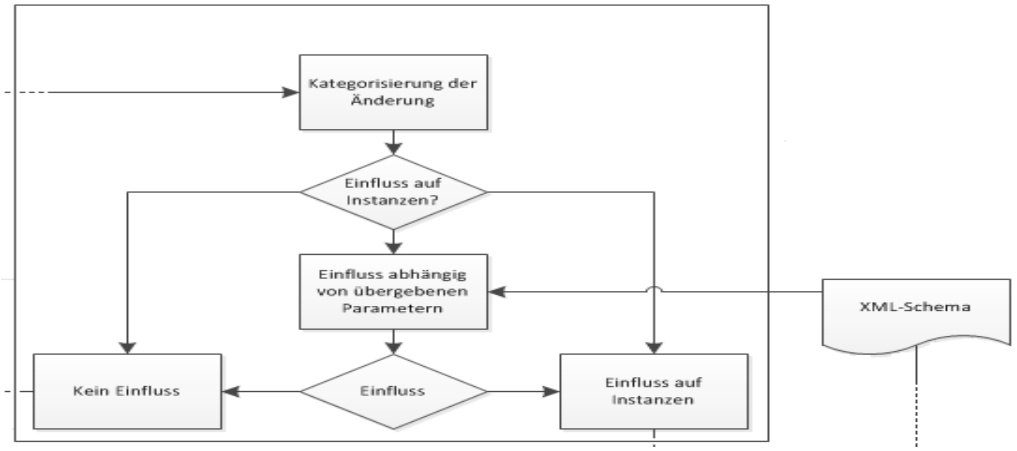
Konvertierung *.ses → Java-Objekte

```
<insert_after select="//xsd:complexType[@name =
'EmxComplexType_4']/xsd:sequence"
after="xsd:element[@name = 'EmxElement_20']"
content="&lt;xsd:element ref=&quot;EmxElement_21&quot;
/&gt;" />
```

```
New ChangeObject(ChangeType.addElementToComplexElement,
{EmxElement_21, EmxComplexType_4})
```

Architektur - Kategorisierung

- Ziel der Kategorisierung: unnötige Berechnungen vermeiden



- Schemaänderung ohne Einfluss auf Dokumente:
  - Keine weiteren Berechnungen nötig
  - Ggf. erst unter Betrachtung des Schemas erkennbar
  - Bsp.: Löschen eines Attributes mit use=”prohibited“
  - Fortfahren mit nächsten Evolutionsschritt
- Schemaänderung mit Einfluss auf Instanzen:
  - Berechnung des Aufwandes für diesen Evolutionsschritt

Architektur – Kategorisierung (2)

- Darstellung einzelner Schemaänderung:

New ChangeObject(ChangeType.addElementToComplexElement, {EmxElement_21, EmxComplexType_4})

  - Name der Änderung
    - Identifikation
    - Kategorisierung
  - Liste mit Parameter (betroffene Elemente, Werte, etc)
    - Berechnung
    - Ggf. zur Kategorisierung

Architektur – Kategorisierung (3)

- Umsetzung der Kategorisierung:
  - Zuordnung: Name der Schemänderung → Zahl

```
enum ChangeType{
    addAnnotation(10001), deleteAnnotation(10002),...
    addLocalElementGroupDefinition(20109),...
    public int category;
    private ChangeType(int value){
    this.category = value;
    }
};
```

  - 1xxxx – Keine Auswirkung auf Instanzen
  - 2xxxx – Auswirkung abhängig von Parametern
  - 3xxxx – Auf jeden Fall Auswirkung auf Instanzen

Architektur – Kategorisierung (3)

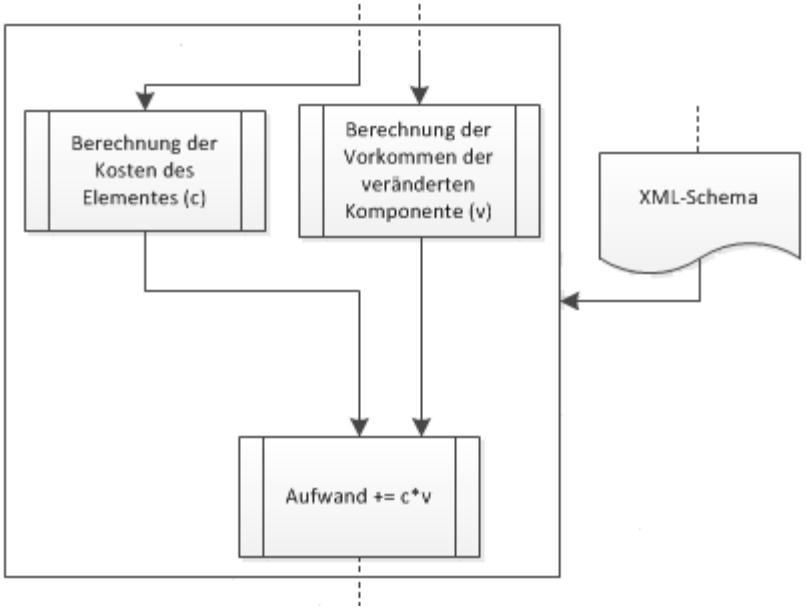
- Mehrstufige Abfrage des Typs der Änderung

```
public double calculateCost(ChangeObject evolutionStep){
  double cost = -1;
  switch(evolutionStep.getType().category/10000){ //Hauptkategorie
    case 1: return 0;//Änderungen ohne Auswirkungen auf Instanzen
    case 2:switch(evolutionStep.getType().category/100){
      case 201: //Änderungen am Schema{
        switch(evolutionStep.getType()){
          case addLocalTypeDefinition:{
            cost = addLocalTypeDefinition();
            return cost;
          }
          ...
        }
      }
    ...
    }
  }
```

- Update: Abbilden auf Zahl entfällt mit Java7, da switch-Anweisungen auf Strings möglich sind

Architektur – Aufwand abschätzen

- Abschätzung, wie oft das veränderte Objekt oder Attribut in einem Dokument vorkommt
- Ermittlung der Kosten eines einzelnen Elementes inkl.:
  - Attribute
  - Sub-Elemente
- Berücksichtigung der Optionalität der Komponenten
  - z.B. *use*-Wert von Attributen
  - *minOccurs*- und *maxOccurs*-Werte
  - ComplexType: *sequence*, *choice* oder *all*



Architektur – Aufwand abschätzen (2)

- Beispiel: Element in complexType einfügen

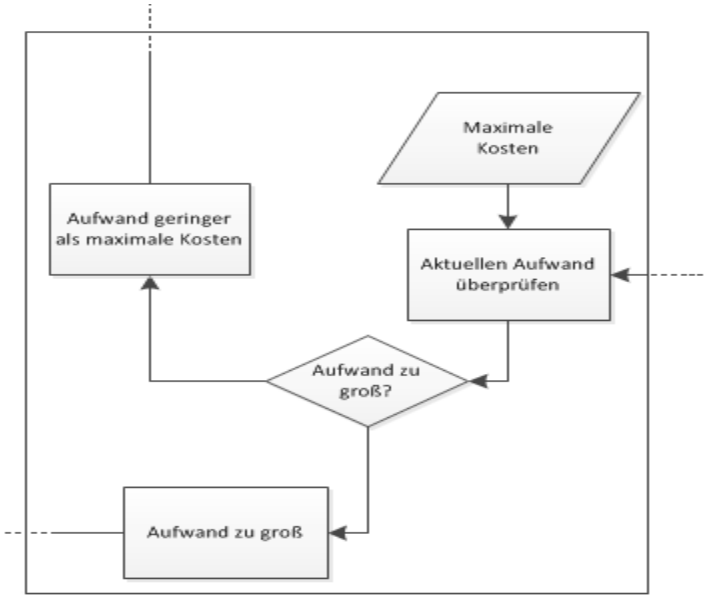
```
public double addElementToComplexElement(String element, String parent){
  String type = sParser.getCEtype(parent);
  if(type.equals("xsd:sequence")){
    return sParser.calc_num_elements(parent)*sParser.get_element_cost(element);
  }
  if(type.equals("xsd:all")){
    return 0;
  }
  if(type.equals("xsd:choice")){
    return 0;
  }
  return -1; //Fehler
}
```

Architektur – Aufwand abschätzen (3)

- Zwei Faktoren bestimmen Aufwand der Änderung:
  - Wie oft kommt veränderte Komponente in einer Instanz vor?
    - Überprüfung der minOccurs- und maxOccurs-Werte sowie des Typs der complexTyps aller Elemente von der betroffenen Komponente aufsteigend bis zur Wurzel des Schemas
  - Welchen Aufwand verursacht die betroffenen Komponente?
    - Insbesondere bei Insert- und Delete-Operationen
    - Rekursive Ermittlung aller Attribute und Sub-Elemente
- Gesamtkosten = Vorkommen*Aufwand

Architektur – Überprüfen der Kosten

- Nach Berechnung der Kosten eines Evolutionsschrittes erfolgt Überprüfung der Gesamtkosten
- Abgleich mit benutzerdefinierten maximalen Kosten
  - Akueller Gesamtaufwand geringer als maximale Kosten → nächsten Evolutionsschritt aus der Liste bearbeiten
  - Aktueller Gesamtaufwand überschreitet maximale Kosten → Abbruch der Schemaevolution, Ausgabe an Nutzer, dass Versionierung zu bevorzugen ist.



Architektur – Überprüfen der Kosten (2)

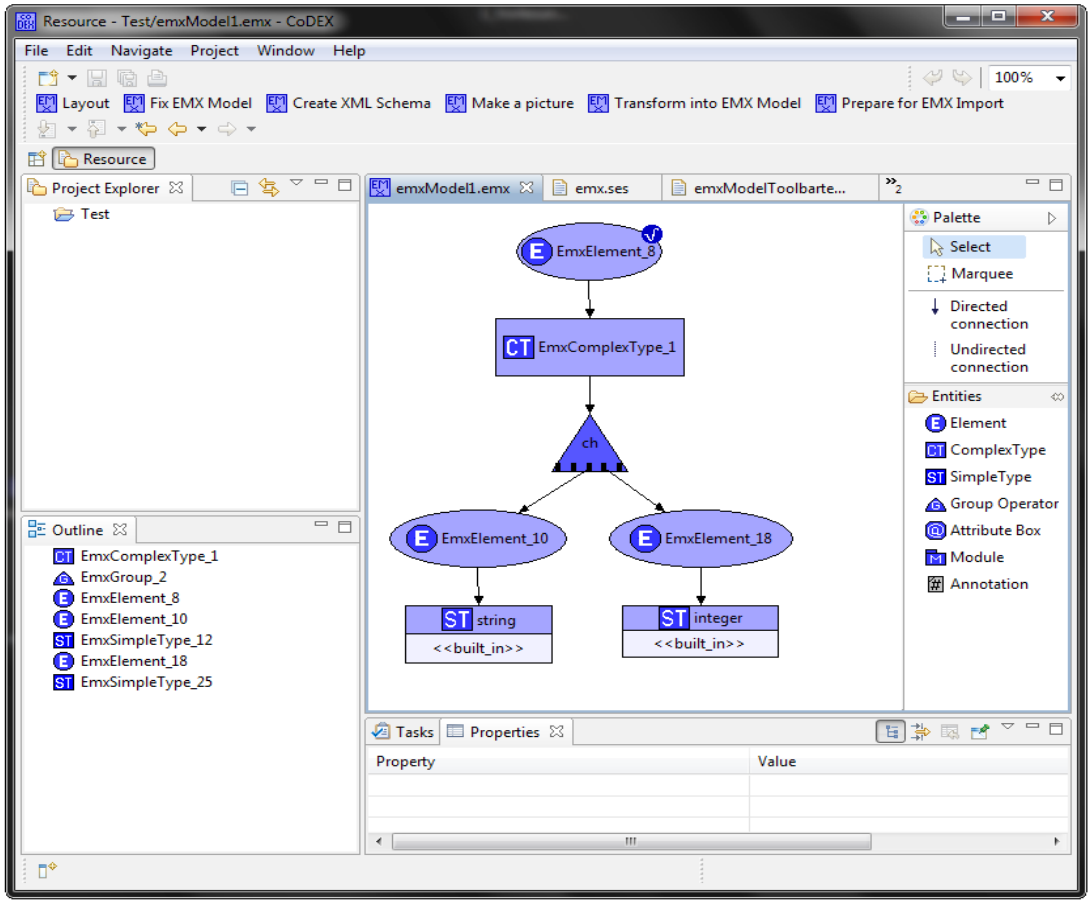
- Überprüfung der Kosten in Schleife eingebaut
- Werden benutzerdefinierte Kosten überschritten, wird Schleife verlassen und Evolution somit abgebrochen

```
public static boolean calcAllCostWithLimit(Collection<ChangeObject> evolutionSteps,
int limit){
  boolean isHigh = false;
  double sumOfAllSteps = 0;
  for(Iterator<ChangeObject> iterator = evolutionSteps.iterator();iterator.hasNext();)
  {
    sumOfAllSteps += calc.calculateCost(iterator.next());
    if(sumOfAllSteps>limit){
      isHigh = true;
      break;
     }
  }
  return isHigh;
}
```

- isHigh = Versionierung, !isHigh = Evolution durchführen

Umsetzung in CodeX

- Live-Präsentation



Quellen

- [ACP09] Antonio Cicchetti, Davide Di Ruscio und Alfonso Pierantonio: Managing Dependent Changes in Coupled Evolution. In: ICMT 2009, Seiten 35 - 51, 2009.
- [GG05] Giovanna Guerrini, Marco Mesiti, Daniele Rossi: Impact of XML Schema Evolution on Valid Documents. In: WIDM 05, 2005.
- [GG06] Giovanna Guerrini, Marco Mesiti, Daniele Rossi: XML Schema Evolution, 2006.
- [Har07] Hartung, Michael: XML Schema Evolution. Technischer Bericht, Universität Leipzig - Abteilung Datenbanken, 2007.
- [HS01] Hong Su, Diane Kramer, Li Chen Kajal Claypool Elke A. Rundensteiner: XEM: Managing the Evolution of XML Documents. In: Proceedings of the 11th International Workshop on Research Issues in Data Engineering, 2001.
- [Mal10] Malý, Jakub: XML Schema Evolution. Diplomarbeit, Univerzita Karlova v Praze, 2010.
- [MMM07] Mirella M. Moro, Susan Malaika, Lipyeow Lim: Preserving XML Queries during Schema Evolution. In: WWW 2007, 2007.
- [MN09] Martin Ne£aský, Irena Mlýnková: Five-Level Multi-Application Schema Evolution. In: Dateso 2009, Seiten 90 - 104, 2009.
- [MT04] Marvin Tan, Angela Goh: Keeping Pace with Evolving XML-Based Specications. In: EDBT 2004 Workshops, 2004.
- [PK05] Peter Kunzmann, Franz-Peter Burkard, Franz Wiedmann: dtv-Atlas Philosophie. Deutscher Taschenbuch Verlag, 2005.
- [Wil06] Will, Christian: Entwicklung und Implementierung einer Sprache zur Evolution von XML-Schemata. Diplomarbeit, Universität Rostock, 2006.
- [Zei01] Zeitz, Andre: Evolution von XML-Dokumenten. Studienarbeit, Universität Rostock, 2001.
- [W3C] David C. Fallside, Priscilla Walmsley: XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/, zuletzt aufgerufen am 22.10.2011.
- [TN11] Thomas Nösinger: persönliche Mitteilung vom 21.10.2011

Danke für Ihre Aufmerksamkeit.

Fragen?