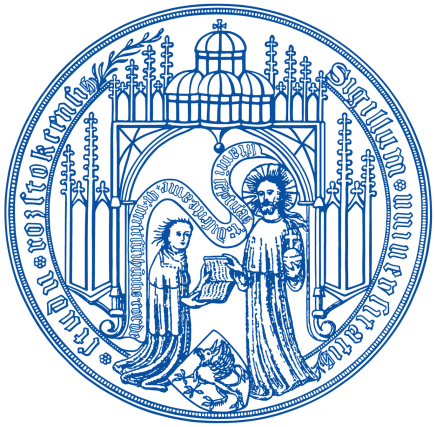
XML-Schema Evolution: Kategorisierung und Bewertung

Literaturarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik



vorgelegt von:	Hannes Grunert
Matrikelnummer:	7200076
geboren am:	30.08.1987 in Ribnitz-Damgarten
Gutachter:	Dr. -Ing habil. Meike Klettke
Betreuer:	Dipl. -Inf. Thomas Nösinger
Abgabedatum:	30.06.2011

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, den 1. Juli 2011

Inhaltsverzeichnis

CR-Klassifikation

- D.2.8 Metrics
- H.3.3 Information Search and Retrieval
- I.7.1 Document and Text Editing

Schlüsselwörter

XML, XML-Schema, Schema-Evolution, Klassifikation, Metrik

Keywords

XML, XML-Schema, schema-evolution, classification, metric

Kapitel 1

Einleitung

Für das Übertragen von Informationen zwischen Computern, insbesondere über das Internet, aber auch für den Austausch von Daten zwischen Programmen und Programmteilen innerhalb eines Computersystems, hat sich in den letzten Jahren XML als Austauschformat etabliert. Die Struktur dieser Dokumente kann durch eine Schemasprache, wie XML-Schema, beschrieben werden. Sofern sich ein XML-Dokument aus einem XML-Schema herleiten lässt, wird dieses Dokument als gültige Instanz bezüglich dieses Schemas bezeichnet.

XML-Schemata für eine Anwendung können sich über die Zeit verändern. Die Gründe hierfür können vielfältig sein. Zum einen können sich die Anforderungen an die Anwendung ändern, aber auch Fehler beim Entwurf und deren Korrigierung führen zu veränderten XML-Schmata. Neue Informationen werden abgespeichert, andere entfernt und andere nur leicht modifiziert. Durch diese Änderungsoperationen ist die Gültigkeit bestehender Dokumente nicht mehr gesichert. Die Instanzen müssen auf ihre Gültigkeit geprüft und gegebenfalls angepasst werden. Dieses Vorgehen wird als XML-Schema-Evolution bezeichnet.

Die Evolutionsschritte lassen sich dabei unter Betrachtung verschiedener Kriterien in Kategorien einteilen. In dieser Literaturarbeit wird ein Überblick über die Kategorisierungsansätze verschiedener universitärer Forschungsgruppen geschaffen und analysiert, inwieweit diese Kategorisierungen die Menge der Schemaevolutionsschritte abdecken.

Die weitere Arbeit ist wie folgt aufgebaut:

- Kapitel 2 führt kurz in die Thematik ein.
- Kapitel 3 widmet sich bestehenden Kategorisierungen und vergleicht sie miteinander.
- Kapitel 4 dient zur Zusammenfassung und gibt einen Ausblick auf die Bachelorarbeit.

Kapitel 2

Grundlagen der Schemaevolution

In Kapitel 1 wurde bereits kurz auf die XML-Schemaevoultion eingegangen. An dieser Stelle wird eine Ursache der Evolution eine XML-Schemas und der Auswirkung auf dazugehörige Instanzen anhand eines (fiktiven) Beispieles erläutert.

In Online-Rollenspielen können sich mehrere Spieler (player) zu einer Gruppe (raid) zusammenschließen um Aufgaben zu lösen, die sie alleine nicht schaffen würden. Dabei müssen sie meist computergesteuerte Monster (boss) besiegen. Das XML-Schema, welches in der Abbildung 2.1 dargestellt ist, verdeutlicht diesen Sachverhalt. Für eine Gruppe wird das aktuelle Datum und eine Liste der Begegnungen (encounter) mit den Monstern abgespeichert. Eine Begenung besteht aus dem Namen des Gegners und einer Liste von maximal 25 Spielern, die auch wirklich am Kampf teilgenommen haben. Zu jedem Spieler wird der Name, die Charakterrolle (class) und die Gesundheit (health) erfasst.

Eine konkrete Instanz, die bezüglich dieses Schemas gültig ist, ist in Abbildung 2.2 dargestellt. Das Abspeichern der Informationen, wann welcher Spieler an einem Kampf beteiligt war, können durch externe Programme, wie spezielle Content-Management-Systeme (CMS, siehe [Kna]), ausgelesen werden um statistische Informationen zum Spiel zu erstellen.

Angenommen es kommt zu einer Änderung in einem CMS, und der Name der Spieler wird nun nicht mehr über ein Attribut, sondern über den Text in einem neuen Element aus dem Schema ausgelesen. Die Änderung im XML-Schema würde nun so aussehen, dass das Attribut *name* im Element *player* entfernt und dort als neues Subelement eingefügt wird. Das aktualisierte XML-Schema ist in Abbildung 2.3 auszugsweise dargestellt.

Durch diesen Evolutionsschritt sind allerdings alle bisher existierenden Instanzen ungültig bezüglich des neuen Schemas, da sie nun ein nicht definiertes Attribut besitzen, dafür aber nicht über das neue Element verfügen. Folglich müssen nun auch alle Dokumente entsprechend der Schemaevolution angepasst werden. Das angepasste Dokument, auf welches das CMS wieder zugreifen kann, zeigt Abbildung 2.4.

Oben genannter Evolutionsschritt ist nur einer von vielen Änderungen, die an einem XML-Schema durchgeführt werden können. Diese Operation hat lediglich die Position einer Information verändert. Andere haben aber auch Einfluss darauf, ob sich die Informationen in einem XML-Dokument ändern können. So kann es auch zu Informationsverlust kommen, wenn beispielsweise nicht-optionale Elemente aus dem Schema gelöscht werden oder die Länge eines Strings gekürzt wird. Eine andere Operation kann aber auch dafür sorgen, dass neue Informationen hinzugefügt werden. Dies ist nicht in allen Fällen automatisch möglich und fordert, dass ein Benutzer die Daten hinzufügen muss. Im folgenden Kapitel werden verschiedene Ansätze vorgestellt, Änderungen an XML-Schemata zu unterteilen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="
    qualified">
 <xsd:element name="raid">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="datum"/>
    <xsd:element ref="encounterlist"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="datum" type="xsd:date"/>
 <xsd:element name="encounterlist">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="encounter"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="encounter">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="boss"/>
    <xsd:element ref="playerlist"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="playerlist">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="player" minOccur="2" maxOccur="25"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="player">
  <xsd:complexType>
  <xs:attribute name="name" type="xsd:string"/>
   <xsd:sequence>
    <xsd:element ref="'class"'/>
    <xsd:element ref="'health"'/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:simpleType name="class">
  <xsd:restriction base="xs:string">
   <xsd:enumeration value="Magier"/>
   <xsd:enumeration value="Priester"/>
   <xsd:enumeration value="Krieger"/>
  </xsd:restriction>
 </xsd:simpleType>
 <xsd:simpleType name="health">
  <xsd:restriction base="xs:decimal">
   <xsd:minInclusive value="0"/>
   <xsd:maxInclusive value="100"/>
  </xsd:restriction>
 </xsd:simpleType>
 <xsd:element name="boss" type="xsd:string"/>
<xsd:schema/>
```

Abbildung 2.1: Alte Version des XML-Schemas

```
<?xml version="1.0" encoding="UTF-8"?>
<raid>
 <datum>2011-05-25</datum>
 <encounterlist>
  <encounter>
   <boss>Kleiner Wabbelschleim</boss>
   <playerlist>
    <player name="Datenbanker">
     <class>Magier</class>
     <health>100</health>
    </player>
    <player name="Softi">
     <class>Priester</class>
     <health>42</health>
    </player>
   </playerlist>
  </encounter>
  <encounter>
   ...
  </encounter>
 </encounterlist>
</raid>
```

Abbildung 2.2: XML-Dokument nach der alten XML-Schema-Version

```
<xsd:element name="player">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="name"/>
   <xsd:element ref="class"/>
   <xsd:element ref="health"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Abbildung 2.3: Neue Version des XML-Schemas (Ausschnitt)

```
<player>
<name>Datenbanker</name>
  <class>Magier</class>
  <health>100</health>
</player>
```

Abbildung 2.4: XML-Dokument nach der neuen XML-Schema-Version (Ausschnitt)

Kapitel 3

Kategorisierungen von Evolutionsschritten

Der Ursprung des Kategorisierungsbegriffes reicht bis in die Antike zurück. In seiner Schrift „Die Kategorien“ unterteilt der Philosoph Aristoteles, der als Begründer der Kategorienlehre gilt, die Eigenschaften eines Begriffes in zehn Kategorien: „Jedes ohne Verbindung gesprochene Wort bezeichnet entweder eine Substanz oder eine Qualität oder eine Quantität oder eine Relation oder ein Wo oder ein Wann oder eine Lage oder ein Haben oder oder ein Wirken oder ein Leiden“ (vgl. [PK05]).

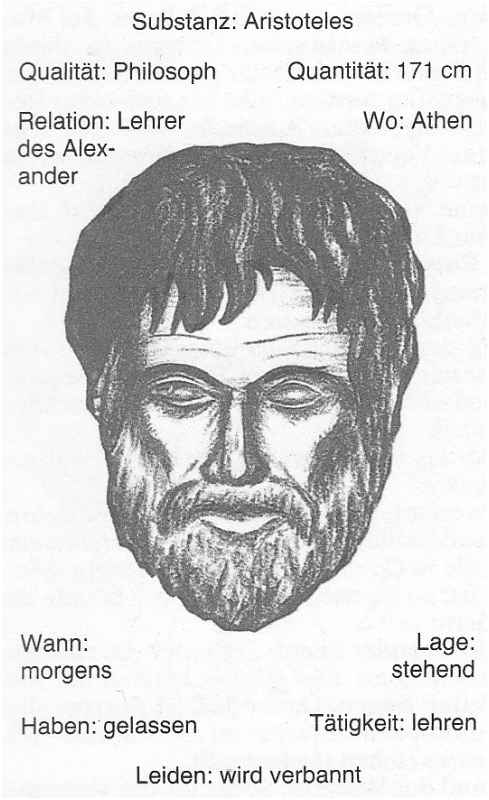


Abbildung 3.1: Kategorien nach Aristoteles, entnommen aus [PK05]

Diese Literaturarbeit dient als Grundlage für die Entwicklung einer Metrik, mit der die Kosten eines einzelnen Schemaevolutionsschrittes abgeschätzt werden könne. Es wird zunächst eine Kategorisierung der Änderungen benötigt, die in dieser Arbeit erarbeitet wird. Die Unterteilung nach Aristoteles ist dafür freilich viel zu allgemein; für die Abschätzung werden andere Kriterien benötigt. Allerdings lassen sich auch Parallelen zu Aristoteles ziehen. Folgende Kriterien sind für die Betrachtung der Evolution möglich:

1. die Informationskapazität - Quantität (gehen Informationen verloren oder kommen neue hinzu?),
2. die betroffene Komponente - Wo wird etwas geändert (einfacher Typ, komplexer Typ, Attributtyp)?,
3. die Art der Änderung - Wirken (Findet eine Lösch-, Hinzufügen- oder Update-Operation statt?),
4. und die tatsächliche Auswirkung auf Instanzen - Leiden (sind die XML-Dokumente weiterhin gültig?).

Kapazitätsänderungen lassen sich nach [Har07] in informationserhaltenden, informationsreduzierenden, informationserweiternden und informationsverändernden Operationen unterteilen. Diese Klassifikation gibt unter anderem Aufschluss darüber, ob es während der Evolution eines XML-Schemas zu Informationsverlust oder Kompabilitätsproblemen, beispielsweise durch neue, nicht-optionale Elemente kommt.

Die tatsächliche Auswirkung auf Instanzen ist eng mit der Informationskapazität verbunden. Eine gesonderte Betrachtung ist vor allem bei informationsverändernden Operationen von Bedeutung, da sich die Auswirkungen einer Schemaänderung erst erkennen lässt, wenn das XML-Schema in Verbindung mit den Instanzen betrachtet wird. Anhand der Art der Änderung lassen sich Rückschlüsse auf die Kapazität ziehen. So kann bereits ohne nähere Betrachtung der Operationen eine Vorauswahl für eine Klassifizierung der Informationskapazität erfolgen.

Eine Betrachtung nach der betroffenen Komponente spielt für die Entwicklung einer Metrik nur eine untergeordnete Rolle. Allerdings lassen sich nicht alle Schemaevolutionsschritte auf jede Komponente anwenden. Dadurch ist ebenfalls eine Vorauswahl für die Betrachtung der Informationskapazität möglich.

Im nächsten Abschnitt wird eine Übersicht über die Kategorisierungen von anderen Forschungsgruppen skizziert. Außerdem wird untersucht, ob die angebenen Evolutionsschritte auch alle möglichen Änderungen an XML-Schemata umfassen und inwiefern sich die vorgeschlagene Kategorisierung jeweils für die Kostenabschätzung eignet.

3.1 Kategorisierungen von anderen Forschungsgruppen

3.1.1 XEM

In dem Artikel „XEM: Managing the Evolution of XML Documents“ von Su, Kramer et al. [HS01] wird der Prototyp einer XML-Evolutionssoftware vorgestellt, dem XML-Evolution Manager (XEM). Der Prototyp bestimmt die Änderungen an einer DTD (Document Type Definition) und nimmt anschließend die Anpassung der Dokumente vor.

Die Autoren unterteilen die Schema-Änderungen in einfache und komplexe, wobei letztere die Hintereinanderausführung von einfachen Schema-Änderungen sind. Zu den Evolutionsschritten gehören das Hinzufügen, Löschen und Ändern von Elementen und Attributen, sowie das Umgruppieren von Elementen [1].

In Abbildung 3.2 sind neben der Kategorisierung der Evolutionsschritte auch die Operationen auf den XML-Instanzen dargestellt. In der linken Tabellenspalte steht der Operator mit den zu übergebenen Parametern, auf der rechten die Beschreibung des Evolutionschrittes bzw. der Änderung an den Dokumenten. So steht z. B. die Operation *destroyDTDAtt(u, E)* für das Löschen des Attributes u im Element E.

Für jede Operation wird die Semantik der damit verbundenen Änderung analysiert, damit Vorbedingungen angegeben werden können, die die strukturelle Konsistenz sicherstellt. Die Auflistung der

[1] Das Umgruppieren von Elementen hat dabei keinen Einfluss auf Instanzen

DTD Operation	Description
createDTDEl(*u*)	Create element with name *u*
destroyDTDEl(*u*)	Destroy element with name *u*
renameDTDEl(*u, u'*)	Rename element from name *u* to *u'*
insertDTDEl(*E, pos, P, q, d*)	Add element *E* at position *pos* to parent *P* with quantifier *q* and default value *d*
removeDTDEl(*E, pos*)	Remove sub-element at position *pos* in parent *E*
changeQuant(*E, pos, q, d*)	Change quantifier of content particle at position *pos* in parent *E* to quantifier *q* with default value *d*
convertToGroup(*E, start, end*)	Group sub-elements from position *start* to position *end* in parent *E* into a list group
flattenGroup(*E, pos*)	Flatten group at position *pos* in element *E* to a list of sub-elements
changeGroupQuant(*E, pos, q*)	Change quantifier of group at position *pos* in element *E* to *q*
addDTDAtt(*u, E, t, d, v*)	Add attribute with name *u* to element *E* with type *t*, default type *d*, and default value *v*
destroyDTDAtt(*u, E*)	Destroy attribute with name *u* from element *E*
changeAttDefType(*u, E, t, v*)	Change element *E*'s attribute *u*'s type to *t*, with default value *v*
changeAttDefValue(*u, E, v*)	Change element *E*'s attribute *u*'s default value to *v*
changeAttFixedValue(*u, E, v*)	Change element *E*'s attribute *u*'s fixed value to *v*

XML Data Operation	Description
addDataAtt(*a, v, pos*)	Add attribute with name *a* with value *v* to position *pos*
destroyDataAtt(*a*)	Destroy attribute *a*
changeDataAtt(*a, v, e*)	Change attribute *a*'s value in element *e* to *v*
addDataEl(*e, pos*)	Add element *e* at position *pos*
destroyDataEl(*e*)	Destroy element *e p*
changeDataEl(*e, v*)	Change element *e*'s value to *v*

Abbildung 3.2: Einfache Änderungen an einer DTD in XEM [HS01]

Evolutionsschritte wird als völlständig bewiesen, d.h. aus den angegebenen Operationen lassen sich alle möglichen Änderungen an einer DTD darstellen. Außerdem ist die Menge der Operationen minimal und wohlgeformt.

Die Kategorisierung in XEM ist ungeeignet um eine Metrik zu entwickeln, da die Änderungen sich auf eine DTD und nicht auf eine XML-Schema-Datei beziehen. Des Weiteren werden keine Aussagen zur Informationskapazität getroffen, was für die spätere Aufstellung einer Metrik ein wichtiges Kriterium darstellt.

3.1.2 Evolutionskategorien nach Andre Zeitz

Andre Zeitz erstellte in seiner Studienarbeit [Zei01] ebenfalls eine Kategorisierung für Änderungen an einer DTD. Dabei ordnet er die Evolutionsschritte zunächst nach der veränderten Komponente, anschließend nach der Art der Änderung. Die einzelnen Operationen werden weiterhin näher bezüglich ihrer Informationskapazität und der Auswirkungen auf existierende Instanzen untersucht. Abbildung 3.3 gibt einen Überblick über die vorgeschlagene Kategorisierung.

Diese Klassifizierung betrachtet sowohl die Art der Änderung, die veränderte Komponente, als auch die Auswirkung auf die Informationskapazität und die Instanzen. Damit sind alle Kriterien erfüllt. Allerdings

- Umformung auf Attributebene
  - Änderung des Attributnamens
  - Änderung des Attributwertes
  - Änderung der Attributreihenfolge
  - Hinzufügen neuer Attribute
  - Entfernen vorhandener Attribute
  - Änderung der Default Declaration
  - Änderung des Attributtyps
- Umformung auf Elementebene
  - Änderung des Elementnamens
  - Änderung des Quantors
  - Änderung der Elementreihenfolge
  - Hinzufügen eines Elementes
  - Löschen eines Elementes
  - Änderung zwischen Alternative und Sequenz
  - Änderung des Content Type
- Umformung auf Entitätsebene
  - Allgemeine Entitäten
  - Parameter-Entitäten
  - Nicht analysierte Entitäten

Abbildung 3.3: Kategorisierung nach Andre Zeitz [Zei01]

behandelt die Studienarbeit ausschließlich die Schemaevolution anhand einer DTD, nicht an einem XML-Schema. Daher ist an dieser Kategorisierung nur der Aufbau, aber nicht die Zuordnung der Operationen zu den Kategorisierungskriterien interessant.

3.1.3 Kategorisierung nach Tan und Goh

Marvin Tan and Angela Goh stellen in ihrem Artikel „Keeping Pace with Evolving XML-Based Specifications“ [MT04] ein Framework für die XML-Schema-Evolution vor. Die Änderungen, die zwischen zwei Versionen eines XML-Schemas auftreten können, werden dabei in die drei Arten unterteilt, die in Abbildung 3.4 dargestellt sind. Während die strukturellen Änderungen(structural) das Löschen und Hinzufügen von Elementen und Atrributen beinhaltet, umfassen die anderen Kategorien das Verschieben und Verändern(migratory), sowie das Umbenennen(sedentary) von Attributen und Elementen. Neben diesen drei Kategorien werden außerdem noch semantische Änderungen betrachtet, die zwar keinen direkten Einfluss auf Instanzen haben, aber als Nebeneffekt von den anderen Änderungsoperationen auftreten können. Grund hierfür ist die sich ändernde Interpretation von Elementen in der Schemabeschreibung. Tan und Goh geben als Beispiel für eine semantische Änderung die Erweiterung des Begriffes „Entität“ an. Während dieser in der alten Version eines XML-Schemas nur Betriebe und Unternehmen bezeichnet, könnte er in einer aktuelleren Version auch auf Personen ausgeweitet werden.

Diese Kategorisierung ist aus zwei Gründen ungeeignet. Zum einen werden nur die Art der Änderung

Migratory Changes	Structural Changes	Sedentary Changes
• Morphing of an element to an attribute. • Migration of a sub-element from one element to another. • Modification of an attribute or element.	• Addition of new elements, sub-elements and attributes. • Removal of elements, sub-elements and attributes.	• Renaming of elements. • Renaming of attributes • Change of simple data type.

Abbildung 3.4: Kategorisierung nach Tan und Goh (aus [MT04])

und das zu betroffene Element betrachtet, während den anderen Kriterien, dem Informationsgehalt und der Auswirkung auf Instanzen, keine Bedeutung beigemessen wird. Außerdem sind die angegebenen Evolutionsschritte unvollständig. So werden beispielsweise keine Angaben zu den Konzepten Beziehungen und Identitätsdefinitionen gemacht.

3.1.4 Kategorisierung nach Guerrini, Mesiti und Rossi

Giovanna Guerrini, Marco Mesiti und Daniele Rossi [GG05, GG06] untersuchen ebenfalls den Einfluss von XML-Schema-Änderung auf Dokumente. Die von ihnen vorgeschlagene Kategorisierung ist in Form einer Tabelle (siehe Abbildung 3.5) angeordnet, die die atomaren Evolutionsschritte enthält. In den Spalten sind die Änderungen nach Einfüge-, Modifikations- und Löschoperationen unterteilt, in den Zeilen nach der beteiligten Komponente (Simple Type, Complex Type, Element). Evolutionsschritte, die keinen Einfluss auf Instanzen haben, werden mit einem Stern gekennzeichnet. Komplexere Änderungen werden, wie bereits in XEM, als Folge von atomaren Schritten aufgefasst.

	Insertion	Modification	Deletion
Simple Type	*insert_glob_simple_type** *insert_new_member_type**	*change_restriction* *change_base_type* *rename_type** *change_member_type* *global_to_local** *local_to_global**	*remove_type** *remove_member_type**
Complex Type	*insert_glob_complex_type** *insert_local_elem* *insert_ref_elem* *insert_operator*	*rename_local_elem* *rename_global_type** *change_type_local_elem* *change_cardinality* *change_operator* *global_to_local** *local_to_global**	*remove_element* *remove_operator* *remove_substructure* *remove_type**
Element	*insert_glob_elem*	*rename_glob_elem** *change_type_glob_elem* *ref_to_local** *local_to_ref**	*remove_glob_elem**

Abbildung 3.5: Kategorisierung nach Guerrini, Mesiti und Rossi [GG05]

Die Unterteilung Schema-Änderungen, welche die Gültigkeit der XML-Dokumente bezüglich des neuen Schemas betrachten, ist ein guter Ansatz für die Entwicklung einer Metrik. Allerdings ist diese Kategorisierung weder vollständig, da keine weiterführenden Konzepte und Eigenschaften wie Schlüssel oder Namensräume in der Auflistung der Operationen vorkommen, noch eine Betrachtung der Informationskapazität erfolgt.

3.1.5 Universität Leipzig

Michael Hartung [Har07] unterteilt Schemänderungen nach der Art der Änderung und nach der Auswirkung auf die Informationskapazität. Evolutionsschritte, die das Hinzufügen von Komponenten erwir-

ken, ordnet er stets eine informationserweiternde Wirkung zu, während Änderungen, die Komponenten löschen, eine informationsreduzierende Auswirkung haben. Schemaänderungen, die ein Element oder Attribut modifizieren, können entweder eine informationsreduzierende, -erweiternde, -erhaltende oder -verändernde Wirkung haben. Diese Zuordnung ist in Abbildung 3.6 zu sehen.

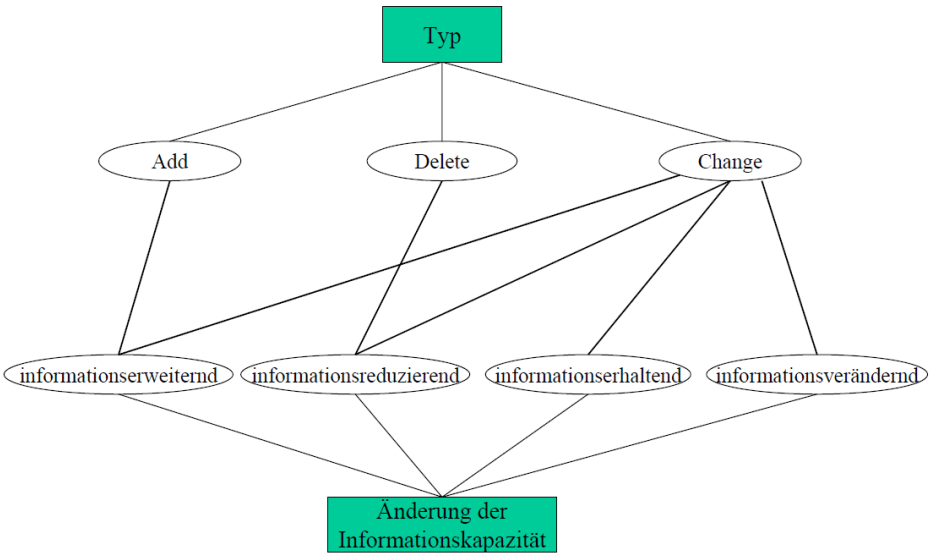


Abbildung 3.6: Kategorisierung nach Michael Hartung (aus [Har07])

Zu den informationsreduzierenden und -erweiternden Evolutionsschritten zählt Hartung auch solche Operationen, die sich auf optionale Komponenten beziehen und die bezüglich eines XML-Dokumentes auch eine informationserhaltenden Effekt hätten. Die tatsächliche Auswirkungen auf XML-Dokumente wird von ihm in seiner Kategorisierung nicht weiter betrachtet.

3.1.6 Kategorisierung nach Moro, Malaika und Lim

Mirella M. Moro, Susan Malaika, Lipyeow Lim beschäftigen sich in [MMM07] mit der Auswirkung von XML-Schema-Änderungen auf Anfragen an XML-Dokumente. Obwohl ihr Fokus auf die Anfragen liegt, führen sie auch eine Kategorisierung für Schemaänderungen ein, die in Abbildung 3.7 dargestellt ist. Dabei unterscheiden sie zunächst zwischen einfachen und komplexen Änderungen. Anschließend unterteilen sie die Änderungen nach der Art und geben zu jeder einzelnen an, ob diese eine Auswirkung auf Instanzen hat oder nicht. Eine Angabe konkreter Evolutionsschritte erfolgt nicht; die Angaben sind allgemeiner gehalten (beispielsweise „Hinzufügen neuer Konstrukte“ unter Extension). Zu den einfachen Änderungen zählen Operationen zum Hinzufügen, Löschen und Ändern von Elementen und Attributen. Komplexe Änderungen sind nicht, wie in den Kategorisierungen zuvor, die Hintereinanderausführung von einfachen Evolutionsschritten, sondern sie umfassen Konzepte wie Namensräume und das Ändern von Kardinalitäten.

Da sich Moro, Malaika und Lim in ihrer Arbeit auf den Einfluss der Schemaänderungen auf Anfragen fokusieren, gehen sie nicht weiter auf die Auswirkung auf Instanzen ein. Ebenso werden auch keine Aussagen zur Informationskapazität getroffen, wodurch diese Kategorisierung auch nicht für die Entwicklung einer Metrik eignet.

- Basic Changes
  1. Refinement
  2. Removal
  3. Extension
  4. Reinterpretation
  5. Redefinition
- Complex Changes
  1. Element Composition and Decomposition
  2. Renaming
  3. Optionality
  4. Renumbering
  5. Retyping
  6. Namespaces
  7. Default values
  8. Reordering

Abbildung 3.7: Kategorisierung nach Moro et al. [MMM07]

3.1.7 Kategorisierung auf Metamodellen

Antonio Cicchetti, Davide Di Ruscio, and Alfonso Pierantonio untersuchen in ihrer Arbeit „Managing Dependent Changes in Coupled Evolution“ [ACP09] die Evolution auf Metamodellen und wie die dazugehörigen Modelle (wie ein XML-Schema) davon betroffen sind. Sie geben dabei zwei Kriterien an, nach denen Änderungen unterschieden werden können. Zum einen kann untersucht werden, ob der Evolutionsschritt

- keine Auswirkung auf Modelle
- Auswirkung auf Modelle, die automatisch behoben werden kann,
- Auswirkung auf Modelle, die nicht automatisch behoben werden kann,

hat. Das zweite Kriterium betrachtet die Art der Änderung, die, wie in den anderen Kategorisierungen, in Operationen unterteilt werden, die Komponenten hinzufügen, löschen oder modifizieren. Da sich sich dies Änderungen auf Metamodelle beziehen, wird an dieser Stelle auf eine Auflistung der Operationen verzichtet. Aus diesem Grund ist diese Kategorisierung ebenfalls ungeeignet, um eine Metrik für XML-Schema-Änderungen zu entwickeln. Die Änderungen zunächst nach ihrer tatsächlichen Auswirkung zu kategorisieren, ist jedoch ein guter Ansatz, da sich so das tatsächliche Ausmaß zunächst gut abschätzen lässt.

3.1.8 Kategorisierungen der Universität Prag

Martin Nečaský und Irena Mlýnková schlagen für die Bewältigung der XML-Schema-Evolution eine Fünf-Schichten-Architektur [MN09] vor. Ziel dieser Architektur ist es, nicht nur die Änderungen an einem XML-Schema zu betrachten, sondern auch um Auswirkungen auf andere XML-Schemata und Anfragen festzustellen.

Auf der untersten Ebene (extensional level) befinden sich die XML-Dokumente, während sich die dazugehörigen Schemata in der mittleren Ebene (logical level) befinden. Diese beiden Ebenen sind miteinander verbunden, sodass eine Änderung am XML-Schema direkte Auswirkungen auf die Instanzen hat.

Für die einzelnen Ebenen existieren verschiedene Änderungsoperationen. Abbildung 3.8 gibt eine Übersicht über die Änderungen auf der Schema-Ebene. Neben dem bekannten Hinzufügen, Löschen und Ändern gibt es zusätzlich noch eine Operation für das Ändern der Kardinalität. In den anderen Kategorisierungen war diese in der Menge der Änderungsoperationen enthalten.

- Adding: Hinzufügen eines neuen Elementes, Attributes, choice- oder sequence-Operators
- Removal: Löschen eines Elementes, Attributes, choice- oder sequence-Operators
- Extension: Hinzufügen eines simple oder complex types
- Renumbering: Ändern der Kardinalität eines Elementes
- Retyping: Ändern eines simple types

Abbildung 3.8: Kategorisierung nach Martin Nečaský und Irena Mlýnková (aus [MN09])

Wie bereits in den meisten Kategorisierungen zuvor, ist diese Unterteilung ebenfalls nicht für eine Kostenabschätzung geeignet, da sie lediglich eine Unterteilung nach der Art der Änderung vornimmt. Außerdem erfolgt auch keine Angabe zu weiterführenden Konzepten (zum Beispiel Namespaces und Beziehungen).

Jakub Malý verfolgt in seiner Diplomarbeit [Mal10] den Ansatz der Fünf-Schichten-Architektur weiter. Dabei schlägt er auch eine ähnliche Kategorisierung vor, die die Änderungen in die vier Gruppen Hinzufügen, Löschen, Verschieben und Ändern unterteilt:

- Addition - Hinzufügen eines neuen Konstruktes
- Removal - Löschen eines Konstruktes
- Migratory - Verschieben eines Konstruktes
- Sedentary - Verändern eines Konstruktes

Der Term *Konstrukt* bezieht sich dabei sowohl auf Komponenten im XML-Schema als auch auf Operationen in denanderen Ebenen. Daher handelt es sich hier um die Kategorisierung allgemeiner Operationen.

Im weiteren Verlauf seiner Arbeit stellt er zudem noch eine Übersicht über Operationen vor, die keine Auswirkungen auf Instanzen von XML-Schemata haben. Dazu gehören

1. Änderungen die nur beim Erstellen neuer Elemente oder Attribute im XML-Dokument auftreten
2. Änderungen, die zusätzliche Optionen einführen
3. Änderungen, die Kardinalitätsangaben erweitern
4. Änderungen, die optionale Elemente einfügen

Zu der ersten Gruppe gehört das Ändern eines Default-Wertes, da dieser nicht nachträglich im XML-Dokument geändert wird. In die zweite Kategorie fallen Operationen wie das Hinzufügen von Elementen zu einem choice-Operator oder das Generalisieren eines Attributtypes. Das Ändern einer Kardinalität hat keine Auswirkung auf Instanzen, falls der minOccur-Wert verringert oder der maxOccur-Wert vergrößert wird. Zu der letzten Gruppe gehören unter anderem das Hinzufügen von optionalen Attributen.

Letztgenannte Kategorisierung ist äußerst hilfreich für die Entwicklung einer Metrik, da hier Angaben zu konkreten Operationen gemacht werden, die die Gültigkeit eines XML-Dokumentes nicht beeinflussen. Leider erfolgt keine Angabe der Evolutionsschritte, wodurch die Kategorisierung nicht vollständig ist. Zudem wird keine Angabe zur Informationskapazität gemacht, wodurch sich nicht ermitteln lässt, wie sich der Informationsgehalt durch einen Evolutionsschritt ändert.

3.1.9 Universität Rostock

Die Kategorisierung nach Christian Will [Wil06] ist die umfangreichste der hier vorgestellten. In seiner Diplomarbeit betrachtet er neben den Änderungen an einfach und komplexen Typen sowie Attributen auch Notationen, Anmerkungen, Beziehungen und Identitätsdefinitionen. Zunächst erfolgt eine Klassifikation nach dem Typ der Änderung, anschließend nach der Art der Änderung. Eine komplette Übersicht wird in Abbildung 3.9 dargestellt.

Will stuft anschließend die Evolutionsschritte auch nach Auswirkungen auf die Instanzen ein. Dabei unterscheidet er, ob die Änderungen niemals, möglicherweise oder immer Auswirkungen auf die Dokumente haben. Außerdem hebt Will hervor, dass einige Änderungen nicht automatisch umgesetzt werden können und somit ein manueller Eingriff seitens des Benutzers erfolgen muss. Für dieses Kriterium werden allerdings nur Beispiele angegeben, eine explizite Zuordnung zwischen Operation und Auswirkung findet nicht statt. Angaben zur Informationskapazität fehlen ebenfalls, daher ist diese Kategorisierung auch für die Aufstellung einer Metrik ungeeignet.

3.2 Abschließende Analyse

Im vorherigen Abschnitten wurden Kategorisierungen von Schemaevolutionsschritten verschiedener Forschungsgruppen vorgestellt und nach den zu Beginn des Kapitels genannten Kriterien untersucht. Ein Überblick zu den Ergebnissen findet sich in Tabelle 3.1. Zusätzlich wird aufgeführt, auf welcher Schemasprache die Kategorisierung aufsetzt, und ob die angegebenen Operationen den vollen Funktionsumfang der Schemasprache widerspiegeln.

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	+	+	+	-	(+)

Tabelle 3.1: Überischt über erfüllte Kriterien

Legende:

+	Kriterium wurde betrachtet
(+)	Kriterium wurde teilweise betrachtet
-	Kriterium wurde nicht beachtet

Leider kann keine der untersuchten Kategorisierungen alle Anforderungen erfüllen, die für die Entwicklung einer Metrik notwendig sind. Aus diesem Grund ist es notwendig, eine eigene Kategorisierung zu entwickeln, die für die Kostenabschätzung von Schemaevolutionsschritten geeignet ist.

1. Ändern des Schemas
   - Hinzufügen und Entfernen von Typdefinitionen, Attributdeklarationen und Elementdeklarationen
   - Hinzufügen und Entfernen von Attributgruppen- und Elementgruppen-Definitionen
   - Hinzufügen und Entfernen von Notationen und Anmerkungen
2. Ändern des Inhaltes einer Typdefinition
   (a) Ändern der Attributmenge
      - Hinzufügen, Löschen und Umbenennen eines neuen Attributes
      - Ändern einer Attributeinheit
        – Ändern der Attributverwendung, der Wertebereichsbeschränkung und der Typdefinition
      - Fremde Attribute
        – Hinzufügen, Entfernen und Ändern einer Attribut-Wildcard

   (b) Ändern der Elementgruppe (Komplexer Inhalt)
      - Ändern des Typs (Sequenz, Alternative, Menge)
      - Ändern des minimalen und maximalen Vorkommens
      - Hinzufügen, Entfernen und Umbenennen eines neuen Elementes
      - Hinzufügen und Entfernen einer Elementgruppe
      - Fremde Elemente
        – Hinzufügen und Entfernen einer Wildcard

   (c) Ändern des Inhalts der Elemente
      - Ändern der zugewiesenen Typdefinition
      - Ändern des minimalen und maximalen Vorkommens
      - Ändern der Nullwertfähigkeit

   (d) Ändern des einfachen Datentyps (Einfacher Inhalt)
      - Ändern des Typs (Atomar, Liste, Vereinigung)
      - Grundlegende und einschränkende Fassetten

   (e) Umbenennung der Typdefinition

   (f) Ändern des Ziel-Namensraumes

   (g) Festlegen des Inhaltstyps und der Eigenschaft Abgeschlossenheit
3. Ändern der Typhierarchie
   - Einfache Typdefinition zu einer Vereinigung hinzufügen oder aus einer Vereinigung entfernen
   - Komplexe Typdefinition in Hierarchie als Erweiterung oder Einschränkung einfügen
   - Entfernen einer Typdefinition aus der Hierarchie
4. Ändern von Beziehungen
   - Hinzufügen und Entfernen eines Primärschlüssels (key)
   - Hinzufügen und Entfernen eines optionalen Primärschlüssels (unique)
   - Hinzufügen und Entfernen eines Fremdschlüssels (keyref)
5. Ändern von Identitätsdefinitionen
   - Hinzufügen und Entfernen einer Identität (ID)
   - Hinzufügen und Entfernen einer Identitätsreferenz (IDREF)

Abbildung 3.9: Kategorisierung nach Christian Will [Wil06]

Kapitel 4

Ausblick

Diese Literaturarbeit dient zur Vorbereitung meine Bachelorarbeit an der Universität Rostock. Während die Literaturarbeit einen Überblick über die Kategorisierungsansätze verschiedener Forschungsgruppen zur XML-Schemaevolution verschafft, ist es das Ziel der Bachelorarbeit eine Metrik zu entwickeln, um die Auswirkungen von XML-Schemaevolutionsschritten auf bestehende Instanzen zu untersuchen.

Die meisten der hier vorgestellten Kategorisierungen beschränken sich auf die Unterteilung der Evolutionsschritte nach dem Typ der zu ändernen Komponente und der Art der Änderung, während die Auswirkung auf Instanzen und die Veränderung der Informationskapazität meist nur kurz erwähnt oder gar nicht behandelt wird. Dabei sind die beiden letztgenannten Kriterien die entscheidenen für die Entwicklung einer Metrik, da sie das tatsächliche Ausmaß auf XML-Instanzen betrachten.

Als Grundlage für die zu betrachtende Operationsmenge dient die Arbeit von Christian Will [Wil06], da sie eine vollständige Übersicht über auftretende Schemaevolutionsschritte bietet. Die zu entwickelnde Klassifikation wird die Evolutionsschritte zunächst nach deren Auswirkungen auf bestehende XML-Dokumente unterteilen. Dabei können folgende drei Fälle, die sich an der Arbeit von Jakub Malý [Mal10] orientieren, auftreten:

1. Schemaänderungen ohne Auswirkungen auf Instanzen
2. Schemaänderungen, die möglicherweise Auswirkungen auf Instanzen haben
3. Schemaänderungen, die immer Auswirkungen auf Instanzen haben

Evolutionsschritte, die die Gültigkeit von XML-Dokumenten nicht beeinflussen und somit keinen Änderungsaufwand verursachen, werden nicht weiter unterteilt. Alle weiteren Änderungen werden anschließend nach ihrem Einfluss auf die Informationskapazität hin untersucht. Interessant sind hierbei die Änderungen, die möglichweise Einfluss auf Instanzen haben und eine informationsverändernde oder informationserhaltende Wirkung haben. Diese Änderungen können sich, je nachdem wie die konkreten Instanzen aussehen, nur auf einen Teil der zum Schema gehörenden Dokumente auswirken. Dies kann beispielsweise auftreten, wenn der maxOccur-Wert eines Elementes verringert wird. In diesem Fall sind nur die Elemente der Dokumente betroffen, die den neuen maxOccur-Wert überschreiten.

Der weitere Verlauf der Bachelorarbeit gestaltet sich wie folgt: Zunächst werden die Evolutionsschritte nach ihrer Auswirkung auf die Informationskapazität untersucht. Anschließend erfolgt die eigentliche Kategorisierung und die Entwicklung der daraus abzuleitenden Metrik. Die Implementation eines Prototyps zur Kostenabschätzung wird die Arbeit abrunden.

Abbildungsverzeichnis

Literaturverzeichnis

[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.

[Kna] Knaak, Stefan: *EQDKPplus.* http://www.eqdkp-plus.com/news.php?[de], zuletzt aufgerufen am 28.06.2011.

[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 Nečaský, 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 Specifications.* 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.