XML-basierte Sichtweise auf die einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen

Aflwatt Haidalla

der Fakultät für Elektrotechnik und Informationstechnik
der Universität der Bundeswehr München
zur Erlangung des akademischen Grades eines

Doktor-Ingenieur
(Dr.-Ing.)

vorgelegte Dissertation

UNIVERSITÄT DER BUNDESWEHR MÜNCHEN
Fakultät für Elektrotechnik und Informationstechnik

XML-basierte Sichtweise auf die einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen

Aflwatt Haidalla

Vorsitzender des Promotionsausschusses:	Prof. Dr.-Ing. Ignaz Eisele
1. Berichterstatter:	Prof. Dr. Dr. Stefan. Schäffler
2. Berichterstatter:	Priv.-Doz. Dr.rer.nat. Dr.-Ing. David Meintrup

Tag der Prüfung: 12.12.2005

Mit der Promotion erlangter akademischer Grad:
Doktor-Ingenieur
(Dr.-Ing.)

Neubiberg, den 10.10.2005

Danksagung

Meinem Doktorvater, Prof. Dr. Dr. Stephan Schäffler, bin ich zu besonderem Dank verpflichtet.

Des weiteren bedanke ich mich bei Herrn Priv.-Doz. Dr.rer.nat. Dr.-Ing. David Meintrup für viele hilfreiche Diskussionen und Anregungen und für die immer nette und freundliche Unterstützung.

Ich danke auch Dr.-Ing. F.D. Lange ebenso für seine stete freundliche Unterstützung und besonderes für seine kritischen Anmerkungen.

Mein Dank gilt Katharina Lange, die mich von Anfang an in meiner Absicht unterstützt hat, diese Arbeit zu beginnen.

Ich danke meiner Familie, die mir stets den Rücken freigehalten hat.

Letztlich bedanke ich mich bei allen, die mich bei dieser Arbeit unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

Aufgrund der zunehmenden Bedeutung der ambienten Beleuchtungssysteme in fast allen Bereichen des täglichen Lebens und vor allem in der Automobilindustrie, ist die Zulassung dieses interessanten Dissertationsthemas "XML-basierte Sichtweise auf die einzelnen Phasen des PLZ von ambienten Beleuchtungssystemen" zu Stande gekommen, da es bis zum heutigen Zeitpunkt weltweit keine Methodik gibt, die sich mit dem PLZ von ambienten Beleuchtungssystemen hinsichtlich des Projektmanagements und gleichzeitig hinsichtlich der XML-basierten Datenstrukturen beschäftigt.

In dieser Arbeit wird eine speziell auf die effiziente Nutzung von XML bei dem Produktlebenszyklus von ambienten Beleuchtungssystemen ausgerichtete Methodik zur Definition und Kombination der leistungsfähigen XML-Dokumente vorgestellt.

Wie für das wissenschaftliche Rechnen typisch, sind für die Entwicklung dieser Methodik sowohl fundierte Kenntnisse der zugrunde liegenden Informatik, Projektmanagement als auch der Physik notwendig. Mit diesen Kenntnissen ist eine interdisziplinäre Synthese der verschiedenen Ansätze möglich. Im Idealfall summieren sich dabei die Vorzüge der Verfahren, und einzelne Nachteile werden deutlich verringert oder gar eliminiert. Für die erfolgreiche Anwendung der entwickelten Verfahren ist es weiterhin von Nutzen, spezifisches Wissen über das zu lösende Problem einfließen zu lassen.

Eine spezielle Einsatzmöglichkeit von XML bei dem Produktlebenszyklus von ambienten Beleuchtungssystemen, wird im Rahmen dieser Arbeit genauer untersucht. Allerdings geht es hier nicht um ihre Parallelisierungseigenschaften, sondern um die effiziente Nutzung der hierarchisch strukturierten XML-Dokumentenstrukturen moderner lichttechnischen Daten und aller anderen Daten, die entlang des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen.

Für die verteilte Produktentwicklung wird das Internet wegen seiner interaktiven Recherchemächtigkeit in weichen Informationen und seiner Kommunikationsfähigkeiten allgemein als Medium akzeptiert. Ergänzend sind jedoch fachliche Dienste erforderlich, um zwischen den beteiligten Betrieben u.a.

die semantisch unterschiedlich definierten Produkt-, Prozess- und Ressourcendaten zu harmonisieren, die Berechnungs-, Koordinations- und Dokumentationsfunktionen zu vereinheitlichen.

Durch den Einsatz von XML bei dem Produktlebenszyklus von ambienten Beleuchtungssystemen hat man einen sehr viel leichteren Zugang, um verteilte Produktentwicklung von ambienten Beleuchtungssystemen zu schaffen.

Angesichts des wachsenden Zeit-, Kosten- und Qualitätswettbewerbs benötigen die Unternehmen verbesserte Kommunikations- und Koordinationsmöglichkeiten in der Entwicklung. Als Medium wird zunehmend das Internet genutzt. Dieses ermöglicht es kostengünstig, komfortabel und zeitunabhängig mit weltweit verteilten Zulieferern und Kunden zu kommunizieren und die Arbeitsteilung zwischen Tochtergesellschaften zu koordinieren. Allerdings bietet das Internet nur eine Basistechnik, dessen Dienste (u.a. E-Mail, Dateitransfer) reichen für Konstruktionsprozesse nicht aus. Daher sind fachliche Dienste zu ergänzen, um die Aufgaben zu lösen, die zwischen den Betrieben daraus resultieren, dass die Produkt-, Prozess- und Ressourcendaten semantisch unterschiedlich definiert sowie in heterogenen DV-Formaten gehalten werden, die Berechnungs-, Koordinations- und Dokumentationsfunktionen verschieden praktiziert werden.

Die heterogenen Daten- und Funktionsstrukturen resultieren aus Unterschieden bei den Problemsichten der beteiligten Disziplinen (Informatiker, Maschinenbauer etc.), den Entwicklungskulturen (z.B. Entwurfsstile und –Abläufe, Projektmanagement) und den Informationssystemen (IS).

Eine DV-technische Standardisierung von Datenstrukturen muss daher durch eine fachliche Normierung ergänzt werden. Daher ist die Notwendigkeit des XML-Konverters von ambienten Beleuchtungssystemen.

Die intelligente Datenbereitstellung, - aufbereitung, weitergabe entlang des gesamten Produktlebenszyklus von ambienten Beleuchtungssystemen in einer gültigen XML-basierten Datenstruktur, die Organisation und Verwaltung der immensen Datenmengen, die in jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen, die Protokolle des Datenaustausches zwischen den Phasen des Produktlebenszyklus und für die verteilte Produktentwicklung, sowie die Automatisierung dieses Datenaustausches, sind der Kern dieser neuen Methodik.

Das Herzstück dieser Arbeit ist es, ein neues XML-Dateiformat für den Produktlebenszyklus von ambienten Beleuchtungssystemen zu entwickeln. Was man im Rahmen dieser Arbeit durch eine genaue Spezifikation von XML erreichen kann. Durch das neue XML-Dateiformat von ambienten Beleuchtungssystemen ist es unter anderen möglich, Daten zwischen den einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen reibungslos auszutauschen. Dadurch ist dieser Datenaustausch auch plattformunabhängig.
Es bleibt hier zu erwähnen, dass die rasante Entwicklung von XML und die Tatsache, dass XML eine ganz neue Wissenschaft ist, eine stete Herausforderung waren, die dieser Arbeit von Anfang an bis zum Ende begleitet hat.

2 Einsatzpotenziale von XML

2.1 XML

2.1.1 Historische Entwicklung

Seit dem ersten XML-Entwurf im November 1996 ist über kaum eine andere Sprache in den letzten Jahren so viel diskutiert worden, wie über die Extensible Markup Language XML. Sie wurde schon Anfang 1998 vom World Web Consortium (W3C) standardisiert, doch erst seit Anfang 1999 ist das Thema richtig interessant, seit der Internet Explorer 5 von Microsoft als erster Browser XML-Dateien darstellen kann. Mittlerweile haben viele andere Hersteller nachgezogen, und XML ist die Basis für viele neue Sprachen wie SVG und WML. Viele andere Anwendungsprogramme bieten mittlerweile auch die Möglichkeit, XML-Dateien einzulesen und zu speichern. Andererseits bezeichnet man mit XML auch eine unbestimmt große und stetig wachsende Gruppe von Konzepten und Sprachen, von denen XML nur die gemeinsame Grundlage ist (*Abb. 2.1*).

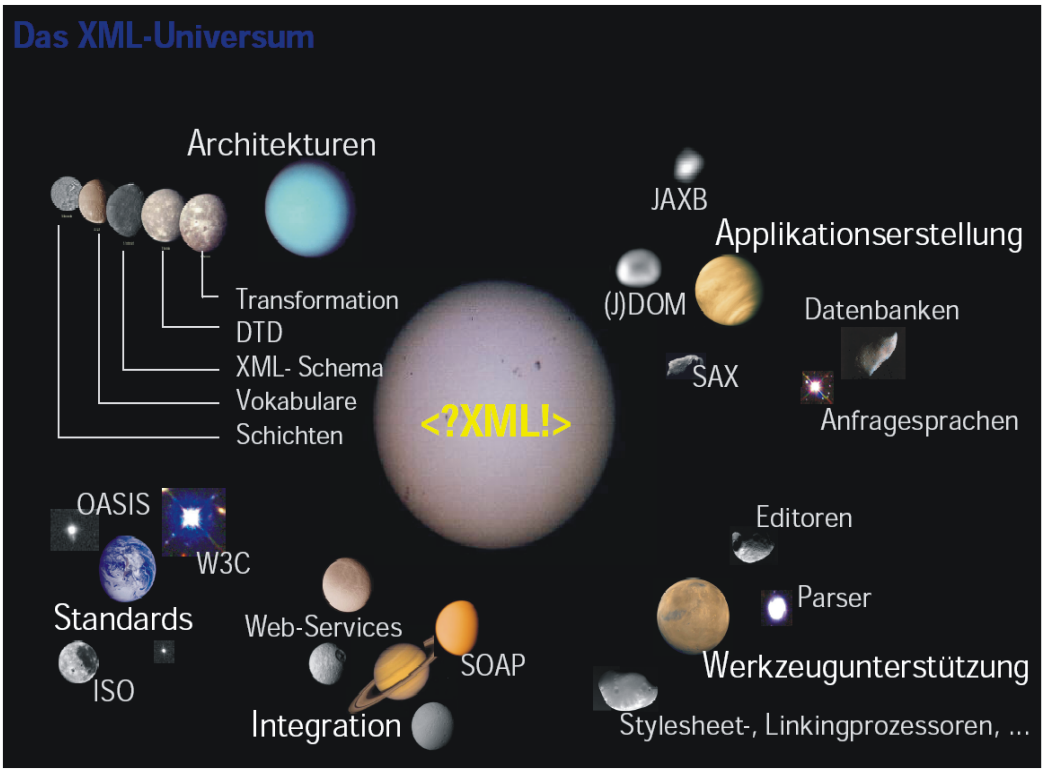

Abb. 2.1: XML-Universum

XML ist Grundlage all dieser Sprachen und Konzepte (und noch einiger anderer), entweder weil es sich bei ihnen um direkte Anwendungen von XML handelt oder weil sie XML konzeptionell voraussetzen und funktional ergänzen.
Die Unterscheidung von Co-Standards und XML-Applikationen ist etwas willkürlich. Co-Standards sind überwiegend Sprachen, die ihrerseits zur Definition anderer Sprachen verwendet werden (können), während man sich unter XML-Applikationen Sprachen vorzustellen hat, mit denen man keine Sprachen definiert, sondern (ähnlich wie mit HTML) Dokumente auszeichnet.
XML steht also nicht für XML, d.h. seit dem ersten XML-Entwurf steht XML für ein expandierendes Universum von Konzepten und Sprachen (*Abb. 2.1*), die nicht nur das Internet, sondern weite Bereiche der Informationstechnologie auf eine neue Grundlage stellen. Deswegen lässt sich das XML-Universum trotz des kleinen XML-Alters nicht mehr in einem einzigen Buch darstellen – jedenfalls nicht mit Anspruch auf Verständlichkeit und Praxisnähe. Zudem sind nicht alle Gebiete des XML-Universums für alle Reisenden und künftigen Bewohner gleichermaßen relevant.

2.1.2 Was ist XML ?

XML(eXtensible Markup Language, zu Deutsch etwa erweiterbare Auszeichnungssprache) ist eine erweiterbare, plattformübergreifende und programmiersprachenunabhängige Metasprache zur Beschreibung von Auszeichnungssprachen. Sie ermöglicht die Strukturierung von Daten in Dokumenten, sodass diese Daten mit unterschiedlichen Anwendungen über das Internet ausgetauscht werden können.
XML ist also ein Satz von Regeln zur Definition von semantischen Tags, die ein Dokument in Teile zerlegen und die verschiedenen Teile des Dokuments beschreiben. Es ist eine Meta-Auszeichnungssprache, die eine Syntax definiert, mit der andere domänenspezifische, semantische, strukturierte Auszeichnungssprachen definiert werden können.
XML ist eine Meta- Auszeichnungssprache, in der man die benötigten Tags nach Bedarf definieren kann. Diese Tags müssen gewisse allgemeine Regeln erfüllen,

aber diese bieten recht flexible Anpassungsmöglichkeiten. Wenn man beispielsweise an einer Genealogie arbeitet und Personen, Geburtstage, Familien, Hochzeiten, Scheidungen usw. beschreiben muss, kann man für jedes dieser Objekte Tags erstellen. Man braucht also seine Daten nicht in Absätze, Listeneinträge, bestimmte Auszeichnungen oder andere sehr allgemeine Kategorien zu zwängen.

Die Tags, die man erstellt, können in einer Dokumententyp-Definition (DTD) dokumentiert werden.

XML definiert eine Metasyntax der domänenspezifische Auszeichnungssprachen wie MathML und CML (*Abb. 2.2*) folgen müssen. Falls eine Anwendung diese Metasyntax versteht, versteht sie automatisch alle Sprachen, die von dieser Metasprache abgeleitet sind. Ein Browser braucht nicht von Vorneherein alle einzelnen Tags zu verstehen, die von Tausenden verschiedener Auszeichnungssprachen verwendet werden könnten. Stattdessen lernt der die Tags, die von einem vorliegenden Dokument verwendet werden, beim Lesen des Dokuments oder seiner DTD kennen. Die detaillierten Anweisungen, wie der Inhalt dieser Tags angezeigt werden soll, werden in einem separaten Stylesheet übermittelt, das dem Dokument beigefügt ist.

Betrachtet man beispielsweise Schrödingers Gleichung:

$$i\hbar\frac{\partial\psi(r,t)}{\partial t} = -\frac{\hbar^2}{2\cdot m}\frac{\psi^2(r,t)}{\partial x^2} + V(r)\psi(r,t)$$

Wissenschaftliche Fachaufsätze enthalten viele Gleichungen dieser Art, aber Wissenschaftler mussten jahrelang auf die Unterstützung der einfachsten mathematischen Formeln durch die Browser-Anbieter warten. Ambiente Beleuchtungssysteme befinden sich in einer ähnlich misslichen Lage, da weder der Netscape Navigator noch der Internet Explorer die notwendigen Datenformate und Datenstrukturen unterstützen.

XML bedeutet, dass man nicht darauf warten muss, dass die Browser-Anbieter endlich das in die Browser einbauen, was man für seine Arbeit benötigt. Man kann die benötigten Tags bei Bedarf erfinden und dem Browser mitteilen, wie diese Tags angezeigt werden sollen.

Anders definiert ist XML eine Methode, um strukturierte Daten in einer Textdatei darzustellen. Unter "strukturierten Daten" sind lichttechnische Daten, Kalkulationstabellen, Adressbücher, Konfigurationsparameter, finanzielle Transaktionen, technische Zeichnungen usw. zu verstehen. Programme, die solche Daten erzeugen, speichern sie oft auch auf der Festplatte ab, wofür sie entweder ein binäres oder ein Textformat benutzen können. Letzteres erlaubt es Ihnen, im Bedarfsfall die Daten ohne das Programm, das sie generierte, anzusehen. XML ist eine Menge von Regeln, Richtlinien, Konventionen - oder wie man dies auch immer bezeichnen will - für die Erstellung von Textformaten für solche Daten. Es entstehen so Dateien, die leicht zu generieren und (von einem Computer) zu lesen sind, die eindeutig sind und die üblichen Schwierigkeiten wie unzureichende Erweiterbarkeit, fehlende Unterstützung für Internationalisierung / Lokalisierung und Plattformabhängigkeit vermeiden.

XML ist für große und komplexe Dokumente ideal geeignet, weil die Daten strukturiert sind. In XML kann man nicht nur ein Vokabular festlegen, das die Elemente in dem Dokument definiert, sondern Sie können auch die Beziehungen zwischen Elementen definieren. XML stellt auch einen client-seitigen Include-Mechanismus bereit, der Daten aus mehreren Quellen zusammenzieht und wie ein einziges Dokument anzeigt. Die Daten können bei diesem Prozessor sogar umgeordnet werden. Teile der Daten können in Abhängigkeit von den Aktionen des Benutzers angezeigt oder verborgen werden. Dies ist außergewöhnlich nützlich, wenn man mit großen Informationenspeichern arbeitet, wie die Daten, die in jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen.

XML bietet einen Kompromiss zwischen Einfachheit und Flexibilität und ist eine leicht verständliche Metasprache, die nach Belieben erweitert werden kann. Es liegt also kein festgelegter Satz von Markup-Befehlen vor. Nach dem Datenaustausch (zum Beispiel im Netz oder zwischen den Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen) kann man das plattformunabhängige XML in alle gängigen Ausgabe-Formate transformieren. Man muss also seine Daten nicht in mehreren Formaten speichern, da man sie aus XML konstruieren kann. Man erreicht eine Trennung der Daten und der

Präsentation. Die Daten werden in XML und die Präsentationsinformationen in einem XML-Transformation-Dokument gespeichert.

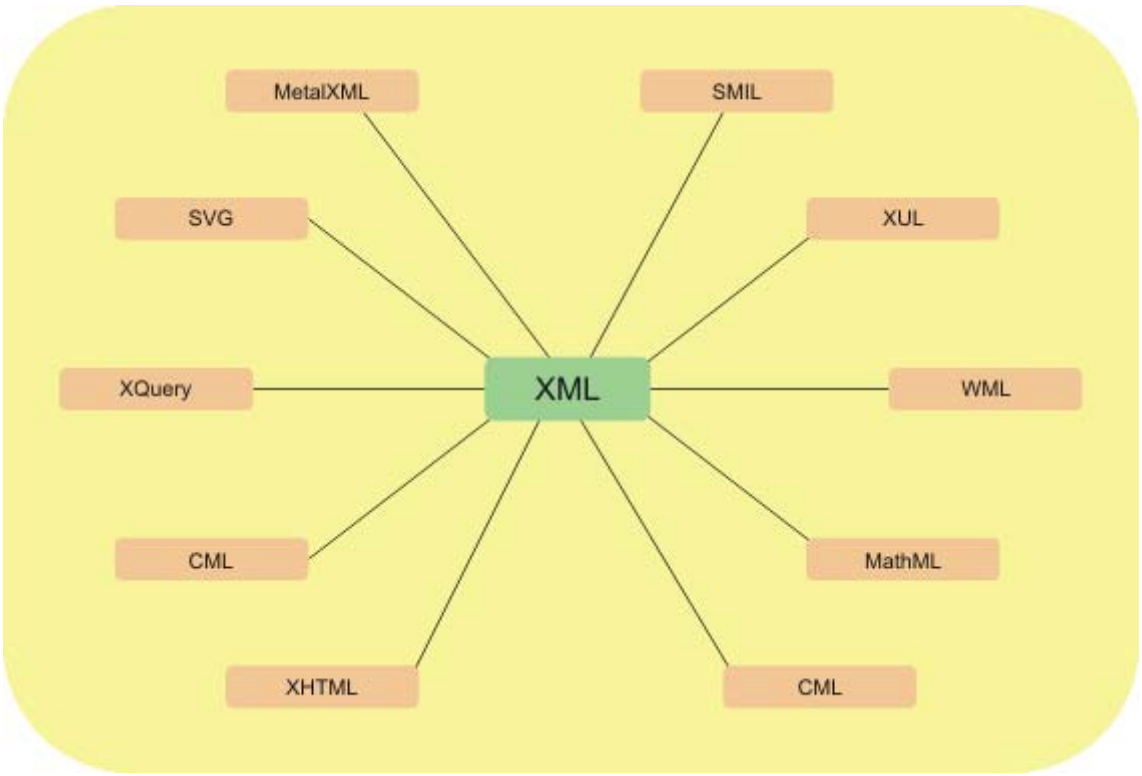

Abb. 2.2: XML- und einige XML-basierten Sprachen

XML-Dokumente müssen im Geensatz zu weniger restriktiven HTML-Dokumenten immer wohlgeformt sein, d.h. dass sie sich exakt an die syntaktischen Regeln des Standards halten. Zusätzlich sollten XML-Dokumente gültig sein und den Regeln einer DTD oder XML-Schema folgen. Dokumenten-Typ-Definitionen und XMLSchema sind formale Spezifikationen aller in einem Dokumenttyp erlaubten Strukturen. Man spricht auch von einer formalen Grammatik, in der zulässige Elementtypen, Attribute und deren Verschachtelung definiert sind. DTD und XML-Schema stellen somit sicher, dass verarbeitende Anwendungen XML-Dokumente entsprechend der zu Grunde liegenden Grammatik interpretieren und verstehen können.

2.1.3 Grundlagen

In diesem Unterkapitel werden die wichtigsten und notwendigen XML-Grundlagen und Sprachkonstrukte im Einzelnen vorgestellt.
Diese Einführung ist wichtig, um sich grob mit dem Thema XML zu befassen und um das Prinzip des XML-Konverters von ambienten Beleuchtungssystemen später besser verstehen zu können.

1. XML-Dokumente

Das folgende Beispiel zeigt ein typisches XML Dokument:

```
<?xml version="1.0" ?>
<document class="H.3.3">
    <author>David Meintrup</author>
    <title>XML Retrieval</title>
    <chapter>
        <heading>Introduction</heading>
        This text explains all about XML and IR.
    </chapter>
    <chapter>
        <heading>Extensible Style
        Language</heading>
      <section>
        <heading>Examples</heading>
      </section>
      <section>
          <heading>Syntax</heading>
      </section>
    </chapter>
</document>
```

Im Gegensatz zu HTML wird in XML zwischen Groß- und Kleinschreibung unterschieden. Die erste Zeile gibt an, um welche XML-Version es sich bei den vorliegenden Dokument handelt.

```
<?xml version="1.0" ?>
```

Bislang gibt es nur die Version "1.0". Nach dieser sogenannten Processing Instruction beginnt der eigentliche Inhalt des XML-Dokumentes, der aus einer Reihe von Markup Befehlen und den Daten besteht, die in einer baumartigen Struktur angeordnet werden.

Die eigentlichen Daten stehen zwischen dem Markup, z.B.

```
<heading>Syntax</heading>.
```

XML ist in Bezug auf die Syntax strenger als HTML, da jedes geöffnete Tag wieder geschlossen werden muss und keine überlappenden Tags (z.B. `<a><b></a></b>`) erlaubt sind.

2. XSL(T)

wenn von XML die Rede ist , fällt auch fast immer der Begriff XSLT. Hierbei handelt es sich um eine weitere Sprache, die die graphische Darstellung der XML-Inhalte vornimmt. XSLT steht für *Extensible Stylesheet Language Transformation* und nimmt die Transformierung der XML-Inhalte in eine andere beliebige Darstellungssprache vor: WML, PDF, ASCII, HTML usw.. XSLT ist mittlerweile auch ein Standard das W3-Konsortiums und wurde im November 1999 spezifiziert.
Um ein XML-Dokument darzustellen braucht es zwei Dinge:

- Eine Vorschrift, die beschreibt, wie die Informationen im XML-Dokument darzustellen sind. Diese Vorschrift wird typischerweise mit Hilfe von XSL beziehungsweise XSLT verfasst. Die Vorschrift heißt im Fachjargon XSL Stylesheet. Stylesheet ist Englisch für "Druckformatvorlage", "Layout-Datei" oder "Stilvorlage".
- Ein Werkzeug, welches das XML-Dokument aufgrund der Vorschrift darstellen oder in ein anderes Dokument umwandeln kann. Moderne Web-Browser können XML-Dokumente direkt darstellen. Daneben gibt es diverse Werkzeuge, die ein XML-Dokument aufgrund einer XSL(T)-Vorschrift in ein anderes Format umwandeln können.

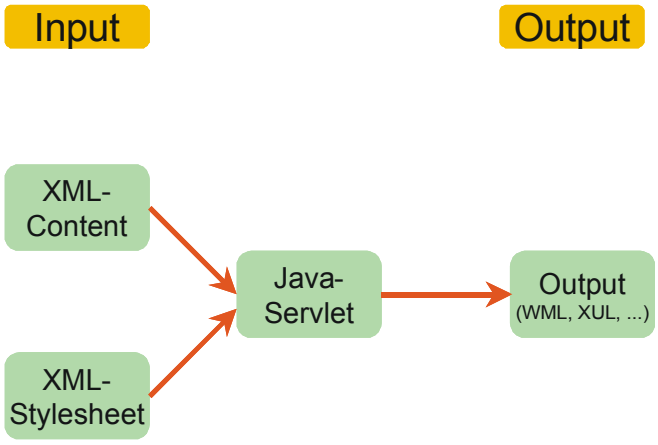

Abb. 2.3: XML und XSLT

3. Stylesheet und seines Aufbaus

Stylesheet ist eine Zusammenstellung von Formatierungsregeln, die angeben, wie Elemente im jeweiligen Zielmedium dargestellt werden sollen. Die Optionen des zunächst gebräuchlichen CSS-Standards werden durch XSL und XSLT wesentlich erweitert.

Im Folgenden soll der allgemeine Aufbau eines XSL-Stylesheets vorgestellt werden. Nach der XML-Deklaration folgt das so genannte Dokumentenelement mit der XSLT-Namensraumdeklaration. Dabei ist auch die Angabe der XSLT-Version erforderlich. Es folgen Top-Level-Elemente mit den eigentlichen Template-Regeln (template rules). Die Template-Regeln wiederum sind durch XSLT-Instruktionen und Ausgabebaumvorlage (result tree template) gekennzeichnet.

Ein XSLT-Stylesheet besteht aus einer Serie von Templates. Ein Template besteht aus einem Muster (Pattern), das XML-Elemente im XML-Quelldokument selektiert und einem Körper, der spezifiziert, was für eine Ausgabe generiert werden soll, wenn dieses Muster im Quelldokument angetroffen wird. Dabei parst der XSL-Prozessor die gesamte Struktur eines XML-Dokuments, vergleicht sie mit

den Mustern der Template-Regeln, ersetzt Inhalte und gibt anschließend das neu erzeugte Dokument aus. Der XSLT-Prozess beginnt im Wurzelknoten des Quelldokuments und durchläuft den Baum in Präfix-Reihenfolge und sucht dabei nach entsprechenden übereinstimmenden Produktionsregeln im Stylesheet. Danach wird der Körper abgearbeitet, wobei wieder andere Templates aufgerufen werden können.
Template-Regeln werden mit dem Element xsl:template definiert. Sie legen fest, welche Knotentypen des Ausgangsdokuments den Transformationsvorgang durchlaufen. Es folgt die Definition des Musters (Pattern) über “match“. Match gibt an, auf welche Elemente eine Template-Regel angewendet werden soll.

4. DTD

Die Abkürzung DTD steht für *Document Type Definition* – und dies ist nichts anders als ein Bestandteil von XML, in dem die konkreten Tags, die im XML-Code verwendet werden, definiert werden. Die DTD ist in der Regel ein Bestandteil des XML-Codes, kann aber auch in eine separate Datei ausgelagert werden.
Mit Hilfe einer DTD wird die Struktur von XML-Dokumenten verbindlich und zentral festgelegt. Alle XML-Dokumente desselben Typs können anhand derselben DTD bezüglich ihrer Struktur geprüft werden. Das ist hilfreich, weil viele XML-Editoren und andere XML-verarbeitende Tools eine solche Prüfung nach Wunsch automatisch vornehmen können. Auf diese Weise wird man als Autor oder Benutzer von XML-basierten Informationen sofort auf allfällige Strukturverletzungen aufmerksam gemacht.
Die automatisierte Prüfung ist insbesondere dann nützlich, wenn Informationen von anderen Anwendungen empfangen werden und vor der Weiterverarbeitung im Hinblick auf die strukturelle Gültigkeit geprüft werden sollen. Außerdem sind DTDs ein hilfreiches Werkzeug, wenn sich mehrere Organisationen über ein gemeinsam unterstütztes Format zum Datenaustausch einigen wollen. In solchen Fällen legt die DTD das gemeinsame Format eindeutig und formalisiert fest.
Trotz der erwähnten Vorteile ist die Benutzung von DTDs nicht obligatorisch. XML-Dokumente lassen sich auch ohne zugehörige DTD erstellen und verarbeiten.

DTDs kann man insofern als eine optionale aber nutzbringende Zusatztechnik betrachten. Übrigens gibt es unter dem Namen XML-Schema einen modernen Standard des W3C, der dieselbe Funktion wie die DTDs erfüllt.
Im Folgenden wird die Grundfunktionalität einer DTD anhand eines Beispiels demonstriert. Angenommen, man möchte XML-Dokumente für die Speicherung von Adressen verwenden.

```
<?xml version="1.0" ?>
<adresse>
     <name>Hans Mustermann</name>
     <strasse>Musterweg 15</strasse>
     <ort>München</ort>
</adresse>
```

Eine DTD für diesen Dokumenttyp wird wie folgt definiert:

```
<!DOCTYPE adressen [
<!ELEMENT adresse (name, strasse, ort)>
    <!ELEMENT name #PCDATA>
    <!ELEMENT strasse #PCDATA>
    <!ELEMENT ort #PCDATA>
]>
```

Hier wird festgelegt, dass das Wurzelelement `adresse` vom Dokumenttyp `adressen` die Kindelemente `name, strasse` und `ort` besitzt. Die DTD lässt sich direkt in das XML Dokument mit einbinden.

```
<?xml version="1.0" ?>
<!DOCTYPE adressen [
<!ELEMENT adresse (name, strasse, ort)>
     <!ELEMENT name #PCDATA>
     <!ELEMENT strasse #PCDATA>
     <!ELEMENT ort #PCDATA>
]>

<adresse>
    <name>Hans Mustermann</name>
    <strasse>Musterweg 15</strasse>
    <ort>München</ort>
</adresse>
```

Alternativ kann die DTD in einer externen Datei abgelegt und in das Dokument über einen Verweis eingebunden werden.

```
<?xml version="1.0" ?>
<!DOCTYPE adressen SYSTEM="adressen.dtd">

<adresse>
```

```
    <name>Hans Mustermann</name>
    <strasse> Musterweg 15</strasse>
    <ort>München</ort>
</adresse>
```

Dieses Beispiel sollte nur einen kleinen Einblick in die Erstellung von DTDs geben. Die Wohlgeformtheit von Dokumenten ist bezüglich RDF das entscheidende Kriterium.

5. Wohlgeformtheit (Validität)

Bislang ist nur die grobe syntaktische Struktur von XML Dokumenten thematisiert worden. Dazu gehört etwa die Unterscheidung zwischen Groß- und Kleinschreibung, die notwendige Baumstruktur der Tags usw.. Entspricht ein XML Dokument den im letzten Abschnitt erläuterten Bedingungen, so bezeichnet man dieses Dokument als wohlgeformt (well-formed).
Dieser Korrektheitsbegriff reicht in vielen Fällen jedoch nicht aus, da er keine Informationen über die erlaubten Tags und ihre Beziehung zueinander repräsentieren kann. Aus diesem Grund gibt es in XML die Möglichkeit, die komplette syntaktische Struktur von XML-Dokumenten in einer Document Type Definition (DTD) festzulegen. Für jedes XML-Dokument ist es dann entscheidbar, ob es einer bestimmten DTD entspricht oder nicht. Zum Beispiel ist in der DTD von XHTML genau festgelegt, welche Tags für die Erstellung eines XHTML-Dokumentes benutzt werden dürfen. Die Gültigkeit (Valid) eines XML-Dokumentes bezüglich einer DTD schließt die Wohlgeformtheit des Dokumentes ein. Die Umkehrung gilt im Allgemeinen nicht.

6. Trennung von Struktur, Inhalt und Format

Ein fundamentaler Vorteil von XML liegt in der rigorosen Trennung von Struktur, Inhalten und Format. Diese Trennung vereinfacht das effiziente und flexible Arbeiten mit Informationen. XML und die begleitenden Standards unterstützen

diese Trennung - teilweise erzwingen sie sie sogar. Die folgende Abbildung (*Abb. 2.4*) zeigt eine schematische Übersicht:

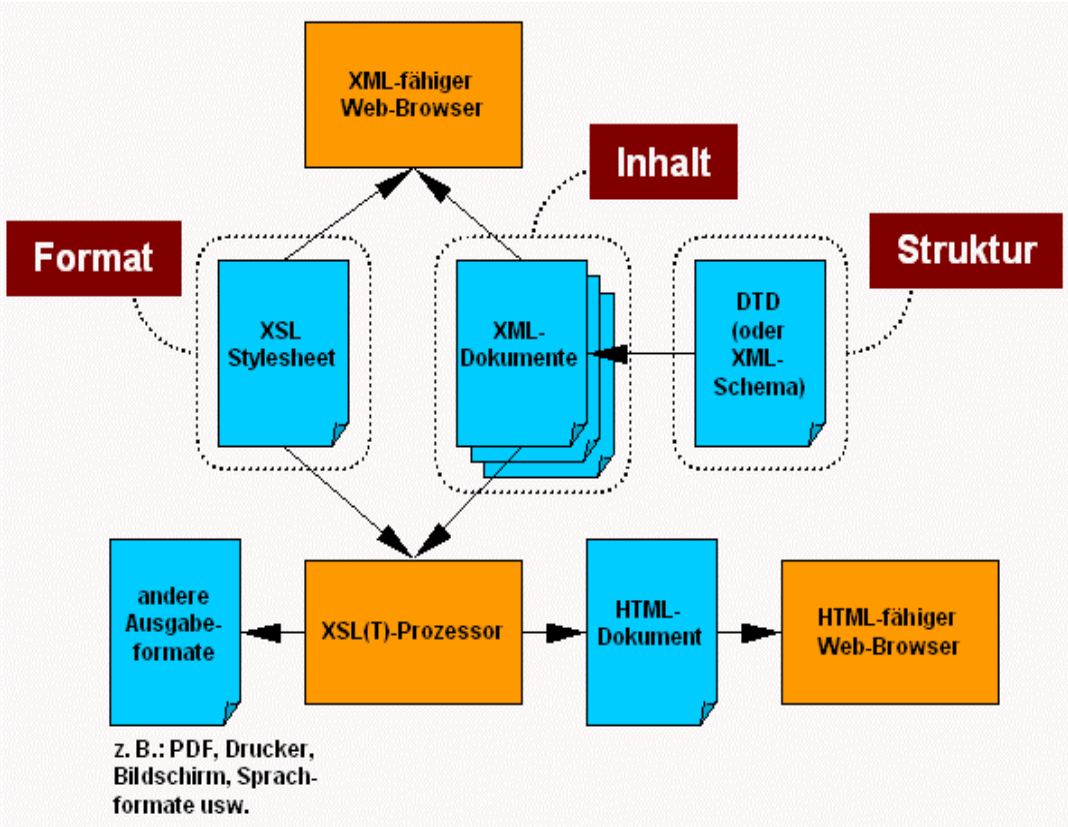

Abb. 2.4: XML-Trennung von Struktur, Inhalten und Format

Im Mittelpunkt stehen die DTD (oder ein XML-Schema), die XML-Dokumente sowie die XSL-Stylesheets. Die DTD gibt die Struktur der zu verarbeitenden Informationen vor. Die XML-Dokumente enthalten die eigentlichen Informationen. Die XSL-Stylesheets legen fest, wie die Informationen pro Ausgabeform dargestellt werden sollen.

7. Tags und Schemata

Tags: Ein von Spitzen Klammern umschlossener Elementname, der zur Auszeichnung eines Dokuments verwendet wird. In XML ist dabei die Wahl der Elementnamen prinzipiell frei, während HTML nur mit vorgegebenen Elementnamen arbeitet. Durch DTDs oder XML Schemas kann die Zahl der gültigen Tags eingeschränkt werden.
Die Struktur der Dokumente ist durch die Tags gegeben und entspricht einem Baum. Insbesondere muss es ein Wurzelelement geben. Der eigentliche Inhalt der Daten steht innerhalb der Tags. Ein leeres Tag <a> kann durch den Ausdruck <a/> abgekürzt werden. Jeder Tag muss geschlossen werden und es sind keine überlappenden Tags (z.B. <a>) erlaubt.

XML-Schema: Bei dem XML-Schema handelt es sich um die weiterentwicklung der DTD. XML-Schema bietet die Möglichkeiten, Tags genauer zu definieren, indem insbesondere der Inhalt eines Tags stärker berücksichtigt wird, als es bei der DTD der Fall ist. Es ist davon auszugehen, dass die DTD in Zukunft durch das XML-Schema ersetzt werden wird.

8. Attribute, Textknoten und Kommentare

Attribute:
Tags können Attribute besitzen. Diese spezifizieren meistens die Daten, die der Tag enthält, etwas genauer. Ein Attribut darf höchstens einmal auftreten, d.h. eine Konstruktion wie `<mitarbeiter name="hans" name="emil">` ist verboten. Ob es sinnvoller ist, Daten in Attribute oder in eigenständigen Tags zu speichern, hängt von der jeweiligen Anwendung ab und ist oft nicht einfach zu entscheiden.
Kommentare:
Kommentare `<!-- Hier kommt ein Kommentar -->` können wie normale Tags an beliebiger Stelle im XML Dokument eingefügt werden.

Textknoten:

Analog zu HTML kann man auch bei XML Markup und Text vermischen:

```
<chapter>
         <heading>Introduction</heading>
         This text explains all about XML and IR.
</chapter>
```

Diesen Text bezeichnet man als Textknoten.

9. CDATA

Unter Character Data (CDATA) lassen sich beliebige Zeichenketten abspeichern, die beim Parsen des XML-Dokumentes nicht weiter überprüft werden. Auf diese Weise lässt sich zum Beispiel ein nicht korrekter XML-Textabschnitt in einem XML-Dokument abspeichern:

```
<oldHTML >
<![CDATA[
          <html>
          <p>Introduction
          <br>This text explains <BR>
          all about XML and IR.
          <HTML>
</oldHTML >
```

10. Baumstruktur

Jedes XML Dokument lässt sich als Baum darstellen, dessen Knoten den oben vorgestellten Typen entsprechen.

```
<document class="H.3.3">
    <author>David Meintrup </author>
    <title>XML Retrieval</title>
    <chapter>
         <heading>Introduction</heading>
         This text explains all about XML and IR.
    </chapter>
</document>
```

Die Baumstruktur dieses XML Dokumentes sieht wie folgt aus:

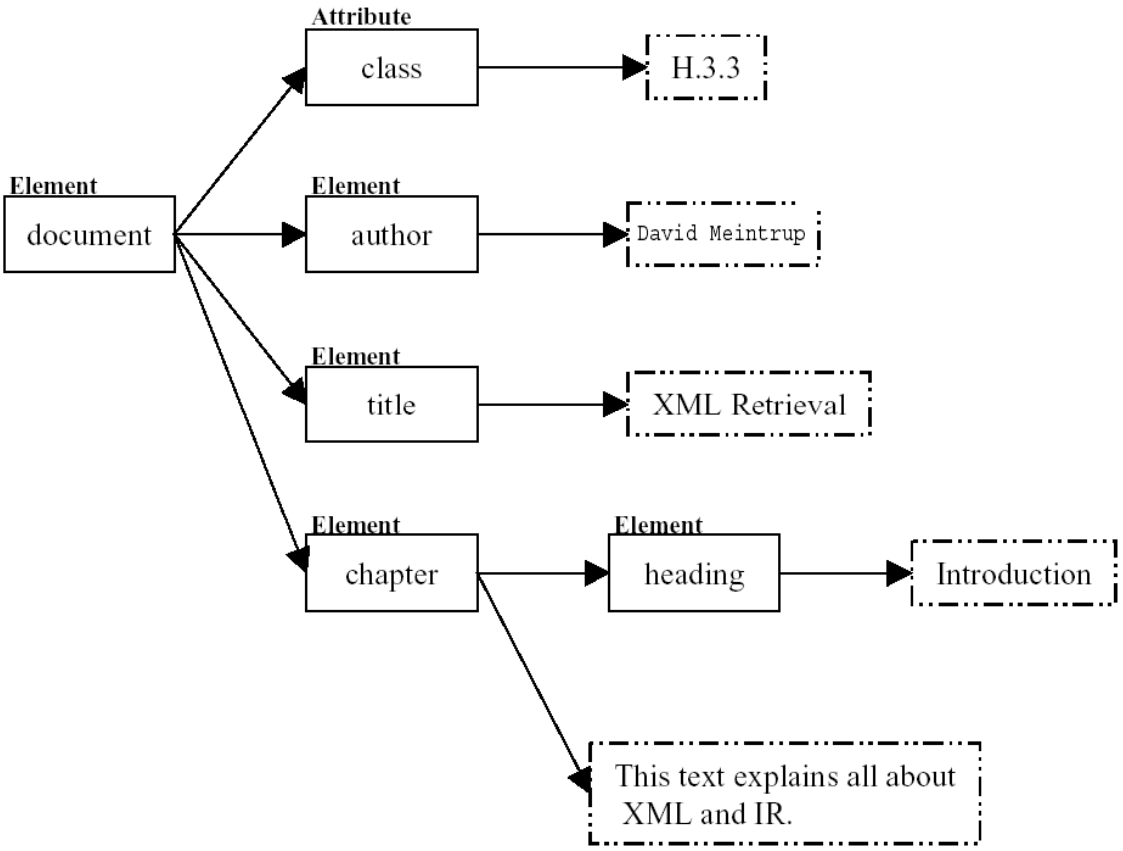

Abb. 2.5: Baumstruktur eines XML-Dokumentes

11. XML-Editor, XSLT-Editor

XML-Editor: Hierbei handelt es sich um eine Variante des herkömmlichen ASCII-Editors. Über XML-Editoren können XML-Dateien erstellt werden. Der Vorteil von XML-Editoren liegt darin, dass sie die Syntax der erstellten XML-Dateien prüfen.

XSLT-Editor: Hierbei handelt es sich wie bei den XML-Editoren um spezielle Editoren, die bei der Erstellung von XSLT-Dateien helfen. XSLT ist komplexer als XML, und hier bieten einige XSLT-Editoren eine graphische Unterstützung bei der Erstellung der XSLT-Dateien an.

12. Parser und Prozessor

Parser: Sobald von XML die Rede ist, ist auch von einem XML-Parser die Rede. Es handelt sich um ein separates Programm. Der Parser macht nichts anderes, als die Syntax einer XML-Datei auf deren Korrektheit zu prüfen. Dazu geht der Parser die Datei durch und prüft die erstellten XML-Tags. Bei Problemen meldet der Parser einen Fehler. Der Parser ist in den XML-Editoren integriert. Dies gilt auch für die Web-Browser. Microsoft verwendet zum Beispiel im Internet Explorer den eigenen msxml, während im Netscape-6-Browser der freie XML-Parser XP von James Clark enthalten ist.

Prozessor: der Begriff Prozessor taucht in Verbindung mit XSLT auf. Der XSLT-Prozessor ist ähnlich wie der XML-Parser ein kleines Programm, das zunächst die Syntax einer XSLT-Datei prüft und anschließend die konkrete Formung in eine der gewünschten Ausgabeformate (WML, PDF, HTML,...) übernimmt. Der XSLT-Prozessor übernimmt konkrete Arbeit für die Transformation. Der Microsoft XML-Parser msxml enthält neben dem XML-Parser auch einen XSLT-Prozessor. Der Netscape-6-Bowser dagegen verwendet einen separaten Prozessor, der ebenfalls frei verfügbar ist: Transformiix.
Und das Ganze zusammenzufassen: Ein XML-Dokument wird mit einem Editor erstellt. Der XML-Parser liest das Dokument ein und wandelt es in einem Baum von Elementen um. Der Parser übergibt den Baum an den Browser, der ihn anzeigt.

13. XML-Link

Die XML Linking Languge Xlink bietet eine Beschreibung an, wie Hyperlinks verschiedener Art innerhalb von XML verwendet werden können. Links in XML gehen über die aus HTML bekannten Hyperlinks weit hinaus und bieten eine viel differenziertere Steuerung.

14. Namespaces

Ein Problem mit Namensräumen ergibt sich immer dann, wenn in einem XML Dokument Tags aus verschiedenen Communities benutzt werden sollen. Zum Beispiel könnte man eine Dublin Core Metadatenbeschreibung sehr einfach als XML-Dokument repräsentieren:

```
<?xml version="1.0" ?>
<metadata>
    <creator>Bingo Mustermann</creator>
    <title>XML </title>
</metadata>
```

Um auszudrücken, wo diese Tags definiert wurden, kann man einen Namespaces angeben. Ein Namespace beschreibt eine Menge von Elementen und Attributen, Die einer URI zugeordnet sind.

```
<?xml version="1.0" ?>
<metadata xmlns:dc="http://ahrechner.org/dc/elements/">
    <dc:creator> Bingo Mustermann</dc:creator>
    <dc:title>XML </dc:title>
</metadata>
```

Nun lassen sich auch andere Tags für die Beschreibung von Metadaten nutzen. Zum Beispiel könnte man den Namen des Creators in Vor- und Nachname auflösen, indem man vCard Elemente in die Beschreibung aufnimmt.

```
<?xml version="1.0" ?>
<metadata xmlns:dc=" http://ahrechner.org/dc/elements/
          xmlns:vCard="http://imc.org/vCard/3.0#">
    <dc:creator>
         <vCard:Given> Bingo </vCard:Given>
         <vCard:Family>Mustermann </vCard:Family>
    </dc:creator>
    <dc:title>XML </dc:title>
</metadata>
```

Durch den Einsatz von Namespaces lassen sich Tags benutzen, die in verschiedenen Communities definiert wurden. Offensichtlich führt ein solcher Mix

von Tags im Allgemeinen nicht zu einem bezüglich einer DTD gültigen XML Dokument.

2.2 XML und Business-Intelligence-Systemen

Wesentlicher Gegenstand von Business-Intelligence-(BI-)Systemen sind die Datensammlung, -speicherung, -aufbereitung und -darstellung in einem entscheidungsunterstützenden Kontext. Inhaltliche Anforderungen richten sich zusätzlich auf die Flexibilität und Beherrschbarkeit der Modelle und Anwendungen. Wenn dies in verschiedenen Bereichen von Unternehmen und zusätzlich auch unternehmensübergreifend realisiert werden soll, stellen Komplexität und fehlende

Standards zur Interoperabilität Hemmnisse im Aufbau und dem Einsatz von Business-Intelligence-Systemen dar. Standradtechnologien ermöglichen die Erzielung von Netzeffekten und erleichtern die Einbindung heterogener Datenquellen und Anwenderwerkzeuge. Dies führt mittelfristig zu Komplexitätsreduktion und somit zu Kostenersparnis. Mit XML steht eine Standardsprache zur semantischen Auszeichnung zur Verfügung, mit deren Hilfe sich der Datenaustausch zwischen den unterschiedlichsten Anwendungen und proprietären Systemen organisieren lässt.

In diesem Unterkapitel wird der aktuelle Stand des Einsatzes von XML im Rahmen von Business-Intelligence-Systemen dargestellt.

Es werden im weiteren Verlauf dieses Kapitels eine kurze Darstellung der Grundlagen von Business-Intelligence-Systemen sowie die wichtigsten Anwendungsgebiete des XML-Einsatzes in BI-Systemen beschrieben. Das Unterkapitel 2.3 stellt weiterhin den Einsatz von XML-Core-Standards zum Datenaustausch sowie vier etablierte BI-spezifische Austauschprotokolle vor.

Jede Standardisierung kann ihre Potenziale nur ausschöpfen, wenn sie von einer ausreichend große Anzahl an verfügbaren und marktrelevanten Softwarewerkzeugen umgesetzt wird. Deshalb zeigt Unterkapitel 2.5 Die Ergebnisse einer Erhebung zum aktuellen und geplanten Stand der Umsetzung der beschriebenen Spezifikationen bei den umsatzstärksten Business-Intelligence Softwareanbietern weltweit.

Dieses Kapitel wird mit einer zusammenfassenden Betrachtung der Einflussdimensionen von XML auf BI-Systeme und einem Ausblick auf zukünftig zu erwartende Entwicklungen beendet.

Business-Intelligence (BI) beschreibt in einem umfassenden Verständnis die Integration, Speicherung, Aufbereitung und Darstellung entscheidungsrelevanter Information zur Planung, Kontrolle und Steuerung eines Unternehmens. Business-Intelligence umfasst zahlreiche Teilsysteme und –prozesse zur Umsetzung dieser Aufgaben und hat sich daher als Sammelbegriff für ein weites Spektrum unterschiedlicher Software-Werkzeuge, Informationen und Methoden zur Unternehmenssteuerung etabliert.

Technologisch gesehen umfasst Business-Intelligence alle durch Informationstechnologie unterstützten Aktivitäten der Integration, Speicherung, Aufbereitung und Darstellung von Informationen im Rahmen der betrieblichen Entscheidungsunterstützung. Sowohl Prozesse als auch Systemkomponenten von Business-Intelligence-Systemen können anhand eines Fünfschichten-Modells systematisiert werden. Die Abbildung *Abb. 2.6* soll die Einsatzmöglichkeiten von XML in BI-Systemen anhand des oben genannten Fünfschichten-Modells schematisch darstellen.

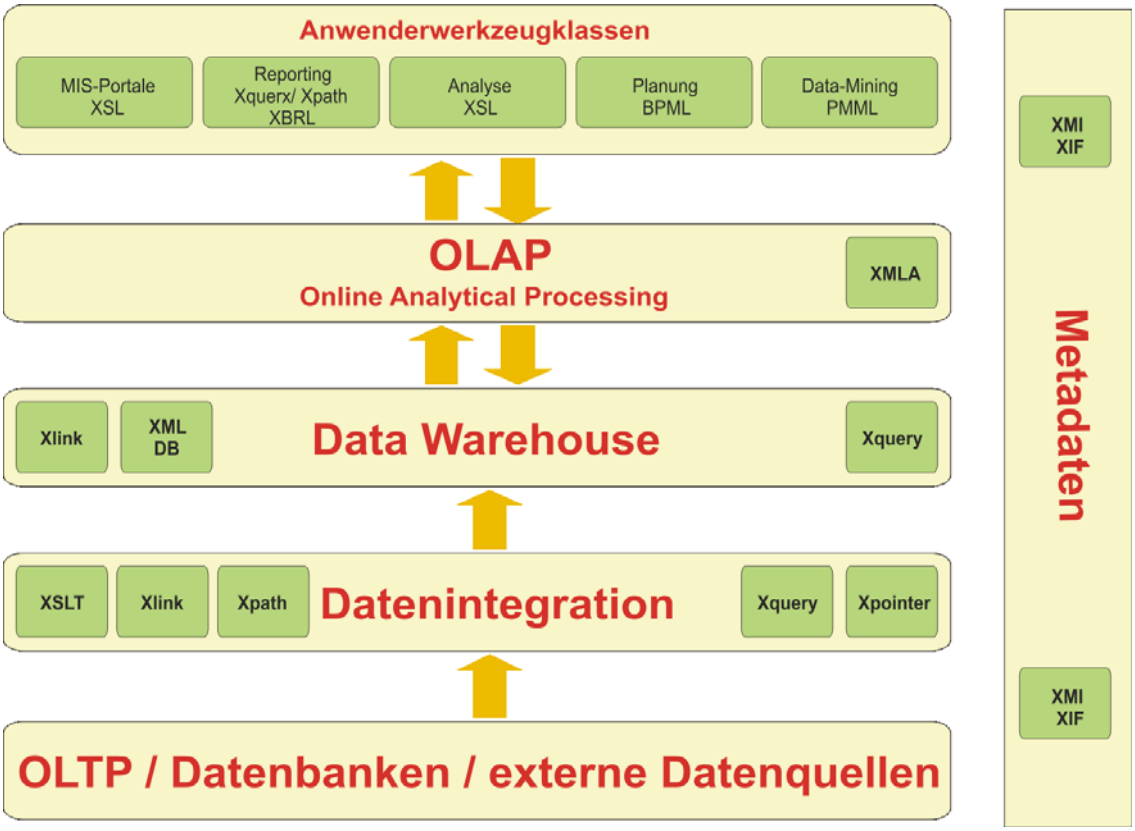

Abb. 2.6: Einsatzmöglichkeiten von XML in BI-Systemen

Die Datenintegration aus operativen Systemen, Datenspeicherung und –modellierung im Data-Warehouse oder Data-Marts sowie die analyseorientierte Aufbereitung von Daten im Sinne des Online Analytical Processing (OLAP) stellen in diesem Zusammenhang die Hauptprozesse und Ebenen dar. Darauf aufbauend stellen verschiedene Werkzeuge Funktionalität für Endanwender bereit. Wichtige Gruppen innerhalb der Anwenderwerkzeuge sind mit wachsender Komplexität und

wachsenden Freiheitsgraden für den Anwender: MIS-Portale, Reporting, Analyse, Planung und Data-Mining.

Wie schon im Unterkapitel 2.1 erwähnt, ermöglicht XML die Strukturierung von Daten in Dokumenten, sodass diese Daten mit unterschiedlichen Anwendungen über das Internet ausgetauscht werden können. Die notwendige Offenheit und Erweiterbarkeit der Metasprache führt zu der besonderen Herausforderung, auf Basis des technischen Standards auch semantische Standards zu definieren, die anwendungs- und herstellerübergreifende Akzeptanz gemeinsamer Inhaltsmodelle erreichen. Während andere Anwendungsbereiche, wie beispielsweise der überbetriebliche Austausch operativer Geschäftsdaten, in diesem Punkt schon relativ weit fortgeschritten sind(z.B. ebXML), steht der XML-Einsatz in Business-Intelligence-(BI-)Systemen erst am Anfang.

Die Untersuchung im Rahmen dieses Unterkapitels fokussiert sich auf die durch das World wide web Konsortiums (W3C) empfohlenen Core-Standards, sowie kommerziell verfügbare BI-spezifische Protokolle. Als Core-Standards werden diejenigen Spezifikationen bezeichnet, welche die Basis der durch das W3C empfohlenen Standards bilden und als wesentlich oder fundamental für den XML-Standard angesehen werden können. BI-spezifische Umsetzung gehen i. d. R. aus der Initiative eines oder mehrerer Anbieter hervor.

Eine Übersicht sowie eine kurze Beschreibung der untersuchten Standards zeigt die Abbildung (*Abb. 2.7*).

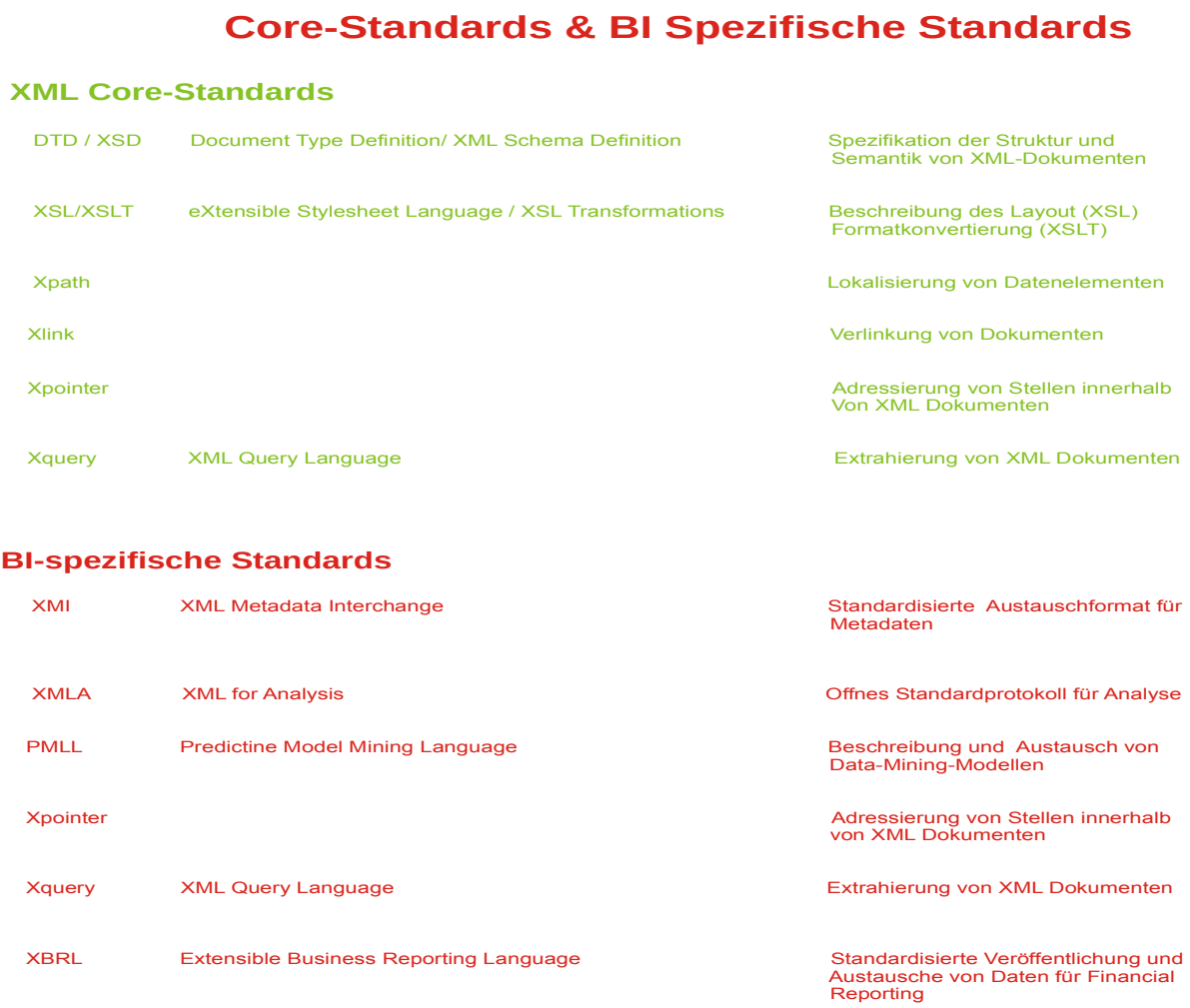

Abb. 2.7: Core-Standards und BI Spezifische Standards

2.3 XML-Einsatz und -Anwendungsgebiete in Business-Intelligence-Systemen

Das in der Abbildung *Abb. 2.6* vorgestellte Fünf-Schichten-Modell für Business-Intelligence-Systeme definiert den Rahmen für die Untersuchung des XML-Einsatzes. Zunächst sollen die Einsatzpotenziale von XML innerhalb BI-typischer Anwendungsgebiete vorgestellt werden. Innerhalb dieser Anwendungsgebiete dient XML als Auszeichnungssprache insbesondere zur Definition von Austauschprotokollen.
Unterkapitel 2.4.1 beschreit den Einsatz von Core-Standards sowie der vier ausformulierten und von Standardisierungsgremien wie der OMG akzeptierten BI-Spezifischen Austauschprotokolle XMLA, PMML, XMI und XBRL.

2.3.1 Datenintegration

Die Datenintegration fungiert als Bindeglied zwischen datenliefernden Vorsystemen, Datenspeicherung wie Data-Warehouses und Anwendungswerkzeugen zur Ausgabe und Weiterverarbeitung von Daten. In diesem kommen alle im Unterkapitel 2.4 vorgestellten Austauschprotokolle zum Einsatz. Die konkrete Auswahl und der Einsatz der Protokolle ist von der individuellen Systemlandschaft und den Aufgabe des Systems abhängig.

2.3.2 Datenmodellierung

Zu den wichtigsten Aufgaben in Rahmen der Konzeption von Business-Intelligence-Systemen zählt die multidimensionale Datenmodellierung. Dabei erschweren proprietäre Speicherformen der modellbeschreibenden Metadaten der Softwareanbieter insbesondere den Austausch von Modellinformationen zwischen verschiedenen Werkzeugen. An dieser Stelle setzt die Standardisierungsinitiative CWM (Common Warehouse Metamodel) der Object Management Group (OMG) an.

Während das im CWM beschrieben „Retional Model“ zur Beschreibung von relationalen Datenbankstrukturen eingesetzt wird, bietet das „OLAP Model“ Möglichkeiten zur Modellierung multidimensionaler Datenbanken. Der Austausch der Modelle erfolgt im standardisierten XML Metadata Interchange (XML-) Format. Der Bereich der XML-basierten multidimensionalen Modellierung steht erst am Anfang der Entwicklung. Zugehörige Forschungsvorhaben sind Xcube, MetaCube-X sowie CataCubeX.

2.3.3 Nutzdaten-Speicherung

Für die persistente Abgabe von Nutzdaten in XML-Dokumenten existieren unterschiedliche Architekturmodelle. Auf der einen Seite bieten die klassischen Datenbankhersteller Erweiterungen ihrer Datenbanken zur relationalen bzw. objektorientierten Speicherung und zur Reproduktion von XML-Dokumenten an (sog. XML-Enabled-Databases). Auf der anderen Seite versuchen sich neue Anbieter mit nativen XML-Datenbanken (z.B. die Software mit der Datenbank „Tamino“) und SQL-Gateways zu positionieren.
Zur Identifizierung der geeigneten Speicherform spielt die Dokumentstruktur bzw. der Dokumentcharakter eine entscheidende Rolle. Mit zunehmendem Strukturierungsgrad der Inhalte können dokumentzentrierte, semistrukturierte und datenzentrierte XML-Dokumente unterschieden werden. Je nach Anwendungszweck und Dokumentcharakter stehen unterschiedliche Optionen zur Verfügung:

a) Speicherung des XML-Dokumentes als Ganzes, d.h. als character large object (CLOB) in relationalen Datenbanken.
b) Speicherung der Graphenstruktur eines XML-Dokumentes durch generische Zerlegung oder auf Basis des Document Object Modell (DOM).
c) Speicherung auf Basis von DTD/XML-Schema oder durch Mapping der Dokumentinformationen auf entsprechende Datenbanktabellen.

Ursprüngliche XML-Datenbanken können Dokumente ohne vorherige Transformation speichern, was kurze Antwortzeiten und einen effizienten Umgang mit unstrukturierten Dokumenten ermöglicht.

Zusätzlich erfüllt die XML-Datenbank die Funktion eines Mediators, indem sie eine logische Sicht auf interne oder externe Datenquellen zur Verfügung stellt und die Anfragen ggf. durchreicht. Damit eignen sich XML-Datenbanken vor allem als Datenspeicher für Content-Management-Systeme oder Document-Management-Systeme in Ergänzung vorhandener relationaler Data-Warehouse-Systeme.

2.3.4 Metadatenverwaltung

Der entscheidende Vorteil des XML-Einsatzes im Bereich der Metadatenverwaltung ergibt sich aus der Definition der Metasprache. Ein beliebiges XML-Dokument repräsentiert immer auch Metadaten. Die semantische Auszeichnung bzw. die DTD/XSD eines Dokumentes liefert Informationen über die zugrunde liegenden Daten, die sich z.B. einfach in Retrieval-Prozesse einbinden lassen. Somit ist XML auch die Basis zukünftiger Entwicklungen, deren Hauptziel die weitgehend generische Repräsentation, Speicherung und Verarbeitung von internen und in zunehmendem Maße auch externen Metadaten ist. Dieses Synergiepotenzial nutzt die Standardisierungsinitiative CWM. Hier wird XML zwar nicht als Speicher, aber als standardisiertes Austauschformat für Metadaten verwendet, um die Interoperabilität zwischen Repositorien unterschiedlicher Hersteller zu gewährleisten.

2.3.5 Abfrage und Präsentation

Für Reportingaufgaben oder zur logischen Integration von XML-Dokumenten in ein BI-System werden entsprechende Anfragesprachen benötigt, die Dokumentteile oder Daten extrahieren können. Der Prozess der Entwicklung von XML-Anfragesprachen ist noch nicht abgeschlossen. Er wird geprägt von einer Vielzahl von Sprachvorschlägen (XML-QL oder XQL) und Forschungsprojekten (z.B. XMDQL(Multidimensional Query Language)) und SQL_{MD} zur Einbeziehung von XML-Daten in multidimensionalen Datenbankabfragen.

- *Xpath:* Als sog. pfadorientierte Lokatorsprache kann Xpath beliebige Dokumentteile innerhalb der Baumstruktur auffinden. Die Bezeichnung der Dokumentteile erfolgt durch den Location Path, der als Ergebnis eine Menge von Nodes liefert und somit eine bestimmte Stelle im XML-Dokument beschreibt.
- *Xquery:* Im Gegensatz zur Lokatorsprache orientiert sich die Anfragesprache Xquery an klassischen deskriptiven Sprachen. Die Ähnlichkeit zu SQL spiegelt sich z.B. in der bekannten Anfragekonzeption SELECT ...FROM...WHERE wieder. Xquery kann sowohl für XML-Dokumente als auch für XML-Datenbanken eingesetzt werden und liefert das Ergebnis der Anfrage wiederum im XML-Format zur weiteren Verarbeitung.
- *XSL/XSLT:* Ein besonderes Merkmal von XML ist die Trennung von Struktur, Inhalt und Layout, wodurch eine medienneutrale Bereitstellung von XML-Daten ermöglicht wird. Für den Einsatz in BI-Sytemen bedeutet dies im Wesentlichen eine vereinfachte Einbindung heterogener Präsentationsplattformen. Basierend auf einer einheitlichen Datenbasis können für jeden Präsentationszweck entsprechende Transformationen vorgenommen werden. Die Umsetzung erfolgt weitgehend mit Core-Standards. Reportingdaten können beispielsweise mit Xquery extrahiert und anschließend durch XSLT in das HTML-Format transformiert und somit für das Web-Reporting oder für eine Management-Portal-Anwendung bereitgestellt werden. Der Core-Standard XSL wird in einem zweiten Verarbeitungsprozess dazu verwendet, die transformierten Daten mittels der XSL Formatting Objects (XSL-FO) in ein spezifisches Ausgabeformat zu überführen. Dadurch kann der gleiche Inhalt nicht nur als HTML-Webseite angezeigt, sondern auch im PDF-Format zur druckreifen Aufbereitung oder als WAP-Seite zur Präsentation von Reports auf einem mobilen Endgerät genutzt werden.

2.3.6 Anwendungskonfiguration

Die Fähigkeit von XML, selbstbeschreibende Daten in Dokumenten strukturiert zu speichern, eröffnet eine fast unüberschaubare Zahl von Anwendungsfeldern. Ein Spezialbereich scheint auch für Business-Intelligence-Systeme immer breitere Anwendung zu finden: die Konfiguration der BI-Software. Dabei ist die Metasprache nicht nur dazu geignet, die heute verbreiteten Textdateien zur Konfiguration von Anwendungen zu ersetzen, so wie es Microsoft mit .NET schon weitgehend praktiziert. Vielmehr wird XML immer häufiger auch anwendungsintern als Kommunikationsprotokoll eingesetzt. Die Brandbreite der Einsatzmöglichkeiten reicht von der Definition des Aufbaus von Benutzeroberflächen über die Steuerung des Verhaltens von Anwendungen bis hin zur Speicherung von benutzerspezifischen Einstellungen.
Zusätzlich ist auch der Einsatz von XML zur Unterstützung von Workflowfunktionen denkbar. Für diesen Zweck kann die Sprache BPML (Business-Process Modelling Language) zum Einsatz kommen, um z.B. kollaborative Planungsprozesse zu steuern.

2.4 Austauschprotokolle

XML-Protokolle zum Datenaustausch zwischen den verschiedenen Komponenten in Business-Intelligence-Systemen können in Core-Standards und BI-spezifische Konventionen unterschieden werden. Letztere untergliedern sich wiederum in technische und inhaltliche Definitionen.
In der Abbildung *Abb.2.8* sind die verschiedenen XML-Protokolle zum Datenaustausch zwischen den verschiedenen Komponenten in Business-Intelligence-Systemen dargestellt.

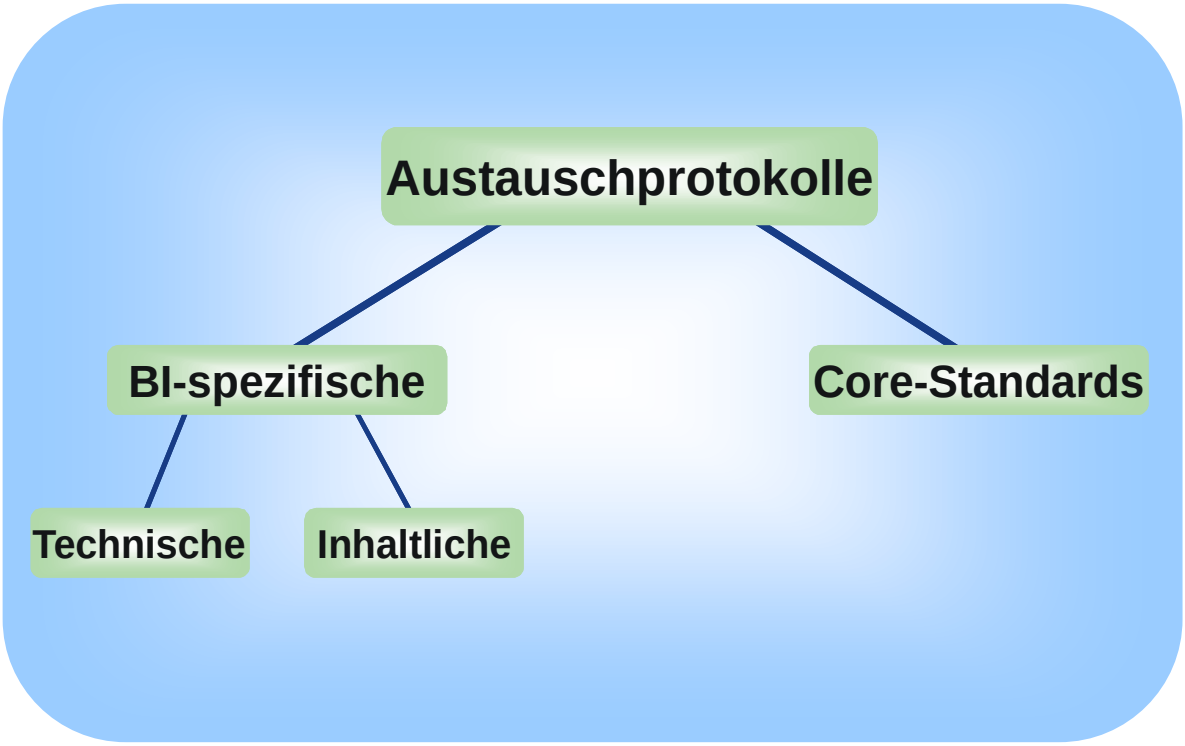

Abb. 2.8: Klassifikation der XML-Datenaustauschprotokolle im BI-Umfeld

2.4.1 Core-Standars

Im Gegensatz zu den Core-Standards, die im wesentlichen anwendungsneutral einsetzbar sind, sollen die spezifischen Austauschprotokolle die Interoperabilität

für BI-Systeme verbessern und den Zugriff auf analytische Daten über das Internet ermöglichen. Nahezu alle Hersteller von OLTP-Systemen, Datenbanken und Datenintegrationswerkzeuge bieten in ihren Produkten Möglichkeiten zur Übergabe und Verarbeitung von XML-Dokumenten. So ist es z.B. mit dem SAP Business Connector möglich, XML-Dokumente für beliebige Geschäftsbelege zu generieren und für den Import in ein Data-warehouse bereitzustellen. XML-Erweiterungen von relationalen Datenbanken können solche Dokumente dann direkt in das jeweilige Data-Warehouse laden. Spezialwerkzeuge zur Datenintegration gehen noch hierüber hinaus. Sie können XML-Dokumente verarbeiten und nach der Durchführung von Extraktions- und Transformationsprozessen auch wieder generieren. Dadurch ergeben sich fast beliebige Architekturvarianten für den Einsatz von XML-Dokumenten zwischen den beteiligten Systemkomponenten.

Die Core-Standards können für den Datenintegrationsprozesse besonders in den Bereichen Extraktion, Transformation und zur Verknüpfung von XML-Dokumenten eingesetzt werden:

- Einsatz eines XSLT-Prozessors zur regelbasierten Umwandlung von Quelldokumenten in ein gewünschtes Zielformat, d.h. die Selektion einzelner Bestandteile, deren Umordnung und die Ableitung neuer Inhalte.
- Verknüpfung von Datenquellen mittels Xlink. Ein Link könnte z.B. auf verschiedene Versionen eines Dokuments verweisen und diese regelbasiert in ein Data-Warehouse integrieren.
- Extraktion beliebiger Dokumentteile mittels Xpath oder Verweis auf definierte Elemente innerhalb einer Dokumentinstanz mittels Xpointer.

Das besondere Potenzial von XML liegt im Aufbau generischer Schnittstellen. Zusätzliche Markierungen innerhalb der Dokumente werden bei Bedarf ohne Migrationsaufwand von Zielsystem entweder automatisch verarbeitet oder einfach ignoriert. Die Internetfähigkeit von XML erlaubt darüber hinaus die Adressierung und Anbindung von Unternehmensdatenquellen auch über eine Firewall hinaus.

Externe Daten liegen in operativen und analytischen Systemen von Partnerunternehmen oder Daten- und Dokumentsammlungen wie dem World Wide Web (WWW) vor. Marktanalysen, lichttechnische Daten oder Geometriedaten könnten beispielsweise interne Controllingdaten ergänzen und

somit die Qualität der Entscheidungsgrundlagen erhöhen. Die Integration solcher Daten ist allerdings eine komplexe Aufgabe, da bei allen externen Systemen zunächst ein Austauschprotokoll definiert werden muss. Ohne die semantische Auszeichnung der Daten stellt die Integration daher eine kaum zu bewältigende Aufgabe dar. Dies kann sich ändern, sobald immer mehr Daten in XML-Format zur Verfügung stehen.
Der Grad der Dynamik der Web-Daten und zugrundeliegende Datenmenge bestimmen die Art der Integration in ein Data-Warehouse. Die Varianten reichen vom vollständigen Download der Web-Daten bis hin zur dynamischen Integration. Die hohe Komplexität dieser Aufgabe spiegelt sich in unterschiedlichen Forschungsansätzen wieder: z.B. MIX (Representation Model for Integration of Web-Data), Xyleme (Dynamic Data Warehouse for XML).

2.4.2 XMLA (XML for Analysis)

im Rahmen einer Standardisierungsinitiative unter der Führung der Softwarehersteller Microsoft, Hyperion und SAS im XMLA Council wurde im Frühjahr 2001 der Standard XML for Analysis (XMLA) realisiert. Das Ziel dieser Kooperation, die inzwischen durch 27 Business-Intelligence-Anbieter unterstützt wird, ist die Definition eines offenen, XML-basierten Standardprotokolls für analytische Applikationen.
XMLA erlaubt einen standardisierten Zugriff auf multidimensionale Datenbestände, unabhängig von der verwendeten Plattform und Infrastruktur. Der Ursprung des Protokolls ist in Microsoft´s OLE DB für OLAP (ODBO) zu finden, der Schnittstellendefinition zur multidimensionalen Datenbank SQL Server Analysis Services. Insofern ist XML for Analysis auch als proprietär einzustufen, wobei die zumindest angekündigte Unterstützung dieses Protokolls durch fast alle Hersteller von Business-Intelligence-Produkten und ihre Einbeziehung zur Weiterentwicklung im XML Council die Entstehung eines Industrie-Standards vermuten lässt.
Wesentliche Eigenschaften von XMLA sind:

- Plattformunabhängigkeit durch den Einsatz der Internetprotokolle http bzw. HTTPS,

- Nutzung von XML als Kommunikationsprotokoll in einer n-fachen Architektur,
- Thin-Client Achitektur für installationsfreie Anwendungen.
- Einsatzmöglichkeit jeder Programmiersprache zur Erzeugung und Verarbeitung von XMLA,
- Einsatz von SOAP (Simple Objects Acess Protokoll) als einfachem und auf XML beruhendem Mechanismus zum Austausch strukturierter Informationen sowie
- Stateless-Architektur und Connection-Pooling für einen schonenden Umgang mit Server-Ressourcen .

Das Release 1.1 stellt zwei Verfahren für den Zugriff auf multidimensionale Daten zur Verfügung:

a) DISCOVER dient dazu, Informationen über Datenquellen zu sammeln,
b) EXECUTE wird verwendet, um Datenquellen abzufragen.

Alle Anfragen werden über einen Web-Server an den XMLA-Server (Provider) weitergereicht und dort sowohl in schreibender als auch in lesender Weise verarbeitet. Die Ergebnisse werden anschließend wieder im XML-Format als Result-Set in tabellarischer oder hierarchischer Form zurückgeliefert (*Abb.2.9*). Das Result-Set kann, je nach Anwendungszweck, in einer BI-Anwendung weiterverarbeitet oder z.B. mit XSLT für einen Web-Browser aufbereitet werden.
Als inhaltliche Spezifikation kommt die von Microsoft definierte Abfragesprache für multidimensionale Datenbanken MDX (Multidimensional Expressions) zum Einsatz, im Rahmen von XMLA in ihrer für den XML-Einsatz angepassten Form unter dem Namen mdXML. Es können aber auch providerspezifische Anfragesprachen eingesetzt werden.
XMLA ist Teil der von Microsoft propagierten .NET-Initiative, d.h. der Umsetzung des Prinzips der Webservices. Als unabhängige und in sich geschlossene Softwarekomponenten dienen BI-Webservices zur Bereitstellung dynamischer Anwendungen im Internet und der jederzeitigen Verfügbarkeit von analytischen Diensten.

Die Einsatzmöglichkeiten von XMLA beschränken sich nicht nur auf die Abfrage multidimensionaler Datenbestände, sondern können auch auf Datenquellen für das Data-Mining ausgedehnt werden. Mit dem Eintritt von SAS Institute als einer der führenden Anbieter im Data-Mining-Markt in die XMLA-Initiative wurde ein weitere Grundstein zur Weiterentwicklung von XMLA für den Data-Mining-Einsatz gelegt.

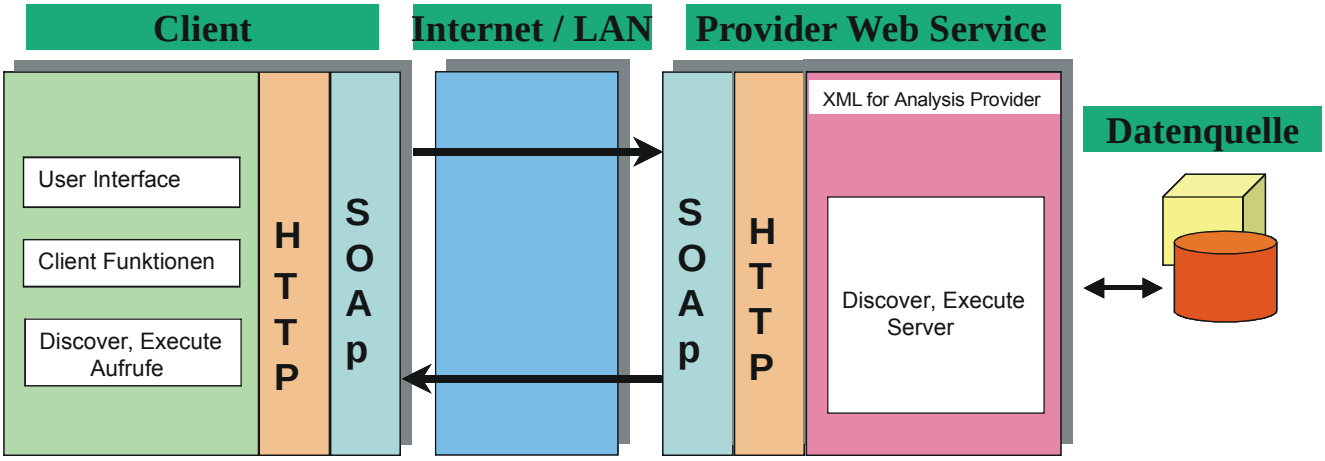

Abb. 2.9: Achitekturschema XML for Analysis

2.4.3 PMML (Predictive Model Markup Language)

Im Bereich der Data-Mining-Werkzeuge besteht der Bedarf, Modellparameter zwischen Data-Mining- und anderen Anwendungen, Datenbanken und Visualisierungswerkzeugen auszutauschen. Zu diesem Zweck wurde durch die Data-Mining Group (DMG), ein Zusammenschluss namhafter Hersteller aus dem Softwaresegment, mit PMML eine auf XML basierende Markierungssprache zur Beschreibung von Data-Mining-Modellen spezifiziert. Mit der Hilfe von PMML werden semantische Modelle erstellt (z.B. zur Klassifikation mittels Entscheidungsbäumen), aus denen Vorhersagemodelle (prädikative Modelle)

abgeleitet werden. Die Struktur der Modellbeschreibungen, die als Document Type Definition (DTD) gespeichert werden, lässt sich in drei Bereiche einleiten:

- Den Dateninput der DM-Modelle
- Die Transformationen, die zuvor verwendet wurden, um die Daten für das DM vorzubereiten
- Und die Parameter, mit denen die jeweiligen Modelle definiert werden.

Integriert werden diese Modelle in relationale Datenbanken oder auch Anwendungen wie Customer-Relationship-Management-(CRM)-Systeme, die auf Data-Mining-Modelle zugreifen.

2.4.4 XMI (XML Metadata Interchange)

Der Einsatz von BI-Werkzeugen unterschiedlicher Hersteller erfordert auch im Bereich der Modellierungssprachen und Metadaten einen standardisierten Datenaustausch. Die meisten BI-Komponenten generieren semantische oder technische Metadaten, häufig ergänzt um explizit definierte. Besondere Bedeutung hat dabei das Metadaten-Repositorium, um das als Integrationsplattform der konsistenten Bereitstellung von Metadaten dient. Der Standard XMI (XML Metadata Interchange) wurde durch das Industriekonsortium OMG (Object Management Group) erarbeitet und dient der Beschreibung und dem Austausch objektorientierte Modelle bzw. Metamodelle. Dabei wird UML (Unified Modelling Language) zur graphischen Repräsentation der Modelle und MOF (Meta Object Facility) zur Spezifikation von Metamodellen verwendet.

Die innerhalb der OMG ins Leben gerufene Standardisierungsinitiative CWM (Common Warehouse Metamodel) hat sich auf den Metadatenaustausch im BI-Bereich fokussiert. Das Ziel dieser Initiative, ist es, den herstellerübergreifenden Austausch von Metadaten in einem heterogenen Data-Warehouse-Umfeld zu gewährleisten. Zahlreiche Teilmodelle im CWM definieren die verschiedensten Aspekte von Business-Intelligence-Systemen – von der Beschreibung von Datenstrukturen und Modellen über Transformationen und Informationsvisualisierungen bis hin zu regelmäßigen Datenintegrationsprozessen.

Die Initiative erreicht durch den standardisierten Austausch von technischen und semantischen Metadaten eine weitgehend vertikale Integration der BI-Systeme. Das Fernziel ist die automatische Generierung der Systemarchitektur.
Ein Vorläufer ist das Open Information Model (OIM) der Standardisierungsinitiative Meta Data Coalition (MDC). Hier dient XIF (XML Interchange Format) bzw. MDC XML Encoding zum standardisierten Austausch von Modellen und Metadaten in heterogenen Data-Warehouse-Umgebungen. Über die UML-basierte Beschreibung der Informationsmodelle werden auch hier spezifische Teilmodelle abgeleitet. Inzwischen hat sich aber die MDC der OMG-Initiative angeschlossen, um zukünftig XMI als alleinigen Standard weiterzuentwickeln.

2.4.5 XBRL (eXtensible Business Reporting Language)

Neben den technisch orientierten Austauschprotokollen, die primär Schnittstellen und Befehlssätze definieren, existiert mit XBRL (eXtensible Business Reporting Language) auch ein inhaltlicher Standard für den relevanten Teilbereich der Finanzberichterstattung. XBRL unterstützt insbesondere die standardisierte Verbreitung, Veröffentlichung und Auswertung von Daten des externen Berichtswesens. Auf diese Weise ermöglicht der Einsatz von XBRL einem Unternehmen eine standardisierte Aufbereitung und Mehrfachverwendung der Jahresabschlussdaten, z.B. zur Veröffentlichung im Internet, für Zwecke der Wirtschaftsprüfung, zur Weitergabe an Kreditgeber, die Börsenaufsicht oder die Behörden. Auf der anderen Seite wird es für Konsumenten der Jahresabschlussinformationen einfacher, diese automatisch zu verarbeiten und zu vergleichen (z.B. im Rahmen des Investment Research).
Die Basis eines XBRL-Dokuments bildet die *Taxonomy.* Sie definiert die Elemente und Attribute, die ein XBRL-Dokument enthalten soll, sowie ihre Beziehungen zueinander. Die Taxonomy stellt somit die Modellbeschreibung (DTD/XSD) eines XBRL-Dokuments dar und legt z.B. den Namen des Unternehmens, die Positionen der Gewinn- und Verlustrechnung sowie der Bilanz und deren Beziehungen fest. XBRL-Taxonomies liegen für den internationalen Rechnungslegungsstandard

IAS/IFRS sowie für die verschiedenen nationalen Vorschriften wie US-GAAP oder HGB vor.

Die Erweiterbarkeit von XML ermöglicht den Einsatz von XBRL auch für das interne Reporting, indem die vorhandene Taxonomy um eine unternehmensspezifische Semantik erweitert wird. Auch hier ist das Ziel die Erhöhung der Transparenz der unterschiedlichen Berichtseinheiten und eine inhaltliche Standardisierung des Datenaustausches.

2.5 Umsetzung in verfügbaren Softwarewerkzeugen

Stand und Entwicklung der Umsetzung der beschriebenen Einsatzmöglichkeiten von XML in Business-Intelligence-Systemen lässt sich an aktuell verfügbaren Produkteigenschaften sowie Ankündigungen der Softwarehersteller für zukünftig erscheinende Releases der Produkte untersuchen. Die Betrachtung beschränkt sich dabei auf die weltweit 14 größten Anbieter von Business-Intelligence-Software, gemessen am Softwarelizenz- und Wartungsumsatz. Untersucht wird jeweils das gesamte Produktspektrum, das zum Aufbau entscheidungsorientierter Informationensysteme eingesetzt werden kann, also insbesondere Werkzeuge zur Datenintegration, Datenbanken zur Speicherung sowie Anwenderwerkzeuge für Reporting, Analyse, Planung bis hin zu Data-Mining.

Die Abbildung *Abb. 2.10* zeigt die Anbindung von XML an die verschiedenen Programmiersprachen.

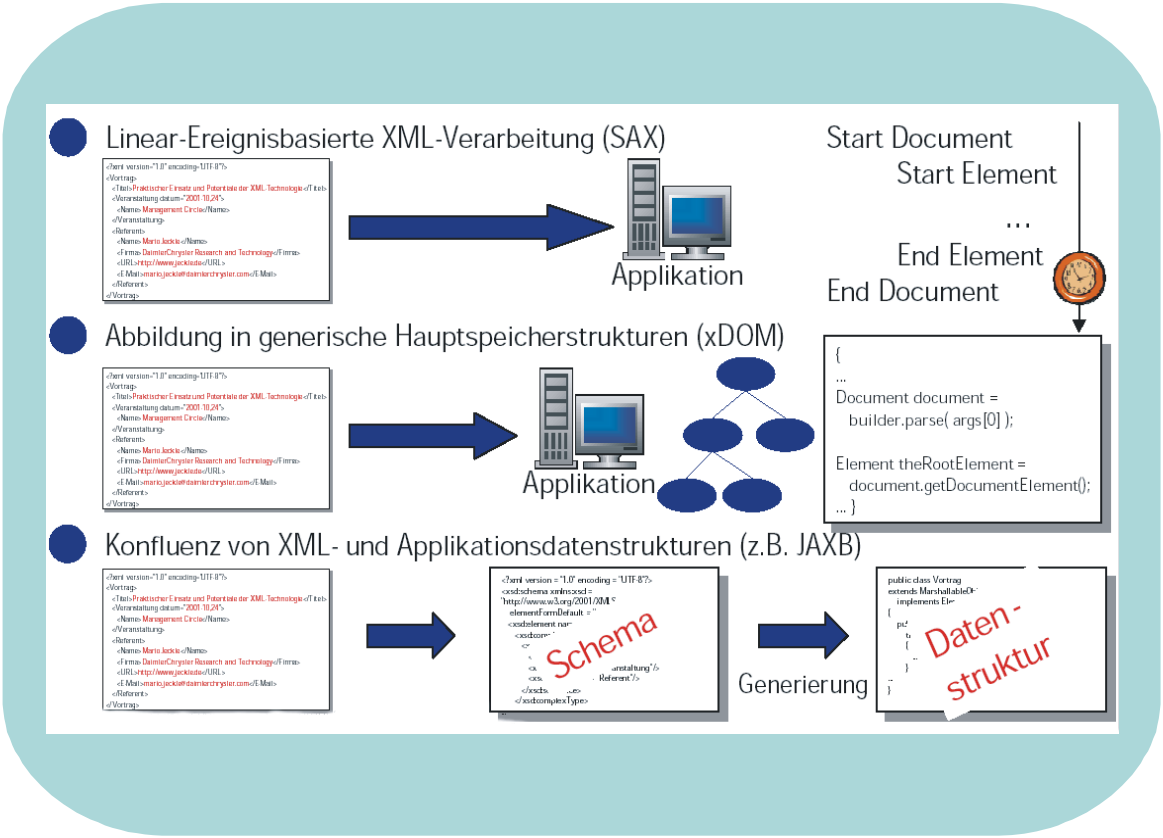

Abb. 2.10: Anbindung von XML an Programmiersprachen

Zusammenfassend kann festgehalten werden:

Datenintegration:

- XML-Dateien mit DTD können von allen Softwareanbietern verarbeitet werden,
- Die XML-Schema-(XSD-) und Web-service-Unterstützung ist dagegen noch wenig verarbeitet.

Speicherung:

- XML-Generatoren und –Datentypen sind bei fast allen Anbietern mit relationalen Speicherkomponenten vorhanden;
- Eine datenbankseitige Unterstützung von XMLA ist bei vielen Anbietern multidimensionaler Datenbanken gegeben.

Abfrage und Präsentation:

- XQL oder Xquery als Abfragesprachen werden kaum unterstützt.
- Die Unterstützung für XMLA als Austauschstandard für BI-Clients findet breite Unterstützung, allerdings hauptsächlich in einer Vorversions-/Ankündigungsphase. Tatsächliche Realisierungen sind erst selten anzutreffen.
- Business-intelligence-Webservices werden von den meisten BI-Anbietern als Entwicklungsziel angegeben. Teilweise sind in den Entwicklungswerkzeugen bereits Möglichkeiten enthalten, Reporting- oder Analysefunktionen als Webservice zur Verfügung zu stellen. Eine Umsetzung im Bereich der Transformation sowie die Integration von Webservices anderer Anwendungen sind dagegen kaum vorhanden.
- Im Rahmen der inzwischen weitgehend webbasierten Anwenderwerkzeuge wird XML sehr häufig zum Datenaustausch aber auch zur Applikationssteuerung eingesetzt.
- PMML-Erzeugung wird noch in geringem Umfang unterstützt.
- XML-Export ist weit verbreitet.

Metadaten:

- Der XML-basierte Metadatenaustausch per CWM/XMI-Spezifikation ist inzwischen bei vielen Werkzeugen umgesetzt. Eigene Definitionen und ältere XIF-Standard sind dagegen selten unterstützt.
- Eine Unterstützung von Webservices im Bereich Repositrien und Metadatenaustausch ist kaum anzutreffen.

Die beschriebenen Einsatzmöglichkeit und aktuell bereits erreichte Umsetzung von XML in BI-Systemen zeigen die Potenziale dieser Technologie auf. Für die vorherrschenden Integrationsaufgaben bietet XML auch im BI-Bereich eine Kommunikationsform, die durch ihre Offenheit, Flexibilität, Erweiterbarkeit und Plattformunabhängigkeit für eine weite Verbreitung sorgen wird. Dabei spielt XML zukünftig nicht nur in Ergänzung von vorhandenen Systemen eine Rolle, sondern generiert durch die dann mögliche Interoperabilität und Internetfähigkeit Synergien und ganz neue Anwendungsfelder. Wesentliche Einflussdimensionen sind dabei:

- ***Externalisierung:*** XML-Technologien erleichtern deutlich die Bereitstellung und Verfügbarkeit von Informationen aus Business-Inteligence-Systemen für Nutzergruppen außerhalb der Unternehmensgrenzen. Diese auch als „Collaborative Business Intelligence“ bezeichnete Externalisierung von Informationen ist eine der Hauptanforderungen an zukünftige BI-Systeme. Mit der Verfügbarkeit von BI-Webservices wird es dann z.B. für alle Teilnehmer einer Supply-Chain einfacher, die Informationsversorgung sicherzustellen. Neue Teilnehmer können leichter integriert werden, indem sie ihre BI-Webservices in einem UDDI-Verzeichnis registrieren.
- ***Integration*****:** XML als Standardprotokoll für BI-Systeme sorgt für eine vereinfachte Einbindung der Systeme in die IT-Infrastruktur eines Unternehmens sowie auch in unternehmensübergreifende Integrationsszenarien. Eine besondere Rolle spielen hierbei BI-Webservices zur vereinfachten und dynamischen Integration von Datenquellen und analytischen Funktionen. Die Integrationspotenziale von XML beziehen sich auf alle Ebenen der BI-Systeme im Rahmen der Anwendungs- und Datenintegration. Als semistrukturiertes Dateiformat können auch strukturierte und unstrukturierte Daten zur Verfügung

quantitativer und qualitativer Informationen in Business-Inteligence-Systemen integriert werden.

- ***Standardisierung:*** eine Standardisierung von Systemkomponenten und Austauschprotokollen sorgt für eine Erzielung von Netzeffekten, die sowohl Anwendern als auch Softwareherstellern zu Gute kommen. Zu nennen sind insbesondere Plattform- und Programmiersprachenunabhängigkeit sowie die Internetfähigkeit, die dazu führen, dass BI-Anwendungen nicht mehr für unterschiedliche Einsatzbereiche jeweils neu konzipiert werden müssen. Die Wiederverwendbarkeit von Schnittstellen und Anwendungen und Verfügbarkeit von Metadaten verbessern sich, sodass die Kombinationsfähigkeit von Systemen unterschiedlicher Hersteller deutlich vereinfacht oder überhaupt erst möglich wird. Das Fernziel einer generischen Architektur (Model-Driven Architecture) rückt durch den Einsatz von XML zumindest ein Stück näher.
- ***Rationalisierung:*** Rationalisierungspotenziale durch den Einsatz von XML entstehen in Form von Kosten-und Zeitersparnis beim Aufbau und Betrieb von BI-Systemen. Standardisierte APIs und Abfragesprachen vermeiden Mehrfachentwicklungen und die daraus resultierende Komplexität. Neue Systemarchitekturvarianten ermöglichen die Virtualisierung von Komponenten wie einer zentralen Datenbank und Applikationsservern. An die Stelle von festen Systemstrukturen treten dynamisch und individuell zusammenstellbare Webservices.

Der größte Vorteil von XML, nämlich die Erweiterbarkeit, ist aber auch gleichzeitig die größte Schwäche. Die Zahl der bereits jetzt vorhandenen und teilweise konkurrierenden XML-Standards zeigt, dass das Problem der Integration von BI-Systemen nicht automatisch durch die Einführung von XML gelöst wird. Der Schwerpunkt der Diskussion wird daher zukünftig auf der Durchsetzung einheitlicher Semantiken liegen. Die inzwischen erfreulich breite Unterstützung von CWM/XMI und die Potenziale von XMLA und XBRL zeigen hier den Weg. Die größte, aber auch technisch lösbare, Herausforderung liegt in der Bewältigung der deutlich vergrößerten Datenmengen, die durch die XML-Auszeichnung generiert werden.

Die aufgezeigten Potenziale von XML für Anwender und Entwickler bieten ganz neue Möglichkeiten in einer neuen Generation von BI-System, zu denen die ambienten Beleuchtungssystemen mit Sicherheit gehören.

2.6 XML und SOAP

SOAP beschreibt das Verpacken, Codieren und den Austausch von strukturierten Daten wie einem XML Dokument. SOAP hat den Vorteil, dass es generell genug gehalten ist, um quasi alle strukturierten Daten transportieren zu können. Daher hat es sich inzwischen zu dem Standardrahmen für den Nachrichtentransport der Webdienste von Microsoft, Sun, IBM, Oracle, HP und vielen anderen Unternehmen entwickelt.

SOAP-Nachrichten bestehen aus zwei notwendigen Teilen: Zum einen dem Envelope, der die Daten in sich verpackt und zum anderen dem Body, der die eigentlichen Daten beinhaltet. Als drittes optionales Element gibt es noch den Header (können auch mehrere sein), der zusätzliche Informationen zu der Nachricht, den Daten oder Bearbeitungsinstruktionen beinhalten kann. Ein Beispiel einer SOAP-Nachricht ist in Abbildung Abb.2.11 schematisch dargestellt.

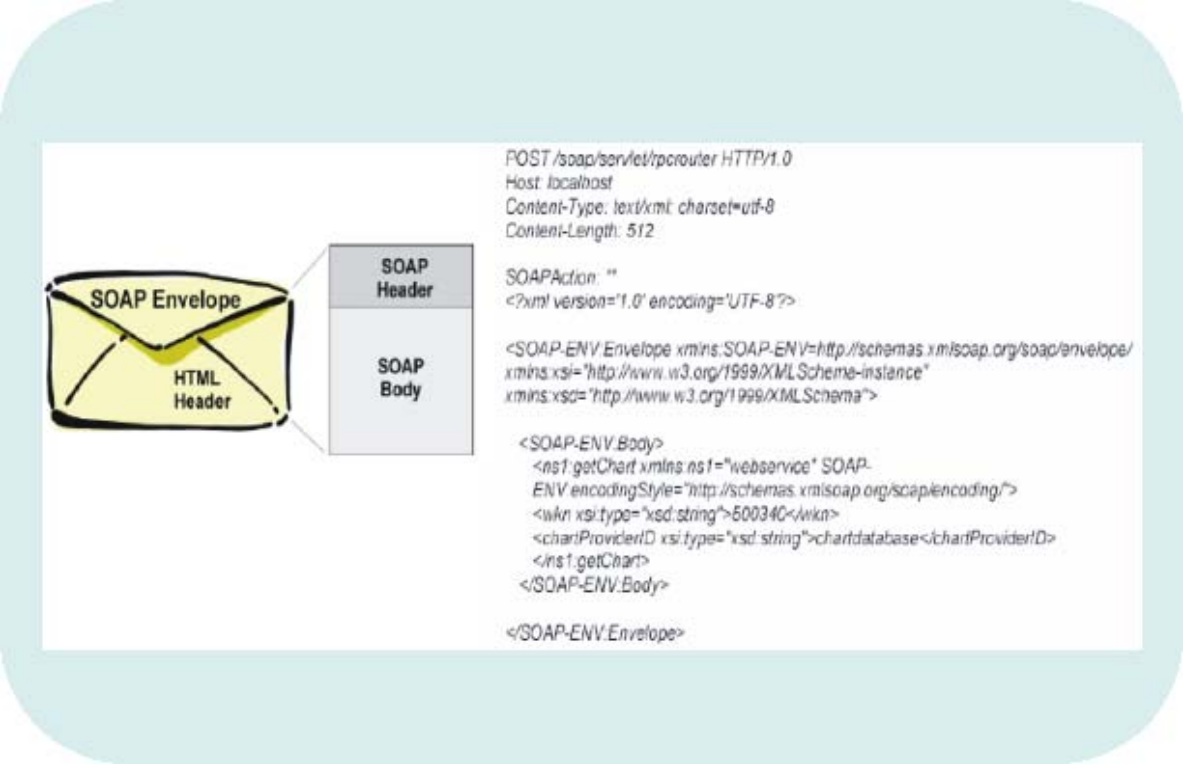

Abb. 2.11: Aufbau einer SOAP-Nachricht

SOAP behandelt nicht die Semantik oder Struktur der in ihm liegenden Daten, sondern dient lediglich als eine Schnittstelle für den Datenaustausch und gibt Anweisungen, wie mit ankommenden Daten umgegangen werden soll. Dabei können die XML-Daten direkt durch einen Parser oder andere vergleichbare XML-Tools manipuliert werden. Sie werden aber im Rahmen dieser Arbeit durch den XML-Konverter, der die Daten in das zur Bearbeitung durch die Anwendersoftware geforderte Format wandelt, vor dem Schicken behandelt werden. Die Software bearbeitet dann die empfangenen Daten, die sie von dem XML-Konverter als Ergebnis bekommen hat. Der XML-Konverter übersetzt die Daten zur Weitergabe wieder in XML/SOAP.

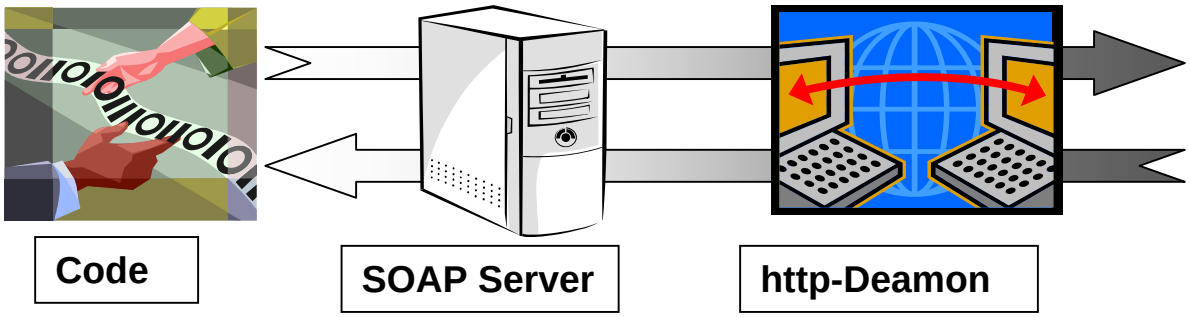

Abb. 2.12: XML/SOAP – SOAP/XML Processing

Es gibt für alle verbreiteten Programmiersprachen und Softwareentwicklungsumgebungen SOAP Toolkits. Den Transport des SOAP-Umschlages der Nachricht können verbreitete Transportprotokolle wie FTP, HTTP, IO, Jabber, SMTP, POP3 oder TCP übernehmen.
Auf diese Weise wird es möglich, beliebige Daten plattform-, netz- und produktübergreifend mit den bisherigen Transportumgebungen auszutauschen. Ein bisher unbekanntes Maß an Flexibilität und Transparenz wird die Folge sein.

2.7 Bewertung

die Untersuchung der BI-spezifischen Anwendungsgebiete zeigt, dass für XML in allen Bereichen ein bereites Einsatzspektrum besteht, insbesondere bei den angestrebten Ziele bei den ambienten Beleuchtungssystemen. Schwerpunkte liegen dabei in der Definition von Austauschprotokollen für die Integration und Präsentation von Nutz- und Metadaten.

Der Einsatz von XML-Core-Standards erleichtert die Integration interner und externer Daten durch die Standardisierung der Austauschformate. Die Erstellungs- und Betriebskosten im Data-Warehouse können durch den Aufbau dynamischer stabilerer Anwendungen mittels generischer Schnittstellen reduziert werden. Ein weiterer Vorteil ergibt sich durch die Möglichkeit der Validierung von XML-Dokumenten, die zu einer höheren Datenqualität führt. Für komplexere Manipulationen der Quelldaten sind die Core-standards allerdings nicht geeignet.

Kritisch zu bewerten ist die große Menge an Overhead-Informationen zur Inhaltsauszeichnung in XML-Dokumenten. Nur ein kleiner Teil (teilweise nur 10%) der Dokumentengröße wird durch die (Nutz-) Daten selbst repräsentiert. In der Praxis können dadurch sehr leicht große Datenvolumina entstehen, weshalb die Quelldaten daher in viele kleinere Dokumente zerlegt werden müssen. Die Dauer von Datentransfer- und Einfügeoperationen können somit kritische Größen erreichen. Die Zeitfenster für die Aktualisierung von Data-Warehouse sind hier der limitierende Faktor. Größere Bandbreiten im LAN und im Internet, Kompressionsverfahren sowie neue Ansätze zur Echtzeit-Übertragung kleiner Datenmengen (Real-Time Data-Warehouse) lassen diesen Nachteil in Zukunft aber zumindest im Bereich der Datenübertragung immer weniger gewichtig werden.

Der Aufbau eines XML-Repositoriums ist obligatorisch. Nur so lassen sich Element und Attribute sowie unterschiedliche DTD-Versionen verwalten.

BI-spezifische Austauschprotokolle werden in wachsendem Maße eingesetzt und sorgen insbesondere für Interoperabilität in heterogenen Systemumgebungen.

Durch den Einsatz von XMLA können Anwender vor allem von einen Investitionsschutz profitieren: Front-End-Werkzeuge oder Management-Portale müssen nicht für jede OLAP-Datenbank angepasst werden und Entwickler können

ihr vorhandenes Wissen bezüglich der XML-Standards und Webservices mit einbringen.

Aus der technischen Perspektive heraus betrachtet, existieren noch eine Reihe von Problemen, die den Einsatz von XMLA verzögern oder gar verhindern können:

- Im Vergleich zur klassischen 2-Tier-Anwendung ist die Übertragung und Verarbeitung von größeren Datenmengen im textbasierten XMLA-Format zeitkritisch.
- Eine von einem Hersteller dominierte API unterstützt die Kernfunktionen der eigenen OLAP-Datenbank. Sobald aber spezifische Funktionen anderer Anbieter benötigt werden, müssen Softwareentwickler weiterhin auf Original-APIs zurückgreifen.
- Die Funktionen der aktuellen Version sind auf die Identifizierung und Abfrage bestehender Datenquellen begrenzt. Umfangreiche Datenmodellierungsfunktionen fehlen ebenso, wie die Einbindung in Metadatenstandards.

Konkurrenz erhält XMLA zusätzlich von der JOLAP-Initiative, die eine auf Java basierende Spezifikation propagiert. JOLAP stellt z.B. Funktionen zum Metadatenaustausch und zur Datenmodellierung zur Verfügung und hat damit einen umfassenden Ansatz.

XMI als Protokoll für den Metadatenaustausch sorgt für ein Zusammenspiel von Repositorien verschiedener Hersteller. Damit kann der Nutzdatenaustausch um einen Metadatenaustausch ergänzt werden, was gerade in Business-Intelligence-systemen häufig gefragt ist, um technische und semantische Informationen systemübergreifend verfügbar zu machen. Die geplante inhaltliche Erweiterung des Standards kann eine breitere Nutzung zusätzlich zum reinen Modellaustausch bewirken und die Einsatzpotenziale noch deutlich verbessern.

PMML sorgt für die Austauschbarkeit von Data-Mining-Modellen zwischen verschiedenen Applikationen und Datenbanken. Auf diese Weise können Mining-Modelle z.B. zum Scoring von Kunden oder der Betrugsentdeckung direkt in ausführende Applikationen integriert werden. Vor allem der operative Einsatz wird ohne weitern Zeitverzug unterstützt. Bisher mussten in Data-Mining-Werkzeugen aufgebaute Modelle, soweit überhaupt möglich, relativ aufwendig in einer

Programmiersprache exportiert und dann manuell in die andere Applikation integriert werden.

XBRL definiert einen Standard zum Austausch von Finanzinformationen im Rahmen des Jahresabschlusses zur Verbesserung der Transparenz und Vergleichbarkeit. Momentan ist noch unklar, ob ohne Zwang des Gesetzgebers eine breite Unterstützung durch die Unternehmen erreicht werden kann. Problematisch stellt sich auch die Abbildung der teilweise umfassenden Erläuterungen zu einzelnen Berichtspositionen dar.

Die Abbildung *Abb. 2.6* im Unterkapitel 2.2 zeigt umfassend die Einsatzmöglichkeiten der weltweit verfügbaren XML-Standards und –Anwendungen.

3 Datenformate von Lichttechnischen Messdaten

Die Definition von Datenformaten zur Ablage der lichttechnischen Messdaten hat den Vorteil, dass der Entwickler nicht mehr nur auf die werkseigenen Konstruktionsdaten zurückgreifen muss. Ihm steht nun vielmehr die Option offen, plattformunabhängig auf einen globalen Datenpool zuzugreifen. Da man in der Lichttechnik meist davon ausgehen kann, dass in dem derzeit betriebenen Forschungs- und Entwicklungsbereich auch schon andere in ähnliche Richtung und mit ähnlichen Ansätzen gearbeitet haben oder zumindest durch ihre Tätigkeit etwas dazu beisteuern könnten, bietet sich durch die globale Vernetzung via Internet die Option, seine Arbeit auf einer kombinierten Wissensbasis aufzubauen. Sowohl der Datentransfer unter Softwareprodukten als auch den Abteilungen des Betriebes sind nun ohne weitere Dokumentation der gelieferten Rohdaten möglich, da sie durch die Ablage in einem vordefinierten Format schon geordnet und definiert geliefert werden.

3.1 Historische Entwicklung der Datenformate

Im Jahr 1986 veröffentlichte die IESNA einen Vorschlag für einen Standard zum Datentransfer von lichttechnischen Daten. Dieser wurde als IES LM-63-1986 von der lichttechnischen Softwareentwicklergemeinde als Standard in Nordamerika akzeptiert. Diesem schlossen sich darauf die Hersteller an, wodurch ein homogener Datenaustausch zwischen den Simulationen und den Produktionsstätten möglich wurde.
In den folgenden Jahren 1991 und 1995 wurde der Standard mehrfach von der IESNA durch Eingaben aus der User-Gemeinde überarbeitet. Die neueste Version dieser Standardserie ist hierbei der ANSI/IESNA LM 63-2002 vom August 2002.
Aus dieser Anfangsserie haben sich diverse Derivate entwickelt, die spezielle Zusätze haben, um ihrem Einsatzgebiet gerecht zu werden. So entstand der LM

74-05 zum Transfer von Leuchtkörperkomponentendaten, und schließlich wurde der LM 75-01 zur Beschreibung von Goniophotometerdaten verabschiedet.
In England entwickelte sich ab 1988 eine eigene Domäne, deren Gemeinschaft den CIBSE TM14:1988 hervorbrachte. Dieser findet im Allgemeinen nur im Königreich Verwendung.
1990 entwickelte Axel Stockmar von LCI Software Engineering den Standard EULUMDAT.
Obwohl hinter EULUMDAT keine Organisation steht, die ihn unterhält und es auch keine Dokumentation gibt, ist es umso erstaunlicher, dass ihn die europäische Lichtgemeinschaft als de facto Standard angenommen hat.
Schlussendlich konnten die Franzosen es nicht nehmen lassen, sie hatten die Absicht ihren eigenen Standard zu entwickeln. Der aus dem Jahr 1993 stammende CIE 102-1993 mag zwar ein gutes und verständliches Format sein, doch gibt es praktisch niemanden, der ihn nutzt.

3.2 Vorstellung der bekannteren Datenformate

Im Prinzip kann man aus der großen Vielfalt der selbsternannten Standards nur vier wirklich als solche bezeichnen, da die anderen eher firmeninterne denn globale Formate sind. Es werden daher in der folgenden Betrachtung auch nur diese vier aufgeführt.

3.2.1 IESNA LM xx-xxxx (63-1995, 63-2001, 63-2002, 74-05, 75-01)

Die IESNA entwickelt in einer ständigen Dynamik neue Standards und Verfahren für die Lichttechnik von Nordamerika, deren sich aber auch Hersteller in Übersee bedienen. Auf diesem Wege wird versucht, Ordnung in das mitunter ausufernde Chaos der Datenformate und Messverfahren zu bringen. Da sich die Lichttechnik nicht nur mit Goniophotometern oder ambienter Beleuchtung befasst, verbietet die

große Vielzahl der Verfahrensvorschläge und anderer Vorgaben es, sie alle hier zu erwähnen. Daher wird hier auf die essentiell wichtigen Punkte für das Vermessen eines Körpers mit einem Goniophotometer und der anschließenden Ablage und strukturierten Speicherung der Daten beschränkt.

Man muss hier dazu zunächst den LM 75-01 erwähnen, der das Verfahren der Vermessung mittels eines Goniophotometers behandelt. In ihm werden unter anderem die zu wählenden Koordinaten und Positionen festgelegt und eine Einweisung in den geforderten Ablauf gegeben.

Der LM 63-2002 ist die neueste Variante der LM 63-xxxx Standardserie, die sich mit dem Ablegen, also dem Dateiformat für lichttechnische Daten an sich beschäftigt. Allerdings scheint es so zu sein, als dass sich die LM 63-1995 Variante bei vielen eingesessenen Herstellern als Primärdatenformat hält. Dies könnte dadurch begründet werden, dass diese Hersteller ältere Softwareprodukte nutzen, die die neueren Standards noch nicht ausgeben und einlesen können und sie daher ihre Datenbestände im alten Format belassen.

Ein neuer und weiterer wichtiger Vorschlag dieses Gremiums ist der LM 74-05. In ihm wird die Ablage der Daten von Leuchtkörperkomponenten in einem flexiblen strukturiertem XML-Format beschrieben. Dies ist sowohl für kleine wie große Datenmengen geeignet. Der größte Vorteil hierbei ist aber die XML-Struktur. Durch diese omnipräsente Dokumentenbeschreibungssprache wird es für eine existente oder zukünftige Software leichter möglich, über Plattformgrenzen hinweg diese Daten auszulesen und zu implementieren. Es ist davon auszugehen, dass die zukünftigen Programme XML-basierende Datenbestände verwalten werden.

Bei Verfahrensfragen und für zukünftige Entwicklungen kann man sich zuerst immer an die IESNA wenden, da sie auf ihrer Homepage für inzwischen beinahe jeden Bereich der Lichttechnik einen Vorschlag im Angebot hat.

Die Rohdaten werden in ASCII abgelegt, so dass es mit einem gewöhnlichen Editor möglich ist, die Daten manuell zu untersuchen. Die Messwerte werden hierbei in Matrizenform zusammengefasst, was eine manuelle Betrachtung vereinfacht.

Ein großer Nachteil bei den IESNA-Standards ist jedoch, dass nur die zur Validierung existenter Dateien nötigen Dokumentationen frei zugänglich sind. Die eigentlichen Erzeugungsdokumentationen werden nur kommerziell vertrieben.

3.2.2 EULUMDAT

„ Proposal for a data format for exchange of luminaire data (interior, exterior, and/or road lighting luminaires) under the operating systems MS-DOS 2.x/3.xx under condition of unequivocal coordination between luminaire and data set.”
EULUMDAT ist der de facto Standard für Europa, ausgenommen England, und ist seit seiner Einführung 1990 unverändert in Gebrauch geblieben. Axel Stockmar hatte zwar ein Update EULUMDAT/2 genannt im Jahr 1998 herausgebracht, welches viele neue Aspekte berücksichtigt, doch wurde es von der Gemeinschaft nicht erkennbar übernommen.
EULUMDAT ist derart weit verbreitet, dass man es auf beinahe jeder Herstellerseite als Primärformat, mindestens aber als Sekundärformat zum Download von Objektdaten findet. Genau wie bei IESNA liegen auch hier die Rohdaten in ASCII vor. Dies macht es wieder möglich, die Daten mit jedem Texteditor manuell zu untersuchen.
Ein großer Vorteil von EULUMDAT ist, dass dieser Standard nicht-kommerziell ist und daher die Dokumentation frei verfügbar ist. Die einzig bekannte ist via Internet zu beziehen und im Anhang zu finden. Und das ist dies wohl einer der Gründe, warum es im Vergleich zu anderen wohlgeformteren kommerziellen Produkten sich halten und durchsetzen konnte.
Ein Nachteil bei dem EULUMDAT-Format ist jedoch, dass die Messwerte im Textdokument untereinander geschrieben werden, so dass eine Datei, wenn auch klein im Speicherbedarf leicht unleserlich wird, wenn man sie nicht aufbereitet. Die manuelle Untersuchung wird dadurch erschwert. Für die maschinelle Verarbeitung ist es jedoch uninteressant.
Es gibt für EULUMDAT ein frei verfügbares Tool, das die Daten von EULUMDAT in das IESNA LM 63-95 konvertieren kann. Dadurch wird es möglich, diese Datenbestände gegenseitig zu ergänzen.

3.2.3 CIBSE TM14:1988

„ CIBSE standard file format for the electronic transfer of luminaire photometric data“ Dieser kommerzielle Standard ist das offizielle Format für England. Er nähert sich in vielen Punkten dem Standard der IESNA an. Aus Copyright-Gründen ist jedoch nur die Dokumentation zur Validierung existenter Daten frei zugänglich. Um einen eigenen Datenbestand zu bauen, müsste man die kommerzielle Dokumentation beziehen.
Deswegen gibt es weltweit kein Hersteller, der CIBSE als einziges Format seines Produktes angeboten hat. Stattdessen standen zumindest als Sekundäroption IESNA LM 63-xxxx oder EULUMDAT Dateien zu Verfügung.

3.2.4 CIE 102-1993

Leider gibt es nicht all zu viel über den angeblichen Weltstandard zu berichten, da im Rahmen der Recherche bei dieser Arbeit gar keinen Hersteller oder Entwickler gefunden wurde, der ihn zu benutzen scheint.

3.3 Vergleich und Beurteilung

Es scheint so, als dass die meisten Hersteller und Softwareentwickler sich auf zwei Datenformate geeinigt haben. Dies sind zum einen die LM-Serie der IESNA und zum anderen der Standard EULUMDAT von Axel Stockmar. Die anderen beiden, CIBSE und CIE, spielen in der globalen und speziell der europäischen Hersteller- und Softwareentwicklergemeinde eine eher untergeordnete Rolle. Auch bei Herstellern in England, der Domäne von CIBSE, war einer der beiden ersteren Genannten zumindest als Sekundärformat zum Download angeboten worden. Auf vielen Herstellerseiten finden sich sogar beide zusammen. Während die Standards der IESNA von einem Gremium betreut werden und ständig

weiterentwickelt werden, ist EULUMDAT seit seiner Einführung unverändert geblieben.

Der LM 63-xx (63-1995, 63-2001, 63-2002) Standard der IESNA für die Speicherung der Daten hat durch das wiederholt tagende Gremium den Vorteil, dass er stets den Erfordernissen der Zeit entspricht. Allerdings scheint die Lichttechnikgemeinde nicht willens zu sein, mit der Entwicklung Schritt zu halten, und so finden sich vornehmlich ältere Datenformate unter den Produktangaben. Dies stellt bei IESNA LM 63-xx (63-1995, 63-2001, 63-2002) zunächst kein Problem dar, da hier ein Datenfeld existiert, das auf die benutzte Version hinweist. Die neueren Programme sind so in der Lage, die alten Daten richtig auszulesen. Allerdings wird im Laufe der Zeit das Problem für die Hersteller auftauchen, ihre ganzen Datenbestände zu aktualisieren, um mit den neueren Softwareprodukten mitzuhalten und damit zusammenhängend ihre eigene scheinbar veraltete CAE/CAD-Umgebung zu ersetzen. Die alte Software wird in baldiger Zukunft wohl nicht mehr in der Lage sein, die neuen Datenformate zu lesen, so dass der betroffene Hersteller oder Entwickler dann nicht mehr auf die volle Bandbreite des globalen Datenpools zugreifen kann.

Zukunftssicher macht die IESNA ihre Standards zudem durch die Einbeziehung von XML. Diese universelle Dokumentenbeschreibungssprache erlaubt es zukünftigen Lösungen, plattformunabhängig die Daten zu verarbeiten. Die Einführung von XML ist mit dem Standard LM 74-05 geschehen und bietet dadurch die Option für eine weitere Aufwertung dieses Formates. In Verbindung mit SOAP wird es auf einer Windowsplattform möglich sein, dezentrale Aufgabenteilung einzuführen. Mit SOAP wird es möglich sein, anwendungsübergreifend die XML Daten ohne vorige Konvertierung auszutauschen.

EULUMDAT hingegen ist wie schon erwähnt seit seiner Einführung ungebrochen mit der alten Version und Dokumentation in Gebrauch. Der Vorteil bei der Statik dieses Standards liegt hierbei darin, dass die Softwareprodukte und Herstellerdatenbanken nicht ständig überarbeitet werden müssen, um dieses Format zu lesen. Doch ist davon auszugehen, dass sich dieser Standard in mittelfristiger Sicht als nicht flexibel genug erweisen wird und entweder ersetzt werden wird oder die Gemeinde sich nach einem anderen Format orientieren wird.

XML als Beschreibungssprache für die Daten ist in EULUMDAT überhaupt nicht vorgesehen. EULUMDAT ist daher nur als mittelfristige Option zu sehen, an der man derzeit aber nicht vorbeikommt. Die Vorteile wie offene Dokumentation und Omnipräsenz schlagen derzeit jede Weitsicht.

Diese beiden Standards sind durch ihre allgemeine Verbreitung im Laufe der Zeit miteinander so weit zusammengewachsen, das es einen weiteren Vorteil gibt, der für die beiden spricht. Es sind die Vielzahl an kostenlosen Tools, die es zum Konvertieren der Daten vom einen in das andere Format gibt.

4 Ambiente Beleuchtungssysteme

4.1 Definition

Am Anfang dieses Kapitels wird zum besseren Verständnis zuerst einmal geklärt, was man sich unter dem Begriff der Ambienten Beleuchtungssysteme im Automobil vorzustellen hat. Im Automobil treten im wesentlichen drei Einsatzfelder für Licht auf. Das erste Feld ist das älteste und umfasst die funktionelle Außenbeleuchtung. Hierzu gehören zum Beispiel die Scheinwerfer, Nebelscheinwerfer, Fahrtrichtungsanzeiger und Bremsleuchten. Seit kurzem gehören dazu auch die Infrarotsysteme für die bessere Sicht bei Nacht. Das zweite Feld beschäftigt sich mit der Übertragung von Daten und Signalen über Lichtwellenleiter als Alternative zur konventionellen Verdrahtung von Kupferrundleitungen. Bei dem dritten Einsatzfeld handelt es sich um die so genannten „Ambienten Beleuchtungssysteme“.
Der Begriff Ambiente ist dabei aus der Begriffswelt der konventionellen Wohnraumbeleuchtung abgeleitet. In den Wohnräumen wird mit Beleuchtungsmitteln ein angenehmes „Ambiente“ geschaffen. Dabei spielen vor allem die Farben und die gleichmäßige Ausleuchtung eine große Rolle. Aufgrund der Entwicklungen der letzten Jahre ist es heute möglich, dies auch im Automobil umzusetzen. Der Begriff der „Ambienten Beleuchtungssysteme im Automobil“ steht also für die Lichtgestaltung im Interieur des Automobils, wobei immer mehr das Gesamtkonzept in den Vordergrund rückt. Dabei ist eine abgestimmte Gestaltung der Be- und Hinterleuchtung von Fahrerunterstützungs- und Informationssystemen (Anzeigeinstrumente konventionell (Analoginstrumente) oder Digitalvarianten, Displays, Multimedia, Navigation etc.), der Beleuchtung von Bedienteilen (Knöpfen, Schaltern), der Beleuchtung von Komfortausstattung (Spiegel, Getränkehalter, Aschenbecher etc.) und der Fahrzeuginnenbeleuchtung (Innenraum, Ablagefach, Türgriffmulden etc.) das Ziel, das zu erreichen gilt. Hierbei wird allerdings nicht nur auf die Ziele des Designs (Ästhetik, Lichtwirkung, Raumeindruck etc.) geachtet, sondern auch auf die Sicherheit und die Ergonomie.

Die Beleuchtung im Innern des Automobils kann mit den folgenden Technologien verwirklicht werden. Zum ersten ist da die konventionelle Art mit Glühlampen bzw. Miniaturglühlampen zu nennen, die seit Beginn zur Verfügung standen. In der folgenden Zeit, wurden einzelne LEDs eingesetzt, um eine direkte Beleuchtung eines Objekts zu ermöglichen. Bei der dritten Möglichkeit handelt es sich um ein System aus LEDs und einem dreidimensionalen Lichtwellenleiter aus Kunststoff. Diese Möglichkeit ist im Moment Stand der Technik und wird im Allgemeinen für das genannte Feld eingesetzt. In der näheren Zukunft können Kunststoffe eingesetzt werden, die leiten und dabei leuchten (OLEDs, Polytronic). Die ersten Anwendungen sind dabei bereits auf dem Markt (Beispiel: leuchtende Handtasche).
In den folgenden Bildern werden Beispiele für Ambiente Beleuchtungssysteme gegeben.

Abb.4.1: Das linke Bild zeigt die Beleuchtung einer Türgriffmulde des Herstellers Skoda, das mittlere das beleuchtete Armaturenbrett eines Audi A6 und das rechte Bild die Belüftungsrosetten eines Daimler-Chrysler.

Ambiente Beleuchtungssysteme schaffen also für das Empfinden des Betrachters eine beruhigende, auf Glücksgefühle einwirkende Umgebung. Deswegen macht sich die Automobilindustrie seit wenigen Jahren diese Eigenschaft mittels moderner Technologien zunutze.

4.2 Photometrie und Ambiente Beleuchtungssysteme

4.2.1 Grundlagen

Dieser Abschnitt dient zur Begriffserklärung der Photometrie sowie zum besseren Verständnis der photometrischen Messgrößen. Das Wissen über die Größen ist notwendig, um die Daten, die in der Simulationsphase sowie in der Messungsphase auffallen, zu verstehen und um aus den Messgrößen weitere Größen ableiten zu können, wenn dies erforderlich oder gewünscht werden sollte.
In der Optik wird unterschieden zwischen den strahlungsphysikalischen und den lichttechnischen Größen. Die strahlungsphysikalischen Größen betrachten das Licht im wesentlichen als eine Erscheinungsform der elektromagnetischen Welle. Es gibt eine Vielzahl von Strahlungsempfängern, die eine objektive Messung der energetischen Größen ermöglichen. Hierzu gehören z.B. die lichtempfindlichen Schichten von Fotozellen. Die energetischen Größen der Strahlungsphysik sind Strahlungsenergie, Strahlungsleistung, Spezifische Ausstrahlung, Strahlstärke, Strahldichte, Bestrahlungsstärke, Bestrahlung, Raumbestrahlungsstärke, Raumbestrahlung und Bestrahlungsvektor. Da in dieser Arbeit die strahlungsphysikalischen Größen keine besondere Rolle spielen (Gründe: menschliches Sehempfinden), werden diese hier nicht genauer beschrieben. Die Definitionen können bei Bedarf jedoch der DIN 5031 Teil 1 entnommen werden.
Die lichttechnischen Größen leiten sich von den strahlungsphysikalischen Größen ab. Hierbei wird das Licht jedoch nicht objektiv als Welle betrachtet, sondern erhält durch den subjektiven Eindruck des Auges eine Bewertung. Zu den lichttechnischen Größen gehören Lichtmenge, Lichtstrom, Spezifische Lichtausstrahlung, Lichtstärke, Leuchtdichte, Beleuchtungsstärke, Raumbeleuchtungsstärke, Zylindrische Beleuchtungsstärke, Beleuchtungsvektor und Belichtung. Die wichtigsten photometrischen Größen für die gestellte Aufgabe sind Lichtmenge, Lichtstrom, Lichtstärke, Leuchtdichte und Beleuchtungsstärke. Diese werden im Unterkapitel 4.2.2 genauer beschrieben. Die Definition der anderen photometrischen Größen können der DIN-Norm 5031 Teil 3 Seite 7ff. entnommen werden.

Die Formelzeichen sind für die strahlungsphysikalischen und die photometrischen Messgrößen gleich. Zur leichteren Verständlichkeit können jedoch Indizes für die beiden Größen verwendet werden, um eine Verwechslung auszuschließen. Hierbei wird für die strahlungsphysikalischen Größen der Buchstabe „e“ (energetisch) als Index und für die photometrischen Größen der Buchstabe „v“ (visuell) benutzt.

Da das menschliche Auge nur einen sehr kleinen Bereich des elektromagnetischen Spektrums wahrnehmen kann, muss bei der Bewertung von leuchtenden Objekten streng zwischen der strahlungsphysikalischen und der photometrischen Bewertung unterschieden werden. Dieser Bereich liegt zwischen etwa 380nm (blaues Licht) und 780nm (rotes Licht), wobei in diesem Bereich die verschiedenen Wellenlängen sehr unterschiedlich stark bewertet werden. Weiterhin muss bei der Bewertung zwischen Tages- und Nachtsehen unterschieden werden. Wie die Begriffe vom Namen her andeuten, muss unterschieden werden, ob die Bewertung bei Tageslicht oder bei Nacht durchgeführt wird.

Bei Tageslicht liegt die Wellenlänge, bei der das menschliche Auge am empfindlichsten reagiert bei 555nm (Zitronengelb) und die Wellenlängen am Rand des sichtbaren Spektrums werden nur sehr gering bewertet. Bei Nachtsehen liegt die empfindlichste Wellenlänge bei 507nm. Auch beim Nachtsehen werden die Wellenlängen am Rand des Spektrums nur noch sehr gering bewertet. Der Hellempfindlichkeitsgrad V(λ) der jeweiligen Wellenlängen kann entweder aus Tabellen oder aus der so genannten V(λ)-Kurve abgelesen werden. Aus den oben genannten Gründen folgt, dass es jeweils eine Kurve für Tages- und eine für Nachtsehen gibt. Bei der Bewertung von Lichtquellen werden im Allgemeinen jedoch immer Messsensoren benutzt, deren spektrale Empfindlichkeit bestmöglich an die Hellempfindlichkeitskurve für das Tagessehen angepasst sind. Die V(λ)-Kurven für das Tages- und Nachtsehen ist auf der Abbildung *Abb.4.2* zu sehen. Die blaue Kurve gibt hierbei die Empfindlichkeit des Auges bei Nacht und die rote die bei Tag an. Bei Bedarf können die Tabellen für das Tages- und das Nachtsehen der Tabelle 2 in der DIN 5031 Teil 3 (Seite 3ff.) entnommen werden.

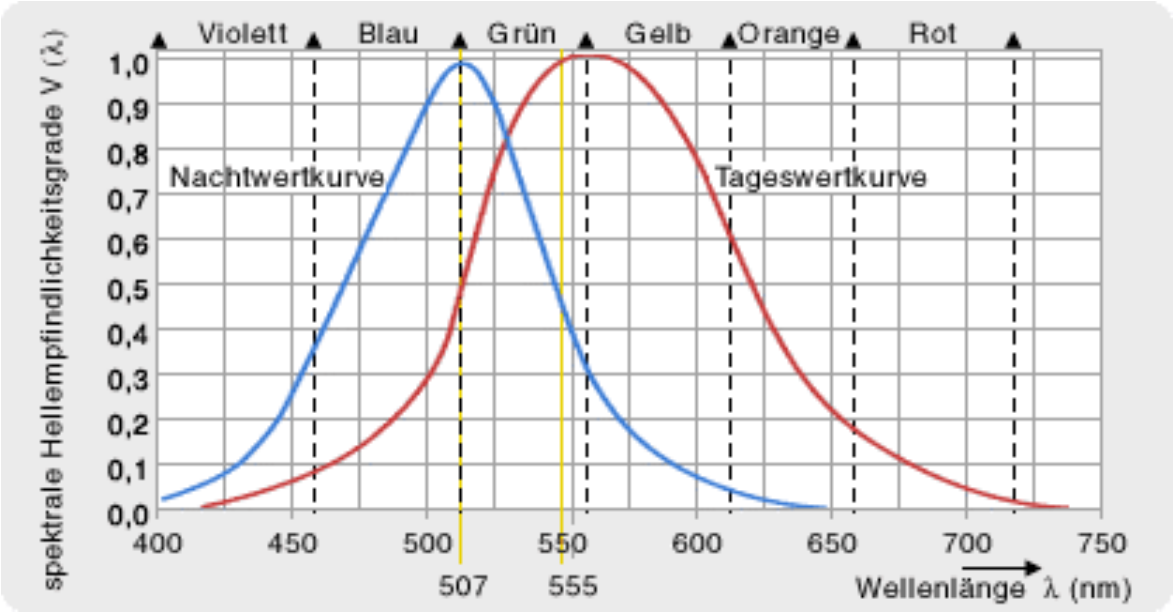

Abb. 4.2: Hellempfindlichkeitskurve des menschlichen Auges

4.2.2 Photometrische Messdaten von Ambienten Beleuchtungssystemen

In diesem Unterkapitel werden die für die ambienten Beleuchtungssysteme wichtigsten photometrischen Größen erklärt (diese Größen fallen in der Simulationsphase sowie in der Messungsphase auf), die Beziehung zwischen den einzelnen aufgezeigt und Möglichkeiten der Messung angegeben.
Die mathematischen Beziehungen der wichtigsten photometrischen Größen untereinander werden in der folgenden Tabelle angegeben. Eine vollständige Tabelle mit allen photometrischen Größen ist in der DIN-Norm 5031 Teil 3 Seite 7f. oder in Büchern über die Grundlagen der Photometrie zu finden.

Größe	Formelzeichen	Einheit	Beziehung
Lichtmenge	Q	Lumensekunde $[lm \cdot s]$	$Q = K_m \cdot \int Q_{e\lambda} \cdot V(\lambda) \cdot d\lambda$

Lichtstrom	Φ	Lumen $[lm]$	$\Phi = \frac{dQ}{dt}$
Lichtstärke	I	Candela $[cd]$	$I = \frac{d\Phi}{d\Omega_1}$
Leuchtdichte	L	Candela/Quadratmeter $\left[\frac{cd}{m^2}\right]$	$L = \frac{d^2\Phi}{dA_1 \cdot \cos(\varepsilon_1) \cdot d\Omega_1}$ $L = \frac{d^2\Phi}{dA_2 \cdot \cos(\varepsilon_2) \cdot d\Omega_2}$
Beleuchtungsstärke	E	Lumen/Quadratmeter $\left[\frac{lm}{m^2}\right]$	$E = \frac{d\Phi}{dA_2}$

Tab4.1: Photometrische Größen und ihre Beziehung untereinander

Wie in dieser Tabelle gut zu erkennen, ist der Lichtstrom eine sehr wesentliche Größe, da aus diesem alle wichtigen Größen zu bestimmen sind. Eine besondere Rolle spielt das photometrische Grundgesetz (siehe Gleichung (4-1)), welches den Lichtstrom mit der Leuchtdichte verknüpft.

$$d^2\Phi = L \cdot \frac{dA_1 \cdot \cos(\varepsilon_1) \cdot dA_2 \cdot \cos(\varepsilon_2)}{r^2} \cdot \Omega_0 \qquad (4\text{-}1)$$

Das Gesetz gilt streng genommen nur im Vakuum. In Materie gilt es nur, bei Vernachlässigung von Absorption, Streuung und Lumineszenz.
Aus dem photometrischen Grundgesetz kann das so genannte photometrische Entfernungsgesetz abgeleitet werden. Diese lautet wie folgt:

$$E = \frac{I}{r^2} \cdot \cos(\varepsilon_2) \cdot \Omega_0 \quad (4\text{-}2)$$

Diese Beziehung hat allerdings nur Gültigkeit, wenn der Abstand zwischen der Lichtquelle und dem Empfänger größer als die photometrische Grenzentfernung ist. Die photometrische Grenzentfernung gibt den Punkt an, von dem eine

Lichtquelle als Punktlichtquelle angesehen werden kann. Dabei ist die photometrische Grenzentfernung abhängig von dem zugelassenen Fehler, der Ausdehnung der Lichtquelle, der räumlichen Lichtstärkeverteilung und der örtlichen Leuchtdichteverteilung der Lichtquelle.
Zusätzlich beeinflussen die Empfängereigenschaften bei der Messung der Beleuchtungsstärke die photometrische Grenzentfernung.
Eine Formel zur Berechnung der photometrischen Grenzentfernung eines kreisrunden Lambert-Strahlers in Abhängigkeit vom zugelassenen Fehler lautet:

$$a_G = \frac{R}{\sqrt{F_{rel}}} \quad \textit{(4-3)}$$

Hierbei gilt:

a_G : Entfernung zwischen Lichtquelle und Empfänger
R : Radius der betrachteten Kreisfläche
F_{rel} : max. vorgegebener relativer Fehler

Aus dieser Formel lässt sich herleiten, dass bei einem zulässigen Fehler von 0,25 %, die Entfernung zwischen der Lichtquelle und dem Empfänger mindestens dem 20 fachen Radius der kreisrunden Lichtquelle betragen muss. Dies heißt, dass die Messentfernung mindestens das 10 fache der maximalen linearen Ausdehnung des Messobjektes betragen muss.
Ein Lambert-Strahler ist eine Lichtquelle, bei der die Leuchtdichte für jede Richtung konstant und die Lichtstärke kosinusförmig vom Betrachtungswinkel abhängig ist.
Wenn die zu vermessende Lichtquelle einem Lambert-Strahler nicht hinreichend genau entspricht, kann die oben genannte Gleichung zur Bestimmung der photometrischen Grenzentfernung nicht benutzt werden. In diesem Fall gibt es jedoch eine praktische Möglichkeit zur Bestimmung der photometrischen Grenzentfernung. Es gilt, dass die photometrische Grenzentfernung erreicht ist, wenn sich das Produkt aus dem Quadrat des Abstandes und der Beleuchtungsstärke mit zunehmendem Messabstand nur noch im Rahmen des zugelassenen Fehlers ändert. Dieses Verfahren wird bevorzugt bei komplizierten

Lichtquellen verwendet, da so die detaillierten Kenntnisse des Lichtstärkeverteilungskörpers für die Berechnung der photometrischen Grenzentfernung nicht benötigt werden.

Da die Lichtleiter bei den ambienten Beleuchtungssystemen so konstruiert werden, dass sie eine bestimmte Abstrahlcharakteristik vorweisen, kann man diese nicht mit einem Lambert-Strahler vergleichen. Aus diesem Grund kann man die oben genannte Formel für unsere Zwecke nicht einsetzen. Man kann jedoch aus den oben genannten Punkten schließen, dass die Messentfernung mindestens das 10 fache der maximalen linearen Ausdehnung betragen muss, um außerhalb der photometrischen Grenzentfernung messen zu können. Die Verfahren, die eine Messung außerhalb der photometrischen Grenzentfernung erfordern, gehören zum Feld der Fernfeldphotometrie.

Es gibt allerdings auch Messverfahren, die eine Messung innerhalb der photometrischen Grenzentfernung zulassen. Diese Verfahren gehören zu der Gruppe der Nahfeldphotometrie. Der Vorteil dieser Methoden liegt darin, dass die Abmessungen des Messaufbaus deutlich kleiner werden können.

Diese Messverfahren werden aber im Rahmen dieser Arbeit nicht genauer untersucht.

5 XML und der Produktlebenszyklus von ambienten Beleuchtungssystemen

5.1 Der Produktlebenszyklus von Ambienten Beleuchtungssystemen

5.1.1 Grundlagen

In diesem Unterkapitel wird der Produktlebenszyklus allgemein definiert um Aufschlüsse auf den Entwicklungsprozess von ambienten Beleuchtungssystemen aufnehmen zu können.

Der Begriff Produktlebenszyklus ist die Übersetzung des gebräuchlichen amerikanischen Begriffs Product lifecycle. Und wird als der Weg eines Produktes von der Rohstoffgewinnung bis zur Entsorgung definiert. Trotz obiger Definition wird der Begriff des Produktlebenszyklus unterschiedlich interpretiert, je nachdem, ob mit "Produkt" ein genau identifizierbarer Gegenstand (z.B. der "Kölner Dom"), oder eine Serie (z.B. "Lichtleiter") gemeint ist.

Während bei der Produktserie die Bezeichnung "Zyklus" fraglos gerechtfertigt erscheint, da sie mehrmals die unterschiedlichen Phasen durchläuft, muss man bei konkreten Gegenständen verallgemeinern (z.B. Bauwerke im allgemeinen), um nicht von einem einmaligen Lebensweg sprechen zu müssen.

Für den Produktlebenszyklus werden gemeinhin folgende Phasen angegeben:

1. Entwicklung, Planung
2. Produktion, Bau
3. Inbetriebnahme, Vermarktung
4. Betrieb, Vertrieb
5. Einstellung des Vertriebs, Außerdienststellung, Außerbetriebnahme, Stilllegung
6. Rückbau, Einstellung der Serviceleistungen
7. Entsorgung

Das Konzept des Produktlebenszyklus unterstellt, dass jedes Produkt und jede Dienstleistung beginnend mit der Markteinführung einen bestimmten Lebenszyklus durchläuft. Diesen unterteilt man in die fünf Phasen Einführung, Wachstum, Reife, Sättigung und Degeneration. Die Dauer der einzelnen Phasen sowie die Höhe des Umsatzes wird maßgeblich von den Marketingaktivitäten bestimmt. Mit dem Lebenszykluskonzept hat das Management ein Modell zur Verfügung, das die Zusammenstellung eines phasengerechten Marketingmixes unterstützt. Darüber hinaus erlaubt es realitätsnähere Prognosen der Absatzzahlen und Umsätze, als es beispielsweise die lineare Regression tun würde.

Die Einführungsphase ist durch hohe Investitionen für Aufbau von Produktion und Vertrieb sowie hohe Stückkosten durch geringe Produktionsmengen charakterisiert. Das Produkt ist meist eine Neuheit und hat einen geringen Bekanntheitsgrad. Der Umsatz steigt daher nur langsam an. Das Ende der Phase ist durch Erreichen der Gewinnschwelle gekennzeichnet.

Das Unternehmen muss das Kundenverhalten genau beobachten. Bei ungünstiger Entwicklung ist entweder der Ausstieg aus dem Markt oder Intensivierung der Marketingaktivitäten notwendig. Wichtigstes Ziel der Einführungsphase ist es, die sogenannte "kritische Masse" zu erreichen. Damit ist eine ausreichende Anzahl von Kunden gemeint, die die Akzeptanz des Produktes am Markt sicherstellt und es in die Wachstumsphase überführen. Tritt dies nicht ein, stirbt das Produkt bereits zu diesem Zeitpunkt ab, da es sich z.B. nicht gegen Alternativprodukte durchsetzen konnte.

In der Wachstumsphase steigt der Umsatz stark an und das Gewinnmaximum wird erreicht. Der Bekanntheitsgrad des Produktes steigt und bringt damit erste Wettbewerber auf den Plan. Die Marktstruktur entwickelt sich zum Oligopol. In dieser Phase kommt es darauf an, weitere Marktanteile hinzuzugewinnen und den Wettbewerbsvorteil zu erhalten. Dies wird vorrangig über Produktverbesserungen erreicht.

Während der Reifephase nehmen die Gewinne wieder ab und das Umsatzwachstum verlangsamt sich. Durch neu eintretende Konkurrenten bildet sich eine polypolistische Marktstruktur heraus und der Verdrängungswettbewerb setzt ein. Ein weiteres Wachstum kann nur mit sehr großen Anstrengungen erreicht werden. Am Ende dieser Phase ist das Umsatzmaximum erreicht.

Da der gestiegene Wettbewerbsdruck ohnehin zur Erosion der Gewinnmarge führen wird, ist in dieser Phase eine agressive Preispolitik kontraproduktiv. Ziel der Marketingmaßnahmen muss die Verteidigung der Marktanteile sein, da die Gewinnung neuer Kunden nur zu Lasten der Wettbewerber und daher nicht ohne Gegenwehr erfolgen wird. Mit weiteren Produktverbesserungen kann das Unternehmen diese Phase verlängern oder durch Bildung von Produktvarianten neue Märkte erschließen.

Stagnierendes bzw. negatives Umsatzwachstum charakterisiert die Sättigungsphase. An deren Ende wird die Verlustschwelle erreicht. Kurzfristig erzielbare Gewinne sollten realisiert werden, mittelfristig ist ein Ausstieg aus dem Markt sinnvoll. Das Unternehmen sollte jetzt die Marketingaktivitäten zurückfahren, um die Erträge in dieser Phase zu erhöhen. Darüber hinaus sind Kostensenkungen durch effizientere Produktionsverfahren und Skaleneffekte möglich.

In der Degeneration schließlich werden nur noch Verluste erwirtschaftet und die Umsätze sind stark fallend. Andere Unternehmen bringen Produkte an den Markt, die dem eigenen überlegen sind. Gleichzeitig scheiden auch andere Anbieter aus dem Markt aus, so dass die Marktstruktur wieder oligopolistisch wird. Die einzig sinnvolle Option für das Unternehmen besteht im Ausstieg aus diesem Markt.

Allerdings kann es auch bei sinkenden Absatzzahlen noch profitable Produkte geben, nämlich dann, wenn alle Mitbewerber aus dem Markt ausgeschieden sind. Dann kann der letzte Anbieter die verbliebene Nachfrage bedienen, wobei die Produktionsverfahren bereits optimiert sind und keine größeren Investitionen notwendig sind.

Die Abbildung *Abb. 5.1* soll die oben genannten Begriffe näher erläutern und schematisch darstellen.

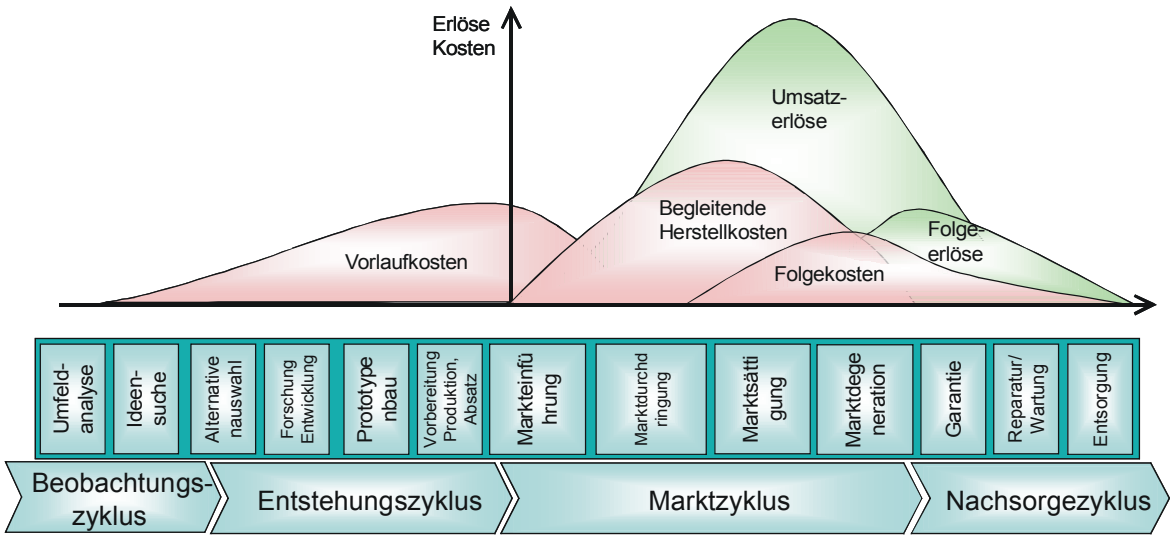

Abb. 5.1: Der Produktlebenszyklus

Der oben dargestellte Verlauf ist idealtypisch und die wenigsten Produkte oder Dienstleistungen folgen diesem Verlauf exakt. Zudem sind die Zeitpunkte der Phasenübergänge schwierig zu erkennen. Wenn eine Analyse nach dem Produktlebenszyklus-Konzept vorgenommen wird, sollte man unbedingt darauf achten, dass man genau abgrenzt, was das "Produkt" ausmacht. Damit wird vermieden, dass man am Ende "Äpfel mit Birnen" vergleicht und zu falschen Ergebnissen kommt.

5.1.2 Der Produktlebenszyklus von ambienten Beleuchtungssystemen

Aneinandergereiht oder verknüpft ergeben die Prozesse als Glieder die Entwicklungsprozesskette von ambienten Beleuchtungssystemen, an deren Anfang gewöhnlich die Planung, die durch eine Kundenanfrage, eine Produktidee oder eine Marktanalyse der Marketingabteilung entsteht, steht, und an deren Ende das fertige Produkt (*Abb. 5.2* und *Abb. 5.3*).
Jeder Geschäftsprozess als kleinste Einheit hat je einen oder mehrere Vorgänger und Nachfolger und steht mit diesen in einer engen Beziehung.
Der vom Lieferantenprozess erzeugte Umfang an Output, der an Kundenprozess (als dessen Input) übergeben wird, bezeichnet man als Leistungsobjekt(e). Dies stellt – aus prozessorientierter Sicht – die Quelle des Prozesselements dar. Durch den Transformationsprozess mit den Parametern Aktivitäten, Zeit, Ressourcen und Information erfährt das Leistungsobjekt einen Wertzuwachs.

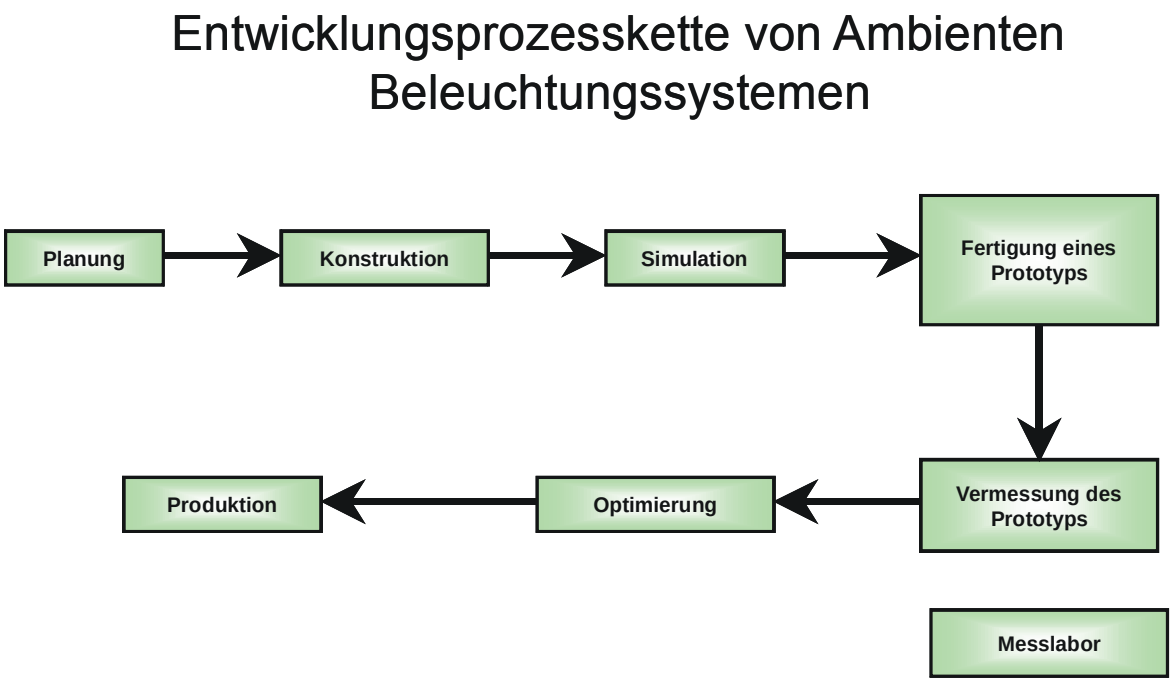

Abb. 5.2: Entwicklungsprozesskette von ABS

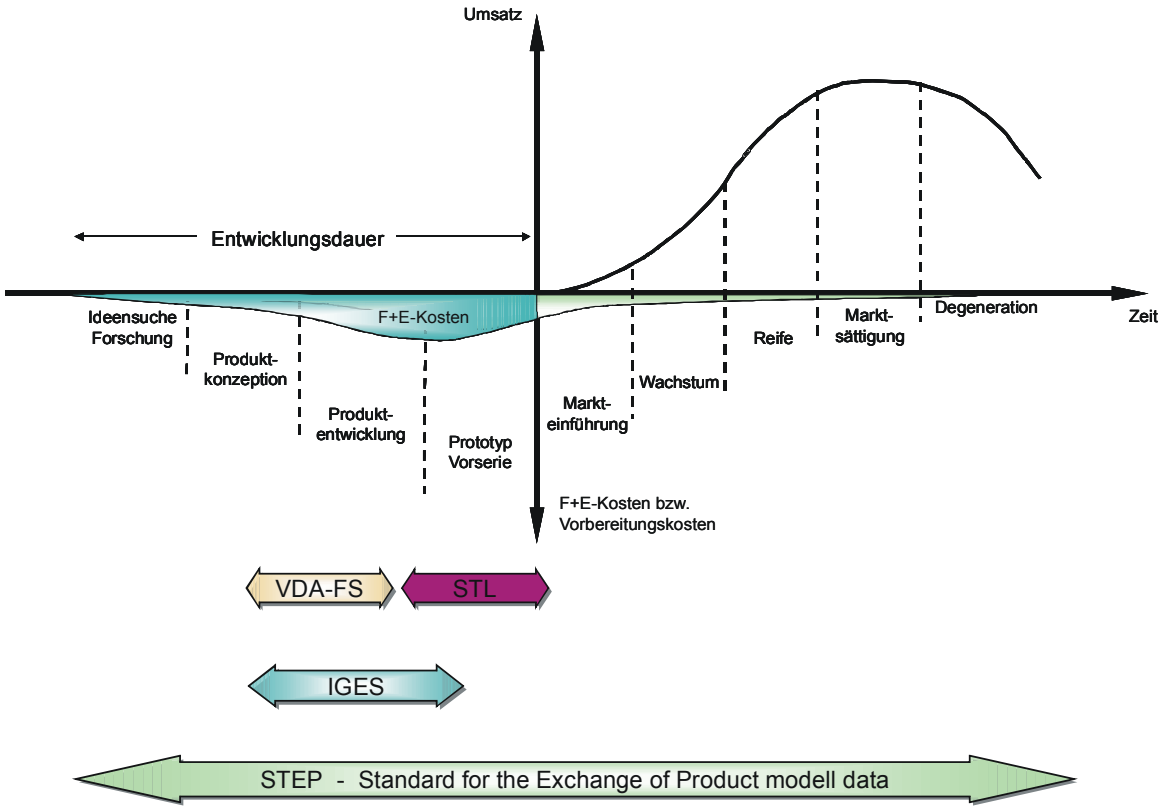

Abb. 5.3: Der Produktlebenszyklus von ABS mit den verschiedenen Datenformaten

In jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen fallen viele Daten unterschiedlicher Datentypen und unterschiedlicher Datenformate auf. Was die Notwendigkeit des XML-Konverters, der u.a. den Datenaustausch zwischen den einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen reibungslos ermöglicht, begründet.
Die Abbildung *Abb. 5.4* gibt einen gesamten Blick auf die unterschiedlichen Daten, die in jeder Phase auffallen.

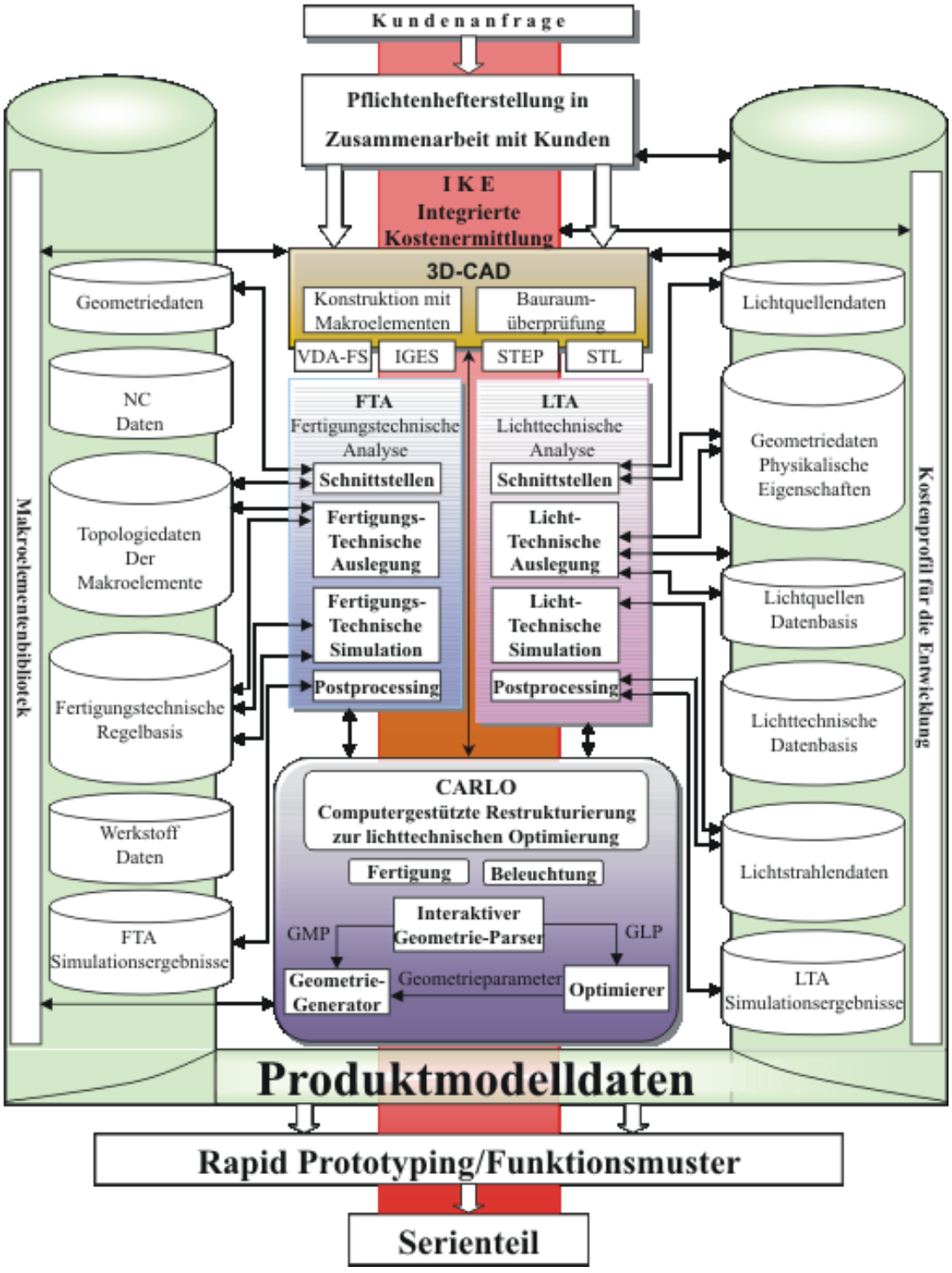

Abb. 5.4: Der Produktlebenszyklus von ABS mit seinem Datenfluss

Die lichttechnische Analyse von Beleuchtungssystemen und das damit verbundene Werkzeug ist im industriellen Einsatz kein „Stand alone“-Verfahren sondern ein Glied der gesamten Produktentwicklungskette. Das heißt, dass die optische Simulationssoftware in das Gesamtkonzept „Entwicklungsprozess“ integriert werden muss und mit Programmen für die mechanische Konstruktion,

mit Software für die Fertigung und mit Programmen für die Optimierung der Gestalt und der Funktion eines optischen Systems Hand in Hand arbeiten muss. Bereits hier wird klar, dass den Programmschnittstellen besondere Bedeutung zukommt. In welches Umfeld eine lichttechnische Simulationssoftware eingebettet ist, zeigt das Schema eines typischen Entwicklungsprozesses in *Abb. 5.5*.

Wie bei den meisten Produkten läuft auch die Entwicklung optischer Systeme iterativ ab. In Abbildung *Abb. 5.5* ist dies durch eine Kreisdarstellung angedeutet. Dem aus kreativen Konstrukteursköpfen erwachsenen Prinzipentwurf eines optischen Systems folgt die mechanische Konstruktion der Bauteile, die mit üblichen CAD-Konstruktionsprogrammen in 3 räumliche Dimensionen umgesetzt wird. Eine Verkürzung der Konstruktionszeit und eine Verbesserung der Qualität wird hierbei durch die Verwendung von bewährten Makroelementen erzielt. Makroelemente sind bei der Lichtleiterentwicklung unter anderem Elemente zur Ein- oder Auskopplung des Lichts.

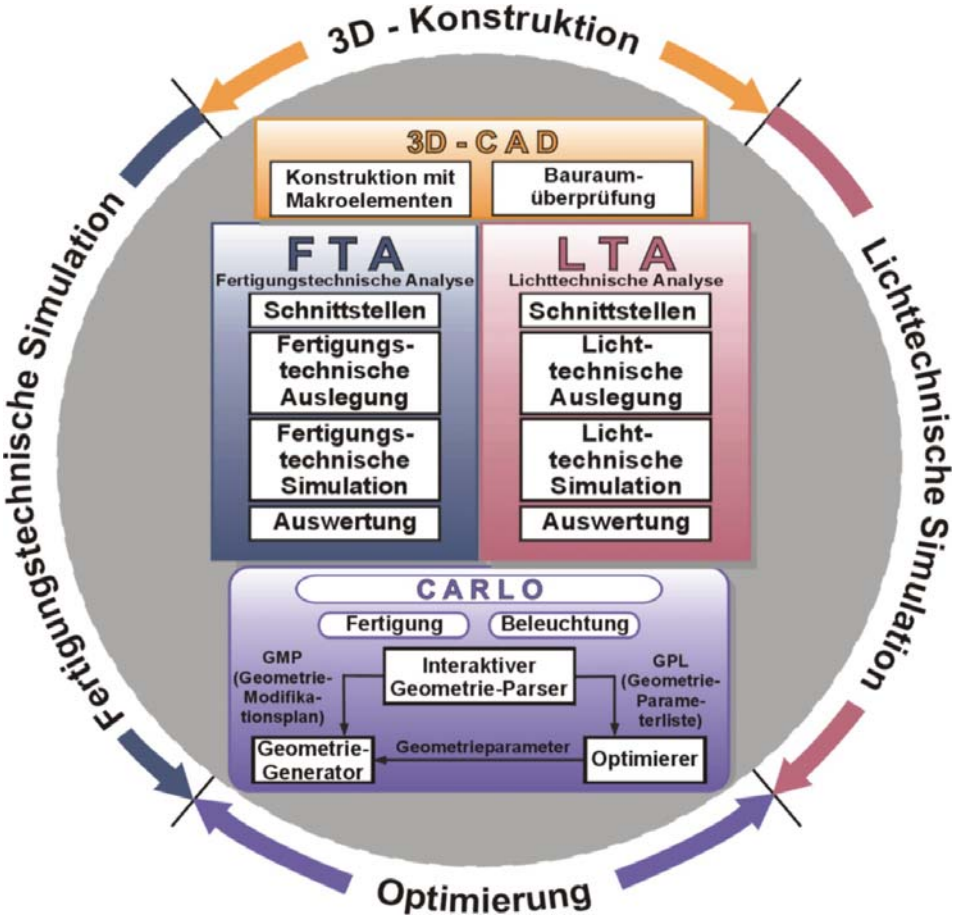

Abb. 5.5: Entwicklungsprozess von Beleuchtungssystemen

Die auf die CAD-Konstruktion aufbauende Entwicklungsphase spaltet sich auf in die parallel laufenden Stränge der lichttechnischen Analyse (LTA) und der fertigungstechnischen Analyse (FTA). Die LTA soll gewährleisten, dass die CAD-Konstruktion die ihr zugedachten optischen Funktionen erfüllt. Die FTA sichert die Herstellbarkeit der konstruierten Bauteile. Bauteiländerungen infolge FTA oder LTA beeinflussen sich gegenseitig stark. Man kann sich leicht vorstellen, dass einerseits eine für die optische Funktion theoretisch ideale Bauteilgestalt in der Praxis oft nicht hergestellt werden kann, andererseits ein einwandfrei zu fertigendes Bauteil die Optik stark beeinträchtigt – man denke nur an die bei Spritzgussteilen notwendigen Kantenradien. Bei Serienbauteilen kommen als weitere Faktoren Fertigungstoleranzen ins Spiel. Das bedeutet, dass die LTA nicht nur für eine einzige Gestalt des optischen Systems sondern für das gesamte Toleranzspektrum durchgeführt werden muss (Toleranzanalyse). Die aus FTA und LTA resultierenden Gestaltänderungen werden in der Optimierungsphase (in *Abb. 5.6* CARLO genannt) miteinander verheiratet. Optimierung bedeutet hier, dass die Geometrie des optischen Systems mit CAD so lange modifiziert wird, bis erstens die funktionelle Aufgabe des optischen Systems über die gesamte durch die Fertigung bestimmte Toleranzbreite bestmöglich erfüllt wird und zweitens die Bauteile möglichst einfach und kostengünstig herzustellen sind.

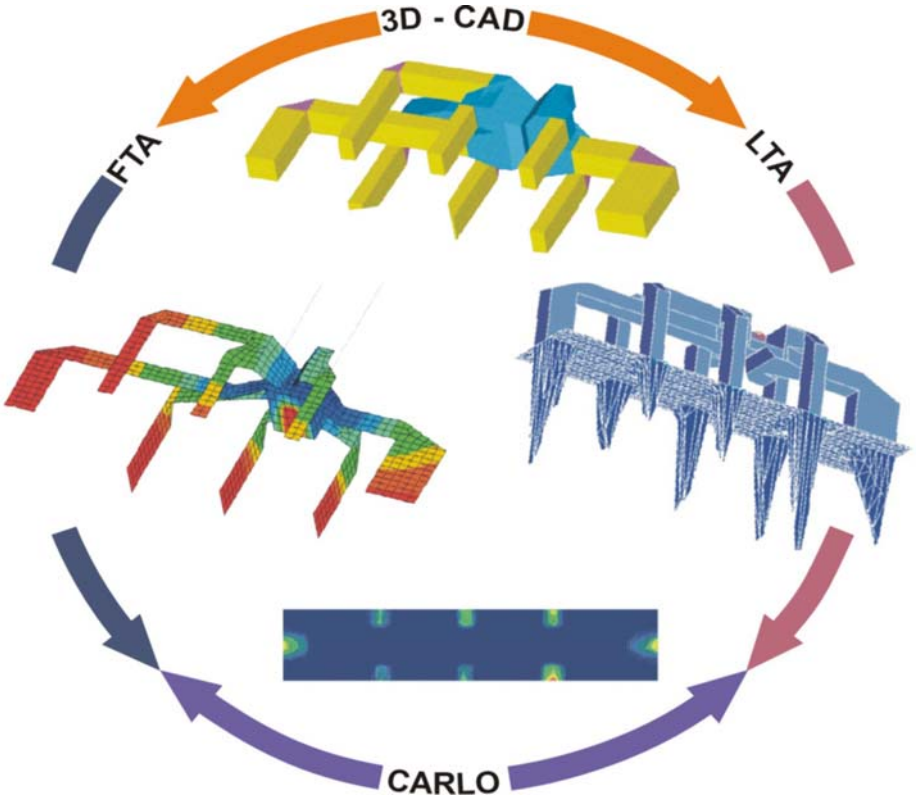

Abb. 5.6: Veranschaulicht werden die in Abb. 5.5 gezeigten Prozessschritte an einem Lichtleiter, der zur Tastaturbeleuchtung eines Autoradios entworfen wurde. Im Bild oben ist die 3D-Teilekonstruktion, links (FTA) der Fließfrontverlauf beim Spritzgießen und rechts (LTA) ein Gebirgediagramm sowie unten ein Kontourplot der Leuchtdichte über der Lichtauskoppelfläche nach mittels CARLO erfolgter Optimierung abgebildet.

Anforderungen an lichttechnische Software:

Der lichttechnischen Software fallen im beschriebenen Entwicklungsprozess die Aufgaben LTA und Toleranzanalyse zu. Über den Entwicklungsprozess hinaus dient die Software zur Präsentation von virtuellen Prototypen gegenüber Kunden. In der Automobilindustrie wird die Innenraumbeleuchtung und die Beleuchtung der Bedienkonsolen von Fahrzeugen in zunehmendem Maße als ästhetisches Gestaltungsmittel begriffen. Infolgedessen wird die Software in Zukunft auch für Designstudien des Gesamtinterieurs dienlich sein.
Neben dem mittlerweile beschriebenen äußeren Einsatzfeld für lichttechnische Software sollen im folgenden auch die eigentlichen internen Anforderungen an Software für die lichttechnische Analyse abgesteckt werden. Dazu ein kurzer Abriss, welche Aspekte beim Aufstellen der Anforderungen an optische Simulationssoftware berücksichtigt werden müssen. Dieser Abriss bietet nebenbei einen Einblick in das Leistungsspektrum optischer Software und ist beim Einordnen der Spezifika und Features eines Softwarepaketes hilfreich.
Eine der Anfangsfragen ist die nach der prinzipiellen Aufgabenstellung des optischen Systems.

- Beleuchtungssysteme lenken ihre Strahlen von einer Lichtquelle auf eine Zielfläche, wobei die Abbildung der Lichtquelle unerwünscht und meist eine bestimmte vorgegebene Ausleuchtung der Zielfläche gefordert ist.
- Systeme der abbildenden Optik führen Strahlen von einem leuchtenden oder beleuchteten Objekt zu einem Empfänger, wobei eine möglichst exakte Abbildung des Objekts gewünscht ist.
- Systeme für Photonik spielen in diesem Zusammenhang keine Rolle, sind aber hier der Vollständigkeit halber mit aufgeführt.

Ein anderer Anforderungskatalog deckt die benötigte Physik ab.

- Wie soll ein Lichtstrahl simuliert werden?
  Ein Lichtbündel bzw. Lichtstrahl kann prinzipiell durch seine Eigenschaften "Ort im Raum", „Richtung", „Strahldivergenz", „Strahldurchmesser", „Energieverteilung über den Strahlquerschnitt", „Phase", „Wellenlänge" und „Polarisation" beschrieben werden. Im Fernfeld, das heißt bei ebenen Wellenfronten, können folgende Einschränkungen getroffen werden. Bei der geometrischen Optik wird ein Lichtstrahl mit infinitesimalem Querschnitt gekennzeichnet durch die Eigenschaften „Ort im Raum", „Richtung", „nominelle Leistung", und „Farbe". Für Beleuchtungssysteme genügt die geometrische Optik. Bei der Wellenoptik wird ein Lichtstrahl als Normale auf ebene Wellenfronten mit infinitesimalem Querschnitt gedacht und gekennzeichnet durch die Eigenschaften „Ort im Raum", „Richtung", „Energie", „Phase", „Wellenlänge" und „Polarisation". Teilcheneigenschaften des Lichts wurden bis heute seltener benötigt und sind deshalb nur in Spezialprogrammen, meist im Forschungsumfeld zu finden. Die Berücksichtigung der Teilcheneigenschaften des Lichts wird notwendig, falls Strahlung oder die Wechselwirkungen zwischen Licht und Materie, wie Absorption, technisch in Detektoren genutzt, oder die Emission, technisch genutzt in Lichtquellen, genauer betrachtet werden sollen.
- Welche Wechselwirkungen zwischen Licht und Materie werden benötigt?
  Absorption, Reflexion, Totalreflexion, Transmission, Brechung, Dispersion, Streuung (an Oberflächen/ im Volumen), Beugung.

Außerdem ist die Frage nach den erforderlichen Methoden zur Strahlverfolgung (Ray Tracing) zu stellen. Beim Ray Tracing wird der Weg eines einzelnen Strahls, ausgehend von der Lichtquelle zu jeder Grenzfläche die er trifft, berechnet, und dies solange bis er absorbiert wird oder nicht weiter verfolgt werden kann. Unterschieden wird zwischen sequentiellem und nicht-sequentiellem Ray Tracing. Beim Sequentiellen liegt der Strahlverlauf von vornherein fest, womit die Grenzflächen, die der Strahl trifft, in einer fest vorgegebenen Reihe folgen – beispielsweise in einer Linse. Sequentielle Strahlverläufe sind relativ einfach und

schnell zu berechnen. Bei der nichtsequentiellen Methode steht die Reihenfolge der getroffenen Grenzflächen nicht von vornherein fest. Der Weg eines Strahls muss aktiv verfolgt werden um festzustellen ob er irgendeine Grenzfläche trifft. Bei Treffern wechselwirkt er mit der Grenzfläche, seine Eigenschaften ändern sich und werden neu berechnet. Ausgehend vom Auftreffpunkt wird dann die nächste Grenzfläche gesucht. Nichtsequentielle Strahlverfolgung ist relativ speicher- und zeithungrig.

Weitere Anforderungen ergeben sich aus den Zielgrößen die analysiert werden sollen. Unterschieden wird zwischen physikalischen Größen und photometrischen. Nicht jedes Simulationspaket kann beide Gruppen berechnen. Die Photometrie (Unterkapitel 4.2) interessiert sich nicht für die physikalischen Eigenschaften der Lichtstrahlung allgemein, sondern nur für den Teil davon, der von unserem Auge wahrgenommen wird. Werden physikalische Größen auf die spektrale Empfindlichkeit des menschlichen Auges bezogen, spricht man von photometrischen Größen. Photometrische Größen sind also physiologische Größen. Aus einer physikalischen Größe, beispielsweise dem auf die Fläche bezogenen Strahlungsfluss in [W/m²], wird so die in Lux [lx] gemessene photometrische Größe Lichtstromdichte, auch Lichtintensität genannt. Sämtliche Beleuchtungssysteme werden mit photometrischen Größen beschrieben.

In manchen Softwarepaketen sind die virtuellen Abbilder von Fertig- oder Makrobauteilen sowie Materialbibliotheken über Herstellerkataloge eingebunden. Hier ist die Rede von Lichtquellen, Strahlteilern, Reflektoren, Spiegeln, Polarisatoren, Empfängern etc. Der Einsatz solcher Bibliotheken erspart erstens aufwendige Detailkonstruktionen und ermöglicht zweitens die schnelle Variantenkonstruktion eines optischen Systems.

Für die eigentliche Analyse stellen die Programmpakete dem Konstrukteur eine Vielzahl von Hilfsmitteln, von einfachen Ergebnisplots über aufwendige Grafiken bis zu ausgefeilten mathematischen Algorithmen, zur Verfügung. Wie hier die Schwerpunkte zu setzen sind, obliegt dem Geschmack des Anwenders. An dieser Stelle wird nur auf einige „Goodies“ verwiesen, die von manchen Softwareherstellern angeboten werden. Für Serienbauteile sehr hilfreich ist eine von der Software unterstützte Toleranzanalyse. Gleichermaßen hilfreich ist die Berechnung der Auswirkungen von thermischen (Änderung der

Umgebungstemperatur oder Erwärmung einzelner Bauteile durch die Bestrahlung) und mechanischen (Druck, Spannungen, Schwingungen) Störungen. Ebenfalls sehr ansprechend sind Echtfarbdarstellungen, die eine „virtual-reality“ Darstellung von Beleuchtungssystemen für Präsentationszwecke ermöglichen.

Die im Vorangehenden aufgeführten Aspekte umreißen die Möglichkeiten, Simulationssoftware für allgemeinere Anwendungsfälle zu charakterisieren. Daneben gibt es natürlich noch Software für Spezialanwendungen, wie aktive lichterzeugende Elemente, Laser, Faserleiter, dünne Schichten, Detektoren etc.

Last but not least sind noch Anforderungen an die Gestaltung der Benutzerschnittstelle, Erweiterbarkeit, Nutzung des Programms zu Dokumentationszwecken und Programmierbarkeit über Makros oder freie Programmiersprachen zu stellen. Schließlich soll der Anwender in der Lage sein, mit seinem täglichen Werkzeug zu arbeiten.

Bei den meisten lichttechnischen Programmen werden komplette Beleuchtungssysteme aus aktiven (Lichtquelle: LEDs oder Miniaturglühlampen) und passiven optischen Elementen (Lichtleiter, Filterplatten, Reflektoren, Absorber) simuliert. Zur Analyse von Beleuchtungssystemen genügen die Methoden der geometrischen, hier allerdings nichtabbildenden, Optik, Wellen- oder Teilcheneigenschaften des Lichts spielen keine Rolle. An Wechselwirkungen zwischen Licht und Materie sollen Absorption, Reflexion, Totalreflexion, Brechung, Streuung an Oberflächen und im Volumen implementiert sein. Strahlverfolgung soll sowohl sequentiell als auch nichtsequentiell möglich sein.

Analysiert werden ausschließlich photometrische Größen. Kataloge für Lichtquellen, insbesondere für LEDs, sind erwünscht. Das gilt ebenso für eine softwareunterstützte Toleranzanalyse sowie Echtfarbdarstellung für virtual reality Szenarien. Wichtig für den industriellen Einsatz des Softwarepakets ist die benötigte Dauer und die Genauigkeit einer Simulation.

Nachdem in den obigen Abschnitten der Aufbau des Produktlebenszyklus von ambienten Beleuchtungssystemen sowie die Anforderungen an die lichttechnischen Software erläutert wurden, wird im nächsten Abschnitt die Datenverwaltung, die entlang des Produktlebenszyklus von ambienten Beleuchtungssystemen im Rahmen dieser Arbeit geschieht, vorgestellt.

Der XML-Konverter stellt einen Brückenschlag zwischen den Einzelanwendungen dar und hat unter anderem die Aufgabe, den Austausch von Daten über Grenzen von Applikationen, Betriebsystemen und Hardwareplattformen zu ermöglichen. Er stellt jedem Benutzer und seiner Anwendung rechtzeitig und im nötigen Umfang und Abstraktionsgrad Daten zur Verfügung.

Gewöhnlich erfolgt anfangs die Sammlung vieler Daten und Informationen auf Papier und deren erste datentechnische Umsetzung im CAD-System (*Abb. 5.7*). Die vom CAD-System erzeugten Geometriedateien werden lokal im Austauschformat auf Festplatte gespeichert und von dort, nach dem Wechsel der Applikation, wieder geladen. Die Ergebnisse werden gesichtet und aus diesen Änderungen am Entwurf abgeleitet (Verbesserungsvorschläge). Jede Einzelapplikation gibt ihre „Datenpakete“, oder eben nur Teile davon, an die nächste weiter. Dieser Umstand wird in *Abb. 5.7* verdeutlicht, wo die CAD-erzeugte Geometrie zwar an die lichttechnische Analyse vollständig als ‚Datei’ weitergegeben werden kann, an die fertigungstechnische Analyse jedoch nur einzelne Daten.

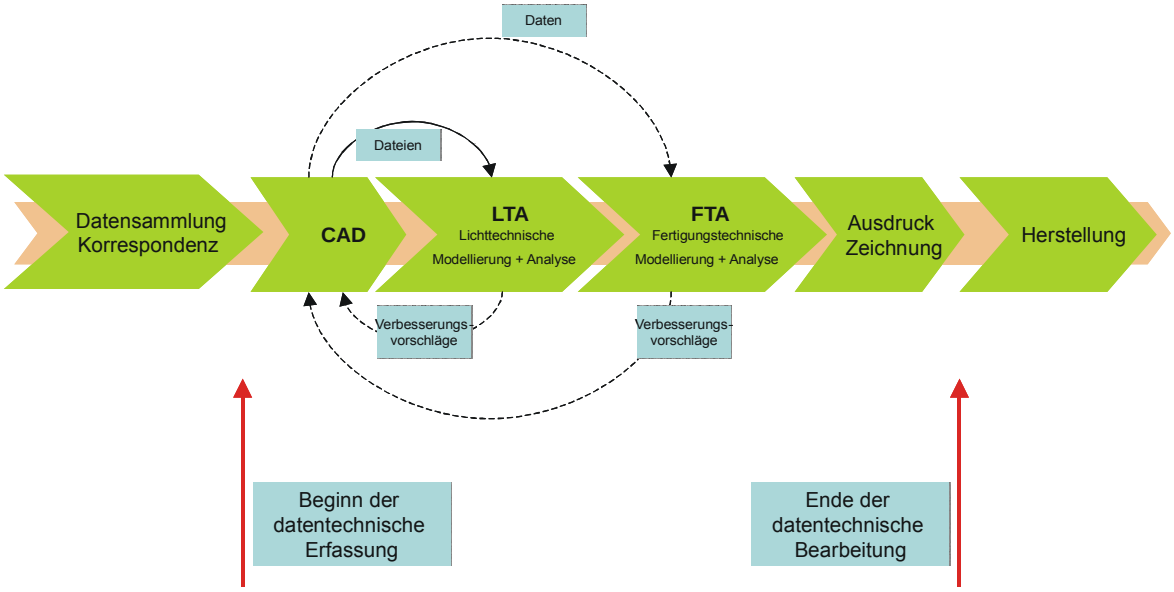

Abb. 5.7: Beginn und Ende der Datenbearbeitung

In der durchgängigen Softwareunterstützung gehen Daten einen anderen Weg. Der hier entwickelte XML-Konverter verlangt für die Datenintegration, dass alle Daten des Projektes über die Datenbank stets verfügbar sind. Von jeder Applikation, d.h. von jedem an der Produkterstellung beteiligten, informationsverarbeitenden Prozess, müssen sie gelesen und/oder bearbeitet und anschließend, ggf. verändert, zurückgereicht werden können. Voraussetzung hierfür ist ein Datenmodell oder Produktdatenmanagementsystem (PDMS), das, zugeschnitten auf das jeweilige Produkt oder Produktpalette, die benötigten Informationen für alle relevanten Applikationen zugänglich macht (*Abb. 5.8*). grundsätzlich enthält das PDMS alle zum Projekt / Produkt gehörenden oder im Zusammenhang mit diesem erzeugten Daten, jeder Prozess kann prinzipiell auf alle zuvor gewonnen Informationen zugreifen.

Vor dem Erzeugen von Geometrien muss die Erfassung von Daten treten, die das zukünftige Produkt zunächst hauptsächlich qualitativ beschreiben und Schritt für Schritt mit quantitativen Daten vervollständigen. Dies erfordert eine frühe Eingabe der Daten in das Computersystem, macht aber dadurch die einmal erfassten Informationen stets zugänglich und wiederverwendbar. Neue Informationen können sofort auf diesen aufbauen. Zu den rein technischen Beschreibungen treten administrative hinzu, auch kalkulatorische, um stets alle Daten ‚beisammen' zu haben, um deren Konsistenz, d.h. Wiederspruchsfreiheit sicherzustellen.

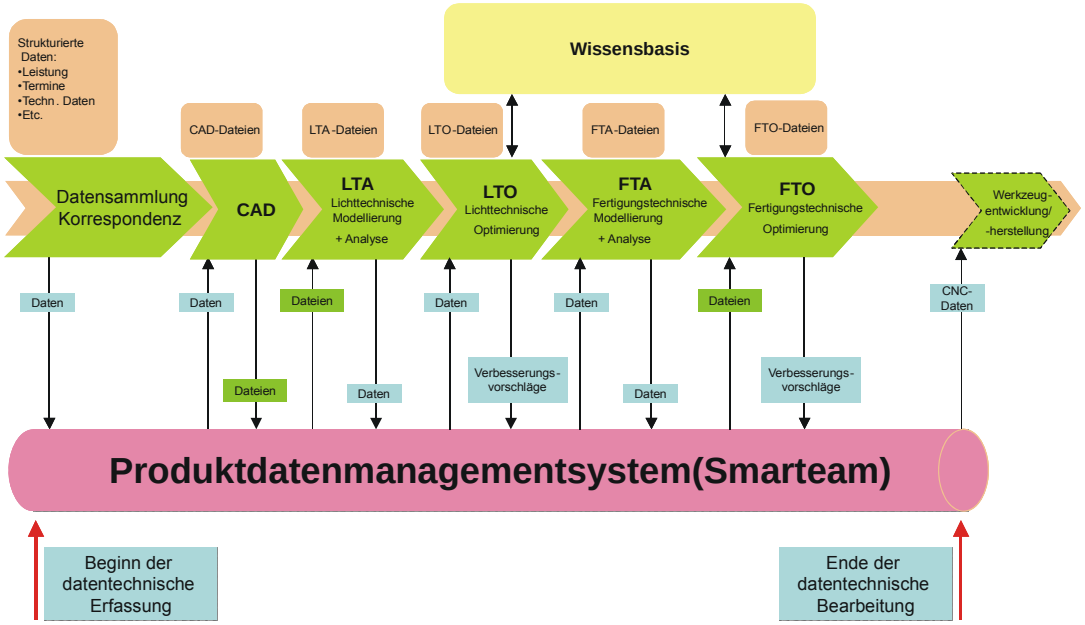

Abb. 5.8: Durchgängige Datenverwaltung

5.1.3 Anforderung an die XML-Dokumentenstrukturen der einzelnen Phasen des Produktlebenszyklus von Ambienten Beleuchtungssystemen

Die geeignete Strukturierung von Informationen stellt oft eine gewisse Herausforderung dar. Es lohnt sich, die gewählte Strukturierung zu hinterfragen und allfällige Varianten bezüglich ihrer Konsequenzen zu analysieren. Nur so hat man eine gute Einsatzmöglichkeit und somit erfolgreiche Ergebnisse. Es werden hier einige Fachbegriffe verwendet, deren Erläuterung wichtig ist, damit der Aufbau sowie das Prinzip des XML-Konverters besser verstanden kann.
In diesem Unterkapitel sollen die Anforderungen an die XML-Dokumentenstrukturen der einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen untersucht werden. Es wird hier ein gangbarer Weg für die Entwicklung von den XML-Dokumentstrukturen der Daten einzelner Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen gezeigt, um den Datenaustausch zwischen den einzelnen Phasen des Produktlebenszyklus zu ermöglichen und um den Produktlebenszyklus von ambienten Beleuchtungssystemen in die verteilte Produktentwicklung integrieren zu können.
Es sollten alle Abläufe und die Organisation der Daten, die in jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen, bzw. ihren Modulen klar beschrieben und ggf. variiert werden. Es ist hier auch wichtig zu erwähnen, dass die Daten, die in jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen, ganz unterschiedliche Datentypen und Datenformate sind (Geometriedaten, Lichttechnische Daten, ...). Für die spätere Umsetzung (softwaretechnisch) egal in welche Programmiersprache, wäre das eine fast unlösbare Problematik, durch diese Vorgehensweise und durch den Einsatz des XML-Konverters hat man eine elegante Lösung, die die spätere Implementierung so leicht macht wie noch nie zuvor.
Um diese Untersuchung erfolgreich durchführen zu können, ist die folgende Fragestellungen wichtig:

- Welche Daten werden wann und wo erzeugt?

- Welche Daten werden wo und in welcher Form (Format) gespeichert?
- Welche Daten werden wann und wo benötigt?
- Welche Forderungen sind an das Datenmodell für einen Informationensaustausch zu stellen?
- Welche Anwendungen auf welcher Plattform sind vorhanden bzw. werden benötigt?
- Welche Hardware- und welche Softwareschnittstellen treten die Anwendungen zum Datenaustausch in Kontakt?

Ausgehend von der beschriebenen Situation soll über eine Neustrukturierung der Datenhaltung der Einsatz von Informationstechnik ermöglicht werden. Damit werden Verfügbarkeit und Wiederverwendbarkeit von Daten und Informationen im Rahmen des Produktlebenszyklus sowie zwischen den unterschiedlichen Plattformen sichergestellt, was Mehrfacharbeit und Liegezeiten aufgrund von Informationsdefiziten vermeidet und so letztendlich zur Beschleunigung der Produktrealisation führt.
Die vorliegende Arbeit beinhaltet das Konzept einer Softwareunterstützung des Produktlebenszyklus von ambienten Beleuchtungssystemen. Es werden folgende Vorgehensweise gewählt:

1. Untersuchung aller auffallenden Daten in allen Phasen des Produktlebenszyklus
2. Identifizierung von Datenwegen und Datentypen
3. Gestaltung einer prozessübergreifenden Datenstruktur.

Es werden vor allem lichttechnische Daten in der lichttechnische Simulationsphase, geometrische Daten in der 3D-Konstruktionsphase und in der fertigungstechnischen Simulationsphase und lichttechnische sowie geometrische Daten in der Optimierungsphase und normale Daten in allen Phasen berücksichtigt.
Da im Rahmen dieser Arbeit unter anderem eine prozessübergreifende Datenstruktur angestrebt wird, sind XML-Dokumente, wie aus den Kapiteln Kap.2 und Kap.3 leicht erkannt werden kann, genau das richtige, weil eine direkte

Speicherung als XML möglich ist, weil Abfragen durch XML Anfragesprachen (wie z.B. Xquery und XPath) auch möglich sind, weil Programmierschnittstellen (wie z.B. SAX und DOM) existieren, weil XML-Dokumente Wiederherstellbar sind, weil eine schnelle und sichere Bearbeitung der Daten durch den Einsatz von XML gewährgeleistet ist und weil Mehrbenutzerbetrieb möglich ist. Der Einsatz des XML-Konverters deckt den gesamten Produktlebenszyklus von ambienten Beleuchtungssystemen von der Planungsphase bis zur Herstellungsphase ab. In der folgenden Abbildung *Abb.5.9* ist die XML-basierte Beschreibung des XML-Konverters schematisch dargestellt.

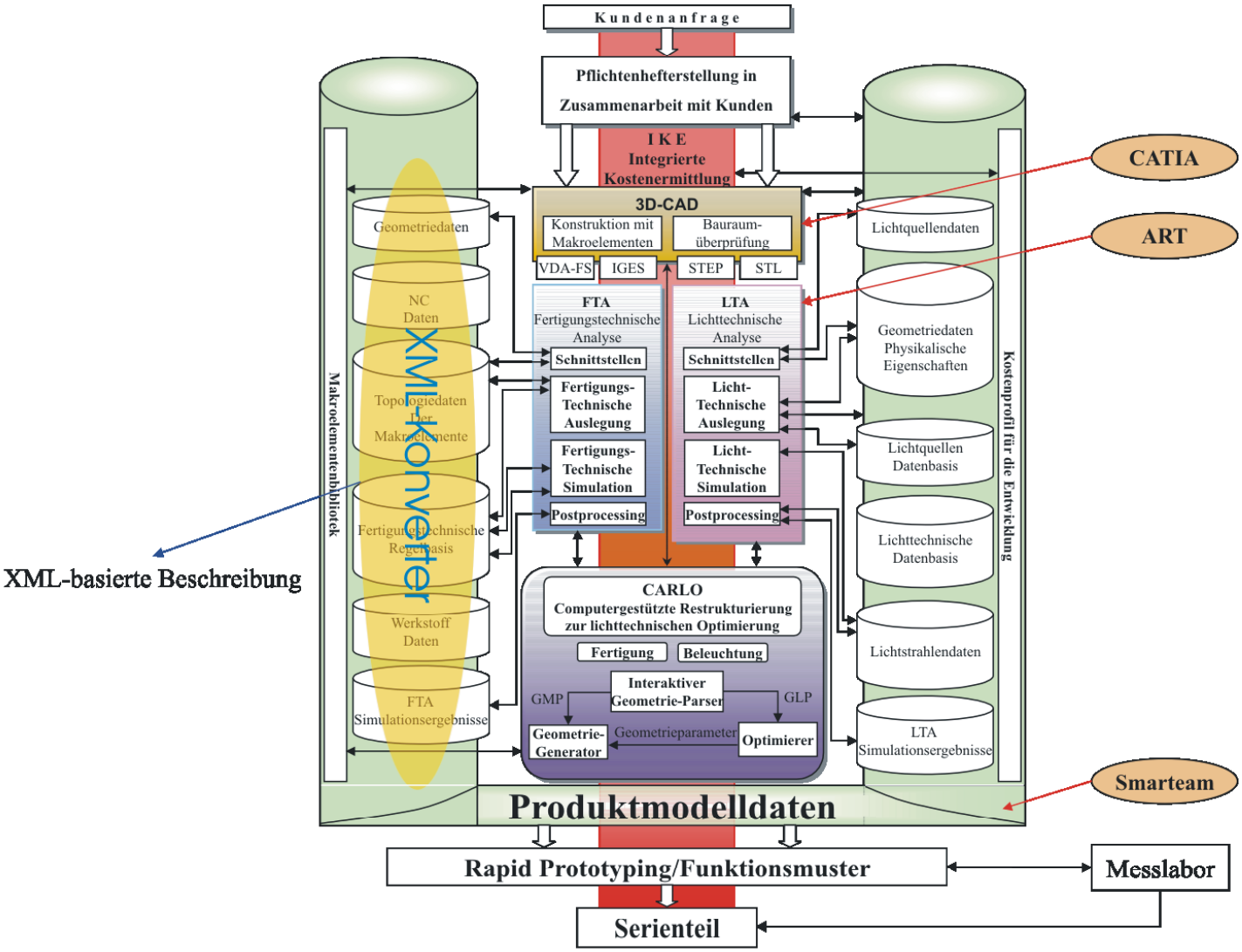

Abb. 5.9: Die XML-basierte Beschreibung

Bei der in dieser Arbeit benutzten Vorgehensweise werden für jede Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen mehrere XML-Dokumentenstrukturen, die die spezifischen Eigenschaften der jeweiligen Phase berücksichtigen, erzeugt. Es werden z.B. in der 3D-Konstruktionsphase (CAD-Phase) hauptwiegend Geometriedokumente erzeugt, in der Lichttechnischen Simulationsphase werden dagegen hauptwiegend Lichttechnische Dokumente erzeugt und in der Fertigungstechnischen Simulationsphase werden Lichttechnische Dokumente sowie Geometriedokumente erzeugt. Der XML-Konverter ist so konzipiert, dass er alle Daten, die bei jeder Phase des Produktlebenszyklus auffallen, automatisch bearbeitet. Denn nur so ist der Datenaustausch zwischen den einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen reibungslos garantiert. Für die verteilte Produktentwicklung ist diese Vorgehensweise auch gerade das richtige, es können von jedem Beteiligten Datenzugriffe auf einzelne Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen unternommen werden. In der Abbildung *Abb.5.10* ist das Prinzip des Erzeugens der XML-Dokumentenstrukturen durch den XML-Konverter schematisch dargestellt.

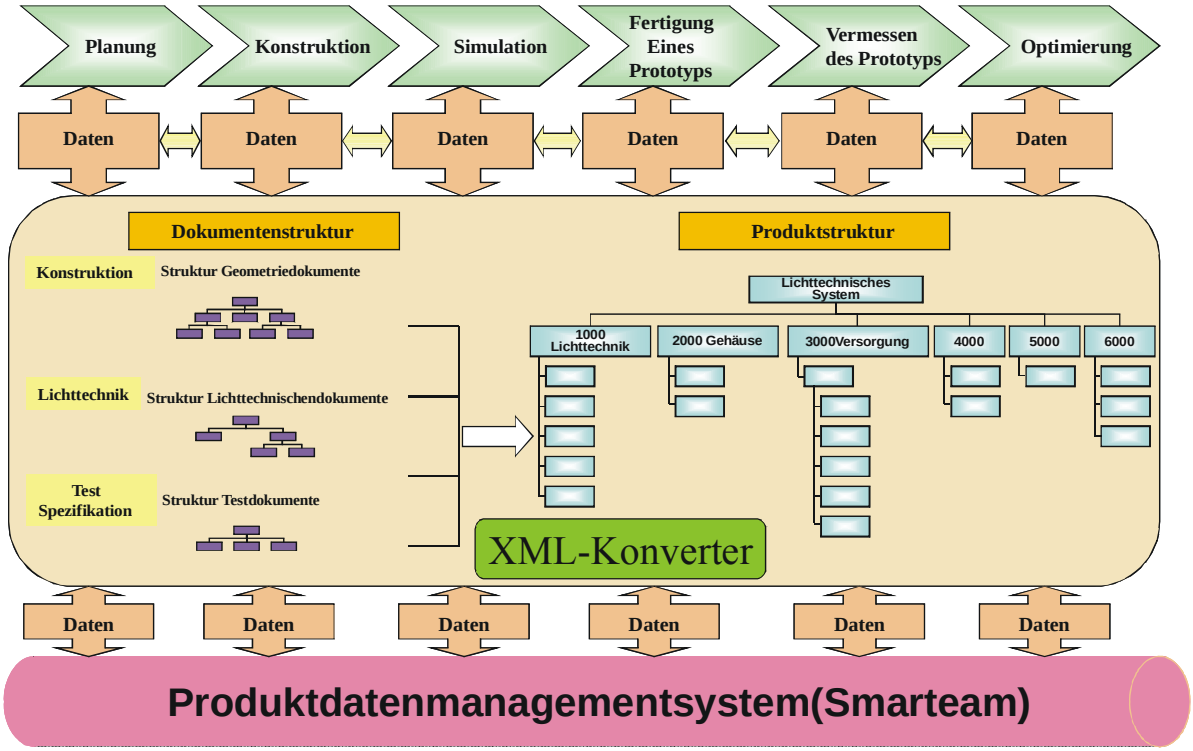

Abb. 5.10: Das Prinzip des Erzeugens der XML-Dokumentenstrukturen

Wie aus der obigen Abbildung zu sehen ist, werden in der linken Seite der Abbildung die XML-Dokumentenstrukturen erzeugt, aus diesen XML-Dokumentenstrukturen wird wie auf der rechten Seite der Abbildung das lichttechnische System von ambienten Beleuchtungssystemen (*Abb. 5.11*) erzeugt.
Das lichttechnische System von ambienten Beleuchtungssystemen besteht aus 6 wichtigen Bestandteilen:

1. Lichttechnik
2. Gehäuse
3. Versorgung
4. Steuerung und Regelung
5. Assembling
6. Baugruppen zur Reduzierung von neben Effekten

Und jeder dieser Teile besteht wiederum aus weiteren wichtigen Teile, wie man auf der Abbildung *Abb. 5.11* sehen kann.
In jeder Phase auffallen viele Datenmengen unterschiedlicher Typen und unterschiedlicher Datenformate, weil am Entstehungsprozess eines technischen Dokumentes unterschiedliche Informationen Texte, Grafiken, Bilder, Zeichnungen, technische Daten und andere Produktinformationen beteiligt sind. Der XML-Konverter legt die Informationen als Bausteine medienneutral in einer XML-Dokumentenstruktur ab und stellt sie an jedem Beteiligten bereit. Durch diese Vorgehensweise wird immer auf die aktuellen und korrekten Informationen zugegriffen.
Der XML-Konverter erzeugt strukturierte Dokumente, die aus verschiedenen Informationsobjekten bestehen. Es können mehrere dieser Bausteine zusammengefasst und als beliebige Sinneinheiten klassifiziert und gespeichert werden und als solche in anderen Dokumenten wieder verwendet werden. Die Objekte werden bezüglich der Produktstruktur, des Kontextes innerhalb der Dokumente und nach Art und Inhalt der Informationen strukturiert. Dies bedeutet eine konsequente Trennung von Inhalt und Layout sowie von Dokumenten- und Produktstruktur.

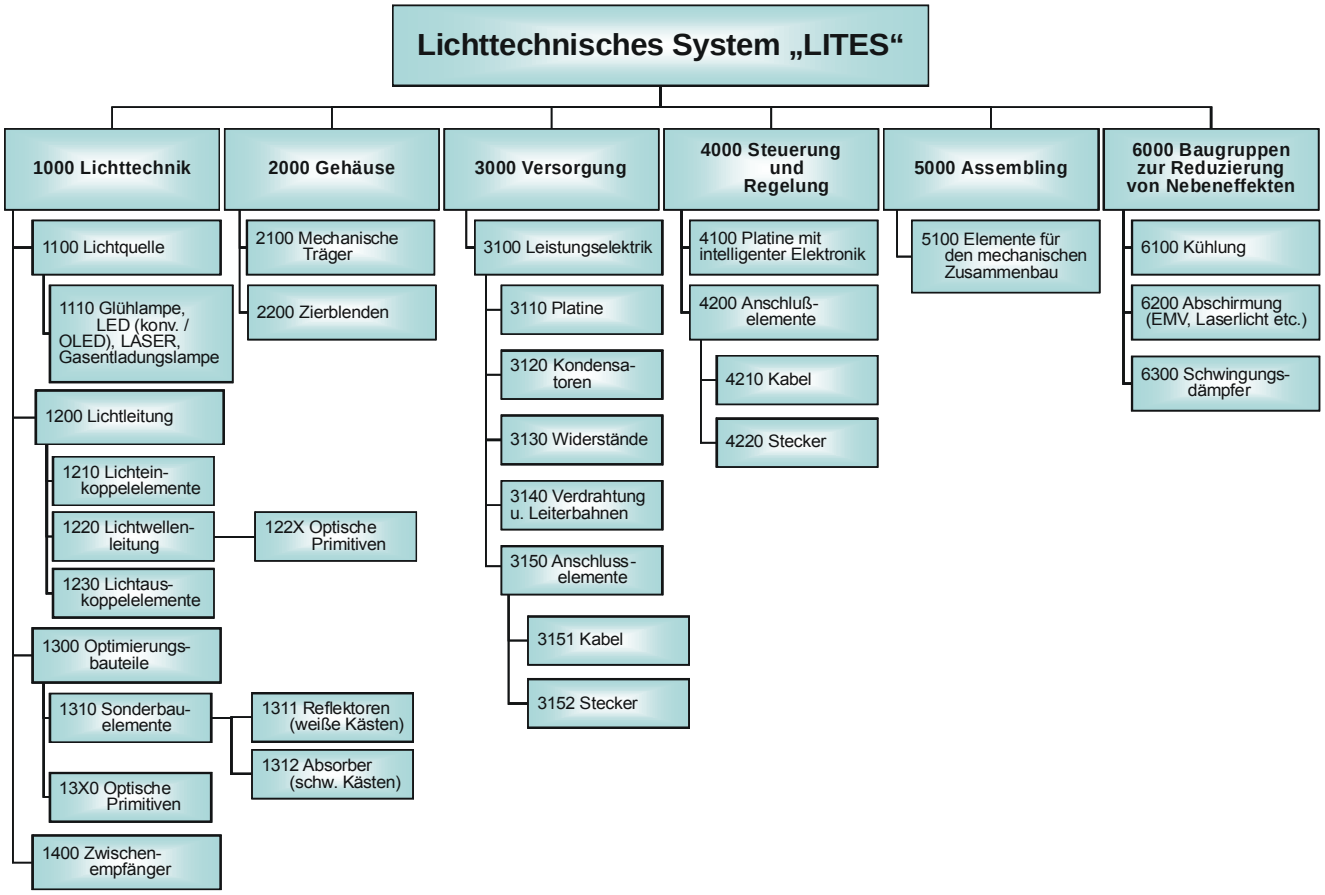

Abb. 5.11: Das Lichttechnische System von ABS

5.2 XML-Konverter von Ambienten Beleuchtungssystemen

5.2.1 Funktionsweise des XML-Konverters

Bei dem XML-Konverter wird von Anfang an die Idee des „store once, use everywhere“ verfolgt, dies bedeutet: es werden die Daten und Dokumente, die in jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen, unabhängig von ihren unterschiedlichen Datentypen und unabhängig von der späteren Anwendung in einem XML-Dokument gespeichert und werden erst beim konkreten Anliegen in das gewünschte Ausgabeformat generiert.
Diese Vorgehensweise hilft dabei nicht nur Zeit zu sparen und den reibungslosen Datenaustausch zwischen den verschiedenen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen sondern sie hilft auch dabei den gesamten Produktlebenszyklus von ambienten Beleuchtungssystemen in die verteilte Produktentwicklung zu integrieren. Man speichert Texte, Daten, Unterlagen, Grafiken, Geometriedaten, lichttechnische Daten, u.s.w. in XML-Dokumenten und modifiziert diese Daten an zentraler Stelle. Dadurch reduziert man Fehler, vermeidet doppelte Arbeit und Redundanzen und verringert die Kosten für Übersetzung und Speicherung in mehrere Formate –wie die Abbildung *Abb. 5.12* zeigt-, die „just in time“ bei Gebrauch aus den XML-Dokumenten erstellt werden. Die XML-Dokumente kann man dann mit Hilfe von dem XML-Konverter und Transformationssprachen in beliebige Ausgabeformate umwandeln. Ausgabeformate wären zum Beispiel Webausgabe in HTML, der Druckausgabe in PDF und usw., man kann auch zahlreiche andere Formate aus XML erzeugen.
Diese Vorgehensweise bei dem XML-Konverter ermöglicht es, dass die ambienten Beleuchtungssysteme für web-basierte Systeme plattformunabhängig fähig gemacht werden, sie ermöglicht es auch, dass die ambienten Beleuchtungssysteme erweiterbar und für neue Problemstellungen anpassbar sind. Davon ausgehend können auch ohne große Mühe neue Publishing-Plattformen erstellt werden.

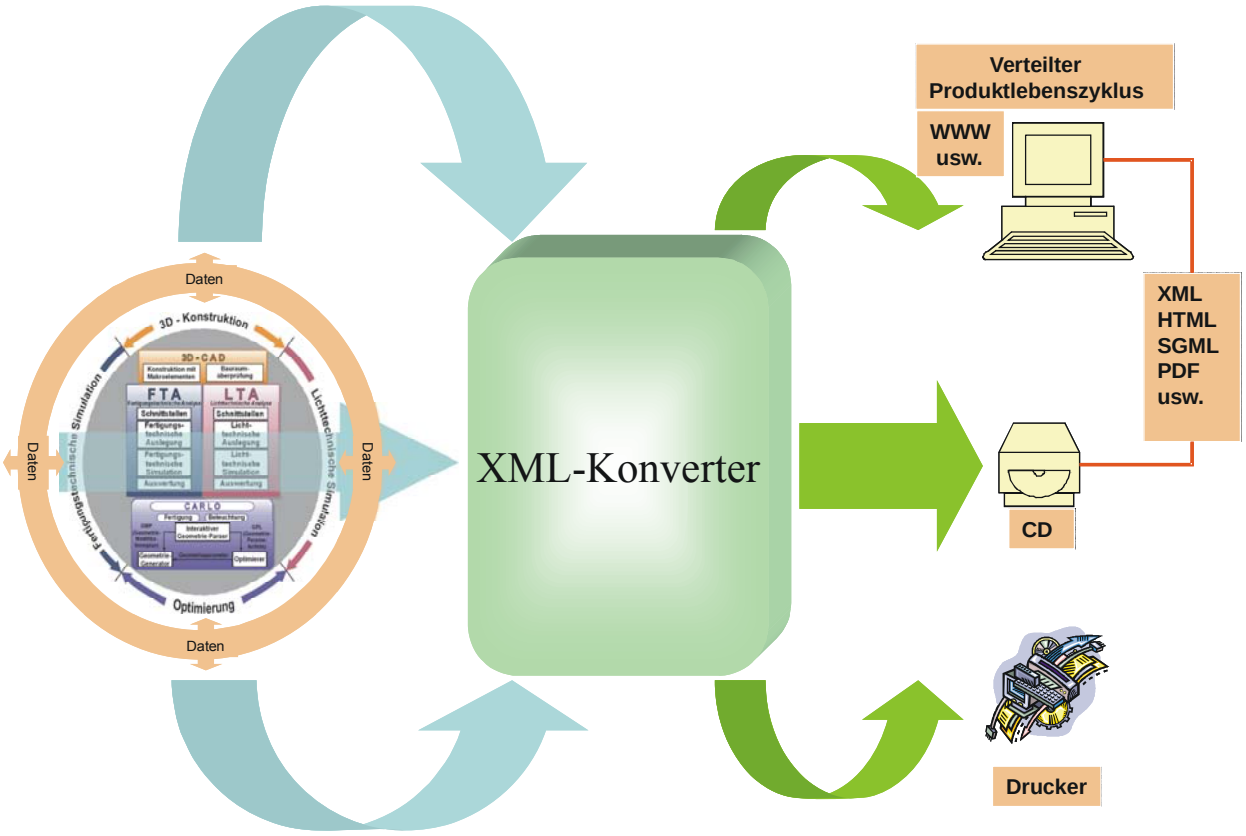

Abb. 5.12: Prinzip des XML-Konverters

Der XML-Konverter stellt für web-basierte Systeme eine Plattform dar. Die Komponenten des XML-Konverters bieten Schnittstellen um vorhandene ABS-Systeme zu erweitern und an neue Problemstellungen anzupassen oder neue Publishing-Plattformen zu erstellen. Im nächsten Unterkapitel(Kap. 5.2.2) wird der Aufbau des XML-Konverters detailliert vorgestellt. Der XML-Konverter wurde so konzipiert, dass die Erstellung von hochwertigen XML-Lösungen für den Produktlebenszyklus von ambienten Beleuchtungssystemen gewährgeleistet wird. So stellt er eine Austauschplattform für ABS-Projekte dar und eine der wichtigsten Teilaufgaben des XML-Konverters ist es, die Integration der ambienten Beleuchtungssysteme in die verteilte Produktentwicklung sowie die dynamische Gestaltung von Webseiten. Angeforderte Ressourcen werden je nach Anfrage für die Nutzung aufbereitet. Ein wesentliches Konzept des XML-Konverters ist der

modulare Aufbau mit Standardschnittstellen, so dass einzelne Komponenten ausgetauscht werden können. Der XML-Konverter baut auf dem Prinzip vom Single-Source-Publishing auf.

Das Prinzip des Single-Source-Publishings besteht darin, dass die zugrunde liegenden Daten Format unabhängig in einer gemeinsamen Quelle verwaltet werden.

Mit dem XML-Konverter ist die Erstellung von Webseiten für den Zugang zu der verteilten Produktentwicklung leicht gemacht. Man möchte eine Separation of Concerns (SoC), d. h. Trennung von *Inhalt*, *Logik* und *Darstellung* bei der Entwicklung von Webseiten erreichen. Die Abbildung *Abb.5.13* zeigt einen Überblick über diese Aufgabenteilung.

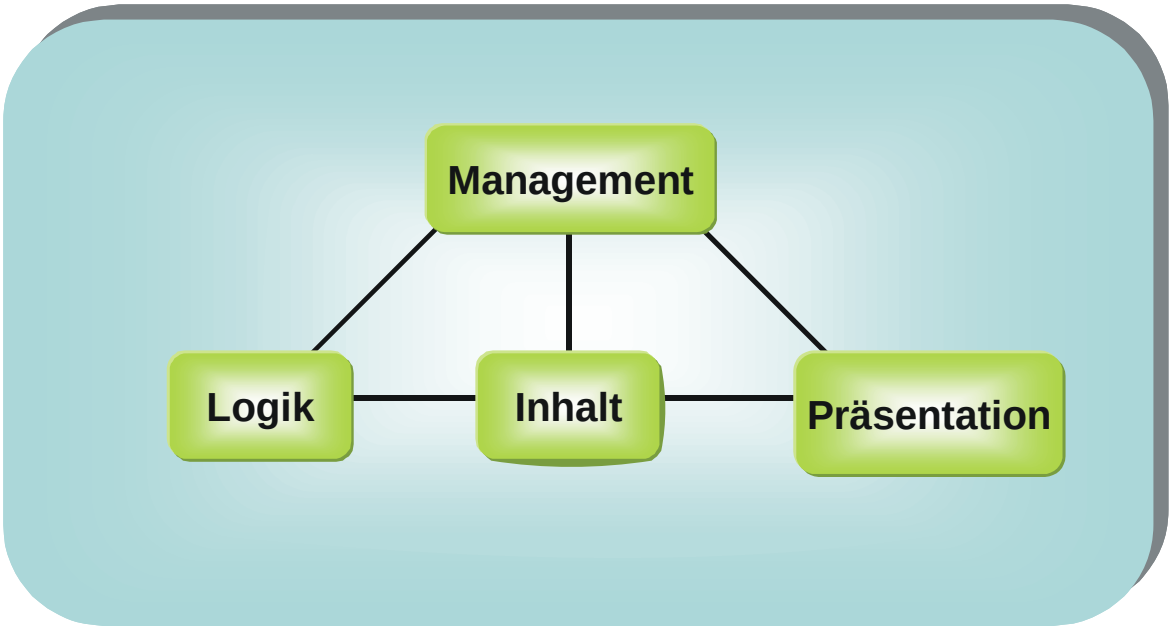

Abb. 5.13: Separate Entwicklung von Webseiten für verteilte Produktentwicklung sowie für den Datenaustausch zwischen den Phasen von ABS

Es soll eine klare Trennung der verschiedenen Arbeitsbereiche in einem Entwicklungsteam von Webseiten oder von der Produktentwicklung entstehen.

Das Management kümmert sich um den Ablauf, Aufbau aller Seiten und organisiert den gesamten Entwicklungsablauf.
Der redaktionelle Teil soll die Inhalte in einem XML-Dokument erstellen. Die Redakteure brauchen sich dabei nur auf den Inhalt-Teil zu konzentrieren und lassen alle Designentscheidungen bei der Entwicklung ihrer XML-Dokumente außer Acht. Eine weitere Arbeitsgruppe organisiert die dynamischen Inhalte einer Seite.
Dieser Bereich wird im Logik-Teil bearbeitet. Der dynamische Webinhalt wird mittels XSP (eXtensible Server Pages) entworfen. Der Präsentation-Bereich kümmert sich um die Präsentation und Grafik der Webseiten. Sie entwerfen hauptsächlich Stylesheets, um die XML-Dokumente zu transformieren.
Durch die Verwendung des XML-Konverters kann man nun auch leicht externe Datenquellen einbinden, wie z.B.: Filesysteme, native XML Datenbanken und netzwerkbasierte Quellen. Je nach Anfrage eines Clients kann das gewünschte Ausgabeformat der Webseiten geändert werden. Die XML-Dokumente lassen sich leicht in unterschiedliche Präsentationsformate überführen und ausliefern (in HTML, WML, PDF, SVG, RTF).
Um allerdings die Verarbeitung eines XML-Dokuments in dem XML-Konverter zu verdeutlichen wird auf den Verarbeitungsprozess des XML-Konverters eingegangen. Bei dem XML-Konverter kann man den Verarbeitungsprozess sehr plastisch und anschaulich darstellen und danach werden seine Komponenten erklärt.
Eines der Hauptziele bei der Entwicklung von dem XML-Konverter ist es, eine bessere Performance zu erreichen. Anstatt der Benutzung eines DOM-Parsers, der im Visier war, ist ein SAX-Parser verwendet worden, um den Speicherverbrauch zu verringern. Die Carbage Collection wird durch SAX-Verarbeitung ebenfalls geringer, weil nicht das komplette XML-Dokument im Speicher gehalten wird.
Außerdem wurde nun eine simultane Verarbeitung mehrerer Anfragen ermöglicht und das spät verwendete Framework ist skalierbar je nach Lastsituation. Die Antwort wird nun schon während der Bearbeitung gesendet. Man wartet nicht bis ein Auftrag komplett bearbeitet wurde, sondern sendet dem Client schon Zwischenergebnisse, wodurch natürlich die Performance der Verarbeitung

gesteigert wird. Bei dem XML-Konverter ist diese Vorgehensweise total erforderlich besonders bei inkrementeller Verarbeitung, wenn die Ergebnisse einer ersten Verarbeitung nochmals transformiert und verändert werden müssen. Bei dem XML-Konverter gibt es eine Erkennung von wiederverwendbarem Code. Durch diese „Hot Spot“-Erkennung wird eine schnellere Verarbeitung durch Speicherung relevanter Codefragmente erreicht.

5.2.2 Aufbau des XML-Konverters

Der XML-Konverter ist so entwickelt, dass beliebige Frameworks, die in verschiedene Sprachen programmiert sind, Anwendungen finden, um in einer Verarbeitungskette verschiedene Prozessoren einzusetzen, ähnlich einer Pipe, die man sich als einen röhrenartigen Datenkanal vorstellen kann (*Abb. 5.14*). Im folgenden wird das Bearbeitungsprinzip des XML-Konverters, das auf dem Konzept von Pipelines basiert, erläutert. Der Informationenaustausch zwischen den Prozessoren muss gewährgeleistet werden obwohl sie (die Prozessoren) unabhängig voneinander sind.
Die Kommunikation zwischen den Prozessoren kann mit verschiedenen Methoden erreicht werden.

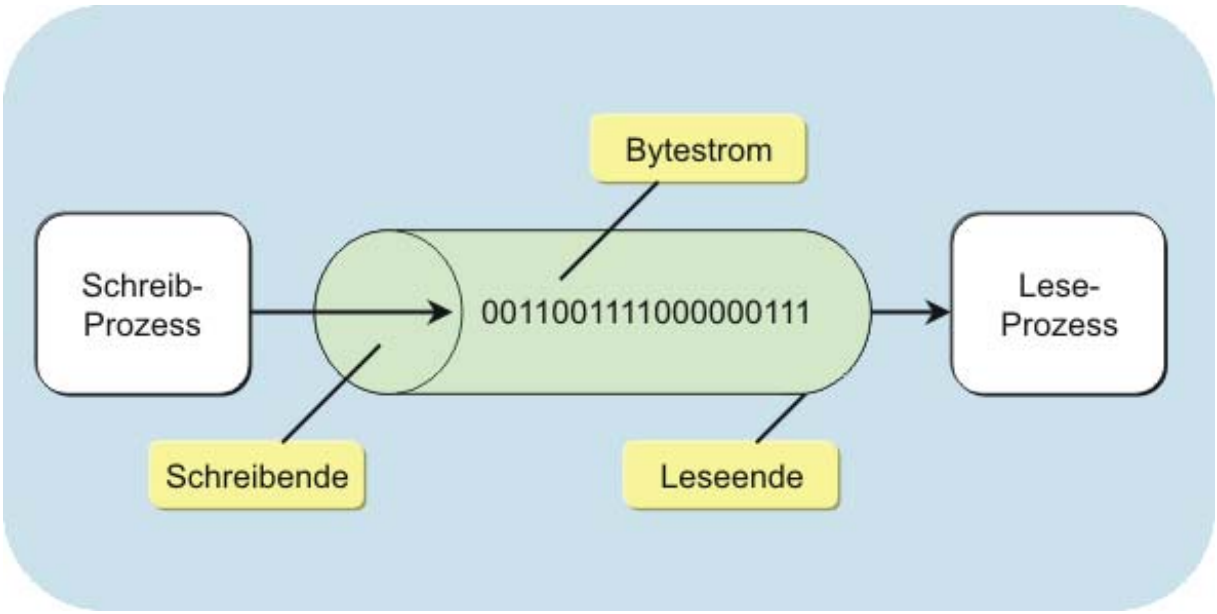

Abb. 5.14: Pipebearbeitungsprinzip bei dem XML-Konverter

Ein Prozess schreibt die Daten in die Pipe und ein anderer Prozess kann diese Daten in der Reihenfolge auslesen, in der sie vom anderen Prozess geschrieben wurden. Die meisten Pipes sind unidirektional, so dass die Daten nur in eine Richtung übermittelt werden. Eine Pipe hat für einen Prozess das Aussehen einer Datei, auf die er schreibt oder liest. Außer dem Positionieren kann darauf jede

Dateioperation erfolgen. Beim Lesen und Schreiben muss allerdings auf die Größe des Pipe-Buffers geachtet werden. Dieser Pipe-Buffer ist abhängig vom System (meistens 4kB oder 8kB groß). Prozesse, die mit einer Pipe arbeiten, werden in bestimmten Situationen vom System gesteuert. Ein Prozess, der aus einer leeren Pipe lesen will, muss warten, bis von einem anderen Prozess in die Pipe geschrieben wurde. Ein Prozess, der in eine Pipe schreiben will, muss warten, wenn der Pipe-Buffer voll ist.

Es gibt zwei Arten von Pipes, unbenannte und benannte. Die unbenannte Pipe (manchmal auch einfache Pipe genannt) hat einige Einschränkungen. Die Lebensdauer einer unbenannten Pipe ist abhängig von der Lebensdauer der Prozesse die mit ihr arbeiten. Sind alle Prozesse beendet, die mit der Pipe arbeiten, so wird die Pipe gelöscht. Es gibt meistens einen schreibenden und einen lesenden Prozess.

Die Kommunikation über eine unbenannte Pipe ist nur für Prozesse möglich, die mit einander "verwandt" sind. Dies gilt für Prozesse, die eine Vater Sohn Beziehung zueinander besitzen, für Sohnprozesse die den selben Vater haben und für Enkelprozesse. In allen Fällen richtet der Vaterprozess die Pipe ein.

Mit einem pipe() Aufruf besitzt ein Prozess eine Pipe zu sich selbst, aus der er mit fd[0] Daten lesen kann. Mit fd[1] kann er Daten in diese Pipe schreiben. Die Abbildung *Abb. 5.15* soll dieses Verhalten schematisch darstellen.

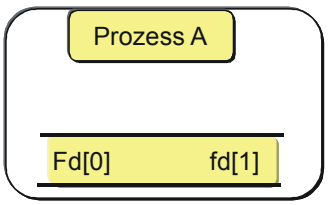

Abb. 5.15: Vaterprozess (Prozess A) richte Pipe ein

Diese Pipe erhält dann einen Sinn, wenn der Vaterprozess durch einen fork() Aufruf einen Sohnprozess kreiert, der mit dem Vaterprozess Daten austauscht. Dieser Sohnprozess erbt die Pipe seines Vaters. Die Abbildung *Abb.5.16* zeigt die Verwendung einer Pipe zwischen Vater und Sohn. Der Sohnprozess (Prozess B1) sendet Daten an den Vaterprozess.

Die Richtung des Datenstromes wird dadurch beeinflusst welcher Prozess die Lese-bzw. Schreibseite der Pipe schließt.
Mit dem Aufruf close(fd[0]) wird vom Sohnprozess die Leseseite der Pipe geschlossen. Der Vaterprozess schließt die Schreibseite der Pipe mit dem Aufruf close(fd[1]).

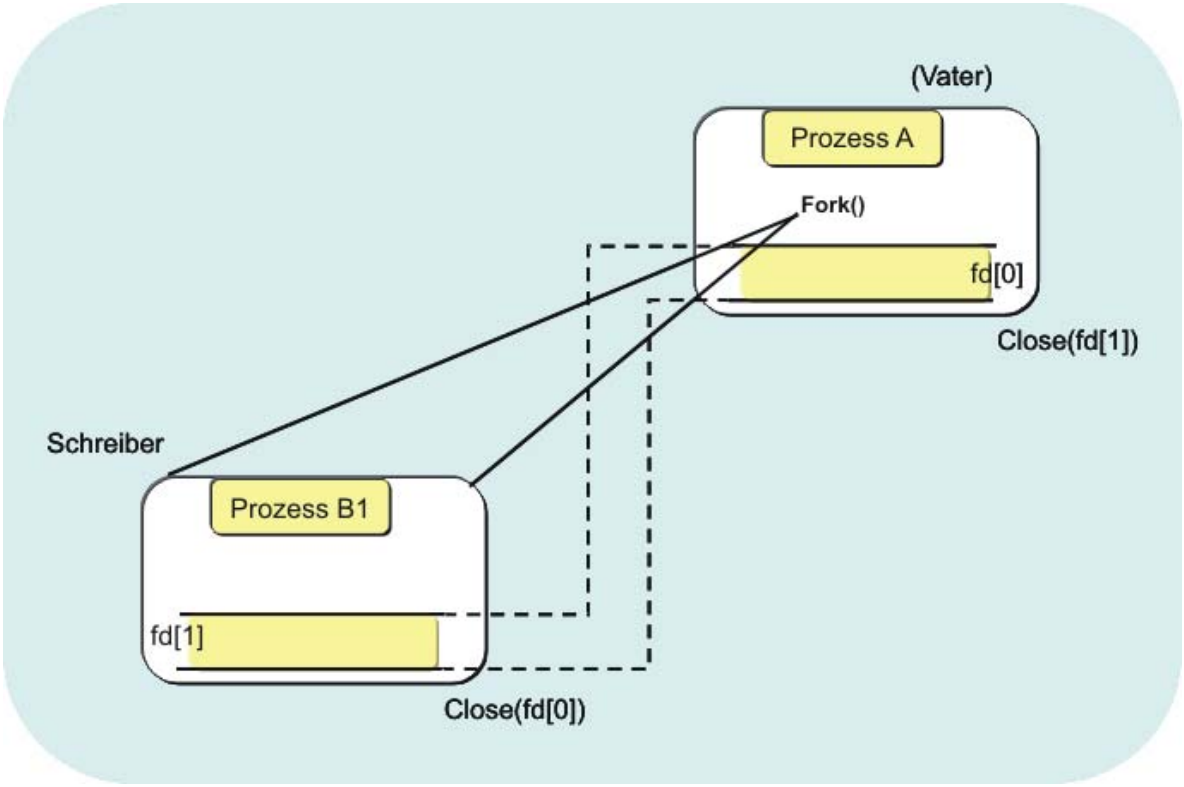

Abb. 5.16: Herstellen einer Pipe zwischen Vater und Sohn

Sollen zwei Söhne durch eine unbenannte Pipe miteinander kommunizieren, so müssen folgende Schritte ausgeführt werden. Die Abbildung *Abb. 5.17* stellt dieses Sachverhalten schematisch dar.

1. Vaterprozess richtet durch den Aufruf pipe() eine Pipe ein.
2. Der Vaterprozess kreiert durch fork() einen "Schreib-Sohn".
3. Der Vaterprozess schließt durch close(fd[1]) die Schreibseite der Pipe.
4. Der "Schreib-Sohn" schließt die Leseseite durch close(fd[0]).
5. Der Vaterprozess kreiert nun durch fork() einen "Lese-Sohn".

6. Der Vaterprozess schließt durch close(fd[0]) nun auch die Leseseite der Pipe.
7. Dieser "Lese-Sohn" schließt durch close(fd[1]) die Schreibseite der Pipe.

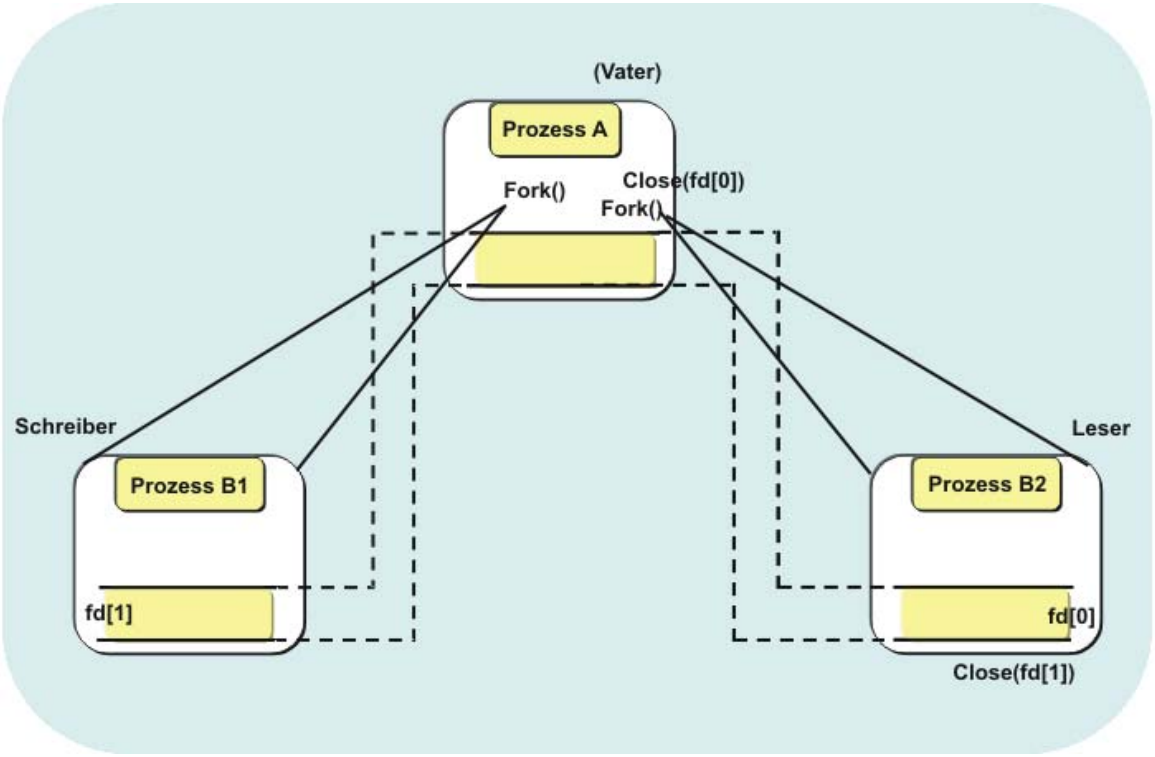

Abb. 5.17: Herstellen einer Pipe-Verbindung zwischen "Schreib-Sohn" und "Lese-Sohn"

Die so erstellte Pipe bildet nun eine Kommunikationsverbindung zwischen dem 1. Sohn (Schreibprozess) und dem 2. Sohn (Leseprozess). Der Vaterprozess hat nach dem Erstellen keinen Einfluss auf die Pipe, da er die Lese- und Schreibseite geschlossen hat.

Eine benannte Pipe ist eine Erweiterung gegenüber einer unbenannten Pipe. Sie besitzt einen angelegten Geräteeintrag vom Typ FIFO (**F**irst **I**n **F**irst **O**ut) und hat einen entsprechenden Namen, mit dem sie von jedem Prozess durch open() angesprochen werden kann. Dieser Name wird beim Aufruf des 1s –1 Kommandos angezeigt und durch ein p als Typenangabe gekennzeichnet. Eine benannte Pipe wird vom System nicht automatisch gelöscht, wenn alle Prozesse

beendet sind. Durch den Aufruf unlink() muss der Anwender die benannte Pipe innerhalb eines Prozesses selber löschen. Eine Löschung der benannten Pipe ist auch von der Kommandooberfläche durch den Befehl rm möglich.

Bei dieser Art der Kommunikation werden die Daten an Nachrichtenspeicher, sogenannte "Message Queues", gesendet und können dort von anderen Prozessen abgeholt werden. Dies wird durch ein Array verwaltet (Message-Queue-Tabelle), indem Daten über jede Message Queue stehen. Die Nachrichten bestehen aus einem Nachrichtenkopf (Message Header) und einem Nachrichtentext. Im Message Header sind Informationen, wie Typ, Größe der Nachricht und ein Zeiger auf den Speicherbereich, wo die Nachricht steht, enthalten.

Die Abbildung *Abb. 5.18* zeigt den Aufbau von Message Queues bei der internen Kommunikation in dem XML-Konverter.

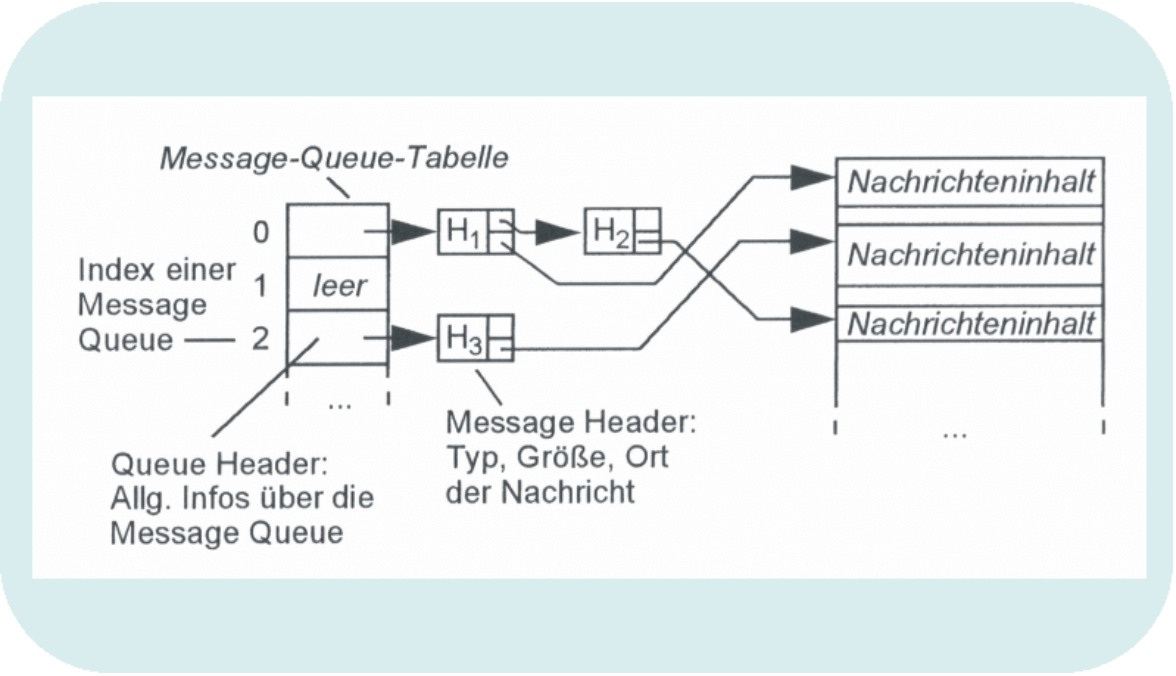

Abb. 5.18: Aufbau von Message Queues bei der internen Kommunikation im XML-Konverter

Bei dem XML-Konverter wird auch das Prinzip des Shared Memory (*Abb. 5.19*) benutzt. Bei diesem Prinzip benutzen die Prozesse einen gemeinsamen Speicherbereich auf den sie zugreifen können. Dieser Speicherbereich muss

durch das Betriebssystem gekennzeichnet bzw. registriert werden. Erfolgt der Zugriff auf diesen Speicherbereich durch mehrere Prozesse, müssen diese synchronisiert werden.

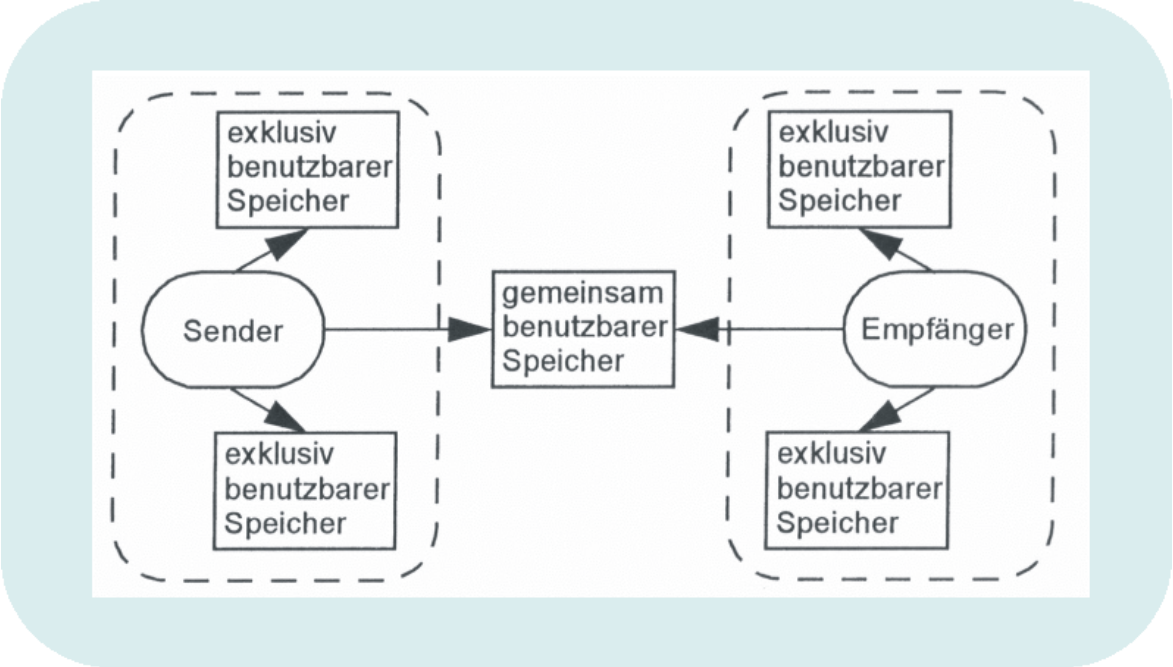

Abb. 5.19: Grundprinzip von Shared Memory bei dem XML-Konverter

Um die Kommunikation für die verteilte Produktentwicklung von den ambienten Beleuchtungssystemen zu realisieren, wird bei dem XML-Konverter den Socketprinzip benutzt. Grundlegend für Netzwerktechniken sind die Referenzmodelle ISO/OSI und TCP/IP. Die wesentliche Idee ist hierbei, datenverarbeitungsorientierte Funktionen, transportorientierte Funktionen und physikalische Übertragung getrennt voneinander zu betrachten und zu realisieren. So wird die Komplexität bei der Planung und dem Entwurf des Rechnernetzes reduziert.

Der automatisierte Austausch von Daten zwischen Maschinen ist seit mehreren Jahrzehnten für Geschäftstransaktionen unerlässlich. Electronical Data Interchange (EDI) ist hierbei der Obergriff für den „unternehmensübergreifenden Austausch von strukturierten Geschäftsdokumenten zwischen

Rechneranwendungen unter Nutzung von Datenformatstandards und Kommunikationswegen“. Als Kommunikationsweg hat sich in den letzten Jahren das Internet mit seinen Standardprotokollen weltweit etabliert. Somit steht den Geschäftspartnern ein weitverbreitetes kostengünstiges Medium zur Verfügung.
Ein Socket kann als Datenpunkt zur Kommunikation zwischen Prozessen betrachtet werden. Sockets ermöglichen eine bidirektionale Kommunikation sowohl lokal als auch innerhalb eines Netzwerkes. Der vom Benutzer aus sichtbare Teil der Kommunikation besteht aus drei Teilen (*Abb. 5.20*):

- dem Socket-Kopf (Socket Layer),
- dem Protokollteil (Protocol Layer),
- dem Gerätetreiber (Device Layer).

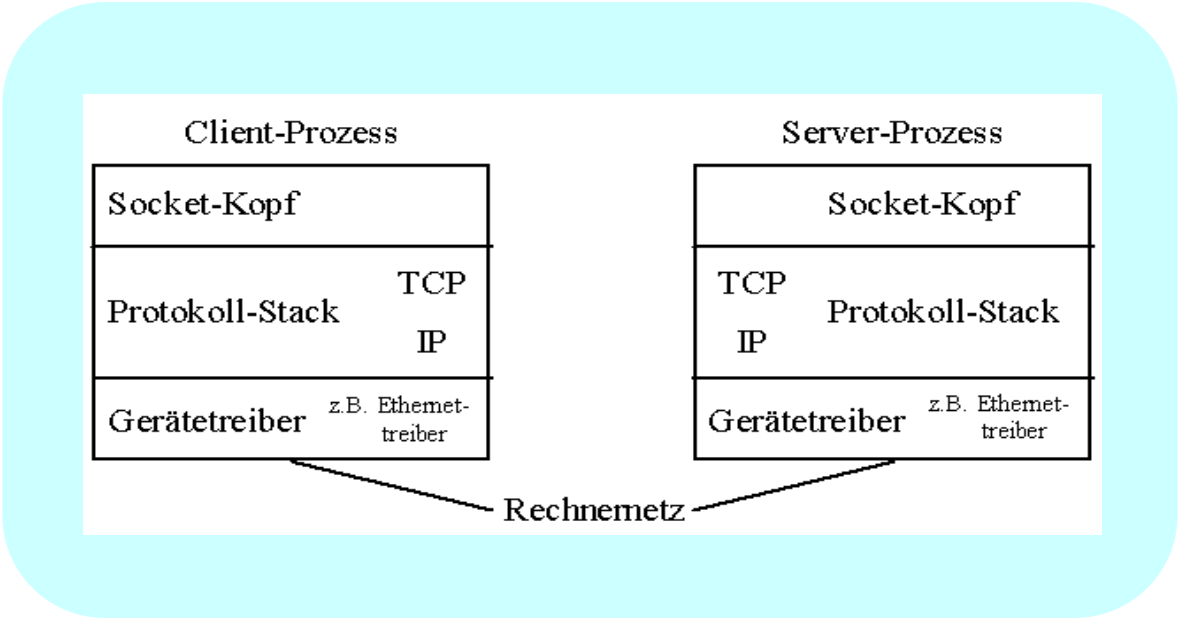

Abb. 5.20: Das Socket-Modell am Beispiel TCP/IP bei dem XML-Konverter

Der Socket-Kopf bildet die Schnittstelle zwischen den Betriebssystemaufrufen und den weiter unten liegenden Schichten. Welche Kombinationen von Sockets, Protokoll und Treiber, möglich sind, wird bei der Systemgenerierung festgelegt. Sockets mit gleicher Charakteristika, bezüglich Adressierung und des Protokolladressformates, werden zu Bereichen, sogenannten Domains,

zusammengefasst. Die Betriebsystemdomäne dient dabei zur lokalen Kommunikation zwischen Prozessen. Die Internetdomäne dient zur Kommunikation über ein Netzwerk für die verteilte Produktentwicklung von den ambienten Beleuchtungssystemen.

Eine Netzkommunikation läuft bei dem XML-Konverter so ab, dass ein Server-Pozess einen Kommunikationspunkt (Socket) aufbaut. Ein Client-Prozess koppelt sich ebenfalls an einen (lokalen) Kommunikationspunkt (Socket) und beantragt einen Verbindungsaufbau zu dem Socket des Server-Prozesses.

Der Server-Prozess macht mit dem Aufruf listen() dem System bekannt, dass er Verbindungen akzeptieren will und gibt die Länge einer Warteschlange an. Der Aufruf accept() erfolgt, wenn ein Client-Prozess eine Verbindung anfordert.

Der accept()-Aufruf liefert nach einem Verbindungsaufbau dem Server-Prozess einen neuen Socket-Deskriptor (analog zu einem Dateideskriptor) für einen anderen Socket zurück, über den nun die Kommunikation mit dem Client-Prozess erfolgen kann. Wie auf der Abbildung *Abb. 5.21* zu sehen ist, ist der Socket, an dem der Server-Prozess auf Verbindung wartet, und der Socket, über den nach einem Verbindungsaufbau die Kommunikation stattfindet, auf der Serverseite nicht identisch.

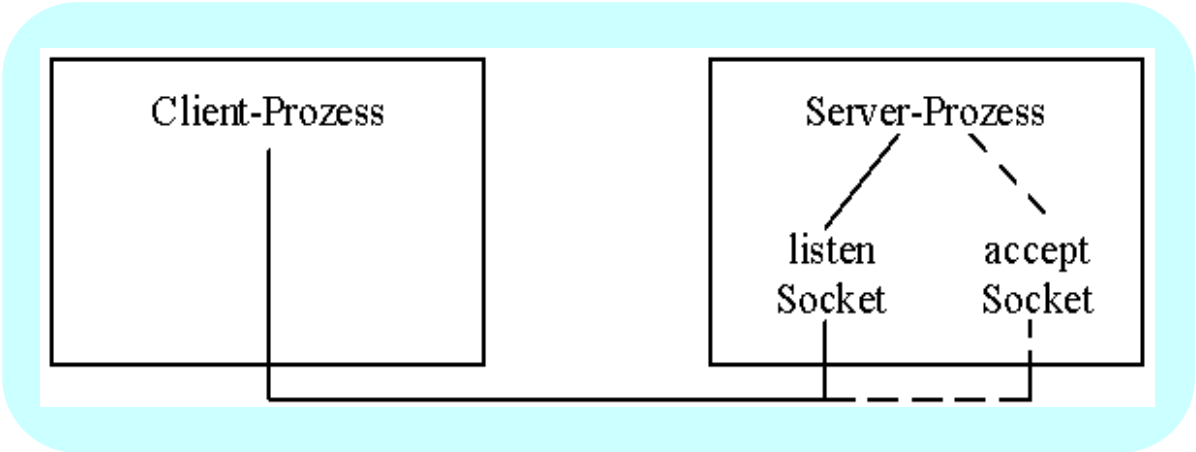

Abb. 5.21: Client- Server -Kommunikation bei dem XML-Konverter

Am ende dieses Abschnittes wird das Streamprinzip erläutert. Ein Stream ist ein Pseudotreiber im Betriebssystemkern, wobei der Begriff Pseudo hierbei verwendet

wird, weil zunächst hinter dem Treiber gar kein physikalisches Gerät steht, sondern nur eine Reihe von Softwarefunktionen. Der Treiber stellt dabei eine Schnittstelle zwischen Benutzerprogramm und Beriebssystem zur Verfügung. Über diese Schnittstelle können Daten(ströme) in beide Richtungen und volldublex ausgetauscht werden.
Ein Datenweg, der mit einem Stream-Mechanismus aufgebaut wurde besteht aus folgenden Komponenten (*Abb. 5.22*):

- dem Stream-Kopf (Stream Head),
- einem oder mehrere optionalen Verarbeitungsmodulen,
- einem an dem Stream angekoppelten Treiber.

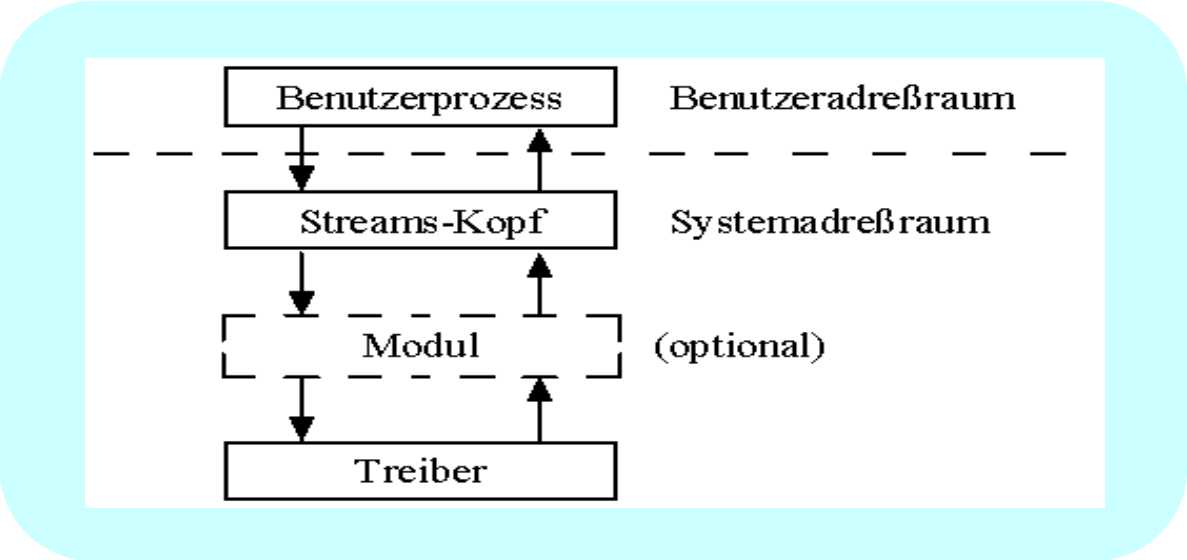

Abb. 5.22: Schemabild eines Streams

Der Treiber kann dabei ein Gerätetreiber für ein physikalisches Gerät oder ein Pseudotreiber sein. Eine mögliche Funktion eines Verarbeitungsmoduls kann z.B. in einem Netzwerk die Abarbeitung eines Netzwerkprotokolls sein.
Eine wesentliche Eigenschaft des Streams-Mechanismus ist der, dass Verarbeitungsmodule dynamisch in den Verarbeitungsstrom eingeschaltet und wieder entfernt werden können. Wird ein neues Verarbeitungsmodul eingefügt,

geschieht dieses unmittelbar hinter dem Kopfmodul. Bereits vorhandene Verarbeitungsmodule werden dadurch nach "unten" verschoben.

Nachdem am Anfang dieses Abschnittes die Kommunikationsprinzipien, die im Rahmen des XML-Konverters geschehen, erläutert sind, wird nun über den Ablaufmechanismus von dem XML-Konverter näher eingegangen.

Über sogenannte Processing Instructions (PI) wird Der XML-Konverter mitgeteilt, welcher Prozessor im nächsten Schritt das XML-Dokument, bzw. den erzeugten XML-Baum bearbeiten soll. Sind alle PI´s abgearbeitet wird das Dokument an den Client gesendet. Auf der Abbildung *Abb. 5.23* wird der Aufbau sowie die Verarbeitungsstruktur von dem XML-Konverter präziser dargestellt.

Die Processing Instructions (PI) geben darüber Aufschluss, wie das folgende XML-Dokument verarbeitet werden soll. Die erste Zeile eines XML Dokumentes muss die Processing Instruction <?xml version="1.0" ?> enthalten. Der innerhalb eines Dokumentes verwendete Zeichensatz kann über das Schlüsselwort encoding in der PI definiert werden, z.B. <?xml version="1.0" encoding="UTF-8"?>. Wird kein Zeichensatz spezifiziert, erwartet der XML Parser UTF-8. Es können beliebige weitere PIs angegeben werden, die Metadaten über das XML Dokument enthalten. Sie dürfen nur nicht mit dem reservierten Wort xml starten.

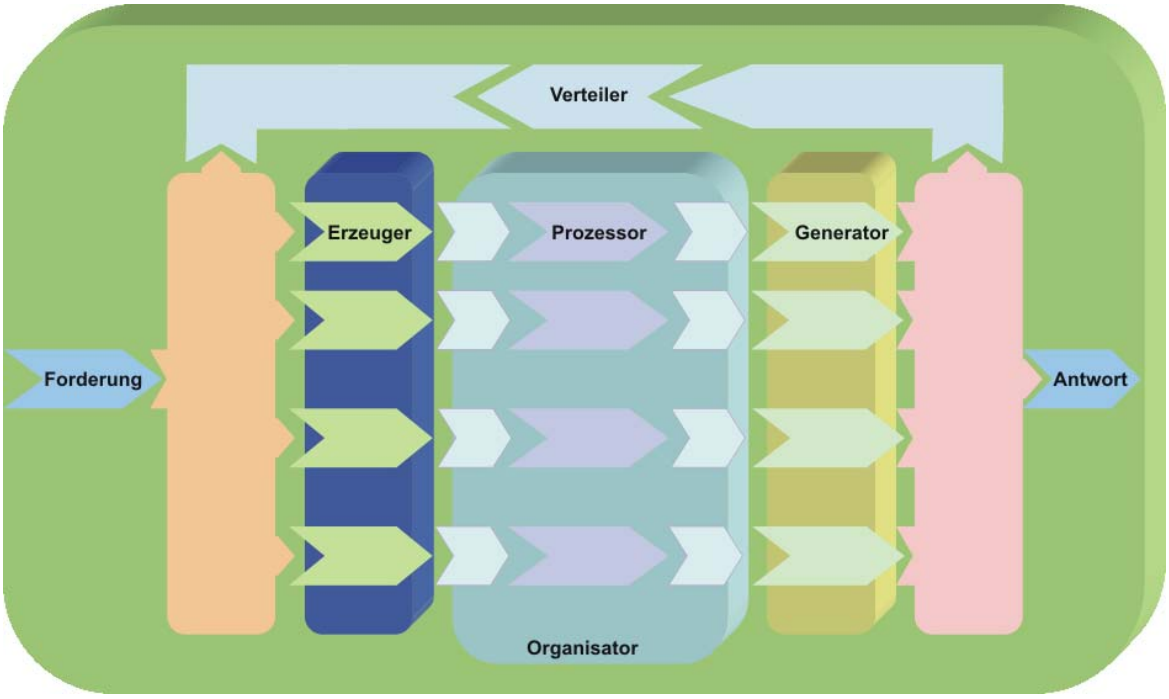

Abb. 5.23: Aufbau und Verarbeitungsstruktur des XML-Konverters

Der XML-Konverter besteht aus sechs verschiedenen Komponenten, die im folgenden näher erläutert werden.

Forderung:

Die Forderung enthält alle für die Verarbeitung der Anfrage wichtigen Informationen (welche URI, Typ des Browser, gewünschtes Ausgabeformat des Ergebnisses, welcher Erzeuger aktiviert werden soll, usw.) und wird vom Client an den Server oder von einer Phase an die nächste Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen geschickt.

Erzeuger:

Der Erzeuger bearbeitet den eingegangenen Forderungen und liefert als Ergebnis ein XML-Dokument, der Inhalt dieses XML-Dokumentes ist abhängig vom entsprechenden Erzeuger. In dem Erzeuger werden die eingegangenen Daten und Forderungen zuerst an den sogenannten XML-Generator, der zum Beispiel für das Generieren der Web-Seiten (HTML-Seiten, ...) bzw. anderer Dokumente (CAD-Dokumente, lichttechnische Dokumente, ...) zunächst gestartet wird. Dieser bringt die eingegangenen Daten gemäß den Forderungen in ein vordefiniertes XML-Format. Der Dokumentengenerator erzeugt schließlich aus dieser XML-Datei und einer vordefinierten XSL-Datei die Web-Seiten bzw. ein anderes Dokument im gewünschten Format. Im Erzeuger wird die Generierung der XML-Dateien als Zwischenschritt für die Generierung von Dokumenten in beliebigen Formaten gemacht. In der Abbildung *Abb. 5.24* ist in dem rechten Teil (im Erzeuger) dieses Sachverhalten schematisch erläutert. In der Abbildung *Abb. 5.25* ist auch das Sachverhalten noch mal in dem rechten Teil (im Erzeuger) erläutert, in dieser Abbildung ist ebenfalls das Multi-Pipe-Konzept schematisch dargestellt.
Der Erzeuger gibt sein Endergebnis an den Organisator für die weiteren Bearbeitungsschritte weiter.

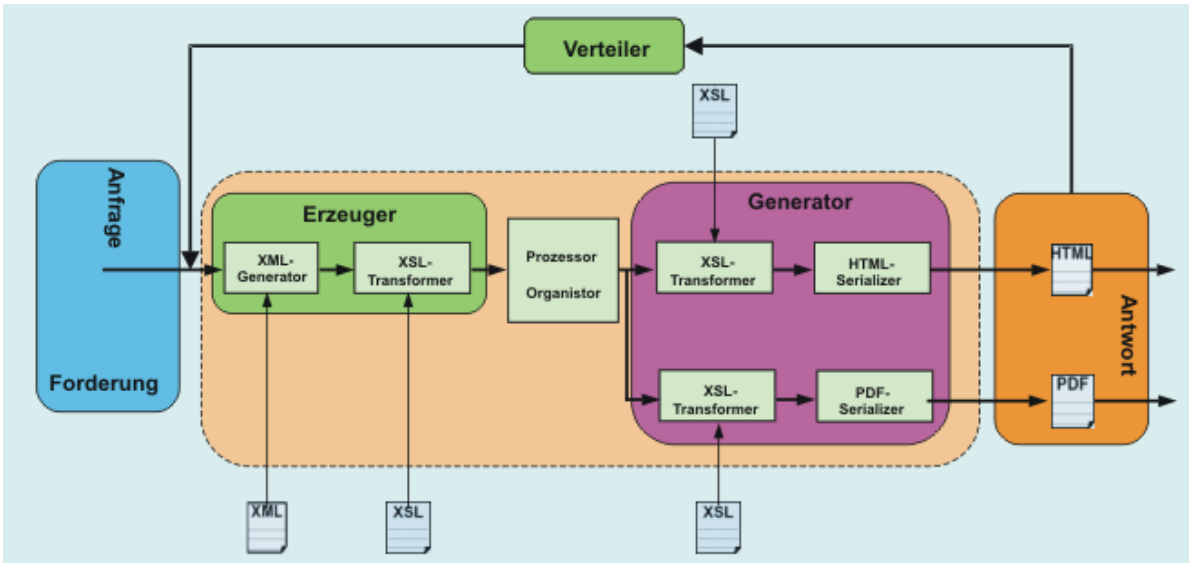

Abb. 5.24: Interne Verarbeitungsstruktur des XML-Konverters

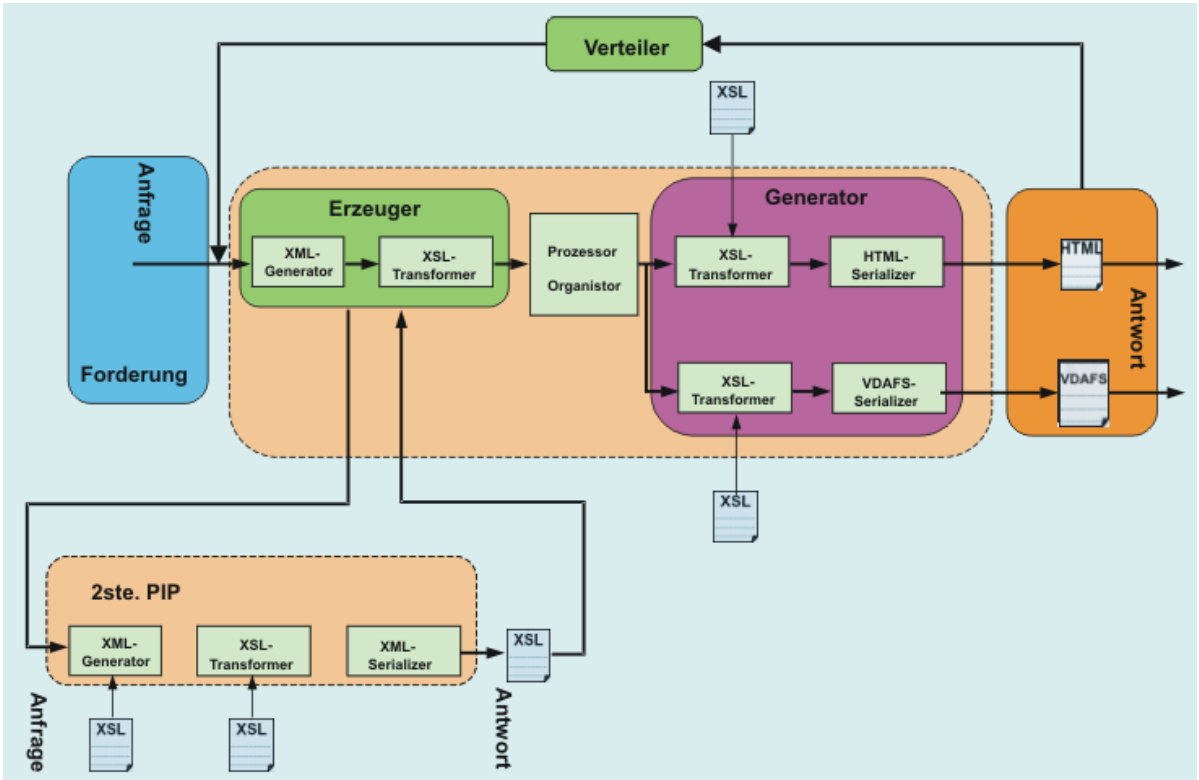

Abb. 5.25: Interne Multi-Pipe Verarbeitungsstruktur des XML-Konverters

Organisator :

Der Organisator bestimmt auf Basis von XML-PI´s, welcher Prozessor das XML-Dokument weiterverarbeiten soll. Nachdem die Verarbeitung der eingegangenen Daten in Prozessoren abgelaufen ist, ist es die Aufgabe des Organisators, die überarbeiteten Dokumente an den vorgegebenen Generator weiterzuleiten. Der Organisator organisiert nicht nur die Prozessore sondern er stellt eine wichtige Brücke zwischen dem Erzeuger und dem Generator.

Generator:

Der Generator liest den Input-Stream – dieser Input-Stream kann aus dem Dateisystem oder über eine HTTP-Verbindung (zum Beispiel verteilte Produktentwicklung) geladen, oder spontan aus einer objektorientierten oder relationalen Datenbank (Kommunikation zwischen den einzelnen Phasen des Produktlebenszyklus) generiert werden - und liefert einen XML-Stream zurück, der von der nachgeschalteten Komponente in dem XML-Konverter verarbeitet wird. Der Generator wandelt also das XML-Dokument in einen für den Client oder für den Browser verarbeitbaren Datenstrom um (MIME Typ setzen, ausführbarer Code, usw.). Der richtige Generator wird mittels eines PI-Tag bestimmt. In dem Generator wird das XML-Dokument zuerst an den XSL-Transformer geschickt, der das eingegangen XML-Dokument gemäß den Forderungen verarbeitet und an den PDF- , XML-, HTML-, ... oder an den VDAFS-Serializer, je nachdem welches Format gewünscht ist, weiterschickt. Der Serializer wandelt das XML-Dokument in das gewünschte Format um. Nachdem der Serializer die Umwandlung abgeschlossen hat, schickt der Generator das Endergebnis an die Antwort weiter.
Der Transformer ist ein der wichtigsten Komponenten in dem XML-Konverter. Er liest den SAX-Stream, der vom Generator geliefert wird ein und kann diesen SAX-Stream dann verarbeiten. Ein Transformer kann den eingehenden SAX-Stream direkt verändern, neue XML-Elemente einfügen oder bestehende herausnehmen. Das Ergebnis sendet er an die nächste Komponente in den XML-Konvert. Anderes formuliert, der Transformer akzeptiert und erzeugt XML. So kann die Ausgabe

(SAX-Stream) eines Transformers einem anderen weitergeleitet werden. Mit dem ersten Transformer könnte man z.B. die Daten filtern, mit den zweiten umsortieren und mit den dritten schließlich Formatierungsinformationen hinzufügen. Der Serializer ist die letzte Komponente in dem XML-Konverter. Er wandelt den eingegangenen Daten gemäß den Forderungen in ein entsprechendes Ausgabeformat um.

Antwort:

Die Antwort fasst das neu erzeugte Dokument mit weiteren Eigenschaften wie Länge, MIME Typ usw. zusammen und schickt es an den Verteiler, falls es weitere Verarbeitungsschritte notwendig sind, sonst schließt sie den Verarbeitungsprozess ab.

Verteiler:

Der Verteiler prüft, ob die Dokumente einen evtl. zweiten Verarbeitungsschritt (inkrementelle Verarbeitung) benötigen. Wenn weitere Verarbeitungsschritte benötigt sind, gliedert der Verteiler die Dokumente wieder in die Verarbeitungskette ein und schickt diese Dokumente mit zusätzlichen PI´s wieder an den Erzeuger. Falls es gar keine Verarbeitungsschritte der Dokumente mehr nötig sind, werden diese an die Antwort weiter geschickt.

Prozessor:

Der Prozessor, der für die Verarbeitung benötigt wird, wird über die PI´s bestimmt. Das folgende Tag <?XMLKonverter-process type="xslt"?> enthält die Information, welcher Prozessor benutzt werden soll (in dem XML-Konverter wird z.B. u.a. der XSL/T-Prozessor verwendet).

- XSL/T-Prozessor: XSLT-Transformationen werden angewendet.
- XSP-Prozessor: verarbeitet XSP-Seiten.
- SQL-Prozessor: wickelt SQL-Datenbankabfragen mittels einfacher Tag-Sprache ab.
- LDAP-Prozessor: ermöglicht den Zugriff auf LDAP-Verzeichnisse. Liefert ein XML-Dokument zurück.

Durch ein PI-Tag kann zusätzlich bei einer Webausgabe angegeben werden, welcher Browser die Forderung stellte, um die Ausgabe dementsprechend anzupassen.
Ein Client schickt also eine Forderung an den Server mit den Informationen welches Ausgabeformat er zurück erhalten möchte. Der XML-Konverter arbeitet den Auftrag ab und sendet dem Client das gewünschte Ausgabeformat. Der XML-Konverter stellt somit die Möglichkeit zur Verfügung sehr mächtige Frameworks einzubinden um mit XML-Dokumenten arbeiten zu können.

6 Zusammenfassung

Ziel dieser Arbeit war es, eine XML-basierte Sichtweise auf die einzelnen Phasen des Produktlebenszyklus von ambienten Beleuchtungssystemen zu realisieren. Dies hat zu der Entwicklung des XML-Konverters geführt, der speziell für den Produktentwicklungsprozess von ambienten Beleuchtungssystemen entwickelt wurde. Ausgangspunkt waren die intelligente Datenbereitstellung, - aufbereitung, weitergabe entlang des gesamten Produktlebenszyklus von ambienten Beleuchtungssystemen in einer gültigen XML-basierten Datenstruktur, die Organisation und Verwaltung der immensen Datenmengen, die in jeder Phase des Produktlebenszyklus von ambienten Beleuchtungssystemen auffallen, die Protokolle des Datenaustausches zwischen den Phasen des Produktlebenszyklus und für die verteilte Produktentwicklung, sowie die Automatisierung dieses Datenaustausches. Dieser Beitrag ist insbesondere wichtig, weil er der erste ist, der diese Thematik in ihrer Gesamtheit betrachtet, analysiert und behandelt. Er liefert eine konkrete Lösung der Problemstellung und somit stellt er die ersten Schritte in diese Thematik dar.

Seit dem Anfang dieser Arbeit lag der Fokus immer auf der erweiterbaren Auszeichnungssprache XML, weil XML flexible Mechanismen für den Datenaustausch und die Aufbereitung von Inhalten für die unterschiedlichsten Zielmedien liefert und weil sie viele Potentiale zu versprechen scheint.

Über die Trennung von Inhalt, Struktur und Erscheinungsbild gelingt es, im XML dies und mehr zu realisieren. Die Erweiterbarkeit von XML an bestehende Problemlösungen ist ein weiterer wichtiger Aspekt bei der Weiterentwicklung und Verbreitung von den XML-Dokumentenstrukturen. Im Laufe dieser Arbeit wird u.a. ganz klar, dass XML die Zukunft insbesondere im Bereich des Daten- und Dokumentenaustausch und der Präsentation von Daten nach aktuellem Stand der Technik gehört.

Der Kapitel zwei (Kap.2) gibt einen Einblick in die vielfältigen Verarbeitungs- und Einsatzmöglichkeiten von XML, insbesondere gibt er einen wichtigen Einblick in den XML-Einsatz bei den Business-Intelligence-Systemen. Es wurden die wichtigsten Grundlagen von XML und die wichtigsten Techniken wie zum Beispiel

die Idee des Single-Source-Publishings vorgestellt. Die Techniken, die bei der Entwicklung des XML-Konverters eingeführt sind, werden ganz gezielt realisiert und eingesetzt um die vorstehende Aufgabe zu bewältigen. So wird z.B. die Technik des „Store once, use everywhere“ in dem XML-Konverter eingeführt, sodass die Kommunikation zwischen den unterschiedlichsten Applikationen sowie zwischen den unterschiedlichsten Plattformen über die XML-Dokumente realisiert werden kann.

Im Kapitel drei (Kap.3) und im Kapitel vier (Kap.4) werden jeweils die Datenformate von Lichttechnischen Messdaten, die bei den ambienten Beleuchtungssystemen benutzt werden, und die ambienten Beleuchtungssystemen ganz gezielt behandelt. In dem Kapitel Kap.3 wird eine Untersuchung aller lichttechnischen Daten, die weltweit existieren, durchgeführt. Im Rahmen dieser Untersuchung werden alle lichttechnischen Daten vorgestellt und analysiert und werden dann am Ende miteinander verglichen. Dieser Kapitel (Kap. 3) wird mit einer gesamten Bewertung aller lichttechnischen Messdaten abgeschlossen.

In dem Kapitel Kap.4 werden die ambienten Beleuchtungssystemen behandelt. Am Anfang dieses Kapitels wurden die Grundlagen von den ambienten Beleuchtungssystemen detailliert vorgestellt und an seinem Ende wurden die photometrischen Messdaten von Ambienten Beleuchtungssystemen auch detailliert vorgestellt.

Der Kapitel fünf (Kap.5) gibt am Anfang einen weiten Einblick in die Produktentwicklungskette von ambienten Beleuchtungssystemen hinsichtlich des Produktmanagements. Es werden zuerst die wichtigsten Grundlagen vorgestellt und dann wird der Produktlebenszyklus von ambienten Beleuchtungssystemen detailliert behandelt. Am Ende des ersten Abschnittes dieses Kapitels werden die Anforderungen an die XML-Dokumentenstrukturen der einzelnen Phasen des Produktlebenszyklus von Ambienten Beleuchtungssystemen ganz detailliert untersucht.

Am zweiten Abschnitt dieses Kapitels wird der XML-Konverter vorgestellt. Es ging hier vornehmlich um den Nachweis der prinzipiellen Funktionsfähigkeit des neuen Konzepts bei den ambienten Beleuchtungssystemen. Am Anfang dieses Abschnittes werden das Prinzip und die Funktionsweise des XML-Konverters

detailliert vorgestellt. Der XML-Konverter bekommt als Input Daten versehen mit bestimmten Forderungen, er bearbeitet diese eingegangenen Daten gemäß den mit gelieferten Forderungen und liefert sein Ergebnis dementsprechend als Output aus. Am Ende dieses Abschnittes wird der Aufbau des XML-Konverters vorgestellt. Es wurden vor allem die sechs Komponenten (die Forderung, der Prozessor, der Organisator, der Generator, die Antwort und der Verteiler) des XML-Konverters vorgestellt und ihre Funktionen werden beschrieben.

Der hier vorgestellte XML-Konverter ist erst ein Einstieg in die Thematik und stellt somit die erste Lösung dieser Problematik dar. Für die weitere Entwicklung des XML-Konverters ist eine Verbesserung der inkrementellen Verarbeitung denkbar.

7 Verzeichnisse

7.1 Formelzeichen- und Symbolverzeichnis

API	Applikation Program Interfaces
BI	Business-Intelligence
BPML	Business-Process Modelling Language
CLOB	Character Large Object
CSS	Cascading Style Sheet
CWM	Common Warehouse Metamodel
DOM	Document Object Model
DTD	Document Type Definition
EDI	Electronical Data Interchange
HTML	Hypertext Markup Language, Sprache zur Präsentation von Informationen
HTTP	Hypertext Transfer Protocol
IS	Informationssystem
LDAP	Lightweight Directory Access Protocol
MDC	Meta Data Coalition
MDX	Multidimensional Expressions
MOF	Meta Object Facility
NAMENSRAUM	Definitionsbereich einer Bezeichnung (Namespace)
OIM	Open Information Model
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
OMG	Object Management Group
PDF	Portable Document Format
PDMS	Produktdatenmanagementsystem
PMML	Predictive Modeling Markup Language
RDF	Resource Description Framework

SAX	Simple API for XML
SEMANTIK	Wortbedeutungslehre
SOAP	Simple Object Protocol, mit Hilfe von SOAP erfolgt der Zugriff auf XML-Objekte
SQL	Structured Query Language
SYNTAX	Lehre von Satzaufbau, zulässigen Satzarten und Wortarten
TCP	Transmission Control Protocol
UML	Unified Modelling Language
URI	Uniform Resources Identifier
WAP	Wireless Application Protokoll
W3C	Wide Wide Web Consortium
WML	Website Meta Language
WWW	Wide Wide Web
XBRL	eXtensible Business Reporting Language
XIF	XML Interchange Format
XLINK	eXtensible Linking Language
XMDQL	Multidimensional Query Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language, Syntax für hierarchische Dokumente und zum Strukturieren von Informationen bzw. Dokumenten
XMLA	XML for Analysis
XML-QL	Query Language for XML
XPOINTER	eXtensible Pointer
XQL	XML Query Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language, Sprache zum transformieren von XML in beliebiges Zielformat
XSL-FO	eXtensible Stylesheet Language:Formating Objects, Sprache um 2D Layout von Text zu beschreiben, egal ob Printmedium, wie

	Drucker oder Digitalme-dium, wie PDF
XSLT	eXtensible Stylesheet Language Transformation, Weiterentwicklung von XSL
XSP	eXtensible Server Pages

A1	Strahlerfläche
A2	Empfängerfläche
cd	Candela
CIE	Comission Internationale de l’Eclairage
DIN	Deutsche Industrie-Norm
E	Beleuchtungsstärke
Frel	Relativer Fehler
I	Lichtstärke
L	Leuchtdichte
LED	Light Emitting Diode (Leuchtdiode, Lumineszenzdiode)
lm	Lumen
lx	Lux
m	Meter
OLED	Organic Light Emitting Diode (organische Leuchtdiode)
Q	Lichtmenge
r	Radius
s	Sekunde
V(λ)	Spektrale Hellempfindlichkeit
ε1	Winkel zwischen Ausstrahlungsrichtung und Normal auf dA1
ε2	Winkel zwischen Lichteinfallsrichtung und Normal auf dA2
λ	Wellenlänge
Φ	Lichtstrom
Ω	Raumwinkel

Ω_0	Einheitsraumwinkel

7.2 Tabellen- und Abbildungsverzeichnis

7.3 Literaturverzeichnisse

1. Harald Schöning, XML und Datenbanken, Hanser Verlag, 2003, ISBN 3-446-22008-9
2. Sven-Holger Kloss, Ein Goniophotometer zur Messung des Lichtstromes und der Lichtstärkeverteilung von hohlen Lichtleitern , Dissertation Technische Universität Berlin, 2001
3. Michael Seeboerger-Weichselbaum, XML, moderne Industrie Buch AG & Co. KG Verlag, 2001, ISBN 3-8266-7201-1
4. Eugen Hecht, Optik, Oldenburg Verlag, 2001, ISBN 3-486-24917-7
5. Andreas Heuer und Gunter Saake, Datenbanken Konzepte und Sprachen, MITP Verlag, 1997, ISBN 3-8266-0349-4
6. Hering/ Gutekunst/ Dyllong, Informatik für Ingenieure, VDI Verlag, ISBN 3-18-400944-07 IEEE – THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC., New York, NY: IEEE Std 1219-1998:
7. ISO – INTERNATIONAL ORGANIZATION IEEE Standard for Software Maintenance. Schweiz: (ISO/IEC 12207) Standard for Information Technology – Software
8. ISO – INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Genf, Schweiz: (ISO/IEC 14764) Standard for Information Technology – Software maintenance
9. KAJKO-MATTSSON: Corrective maintenance maturity model: Problem management. Doktorarbeit, Department of Computer and Systems Sciences (DSV), Stockholm University and Royal Institute of Technology, Stockholm, Sweden , 2001
10. HARJANI, D.-R. und J.-P. QUEILLE: A Process Model for the Maintenance of Large Space Systems Software. 1992. Erschienen in: Proceedings of the ICSM, Orlando, Florida, 1992
11. Thomas Michel, XML kompakt, Hanser Verlag, 1999, ISBN 3-446-21302-3
12. Elliotte Rusty Harold, XML Bibel, mitp Verlag, 2000, ISBN 3-8266-0627-2

13. Helmut Vonhoegen, Einstieg in XML, Galileo Computing Verlag, 2002, ISBN 3-89842-137-6
14. Dirk Ammelburger, XML Grundlagen der Sprache und Anwendungen in der Praxis, Hanser Verlag, 2004, ISBN 3-446-22562-5
15. Al-Akaidi, M. M.; Seager, R. D.: Mathematical Modelling of Light Sources(s); o. J.
16. Al-Akaidi, M. M.; Seager, R. D.: Ray Tracing Algorithms; o. J.
17. Alps Electric Co., Ltd., Tokio: Beleuchtungslicht-Leiteinrichtung 1990
18. Amanatides, John: Realism in Computer Graphics: A Survey; o. J.
19. Barr, Alan H.: Ray Tracing Deformed Surfaces; in: Computer Graphics; Band 20, (1986), Heft 4, S. 287-296; 1986
20. Barsky, Brian A.: A Description and Evaluation of Various 3-D Models; o. J.
21. Barth, W.; Stürzlinger, W.: Efficient Ray Tracing for Bezier and B-spline Surfaces; in: Comput. & Graphics; Band 17, (1993), Heft 4, S. 423-430; 1993
22. Baum, Dan: 3D Graphics Hardware: Where We Have Been, Where We Are Now, and Where We Are Going; in: Computer Graphics; (1998), S. 65-66; 1998
23. Carson, Steve: The History of Computer Graphics Standards Development; in: Computer Graphics; (1998), S. 34-38; 1998
24. Chen, John J.; Ozsoy, Tulga M.: An Intersection Algorithm for C2 Parametric Surface; o. J.
25. Claussen, Ute; Pöpsel, Josef: Der Realität auf der Spur Radiosity – Ein Verfahren zur globalen Beleuchtungsberechnung - Teil 1: Geschichtliches und Prinzipielles; in: c`t; (1991), Heft 6, S. 204-214; 1991
26. Claussen, Ute; Pöpsel, Josef: Der Realität auf der Spur Radiosity – Ein Verfahren zur globalen Beleuchtungsberechnung - Teil 2: Umsetzung in die Programmierpraxis; in: c`t; (1991), Heft 8, S. 196-208; 1991
27. Cohen, Elaine; Lyche, Tom: Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics; o. J.

28. Cohen, Michael F.; Chen, Shenchang Eric; Wallace, John R.; Greenberg, Donald P.: A Progressive Refinement Approach to Fast Radiosity Image Generation; in: Computer Graphics; Band 22, (1988), Heft 4, S. 75-84; 1988
29. Cohen, Michael F.; Greenberg, Donald P.: The Hemi-Cube - A Radiosity Solution for Complex Environments; in: Computer Graphics; Band 19, (1985), Heft 3, S. 31-40; 1985
30. Cook, Robert L.; Porter, Thomas; Carpenter, Loren: Distributed Ray Tracing; in: Computer Graphics; Band 18, (1984), Heft 3, S. 137-145; 1984
31. Cook, Robert L.; Torrance, Kenneth E.: A Reflectance Model for Computer Graphics; in: ACM Transaction on Graphics; Band 1, (1982), Heft 1, S. 7-24; 1982
32. Coté, Marie; Pagano, Robert J.; Stevenson, Michael A.: Optical system performance visualization; 1999
33. Coté, Marie; Tesar, John: Optical System Image Irradiance Simulations; in: Proceedings of the SPIE; Band 3482, (1998), S. 397-409; 1998
34. Dambach-Werke GmbH: Leuchtkasten 1990
35. Dehoff, Peter; Egger, Wolfgang: Geben mit dem Computer gerechnete Leuchtdichteverteilungen den korrekten Helligkeitseindruck wieder?; o. J.
36. Delacour, Jacques; Ungar, Serge; Mathieu, Gilles; Hasna, Günther; Martinez, Pascal; Roche, Jean-Christophe: Front Panel engineering with CAD simulation tool; in: Proceedings of the SPIE; Band 3636, (1999), S. 11-21; 1999
37. Dewald, Scott: Using ZEMAX Image Analysis and user-defined surfaces for projection lens design and evaluation for Digital Light Processing TM projection systems; in: Optical Engineering ; Band 39, (2000), Heft 7, S. 1802-1807; 2000
38. Eberle Spencer, Domina ; Gaston, Eugene A.: Diffuse and specular components of reflectance: why and how; in: c`t; (1978), July, S. 240-248; 1978
39. Erdei, Gábor; Szarvas, Gábor; Barabás, Miklós: Extensions to a lens design program for the ray-optical design of waveguide lenses and prism couplers; in: Journal of Modern Optics; Band 44, (1997), Heft 2, S. 415-430; 1997

40. Filip, Daniel J.: Blending Parametric Surfaces; in: ACM Transaction on Graphics; Band 8, (1989), Heft 3, S. 164-173; 1989
41. Flowers, Ken: Programming with the Motif Toolkit; in: UNIX WORLD; (1990), November, S. 135-144; 1990
42. Flowers, Ken: Using Motif´s User Interface Language; in: UNIX WORLD; (1990), December, S. 119-131; 1990
43. Forney, Paul: Integrated Optical Design; in: Proceedings of the SPIE; Band 3435, (1998), S. 2-8; 1998
44. Fuhr, Richard D.; Hsieh, Lwo; Kallay, Michael: Object-oriented paradigm for NURBS curve and surface design; in: Computer-Aided Design; Band 27, (1995), Heft 2, S. 95-100; 1995
45. Fujimoto, Akira; Tanaka, Takayuki; Ivata, Kansei: ARTS: Accelerated Ray-Tracing System; o. J.
46. Glassner, Andrew S.: Space Subdivision for Fast Ray Tracing; o. J.
47. Goldsmith, Jeffrey; Salmon, John: Automatic Creation of Object Hierarchies for Ray Tracing; o. J.
48. Goral, Cindy M.; Torrance, Kenneth E.; Greenberg, Donald P.; Battaile, Bennet: Modeling the Interaction of Light Between Diffuse Surfaces; in: Computer Graphics; Band 18, (1984), Heft 3, S. 213-222; 1984
49. Greene, Ned: Environment Mapping and Other Applications of World Projections; o. J. 55 Haines, Eric A.; Greenberg, Donald P.: The Light Buffer: A Shadow-Testing Accelerator; o. J.
50. Hamann, Bernd; Tsai, Po-Yu: A tesselation algorithm for the representation of trimmed NURBS surfaces with arbitrary trimming curves; in: Computer-Aided Design; Band 28, (1996), Heft 6/7, S. 461-472; 1996
51. Hankins, J. A.; Rigler, A. K.: Splines in Optical Ray Tracing Models; o. J.
52. Harrison, Nigel: Software makes light work of optical design; in: Scientific Computing World; (1995), Heft 8, S. 38-40; 1995
53. He, Xiao D.; Heynen, Patrick O.; Phillips, Richard L.; Torrance, Kenneth E.; Salesin, David H.; Greenberg, Donald P.: A Fast and Accurate Light Reflection Model; in: Computer Graphics; Band 26, (1992), Heft 2, S. 253-254; 1992

54. He, Xiao D.; Torrance, Kenneth E.; Sillion, Francois; Greenberg, Donald P.: A Comprehensive Physical Model for Light Reflection; in: Computer Graphics; Band 25, (1991), Heft 4, S. 175-186; 1991
55. Heckbert, Paul S.: Survey of Texture Mapping; o. J.
56. Heckbert, Paul S.: Adaptive Radiosity Textures for Bidirectional Ray Tracing; in: Computer 63 Graphics; Band 24, (1990), Heft 4, S. 145-154; 1990
57. Peter Rechenberg, G. Pomberger, Informatikhandbuch, Hanser Verlag, 1999, ISBN 3-446-19601-3
58. Stephan Schwalm, Carsten Bange, Wirtschaftsinformatik 46 (2004), S. 5-14
59. ICCM-Electronics GmbH: Beleuchtungseinrichtung für LCD-Displays oder dergleichen 1990
60. Immel, David S.; Cohen, Michael F.; Greenberg, Donald P.: A Radiosity Method für Non-Diffuse Environments; in: Computer Graphics; Band 20, (1986), Heft 4, S. 133-142; 1986
61. Isshiki, Masaki; Ono, Hiroki; Nakadate, Suezou: Lens Design: An Attempt to Use "Escape Function" as a Tool in Global Optimization; in: OPTICAL REVIEW; Band 2, (1995), Heft 1, S. 47-51; 1995
62. ITT Industrie Riunite S.r.l, Turin: Leuchtringkörper für Bordzubehör von Fahrzeugen 1987
63. Jansen, Frederik W.: Data structures for ray tracing; 1986
64. Jensen, Henrik Wann; Christensen, Niels Jorgen: Photon maps in bidirectional Monte Carlo Ray Tracing of complex Objects; in: Comput. & Graphics; Band 19, (1995), Heft 2, S. 215-224; 1995
65. Kajiya, James T.: The Rendering Equation; in: Computer Graphics; Band 20, (1986), Heft 4, S. 143-150; 1876
66. Kajiya, James T.: Anisotropic Reflection Models; in Computer Graphics; Band 19, (1985), Heft 3, S. 15-21; 1985
67. Kay, Timothy L.; Kajiya, James T.: Ray Tracing Complex Scenes; in: Computer Graphics; Band 20, (1986), Heft 4, S. 269-278; 1986
68. Kirk, D. B.: Realistic Simulation Using Ray Tracing; o. J.

69. Kumar, Subodh; Manocha, Dinesh: Efficient rendering of trimmed NURBS surfaces Research; in: Computer-Aided Design; Band 27, (1995), Heft 7, S. 509-521; 1995
70. Lee, Randy B.; Fredricks, David A.: SPECIAL FEATURE - Intersection of Parametric Surfaces and a Plane; o. J
71. Levner, Geoff; Tassinari, Paolo; Marini, Daniele: A Simple, General Method for Ray Tracing Bicubic Surfaces; o. J
72. Li, Yun; Gargiulo, Edward P.; Keefe, Michael: Studies in Direct Tooling Using Stereolithography; in: Journal of Manufacturing Science and Engineering; Band 122, (2000), Heft 2, S. 316-322; 2000
73. Lin, Psang Dain; Liao, Te-tan: Skew Ray Tracing and Sensitivity Analysis of Geometrical Optics; in: Journal of Manufacturing Science and Engineering; Band 122, (2000), Heft 2, S. 338-349; 2000
74. Marcus, Aaron: Designing Graphical User Interfaces; in: UNIX WORLD; (1990), August, S. 107-115; 1990
75. Marcus, Aaron: Designing Graphical User Interfaces; in: UNIX WORLD; (1990), September, S. 121-127; 1990
76. Marcus, Aaron: Designing Graphical User Interfaces; in: UNIX WORLD; (1990), October, S. 135-138; 1990
77. Meyer, Alan: A Linear Time Oslo Algorithm; in: ACM Transaction on Graphics; Band 10, (1991), Heft 3, S. 312-318; 1991
78. Mori, Kei: Lichtverteileranordnung zur gleichmäßigen Beleuchtung 1983
79. Peterson, Gary L.: Stray light calculation methods with optical ray trace software; 1999
80. Peterson, John W.: Ray Tracing General B-Splines; 1986
81. Philips Patentverwaltung: Lichtleitkörper für Frontblenden elektrischer Geräte 1989
82. Uli Shell, Objektorientierte Softwareentwicklung mit C++, von der Analyse zum Produkt, dpunkt.verlag, 1998, ISBN 3-920993-76-4