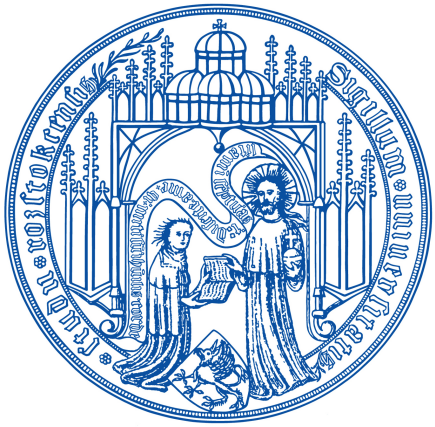
XML-Schema Evolution: Kategorisierung und Bewertung

Bachelorarbeit

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
	Prof. Dr.-Ing. habil. Gero Mühl
Betreuer:	Dipl.-Inf. Thomas Nösinger
Abgabedatum:	23.09.2011

Selbstständigkeitserklärung

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

Rostock, den 21. Oktober 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.

Ziel dieser Arbeit ist die Entwicklung einer Metrik, mittels der man die Auswirkungen von XML-Schema-Änderungen auf Instanzen abschätzen kann. Dazu werden zunächst die Evolutionsschritte in Kategorien eingeordnet, mit denen man die Kosten prognostizieren kann. Es wird zunächst ein Überblick über die Kategorisierungsansätze verschiedener universitärer Forschungsgruppen geschaffen und analysiert, inwieweit diese Kategorisierungen die Menge der Schemaevolutionsschritte abdecken und ob der Ansatz für die Aufstellung einer Metrik geeignet ist.

Danach werden die einzelnen Schemaevolutionsschritte hinsichtlich der Kriterien Kapazität, Informationsgehalt und den Einfluss auf die XML-Instanzen untersucht und in die gewählte Kategoriesierung eingeordnet. Abschließend wird eine Metrik ausgearbeitet, die auf der Klassifizierung aufsetzt und die Kosten der XML-Evolutionsschritte berechnet. Zusätzlich wird ein Prototyp implementiert, der die geschaffene Metrik umsetzt.

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.
- In Kapitel 4 werden die Evolutionsschritte nach ihren Eigenschaften hin untersucht.
- Kapitel 5 stellt den entwicklelten Prototypen vor.
- Kapitel 6 dient zur Zusammenfassung.

Kapitel 2

Grundlagen der Schemaevolution

In Kapitel 1 wurde bereits kurz auf die XML-Schemaevolution 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 Begegnung 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 5.3 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" minOccurs="1" maxOccurs="unbounded"/>
   </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" minOccurs="2" maxOccurs="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>

<xsd:element name="name" type="xsd:string"/>
```

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]).

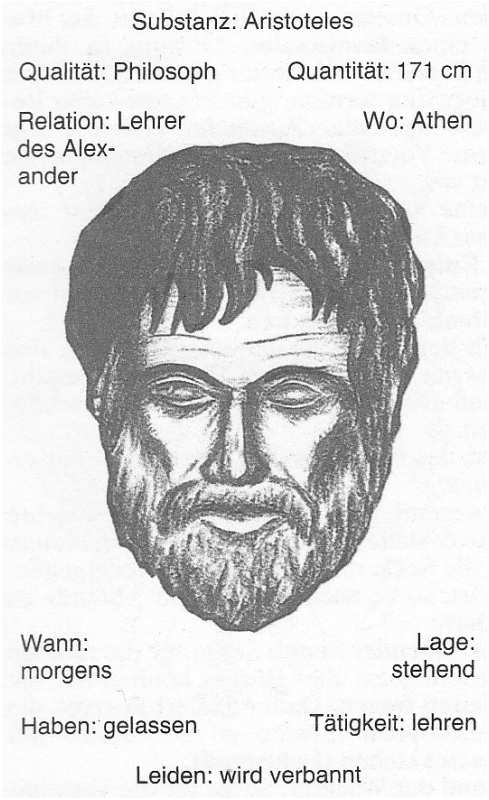


Abbildung 3.1: Kategorien nach Aristoteles, entnommen aus [PK05]

Diese Bachelorarbeit 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 (covertToGroup, flattenGroup) 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 vollständig 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.

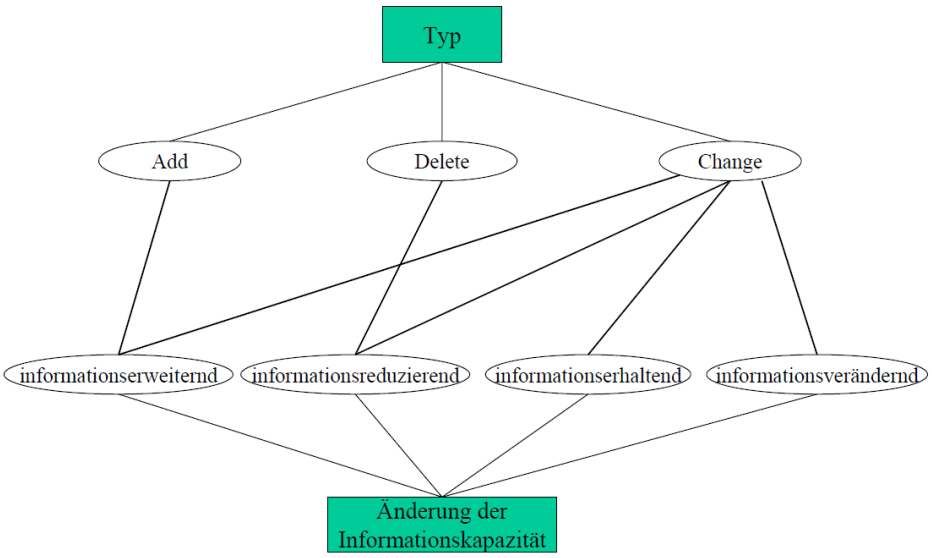


Abbildung 3.6: Kategorisierung nach Michael Hartung (aus [Har07])

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 diese 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 [Wil06b] 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. Als Grundlage für

weitere Betrachtungen wird die Kategorisierung von Christian Will gewählt, da sie die Änderungen an einem XML-Schema vollständig erfasst.

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 (*sequence*), Alternative (*choice*), Menge(*all*))
      - 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 [Wil06b]

Kapitel 4

Untersuchung

In diesem Kapitel werden die einzelnen Änderungen an einem XML-Schema hinsichtlich ihrer Auswirkung auf die Kapazität des Schemas und ihrem tatsächlichen Einfluss auf die dazugehörigen Instanzen untersucht. Für jede Änderung wird zudem der Aufwand, getrennt nach Update-, Lösch- und Einfüge-Kosten, angegeben.

Die betrachtete Operationsmenge orientiert sich an der Kategorisierung von Christian Will [Wil06a], unterteilt einige Schemaänderungen aber noch weiter. Die untersuchten Komponenten liegen der Definition des W3C (siehe [DCF], [PVB], [HST] und [TB]) zugrunde.

4.1 Ändern des Schemas

Beim Hinzufügen einer Definition bzw. Deklaration muss unterschieden werden, ob diese lokal oder global erfolgt, da die Auswirkungen auf Instanzen und Informationskapazität unterschiedlich ist. Wird eine globale Definition gelöscht, so muss überprüft werden, wo diese verwendet wird und der Lösch-Aufwand an allen Stellen berechnet bzw. abgeschätzt werden.

Eine Zusammenfassung der nachfolgend betrachteten Evolutionsschritte gibt die Tabelle 4.1.

Hinzufügen einer globalen Definition bzw. Deklaration

Wird im Schema eine neue Komponente, wie z.B. ein neues Attribut oder eine neue Elementgruppe, global definiert, so bestehen zunächst keine Referenzen auf diese Komponente; sie wird nicht benutzt. Dadurch werden die Instanzen nicht beeinflusst und die Informationskapazität bleibt konstant.

Hinzufügen einer Typdefinition

Das Hinzufügen einer neuen Typdefinition (*simpleType*) hat keinen Einfluss auf Instanzen, da bestehende Attribute und Elemente ihn noch nicht benutzen. Es besteht entsprechend kein Änderungsaufwand und die Kapazität bleibt erhalten.

Entfernen einer Typdefinition

Wird eine Typdefinition aus dem Schema entfernt, so müssen alle damit verbundenen Elemente und Attribute (und ggf. weitere Typdefinitionen) angepasst werden. Dies kann entweder dadurch geschehen, dass alle betroffenen Komponenten gelöscht oder angepasst werden. Die Anpassung kann z.B. dadurch erfolgen, dass der Typ der Elemente und Attribute auf einen verwandten vordefinierten Datentyp (notfalls *anyType*) geändert wird. Die erste Variante verursacht einen Lösch-Aufwand in Höhe aller gelöschten Attribute und Elemente, außerdem wird die Informationskapazität reduziert. Letztere Variante verursacht

keinen Änderungsaufwand und erhält die Kapazität, benötigt aber weitere Änderungen im XML-Schema um die Datentypen zu ändern. In der prototypischen Umsetzung wird die zweite Version umgesetzt.

Hinzufügen einer lokalen Attributdeklaration

Beim Hinzufügen einer lokalen Attributdeklaration muss zunächst überprüft werden, wie oft das Attribut in den Instanzen verwendet werden wird. Ist das Attribut als optional gekennzeichnet, so müssen die Instanzen nicht geändert werden. Falls das Attribut nicht optional ist, betragen die Hinzufüge-Kosten pro Vorkommen *1* (dem Namen des Attributes) sofern kein default-Wert angegeben ist, *2* mit default-Wert. Die Kapazität wird durch die Änderungen erweitert.

Entfernen einer Attributdeklaration

Das Entfernen einer Attributdeklaration löscht alle Attribute, die diese Deklaration verwendet. Ist das Attribut optional, so muss der Anteil der tatsächlich vorkommenden Attribute berechnet bzw. abgeschätzt werden. Es wird weiterhin angenommen, dass für jedes vorkommende Attribut auch ein Wert vorliegt. Die Kosten für das Entfernen eines Attributes mit Wert wird mit *2* angeben (1 für den Namen des Attributes, 1 für den Wert). Die Informationskapazität wird durch diese Änderung reduziert.

Hinzufügen einer lokalen Elementdeklaration

Wird ein einfaches Element (*Element ohne complexType*) lokal in das Schema eingefügt, so muss ermittelt werden, wie oft er in den Instanzen verwendet wird. Sofern das Element mindestens einmal vorkommen wird, müssen die Instanzen durch das Hinzufügen des Elementes verändert werden. Der Einfüge-Aufwand wird mit *2 bis 3* gewertet. Die Kosten ergeben sich aus dem Start- (1) und dem Endtag (1) und, sofern ein *default*- oder ein *fixed*-Wert angegeben ist, aus dem Einfügen der Information. Die Informationskapazität wird durch diese Änderung erweitert.

Entfernen einer Elementdeklaration

Beim Entfernen eines einfachen Elementes muss, wie bereits beim Einfügen, überprüft werden, wie oft dieses verwendet wird. Sofern das Element mindestens einmal vorkommt, werden die Instanzen durch das Löschen des Elementes verändert. Der Lösch-Aufwand wird mit *3* gewertet. Die Kosten ergeben sich aus dem Start- (1) und dem Endtag (1). Es wird weiterhin angenommen, dass das Element auch einen Inhalt hat (1). Alternativ könnte der Anteil der Elemente mit Inhalt berechnet werden. Dann würde der Lösch-Aufwand zwischen *2 und 3* liegen.

Hinzufügen einer lokalen Attributgruppen-Definition

Wird eine Attributgruppe lokal eingefügt, entspricht dies dem Einfügen mehrerer Attribute. Die Instanzen müssen entsprechend angepasst werden. Da das Hinzufügen eines Attributes eine erweiternde Wirkung auf die Informationskapazität hat, gilt dies auch für das Hinzufügen einer Attributgruppe. Die Kosten ergeben sich aus der Summe der hinzugefügten Attribute.

Entfernen einer Attributgruppen-Definition

Ähnlich wie bei Löschung einer Attributdeklaration verhält es sich bei dem Entfernen einer Attributgruppen-Definition. Es muss bestimmt werden, wie oft die Attribute in den den Instanzen verwendet werden. Der Lösch-Aufwand ergibt sich aus der Anzahl der Attribute und deren Häufigkeit. Die Informationskapazität wird durch das Entfernen der Attribute reduziert.

Operation	Instanz-anpassung	Kapazität	Aufwand		
			Update	**Löschen**	**Hinzufügen**
Hinzufügen einer globalen Definition	nein	erhaltend	0	0	0
Hinzufügen einer Datentypdefinition	nein	erhaltend	0	0	0
Entfernen eines einfachen Datentyps	ja	reduzierend	0	(#Attribute löschen + #Elemente löschen) die diesen Typ benutzen	0
Hinzufügen einer lokalen Attributdeklaration	ja/nein	erhaltend	0	0	#Attribute*2
Entfernen einer Attributdeklaration	ja/nein	reduzierend	0	#Attribute*2	0
Hinzufügen einer lokalen Elementdeklaration	ja/nein	erhaltend	0	0	#Elemente * (#Attribute hinzufügen + Zeichenketten + Tags)
Entfernen einer Elementdeklaration	ja	reduzierend	0	#Elemente * (#Attribute löschen + Zeichenketten + Tags)	0
Hinzufügen einer lokalen Attributgruppen-Definition	ja/nein	erhaltend	0	0	#Gruppen * (#Attribute hinzufügen)
Entfernen einer Attributgruppen-Definition	ja/nein	reduzierend	0	#Gruppen * (#Attribute löschen)	0
Hinzufügen einer lokalen Elementgruppen-Definition	ja/nein	erhaltend	0	0	#Elementgruppen * (#Attribute hinzufügen + #Elemente hinzufügen + Tags)
Entfernen einer Elementgruppen-Definition	ja	reduzierend	0	#Elementgruppen * (#Attribute löschen + #Elemente löschen + Tags)	0
Hinzufügen einer Notation	nein	erhaltend	0	0	0
Entfernen einer Notation	nein	erhaltend	0	0	0
Hinzufügen einer Anmerkung	nein	erhaltend	0	0	0
Entfernen einer Anmerkung	nein	erhaltend	0	0	0

Tabelle 4.1: Ändern des Schemas

Hinzufügen einer lokalen Elementgruppen-Definition

Wird eine neue lokale Elementgruppe eingefügt, so muss ermittelt werden, wie oft die Elementgruppe in einer Instanz auftaucht. Die Kosten, die beim Entfernen einer einzelnen Elementgruppe entstehen, lässt sich unter dem Abschnitt 4.2.2 nachlesen. Die Informationskapazität wird durch diesen Evolutionsschritt erweitert.

Entfernen einer Elementgruppen-Definition

Auch beim Löschen einer Elementgruppe muss bestimmt werden, wie oft diese in den zum Schema gehörenden Dokumenten vorkommen. Die Informationskapazität wird durch diese Schemänderung reduziert. Genaue Angaben zur Berechnung des Aufwandes befindet sich unter dem Abschnitt 4.2.2.

Hinzufügen und Entfernen einer Notation oder Anmerkung

Notationen und Anmerkungen sind lediglich Teil der XML-Schema Dateien; in Instanzen kommen sie nicht vor. Anmerkungen dienen zum einen der Dokumentation (*documentation*) und bieten Informationen für Hilfsprogramme (*appInfo*). Ähnliches gilt für Notationen. Das Hinzufügen und Entfernen dieser zwei Komponenten aus dem Schema hat keinen Einfluss auf dazugehörige Instanzen und die Informationskapazität.

4.2 Ändern des Inhaltes einer Typdefinition

4.2.1 Ändern der Attributmenge

Hinzufügen eines neuen Attributes Wird einem Element ein neues Attribut hinzugefügt, so muss zunächst überprüft werden, ob es optional ist und über einen default- oder fixed-Wert verfügt. Ist das Attribut optional, so müssen die Instanzen nicht angepasst werden. Falls das Attribut nicht-optional ist, so muss der Name des Attributes gefolgt von einem Gleichheitszeichen und zweier Hochkommata in jedes Vorkommen des Elementes eingetragen werden. Existiert zusätzlich noch ein default- oder fixed-Wert, so muss dieser zusätzlich eingetragen werden. Der Einfüge-Aufwand wird für optionale Attribute mit *0*, für nicht-optionale Attribute ohne Wert mit *1* und für nicht-optionale Attribute mit *2* pro Vorkommen abgeschätzt. Die Informationskapazität wird durch diese Schemaänderung erhöht.

Löschen eines Attributes Sobald ein Attribut aus einem XML-Schema gelöscht wird, müssen auch alle Vorkommen in Instanzen entfernt werden. Es wird dabei angenommen, dass jedes Vorkommen auch über einen Attributwert verfügt. Das Entfernen eines Attribute verursacht dabei immer einen Löschaufwand von *2* (1 für Attributname, 1 für Attributwert). Die Häufigkeit eines optionalen Attributes muss zudem geringer abgeschätzt werden, wenn seine Anzahl nicht durch Betrachtung konkreter Instanzen ermittelt wird. Die Informationskapazität wird durch diese Änderung reduziert.

Umbenennen eines Attributes Durch das Umbenennen eines Attributes im Schema, müssen von allen Attributen in den Instanzen der Attributname geändert werden. Da nur der Name, aber nicht der Wert geändert wird, beträgt der Update-Aufwand pro verändertem Attribut lediglich *1*. Die Informationskapazität bleibt durch das Umbenennen des Attributnamens erhalten.

Ändern der Attributverwendung und der Wertebereichsbeschränkung Attribute können durch die Angabe von use-, default-, und fixed-Werten weiter spezifiziert werden. Einige dieser Änderungen haben Auswirkungen auf Instanzen. Folgende Möglichkeiten gibt es zur Änderung des use-Wertes:

1. Setzen des use-Wertes auf optional: Keine Änderung, da der use-Wert standardmäßig optional ist. Die Informationskapazität bleibt erhalten.

2. Setzen des use-Wertes auf required: Es muss überprüft werden, wieviele Attribute tatsächlich eingetragen wurden. Dieser Anteil wird mit 0,5 abgeschätzt. Entsprechend müssen die fehlenden Attribute in den Instanzen ergänzt werden. Die Informationskapazität wird reduziert, da diese Operation nicht rückgänig gemacht werden kann.

3. Setzen des use-Wertes auf prohibited: Alle Vorkommen des Attributes müssen gelöscht werden. Die Informationskapazität wird reduziert.

4. Setzen des use-Wertes von optional auf required: Entspricht dem erstmaligen Setzen des use-Wertes auf required (siehe 2.).

5. Setzen des use-Wertes von optional auf prohibited: Das Attribut kommt vor der Änderung nicht in allen Elementen vor. Es müssen daher nur die Elemente angepasst werden, die das Attribut besitzen. Die Informationskapazität wird hierdurch reduziert.

6. Setzen des use-Wertes von required auf optional: Keine Änderung, da Attribut vorher schon eingetragen ist. Die Informationskapazität wird allerdings erweitert, da nun auch Elemente ohne das Attribut erlaubt sind.

7. Setzen des use-Wertes von required auf prohibited: Alle Vorkommen des Attributes müssen gelöscht werden. Die Informationskapazität wird reduziert.

8. Setzen des use-Wertes von prohibited auf optional: Da das Attribut vor der Änderung nicht vorhanden war und nach dem neuen Schema nicht unbedingt in den Instanzen vorkommen muss, erfolgt keine Änderung an den Dokumenten. Die Informationskapazität wird erweitert, da das Attribut nun erlaubt ist.

9. Setzen des use-Wertes von prohibited auf required: In allen Elementen, die das Attribut verwenden, muss ein Wert eingetragen werden, da dieser vor der Änderung nicht vorhanden war. Die Informationskapazität wird erweitert.

10. Löschen des use-Wertes optional: Da der Standardwert für use optional entspricht, ändert sich durch diese Operation nichts an den Instanzen.

11. Löschen des use-Wertes required: Diese Operation entspricht dem Setzen des use-Wertes von required auf optional (siehe 6.).

12. Löschen des use-Wertes prohibited: Diese Operation entspricht dem Setzen des use-Wertes von prohibited auf optional (siehe 8.).

Wird der default-Wert geändert, hat dies keinen Einfluss auf bestehende Dokumente, da der Wert nur bei der Generierung eines neuen Attributes verwendet wird, wenn kein anderer Wert angegeben wurde. Von bestehenden Attributen kann nicht ermittelt werden, ob ihr Attributwert aus der default-Angabe hervorgeht.

Für geänderte fixed-Werte müssen alle betroffenen Attributwerte geändert werden. Die Informationskapazität bleibt durch diese Änderung erhalten, da der ursprüngliche fixed-Wert wieder hergestellt werden kann. Wird fixed erstmalig gesetzt, müssen auch alle Attributwerte angepasst werden, die Informationskapazität wird jedoch reduziert, da die alten Werte nicht wieder hergestellt werden können.

Ändern der Typdefinition Wird der Typ eines Attributes geändert, muss überprüft werden, ob die Attributwerte der Instanzen mit dem neuen Typ kompatibel sind. Falls ja, erfolgt keine Anpassung der Instanzen und die Änderung erhält die Kapazität. Dies ist insbesondere der Fall, wenn ein Subtyp generalisiert wird. Sind die Typen nicht kompatibel, so muss ein Defaultwert generiert werden. Der Evolutionsschritt führt zu Änderungen an den Instanzen. Die Informationskapazität wird durch diese Evolution verändert, da der Attributwert nicht wiederhergestellt werden kann.

Fremde Attribute

Hinzufügen einer Attribut-Wildcard Nach Christian Will ist die Verwendung von Attributen, die durch eine Wildcard in ein XML-Schema eingefügt werden, immer optional, wodurch keine Instanzanpassungen notwendig sind. Die Informationskapazität wird durch diesen Evolutionsschritt erweitert.

Entfernen einer Attribut-Wildcard Das Entfernen einer Attribut-Wildcard führt dazu, dass alle Attribute, auf die die Wildcard passt, aus den Instanzen entfernt werden müssen. Dieser Vorgang entspricht dem Löschen von optionalen Attributen.

Ändern der Attribut-Wildcard Änderungen an der Attribut-Wildcard betreffen die erlaubten Namensräume und das Ändern der Validierungsvorschrift. Christian Will erläutert in seiner Diplomarbeit ([Wil06a]), dass Veränderungen an der Validierungsvorschrift entweder keinen Einfluss auf die Instanzen haben oder die Schemaevolution abbricht. In beiden Fällen bleibt die Informationskapazität erhalten. Werden zu den erlaubten Namensräumen neue hinzugefügt, wird die Kapazität des Schemas erweitert, die Instanzen bleiben jedoch unverändert. Werden erlaubte Namensräume entfernt, müssen die betroffenen Attribute des Elementes angepasst und eventuell sogar gelöscht werden. Dadurch verringert sich die Informationskapazität.

4.2.2 Ändern der Elementgruppe (Komplexer Inhalt)

Ändern des Typs (Sequenz, Alternative, Menge) Die Elemente eines komplexen Typs können in einer Sequenz (*sequence*) angeordnet werden, d.h. sie treten in einer festen Reihenfolge auf. In einigen Anwendungsfällen kann es jedoch sein, dass nicht alle Elemente benötigt werden und deren Reihenfolge beliebig sein kann (Menge, *all*), oder dass genau ein Element aus einer Liste benötigt wird (Alternative, *choice*). Beim Ändern des Typs können folgende sechs Fälle auftreten:

1. Eine Sequenz wird zu einer Menge
2. Eine Sequenz wird zu einer Alternative
3. Eine Alternative wird zu einer Menge
4. Eine Alternative wird zu einer Sequenz
5. Eine Menge wird zu einer Sequenz
6. Eine Menge wird zu einer Alternative

Sequenz zu Menge Wird der Typ eines komplexen Elementes von *sequence* auf *all* geändert, so erfolgt in den Instanzen keine Änderung. In der Menge befinden sich nach der Änderung alle Elemente in einer festen Reihenfolge. Es müssen keine neuen Elemente hinzugefügt werden und keine bestehenden gelöscht werden. Da an den Instanzen keine Anpassung erfolgt, ist die Schemänderung kapazitätserhaltend.

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Hinzufügen eines neuen Attributes	ja, sofern nicht-optional	erweiternd	0	0	2
Löschen eines Attributes	ja, sofern nicht-optional	reduzierend	0	2	0
Umbenennen eines Attributes	ja, sofern nicht-optional	erhaltend	1	0	0
Ändern der Attributverwendung	siehe Tabelle 4.3				
Ändern des fixed-Wertes	ja	erhaltend / reduzierend	#Attribute	0	0
Ändern des default-Wertes	nein	erhaltend	0	0	0
Ändern der Typdefinition	evtl.	erhaltend / verändernd	#Attribute	0	0
Hinzufügen einer Attribut-Wildcard	nein	erweiternd	0	0	0
Entfernen einer Attribut-Wildcard	ja	reduzierend	0	# nicht-optionaler Attribute löschen	0
Ändern der Attribut-Wildcard (Validierung)	nein	erhaltend			
Ändern der Attribut-Wildcard (Namensräume)	evtl.	Namensräume hinzufügen : erweiternd, Namensräume entfernen: reduzierend	1 (Attribut ändern)	2 (Attribut entfernen	0

Tabelle 4.2: Ändern der Attributmenge

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
use="optional" einfügen	nein	erhaltend	0	0	0
use="required" einfügen	ja	reduzierend	0	0	0,5*#Attribute einfügen
use="prohibited" einfügen	ja	reduzierend	0	0,5*#Attribute löschen	0
"optional" → "required"	ja	reduzierend	0	0	0,5*#Attribute einfügen
"optional" → "prohibited"	ja	reduzierend	0	0,5*#Attribute löschen	0
"required" → "optional"	nein	erweiternd	0	0	0
"required" → "prohibited"	ja	reduzierend	0	#Attribute löschen	0
"prohibited" → "optional"	nein	erweiternd	0	0	0
"prohibited" → "required"	ja	erweiternd	0	0	#Attribute einfügen
use="optional" entfernen	nein	erhaltend	0	0	0
use="required" entfernen	nein	erweiternd	0	0	0
use="prohibited" entfernen	nein	erweiternd	0	0	0

Tabelle 4.3: Ändern der Attributverwendung

Sequenz zu Alternative Bei der Änderung des Typs von *sequence* auf *choice* muss für die Instanzen eines der bestehenden Elemente ausgewählt werden, welches erhalten bleibt, während alle anderen Elemente gelöscht werden müssen. Der Lösch-Aufwand beträgt entsprechend die Summe aller gelöschten Elemente pro Vorkommen des komplexen Typs. Durch das Löschen kann die Änderung nicht rückgängig gemacht werden ohne die Instanz erneut anzupassen, daher ist die Operation kapazitätsreduzierend.

Alternative zu Menge Die Menge, die sich durch das Ändern des Typs von *choice* zu *all* ergibt, besteht aus dem jeweilig zuvor ausgewähltem Element. Es erfolgen keine Anpassungen an den Instanzen und die Kapazität bleibt gleich.

Alternative zu Sequenz Wird der Typ eines komplexen Elementes von *choice* auf *sequence* geändert, so fehlen in den Instanzen sämtliche Elemente, mit Ausnahme des Elementes, welches vor der Änderung mittels *choice* ausgewählt wurde. Entsprechend müssen in den Instanzen alle fehlenden Elemente mittels Einfüge-Operationen hinzugefügt werden. Diese Änderung ist das Gegenstück zum Ändern des Typs von Sequenz auf Alternative. Durch das Hinzufügen der Elemente kann sie ebenfalls nicht ohne Anpassungen der Instanzen rückgängig gemacht werden und erweitert daher die Kapazität.

Menge zu Sequenz Die Möglichkeit den Typ eines komplexen Elementes von *all* nach *sequence* zu ändern sorgt dafür, dass in den Instanzen gleich zwei Änderungen erfolgen können, aber nicht unbedingt auftreten müssen. Da in einer Sequenz alle Elemente vorkommen müssen, in der Menge aber nicht unbedingt, müssen zum einen alle fehlenden Elemente ergänzt werden. Falls sie schon vorhanden sind, erfolgt keine Änderung. Außerdem müssen die Elemente ggf. umgeordnet werden, da die Sequenz eine bestimmte Reihenfolge der Elemente erwartet. Das Umsortieren ergibt einen Update-Aufwand, der sich nach der Anzahl der Elemente im Komplexen Typ und dem verwendeten Sortieralgorithmus richtet. Die Instanzen müssen nur angepasst werden, wenn Elemente fehlen und eine Umsortierung notwendig ist. Durch das Hinzufügen von Elementen ist die Änderung kapazitätserweiternd. Der Gesamtaufwand ergibt sich aus dem Sortieren und dem Hinzufügen der Elemente.

Menge zu Alternative Wird eine Änderung des Typs eines komplexen Elementes von *all* auf *choice* gesetzt, so muss zunächst überprüft werden, ob in einer Instanz ein Sub-Element aus dem komplexen Typen vorkommt. Ist dies nicht der Fall, so muss ein neues Element im komplexen Typen eingefügt werden. Die Änderung ist dann kapazitätserhaltend, da die Menge auch mit einem Element weiterhin gültig ist. Falls mindestens ein Element in der Menge enthalten ist, so muss eines ausgewählt werden und alle weiteren werden entfernt. Es liegt dann eine Reduzierung der Kapazität vor. Sofern nicht genau ein Element im komplexen Typen enthalten ist, erfolgt eine Anpassung der Instanzen. Der Aufwand ergibt sich nach der Unterscheidung nach kapazitätserhaltender oder -reduzierender Wirkung.

Ändern des minimalen und maximalen Vorkommens Das Ändern des minimalen und maximalen Vorkommens hat, je nachdem wie die Werte neu gesetzt werden, eine unterschiedliche Auswirkung auf die Instanzen und die Informationskapazität.

Verringern des minimalen Vorkommens Diese Operation hat keine Auswirkung auf bestehende Instanzen. Durch die Verringerung des minOccurs-Wertes eines Elementes ist es möglich, dass auch eine geringere Anzahl dieses Elementes in den Dokumenten vorkommen kann, da aber dort diese Anzahl bereits durch den alten minOccurs-Wert erreicht werden musste, gilt dies auch für die neue Schema-Definition. Die Informationskapazität wird durch diese Änderung erweitert.

Erhöhen des minimalen Vorkommens Wird das minimale Vorkommen einer Elementgruppe erhöht, so müssen die Instanzen ggf. angepasst werden. Dies ist der Fall, wenn es in den Instanzen Elementgruppen gibt, deren Anzahl unterhalb des neuen minOccurs-Wert liegen. Der Anteil dieser Komponenten wird mittels Gleichverteilung abgeschätzt. Dieser Anteil ergibt sich aus dem alten und neuen minOccurs-Werten im Verhältnis zum maxOccurs-Wert: $\frac{\sum_{i=1}^{minOccurs_{neu}-minOccurs_{alt}} i}{maxOccurs-minOccurs_{alt}+1}$. Es müssen so viele neue Elementgruppen hinzugefügt werden, wie die Differenz aus alten und neuem minOccurs-Wert beträgt. Die Informationskapazität wird durch diesen Evolutionsschritt verringert, da die hinzugefügten Elementgruppen bei einer Zurücksetzung des minOccurs-Wertes nicht gelöscht werden würden.

Verringern des maximalen Vorkommens Das Verringern des maximalen Vorkommens einer Elementgruppe hat Einfluss auf diejenige Komponenten einer Instanz, in der die betroffene Elementgruppe nach der Schemänderung öfter vorkommt, als der neue maxOccurs-Wert erlaubt. Der Anteil lässt sich wie folgt berechnen: $\frac{maxOccurs_{alt}-maxOccurs_{neu}}{maxOccurs_{alt}} = 1 - \frac{maxOccurs_{neu}}{maxOccurs_{alt}}$. Die Differenz aus altem und neuen maxOccurs-Wert gibt die Menge der zu löschenden Elementgruppen an. Die Informationskapazität wird durch diese Änderung verringert, da die gelöschten Elementgruppen nicht wieder hergestellt werden können.

Erhöhen des maximalen Vorkommens Eine Erhöhung des maximalen Vorkommens einer Elementgruppe hat keine Auswirkung auf Instanzen, da der maxOccurs-Wert nur angibt, wie viele Elementgruppen vorkommen können. Die Informationskapazität wird durch diese Operation erweitert, da sie die Umkehroperation zum Verringern des maximalen Vorkommens ist.

Hinzufügen eines neuen Elements Wird ein Element eine Elementgruppe hinzugefügt, so muss unterschieden werden, ob es sich bei der Elementgruppe um eine Sequenz, einer Menge oder einer Alternative handelt. Das Hinzufügen eines einzelnen Elementes verursacht Kosten für das Erstellen der Start- und Endtags und das Einfügen der zum Element gehörigen Attribute. Ist das Element selbst eine Elementgruppe, müssen für alle untergeordneten Elemente die Einfügekosten hinzu addiert werden. Enthält das Element lediglich einen einfachen Wert, müssen nur die Kosten für das Löschen dieses Wertes zu den Gesamtkosten hinzugefügt werden. Die Informationskapazität wird durch diese Änderung erweitert.

Alternative Ist die Elementgruppe eine Alternative, so wurde in den bestehenden Instanzen bereits ein Element aus der Alternative ausgewählt. Das neue Element wird entsprechend keine Kosten verursachen, da es nicht auftreten wird, wenn die Instanzen geändert werden.

Menge In einer Menge müssen nicht alle Elemente vorkommen. Dementsprechend muss das im Schema neu eingefügte Element nicht in den Instanzen eingefügt werden, wodurch auch keine Kosten entstehen.

Sequenz Wird ein Element einer Sequenz hinzugefügt, so wird es immer in den Instanzen vorkommen, sofern die dazugehörige Elementgruppe in den Instanzen verwendet wird. Der Einfügeaufwand ergibt sich aus dem Vorkommen der Elementgruppe multipliziert mit dem Kosten für das Hinzufügen eines einzelnen Elementes wie oben beschrieben.

Entfernen eines Elementes Das Entfernen eines Elementes verursacht Kosten für das Löschen der Start- und Endtags und der dazugehörigen Attribute. Ist das Element selbst eine Elementgruppe, müssen die Kosten für die dazugehörigen Elemente mitberechnet werden. Andernfalls entstehen nur die Kosten für das Entfernen eines einfachen Wertes. Wie bereits beim Hinzufügen eines Elementes muss man auch beim Entfernen den Typ der übergeordneten Elementgruppe beachten. Die Informationskapazität wird in allen drei Fällen reduziert.

Alternative In einer Alternative ist nicht sichergestellt, ob das zu löschende Element auch ausgewählt wurde. Der Anteil der betroffenen Alternativen wird mit $\frac{1}{|Anzahl der Elemente in der Elementgruppe|}$ abgeschätzt. Der gesamte Löschaufwand ergibt sich aus diesem Anteil multipliziert mit dem Löschen eines einzelnen Elementes.

Menge Auch in einer Menge muss das zu löschende Element nicht immer vorkommen. Der Anteil, der mit 0,5 abgeschätzt wird (Element kommt mit gleicher Wahrscheinlichkeit vor oder nicht vor), wird mit den Kosten für das Löschen eines Elementes multipliziert um den Gesamtaufwand zu bestimmen.

Sequenz Ist der Typ der übergeordneten Elementgruppe eine Sequenz, so kommt das zu löschende Element immer in den Instanzen vor. Entsprechend muss der Aufwand für das Löschen eines einzelnen Elementes nicht mit einem Faktor multipliziert werden.

Umbenennen eines Elementes Wird ein Element umbenannt, so müssen in den Instanzen die Start- und Endtags angepasst werden. Die Informationskapazität bleibt durch diesen Evolutionsschritt erhalten. Ob, und wie oft ein Element betroffen ist, hängt wie beim Löschen vom Typ der Elementgruppe ab. Der Anteil für die Sequenz, Menge und Alternative entspricht dem Löschen eines Elementes.

Hinzufügen einer Elementgruppe Beim Hinzufügen einer Elementgruppe in ein XML-Schema wird die Informationskapazität immer erweitert. Auch die Instanzen müssen durch diese Änderung angepasst werden. Der Aufwand für das Einfügen einer Gruppe lässt sich allerdings erst durch die den Typ der Elementgruppe feststellen. Die Einfügekosten ergeben sich aus dem Hinzufügen des Start- und Endtags der Elementgruppe und der dazugehörigen Attribute. Weitere Kosten entstehen für das Hinzufügen der untergeordneten Elemente, der je nach Gruppentyp anders ausfällt.

Menge Bei den Elementen einer Menge handelt es sich um optionale Komponenten. Dadurch muss beim Einfügen der Elementgruppe während der Schemaevolution keines dieser Elemente erstellt werden, wodurch keine Kosten entstehen.

Sequenz Zu den Basis-Kosten für das Einfügen einer Elementgruppe kommen die Kosten für das Einfügen aller dazugehörigen Elemente.

Alternative Die Elemente der Elementgruppe können einen unterschiedlichen Einfügeaufwand verursachen. Um den Aufwand abzuschätzen, werden die Kosten für alle Elemente ermittelt und durch die Zahl der Elemente geteilt. Der so gebildete Mittelwert stellt den Aufwand da, der entstehen würde, wenn alle Elemente in etwa gleich oft in den Instanzen vorkommen würden.

Entfernen einer Elementgruppe Wird eine Elementgruppe aus dem Schema entfernt, müssen die Start- und Endtags, sowie die zur Gruppe gehörenden Attribute gelöscht werden. Außerdem müssen alle untergeordneten Elemente entfernt werden, deren Aufwand je nach Typ der Elementgruppe unterschiedlich abgeschätzt wird. Die Informationskapazität wird durch diesen Evolutionsschritt reduziert.

Sequenz Die Kosten für das Entfernen einer Sequenz entspricht der Summe für das Entfernen aller Elemente aus der Sequenz.

Menge Der Aufwand ergibt sich aus den Kosten, der entsteht, wenn alle Elemente entfernt werden, wobei der Faktor von 0.5, der beim Löschen eines einzelnen Elementes aus einer Menge auftritt, vor diese Summe gezogen werden kann. Damit entspricht der abgeschätzte Aufwand für das Entfernen einer Menge der Hälfte für das Entfernen einer Sequenz.

Alternative Der durchschnittliche Aufwand für das Löschen eines der Elemente aus der Elementgruppe entspricht in einer Alternative dem Mittelwert für das Löschen aller Elemente.

Fremde Elemente

Hinzufügen einer Wildcard Wird eine Wildcard hinzugefügt, so müssen so viele Elemente eingefügt werden, wie der minOccurs-Wert beträgt. Für Wildcards, die keine Elemente vorgeben, wird laut Will keine Schemaevolution durchgeführt. Die Informationskapazität wird hierdurch erweitert.

Entfernen einer Wildcard Das Entfernen einer Wildcard aus dem Schema reduziert die Kapazität des Schemas. Alle nicht mehr erlaubten Elemente müssen aus dem Schema wieder entfernt werden.

4.2.3 Ändern des Inhalts der Elemente

Ändern der zugewiesenen Typdefinition Wird der Typ für den Inhalt eines Elementes geändert, muss überprüft werden, ob der neue Typ mit dem alten verträglich ist. Die Beziehung der einzelnen Datentypen lässt sich unter [HST] nachlesen. Sind die Typen miteinander verträglich, müssen die Instanzen nicht angepasst werden. Wenn nicht, muss für das Element ein neuer Wert generiert werden oder der Wert gelöscht werden. In beiden Fällen wird der Aufwand mit 1 abgeschätzt.

Ändern des minimalen und maximalen Vorkommens Der Aufwand für das Ändern des minimalen und maximalen Vorkommens eines Elementes entspricht von der Berechnung her dem der Elementgruppe (siehe 4.2.2).

Ändern der Nullwertfähigkeit Um zu unterscheiden, ob der Wert eines Elementes nicht angegeben oder nicht bekannt ist, gibt es in XML-Schema die Eigenschaft nillable. Wird der Wert auf true gesetzt, erscheint in den Elementen das Attribut xsi:nil. Der dazugehörige Attributwert gibt an, ob ein NULL-Wert vor liegt oder nicht.

1. keine Angabe zur Nullwertfähigkeit → nillable="false"
2. keine Angabe zur Nullwertfähigkeit →nillable="true"
3. nillable="false" → nillable="true"
4. nillable="false" → keine Angabe zur Nullwertfähigkeit
5. nillable="true" → nillable="false"
6. nillable="true" → keine Angabe zur Nullwertfähigkeit

zu 1.) Da der Standardwert für nillable false ist, findet keine Änderung an der Nullwertfähigkeit des Elementes statt. Folglich brauchen die Instanzen nicht angepasst zu werden.

zu 2.) Durch das Setzen von *nillable="true"* muss dem Element das Attribut *xsi:nil* hinzugefügt werden. Dadurch müssen die Instanzen angepasst werden und die Änderung erweitert die Informationskapazität. Die Kosten für die Anpassung entsprechen denen für das Hinzufügen eines Attributes.

zu 3.) Das Ändern des Attributwertes von false auf true entspricht dem erstmaligen Setzen von nillable="true" (siehe Fall 2).

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Ändern des Typs (Sequenz, Alternative, Menge)	siehe Tabelle 4.5				
Verringern des minimalen Vorkommens	nein	erweiternd	0	0	0
Erhöhen des minimalen Vorkommens	ja	reduzierend	0	0	maximal $minOccurs_{neu}$ - $minOccurs_{alt}$ Elementgruppen
Verringern des maximalen Vorkommens	ja	reduzierend	0	maximal $maxOccurs_{alt}$ - $minOccurs_{neu}$ Elementgruppen	0
Erhöhen des maximalen Vorkommens	nein	erweiternd	0	0	0
Hinzufügen eines neuen Elements (in Alternative)	nein	erweiternd	0	0	0
Hinzufügen eines neuen Elements (in Menge)	nein	erweiternd	0	0	0
Hinzufügen eines neuen Elements (in Sequenz)	ja	erweiternd	0	0	#Elementgruppen * Element hinzufügen
Entfernen eines Elementes (aus Alternative)	ja	reduzierend	0	$\frac{1}{\lvert\#Elemente\rvert}$	0
Entfernen eines Elementes (aus Menge)	ja	reduzierend	0	#Elementgruppen *0,5 * #Elemente	0
Entfernen eines Elementes (aus Sequenz)	ja	reduzierend	0	#Elementgruppen * #Elemente	0
Umbenennen eines Elementes	ja	erhaltend	2	0	0
Hinzufügen einer Elementgruppe (Alternative)	ja	erweiternd	0	0	∅ Element einfügen
Hinzufügen einer Elementgruppe (Menge)	nein	erweiternd	0	0	0
Hinzufügen einer Elementgruppe (Sequenz)	ja	erweiternd	0	0	$\sum$ Element einfügen
Entfernen einer Elementgruppe (Alternative)	ja	reduzierend	0	∅ Element löschen	0
Entfernen einer Elementgruppe (Menge)	ja	reduzierend	0		0 0,5 * $\sum$ Element löschen
Entfernen einer Elementgruppe (Sequenz)	ja	reduzierend	0	$\sum$ Element löschen	0
Hinzufügen einer Wildcard	ja	erweiternd	0	0	# Elemente einfügen
Entfernen einer Wildcard	ja	reduzierend	0	#Elemente entfernen	0

Tabelle 4.4: Ändern der Elementgruppe

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Sequenz → Menge	nein	erhaltend	0	0	0
Sequenz → Alternative	ja	reduzierend	0	#Elemente-1 löschen	0
Alternative → Menge	nein	erhaltend	0	0	0
Alternative → Sequenz	ja	erweiternd	0	0	#Elemente-1 hinzufügen
Menge → Sequenz	evtl.	erweiternd	Umordnen der Elemente	0	#fehlende Elemente hinzufügen
Menge → Alternative	ja	erhaltend/ reduzierend	0	#vorhandene Elemente-1 löschen	Element hinzufügen

Tabelle 4.5: Ändern des Typs einer Elementgruppe

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Ändern der zugewiesenen Typdefinition	evtl.	erhaltend / verändernd	0 oder 1	0	0
Ändern des minimalen und maximalen Vorkommens	entspricht den Angaben in Tabelle 4.4				
Ändern der Nullwertfähigkeit	siehe Tabelle 4.7				

Tabelle 4.6: Ändern des Inhalts der Elemente

zu 4.) Wird im XML-Schema die Angabe nillable="false" gelöscht, wird der default-Wert für nillable benutzt. Da dieser ebenfalls false ist, findet keine Änderung am Schema statt.

zu 5.) Durch das Ändern des Attributwertes von nillable von true auf false muss in dem betroffenen Element das Attribut *xsi:nil* gelöscht werden. Die Kosten für diese Anpassung entsprechen denen für das Löschen eines einfachen Attributes. Die Informationskapazität wird hierdurch reduziert.

zu 6.) Dieser Evolutionsschritt entspricht dem Fall 5.

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Setzen von nillable="false"	nein	erhaltend	0	0	0
Setzen von nillable="true"	ja	erweiternd	0	0	Hinzufügen eines Attributes
nillable="true" → nillable="false"	ja	reduzierend	0	Löschen eines Attributes	0
nillable="false" → nillable="true"	ja	erweiternd	0	0	Hinzufügen eines Attributes
Löschen von nillable="false"	nein	erhaltend	0	0	0
Löschen von nillable="true"	ja	reduzierend	0	Löschen eines Attributes	0

Tabelle 4.7: Ändern der Nullwertfähigkeit

4.2.4 Ändern des einfachen Datentyps (Einfacher Inhalt)

Wird ein einfacher Datentyp verändert, muss überprüft werden, ob Attribute oder Elemente angepasst werden müssen. Ist dies der Fall, muss die Anzahl der veränderten Elemente und Attribute ermittelt werden.

Ändern des Typs (Atomar, Liste, Vereinigung) Ein einfacher Datentyp ist entweder atomar, oder kommt als Liste oder Vereinigung vor. Die sechs Möglichkeiten zur Änderung lassen sich der Tabelle 4.8 hinsichtlich ihrer Informationskapazität und Einfluss auf die Instanzen. Eine detaillierte Beschreibung kann der Diplomarbeit von Christian Will ([Wil06a]) entnommen werden.

Grundlegende Fassetten Wird die grundlegende Fassette eines einfachen Datentyps geändert, entspricht dies dem Ändern der Typdefinition wie bei einem Attribut (siehe 4.2.1).

Einschränkende Fassetten Durch zwölf einschränkende Fassetten, wie beispielsweise der minimalen und maximalen Länge, lassen sich einfache Datentypen genauer beschreiben. Eine genauere Betrachtung für jede einzelne einschränkende Fassetten würde den Rahmen dieser Bachelorarbeit überschreiten. Es wird daher angenommen, dass etwa 90% aller Attributwerte und Elemente von einer Änderung an den Fassetten betroffen sind.

Operation	Instanz-anpassung	Kapazität	Aufwand		
			Update	**Löschen**	**Hinzufügen**
atomar zu Liste	nein	erweiternd	0	0	0
atomar zu Vereinigung	nein	erweiternd	0	0	0
Liste zu Vereinigung	ja	verändernd	0	0	0
Liste zu atomar	ja	reduzierend	0	Listenlänge-1	0
Vereinigung zu atomar	ja	reduzierend	Anzahl der betroffenen Instanzwerte	0	0
Vereinigung zu Liste	ja	verändernd	Anzahl der betroffenen Instanzwerte	0	0

Tabelle 4.8: Ändern des Typs eines Datentyps

Operation	Instanz-anpassung	Kapazität	Aufwand		
			Update	**Löschen**	**Hinzufügen**
Ändern des Typs	siehe Tabelle 4.8				
Grundlegende Fassetten	siehe Abschnitt Ändern einer Typdefinition				
Einschränkende Fassetten	ja	verändernd	Anteil * Zeichenkette anpassen	0	0

Tabelle 4.9: Ändern des einfachen Datentyps

4.2.5 Umbenennung der Typdefinition

Diese Operation hat keinen Einfluss auf bestehende Instanzen, da Typdefinitionen nur im Schema selbst verwendet werden. Die Informationskapazität bleibt entsprechend erhalten.

4.2.6 Ändern des Ziel-Namensraumes

Diese Schemänderung hat keinen Einfluss auf Instanzen, wodurch die Informationskapazität erhalten bleibt und keine Kosten entstehen.

4.2.7 Festlegen des Inhaltstyps

Beim Festlegen des Inhaltstyps ergibt sich der Typ meistens bereits aus der Typdefinition (einfacher oder komplexer Inhalt). Eine Sonderrolle nimmt die Eigenschaft mixed ein. Ist der Attributwert von mixed auf true gesetzt, können in einem komplexen Typ neben Elementen auch Zeichenketten vorkommen. Wird keine Angabe zum mixed-Wert gemacht, ist gemischter Inhalt standardmäßig verboten. Wird mixed von false auf true gesetzt, erfolgt keine Anpassung der Instanzen (und somit entstehen auch keine Kosten) und die Informationskapazität wird erweitert. Erfolgt eine Änderung des Wertes von true auf false, wird die Kapazität reduziert und alle Zeichenketten im betroffenen Element müssen gelöscht werden. Dieser Anteil wird mit 0,3 abgeschätzt.

4.2.8 Festlegen der Eigenschaft Abgeschlossenheit

Eine Typdefinition kann dadurch, dass sie als abgeschlossen definiert wurde, nicht in einer anderen Typdefinition vorkommen. Wird der Typ bereits in einer anderen Typdefinition verwendet, obwohl sie durch eine Schamaänderung abgeschlossen wird, so wird die Schmaänderung verworfen. Ist die Typdefinition

durch eine Schemaevolution nicht mehr abgeschlossen, so istkeine Änderung der Instanzen notwendig. Die Informationskapazität bleibt erhalten.

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Umbenennung der Typdefinition	nein	erhaltend	0	0	0
Ändern des Ziel-Namensraumes	nein	erhaltend	0	0	0
Festlegen des Inhaltstyps auf mixed="true"	nein	erweiternd	0	0	0
Festlegen des Inhaltstyps auf mixed="false"	ja	reduzierend	0	Löschen der Zeichendaten	0
Festlegen der Eigenschaft Abgeschlossenheit	nein	erhaltend	0	0	0

Tabelle 4.10: Weitere Operationen

4.3 Ändern der Typhierarchie

Einfache Typdefinition zu einer Vereinigung hinzufügen

Eine Anpassung der Instanzen ist durch das Hinzufügen einer Typdefinition in eine Vereinigung nicht notwendig. Die Kapazität wird allerdings erweitert, da die Vereinigung einen größeren Wertebereich besitzt.

Einfache Typdefinition aus einer Vereinigung entfernen

Durch diese Operation verkleinert sich der Wertebereich der Vereinigung, wodurch sich die Informationskapazität reduziert. Für alle Elemente, die nun außerhalb des Wertebereichs liegen, muss ein neuer Wert generiert werden. Dieser Anteil wird $\frac{1}{|AnzahlderDatentypenin der Vereinigung vor dem Entfernen|}$ abgeschätzt.

Komplexe Typdefinition in Hierarchie als Erweiterung oder Einschränkung einfügen

Diese Operation hat keinen Einfluss auf bestehende XML-Dokumente. Die Informationskapazität bleibt erhalten, da kein bestehendes Attribut oder Element diesen Datentyp benutzt.

Entfernen einer Typdefinition aus der Hierarchie

Ist die zu entfernende Typdefinition eine Erweiterung eines bestehenden Typs, so müssen die Elemente und Attribute, die diesen Typ verwenden, entweder angepasst oder aus dem Schema gelöscht werden. Christian Will schlägt in [Wil06a] vor, die betroffenen Elemente zu löschen. Der gesamte Löschaufwand entspricht der Summe der entfernten Elemente und Attribute. Die Informationskapazität wird hierdurch reduziert. Wenn die Typdefinition eine Einschränkung eines bestehenden Typs ist, müssen die Instanzen nicht angepasst werden und die Informationskapazität bleibt erhalten.

Operation	Instanz-anpassung	Kapazität	Aufwand		
			Update	**Löschen**	**Hinzufügen**
Einfache Typdefinition zu einer Vereinigung hinzufügen	nein	erweiternd	0	0	0
Einfache Typdefinition aus einer Vereinigung entfernen	ja	reduzierend	(#betroffener Elemente bzw. Attribute) * $\frac{1}{\lvert AnzahlDatentypen \rvert}$	0	0
Komplexe Typdefinition in Hierarchie als Erweiterung einfügen	nein	erhaltend	0	0	0
Komplexe Typdefinition in Hierarchie als Einschränkung einfügen	nein	erhaltend	0	0	0
Entfernen einer erweiternden Typdefinition aus der Hierarchie	ja	reduzierend	0	# betroffende Attribute und Elemente	0
Entfernen einer einschränkenden Typdefinition aus der Hierarchie	nein	erhaltend	0	0	0

Tabelle 4.11: Ändern der Typhierarchie

4.4 Ändern von Beziehungen

Hinzufügen eines Primärschlüssels(key)

Wird einer Schemakomponente ein Primärschlüssel hinzugefügt, wird die Informationskapazität durch diesen Evolutionsschritt reduziert, da nun gefordert wird, dass eine bereits vorhandene Schemakomponente eindeutig sein soll. In den Instanzen müssen eventuell vorkommende Duplikate entfernt werden. Da keine Informationen über den Inhalt der betroffenen Elemente vorliegen, muss der Anteil der Duplikate, die aus den Instanzen entfernt werden müssen, abgeschätzt werden. Es wird angenommen, dass etwa jedes 10. Element eines simpleType die gleichen Werte aufweist. Bei einem complexType ergibt sich der Anteil aus den dazugehörigen Kind-Elementen. Besteht beispielsweise ein complexType aus der Sequenz von zwei simpleTypes, ergibt sich ein Duplikat-Anteil von 0,1*0,1 = 0,01 $\hat{=}$ 1%. Dieser Anteil wird mit der Anzahl aller betroffenen Komponenten multipiziert um die Zahl der Duplikate abzuschätzen.

Entfernen eines Primärschlüssels(key)

Durch das Entfernen eines Primärschlüssels bleibt die Informationskapazität des Schemas erhalten und auch die Instanzen bleiben unverändert. Allerdings müssen die zu diesem Primärschlüssel gehörenden Fremdschlüssel entfernt werden.

Hinzufügen und Entfernen eines optionalen Primärschlüssels(unique)

Diese Operationen entsprechen hinsichtlich der Anpassung der Instanzen und der Veränderung der Informationskapazität den Evolutionsschritten für nicht-optionale Primärschlüssel.

Hinzufügen eines Fremdschlüssels

Wird einer Schemakomponente ein Fremdschlüssel eingefügt, müssen die referenzierten Elemente an die Stelle des Fremdschlüssels eingefügt werden. Die gesamten Kosten ergeben sich aus dem Aufwand für das Hinzufügen eines einzelnen Elementes und der Anzahl der neuen Elemente. Die Informationskapazität bleibt erhalten, da keine neuen Elemente auftauchen.

Entfernen eines Fremdschlüssels

Durch das Entfernen eines Fremdschlüssels müssen in den Instanzen die kopierten Komponenten wieder gelöscht werden, während die referenzierten Elemente erhalten bleiben. Dadurch bleibt die Informationskapazität erhalten. Der Aufwand für das Aktualisieren der Instanzen entspricht den Kosten für das Löschen der referenzierenden Elemente.

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Hinzufügen eines (optionalen) Primärschlüssels	ja	reduzierend	0	(Anteil der Duplikate)*(Anzahl der Elemente)	0
Entfernen eines (optionalen) Primärschlüssels	nein	erweiternd	0	0	0
Hinzufügen eines Fremdschlüssels	ja	erhaltend	refenziertes Element einfügen	0	0
Entfernen eines Fremdschlüssels	ja	erhaltend	referenziertes Element löschen	0	0

Tabelle 4.12: Ändern von Beziehungen

4.5 Ändern von Identitätsdefinitionen

Hinzufügen einer Identität

Das Hinzufügen einer neuen Identität hat weder Einfluss auf die XML-Dokumente noch auf die Informationskapazität, da zum Zeitpunkt keine Identitätsreferenzen auf die Identität verweisen.

Entfernen der Identität

Durch das Löschen einer Identität müssen alle Identitätsreferenzen, die sich auf die Identität beziehen, entfernt werden. Das Löschen der Identität selbst verursacht hingegen keinen Aufwand und erhält die Informationskapazität.

Hinzufügen einer Identitätsreferenz

Wird eine Identitätsreferenz einem Element hinzugefügt, muss das referenzierte Element an dieser Stelle eingefügt werden. Identitätsreferenzen entsprechen vom Aufwand und der Informationskapazität her den Fremdschlüsseln, nur das ID's sich auf ein einzelnes Element beziehen.

Entfernen einer Identitätsreferenz

Durch das Entfernen einer Identitätsreferenz ist es notwendig die Elemente, welche die Referenz benutzen, aus den Instanzen zu entfernen. Die Informationskapazität des Schemas bleibt erhalten, da die referenzierten Elmente weiterhin in den Instanzen vorkommen. Es entstehen Kosten für das Entfernen der Elemente.

Operation	**Instanz-anpassung**	**Kapazität**	**Aufwand**		
			Update	**Löschen**	**Hinzufügen**
Hinzufügen einer Identität	nein	erhaltend	0	0	0
Entfernen der Identität	nein	erhaltend	0	0	0
Hinzufügen einer Identitätsreferenz	ja	erhaltend	Hinzufügen des referenzierten Elementes	0	0
Entfernen einer Identitätsreferenz	ja	erhaltend	Löschen des referenzierten Elementes	0	0

Tabelle 4.13: Ändern von Identitätsdefinitionen

Kapitel 5

Prototyp

Die Kosten eines einzelnen Evolutionschrittes lassen sich in den meisten Fällen leicht überblicken. Wird ein XML-Schema überarbeitet, kommt es häufig vor, dass viele Veränderungen auftreten und die zum Schema gehörigen Instanzen entsprechend oft angepasst werden müssen. Um den Aufwand weiterhin überschauen zu können, wird in diesem Kapitel ein Prototyp vorgestellt, der in Abbildung 5.1 mit seinen einzelnen Modulen dargestellt ist. Ziel des Prototypen ist es, dass der Benutzer, der ein XML-Schema ändern will, die maximalen Kosten vorgeben kann und das Programm bereits nach dem Evolutionsschritt abbricht, der diese Angabe übersteigt. Der Prototyp ist Teil des Projektes *CodeX* und ist dort als Plugin integriert. CodeX ist konzeptueller Editor, der im Rahmen der Diplomarbeit von Robert Stephan ([Ste06]) entstand und durch weitere studentische Arbeiten in seinen Funktionalitäten ergänzt wurde. Die Implementation erfolgte in Java und basiert auf dem Eclipse Framework. Wird ein XML-Schema verändert, generiert CodeX eine Folge von Evolutionsschritten, die zusammen mit dem XML-Schema und der Eingabe der maximalen Kosten durch den Benutzer die Parameter für das neue Plugin darstellen.

5.1 Architektur

Ermittlung der Evolutionsschritte

Alle Evolutionsschritte liegen in einer Liste vor, welche sequentiell durchlaufen wird. Bei Programmstart und nach der Abarbeitung einer Schemänderung wird überprüft, ob ein weiterer Evolutionsschritt abgearbeitet werden muss, sofern die maximalen Kosten noch nicht überschritten wurden. Liegt ein weiterer Schritt vor, so wird dieser anschließend kategorisiert. Falls nicht, wurde die gesamte Schemaevolution abgeschlossen ohne dabei dem vom Benutzer vorgegenen Aufwand zu übersteigen. In diesem Fall erfolgt die Ausgabe, dass die Schemaevolution durchgeführt werden sollte.

Kategorisierung

Die Kategorisierung eines Evolutionschrittes entscheidet, ob dieser Einfluss auf die zum XML-Schema zugehörigen Instanzen hat. Dabei wird zunächst überprüft, ob die Art der Änderung unabhängig von der betroffenen Schemakomponente die Instanzen beeinflussen kann. dabei können drei Fälle auftreten:

1. Die Änderung hat niemals Einfluss auf die Instanzen.
2. Der Einfluss auf die Instanzen ist erst bei näherer Betrachtung des XML-Schema erkennbar, d. h., es werden zunächst die Parameter der aktuellen Schemaänderung mit den dazugehörigen Einträgen im Schema verglichen, um zu entscheiden, ob eine Anpassung der Instanzen notwendig ist.
3. Die Änderung wird immer die Instanzen beeinflussen.

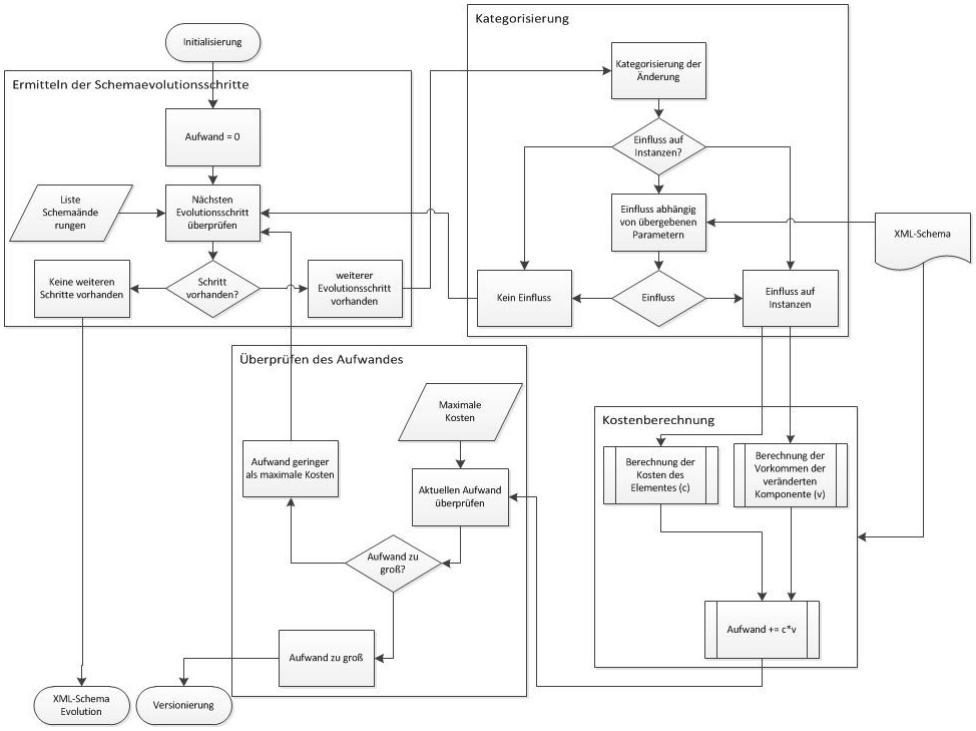


Abbildung 5.1: Aufbau des Prototypen

Hat eine Schemaänderung unter keinen Umständen Einfluss auf Instanzen (wie das Hinzufügen einer Annotation), so verursacht dieser Evolutionsschrittes keine Kosten und es wird der nächste Schritt, sofern vorhanden, ermittelt.

Evolutionsschritte die unabhängig von ihren Parametern immer die zum Schema gehörenden Dokumente verändern, können sofort hinsichtlich ihrer Kosten berechnet werden.

Für die restlichen Änderungen am XML-Schema-Änderungen müssen die übergebenen Parameter mit dem Schema abgeglichen werden, um zu überprüfen, ob Instanzen von der Änderung wirklich betroffen sind. Stellt sich heraus, dass dabei keine Instanzen bertoffen sind, entstehen keine Kosten für den Evolutionsschritt und es kann mit der nächsten Schemaänderung fortgesetzt werden. Andernfalls folgt die Berechnung des Aufwandes.

Berechnung der Kosten eines Einzelschrittes

Für die Berechnung der Kosten eines Einzelschrittes müssen zwei Berechnungen durchgeführt werden. Neben der Art der Änderung wird das XML-Schema benötigt, welches evolutioniert wird.

Abschätzung der Anzahl der veränderten Komponenten in den Instanzen Zum einen muss abgeschätzt werden, wie oft die veränderte Komponente in den Instanzen vorraussichtlich vorkommen wird. Dazu müssen alle Elemente, die sich zwischen der Wurzel des Schemas und der betrachteten Komponente befinden, hinsichtlich ihrer minOccurs- und maxOccurs-Werte untersucht werden. Aus diesen Werten lassen sich für jedes Element ein Mittelwert((maxOccurs-minOccurs):2) bilden. Außerdem muss beachtet werden, dass einzelne Elemente sich in einer Sequenz, Menge oder Alternative befinden und somit nicht unbedingt in den Instanzen vorkommen müssen.

- Ein Element in einer Sequenz muss immer genau einmal vorkommen, die Wahrscheinlichkeit, dass es in den Instanzen vorkommt, beträgt daher 1.

- In einer Menge kommt ein Element maximal einmal vor. Ohne statistische Informationen sind keine genaue Angaben zum tatsächlichen Vorkommen möglich; die Wahrscheinlichkeit eines Auftretens wird daher mit 0,5 abgeschätzt

- Wird als Typ eine Alternative angegeben, so wird genau eine der angebenen Elemente ausgewählt. Ohne genauere Verteilungsangaben wird das Vorkommen mit $\frac{1}{|AnzahlderElementeimkomplexenTyp|}$ abgeschätzt

Das gesamte Vorkommen der veränderten Komponente ergibt sich aus dem Produkt der Mittelwerte und der Abschätzung für die komplexen Typen.

Kosten eines Elementes Die Kosten eines einzelnen Elementes ergeben sich entsprechend der Untersuchung in Kapitel 4. Abschätzungen mittels minOccurs- und maxOccurs-Werten sowie dem Typ von komplexen Elementen innerhalb der betrachteten Komponente gelten analog.

Die Gesamtkosten eines Evolutionsschrittes ergeben sich aus der abgeschätzten Anzahl der Komponenten multipliziert mit den Kosten einer einzelnen Komponente. Liegt der Aufwand für einen Evolutionsschritt vor, wird dieser zum Gesamtaufwand hinzugefügt.

Überprüfen des Gesamtaufwandes

Der Gesamtaufwand wird nach jedem Evolutionsschritt überprüft, welcher Kosten verursacht. Dabei wird überprüft, ob der aktuelle Aufwand die vom Benutzer übergebenen maximalen Kosten übersteigt. Ist der Aufwand zu groß, werden keine weiteren Evolutionsschritte durchgeführt und es erfolgt eine Ausgabe, dass statt einer Evolution des XML-Schemas eine Versionierung durchgeführt werden sollte. Die Versionierung und die dafür notwendigen Betrachtungen sind nicht Bestandteil der vorliegenden Arbeit. Werden die maximalen Kosten nicht überstiegen, wird, sofern vorhanden, der nächste Evolutionsschritt abgearbeitet.

5.2 Beispiel

Nachfolgend wird der Programmablauf des Prototyps durch ein kurzes Beispiel demonstriert.

Eingabeparameter Als Ausgangsschema dient das XML-Schema aus Kapitel 1 (siehe 2.1). Es werden folgende drei Evolutionsschritte durchgeführt:

1. Dem Schema wird eine Annotation im simpleType „class“ eingefügt.

2. Der minOccurs-Wert in der Referenz zu „player“ wird auf 1 gesetzt.

3. Das Attribut „name“ im Element „player“ wird in „nickname“ umbenannt.

Die maximalen Kosten für die gesamte Schemaevolution soll den Wert 42 nicht überschreiten.

Programmablauf Mit dem Start des Programmes werden die XML-Schema-Datei, die maximalen Kosten und die Liste der Schemänderungen mit übergeben. Der aktuelle Gesamtaufwand wird mit 0 initialisiert. Anschließend wird überprüft, ob ein Evolutionsschritt in der Liste auftaucht.

Die Evolutionsschritte werden intern als eigener Datentyp dargestellt und zusammen mit ihren, sofern vorhanden, Parametern in die Liste der Evolutionsschritte eingetragen. Über diese Liste wird solange iteriert, bis entweder alle Schritte abgearbeitet wurden oder nach einem Evolutionsschritt die maximalen Kosten überschritten wurden.

Da ein Schritt vorhanden ist, wird dieser anschließend kategorisiert. In der ersten Stufe wird überprüft, ob er niemals Einfluss oder immer Einfluss auf mögliche Instanzen vom XML-Schema hat, unabhängig davon wie das Schema aufgebaut ist, oder ein Einfluss erst erkennbar ist, wenn man seine Parameter mit dem XML-Schema abgleicht.

Der erste Schritt ist das Hinzufügen einer Annotation. Dieser wird intern durch *addAnnotation* repräsentiert und besitzt keine Parameter. Nach der Aufwandsuntersuchung in Kapitel 4 verursacht diese Operation keine Kosten, da Instanzen niemals betroffen sind. Aus diesem Grund bleibt der aktuelle Aufwand bei 0. Eine Überprüfung, ob der aktuelle Aufwand die maximalen Kosten übersteigt, ist dadurch auch nicht nötig.

Anschließend wird überprüft, ob weitere Evolutionsschritte vorliegen. Da dies der Fall ist, wird der nachfolgende Schritt, das Ändern des minOccurs-Wertes, kategorisiert. Diese Evolution ist eine Änderung, deren tatsächliches Wirken auf die Instanzen sich erst in Verbindung mit der XML-Schema Datei bestimmen lässt. Der Änderung *changeElementMinOccur* werden zwei Parameter übergeben: der neue minOccurs-Wert und das betroffene Element. Nun wird das XML-Schema eingelesen und der aktuelle minOccurs-Wert des Elementes „player“ ermittelt. Dieser beträgt 2 und ist somit größer als der neue Wert. Dadurch hat dieser Evolutionsschritt auch keinen Einfluss auf die Instanzen und verursacht keine Kosten.

Daran anknüpfend folgt der letzte Evolutionsschritt, das Umbenennen eines Attributes. Dies ist eine Änderung, die abhängig von ihren Parameter (Attributverwendung) die Instanzen beeinflusst. Ist der use-Wert nicht auf prohibited gesetzt, folgt die Berechnung der Kosten dieses Schrittes, ansonsten entstehen keine Kosten. Um den Aufwand bestimmen zu können, muss zum einen das Vorkommen des Attributes in den Instanzen abgeschätzt werden und der Aufwand für die Änderung ermittelt werden.

Das Umbenennen eines Attributes wurde in Kapitel 4 mit dem Kostenfaktor 1 für jedes Vorkommen abgeschätzt. Für die Abschätzung der betroffenen Elemente wird eine Berechnung mittels Gleichverteilung wie oben beschrieben durchgeführt. Das Element „player“, welches das Attribut „name“ enthält, kommt im komplexen Typ „playerlist“ mindestens zwei, maximal jedoch 25 mal vor (siehe Abbildung 5.2). Dies ergibt ein durchschnittliches Auftreten von 13,5 Elementen des Typ „player“. Da der komplexe Typ „playerlist“ eine Sequenz enthält, in der „player“ vorkommt, wird dieser Wert mit 1 multipliziert.

```
<xsd:element name="playerlist">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="player" minOccurs="2" maxOccurs="25"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Abbildung 5.2: Ausschnitt aus dem XML-Schema - Elementgruppe „playerlist“

Die „playerlist“ ist Bestandteil der Sequenz, die sich im Element „encounter“ befindet, wodurch das Vorkommen des Attributes „name“ bei 13,5 unverändert bleibt. „encounter“ erscheint in der übergeordneten „encounterlist“ als Teil einer Sequenz mindestens einmal vor (siehe Abbildung 5.3). Der dazugehörige maxOccurs-Wert ist mit „unbounded“ angegeben. Da der Prototyp keine Informationen aus Instanzen bezieht um ein maximales Vorkommen abschätzen zu können, wird der maxOccur-Wert mit dem fünf-

fachen des minOccurs-Wertes abgeschätzt. Dieser Faktor kann bei Bedarf angepasst werden. Das bisherige Vorkommen von 13,5 Attributen wird entsprechend mit dem Wert 3 multipliziert ((5+1):2 = 3). Dementsprechend kommt des Attribut „name“ in einer „encounterlist“ ca 40,5-mal vor. Dieses Element ist wiederum Teil der Sequenz im komplexen Element „raid“, wodurch der Wert aber unverändert bleibt. Da „raid“ das Wurzelelement des Schemas ist, kommt das Attribut „name“ in einer dazugehörigen Instanz etwa 40,5-mal vor.

```
<xsd:element name="encounterlist">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="encounter" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Abbildung 5.3: Ausschnitt aus dem XML-Schema - Elementgruppe „encounterlist“

Anschließend wird der Wert 40,5 zum aktuellen Aufwand hinzuaddiert. Da zuvor kein Evolutionsschritt Kosten verursachte, beträgt der aktuelle Aufwand nun 40,5. Dieser Wert wird nun mit dem maximalen Aufwand verglichen, der vom Benutzer mit 42 festgelegt wurde. Da der aktuelle Aufwand weiterhin kleiner ist als die vorgegeben Kosten, wird die Evolution fortgesetzt.

Da kein weiterer Evolutionsschritt abgearbeitet werden muss, terminiert das Programm mit der Ausgabe, dass eine XML-Schema-Evolution durchgeführt werden sollte.

Kapitel 6

Schlussbetrachtung

6.1 Zusammenfassung

Diese Bachelorarbeit gibt einen Überblick über mögliche Kategorisierungen von XML-Evolutionsschritten und untersucht, welcher Aufwand entsteht, wenn Dokumente an das neue Schema angepasst werden müssen. Es wird eine Übersicht über die Kategorisierungsansätze verschiedener Forschungsgruppen gegeben. Speziell wurden die Schemänderungen in Bezug auf ihrer Auswirkung auf die Informationskapazität und der Anpassung der Instanzen hinsichtlich Update-, Lösch- und Einfügeaufwand analysiert.

6.2 Weitere Ideen

Die prototypische Implementierung konzentriert sich darauf, die Kosten einer betroffenen Komponente in einer Instanz und die Anzahl der betroffenen Komponenten anhand des dazugehörigen XML-Schemas zu ermitteln. Dabei werden mathematische Verteilungsfunktionen mit Abschätzungen verbunden.

- Besitzt ein Element einen maxOcurrs-wert, der auf „unbounded“ gesetzt ist, berechnet der Prototyp einen numerischen Wert in Abhängigkeit vom minOccurs-Wert. Um eine ungenaue Abschätzung zu vermeiden, werden statistische Informationen über die tatsächlich auftretenden Werte benötigt. Alternativ könnte das Programm während der Kalkulierung der Kosten den Benutzer nach einem maxOccurs-Wert abfragen, wenn das Element erstmals in einer Berechnung vorkommt.
- Wird das Vorkommen eines Elementes mittels minOccurs- und maxOccurs-Werten abgeschätzt, schätzt der Prototyp das Vorkommen mittels Gleichverteilung ab. Alternativ könnte man andere Verteilungsfunktionen implementieren, zwischen denen der Benutzer wählen kann oder auf statistische Informationen zurückgreifen. Gleiches gilt für die Abschätzung für Alternativen und Mengen in einem komplexen Element.
- Durch weitere statistische Informationen lässt sich auch im Vorfeld klären, ob eine Änderung überhaupt Auswirkungen auf Instanzen besitzt. Falls ja, lässt sich auch der Anteil der betroffenen Dokumente bestimmen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[ACP09] Antonio Cicchetti, Davide Di Ruscio und Alfonso Pierantonio: *Managing Dependent Changes in Coupled Evolution.* In: *ICMT 2009*, Seiten 35 – 51, 2009.

[DCF] 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 22.08.2011.

[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.

[HST] Henry S. Thompson, David Beech: *XML Schema Part 1: Structures Second Edition.* http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/structures.html, zuletzt aufgerufen am 19.09.2011.

[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.

[PVB] Paul V. Biron, Ashok Malhotra: *XML Schema Part 2: Datatypes Second Edition.* http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html, zuletzt aufgerufen am 19.09.2011.

[Ste06] STEPHAN, ROBERT: *Entwicklung und Implementierung einer Methode zum konzeptuellen Entwurf von XML-Schemas.* Diplomarbeit, Universität Rostock, 2006.

[TB] TOBY BAIER, MICHAEL EBERT: *XML Schema Teil 0: Einführung.* http://www.edition-w3.de/TR/2001/REC-xmlschema-0-20010502/, zuletzt aufgerufen am 19.09.2011.

[Wil06a] WILL, CHRISTIAN: *Ableitung von Schemaevolutionsschritten aus XML-Updateoperationen.* Studienarbeit, Universität Rostock, 2006.

[Wil06b] 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.