eCH-0062: Design von XML Schemas - Zusammenfassung

Name	Design von XML Schemas – Zusammenfassung
Standard-Nummer	eCH-0062
Kategorie	Hilfsmittel
Reifegrad	Experimentell
Version	1.0
Status	Aufgehoben
Beschluss am	
Ausgabedatum	
Ersetzt Standard	-
Sprachen	Deutsch
Autoren	Fachgruppe XML Pina Alexander - Unisys (Schweiz) AG – http://www.unisys.ch
Herausgeber / Vertrieb	Verein eCH, Amthausgasse 18, 3011 Bern T 031 560 00 20, F 031 560 00 25 www.ech.ch/ info@ech.ch

Zusammenfassung

Das vorliegende Dokument wurde in den eCH-0018 ab Version 2.0 integriert.

Inhaltsverzeichnis

1 Status des Dokuments

Aufgehoben: Das Dokument wurde von eCH zurückgezogen. Es darf nicht mehr genutzt werden.

1.1 Terminologie der Empfehlungen

Richtlinien in diesem Dokument werden gemäss der Terminologie aus [RFC2119] angegeben, dabei kommen die folgenden Ausdrücke zur Anwendung, die durch GROSSSCHREIBUNG als Wörter mit den folgenden Bedeutungen kenntlich gemacht werden (Zitat aus [RFC2119]):

- **MUST**: This word, or the terms "**REQUIRED**" or "**SHALL**", mean that the definition is an absolute requirement of the specification.
- **MUST NOT**: This phrase, or the phrase "**SHALL NOT**", mean that that definition is an absolute prohibition of the specification.
- **SHOULD**: This word, or the adjective "**RECOMMENDED**", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT**: This phrase, or the phrase "**NOT RECOMMENDED**" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY**: This word, or the adjective "**OPTIONAL**", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

2 Einleitung

Dieses Dokument gibt in einer kompakten Form den Inhalt des Dokumentes eCH0035 Design von XML Schemas wieder. Es ersetzt in keinem Fall die in eCH0035 normierten Aussagen. Die beiden Dokumente sind als komplementär zu betrachten.

Dieses Dokument kann als Zusammenfassung dienen, welche schnell über Do's und Dont's im XML Schema Bereich Auskunft gibt.

Für eine komplette und detaillierte Übersicht sollte eCH0035 konsultiert werden. Dieses Dokument kondensiert die Aussagen auf eine rein zielorientierte Sicht, und erklärt die Motivationen hinter den Normierungen des eCH0035 Dokumentes.

Alle relevanten Empfehlungen werden erwähnt, dies erlaubt einer/m nicht versierten Benutzer/inn, ohne detaillierte Kenntnisse der Funktionsweisen bereits eigene Schemas zu spezifizieren, und Fehlerquellen von vornherein auszuschliessen.

3 Zusammenfassung der Empfehlungen

3.1 Versionen von XML und XML Schema

3.1.1 Zusammenfassung

XML Schemas werden in XML geschrieben. XML selber kann in verschiedenen Versionen angewendet werden. Aktuell sind dies Version 1.0 und Version 1.1. XML Version 1.1 wird praktisch nur in Zusammenhang mit Grossrechner Systemen benötigt.

Generell soll Bundesweit die gleiche Version zur Anwendung kommen. Aus diesem Grund ist Version 1.0 des XML Standards zu bevorzugen.

3.1.2 Empfehlung

SHOULD - Es sollte bevorzugt XML Schemas in der Version 1.0 des XML Standards geschrieben werden.

MAY: Ist die Verwendung von XML 1.1 unbedingt notwendig, so kann die Verarbeitung von XML 1.1 in der durch eine W3C Note [xml11schema10] beschriebenen Weise mit XML Schema 1.0 kombiniert werden.

3.2 XML Schema und XML Namespaces

3.2.1 Zusammenfassung

XML Instanz Dokumente einer XML Schema Spezifikation verweisen auf einen „targetNamespace“. Als XML Instanz Dokumente werden XML Dokumente bezeichnet, welche zu einer XML Schema Spezifikation zugehörig sind, also gegenüber der XML Schema Spezifikation gültig und valide sind.

Um eine Katalogisierung der Schemas im Bundesumfed durchführen zu können, soll der Namespace auf eine „wenn möglich“ erreichbare URL verweisen. Hinter dieser URL soll eine Beschreibung der Spezifikation auffindbar sein welche nach [eCH033] erstellt worden ist. Nachfolgend ein mögliches Beispiel einer solchen Namespace Einbindung.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="0"
           targetNamespace="http://www.ech.ch/xmlns/minorversion/v1">
  <xs:attribute name="minorVersion" type="xs:token"/>
</xs:schema>
```

3.2.2 Empfehlung

SHOULD – Der Namespace eines XML Schemas sollte nicht direkt auf das Schema zeigen, sondern auf eine Beschreibung gemäss [eCH033].

3.3 Lokale/Globale Deklaration von Elementen/Typen

3.3.1 Zusammenfassung

Es gibt verschiedene Design Varianten wie eine XML Schema Spezifikation erstellt werden kann. Je nach Variante befinden wir uns auf der einen oder anderen Seite eines polarisierten Spektrums. Unter Design Varianten werden hier Muster/Patterns verstanden wie ein XML Schema Spezifikation erstellt werden kann.

Als Beispiel ist das „Russian Doll“ Muster komplementär zu dem Salami Slice“ Muster. Das „Venetian Blind“ Muster befindet sich in der Mitte dieser beiden Pole. Das „Venetian Blind“ Muster erlaubt gezielte Wiederverwendung von Typen sowie eben auch gezielte nicht Wiederverwendung von Typen. In einem „Russian Doll“ Muster kann praktisch nichts wieder verwendet werden und in einem „Salami Slice“ Muster alles.

Durch den Grundsatz, dass ein fachliches Modell angewendet wird, ergibt sich der Einsatz des „Venetian Blind“ Musters. Da das „Venetian Blind“ Muster explizit erlaubt die Spezifikation so zu Strukturieren, dass Typen gekapselt und auch gezielt wieder verwendet werden kann.

Als Beispiel kann die Strukturierung natürliche Person verwendet werden. Natürliche Personen besitzen meist eine oder mehrere Adressen. Nun kann die Adresse so gestaltet werden (Durch Erstellung eines Typs Adresse), dass sie z.bsp auch für eine juristische Person wieder verwendet werden kann.

- In einem „Russian Doll“ Modell ist nur das Root-Element wieder verwendbar, also die Struktur „Person“ als ganzes.
- In einem „Salami Slice“ Modell sind alle Elemente wieder verwendbar, so kann die Postleitzahl mit dem Personennamen kombiniert werden, so lassen sich neue Strukturen spezifizieren, welche im Kontext des fachlichen Modells absolut keine Daseinsberechtigung führen.

Die genauen Funktionsweisen dieser Muster können in [eCH0035] nachgeschlagen werden.

3.3.2 Empfehlung

SHOULD: Das vorherrschende Design-Muster sollte das „Venetian Blind“ Design sein, das Typen generell global, und Elemente und Attribute je nach Bedarf lokal oder global definiert (global nur dann, wenn sie an verschiedenen Stellen wiederverwendet werden). Typen sind also der Aspekt eines Schemas, der speziell hervorgehoben wird und durch die generell globale Definition zur Wiederverwendung gut geeignet ist.

MAY: Sollen Elemente wiederverwendet werden, so können auch diese global definiert werden (dies ist z.B. bei Rekursionen unerlässlich). Dabei ist jedoch darauf zu achten, dass im Venetian Blinds Design die Typen die primäre Art der Wiederverwendung darstellen, so dass jeder dieser Fälle einzeln betrachtet werden sollte.

SHOULD NOT: Bei der lokalen Definition von Elementen oder Attributen sollte darauf geachtet werden, dass keine Elemente oder Attribute gleichen Namens lokal definiert werden, weil dies bei Benutzern des Schemas Irritationen erzeugen kann. Handelt es sich um das gleiche

Element, sollte eine globale Definition erfolgen, die referenziert wird. Handelt es sich um unterschiedliche Konzepte, sollten diese nicht mit dem gleichen Namen benannt werden.

3.4 Namens- vs. typbasierte Verarbeitung

3.4.1 Zusammenfassung

Es gibt folgende Arten XML Instanz Dokumente zu Verarbeiten.

- Typbasierend
- Namensbasierend

Der Einsatz der jeweiligen Verarbeitungsmethoden ist abhängig von den eingesetzten Hilfsmitteln und Programmiersprachen/Frameworks. In typisierten Umgebungen wie z.bsp JAVA oder .NET kann typisiert gearbeitet werden, dies muss aber nicht zwingend sein. In nicht typisierten Umgebungen wie PHP, Perl oder andere Skriptsprachen kann nur namensbasierend gearbeitet werden. Generell kann gesagt werden dass die Typbasierende Verarbeitung immer stärker durch Entwicklungsumgebungen und Frameworks gefördert wird.

Ein Hinweis auf Typbasierende Verarbeitung ist die Verwendung von Vererbungen in der XML Schema Spezifikation. Diese Mechaniken erfordern, dass ein XML Parser zur Laufzeit den aktuellen Typ einer XML Elementen Struktur ermitteln muss, also ein klassisches „Late Binding“ zur Laufzeit erfolgen muss.

Weiterhin unterstützen nicht alle XML Sprachen wie XSLT, XQUERY die komplette Typbasierende Verarbeitung, aus diesem Grund ist der Einsatz von Typbasierender Verarbeitung zu dokumentieren.

3.4.2 Empfehlung

MUST – Werden Mechaniken in der XML Schema Spezifikation verwendet, welche Typbasierende Verarbeitung erfordern so ist dies explizit in der Dokumentation zu beschreiben.

3.5 Type Substitution Mechanismen in XML Schema

3.5.1 Zusammenfassung

Bei dem Einsatz von typisierten XML Schema Spezifikationen können folgende Arten der Substitution von Typen angewendet werden.

- Elementsubtitution
- Typsubstitution

Generell wird davon ausgegangen das Typsubstition nicht nötig sind. In Spezialfällen welche sich konsequent am objektorientierten Paradigma ausrichten, können sie vorkommen. Dies vor allem bei dem Einsatz von programmiertechnischen Frameworks, wie die XML Serialisierung von JAVA oder .NET

Wenn dennoch Substitutionen eingesetzt werden so ist aus dem Grund der Lesbarkeit darauf zu achten das durchgehend nur eine Art der Substitution eingesetzt wird.

3.5.2 Empfehlung

MUST: Wird einer der beiden Mechanismen eingesetzt (`xsi:type` oder Ersetzungsgruppen), so muss nur dieser Mechanismus eingesetzt werden. Eine Mischung beider Stile würde eine Schema schwer zu verstehen und den Umgang damit unnötig kompliziert machen.

SHOULD NOT: Generell sollten Mechanismen der `xsi:type` Type Substitution vermieden werden, solange nicht gewichtige Gründe dafür sprechen. Diese Mechanismen erschweren die Verarbeitung und das Verständnis des Schemas und der durch das Schema definierten Instanzen.

SHOULD NOT: Generell sollten Mechanismen der Substitution Groups vermieden werden, solange nicht gewichtige Gründe dafür sprechen. Diese Mechanismen erschweren die Verarbeitung und das Verständnis des Schemas und der durch das Schema definierten Instanzen.

MUST: Werden Mechanismen der Type Substitution verwendet, so ist dies deutlich und angemessen zu dokumentieren, so dass Benutzer des Schemas deutlich über diesen Aspekt des Schemas informiert werden.

3.6 Root Elemente

3.6.1 Zusammenfassung

XML Schema Spezifikationen können so ausgelegt werden, dass mehrere Root-Elemente vorhanden sind. Als Root-Elemente werden solche Elemente bezeichnet, welche in einer XML Dokumenteninstanz immer als erstes Element genutzt werden müssen. Der Einsatz von mehreren Root-Elementen erlaubt, dass eine XML Schema Spezifikation mehrere XML Dokumenteninstanzen beschreiben kann.

Wir möchten diesen Mechanismus nicht verbieten, dennoch ist darauf zu achten, dass bei dem Einsatz von mehreren Root-Elementen dies auch klar dokumentiert ist. Weiterhin dass Benutzer dieser XML Schema Spezifikation darauf hingewiesen werden, wann und wo im Ablauf welches Root-Element verwendet werden muss.

3.6.2 Empfehlung

MAY: Schemas dürfen so designed werden, dass mehrere Elemente als Root Elemente vorkommen dürfen. Auf diese Weise kann ein Schema dazu dienen, Instanzen mit verschiedenen Root Elementen zu validieren.

SHOULD: Root Elemente in einem Schema sollten markiert werden. Da es dafür keinen durch XML Schema definierten Mechanismus gibt, muss dies in einem Kommentar geschehen, der die potentiellen Root Elemente kennzeichnet.

3.7 Verarbeitungseinheiten

3.7.1 Zusammenfassung

Mit Verarbeitungseinheiten bezeichnen wir die Strukturierung eines Schemas, respektive Teile eines Schemas welche strukturiert eine fachliche Datenstruktur abbilden.

Zum Beispiel kann ein Partner oder ein Konto eine Verarbeitungseinheit sein oder auch eine IBAN-Nummer.

Die Wahl wie eine XML Schema Spezifikation in solche Verarbeitungseinheiten gegliedert wird ist primär davon abhängig was fachlich modelliert werden soll. Zu der Modellierung der Fachlichkeit kann der Geschäftsmodell Ansatz verwendet werden.

3.7.2 Empfehlung

SHOULD: Die Verarbeitungseinheiten, die bei der Definition eines Schemas gewählt werden, sollten sich daran orientieren, in welcher Weise die Daten aus fachlicher Sicht strukturiert sind. Die Wahl der Verarbeitungseinheiten sollte das Ziel haben, die für die Verarbeitung relevanten Strukturen durch XML Markup zu kennzeichnen, so dass Applikationen soweit möglich keine über XML-Technologien hinausgehenden Methoden brauchen, um die fachlich relevanten Strukturen zu identifizieren und mit ihnen zu arbeiten.

3.8 Elemente oder Attribute?

3.8.1 Zusammenfassung

Obwohl der Einsatz von Attributen zu technisch kleineren XML Dokumentinstanzen führt, postuliert eCH den Einsatz von Elementen. Durch den Einsatz von Elementen lassen sich einfachere und vorallem wiedererkennbare Strukturen abbilden.

3.8.2 Empfehlung

SHOULD – Generell sollten Elemente zur Strukturierung verwendet werden, da diese im Bedarfsfall einfacher erweitert werden können, z.B. mit zusätzlichen Attributen, oder durch die Unterteilung in feiner gegliederte Strukturen.

MAY - Attribute sollten mit Vorsicht verwendet werden. Sie unterliegen deutlichen Einschränkungen (nicht wiederholbar, keine Möglichkeit der weiteren Strukturierung) und bilden daher bei späteren Änderungen einen Schemas u.U. eine Stelle, an der gewünschte Änderungen nicht mehr einfach durchführbar sind

3.9 Repräsentation leerer Werte

3.9.1 Zusammenfassung

Es existiert in der XML Schema Spezifikation eine integrierte Mechanik wie mit „nil“-Werten umgegangen werden kann. In der Praxis wird aber dieser xsi:null Mechanismus selten an-

gewendet. Aus diesem Grund verzichtet eCH auch auf die Darstellung von leeren Werten mittels des xsi:null Mechanismus.

3.9.2 Empfehlung

SHOULD - Die Darstellung von „null“-Werten sollte mit optionalen Elementen oder Attributen erfolgen, d.h. durch Nichterscheinen der jeweiligen Komponenten, oder durch leeren Inhalt der jeweiligen Komponenten.

SHOULD NOT - Der `xsi:null` Mechanismus sollte vermieden werden.

3.10 Wertelisten

3.10.1 Zusammenfassung

Normalerweise sind Wertelisten Stammdaten. Dies bedeutet, dass Inhalte der jeweiligen Listen der Pflege unterworfen sind.

Nun gibt es Inhalte welche über mehr Stabilität verfügen als andere. Als Beispiel kann eine Liste der Sprachkodierungen im Gegensatz zu einer Liste von Mitarbeitern in der Bundesverwaltung aufgeführt werden. Je nach Änderungspotential welche die Inhalte der Liste besitzen ist ein anderes Vorgehen zu wählen.

- Wenn abzusehen ist, dass die Liste selten bis nie ändert, so kann eine statische Liste mit einem „Enumeration Factet“ dargestellt werden.
- Handelt es sich aber um Listen welche Änderungen unterworfen sind, so sollte diese vorzugsweise als Referenzwerte oder Codetabellen extern gepflegt werden.

3.10.2 Empfehlung

SHOULD: Handelt es sich bei Wertelisten um statische Listen, deren Werte abschliessend bekannt und aller Voraussicht nach stabil sind, so sollten diese Werte im Schema mit einem Simple Type und Enumeration Facets aufgezählt werden. Um eine solche Werteliste besser zu verwalten und bessere Zugriffskontrollen durchführen zu können, ist es im Allgemeinen sinnvoll, sie in ein eigenständiges Schema-Dokument auszulagern

SHOULD: Handelt es sich bei Wertelisten um dynamische Listen, deren Werte nicht abschliessend bekannt sind oder sich verändern können, so sollten sie nur durch eine möglichst exakte lexikalische Einschränkung und den Verweis auf eine externe Liste definiert werden. Auf diese Weise können den Wertelisten neue Werte hinzugefügt werden, ohne dass sich am Schema etwas ändern muss

3.11 Sprachmarkierung von Inhalten

3.11.1 Zusammenfassung

Sprachmarkierungen erlauben Inhalte von Elementen eindeutig mit der Sprache des Inhaltes zu bezeichnen. Dies erlaubt mehrsprachige Inhalte eindeutig und normiert darzustellen. Die

Sprachmarkierung wird mittels des xml:lang Attributes einem Element zugeordnet. Die Sprachmarkierung ist normiert nach ISO-3166.

Die native Unterstützung des ISO Codes für die Sprachmarkierungen sollte angewendet werden. In [eCH-0050] wurde ein entsprechendes Schema definiert, das zur Wiederverwendung im Rahmen von eCH bereitgestellt wird.

3.11.2 Empfehlung

SHOULD - Sollen Inhalte mit Sprachmarkierungen versehen werden, so sollten diese Markierungen in einem `xml:lang` Attribut angebracht werden, da dieses im XML Standard selber definiert ist und eine allgemeine Konvention für Sprachmarkierungen darstellt.

3.12 Versionierung von XML Schemas

3.12.1 Zusammenfassung

Eine XML Schema Spezifikation wird über kurz oder lang eine neue Version und dementsprechend ein Release benötigen. Je nach Einsatzgebiet und Verbreitungsgrad der XML Schema Spezifikation, werden verschiedenen Versionen gleichzeitig oder nur die aktuellste Version in Betrieb sein.

Um diesen Sachverhalt eindeutig abbilden zu können, wird eine Versionierung über den Namespace angestrebt. Dies bedeutet, dass die Versionsangaben der XML Schema Spezifikation in den Namespace einkodiert sind. XML Instanzdokumente validieren gegen die jeweilige Version der XML Schema Spezifikation.

Es wird prinzipiell unterschieden zwischen „Major Version“ (Neue Version) und „Minor Version“ (Bugfix).

Durch diese Herangehensweise wird eine typisierte und stabile Schnittstelle erzwungen, da „Minor Releases“ nur erzeugt werden können, wenn die Änderungen des XML Schemas immer noch gegen vorangehende Versionen der XML Instanzdokumente validieren.

Diese Herangehensweise erlaubt dass sich Softwarehersteller auf stabile Spezifikationen (Schnittstellen) verlassen können und ihre Produkteversionen mit der Versionierung der XML Schema Spezifikation abstimmen können.

Es ist anzumerken das während der Entwicklungsphase diese Art der Versionierung nicht zwingend anzuwenden ist, da hier Änderungen schnell und iterativ eingeführt werden.

3.12.2 Empfehlung

SHOULD: Ändert sich ein Schema in einer Weise, dass bei der Validierung und Interpretation gemäss dem neuen Schema auch Instanzen nach dem alten Schema korrekt verarbeitet werden können, so sollte eine neue Minor Version erzeugt werden.

SHOULD: Ändert sich ein Schema in einer Weise, dass bei der Validierung und Interpretation gemäss dem neuen Schema Instanzen nach dem alten Schema nicht mehr korrekt verarbeitet werden können, so sollte eine neue Major Version erzeugt werden.

SHOULD NOT: Sind vormals erlaubte Werte oder Strukturen in einer neuen Version verboten, so sollten sie nicht in der Schema-Definition verboten werden, sondern auf der Anwendungsebene als deprecated markiert werden, so dass alte Instanzen weiterhin verarbeitet werden können.

MAY: Die obigen Empfehlungen sind nur für produktiv eingesetzte und entwickelte Schemas anwendbar. Für die Entwicklungsphase vor dem produktiven Einsatz und für Schemas ausserhalb des produktiven Einsatzes ist ein einfacheres Vorgehen möglich.

MUST: Um die Markierung der Minor Version eines Schemas und dessen Instanzen zu ermöglichen (die Major Version wird über den Namespace Name markiert), wird das `version` Attribut des Schemas verwendet, das nur die Minor Version enthält. Die Minor Version in Instanzen wird durch eine Markierung vorgenommen, die im Schema selbst vorgesehen sein muss, am besten aber durch die Verwendung des in [eCH-0050] vorgesehenen Attributs erfolgen sollte.

3.13 Strukturierung von Schema-Dokumenten und Schemas

3.13.1 Zusammenfassung

XML Schema kann einerseits physikalisch strukturiert werden als auch logisch.

- Die physikalische Strukturierung trennt das Schema in einzelne Dateien auf, welche über „Include“ Statements wieder eingebunden werden können.
- Die logische Strukturierung bezieht sich auf die Modellebene, also wie wird mit Typen, Elementen und Attributen umgegangen.

Kleinere Schema Spezifikationen können sicherlich in einer physikalischen Schema Datei erstellt werden. Sobald aber grösserere Projekte zu realisieren sind, kann es von Vorteil sein, die physikalische Struktur aufzuteilen. Dies kann nach diversen Kriterien geschehen

- Verschiedene Teams spezifizieren Teile des XML Schemas unabhängig.
- Physikalische Trennung nach Prozessen und Schnittstellen

Im Generellen lässt sich sagen, dass ab einer bestimmten Grösse der Schema Spezifikation, physikalische Trennung nötig wird, um eine strukturierte Arbeitsweise zu erlauben.

Weiterhin sollte soweit wie möglich bereits existierende Arbeiten und Standards aufgegriffen werden, wie z. Bsp. die Einbindung des Meldewesenstandards. Die Voraussetzung ist natürlich das die jeweilige XML Schema Spezifikation nennenswerte Synergien besitzt.

3.13.2 Empfehlung

SHOULD: Aus Gründen der Modularisierung sollten Schema-Dokumente, die über eine gewisse Grösse hinausgehen, unter Verwendung von `xs:include` strukturiert werden. Dies erlaubt übersichtlichere Schemas und zudem die Wiederverwendung von Teilen des Schemas .

SHOULD NOT: Der `xs:redefine` Mechanismus von XML Schema sollte vermieden werden, da er zu komplizierten Abhängigkeiten zwischen verschiedenen Schema-Dokumenten führt.

SHOULD NOT: Chamäleon Schemas (d.h. Schema-Dokumente ohne `targetNamespace`, die daher den Namespace des Schema-Dokumenten annehmen, in das sie eingebunden werden), sollten vermieden werden.

MAY: Arbeiten verschiedene Teile eines Teams an der Entwicklung oder Weiterentwicklung eines Schemas, so kann dies ebenfalls ein Grund für die Modularisierung eines Schemas sein. In diesem Fall wird sich die Modularisierung an der inhaltlichen Zuteilung der verschiedenen Teile des Schemas an verschiedene Teile des Teams orientieren.

SHOULD: Soweit möglich, sollten in einem neuen Schema existierende Lösungen für Teilaspekte des Schemas durch das Importieren der betreffenden Schemas wiederverwendet werden. Dies erleichtert die Entwicklung von Schemas, und ist auch für die Anwender eines Schemas eine Erleichterung, weil diese bereits bekannten Schemas benutzen und auf diese Weise bestehendes Know-how und bestehenden Code wiederverwenden können.

3.14 Identifikation und Referenzen und deren Implementierung

3.14.1 Zusammenfassung

Referentielle Beziehungen werden eingesetzt, wenn eine rein hierarchische Darstellung der Daten zuviel Speicherbedarf verbrauchen würde, respektive eine hierarchische Darstellung aller Daten nicht möglich ist. Zum Beispiel ist es nicht praktisch und auch nicht bedienerfreundlich alle Organisationseinheiten und Mitarbeiter aus dem Staatkalender auf einmal zu laden und darzustellen. Es wird folglich mit Referenzen gearbeitet. Die Mitarbeiterdaten zu einer Organisationseinheit werden erst geladen und angezeigt wenn sie benötigt werden. Wenn dieses Prinzip weiterverfolgt wird gilt es natürlich auch für die XML Schema Spezifikation. Stellen Sie sich vor wie Sie eine XML Schema Spezifikation für die Daten im Staatskalender aussehen sollte. Eine Organisationseinheit besitzt eine Liste von Referenzen auf Mitarbeiter.

Es ist anzumerken das der Umgang mit Referenzen auch noch weitere Stolperfallen aufweist. Wenn wir nun von einer Datenübertragung ausgehen bei welcher Mitarbeiter aus dem Staatskalender übertragen werden, welche Referenzen auf ihre Organisationseinheit besitzen (zu welcher Organisation gehört Mitarbeiter A), so kann es geschehen das die Empfängerseite die Organsiationseinheit in welcher ein bestimmter Mitarbeiter ist noch gar nicht erhalten hat und dementsprechend kann die Referenz nicht aufgelöst werden. Nun was soll die Empfängerseite nun tun. Hier bestehen drei Möglichkeiten.

- Zum einen sicherstellen das alles in einem „Rutsch“ ankommt, somit sind wir wieder beim hierarchischen Modell
- Geordnete sequentielle Übertragung, was wiederum mehr Aufwand bei der Implementierung mit sich bringt.
- Einfach ignorieren, dies funktioniert aber nur in den seltensten Fällen.

Wie können nun solche Sachverhalte in XML Schema abgebildet werden. Eine Möglichkeit sind die sogenannten Identity Contraints, welche genau solche Referenzen einschränken und eine Eindeutigkeit fordern. Generell handle es sich bei den Identity Constraint um Einschränkungen und keine volle Modellierungsmethodik um Referenzen abzubilden. Die Konsumenten der Spezifikation müssen ihre eigene Logik für den Umgang mit diesen Referenzen implementieren.

Beim Einsatz von Identity Constrainsts ist sichergestellt das wenn eine Referenz angewendet wird, im XML Instanz Dokument auch ein Wert dafür vorhanden sein muss. Die Menge der übertragenen Information ist dieselbe wie im hierarchischen Modell.

3.14.2 Empfehlung

MAY: Ist es aus Sicht von Implementierungsüberlegungen angebracht, auf hierarchische Darstellungen zu verzichten, so ist es erlaubt, auf die hierarchische Darstellung von Daten zu verzichten. Dies sollte jedoch nur mit Vorsicht und Bedacht geschehen, weil sich die durch die Implementierungsüberlegungen gegebenen Randbedingungen ändern können (z.B. bei der Umstellung auf XML-Datenbanken), und das Schema dann nicht mehr zur geänderten Umgebung passt.

SHOULD: Werden Referenzen verwendet, so sollten die Randbedingungen der Referenzen so gut wie möglich mit Hilfe von Identity Constraints beschrieben werden. Weitergehende Randbedingungen sollten dokumentiert werden.

SHOULD: Sind Schemas hauptsächlich auf Endbenutzer ausgerichtet, z.B. im Fall von XML-Dokumenten, die von Menschen verwendet (betrachtet oder editiert) werden, z.B. Konfigurationsdateien oder Web-orientierte Dokumente, so sollte die Modellierung an den entsprechenden Stellen eher hierarchisch sein. Hierarchische Strukturen sind für Menschen einfacher zu verstehen als durch viele Referenzen miteinander vernetzte Strukturen.

SHOULD: Sind Schemas hauptsächlich für die maschinelle Weiterverarbeitung bestimmt, so kann das Modell eher flacher definiert werden als für den hauptsächlich auf Endbenutzer ausgerichteten Fall. Allerdings sollte an den Stellen, wo es eine fachlich begründete, dem Basismodell innewohnende hierarchische Struktur gibt, auch eine solche Struktur im XML definiert werden.

SHOULD: Soll die Eindeutigkeit, Existenz, oder Referenzierung von Namen garantiert werden, so sollten XML Schema Identity Constraints benutzt werden, um diese Constraints zu definieren.

SHOULD NOT: Der DTD Mechanismus der `ID/IDREF` Attribute sollte nicht verwendet werden, da er starken Einschränkungen unterliegt und es zudem nicht erlaubt, die Constraints so genau zu definieren, wie es mit den Identity Constraints möglich ist.

MAY: Sind durch die Applikation Anforderungen gegeben, die sich durch XML Schemas Identity Constraints nicht implementieren lassen (z.B. Eindeutigkeit oder Referenzierung über mehrere Dokumente hinweg), so können diese durch externe Identifikation und externe Constraints definiert werden. Für die durch diese externen Definitionen erfassten Strukturen sollten dann eigene Typen verwendet werden, und im Schema sollte deutlich dokumentiert sein, dass die Werte dieser Typen Constraints unterworfen sind, die ausserhalb des Schemas definiert sind.

SHOULD: Sollen Identity Constraints in zur Wiederverwendung bestimmten Typen vorkommen, so kann dies am besten dadurch dokumentiert werden, dass ein Beispielelement dieses Typs definiert wird, das die Identity Constraints enthält. Im Typ kann dann durch einen Kommentar darauf verwiesen werden, dass bei einer Wiederverwendung des Typs die Identity Constraints aus dem Beispielelement kopiert werden müssen.

3.15 Offene XML Schemas

3.15.1 Zusammenfassung

Unter offenen XML Schemas werden Spezifikation verstanden, welche tolerant gegenüber Modifikation sind. Die Funktionsweise dieser Schemas beruht auf dem simplen Prinzip der Platzhalter, auch Wildcards genannt.

Im Prinzip handelt es sich um die Einbindung eines Elements welches vom Typ „any“, welches als Wildcard bezeichnet ist. Dies kann Vor- und Nachteile besitzen.

Der Vorteil das man generische Schemas spezifizieren kann, und die Details der Implementation überlässt, ist auch gerade der Nachteil: Werden die Schemas zu generisch spezifiziert, führt dies dazu dass alles Implementationsspezifisch gelöst wird.

Generell ausgedrückt ist der Einsatz von offenen Schemas erwünscht, aber es sollte mit Vorsicht gehandhabt werden, welche Teile eines Schemas generisch gestaltet werden und welche nicht.

3.15.2 Empfehlung

MAY: Ist ein offenes Schema erwünscht, so dass in Instanzen des Schemas Inhalte erscheinen dürfen, die nicht detailliert im Schema definiert sind, so kann die über Wildcards für Elemente und/oder Attribute erreicht werden. Diese Wildcards können einerseits dazu dienen, Offenheit gegenüber nicht bekannten Inhalten zu bewahren, andererseits aber auch dazu, Offenheit gegenüber vorhergesehenen zukünftigen Erweiterungen zu erreichen.

4 Haftungsausschluss/Hinweise auf Rechte Dritter

eCH-Standards, welche der Verein **eCH** dem Benutzer zur unentgeltlichen Nutzung zur Verfügung stellt, oder welche **eCH** referenziert, haben nur den Status von Empfehlungen. Der Verein **eCH** haftet in keinem Fall für Entscheidungen oder Massnahmen, welche der Benutzer auf Grund dieser Dokumente trifft und / oder ergreift. Der Benutzer ist verpflichtet, die Dokumente vor deren Nutzung selbst zu überprüfen und sich gegebenenfalls beraten zu lassen. **eCH**-Standards können und sollen die technische, organisatorische oder juristische Beratung im konkreten Einzelfall nicht ersetzen.

In **eCH**-Standards referenzierte Dokumente, Verfahren, Methoden, Produkte und Standards sind unter Umständen markenrechtlich, urheberrechtlich oder patentrechtlich geschützt. Es liegt in der ausschliesslichen Verantwortlichkeit des Benutzers, sich die allenfalls erforderlichen Rechte bei den jeweils berechtigten Personen und/oder Organisationen zu beschaffen.

Obwohl der Verein **eCH** all seine Sorgfalt darauf verwendet, die **eCH**-Standards sorgfältig auszuarbeiten, kann keine Zusicherung oder Garantie auf Aktualität, Vollständigkeit, Richtigkeit bzw. Fehlerfreiheit der zur Verfügung gestellten Informationen und Dokumente gegeben werden. Der Inhalt von **eCH**-Standards kann jederzeit und ohne Ankündigung geändert werden.

Jede Haftung für Schäden, welche dem Benutzer aus dem Gebrauch der **eCH**-Standards entstehen ist, soweit gesetzlich zulässig, wegbedungen.

5 Urheberrechte

Wer **eCH**-Standards erarbeitet, behält das geistige Eigentum an diesen. Allerdings verpflichtet sich der Erarbeitende sein betreffendes geistiges Eigentum oder seine Rechte an geistigem Eigentum anderer, sofern möglich, den jeweiligen Fachgruppen und dem Verein **eCH** kostenlos zur uneingeschränkten Nutzung und Weiterentwicklung im Rahmen des Vereinszweckes zur Verfügung zu stellen.

Die von den Fachgruppen erarbeiteten Standards können unter Nennung der jeweiligen Urheber von **eCH** unentgeltlich und uneingeschränkt genutzt, weiterverbreitet und weiterentwickelt werden.

eCH-Standards sind vollständig dokumentiert und frei von lizenz- und/oder patentrechtlichen Einschränkungen. Die dazugehörige Dokumentation kann unentgeltlich bezogen werden.

Diese Bestimmungen gelten ausschliesslich für die von **eCH** erarbeiteten Standards, nicht jedoch für Standards oder Produkte Dritter, auf welche in den **eCH**-Standards Bezug genommen wird. Die Standards enthalten die entsprechenden Hinweise auf die Rechte Dritter.

Anhang A – Referenzen & Bibliographie

[eCH-0018] Erik Wilde, Hanspeter Salvisberg, Alexander Pina, *XML Best Practices*, eCH, Berne, Switzerland, eCH-0018, August 2005

[eCH-0033] Erik Wilde, *Beschreibung von XML Namespaces*, eCH, Berne, Switzerland, eCH-0033, 2006.

[eCH-0050] Erik Wilde, *Hilfskomponenten zur Konstruktion von XML Schemas*, eCH, Berne, Switzerland, eCH-0050, 2006.

[ISO19757-2] International Organization for Standardization, *Information Technology — Document Schema Definition Languages (DSDL) — Part 2: Grammar-based Validation — RELAX NG*, ISO/IEC 19757-2, November 2003.

[ISO19757-3] International Organization for Standardization, *Information Technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based Validation — Schematron*, ISO/IEC 19757-3, June 2005.

[RFC2119] Scott O. Bradner, *Key Words for use in RFCs to Indicate Requirement Levels*, Internet RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt

[RFC3066] Harald Tveit Alvestrand, *Tags for the Identification of Languages*, Internet RFC 3066, January 2001. http://www.ietf.org/rfc/rfc3066.txt

[RFC3986] Tim Berners-Lee, Roy Fielding, Larry Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, Internet RFC 3986, January 2005. http://www.ietf.org/rfc/rfc3986.txt

[xml10third] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, *Extensible Markup Language (XML) 1.0 (Third Edition)*, World Wide Web Consortium, Recommendation REC-xml-20040204, February 2004. http://www.w3.org/TR/2004/REC-xml-20040204

[xml11] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan, *Extensible Markup Language (XML) 1.1*, World Wide Web Consortium, Recommendation REC-xml11-20040204, February 2004. http://www.w3.org/TR/2004/REC-xml11-20040204

[xml11schema10] Henry S. Thompson, *Processing XML 1.1 Documents with XML Schema 1.0 Processors*, World Wide Web Consortium, Note NOTE-xml11schema10-20050511, May 2005. http://www.w3.org/TR/2005/NOTE-xml11schema10-20050511

[xmlns] Tim Bray, Dave Hollander, Andrew Layman, *Namespaces in XML*, World Wide Web Consortium, Recommendation REC-xml-names-19990114, January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114

[xmlns11] Tim Bray, Dave Hollander, Andrew Layman, Richard Tobin, *Namespaces in XML 1.1*, World Wide Web Consortium, Recommendation REC-xml-names11-20040204, February 2004. http://www.w3.org/TR/2004/REC-xml-names11-20040204

[xmlschema1sec] Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, *XML Schema Part 1: Structures Second Edition*, World Wide Web Consortium, Recommendation REC-xmlschema-1-20041028, October 2004. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028

[xmlschema2sec] Paul V. Biron, Ashok Malhotra, *XML Schema Part 2: Datatypes Second Edition*, World Wide Web Consortium, Recommendation REC-xmlschema-2-20041028, October 2004. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028

Anhang B – Mitarbeit & Überprüfung

Remo Dick, ISC EJPD

Claude Eisenhut, Eisenhut Informatik

Urs Gähler, VRSG

Jürg Hotz, Kanton Thurgau

Willy Müller, ISB

Hubert Münst, Data Factory

Alexander Pina, Unisys (Schweiz) AG

Hanspeter Salvisberg, Unisys (Schweiz) AG

Hans Ulrich Wiedmer, KOGIS - LT

Erik Wilde, ETH Zürich

Anhang C – Abkürzungen & Glossar

Ein kommentiertes, mit weiterführenden Links versehenes und wesentlich ausführlicheres Abkürzungsverzeichnis und Glossar findet sich auf dem Web unter http://dret.net/glossary/.

DOM	Document Object Model
DSDL	Document Schema Definition Languages
DTD	Document Type Definition
RDBMS	Relational Database Management System
URI	Universal Resource Identifier
XML	Extensible Markup Language
XSD	XML Schema Definition Language, häufig verwendete (aber nicht offizielle) Abkürzung für XML Schema