Ein interdisziplinärer Ansatz zum Conformance Testing mit beispielhafter Umsetzung in den Bereichen E-Procurement und E-Learning

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers im Studiengang Informatik

vorgelegt von

Sascha Zimmermann

Matrikelnummer 203110639

Betreuer:	Ansgar Mondorf Institut für Wirtschafts- und Verwaltungsinformatik
Erstgutachter:	Prof. Dr. Maria A. Wimmer Forschungsgruppe Verwaltungsinformatik, Institut für Wirtschafts- und Verwaltungsinformatik
Zweitgutachter:	Dr. Ingo Dahn Institut für Wissensmedien
Beginn der Arbeit:	23.02.2012
Ende der Arbeit:	06.11.2012

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Forschungsgruppe für Qualifikationsarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Koblenz, den 06.11.2012) (Unterschrift)

Zusammenfassung

Standards haben längst Einzug in das Gebiet der Informatik gehalten. Verschiedenste Organisationen beschäftigen sich mit der Beschließung von Normen für die standardisierte Lösung von Problemen in der Informatik. Ein wichtiger Teilaspekt ist die Spezifizierung von Datenformaten, die die Interoperabilität von Programmen sichern kann. Dabei existieren viele verschiedene Schemasprachen, die auf unterschiedliche Anforderungen ausgelegt sind, und mit deren Hilfe die Menge der konformen Dokumente beschrieben werden kann. Die Kombination mehrerer Schemasprachen ist sinnvoll, da die Mächtigkeit einer Sprache oftmals nicht ausreicht, um die Anforderungen zu erfassen.
So auch im Fall der, von IMS entwickelten, Spezifikation *Common-Cartridge*. Das Common-Cartridge-Format beschreibt valide Zip-Pakete, die dazu genutzt werden können, verschiedene Lernobjekte zu aggregieren und als zusammenhängende Lerneinheit in eine Lernplattform zu importieren. Die Spezifikation benutzt bereits vorhandene und bewährte Spezifikationen, um Teile gültiger XML-Dokumente zu beschreiben. Diese sind selbst Teil eines Pakets und können Referenzen enthalten, die den Inhalt mit der Paketstruktur in Verbindung bringen. Dabei wird das Common-Cartridge-Format durch ein sogenanntes *Domain Profile* erfasst. Ein solches erlaubt die Anpassung einer oder mehrerer Standards bzw. Spezifikationen, um die Bedürfnisse einer bestimmten Domäne abzudecken. Zudem können sogenannte Testregeln definiert werden, die für die Bestimmung der für ein Datenpaket auszuführenden Testaufgaben genutzt werden können. Letztlich kann die automatische Erstellung eines Testsystems veranlasst werden, welches diese Aufgaben ausführen kann. Somit stellt sich die Frage, ob ein Domain-Profile ebenso dazu eignet ist, die Anforderungen anderer paket-basierter Datenformate zu erfassen, und ob die nötigen Änderungen an den unterstützenden Tools vorgenommen werden können. Schließlich würde dies die Überprüfung gestellter Anforderungen ermöglichen.
Diese Arbeit soll sich mit einer beispielhaften Anwendung des Verfahrens befassen, um diese Fragen nach Möglichkeit zu beantworten. Dazu soll der verwendete Ansatz auf die Datenformate übertragen werden, die Teil der Spezifikation des *Virtual Company Dossiers* - VCD - sind. Diese legen ebenso paket-basierte Formate fest, die innerhalb eines elektronischen Ausschreibungsprozesses verwendet werden. Sie ermöglichen die Erfassung von Evidenzen, die die notwendige Qualifikation eines Bewerbers, sowie die Erfüllung der, mit dem Ausschreibungsverfahren verbundenen Kriterien nachweisen.
Im praktischen Teil werden somit zuerst die Ähnlichkeiten der beiden Formate herausgearbeitet. Diese dienten der Identifizierung abstrakter Anforderungen, die, soweit möglich, mit den gleichen Formalismen festgehalten wurden. Dabei wurden Änderungen an den unterstützenden Tools vorgenommen, um die Erfassung aller VCD-spezifischen Anforderungen zu ermöglichen. So wurden das Format, und das Tool zur Erstellung von Applikationsprofilen, angepasst. Schließlich erfolgten Änderungen am generischen Testsystem, welches zur Herleitung konkreter Testsysteme genutzt wird.
Letztlich stellte sich heraus, dass ähnliche Anforderungen bestehen, und die unterstützenden Tools in einer Weise angepasst werden konnten, die die Menge der erfassbaren Anforderungen erweitert. Da der Hintergrund der beteiligten Spezifikationen sich zu-

dem signifikant unterscheidet, ist davon auszugehen, dass das Verfahren in einem weiten Spektrum von Anwendungsfällen eingesetzt werden kann.

Abstract

Standards are widely-used in the computer science and IT industry. Different organizations like the International Organization for Standardization (SO) are involved in the development of computer related standards. An important domain of standardization is the specification of data formats enabling the exchange of information between different applications. Such formats can be expressed in a variety of schema languages thereby defining sets of conformant documents. Often the use of multiple schema languages is required due to their varying expressive power and different kind of validation requirements.
This also holds for the Specification Common Cartridge which is maintained by the IMS Global Learning Consortium. The specification defines valid zip packages that can be used to aggregate different learning objects. These learning objects are represented by a set of files which are a part of the package and can be imported into a learning management system. The specification makes use of other specifications to constrain the contents of valid documents. Such documents are expressed in the eXtensible Markup Language and may contain references to other files also part of the package.
The specification itself is a so called domain profile. A domain profile allows the modification of one or more specifications to meet the needs of a particular community. Test rules can be used to determine a set of tasks in order to validate a concrete package. The execution is done by a testsystem which, as we will show, can be created automatically. Hence this method may apply to other package based data formats that are defined as a part of a specification.
This work will examine the applicability of this generic test method to the data formats that are introduced by the so called Virtual Company Dossier. These formats are used in processes related to public e-procurement. They allow the packaging of evidences that are needed to prove the fulfillment of criteria related to a public tender.
The work first examines the requirements that are common to both specifications. This will introduce a new view on the requirements by introducing a higher level of abstraction. The identified requirements will then be used to create different domain profiles each capturing the requirements of a package based data format.
The process is normally guided by supporting tools that ease the capturing of a domain profile and the creation of testsystems. These tools will be adapted to support the new requirements. Furtheron the generic testsystem will be modified. This system is used as a basis when a concrete testsystem is created.
Finally the author comes to a positive conclusion. Common requirements have been identified and captured. The involved systems have been adapted allowing the capturing of further types of requirements the which haven't been supported before. Furthermore the background of the specifications quite differ. This indicates that the use of domain profiles and generic test technologies may be suitable in a wide variety of other contexts.

Inhaltsverzeichnis

1 Einleitung

Laut *Lampathaki et al.* besteht seit Ende der sechziger Jahre ein wissenschaftliches Interesse an den Aspekten der Datenmodellierung. Dieses Interesse begann mit der Einführung des *elektronischen Datenaustauschs* - EDI, welcher den, von Software, Hardware oder Kommunikationsnetzwerken unabhängigen Austausch von Geschäftsdokumenten erlaubt (vgl. [47]). In der heutigen Zeit wird häufig die *eXtensible Markup Language* - XML[1] - genutzt, wenn Daten zwischen verschiedenen Systemen oder Softwaremodulen ausgetauscht werden müssen[77]. Dabei können nahezu alle, durch aktuelle Programme bearbeitete Daten in die XML übersetzt werden[57]. Die Modellierung von Daten kann jedoch im Voraus und auf verschiedenen Ebenen erfolgen. So stellen *Lampathaki et al.* einen Top-Down-Ansatz vor, der als erstes die Semantik und die Konzepte hinter den relevanten Daten betrachtet (*konzeptuelle Ebene*). Darauf aufbauend können die auszutauschenden Daten und deren Struktur verstanden werden (*logische Ebene*). Diese Aspekte können schließlich mit verschiedenen Datenmodellen festgehalten und mit einer XML-Syntax verbunden werden (vgl. Kapitel 2.2.1 und [47]).
Dabei kann ein Datenmodell Teil einer Spezifikation sein, die wiederum als Grundlage dienen kann, wenn verarbeitende Programme implementiert werden sollen (vgl. Szenario von *Duval und Verbert* in Kapitel 2.2.3 bzw. [30] - sowie Kapitel 2.3.2). An eine Implementierung können wiederum Anforderungen gestellt werden, die mit den formalen Aussagen der Spezifikation verbunden sind (vgl. Kapitel 2.3.2 und Begriff *Test Assertion* in [42]). Zum Beispiel kann die Erstellung korrekter XML-Dokumente bzw. -Nachrichten gefordert sein (vgl. Kapitel 2.3.2 und [42]).
Diese Arbeit beschäftigt sich mit zwei konkreten Spezifikationen: dem sogenannten *Virtual Company Dossier* - VCD - und der Spezifikation *IMS Common Cartridge* - IMS-CC. Beide Spezifikationen führen verschiedene paket-basierte Datenformate ein. Dabei kann der Inhalt eines Datenpakets durch weitere XML-basierte Dateien beschrieben werden, die im Folgenden auch als Meta-Dateien bezeichnet werden sollen (vgl. Kapitel 3.1.3 und Kapitel 3.2.3).
Die Spezifikation VCD wurde innerhalb des *Pan European Public Procurement Online* - PEPPOL - Projekts erstellt. Der Hintergrund ist durch Prozesse gegeben, die für die Abwicklung einer europaweiten, elektronischen, sowie öffentlichen Ausschreibung genutzt werden können (vgl. Kapitel 3.1.1). Dabei steht die Teilnahme an einer öffentlichen Ausschreibung im Mittelpunkt, für die verschiedene Nachweise erbracht werden müssen. Diese können neben anderen Dokumenten zu einem Datenpaket - auch VCD-Package genannt - zusammengestellt werden (vgl. Kapitel 3.1.1 und Kapitel 3.1.3). Dabei gibt die Spezifikation zwei verschiedene XML-Formate vor, die das Format der Meta-Dateien festhalten (vgl. Kapitel 3.3).
Anders verhält es sich bei der Spezifikation *IMS Common Cartridge* - IMS-CC. Bei dieser stehen die Zusammenstellung von Lerninhalten, deren Import in ein Lernmanagement-

[1] Die Spezifikation XML 1.0 wurde laut *Lampathaki et al.* im Jahr 1998 veröffentlicht. Die aktuelle W3C Recommendation kann unter *http://www.w3.org/TR/REC-xml/* eingesehen werden (zuletzt geprüft am *03.11.2012*).

System und ihre anschließende Verwendung im Mittelpunkt (vgl. Kapitel 3.2.1). Dabei wird das Format einer XML-basierten Meta-Datei vorgegeben, welche zur Beschreibung der, in einem Datenpaket enthaltenden Ressourcen, und für die Definition einer, die Ressourcen betreffenden, Organisation genutzt werden kann (vgl. Kapitel 3.2.3 sowie [64] und [73]).
Beide Spezifikationen legen Modelle fest, die der *konzeptuellen Ebene* und der *logischen Ebene* zugeordnet werden können (vgl. Kapitel 3.1.3 und Kapitel 3.2.3). Diese geben u.a. die möglichen Inhalte der Meta-Dateien vor. Dabei wurden die erstellten Modelle für die Herleitung der syntaktischen Form genutzt, die wiederum auf der XML basiert (vgl. Syntax-Binding zu UBL[13] und IMS-CC-Spezifikation[64]). Die Überprüfung der XML-Dokumente wird in beiden Fällen berücksichtigt (vgl. S. 62 in [52] und Kapitel *Conformance* in [64]). Einerseits können Software-Komponenten geprüft werden, die konforme Daten erzeugen und verarbeiten müssen (vgl. S. 62ff in [52] und Kapitel *LMS Compliance* in [64]). Andererseits kann die manuelle Erstellung von Datenpaketen durch eine Validierung erleichtert werden. Denn in beiden Fällen wurden Dokumente erstellt, die einen Teil der Anforderungen festhalten und in verschiedenen Schemasprachen - z.B. XML-Schema - vefasst sind (vgl. S. 46 in [13] und S.71ff in [64]).
Die Mächtigkeit dieser Sprachen reicht allerdings nicht aus, um alle gestellten Anforderungen zu berücksichtigen (vgl. Kapitel 3.3). Ein Beispiel ist die Überprüfung von Datei-Referenzen, die von einer Meta-Datei ausgehen können. Diese können einen Bezug zur Struktur, und dem Inhalt, eines Datenpakets herstellen (vgl. Kapitel 3.1.3). Solche Anforderungen können jedoch nicht ohne weitere Maßnahmen automatisch geprüft werden. Zudem ist ein Testablauf notwendig, wenn verschiedene XML-Dokumente gegen mehrere, in verschiedenen Schemasprachen verfasste, Dokumente validiert werden sollen[2].
Diese Probleme wurden im Fall der Spezifikation IMS-CC auf eine besondere Art und Weise gelöst. Die Spezifikation ist ein sogenanntes *Domain-Profile* (vgl. [24]). Ein solches erlaubt die Aggregierung von verschiedenen Spezifikationen und deren Modifizierung (vgl. Kapitel 2.4.3). Als Teil der Modifikationen können Anforderungen erfasst werden, die z.B. die Struktur eines Datenpakets betreffen (vgl. Modifikation ACM.4 in Kapitel 2.4.2 und S. 22 in [65]). Dabei adressiert solch eine Modifikation ein Datenelement, das wiederum durch ein Informationsmodell[3] vorgegeben wird (vgl. Kapitel 2.4.1). Die Modifikationen können ebenso auf syntaktischer Ebene erfasst werden, wenn jeweils eine Verbindung zwischen einem Informationsmodell und einer XML-Syntax besteht (vgl. S. 11 in [65]). Letztlich kann ein generisches Testverfahren genutzt werden, welches auf einem *Domain-Profile* basiert. Mit diesem kann automatisch ein Testsystem erzeugt werden, welches für die Überprüfung der gestellten Anforderungen genutzt werden kann (vgl. Kapitel 4).

[2]Das Testing Framework for Global eBusiness Interoperability Test Beds - GIT-B - führt Konzepte ein, die für eine solche Validierung genutzt werden können (vgl. Kapitel 2.3.2 und [42]). Diese werden mit der, in dieser Arbeit vorgestellten Lösung, verglichen (vgl. Kapitel 4).

[3]Ein solches ist eine abstrakte Informationsstruktur, die nicht an eine spezifische Technologie gebunden ist (vgl. S. 11 in [63]) und gehört zu einer Spezifikation (vgl. Begriff *Learning Technology Specification* - S. 6 in [65]).

Das Ziel dieser Arbeit ist, den verwendeten Ansatz auf die VCD-Spezifikation zu übertragen. Mit der erfolgreichen Anwendung soll gezeigt werden, dass dieser Ansatz ebenso für andere paket-basierte Datenformate verwendet werden kann, wenn diese mit einem anderen Hintergrund erstellt wurden. Es wird davon ausgegangen, dass ähnliche Anforderungen bestehen, wenn diese auf einer abstrakteren Ebene betrachtet werden, die die Bedeutung der Elemente nicht im Detail mit einbezieht. Eine Anwendbarkeit geht letztlich mit der Überprüfbarkeit der gestellten Anforderungen einher, was z.B. die parallele Entwicklung der Datenformate und der Implementierungen erlaubt.
Somit werden als erstes die beiden Spezifikationen, und die zugehörigen Datenformate vorgestellt (vgl. Kapitel 3.1.1 und Kapitel 3.2.1). Daraufhin werden die Anforderungen auf einer abstrakteren Ebene verglichen (vgl. Kapitel 3.3). Die dabei gewonnenen Erkenntnisse führen schließlich zur Bestimmung der gemeinsamen Anforderungen (vgl. Kapitel 3.4). Für diese werden letztlich die Arten von Modifikationen vorgeschlagen, die bereits bei der Spezifikation IMS-CC verwendet wurden. Dabei stellt sich heraus, dass durchaus gleichgeartete Anforderungen bestehen, die mit ähnlichen Modifikationen erfasst werden können. Die restlichen Anforderungen werden, soweit möglich, durch andere Modifikationen erfasst (vgl. Kapitel 3.4.3).
Kapitel 4 stellt schließlich das generische Testverfahren dar, welches zur automatischen Erstellung konkreter Testsysteme genutzt werden kann (vgl. Kapitel 4). Im Rahmen der Beschreibung werden die zu erfassenden Modifikationen mit den Testaufgaben in Verbindung gebracht, die für eine Überprüfung auszuführen sind (vgl. Kapitel 4.1 und Kapitel 4.2). Gleichzeitig werden die Probleme herausgearbeitet, die für die Erfassung und die Überprüfung der Modifikationen zu lösen sind (vgl. Kapitel 4.2). Der Ablauf eines Tests wird schließlich in Kapitel 4.3 erklärt. Zuletzt werden der verwendete Arbeitsablauf und die vorgenommenen Änderungen vorgestellt (vgl. Kapitel 4.4).

`Struktur der Arbeit`

Kapitel 2 stellt theoretische Grundlagen vor. Dabei wird in Kapitel 2.1 der Begriff der Interoperabilität behandelt. Die in der Literatur zu findenden Definitionen sind von ihrer Komplexität her sehr verschieden. So können diese den essentiellen Austausch von Daten (vgl. Definition des *IEEE*[2]) oder gar die Zusammenarbeit verschiedener Geschäftsprozesse und Organisationen erfassen (vgl. Definition der *e-Government Working Group of the European Public Administration Network* - EPAN[41]). Letztlich stellt sich die Frage, mit welchen Mitteln Interoperabilität gewährleistet werden kann. Dabei können Geschäftsprozesse koordiniert oder der Informationsfluss gefördert werden (vgl. *information* und *process integration* - [46]). Kapitel 2.2 beschäftigt sich schließlich mit der Modellierung von Datenformaten, und dem Kontext, in dem eine Modellierung stattfindet. So werden zwei mögliche Modelle vorgestellt: *Lampathaki et al.* behandeln die Modellierung von Daten, die Teil einer typischen Business-Transaktion sind, und schlagen u.a. die Verwendung von Ontologien, und sogenannten *Business Infomration Entities* vor (vgl. [47]). *Routledge et al.* nutzen dagegen die Unified Modeling Language - UML, um verschiedene Modelle festzuhalten (vgl. [67]). Letztlich wird ein Lifecycle-Modell

von *Söderström* vorgestellt, welches auf verschiedenen Standardisierungsprozessen basiert (vgl. [74]). Die Datenmodellierung kann ein Teil eines solchen Prozesses sein (vgl. Phase *Develop Standard* in [74]). Kapitel 2.3 stellt schließlich Möglichkeiten vor, wie Anforderungen auf syntaktischer Ebene erfasst und getestet werden können. Die formale Erfassung kann mit Hilfe von Schemasprachen geschehen, wobei das Testen der Anforderungen mit den im GIT-B-Framework eingeführten Konzepten realisiert werden kann[4]. Schließlich können Tests zu Zertifikaten führen, die den Vorteil des Conformance-Testings zeigen: sie ermöglichen die Weitergabe von Vertrauen (vgl. [33] und [34]). Kapitel 2.4 stellt das Konzept des *Domain Profile* vor. Im Rahmen dieser Arbeit wurden mehrere dieser Profile verwendet, um alle Anforderungen der, durch die VCD-Spezifikation vorgegebenen, Datenformate zu erfassen. Auch das vorzustellende Testverfahren baut auf diesem Format auf (vgl. Kapitel 4).

Kapitel 3 stellt die Spezifikationen *IMS Common Cartdige* - IMS-CC - und *Virtual Company Dossier* - VCD - vor, wobei die Spezifikation IMS-CC als *Domain Profile* festgehalten wurde. Zum einen wird der *Hintergrund zu den eingeführten Datenformaten* betrachtet (Kapitel 3.1.1 und Kapitel 3.2.1). Dieser schließt ebenso die Beschreibung von relevanten Personen und IKT-Systemen mit ein, die an einem Datenaustausch beteiligt sein können. Dabei müssen konforme Daten durch verschiedene Implementierungen verarbeitet werden. Zudem werden die *Lösungen* behandelt, *die letztlich die Interoperabilität sichern sollen* (Kapitel 3.1.2 und Kapitel 3.2.2). Diese werden den verschiedenen Interoperabilitätsstufen zugeordnet. Schließlich wird näher auf *das zugrunde liegende Datenformat* eingegangen, welches den Aufbau der Datenpakete beschreibt (Kapitel 3.1.3 und Kapitel 3.2.3). In Kapitel 3.3 werden die Gemeinsamkeiten der beiden Formate vorgestellt, während die Anforderungen in Kapitel 3.4 nochmals mit verschiedenen Modifikationen formalisiert werden.

Kapitel 4 stellt letztlich den Ansatz vor, der für die Erfassung der Modifikationen und die Herleitung entsprechender Testsysteme verwendet werden kann. Von besonderem Interesse sind die Anpassungen, die am Verfahren selbst erfolgten und die Überprüfung der durch die VCD-Spezifikation vorgegebenen Anforderungen ermöglichen. Diese Anpassungen werden in Kapitel 4.4 vorgestellt.

Kapitel 5 bietet schließlich eine Reflektion und einen Ausblick. Die Erfassung der Anforderungen, und die nötigen Anpassungen, werden nochmals im Kontext der Arbeit betrachtet. Der Autor kommt zum Schluss, dass die Anwendung der gezeigten Methodik ebenso für andere paket-basierte Datenformate in Frage kommt. Die Anwendbarkeit hängt jedoch zum Großteil von den gestellten Anforderungen ab. Zudem kann die Verwendung mit Anpassungen einhergehen und die Entwicklung modularer Testkomponenten voraussetzen. Letztlich werden Möglichkeiten aufgezeigt, die die bestehende Lösung sinnvoll erweitern.

[4] Das Framework behandelt weit mehr als die syntaktischen Anforderungen (vgl. [42]).

2 Theoretische Grundlagen

2.1 Der Begriff der Interoperabilität

2.1.1 Definitionen im Bereich der Informatik

Ist die Zusammenarbeit verschiedener Organisationen gewünscht, so steht der Austausch von Informationen oft im Mittelpunkt. Dieser wird meist durch IKT (Informations- und Kommunikationstechnik)-Systeme vollzogen, die in der Lage sein müssen, Daten in einem bekannten Format zu versenden, zu empfangen, und auszuwerten. Dieser Aspekt wird in einer Definition des Interoperabilitäts-Begriffs berücksichtigt, die von dem *Institute of Electrical and Electronics Engineers* - IEEE - getroffen wird, und häufig referenziert wird[2]:

> „The ability of two or more systems or components to exchange information and to use the information that has been exchanged."

Der Umfang der möglichen Zusammenarbeit beteiligter Systeme wird von *Scholl et al.* näher beschrieben[69]:

> „Interoperability has been distinguished from the act of interoperation as the capacity to engage in interoperation[...], that is, the more capable processes and systems are to engage in interoperation, the more they are interoperable (ibid)."

Die beiden vorgestellten Definitionen zeigen grundlegende Eigenschaften auf, die in vielen anderen Definitionen zu finden sind. Diese bringen jedoch verschiedene Sichtweisen mit ein, und konzentrieren sich auf zwei Aspekte:

- das Verhalten zusammenarbeitender IKT-Systeme, und
- den zugrunde liegenden Austausch von Informationen.

Dabei unterscheiden sich die Definitionen in der Komplexität der Aussagen, oder ergänzen sich gegenseitig. In letzterem Fall können die Aussagen ebenso in einer Teilmengenrelation zueinander stehen. Daher sollen verschiedene Definitionen begutachtet werden, um einen umfassenderen Überblick zu gewährleisten.
Asuncion et al. fassen z.B. verschiedene Definitionen des Begriffs zusammen, die durch das IEEE (Institute of Electrical and Electronics Engineers), die ISO (International Organization for Standardization), die OpenGroup, oder allgemeiner im Kontext von Dienstorientierten Architekturen formuliert wurden, und finden dabei einen Konsens[62], der sich ebenso auf eine Arbeit von *Pokraev* stützt[9]:

> „Summarizing, interoperability allows some form of interaction between two or more systems so as to achieve some goal without having to know the uni-

queness of the interacting systems.“

Der gefundene Konsens beleuchtet die Zusammenarbeit verschiedener Systeme, beschäftigt sich aber zudem mit der Einzigartigkeit dieser. Während die erstgenannte Definition (des *IEEE*) den *sinnvollen* Austausch von Daten erfasst, und *Scholl et al.* die Zusammenarbeit verschiedener Systeme betrachten, bringt der durch *Asuncion et al.* gefundene Konsens die Beschaffenheit der einzelnen Systeme mit ein, deren Kenntnis keine Vorraussetzung sein sollte, um den Austausch von Daten zu realisieren.
Dies ist ein besonderes Kriterium, das zwangsweise zu einer vorher getroffenen Vereinbarung führt, die z.B. ein Datenformat und vorgesehene Kommunikationsmuster festlegt. Denn der Effekt einer Nachricht muss stets definiert sein, auch wenn keine Details über den Empfänger vorliegen. *Asuncion et al.* beschreiben diesen Zusammenhang im Detail[9]:

> „A message intention contains what the sender expects the effect of the message will be or the intended use of data on the receiver. [...] The actual effect of the message must thus be compatible with its desired intention.“

Dabei ist der Kontext des Nachrichtenaustauschs wichtig, da er laut den Autoren die Bedeutung einer Nachricht beeinflussen kann. Ein einfaches Beispiel wäre die Verwendung von Kundenspezifischen Daten innerhalb einer Nachricht. Je nach der statt findenden Kommunikation könnten diese mit verschiedenen Absichten gesendet werden (z.B. im Rahmen einer Gutschrift oder einer Rechnungsstellung).
Der Inhalt einer gesendeten Nachricht kann aber ebenso eine Anfrage darstellen, die dazu führt, dass eine bestimmte Funktionalität ausgeführt wird. So kann Funktionalität in separaten IKT-Systemen ausgelagert werden. Dieser Aspekt ist Teil einer Definition, die durch die *e-Government Working Group of the European Public Administration Network* - EPAN - getroffen wurde[41]:

> „Interoperability is the ability of a system or process to use information and/or functionality of another system or process through the adherence to common standards.“

Die EPAN geht nicht nur auf IKT-Systeme ein, sondern bezieht den Geschäftsprozess mit ein. Zuerst sollen die Begriffe *Standard* und *Geschäftsprozess* erklärt werden.

Geschäftsprozess „Ein Prozess ist die inhaltlich abgeschlossene, zeitliche und sachlogische Folge von Aktivitäten, die zur Bearbeitung eines prozessprägenden betriebswirtschaftlichen Objektes wichtig sind. [...] Ein Geschäftsprozess ist ein spezieller Prozess, der durch die obersten Ziele der Unternehmung geprägt wird.“ (nach *Becker und Kahn*[10])

Standard „A 'standard' is to be understood, for the present purposes, as a set of technical specifications adhered to by a producer, either tacitly or as a result of a formal agreement.“ (nach *David und Greenstein*[25])

D.h. Interoperabilität kann ermöglichen, dass innerhalb eines Geschäftsprozesses Informationen und Funktionalitäten genutzt werden, die durch andere Geschäftsprozesse bereitgestellt werden. Da ein Prozess eine Abfolge von Aktivitäten ist, wird auch die Nutzung der bereitgestellten Mittel durch eine Aktivität beschrieben, oder sie ist Teil einer Aktivität. Dabei ist der Zusammenhang mit den IKT-Systemen interessant, die den Austausch der notwendigen Informationen ermöglichen.
Einen näheren Einblick in diesen Zusammenhang bietet z.B. das EIF (European Interoperability Framework), welches eine vereinbarte Herangehensweise an den Begriff der Interoperabilität festlegt, die dazu genutzt werden kann, die Erbringung gemeinsamer öffentlicher Dienstleistungen zu verwirklichen[32]. Somit stellt es einen Leitfaden dar, der die Zusammenarbeit öffentlicher Verwaltungen vereinfachen soll. Dabei geht das Framework zuerst auf die notwendige Interaktion der verschiedenen Organisationen ein, und nutzt eine Definition des Interoperabilitäts-Begriffs, die durch eine Entscheidung des *Rats der Europäischen Union* und des *Europäischen Parlaments* festgehalten wurde (vgl. Article 2 in [1] - S. 20):

> „'interoperability' means the ability of disparate and diverse organisations to interact towards mutually beneficial and agreed common goals, involving the sharing of information and knowledge between the organisations, through the business processes they support, by means of the exchange of data between their respective ICT systems“

Der eigentliche Nachrichtenaustausch ist somit ein Teil der Interaktion, deren Kontext genauer definiert wird. Der Austausch von Informationen soll durch die IKT-Systeme der einzelnen Organisationen verwirklicht werden. Die Zusammenarbeit verschiedener Geschäftsprozesse setzt also zwei Dinge voraus (hergeleitet aufgrund der Definition in [1]):

- die Fähigkeit der Interaktion der einzelnen Systeme, und
- die Abgestimmtheit der zusammenarbeitenden Geschäftsprozesse, in deren Rahmen Interaktionen stattfinden.

Diese zwei Bedingungen sind eine nähere Beschreibung der Voraussetzungen, die für eine erfolgreiche Zusammenarbeit erfüllt sein müssen. Zudem bietet jede der beiden Voraussetzungen eine spezialisierte Sicht auf das Problem, wobei die erfolgreiche Koordinierung von Geschäftsprozessen die Kommunikationsfähigkeit der IKT-Systeme voraussetzt.
Es wurden bereits viele Modelle zur Aufteilung gegebener Interoperabilitäts-Vorraussetzungen vorgestellt und in unterschiedlichen Quellen näher beschrieben. Auch das EIF führt die sogenannten *Interoperabilitätsstufen* ein, die u.a. eine Arbeitsteilung ermöglichen:

- Zuerst kann die Realisierung eines Datenaustauschs in Angriff genommen werden.
- Diese kann dann innerhalb der zu koordinierenden Geschäftsprozesse verwendet werden.

Dabei werden weitere Stufen eingeführt, die im nächsten Kapitel vorgestellt werden sollen. Denn die Unterteilung des Interoperabilitätsbegriffs führt ebenso zu einer möglichen Struktur von Testverfahren, mit deren Hilfe der Erfüllungsgrad der gestellten Interoperabilitätsanforderungen überprüft werden kann.

Abschließend sollen die für diese Arbeit wichtigen Fakten zusammengefasst werden:

- Der Begriff der Interoperabilität bezeichnet entweder einen wünschenswerten Zustand bzgl. der Kommunikationsfähigkeit verschiedener IKT-Systeme (vgl. Definitionen in [62] und [9]), oder ein Maß für diese (vgl. Definitionen in [69] und [70]).
- Jede Kommunikation, und damit jeder Austausch von Informationen, findet zwecks eines zu erreichenden Zieles statt (vgl. Article 2 in [1] - S. 20).
- Der Kontext einer Kommunikation kann durch einen Geschäftsprozess definiert sein (vgl. Article 2 in [1] - S. 20).
- Ist die Kenntnis der Einzigartigkeit bestehender IKT-Systeme keine Voraussetzung, so kann die Realisierung jedes weiteren Systems unabhängig von diesen erfolgen (hergeleitet aufgrund der Definition in [9]). Im Idealfall ist somit keine Kommunikation mit den bereitstellenden Organisationen nötig.
- In diesem Fall werden die Art der zu verarbeitenden Nachrichten durch das vorgegebene Ziel eingegrenzt, und
- das Format der zu verarbeitenden Nachrichten muss vor der Umsetzung eines verarbeitenden Systems fest stehen (hergeleitet aufgrund der Definitionen in [9]).

2.1.2 Ansätze zur näheren Eingrenzung und ihre Schwerpunkte

Die Kommunikationsfähigkeit verschiedener, an einer Zusammenarbeit interessierter, Organisationen ist ein zentraler Aspekt der gezeigten Interoperabilitätsanforderungen. Diese schließt meist die Kommunikationsfähigkeit verschiedener IKT-Systeme mit ein. *Stegwee und Rukanova* gewähren einen näheren Einblick in mögliche Interaktionen, indem sie eine Klassifizierung betrieblicher Systemen vorschlagen, und somit potentielle, an einer Kommunikation beteiligte, Subsysteme identifizieren. Dabei bauen Sie auf einer häufig zitierten Definition des Interoperabilitäts-Begriffs auf, die durch das IEEE getroffen wurde (vgl. [2]) und bereits in Kapitel 2.1.1 vorgestellt wurde. Für die Bildung einer genaueren Definition wird der Begriff des *Systems* erweitert und als *Soziotechnisches System* verstanden, in welchem „Menschen auf ein geteiltes Ziel hinarbeiten, indem

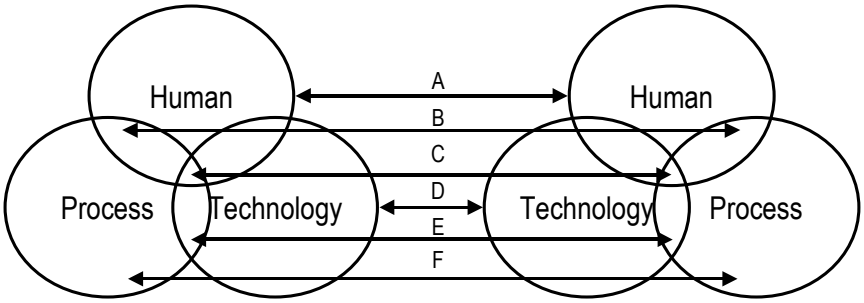


Abbildung 2.1: Abbildung 1 aus [76]: Bestimmung verschiedener Kommunikationsarten zwischen soziotechnischen Systemen

sie eine gemeinsame Menge von auszuführenden Prozessen nutzen und ihre Aufgaben koordinieren"[76].
Dies führt zur Identifizierung der in Abbildung 2.1 auf Seite 9 abgebildeten Subsysteme, die laut den Autoren ebenso durch *Warboys et al.*[78] bestimmt wurden. Dabei stellt jede Seite ein soziotechnisches System dar, und jede Art der möglichen Kommunikation wird entweder durch eine Schnittmenge (zwischen zwei Subsystemen) oder einem Pfeil dargestellt. Während jede Schnittmenge eine Kommunikationsart innerhalb eines Betriebes zeigt, steht jeder Pfeil für eine Art der Kommunikation, die zwischen zwei unterschiedlichen Betrieben stattfinden kann.
Näher behandelt werden jedoch nur jene Kommunikationsarten, die zwischen zwei gleichartigen Subsystemen, z.B. zwischen zwei Menschen, stattfinden können. Für diese (Pfeile `A`, `D` und `F`) werden verschiedene Stufen der Interoperabilität definiert, die anhand verschiedener Arbeiten herausgearbeitet wurden.
In der Literatur finden sich viele Ansätze, die ebenfalls sogenannte Interoperabilitätsstufen definieren. Dabei werden Anforderungen für verschiedene Kommunikationsarten gestellt. Die in der gesichteten Literatur definierten Stufen werden bis auf Ausnahmen mit den gleichen Begriffen bezeichnet, falls die mit Ihnen verbundenen Anforderungen zum Großteil übereinstimmen. Somit ist es dienlich, sich an einer Definition zu orientieren, die das umfassendste Modell (im Sinne der behandelten Problematiken) vorstellt. Dieses ist ein Teil des EIF und wird in Abbildung 2.2 auf Seite 10 veranschaulicht.
Die meisten Stufen finden sich ebenfalls in anderen Quellen wieder. Daher sollen die Unterschiede der vorgestellten Anforderungen aufgezeigt werden. Höher gelegene Ebenen sollen zuerst besprochen werden, da sie weniger Anforderungen an die IKT-Systeme stellen. Vielmehr treten die Gründe und der Kontext eines Datenaustauschs hervor. Die Form der geteilten Daten wird auf der syntaktischen Ebene behandelt.

Die *gesetzliche Ebene* findet sich ausschließlich im EIF wieder und betrifft die Rechtskräftigkeit ausgetauschter Dokumente. Diese muss über Staatsgrenzen hinweg Bestand haben. Zudem können unterschiedliche Bestimmungen zum Datenschutz bestehen, die bei einem Austausch respektiert werden müssen[32]. Da die rechtliche Anerkennung von Dokumenten eine Grundlage für die Zusammenarbeit zweier öffentlicher Verwaltungen

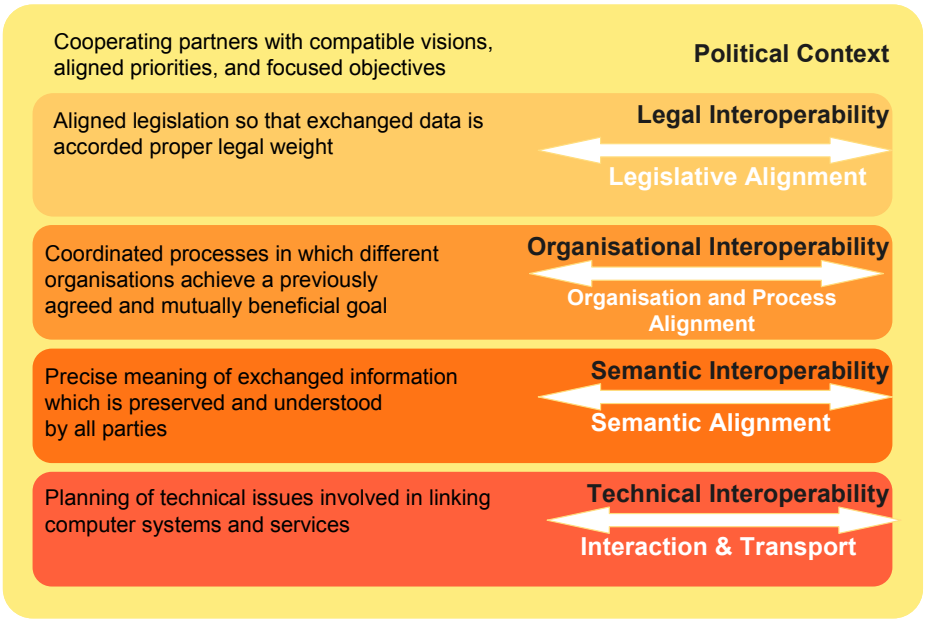


Abbildung 2.2: Abbildung 4-2 aus [32]: Die im EIF vorgestellten Interoperabilitätsstufen

darstellt, kommen die in Abbildung 2.1 auf Seite 9 gezeigten Kommunikationsarten nur beim Bestehen einer solchen zu stande. Die zu beachtenden Datenschutzbestimmungen können sich in allen Kommunikationsarten widerspiegeln: so können diese sowohl den internen als auch den organisationsübergreifenden Austausch von Dokumenten einschränken.

Die *organisatorische Ebene* betrifft die in Abbildung 2.1 auf Seite 9 gezeigte Kommunikationsart `F`. Nach *Gottschalk* stellt organisatorische Interoperabilität ein Maß für die Kommunikationsfähigkeit zweier Unternehmen dar, wenn diese unterschiedliche Arbeitsabläufe verwenden[61]. *Wimmer und Liehmann* ordnen dieser Ebene die „Abstimmung übergreifender Geschäftsprozessmodelle und Informationsarchitekturen“ zu, die zu einer Verbesserung der Interoperabilität führen können. Dabei stellt die Einhaltung der organisatorischen Ziele eine gesonderte Herausforderung dar[79]. Die im EIF zu findende Definition zieht ebenso die Erstellung neuer Geschäfts-Prozesse in Erwägung und betont, dass die resultierende Zusammenarbeit möglichst *effektiv* und *effizient* sein sollte. Im größeren Kontext ist daher auch die Kommunikation zwischen allen Subsystemen der beiden soziotechnischen Systeme (der zusammenarbeitenden Organisationen) betroffen, da die Abstimmung der Geschäftsprozesse ebenso die Interaktionen betrifft, die im Rahmen dieser stattfinden. Dies kann sowohl Mitarbeiter als auch IKT-Systeme betreffen.
Eine in diesem Kontext einzuordnende Stufe der Interoperabilität ist die der *pragmatischen Interoperabilität*, die von *Asuncion und Sinderen* näher untersucht wird. Die Autoren berufen sich auf *Charles Morris*, nach dem die Pragmatik dem Studium der

„menschlichen Interpretation (nicht-) liniguistischer Zeichen“[8] diene, die nach *Chandler* für etwas anderes stehen, und die Form von Wörtern, Bildern, Geräuschen, Gestiken und Objekten annehmen können[14][5]. Der Zusammenhang mit dem Thema der Interoperabilität ergibt sich, wenn ein technisches System „mit seiner Umgebung kommuniziert, indem es Nachrichten austauscht, die aus Zeichen bestehen“[8]. Somit ist die in Abbildung 2.1 auf Seite 9 gezeigte Kommunikationsart C betroffen. Die einwandfreie Kommunikation auf dieser Ebene setzt nach *Asuncion* die Kompatibilität zwischen dem beabsichtigten Effekt, und der wirklichen Verwendung einer, innerhalb eines geteilten Kontextes, ausgetauschten Nachricht voraus[8].
Diese muss ebenso zwischen IKT-Systemen bestehen, die innerhalb verschiedener Geschäftsprozesse zusammenarbeiten. Dieser Aspekt ist Teil einer Betrachtung durch *Asuncion und Sinderen*. Sie unternehmen eine Literaturrecherche zum Begriff der pragmatischen Interoperabilität, und bestimmen verschiedene Anforderungen. Bei der Interaktion zweier Programme spielt der beabsichtigte Effekt einer Nachricht ebenfalls eine zentrale Rolle. Bei der Kommunikation zweier Unternehmen bestehen jedoch weitere Anforderungen, die aus verschiedenen Arbeiten zusammengefasst werden. Die geschäftlichen Absichten und Regeln, und die Strategie der Unternehmen, sollten miteinander vereinbar sein[66]. Zudem sollte ein grundlegendes Verständnis der angebotenen Dienste bestehen[49], und Aspekte wie die Vertrauenswürdigkeit, Bereitschaft, und das Ansehen eines Unternehmens beachtet werden[66].

Die *semantische Ebene* betrifft „die Bedeutung von Datenelementen und die Beziehung zwischen Ihnen“[32]. Sie stellt sicher, dass Datenelemente in gleicher Weise verstanden werden, und schließt die Entwicklung eines Vokabulars mit ein[32]. Ein solches nimmt die zentrale Rolle in der Definition nach *Gottschalk* ein[61]:

> „Semantic interoperability is defined as the extent to which information systems using different terminology are able to communicate.“

Wimmer und Liehmann merken an, dass die „Abstimmung der Datenstrukturen und der Interpretation von Daten und Informationen“ eine Grundlage für die Bearbeitung von Dienstleistungen darstellt[79]. Die Anforderungen auf der semantischen Ebene kann mehrere Kommunikationsarten betreffen. Müssen von einem Mitarbeiter kodierte und gesendete Nachrichten durch einen Empfänger verarbeitet werden, so betrifft dies die Kommunikationsart A, die kohärente Verwendung von Daten in Geschäftsprozessen betrifft dagegen die Kommunikationsart F. Aber auch die Kommunikationsart D ist betroffen, wenn Daten durch Mitarbeiter verarbeitet werden, und dies im Rahmen eines Geschäftsprozesses geschieht.

Die *syntaktische Ebene* betrifft „die Beschreibung des exakten Formats der auszutauschenden Nachrichten, die in Form von Grammatiken, Formaten und Schemata erfolgen

[5] Aus dem Kapitel *Definitions* auf Seite *2*.

kann“[32]. Denn an einer Kommunikation beteiligte IKT-Systeme sollten die auszutauschenden Daten in einer kompatiblen Art und Weise strukturieren[9]. Findet die Kodierung der Daten ausschließlich durch IKT-Systeme statt, so betrifft dies die Kommunikationsart D. Es kann aber ebenso notwendig sein, dass Nachrichten durch Mitarbeiter erstellt werden, so dass auch die Kommunikationsart C genutzt wird. Natürlich macht dies nur Sinn, wenn eine Semantik vorliegt, die es einem Mitarbeiter erlaubt, eine sinnvolle Nachricht zu formulieren.

Die *technische Ebene* deckt „die technischen Aspekte der Verbindung von Informationssystemen“ ab[32]. Nach *Gottschalk* gibt es bereits zahlreiche Produkte und technische Lösungen, die die physikalische Verbindung und die Kommunikation von Systemen betreffen[61].

2.1.3 Das Teilen von Informationen auf der syntaktischen Ebene

Die im letzten Kapitel eingeführten Interoperabilitätsstufen führen verschiedene, aufeinander aufbauende, Bedingungen ein, die für eine erfolgreiche Zusammenarbeit zweier Organisationen erfüllt sein müssen. Die vorgestellten Bedingungen werden zu Ebenen zusammengefasst, die verschiedene Sichtweisen auf die gestellten Anforderungen ermöglichen. Dies erlaubt eine Arbeitsteilung, sowie die Koordinierung der Aktivitäten, die zu einer Umsetzung führen.
Dabei ist die Absprache der Repräsentation und der Bedeutung auszutauschender Informationen von grundlegender Bedeutung. Denn eine solche ist notwendig, wenn beteiligte Geschäftsprozesse koordiniert werden sollen. Laut *Glushko und McGrath* können geschäftliche Beziehungen gar als eine Abfolge auszutauschender Dokumente konzipiert werden. Denn diese können als „die öffentliche Schnittstelle zu den jeweiligen Geschäftsprozessen“ dienen, sowie die Details einer spezifischen Implementation verstecken. Eine Verbindung zu den verabeitenden Prozessen kann mit *Document Engineering Model*s vollzogen werden, die wiederum die Geschäftsprozesse beschreiben, die die Dokumente Erstellen und Konsumieren. Als Beispiel wird eine Kommunikation zwischen zwei Betrieben genannt, die den Versand einer Bestellung und einer darauf folgenden Bestätigung beinhaltet. (vgl. [35])
Es ergibt sich somit die Frage, welcher Weg eingeschlagen werden soll, wenn die Zusammenarbeit verschiedener Organisationen gewünscht ist. *Klischewski* beschäftigt sich näher mit 2 „prominenten Konzepten“, die die unterschiedlichen Schwerpunkte unterstreichen: `information integration` und `process integration`[46]. Dabei wird Integration als Resultat von Kooperation und Interoperabilität angesehen, die sich mit[46]

- „einer gemeinsamen organisations-übergreifenden Strategie und ihrer Implementation“ (Kooperation), und
- „den technischen Mitteln“ beschäftigen, die „IT-Systemen den Austausch von Nachrichten“ erlauben. (Interoperabilität)

Diese Definition trennt die Prozessebene von der technischen und syntaktischen Ebene, wobei sich Entscheidungen über die auszutauschenden Informationen im Format der

Dokumente wiederspiegeln. *Process integration* und *information integration* stellen 2 verschiedene Lösungsansätze für das Erreichen von Integration dar.
Process integration stellt die Verknüpfung von Schritten und Arbeitsgängen verschiedener Prozesse in den Mittelpunkt, die über technische und organisatorische Grenzen hinweg erfolgen soll. *Informationen integration* verschiebt den Fokus auf die Förderung des Informationsflusses, die durch den Zugang zu strukturierten Informationsquellen verwirklicht werden kann. Dabei können die, an einem Informationsfluss beteiligten, Partner entweder Organisationen, Menschen oder Maschinen sein (vgl. *Klischewski*[46]).
Sind die eigentlichen Informationen in Dokumenten beschrieben, so beschreibt *Information integration* einen eher dokument-zentrierten Ansatz, so wie er auch bei *Glushko und McGrath* zu finden ist. Denn laut *Klischewski* sollen Informationen primär als geteilte Resource in Erscheinung treten. Ein Ansatz dazu findet sich in einem Zitat von *Jhingran et al.*: nach den Autoren gehört die physikalische, bzw. logische, Verbindung komplementärer Daten zu den Aufgaben des *information integration*. Dadurch wäre es Programmen möglich, auf alle relevanten Daten eines Betriebes zuzugreifen (*Jhingran et al.* zitiert durch *Klischewski*[46]).
Klischewski unternimmt eine Literaturrecherche und nennt zentrale Aspekte der Analyse und Modellierung. Dazu zählen u.a. Datendefinitionen, Informationsmodelle, und die Klassifizierung von Verwaltungsinformationen. Gleichzeitg werden verschiedene Technologien empfohlen, die zur Lösung von, durch den Ansatz abgeleiteten, Interoperabilitätsbedingungen, genutzt werden können. Eine dieser Technologien ist die eXtensible Markup Language - `XML` [46], die laut dem W3C ein einfaches, sehr flexibles, von SGML abgeleitetes, Textformat ist, und als *W3C Recommendation* vorliegt[6]. Eine W3C-Recommendation ist laut dem W3C mit einem Standard vergleichbar, der durch andere Organisationen veröffentlicht wurde[7]. Die Flexibilität wird ebenso durch *Klarlund et al.* bestätigt, nach dem nahezu alle, durch aktuelle Programm bearbeitete, Daten nach XML übersetzt werden können. Und damit durch verschiedene, zu unterschiedlichen Betrieben, bzw. Organisationen, gehörende, Programme ausgetauscht werden können. (vgl. [57])
Laut *Völkel* ist dies nicht nur möglich, sondern ein häufiges Einsatzgebiet. Er stellt fest, dass XML für den Austausch von Daten, und für die Kommunikation von Softwaremodulen eingesetzt wird. Zudem können die an einem Datentausch beteiligten Programme in unterschiedlichen Programmiersprachen implementiert sein (vgl. S. 35 in [77]). Dieser Aspekt findet sich ebenso in den Bedingungen des EIF wieder: es werden keine Einschränkungen bzgl. der zu verwendenden Programmiersprachen eingeführt. Die relevanten Interoperabilitätsstufen konzentrieren sich ausschließlich auf die Kommunikation zweier IKT-Systeme.
Somit scheint die XML ein geeignetes Mittel für die Kodierung auszutauschender Informationen zu sein. Auch die später einzuführenden paket-basierten Datenformate setzen auf die XML, um verschiedenste Informationen (u.a. Informationen über den Paketinhalt) zu übermitteln (vgl. Kapitel 3.1.3 sowie Kapitel 3.2.3). Ein Problem ist jedoch

[6] Vgl. *http://www.w3.org/XML/ (zuletzt geprüft am* 25.10.2012*).*
[7] *Vgl. http://www.w3.org/2005/10/Process-20051014/tr.html#q74 (zuletzt geprüft am* 25.10.2012*).*

die Semantik der auszutauschenden Informationen. So können Konflikte bei der Interpretation auftreten. *H. Wache et al.* gehen näher auf diese ein. Sie stellen fest, dass semantische Konflikte immer dann auftauchen, wenn die Interpretation aufgrund eines unterschiedlichen Kontextes verschieden ist. In diesem Rahmen zitieren sie *Goh*, welcher „3 Hauptursachen für semantische Heterogenität identifiziert“ (*Goh* zitiert in *Wache et al.*[38]):

Confounding Conflicts Informationselemente scheinen die gleiche Bedeutung zu haben, unterscheiden sich aber in der Realität.

Scaling Conflicts Werte werden mittels verschiedener Referenz-Systeme erfasst (z.B. in verschiedenen Währungen).

Naming Conflicts Die Namensgebung der Informationen unterscheidet sich signifikant. (Beispiel: Synonyme und Homonyme)

Wie im folgenden gezeigt werden soll, können Teile dieser Probleme mit der Hilfe von *Ontologien* gelöst werden. Auch *Klischewski et al.* nennen die Nutzung von Ontologien und Thesauri als Beispiele, um die Klassifizierung administrativer Informationen zu verwirklichen (vgl. [46]). Eine Ontologie kann somit die Interpretation von Informationen einschränken. Schließlich ist die Klassifizierung von Informationen mit einem Gewinn an Semantik verbunden.
Eine genauere Einordnung des Ontologie-Begriffs wird durch *Obrst* getroffen[59], die von *Höfferer* zusammengefasst wird. Laut *Höfferer* definiert *Obrst* ein sogenanntes „ontology spectrum“, dessen Enden für Ontologien mit „schwacher“ sowie „starker Semantik“ stehen. „Eine kontrollierte Liste von Wörtern“ wird dem unteren Ende zugeordnet. Darauf aufbauend werden verschiedene Arten von Ontologien vorgestellt, die sich in der Art der unterstützten Beziehungen (zwischen den Wörten) unterscheiden. Desto mehr Beziehungsarten unterstützt werden, desto höher ist die Einordnung im Spektrum. Ein Beispiel ist der *Thesaurus*, welcher linguistische Beziehungen unterstützt, die z.B. die Erfassung von Synonymen und Homonymen ermöglichen[40]. Ein Thesaurus kann somit für die Lösung der, durch *Goh* erwähnten, Namenskonflilkte dienen, wenn er gemeinsam entworfen bzw. gepflegt wird.
Der Begriff der Ontologie wird in der Literatur jedoch allgemeiner formuliert, und geht über die Erfassung von Wortbeziehungen hinaus. Eine oft zu findende Definition ist die von *Gruber*: „An ontology is an explicit specification of a conceptualization“[37]. *Völkel* fügt hinzu, dass „eine Ontologie meist ein geteiltes Modell einer bestimmten Domäne ist, auf welches sich verschiedene Parteien geeinigt haben“ (vgl. S. 39 in [77]). Ein solches besteht laut *Völkel* aus formalen Aussagen und Konzepten, wobei letztere durch natürlichsprachlichen Bezeichnungen repräsentiert werden (vgl. S. 22 in [77]). *Haslhofer und Klas* berufen sich auf *Noy und Klein*[58], nach denen eine Ontologie ein logisches System ist, dass eine Menge von Axiomen definiert, die Schlussfolgerungen über eine vorgegebene Menge von Fakten erlauben (vgl. [58] - inhaltlich zitiert durch [39]).
Diese Definitionen behandeln die Arbeit mit vorhandenem Wissen. Das Wissen einer Domäne kann in eine Repräsentation überführt, und zudem konsistent gehalten werden.

Letztendlich wird gar die Herleitung neuen Wissens unterstützt (Aussagen hergeleitet aufgrund von [58] und [77]). Dies erleichtert die Arbeit mit ausgetauschten Informationen. Sowohl der Empfänger, als auch der Sender einer Nachricht können Begriffe im Zusammenhang (mit der Domäne) verstehen und verwenden (hergeleitet aufgrund der vorigen Aussage und [77]). Nachrichten können wiederum mit Hilfe der XML kodiert sein (vgl. Aussage von *Klarlund et al.*[57]). Eine Ontologie unterstützt somit die Einigung bzgl. eines Datenformats.
Ontologien werden in Kapitel 3.1 eine Rolle spielen, da diese bei einem der zu behandelnden Datenformate, dem *Virtual Company Dossier* (VCD), zum Einsatz kommen. Zudem verwendet dieses Format verschiedene Vokabulare, die bei einem konkreten Test (von Datenpaketen) berücksichtigt werden müssen (vgl. Kapitel *5.4.1* und Kapitel *5.3.4.4* in [52]). Der Austausch erfolgt zwischen koordinierten Geschäftsprozessen, wobei die mit dem VCD entworfene Architektur die wichtigen Schritte vorgibt (vgl. Kapitel 5.3.1 in [52]).
Sowohl beim VCD, als auch dem in Kapitel 3.2 behandelten Datenformat *IMS Common Cartridge* (IMS-CC), spielen die auszutauschenden Informationen eine zentrale Rolle. Bei dem Format IMS-CC steht gar der Import von Lerninhalten (in eine Lernplattform), sowie die anschließende Nutzung durch verschiedene Beteiligte im Vordergrund. Die behandelten „Use-Cases“ können in einer einzigen Abbildung zusammengefasst werden. Dabei zeigt sich ein linearer Ablauf, der die verschiedenen Use-Cases miteinander verbindet. Innerhalb dieses Ablaufes werden die am Lernprozess beteiligten Nutzer in Ihrer Arbeit unterstützt (vgl. Kapitel 3.2.1 und [64]). Somit kommt die Spezifikation ohne die Vorgabe von Geschäftsprozessen aus. Wie bei dem Ansatz des *information integration* ist der Fluß der Informationen von besonderer Bedeutung.
Die folgenden Kapiteln sollen näher auf die syntaktische Ebene eingehen, da die herauszuarbeitenden Anforderungen eher syntaktischer Natur sind.

2.2 Realisierung gestellter Interoperabilitäts-Anforderungen

Für die Zusammenarbeit verschiedener Organisationen kann die Lösung unterschiedlichster Probleme erforderlich sein. Die in Kapitel Kapitel 2.1.2 eingeführten *Interoperabilitätsstufen* erlaubten eine Klassifizierung, sowie eine nähere Unterteilung, der zu stellenden Bedingungen. Dadurch wurden verschiedene Sichtweisen, als auch eine Herangehensweise deutlich, die eine Umsetzung erlaubt. Zugleich wurden die vorgestellten Stufen im Zusammenhang mit möglichen Kommunikationsarten betrachtet. Dazu wurde auf den Begriff des *soziotechnischen Systems*[76] zurückgegriffen, der eine Kategorisierung der verschiedenen Kommunikationsarten, sei es innerhalb eines Betriebes, oder zwischen verschiedenen Betrieben, erlaubt. Dies ermöglichte eine Fokussierung auf verschiedene Anwendungskontexte. Zum Beispiel kann die Kommunikation zweier Individuen betrachtet werden, die durch verschiedene IKT-Systeme gestützt wird. Eine solche muss nicht unbedingt durch einen Geschäftsprozess erfasst sein.
Ein diesbezüglich häufig anzutreffendes Beispiel ist die Verwendung von Textverarbeitungen, die den Im- und Export des *Open-Document Formats*[8] - ODF - unterstützten[9]. Durch einen Im- bzw. Export können verschiedene Textdokumente ausgetauscht werden, wobei der Austausch vermutlich so fein-granular ist, dass dessen Erfassung in einem Geschäftsprozess nicht immer Sinn macht. Auch die Spezifikation IMS-CC enthält keine Beschreibung von Geschäftsprozessen. Vielmehr stützt sich die Spezifikation auf 5 einzelne Use-Cases, bei denen die Verwendung der, durch eine Common Cartridge - CC - aggregierten, Inhalte im Mittelpunkt steht (vgl. *Beschreibung der Use-Cases* - S. 12-18 in [64]).
Somit wurde in Kapitel 2.1.3 näher auf die Aktivität `information integration` eingegangen, die den Informationsfluss zwischen den, an einer Kommunikation beteiligten, Partnerm fördern soll, und somit die Informationen in den Vordergrund stellt (vgl. [46]). *Klischewski* empfiehlt diesbezüglich verschiedene Technologien (vgl. [46]), die teilweise bei den Spezifikationen *IMS Common Cartridge* - IMS-CC - und *Virtual Company Dossier* - VCD - zum Einsatz kommen. Dazu gehören die *XML*, sowie *RDF/S*[10][11] und *OWL*[12] (vgl. [46]). Die XML wird in beiden Spezifikationen genutzt (vgl. [64], [13], Kapitel 3.1.3, sowie Kapitel 3.2.3). Die beiden anderen Technologien werden in der VCD-Spezifikation verwendet (vgl. S. 19f, 58ff in [54]).
In den folgenden Kapiteln soll auf die folgenden Aspekte eingegangen werden:

- Kapitel 2.2.1 soll näher auf die Modellierung der Dokumente eingehen, die die auszutauschenden Informationen beinhalten. Der vorgestellte Ansatz wird größtenteils auf einer Arbeit von *Lampathaki et al.* basieren, die sich vornehmlich mit Business-Standards beschäftigen (vgl. [47]). Viele der vorgestellten Aspekte und Technologi-

[8] Dieses liegt in verschiedenen Versionen vor, die unter *https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office* verfügbar sind (zuletzt geprüft am *21.10.2012*).
[9] *Apache OpenOffice Writer* ist ein Beispiel für eine Textverarbeitung, die das ODF-Format unterstützt (vgl. *http://www.openoffice.org/* - zuletzt geprüft am *21.10.2012*).
[10] Vgl. *http://www.w3.org/RDF/* (zuletzt geprüft am *21.10.2012*).
[11] Vgl. *http://www.w3.org/TR/rdf-schema/* (zuletzt geprüft am *21.10.2012*).
[12] Vgl. *http://www.w3.org/TR/owl2-overview/* (zuletzt geprüft am *21.10.2012*).

en stehen im engen Zusammenhang mit der später vorgestellten Spezifikation des Virtual Company Dossiers (vgl. Kapitel 3.1.3).

- Kapitel 2.2.2 wird einen alternativen Ansatz zur Datenmodellierung vorstellen, der jedoch große Gemeinsamkeiten mit dem bereits vorgestellten Ansatz hat. Dabei wird vornehmlich die Unified Modeling Language - UML - genutzt, um verschiedene Modelle zu beschreiben (vgl. [67]). Ähnliche Modelle finden sich ebenso bei der *IMS Content Packaging*-Spezifikation - IMS-CP, die als Basis für die Spezifikation IMS-CC dient (vgl. [18], [73], [64], Kapitel 3.2.3).

- Kapitel 2.2.3 wird den Kontext der Datenmodellierung näher erläutern. Die Erstellung eines Datenformats ist nur eine Aktivität, die mit anderen, ebenso wichtigen, Aktivitäten in Verbindung steht (vgl. [74] sowie [75]).

2.2.1 Modellierung formaler Dokumente

Der Austausch von Informationen setzt die Einigung auf ein konkretes, zu verwendendes Datenformat, voraus (vgl. Kapitel 2.1.1). Die Represäntation der auszutauschenden Informationen kann auf der eXtensible Markup Language - XML - basieren. Diese ist eine W3C-Recommendation und laut dem W3C eine überaus flexible Sprache. Zudem kommt sie laut *Klarlund et al.* für den Austausch verschiedenster Informationen in Frage (vgl. [57]). Ein solcher Austausch ist laut *Völkel* ein häufiges Einsatzgebiet (vgl. [77])[13]. *Lampathaki et al.* zeigen allerdings die Grenzen der XML auf. Auch wenn geschäftliche Informationen mittels der XML ausgedrückt werden, und diese durch verschiedene IKT-Systeme ausgetauscht werden, besteht keine Garantie dafür, dass die Informationen in gleicher Weise verstanden werden. Denn die XML alleine gibt weder die Modellierung, die Benennung, noch die Strukturierung der zu Grunde liegenden Geschäfts-Informationen vor (vgl. [47]).
Mit dieser Aussage gehen *Lampathaki et al.* näher auf den Inhalt von Dokumenten ein und nennen bereits die Aspekte der Strukturierung, und der Benennung, auszutauschender Informationen. Diese Aspekte werden näher durch *Glushko und McGrath* behandelt. Die Autoren beschäftigen sich mit dem Aufbau „datenintensiver, elektronischer Dokumente“, die zur elektronischen Abwicklung von Geschäften genutzt werden (vgl. [35]). Dabei stellen sie fest, dass die in einem Dokument enthaltenen Informationen in 3 verschiedene Kategorien eingeteilt werden können[35]:

- *Content components*: die Teile der eigentlichen Informationen eines Dokuments - „the *what is it* information“

- *Structure components*: die Anordnung der Inhalte - „the *where it is* information“

- *Presentation components*: die Formatierung und Visualisierung der Struktur- und Inhalts-Komponenten - „the *what does it look like* information“

[13] Dieser Absatz fasst Fakten aus Kapitel 2.1.3 zusammen.

Diese Komponenten haben laut *Glushko und McGrath* eine unterschiedliche Gewichtung, wenn es um datenintensive, elektronische Dokumente geht. Die Inhalte seien von zentraler Bedeutung. Struktur-Komponenten würden oftmals als Container für Inhalts-Komponenten genutzt. Die Repräsentation der Informationen sei jedoch weniger wichtig (vgl. [35]).
Ein Datenformat muss somit vor allem die *Struktur* der Dokumente, sowie die *Art der zu teilenden Informationen* berücksichtigen. Diese Aspekte werden durch *Lampathaki et al.* aufgegriffen und mit dem Akt der Datenmodellierung in Verbindung gebracht. Dabei nennen sie zwei verschiedene Herangehensweisen, die für die Modellierung auszutauschender Informationen verwendet werden können[47]:

- Der auf *Objektivität* basierende Ansatz basiert auf Entitäten und gibt Datenmodelle vor, die ein Spiegel der Realität sein sollen (Bezug auf *Klein*[45]).
- Der auf *Subjektivität* oder Regeln basierte Ansatz sieht Datenmodellierung als die Formalisierung der Bedeutung auszutauschender Nachrichten an.

Die Autoren verwenden jedoch eine andere Definition. *Lampathaki et al.* sehen die Datenmodellierung als die methodisch ausgereifte, und standardisierte Beschreibung von Daten an, die während einer typischen Geschäftstransaktion gespeichert, als auch kommuniziert werden (vgl. [47]). Diese Definition ist allgemeiner und nennt keine Herangehensweise an das Problem der Datenmodellierung. Ob Daten im voraus definiert, oder aufgrund konkreter Dokumente bestimmt wurden, das Format, und die Bedeutung der Daten, muss in beiden Fällen festgehalten werden.
Dabei verfolgen *Lampathaki et al.* einen Top-Down Ansatz, und stützen sich auf die Erkenntnisse von *Glushko und McGrath*[35], und *Simision und Witt*[72], um die in Abbildung 2.3 auf Seite 19 dargestellten Datenmodelle einzuführen[47]:

- *Das konzeptuelle Datenmodell* beschreibt die Daten von einer abstrakten und semantisch bereicherten Sichtweise (vgl. *Content Components*).
- *Das logische Datenmodell* ruft die Datenstrukturen ins Leben, die während einer Transaktion ausgetauscht werden (vgl. *Structure Components*).
- *Das Physikalische Datenmodell* beschreibt die Speicherung der Daten (vgl. *Representation Components*).

Die gezeigten Konstrukte *Collections*, *Objects* und *Atoms* werden nach *Lampathaki et al.* für jede Abstraktionsebene eingeführt. Die auszutauschenden Informationen werden durch Atome modelliert, die zu logischen Mengen (Objects) zusammengefasst, und wiederum zu Collections kombiniert werden können (vgl. [47]).
Die konkrete Repräsentation der auszutauschenden Nachrichten kann durch XML-Dokumente, die Speicherung der enthaltenen Informationen mittels einer Datenbank erfolgen (vgl. [47]). Dabei können verschiedene, in Abbildung 2.3 auf Seite 19 genannte, Sprachen genutzt werden, um die Modelle zu erfassen:

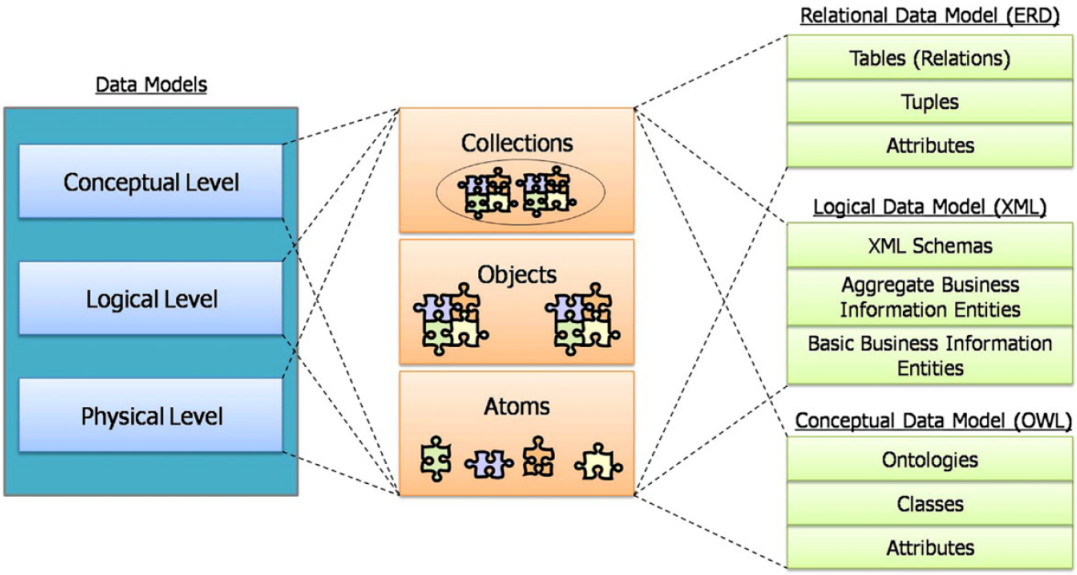


Abbildung 2.3: Abbildung 8 aus [47]: Data Modelling Granularity Levels

- auf dem *konzeptuellen Level* spielen Ontologien, Klassen, und deren Attribute eine Rolle (vgl. [47] und Abbildung 2.3 auf Seite 19), die laut der Abbildung in OWL formuliert werden. Dies deckt sich mit den, durch *Klischewski* vorgeschlagenen, Sprachen, zu denen neben OWL auch RDF/S gehört (vgl. [46]).

- auf dem *logischen Level* spielen XML-Schemata und Einheiten von Geschäftsinformationen eine Rolle (vgl. [47] und Abbildung 2.3 auf Seite 19). Dabei kann ein XML-Schema die Form der gültigen XML-Dokumente (der auszutauschenden Nachrichten) festhalten (vgl. [47]). Für die Modellierung der Informationen existieren bereits spezialisierte Lösungen. Ein Beispiel ist die *Core Components Technical Specification* - CCTS[14]. Diese ist laut *Schroth et al.* eine Methode, die Metamodelle und Regeln für die Definition von semantisch eindeutigen, syntax-unabhängigen, Geschäftsinformationen umfasst (vgl. [71]). Die CCTS steht im engen Zusammenhang mit der VCD-Spezifikation, da diese die XML-Schemata der Universal Business Language - *UBL* - nutzt (vgl. S. 39 in [13] sowie Kapitel 3.1.3). Die UBL ist laut *Lampathaki et al.* die erste Standard-Implementierung der *CCTS* (vgl. 1050 in [47]), und stellt eine Bibliothek von XML-Schemata bereit, die wiederverwendbare Datenkomponenten definieren (vgl. [71]). Die in der Abbildung genannten *Business Information Entities* - BIE - werden ebenfalls in der *CCTS* definiert[14].

- auf dem *physischen Level* können Entity Relationship Diagramme - *ERD*[15] - verwendet werden. Diese beschreiben die Tabellen einer Datenbank, deren Instanzen

[14]Die Spezifikation ist unter *http://www.sdn.sap.com/irj/sdn?rid=/webcontent/uuid/1baa57f9-0a01-0010-1684-c42a08982294* verfügbar (zuletzt geprüft am *21.10.2012*).

die zu speichernden Informationen aufnehmen sollen (vgl. [47] und Abbildung 2.3 auf Seite 19).

Dabei ist die UBL von besonderer Bedeutung für diese Arbeit. Denn die UBL wird für die präzise, und standard-konforme, Definition zahlreicher Elemente genutzt, die Teil eines eingeführten Datenmodells der VCD-Spezifikation sind (vgl. S. 45 in [52]). Daher sollen die *Core Components Library* - CCL, und die CCTS näher besprochen werden. Laut *Schroth et al.* ist die CCL ein Repository für *Core Components* - CC, welche wiederum generische, sowie geschäftliche, Datenkomponenten sind. Dabei beinhalte die CCL nur kontext-unabhängige Datenvorlagen, die generell gültig, und zudem syntax-unabhängig sind (vgl. S. 3 in [71]).
Die CCTS ist nach *Schroth et al.* eine Methode für die Definition von semantisch eindeutigen, syntax-unabhängigen, Geschäftsinformationen. Sie stelle Anleitungen für die korrekte Benennung und Kombination von CCs bereit. Zudem führe sie Wege ein, um kontext-spezifische Einschränkungen anzuwenden. Somit können Datenvorlagen auf die individuellen Anforderungen eines Nutzers beschränkt werden (vgl. [71]). Zur Erklärung verwenden *Schroth et al.* ein Beispiel, welches 3 verschiedene Arten von CCs beschreibt[71]:

- sogenannte *Aggregate Core Components* (ACC) bestehen aus zahlreichen anderen Komponenten.
- sogenannte *Basic Core Components* (BBC) sind Datenelemente, die nicht mehr in andere Komponenten zerlegt und in einer ACC aggregiert werden können.
- sogenannte *Association Core Components* (ASCC) können im Gegensatz zu BCC in weitere Komponenten zerlegt werden, und durch eine ACC aggregiert werden.

Durch die Spezifizierung von kontext-spezifischen Einschränkungen werden sogenannte Aggregate Business Information Entities - ABIEs, BIEs, sowie Associated BIEs definiert (vgl. [71]). Abbildung 2.4 auf Seite 21 ist der Arbeit von *Schroth et al.* entnommen und zeigt die Beziehung zwischen verschiedenen CCs und BIEs.

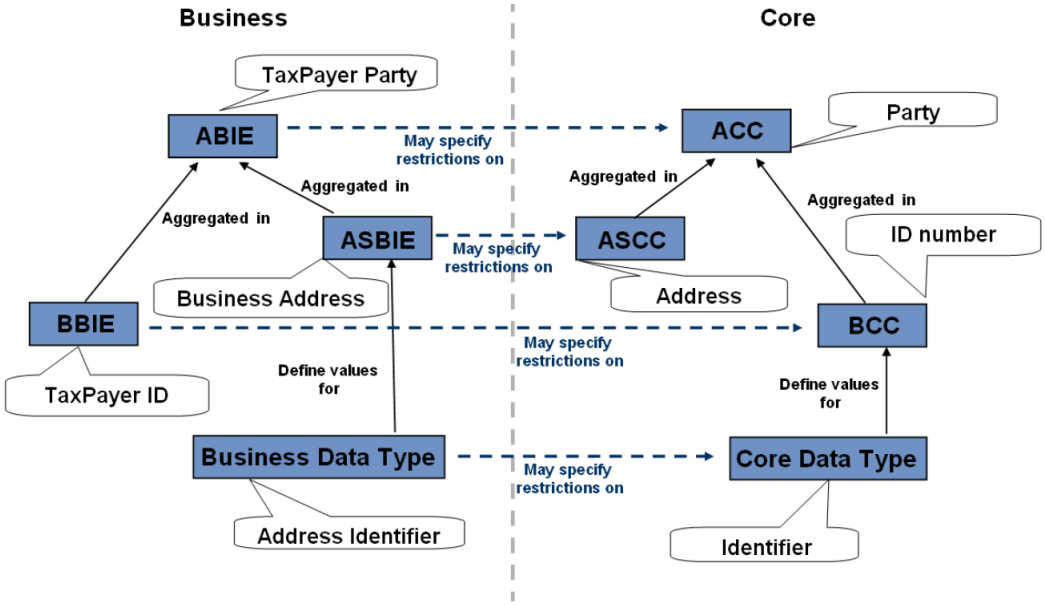


Abbildung 2.4: Abbildung 1 aus [71]: CCTS Meta Model Entities

2.2.2 Nutzung der Unified Modeling Language

Im letzten Kapitel wurde ein, durch *Lampathaki et al.* herausgearbeiteter, Ansatz zur Datenmodellierung vorgestellt. Das Ziel des vorgestellten Ansatzes ist die Modellierung von Daten, die im Rahmen einer typischen (Business-)Transaktion ausgetauscht, und gespeichert werden (vgl. [47]). Die vorgestellte CCTS ist für diesen Ansatz eine geeignete Teillösung. Denn sie stellt eine Methodik dar, die die Definition semantisch eindeutiger, syntax-unabhängiger, Geschäftsinformationen erlaubt (vgl. [71]). Zudem existiert eine Implementierung (UBL), die eine Bibliothek von XML-Schemata bereitstellt (vgl. [47] und [71]).
Im Gegensatz zu dem vorgestellten Ansatz kommen jedoch auch andere Lösungen in Frage, um die Datenmodelle auf der konzeptuellen und logischen Ebene zu erfassen. Dieses Kapitel soll näher auf die Verwendung der *Unified Modeling Language* - UML[15] - eingehen, die u.a. in einem Ansatz verwendet wird, der durch *Routledge et al.* vorgestellt wird (vgl. [67]).
Den Autoren nach existierten bereits (2002) zahlreiche Anwendungen, die es dem Nutzer erlaubten, ein XML-Schema zu visualisieren. Dabei verfolgten diese Anwendungen einen grafischen, sowie baum-basierten, Ansatz. Der baum-basierte Ansatz bringt nach *Routledge et al.* jedoch Nachteile mit sich. Der Nutzer sei immernoch mit Aspekten der Umsetzung konfrontiert. Ein besserer Ansatz sei die Herleitung der physischen Datenstrukturen, die aufgrund eines konzeptuellen Modells erfolgen kann (vgl. [67]).
Letztendlich stellen *Routledge et al.* einen solchen Ansatz vor, wobei die Autoren auf

[15]Die UML ist eine Modellierungssprache, die durch die Mitglieder der Object Management Group - OMG - definiert, und gepflegt wird. Vgl. *http://www.uml.org/* (zuletzt geprüft am *21.10.2012*).

einem Vorgehen aufbauen, das von *Bird et al.* vorgestellt wurde (vgl. [11][16]). Der Ansatz von *Routledge et al.* erlaubt eine Transformation von, der konzeptuellen Ebene zugeordneten, UML-Klassendiagrammen. Diese geschieht schrittweise, und endet mit der Erstellung eines XML-Schemas. Der Prozess schließt verschiedene Modellierungsebenen mit ein, die in Abbildung 2.5 auf Seite 22 abgebildet sind. Die Transformationsschritte sind mit Pfeilen gekennzeichnet und werden mittels eines Beispiels erklärt:

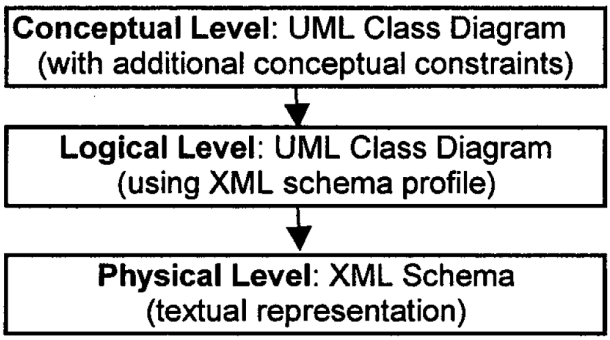


Abbildung 2.5: Abbildung 1 aus [67]: Three level design approach

1. Als erstes kann ein UML-Klassendiagramm erstellt werden, um den Gegenstandsbereich in Form von Objekten und Beziehungen aus der realen Welt zu beschreiben (konzeptuelle Ebene - vgl. [67]). *Routledge et al.* zeigen das in Abbildung 2.6a auf Seite 23 abgebildete Diagramm, welches ein Modell für die Bewertung von Studenten darstellt. Die Autoren merken weiterhin an, dass im Beispielmodell die Verwendung von nicht-standardisierten Annotationen notwendig war, um manche der allgemeinen konzeptuellen Einschränkungen zu repräsentieren (vgl. [67]). Dies kann somit ebenso für andere Modelle gelten. Dabei sollte die Modellierung, und die anschließende Überprüfung, unter Einbeziehung eines Domänenexperten erfolgen (vgl. [67]).

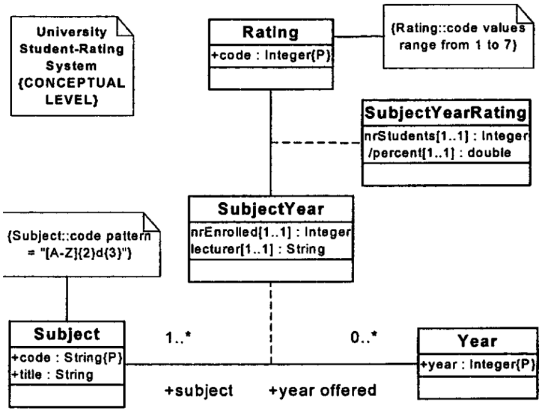
2. Im nächsten Schritt kann die Herleitung eines Diagramms erfolgen, dass der *logischen Ebene* zugeordnet wird. So muss sich ein Designer nicht mit den Aspekten der Umsetzung befassen (vgl. [67]). Laut *Routledge et al.* beschreibt ein solches die physischen Datenstrukturen, wobei dies in einer abstrakten, und oft auch grafischen, Weise geschieht. In dem, durch die Autoren, vorgestellten Ansatz ist dieses Modell eine direkte Repräsentation der XML-Schema-Datenstrukturen. Die Beschreibung als UML-Klassendiagramm kann aufgrund eines, durch die Autoren vorgestellten, UML Profils erfolgen[17]. *Routledge et al.* zeigen ebenso ein Diagramm, dass aufgrund des Beispiels hergeleitet wurde. Aufgrund der Größe, und der späteren Behandlung

[16] *Bird et al.* nutzen das sogenannte *Object Role Modelling*, welches eine grafische, sowie konzeptuelle, Modellierungstechnik ist (vgl. [11]).

[17] Nach *Routledge et al.* kann die UML durch die Nutzung von sogenannten Stereotypen erweitert werden. Diese sind wiederum Teil eines UML-Profils.

von XML-Schema (siehe Kapitel 2.3.1), soll dieses aber nicht gezeigt werden (vgl. [67]).

3. Im letzten Schritt kann ein XML-Schema erstellt werden, dass der *physischen Ebene* zugeordnet ist (vgl. [67]). Abbildung 2.6b auf Seite 23 zeigt eine gültige Instanz, die durch das XML-Schema erfasst wird, dass aufgrund des in Abbildung 2.6a auf Seite 23 gezeigten Diagramms hergeleitet wurde. Die Autoren führen die vollständige Instanz als Beispiel an.



(a) Abbildung 2 aus [67]: Example conceptial UML class diagram

```
<report>
  <subject code="CS 100">
    <title>Introduction to Computer
      Science</title>
    <year year="1982"> <subjectYear>
      <lecturer>P.L.Cook</lecturer>
      <nrEnrolled>200</nrEnrolled>
      <ratings code="7"> <
        subjectYearRating>
          <nrStudents>10</nrStudents>
          <percent>5.00</percent>
        </subjectYearRating> </ratings>
          ...
      </subjectYear> </year> </subject>
</report>
```

(b) Ein Auschnitt einer XML-Instanz, die dem hergeleiteten XML-Schema entspricht

Für *Routledge et al.* steht der Entwurf des XML-Schemas im Vordergrund, der mit Hilfe der vorgestellten Lösung auf eine visuelle Art erfolgen kann. Somit kann sich ein Datenmodellierer auf die konzeptuelle Modellierung der Domäne konzentrieren, und muss sich nicht mit der Umsetzung beschäftigen (vgl. [67]). *Lampathaki et al.* verbinden die verschiedenen Ebenen jedoch mit einer anderen Bedeutung (vgl. [47] sowie Kapitel 2.2.1). Im Ansatz von *Routledge et al.* wird auf konzeptueller Ebene ein UML-Klassendiagramm erstellt, während *Lampathaki et al.* die Verwendung von Ontologien vorschlagen. Zudem ist die physische Ebene nach *Routledge et al.* durch ein XML-Schema abgedeckt, während im Ansatz von *Lampathaki et al.* die Speicherung der Daten im Mittelpunkt steht. Letztendlich nehmen *Lampathaki et al.* eine Unterteilung in *Atoms*, *Objects* und *Collections* vor. Diese erlaubt, dass zuerst die Bedeutung der Informationen, dann deren Strukturierung, und schließlich deren Syntax definiert werden kann (vgl. Kapitel 2.2.1).
Durch das Erstellen eines konzeptuellen UML-Klassendiagramms werden diese Aspekte ebenso spezifiziert. Zum Beispiel fasst eine Klasse eine Menge von Attributen zusammen, die jeweils in ein XML-Element oder ein XML-Attribut überführt werden können (vgl. [67]). Gleichzeitig kann eine Klasse aber ebenso für konkrete Objekte stehen. Sie ist ein Teil der Beschreibung eines Gegenstandsbereich, da ein solcher durch ein UML-Klassendiagramm modelliert wird (vgl. *Routledge*[67]). D.h. sie ist mit einer Bedeutung

verbunden. Im Ansatz von *Lampathaki et al.* wird die Semantik jedoch mit der Hilfe von Ontologien definiert.
Beide Ansätze erlauben zuerst die Erstellung eines konzeptuellen Modells, welches die Struktur der konkreten Dokumente außen vor lässt. Nach *Routledge et al.* wird die Struktur der ausgetauschten Dokumente durch das hergeleitete Diagramm festgelegt, das mit der logischen Ebene in Verbindung gebracht wird. Für die Transformation können Optionen verwendet werden, jedoch gibt der Algorithmus eine Methodik für die Überführung vor. Somit ist die Struktur durch den Algorithmus gegeben (vgl. [67]). Die UBL gibt dagegen die Struktur von unterstützten Geschäftsdokumenten vor, und ist eine Implementierung der CCTS (vgl. [47] und [71]). Die Struktur der auszutauschenden Daten wird somit durch die bereitgestellten XML-Schemata beschrieben.
Neben der Arbeit von *Routledge et al.* existieren zahlreiche Beiträge, die sich mit einer Überführung von UML-Klassendiagrammen beschäftigen. Dies ist das Ergebnis eines Beitrags von *Dominguez et al.*. Die Autoren untersuchen in der Literatur zu findende Herangehensweisen. Dabei stellen sie fest, dass ausschließlich die Transformation von UML-Klassendiagrammen besprochen wird (vgl. [27]). Laut den Autoren ist dies verständlich. Die Transformation beliebiger UML-Modelle sei alles andere als trivial. Wichtige Unterschiede zwischen UML und XML-Schema seien ein Grund für die Komplexität. Dabei beziehen sich die Autoren auf *Salim et al.*, die diese Unterschiede beschreiben (vgl. [68])[18]:

- UML ist für die konzeptuelle Modellierung gedacht, die in der Design- und Entwicklungsphase (*design development stage*) stattfindet (vgl. [12]). Dieser Aspekt wird z.B. durch *Pagano und Brüggemann-Klein* angesprochen. Die Autoren behandeln ebenso die Transformation von UML-Klassendiagrammen. Ein hergeleitetes XML-Schema gibt jedoch die Repräsentation persistenter XML-Dokumente vor. Diese sollen mit Hilfe einer XML-Datenbank gespeichert werden (vgl. [60]). Dabei sei es von Vorteil, wenn die Datenmodellierung in die Systemmodellierung mit einbezogen wird. Bei der Nutzung von Schemasprachen (wie z.B. XML-Schema) sei eine Bindung an die verwendete Technologie gegeben, und die Modellierung von Dokumenten sei nicht in die Systemmodellierung integriert. Diese Nachteile würden auch dann bestehen, wenn die Integrierung von XML-Modellierungs-Werkzeugen erfolgt (vgl. [60]).

- XML-Schema ist eine text-basierte Sprache (vgl. [68]). Zudem beinhaltet XML-Schema mehr programmatische, sowie syntaktische Details, die generell nicht durch UML erfasst werden (vgl. [68]). Dies führt u.a. zu der bereits besprochenen Konsequenz: die Syntax muss durch einen Algorithmus vorgegeben werden. Ein UML-Klassendiagramm kann auf verschiedenste XML-Strukturen abgebildet werden (vgl. [67]).

Somit existieren verschiedene Vor- und Nachteile, die durch die Nutzung von UML-Klassendiagrammen entstehen:

[18] Die Unterschiede sollen nicht im Detail besprochen werden.

- Die Modellierung wird einfacher, da sie grafisch vollzogen werden kann.
- Gleichzeitig kann der Gegenstandsbereich an sich modelliert werden (vgl. [67]).
- Die Modellierung eines verarbeitenden Systems kann zusammen mit der Datenmodellierung erfolgen, da keine zusätzlichen Tools integriert werden müssen (dieser Aspekt wurde durch *Pagano und Brüggemann-Klein*[60] genannt).
- Es stellt sich die Frage, ob das konzeptuelle Diagramm als Grundlage dienen kann, wenn verarbeitende Programme modelliert werden sollen. Dies wäre ein weiterer Vorteil. Der Ansatz von *Routledge et al.* legt dies nahe. Allerdings müssen die nichtstandardisierten Annotationen berücksichtigt werden, die laut dem Autor benötigt werden (vgl. [67]).
- Die Struktur der XML-Instanzen wird jedoch durch den Algorithmus vorgegeben, der die Transformation durchführt (vgl. [67]). Somit besteht ein Vorteil, wenn die UBL verwendet wird. Denn die Struktur der Dokumente wird durch die bereitgestellten XML-Schemata vorgegeben (vgl. [71]) (diese können für eine Validierung genutzt werden).
- Der Ansatz von *Lampathaki et al.* hat den Vorteil eines einheitlichen Konzepts: die Einteilung in *Atoms*, *Objects* und *Collections* bezieht alle Ebenen mit ein (vgl. [47]). Dies ist ebenso an Abbildung 2.3 auf Seite 19 ersichtlich, und wird nochmals in Tabelle 1 auf Seite 25 zusammengefasst (wobei BIEs durch XML repräsentiert werden).
- Die UBL ist allerdings nur dann eine geeignete Lösung, wenn die im Rahmen einer Businesss-Transaktion auszutauschenden Nachrichten modelliert werden sollen (da die UBL auf Geschäftsinformationen aufbaut - vgl. [71]).

Ebene	Atom	Object	Collection	Modell / Sprache
konzeptuell	Attribut	Klasse	Ontologie	OWL
logisch	Basic BIE	Agreggate BIE	XML-Schema	XML
physisch	Attribut	Tupel	Tabellen	ERD

Tabelle 1: Die Unterteilung nach *Lampathaki et al.* betrifft alle Modellierungsebenen (vgl. Abbildung 2.3 auf Seite 19)

2.2.3 Ansätze zum Conformance Testing im Kontext der Standardisierung

In den letzten beiden Kapiteln wurde der Akt der Datenmodellierung beschrieben. Dieser ermöglicht die Erstellung verschiedener Datenmodelle, die in Ihrer Gesamtheit ein Datenformat beschreiben. Dabei wurden sowohl die auszutauschenden Informationen, ihre Bedeutung, und ihre physische Speicherung abgedeckt (vgl. Kapitel 2.2.1 sowie Kapitel 2.2.2). Die Erstellung solcher Datenmodelle geht jedoch mit den Interessen der

Beteiligten einher. Dies ist ein wichtiger Punkt, da die Modellierung nicht unbedingt durch eine Person erfolgen muss. Zudem ist die Datenmodellierung eine Aktivität, die im Zusammenhang mit anderen Aktivitäten betrachtet werden kann (vgl. [74] und [30]). Diese Aspekte werden z.B. durch *Söderström* herausgearbeitet. Die Autorin beschäftigt sich näher mit *B2B-Standards*. Laut *Söderström* enthält ein B2B Standard einen Leitfaden, der beschreibt, wie die Kommunikation, und die zwischen den Organisationen verschickten Informationen, strukturiert und verwaltet werden sollen. Der in dieser Arbeit wichtige Aspekt ist der „Nachrichteninhalt der Dokumente, die während einer Kommunikation empfangen sowie verschickt werden“ (vgl. [74]). Im Anwendungskontext der Arbeit kann der Austausch innerhalb von Geschäftsprozessen (vgl. Spezifikation VCD - S. 22-25 in [52]), und zwischen IT-Systemen und Individuen (vgl. Spezifikation IMS-CC - S. 11-18 in [64]) stattfinden[19].
Laut *Söderström* durchläuft ein Standard bestimmte Phasen und Aktivitäten. Dabei wird üblicherweise der Begriff des Lebenszyklus verwendet, um die verschiedenen Phasen für ein Produkt oder einen Prozess zu benennen (vgl. [74]). Die Autorin untersucht 7 verschiedene Modelle von Lebenszyklen, die in der Literatur vorgestellt wurden. Diese werden verglichen, und ein generelles Model abstrahiert. Am Ende wird dieses um weitere Aktivitäten erweitert. (vgl. [74]).
Das hergeleitete Modell ist in Abbildung 2.6 auf Seite 27 abgebildet. Diese Abbildung wurde allerdings einer nachfolgenden Arbeit von *Söderström* entnommen, die die gezeigten Aktivitäten mit verschiedenen Stakeholdern in Verbindung bringt (vgl. [75]). Die Aktivitäten sollen im folgenden kurz vorgestellt werden[74]:

- In der „Initiation Phase“ wird „die Idee des Standards geboren und die generellen Konditionen für die kommenden Aktivitäten festgesetzt“.
- Die „*Development Phase* beinhaltet die Erstellung der Standard Spezifikationen, die die Essenz der Standards sind“.
- In der Phase „*Develop Product(s)* werden Produkte auf der Basis von einem oder mehreren Standards erstellt“. Allerdings sind diese keine notwendige Anforderung für die Implementierung eines Standards.
- Die Phase „Implementation“ umfasst die Implementierung von Standards oder Standard-basierten Produkten, die innerhalb der Organisationen vollzogen wird.
- Die Phase „Use“ beinhaltet die „tatsächliche Nutzung von Standards und den zugehörigen Produkten“.
- Die Phase „Feedback“ schließt das Feedback von Nutzern ein. Dieses basiert auf den Erfahrungen der Nutzer und kann als Beitrag für sich kontinuierlich entwickelnde Standards dienen.

[19]Da *Söderström* B2B Standards betrachtet, wird im Verlauf des Kapitels ein Modell aus dem E-Learning Bereich vorgestellt.

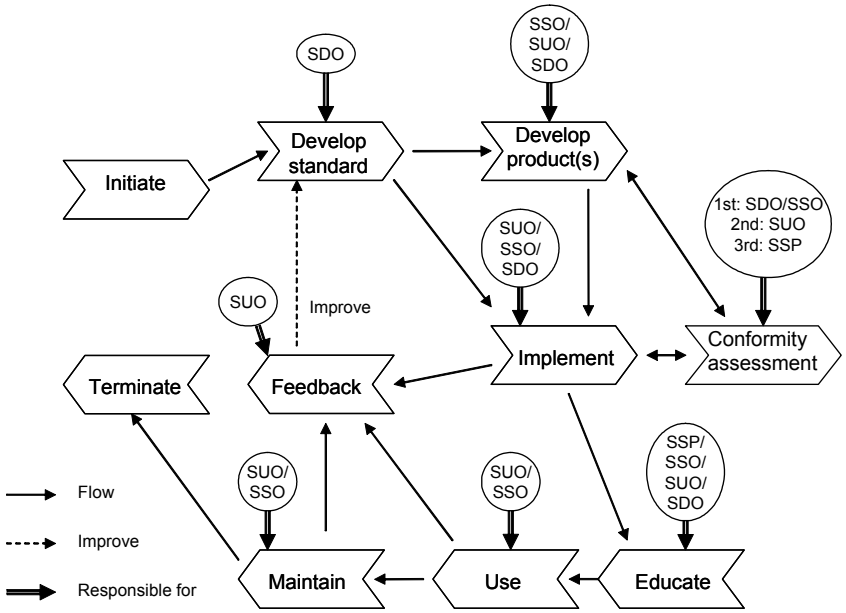


Abbildung 2.6: Abbildung 1 aus [75]: Summary of connections between process and roles

- Die Phase „Education“ adressiert Wissensprobleme der Nutzer. Zum Beispiel wenn diese nicht wissen, wie Standards funktionieren.
- In der Phase „Maintenance“ steht die Aufrechterhaltung der Operation eines Standards / Systems im Mittelpunkt.
- In der Phase „Conformity Assessment“ steht die gleichnamige Aktivität im Vordergrund, die mittels verschiedener Techniken vollzogen werden kann.

Da das vorgestellte Modell sehr umfassend ist, sollen nur jene Aktivitäten betrachtet werden, die im engen Zusammenhang mit der Thematik stehen. Im Detail sind dies die folgenden:

- Die *Entwicklung eines Standards*, da diese den Entwurf von Datenmodellen mit einschließen kann[20]. Auch wenn bereits relevante Standards existieren, kann die Anpassung, oder die Wiederverwendung, von Standards zu einem neuen Standard, oder einer neuen Spezifikation führen (vgl. [30] und [29] - sowie Kapitel 2.4.1).
- Die *Implementierung eines Standards*, da eine Implementierung auf Konformität getestet werden kann. Dies gilt ebenso für Produkte, die aufgrund von Standards erstellt werden (vgl. [74], sowie Kapitel 2.3.2). Die Differenzierung der beiden Phasen findet ebenso Verwendung bei der VCD-Spezifikation: im Rahmen der Spezifikation

[20] Vgl. Definition eines B2B Standards nach Söderström. Die Datenmodellierung betrifft u.a. die Strukturierung der auszutauschenden Informationen, die schließlich in Dokumente eingebettet werden müssen.

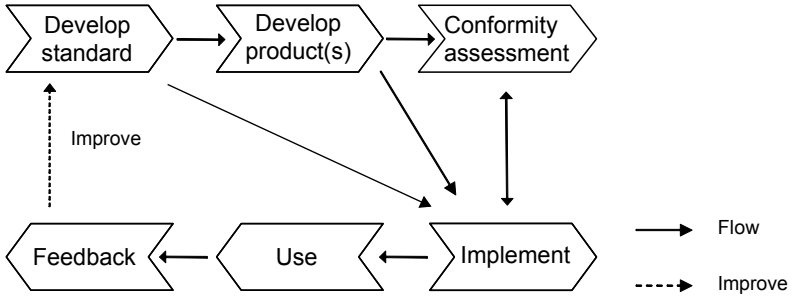
wurden Referenzimplementierungen entwickelt, die für den Aufbau einer konformen Infrastruktur, und konkreter Implementierungen genutzt werden können (vgl. S. 42 [52]). Zudem besteht eine Wechselwirkung zwischen der Phase des Conformity Assessments und der Phase der Implementierung (vgl. Abbildung 2.6 auf Seite 27).

- Die *Aktivität des Conformity Assessments*, da diese das Testen von Implementierungen beinhalten kann (vgl. [74]). Ein Teil solcher Tests kann u.U. mit der in dieser Arbeit vorgestellten Methodik ausgeführt werden. Genauer gesagt können Dokumente getestet werden, die durch, an einem Austausch beteiligte, IKT-Systeme erstellt wurden. Das Testverfahren muss den Typ der Dokumente allerdings unterstützten (vgl. Kapitel 4).

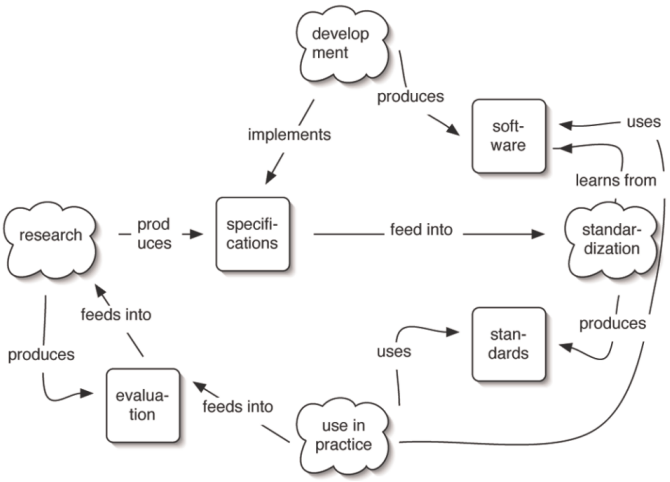
- Die *Nutzung einer Implementierung durch verschiedene Nutzer*. Nutzer können in der darauffolgenden Phase ein Feedback geben. Ein solches kann zu Änderungen am Standard, den Implementierungen, sowie an den, zu testenden, Anforderungen führen (vgl. [74]). Ein Feedback kann ebenso auf der Aktivität des Confirmity Assessments basieren: die Nutzer selbst können Tests ausführen und Ihre Erfahrungen weitergeben. Dies wird bereits durch Abbildung 2.6 auf Seite 27 erfasst. Laut *Söderström* können *Standard User Organisations* ebenso an der Phase des Conformity Assessments interessiert sein (vgl. [75] sowie Abbildung 2.6 auf Seite 27).

Im folgenden soll der Lebenszyklus besprochen werden, der nur die genannten Phasen enthält. Dieser ist in Abbildung 2.7a auf Seite 29 abgebildet, und soll mit einem, durch *Duval und Verbert*[30] geschilderten, Szenario verglichen werden.
Die Autoren beschäftigen sich mit Standards im E-Learning Bereich, und betonen den Unterschied zwischen Standard und Forschung. Dazu wird das in Abbildung 2.7b auf Seite 29 gezeigte Szenario verwendet, welches, wie das Modell von *Söderström*, verschiedene Aktivitäten bzgl. eines Standards / einer Spezifikation, und deren Zusammenhänge aufzeigt[30].
Demnach können Spezifikationen ein Produkt von Forschung sein, und können wiederum durch Software Artefakte implementiert werden. Die durch die Entwicklung, und die Nutzung der Implementierungen resultierende, Erfahrung kann zu einer neuen Version der Spezifikation führen. An einem bestimmten Punkt können die Spezifikationen als Eingabe für die Standardisierung dienen (vgl. [30]). Die Abbildung zeigt mehrere Aktivitäten auf, die ebenfalls im Modell von *Söderström* vorkommen:

- Die Phase *Develop Standards* (*Söderström*) kann mit der Aktivität *standardization* (*Duval und Verbert*) verglichen werden. Im Szenario von *Duval und Verbert* können während der Implementierung gesammelte Erfahrungen berücksichtigt werden, die jedoch als erstes in eine neue Version der Spezifikation einfließen. Der Einfluß auf den Standardisierungsprozess erfolgt dann über einen weiteren Schritt (vgl. [30]). Sowohl *Duval und Verbert* als auch *Söderström* berücksichtigen das Feedback, das durch die Nutzer der Implementierungen erfolgen kann (vgl. [74] und [30]).



(a) Ein Ausschnitt des Lifecycle Modells nach Söderström (2004)



(b) Abbildung 3 aus [30]: Standards are not research (2008)

Abbildung 2.7: Ein Vergleich von Aktivitäten, die mit einem Standard und dessen Entwicklung in Verbindung stehen

- Die Phasen *Develop Products* und *Implement* (*Söderström*) können mit der Aktivität *development* (*Duval und Verbert*) verglichen werden. Dabei fassen *Duval und Verbert* Standard-basierte Produkte und Implementierungen zusammen, indem sie den Begriff des *Software-Artefakts* verwenden. Allerdings können ebenso Spezifikationen implementiert werden, die später für die Standardisierung verwendet werden (vgl. [30]).

- Die Phase *Use* (*Söderström*) kann mit der Aktivität *use in practice* (*Duval und Verbert*) verglichen werden. *Duval und Verbert* ziehen die direkte Nutzung des Standards in Erwägung (vgl. Abbildung 2.7b auf Seite 29). *Söderström* betont, dass diese Phase nur durch wenige der untersuchten Arbeiten erfasst wird. Zudem würden diese meist nicht genau definieren, was unter der Nutzung eines Standards zu verstehen ist. Im abstrahierten Modell wird auf die Nutzung der Implementierungen eingegangen (vgl. [74]). Das Feedback der Nutzer wird durch beide Arbeiten

berücksichtigt (vgl. Aktivität *evaluation* in Abbildung 2.7b auf Seite 29, sowie [74]).

- Die Phase *Feedback* (*Söderström*) ist durch die Auswirkung des Feedbacks (*Duval und Verbert*) erfasst. Dieses wird in Evaluationen berücksichtigt und kann Einfluß auf die Forschung nehmen, wodurch neue Versionen einer Spezifikation entstehen können (vgl. Abbildung 2.7b auf Seite 29).

Ein wesentlicher Unterschied zwischen dem Modell nach *Söderström* und dem Szenario von *Duval und Verbert* besteht in der Behandlung von Spezifikationen und Standards. Während *Söderström* ausschließlich Standards behandelt, beschreiben *Duval und Verbert* den Zusammenhang zwischen Standards und Spezifikationen. Zudem fehlt im Szenario von *Duval und Verbert* die Aktivität des *Conformity Assessments* (vgl. Abbildung 2.7b auf Seite 29 und Abbildung 2.7a auf Seite 29, sowie [74] und [30]).
Diese soll allerdings nicht alleine im Mittelpunkt der Betrachtung stehen. Nach *Söderström* hilft ein Lebenszyklus-Modell dabei Standards, und deren Bezug zur Umgebung, zu verstehen. Laut der Autorin ist die Auswirkung von Aktivitäten einer Phase auf die Aktivitäten einer anderen Phase ein Beispiel (vgl. [74]). Somit ist die Wechselwirkung zwischen der Aktivität des Conformity Assessments und den anderen, durch die in Abbildung 2.7a auf Seite 29 zusammengefassten, Aktivitäten von Belang. Diese wurde zwar zum Teil erarbeitet, jedoch spielen ebenso verschiedene Stakeholder eine Rolle, die unterschiedliche Interessen verfolgen, und somit andere Sichtweisen auf diese Aktivitäten haben können (vgl. [75]). Daher soll näher auf die durch *Söderström* vorgestellten Stakeholder eingegangen werden, die mit einer der in Abbildung 2.7a auf Seite 29 gezeigten Phasen in Verbindung stehen[75]:

- *Standards developing organizations* - SDOs - können in die Phasen *Develop product(s)*, *Conformity assessment* und *Implement* involviert sein. Sie sind für die Entwicklung eines Standards verantwortlich. In der Phase *Develop Products* können Referenzimplementierungen von Interesse sein. Diese können laut *Söderström* eine Hilfe sein, um die ordnungsgemäße Funktion von Standards zu bestätigen. Zudem können sie ebenso Aktivitäten der Phase *Conformity Assessment* durchführen.

- *Standards software organizations* - SSOs - können in die Phasen *Develop products*, *Conformity assessment*, *Implement* und *Use* involviert sein. Sie sind meist für Aktivitäten der Phase *Develop Products* verantwortlich. In der Phase *Conformity assessment* können Standard-basierte Produkte getestet werden. Werden Implementierungen für *Standard User Organizations* - SUOs - angeboten, so werden Aktivitäten der Phase *Implement* übernommen. Schließlich können Standards in der täglichen Arbeit genutzt werden, wodurch eine SSO ebenso zu einer SUO wird.

- *Standard user organizations* - SUOs - können in die Phasen *Develop product(s)*, *Conformity assessment*, *Implement* und *Use* involviert sein. Fortgeschrittene SUOs können betriebsinterne Produkte erstellen. Durch die eigene Umsetzung der Standard-Spezifikationen können sie an der Implementierungs-Phase teilnehmen. Ebenso können Standards und/oder die erworbenen Produkte getestet werden.

Durch die Möglichkeit der eigenen Umsetzung kommt CA ebenso für die selbst erstellten Produkte und Implementierungen in Frage. Zuletzt spielen SUOs eine entscheidende Rolle, da sie die Standards in Ihrer alltäglichen Arbeit verwenden. Mit Ihrer praktischen Erfahrung können sie eine Rückmeldung dazu geben, ob ein Standard / Produkt in der Praxis funktioniert.

- *Standard Service Providers* können letztlich die Aktivitäten des *Conformity Assessments* übernehmen. Ein SSP sollte nicht in den Standardisierungsprozess involviert sein, um die Objektivität zu sicher zu stellen.

Somit können sämtliche, in der Abbildung aufgeführten, Stakeholder ein Interesse an *Conformity Assessment* haben. Wer letztendlich die Tests übernimmt, hängt eng damit zusammen, ob die eigentliche Implementierung aufgeteilt wird. Werden Referenzimplementierungen durch eine SDO erstellt, greift CA bereits bei der Erstellung eines Standards. Bei der Einführung standard-basierter Produkte ergeben sich Einsatzgebiete für SSOs. Letztlich können die Nutzer selbst eine Implementierung in Angriff nehmen, und CA durchführen, falls die dazu notwendigen Mittel bereit gestellt werden (vgl. letzte Aufzählung und [75]).
Änderungen an einem Standard können, wie bereits herausgearbeitet, durch das Feedback von SUOs und SSOs (vgl. [75]), oder aufgrund der, während der Implementierung gesammelten, Erfahrungen (vgl. [30]) erfolgen. Diese Änderungen können somit alle Stakeholder betreffen, da sich die Änderung eines Standards ebenso auf die zu testenden Anforderungen auswirken können. Werden die Tests jedoch an einer Stelle durchgeführt, wirken sich die Änderungen nur auf eine Organisation aus. Die Auslagerung des CAs kann wie besprochen durch SSP geschehen, die möglichst unabhängig sein sollten, um die notwendige Objektivität zu gewährleisten (vgl. [75]).
Wird ein Bezug zu Datenformaten hergestellt, so ist die Datenmodellierung interessant, die zu Diesem geführt hat. Eine Implementierung muss Nachrichten in korrekter Weise verstehen und verarbeiten können, wenn die Interoperabilität verschiedener IKT-Systeme gefordert ist (vgl. Kapitel 2.1.1 sowie [2]). Ein B2B Standard gibt nach *Söderström* die auszutauschenden Informationen, sowie deren Struktur, vor (vgl. [74]). Dabei kann es durchaus vorkommen, dass nur eine Teilmenge der möglichen Optionen implementiert wird, da der Standard jeden erdenklichen Fall abdeckt (vgl. [29]). Solche Aspekte sollen in Kapitel 2.4.1 behandelt werden, da die Erstellung von *Applikationsprofilen* u.U. eine Lösung sein kann (vgl. [29] und Kapitel 2.4.1).
Weitere interessante Einwände sind bei *Jakobs* zu finden. So ist die Koordinierung von Standardisierungsaktivitäten nicht immer erwünscht[21], oder der Einfluß nicht immer demokratisch geregelt. Solche und andere Aspekte werden durch eine umfassende Sammlung von Attributen erfasst, die Eigenschaften einer *Standard Setting Body* - SSB - beschreiben (vgl. [74]). Auch *Jakobs* hält die Aktivität des CA für einen wichtigen Schritt. Der Autor hebt hervor, dass interoperable Implementierungen eines Standards, und der Beweis der Konformität dieser, ebenso wichtig sind, wie der Basis Standard an sich.

[21] Dies betrifft laut Jakobs nur sogenannte *Consortia*. Im Gegensatz dazu sind die Aktivitäten von SDOs in einer hohen Weise koordiniert (vgl. [43]).

Letztlich identifiziert der Autor verschiedene Interessen, die durch die am Standardisierungsprozess beteiligten Firmen verfolgt werden können. Unter den Beteiligten finden sich ebenso größere Nutzer und Hersteller, die an den durch *Söderström* geschilderten Aktivitäten Teil haben können (zusammengefasst aus [43]). Nutzer sind somit nicht unbedingt reine Konsumenten von Produkten, streben allerdings nach *Jakobs* keine Beeinflussung der Entwicklung an. Im Vordergrund stehen u.a. die Vertretung ähnlicher Interessen, sowie der Zugang zu den relevanten Informationen (vgl. [43]).

2.3 Beschreibung und Verifizierung von Datenformaten

Im letzten Kapitel wurde näher auf das Thema der Datenmodellierung eingegangen. Die vorgestellten Ansätze unterteilen den Prozess, indem sie verschiedene Abstraktionsebenen einführen. Als erstes wurde der Ansatz von *Lampathaki et al.* besprochen. Dieser konzentriert sich auf Daten, die während einer typischen Business-Transaktion ausgetauscht werden. Für die Erfassung der Modelle werden Ontologien, XML-Schemata und ER-Diagramme empfohlen. Verwendung findet ebenso die CCTS, die in der UBL eine Implementierung findet (vgl. Kapitel 2.2.1 und [47]). Der Ansatz von *Routledge et al.* baut jedoch auf UML-Klassendiagrammen auf, die den Gegenstandsbereich einer Domäne modellieren. Aus einem konzeptuellen Diagramm kann im Anschluß ein weiteres Diagramm hergeleitet werden, dass die XML-Schema Strukturen widerspiegelt. Dieses kann letztendlich für die Herleitung des eigentlichen XML-Schemas verwendet werden (vgl. [67] und Kapitel 2.2.2).
In beiden Fällen wird ein XML-Schema verwendet, um das Format der konkreten Dokumente vorzugeben. Auch die in Kapitel 2.4.1 vorgestellten Applikationsprofile basieren auf einem XML-Schema. Diese stellen eine Grundlage für das in Kapitel 4 vorgestellte Testverfahren dar, welches die Validierung konkreter Dokumente erlaubt. Ein Applikationsprofil erlaubt die Nutzung verschiedener Schemasprachen (wie XML-Schema). D.h. es können mehrere formale Spezifikationen verwendet werden, um die Anforderungen an einzelne Dokumente festzulegen (vgl. Kapitel 4.1). Der Austausch der konkreten XML-Dokumente wird nicht selten durch Use-Cases (Beispiel IMS-CC - vgl. S. 11-18 in [64]) oder Geschäftssprozesse (Beispiel VCD - vgl. S. 22-25 in [52]) beschrieben. Dabei spiegelt die Syntax der Dokumente auch Anforderungen wieder, die auf der konzeptuellen Ebene festgelegt wurden. Wenn die Beschreibung eines Kunden z.B. eine Adresse aufweisen muss, so wirkt sich dies ebenso auf die Syntax aus.
Die Konzepte des vorzustellenden Testverfahrens können mit einem Teil der, durch das *Testing Framework for Interoperability Test Bed*s - GITB, eingeführten Konzepte verglichen werden (vgl. Kapitel 4.2). Das zu beschreibende Testverfahren wurde jedoch unabhängig von diesem entwickelt. Das GITB beschäftigt sich mit Tests, die aufgrund von eBusiness-Spezifikationen erfolgen sollen (vgl. [42]). Das später zu verwendende Testsystem wurde jedoch für das Testen von Common Cartridges (IMS-CC) verwendet (vgl. [24]). Dies legt nahe, dass es gemeinsame Probleme gibt, die sowohl im E-Learning, als auch im E-Procurement Bereich, zu finden sind.
Die nächsten Kapitel erläutern die beschriebenen Zusammenhänge im Detail. Ihr Inhalt soll im folgenden zusammengefasst werden:

- Kapitel 2.3.1 wird verschiedene Schemasprachen vorstellen. Allerdings sollen die Sprachen XML-Schema und Schematron bevorzugt behandelt werden. Denn diese kommen bei den zu besprechenden Spezifikationen IMS-CC und VCD zum Einsatz (vgl. [64] und S. 46 in [13]).

- Kapitel 2.3.2 wird grundlegende Begriffe erklären, die in einem engen Zusammenhang mit dem später vorzustellenden Testverfahren (vgl. Kapitel 4) stehen.

- Kapitel 2.3.3 wird näher auf Zertifikate eingehen. Verarbeitende Programme sollten konforme Daten in vorhergesehener Weise verarbeiten können, wenn die Interoperabilität zwischen diesen gewünscht ist (vgl. Kapitel 2.1.1 und [2]). Durch entsprechende Tests kann die Existenz von bestimmten Fehlern überprüft (vgl. [50]), und das Vertrauen in eine Implementierung gestärkt werden (vgl. [33] und [34]). Dabei kann ein unabhängiges Testlabor mit den Tests beauftragt werden. Eine Zertifizierungsstelle kann auf die Ergebnisse zurückgreifen und ein Zertifikat ausstellen. (vgl. [34] und [33]).

2.3.1 Erfassung der Anforderungen durch formale Sprachen

Die formale Beschreibung der syntaktischen Anforderungen ist ein wichtiger Schritt in der Datenmodellierung. Sie erlaubt u.U. die Validierung konkreter Dokumente und Nachrichten. Im Fall des IMS-CC-Formats und des VCD werden solche u.a. durch Implementierungen erstellt, die dann auf die Einhaltung des Formats hin getestet werden können (vgl. Komponente *VCD Builder* - S. 50 in [52] - sowie Begriff des *Authoring Tools*[22]). Die vorgestellten Ansätze zur Datenmodellierung schlugen die Verwendung der Sprache XML-Schema vor, um das Format der Instanzen festzuhalten (vgl. [47] und [67]). Dabei wurde die Erstellung, bzw. die Herleitung, des XML-Schemas der logischen (*Lampathaki et al.*[47]), sowie der physischen Ebene (*Routledge et al.*[67]), zugeordnet. *Routledge et al.* verwendeten auf der logischen Ebene jedoch ein UML-Klassendiagramm, welches die Struktur des XML-Schemas genau widerspiegelt (vgl. [67]). Somit lag der Unterschied letztendlich in der Repräsentation (UML - XML). Eine erste Einführung der Syntax findet somit bei beiden Ansätzen auf der logischen Ebene statt.
Neben XML-Schema existieren jedoch viele andere Schemasprachen, die für die Beschreibung gültiger XML-Dokumente genutzt werden können. *Klarlund et al.*, *Lee und Chu*, und *Coen et al.* stellen verschiedene Schemasprachen vor (vgl. [48][16][57]). Dabei ist die Existenz mehrerer Schemasprachen durchaus sinnvoll. Denn laut *Klarlund et al.* müssen zahlreiche, sowie grundlegende, Entscheidungen getroffen werden, die die Gestaltung einer Schemasprache betreffen (vgl. [57]). *Lee und Chu* merken zudem an, dass die Philosophie, die hinter dem Entwurf einer jeden, durch die Autoren betrachteten, Sprache steht, unterschiedlich ist (vgl. [48]). *Coen et al.* verweisen gar darauf, dass jede Sprache auf eine unterschiedliche Art von Validierungs-Anforderungen zugeschnitten ist. Zudem stelle jede Sprache eine große Menge von Features bereit, die bei den anderen Sprachen oftmals nicht zu finden seien (vgl. [16]). Dabei wird der Begriff der Validierung durch *Coen et al.* wie folgt definiert[16]:

> „Validating a document is, in XML parlance, the process of verifying whether an XML document accords to a set of structural and content rules expressed in one of the many schema languages proposed by a number of different organizations and individuals.“

[22] Auf der, zum ASPECT-Projekt gehörenden, Seite *http://www.aspect-project.org/node/71#2* finden sich meherere Beispiele für solche Tools (zuletzt geprüft am *25.10.2012*).

Gleichzeitig wird zwischen zwei verschiedenen Typen von Schemasprachen unterschieden[16]:

- *Grammatik-basierte Sprachen*, mit deren Hilfe eine formale, kontextfreie, Grammatik spezifiziert werden kann, die auf Top-Down Produktionsregeln basiert. Zu dieser Kategorie zählen XML-Schema, Relax NG[23] und DTDs[24].
- *Regel-basierte Sprachen*, mit deren Hilfe eine Liste von Regeln angegeben werden kann. Diese müssen für ein Dokument erfüllt sein. Hierbei wird nochmals zwischen *offenen* sowie *geschlossenen* Spezifikationen unterschieden[25]. Dieser Sachverhalt wird durch *Klarlund et al.* mit den Begriffen *Closed* bzw. *Open World Assumption* näher beschrieben: bei der ersten Annahme dürfen keine Inhalte vorkommen, deren Erscheinen nicht explizit erlaubt wurde. Bei letzterer Annahme verhält es sich entgegen gesetzt (vgl. [57]). Diese Kategorie beinhaltet nach *Coen et al.* die Sprachen xlinkit[26] und Schematron[27].

Die Autoren *Lee und Chu* verfeinern diese Einteilung nochmals. Sie zeigen das in Abbildung 2.8 auf Seite 35 gezeigte Diagramm, um 6 verschiedene Sprachen näher einordnen zu können[48]:

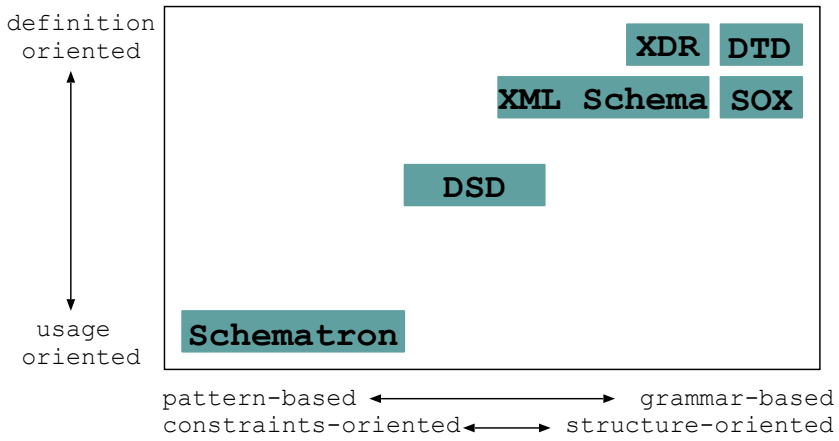


Abbildung 2.8: Abbildung 1 aus [48]: XML schema languages classification

- Die *grammatik-basierte* Gruppe von Sprachen hat den Autoren nach Vorteile, wenn XML-Anfragen erstellt werden sollen. Denn sowohl die Struktur, als auch die Definition des Schemas, seien in diesem Fall bekannt.

[23] Die Spezifikation ist unter *http://relaxng.org/* zu finden (zuletzt geprüft am *23.10.2012*).

[24] Die Beschreibung gültiger DTDs (Document Type Definitions) ist Teil der W3C Recommendation, die die Sprache XML vorstellt: *http://www.w3.org/TR/REC-xml/#sec-prolog-dtd* (zuletzt geprüft am *23.10.2012*).

[25] *Coen et al.* beziehen sich für diese Aussagen auf *N. Walsh and J. Cowan*: *http://nwalsh.com/xml2001/schematownhall/slides/* (zuletzt geprüft am *23.10.2012*).

[26] Xlinkit ist genau genommen ein leicht gewichtiger Applikations-Server, der die regel-basierte Erzeugung von Links bereitstellt, und die Konsistenz verteilter Web-Inhalte überprüft (vgl. [56]).

[27] Die Homepage von Schematron ist unter *http://code.google.com/p/schematron/* zu finden. Laut dieser ist Schematron ein ISO Standard (zuletzt geprüft am *23.10.2012*).

- Dagegen könne die *pattern-basierte* Gruppe mit einer starken Ausdruckskraft aufwarten, wenn es um Einschränkungen (Constraints) gehe.

Demnnach ist *XML-Schema* eine *grammatik-basierte*, sowie *definitions-* und *strukturorientierte* Sprache. *Schematron* ist dagegen eine *pattern-basierte*, *validierungs-* und *constraints-orientierte* Sprache (vgl. [48] und Abbildung 2.8 auf Seite 35).
Dabei ist XML-Schema laut *Coen et al.* eine W3C Recommendation[28], die darauf abzielte, die sogenannten DTDs zu ersetzen. Ein möglicher Grund wird durch den Autor selbst angeführt. Demnach sei die Kontrolle über Attribute und Datenelemente sehr beschränkt und qualifizierte Elemente würden nicht unterstützt (vgl. [16]). *Klarlund et al.* führen den letzten Punkt näher aus. Der erlaubte Inhalt sei für Elemente mit demselben Namen stets gleich, unabhängig davon, wo diese im Dokument verwendet werden (vgl. [57]). Diese Einschränkung besteht in XML-Schema nicht. Laut dem Autor erfolgt eine Trennung von Tags und Typen (vgl. [57]).
Zudem unterscheidet sich ein XML-Schema in weiteren Punkten von einer DTD. Laut *Coen et al.* ist ein XML-Schema, im Gegensatz zu einer DTD, in XML verfasst. Dies würde die Lesbarkeit und die Knappheit der Sprache verschlechtern, die Fexibilität und die automatische Verarbeitung jedoch signifikant verbessern (vgl. [16]). Zudem führt die Sprache sogenannte Namespaces ein, um Namen voneinander unterscheiden zu können, wenn diese der Metasprache selbst, oder der zu definierenden Sprache angehören (vgl. [57]).
Dabei besteht der größte Unterschied in den unterstützten Typen. Nach *Coen et al.* stellen DTDs, bis auf *Strings*, *leerzeichen-freie Strings* sowie *Aufzählungen von Strings*, keine Datentypen zur Verfügung, die „der Rede wert sind“. Zudem biete XML-Schema die besten eingebauten Datentypen, sowie die durchdachtesten Mechanismen, wenn es um benutzer-definierte Typen geht[29] (vgl. [16]). Dabei unterscheidet XML-Schema laut *Coen et al.* zwischen

- einfachen Typen (Strings mit zahlreichen Constraints), und
- komplexen Typen (Mark-up Subtrukturen eines XML Dokuments, die Elemente, Attribute und Textknoten beinhalten)[16].

Dabei lägen die wirklichen Stärken von XML-Schema in der ergiebigen Menge eingebauter einfacher Typen, und der Zahl der Facetten, die auf diese angewendet werden können (vgl. [16]). Die Autoren merken an, dass Schema-Autoren dazu ermutigt werden, neue Typen von existierenden herzuleiten, und deren Werte durch eine der vielen bereitgestellten Facetten einzuschränken (vgl. [16]). Dabei kann der Typ eines Elements laut *Klarlund et al.* vom Kontextpfad - einer Folge von Vorfahren - abhängig gemacht werden (vgl. [57]).
Die Zuordnung der Typinformationen erfolgt laut *Coen et al.* während der Validierung. Laut dem Autor definiert XML-Schema das sogenannte *Post Schema Validation Infoset* -

[28] Vgl. *http://www.w3.org/XML/Schema.html* (zuletzt geprüft am *24.10.2012*).
[29] Es wurden DTDs, sowie die Sprachen XML-Schema, Relax NG, DSD (diese Sprache wird durch *Klarlund et al.*[44] näher besprochen), Schematron, und xlinkit betrachtet (vgl. [16]).

PSVI, die zusätzlichen Informationen, die den Knoten eines XML-Dokuments durch eine Validierung hinzufügt werden (vgl. [16]). Um eine mögliche Verwendung dieser Informationen zu erläutern, soll das *Document Object Model* - DOM - vorgestellt werden. *Klarlund et al.* gehen näher auf dieses ein. Für einen Programmierer sei ein XML-Knoten ein Objekt, welches physisch ein Teil von zugeteiltem Speicher sei. Innerhalb des DOM würde ein solches verschiedene Zeiger beinhalten, so dass die Kinder, Attribute und Namespace-Knoten erreicht werden könnten (vgl. [57]). Nach einer erfolgreichen Validierung kann, je nach Implementierung, auf die zusätzlichen Informationen zugegriffen werden. Ein Beispiel ist die von der Apache Foundation bereitgestellte Bibliothek *Xerces 2*, die für das Parsen, die Validierung, und die Manipulation von XML-Dokumenten genutzt werden kann[30]. Die Java-Version des Parses implementiert die XML-Schema-API[31], die eine Programmierschnittstelle, sowie die notwendigen Schritte vorgibt, um an die zusätzlichen Informationen zu gelangen[32]. Ein Beispiel wäre das DOM-Interface *dom.Element*. Der XML-Schema-API nach sollte eine Klasse, die dieses Interface implementiert, ebenso das Interface *ElementPSVI* implementieren. So lassen sich die zusätzlichen Informationen nach einem Cast erfragen.
Somit bietet XML-Schema eine gute Grundlage, um die Struktur eines XML-Dokumentes festzuhalten. Das Schema kann ebenfalls für die Validierung konkreter Dokumente genutzt werden. Die Anforderungen des VCDs umfassen jedoch *co-occurence constraints* - auch *Co-Constraints* genannt (vgl. Anforderungen des VCD - Kapitel 3.4.3 - z.B. *VCD-02-R004*). *Co-Constraints* ermöglichen laut *Coen et al.*, Einschränkungen für Elemente und Attribute zu definieren, die wiederum auf der Existenz, oder dem Inhalt, anderer Elemente und Attribute basieren. Nach *Coen et al.* unterstützt XML-Schema diese Art von Einschränkungen jedoch nicht (vgl. [16]). Der Autor berichtet, dass *Co-Constraints* häufig, und in vielen Dokumenttypen genutzt werden. Aufgrund der fehlenden Unterstützung würden diese in natürlicher Sprache formuliert und durch spezielle Code-Module der relevanten Programme überprüft (vgl. [16]). Somit besteht auch im allgemeinen ein hoher Bedarf für solche Einschränkungen.
Eine durch *Coen et al.* vorgeschlagene Lösung ist die gleichzeitige Verwendung von Schematron und XML-Schema. Den Autoren nach ist Schematron eine regel-basierte Sprache, die hauptsächlich für die Überprüfung von Co-Constraints (in XML-Dokumenten) genutzt würde (vgl. [16]). *Lee und Chu* fassen die Vorteile der beiden Sprachen kurz zusammen[48]:

- *XML-Schema* unterstützt Features für Schema-Datentypen / -Struktur in vollem Maße, während
- *Schematron* eine sehr flexible und pattern-basierte Sprache bereitstellt, die die detaillierte Semantik eines Schemas beschreiben kann.

[30] Siehe *http://xerces.apache.org/ (zuletzt geprüft am* 24.10.2012*).*

[31] Siehe *http://www.w3.org/Submission/2004/SUBM-xmlschema-api-20040122/* (zuletzt geprüft am *24.10.2012*).

[32] Ein Teil des vorzustellenden Testsystems macht Gebrauch von dieser Schnittstelle, um spezielle Einschränkungen - sogenannte *Additional Constraints* - zu überprüfen. Diese werden in Kapitel 2.4.1 vorgestellt.

Die Beschreibung der Semantik ist möglich, da Schematron die Dokumentation der gestellten Anforderungen ermöglicht (vgl. [48]). *Lee und Chu* merken jedoch an, dass Schematron den Aspekt der Schema-Definition komplett ignoriert. Somit bestehen Bedenken, sie als eine Universalsprache zu betratchen (vgl. [48]).
Die Nutzung der recht neuen XML-Schema Version 1.1[33] kann ebenso eine Lösung darstellen. Diese Version führt laut *Delima et al.* das Konzept der Assertions und Typ-Alternativen ein[26]:

- In XML-Schema 1.1 können Element-Deklarationen eine Typtabelle haben, die u.a. eine Folge von *Typ-Alternativen* beinhaltet. Jede dieser Alternativen spezifiziert einen eingeschränkten[34] XPath. Die zu verwendende Deklaration wird aufgrund der XPath-Ausdrücke bestimmt[35].
- *Assertions* erlauben die Kontrolle über das Vorkommen, und den Wert, von Elementen und Attributen, und basieren ebenso auf eingeschränkten[36] XPath 2.0 Ausdrücken.

Ein Nachteil ist jedoch, dass die verwendeten XPath-Ausdrücke in Ihrer Mächtigkeit beschränkt sind. Aus diesem Grund wird für die VCD Spezifikation ein zusätzliches Schematron-Schema verwendet.

2.3.2 Testen von Implementierungen

Im letzten Kapitel wurden die Schemasprachen XML-Schema und Schematron besprochen. Dabei hatte sich herausgestellt, dass beide Sprachen verschiedene Stärken haben. Während XML-Schema mit einem komplexen Typsystem aufwartet, erlaubt Schematron die Definition sogenannter *co-occurrence constraints* (vgl. [16] und [48]). Dies führte zur Schlussfolgerung, dass die Nutzung beider Sprachen durchaus sinnvoll sein kann (vgl. [16]). Jedoch kann es vorkommen, dass die Mächtigkeit beider Sprachen nicht ausreichend ist. Zum Beispiel dann, wenn Referenzen überprüft werden sollen, die sich auf das Dateisystem beziehen. Dies ist bei den Spezifikationen IMS-CC und VCD der Fall (vgl. Anforderungen für *externe Referenzen* in Kapitel 3.4.2 und Kapitel 3.4.3). Solche Anforderungen werden in der später vorzustellenden Lösung durch sogenannte *Additional Constraints* (vgl. Kapitel 2.4.2) erfasst werden. Diese werden ebenso mit Hilfe der XML festgehalten, und erweitern die Menge der erfassbaren Anforderungen[37].
Somit kann eine Spezifikation zahlreiche formale Dokumente beinhalten, die in verschiedenen Schemasprachen formuliert sind. Bei der Überprüfung eines Dokuments müssen somit verschiedene Tests bemüht werden. Auch können auszuführende Tests von dem Inhalt konkreter Dokumente abhängen (vgl. Anbindung der QTI-Tests in [64]). Dies legt

[33] Diese ist ebenso eine W3C Recommendation. Vgl. *http://www.w3.org/TR/xmlschema11-1/* (zuletzt geprüft am *24.10.2012*).
[34] Vgl. *http://www.w3.org/TR/xmlschema11-1/#cvc-assertion* (zuletzt geprüft am *24.10.2012*).
[35] Es wird diejenige Alternative gewählt, für die die Auswertung `true` ergibt.
[36] Vgl. *http://www.w3.org/TR/xmlschema11-1/#ta-props-correct* (zuletzt geprüft am *24.10.2012*).
[37] In Kapitel 4 wird näher auf das Format, und die Überprüfung, der Anforderungen eingegangen.

nahe, eine Methodik für das Testen zu spezifizieren. In diesem Kontext sollen die Arbeit von *Gebase et al.*[50], sowie das Testing Framework for Global eBusiness Interoperability Test Beds (*Ivezic et al.*[42]) - GITB - näher betrachtet werden.
Doch zuerst soll der Begriff des Conformance Testings näher erläutert werden. *Gebase et al.* zitieren die *ISO*, um den Begriff *Conformance* zu definieren. Demnach ist Conformance als die Erfüllung von spezifizierten Anforderungen definiert, die Produkte, Prozesse oder Services betreffen (vgl. ISO/IEC 17000:2004[5]). Der Begriff des *Conformance Testings* wird durch *Gebase et al.* wie folgt definiert[50]:

> „Conformance testing is a way to determine directly or indirectly that all relevant requirements in a standard have been implemented correctly."

Gebase et al. merken an, dass Conformance Testing als Black-Box Testing anzusehen sei. Der Tester habe keine Kenntnis über die interne Struktur der Implementierung und keinen Zugriff auf den Sourcecode. Es würden Testfälle verwendet, die aus korrekten und inkorrekten Eingaben bestehen. Nach Ausführung der Tests können die Ausgaben auf Korrektheit hin untersucht werden, wozu der Standard bemüht werden kann. Somit könne Conformance Testing nur die Existenz von Fehlern, aber nicht deren Abwesenheit nachweisen (vgl. [50]). Laut den Autoren sei Conformance Testing eine Voraussetzung für das sogenannte Interoperability Testing, welches sicherstellen solle, dass verschiedene Systeme „zusammenarbeiten" können. Conformance und Interoperability Testing könnten u.a. gewährleisten, dass Nachrichten durch Systeme empfangen und verschickt werden können, und diese mit passenden Nachrichten antworten können (vgl. [50]).
Gebase et al. stellen zudem Teststrategien für den „weit verbreiteten" HL7 Nachrichten-Standard vor. Dieser ermöglicht, dass klinische und administrative Informationen zwischen Programmen im Gesundheitswesen ausgetauscht werden können (vgl. [50]). Der Standard selbst wird durch die SDO *Health Level Seven International* bereitgestellt, die von dem *American National Standards Institute* - ANSI - anerkannt wurde[38]. Dabei wurden laut *Ivezic et al.* Geschäftsprozesse festgehalten, die die im „wirklichen Leben" ausgeführten Prozesse reflektieren (vgl. S. 102 in [42]). Usage Scenarios von Health Level 7 werden im, durch das GITB Projekt verfolgten, Ansatz berücksichtigt, der letztendlich die detaillierte Spezifikation des GITB-Testing Frameworks vorsieht[39] (vgl. S.10 in [42]). Auch das Projekt PEPPOL, in dessen Rahmen die Spezifikation VCD entworfen wurde (vgl. [52]), leistete einen Beitrag (vgl. S. 10 in [42]). Dabei wird das Langzeit-Ziel des Projekts wie folgt beschrieben[42]:

> „The long-term objective is to establish a shared and Global eBusiness Interoperability Test Bed (GITB) infrastructure to support conformance and interoperability testing of eBusiness Specifications and their implementation

[38] Die Angaben können auf der Webseite der Organisation überprüft werden. Vgl. *http://www.hl7.org/about/FAQs/index.cfm?ref=nav* (zuletzt geprüft am *24.10.2012*). Der Prozess der sogenannten *accreditation* wird in Kapitel 2.3.3 näher besprochen werden.

[39] Das Projekt sieht ebenso eine Implementierung vor, die als ein Konzeptnachweis (proof-of-concept) dienen soll.

by software vendors and end-users.“

Dabei werden einige Begriffe definiert, die ein besseres Verständnis der, in dieser Arbeit zu behandelnden, Problematik ermöglichen. Auf diese soll im folgenden näher eingegangen werden, wobei ebenso ihre Rolle (im jeweiligen Kontext) besprochen werden soll. Unter diesen ist der Begriff der *eBusiness-Spezifikation* von zentraler Bedeutung[42]:

> „An eBusiness Specification is any agreement or mode of operation that needs to be in place between two or more partners in order to conduct eBusiness transactions.“

Dabei hängt eine eBusiness-Spezifikation nach *Ivezic et al.* mit einer, oder mehreren, Ebenen des sogenannten eBusiness-Interoperability-Stacks zusammen (vgl. S. 20 in [42]). Die Ebenen betreffen sowohl Geschäftsprozesse, Geschäftsdokumente, den Transport und die Kommunikation, und stehen damit mit den in Kapitel 2.1.2 eingeführten Interoperabilitätsstufen in Verbindung (vgl. S. 8 in [42]). Dabei kommt es häufiger vor, dass eine eBusiness Spezifikation eine Menge von Standards, oder Profile von diesen, beinhaltet (vgl. S. 13 in [42]). Darauf aufbauend wird der Begriff des sogenannten *System under test* - SUT - definiert[42]:

> „An implementation of one or more eBusiness Specifications, which are part of an eBusiness system which is to be evaluated by testing.“

Der Begriff des Conformance Testings wird analog zu *Gebase et al.* definiert, jedoch genauer spezifiziert. *Conformance Testing* ist demnach ein spezieller Prozess[42]:

> „Process of verifying that an implementation of a specification (SUT) fulfills the requirements of this specification, or of a subset of these in case of a particular conformance profile or level.“

Die Autoren führen weiterhin aus, dass diese Tests üblicherweise mit Hilfe eines, mit dem SUT verbundenen, Test Beds statt finden (vgl. S. 13 in [42]). Es können jedoch ebenso sogenannte Test Items getestet werden. Diese stellen den Autoren nach eine Einheit von zu verifizierenden Daten dar. Als Beispiele werden u.a. Dokumente und XML-Fragmente genannt (vgl. S.16 in [42]). Ein Anwendungsszenario wird in Abbildung 2.9 auf Seite 41 gezeigt.
Laut *Ivezic et al.* ist das zu testende System bei diesem Szenario nicht mit dem Test Bed verbunden. Das zu testende Artefakt (Test Item) wird durch jemanden, der an den Tests beteiligt ist (Test Participant), beschafft (vgl. S. 64 in [42]). Ein anderes Szenario ergibt sich, wenn die Test Items durch das Test Bed eingeholt werden sollen. Ein Beispiel für ein solches Szenario ist in Abbildung 2.10 auf Seite 41 abgebildet.

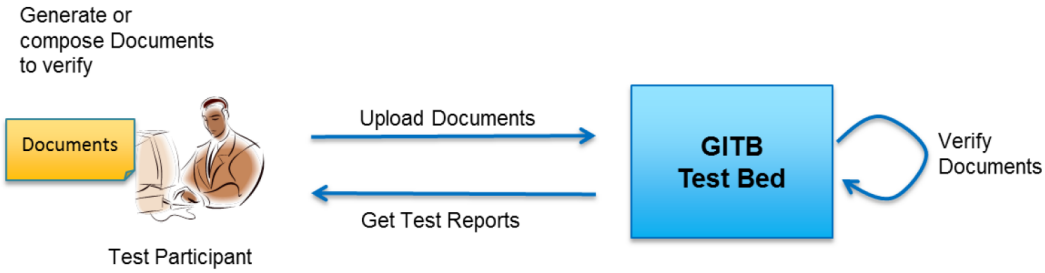


Abbildung 2.9: Abbildung 10-2 aus [42]: Workflow of a standalone document validation

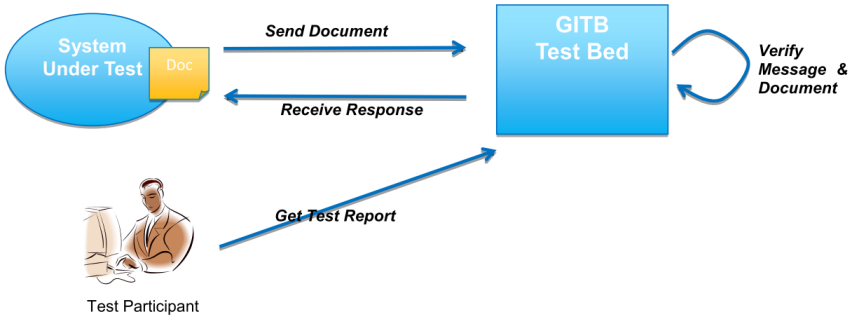


Abbildung 2.10: Abbildung 10-3 aus [42]: Sample workflow in interactive Conformance Testing

In diesem Szenario kommuniziert das Test Bed mit dem SUT und validiert die Ausgaben. Der Test Participant interagiert jedoch lediglich mit dem Test Bed (vgl. S. 64 in [42]). Es stellt sich jedoch die Frage, wie die formale Erfassung der Anforderungen und die Validierung realisiert werden sollen. Beide Aspekte werden durch *Ivezic et al.* genauer spezifiziert.
Laut den Autoren halten sogenannte Document Assertion Sets - DAS(s) - die formalen Anforderungen fest. Denn ein solches beinhaltet mehrere Artefakte, die zur Validierung genutzt werden können. Diese Artefakte können z.B. XML-Schemata oder Code-Listen[40] sein (vgl. S. 14 in [42]). Das Konzept wird verallgemeinert, indem der Begriff der Test Assertion eingeführt wird. Eine solche ist laut *Ivezic et al.* „ein testbarer oder messbarer Ausdruck“, der häufig in einer semi-formalen Repräsentation vorliegt. Mit der Hilfe eines solchen Ausdrucks solle eine Implementierung hinsichtlich der Einhaltung einer normativen Aussage (die Teil der Spezifikation ist) beurteilt werden (vgl. S. 15 in [42]).
Test Assertions können wiederum zur Erstellung von sogenannten Test Cases genutzt werden (vgl. S. 63 in [42]). Diese stellen laut den Autoren „eine ausführbare Einheit der Verifizierung und/oder der Interaktion mit einem SUT“ dar (vgl. S. 15 in [42]). Nach *Ivezic et al.* kann eine sogenannte Test Suite auf diese Test Cases zurück greifen. Dabei definiert eine solche einen Ablauf von Test Case- oder Document Validator-Ausführungen,

[40]Ein Beispiel ist das *Genericode*-Format - vgl. *http://docs.oasis-open.org/codelist/ns/genericode/1.0/* (zuletzt geprüft am *24.10.2012*). Dieses wurde für die VCD-Spezifikation benutzt - vgl. [13].

um ein, oder mehrere, SUTs gegen eine, oder mehrere, eBusiness-Spezifikationen zu prüfen (vgl. S. 16 in [42]).
Dabei wird ein Document Validator als ein Software-Programm definiert, welches „einige Aspekte der Dokument-Anforderungen überprüfen kann“. Diese Anforderungen werden laut den Autoren durch sogenannte Validation-Assertions festgehalten. Als Beispiele werden ein XML-Schema und Konsistenzregeln angegeben (vgl. S.14 in [42]). Insgesamt wird der Prozess durch Abbildung 2.11 auf Seite 42 verdeutlicht[42]:

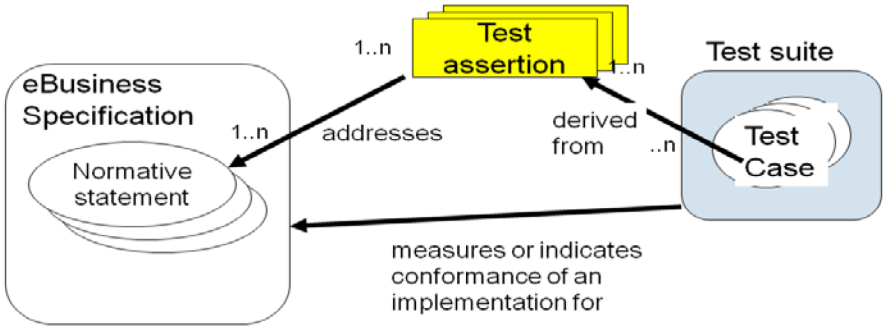


Abbildung 2.11: Abbildung 10-1 aus [42]: The Role Of Test Assertions

- Im Idealfall wird eine Menge von Test Assertions definiert, bevor eine Test Suite und Test Cases erstellt werden.
- Diese dienen als Grundlage für die Herleitung von Test Cases.
- Aufbauend auf den Test Cases kann eine Test Suite erstellt werden. Das Vertrauen in die resultierende Test Suite wird durch Test Assertions verbessert.

Die Autoren merken an, dass Test Assertions als Basis für eine Coverage-Analyse dienen können. Mit einer solchen kann abgeschätzt werden, in welchem Umfang die Spezifikation getestet wurde (vgl. S. 63 in [42]). Letztendlich wird eine Test Suite durch eine *Test Suite Engine* ausgeführt[42]:

> „A Test Suite Engine (or ”Test Suite Driver”) is a processor that can execute a Test Suite, or has control of the Test Suite main process execution in case it delegates part of the execution [...]“

Eine Delegation kann z.B. erfolgen, wenn eine Validierungs-Aufgabe an einen Document Validator übergeben wird (vgl. S. 16 in [42]). Das Ergebnis einer solchen Ausführung kann wiederum zu einem Test Report führen. Ein solcher kann ebenso das Resultat sein, wenn Test Items, oder die Ausgabe von einem, oder mehreren, SUT(s), überprüft wird (vgl. S. 16 in [42]).
Die Validierung der Dokumente kann jedoch von der Interaktion (mit einem SUT) getrennt werden. Der Ansatz wird Two-Phase Testing genannt und mit Hilfe von Abbildung 2.12 auf Seite 43 erklärt[42]:

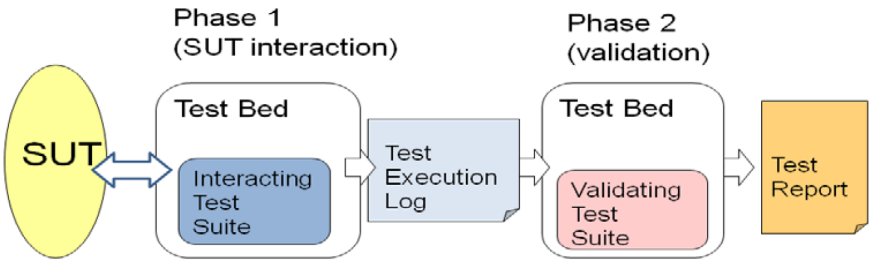


Abbildung 2.12: Abbildung 10-5 aus [42]: Two-phase testing

- In der *ersten Phase* erfolgt die Interaktion mit dem SUT. Die dafür zuständige Test Suite erstellt eine Aufzeichnung der Ausführung.
- In der *zweiten Phase* wird diese Aufzeichnung validiert und ein Test Report erstellt.

Somit kann eine Test Suite ebenso in einem Runtime-Szenario eingesetzt werden. Dies ist wichtig, denn laut den Autoren kann das Testen von Dokumenten bzw. Nachrichten-Instanzen nicht sicherstellen, dass ein SUT valide Dokumente bzw. Nachrichten generieren kann (vgl. S. 66 in [42]). Mittels des Two-phase testings können die beiden Aufgaben jedoch voneinander separiert und bestehende Lösungen wiederverwendet werden.
Letztlich stellen *Ivezic et al.* genauere Definitionen der besprochenen Begriffe vor. Doch die Verwendung der abstrakteren Definitionen erlaubt eine Verbindung zu der vorzustellenden Methodik. Dabei wird ein Teil der zu behandelnden Problematik in einem Beispiel besprochen, dass sich auf Abbildung 2.13 auf Seite 44 bezieht.
Die Abbildung soll die Nutzung sogenannter *Testing Capability Components* demonstrieren, welche von den Autoren ebenfalls als *Test-Plugins* bezeichnet werden. Nach *Ivezic et al.* können mehrere Instanzen einer Komponente Teil einer operativen Test-Bed-Plattform sein. So könnten beispielsweise mehrere Validatoren, z.B. ein Validator für XML-Schema, und ein Validator für Schematron, deployed werden. Dabei wird eine Test-Suite ebenfalls als Plugin behandelt (vgl. S.49 in [42]).
Diese Komponenten werden den Autoren nach durch die in der Abbildung gezeigte *Test-Bed-Core-Platform* genutzt. Um dies zu ermöglichen müssen die einzelnen Komponenten eine standardisierte *Core-Plugin-API* implementieren, die der Plattform gestattet, gleiche Arten von Plugins in gleicher Weise anzusteuern. Damit die Plugins aber ebenso die Core-Plattform ansprechen können, implementiert diese sogenannte *Core-Platform-APIs* (vgl. S. 49 in [42]). Letztendlich wird das in der Abbildung gezeigte Szenario wie folgt erklärt[42]:

1. Die Komponente *Test-Control-Manager* führt die Test-Suite *HL7-interop* aus, und benutzt dafür die Test-Suite-Engine *TDL1*.
2. Ein Test-Case dieser Test-Suite benötigt eine Art der Validierung - eine *Testing Capability*, die mit dem Namen *HL7-doc* identifiziert wird.
3. Somit kommuniziert die Engine mit der Komponente *Test-Control-Manager* und fragt eine entsprechende Validierung an.

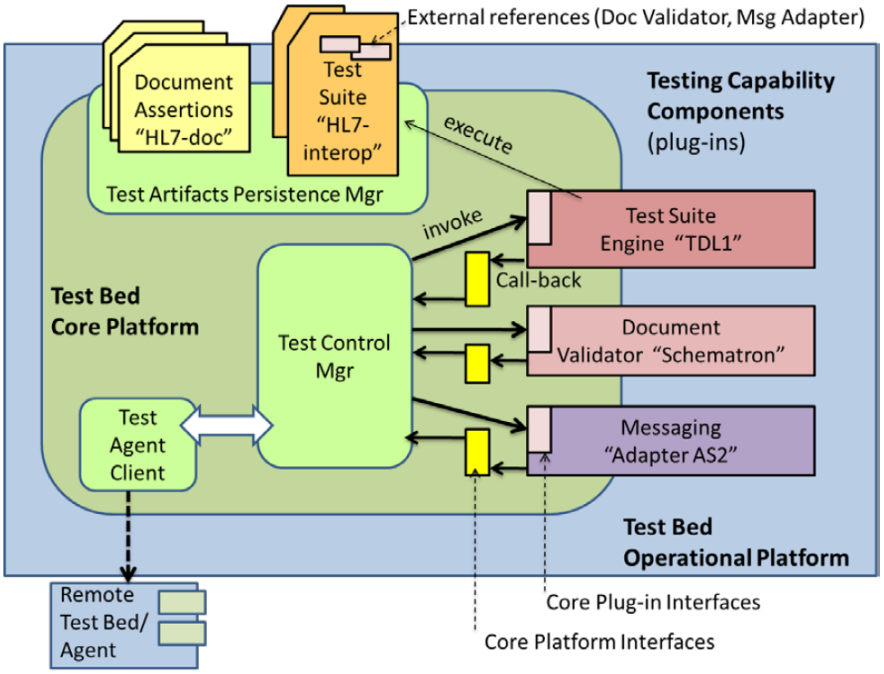


Abbildung 2.13: Abbildung 8-2 aus [42]: Coordination of Plug-in Components via Core Interfaces

4. Diese macht eine geeignete Komponente aus und weist die Validierung an, die in diesem Fall durch einen Schematron-Validator ausgeführt wird.

5. Letztendlich wird das Ergebnis an die Test-Suite-Engine übergeben.

2.3.3 Bereitstellung von Zertifikaten

Im letzten Kapitel wurde das GIT-B Projekt näher betrachtet. Dabei wurden viele grundlegende Begriffe erklärt, die für die Beschreibung des GIT-B Frameworks genutzt wurden. Der Schwerpunkt lag auf dem Thema Conformance Testing, da dieses in dieser Arbeit näher behandelt werden soll. Neben den vorgestellten Begriffen beschreibt das Framework ebenso Methodiken, um die Anforderungen einer Spezifikation festzuhalten, und konkrete Dokumente auf die Erfüllung hin zu testen (vgl. [42] und Kapitel 2.3.2).
In Kapitel 2.2.3 wurden verschiedene Stakeholder herausgearbeitet, die ein Interesse an Conformity Assessment haben können. Dabei steht der Begriff Conformity Assessment für die Prozesse und Richtlinien, die für das Conformance Testing umzusetzen sind (vgl. [50]). Es stellte sich heraus, dass sowohl *Standard Development Organizations* - SDOs, *Standard User Organizations* - SUOs, als auch *Standard Software Organizations* - SSOs - betroffen sein können.
Ein Beispiel wäre eine SSO, die ein Software-Framework bereitstellt. Ein solches kann z.B. Teile eines Standards implementieren, und die Erstellung eines standard-konformen

End-Produkts ermöglichen (vgl. Phase *Develop Products* in [75] sowie Kapitel 2.3.2). In diesem Fall muss sich eine SUO darauf verlassen können, dass sich das Framework an den Standard hält. Somit liegt das Interesse der SUO in einer Bestätigung, wogegen die SSO darin interessiert ist, diese bereitzustellen (hergeleitet aufgrund von [33] und [34]). Dieser Zusammenhang wird durch *Gebase et al.*[50], *Fabbrini et al.*[33] und *Fusani und Marchetti*[34] näher beschrieben. Nach *Gebase et al.* wird Conformance Testing oftmals durch sogenannte Testlabore ausgeführt. Diese zertifizieren, dass eine Implementierung die Tests bestanden hat. Zudem merken die Autoren an, dass die Tests als Kommunikation zwischen Käufern und Verkäufern dienen (vgl. [50]). Die Auslagerung von Tests wurde ebenso durch *Söderström* behandelt und in Kapitel 2.2.3 dargelegt. Nach der Autorin können sogenannte *Standard Service Provider* - SSP - für die Ausführung des Conformity Assessments verantwortlich sein (vgl. [75]). Die Beziehung zwischen Conformance Testing, Conformity Assessment, und der Zertifizierung werden mit Hilfe von Abbildung 2.14 auf Seite 45 verdeutlicht.

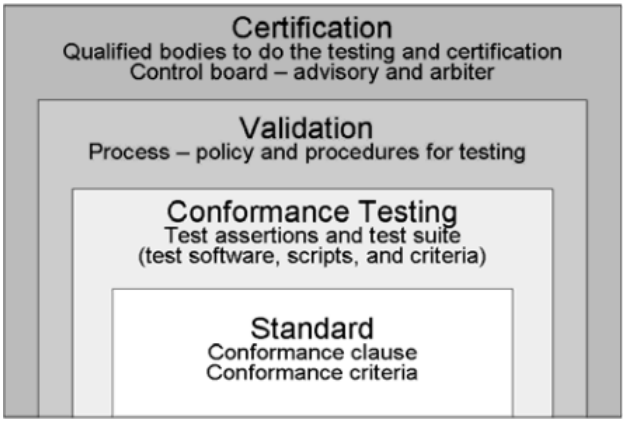


Abbildung 2.14: Abbildung 1 aus [50]: Certification Building Blocks

Die Aktivitäten bauen laut *Gebase et al.* aufeinander auf. So kann Conformance Testing nur dann erfolgen, wenn der Standard klare, und testbare, Bedingungen festlegt (vgl. [50]). Dabei ist Conformity Assessment „eine Aktivität, die demonstriert, dass spezifische, auf Produkte, Prozesse, Systeme, Personen oder Institute bezogene, Anforderungen erfüllt sind“ (frei übersetztes Zitat aus [5]). Die Zertifizierung an sich ist wiederum eine Prozedur, mit welcher eine Drittpartei die Erfüllung der Anforderungen schriftlich bestätigen kann (vgl. [3]). Somit kann das Ausführen von Conformity Assessment zur Bestätigung führen (vgl. [33]). Dabei ist eine Zertifizierungsstelle „ein Organismus mit internen Regeln, humanen / infrastrukturellen Ressourcen und Kompetenzen, um die Zertifzierungsprozesse durchzuführen“ (vgl. [33]).
Laut *Fabbrini et al.* können diese internen Regeln wiederum konform zu bestimmten Standards sein. In diesem Fall könne eine Zertifizierungsstelle „akkreditiert“ werden, womit Ihr bescheinigt wird, dass sie dazu imstande ist, eine Zertifizierung durchzuführen. Die Akkreditierung erfolgt durch ein Akkreditierungs-Institut, wobei ein solches wieder-

um durch ein anderes Akkreditierungs-Institut akkreditiert wird. (vgl. [33]).
Laut *Fabbrini et al.* hat es Vorteile, wenn die Zertifizierung unter Einhaltung von Standards geschieht. Denn dies ermöglicht den Vergleich von Resultaten, und das Ergebnis sei reproduzierbar. Zudem wäre es wichtig, dass die Zertifzierung denselben Regeln folgt, wenn Anforderungen geprüft werden, die Objekte vom gleichen Typ betreffen (vgl. [33]). Abbildung 2.15 auf Seite 46 fasst die erklärten Sachverhalte zusammen und ist der Arbeit von *Fabbrini et al.* entnommen. Jedoch stellt sich die Frage, mit welchen Folgen die Zertifizierung verbunden ist. Denn laut *Gebase et al.* kann Conformance Testing, im Bereich der Programmierung, nur die Existenz von Fehlern, aber nicht deren Abwesenheit nachweisen (vgl. [50] und Kapitel 2.3.2). Auch *Fabbrini et al.* merken an, dass der zusätzliche Wert eines Produkts von dem Wert abhängt, der den Anforderungsstandards zu Teil wird (vgl. [33]). Nach *Fusani und Marchetti* kann Conformity Assessment somit als Weitergabe von Vertrauen verstanden werden (vgl. [34]).

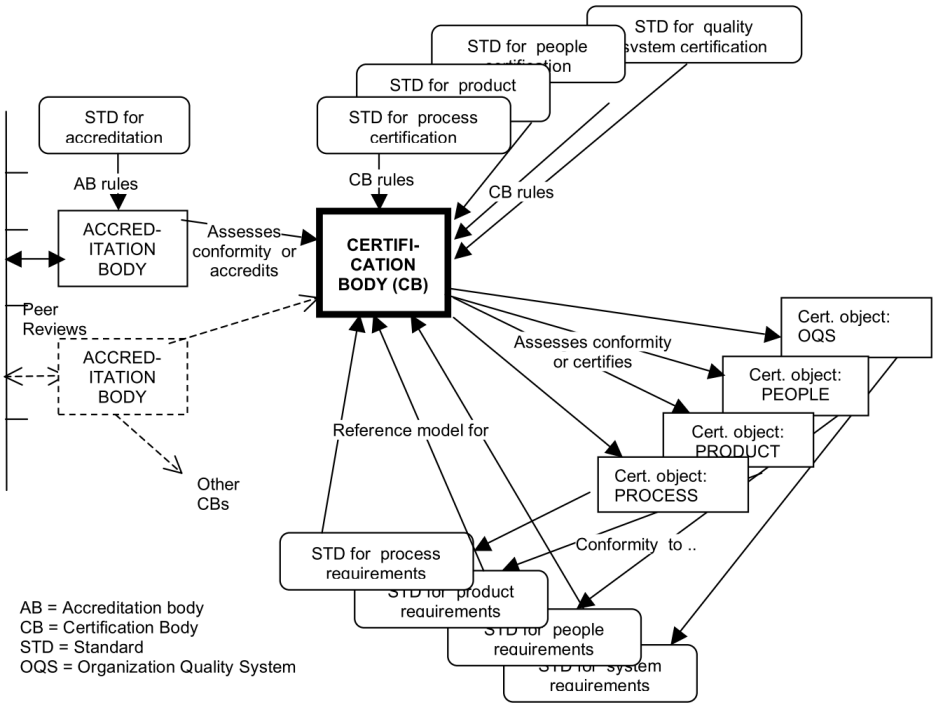


Abbildung 2.15: Abbildung 1 aus [33]: Certification and Accreditation Scenario (simplified)

Die Autoren gehen näher auf den Softwarebereich ein und betrachten die Auswirkung verschiedener Faktoren, die die Zertifizierung von *Open-Source-Software* - OSS - und *Closed-Source-Software* - CSS - betreffen. Sie betonen, dass jeder Stakeholder eine andere Ansicht von Vertrauen und Zuversicht hat. Z.B. könne ein Entwickler die Zuversicht anstreben, dass seine Software Deadlock-frei ist, während der Nutzer den Anspruch haben kann, dass die Software sich sicher und effizient verhält (vgl. [34]). Dies kann eben-

so im Zusammenhang mit dem Modell nach *Söderström* betrachtet werden. Z.B. kann eine SUO Software erwerben, die von einer SSO bereitgestellt wird (vgl. [75] und Kapitel 2.2.3).
Die Autoren verdeutlichen die verschiedenen Sichtweisen, indem sie das in Abbildung 2.16 auf Seite 47 gezeigte Szenario vorstellen, welches den Autoren nach typisch sei. In diesem habe der Zulieferer die Zuversicht, dass sein Produkt gewisse Eigenschaften hat, und möchte diese Zuverischt an den Kunden weitergeben. Letztendlich wird diese Aufgabe durch eine Zertifizierungsstelle übernommen, die den Autoren nach als eine Art „Katalysator“ agiert (vgl. [33]).

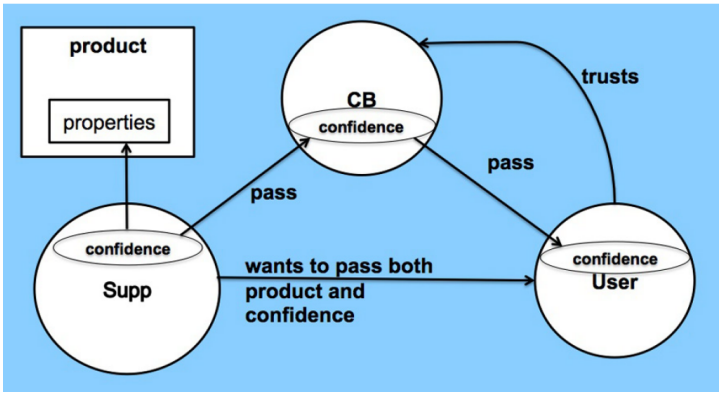


Abbildung 2.16: Abbildung 2 aus [34]: Confidence transfer

Ein einfacheres Szenario wird in Abbildung 2.17 auf Seite 48 dargestellt, welches der Arbeit von *Fabbrini et al.* entnommen wurde. In diesem stellt der Zulieferer die Zertifikate selbst aus. Laut den Autoren kann dies nat. zur Folge haben, dass das Vertrauen der Kunden geringer ausfällt. Es bestünde kein Beweis dafür, das der Zulieferer sich an kontrollierte und verifizierte Prozesse hält, um das Objekt zu produzieren und zu zertifizieren (vgl. [33]).
Fusani und Marchetti gehen ebenso näher auf die Verifizierbarket von Eigenschaften ein und führen aus, dass eine Eigenschaft nicht immer beweisbar sein muss (vgl. [34]). Dieser Aspekt wird durch die, bereits vorgestellte, Definition des Conformity Assessment-Begriffs berücksichtigt. Denn diese umschliesst lediglich eine Demonstration, die die Einhaltung der Anforderungen zeigen soll (vgl. Definition der ISO - [5]).

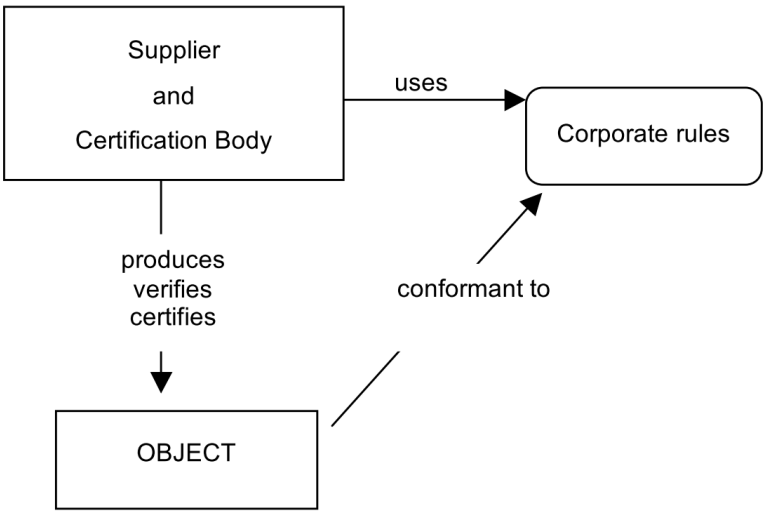


Abbildung 2.17: Abbildung 2 aus [33]: Simpler Certification scheme

2.4 Anpassung bestehender Datenformate

In den letzten Kapiteln (vgl. Kapitel 2.2.1 und Kapitel 2.2.2) wurde näher auf die Datenmodellierung eingegangen. Dabei wurden die Ansätze von *Lampathaki et al.* und *Routledge et al.* vorgestellt, die drei unterschiedliche Abstraktionsebenen einführten. Die Bedeutung der auszutauschenden Daten wurde mit konzeptuellen Modellen, die Repräsentation der Daten durch logische Modelle erfasst. Als Schemasprache wurde XML-Schema vorgeschlagen (vgl. [47] und [67]).
Kapitel 2.3.1 stellte die Schemasprachen XML-Schema und Schematron ausführlicher vor. Dabei stellte sich heraus, dass die Philosophie einer jeden Sprache anders ist und jede Sprache verschiedene Arten von Validierungsanforderungen unterstützt (vgl. [48] und [16]). Somit wurde die Verwendung mehrerer Sprachen als sinnvoll erachtet, wobei die Kombination von XML-Schema und Schematron durch *Coen et al.* vorgeschlagen wurde (vgl. [16]).
In Kapitel 2.3.2 wurden Konzepte für die Realisierung von Tests eingeführt. Diese wurden durch das GITB-Framework beschrieben, welches Conformance- und Interoperability-Tests von eBusiness-Spezifikationen, sowie deren Implentierung, unterstützen soll (vgl. S. 8 in [42]). Dabei wurden sogenannte Test Assertions eingeführt, welche die testbaren Anforderungen beschreiben, und auf eine formale Schreibweise verweisen (vgl. S. 15 in [42]). Im einfachsten Fall sind solche Assertions durch verschiedene Dokumente abgedeckt, die in unterschiedlichen Schemasprachen verfasst sind (vgl. Begriff *Document-Validator* bzw. *Validation-Assertion* - S.14 in [42]). Somit können XML-Schema und Schematron verwendet werden, um die Anforderungen zu erfassen, und einen konkreten Testablauf zu realisieren (vgl. *Test Bed Operational Platform* - Abbildung 2.13 auf Seite 44).
Anschließend wurde das Thema der Zertifizierung behandelt (vgl. Kapitel 2.3.3). Ein

Zertifikat kann das Ergebnis von Tests bestätigen (vgl. Zusammenhang von *Conformity Assessment* und *Zertifizierung*[33]), und die Weitergabe von Vertrauen ermöglichen (vgl. Abbildung 2.16 auf Seite 47 und [34]). Dabei können die Tests von einer unabhängigen Zertifizierungsstelle ausgeführt werden, die sich selbst an vorgeschriebene Standards hält (vgl. *Akkreditierung* von Zertifizierungsstellen[33]). Dies ermöglicht ein höheres Vertrauen in die Zertifikate und führt zu reproduzierbaren, und damit zu vergleichbaren Ergebnissen (vgl. [33]).
Letztendlich wurden all diese Schritte in einen Zusammenhang gebracht, indem dass durch *Söderström* eingeführte Modell (vgl. Kapitel 2.2.3) vorgestellt wurde. Dieses beschreibt den Lebenszyklus eines Standards, und verbindet die verschiedenen Aktivitäten, die während des Lebenszyklus ausgeführt werden (vgl. Abbildung 2.6 auf Seite 27 und [75]). Die für diese Arbeit relevanten Aktivitäten wurden herausgearbeitet, und in einer Abbildung festgehalten (vgl. Abbildung 2.7a auf Seite 29). Während die Datenmodellierung der Phase *Develop Standard* zugeordnet werden kann, können die Tests der Phase *Conformity Assessment* zugeordnet werden (vgl. [74] und Definition von *Conformity Assessment* in [5]). Dabei wurde festgestellt, dass Wechselwirkungen zwischen den Aktivitäten bestehen, und verschiedene Stakeholder existieren, die unterschiedliche Interessen an diesen Aktivitäten haben (vgl. [75]).
Die folgenden Kapitel sollen jedoch näher auf die Wiederverwendung existierender Spezifikationen und Standards eingehen. Im Bereich der Datenformate kann eine solche durch die Nutzung von Applikationsprofilen erfolgen (vgl. [29]). Dabei erlaubt eine bestimmte, in Kapitel 2.4.1 vorgestellte, Form von Applikationsprofilen die Aggregation, und Modifikation, existierender Spezifikationen (vgl. Begriff *Domain-Profile* - [23]). Durch die Wiederverwendung existierender Spezifikationen und Standards kann die Erfahrung, die in die Erstellung dieser eingeflossen ist, genutzt werden. Zudem besteht der Vorteil, dass kein proprietäres Format zum Einsatz kommt. Die Zusammenarbeit von Implementierungen kann gewährleistet werden, wenn diese auf denselben Spezifikationen aufbauen, und gewisse Regeln eingehalten werden (vgl. Abbildung 2.19b auf Seite 53 und [65]). Die folgenden Kapitel sollen diese Möglichkeit im Detail beschreiben und sind wie folgt aufgebaut:

- Kapitel 2.4.1 wird den Begriff des Applikationsprofils vorstellen und sich an die Definitionen halten, die Teil der *IMS Application Profile Guidelines* ([63] sowie [65]) sind. Neben dem Begriff soll ebenso der Kontext beschrieben werden, in dem sich die Erstellung eines Applikationsprofils auszahlt.

- Kapitel 2.4.2 wird spezielle Modifikationen behandeln, die die Zahl der gültigen Dokumente einschränken. Die Änderungen beziehen sich dabei auf ein sogenanntes Informationsmodell (vgl. Begriff *Learning Technology Specification* - S. 6 in [65]), welches einem Modell auf konzeptueller Ebene entspricht (Herleitung im Kapitel). Da die Modifikationen restriktiv sind, entsprechen konforme Dokumente ebenso der Basisspezifikation (vgl. Erklärung restriktiver Profile - S. 9-11 in [65]).

- Kapitel 2.4.3 wird sogenannte XML-Bindings und deren Modifikation behandeln. Zu einem Informationsmodell kann ein Binding erstellt werden, welches das Modell

mit einer konkreten Technologie verbindet. Zum Beispiel können die im Informationsmodell festgehaltenen Informationen zum Teil in ein XML-Schema übertragen werden. Dies kann ebenso für die zutreffenden Modifikationen geschehen, so dass ein separates XML-Dokument die vorgenommenen Änderungen beschreibt (vgl. [63]).

2.4.1 Der Begriff des Applikationsprofils

Laut *Dahn* kann eine Spezifikation nicht alle Einsatzgebiete, und die mit Ihnen verbundenen Anforderungen, berücksichtigen. Denn diese stehen nicht von vorne herein fest (vgl. [23]). Dies lässt sich am Modell von Söderström verdeutlichen, dass in Kapitel 2.2.3 vorgestellt wurde. So nehmen verschiedene Stakeholder an den Aktivitäten teil, die zum Lebenszyklus eines Standards gehören (vgl. [75]). In der Phase „Develop Standard“ wird die Entwicklung des Standards realisiert (vgl. [75]). Dies geschieht mittels der Einflussnahme verschiedener Beteiligter, die durch *Jakobs* in verschiedene Gruppen eingeteilt wurden. Sogenannte *Leaders* nehmen dabei den größten Einfluß auf den Standardisierungsprozess. Zu den strategischen Zielen gehören die Schaffung eines Marktes, sowie die Erstellung eines erfolgreichen Standards (vgl. [43]).
Somit fließen maßgeblich die Interessen der, zu dieser Gruppe gehörenden, Firmen in die Standardisierung, und somit, in ein zu entwickelndes Datenformat, mit ein. Aufgrund dessen werden die Einsatzgebiete des Standards durch die Interessen dieser Gruppe vorgegeben (Aussagen hergeleitet aufgrund von [43]). Somit kann es vorkommen, dass der Standard Optionen vorsieht, die in einem anderen Kontext nicht benötigt werden (vgl. [29]), oder das Anforderungen eines Einsatzgebietes nicht erfasst werden (hergeleitet aufgrund der vorigen Aussage). Eine Lösung wäre, sich an den Aktivitäten des Lebenszyklus zu beteiligen, wenn neue Versionen eines Standards entwickelt werden (vgl. Modell von Söderström[74][75]).
Eine Beteiligung am Standardisierungsprozess (Phase „Develop Standards“ in Abbildung 2.7a auf Seite 29) ist allerdings nicht immer praktikabel, oder nicht möglich, da der Standard bereits in einer finalen Version vorliegt. In einem solchen Fall kann sich die Erstellung eines Applikationsprofils anbieten. Ein solches kann die Vorgaben eines Standards einschränken, so dass die spezifischen Anforderungen und Einschränkungen einer Gemeinschaft berücksichtigt werden (vgl. [29] - inhaltlich zitiert durch [30]). Die Verwendung einer vorhandenen Spezifikation sichert jedoch nicht die Inteorperabilität mit anderen Systemen (vgl. [23]). Allerdings verhindert die Nutzung (bzw. die Erstellung) verschiedener Spezifikationen und Standards die Möglichkeit der Interoperabilität (vgl. [23]). Somit ist die Entwicklung eines Applikationsprofils ggf. vorzuziehen, wenn bereits eine geeignete Spezifikation existiert[41].
Bei der Erstellung eines Applikationsprofils werden Einschränkungen an einem *Informationsmodell* vorgenommen (vgl. S. 8 in [63]). Ein solches ist laut *Riley et al.* eine abstrakte Informationsstruktur, die nicht an eine spezifische Technologie gebunden ist

[41] In den *Application Profilling Guidelines* (Kapitel „Things to Consider Before Developing a Profile“) wird auf weitere Gründe eingegangen, die für, und gegen, die Entwicklung eines Applikationsprofils sprechen (vgl. [63]).

(vgl. S. 11 in [63]). Zudem kann ein Informationsmodell einen Teil der Semantik, ein konzeptuelles Schema, Daten-Elemente, sowie UML-Use-Cases abdecken (vgl. S.6 in [63][42]). Dabei ist ein Informationsmodell Teil einer Spezifikation[43], die der konzeptuellen Ebene zuzuordnen ist. Denn die restlichen, durch *Lampathaki et al.* und *Routledge et al.*, eingeführten Datenmodelle (vgl. Kapitel 2.2.1 und Kapitel 2.2.2) geben die Struktur der Dokumente vor. Dabei unterscheiden *Riley et al.* zwischen zwei verschiedenen Arten von Applikationsprofilen[65]:

- ein sogenanntes *Data Profile* besteht aus einem Profil, dass sich auf ein Informationsmodell bezieht, und ebenso als „Conceptual Data Schema“ bezeichnet wird.
- ein sogenanntes *Bound Data Profile* beinhaltet mindestens ein zusätzliches Profil, welches sich auf ein Binding bezieht. Es können jedoch mehrere Profile für unterschiedliche Bindings exisitieren.

Riley et al. behandeln jedoch ausschließlich die Möglichkeit eines XML-Bindings, welches ein Informationsmodell mit geeigneten XML-Strukturen in Verbindung bringt (vgl. S. 8 in [65]). Die Dokumentation eines Bindings kann der logischen Ebene zugeordnet werden. Denn nach *Lampathaki et al.* führt das logische Modell ein XML-Schema ein, welches die Struktur valider Dokumente festhält (vgl. Abbildung 2.3 auf Seite 19). Auch nach *Routledge et al.* wird die Struktur der Dokumente bereits auf logischer Ebene festgehalten (vgl. Abbildung 2.5 auf Seite 22). Somit kann jeweils ein Profil für das konzeptuelle Modell, als auch für das XML-Schema, dass die konkrete Struktur der auszutauschenden Informationen vorgibt, festgehalten werden.
Dabei kann der Prozess des *Application Profiling*[44] laut *Riley et al.* mit 3 Aktivitäten zusammengefasst werden. Davon sind die ersten beiden für diese Arbeit relevant[63][45]:

- der *Localization* - der Spezialisierung von einem oder mehreren konzeptuellen Daten-Schemata (*Conceptual Data Schemas*). Diese muss die genauen Anforderungen der Gemeinschaft berücksichtigen. Das Ergebnis ist ein hergeleitetes Schema.
- der *Representation* - dem Abbilden des (bzw. der) lokalisierten konzeptuellen Schemas (bzw. Schemata) auf ein generisches, für den Austausch ausgelegtes, Binding.

Laut *Riley et al.* können etliche Operationen genutzt werden, um ein Informationsmodell anzupassen. Jede dieser Operationen hat jedoch eine Auswirkung auf die Kompatibilität (vgl. S. 14 in [65]). Es ist jedoch vorzuziehen, dass ein Applikationsprofil kompatibel mit der Basisspezifikation ist (vgl. S. 19 in [63]). Im Detail wird dies mit Systemen erklärt, die Daten *empfangen* und *versenden*. Die Realisierung der Kommunikation tritt dabei in den Hintergrund (vgl. S. 8ff in [65]).

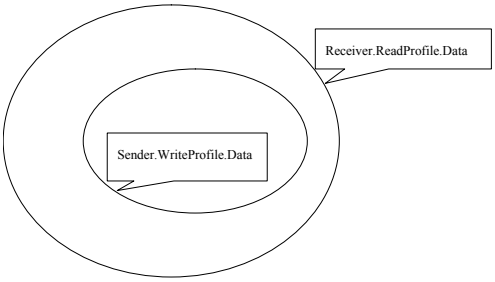
[42] Diese Aspekte werden durch ein Informationsmodell abgedeckt, welches Teil einer sogenannten *Learning Technology Specification* ist.

[43] Dies gilt für *Learning Technology Specifications* (vgl. S. 6 in [63]).

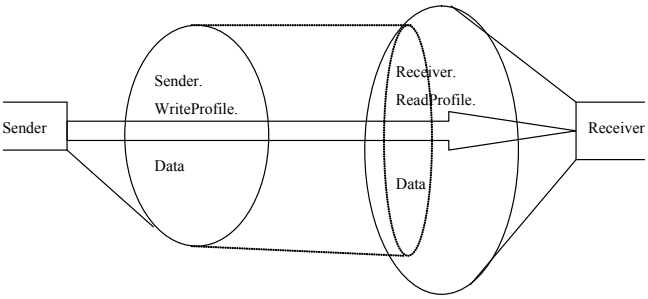
[44] Die Erstellung eines Applikationsprofils wird mit dem Begriff *Application Profiling* bezeichnet (vgl. S. 8 in [63]).

[45] Die dritte Aktivität beschäftigt sich mit abstrakten Interfaces und Service-Modellen, die nicht näher betrachtet werden sollen.

Dabei unterscheiden *Riley et al.* zwischen dem Lese- und Schreibprofil eines Programms. Ein solches wird durch ein gebundenes bzw. ungebundenes Profil vorgegeben, und beschreibt die Daten, die das jew. Programm lesen bzw. schreiben kann (vgl. S. 8 in [65]). Diese Daten werden durch *Riley et al.* mit einer Schreibweise verbunden: $DP.Data$ bezeichnet die Daten, die konform zum Profil DP sind. Damit eine erfolgreiche Kommunikation stattfinden kann, muss der Empfänger alle potentiellen Daten lesen können, die der Sender versenden kann. D.h. es muss $Sender.Schreibprofil.Data \subseteq Empfänger.Leseprofil.Data$ gelten. Der Zusammenhang wird durch Abbildung 2.18a auf Seite 52 verdeutlicht, die den Application Profiling Guidelines entnommen ist (vgl. [65]).
Diese Grafik wird ebenso zur Darstellung der Kommunikation benutzt, die in Abbildung 2.18b auf Seite 52 gezeigt wird. Die Teilmengenrelation wird hier übernommen und mit Hilfe der Projektion verdeutlicht.



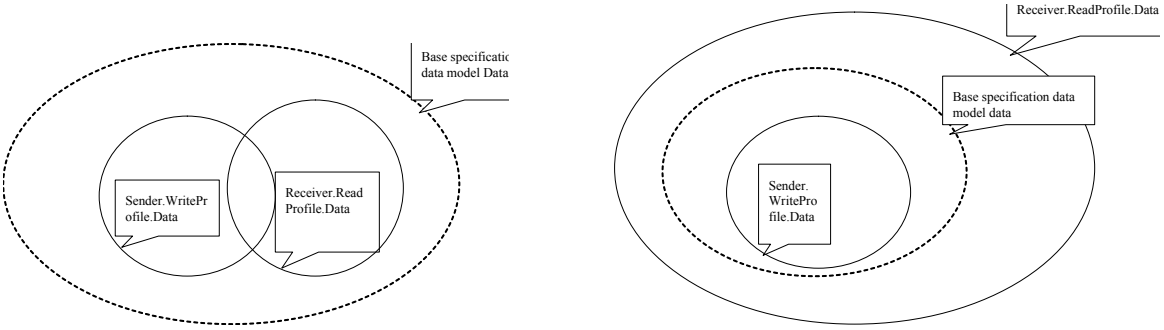
(a) Abbildung 2.1 aus [65] (S. 9): Data sets in an interoperable setting



(b) Abbildung 2.2 aus [65] (S. 9): Data communication in an interoperable setting

Abbildung 2.18: Vorraussetzungen für eine erfolgreiche Kommunikation von Implementierungen

Die Profile sollten jedoch im Zusammenhang mit der Basisspezifikation betrachtet werden. Ein mögliches Szenario wird in Abbildung 2.19b auf Seite 53 gezeigt. Der Empfänger kann alle Daten lesen, die konform zur Basisspezifikation sind. Es können gar Daten gelesen werden, die durch diese nicht erfasst werden. In diesem Fall wird das Leseprofil des Empfängers als *extensive* bezeichnet. Anders verhält es sich mit dem Sendeprofil des Senders. Alle potentiellen Daten des Senders sind konform zur Basisspezifikation. In diesem Fall heißt es, dass das Sendeprofil *restriktiv* ist (vgl. [65]).
Das gezeigte Szenario deckt sich somit mit dem Kommunikationsszenario, dass in Abbildung 2.18b auf Seite 52 abgebildet ist. Abbildung 2.19a auf Seite 53 zeigt jedoch, dass Implementierungen restriktiver Profile nicht kompatibel sein müssen. In der Abbildung wird das Schreibprofil des Senders nicht durch das Leseprofil des Empfängers erfasst. Das Szenario ermöglicht jedoch, dass Werkzeuge und Produkte, die zur Basisspezifikation konforme Daten verarbeiten können, ebenso mit Daten arbeiten können, die konform zu einem der beiden Profile sind (vgl. S. 14 in [65]). D.h. wenn eine Gemeinschaft restriktive Profile entwickelt, die auf derselben Basisspezifikation aufbauen, dann können Tools, Produkte und Services bereitgestellt werden, die auf der Basisspezifikation ba-



(a) Abbildung 2.3 aus [65] (S. 10): Two restrictive data profiles of a base specification data model resulting in non-interoperable systems

(b) Abbildung 2.4 aus [65] (S. 10): An extensive and a restrictive profile of a base specification data model resulting in interoperable applications

Abbildung 2.19: Interoperabilität bezüglich der Basisspezifikation

sieren. Daher sollen in dieser Arbeit lediglich Modifikationen behandelt werden, die zu einem *restrikitiven* Profil führen. Solche Modifikationen werden laut *Riley et al.* auch *restriktive Modifikationen* genannt.

2.4.2 Restriktive Modifikationen

Im folgenden sollen die restriktiven Modifikationen herausgearbeitet werden, die zumeist Teil der Modifikationen sind, die in den Application Profiling Guidelines[65] vorgestellt werden. Es kann jedoch vorkommen, dass eine Modifikation nur in bestimmten Fällen restriktiv ist.
Für die Erklärung bestimmter text-basierter Modifikationen ist zudem die Unterscheidung zwischen dem *Typ* und der *Darstellung* einer Information oftmals hilfreich. Diese Unterscheidung wird z.B. durch den 2ten Teil der W3C-Recommendation für XML-Schema[46] eingeführt:

- Elemente des **Wertebereichs** eines *Datentyps* sind eindeutig betreffend Ihrer Bedeutung,
- während die Elemente des **Lexikalischen Bereichs** die Repräsentation der Elemente des *Wertebereichs* sind.

Im englischen wird der Wertebereich als *value space*, und der lexikalische Bereich als *lexical space* bezeichnet. Nach dieser Definition kann die Einschränkung von gültigen Texten auch den Wertebereich einschränken. Ein Beispiel wäre, wenn eine bestimmte Zahl auf das Intervall 1-100 eingeschränkt wird. Eine solche Vorgabe kann ebenso durch einen Regulären Ausdruck formuliert werden. Dieser arbeitet dann allerdings auf der *Repräsentation* der Zahlen.

[46] Siehe *http://www.w3.org/TR/xmlschema-2/* (zuletzt geprüft am *25.10.2012*).

Modifikationen von Text-Elementen (TM) Ein Informationsmodell kann *Text-Elemente* enthalten, welche eine Menge von gültigen Texten definieren. Die gültigen Texte können durch die folgenden Modifikationen weiter eingeschränkt werden:

Beschränkung der Länge von Text Definition einer Textlänge, die nicht überschritten werden darf (vgl. S. 14s in [65]). Falls eine solche Einschränkung besteht, sollte die neue Länge kleiner als die alte Länge sein. (TM.1)

Änderung des Datentyps von Text Änderung des *Typs* der Information, die durch einen gültigen Text repräsentiert wird. Ein Beispiel ist die Verwendung von reellen Zahlen. Diese kann auf die Verwendung natürlicher (genauer: Integer-) Zahlen eingeschränkt werden (vgl. S. 15f in [65]). Da der Wertebereich der natürlichen Zahlen eine Teilmenge des Wertebereichs der reellen Zahlen ist, wäre eine solche Modifikation restriktiv (vgl. S. 15 in [65]). Laut *Riley et al.* kann eine Änderung des Datentyps effizienter, wie die Einführung von vielen, komplexen, Einschränkungen sein (vgl. S. 15f in [65]). Somit sprechen *Riley et al.* den Vorteil an, auf dem *Wertebereich* zu arbeiten. Wird auf dem *lexikalischen Bereich* gearbeitet, müssten u.U. verschiedene Darstellungen einer Zahl erfasst werden. (TM.2)

Definition eines Default Wertes Definition eines Wertes, der genau dann benutzt werden soll, wenn eine dem Element zugeordnete Stelle im Dokument keinen Text beinhaltet (vgl. S. 20 in [65]). (TM.3)

Einschränkung der Nachkommastellen und der Genauigkeit von Zahlen Diese Modifikation kann auf Elemente angewendet werden, deren Wertebereich aus reellen Zahlen besteht. Für gültige Zahlen kann eine Anzahl von Nachkommastellen, sowie eine Genauigkeit, definiert werden (vgl. S. 16 in [65]). Falls eine solche Modifikation bereits besteht, müssen die neuen Werte kleiner als die der alten Modifikation sein. Laut *Riley et al.* kann eine Einschränkung der Genauigkeit zu Rundungsfehlern führen, wenn eine Implementierung des Profils in Prozessen genutzt wird, die der Basisspezifikation entsprechen (vgl. S. 16 in [65]). (TM.4)

Definition eines Intervalls für Zahlen Diese Modifikation kann auf Elemente angewendet werden, deren Wertebereich aus reellen bzw. Integer-Zahlen besteht. Im Rahmen der Modifikation wird ein Interval für gültige Zahlen definiert oder geändert (vgl. S. 17 in [65]). Falls eine solche Modifikation bereits vorhanden ist, muss das neue Intervall im alten Intervall enthalten sein. (TM.5)

Definition eines Ausdrucks für gültige Texte Definition eines formalen Ausdrucks, der gültige Texte beschreibt. Ein solcher Ausdruck kann z.B. ein regulärer Ausdruck sein (vgl. S. 19 in [65]). Falls bereits eine solche Modifikation existiert, muss die Menge der gültigen Texte bzgl. des neuen Ausdrucks in der Menge der Texte enthalten sein, die bzgl. des alten Ausdrucks gültig sind. (TM.6)

Modifikationen von natürlichsprachlichen Text-Elementen (NTM) Modifikationen diesen Typs ändern *Text-Elemente*, die für natürlichsprachliche Beschreibungen gedacht sind.

Festlegen einer Sprache für natürlichsprachliche Inhalte Entsprechende Texte müssen in der vorgegebenen Sprache verfasst sein (vgl. S. 20 in [65]). (NTM.1)

Festlegen einer Sprache für alle natürlichsprachlichen Inhalte eines Profils Alle Texte, die Teil des *Profils* sind, müssen in der vorgegebenen Sprache verfasst sein (vgl. S. 20 in [65]). (NTM.2)

Modifikationen der Struktur eines Informationsmodells (SM) Diese Modifikationen betreffen die Struktur, die durch ein Informationsmodell festgelegt wird.

Vorschreiben von optionalen Elementen Diese Art von Modifikation erlaubt, dass ein optionales Element in ein verbindliches Element geändert wird (vgl. S. 17 in [65]). Laut *Riley et al.* können verbindliche Elemente die Menge der gültigen Kombinationen, die sich aus den optionalen Elementen ergibt, einschränken. Somit muss eine Implementierung nicht alle Kombinationen unterstützen (vgl. S. 17 in [65]). (SM.1)

Verbieten von optionalen Elementen Mit Hilfe einer solchen Modifikation kann ein optionales Element verboten werden. Alternativ kann festgelegt werden, dass das Element zwar vorkommen darf, aber stets leer sein muss. Die Spezifikation wird durch diese Modifikation vereinfacht (vgl. S. 18 in [65]). Wenn optionale Elemente entfernt werden, ist die Implementierung des Profils einfacher. (SM.2)

Vorgabe einer Reihenfolge bzgl. einer Menge von Elementen Eine Modifikation dieser Art betrifft eine Menge von *Elementen*. Diese sollen in einer Instanz in einer bestimmten Reihenfolge erscheinen (vgl. S. 19 in [65]). Laut *Riley et al.* kann es sein, dass diese Modifikation nicht einfach umzusetzen ist. Dies kommt auf die verwendete Technologie an, die für das Binding genutzt wird (vgl. S. 19 in [65]). (SM.3)

Wiederverwendung von Elementen und Strukturen Das Referenzieren oder Einfügen von Elementen und Strukturen, die in anderen Datenmodellen definiert wurden (vgl. S. 21 in [65]). Laut *Riley et al.* beinhalten viele Spezifikationen sogenannte *extension points*, die für das Hinzufügen von Elementen aus anderen Spezifikationen, oder aus dem eigenen Informationsmodell geeignet sind (vgl. S. 21 in [65]). Diese sollten genutzt werden, da die sonstige Folge ist, dass Implementierungen der Basisspezifikation nicht mehr mit profil-konformen Daten umgehen können. Denn das Schreibprofil wäre nicht mehr restriktiv zur Basisspezifikation (siehe Kapitel 2.4.1). (SM.4)

Einschränkung des Rekursionslevels für rekursive Elemente Diese Art der Modifikation betrifft Elemente, die Instanzen von sich selbst beinhalten. Dabei kann eine maximale Verschachtelungstiefe angegeben werden (vgl. S. 21 in [65]). (SM.5)

Bedingte Modifikationen (BM) Bedingte Modifikationen sind nur anzuwenden, wenn eine *inhaltliche Bedingung* erfüllt ist.

Bedingtes Vorschreiben von optionalen Elementen Diese Modifikation ähnelt der Modifikation, die unter dem Punkt *"Vorschreiben von optionalen Elementen"* beschrieben wurde. Der Unterschied ist, dass eine zusätzliche *inhaltliche Bedingung* angegeben wird, deren Erfüllung über die Anwendung der Modifikation entscheidet. Ob die Modifikation anzuwenden ist, ergibt sich somit erst durch den Inhalt einer Instanz, und steht nicht von vorne herein fest (vgl. S. 17 in [65]). (BM.1)

Additional Constraints (ACM) Modifikationen, die mit der Hilfe von *Additional Constraints* ausgedrückt werden können. Diese setzen allerdings ein XML-Binding voraus, und werden in Kapitel 2.4.3 besprochen.

Nutzung eines kontrollierten Vokabulars Für ein Element kann ein Vokabular festgelegt, oder ein existierendes, restriktiveres Vokabular, verwendet werden. Handelt es sich um ein *Text-Element* (vom Datentyp *String*), kann eine Liste von zulässigen Strings vorgegeben werden (vgl. S. 15 in [65]). *Riley et al.* merken zudem an, dass externe Vokabulare referenziert werden können, was die Anpassung existierender Systeme vereinfachen würde. Zudem können Vokabulare spezifisch für die relevante Domäne sein (vgl. S. 15 in [65]), und können bei der Definition der Semantik helfen (vgl. Kapitel 2.1.3). So können die im Binding verwendeten Begriffe mit Bedeutung versehen werden. (ACM.1)

Nutzung eines vorgeschriebenen Vokabulars Für ein Element kann ein spezifisches, kontrolliertes, Vokabular vorgegeben werden. Dieses definiert die zulässigen Werte (vgl. S. 18 in [65]). (ACM.2)

Einschränkung der Art referenzierter Elemente Falls ein Element eine Referenz beinhalten kann, so kann der Typ des Zielelements eingeschränkt werden (vgl. S. 21f in [65]). (ACM.3)

Einschränkung der Struktur eines Datenpakets Falls Daten als eine Menge von Dateien ausgetauscht werden, können zusätzliche Einschränkungen auf der Struktur, und den Inhalten eines Datenpakets, definiert werden (vgl. S. 22 in [65]). (ACM.4)

Allgemeine Modifikationen (AM) Diese Modifikationen lassen sich nicht in die vorher eingeführten Kategorien einordnen.

Verdeutlichung der Bedeutung eines Elements Es kann eine zusätzliche Beschreibung für ein Element verfasst werden, um eine spezifische Interpretation festzulegen (vgl. S. 16 in [65]). Laut *Riley et al.* kann eine solche Beschreibung dazu beitragen, dass die vorgesehene Nutzung des Elements klarer und eindeutiger wird. Die Bedeutung sollte jedoch nicht geändert werden (vgl. S. 16 in [65]). (AM.1)

2.4.3 Regeln für Datenpakete und Inhalte

In Kapitel 2.4.2 wurden Modifikationen vorgestellt, die die Anpassung eines Informationsmodells erlauben. Alle vorgestellten Modifikationen sind restriktiver Natur, so dass es möglich ist, dass konforme Daten durch eine Implementierung der Basisspezifikation verarbeitet werden können (vgl. Abbildung 2.19b auf Seite 53). Zu einem sogenannten *Bound Data Profile* gehört jedoch ein weiteres Profil, welches die Änderungen bzgl. des Bindings ausdrückt (vgl. S. 8 in [65]). Die Erstellung eines solchen Profils wird vor allem dann empfohlen, wenn die Interoperabilität zwischen verschiedenen Parteien gewährleistet werden soll (vgl. S. 11 in [65])[47].
Dabei wird in den Application Profiling Guidelines ausschließlich auf die Möglichkeit eines XML-Bindings eingegangen, welches durch ein XML-Schema vorgegeben ist (vgl. S. 11 in [65]). Die Änderungen am Basisschema werden in einem Profil beschrieben, das ebenfalls in XML verfasst ist (vgl. S. 23 in [65]). Dabei kann das Basisschema aus mehreren XML-Schemata aufgebaut sein[48].
Die Änderungen am Basisschema werden durch eine formale Notation der vorgestellten Modifikationen (vgl. Kapitel 2.4.2) zum Ausdruck gebracht. Dabei sind die Modifikationen SM.4 und ACM.4 von besonderer Bedeutung, da diese Einschränkungen erlauben, die mehrere Datenmodelle betreffen:

- Ein Informationsmodell kann zulässige Mittel definieren, die eine Erweiterung des Modells erlauben (vgl. S. 10 in [65]). D.h. an spezifischen Stellen des Informationsmodells können beliebige Inhalte eingefügt werden (vgl. S. 12 in [65]). Ein restriktives Profil kann somit die Inhalte, die an solchen Stellen erlaubt sind, einschränken (vgl. S. 12 in [65]).
  Dies ist ebenso möglich, wenn ein Profil für das Basisschema erstellt wird. Ein XML-Schema kann sogenannte *Wildcards* definieren, die z.B. die Nutzung von XML-Elementen erlauben können, die wiederum in anderen XML-Schemata definiert wurden[49]. Die Identifizierung der Elemente geschieht durch die Festlegung von Namespaces. Es können jedoch Hinweise angegeben werden, die für eine Bestimmung der, bei einer Validierung zu verwendenden, XML-Schemata genutzt werden können[50]. Ein Applikationsprofil erlaubt die Modifizierung solcher Wildcards, so dass nur noch XML-Elemente zugelassen werden, die konform zu einem bestimmten Profil sind (vgl. Modifikation SM.4).

- Ein Informationsmodell kann zudem die Nutzung von Referenzen erlauben, für die bestimmte Auflagen definiert werden können. Gültige Referenzen können durch

[47] Alternativ können ein Profil für das Informationsmodell, und ein neues Binding, erstellt werden (vgl. S. 11 in [65]).

[48] XML-Schema ist eine W3C-Recommendation, die unter *http://www.w3.org/XML/Schema* eingesehen werden kann (zuletzt geprüft am *25.10.2012*). Ein XML-Schema kann andere XML-Schemata *importieren*, *inkludieren*, oder *Wildcards* verwenden. Diese Möglichkeiten werden im ersten Teil der Recommendation - *Part 1: Structures* - beschrieben (zuletzt geprüft am *25.10.2012*).

[49] Vgl. *http://www.w3.org/TR/xmlschema11-1/#Wildcards* (zuletzt geprüft am *25.10.2012*).

[50] Vgl. *http://www.w3.org/TR/xmlschema11-1/#xsi_schemaLocation* (zuletzt geprüft am *25.10.2012*).

eine Modifikation des Typs ACM.4 eingeschränkt werden. Dabei werden Bedingungen angegeben, die für das Ziel einer solchen Referenz gelten müssen. Eine solche Bedingung kann z.B. festlegen, dass eine referenzierte Datei ein Dokument sein muss, welches wiederum einem anderen Profil entsprechen muss (vgl. Einbindung von QTI-Tests in IMS-CC - *Dahn*[23]). Die Gültigkeit einer solchen Referenz kann somit nur durch einen Test der referenzierten Datei geprüft werden[51].

In beiden Fällen kann sich ein Szenario ergeben, welches mehrere Profile miteinander verbindet. Laut *Dahn* wird ein Profil „von mehreren Spezifikationen und ihrer gemeinsamen Verwendung“ auch als *Domain-Profile* bezeichnet (vgl. [23]). Abbildung 2.20 auf Seite 58 veranschaulicht den möglichen Aufbau eines solchen:

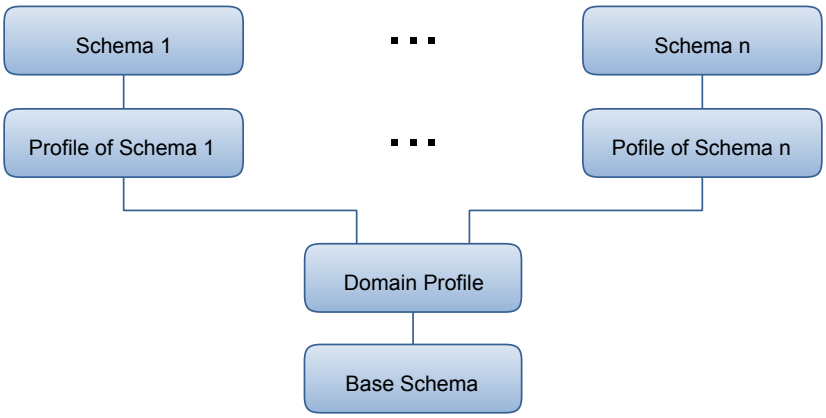


Abbildung 2.20: Domain-Profile, welches verschiedene Profile zusammenbringt

- Es existieren verschiedene XML-Schemata ($Schema_1$ bis $Schema_n$), welche die Bindings der beteiligten Datenmodelle beschreiben.
- Die Änderungen werden in separaten Profilen festgehalten ($ProfileofSchema_1$ bis $ProfileofSchema_n$), die durch ein, auf dem Bassischema basierendes, Hauptprofil ($DomainProfile$) zusammengebracht werden.

Somit kann ein Domain-Profile genutzt werden, um verschiedene Bindings zu agreggieren, und zu modifizieren. Zudem kann eine Entsprechung zwischen den Modifikationen des Informationsmodells, und denen des Basisschemas bestehen. Die Erstellung eines Domain-Profiles ist somit auf konzeptueller, sowie auf logischer Ebene möglich.

[51] Wenn die Referenz auf keine Datei verweist, oder formell ungültig ist, muss nat. kein Test erfolgen.

3 Zwei Spezifikationen im Vergleich

Im folgenden sollen zwei Spezifikationen näher betrachtet werden, die aus zwei unterschiedlichen Bereichen stammen:

- die Spezifikation *VCD* ist im Rahmen des Projektes PEPPOL entstanden, welches durch die europäische Kommission gefördert wird (vgl. [7]). Das Ziel des Projekts ist die Aufstellung einer europaweiten, sowie beispielhaften Lösung, die, gemeinsam mit den bereits existierenden (nationalen) Lösungen, die europaweite, öffentliche, sowie interoperable, Auftragsvergabe fördern soll (vgl. [53]). Für Firmen, die an der elektronischen, sowie öffentlichen, Auftragsvergabe teilnehmen wollen, scheinen gemeinsame Standards für den elektronischen Datenaustausch ein Schlüssel-Element zu sein (laut *Mondorf et al.* - vgl. S.20 in [52]). Das VCD ist ein solches Datenformat, welches das Sammeln von Nachweisen und Erklärungen ermöglicht. Diese können mittels eines sogenannten VCD-Containers zusammengefasst, und an eine, für die Bearbeitung zuständige, Stelle übermittelt werden (vgl. [7]). (vgl. Kapitel 3.1)

- die Spezifikation *IMS-Common Cartridge* - IMS-CC - ist ein Profil von mehreren Spezifikationen, die durch ein Profil der IMS Content Packaging - IMS-CP - Spezifikation, verbunden sind (vgl. S. 65 in [24]). Dieses legt ein Format für Datenpakete - sogenannte *Common Cartridge*s - fest, welche die Zusammenstellung von Lerninhalten ermöglichen (vgl. [23]). Dabei schränkt das Profil das Format der IMS-CP-Spezifikation weiter ein. Zum Beispiel werden nur bestimmte Lerninhalte unterstützt (vgl. S. 28 in [64]), oder die Beschreibung der Inhalte vereinfacht (vgl. *Common Cartridge Information Model* - S. 29 in [64]). Die Verarbeitung der Datenpakete wird ebenso berücksichtigt. Diese kann durch ein sogenanntes *Lernmanagement-System* - LMS - erfolgen (vgl. S. 7 in [64]). Ein solches ist ein Programm, welches die Zuweisung von Inhalten (an Studenten), das Lernen an sich, und die Auswertung der Lernergebnisse erlaubt (vgl. S. 10 in [64]). Für die Beschreibung der Verarbeitung werden verschiedene Use-Cases definiert, die verschiedene Abläufe vorgeben. Die von einem LMS zu unterstützenden Aktionen sind ein Teil dieser Abläufe, die somit einen Kontext definieren. Die Abläufe erfassen z.B. den Import, als auch die weitere Arbeit mit den importierten Inhalten (vgl. Beschreibung der Use-Cases - S. 11-18 in [64]). Somit werden Anforderungen an ein LMS gestellt, wenn ein solches die Verarbeitung von Common Cartridges unterstützen soll. (vgl. Kapitel 3.2)

Obwohl die beiden Spezifikationen aus unterschiedlichen Bereichen stammen, sind Ähnlichkeiten in den zugehörigen Datenformaten zu finden. So stützen sich die Formate auf XML-Dateien, die zusammen den Paketinhalt beschreiben. Dabei gelten unterschiedliche Anforderungen, die sich auf die Beschreibung des Paketinhalts, die von einer Beschreibung ausgehenden Referenzen, und den Aufbau eines Datenpakets beziehen. Bei einer näheren Betrachtung stellt sich heraus, dass beide Formate eine Reihe ähnlicher

Anforderungen definieren. Dies beruht auf ähnlichen Lösungen, die in beiden Datenformaten zum Einsatz kommen(vgl. Kapitel 3.3).
Diese Gemeinsamkeiten dienen letztlich der Kategorisierung der Anforderungen. Allerdings können nicht alle Anforderungen durch ein XML-Schema erfasst werden. Somit sollen diese Anforderungen mit der Hilfe von verschiedenen Applikationsprofilen (vgl. Kapitel 2.4.1) festgehalten werden. Letztlich werden restriktive Modifikationen (vgl. Kapitel 2.4.2) vorgeschlagen, die im Rahmen dieser Profile verwendet werden können (vgl. Kapitel 3.4).

3.1 Das Virtual Company Dossier

3.1.1 PEPPOL im Kontext des E-Procurement

Laut *Mondorf et al.* verfolgt die europäische Kommission das wichtige Ziel, die elektronische, europaweite, und somit grenzüberschreitende, Beschaffung zu ermöglichen. Dies soll erreicht werden, indem gemeinsame Prinzipien und technische Lösungen erstellt werden, die dann in den einzelnen Mitgliedsstaaten angewendet werden (vgl. [7]). Das Projekt PEPPOL soll zur Umsetzung dieser Ziele beitragen, und wird durch die europäische Kommission gefördert (vgl. [7]). Ein Ziel des Projektes ist die Aufstellung einer europaweiten, sowie beispielhaften Lösung, die, gemeinsam mit den bereits existierenden (nationalen) Lösungen, die europaweite, interoperable, elektronische, und öffentliche Auftragsvergabe fördern soll (vgl. [53]). Laut dem später (2010) erschienenen eGovernment-Aktionsplan sollen Unternehmen unterstützt werden, indem sie den gesamten öffentlichen Beschaffungszyklus online abwickeln können (vgl. [31]). Dabei soll die elektronische Kommunikation zwischen den Unternehmen, und den, der europäischen Regierung angehörigen, Institutionen, während des gesamten Prozesses ermöglicht werden (vgl. [7]).
Dabei wird der Beschaffungs-Prozess in drei Hauptphasen unterteilt (vgl. [7]). Das sogenannte VCD-Konzept soll die Prozesse unterstützen, die in der ersten Phase, der sogenannten *Pre-Awarding*-Phase, anfallen (vgl. S. 8 in [13]). Der primäre Einsatzbereich des VCDs wird durch *Brutti et al.* mittels der sogenannten Tendering-Phase näher beschrieben (vgl. S. 21 in [13]). Die aufgeführten Aktivitäten finden sich jedoch ebenso im, durch *Mondorf et al.* beschriebenen, Beschaffungsprozess wieder, und können der Pre-Awarding-Phase zugeordnet werden (vgl. S.21 in [13] sowie S. 4 in [7]). Abbildung 3.1 auf Seite 61 zeigt ein, der Arbeit von *Brutti et al.* entnommenes, Use-Case-Diagramm, welches den Prozess verdeutlicht.
Dabei liegt der Fokus des VCDs auf der Aktivität „Collect evidences and submit qualification information“ (vgl. S. 21 in [13]). Dieser Schritt beschreibt eine Aktion eines Anbieters, der sich auf die Vergabe eines Auftrags hin bewerben muss, um an einer öffentlichen Ausschreibung teilzunehmen (vgl S. 21 in [13]). Dazu muss der Anbieter die Erfüllung von bestimmten, durch den Customer vorgegebenen, Bedingungen nachweisen (vgl S. 21 in [13]). Es kann jedoch vorkommen, dass der Anbieter in einem anderen Mitgliedsstaat ansässig ist, da die Ausschreibung EU-weit erfolgt (vgl. [7]). Daher werden die Bedingungen in Form von Kriterien angegeben, die in der EU-Direktive *2004/18/EC*

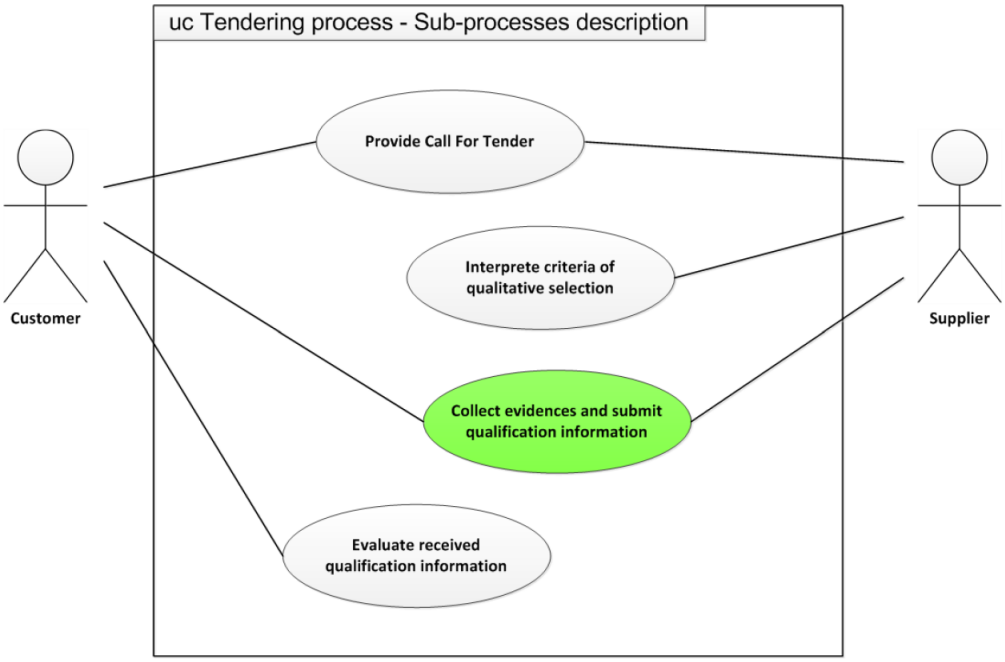


Abbildung 3.1: Abbildung 6 aus [13]: Tendering process use case - Sub process description

definiert sind, und in nationale, die Beschaffung betreffende, Bestimmungen umgesetzt sind (vgl S. 7 in [13]). Letztlich muss der Anbieter die Kriterien interpretieren, die Nachweise einholen, und diese in einem sogenannten VCD-Paket an den Customer übertragen (vgl. S. 7 in [13]).
Im Fall des, durch das PEPPOL-Projekt adressierten, eProcurement-Prozesses ist der Kunde ein öffentlicher Auftraggeber (Contracting Authority), und der Anbieter ein Wirtschaftsteilnehmer (Economic Operator) (vgl. [7]). Das Projekt stellt Software-Komponenten bereit, die einen Economic Operator beim Sammeln der Evidenzen, und von zusätzlichen Dokumenten, unterstützen sollen (vgl. S. 18 in [13]). *Brutti et al.* nennen die hauptsächlichen Komponenten (vgl. S. 18 in [13]), wobei 2 der Komponenten erklärt werden sollen:

- das *europäische VCD System* (EVS) bietet Unterstützung bei der Herleitung der benötigten Evidenzen (vgl. S. 18 in [13]). Evidenzen weisen die Erfüllung von nationalen Kriterien nach, welche durch eine Abbildung mit den europäischen Kriterien verbunden sind (vgl. S. 17 in [13]). Die Herleitung erfolgt in Übereinstimmung mit einer Reihe von Regeln (dem sogenannten Rule-Set) und im Einklang mit den europäischen, sowie den nationalen, Bestimmungen zur Beschaffung (vgl. S. 18 in [13] - dieser Punkt wird in Kapitel 3.1.2 genauer behandelt werden).
- das *nationale VCD System* (NVS) bietet diverse Funktionalitäten und unterstützt den Economic Operator in einem Prozess, der von der Auswahl der Kriterien bis

zur Komplettierung eines validierten VCD-Containers reicht (vgl. S. 18 in [13]).

Dabei ist ein Szenario vorgesehen, in dem das NVS als die einzige Anlaufstelle für den Economic Operator dient (vgl. S. 47 in [54]). In diesem Fall findet eine direkte Kommunikation zwischen dem EVS und dem NVS statt. Diese wird mit Hilfe von Abbildung 3.2 auf Seite 62 erklärt:

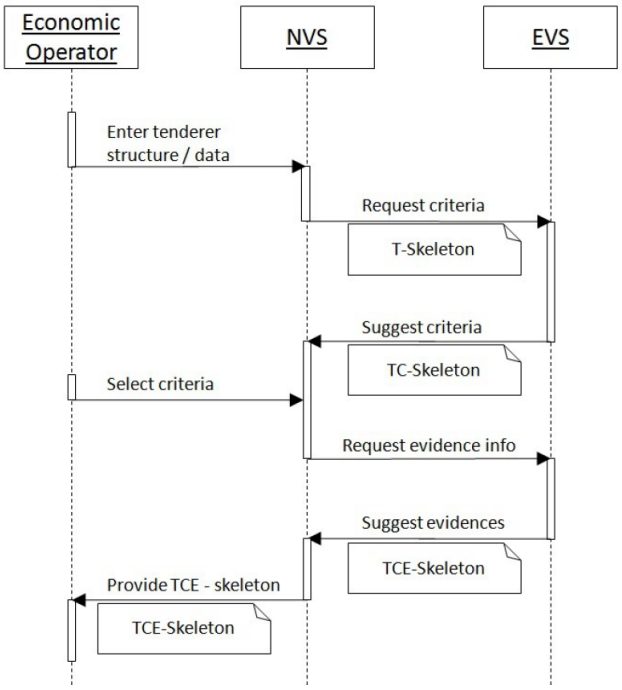


Abbildung 3.2: Abbildung 19 aus [13]: EVS/NVS interaction

- Im ersten Schritt stellt der Economic Operator Informationen zur sogenannten *tenderer structure* bereit (vgl. S. 48 in [52]). Diese kann verschiedene Economic Operator und deren Beziehungen erfassen. Z.B. dann, wenn mehrere Economic Operator ein spezifisches Angebot vorbereiten, und die Anbieter sich aus einem Hauptunternehmer und mehreren Subunternehmern zusammen setzen (Spezialfall der Definition - vgl. S. 48 in [52]).
- Aufgrund der Angaben wird ein T-Skeleton erstellt, welches an das EVS geschickt wird (vgl. Abbildung 3.2 auf Seite 62 sowie S. 47 in [54]). Das EVS übermittelt dem NVS ein sogenanntes TC-Skeleton, welches Informationen über die nachzuweisenden Kriterien enthält (vgl. S. 48 in [52]).
- Im nächsten Schritt erstellt das EVS ein sogenanntes TCE-Skeleton, welches Informationen zu den Evidenzen beinhaltet, die durch jeden Economic Operator bereitgestellt werden müssen. Die Erstellung des TCE-Skeleton geschieht aufgrund einer Selektion von Kriterien (vgl. Abbildung 3.2 auf Seite 62).

Somit werden schrittweise Informationen zum Economic Operator, den zu erfüllenden Kriterien, sowie den passenden Evidenzen hinzugefügt (vgl. S. 44 in [52]). Dabei ist jedes dieser Formate eine Untermenge eines vollen VCD-Pakets (vgl. S. 44 in [52]).
Die Funktionsweise des NVS und des EVS werden jedoch als Spezifikation eingeführt (vgl. S. 54 in [52]). Die Funktionalität des NVS wird durch den sogenannten VCD-Designer und den VCD-Builder implementiert, die zur Referenzimplementierung des NVS gehören (vgl. S. 57 in [52]). Der Economic Operator stellt die im ersten Schritt benötigten Informationen dem VCD-Designer zur Verfügung, der die Implementierung des EVS aufruft. Der VCD-Builder dient der Vervollständigung eines TCE-Skeletons (vgl. S. 59 in [52]).
Neben den genannten Implementierungen existieren ebenso Implementierungen für weitere, im Rahmen dieser Arbeit nicht besprochene, Komponenten (vgl. S. 58-60 in [52]).
Die Entwicklung der Implementierungen geschah parallel zur Entwicklung der Spezifikation (vgl. S. 42 in [52]). Dabei werden die folgenden Vorteile genannt (vgl. S. 42 in [52]):

- Die Referenzimplementierungen zeigen, dass die Spezifikation umgesetzt werden kann (vgl. S. 42 in [52]). Dies deckt sich mit einer Aussage von *Söderström*, nach welcher Referenzimplementierungen eine Hilfe sein können, um die ordnungsgemäße Funktion eines Standards zu bestätigten (vgl. Kapitel 2.2.3 sowie [75]). Laut *Mondorf et al.* werden die Referenzimplementierungen ebenso für die IT-Industrie bereitgestellt, die die Implementierungen in ihre eigenen Plattformen und Systeme einarbeiten können (vgl. S. 42 in [52]). D.h. diese Implementierungen sind mit sogenannten Produkten zu vergleichen, die nach *Söderström* in der Phase *Develop Products* entwickelt werden (vgl. Abbildung 2.7a auf Seite 29 sowie [75]). Im Modell von *Söderström* werden diese durch sogenannte *Standard Software Organisations* - SSOs - bereitgestellt (vgl. Kapitel 2.2.3 sowie [75]).

- Die Referenzimplementierungen machen Tests möglich (vgl. S. 42 in [52]). Dabei sind verschiedene Testphasen involviert. Während in der ersten Phase Operationen mit künstlichen Daten und Beteiligten simuliert wurden (vgl. Phase POC - S. 18 in [52]), bauten die darauffolgenden Tests auf Qualifikationsdokumenten auf, die im Rahmen einer Ausschreibung eingereicht wurden (vgl. Phase Test Pilots - S. 18 in [52]). Letztlich sollen EO und CAs mit einbezogen werden, indem diese ihre gegenwärtigen kommerziellen Anwendungen und Daten benutzen (vgl. S. 18 in [52]). D.h. verschiedene Stakeholder sind an den Tests beteiligt. In den ersten beiden Phasen können die Tests ohne die Nutzer erfolgen, da sie auf keinen aktuellen Daten beruhen. Im letzten Abschnitt sind jedoch die Nutzer (vgl. SUOs in Kapitel 2.2.3) betroffen, deren aktuelle Daten als Grundlage dienen. D.h. das im Modell vorgesehene Feedback (vgl. Abbildung 2.7a auf Seite 29) kann in der letzten Phase durch die Benutzer erfolgen. Aufgrund der parallel ausgeführten Tests hat die Phase des *Conformity Assessments* einen festen Platz im Spezifikations-Prozess (vgl. Phase Conformity Assessment in Abbildung 2.7a auf Seite 29).

- Die Referenzimplementierungen helfen dabei, das Vorhaben zu verdeutlichen, und

können als Referenz dienen, wenn andere Implementierungen beurteilt werden sollen (vgl. S. 42 in [52]).

Letztlich arbeiten die Referenzimplementierungen mit verschiedenen Untermengen eines VCD-Pakets (vgl. T-, TC-, TCE-Skeleton) und müssen diese korrekt verarbeiten können (vgl. Verarbeitung und Erzeugung von T-, TC- und TCE-Skeletons durch das NVS und EVS). Genau dieser Aspekt wird durch eine Lösung für Conformance Tests abgedeckt (vgl. S. 62 in [52]). Diese dient u.a. dazu, dass verschiedene Stakeholder, wie Software-, Service- oder Platform-Provider, die Einhaltung der VCD-Spezifikation erklären können (vgl. S. 61 in [52] sowie Kapitel 2.3.3). Dabei wird die, zur Laufzeit stattfindende Validierung, als eine Aufgabe der Softwarekomponenten betrachtet (vgl. S. 63 in [52]). Das GIT-B beinhaltet jedoch eine Alternative zu diesem Ansatz, die als *Two-Phase Testing* bezeichnet wird (vgl. Abbildung 2.12 auf Seite 43). Mittels dieser Lösung können die Conformance Tests von den Interoperabilitätstests separiert werden. Somit ist es theoretisch möglich, die Lösungen zum Conformance Testing wiederzuverwenden (vgl. Kapitel 2.3.2).

3.1.2 Maßnahmen zur Sicherung der Interoperabilität

Laut *Mondorf et al.* werden innerhalb des PEPPOL Projekts verschiedene Profile - PEPPOL BIS (Business Interoperability Specification) - genutzt, die den Beschaffungsprozess in verschiedene Kollaborationen unterteilen (vgl. S. 15 in [52]). Eine PEPPOL BIS behandelt alle Interoperabilitätsebenen (vgl. Kapitel 1.1.2), und definiert die Anforderungen und Spezifikationen im Detail (vgl. S. 15 in [52]). Dabei ist die sogenannte VCD BIS als Unterstützung für die *Pre-Awarding-Phase* vorgesehen (vgl, S. 8 in [13]).
Im letzten Kapitel wurde näher auf Prozesse eingegangen, die einen Teil dieser Phase ausmachen (vgl. Abbildung 3.1 auf Seite 61 und Erklärung). Dabei ist die Anerkennung der Evidenzen wichtig, die durch ein, am Ausschreibungsprozess beteiligtes Unternehmen, erbracht werden. Die Direktive 2004/18/EC legt einen Grundstein für die Lösung, indem sie Prinzipien für die gegenseitige Anerkennung von Evidenzen definiert (vgl. [7]). Das VCD-Konzept soll die Anwendung der Prinzipien vereinfachen (vgl. [7]). Konkret geschieht dies durch die Bereitstellung der Referenzimplementierungen, wobei das EVS eine entscheidende Rolle bei der Herleitung der benötigten Evidenzen spielt (vgl. Erläuterung NVS und EVS - Abbildung 3.2 auf Seite 62).
Die VCD BIS stellt den Zusammenhang zwischen verschiedenen Vorbedingungen und den Lösungen her. Denn sie erklärt zuerst die Bedingungen, die erfüllt sein müssen, um die Interoperabilität in der europaweiten, elektronischen, sowie öffentlichen Auftragsvergabe zu sichern (vgl. S.6 in [13]). Letztlich stellt sie Anleitungen bereit, die für eine Umsetzung genutzt werden können (vgl. S. 6 in [13]). Diese Anleitungen werden den, im EIF (vgl. Abbildung 2.2 auf Seite 10) definierten, Interoperabilitätsstufen zugeordnet, indem das in Abbildung 3.3 auf Seite 65 dargestellte Interoperabilitäts-Framework eingeführt wird.
Dabei behandelt die Spezifikation die Stufen des vorgestellten IOP-Frameworks, welche

ebenfalls den Aufbau vorgeben. Für diese Arbeit ist vor allem die technische Ebene, und deren Beziehung zu den verbleibenden Ebenen, interessant. Zuerst sollen die verschiedenen Ebenen, dann die Beziehungen zur technischen Ebene, besprochen werden.

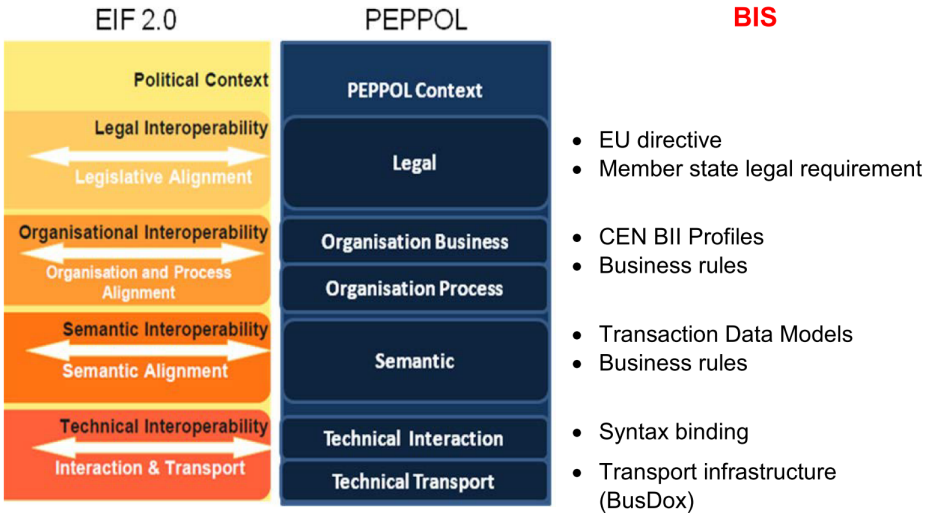


Abbildung 3.3: Abbildung 1 aus [7]: PEPPOL's interoperability framework

Auf der *gesetzlichen Ebene* werden die gesetzlichen Anforderungen bestimmt (vgl. S. 7 in [13]). Als Referenz dient die bereits erwähnte EU-Direktive 2004/18/EC (vgl. S. 12 in [13]), wobei verschiedene Typen von Evidenzen durch eine Taxonomie festgehalten werden (vgl. S. 16 in [13]). Dabei kann eine Evidenz durch den Kandidaten selbst, oder andere, wie z.B. eine Drittpartei, bereitgestellt werden (vgl. S. 17 in [13]). Basierend auf der EU-Direktive wurden die gesetzlichen Abbildungen zu den nationalen Bestimmungen definiert. Diese Abbildungen bestehen aus „eurpäischen“ und „nationalen“ Ontologien, sowie „nationalen“ Abbildungstabellen (vgl. S. 17 in [13]). Die Tabellen selbst basieren auf einem Template, welches Regeln und Prinzipien reflektiert, die bei der Abbildung zu beachten sind. Dabei findet eine Abbildung zwischen den europäischen und nationalen Kriterien statt. Es werden jedoch ebenso die Abbildungen auf die Evidenzen berücksichtigt, die für einen Nachweis genutzt werden können (vgl. S. 17 in [13]). Abbildung 3.4 auf Seite 66 fasst die entstandenden Beziehungen zusammen.
Laut *Mondorf et al.* wird jedoch in verschiedenen Ontologie-Schemata beschrieben, wie die Evidenzen, die Kriterien, sowie die Abbildung zwischen diesen modelliert werden müssen (vgl. S. 55 in [52]). Instanzen (Ontologien) dieser Schemata bilden die grundlegende Menge von Informationen, auf denen die Entscheidungen des EVS beruhen (vgl. S. 55 in [52] - Begriff *reasoners*, sowie Komponenten des EVS - S. 56 in [52]).

Auf der Ebene *Organisation Business* (ORGANISATORISCHE EBENE) wird der Prozess der Angebotsabgabe, und die Rollen der involvierten Geschäftspartner, aufgezeigt (vgl. S. 20-21 in [13]). Die Spezifikation gibt zudem Hinweise darauf, wie gestellte Kriterien interpretiert werden können, und bietet Unterstützung, wenn die benötigten Evidenzen

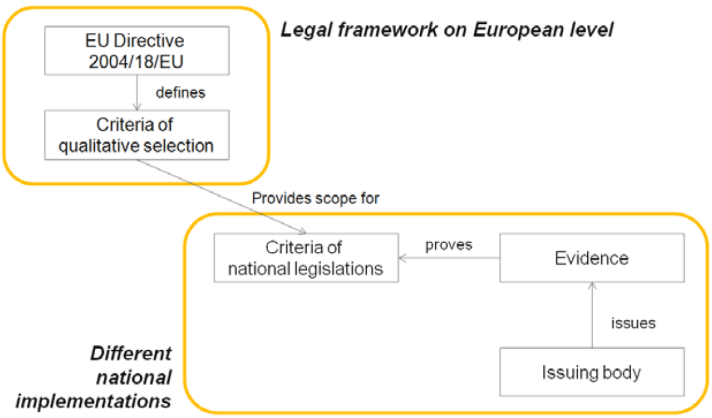


Abbildung 3.4: Abbildung 4 aus [13]: Mapping between different Member State regulations

zusammengetragen, und, als Teil der Qualifikations-Informationen, an den öffentlichen Auftraggeber übertragen werden sollen (vgl. S. 21 in [13]). Dabei werden die Geschäftsbedingungen genannt, die beim Austausch von Qualifikations-Dokumenten beachtet werden müssen, um die gesetzlichen Anforderungen, sowie allgemeine Unternehmensbedürfnisse, zu erfüllen. Letztlich wird beschrieben, wie die Qualifikations-Dokumente (VCDs) zu einem elektronischen Paket (VCD Package) zusammengestellt werden können, ohne das diese Bedingungen verletzt werden (vgl. S. 22 in [13]). Dieser Teil der Spezifikation legt bereits viele Informationen fest, die Teil eines Qualifikationsdokuments sein müssen. Zudem wird auf den Inhalt der elektronischen Pakete eingegangen (vgl. S. 22-24 in [13]).

Auf der Ebene *Organisation Procees* (ORGANISATORISCHE EBENE) wird der Prozess spezifiziert, in dessen Rahmen die elektronischen Pakete, und somit die enthaltenen Qualifikations-Dokumente, ausgetauscht werden (vgl. S. 25 in [13]). Dieser berücksichtigt die Sichtung der öffentlichen Ausschreibung, die Interpretation der Kriterien, sowie das Versenden, und den Erhalt, des zusammen gestellten elektronischen Pakets (vgl. S. 25 in [13]).

Auf der *semantischen Ebene* werden Transaktionsmodelle für VCDs und VCD-Pakete (vgl. S. 52-135 in [13]), sowie Validierungsregeln für diese (vgl. S. 30-33) eingeführt. Zudem wird auf die Nutzung von Bezeichnern (vgl. S. 36 in [13]) und Code-Listen (vgl. S. 37 in [13]) eingegangen.

Auf der *syntaktischen Ebene* wird die Umsetzung der Transaktionsmodelle erwähnt, die mit Hilfe der Universal Business Language - UBL - 2.0 erfolgt (vgl. S. 39 in [13]). Zudem wird ein beispielhaftes VCD-Paket besprochen, um die physikalische Struktur eines solchen, sowie die Nutzung der enthaltenen Informationen, zu zeigen (vgl. S. 46 in [13]).

Die semantische und syntaktische Ebene sind für diese Arbeit von besonderem Interesse und sollen daher in Kapitel 3.1.3 genauer besprochen werden.

3.1.3 Anforderungen an auszutauschende Datenpakete

Die Datenmodelle für die Transaktion basieren jeweils auf einer sogenannten Konzeptdatenbank. Eine solche aggregiert eine Liste von Konzepten aus der VCD-Domäne und den zugehörigen Softwarekomponenten (vgl. S. 27 in [13]), und gibt zudem Regeln an, die für die Nutzung der Konzepte, und für die Erstellung von Werten gelten müssen. Die gesamten Transaktionsmodelle werden in Anhang B der Spezifikation eingeführt (vgl. S. 52-135 in [13]). Laut *Mondorf et al.* nutzt das Datenmodell die Komponenten der UBL, um die zahlreichen Elemente standard-konform zu definieren. VCD-Konzepte wurden nur dann separat spezifiziert, wenn sie nicht durch die UBL-Bibliothek erfasst werden (vgl. S. 45 in [52]). Dabei wird je ein Transaktionsmodell für VCDs und VCD-Pakete definiert, die jeweils in 2 Versionen vorliegen (vgl. S. 52, 88 sowie S. 75, 116 in [13]). Der Zusammenhang zwischen einem VCD und einem VCD-Paket wird u.a. durch Abbildung 3.5 auf Seite 67 dargestellt, die Deliverable 2.4 entnommen wurde (vgl. [52]).

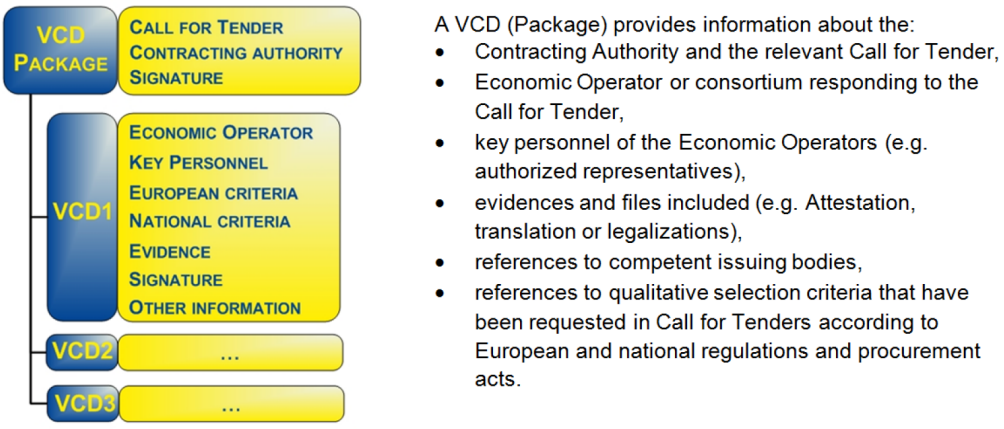


Abbildung 3.5: Abbildung 17 aus [52]: Conceptual view of the VCD data model

Dabei gehen *Mondorf et al.* näher auf den Aufbau eines VCD-Pakets und die enthaltenen Informationen ein. Jedes VCD enthält relevante Informationen über einen Wirtschaftsteilnehmer und seine Fähigkeiten. Weiterhin könne ein VCD-Paket als ein Umschlag betrachtet werden, und enthält u.a. Informationen zur Struktur der Anbieter (*tenderer structure*). Dabei wird die Beschreibung von Bietergemeinschaften und die Auswärtsvergabe von Aufträgen unterstützt (vgl. S. 44 in [52]). Die Informationen werden letztlich mit den enthaltenen VCDs verknüpft (vgl. S. 44 in [52]).
Das Datenmodell ist mit Hilfe von Business Information Entities (vgl. Kapitel 2.2.1 sowie das VCD-Datenmodell[52]) modelliert. Es beschreibt ebenso die Bedeutung der einzelnen

[52]Dieses kann unter *http://www.peppol.eu/peppol_components/peppol-eia/ict-architecture/pre-award-*

BIEs und soll daher verwendet werden, um die Referenzen auf Datenelemente und Dateien zu besprechen. Dazu soll näher auf diejenigen ABIEs eingegangen werden, deren Komponenten im engen Zusammenhang mit den Referenzen stehen:

- *ABIE DocumentReference*: Laut *Mondorf et al.* wird jedes Dokument durch eine Referenz vertreten (vgl. ABIE *DocumentReference* sowie S. 45 in [52]). Zudem wird das physikalische Dokument mit einer URL referenziert, die einen absoluten oder relativen Ort (bzgl. des VCDs) angibt (vgl. BBIE *Attachment.ExternalReference* sowie S. 45 in [52]). Laut *Mondorf et al.* enthält ein VCD Referenzen zu verschiedenen Arten von Dokumenten. Im Datenmodell würden Evidenzen und andere Dokumente jedoch als Gruppe von Dokumenten angesehen, die normalerweise aus einem Kern-Dokument und ergänzenden Dokumenten besteht (vgl. S. 45 in [52] sowie BBIEs *Document.DocumentGroupID*, *Evidence.DocumentGroupID*).

- *ABIE Evidence*: Evidenzen beweisen ein oder mehrere Kriterien (vgl. S. 45 in [52]). Im Datenmodell ist die Angabe mehrerer Evidenz-Dokumente vorgesehen (vgl. ASBIE *Evidence.DocumentReference*). D.h. es erfolgt eine Referenz auf die Dokument-Referenzen, die wiederum auf ein physikalisches Dokument verweisen. Das Datenmodell sieht jedoch ebenso die Angabe der nachzuweisenden Kriterien vor (vgl. S. 45 in [52] sowie BBIE *Criterion.ID* und BBIE *Evidence.Proves. CriterionID*).

- *ABIE Criterion*: Im Datenmodell referenziert jedes Kriterium die jew. Kriterien der nationalen Rechtsvorschrift (bzgl. der Constracting Authroity und des Economic Operator) sowie der europäischen Rechtsvorschrift (vgl. S. 45 in [52] sowie ASBIEs *Criterion.EuropeanRegulation*, *Criterion.EconomicOperatorNational-Regulation*, *Criterion.ContractingAuthorityNationalRegulation*). D.h. das Ergebnis der, durch das EVS vorgenommen, Herleitung wird ebenso durch die, in einem VCD-Paket enthaltenen, VCDs beschrieben. Zu einer Rechtsvorschrift (vgl. ABIE *Regulation*) wird ebenso auf die relevante Ontologie verwiesen (vgl. BBIE *Regulation.URI*).

- *ABIE SingleTenderer*: Durch das Datenmodell wird eine Struktur von Anbietern (EO) unterstützt. Die Struktur kann rekursiv aufgebaut werden (vgl. ASBIE *SingleTenderer.SubcontractorSingleTenderer* - Verweis auf ABIE *SingleTenderer*). Dabei wird innerhalb der Struktur auf das physikalische VCD verwiesen (vgl. S. 45 in [52] sowie BBIE *SingleTenderer.VCDReferenceID*).

Somit werden Dokumente, Evidenzen und Kriterien durch ABIEs modelliert und ebenfalls durch ASBIEs verknüpft. Gleichzeitig definiert die sogenannte Konzeptdatenbank (vgl. S. 52-53 in [52]) die zugehörigen Konzepte: sie dient als Grundlage für die Transaktionsmodelle (vgl. S. 27 in [13]). Referenzen können jedoch ebenso eine Datei als Ziel haben. Zum Beispiel werden die VCDs der, in der *tenderer structure* beschriebenen, Anbieter, und die Nachweise zu den Evidenzen referenziert (vgl. S. 45 in [52]). Zu den

eprocurement/models eingesehen werden (zuletzt geprüft am *25.10.2012*).

Transaktionsmodellen gibt es zudem eine beispielhafte Umsetzung, die in der VCD BIS auszugsweise beschrieben wird (vgl. S. 39-46 [13]). Dazu wird auf den Aufbau eines beispielhaften VCD-Pakets eingegangen, dessen Struktur durch Abbildung 3.6 auf Seite 69 dargestellt wird.

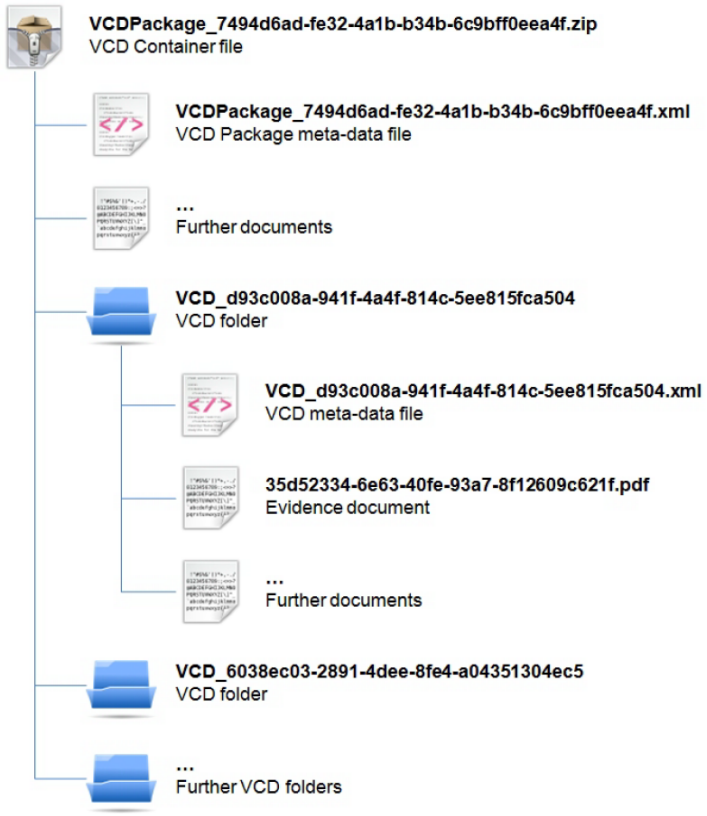


Abbildung 3.6: Abbildung 9 aus [13]: Example folder structure of a VCD Container

Dabei enthalten die dem Muster *VCD_*.xml* entsprechenden Verzeichnisse die VCDs der Wirtschaftsteilnehmer (EO), während die oberste Datei *VCDPackage_....xml* die Informationen zum VCD-Paket enthält (vgl. S. 39 in [13]). Da die UBL keine Dokumenttypen für VCDs und VCD-Pakete vorgibt[53], wurden diese innerhalb des PEPPOL-Projekts definiert (vgl. S. 39 in [13]). Die in den Dateinamen verwendeten IDs dienen ebenso der Referenzierung. So werden die einzelnen VCDs über die im Ordnernamen enthaltende ID referenziert. Die Referenz ist Teil der *tenderer structure*, die im Dokument *VCD-Package_....xml* beschrieben wird (vgl. S. 42 in [13]). Somit werden keine rekursiven Verzeichnisstrukturen genutzt.
Letztlich wurde die Modellierung der Daten auf verschiedenen Ebenen vollzogen:

[53] Die Transaktionsmodelle wurden jedoch mit einem Binding zu UBL 2.0 definiert (vgl. S. 39 in [13]), was durch das Datenmodell ersichtlich wird. In diesem werden die zu benutzenden UBL-Datentypen ausgezeichnet.

- die sogenannte *Konzeptdatenbank* verbindet die Datenmodellierung mit der teilweisen Spezifikation von Softwarekomponenten. Zum Beispiel sollen Entwickler unterstützt werden, die mit der Implementierung von „VCD-Software“ beschäftigt sind (vgl. S. 52-53 in [52]). Dabei basieren die Transaktionsmodelle ebenfalls auf der Konzeptdatenbank. Somit wäre die Konzeptdatenbank als erster Schritt der Modellierung anzusehen und könnte der konzeptuellen Ebene zugeordnet werden (vgl. Abbildung 2.3 auf Seite 19). Neben der Konzeptdatenbank spielen jedoch ebenso Ontologien eine Rolle. Denn zu einer Rechtsvorschrift wird im Datenmodell auf eine entsprechende Ontologie verwiesen (vgl. BBIE *Regulation.URI*). Betroffene Ontologien gehören somit, das Datenmodell betreffend, ebenso zur konzeptuellen Ebene.

- die Transaktionsmodelle können der logischen Ebene zugeordnet werden. Die Transaktionsmodelle geben das physische Format noch nicht vor, da dieses erst durch die Einführung eines Bindings zu UBL 2.0, und der Definition der Dokumenttypen geschieht (vgl. S. 39 in [13]). Durch die Nutzung von BIEs wird jedoch eine (syntax-unabhängige) Strukturierung der Daten vorgegeben (vgl. Definition *logisches Datenmodell* in Kapitel 2.2.1).

- Die Implementierung der Spezifikation wird durch verschiedene XML-Schema-Dateien verwirklicht, wobei jeweils ein XML-Schema für VCD-Pakete und VCDs existiert (vgl. S. 56 in [52]). Zudem werden XML-Schemata für T-, TC- und TCE-Skeletons definiert, die die Basisschemta modifizieren, indem ungenutzte Elemente entfernt werden (vgl. S. 56 in [52]). Diese können ebenfalls der logischen Ebene zugeordnet werden (vgl. Abbildung 2.3 auf Seite 19).

3.2 IMS Common Cartridge

3.2.1 IMS Common Cartridge im Kontext des E-Learning

Laut *Q. Li et al.* spielt die Standardisierung eine wichtige Rolle, um sogenanntes *distance learning* zu fördern (vgl. S. 10:22 in [51]). Zu diesem gehören das verteilte (distributed) und kollaborative (collaborative) Lernen (vgl. S. 10:2 in [51]). Die Ansätze legen unterschiedliche Schwerpunkte, stützen sich jedoch auf Computer- und Kommunikationstechnologien, um die einfachere Teilnahme an Lernaktivitäten zu ermöglichen (vgl. S. 10:2 in [51]). Dabei werden Inhalte mit der Hilfe von sogenannten Lernmanagement-Systemen - LMS - zur Verfügung gestellt, die in einer engen Beziehung zu Content-Management-Systemen - CMS - stehen (vgl. S. 10:3 in [51]).
Solche Systeme unterstützen meist ein Learning Object Repository - LRO, dessen Inhalte mit Hilfe von Standards repräsentiert werden (vgl. S. 10:3 in [51]). Dabei ist ein einzelnes Lernobjekt (laut dem IEEE LOM Standard - vgl. [4]) wie folgt definiert[4]:

> „For this Standard, a learning object is defined as any entity -digital or non-digital- that may be used for learning, education or training.“

Ein zentraler Aspekt von Lernobjekten scheint Ihre Wiederverwendbarkeit zu sein. Denn laut *Duval und Hodgins* werden Lernobjekte mit wiederverwendbaren, multimedialen, Inhalts-Komponenten in Verbindung gebracht. Die Wiederverwendung selbst kann auf unterschiedliche Art und Weise erfolgen, wobei die Autoren verschiedene Arten der Wiedererwendung nennen. Dabei sei der Ausdruck *repurposing* spezifischer, und beschreibe die Fähigkeit, den gleichen Inhalt zu einem Zweck zu nutzen, der sich signifikant von der ursprünglichen Bestimmung unterscheidet (vgl. [28]). Der LOM Standard bietet Unterstützung bei der Beschreibung von Lernobjekten. Er beschreibt Datenelemente, die für die Beschreibung der relevanten Eigenschaften eines Lernobjekts genutzt werden können (vgl. S. 5 in [4]).
Dabei spielt der Austausch von Lernobjekten eine wichtige Rolle. *Q. Li et al.* sprechen an, dass ein gemeinsames Vokabular für LMS und Materialien, sowie eine einheitliche Repräsentation dieser benötigt werden, so dass die Interoperabilität verschiedener LMS, und der Austausch von Lernobjekten ermöglicht werden (vgl. S. 10:22 in [55]). Dabei gehen *Muñoz-Merino et al.* auf die Einheitlichkeit von LMS ein. Sie verweisen dazu auf ein, durch *Dagger et al.* beschriebenes, Szenario, in dem ein LMS auf modularen Komponenten basiert, und Dienste anbietet, die an keine spezifische Platform gebunden sind (vgl. [22] sowie [55]). Diese Philosophie wird ebenso durch das IMS[54] Abstract Framework und die Open Knowledge Initiative - OKI - Architektur verfolgt (vgl. [55]).
Laut *Dagger et al.* identifiziert das IMS-Abstract-Framework die grundlegenden Komponenten und Interfaces eines eLearning Systems, während die OKI Service-Ebenen zur Entwicklung von eLearning-Platformen definiert, die die Funktionalität in verschiedene Module unterteilt (vgl. S. 3 in [22]). Auf einem fein-granulareren Level existieren Standards und Spezifikationen, die die Syntax beschreiben, die die verschiedenen Services implementieren sollten (vgl. S. 3 in [22]). Die Autoren bieten eine kurze Übersicht und nennen verschiedene Spezifikationen, zu denen auch die IMS-Content-Packaging - IMS CP - Spezifikation gehört (vgl. S.3 in [22]). Diese beschreibt ein Format, welches für den Austausch von Lerninhalten benutzt werden kann. Dabei werden die Lerninhalte zu einem Dateipaket zusammengefasst. Enthaltene Dateien können z.B. Aufgaben und deren Lösungen beinhalten (vgl. [23]). Die Spezifikation steht im engen Zusammenhang mit dem, in dieser Arbeit zu besprechenden, IMS-Common-Cartridge - IMS-CC - Profil (vgl. Kapitel 2.4.1 für die Definition eines *Applikationsprofils*). Dieses basiert auf mehreren Spezifikationen (vgl. S. 6 in [64]), und wurde im Einklang mit den IMS-Application-Profiling-Guidelines entwickelt (vgl. S. 6 in [64], Kapitel 2.4.1 sowie [63],[65]). Eine der zu Grunde liegenden Spezifikationen ist die IMS-CP-Spezifikation (vgl. S. 6 in [64]). Laut *Dahn* schließt das Profil mehrere Profile verschiedener Spezifikationen mit ein, die mit Hilfe eines Profils der IMS-CP-Spezifikation verbunden sind (vgl. S. 65 in [24]).
Die Einsatzgebiete der Spezifikation werden durch das in Abbildung 3.7 auf Seite 72 abgebildete Diagramm vorgestellt, welches verschiedene Use-Cases zusammenfasst (vgl. S. 11 in [64]).
Dabei wird ein LMS als ein Programm angesehen, das die Zuweisung von Inhalten (an

[54]IMS ist eine Organisation, die nicht auf Profit ausgerichtet ist, und eine Menge von Spezifikationen definiert hat, die verschiedene Aspekte des E-Learning betreffen (vgl. S. 485 in [55]).

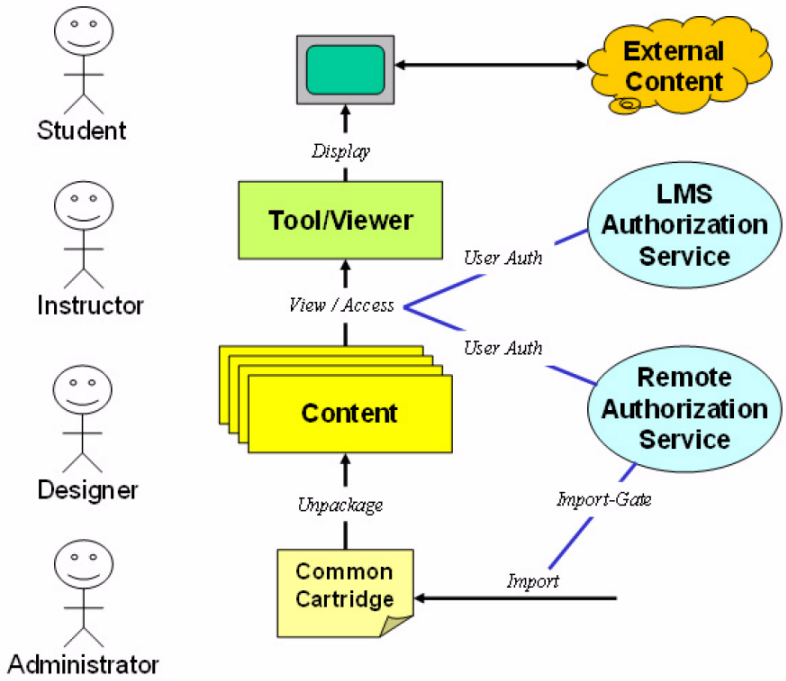


Abbildung 3.7: Abbildung 2.1 aus [64]: Use Case Framework

Studenten), das Lernen an sich, und die Auswertung der Lernergebnisse erlaubt (vgl. S. 10 in [64]). Die Use-Cases beinhalten 2 grundlegende Abläufe, die den Import der, in einer IMS-CC enthaltenen, Inhalte, sowie den Zugriff auf die importierten Inhalte beschreiben (vgl. S. 12 und 13 in [64]). Darauf aufbauend kann die Ausführung von statischen, mittels einer Cartridge importierter, Lerninhalte gefordert sein (vgl. S. 14 in [64]). Letztlich kann es notwendig sein, dynamische Inhalte auszuführen. Dies kann mittels eines client-seitigen Viewers / Players geschehen, oder mit den durch das LMS bereitgestellten Funktionalitäten (vgl. S.15-18 in [64]). Die Use-Cases sollen im folgenden kurz erklärt werden:

- Bei dem *Import einer Cartridge* benutzt ein Dozent (Instructor) ein durch das LMS bereitgestelltes Tool. Dabei kann eine Authentifizierung gefordert sein (vgl. Kante *import-gate* in Abbildung 3.7 auf Seite 72). Der Dozent kann zudem einzelne, in der Cartridge enthaltene, Materialien auswählen. Das LMS muss sicherstellen, dass die benötigten Abhängigkeiten erfüllt sind. (zusammengefasst aus S. 12 in [64])

- eine *Autorisierung* kann auch dann nötig sein, wenn ein Lerner auf einen Kurs (im LMS) zugreifen möchte, und dieser Material einer Cartridge beinhaltet. Denn eine solche kann wiederum Elemente beinhalten, die nur eingeschränkt zugänglich sein sollen (sogenannte *restrictive items*). Dabei wird ein sogenannter Autorisierungs-service genutzt, um ein, durch den Lerner bereitgestelltes Token, zu verifizieren. (zusammengefasst aus S. 13 in [64])

- bei der *Ausführung von statischen Inhalten* rendert das LMS sowohl Navigationselemente, die LMS- als auch Cartridge-spezifisch sind. Als Beispiele werden u.a. Links, Verzeichnisse, und eine Baumansicht genannt. Bei der Wahl bestimmter Links erstellt das LMS passende Darstellungselemente für die jeweiligen Ressourcen. (vgl. S. 14 in [64])

- die *Ausführung von dynamischen Inhalten* steht mit der Absicht eines Studenten in Verbindung, Zugriff auf eine Lernaktivität zu erlangen, und diese abzuschließen. Die Realisierung ist etwas umfangreicher, und soll nicht im Detail besprochen werden. Wichtige Schritte sind die Anzeige eines sogenannten Lernmoduls[55], und die Interaktion mit diesem. Bei der *client-seitigen Ausführung* werden diese Aufgaben durch einen Viewer / Player übernommen. Dabei muss der Kontext im LMS, sowie eine teilweise Kontrolle, an den Viewer / Player übergeben werden. Auch die Ergebnisse müssen dem LMS mitgeteilt werden. Bei der *server-seitigen* Ausführung sind diese Schritte nicht notwendig. (zusammengefasst aus S. 18 in [64])

Der Zusammenhang zwischen diesen Use-Cases und dem IMS-CC Format ergeben sich durch den Import einer Common-Cartridge. Laut *Dahn* wird eine Cartridge „in ihre einzelnen Ressourcen aufgelöst, die dann von Lehrenden frei mit eigenen Ressourcen oder mit den Ressourcen anderer Cartridges kombiniert werden können“. Laut dem Autor übernimmt der Lehrende ebenso die Entscheidung, welche Ressourcen für den Lernenden zugänglich sind (vgl. [23]). Laut der IMS-CC-Spezifikation werden beim Import einer Cartridge proprietäre, implementierungs-abhängige, Datenstrukturen erzeugt, um die Inhalte und die Struktur zu speichern (vgl. S. 12 in [64]). Diese Strukturen werden somit im Rahmen der beschriebenen Use-Cases verwendet.
Dabei können die Lerninhalte bzw. -materialien durch einen Kurs- bzw. Programmdesigner erstellt werden (vgl. S. 11 in [64]). Letztlich obliegt es Administratoren, die Wartung von administrativen Elementen eines LMS zu übernehmen (vgl. S. 11 in [64]).

3.2.2 Maßnahmen zur Sicherung der Interoperabilität

Neben dem Format der auszutauschenden Pakete legt das IMS-CC Profil ebenso Funktionalitäten fest, die von einem LMS unterstützt werden müssen, um einen fehler-freien Import von Inhalten zu gewährleisten (vgl. S. 7 in [64]). Ein spezifisches Runtime-Modell wird allerdings nicht festgelegt (vgl. S. 7 in [64]). Es wird jedoch ein beispielhafter Aufbau gezeigt, welcher die Funktionalitäten bzgl. einer möglichen LMS-Architektur einordnet (vgl. Abbildung 1.1 auf S. 7 in [64]). Die Import-Funktion wird dem LMS selbst, die restlichen Funktionen der Runtime zugeordnet. Eine Ausnahme bildet die Autorisierung, für die ebenso ein externer Service genutzt werden kann.
Die zu unterstützenden Funktionalitäten werden unter anderem durch die Use-Cases deutlich, die im letzten Kapitel verkürzt vorgestellt wurden (vgl. Beschreibung der Use-Cases - S. 11-18 in [64]), und die die Entwicklung vorangetrieben haben (vgl. S. 7 in

[55] Ein Lernmodul ist ein Aggregat von Inhalten und/oder Programmfunktionalitäten, welches eine Lernaktivität repräsentiert, oder ein Teil einer solchen ist (frei übersetzt aus [64]).

[64]). In den Use-Cases werden wiederum Daten bemüht, die aus einer Common Cartridge stammen (vgl. Beschreibung der Use-Cases - S. 11-18 in [64]). Letztlich haben diese eine Repräsentation (vgl. *Appendix B* in [64]).
Diese einzelnen Aspekte lassen sich mit den unteren Ebenen des European Interoperability Frameworks - EIF - in Verbindung bringen. Abbildung 3.8 auf Seite 74 zeigt eine Zuordnung, welche die verschiedenen Teile der IMS-CC-Spezifikation klassifiziert:

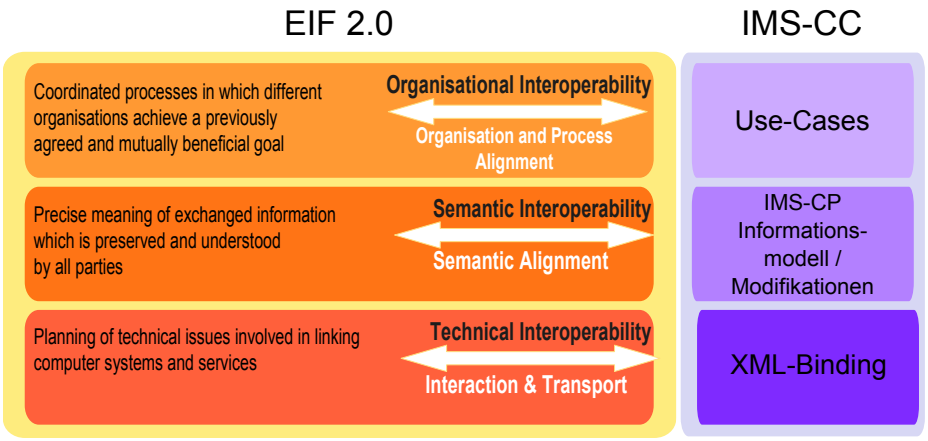


Abbildung 3.8: Klassifizierung der in der IMS-CC Spezifikation beschriebenen IOP-Anforderungen

- die *Use-Cases* können der organisatorischen Ebene zugeordnet werden. Diese erfassen jedoch nicht die Zusammenarbeit verschiedener Organisationen. Vielmehr beschreiben diese Abläufe, in die verschiedene Beteiligte eingebunden sind. Diese werden mittels ihrer Rollen klassifiziert (vgl. S. 11 in [64]). Dabei wird ein LMS als beteiligtes System betrachtet (vgl. S. 11 in [64]). Jeder Use-Case wird mit der Hilfe von verschiedenen Eigenschaften beschrieben, die in Abbildung 3.9 auf Seite 74 gezeigt werden.

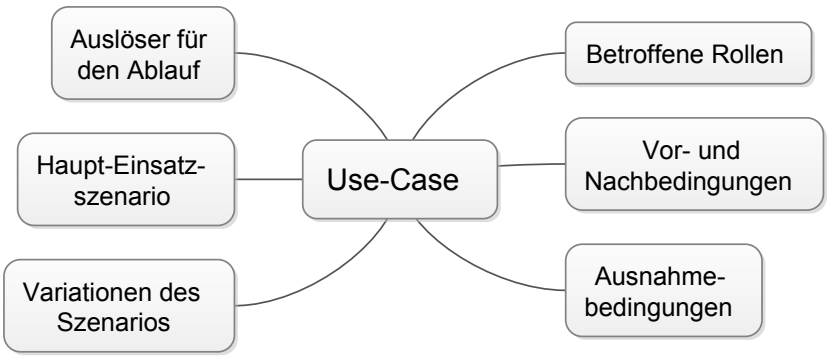


Abbildung 3.9: Teile der Beschreibung eines Use-Cases

Der eigentliche Ablauf wird mit Hilfe eines Haupt-Einsatzszenarios beschrieben, welches verschiedene Aktionen miteinander verbindet. Diese haben eine vorgegebene Reihenfolge und bauen aufeinander auf. Ein Teil der Aktionen beruht auf Daten, die mit einer Common-Cartridge in Verbindung gebracht werden können. Zum Beispiel auf den, beim Import erstellten, Datenstrukturen. Somit wird das Einsatzgebiet der Spezifikation, und die Verwendung der Daten beschrieben, die durch die verschiedenen Beteiligten eines Ablaufs erfolgt.
Zusätzlich werden Variationen des Hauptszenarios berücksichtigt. Diese beschreiben zusätzliche Schritte, die das Hauptszenario ergänzen. Weiterhin werden mögliche Gründe angegeben, die den Anlass für den Start eines Ablaufs bilden können. Ein erfolgreicher Ablauf setzt jedoch gewisse Vorbedingungen voraus. Für den Endzustand existieren jeweils Bedingungen, die die Umstände nach einer erfolgreichen, bzw. fehlgeschlagenen, Abarbeitung beschreiben.
(vgl. Beschreibung der Use-Cases - S. 11-18 in [64])

- das sogenannte *Informationsmodell* der IMS-Content-Packaging - IMS-CP - Spezifikation bildet die Basis für das Common-Cartridge-Profil. Eine Common-Cartridge ist ein IMS-Content-Package, welches einer bestimmten Struktur folgt (vgl. S. 28 in [64]). Dabei gilt es zwischen der Struktur eines Pakets, und der Struktur des sogenannten *Manifests* zu unterscheiden, das den Paketinhalt beschreibt (vgl. S. 13 in [73]). Das Informationsmodell beschreibt den Aufbau eines solchen Manifests (vgl. S. 16ff in [73]). Die beschriebenen Strukturen werden mit Hilfe eines Bindings (vgl. Kapitel 2.4.1) mit einer XML-Repräsentation verbunden (vgl. S. 6 in [73]). In der Beschreibung des Informationsmodells wird auf die möglichen Inhalte eines Pakets eingegangen. Z.B. im Rahmen von Informationselementen, die Referenzen auf, im Paket enthaltene, Dateien beinhalten können (Beispiel: vgl. Definition der Klasse File - S. 29 in [73]). Die vorgenommen Modifikationen (vgl. S. 28-57 in [64]) schränken die gültige Struktur eines Manifests, und die der konformen Dateipakete, ein. Dabei werden andere Spezifikationen, oder Profile von diesen, mit einbezogen. Zum Beispiel werden die Art der unterstützten Ressourcen eingeschränkt (vgl. S. 28 in [64]). D.h. es gelten Einschränkungen für bestimmte Referenzen in einem Manifest, und die referenzierten Inhalte (Beispiel - vgl. *Content Type* Question Bank - S. 35 in [64]). Eine andere Anbindung kann mit der Hilfe von sogenannten Extension-Points umgesetzt werden. Diese ermöglichen die Wiederverwendung von Datenstrukturen, die durch die Spezifikation nicht vorgegeben sind (vgl. S. 63 in [24]). Die IMS-CP-Spezifikation sieht z.B. Extension-Points vor (vgl. Klasse ManifestMetadata - S. 19-20 in [73] sowie Klasse Metadata - S. 29-30 in [73]). Dabei legt die IMS-CC-Spezifikation zu verwendende Datenstrukturen fest (vgl. 36-38 in [73]). Zudem können Anforderungen für angebundene Datenstrukturen, sowie für die Inhalte referenzierter Dateien, gelten, die wiederum durch ein Profil beschrieben sind (vgl. S. 47ff in [64] - CC Profile of QTI und S. 36ff in [64] - LOM Metadata). Diese Kombination von verschiedenen Profilen wird auch *Domain Profile* genannt (vgl. S. 62 in [24]).

- das sogenannte *XML-Binding* gibt die konkrete XML-Repräsentation eines Manifests vor[56]. Auch für andere, angebundene Spezifikationen, existieren Bindings, die die Form von XML-Dateien vorgeben (vgl. Appendix B: S. 72-73 in [73]). Dabei wird ein Applikationsprofil bereitgestellt, das der logischen Ebene zugeordnet (vgl. Kapitel 2.2.1 sowie Kapitel 2.4.3), und durch eine Applikation namens IMS-SchemaProf verarbeitet werden kann. Jedes verwendete Profil basiert auf einem XML-Binding der jeweiligen Spezifikation, welches durch ein XML-Schema vorgegeben ist (vgl. S. 74-88 in [64]). Die Modifikationen der Profile werden mit Hilfe eines XML-Formats festgehalten. Die geänderten XML-Schemata sind ebenso Teil des jeweiligen Profils. Für das Hauptprofil (des IMS-CP-Schemas) werden zudem Schematronregeln angegeben (vgl. S. 109ff in [64]).

3.2.3 Anforderungen an auszutauschende Datenpakete

Für die nähere Beschreibung des Datenformats soll die Beziehung zwischen dem Manifest und der Struktur eines Dateipakets genauer herausgearbeitet werden. Die IMS-CC-Spezifikation veranschaulicht den Aufbau einer Common-Cartridge mit Hilfe von Abbildung 3.10 auf Seite 76, deren Komponenten kurz erläutert werden sollen:

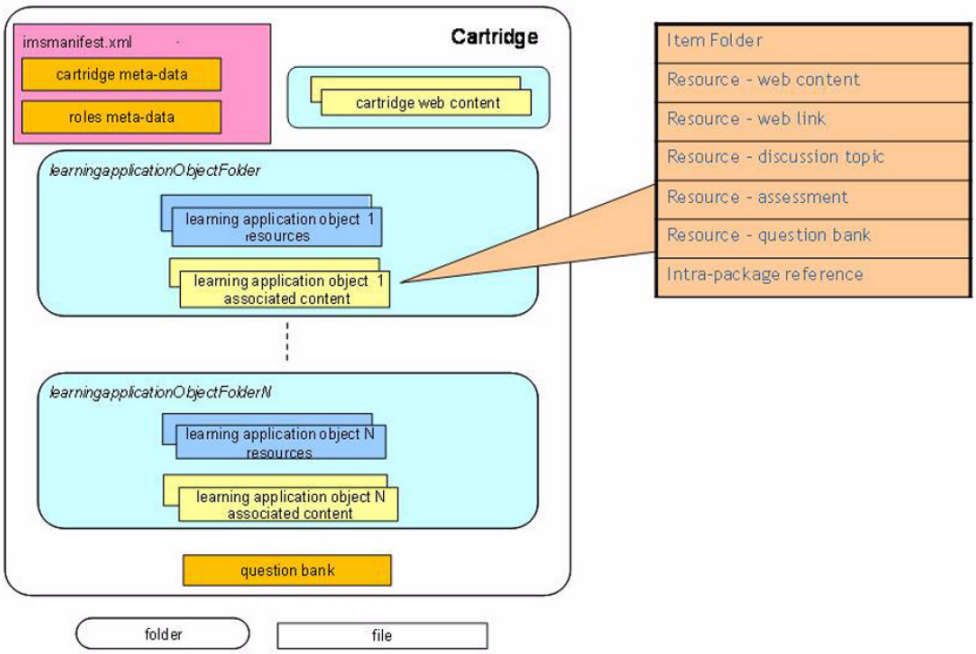


Abbildung 3.10: Abbildung 3.3 aus [64]: Common Cartridge package interchange file.

- das *Manifest* wird durch die Datei IMSMANIFEST.XML beschrieben, die dem XML-

[56] Vgl. S. 14, 16ff in [73]: Die Struktur eines Manifests wird mit Klassen beschrieben, die an spezifische Datenstrukturen gebunden sind.

Binding des, durch die IMS-CC-Spezifikation, modifizierten Informationsmodells entspricht (vgl. Tabelle 3.3.1 auf S. 22 in [64], Erklärung des Manifest-Begriffs auf S. 13 in [73], sowie Erklärung zum Binding auf S. 14 in [73]).

- eine Common-Cartridge kann einen *Web-Content-Ordner* enthalten, der Dateien mit Webinhalten (vgl. S. 21, 23, 34 in [64]), XML-Beschreibungen von HTTP-Links (vgl. S. 21, 23, 34-35, 41-42 in [64]), sowie paket-interne Referenzen beinhalten kann. Letztere verweisen wiederum auf andere Dateien im Paket (vgl. S. 21, 23 in [64]). Referenzen unter den Dateien, sowie von Dateien außerhalb des Ordners, sind erlaubt. Referenzen zu außerhalb gelegenen Dateien hingegen nicht (vgl. S. 23 in [64]).

- ein *Lernobjekt* ist eine Ordnerstruktur, die für die Gruppierung verschiedener, Ressourcen-spezifischer, Dateien genutzt wird (vgl. S. 23 in [64]). Dabei enthält der oberste Ordner ein *descriptor file*. Der Inhalt einer solchen Datei muss evtl. einem anderen Profil entsprechen (Bsp.: Assessment Content File - S. 35 in [64]). Die Dateien im Ordner (*learningapplicationObjectFolder*) sind an das Lernobjekt gebunden, und können nicht von Ressourcen referenziert werden, die sich außerhalb des Ordners befinden (vgl. S. 23 in [64]).

- die sogenannte *Question Bank* ist optional und kann Fragen enthalten, die durch ein Profil der Question & Test Interoperability[57] - QTI - Spezifikation unterstützt werden.

Dabei gruppiert ein Lernobjekt verschiedene Dateien, die der Beschreibung einer Ressource dienen. Diese hat wiederum einen bestimmten Typ (vgl. S.23 sowie S. 34-35 in [64]), und ist durch eine gleichnamige Klasse modelliert (vgl. S. 30 in [64]). Die IMS-CC-Spezifikation nutzt ein UML-Profil, um das Datenmodell zu veranschaulichen (bzgl. des Manifests - vgl. S. 29-31 in [64]). In diesem Kapitel sollen jedoch Abbildungen genutzt werden, die eine weitere Abstraktion darstellen. Abbildung 3.11 auf Seite 78 zeigt wichtige Eigenschaften eines Resource-Objekts.
Die blauen Knoten (bzw. dunkleren) Knoten stehen für die Kinder eines Resource-Objekts, während die hellblauen (bzw. helleren) Knoten für Eigenschaften stehen, die näher betrachtet werden sollen. Der Typ einer Ressource ist eine Eigenschaft, die durch ein Attribut beschrieben wird (vgl. S. 30 in [64]). Der Wert des Attributs muss in einer vorgegebenen Liste von Strings enthalten sein (vgl. Beschreibung der Inhaltstypen - S. 34-35 in [64]). Die zur Ressource gehörenden Dateien werden durch Kindobjekte (File) erfasst, die ein Attribut beinhalten, dass die jeweilige Datei referenziert (vgl. Klasse File - S. 29 in [73]). Bei den Inhaltstypen 'Web Content' und 'Associated Content' kann eine Ressource eine Referenz auf die ausführbare Datei beinhalten (vgl. S. 34 in [64]).
Kindobjekte vom Typ Dependency ermöglichen die Referenzierung von Dateien, die durch andere Ressourcen erfasst werden (vgl. Klasse Dependency - S. 29 in [73]). Das Ziel

[57] Die QTI-Spezifikation ermöglicht die Repräsentation von Fragen- und Test-bezogenen Daten, sowie den zugehörigen Resultaten (vgl. [19]).

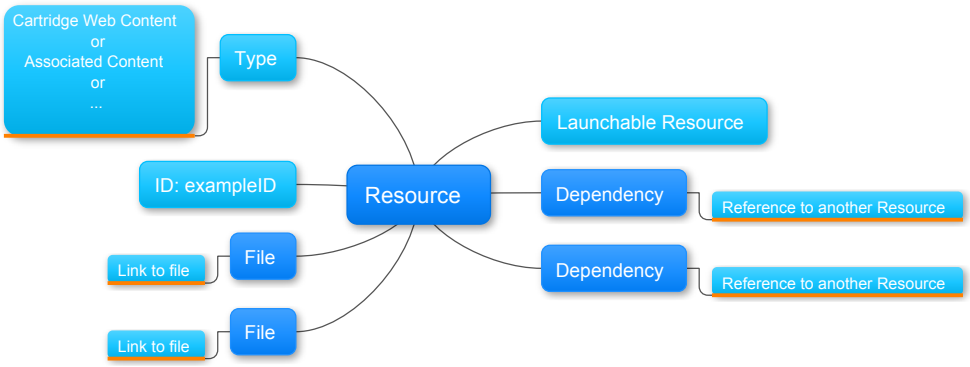


Abbildung 3.11: Mögliche Struktur einer Ressource (im Datenmodell)

wird durch eine ID festgelegt, die durch ein Resource-Objekt definiert werden kann (vgl. Klassen Identifier und IdentifierRef - S. 37-38 in [73][58]). Je nach Typ der Ressource ist die Referenzierung anderer Ressourcen nicht erlaubt, oder eingeschränkt. Einschränkungen betreffen den Typ der Ziel-Ressource. Zum Beispiel dürfen Ressourcen vom Typ 'Web Link' keine anderen Ressourcen referenzieren, während Ressourcen vom Typ 'Associated Content' auf Ressourcen vom Typ 'Web Content' verweisen dürfen (vgl. Beschreibung der Inhaltstypen - S. 34-35 in [64]).
Auch die Nutzung von File-Objekten und einem Link zu einer ausführbaren Datei hängt vom Typ der Ressource ab. Ein Beispiel sind Ressourcen vom Typ 'Discussion Topic'. Eine solche darf keinen Link enthalten, und muss genau ein File-Objekt definieren, welches ein XML-Descriptor-File referenziert (vgl. S. 35 in [64]). D.h. die Vorgaben für den Aufbau eines konkreten Resource-Elements (XML) hängen vom Inhalt eines konkreten Manifest-Dokuments ab[59].
Die Erfassung der Ressourcen ist jedoch nicht die einzige Aufgabe eines Manifests. Ein solches kann ebenso eine Organisation der Inhalte definieren, die die grundlegende Navigationsstruktur festlegt (vgl. S. 28 in [64] sowie Use-Case *Ausführung von statischen Inhalten* in Kapitel 3.2.2). Ein Beispiel für den Aufbau einer solchen Struktur wird in Abbildung 3.12 auf Seite 79 gezeigt. Die dunkelbauen (bzw. dunkleren) Knoten stellen, wie in der vorigen Abbildung auch, Kind-Objekte dar, während die hellblauen (bzw. helleren) Knoten den Typ der Objekte beschreiben.
Ein Manifest kann höchstens eine Organisation definieren, was eine Einschränkung der IMS-CP-Spezifikation darstellt (vgl. S. 32 in [64]). Die Struktur wird durch Objekte vom Typ 'Item' festgelegt, welche für die Beschreibung einer Baumstruktur genutzt werden können (vgl. 34-35 in [64]). Ein Objekt vom Typ 'Organization' darf nur ein einzelnes 'Item'-Objekt beinhalten, welches die Struktur beherbergt (vgl. *Root Folder* - S. 34 in [64]). Für die Beschreibung der eigentlichen Struktur können weitere Objekte vom Typ

[58] Es gelten Einschränkungen für eine Referenz. Vgl. Klasse IdentifierRef (S. 38 in [73]).

[59] Dies ist wichtig für die Anbindung der weiteren Profile. Wird während eines Tests eine Referenz auf ein Descriptor-File gefunden, so muss das Profil bestimmt werden, das für einen Test der Datei verwendet werden kann (vgl. - Realisierung der Additional-Constraints-Tests in Kapitel 4.3).

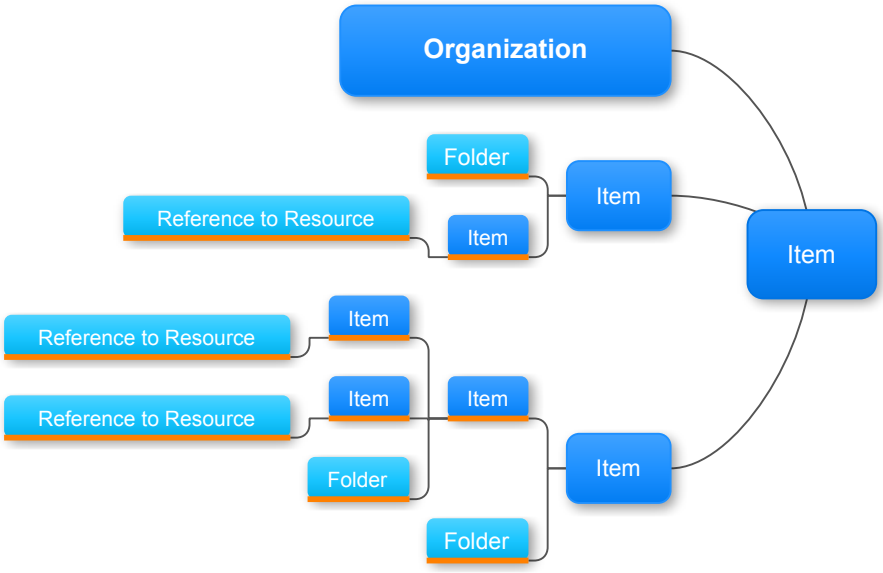


Abbildung 3.12: Mögliche Organisation von Inhalten (im Datenmodell)

'Item' verwendet werden:

- Für *innere Knoten* können 'Item'-Objekte verwendet werden, die keine Ressource referenzieren (vgl. S. 33 in [64]).
- Für *Blätter* muss jedoch eine Referenz zu einer Ressource angegeben werden (vgl. S. 34 in [64]).

Für die Referenz wird der Identifier verwendet, der zum jeweiligen Resource-Objekt gehört (vgl. Klassen Resource, Item, Identifier und IdentifierRef - S. 28, 26, 37, 38 in [73]). In der beispielhaften, in Abbildung 3.11 auf Seite 78 gezeigten, Abstraktion eines Resource-Objekts, hat der Identifier den Wert 'exampleID'. Jedoch dürfen nur Resource-Objekte referenziert werden, die einen anderen Typ wie 'Question-Bank' haben (vgl. S. 35 in [64]). Zudem können Bedingungen für den Aufbau eines Resource-Objekts gelten, wenn dieses aus einem Item-Objekt heraus referenziert wird (vgl. Inhaltstyp *Web Content* - S. 34 in [64])[60].
Die Beschreibungen der Ressourcen und der Organisation wird durch ein Objekt vom Typ 'Manifest' erfasst (vgl. Klasse *Manifest* - S. 29 in [64]). Eine Abstraktion dieses Objekts wird in Abbildung 3.13 auf Seite 80 dargestellt:

- ein Objekt vom Typ *Organizations* beinhaltet die Organisation der Inhalte (welche sich Referenzen auf *Resource-Objekte* zu Nutze macht - vgl. Abbildung 3.13 auf Seite 80).

[60]Diese Auflagen sind weitere Beispiele für Bedingungen, die auf dem Inhalt eines konkreten Manifests aufbauen (vgl. bedingte Modifikationen in Kapitel 2.4.2, sowie entsprechende Modifikationen in Kapitel 3.4.2 und Kapitel 3.4.3).

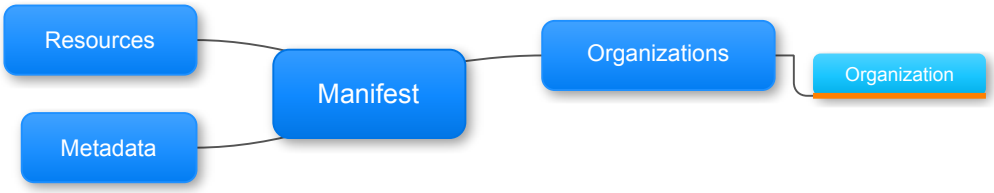


Abbildung 3.13: Vereinfachter Aufbau eines Manifests (Datenmodell)

- ein Objekt vom Typ *Resources* ist der Vater von mehreren *Resource-Objekten* (die zugehörige Dateien mit Hilfe von *File-Objekten* referenzieren - vgl. Beschreibung zu Abbildung 3.12 auf Seite 79).
- ein Objekt vom Typ *ManifestMetadata* beschreibt die Common-Cartridge als ganzes (vgl. Klasse ManifestMetadata - S. 19-20 in [73]). Allerdings wurde diese Klasse modifiziert, und die Art der unterstützten Metadaten eingeschränkt. Diese müssen einem Profil entsprechen, das auf dem Binding der IEEE-LOM-Spezifikation basiert (vgl. S. 36 in [64]).

Abschließend kann festgestellt werden, dass die Datenmodellierung auf verschiedenen Ebenen erfolgt ist, die dem Modell von *Routledge et al.* (vgl. Kapitel 2.2.2) zugeordnet werden können:

- die IMS-CP-Spezifikation kann mit der *konzeptuellen Ebene* in Verbindung gebracht werden. Denn sie nutzt UML-Diagramme, um eine informative Übersicht über das Informationsmodell (vgl. Kapitel 2.4.1) zu bieten. Diese werden als *Platform Independent Models* - PIM - bezeichnet. Jede Klasse wird nochmals normativ definiert. Für die UML-Diagramme wird ein UML-Profil genutzt, die Klassen werden mit Hilfe von Tabellen und vordefinierten Termen spezifiziert. Sowohl das UML-Profil, als auch die verwendeten Terme werden in [17] definiert (vgl. S. 14 in [73]). Das Informationsmodell ist die Grundlage für das Binding[18], welches ebenso mit der Hilfe von UML-Diagrammen ausgedrückt ist, die wiederum als *Platform specific Models* - PSM - bezeichnet werden. Diese erlauben allerdings die Herleitung der XML-Schemata (vgl. [18]). Für diese Art der Beschreibung existiert ebenso ein UML-Profil (vgl. [18]).
- die UML-Diagramme der IMS-CC-Spezifikation können der *logischen Ebene* zugeordnet werden. Diese werden als PSM bereitgestellt, wobei die Änderungen in den UML-Diagrammen, und den näheren Erläuterungen, ausgeführt werden (vgl. [64]). Dabei dient das Binding der IMS-CP-Spezifikation als Grundlage (vgl. UML-Diagramme in [64] und [18]). Somit liegt eine „Zwischenform“ vor, die mit dem konzeptuellen Modell der IMS-CP-Spezifikation in Verbindung gebracht, aber ebenso zur Herleitung der XML-Schemata genutzt werden kann. Die Änderungen am Binding der IMS-CP-Spezifikation fanden im Einklang mit den IMS-Application-Profiling-Guidelines statt (vgl. S. 6 in [64], Kapitel 2.4.1 sowie [63],[65]). Das Profil

schließt jedoch mehrere Spezifikationen, und Profile von diesen, mit ein (vgl. S. 6 in [64]). Verbunden werden diese jedoch durch das Hauptprofil, welches auf der IMS-CP-Spezifikation basiert (vgl. S. 65 in [24]).

- die XML-Schemata können der *physischen Ebene* zugeordnet werden. Sie können aus einem PSM hergeleitet werden (vgl. Begriff PSM in [18]). Das Applikationsprofil beinhaltet die formalen Dateien, die die Änderungen an den XML-Bindings der involvierten Spezifikationen festhalten. Diese sind die formale Repräsentation eines Profils auf logischer Ebene (vgl. Kapitel 2.4.3). Ebenso werden XML-Schemata bereitgestellt, die diese Änderungen bereits reflektieren (vgl. S. 72ff in [64]). Zusätzliche, nicht durch XML-Schema zu erfassende, Bedingungen werden durch ein Schematron-Schema bereitgestellt (vgl. Appendix B - S. 109ff in [64]).

3.3 Gemeinsame Anforderungen

In den letzten beiden Kapiteln wurden die Spezifikationen *Virtual Company Dossier* - VCD - und *IMS-Common-Cartridge* - IMS-CC - vorgestellt (vgl. Kapitel 3.1 und Kapitel 3.2). Beide Spezifikationen legen ein Format für Datenpakete fest, die innerhalb von vorgesehenen Abläufen ausgetauscht werden. Die Abläufe zeigen das jeweilige Einsatzszenario, und gehen auf die Interaktionen ein, die zwischen den verschiedenen beteiligten Personen (klassifiziert nach Rollen) und IKT-Systemen vorgesehen sind (vgl. [52] - z.B. Abbildung 23 auf S. 59 - sowie die Beschreibung der *Use Cases* in [64]). Die Beschreibung dieser Abläufe können der organisatorischen Ebene des EIF zugeordnet werden (vgl. Kapitel 3.2.2, Kapitel 3.1.2, sowie S. 35 in [52]). Das Datenmodell kann dagegen der semantischen Ebene zugeordnet werden (vgl. Begriff *Transaction Data Models* - S. 27 in [13]). Dabei gibt das jeweilige Binding das Format von XML-Dateien vor, die zusammen den Inhalt des Pakets beschreiben (vgl. Kapitel 3.1.3 sowie Kapitel 3.2.3)[61]. Bei der VCD-Spezifikation werden gleich zwei Datenformate eingeführt, wobei die Beschreibung eines VCD-Pakets auf andere Beschreibungen zurückgreifen kann, die ein einzelnes VCD beschreiben (vgl. Kapitel 5 und 6 in [13], sowie Kapitel 3.1.3). Die IMS-CC-Spezifikation legt ein Format für eine sogenannte Manifest-Datei fest (vgl. Kapitel 3.2.3, sowie Erklärung der Datei IMSMANIFEST.XML in [64]). Dabei wird der Inhalt der Datei durch ein sogenanntes *Domain-Profile* (für eine Erklärung siehe Kapitel 2.4.3) vorgegeben, das auf mehreren Spezifikationen basiert (vgl. S. 6 in [64]).
Im folgenden sollen die Gemeinsamkeiten der Datenmodelle[62] beschrieben werden. Insbesondere soll auf Aspekte eingegangen werden, die nur schwer mittels eines XML-Schemas erfasst werden können. Denn diese Aspekte werden im nächsten Kapitel (vgl. Kapitel 3.4) mit den Anforderungen in Verbindung gebracht, die durch verschiedene Applikations-

[61] Die Zuordnung zu den Interoperabilitätsstufen des EIF ist nicht vollständig und konzentriert sich auf das Datenmodell. Z.B. können ebenso die Ontologie und das *Rule-Set* (für eine Erklärung der Begriffe vgl. Kapitel 3.1.2) der semantischen Ebene zugeordnet werden (vgl. S. 55 in [52]).

[62] Die Datenmodelle für VCDs und VCD-Pakete sind unter *http://www.peppol.eu/peppol_components/peppol-eia/eia#ict-architecture/pre-award-eprocurement/models* (zuletzt geprüft am *25.10.2012*) verfügbar.

profile nochmals formalisiert werden sollen[63]. Zu diesen Aspekten gehören die internen und externen Referenzen, die in den beschreibenden XML-Dateien zu finden sind, oder von diesen ausgehen[64]. Für einen Vergleich soll zuerst die Art der möglichen Referenzen beschrieben werden. Dabei soll die IMS-CC-Spezifikation als erstes besprochen werden, da sie bereits Anforderungen erfasst, die für gültige Referenzen gelten müssen (vgl. [64]). Somit kann sie als Anhaltspunkt dienen, wenn ähnliche, auf das VCD bezogene, Anforderungen erfasst werden sollen.
Die grundlegende Struktur eines Manifests wurde bereits in Kapitel 3.2.3 erklärt und mit verschiedenen Abbildungen veranschaulicht (vgl. Abbildung 3.11 auf Seite 78, Abbildung 3.12 auf Seite 79, sowie Abbildung 3.13 auf Seite 80). Auch wurde auf die möglichen Referenzen eingegangen. Ein *Manifest* beschreibt eine *Organisation*, die wiederum *Ressourcen* referenzieren kann. Zudem können Abhängigkeiten unter den Ressourcen bestehen (vgl. Kapitel 3.2.3 sowie [64]). Diese Aspekte werden durch Abbildung 3.14 auf Seite 83 nochmals zusammengefasst, und in einer formelleren Form dargestellt[65].
Dabei hat die grundlegende Klasse den Namen *Manifest*. Ein entsprechendes Objekt hat stets ein Kind vom Typ *Resources* (vgl. Klasse *Manifest* - S. 19-20 in [73]). Ein Objekt vom Typ *Resources* kann wiederum mehrere Kinder vom Typ *Resource* haben, die mit Hilfe von Kindern (Objekte vom Typ *File*) auf Inhalte des Pakets verweisen können. Ein wesentlicher Punkt ist, dass ein Objekt vom Typ *Resource* Abhängigkeiten zu anderen Objekten desselben Typs definieren kann. Zudem hat jedes *Resource*-Objekt einen eindeutigen *Identifier*, der für die Referenzen benutzt werden kann. Diese Referenzen unterliegen jedoch Einschränkungen, die zum Beispiel die Art des Ziel-Objekts vorgeben. Dieser wird durch den Wert des *type*-Attributs festgelegt. Auch der Aufbau eines Resource-Objekts (bzgl. der Kinder) kann von dem Inhalt des Attributs abhängen (vgl. Kapitel 3.2.3 und [64]).
Neben einem Objekt vom Typ *Resources* kann ein Manifest ebenso ein *Organization*-Objekt beinhalten (vgl. Kapitel 3.2.3 und [64]). Dieses wird in der Abbildung nicht separat aufgeführt. Vielmehr wird das Objekt, mitsamt seinen direkten und indirekten Kindern, durch einen Knoten repräsentiert. An der Struktur beteiligte Objekte (vom Typ *Item*) können wiederum Referenzen auf Objekte vom Typ *Resource* enthalten. Diese Referenzen müssen wiederum Auflagen entsprechen.
Somit kann eine interne Referenz von einem Objekt vom Typ *Resource* oder *Item* ausgehen. In beiden Fällen ist ein Objekt vom Typ *Resource* das Ziel. Diese Referenzen gehen mit Bedingungen einher, die den Inhalt eines Attributs, oder den Aufbau eines Objekts mit einbeziehen. Neben den internen Referenzen sind jedoch ebenso die Referenzen von

[63] Es soll jeweils ein Applikationsprofil für T-, TC-, TCE-Skeletons und vollständige VCD-Pakete erstellt werden (für eine Beschreibung der Skeletons siehe Beschreibung der Interaktion zwischen dem EVS und einem NVS - Kapitel 3.1.1).

[64] Zum Beispiel kann XML-Schema nicht für die Erfassung von Dateireferenzen verwendet werden, wenn Auflagen für die referenzierten Dateien gelten, und die Validierung dieser Auflagen gefordert ist (vgl. *http://www.w3.org/TR/xmlschema11-1/* - zuletzt geprüft am *25.10.2012*)

[65] Die Abbildung soll mögliche Objektstrukturen zeigen, die ein Manifest beschreiben. Die einzelnen Objekte sollen den jeweiligen, durch das IMS-CC-Profil modifizierten, Klassen entsprechen (vgl. Modifikation des IMS-CC-Profils[64], sowie die Definitionen der Klassen[73]).

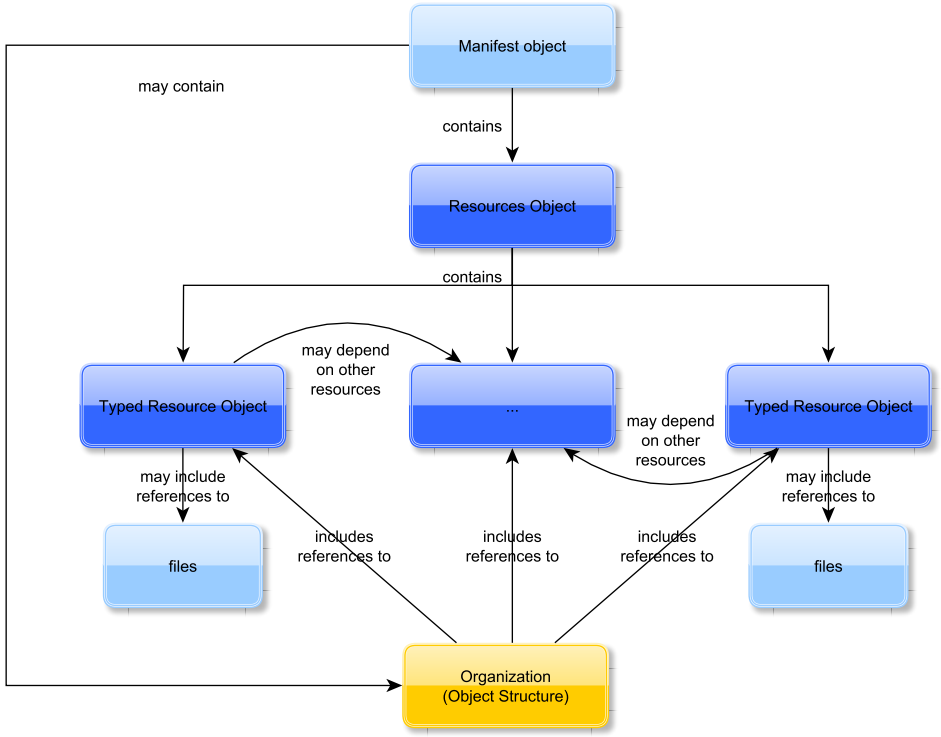


Abbildung 3.14: Referenzen in einem Manifest (Datenmodell IMS-CC)

Bedeutung, die auf externe Dateien verweisen. Diese können ebenso mit Auflagen einhergehen. So kann es sein, dass referenzierte XML-Dateien einem Profil entsprechen müssen, das Teil des IMS-CC-Profils ist (vgl. S.68 in [64]).
Dieser Aufbau soll mit dem Aufbau der Beschreibung eines VCD-Pakets verglichen werden. Abbildung 3.15 auf Seite 84 zeigt einen Auszug aus einer möglichen Beschreibung eines VCD-Pakets (links) und eines Manifests (rechts).
Dabei enthält ein VCD eine sogenannte *Tenderer Structure* (vgl. S. 44 in [52] sowie Kapitel 3.1.3), die durch eine Struktur von BIEs modelliert ist[66]. Wie bei der Organisations-Struktur in einem Manifest, kann die Tenderer-Structure Referenzen beinhalten, die mit der Hilfe von BIEs (vom Typ *VCDReferenceID*) ausgedrückt werden. Im Gegensatz zur Common-Cartridge werden die zu einem VCD zugehörigen Dateien nicht durch die Beschreibung des VCD-Pakets, sondern durch eine eigene Beschreibung erfasst (Knoten *VCD meta description* - vgl. Beschreibung des Bindings in [13]). Ebenso wird jede Datei einzeln referenziert (vgl. BBIEs *SingleTenderer.VCDReferenceID* sowie *BiddingConsortium.VCDReferenceID*). Bei einer Common-Cartridge können dieselben

[66]Die folgenden, VCD- und VCD-Pakete betreffenden, sowie nicht näher belegten, Aussagen basieren auf dem Datenmodell, welches unter *http://www.peppol.eu/peppol_components/peppol-eia/eia#ict-architecture/pre-award-eprocurement/models* heruntergeladen werden kann (zuletzt geprüft am *25.10.2012*).

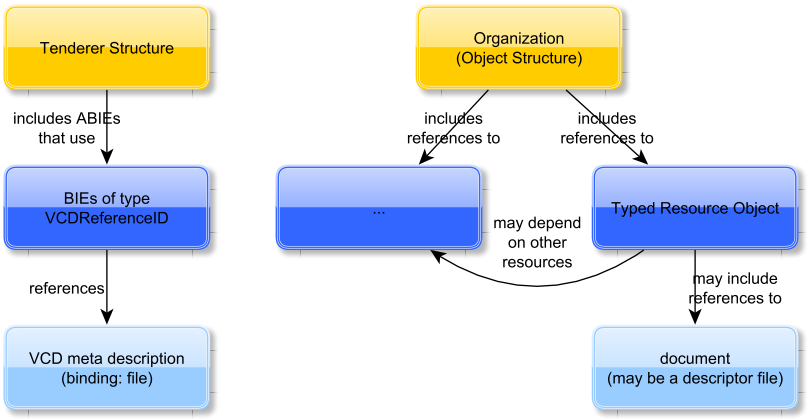


Abbildung 3.15: Vergleich der Referenzen in einem VCD-Paket und einer Common-Cartridge

Resource-Objekte mehr als einmal referenziert werden (vgl. S. 32 in [64]). Allerdings können bestimmte Ressourcen auf XML-Dateien verweisen (durch Objekte vom Typ *File* - vgl. Kapitel 3.2.3), die wiederum Verweise auf Dateien enthalten können, die sich im selben Ordner befinden (vgl. Inhaltstyp 'Discussion Topic' - S. 40 in [64]). Eventuell müssen referenzierte XML-Dateien einem Profil entsprechen (vgl. S. 68 in [64]). Auch die referenzierten VCDs müssen dem Binding des Transaktionsmodells entsprechen, dass den Aufbau einer VCD-Beschreibung erfasst[67]. Neben den gezeigten Referenzen kann ein VCD-Paket ebenso weitere Dateien referenzieren, die keinem Datenmodell entsprechen müssen (vgl. ABIE *AdditionalDocument*). Die Referenzen in einem VCD werden durch Abbildung 3.16 auf Seite 85 veranschaulicht.
Die Beschreibung eines VCDs enthält Beschreibungen der Evidenzen und Kriterien, die mit der Hilfe von verschiedenen Strukturen (die mittels BIEs modelliert sind) verwirklicht werden (vgl. ABIE *Evidence* sowie ASBIE *EconomicOperator.Criterion*). Zwischen diesen erfolgen wiederum Abbildungen, die mit Referenzen umgesetzt sind. Diese machen sich wiederum eindeutige Identifier zu nutze (vgl. BBIE *Criterion.ID* sowie *Evidence.ProvesCriterionID*, *Evidence.DocumentGroupID* sowie *Criterion.ProvingEvidence ID*). Diese Verfahrensweise erinnert an die Realisierung der Abhängigkeiten im IMS-CC-Format. Während eine Common Cartridge - CC - verschiedene Ressourcen beschreibt, und diese Beschreibungen die zugehörigen Dateien zusammenfassen, werden in einem VCD die Evidenzen beschrieben, und die zugehörigen Dateien (vgl. ASBIE *Evidence.DocumentReference* - Kardinalität *1..n*) zusammengefasst. Da eine Evidenz mehrere Kriterien nachweisen kann, können mehrere Verweise auf eine Evidenz erfolgen. Dies erinnert an Ressourcen vom Typ 'Web Content', die durch mehrere Ressourcen referenziert werden können. Bei einer VCD-Beschreibung können die zu einem Kriterium

[67] Dies ist eine der Anforderungen, die mit Hilfe eines Applikationsprofils ausgedrückt werden sollen (vgl. Anforderungen 'VCD-01-R001' und 'VCD-01-R002' in Kapitel 3.4.3).

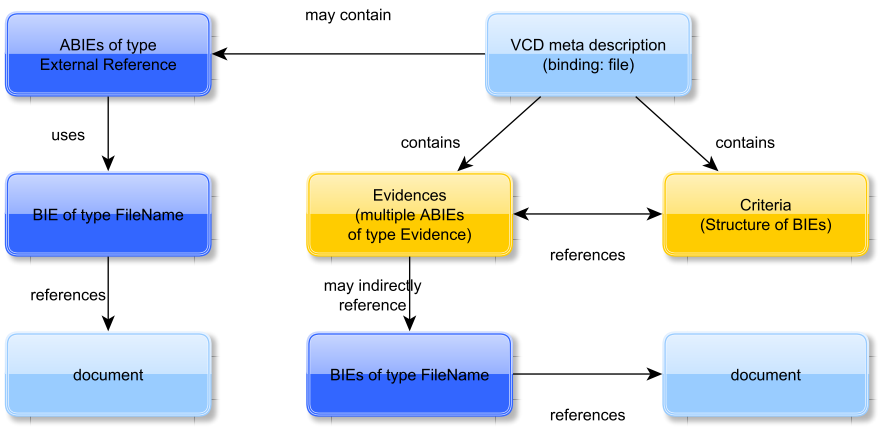


Abbildung 3.16: Referenzen in einem VCD

gehörenden Dateien bestimmt werden, da die Beschreibung eines solchen die benötigten Evidenzen referenziert (vgl. BIE *Criterion.ProvingEvidenceID* - Kardinalität *1–n*). Der Zusammenhang zwischen einem VCD und dem zugehörigen VCD-Paket wird durch Abbildung 3.17 auf Seite 86 veranschaulicht.
Die Referenzierung der, im Paket enthaltenen, Dateien teilt sich auf die Beschreibung des VCD-Pakets, und die Beschreibungen der einzelnen VCDs, auf. Letztere haben, wie Lernobjekte in einer CC auch, ein eigenes Verzeichnis (vgl. Kapitel 6 in [13] sowie Kapitel 3.1.3). Dieses enthält die Dateien, die durch die Beschreibung des jeweiligen VCDs referenziert werden. Dabei wird ein VCD durch einen Identifier, die übrigen Dateien durch relative Dateinamen identifiziert. Der Pfad zu einer VCD-Beschreibung kann mit Hilfe des Identifiers berechnet werden (vgl. Kapitel 6 in [13][68]). In einer CC werden jedoch ausschließlich *Uniform Resource Identifier* - URIs - verwendet, wobei im Paket enthaltene Dateien mit relativen Pfaden eingebunden werden.
Die VCD-Spezifikation sieht jedoch ebenso Anforderungen vor, die weder interne noch externe Referenzen betreffen, und die nicht durch ein XML-Schema erfasst werden können (auf der Ebene des Bindings). Auch für diese Anforderungen bietet sich die Nutzung von *Co-Occurence Constraints* (vgl. Kapitel 2.3.1) an. Zum Beispiel kann in der Beschreibung eines Kriteriums (ABIE vom Typ *Criterion*) angegeben werden, dass alle Evidenzen bereitgestellt wurden (indem der Wert der BBIE *Criterion.AllEvidencesSuppliedIndicator* auf TRUE gesetzt wird). In diesem Fall muss jede Referenz auf eine Evidenz (BBIE *Criterion.Proving EvidenceID*) valide sein (d.h. die zur Referenz benutzte ID muss dem Wert einer BBIE vom Typ *Evidence.DocumentGroupID* entsprechen). Zudem verwendet die VCD-Spezifikation sogenannte Code-Listen. Diese geben eine Liste von bekannten *Codes*

[68] Ein Teil der, im vorzustellenden Testverfahren verwendeten, Komponenten musste angepasst werden, um die Angabe einer Formel, und die Auflösung der IDs zu ermöglichen. Dabei wird die Berechnungsgrundlage mit Hilfe einer formalen Datei festgehalten. Die Berechnungen werden während einem konkreten Test ausgeführt (vgl. Kapitel 4.4).

Abbildung 3.17: Zusammenhang der Datenmodelle (VCD-Package bzw. VCD)

vor, und können für die Einschränkung von Werten genutzt werden (vgl. S. 37 in [13]). Solche vorgegebenen Werte können ebenso mit Enumerationen (XML-Schema[69]) festgehalten werden. Ein Applikationsprofil unterstützt jedoch die Anbindung zur Laufzeit. D.h. die zu verwenden Werte befinden sich in einem sogenannten *Vokabular*, welches für die Überprüfung von Inhalten genutzt wird (vgl. Modifikation ACM.1 in Kapitel 2.4.2). Letztlich kommt bei der VCD-Spezifikation die Unterscheidung zwischen T-, TC-, und TCE-Skeletons, sowie VCD-Paketen hinzu (vgl. Beschreibung der Interaktion zwischen dem EVS und einem NVS in Kapitel 3.1.1). Dabei kann das jeweilige Datenmodell durch Modifikationen am Basismodell beschrieben werden. Dies geschieht, indem das Vorkommen mancher BIEs auf 0 oder 1 gesetzt wird[70]. Leider sind diese Änderungen nicht

[69] Vgl. Definition der *simpleTypes*, sowie Definition der Facet *enumeration* unter *http://www.w3.org/TR/xmlschema11-2/#rf-enumeration* (zuletzt geprüft am *25.10.2012*). Diese Möglichkeit betrifft die Ebene des Bindings.

[70] Dies wird in Validierungsregeln definiert, die unter *http://www.peppol.eu/peppol_components/peppol-eia/eia#conformance-test/pre-award-eprocurement/models* heruntergeladen werden können (zuletzt ge-

immer restriktiv[71]. Der Idealfall wäre die Erfassung der Struktur gewesen, die durch restriktive Profile des VCD-Package-Formats erfolgen hätte können.

3.4 Conformance Testing

Die letzten Kapitel boten einen Überblick über die Spezifikationen VCD und IMS-CC (vgl. Kapitel 3.1 sowie Kapitel 3.2). Dabei wurde näher auf das Datenmodell einer jeden Spezifikation eingegangen (Kapitel 3.1.3 sowie Kapitel 3.2.3), wobei der Fokus auf den Referenzen, und dem Aufbau eines Dateipakets lag. Schließlich wurde ein Vergleich der Datenmodelle angestrebt. Dabei stellte sich heraus, dass beide Formate ähnliche Konzepte nutzen. Zum Beispiel werden Referenzen verwendet, die mit verschiedenen Auflagen verbunden sind, und sowohl auf Datenelemente, als auch externe Dateien, verweisen können (vgl. Kapitel 3.3).
Dieses Kapitel soll eine Grundlage schaffen, die das Testen solcher Anforderungen ermöglicht. Dazu sollen bereits formulierte Anforderungen näher untersucht werden, und schließlich mit den Modifikationen ausgedrückt werden, die in einem Applikationsprofil verwendet werden können (für eine Übersicht vgl. Kapitel 2.4.2)[72]. Die Erfassung der Applikationsprofile und die Herleitung der Testsysteme sollen dann im nächsten Kapitel (vgl. Kapitel 4) besprochen werden. Der verwendete Ansatz erlaubt die Nutzung von Applikationsprofilen, um konkrete Testsysteme herzuleiten. Ein solches kann dazu verwendet werden, konkrete Dateipakete auf gestellte Anforderungen hin zu überprüfen. Allerdings wird sich noch herausstellen, dass Anpassungen an den unterstützenden Tools notwendig sind.

3.4.1 Formalisierung der Anforderungen

Im den nächsten beiden Unterkapiteln sollen die Anforderungen untersucht werden, die durch das IMS-CC Profil und die VCD-Spezifikation gestellt werden[73]. Mögliche Anforderungen sollen in verschiedene Kategorien eingeteilt werden. Dabei sollen nur jene Anforderungen erwähnt werden, auf die bereits in den vorigen Kapiteln eingegangen wurde (vgl. Kapitel 3.1.3 sowie Kapitel 3.2.3). Mit jeder Kategorie sollen ebenso Modifikationen genannt werden, die für die Erfassung der behandelten Anforderungen genutzt werden können (für eine Übersicht - vgl. Kapitel 2.4.2). Somit kann in den folgenden

prüft am *25.10.2012*).

[71] Zum Beispiel wird die BIE *EconomicOperator.Criterion* in einem T-Skeleton verboten. Das Datenmodell sieht jedoch eine Kardinalität von 1..n vor. Eine restriktive Änderung könnte höchstens einen Wert zwischen 1 und unendlich festsetzen.

[72] Ein Teil der Anforderungen wurde bereits durch Schematron-Regeln ausformuliert. Diese sollen übernommen werden.

[73] Die konkreten Anforderungen sollen später gezeigt werden. Diese werden in der IMS-CC-Spezifikation (vgl. [64]), und in einem, im PEPPOL Projekt erarbeiteten, Dokument festgehalten. Letzteres kann unter *https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/2-Conformance_Test/2-CT-PreAward_eProcurement/23-CT-Models/CT%20Models_VCD%20Validation%20Rules%20131.xlsx* heruntergeladen werden (zuletzt geprüft am *12.10.2012*).

Kapiteln eine Klassifizierung aller Anforderungen erfolgen. D.h. die Modifikationen des IMS-CC-Profils, und die, für das VCD zu erstellenden Modifikationen, können anhand der Kategorien eingeordnet, und somit miteinander verglichen werden. Dabei sollen die folgenden Kategorien verwendet werden[74]:

- *interne Referenzen* dienen der Referenzierung von anderen Datenelementen. In beiden Spezifikationen werden Identifier verwendet, um das Ziel einer Referenz festzulegen (Beispiele: BIE *Criterion.ID* / Attribut *Resource.Identifier*). Solche Referenzen können mit verschiedenen Bedingungen einhergehen:
  - eine grundlegende Anforderung ist die *Existenz des Ziel-Elements*. Zum Beispiel sollte ein referenziertes Kriterium auch tatsächlich existieren. In einer CC sollten referenzierte Resource-Objekte existieren, wenn eine Abhängigkeit erklärt wird.
  - darauf aufbauende Anforderungen können *zusätzliche Bedingungen* einführen. Wenn z.B. verschiedene Referenzen existieren, und diese zwischen verschiedenen Arten von Datenelementen bestehen können, sollte sichergestellt sein, dass das Ziel den richtigen Typ hat. Zum Beispiel darf ein *Resource*-Objekt (eines *Manifests*) nur Abhängigkeiten zu anderen *Resource*-Objekten erklären, wenn diese einen erlaubten Typ haben (vgl. Kapitel 3.3). Ein anderes Beispiel ist die Referenzierung von *Resource*-Objekten, die mit Hilfe von *Item*-Objekten erfolgt. Eine solche Referenz ist für Ressourcen vom Typ 'Question Bank' nicht erlaubt (vgl. Kapitel 3.2.3). (vgl. Modifikation ACM3.)
- *externe Referenzen* können auf physikalische Dateien verweisen. Für diese gelten ähnliche Bedingungen wie für interne Referenzen. Denn es können Bedingungen formuliert werden, die das Ziel (die Datei) und die Inhalte mit einbeziehen. Grundlegend kann z.B. die Existenz einer Datei gefordert sein (vgl. IMS-CC-Profil - Regel für die Existenz der Datei IMSMANIFEST.XML[64]). Darauf aufbauend kann z.B. festgelegt werden, dass referenzierte Dateien wiederum auf Eigenschaften hin überprüft werden sollen. Zum Beispiel daraufhin, ob sie einer weiteren Spezifikation entsprechen (vgl. Anbindung des QTI-Profils[64]). (vgl. Modifikation ACM.4)
- *Strukturelle Anforderungen* können ebenso auf inhaltlichen Bedingungen beruhen (vgl. Begriff *Co-Constraints* in Kapitel 2.3.1). Zum Beispiel betrifft dies die Kinder eines *Resource*-Objekts. Ein solches muss auf ein Kind vom Typ *File* verweisen (*1:1* Beziehung - sonstige Kardinalität *0..**), falls die beschriebene Ressource vom Typ 'Discussion Topic' ist[64] (vgl. Attribut *Resource.type* - S. 30 in [64]). (vgl. Modifikationen SM.1, SM.2 und BM.1)
- *Code-Listen* definieren eine Menge von Codes, die für die Einschränkung von Werten dienen können (vgl. S. 37 in [13]). Der Vorteil ist die Anbindung der Code-Listen. Diese können z.B. separat gepflegt werden, und durch die Validierungsre-

[74] Die in der nachfolgenden Aufzählung formulierten Erkenntnisse stützen sich auf die bereits festgehaltenen Anforderungen.

geln mit den Werten verbunden werden. Für das VCD wurde genau dieser Weg eingeschlagen (vgl. Validierungsregeln bzgl. des VCDs / der Skeletons - bzw. Nutzung des Typs *Code.Type* im Datenmodell)[75]. (vergleichbar mit Modifikationen ACM.1 und ACM.2)

3.4.2 Modifikationen des IMS Common Cartridge Profils

Nachfolgend sollen beispielhafte Modifikationen des IMS-CC-Profils aufgeführt werden, die sich den verschiedenen Kategorien zuordnen lassen.

`Interne Referenzen`

Modifikationstyp	Anforderung (vgl. [64])
ACM.3	A Resource object which is a Discussion Topic associated resource may contain Dependency objects which reference Resource objects With Type 'webcontent' or 'associatedcontent/imscc_xmlv1p0/learning-application-resource'.(#S12)
	A Question Bank Resource must not be referenced from an item. (#S11b3)

Tabelle 2: *Interne Referenzen* - Ausgewählte Anforderungen des IMS-CC-Profils

`Externe Referenzen`

Modifikationstyp	Anforderung (vgl. [64])
ACM.4	At the top level of the package there must be a file named Imsmanifest.xml.
	Hat ein Resource-Objekt den Typ *'imsqti_xmlv1p2/imscc_xmlv1p0/assessment'*, so muss dieses ein einzelnes Kind-Objekt vorweisen, welches vom Typ *File* ist, und wiederum eine XML-Datei referenziert, die dem IMS-CC-Profil von QTI 1.2.1 entsprechen muss[76].

Tabelle 3: *Externe Referenzen* - Ausgewählte Anforderungen des IMS-CC-Profils

`Strukturelle Anforderungen`

[75] Dieser Typ definiert eine Reihe von Attributen, mit deren Hilfe die zu verwendende Code-Liste bestimmt werden kann. Der eigentliche Wert ist vom Typ *String*. Vgl. UBL-Schemata, verfügbar unter *http://docs.oasis-open.org/ubl/os-UBL-2.0/* (zuletzt geprüft am 13.10.2012).

[76] Diese Anforderung ist nur im Text der IMS-CC-Spezifikation ausformuliert (vgl. S. 35 in [64]), und nicht Teil des Applikationsprofils. Sie wird jedoch durch eine spätere Version des Profils erfasst.

Modifikationstyp	Anforderung (vgl. [64])
SM.1	Ein Manifest-Objekt hat immer ein ManifestMetadata-Objekt als Kind.
SM.1 - bedingt	For Discussion Topic Resources the Resource object must contain a single File Object [...] (#S06).
SM.2 - bedingt	For Web Link Resources the Resource object [...]. It must contain neither Dependency objects [...] (#S07).

Tabelle 4: *Strukturelle Anforderungen* - Ausgewählte Anforderungen des IMS-CC-Profils

`Sonstige Anforderungen`

Modifikationstyp	Anforderung (vgl. [64])
SM.4	At the manifest level, the Common Cartridge profile of IEEE LOM loose must be used, which corresponds to unqualified Dublin Core[77].

Tabelle 5: *Sonstige Anforderungen* - Ausgewählte Anforderungen des IMS-CC-Profils

3.4.3 Modifikationen für das VCD

Nachfolgend sollen die Modifikationen aufgeführt werden, die für die Erfassung der, durch die VCD-Spezifikation vorgegebenen, Anforderungen genutzt werden können. Die Datenformate geben die Form von verschiedenen Skeleton-Arten, VCDs und VCD-Paketen vor. Da diese Struktur nicht mit restriktiven Profilen erfasst werden kann, sollen 5 verschiedene Profile erstellt werden. Für einen Teil der Anforderungen wurden bereits Schematron-Regeln erstellt, die in die zu erstellenden Applikationsprofile eingebunden werden sollen.

`Interne Referenzen`

Modifikationstyp	Format	Zu erfassende Anforderung[78]
ACM.3	VCD TCE-Skeleton	The Evidence MUST have a ProvesCriterionID referring to an existing Criterion. (VCD-02-R005)

Fortsetzung auf der nächsten Seite

[77] Die Spezifikation IMS-CP sieht vor, dass beschreibende Informationen zu einem Manifest angegeben werden können. Dies wird mittels eines *Extension-Points* realisiert (vgl. Klasse *ManifestMetadata* in [73]). Die Modifikation schränkt die zu verwendenden Metadaten ein (vgl. Beschreibung des *Organization*-Objekts in Kapitel 3.2.3).

Tabelle 6 – *Fortsetzung*

Modifikationstyp	Format	Zu erfassende Anforderung[78]
ACM.3 - bedingt	VCD	If the AllEvidencesSuppliedIndicator of a Criterion is true, then each ProvingEvidenceID of this Criterion MUST Refer to exactly one existing Evidence. (VCD-02-R004)
-	VCD	The cross-references between Criterion and Evidence MUST be coherent. (VCD-02-R028)

Tabelle 6: *Interne Referenzen* - Zu erfassende Anforderungen des VCDs

`Externe Referenzen`

Modifikationstyp	Format	Zu erfassende Anforderung[78]
ACM.4	VCD Package	A VCD that is referenced in a VCD Package XML document by its UUID must exist As a subfolder of the VCD Package folder/zip file having the name "VCD_<uuid>". (VCD-02-R010)
	VCD Package	The XML meta-data file of a single VCD must exist in the corresponding VCD folder Having the name "VCD_<uuid>.xml". (VCD-02-R011)
	VCD-Package	Any XML document SHOULD conform to the corresponding XML schema definitions. (VCD-01-R001 und VCD-01-R002)
	VCD	A file referenced inside a document instance as an external reference must either exist inside the VCD-Container in the same folder like the meta data file or it must exist at the remote location (in case a remote file is referenced). (VCD-02-R017)

Tabelle 7: *Externe Referenzen* - Zu erfassende Anforderungen des VCDs

`Strukturelle Anforderungen`

Modifikationstyp	Format	Zu erfassende Anforderung[78]
SM.1	VCD	The Criterion MUST have a ProvingEvidenceID Referring to an existing Evidence. (VCD-02-R006)

Fortsetzung auf der nächsten Seite

[78]Die einzelnen Anforderungen sind im PEPPOL-Projekt erarbeitet worden, und auf der Seite *http://www.peppol.eu/* verfügbar (vgl. Kapitel 3.4.1).

Tabelle 8 – *Fortsetzung*

Modifikationstyp	Format	Zu erfassende Anforderung[78]
SM.2	VCD VCD-Package	DocumentReference/IssuerParty SHOULD be disabled if Belongs to IssuingService/PolicyDocument. (VCD-02-R002)
SM.1 - bedingt	VCD T-Skeleton TC-Skeleton TCE-Skeleton	The elements PersonDetails.FirstName, PersonDetails.FamilyName, PersonDetails. Birthdate and PersonDetails.PlaceOfBirth MUST exist if PersonDetails.ID is not existing. (VCD-02-R007)
-	VCD-Package	EITHER the SingleTenderer OR the BiddingConsortium MUST exist. (VCD-02-R003)
-	VCD	The attribute LanguageID MUST be present to define the language. (VCD-02-R024)
-	VCD	The party name MUST exist in the Economic Operator. (VCD-02-R043)

Tabelle 8: *Strukturelle Anforderungen* - Zu erfassende Anforderungen des VCDs

`Code-Listen`

Modifikationstyp	Format	Zu erfassende Anforderung[78]
ACM.2	Verschieden	Ein Wert muss einer Code-List entstammen. (VCD-02-R012 – VCD-02-R016, VCD-02-R018 – VCD-02-R023, VCD-02-R025, VCD-02-R026)

`Sonstige Anforderungen`

Modifikationstyp	Format	Zu erfassende Anforderung[78]
TM.6	Alle Formate	The electronic mail address SHOULD conform to the pattern "[^@]+@[^\.]+\..+". (VCD-02-R008)
-	VCD VCD-Package	A VCD/VCD Package SHOULD not contain empty elements. (VCD-02-R029)

Tabelle 10: *Sonstige Anforderungen* - Zu erfassende Anforderungen des VCDs

4 Ein generischer Ansatz zur Validierung

In Kapitel 3 wurden die Spezifikationen IMS-CC und VCD näher betrachtet. Beide Spezifikationen geben Datenformate vor, die wiederum das Format gültiger Datenpakete definieren. Ein wesentlicher Teil ist die Beschreibung eines Datenpakets, welche bei beiden Formaten zum Paket selbst gehört, und durch ein Binding mit einer XML-Syntax verbunden ist. Bei einer Common-Cartridge wird der Paketinhalt durch eine sogenannte Manifest-Datei beschrieben, während das VCD zwei unterschiedliche Formate vorgibt, die für die Beschreibung von VCDs und VCD-Paketen genutzt werden können. (vgl. Kapitel 3.1.3 und Kapitel 3.2.3)
Dabei gelten unterschiedliche Anforderungen, die sich auf die Beschreibung des Paketinhalts, die von einer Beschreibung ausgehenden Referenzen, und den Aufbau eines Datenpakets beziehen. Bei einer näheren Betrachtung stellte sich heraus, dass beide Formate eine Reihe ähnlicher Anforderungen definieren. Dies beruht auf ähnlichen Lösungen, die in beiden Datenformaten zum Einsatz kommen (vgl. Kapitel 3.3). Somit wurden verschiedene Kategorien eingeführt, die die Anforderungen beider Spezifikationen einordnen (vgl. Kapitel 3.4.1). Dabei ist die IMS-CC-Spezifikation ein Profil, welches u.a. auf der IMS-CP-Spezifikation basiert, und verschiedene Modifikationen enthält (vgl. S. 6, 29ff in [64], sowie Kapitel 3.2.1). Diese schränken sowohl das Format der, durch die IMS-CP-Spezifikation vorgesehenen, Datenpakete (Content Packages) und Paketbeschreibungen (Manifest) ein (vgl. Modifikationen des IMS-CC-Profils[64]), wodurch neue Anforderungen festgelegt werden. Eine physikalische Version des Profils kann für die Herleitung eines Testsystems[79] genutzt werden, mit dem die Einhaltung der Anforderungen überprüft werden kann (vgl. [24]). Die Nutzung restriktiver Modifikationen (für eine Erklärung siehe Kapitel 2.4.1) bietet sich somit auch bei den Datenmodellen des VCDs (Skeletons, VCD, VCD-Package) an. Denn konkrete Dokumente und Datenpakete müssen auf Anforderungen hin überprüft werden, die in verschiedenen Schemasprachen vorliegen, oder nicht durch eine solche erfasst werden können (vgl. Anforderungen der Formate[64])[80].
Dieses Kapitel stellt den Ansatz vor, der die Erfassung der Modifikationen, und die Generierung eines Testsystems erlaubt. Dazu werden die Modifikationen mit der Hilfe eines Applikationsprofils erfasst, welches auf dem XML-Binding der jeweiligen Spezifikation (IMS-CC / VCD) basiert. Ein generisches Testsystem stellt die passenden Komponenten bereit, die für die Überprüfung verschiedener Anforderungen genutzt werden können. Eine Anforderung muss jedoch in einer Sprache (z.B. einer Schemasprache) formuliert sein, die von einer der Komponenten unterstützt wird. Die Herleitung der benötigten Dokumente (z.B. verschiedener Schemata) wird von einem unterstützenden Tool vorgenommen. Die Verfahrensweise soll in den folgenden Kapiteln näher vorgestellt werden:

[79] Die dafür notwendigen Komponenten (wie z.B. das generische Testsystem - welches in Kapitel 4.2 vorgestellt wird) wurden zum größten Teil durch den Autor implementiert. Ein Ausnahme ist das Vocabulary-Testsystem, welches der Überprüfung von Werten dient, die einem vorgeschriebenen Vokabular entsprechen müssen. Die Implementierung erfolgte jedoch nicht im Rahmen dieser Arbeit. Kapitel 4.4 erläutert die, im Rahmen dieser Arbeit, vorgenommenen Anpassungen.

[80] Kapitel 3.4.1 enthält einen Verweis auf die WWW-Seite, auf welcher die, im PEPPOL-Projekt erstellten, Validierungsregeln heruntergeladen werden können.

- Kapitel 4.1 wird näher darauf eingehen, wie die zu verwendenden Modifikationen, und das zugrunde liegende Binding, genutzt werden können, um einzelne Testaufgaben zu bestimmen. Eine Ausführung dieser Aufgaben kann für die teilweise Überprüfung eines VCDs, eines VCD-Pakets, oder einer Common-Cartridge genutzt werden.
- Kapitel 4.2 wird den Aufbau des generischen Testsystems beschreiben. Dieses stellt die Komponenten bereit, die für die Bestimmung der Testaufgaben, und deren Ausführung notwendig sind.
- Kapitel 4.3 wird die verschiedenen Phasen beschreiben, die mit der Überprüfung eines Datenpakets einhergehen. Dabei wird der Ablauf durch ein sogenanntes Kontrollsystem gesteuert, welches die Ausführung der verschiedenen Aufgaben arrangiert.
- Kapitel 4.4 wird letztlich die Änderungen vorstellen, die für die Unterstützung der, für das VCD zu verwendenden, Modifikationen notwendig sind. Zum Beispiel muss die Implementierung des generischen Testsystems angepasst werden.

4.1 Modifikations-basierte Bestimmung von Testaufgaben

In Kapitel 3.4.3 wurden verschiedene Modifikationen genannt, die für die Erfassung der, für die Datenmodelle des VCD bestimmten, Anforderungen genutzt werden können. Die Erfassung solcher Modifikationen kann wiederum durch die Erstellung eines Applikationsprofils erfolgen. Ein solches kann auf dem Informationsmodell der jeweiligen Spezifikation basieren, wobei das Ziel einer Modifikation durch ein Element des Informationsmodells angegeben wird (vgl. *Informationsmodell der IMS-CP-Spezifikation*[73], sowie *Datenmodelle für das VCD* in Kapitel 3.4). Schließlich kann ein Applikationsprofil erstellt werden, welches die Änderungen am XML-Binding ausdrückt.
Bei beiden betrachteten Spezifikationen werden XML-Schemata bereitgestellt (vgl. [18] und Kapitel 3.1.3). Diese geben ein Format für XML-Dateien vor, die zur Beschreibung eines Dateipakets (IMS-CC), oder von Teilen eines solchen (VCD, VCD-Package), genutzt werden können (vgl. Kapitel 3.3). Die bereitgestellten XML-Schemata sind die Grundlage für die Erstellung der Applikationsprofile. Dabei ist ein Applikationsprofil ebenfalls ein Dateipaket (vgl. S. 78 in [24]). Die Art der enthaltenen Dateien und deren Beziehungen werden durch Abbildung 4.1 auf Seite 95 veranschaulicht, und sollen im folgenden näher erklärt werden:

- Die *XML-Schemata* beschreiben das XML-Binding der jeweiligen Spezifikation. Sie geben das Format der XML-Dateien vor, die zusammen ein valides Dateipaket (CC / VCD / VCD-Paket) beschreiben. Die verwendete Schema-Sprache ist XML-Schema 1.0, wobei stets ein XML-Schema existiert, welches im Kontext der Applikationsprofile als Basis-Schema bezeichnet wird (vgl. S. 11 in [65]). Dieses kann sich die restlichen XML-Schemata (direkt oder indirekt) zu nutze machen[81],

[81] Vgl. *http://www.w3.org/TR/xmlschema11-1/#compound-schema* (zuletzt geprüft am *19.10.2012*).

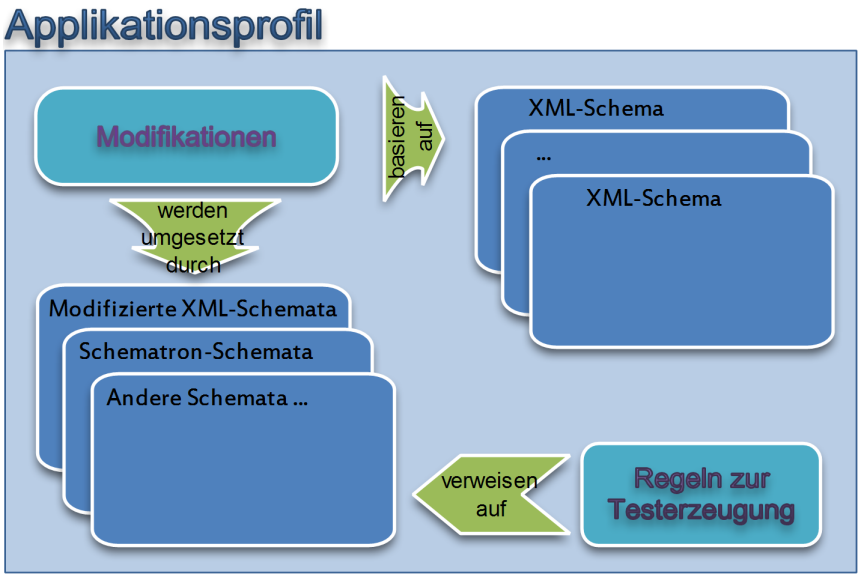


Abbildung 4.1: Aufbau eines Applikationsprofils

und erfasst bereits viele der Anforderungen, die durch die jeweiligen Spezifikationen vorgegebenen werden (vgl. [18] sowie [13]).

- Die *Modifikationen* beschreiben Änderungen an den XML-Schemata (dem XML-Binding). Sie stellen die Umsetzung der Modifikationen dar, die bereits auf dem Informationsmodell definiert wurden (vgl. Kapitel 2.4.1). Diese Modifikationen haben jedoch ein konkretes Element als Ziel, welches durch das Basisschema (oder einer der direkt oder indirekt genutzten XML-Schemata) definiert wird (vgl. Kapitel *Creating a Binding for the Information Model* in [65]). Ein Beispiel wäre die Umsetzung einer Modifikation, die das *type*-Attribut aller Resource-Objekte als Ziel hat. Das type-Attribut wird im Informationsmodell mittels der Klasse *Resource* (vgl. S. 28 in [73]) definiert, und im jeweiligen XML-Schema durch eine Attribut-Deklaration vorgegeben (vgl. [18]). Diese gehört zu einem komplexen Typ, der den Aufbau eines Resource-Elements beschreibt (vgl. Klasse *Resource* in [18] - sowie Typ *Resource.Type*[82]). Dabei schränkt die Modifikationen die gültigen Werte konkreter Attribute ein. Die Definition des XML-Attributs ist somit das Ziel der Umsetzung. Dabei werden die Modifikationen in einer separaten XML-Datei beschrieben, die wiederum einem XML-Schema entspricht (vgl. S. 35ff in [65]). Somit kann ein XML-Schema-Validator genutzt werden, um die Datei auf syntaktische Fehler hin zu prüfen. Diese Überprüfung kann ebenso geschehen, wenn die Datei durch ein unterstützendes Tool verarbeitet werden soll (z.B. durch das Tool SchemaProf).

[82]Das XML-Binding der IMS-CP-Spezifikation wird, wie alle anderen, in dieser Arbeit verwendeten, Spezifikationen von IMS, unter *http://www.imsglobal.org/specifications.html* bereit gestellt (zuletzt geprüft am *19.10.2012*).

- Die *hergeleiteten Dokumente* drücken die Anforderungen des Profils aus, und können in unterschiedlichen Schema-Sprachen formuliert sein (vgl. Kapitel 2.3.1). Ein Teil der Modifikationen kann direkt auf das jeweilige Zielelement (innerhalb eines XML-Schemas) angewendet werden. Die Anwendung solcher Modifikationen führt zu einer Menge von XML-Schemata, die sich an den modifizierten Stellen vom XML-Binding unterscheiden (vgl. Modifikationstyp *XML Schema modifications* - S. 11 in [65]). Ein Profil kann jedoch weitere Modifikationen festhalten, die nicht mit einem neuen XML-Schema ausgedrückt werden können (vgl. Modifikationstyp *XML non-Schema modifications* - S. 11 in [65]). Ein Beispiel sind Auflagen für Dateireferenzen, die mit der Hilfe von sogenannten *Additional Constraints* formuliert werden (*Riley et al.* führen eine eigene Kategorie für diese Modifikationen ein - vgl. S. 11 in [65]). Zum Beispiel können Attribute einen relativen Pfad beinhalten, der für einen Test genutzt werden muss (z.B. um die Existenz der Datei zu überprüfen - vgl. Modifikation „Referenced files must exist.“ - S. 80 in [64]). Dies gilt jedoch ebenso für viele Auflagen, die auf internen Referenzen basieren. Dabei werden Auflagen unterstützt, die mit XML-Schema 1.1 Assertions (vgl. Kapitel 2.3.1) ausgedrückt werden können[83]. Andere Auflagen können z.B. mit einem Schematron-Schema erfasst werden (vgl. Begriff *Co-Constraints* in Kapitel 2.3.1). Denn ein Applikationsprofil kann ebenso Dokumente enthalten, die von Hand erstellt wurden, und ein Teil der Anforderungen festhalten. Diese können auf Modifikationen basieren, die auf der konzeptuellen Ebene erfasst wurden[84].

- Die *Regeln zur Testerzeugung* stellen letztendlich eine Verbindung zwischen den *hergeleiteten Dokumenten* und den Testaufgaben her, die für eine Überprüfung konkreter Datenpakete auszuführen sind (vgl. [24]).

Dabei kann ein Applikationsprofil Teil eines *Domain-Profile* sein, welches verschiedene Applikationsprofile verbindet (vgl. [24]). Die Struktur ergibt sich durch die Verbindungen, die durch einen Teil der restriktiven Modifikationen hergestellt werden können, und in Kapitel 2.4.3 erklärt werden. Letztendlich kann eine Modifikation ein Format für referenzierte Dateien vorschreiben, oder Datenelemente einschränken, die an sogenannten *Extension-Points* eingebunden werden (vgl. Kapitel 2.4.2). Ein Beispiel ist die Referenzierung von QTI-Dateien, die innerhalb einer Manifest-Datei möglich ist (vgl. *Assessment Content Type* - S. 35 in [64]). Diese müssen bei einem Test mit einer Validierung einhergehen. Bei einem VCD-Paket müssen dagegen referenzierte VCDs überprüft werden. D.h. das Profil ist ein *Domain-Profile*, welches sich Modifikationen zu nutze macht, die eine Verbindung zum VCD-Profil herstellen. Da jedoch 3 gültige Untermengen eines VCD-Pakets existieren (T-, TC-, TCE-Skeleton - vgl. Kapitel 3.1.1), mussten

[83] Die Herleitung der benötigten Dokumente erfolgt durch das sogenannte *Schema Transform Tool*, dass durch den Autor implementiert wurde. XML Schema 1.1 wird erst durch die neue Implementierung unterstützt. Die Implementierung erfolgte nicht im Rahmen dieser Arbeit.

[84] Dies ist ebenso eine Neuerung. Das Tool SchemaProf kann zur Erstellung und Bearbeitung von Applikationsprofilen verwendet werden, und unterstützt in der neusten Version den Import von weiteren formalen Dokumenten. Diese Änderung wurde im Rahmen der Diplomarbeit entwickelt, und wird in Kapitel 4.4 näher beschrieben.

4 verschiedene *Domain-Profiles* erstellt werden (vgl. S. 66 in [52]). Die verschiedenen Skeleton-Arten gehen mit Einschränkungen einher, die auf den jeweiligen Datenformaten (für VCD-Pakete und VCDs) formuliert wurden (vgl. Ende von Kapitel 3.3), und durch verschiedene XML-Bindings ausgedrückt wurden (vgl. S. 56 in [52]). D.h. jedes Domain-Profile basiert auf einem dieser XML-Bindings (vgl. S. 66 in [52]).
Dabei werden die Testaufgaben aufgrund der Modifikationen bestimmt, die Teil dieser Profile sind. Der Algorithmus basiert auf sogenannten Regeln, die auf die hergeleiteten Dokumente eines Applikationsprofils verweisen (vgl. *Regeln zur Testerzeugung* - letzte Aufzählung). Da jede Modifikation durch ein hergeleitetes Dokument erfasst wird, besteht ein direkter Zusammenhang zwischen den Modifikationen, und den erzeugten Aufgaben. Dabei besteht jede Regel aus den folgenden, in Abbildung 4.2 auf Seite 97 gezeigten, Angaben[85]:

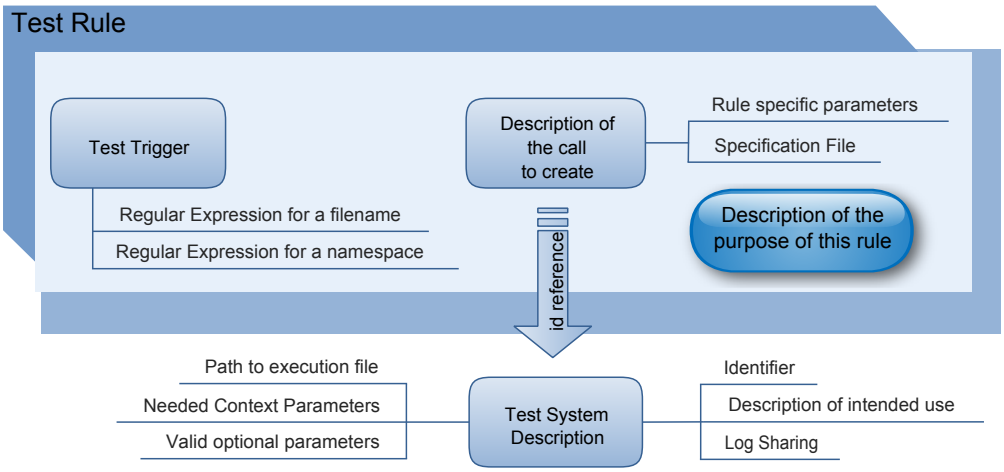


Abbildung 4.2: Aufbau einer Regel, die zur Bestimmung von Testaufgaben verwendet wird

- Ein sogenannter *Test Trigger* definiert einen Filter, welcher bestimmte Dateien eines konkreten Dateipakets erfasst[86]. Dabei können reguläre Ausdrücke verwendet werden, die Einschränkungen auf dem Dateinamen, und dem Namespace von XML-Dateien definieren[87].

- Die *Beschreibung der Testaufgabe* stützt sich auf die hergeleiteten Dateien. Dabei fällt die Aufgabe nur für diejenigen Dateien an, die durch den Filter (vgl. *Test Trigger*) erfasst werden. Eine hergeleitete Datei kann das Format festlegen, dass für die, durch den Filter, erfassten Dateien vorgeschrieben ist. Der Inhalt kann aber ebenso andere Anforderungen beschreiben (z.B. eine maximale Größe, die durch die Ausführung der Aufgabe überprüft wird).

[85] Der Aufbau und die Nutzung der Testregeln wird ebenso in [24] erklärt.
[86] D.h. das Paket gehört zu den Eingaben des Algorithmus
[87] Falls ein Namespace angegeben wurde, werden nur XML-Dateien erfasst.

- Die *Beschreibung des zu verwendenden Testsystems* ist nicht Teil der Regel, sondern wird mit einer ID referenziert. Dies hat den Vorteil, dass unterschiedliche Testsysteme genutzt werden können, wenn die Testaufgabe ausgeführt werden soll.
- Letztlich kann ein natürlichsprachlicher Text angegeben werden, der die *Absicht hinter der Testregel* beschreibt.

Somit werden als erstes die Regeln angewendet, wenn die Testaufgaben für ein Datenpaket bestimmt werden sollen. Diese werden dann letztlich durch einzelne Testsysteme ausgeführt, die mit der Hilfe einer ID identifiziert werden (vgl. S. 80 in [24]). Die einzelnen Testsysteme sind selbst Teil des generischen Testsystems, welches im nächsten Kapitel vorgestellt werden soll. Abbildung 4.3 auf Seite 98 fasst die Vorgehensweise nochmals zusammen. Auch das in der Abbildung gezeigte Kontrollsystem ist Teil des generischen Testsystems, und übernimmt die anfängliche Bestimmung der Testaufgaben (vgl. S. 80 in [24]). Es gilt jedoch zu beachten, dass neue Aufgaben anfallen können, während die anfänglich bestimmten Aufgaben ausgeführt werden[88] (vgl. S. 81 in [24]).

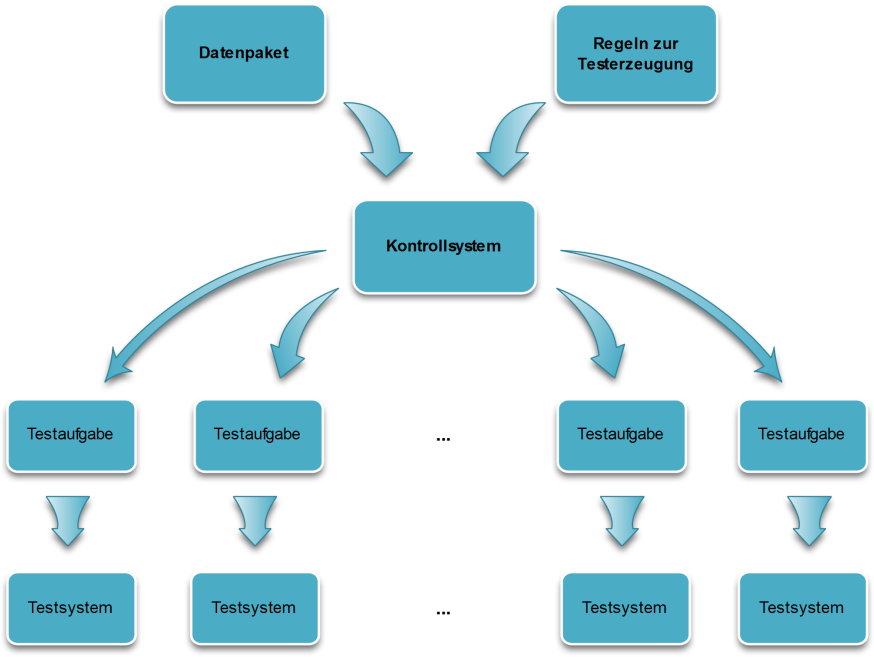


Abbildung 4.3: Initiale Bestimmung der Testaufgaben (für ein Datenpaket)

Letztendlich lassen sich Ähnlichkeiten zum Ansatz des GIT-B-Frameworks ausmachen (vgl. Kapitel 2.3.2). Dieses führt den Begriff des *Document Assertion Set*s - DAS - ein. Ein solches hält die formalen Anforderungen fest, und beinhaltet die Artefakte, die zur Validierung genutzt werden können (vgl. Kapitel 2.3.2 und S. 14 in [42]). Die Menge der

[88] Auf diesen Aspekt wird in Kapitel 4.3 näher eingegangen.

hergeleiteten Dateien umfasst die Schemata, die für die Validierung einzelner Dokumente verwendet werden. Die Schemata halten zudem die Anforderungen fest, die für einzelne Dokumente eines Datenpakets erfüllt sein müssen. Die Anforderungen können wiederum aus den vorgenommen Modifikationen resultieren. Eine Validierung wird in einem Test-Bed (vgl. GIT-B Framework - S. 50 in [42] sowie Kapitel 2.3.2) durch einen sogenannten Document-Validator ausgeführt (vgl. Kapitel 2.3.2 und S. 14 in [42]). Ein solcher wird beim generischen Testsystem durch ein einzelnes Testsystem genutzt, welches wiederum eine ID zugewiesen bekommt, und in einer Testregel referenziert wird. Die Erzeugung der initialen Tests unterscheidet sich vom Ansatz des *GIT-B*-Frameworks. Das Framework führt den Begriff der *Test Suite* ein (vgl. S. 16 in [42] und Kapitel 2.3.2), die durch eine *Test Suite Engine* ausgeführt werden kann (vgl. S. 16 in [42] und Kapitel 2.3.2), und in einer sogenannten *Test Definition Language* verfasst ist (vgl. S. 21 [42] und Kapitel 2.3.2). Eine Test Suite kann u.a. einen Ablauf von Document-Validator-Ausführungen definieren (vgl. S. 15 in [42] und Kapitel 2.3.2). Im Fall des generischen Testsystems dienen die Testregeln als Grundlage, wenn die auszuführenden Tests bestimmt werden sollen.

4.2 Ein modularer Ansatz zur Ausführung anfallender Aufgaben

Im letzten Kapitel wurde der Aufbau eines Applikationsprofils, und die Erzeugung der Testaufgaben eingegangen. Diese wird durch das sogenannte Kontrollsystem übernommen, welches Teil des generischen Testsystems ist. Der zugrunde liegende Algorithmus stützt sich auf sogenannte Testregeln, und nutzt ein Datenpaket als Eingabe. Für jede Testregel wird ein Datei-Filter verwendet, welcher bestimmte Dateien eines Datenpakets erfasst. Zudem wird die Testaufgabe beschrieben, die für die erfassten Dateien ausgeführt werden soll. Diese kann sich auf eine der hergeleiteten Spezifikations-Dateien stützen, die ein Teil der zu überprüfenden Anforderungen enthalten kann. Da diese Anforderungen auf den Modifikationen des Applikationsprofils beruhen können (vgl. Erklärung der *hergeleiteten Dateien* in Kapitel 4.1), kann die Ausführung der Testaufgabe zur Überprüfung verschiedener Modifikationen führen (vgl. Kapitel 4.1).
In diesem Kapitel soll der Aufbau des generischen Testsystems besprochen werden, welches die Bestimmung, und die Ausführung der Testaufgaben ermöglicht, und zum größten Teil durch den Autor implementiert wurde[89]. Ein solches muss mit einem Applikationsprofil, oder einem Domain-Profile konfiguriert werden[90]. Abbildung 4.4 auf Seite 100 zeigt die wichtigsten Komponenten eines konfigurierten Testsystems.

Die Profile enthalten die hergeleiteten Dokumente, welche die zu testenden Anforderungen festhalten. Zudem enthält jedes Profil die Testregeln, die zur Erstellung der Testaufgaben genutzt werden. Dabei liegt jedes Applikationsprofil einzeln vor. Ein

[89] Die Implementierung erfolgte nicht im Rahmen der Diplomarbeit, und schließt bis auf das Vocabulary-Testsystem, alle Komponenten mit ein. Das Vocabulary-Testsystem dient der Überprüfung von Werten, die einem vorgeschriebenen Vokabular entsprechen müssen.

[90] Diese Aufgabe wird von einem weiteren Tool übernommen. Der sogenannte *Packager* kann für die Erstellung eines Testsystems genutzt werden, und greift auf verschiedene generische Testsysteme zurück. Diese liegen auf einem Server, und werden mit der Hilfe von IDs verwaltet.

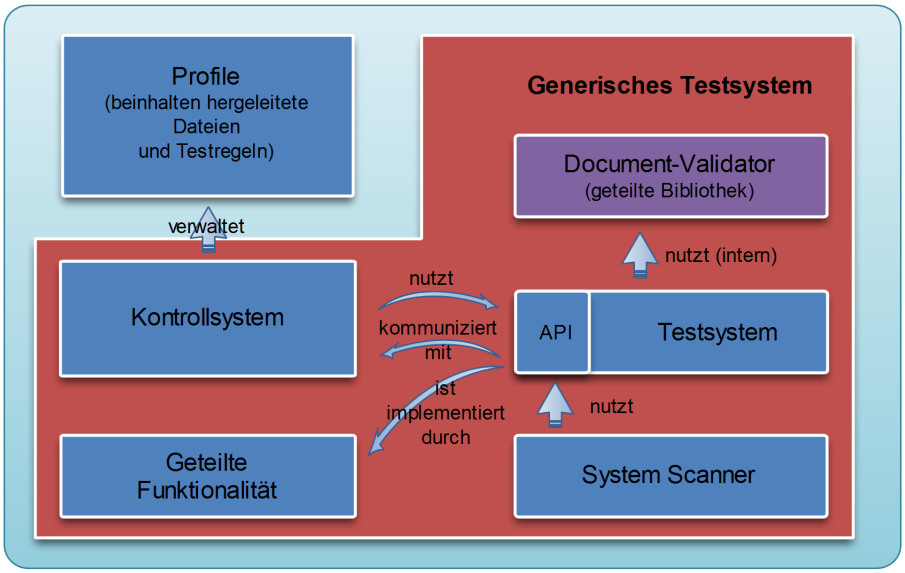


Abbildung 4.4: Grundlegende Komponenten des generischen Testsystems

Domain-Profile wird somit in einzelne Profile zerlegt, wenn ein generisches Testsystem konfiguriert wird. Dies hat verschiedene Vorteile. Zum einen können einzelne Profile überarbeitet und ausgetauscht werden. Zum anderen sind Tests gegen Unterprofile leichter zu realisieren. Im, durch das GIT-B-Framework beschriebenen, Test-Bed werden die Dokumente, die zu einem DAS gehören, in einem Repository gespeichert (vgl. *Test Artifact Persistence Manager (TAPM)* - S. 55, 139ff in [42] sowie Abbildung 2.13 auf Seite 44 in Kapitel 2.3.2). Diese werden durch eine Test-Suite genutzt, die eine Reihe von Validierungen definieren kann. Im generischen Testsystem werden die Profile, und somit die hergeleiteten Dateien, durch das Kontrollsystem verwaltet.

Die Testsysteme sind für die eigentliche Ausführung der Testaufgaben verantwortlich. Dabei implementiert jedes Testsystem einen Teil einer *API* (Java-Interface *ITestSystem*), die durch eine, zur geteilten Funktionalität gehörende, Teilimplementierung (abstrakte Java-Klasse *ATestSystem*) vorgegeben wird. Die Schnittstelle ist von zentraler Bedeutung, und ermöglicht die einheitliche Ausführung von Aufgaben:

- Einzelne Testsysteme sind in der Lage, sich selbst zu beschreiben. Dazu gibt die Schnittstelle (*ITestSystem*) u.a. Methoden vor, die durch ein Testsystem zu implementieren sind, und die für die Erfragung der benötigten Angaben genutzt werden können. Wird das Testsystem mit einem speziellen Parameter aufgerufen, so wird der Aufruf von der Teilimplementierung (*ATestSystem*) abgefangen. Als nächstes wird die Implementierung der Schnittstelle

(*ITestSystem*) genutzt, um eine XML-Beschreibung zusammen zu stellen. Zur Selbstbeschreibung gehören u.a. der Identifier und die Beschreibung der Parameter, die für einen Test benötigt werden.

- Jedes Testsystem kann automatisch initialisiert und ausgeführt werden. Ein Kommandozeilen-Aufruf wird durch die Teilimplementierung (*ATestSystem*) abgefangen, die wiederum auf Methoden zurückgreift, die durch das Testsystem implementiert wurden (*ITestSystem*). Dabei gibt die Teilimplementierung (*ATestSystem*) einen Ablauf vor, der von der Initialisierung der Parameter bis zur Ausführung der Testaufgabe reicht, die über eine spezielle Methode (*ITestSystem.test()*) veranlasst wird. Die Implementierung dieser Methode kann wiederum auf einen *Document-Validator* zurückgreifen[91].
- Jedes Testsystem kann Nachrichten an einen festgelegten Port verschicken. Die Art der Informationen beschränkt sich auf Rückmeldungen (zum Testverlauf) und die gewünschte Veranlassung weiterer Tests. Die Teilimplementierung (*ATestSystem*) gibt Methoden vor, die für die Kommunikation genutzt werden können, und somit jedem Testsystem zur Verfügung stehen.

Dabei können die einzelnen Testsysteme mit sogenannten *Testing Capability Components* verglichen werden, die im GIT-B Framework vorgestellt werden (vgl. S. 49-51 in [42] sowie Abbildung 2.13 auf Seite 44 in Kapitel 2.3.2). Je nach Art der Komponente (bzw. des Plugins) muss eine sogenannte *Core Plugin-in API* implementiert werden, so dass das Plugin von der sogenannten *Core-Platform* kontrolliert werden kann. Dies geschieht mit der Hilfe von Methoden-Aufrufen (vgl. S. 51-54 in [42]). Im generischen Testsystem wird ein einzelnes Testsystem jedoch durch einen Kommandozeilen-Aufruf ausgeführt, welcher die Parameter enthält, die das Verhalten des Systems beeinflussen. Dabei wird der Aufruf durch die Teilimplementierung der *API* (*ATestSystem*) verarbeitet. Zusammen ermöglichen die *API*, und die Teilimplementierung (*ATestSystem*), die Implementierung weiterer Testsysteme, und deren Einbindung in das generische Testsystem. Dies gleicht dem Plugin-Mechanismus eines Test-Beds. Allerdings unterstützt ein solches mehrere Typen von Plugins (vgl. S. 49 ff in [42]).

Der System Scanner nutzt die, durch die Teilimplementierung der *API* (vergleiche Beschreibung der *Testsysteme*), bereitgestellte Funktionalität, um die Selbstbeschreibungen aller Testsysteme zu erfassen. Dazu werden die nötigen Kommandozeilen-Aufrufe abgesetzt, und die Antworten (XML) zu einer XML-Beschreibung zusammengefasst.

Das Kontrollsystem ist die zentrale Komponente des generischen Testsystems. Es übernimmt die Bestimmung der Testaufgaben und sorgt für deren Ausführung. Für die Bestimmung der initialen Testaufgaben ist der Zugriff auf die Testregeln

[91] Es existieren verschiedene Java-Bibliotheken, die eine Validierung gegen unterschiedliche Schemasprachen ermöglichen. Ein Beispiel ist der Xerces2 Java Parser: vgl. *http://xerces.apache.org/xerces2-j/* (zuletzt geprüft am *20.10.2012*)

nötig, die Teil des Hauptprofils sind (jenes, welches die Verbindung zwischen den einzelnen Applikationsprofilen herstellt). Die, für die Tests zu benutzenden, Testsysteme werden in den Testregeln mit einer ID referenziert. Diese IDs können dank der Vorarbeit des *System Scanner*s aufgelöst werden. Der Aufruf eines Testsystems erfolgt über die Kommandozeile, wobei ein solcher durch die Teilimplementierung der *API* (*ATestSystem* - vgl. Beschreibung der *Testsysteme*) verarbeitet wird. Die übergebenen Parameter enthalten ebenso die Pfade, die den temporären Speicherort der, für den Test zu verwendenden, Dateien angeben. Zum Beispiel muss einem Testsystem die zu testende Datei mitgeteilt werden. Auch werden die Reports verwaltet, die durch die einzelnen Testsysteme erstellt werden. Zudem gibt das Kontrollsystem die Rückmeldungen zum Testverlauf weiter, und reiht weitere Testaufgaben ein, wenn dies durch ein Testsystem gewünscht wird. Dabei läuft die Kommunikation über einen Port, der für alle Komponenten gleich ist. Ist ein neuer Profiltest nötig, so werden die Testregeln des entsprechenden Profils verwendet, um die zusätzlichen Testaufgaben zu bestimmen. Somit kann die Liste der auszuführenden Aufgaben stets erweitert werden.
Dabei kann das Kontrollsystem mit der Komponente *Test Control Manager* verglichen werden, die durch das GIT-B-Framework eingeführt wird, und Teil der *Test Bed Core Platform* ist (vgl. S. 50 in [42] sowie Abbildung 2.13 auf Seite 44 in Kapitel 2.3.2) ist. Diese Komponente führt z.B. Plugins vom Typ *Test Suite Engine* und *Document Validator* aus (für eine Beschreibung der Begriffe - siehe Kapitel 4.1). Dabei nutzt sie die *Core Plugin-in API*s, die durch die Plugins implementiert werden. Im generischen Testsystem nutzt das Kontrollsystem jedoch Kommandozeilen-Aufrufe, die durch die Teilimplementierung (*ATestSystem*) einer vorgegebenen *API* (*ITestSystem*) verarbeitet werden (vgl. Beschreibung der *Testsysteme*). Ein weiterer Unterschied ist jedoch, dass die Komponente verschiedene *Core Platform API*s implementiert (vgl. S. 50 in [42] sowie Abbildung 2.13 auf Seite 44 in Kapitel 2.3.2), die von den einzelnen Plugins genutzt werden können. Beim generischen Testsystem werden dem Kontrollsystem jedoch Nachrichten übermittelt, die Rückmeldungen zum Testablauf geben. Auch die zusätzliche Einreihung von Testaufgaben erfolgt über die Interpretation einer Nachricht, die von einem Testsystem ausgeht. Eine *Test Suite* kann hingegen Methoden der *Core Platform API* aufrufen, um z.B. eine Validierung anzufragen (vgl. S. 53-54 in [42]). Zudem gibt das Kontrollsystem vor, wo die Reports abzulegen sind. Im GIT-B-Framework kann die Komponente *Test Control Manager* das Ergebnis erfragen (Methode *GetResult (validation-reference)* der *Core Plug-in API* - vgl. S. 52 in [42]).

Letztlich stellt das generische Testsystem verschiedene Testsysteme zur Verfügung, die für die Ausführung einzelner Testaufgaben genutzt werden können. Der Zusammenhang mit den Modifikationen ergibt sich durch die hergeleiteten Dateien, die für die verschiedenen Tests bemüht werden. Daher soll im folgenden auf die verschiedenen Modifikationen, und ihre Umsetzung eingegangen werden. Die in Kapitel 3.4.1 eingeführten Kategorien konnten zur Einordnung verschiedener Anforderungen verwendet werden, die für das VCD, und das IMS-CC-Profil, bestehen, und mit verschiedenen Modifikationen

umgesetzt wurden bzw. umzusetzen sind. Im folgenden soll erklärt werden, wie diese Modifikationen angewendet werden, und welche Testsysteme für die Überprüfung der resultierenden Anforderungen zuständig sind:

- *Strukturelle Modifikationen* können in das XML-Binding der jeweiligen Basis-Spezifikation übernommen werden. Konkrete Dateien können mit einem XML-Schema-Validator überprüft werden. Dies geschieht durch ein Testsystem, welches eine Java-Bibliothek nutzt, um die eigentliche Validierung vorzunehmen. Ein Beispiel ist die Modifikation SM.1, die sowohl durch das VCD, als auch das IMS-CC-Profil, genutzt wird (vgl. Kapitel 3.4.2 und Kapitel 3.4.3).

- *Bedingte strukturelle Modifikationen* sind nur dann anzuwenden, wenn eine inhaltliche Bedingung erfüllt ist. Diese Art von Modifikation kann nicht immer in ein neues XML-Binding eingearbeitet werden. Falls ein neues XML-Schema erstellt werden kann, müssen sogenannte *Assertions*, oder *Typ-Alternativen*[92] verwendet werden (vgl. Kapitel 2.3.1). Die Modifikation *VCD-02-R007* ist ein Beispiel für diesen Fall. Reichen die Mittel einer Assertion nicht aus, so muss die Modifikation von Hand in ein Schematron-Schema übertragen werden. In diesem Fall werden die Tests durch das Schematron-Testsystem ausgeführt, welches wiederum eine XSLT-Implementierung nutzt[93], um die Überprüfung konkreter Dateien vorzunehmen. Dem Autor sind jedoch keine bedingten, sowie strukturellen, Modifikationen bekannt, die nicht durch Assertions ausgedrückt werden können, und im Rahmen des IMS-CC-Profils, und der VCD-Spezifikation, verwendet wurden.

- Wenn eine Modifikation Anforderungen an *interne Referenzen* stellt, gelten die gleichen Bedingungen, wie für *bedingte strukturelle Modifikationen*. Entweder können die Anforderungen durch ein neues XML-Schema festgehalten werden, oder sie müssen als Teil eines Schematron-Schemas ausgedrückt werden. Die Anforderungen des VCD-Formats enthalten bereits Schematron-Regeln, die für die Überprüfung von internen Referenzen genutzt werden können. Das Tool zur Erstellung und Bearbeitung von Applikationsprofilen (SchemaProf - vgl. Kapitel 4.1) musste jedoch angepasst werden, um die (tool-gestützte) Wiederverwendung dieser Regeln zu ermöglichen. Für das IMS-CC-Profil mussten die bedingten Modifikationen (bzgl. der internen Referenzen) durch Assertions erfasst werden, da die neue Implementierung des STT keine bedingten Modifikationen umsetzen kann[94]. Die, im Rahmen

[92] Diese werden jedoch nicht durch die unterstützenden Tools verarbeitet.

[93] Diese wird unter *http://code.google.com/p/schematron/* bereitgestellt (zuletzt geprüft am *21.10.2012*). XSL Transformation - XSLT - ist eine W3C-Recommendation, und kann unter *http://www.w3.org/TR/xslt20/* eingesehen werden (zuletzt geprüft am *27.10.2012*).

[94] Die Neu-Implementierung war jedoch notwendig (nicht im Rahmen dieser Arbeit). Die, durch die UBL genutzten XML-Schema-Elemente wurden durch die alte Implementierung nicht unterstützt. Zudem war die Berechnung der Instanz-Pfade nicht immer korrekt. Diese beschreiben die Stellen in einem XML-Dokument, an denen ein, im XML-Schema definiertes Element oder Attribut, vorkommen kann. Diese Pfade wurden (beim IMS-CC-Profil) zur Erzeugung von Schematron-Regeln genutzt, die wiederum die bedingten Modifikationen festhielten (vgl. Schematron-Schema in [64]).

dieser Diplomarbeit, vorgenommenen Änderungen werden in Kapitel 4.4 vorgestellt.

- Wenn eine Modifikation Anforderungen an *externe Referenzen* stellt, oder einen Wert mit Hilfe einer *Code-Liste* einschränkt, so können die mit der Modifikation verbundenen Anforderungen mit dem *Additional Constraints*-Formats umgesetzt werden[95]. Dabei werden weitere Arten von Einschränkungen unterstützt. Jede dieser Arten ist mit einem Testsystem verbunden, dass die Überprüfung vornimmt, und dessen Aufruf durch das Constraints-Testsystem veranlasst wird (durch eine Nachricht an das Kontrollsystem). Ein beispielhafter Ablauf wird in Kapitel 4.3 gezeigt. Die Einschränkung von, an einem Extension-Point angebundenen, Datenelementen ist ein weiteres Beispiel für eine solche Anforderung, die jedoch nur beim IMS-CC-Profil verwendet wird. In diesem Punkt mussten Anpassungen erfolgen, da das generische Testsystem keine Code-Listen unterstützt. Dazu wurden die Code-Listen in sogenannte *Vokabulare* (vgl. Kapitel 2.4.2) überführt.

4.3 Testablauf und Kommunikation in einem profil-spezifischen System

Im letzten Kapitel wurden die einzelnen Komponenten des generischen Testsystems besprochen. Dabei nehmen die Testsysteme, und das Kontrollsystem, eine zentrale Rolle ein. Letzteres übernimmt die Kontrolle über den Testablauf, wenn ein konkretes Datenpaket überprüft werden soll (IMS-CC / VCD-Package). Dabei werden die anfänglich auszuführenden Testaufgaben mittels der Testregeln bestimmt (vgl. Kapitel 4.2).
Dieses Kapitel soll den Ablauf eines Tests genauer beschreiben. Dabei lässt sich der Ablauf in chronologische, vom Kontrollsystem auszuführende, Aufgaben unterteilen:

1. Als erstes müssen die Konfiugrationsdateien eingelesen werden. Diese enthalten Informationen zu
   - den Datenpaketen, die durch das Testsystem überprüft werden sollen.
   - dem Verzeichnis, in welchem die Test-Reports abgelegt werden sollen (XML und HTML).
   - einem sogenannten *Listener*, welcher dem Benutzer eine Rückmeldung zum Testverlauf geben soll[96].
   - den sogenannten Log-Leveln, welche eine Schranke für die Meldung von Fehlern festlegen[97].
   - den Profilen, mit denen das Testsystem konfiguriert wurde.
   - den Testsystemen, die durch den System-Scanner gemeldet wurden.

[95] Code-Listen werden durch die, im Rahmen dieser Arbeit erfolgten, Anpassungen unterstützt (vgl. Kapitel 4.4).
[96] Die Meldungen können z.B. in eine Datei geschrieben werden.
[97] Zum Beispiel kann vorgegeben werden, dass nur schwerwiegende Fehler gemeldet werden.

2. Als nächstes werden die Profile, und die Daten der, zu überprüfenden Datenpakete, entpackt. Somit muss dieser Vorgang nicht von den einzelnen Testsystem wiederholt werden.

3. Anschließend werden die Testregeln des Profils eingelesen, dass für einen Test verwendet werden soll[98].

4. Schließlich erfolgt die Erstellung der Testaufgaben, die aufgrund der Testregeln, und den, zu verarbeitenden, Datenpaketen erfolgt.

5. Als nächstes werden die Testaufgaben nacheinander abgearbeitet, wobei für jede Aufgabe die folgenden Schritte anfallen:

   (a) Aufruf des Testsystems, welches in der zugrunde liegenden Testregel referenziert wurde. Die Beschreibung der, im generischen Testsystem registrierten, Testsysteme wurde bereits eingelesen (vgl. Punkt 1), und enthält ebenso die IDs der einzelnen Testsysteme. Eine jede Beschreibung enthält ebenso die Beschreibung der Parameter, die für die Erstellung des Kommandozeilen-Aufrufs benötigt werden. Als Teil dieser Parameter wird z.B. der Pfad zu einer Datei übergeben, die die Fehlermeldungen (XML) aufnehmen soll.

   (b) Aufbau einer Verbindung zum Testsystem. Diese erfolgt über einen bestimmten Port, der durch alle Testsysteme genutzt wird. Auf der Seite des Testsystems wird die Verbindung automatisch initialisiert. Dies geschieht durch die Teilimplementierung eines bestimmten Interfaces (abstrakte Java-Klasse *ATestsystem* - vgl. Kapitel 4.2).

   (c) Erfassung aller, durch das Testsystem versendeten Nachrichten. Diese können Rückmeldungen zum Testverlauf geben, oder Anforderungen für die Einreihung weiterer Tests enthalten. Meldungen werden sofort an den zuständigen *Listener* übergeben, der diese an den Nutzer weitergibt.

   (d) Erstellung der Testaufgaben, die für die Ausführung der angeforderten Tests notwendig sind. Die neuen Testaufgaben werden vor den, noch nicht bearbeiteten, Testaufgaben eingereiht, so dass sie als nächstes ausgeführt werden. Wenn Dateien gegen ein Profil getestet werden müssen, werden die Testregeln des genannten Profils verwendet, um die notwendigen Testaufgaben zu bestimmen. Ein Beispiel ist die Überprüfung von VCDs, die innerhalb der XML-Beschreibung eines VCD-Pakets referenziert werden. Wird eine solche Referenz gefunden, so müssen die, für das referenzierte VCD auszuführenden, Tests bestimmt werden. In diesem Fall werden z.B. ein XML-Schema-Test, ein Schematron-Test, und ein Test gegen Additional-Constraints benötigt (letzterer muss die Referenzen innerhalb des VCDs überprüfen).

6. Erstellung der HTML-Reports. Diese beruht auf verschiedenen Dateien:

[98] Dieses ist bei beiden Applikationsprofilen (IMS-CC / VCD) das Hauptprofil, welches Teil eines Domain-Profils ist.

- den XML-Dateien, die die Fehlerbeschreibungen beinhalten, und die durch die einzelnen Testsysteme erstellt wurden.
- verschiedenen XSLT-Stylesheets[99], die für das Zählen der Fehler genutzt werden können. Diese werden durch die einzelnen Testsysteme bereit gestellt, und arbeiten auf dem jeweiligen XML-Format.
- verschiedenen XSLT-Stylesheets, die für die Erstellung eines einzelnen HTML-Reports verwendet werden können. Diese werden ebenso durch die einzelnen Testsysteme bereit gestellt, und können dazu genutzt werden, die Fehlerbeschreibungen in HTML-Seiten zu transformieren.
- verschiedenen XSLT-Stylesheets, die für die Erstellung der Übersichtsseiten genutzt werden können, die schließlich die einzelnen HTML-Reports verlinken. Diese Stylesheets werden durch das Kontrollsystem bereit gestellt.

Dabei nimmt die Überprüfung der *Additional Constraints* eine besondere Stellung ein. Diese kann durch verschiedenen Testaufgaben gefordert sein (vgl. *Punkt 5*), die aufgrund der Testregeln eines Profils erstellt wurden (vgl. *Punkt 4* und *Punkt 5d*). Das zuständige Testsystem muss u.U. verschiedene Arten von Anforderungen überprüfen. Diese werden jedoch nicht durch das Testsystem selbst, sondern durch andere Testsysteme geprüft. Dabei wird eine neue Datei erstellt, die einen Teil der Anforderungen enthält, und die vom zuständigen Testsystem verwendet wird. Der Ablauf soll mit Hilfe von Abbildung 4.5 auf Seite 107 erklärt werden. Diese zeigt die Aufrufe, und die Nachrichten, die während eines beispielhaften Tests anfallen können:

1. das Kontrollsystem ruft das Testsystem auf, dass für die Überprüfung der *Additional Constraints* zuständig ist (vgl. *Punkt 5a*). Zum Beispiel kann ein Test notwendig sein, der die, in einer XML-Beschreibung eines VCD-Pakets, enthaltenen Datei-Referenzen überprüft.

2. das Testsystem verarbeitet die *Additional Constraints*-Datei, in der die Anforderungen formuliert sind. Diese betreffen bestimmte Inhalte des, zu überprüfenden, Dokuments (die z.B. Teil der XML-Beschreibung eines VCD-Pakets sind). Neben anderen Anforderungen (z.B. für XML-Attribut-Werte, die einem bestimmten Vokabular entstammen müssen[100]), verarbeitet es diejenigen Anforderungen, die die Auflagen für bestimmte Datei-Referenzen definieren.

3. das Testsystem extrahiert die, für die Tests benötigten Inhalte, und schreibt diese mitsamt den Anforderungen in eine separate Datei. Im Beispiel wurden Referenzen auf Dateien, wie z.B. auf ein VCD, gefunden.

[99] Ein XSLT-Stylesheet basiert auf XML, und enthält Elemente, die durch die Sprache *XSLT* vorgegeben sind. Dabei wird eine Transformation definiert, die eine Menge von XML-Dokumenten in weitere XML-Dokumente überführt. XSLT ist eine W3C-Recommendation (vgl. *http://www.w3.org/TR/xslt20/* - zuletzt geprüft am *27.10.2012*).

[100] Code-Listen werden in Vokabulare überführt werden - vgl. Kapitel 4.4.

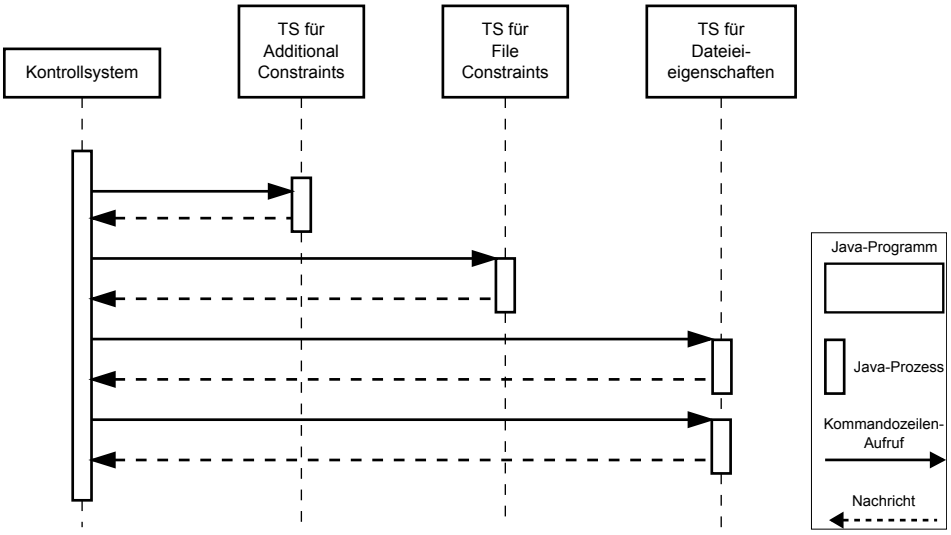


Abbildung 4.5: Delegierung der Aufgaben bei einem *Additional-Constraints*-Test

4. das Testsystem teilt dem Kontrollsystem mit, dass verschiedene Tests notwendig sind. Dazu werden jeweils das zu verwendende Testsystem, und die zu benutzenden Anforderungen, angegeben. Die ID des Testsystems wird dabei aus der Art der Anforderung (String) ermittelt. D.h. es können neue Anforderungen eingeführt werden, die keine Änderung des Testsystems erfordern.

5. das Kontrollsystem erstellt letztlich die neuen Testaufgaben und führt diese als nächstes aus. In dem, in Abbildung 4.5 auf Seite 107 gezeigten, Fall wurden Datei-Referenzen extrahiert, und das zuständige Testsystem aufgerufen.

6. das aufgerufene Testsystem nutzt sowohl die extrahierten Inhalte, als auch die, in der bereitgestellten Datei enthaltenen, Anforderungen, um die Tests auszuführen. Im Fall von gefundenen Datei-Referenzen werden die Tests an ein zuständiges Testsystem delegiert.

4.4 Nötige Anpassungen für das VCD

In den letzten Kapiteln wurde auf verschiedene Lösungen eingegangen. Diese erlauben einen Arbeitsablauf, der von der Erfassung der Anforderungen bis zur Erstellung eines Testsystems reicht, welches schließlich für die Überprüfung der Anforderungen genutzt werden kann. Dabei ist die Erstellung eines *Applikationsprofils* der erste Schritt. Ein solches wird aufgrund einer Basisspezifikation erstellt, und ist ein Dateipaket, welches das XML-Binding der Basisspezifikation, die Modifikationen an diesem, die hergeleiteten Dateien, und die Regeln zur Testerzeugung enthält. Die Modifikationen adressieren Elemente des XML-Bindings, und basieren auf den Modifikationen, die auf der konzeptuellen Ebene definiert wurden. Ein Teil der Modifikationen kann durch ein neues Binding

erfasst werden. Dieses ist Teil der hergeleiteten Dateien, die ebenso die Anforderungen festhalten, die aus den verbleibenden Modifikationen resultieren. Schließlich stellen die Regeln zur Testerzeugung eine Verbindung zum generischen Testsystem, und den potentiellen Dateipaketen her. Sie erlauben die Herleitung von Testaufgaben, die auszuführende Tests beschreiben. Dabei kann das *generische Testsystem* genutzt werden, um ein konkretes Testsystem zu erstellen. Ein solches beinhaltet das sogenannte Kontrollsystem, welches die Bestimmung und die Ausführung der Testaufgaben übernimmt. Dabei nutzt das Kontrollsystem verschiedene Testsysteme, die ein Dokument mit Hilfe der, in einer hergeleiteten Datei formulierten, Anforderungen überprüfen können (vgl. Kapitel 4.1, Kapitel 4.2 und Kapitel 4.3).
Die Erstellung des Applikationsprofils und der entsprechenden Testsysteme kann zudem durch verschiedene Tools unterstützt werden. Dadurch können anfallende Arbeiten automatisiert werden, wodurch letztlich weniger Detailwissen nötig ist, und die Arbeitsteilung erleichtert wird. Dabei existieren bereits unterstützende Tools, die für die Erstellung des IMS-CC-Profils, und der benötigten Testsysteme verwendet wurden (vgl. [24]). Dies geschah im Rahmen eines Ablaufs, der in Abbildung 4.6 auf Seite 108 gezeigt wird:[101]

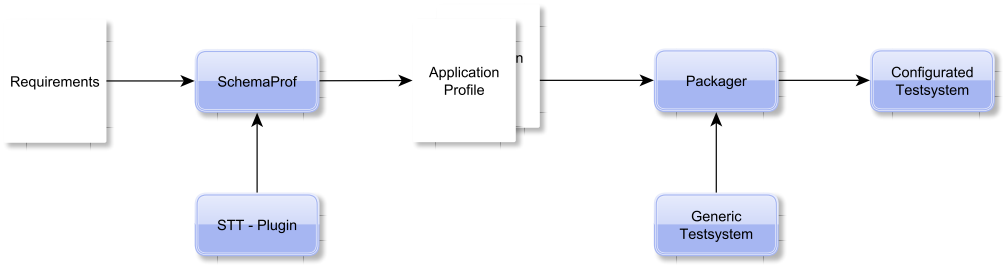


Abbildung 4.6: Angewandte Methode zur Erstellung der Applikationsprofile und der zugehörigen Testsysteme

- Zuerst können die Modifikationen mit dem Tool *SchemaProf* erfasst werden. Die Modifikationen können grafisch erfasst werden, wobei die Komponenten des Basis-Schemas mit einer Baum-Ansicht visualisiert werden. Das Tool ermöglicht ebenso die Erstellung einer Testkonfiguration, die im wesentlichen die Regeln zur Testerzeugung beinhalten[102].

[101] In [24] werden diese Tools ebenso beschrieben. Dabei wird jedoch auf ältere Versionen eingegangen, die mittlerweile überarbeitet wurden. Die Beschreibung kann daher von der, in dieser Arbeit formulierten, Beschreibung abweichen. Dies betrifft z.B. das *Schema Transform Tool* - STT.

[102] Neben den Testregeln müssen ebenso Zuordnungen erfolgen, die einem XML-Schema-Validator dabei helfen, bestimmte Namespaces aufzulösen. Ein Beispiel ist die Einschränkung von Elementen, die an einem Extension-Point eingebunden werden können. Eine Modifikation kann die gültigen Elemente mittels eines XML-Schemas einschränken, welches einen Namespace vorgeben kann. In diesem Fall kann das hergeleitete XML-Schema zwar den Namespace festhalten, einem Document-Validator muss jedoch das XML-Schema mitgeteilt werden, das zur Validierung der Elemente verwendet werden soll. Diese Zuordnung ist ebenso ein Teil der Testkonfiguration.

- das *Schema Transform Tool* - STT - kann zur Umsetzung der Modifikationen genutzt werden. Es wird als Standalone- und als Plugin-Version zur Verfügung gestellt. In dem gezeigten Ablauf wird es zusammen mit SchemaProf genutzt. Bei der Herleitung werden ein neues XML-Schema und eine Additional-Constraints - AC - Datei erzeugt. Das hergeleitete XML-Schema kann Assertions enthalten, wenn inhaltliche Bedingungen angegeben werden. Die AC-Datei enthält die Modifikationen, die nicht mit einem XML-Schema ausgedrückt werden können. Bei der Herleitung kann eine Testkonfiguration erstellt werden, die anschließend in SchemaProf überarbeitet werden kann.

- der sogenannte *Packager* kann dazu genutzt werden, ein konkretes Testsystem zu erstellen. Das Tool verwaltet verschiedene generische Testsysteme, und kann ein solches mit einem Domain-Profile konfigurieren.

Die vorgestellten Programme sollten ebenso zur Erfassung der VCD-bezogenen Modifikationen und der Herleitung entsprechender Testsysteme genutzt werden. Ein Teil der Modifikationen konnten jedoch nicht durch SchemaProf erfasst werden. Das Tool unterstützt lediglich die Erfassung von Modifikationen, die durch das STT unterstützt werden. Zudem konnten die, aus diesen Modifikationen resultierenden, Anforderungen nicht überprüft werden. Im wesentlichen betrifft dies 3 verschiedene Arten von Modifikationen:

- Modifikationen, die auf sogenannte *Co-Constraints* (vgl. Kapitel 2.3.1) zurückgreifen. Die Herleitung neuer Anforderungen ist in vielen Fällen nicht möglich[103]. Im PEPPOL-Projekt wurden jedoch viele der festgelegten Anforderungen in Schematron-Regeln überführt. Das Tool wurde deshalb um eine Import-Möglichkeit erweitert, die die Erfassung der betroffenen Anforderungen erlaubt. Bei der Bearbeitung einer Testregel kann ein Schematron-Schema angegeben werden, welches den hergeleiteten Dateien hinzugefügt wird. Dabei entsprechen die, innerhalb des PEPPOL-Projekts verwendeten, Schemata dem ISO/IEC 19757-3 Standard[6][104]. Dieser erlaubt die Erstellung mehrerer Dateien, die zusammen ein Schematron-Schema beschreiben (vgl. Element *include* in [6]). Liegen mehrere Dateien vor, so werden diese beim Import zu einer Schematron-Datei zusammengefasst (vgl. Schritt 1 in Abbildung 4.7 auf Seite 110). Dabei stellt das generische Testsystem ein Testsystem zur Verfügung, welches für eine Überprüfung von Dateien verwendet werden kann. Dieses unterstützte die, im PEPPOL-Projekt genutzten, Möglichkeiten jedoch nicht. Somit musste die Implementierung des Testsystems angepasst werden. Die Validierung erfolgt in mehreren Schritten, die in Abbil-

[103] Additional-Constraints können von inhaltlichen Bedingungen abhängen. Zudem können sogenannte Assertions definiert werden. Diese beinhalten einen XPath-Ausdruck, der im Kontext eines XML-Elements ausgewertet wird. Das Ergebnis muss dem Wert TRUE entsprechen. Die Erfassung anderer, bedingter, Modifikationen ist jedoch nicht möglich.

[104] Der Standard wird unter *http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html* zur Verfügung gestellt (zuletzt geprüft am 27.10.2012).

dung 4.7 auf Seite 110 gezeigt werden[105]. In jedem Schritt ist die Anwendung eines *XSLT-Stylesheets* notwendig, die stets zu einem neuen XML-Dokument führt:[106]

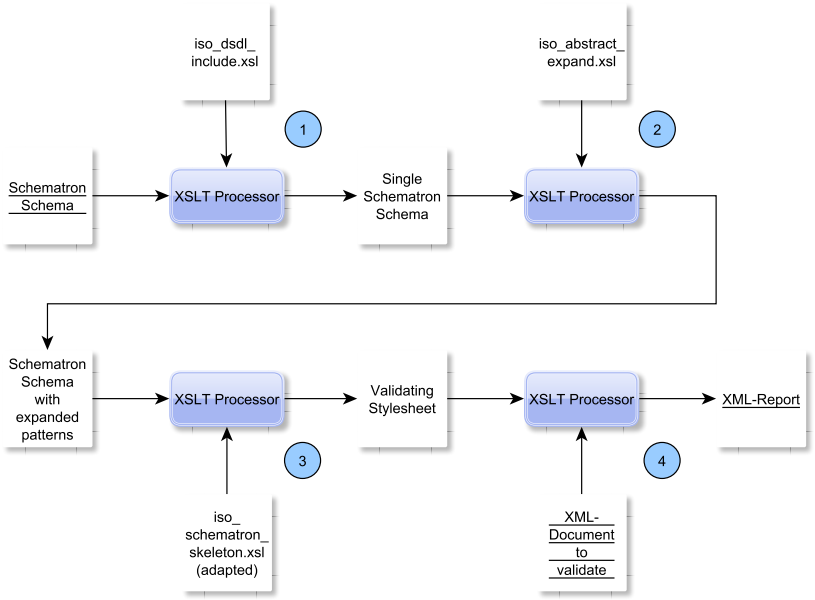


Abbildung 4.7: Anpassung des Schematron-Testsystems: Ablauf einer Schematron-Validierung

1. Im ersten Schritt werden die verschiedenen Teile des Schematron-Schemas zu einer Datei zusammengefasst.
2. Diese können jedoch auf sogenannte *abstract patterns* (vgl. [6]) zurückgreifen, die Vorlagen für bestimmte Sprachkonstrukte (*patterns*) sind. Die Nutzung dieser Vorlagen wird durch eine weitere Transformation berücksichtigt.
3. Im nächsten Schritt wird das XSLT-Stylesheet erstellt, das für die Validierung konkreter XML-Dokumente genutzt werden kann. Im Fall des Schematron-Testsystems wurde das XSLT-Stylesheet in einer vorgesehenen Weise angepasst, so das ein bestimmtes XML-Format genutzt wird, um die Fehler festzuhalten.
4. Im letzten Schritt erfolgt die eigentliche Validierung. Die Ausgabe ist ein XML-Report, der durch die, durch das Testsystem bereitgestellten XSLT-

[105]Diese Schritte sind durch die verwendete XSLT-Implementierung vorgegeben. Die Implementierung wird unter *http://code.google.com/p/schematron/* bereitgestellt. Die beiliegende Readme-Datei erklärt den Ablauf (zuletzt geprüft am *27.10.2012*).

[106]Die Ausführung der XSLT-Stylesheets erfolgt mit einer Java-Bibliothek. Es existieren verschiedene Java-Implementierungen, die die Anwendung von XSLT-Stylesheets erlauben. Ein Beispiel ist Xalan-Java. Die Implementierung wird unter *http://xml.apache.org/xalan-j/* bereitgestellt (zuletzt geprüft am *27.10.2012*).

Stylesheets (für das Zählen der Fehler und die Erstellung der XML-Reports), wiederverwendet werden kann.

Dabei wurden die ersten beiden Transformationen in den Validierungs-Ablauf eingearbeitet. Zudem erfolgten Anpassungen an den XSLT-Stylesheets, die zur Formatierung der HTML-Reports verwendet wurden.

- Modifikationen, die der *Einschränkung von Datei-Referenzen* dienen (vgl. ACM.4). Das Datenmodell der VCD-Spezifikation führt Datenelemente ein, die auf externe Dateien verweisen können. Dabei ist das Datenmodell durch die Transaktionsmodelle, und den entsprechenden XML-Bindings, mit verschiedenen Dokumenttypen verbunden, die für die Beschreibung eines konkreten VCD-Pakets, oder eines der enthaltenen VCDs, genutzt werden können. So kann ein VCD z.B. mit Hilfe der BIE *VCDReferenceID* referenziert werden (vgl. Kapitel 3.1.3). In einem konkreten XML-Dokument wird eine solche Referenz durch den Inhalt eines gleichnamigen XML-Elements repräsentiert (vgl. S. 42 in [13]), wobei das referenzierte VCD während eines Testlaufs überprüft werden muss (vgl. Kapitel 3.4.3). Diese Anforderung kann grundsätzlich durch die Modifikation ACM.4 erfasst werden. Allerdings muss eine Datei-Referenz in der vorgestellten Lösung durch einen *Uniform Resource Locator* - URL - ausgedrückt werden. Ein VCD wird jedoch mit einer ID referenziert, mit deren Hilfe sich der Pfad berechnen lässt (vgl. Kapitel 3.1.3). Somit wurden die folgenden Anpassungen vorgenommen:
  - das *XML-Format gültiger Additional-Constraints-Dateien* wurde geändert. Eine solche beinhaltet die Umsetzung einer derartigen Modifikation. Dem Format wurde die Angabe einer Berechnungsgrundlage hinzugefügt, die für jede Additional-Constraint-Modifikation verwendet werden kann. Im Detail können die, in einem XML-Dokument gefundenen, Werte transformiert werden (vgl. Abbildung 4.5 auf Seite 107 und Erklärung), bevor der eigentliche Test erfolgt. Für die Berechnung kann Javascript-Code angegeben werden, wobei der Inhalt durch eine spezielle Variable referenziert wird.
  - die *Methoden zur Verarbeitung einer Additional-Constraints-Datei* wurden zusammen mit *den zugehörigen Datenmodellen* angepasst. SchemaProf, STT, und das Constraints-Testsystem müssen das neue Format korrekt einlesen (alle Komponeten) bzw. schreiben (SchemaProf und STT) können.
  - für SchemaProf mussten *einzelne Dialoge erstellt bzw. angepasst* werden, um die grafische Erfassung des Javascript-Codes zu ermöglichen. Dieser kann in einem Editor verfasst werden, der u.a. Syntax-Hervorhebung unterstützt, und durch eine Java-Bibliothek zur Verfügung gestellt wird[107].
  - das *Constraints-Testsystem* muss schließlich die Transformationen durchführen und die Ergebnisse für die weitere Verarbeitung der Testaufgaben nutzen.

[107] Diese wird unter einer modifizierten BSD-Lizenz bereitgestellt. Vgl. *http://fifesoft.com/rsyntaxtextarea/index.php* (zuletzt geprüft am *28.10.2012*).

Genauer gesagt müssen die transformierten Werte in eine Datei übernommen werden, die ebenso die, an die Inhalte gestellten, Anforderungen beinhaltet (vgl. Abbildung 4.5 auf Seite 107 und Erklärung).

- Modifikationen, die auf einer *Code-Liste* basieren. Im PEPPOL-Projekt wurden sogenannte Code-Listen erstellt. Diese legen eine Menge von Codes fest, die für die Einschränkung von Werten genutzt werden können (vgl. [13]). Dabei wird das sogenannte *Genericode*-Format verwendet, welches zu einer Spezifikation gehört, die durch das Konsortium *Organization for the Advancement of Structured Information Standards* - OASIS - bereitgestellt wird[108]. In den *Application Profiling Guidelines* wird eine ähnliche Modifikation beschrieben, für die ein kontrolliertes Vokabular genutzt werden kann, um die gültigen Werte eines Informationselements einzuschränken (vgl. Modifikation ACM.2 - bzw. Modifikation ACM.1 - in Kapitel 2.4.2 und [65]). Dabei wird in der Beschreibung auf die *IMS Vocabulary Definition Exchange* - VDEX - Spezifikation verwiesen (vgl. [20]), die Informationen zur empfohlenen Verwendung von Vokabularen enthält (vgl. [64]). Die Spezifikation gibt ein Informationsmodell und ein XML-Binding[21] für Vokabulare vor ([20]). Da auch das Genericode-Format mit einer XML-Syntax verbunden ist, konnte ein XSLT-Stylesheet verwendet werden, um die XML-Repräsentation der Code-Listen in das VDEX-Format (XML) zu überführen. Schließlich kann in SchemaProf ein externer Speicherort angegeben werden, unter dem das hergeleitete Vokabular abgelegt wurde. Im generischen Testsystem gibt es ein spezifisches Testsystem, welches für die Überprüfung von Vokabularen zuständig ist. Dieses musste nicht geändert werden.

[108] Die Spezifikation kann unter *http://docs.oasis-open.org/codelist/ns/genericode/1.0/* eingesehen werden (zuletzt geprüft am *27.10.2012*).

5 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden zwei Spezifikationen betrachtet, die verschiedene paket-basierte Datenformate einführen. Diese greifen jeweils auf XML-basierte Dokumente - bzw. Meta-Dateien - zurück, die Teil eines Datenpakets sind und die Inhalte beschreiben (vgl. Kapitel 3.1.3 und Kapitel 3.2.3).
Dabei wurden diese Spezifikationen vor einem unterschiedlichen Hintergrund entwickelt. So stand bei der Spezifikation IMS Common Cartridge - IMS-CC - die Zusammenstellung von Lerninhalten im Mittelpunkt, die in eine Lernplattform importiert werden können (vgl. Kapitel 3.2.1). Bei der Spezifikation Virtual Company Dossier - VCD - stand dagegen die Teilnahme an einer öffentlichen Ausschreibung im Mittelpunkt, für die verschiedene Nachweise erbracht werden müssen (vgl. Kapitel 3.1.1). Beide Formate dienen jedoch der Zusammenstellung verschiedener Dateien, die innerhalb vorgesehener Abläufe benötigt werden. Zudem muss ein Datenpaket in beiden Fällen einer Reihe von Anforderungen genügen (vgl. Kapitel 3.3).
Die Überprüfung dieser Anforderungen wurde durch die Spezifikation IMS-CC jedoch auf eine besondere Art und Weise gelöst. Die Spezifikation wurde als sogenanntes *Domain Profile* festgehalten (vgl. Kapitel 3.2.2), welches im Zusammenhang mit einem generischen Testverfahren verwendet werden kann. Dieses erlaubt die automatische Erstellung von Testsystemen, wodurch konkrete Datenpakete überprüft können (vgl. Kapitel 4).
Somit stellte sich die Frage nach der allgemeinen Anwendbarkeit des Verfahrens, wenn die Anforderungen eines paket-basierten Datenformats erfasst werden sollen. So sollte die Anwendbarkeit für die Datenformate ermöglicht werden, die durch die VCD-Spezifikation eingeführt werden.
Dabei ist der Begriff des *Domain Profile* von zentraler Bedeutung. Ein solches ist die Basis für die Erstellung konkreter Testsysteme (vgl. Kapitel 4.1). Somit müssen die Anforderungen der VCD-spezifischen Formate mit Hilfe eines *Domain Profile* formalisiert werden, um diese Lösung wiederzuverwenden. Dabei kann ein *Domain Profile* selbst auf verschiedenen Spezifikationen (bzw. Standards) basieren, und erlaubt die Anpassung der zu Grunde liegenden Spezifikationen (vgl. *Domain Profile* - S. 5 in [65]). Diese Anpassungen betreffen jeweils ein Informationsmodell, welches wiederum einer Spezifikation angehört (vgl. Kapitel 2.4.1 und Begriff *Data Profile* - S. 8 in [65][109]). Schließlich können diese Modifikationen angewendet werden, was zu neuen Anforderungen führt. Die für ein Datenpaket bestehenden Anforderungen ergeben sich somit aus den zu Grunde liegenden Spezifikationen und den vorgenommenen Modifikationen.
Dabei ist ein Informationsmodell jedoch eine abstrakte Informationsstruktur, die nicht an eine spezifische Technologie gebunden ist (vgl. S. 11 in [63]). Die konkrete Repräsentation der Daten wird erst durch ein sogenanntes XML-Binding eingeführt, welches eine Verbindung zu einer Syntax herstellt, und mit Hilfe eines XML-Schemas beschrieben werden kann (vgl. *Bound Data Profile* - S. 8, 11 in [65]). Ein solches kann jeweils das Format von XML-Dokumenten festlegen, die Teil eines Pakets sind (vgl. Spezifikation IMS-CC

[109] Ein *Domain-Profile* basiert auf mehreren Applikationsprofilen (vgl. Kapitel 2.4.3 für eine Erklärung). Zudem ist ein Informationsmodell Teil einer *Learning Technology Specification* (vgl. S. 6 in [65]).

- Kapitel 3.2.3).
Aufgrund der verschiedenen XML-Bindings und der vorgenommenen Modifikationen können weitere Dokumente hergeleitet werden, die einen Teil der Modifikationen reflektieren und für eine Validierung genutzt werden können (vgl. Kapitel 4.1 und Kapitel 4.2). Der Zusammenhang mit der Paketstruktur entsteht über die Datei-Referenzen, die Teil einer Meta-Datei sein können (vgl. Spezifikation IMS-CC[64] und Kapitel 3.2.3). Dabei kann eine Modifikation verschiedene Auflagen definieren, die für eine referenzierte Datei gelten müssen (vgl. Kapitel 2.4.3). Letztlich liegt der wesentliche Vorteil dieser Verfahrensweise in dem bereits erwähnten Verfahren, das für die automatisierte Erstellung konkreter Testsysteme genutzt werden kann (vgl. [24] und Kapitel 4).
Die Spezifikation Virtual Company Dossier - VCD - stützt sich jedoch nicht auf ein Informationsmodell. Vielmehr werden die auszutauschenden Daten mit der Hilfe von sogenannten *Business Information Entities* modelliert (vgl. Kapitel 3.1.3), die jedoch ebenso als Ziel einer Modifikation in Frage kommen. Für text-basierte Modifikationen kommen sogenannte *Basic Business Information Entities* - Basic BIEs - in Frage, während sich Struktur-bezogene Modifikationen auf *Aggregate BIEs* beziehen können (vgl. Erklärung der *BIEs* in Kapitel 2.2.1 und Erklärung der Modifikationen in Kapitel 2.4.2). Dabei gibt die VCD-Spezifikation ebenso ein XML-Binding vor, welches die Datenmodelle mit einer XML-Syntax verbindet (vgl Kapitel 3.1.3).
Somit konnte die Erstellung verschiedener *Domain-Profiles* eine Lösung sein, um die Anforderungen der, durch die VCD-Spezifikation eingeführten, paket-basierten Formate festzuhalten. Daher wurde der Versuch unternommen, die Erstellung der verschiedenen *Domain-Profiles*, und die Herleitung der konkreten Testsysteme, zu ermöglichen. Dies sollte ausschließlich mit Mitteln geschehen, die den generischen Ansatz nicht einschränken.
In diesem Sinne wurden die Anforderungen auf einer abstrakten Ebene verglichen (vgl. Kapitel 3.3). Die dabei gewonnenen Erkenntnisse führten schließlich zur Bestimmung der gemeinsamen Anforderungen und der umzusetzenden Modifikationen. Das Ziel einer Modifikation war dabei stets durch verschiedene BIEs gegeben, die ebenso durch die, innerhalb des PEPPOL-Projekts formulierten Anforderungen, referenziert wurden (vgl. Kapitel 3.4).
Für die syntaktische Erfassung der Modifikationen mussten jedoch Änderungen an den unterstützenden Tools erfolgen, die die Arbeit mit einem Domain Profile erleichtern. Dabei sollte die Erstellung eines *Domain-Profile* mit *SchemaProf* und dem *Schema Transform Tool* - STT - erfolgen. Ferner sollte der sogenannte *Packager* genutzt werden, der ein konkretes Testsystem erstellen kann. Dabei greift dieser auf ein modular aufgebautes Testsystem zurück, welches die, für ein Datenpaket notwendigen Tests ausführen kann, und mit dem erstellten *Domain-Profile* konfiguriert werden muss (generisches Testsystem). Eine notwendige Grundlage stellen die, durch das STT, hergeleiteten Dateien dar, die aufgrund der Modifikationen erstellt wurden. Diese sind Teil des *Domain-Profile* und halten die Anforderungen fest, die nach der Einarbeitung der Modifikationen bestehen. Ein erstelltes Testsystem greift auf diese Dateien zurück, um die eigentlichen Tests auszuführen (vgl. Kapitel 4.4).

Somit wurden Änderungen vorgenommen, die die Verwendung der unterstützenden Tools ermöglichen. Diesen gingen jedoch grundlegende Anpassungen voraus, die die manuelle Erfassung der Modifikationen, die manuelle Herleitung der Dateien und die Überprüfung der resultierenden Anforderungen erlauben (vgl. Kapitel 4.4). Somit lassen sich die vorgenommenen Änderungen in zwei Kategorien einordnen:

1. Zum einen wurden Änderungen vorgenommen, um bestimmte Anforderungen zu unterstützen, die den verschiedenen, in Kapitel 3.4.1 eingeführten, Kategorien zugeordnet werden können. Diese können das Format der Applikationsprofile, das Format der hergeleiteten Dateien und das generische Testsystem betreffen, welches die Grundlage für die automatische Erstellung der konkreten Testsysteme darstellt. Ein Profil muss die gewünschten Modifikationen aufnehmen können. Die resultierenden Anforderungen müssen in den hergeleiteten Dateien festgehalten werden können. Diese müssen schließlich durch ein Testsystem überprüft werden können. Dabei können die Anpassungen auf unterschiedliche Art und Weise vorgenommen werden. So erlaubt das generische Testsystem die Einbindung von Komponenten, die zur Überprüfung spezieller Anforderungen genutzt werden können. Sind die zu überprüfenden Anforderungen durch die Implementierung gegeben, so kann die Lösung spezifisch sein. Sinnvoller ist die Einführung, oder die Anpassung, von Formaten, mit deren Hilfe die Anforderungen formal festgehalten werden können. Ein solches Format kann dann durch eine zuständige Komponente des generischen Testsystems verarbeitet werden. Das neue Format kann somit durch andere Spezifikationen genutzt werden. Ein weiterer Vorteil besteht, wenn eine solche Datei aufgrund verschiedener Modifikationen hergeleitet werden kann. In diesem Fall kann ein Zusammenhang zwischen den ursprünglichen Modifikationen und den resultierenden Anforderungen hergestellt werden.

2. Neben diesen Änderungen wurden Anpassungen an den unterstützenden Tools vorgenommen, die Teile anfallender Aufgaben übernehmen oder vereinfachen. So werden die grafische Erstellung eines Domain-Profile (SchemaProf), die automatische Herleitung der, für die Tests benötigten, Dateien (STT), und die Kommandozeilenbasierte Erstellung konkreter Testsysteme (Packager) ermöglicht (vgl. Kapitel 4.4). Solche Änderungen reduzieren das Expertenwissen, dass für eine Anwendung der Lösung nötig ist. So ist ein Nutzer nicht mehr mit allen Details der Lösung konfrontiert. Auf der anderen Seite können sich Änderungen der ersten Art (siehe erster Punkt der Aufzählung) ebenso auf die unterstützenden Tools auswirken. Ist z.B. eine Modifikation anzupassen, so kann ebenso die Anpassung von SchemaProf und dem STT nötig sein.

Unter den vorgestellten Gesichtspunkten können die, im Rahmen dieser Arbeit vorgestellten Lösungen letztendlich näher beurteilt werden. So wurde die Menge der erfassbaren Anforderungen erweitert, um die Überprüfung von Datei-Referenzen zu ermöglichen, die zuerst in einen Dateipfad umgerechnet werden müssen. Diese Änderungen betrafen jedoch viele Bereiche der vorgestellten Lösung. So musste das Format der Applikationsprofile, das Format einer sogenannten Additional-Constraints-Datei, SchemaProf, das

STT und das generische Testsystem angepasst werden. Somit setzte diese Lösung die teilweise Kenntnis der jeweiligen Implementierungen, und die Kenntnis des involvierten Formats voraus. Diese Lösung wurde favorisiert, da die vorhandenen Mechanismen sinnvoll genutzt werden konnten, und die Menge der erfassbaren Anforderungen erweitert wurde.
Ein anderes Bild ergibt sich bzgl. der Änderungen, die die Verwendung von Code-Listen (für eine Erklärung - vgl. S. 37 in [13]) erlauben. Diese konnten in sogenannte Vokabulare (für eine Erklärung - vgl. Kapitel 4.4) überführt werden, die dann im Rahmen einer Modifikation genutzt werden konnten. Dabei basierten beide Formate auf der XML, was eine Transformation ermöglichte, die durch ein XSLT-Stylesheet festgehalten werden konnte. Somit mussten keine Änderungen am eigentlichen Verfahren vorgenommen werden (vgl. Kapitel 4.4).
Letztlich verblieben Anforderungen, die bereits durch Schematron-Regeln erfasst wurden. Diese wurden größtenteils übernommen, und den hergeleiteten Dateien der einzelnen Domain-Profiles hinzugefügt. D.h. ein Großteil dieser Anforderungen wird auf syntaktischer Ebene nicht durch Modifikationen festgehalten. Die Übersetzung dieser Modifikationen und die Anwendung muss somit von Hand erfolgen, wobei der Zusammenhang nicht durch das Domain-Profile ersichtlich wird. Diese Lösung erforderte eine Importfunktion in SchemaProf, und die Anpassung des generischen Testsystems. Letzteres stellte zwar ein Testsystem für Schematron-Tests bereit, dass allerdings das im PEPPOL-Projekt verwendete Format nicht unterstützte. Letztlich wurde für die Anpassung des Testsystems weiteres Detailwissen vorausgesetzt (vgl. Kapitel 3.4.3). Die Menge der erfassbaren Anforderungen wurde jedoch erweitert. Da durch den Import keine Verbindung zu einer, auf der syntaktischen Ebene erfassten Modifikation, besteht sollte jedoch geprüft werden, ob diese Anforderungen nicht in entsprechende Modifikationen übersetzt werden können.
Probleme könnten sich im Allgemeinen durch die Beschränktheit des STT ergeben. Dieses kann einen Großteil der bedingten Modifikationen nicht umsetzen. Es wäre somit sinnvoll, das STT nochmals zu überarbeiten. Die Umsetzung von bedingten Modifikationen ist allerdings sehr komplex. Da diese an den Elementen eines XML-Schemas angebracht werden, ist die Berechnung von Instanz-Pfaden notwendig. Diese beschreiben die Stellen in einem XML-Dokument, an denen ein, in einem XML-Schema definiertes Element oder Attribut, vorkommen kann.
Letztendlich konnte das Verfahren jedoch angewendet werden, um die Anforderungen der VCD-spezifischen Datenformate festzuhalten, und die entsprechenden Testsysteme zu erstellen. Auch wenn die beiden betrachteten Spezifikationen vor einem unterschiedlichen Hintergrund erstellt wurden, ergeben sich bei näherer Betrachtung viele gemeinsame Anforderungen, die auf einer abstrakteren Ebene bestehen. Dabei konnten einige dieser Anforderungen mit ähnlichen Modifikationen erfasst werden, was für die Anwendbarkeit des Verfahrens spricht. Die Unterstützung bedingter Modifikationen sollte jedoch erweitert werden. Für eine solche müssen Schemprof und das STT angepasst werden.

Literatur

[1] *Decision No 922/2009/EC of the European Parliament and of the Council of 16 September 2009 on interoperability solutions for European public administrations (ISA) OJ L 260, 03.10.2009.*

[2] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* IEEE, 1990.

[3] *ISO/IEC Guide 2:1996, Standardization and related activities General vocabulary*, 1996.

[4] *IEEE 1484.12.1-2002: Draft Standard for Learning Object Metadata.* Technischer Bericht, IEEE, 2002.

[5] *ISO/IEC 17000:2004 Conformity assessment - Vocabulary and general principles, first edition*, 2004.

[6] *ISO/IEC 19757-3: Information technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron*, 2006.

[7] Ansgar Mondorf, Daniel M. Schmidt und Maria Wimmer: *Ensuring Sustainable Operation in Complex Environment: the Peppol Project and its VCD System.* In: *The 5th Mediterranean Conference on Information Systems, MCIS 2010, Tel-Aviv-Yaffo Academic College, Tel Aviv, Israel, September 12-14, 2010.* AISeL, 2010. Paper 61.

[8] Asuncion, C. H.: *Pragmatic interoperability in the enterprise – A research agenda.* In: Boldyreff, C., S. Islam, M. Leonard und B. Thalheim (Herausgeber): *23rd International Conference on Advanced Information Systems Engineering Doctoral Consortium (CAiSE-DC), London, United Kingdom*, Band 731 der Reihe *CEUR Workshop Proceedings*, Seite 8. CEUR-WS, June 2011.

[9] Asuncion, Camlon und Marten van Sinderen: *Pragmatic Interoperability: A Systematic Review of Published Definitions.* In: Bernus, Peter, Guy Doumeingts und Mark Fox (Herausgeber): *Enterprise Architecture, Integration and Interoperability*, Band 326 der Reihe *IFIP Advances in Information and Communication Technology*, Seiten 164–175. Springer Boston, 2010.

[10] Becker, Jörg und Dieter Kahn: *Der Prozess im Fokus.* In: Becker, Jörg, Martin Kugeler und Michael Rosemann (Herausgeber): *Prozessmanagement*, Seiten 3–16. Springer Berlin Heidelberg, 2005.

[11] Bird, Linda, Andrew Goodchild und Terry Halpin: *Object role modelling and XML-Schema.* In: *Proceedings of the 19th international conference on Conceptual modeling*, ER'00, Seiten 309–322, Berlin, Heidelberg, 2000. Springer-Verlag.

[12] Booch, G., J. Rumbaugh und I. Jacobson: *The unified modeling language user guide*. The Addison-Wesley object technology series. Addison-Wesley, 1999.

[13] Brutti, A. und D. Reiser: *Virtual Company Dossier BIS*. Technischer Bericht, Adetef, InfoCamere, PEPPOL.AT, UPRC, UKL, 2012.

[14] Chandler, Daniel: *Semiotics for beginners by Daniel Chandler*, 2005.

[15] Chen, Peter Pin-Shan: *The entity-relationship model - toward a unified view of data*. ACM Trans. Database Syst., 1(1):9–36, März 1976.

[16] Coen, Claudio Sacerdoti, Paolo Marinelli und Fabio Vitali: *Schemapath, a minimal extension to xml schema for conditional constraints*. In: *Proceedings of the 13th international conference on World Wide Web*, WWW '04, Seiten 164–174, New York, NY, USA, 2004. ACM.

[17] Colin, Smythe: *IMS GLC Specification Development Note 7: UML Profile for Platform Independent Model Descriptions of Specifications for Data Models*. Technischer Bericht, IMS GLC, 2006.

[18] Colin, Smythe und Neilsen Boyd: *IMS Content Packaging XML Binding - Version 1.2 Public Draft Specification v2.0*. Technischer Bericht, IMS Global Learning Consortium, 2007.

[19] Colin, Smythe, Eric Shepherd, Lane Brewer und Steve Lay: *IMS Question & Test Interoperability: An Overview*. Technischer Bericht, IMS Global Learning Consortium, 2002.

[20] Cooper, Adam: *IMS Vocabulary Definition Exchange Information Model*, Februar 2004.

[21] Cooper, Adam: *IMS Vocabulary Definition Exchange XML Binding*, Februar 2004.

[22] Dagger, Declan, Alexander O'Connor, Seamus Lawless, Eddie Walsh und Vincent P. Wade: *Service-Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services*. IEEE Internet Computing, 11(3):28–35, Mai 2007.

[23] Dahn, Ingo: *Spezifikationen, Normen und Standards zur Unterstützung kollaborativen Lernens*. In: Haake, J., G. Schwabe und M. Wessner (Herausgeber): *CSCL-Kompendium 2.0: Lehr- und Handbuch zum computerunterstützten kooperativen Lernen*, Seiten 374–392. Oldenbourg, 2011.

[24] Dahn, Ingo und Sascha Zimmermann: *Application Profiles and Tailor-Made Conformance Test Systems*. In: Jakobs, Kai (Ed.) (RWTH Aachen University Germany) (Herausgeber): *International Journal of IT Standards and Standardization Research (IJITSR)*, Seiten 60–73, 2010.

[25] David, P.A. und S. Greenstein: *The Economics Of Compatibility Standards: An Introduction To Recent Research 1.* Economics of innovation and new technology, 1(1-2):3–41, 1990.

[26] Delima, Neil, Sandy Gao, Michael Glavassevich und Khaled Noaman: *XML Schema 1.1, Part 2: An introduction to XML Schema 1.1.* IBM developerWorks, 2009.

[27] Domínguez, Eladio, Jorge Lloret, Beatriz Pérez, Áurea Rodríguez, Ángel L. Rubio und María A. Zapata: *A survey of UML models to XML schemas transformations.* In: *Proceedings of the 8th international conference on Web information systems engineering*, WISE'07, Seiten 184–195, Berlin, Heidelberg, 2007. Springer-Verlag.

[28] Duval, Erik und Wayne Hodgins: *A LOM Research Agenda.* In: *WWW (Alternate Paper Tracks)*, 2003.

[29] Duval, Erik, Neil Smith und Marc Van Coillie: *Application Profiles for Learning.* In: *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, ICALT '06, Seiten 242–246, Washington, DC, USA, 2006. IEEE Computer Society.

[30] Duval, Erik und Katrien Verbert: *On the Role of Technical Standards for Learning Technologies.* IEEE Trans. Learn. Technol., 1(4):229–234, Oktober 2008.

[31] Europäische Kommission: *Mitteilung der Kommission an das europäische Parlament, den Rat, den europäischen Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen: Europäischer eGovernment-Aktionsplan 2011-2015*, Dezember 2010.

[32] European Commision: *Commission's Communication on Interoperability, Annex II, European Interoperability Framework*, 2010.

[33] Fabbrini, F., M. Fusani und G. Lami: *Basic Concepts of Software Certification.* In: *Proc. of 1st International Workshop on Software Certfication (CERTSOFT'06)*, Seiten 4–16. McMaster University, 2006.

[34] Fusani, Mario und Eda Marchetti: *Damages and Benefits of Certification: A perspective from an Independent Assessment Body.* ECEASST, 33, 2010.

[35] Glushko, Robert J. und Tim McGrath: *Document engineering for e-business.* In: *Proceedings of the 2002 ACM symposium on Document engineering*, DocEng '02, Seiten 42–48. ACM, 2002.

[36] Goh, Cheng Hian: *Representing and reasoning about semantic conflicts in heterogeneous information systems.* Doktorarbeit, 1997.

[37] Gruber, Thomas R.: *A translation approach to portable ontology specifications.* Knowl. Acquis., 5(2):199–220, Juni 1993.

[38] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann und S. Hübner: *Ontology-Based Integration of Information - A Survey of Existing Approaches.* Seiten 108–117, 2001.

[39] Haslhofer, Bernhard und Wolfgang Klas: *A survey of techniques for achieving metadata interoperability.* ACM Comput. Surv, 42(2):7:1–7:37, 2010.

[40] Hoefferer, P.: *Achieving business process model interoperability using metamodels and ontologies.* In: *Proceedings of the 15th European Conference on Information Systems (ECIS 2007)*, Seiten 1620–1631, 2007.

[41] Irish Presidency of the European Public Administration Network eGovernment Working Group: *European Public Administration Network eGovernment Working Group. Key Principles of an Interoperability Architecture.* Technischer Bericht, EU Institutions, 2004.

[42] Ivezic, Nenad, Oriol Bausà, Hyunbo Cho, Jacques Durand, Michael Dreher, Tim Fowler, Yildiray Kabak und Christine Legner: *Testing Framework for Global eBusiness Interoperability Test Beds (GITB) (CWA 16408:2012).* CEN - European Committee for Standardization.

[43] Jakobs, Kai: *ICT standards development - Finding the best platform.* In: Doumeingts, Guy und Jörg Morel Gérard Müller (Herausgeber): *Enterprise Interoperability: New Challenges and Approaches*, Springer eBooks collection: Engineering, Seiten 543–552. Springer-Verlag London Limited, 2007.

[44] Klarlund, Nils, Anders Møller und Michael I. Schwartzbach: *The DSD Schema Language.* Automated Software Engineering, 9(3):285–319, 2002.

[45] Klein, H.K. und RA Hirschheim: *A comparative framework of data modelling paradigms and approaches.* The Computer Journal, 30(1):8–15, 1987.

[46] Klischewski, Ralf: *Information Integration or Process Integration? How to Achieve Interoperability in Administration.* In: Traunmüller, Roland (Herausgeber): *Electronic Government*, Band 3183 der Reihe *Lecture Notes in Computer Science*, Seiten 57–65. Springer Berlin / Heidelberg, 2004.

[47] Lampathaki, Fenareti, Spiros Mouzakitis, George Gionis, Yannis Charalabidis und Dimitris Askounis: *Business to business interoperability: A current review of XML data integration standards.* Comput. Stand. Interfaces, 31(6):1045–1055, 2009.

[48] Lee, Dongwon und Wesley W. Chu: *Comparative analysis of six XML schema languages.* SIGMOD Rec, 29(3):76–87, 2000.

[49] Legner, C. und K. Wende: *Business Interoperability Framework*. Working Paper, Institut für Wirtschaftsinformatik, Universität St. Gallen, 2007.

[50] Len Gebase, Robert Snelick und Mark Skall: *Conformance Testing and Interoperability: A Case Study in Healthcare Data Exchange*. In: Hamid R. Arabnia und Hassan Reza (Herausgeber): *Proceedings of the 2008 International Conference on Software Engineering Research & Practice, SERP 2008, July 14-17, 2008, Las Vegas Nevada, USA, 2 Volumes*, Seiten 143–151. CSREA Press, 2008.

[51] Li, Qing, Rynson W. H. Lau, Timothy K. Shih und Frederick W. B. Li: *Technology supports for distributed and collaborative learning over the internet*. ACM Trans. Internet Technol., 8(2):10:1–10:24, Februar 2008.

[52] Mondorf, Ansgar, Daniel Reiser und Maria Wimmer: *Deliverable 2.4: Pre Award eProcurement Virtual Company Dossier – Pre Award eProcurement: Virtual Company Dossier - PEPPOL EIA (Enterprise Interoperability Architecture)*, Juni 2011.

[53] Mondorf, Ansgar und Maria Wimmer: *Interoperability in e-tendering: the case of the virtual company dossier*. In: ACM New York (Herausgeber): *Proceedings of the 2nd international conference on Theory and practice of electronic governance*, ICEGOV '08, Seiten 110–116, 2008.

[54] Mondorf, Ansgar und Maria A. Wimmer: *Deliverable 2.2: Specification of architecture and components enabling cross-border VCD*, 2010.

[55] Muñoz-Merino, Pedro J., Carlos Delgado Kloos und Jesús Fernández Naranjo: *Enabling interoperability for LMS educational services*. Comput. Stand. Interfaces, 31(2):484–498, Februar 2009.

[56] Nentwich, Christian, Licia Capra, Wolfgang Emmerich und Anthony Finkelstein: *xlinkit: a consistency checking and smart link generation service*. ACM Trans. Internet Technol., 2(2):151–185, Mai 2002.

[57] Nils Klarlund, Thomas Schwentick und Dan Suciu: *XML: model, schemas, types, logics, and queries*. In: *Logics for Emerging Applications of Databases*, Seiten 1–41. Springer, 2003.

[58] Noy, Natalya F. und Michel Klein: *Ontology Evolution: Not the Same as Schema Evolution*. Knowl. Inf. Syst., 6(4):428–440, Juli 2004.

[59] Obrst, Leo: *Ontologies for semantically interoperable systems*. In: *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, Seiten 366–369, New York, NY, USA, 2003. ACM.

[60] Pagano, Dennis und Anne Brüggemann-Klein: *Engineering Document Applications - From UML Models to XML Schemas*. Band 3 der Reihe *Balisage Series on Markup Technologies*, 2009.

[61] Petter Gottschalk: *Maturity levels for interoperability in digital government.* Government Information Quarterly, 26(1):75–81, 2009.

[62] Pokraev, S.V.: *Model-driven semantic integration of service-oriented applications.* 2009.

[63] Riley, K., S. Wilson, J. Bell, I. Dahn, N. Friesen, P. Hope, J. Merriman, B. Olivier, A. Roberts und Smythe Colin: *IMS Application Profile Guidelines Part 1 - Management Overview*, August 2005.

[64] Riley, Kevin, David Mills und Erik Unjhem: *IMS Common Cartridge Profile: Version 1.0 Final Specification*, Oktober 2008.

[65] Riley, Kevin, Scott Wilson, John Bell, Kerry Blinco, Lorna Campbell, Ingo Dahn, Norm Friesen, Dick Hill, Peter Hope, Jeff Merriman, Bill Olivier, Anthony Roberts und Colin Smythe: *IMS Application Profile Guidelines Part 2 - Technical Manual*, 2005.

[66] Roukolainen, T.: *Modeling Framework for Interoperability Management for Collaborative Computing Environments.* Licentiate Thesis, University of Helsinki, 2009.

[67] Routledge, Nicholas, Linda Bird und Andrew Goodchild: *UML and XML schema.* Aust. Comput. Sci. Commun., 24(2):157–166, Januar 2002.

[68] Salim, Flora Dilys, Rosanne Price, Shonali Krishnaswamy und Maria Indrawan: *UML Documentation Support for XML Schema.* In: *Proceedings of the 2004 Australian Software Engineering Conference*, ASWEC '04, Seiten 211–220, Washington, DC, USA, 2004. IEEE Computer Society.

[69] Scholl, Hans, Herbert Kubicek und Ralf Cimander: *Interoperability, Enterprise Architectures, and IT Governance in Government.* In: Janssen, Marijn, Hans Scholl, Maria Wimmer und Yao-hua Tan (Herausgeber): *Electronic Government*, Band 6846 der Reihe *Lecture Notes in Computer Science*, Seiten 345–354. Springer Berlin / Heidelberg, 2011.

[70] Scholl, Hans J. und Ralf Klischewski: *E-Government Integration and Interoperability: Framing the Research Agenda.* International Journal of Public Administration, 30(8):889–920, 2007.

[71] Schroth, Christoph, G. Pemptroad und Till Janner: *CCTS-based Business Information Modelling for Increasing Cross-Organizational Interoperability.* In: *IESA*, Seiten 467–478. Springer, 2007.

[72] Simsion, G. und G. Witt: *Data modeling essentials.* Morgan Kaufmann, 2004.

[73] Smythe, Colin, Nielsen Boyd, Wilbert Krann, Jan Poston Day und Nigel Ward: *IMS Content Packaging Information Model - Version 1.2 Public Draft v2.0.* Technischer Bericht, IMS Global Learning Consortium, 2007.

[74] Söderström, Eva: *Formulating a General Standards Life Cycle.* In: Persson, Anne und Janis Stirna (Herausgeber): *Advanced Information Systems Engineering*, Band 3084 der Reihe *Lecture Notes in Computer Science*, Seiten 37–49. Springer Berlin / Heidelberg, 2004.

[75] Söderström, Eva: *Connecting B2B standards life cycles with stakeholders.* Interoperability of Enterprise Software and Applications: Workshops of the INTEROP-ESA International Conference, s. 223-234, 2005.

[76] Stegwee, Robert A. und Boriana D. Rukanova: *Identification of different types of standards for domain-specific interoperability.* In: *Proceedings of MIS Quarterly Special Issue on Standard Making: A Critical Research Frontier for Information Systems: Pre-Conference Workshop ICIS 2003*, Seiten 161–170, 2003.

[77] Völkel, Max: *Personal Knowledge Models with Semantic Technologies.*

[78] Warboys, Brian C., Peter Kawelek, Ian Robertson und Robert Greenwood: *Business Information Systems: A Process Approach.* Information Systems Series. McGraw-Hill, 1999.

[79] Wimmer, Maria A., Michael Liehmann und Bernd Martin: *Offene Standards und abgestimmte Spezifikationen - das österreichische Interoperabilitätskonzept.* In: *Multikonferenz Wirtschaftsinformatik*, Seiten 11–21.