eCH-0036: Dokumentation für den XML-orientierten Datenaustausch

Name	Dokumentation für den XML-orientierten Datenaustausch
Standard-Nummer	eCH-0036
Kategorie	Best Practice
Reifegrad	Experimentell
Version	1.00
Status	Aufgehoben
Beschluss am	2018-09-05
Ausgabedatum	2007-04-07
Ersetzt Standard	--
Sprachen	Deutsch
Autoren	Fachgruppe XML Erik Wilde, ETH Zürich (http://dret.net/netdret/)
Herausgeber / Vertrieb	Verein eCH, Amthausgasse 18, 3011 Bern T 031 560 00 20, F 031 560 00 25

Zusammenfassung

Im vorliegenden Dokument wird beschrieben, welche Dokumentation für XML Schemas zu erstellen ist, damit die nötigen Grundlagen für die Implementierung einer Schnittstelle für den XML-basierten Datenaustausch vorhanden sind. Ausgangspunkt ist ein Datenmodell des Ausschnitts der Realität, über welchen Informationen ausgetauscht werden sollen. Davon abgeleitet werden für die jeweils interessierenden Transaktionen Datenmodelle für den Datenaustausch (Austauschmodelle). Ein Austauschmodell wiederum dient als Grundlage für eines oder auch verschiedene Schemas (Austauschschemas). Nur wenn die Modelle wohl definiert und dokumentiert sind, und wenn die Beziehungen zwischen den Modellen (System- und Austauschmodelle) und den Schemas (Austauschmodell und -schemas) wohl definiert und dokumentiert sind, können unabhängige Implementierer die Schnittstelle korrekt umsetzen.

Inhaltsverzeichnis

1 Status des Dokuments

Aufgehoben: Das Dokument wurde von eCH zurückgezogen. Er 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

Die *Extensible Markup Language (XML)* [XML10] ist für sehr viele Szenarien, in denen Daten ausgetauscht werden sollen, das heute gebräuchliche Austauschformat. XML ist eine zeichenbasierte Syntax, die es erlaubt, baumstrukturierte Daten auszutauschen. In seiner einfachsten Variante sind diese Daten nicht einmal typisiert. Seit der Einführung von *XML Schema* [XSD1,XSD2] ist es zwar möglich, die den Daten zugrundeliegenden Datentypen zu in einem Schema zu definieren, dies ändert aber nichts an der Tatsache, dass es sich bei XML um eine reine Syntax handelt. Dies bedeutet, dass das zugrundeliegende Modell (die anwendungsspezifische Semantik) aus einem XML Dokument oder auch aus einem XML Schema nicht ersichtlich wird. Viele der aus Anwendungssicht wichtigen Informationen sind nicht mit Hilfe von XML-Technologien erfassbar und damit auch nicht austauschbar. Dies bedeutet, dass es über die syntaktische Festlegung eines Austauschformates hinaus weitergehende Übereinstimmungen zwischen Teilnehmer in einem Szenario mit XML-basierten Datenaustausch geben muss. Nur dann kann sichergestellt werden, dass über die gemeinsam benutzt Syntax hinaus auch ein gemeinsames Verständnis der ausgetauschten Daten besteht, und dieses gemeinsame Verständnis ist notwendig für einen bei allen Beteiligten korrekten Umgang mit Daten.

2.1 Überblick

Nach heutigem Stand der XML-Technologien gibt es kein allgemein akzeptiertes und anwendbares Verfahren, wie sich ein Modell für den XML-orientierten Datenaustausch auf einer Abstraktionsebene beschreiben lässt, die zum Verständnis der Daten und zur Beschreibung des Modells besser geeignet ist als XML Schema. Es gibt erste Ansätze zu einem strukturierten und systematischen Vorgehen [Glushko2005], jedoch wird es noch einige Zeit dauern, bis es eine etablierte Vorgehensweise geben wird. Bis dahin ist es notwendig, die Dokumentation von Datenmodellen und ihren Zusammenhängen mit XML-Formaten auf eine eher informelle Art vorzunehmen. Das vorliegende Dokument gibt Leitlinien dazu, was zu beachten ist, wenn diese Dokumentation erstellt werden soll.

2.2 Anwendungsgebiet

Jeglicher XML-basierte Datenaustausch kann im Rahmen der bestehenden XML-Technologien nur auf einer rein syntaktischen beschrieben werden. Um eine XML-basierte Schnittstelle vollständig zu beschreiben, muss jedoch auch die Semantik der XML-Inhalte beschrieben sein. Dieses Problem ist abhängig von der Komplexität der auszutauschenden Daten. Für sehr einfache Schnittstellen kann im Extremfall ein „selbstdokumentierendes“ Schema ausreichend sein, die Bedeutung der XML-Strukturen zu beschreiben. In allen anderen Fällen aber wird ein Schema nicht ausreichend sein, um zu verstehen, welche Bedeutung und auch welche Abhängigkeiten und Randbedingungen die beschriebenen Daten haben.

Das vorliegende Dokument beschreibt, welche Aspekte zu beachten sind, wenn die Beschreibung von Daten für den XML-basierten Datenaustausch so erfolgen soll, dass die Dokumentation alleine ausreichend ist, um die Daten verstehen und damit korrekt verarbeiten zu können. Die Annahme dabei ist, dass der Konsument der Dokumentation keine speziellen

Kenntnisse des Anwendungsumfelds hat, aus dem heraus die XML-Daten produziert werden. Trotzdem sollte es ihm möglich sein, mit Hilfe der Dokumentation alle relevanten Informationen zu erhalten, die er benötigt, und die XML-Daten zu verstehen und zu verarbeiten.

Eine solche Dokumentation kann auf diese Weise dazu dienen, die Abhängigkeiten zwischen dem Anbieter und dem Verwender einer XML-orientierten Schnittstelle zu reduzieren, da alle notwendige Information in der Dokumentation verfügbar ist.

2.3 Vorteile

Eine der Grundideen für die Verwendung XML-basierter Schnittstellen ist die Idee lose gekoppelter Systeme, die möglichst wenig über die absoluten Notwendigkeiten hinausgehende Abhängigkeiten haben. XML ist auf der technischen Ebene eine gute Lösung, um diese lose Koppelung zu erreichen, da die Verwendung von XML-basierten Daten in nahezu allen Entwicklungs- und Produktionsumgebungen unterstützt wird. XML selber ist jedoch nur eine Syntax zum Austausch strukturierter Daten, d.h. die Definition und der Austausch von XML-basierten Daten stellt nur sicher, dass ein gemeinsames Verständnis darüber existiert, welche Datenstrukturen erlaubt sind und welche nicht.

Um über diese einfache syntaktische Kompatibilität hinaus auch eine semantische Kompatibilität zu erreichen, muss Dokumentation existieren, die ein gemeinsames Verständnis der Daten auf der Anwendungsebene ermöglicht. Nur wenn eine solche Dokumentation existiert, kann das Ziel lose gekoppelter Systeme erreicht werden, andernfalls ist zwar rein syntaktisch eine einfache Kompatibilität gegeben, aber durch die fehlende Dokumentation der Bedeutung auf der Anwendungsebene entsteht dann eine Abhängigkeit zwischen dem Anbieter und dem Nutzer einer XML-basierten Schnittstelle, die weit stärker ist als die an sich angestrebte lose Koppelung.

Das vorliegende Dokument beschreibt Bereiche, die bei der Dokumentation für den XML-orientierten Datenaustausch beachtet werden sollten. Es wird häufig der Fall sein, dass gewisse Bereiche für konkrete Szenarien nicht beachtet werden müssen und ignoriert werden können. Die Liste an wichtigen Bereichen kann jedoch als eine Art Checklist benutzt werden, um zu überprüfen, ob bei der Dokumentation kein Aspekt vergessen wurde, der für komplette Dokumentation wichtig wäre.

2.4 Schwerpunkte

Wie bereits erwähnt, gibt es noch keine etablierte und vollständige Methodik, wie Dokumentation für den XML-orientierten Datenaustausch erstellt werden sollte. Aus diesem Grund werden im vorliegenden Dokument auch keine Methoden oder Technologien vorgeschrieben, sondern es werden Bereiche identifiziert, in denen Dokumentation vorliegen sollte. In welcher Form diese Dokumentation erstellt wird (informell, halb-formal oder formal) ist jeweils individuell zu entscheiden und hängt zu einem grossen Teil auch davon ab, in welcher Form bereits Dokumentation besteht zu dem Datenmodell, das dokumentiert werden soll. Aus diesem Grund ist der Schwerpunkt auf die Bereiche gelegt, und nicht auf die konkreten Beschreibungsarten, die für diese Bereiche benutzt werden sollen.

3 Ziel/Abgrenzung des Dokuments

Komplexe IT-Systeme basieren immer auf einem Modell, das erstellt wurde, um diejenigen Aspekte der Realität zu repräsentieren, mit denen im Rahmen des IT-Systems umgegangen werden soll. Im Allgemeinen dient dieses Modell Zwecken der Entwicklung und Dokumentation des Systems, so dass es als Basis für Erzeugung, Wartung und Anpassungen des Systems benutzt werden kann. Im folgenden soll dieses Modell als das *Systemmodell* bezeichnet werden, weil es selten mit dem Fokus auf Offenheit des Systems entwickelt wird, sondern meistens mit dem Fokus auf dem für die Entwicklung des IT-Systems gewählten Ausschnitts der Realität, der mit dem Modell abgedeckt werden soll.

Sobald ein IT-System aber mit einem anderen, getrennt entwickelten IT-System Daten austauschen soll, müssen die beiden Systemmodelle der beiden Systeme (die mit grosser Wahrscheinlichkeit nach nicht die gleichen sind, sondern getrennt entwickelt wurden) aufeinander abgebildet werden. Meist geschieht das heute nicht in Form einer direkten Abbildung, sondern durch die Definition eines *Austauschmodells*, und als syntaktische Basis hat sich dort XML [XML10] etabliert. Der Vorteil eines neutralen Austauschmodells ist, dass beim Anbinden neuer IT-Systeme nur die Abbildung des Systemmodells auf das Austauschmodell vorgenommen werden muss, und nicht die individuelle Anbindung an alle bereits am Datenaustausch teilnehmenden IT-Systeme.

Das Ziel des vorliegenden Dokuments ist, den Zusammenhang zwischen einem Systemmodell, einem Austauschmodell und einem XML-basierten Austauschformat so zu beschrieben, dass eindeutig und einfach nachvollzogen werden kann, wie die beiden Modelle und das XML-Format zueinander in Beziehung stehen.

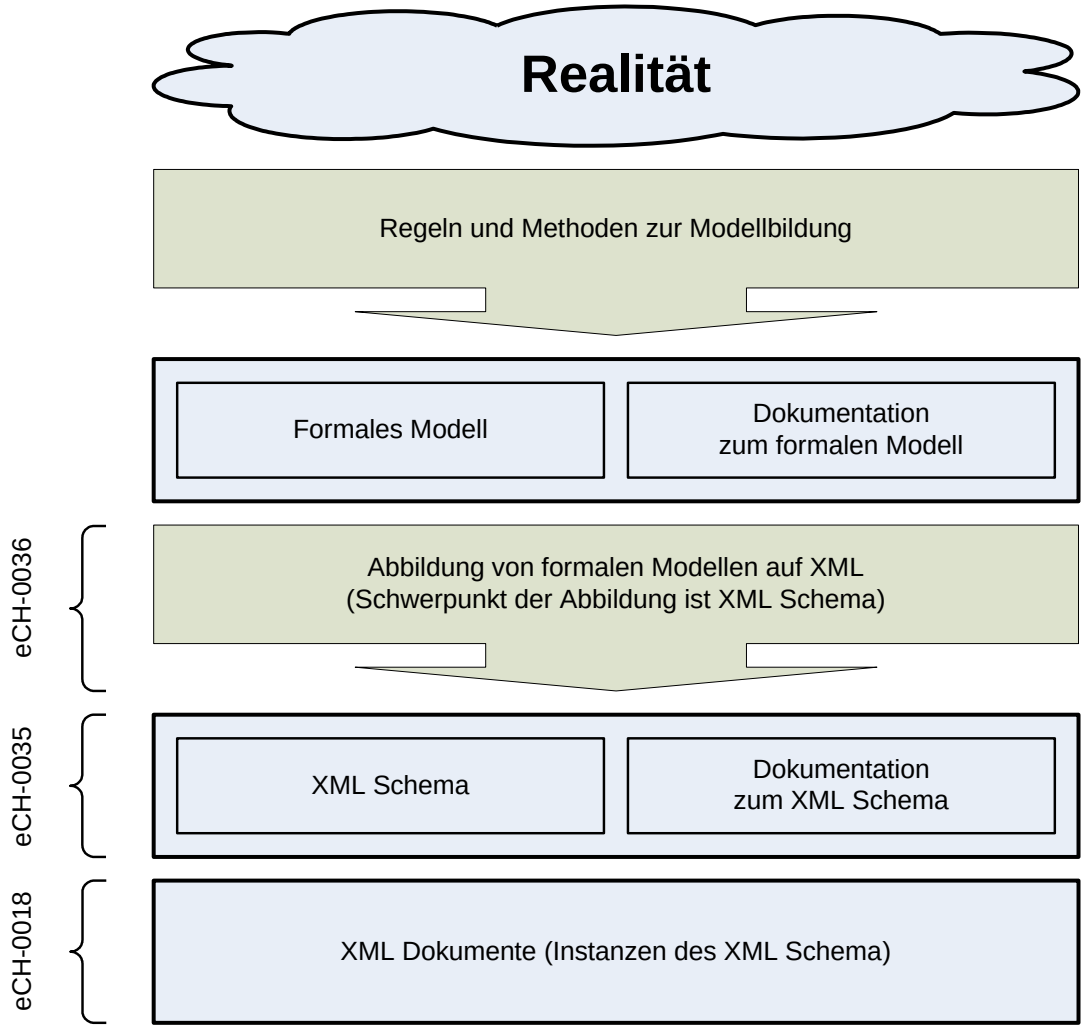
Für eine möglichst einfach zu verstehende Beschreibung eines Modells ist eine formale Beschreibung häufig sehr hilfreich. In vielen Fällen ist eine solche formale Beschreibung aber nicht ausreichend, um die Semantik des Modells vollständig zu erfassen. In diesen Fällen sollte es ergänzend zu der formalen Beschreibung eine semantische Beschreibung geben, die es ermöglicht, die nicht in der formalen Beschreibung erfassten Bedeutungen des Modells zu verstehen. Diese ergänzende Beschreibung wird häufig in natürlicher Sprache verfasst sein.

3.1 Empfehlungen

- **MUST**: Ausser für sehr einfache Modelle (wie z.B. die in [eCH-0050] beschriebenen Hilfskomponenten) muss immer ein formales Modell existieren, das den Ausschnitt der Realität beschreibt, der dann mit einer Abbildung auf ein XML Schema in XML repräsentiert werden kann.
- **SHOULD**: Der Ausschnitt der Realität beschreibt, der dann mit einer Abbildung auf ein XML Schema in XML repräsentiert werden kann, sollte in natürlicher Sprache beschrieben werden.
- **SHOULD**: Als Sprache für die formale Modellierung sollte UML [UML20] verwendet werden, diese Sprache hat sich im Software Engineering etabliert und wird von vielen Tools unterstützt.

4 Beschreibung des zugrunde liegenden Modells

Der generelle Prozess, wie man von der in einem IT-System zu repräsentierenden Realität auf konkrete Datenrepräsentationen kommt, ist in der folgenden Abbildung dargestellt. Dabei wird zunächst davon ausgegangen, dass es sich bei diesem Abbildungsprozess um nur eine Abbildung handelt, die definiert werden muss.



Im Fall von XML-basierten Datenaustausch ist die Problematik allerdings meist komplexer, weil es nicht nur um ein Datenmodell geht, sondern um das System- und das Austauschmodell, die oft nicht identisch sind, aber natürlich in wohl definierter Beziehung zueinander stehen sollten, um beim Austausch von Daten keine Interpretationsfehler zu begehen. Mehr Informationen dazu finden sich in Abschnitt 4.1.

- **SHOULD**: Historisierung ist ein komplexes Problem und führt zwangsläufig zu komplexen Datenmodellen. Aus diesem Grund sollte Historisierung nur dann modelliert werden, wenn es das Anwendungsumfeld erfordert. In allen anderen Fällen sollte ein geschichtsloses (oftmals mit einem Zeitstempel versehenes) Modell vorgezogen werden.

4.1 System- und Austauschmodelle

Wenn zwei oder mehr Partner untereinander XML-Daten austauschen und fehlerfrei weiterverarbeiten möchten, benötigen sie für den betroffenen Ausschnitt der Realität ein gemeinsames Verständnis. Sie müssen sich auf ein gemeinsames Modell einigen: das Austauschmodell. Das Austauschmodell enthält meist nur einen Ausschnitt der Systemmodelle der beteiligten Partner.

In vielen Anwendungsfällen wird es bei der Entwicklung des Systemmodells bereits ähnliche Modelle geben, die in leicht anderen Umgebungen für leicht andere Anwendungsfälle definiert wurden. Um diese verwandten Modelle zu finden, sollte zu Beginn der Modellierungsphase ein gewisser Aufwand betrieben werden. Oftmals wird dieser Aufwand mehr als kompensiert durch den Gewinn, der sich durch das Auffinden ähnlicher Modelle ergibt, unabhängig davon, ob sich diese Modelle direkt wieder verwenden lassen, oder sich durch ihr Studium zumindest neue Einsichten in die Problematik ergeben und neue Lösungsvarianten aufgezeigt werden.

In jedem Fall (Wiederverwendung bestehender Modellteile oder komplette Neudefinition eines eigenen Systemmodells) stellt sich im Rahmen des vorliegenden Dokuments dann die Aufgabe, das für das Software-Engineering erstellte Modell abzubilden auf ein XML-orientiertes Modell für den Datenaustausch, also das Austauschmodell.

Es besteht oftmals also die Notwendigkeit, aus dem Systemmodell ein Austauschmodell zu erzeugen, wobei die Formalisierung des Systemmodells unbestimmt ist (oftmals durch UML definiert), und das Austauschmodell auf der Basis von XML Schema definiert wird. Auch bei der Definition des Austauschmodells stellt sich dann die Frage: Kann ein bestehendes XML-Modell verwendet werden oder zumindest Teile von bestehenden XML-Modellen, oder muss ein komplett neues XML-Modell erstellt werden, um das Austauschmodell zu erzeugen?

Dies kann oftmals am besten durch das Nachvollziehen von Use Cases festgestellt werden. Können die für das zu erstellende Modell definierten Use Cases durch die ähnlichen, bestehenden Modelle erfüllt werden, so ist keine (komplette) Neudefinition notwendig und das bestehende Modell sollte übernommen oder zumindest als Basis verwendet werden. Bestehen nur kleine Differenzen, so können diese oftmals durch das Anpassen des bestehenden Modells beseitigt werden, und es muss ebenfalls keine komplette Neudefinition vorgenommen werden.

Ist eine solche Wiederverwendung bestehender Modelle (die im Bereich von XML oftmals als "Micro-Formats" bezeichnet werden, weil sie kleine Ausschnitte der Realität in einem festgelegten Format beschreiben) möglich, so sollte sie klar dokumentiert werden, so dass eventuell mögliche andere Aspekte der Wiederverwendung aus dem bestehenden Modell ebenfalls ausgenutzt werden können.

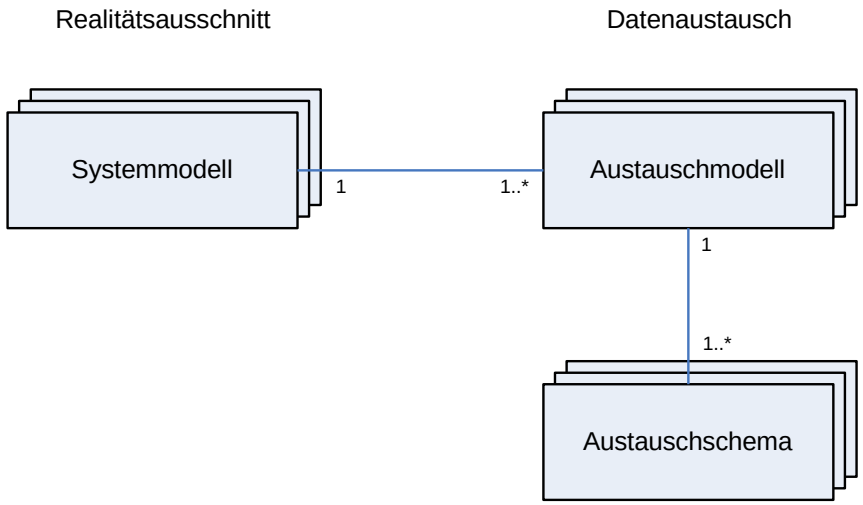
Häufig hat das Systemmodell den Charakter eines Bestandesmodells, ein Austauschmodell den eines Ereignismodells. Dies ist jedoch nicht immer und oft nicht vollständig der Fall.

Meist gibt es zu einem Systemmodell mehrere Austauschmodelle, z.B. wenn im selben Kontext unterschiedliche Formen von Transaktionen abgebildet werden müssen. Es besteht also

zwischen System- und Austauschmodell eine 1:n Beziehung. Die Austauschmodelle können dabei jeweils andere Ausschnitte des Systemmodells betreffen.

Manchmal wird das Austauschmodell auf eine einzige Art und Weise implementiert. In diesem Fall genügt für die Umsetzung ein einziges Schema. Gelegentlich muss das Austauschmodell jedoch in unterschiedlichen Ausprägungen implementiert werden, z.B. weil situationsabhängig unterschiedliche Formate benötigt werden oder die Konsistenzregeln unterschiedlich gehandhabt werden müssen (vgl. dazu Abschnitt 5.2).

Die folgende Darstellung zeigt den soeben beschriebenen Zusammenhang zwischen System- und Austauschmodellen, und zwischen Austauschmodellen und -schemas.



- **MUST**: Bei der Abbildung des System- auf das Austauschmodell muss klar dokumentiert werden, was modelliert wird, und was nicht in die Modellierung einbezogen wird. Relevante Perspektiven sind in dieser Hinsicht Fragen nach gewähltem Ausschnitt der Realität, Fragen des Gesamtbestandes, Ereignissen und Historisierung.
- **MUST**: Sowohl für das System- als auch für das Austauschmodell muss es Beschreibungen geben, die klar definieren, was in diesen Modellen enthalten ist, und wie die Beziehung der beiden Modelle zueinander ist. Für beide Modelle muss es jeweils ein Bestandes- und ein Ereignismodell geben, das klar beschreibt, wie Bestandesdaten aussehen, und welche Ereignisse eintreten können. Zu den Ereignissen gehört dann jeweils die Beschreibung, wie sich diese auf den Bestand auswirken.
- **SHOULD**: Die Beschreibungen für das System- und das Austauschmodell sollten in einer natürlichen Sprache und in einer formalen Beschreibungssprache vorliegen.
- **SHOULD**: Die Beziehungen zwischen dem Systemmodell und Austauschmodellen sollten so genau wie möglich beschrieben werden. Nur auf diese Weise kann bei Änderungen an den Modellen einfach nachvollzogen werden, was dies für Konsequenzen für die Verarbeitung und/oder den Austausch von Daten hat.
- **MAY**: Es können XML Schemas benutzt werden, um die Details der Bestandes- oder Ereignismodelle zu beschreiben. XML Schema ist aufgrund seiner eingebauten Typen gut für diese Aufgabe geeignet, sollte allerdings im Fall komplexerer Strukturen

mit einer graphischen Darstellung dokumentiert werden (wie sie z.B. von gängigen Tools für das Editieren von XML und XML Schemas generiert werden können).

4.2 Revision/Nachführungspolitik des Modells

Wie der gesamte Anwendungsfall, so wird sich auch das Modell im Laufe der Zeit verändern, es wird neuen Anforderungen angepasst werden müssen, und erkannte Schwachstellen werden beseitigt werden müssen. Es ist aus diesem Grund sehr wichtig, festzulegen, in welcher Weise solche Nachführungen ausgeführt werden (wer dafür verantwortlich ist, wer sie ausführt, und wer über erfolgte Änderungen in Kenntnis gesetzt wird), und wie sie sich auf die vom Modell abhängenden Schemas auswirken.

Meist wird eine Änderung des Modells auch eine Änderung der vom Modell abhängenden Schemas zur Folge haben, und aus diesem Grund muss eine klare Nachführungspolitik des Modells existieren, bis hin zu den Auswirkungen auf die Schemas und in der Folge auf die Softwarekomponenten, die mit Instanzen der Schemas arbeiten.

Damit der inhaltliche Zusammenhang von Modell und Schema gewährleistet ist, muss es also auch einen Verweis vom Schema auf das Modell geben, so dass bei geplanten Änderungen am Schema bekannt ist, aus welchem Modell das Schema erzeugt wurde, und damit auch klar ist, an welchem Modell geplante Änderungen zuerst ausgeführt werden müssen, bevor das Schema entsprechend geändert werden kann. Im Kontext von eCH wird dieser Verweis vom Schema auf das Modell über den Namespace Namen gewährleistet, der gemäss [eCH-0033] auf eine Beschreibung des Namespaces verweist, über die dann das Modell gefunden werden kann. Die Indirektion über die Namespace-Beschreibung erlaubt es, Zugriff auf alle relevanten Informationen zum betreffenden Vokabular zu erhalten.

4.2.1 Empfehlung

- **SHOULD**: Änderungen sollten immer zuerst am Modell durchgeführt werden und erst danach als entsprechende Änderung am dazugehörigen Schema.
- **SHOULD**: Die Änderungspolitik eines Modells und des davon abgeleiteten Schemas sollte kommuniziert werden. Es sollte dokumentiert werden, in welcher Weise Änderungen am Modell durchgeführt werden, und in welcher Weise diese Änderungen dann auf das Schema übertragen werden.

4.3 Abhängigkeit des Modells von anderen Modellen

- **MUST**: Ist ein Modell oder ein Schema von anderen Modellen oder Schemas abhängig, so ist diese Abhängigkeit deutlich zu dokumentieren. Nur auf diese Weise kann sichergestellt werden, dass keine Abhängigkeiten entstehen, über die sich der Benutzer eines Schemas oder eines Modells nicht im Klaren ist.

4.4 Zuständigkeit für das Modell

Das formale Modell ist der Ausgangspunkt für jegliche Definition von Implementierungen, aus diesem Grund muss klar geregelt sein, wer für das formale Modell verantwortlich ist, so

dass im Fall von Implementierungsfragen ein Ansprechpartner gefunden werden kann. Auch hier bietet die Information in der Namespace Beschreibung die Möglichkeit, den Ansprechpartner zu finden, da dies ein verpflichtender Bestandteil einer Namespace Beschreibung ist.

- **MUST**: Kontaktmöglichkeiten und Verantwortlichkeiten für ein Modell oder eine Schema müssen deutlich dokumentiert werden. Nur auf diese Weise ich sichergestellt, dass alle Benutzern eines Modells oder eines Schemas kommunizieren können, und dass eine Anlaufstelle existiert, an die sich Interessierte wenden können.

4.5 Redundanz

Es kann durchaus möglich sein, dass ein zusammenhängendes Gesamtmodell auf verschiedene Speichermedien oder verschiedene Speicherorte verteilt werden soll. Dies kann sich aus verschiedenen technischen oder fachlichen Gründen ergeben, und in diesem Fall müssen Mechanismen für Abgleich und Synchronisation der Daten vorgesehen werden. Es muss die Möglichkeit geschaffen werden, wie man aus den verschiedenen Implementierungen durch ein Zusammenführen der Daten wieder zu einem konsistenten und zusammenhängenden Modell gelangen kann.

- **SHOULD**: Bekannte Redundanzen sollten dokumentiert werden. Auf diese Weise kann vermieden werden, dass durch Unkenntnis redundanter Daten Inkonsistenzen erzeugt werden, die bei späterer Interpretation oder Zusammenführung von Daten problematisch wären.

4.6 Identifikatoren

Gibt es verschiedene Datenspeicher (z.B. aufgrund redundanter Speicherung wie in Abschnitt 4.5 beschrieben), so muss eindeutig beschrieben sein, wer Objekte identifiziert, und wie diese Identifikationen kommuniziert werden. Andernfalls besteht die Gefahr, dass redundante Daten unterschiedliche Identifikatoren erhalten und dann nicht mehr erfolgreich zusammengeführt werden können.

- **MUST**: Identifikatoren müssen sich immer auf die Gesamtheit beziehen, innerhalb der sie identifizierend sind. Dies kann u.U. auch bedeuten, dass sie auf eine Autorität verweisen müssen, die diese Gesamtheit definiert oder ihrerseits identifiziert.
- **SHOULD**: Persistente Identifikatoren sollten nicht rein lokale Bedeutung haben. Sie sollten einen klar definierten Identifikations-Herren haben, der sich unabhängig vom Kontext des Identifikators feststellen lässt.
- **MUST**: Werden Identifikatoren verwendet, so müssen deren Eigenschaften und Aufbau beschrieben sein. Dies betrifft insbesondere die Frage, ob es sich um zusammengesetzte Identifikatoren handelt, wie gross die Wahrscheinlichkeit von Duplikaten ist, und ob es eine zentrale oder koordinierte Vergabe der Identifikatoren gibt.
- **SHOULD**: Neben der Erzeugung und dem Aufbau von Identifikatoren muss auch klar geregelt sein, wer die Identifikatoren schliesslich Objekten zuordnet, so dass diese dann über den Identifikator eindeutig identifiziert sind.

4.7 Verantwortlichkeit über Daten/Instanzen

Überlappungen der Verantwortlichkeit können aus verschiedenen Gründen auftreten. Dies kann passieren auf Grund geographischer Überlappungen (Grenzen "gehören" im Allgemeinen beiden angrenzenden Gebieten und müssen koordiniert verschoben werden), fachlicher Überlappungen (Zuständigkeiten verschiedener Instanzen für den gleichen Sachverhalt, oft aus verschiedenen Gründen), oder politischer Überlappungen (verschiedene politische Ebenen können mit dem gleichen Sachverhalt umgehen, eine jede aus ihrer eigenen Perspektive).

- **MUST**: Beim Austausch von Daten von Daten muss klar dokumentiert werden, über welche der Daten als Folge des Austauschs die Datenherrschaft besteht, so dass sie geändert werden können, und welche lediglich Teil des Datenaustauschs sind, damit die interpretiert werden können (die Datenherrschaft ist in diesem Fall aber bei der Datenquelle verblieben).
- **MUST**: Datenherrschaft muss klar definiert sein, es muss also für alle an einem Datenaustausch beteiligten Daten klar definiert sein, wer die Datenherrschaft hat (die Quelle, der Empfänger, oder eine dritte Partei).
- **MUST**: Gibt es Überlappungen der Datenherrschaft (so dass z.B. zwei verschiedene Parteien Änderungen vornehmen können), so ist zu definieren, wie bei Konflikten die Situation bereinigt werden kann.

4.8 Modell-Varianten

- **MUST**: Gibt es für ein zugrunde liegendes Modell verschiedene formale Modell-Varianten, so muss diese Abhängigkeit dokumentiert werden. Nur bei einer solchen Dokumentation kann bei Änderungen an einer Variante erkannt werden, dass die anderen Varianten entsprechend angepasst werden müssen.

5 Abbildung eines Modells auf XML Schema

5.1 Referenzierbarkeit von Identifikatoren

Treten in einem Schema Identifikatoren auf, so können diese Identifikatoren entweder auf interne Objekte verweisen, die direkt referenziert werden können, oder auf externe Objekte, die nicht direkt referenziert werden können, weil sie z.B. in einem anderen System gehalten werden.

- **SHOULD**: Identifikatoren sollten im Schema so gekennzeichnet werden, dass erkenntlich wird, ob sie interne und/oder externe Objekte identifizieren.

5.2 Unterschiedliche Constraints auf verschiedenen Verarbeitungsstufen

In vielen Fällen kann es für ein Modell unterschiedliche Randbedingungen geben, die an Instanzen des Modells gestellt werden, je nachdem, an welcher Stelle in einem Verarbeitungsprozess sich die Instanz befindet. So kann z.B. direkt nach der Eingabe von Daten noch eine grosse Toleranz gegenüber fehlenden, unvollständigen oder fehlerhaften Teilen eines Datensatzes bestehen, währenddessen diese Toleranz nach dem Durchlaufen einer Stufe zur Qualitätssicherung nicht mehr bestehen soll.

Das zugrunde liegende Modell ist in einem solchen Szenario identisch, jedoch werden je nach Verarbeitungsstufe oder anderen Kriterien unterschiedliche Randbedingungen gestellt. Dies entspricht der bereits zuvor beschriebenen Beziehung zwischen Austauschmodellen und Austauschschemas, wo es für ein Austauschmodell verschiedenen Austauschschemas geben kann. In diesem Fall beschreiben die verschiedenen Austauschschemas nicht verschiedene Ausschnitte des Austauschmodells, sondern verschiedene Anforderungen an einen gleich bleibenden Ausschnitt des Austauschmodells. Im Zusammenhang mit XML gibt es für diese Art von Schemavariationen verschiedene Implementierungsmöglichkeiten:

- Es können verschiedene Schema-Varianten erstellt werden, die verschiedenen Randbedingungen enthalten. Ein populäres Beispiel für diesen Ansatz ist (X)HTML, für das es ein *transitional* und ein *strict* Schema gibt. Der Nachteil dieses Ansatzes ist, dass die beiden Schemas keine direkte Abhängigkeit haben, bei der Entwicklung und vor allem auch der Versionierung von Hand darauf geachtet werden muss, dass die Schemas tatsächlich deckungsgleich bleiben mit Ausnahme der Randbedingungen. Für komplexe Schemas ist es eine nicht-triviale Aufgabe, zu überprüfen, ob ein Versionierungsschritt zu Divergenzen verschiedener Schema-Varianten geführt hat. Es ist für komplexe Schemas daher ratsam, Schema-Varianten mit automatisierten oder halb-automatisierten Verfahren zu erzeugen.
- Die strukturelle Beschreibung eines Schemas und die Definition der Randbedingungen können getrennt werden. Für die strukturelle Beschreibung lässt sich XML Schema einsetzen, während die Randbedingungen z.B. in Schematron [ISO19757-3] definiert werden können. Durch diese Trennung von Struktur und Randbedingungen ist es einfacher, die gemeinsam zugrunde liegende Struktur zu erhalten, jedoch muss für Versionierungen darauf geachtet werden, dass die Randbedingungen entsprechend angepasst werden, wenn sich Strukturen ändern. Hilfe kann hier das Einbetten

von Schematron in XML Schema bieten (in Form von `appinfo` Elementen), jedoch gibt es für diese Art der Schemasprachenkombination nur wenig etablierte Verfahren und Unterstützung und sie erfordern daher einiges Detailwissen im Umgang mit verschiedenen XML-Technologien.

Wie eine u.U. gewünschte Realisierung verschiedener Schemavarianten vorgenommen wird, kann aufgrund der schwachen Unterstützung durch etablierte Technologien und Tools nicht vorgegeben werden. Es ist aber zu empfehlen, eine solche Konstruktion zu benutzen, wenn sie durch das Anwendungsszenario sinnvoll erscheint. Auch eine etwas kompliziertere Kombination verschiedener Schemas und u.U. sogar verschiedener Schemasprachen ist besser, als die Kenntnis verschiedenartiger Randbedingungen nicht in Schemas festzuhalten, sondern die Überprüfung in Programmcode zu erwarten.

- **SHOULD**: Gibt es auf verschiedenen Verarbeitungsstufen verschiedene Randbedingungen für Instanzen, so sollte diese Situation am besten durch verschiedene Schema-Varianten abgebildet werden.
- **SHOULD**: In jeder Schema-Variante sollten so viele Randbedingungen wie möglich festgehalten werden, damit möglichst viele Tests durch Validieren und möglichst wenige durch Programmieren implementiert werden können.
- **SHOULD**: Die Schema-Varianten sollten als verschiedene Varianten implementiert werden mit einem dokumentierten Zusammenhang, dies kann entweder über den Namensraum oder über die Dokumentation erfolgen.
- **MUST**: Werden die Schema-Varianten mit dem gleichen Namensraum markiert, so muss es eine maschinenlesbare Markierung im Schema geben, über die ersichtlich wird, um welche Variante es sich handelt.
- **MUST**: Werden die Schema-Varianten mit unterschiedlichen Namespaces markiert, so muss in der Dokumentation festgehalten werden, dass die Schema-Varianten zusammengehörig sind.

5.3 Aktualität und Aktualisierung von Objekten

Werden Objekte verändert, so ist insbesondere bei u.U. vorhandenen Redundanzen wichtig, dass sich feststellen lässt, welche Objekte in welchem Status sind, und auf welche Weise sich der aktuelle Status oder ein historischer Status herstellen lässt. Dazu muss es ein klar definiertes Modell geben, wie mit Änderungen umgegangen wird, und wie mit diesen Änderungen im Rahmen des Gesamtmodells verfahren werden soll.

- **MUST**: Der Status von Objekten muss in der Instanz erkennbar sein, falls diese Information relevant ist.
- **SHOULD**: Die Strategie zur Aktualisierung von Objekten sollte im oder mit dem Schema dokumentiert werden, falls dies relevant ist.

5.4 Beziehungen zwischen Objekten

Beziehungen zwischen Objekten werden in XML üblicherweise auf zwei unterschiedliche Arten dargestellt, einerseits als hierarchische Beziehung durch die XML Baumstruktur, ande-

rerseits als Referenzen, wo die zueinander in Beziehung zu setzenden Strukturen Identifikationen erhalten, und die Beziehungen über Referenzen dargestellt werden (ähnlich dem Schlüssel/Fremdschlüssel Konzept in relationalen Datenbanken).

Während die Repräsentation über Hierarchien durch die Baumstruktur von XML auf 1:n Beziehungen beschränkt ist, können über durch Referenzen beliebige Beziehungen dargestellt werden, im Fall von n:m Beziehungen allerdings auch nur über Konstrukte, die den Hilfstabellen in relationalen Datenbanken ähnlich sind.

Generell finden sich zur Frage der Abbildung von Beziehungen auf die in XML Schema unterstützten Konstrukte und die damit verbundenen Randbedingungen weitergehende Kommentare in [eCH-0035].

Handelt es sich beim Modell um ein Modell, in dem Beziehungen eine zentrale Rolle spielen und zudem nicht unbedingt ein innewohnendes Merkmal der Objekte sind, sonder eigenständige Objekte sein können und sollen, so kann es sich anbieten, die eher einfachen Mechanismen von Hierarchien oder Referenzen nicht zu verwenden, sondern mit komplexeren Mechanismen zu arbeiten, die dafür leistungsfähiger sind. Ein Beispiel dafür ist die *XML Linking Language (XLink)* [XLink10], die definiert, wie Hypermedia Links in XML codiert werden sollen. XLink ist stark an Hypermedia-Anwendungen orientiert und deshalb nicht unmittelbar für andere Anwendungen anwendbar, aber die Grundkonzept ist flexibel und erweiterbar, und bietet einen guten Ausgangspunkt, um komplexe Beziehungen zwischen Objekten ausdrücken zu können und diese als eigenständige Objekte behandeln zu können.

- **SHOULD**: Die Kardinalität von Beziehungen muss dokumentiert werden, falls sie nicht direkt auf Schema-Konstrukte abgebildet werden können.
- **MUST**: Falls Beziehungen mit redundanten Verweisen umgesetzt werden, so muss dies dokumentiert werden, damit die Interpretation und Aktualisierung der Daten darauf Rücksicht nimmt.

5.5 Konsistenzbedingungen

Sehr oft ist es unmöglich, alle im Modell enthaltenen Konsistenzbedingungen im Schema zu implementieren. Dies ist vor allem deshalb der Fall, weil XML Schema (wie in Abschnitt 5.7 beschrieben) technische Einschränkungen hat, und auch über XML Schema hinausgehende deklarative Massnahmen oftmals nicht dazu in der Lage sind, die gesamte Menge an Konsistenzbedingungen des Modells auszudrücken.

5.5.1 Empfehlungen

- **MUST**: Konsistenzbedingungen, die nicht durch das XML Schema oder ergänzende Mechanismen (wie in Abschnitt 5.7 beschrieben) garantiert werden können, müssen so detailliert dokumentiert werden, dass ihre Überprüfung in der Applikation realisiert werden kann.
- **SHOULD**: Konsistenzbedingungen, die nicht durch das XML Schema oder ergänzende Mechanismen garantiert werden können, sollten immer dann überprüft werden, wenn eine Schema-Validierung sowie eine Überprüfung der durch ergänzende Mechanismen definierten Konsistenzbedingungen durchgeführt wird.

5.6 Berechtigung zur Datennutzung

- **MUST**: Einschränkungen der Datennutzung aus Aspekten der Sicherheit oder des Datenschutzes müssen zwischen Partnern eines Datenaustausches separat geregelt werden, sie sind nicht Aspekt eines Schemas oder dessen Dokumentation.

5.7 Einschränkungen von XML Schema

XML Schema ist als Sprache zur Definitionen von Strukturen recht mächtig und ein deutlicher Fortschritt gegenüber der *Document Type Definition (DTD)*, hat jedoch gegenüber anderen Schemasprachen gewissen Einschränkungen. Je nachdem, was für Schemas definiert werden sollen, können die Einschränkungen von XML Schema bedeutungslos sein, oder sie können der Grund dafür sein, dass viel Information aus dem Modell nicht im Schema abgebildet werden kann. Es gibt zwei hauptsächliche Konkurrenten, die sich als Ersatz oder als Ergänzung von XML Schema einsetzen lassen:

- *RELAX NG* [ISO19757-2] ist wie XML Schema grammatikbasiert und erlaubt die Definition eines Schemas durch die Angabe der Regeln, wie Elemente und Attribute zusammengesetzt werden müssen. RELAX NG umgeht einige der Einschränkungen, die XML Schema für die Grammatiken definiert, und erlaubt damit die Definition von Grammatiken, die in XML Schema nicht definiert werden können.
- *Schematron* [ISO19757-3] ist rein regelbasiert und erlaubt es, für beliebige Elemente und Attribute in einem Dokument zu beschreiben, welche Regeln befolgt werden müssen innerhalb von Instanzen. Dabei kann es sich auch um komplizierte und kontextabhängige Ausdrücke handeln, aus diesem Grund kann man mit Schematron viele Regeln definieren, die sich in einer grammatikbasierten Schemasprache nicht ausdrücken lassen. Schematron eignet sich sehr gut als Ergänzung einer grammatikbasierten Sprache, jedoch weniger als einzige Sprache zur Schemadefinition.

Ausgehend von den Randbedingungen, die über XML Schema hinaus definiert werden sollen, ist es also wichtig, die dazu am besten geeignete Schemasprache auszuwählen.

5.7.1 Empfehlungen

- **MUST**: Als Grundlage für Schemas für XML Dokumente muss XML Schema eingesetzt werden, weil es die am weitesten verbreitete Schemasprache ist.
- **SHOULD:** Gibt es Randbedingungen, die nicht durch XML Schema abgebildet werden können, so sollte der Einsatz ergänzender Schemasprachen wie RELAX NG oder Schematron erwogen werden.
- **SHOULD**: Konsistenzbedingungen, die nicht durch das grundlegende XML Schema, jedoch als ergänzende Mechanismen definiert werden, sollten immer dann überprüft werden, wenn eine Schema-Validierung durchgeführt wird.

6 Sicherheitsüberlegungen

6.1 Empfehlungen

- **MUST**: Sind Konsistenzbedingungen für Daten nicht im XML Schema implementiert, so muss dies im Schema dokumentiert werden.
- **SHOULD**: Es sollten so viele Konsistenzbedingungen wie möglich im XML Schema implementiert werden.

7 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.

8 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-0035] Erik Wilde, *Design von XML Schemas*, eCH, Berne, Switzerland, eCH-0035, 2006.

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

[Glushko2005] Robert J. Glushko, Tim McGrath, *Document Engineering*, The MIT Press, Cambridge, Massachusetts, August 2005, ISBN 0-262-07261-0. http://www.docengineering.com/

[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. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37605

[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. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40833

[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

[UML20] *UML 2.0 Superstructure Specification*, Object Management Group, Framingham, Massachusetts, October 2004. http://www.omg.org/cgi-bin/doc?ptc/2004-10-02

[XLink10] Steven J. DeRose, Eve Maler, David Orchard, *XML Linking Language (XLink) Version 1.0*, World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001. http://www.w3.org/TR/2001/REC-xlink-20010627

[XML10] 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

[XML-DSig] Donald E. Eastlake, Joseph M. Reagle, David Solo, *XML-Signature Syntax and Processing*, World Wide Web Consortium, Recommendation REC-xmldsig-core-20020212, February 2002. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212

[XML-Enc] Donald E. Eastlake, Joseph M. Reagle, Takeshi Imamura, Blair Dillaway, Ed Simon, *XML Encryption Syntax and Processing*, World Wide Web Consortium, Recommendation REC-xmlenc-core-20021210, December 2002.

http://www.w3.org/TR/2002/REC-xmlenc-core-20021210

[XSD1] 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

[XSD2] 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

Patrick Ostertag, Etat de Fribourg

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/.

UML	Unified Modeling Language
XML	Extensible Markup Language