Color Management Integration für Database Publishing

Alexander Jacob

Matrikelnummer 234986

Bachelor-Thesis zur Abschlussprüfung
im Sommersemester an der
Bergischen Universität Wuppertal,
Fachbereich E

Abgabedatum: 30.09.2005

1. Prüfer: Prof. Dr. rer. nat. Stefan Brües
2. Prüfer: Dipl.-Ing. Claudio Höcker

BERGISCHE UNIVERSITÄT
WUPPERTAL

✉ 42097 Wuppertal
Telefax (0202) 439-2901
Telefon (0202) 439-0
www.uni-wuppertal.de

Fachbereich E
Elektrotechnik, Informationstechnik, Medientechnik

Prüfungsausschüsse
Kommunikationstechnologie Druck
Druck- und Medientechnologie
Lehramt Drucktechnik

Rainer-Gruenter-Str. 21 FME
42119 Wuppertal
Tel: 0202 439-1958

Anmeldung einer Abschlussarbeit (Bachelor Thesis)

Jacob, Alexander — Name, Vorname
234986 — Matrikelnummer

Gruitener Str. 42, 40699 Erkrath — Straße, Plz / Wohnort

Titel der Thesis: Color Management Integration für Database Publishing

Erläuterungen zur Themenstellung:
(Zielsetzung, Aufgabenschwerpunkte)

Erarbeitung und Implementierung einer Lösung für die Einbettung und Interpretation von ICC-Profilen in einen bestehenden Workflow

- Analyse der verwendeten Systemkomponenten
- Analyse der Probleme bei der Bilddateneinbettung
- Erarbeitung von Lösungsansätzen und -vorschlägen
- Umsetzung einer geeigneten Lösung

[X] Die Arbeit wird in einer Institution außerhalb der Hochschule durchgeführt

@ computational design GmbH, Robertstr. 5a, 42107 Wuppertal
Name, Anschrift der Institution

Frau Hilgers, Herr Yilmaz, Herr Teschke
Ansprechpartner

Ausgabedatum: 1.7.05 — Abgabetermin: 30.9.05

1. Prüfer: Brüß

2. Prüfer: Höckes

Unterschrift des Studierenden

Unterschrift

Unterschrift

Prüfvermerk Prüfungsamt

Prüfvermerk Vors. Prüfungsausschuss

Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und Zitate kenntlich gemacht habe.

Hiermit räume ich der Bergischen Universität Wuppertal ein, die Inhalte dieser Arbeit, unter Einhaltung eventuell vorgegebener Sperrfristen, wissenschaftlich zu nutzen und öffentlich zu machen.

_____

Unterschrift (Alexander Jacob)

Summary

Über ein Content-Management-System (CMS) können Inhalte, wie Texte oder Bilder, eingegeben, editiert und für eine Katalogproduktion aufbereitet werden. Bei einer Ausgabe wird der Content mit Hilfe der Extensible Markup Language (XML) als medienneutralem Informationsträger ausgelesen. XML kann über Stylesheets für verschiedenste Ausgabemedien (Druck, Internet, Mobiltelefone) layoutiert werden. Eine PDF-Ausgabe für den Druck stellt besondere Anforderungen an PDF-Erzeugertools, auch hinsichtlich Farbmanagementfunktionen nach den Spezifikationen des International Color Consortiums (ICC).

Bei einer dynamischen PDF-Generierung über die Software XEP werden ICC-Farbprofile, die in Bildern eingebettet sind, entfernt. Um dies zu verhindern, werden diese Bilder aus dem CMS über die PDF-Programmieranwendung PDFlib für eine weitere Verarbeitung in PDF integriert.

Speziell für die hohen Anforderungen der Druckindustrie wurde der PDF/X-3-Standard entwickelt, durch den eine fehlerfreie Produzierbarkeit garantiert wird. XEP muss für eine PDF/X-3-Erzeugung soweit angepasst werden, wie es die Konfigurationsmöglichkeiten zulassen.

Summary

A content management system (CMS) stores text or images as well as it allows to edit, format and combine this content for catalogue print production.

On output, XML (Extensible Markup Language) plays the role of a media neutral information carrier for the selected content. XML-stylesheets are used to layout the XML-data for various output purposes (e.g. print, internet, cell phones). A PDF-output for printing demands high requirements on the PDF-creation tools, in particular color management functionality as specified by the International Color Consortium (ICC) which is todays' state of the art.

Using XEP, an application for dynamic PDF creation, embedded ICC-profiles in images are not retained. To prevent this, the images stored in the CMS are modified by the programming application PDFlib to PDF-files for further processing while retaining their ICC-profiles.

Today, PDF/X-3 has become a standard for flawless print production workflows. Finally, XEP has to be configured for PDF/X-3 creation as far as possible.

Inhaltsverzeichnis

1. Einleitung 1

2. Der „pablish® - catalog publishing“-Workflow 3

2.1 Funktionsweise 3

2.1.1 Frontend und Content-Management-System 3

2.1.2 Dynamische PDF-Generierung 8

2.1.3 Ablaufbeschreibung 8

2.2 Schwachstellen bei der PDF-Erzeugung 10

2.2.1 Farbprofile und Bilddatenbehandlung 10

3. Workflow-Optimierung im Hinblick auf PDF/X-3 und Color Management 12

3.1 PDF/X-3 und Color Management 12

3.2 Steuerung der PDF-Generierung 14

3.2.1 Software Development Kit (SDK) 14

3.2.2 Anpassen der Workflow-Komponenten 14

3.3 IMG2PDF – Erzeugung von PDF-Dateien aus Bilddaten 17

3.3.1 PDFlib – Einbinden einer PDF-Bibliothek 18

3.3.2 TinyXML 21

3.3.3 Ablaufbeschreibung 22

4. Fazit 24

5. Ausblick 25

6. Literaturverzeichnis 27

7. Abbildungsverzeichnis 28

8. Anhang 29

8.1 Programmierung mit C++ - IMG2PDF 29

8.2 Extensible Markup Language (XML) - Medienneutraler Datenlieferant 32

8.3 Document Type Definition (DTD) 33

1. Einleitung

Database Publishing ist neben Color Management in den letzten Jahren eines der immer wiederkehrenden Schlagworte der Medienindustrie. Besonders für Katalogproduktionen eignet sich das datenbankgestützte Publizieren, um einen hohen Automatisierungsgrad zu erreichen und eine aufwendige, manuelle Bearbeitung abzulösen. Effizient und vorteilhaft ist es hierbei die verwendeten Daten für die Ein-/Ausgabe und Verwaltung medienneutral anzulegen, also auf ausgabespezifische Informationen zu verzichten. Somit lassen sich aus einem vorhandenen Datenbestand die verschiedensten Ausgabeformate generieren (für Internet, Druck, CD-Präsentation, PowerPoint-Präsentation, etc.).

Individuell angepasste Schnittstellen, z.B. ein Webinterface, ermöglichen dem Benutzer eine Eingabe, die auf seine Bedürfnisse zugeschnitten ist. Im Hintergrund verwalten und verknüpfen intelligente Datenbanken die verfügbaren Informationen. Tools zur automatischen „On-Demand"- oder serverseitigen PDF-Generierung können dabei, neben einer Aufbereitung für das Internet oder einer CD-Präsentation, den letzten Schritt des Workflows darstellen. Bei einer automatischen PDF-Ausgabe wünschen sich die meisten Medienunternehmen die Erzeugung eines „ready-to-print" PDF-Dokuments, das direkt an den Druckpartner weitergeschickt werden kann. Gerade hierbei ergeben sich die meisten Probleme, da die technischen Anforderungen an ein druckfertiges PDF sehr hoch sind (Einstellungen für Schrift, Color Management, Bildverarbeitung, etc.). Deshalb ist die Forderung nach der Berücksichtigung der PDF/X-3-Spezifikation immer stärker geworden, die als ISO Norm 15930-3 (PDF/X-3) die Übermittlung von digitalen Druckvorlagen auf der Grundlage von PDF regelt.

Die automatische PDF-Generierung liefert gute visuelle Ergebnisse, dies wirft aber die Frage auf, inwiefern sich diese PDF-Dateien hinsichtlich ihrer Qualität einschätzen lassen. Viele Applikationen bieten heutzutage die Möglichkeit PDF-Dokumente zu erzeugen. Dabei wird aber meistens nicht auf die für die Druckindustrie wichtigen Parameter geachtet bzw. eingegangen. Insbesondere im Hinblick auf die Verarbeitung von Bildern scheint ein großes Defizit zu bestehen. Bei der PDF-Generierung aus einer Datenbank oder von einer Applikation wird allerdings schon der Grundstein für eine weitere Verarbeitung gelegt. Alle Informationen, die dort schon fehlen,

unbeachtet bleiben oder verloren gehen, führen vielleicht zufällig zu einer akzeptablen bis guten, aber sicherlich keiner gezielt gesteuerten Reproduktion. Umso wichtiger ist es daher, auch hierbei der Kontrolle der Farbwiedergabe einen großen Stellenwert beizumessen. Der Zugang zu einem effizienten Color Management bleibt aber oftmals an einer oder mehreren Stellen des Workflows verschlossen.

Diese Arbeit soll am Beispiel der datenbankgestützten Katalogproduktion des Wuppertaler Cross-Media-Unternehmens „@ computational design GmbH“ Probleme, die in einer Database-Publishing-Umgebung mit Color Management auftreten können, verdeutlichen und die Umsetzung einer individuellen Lösung beschreiben. Im Vordergrund steht hierbei nicht nur der Umgang mit ICC-Farbprofilen, sondern auch das Ziel einer optimierten und gezielt gesteuerten Ausgabe von PDF/X-3-Dokumenten.

In einem theoretischen Teil sollen die Grundlagen für das Verständnis des vorhandenen Workflows und die auftretenden Probleme erläutert werden. Die Qualität der automatisch erzeugten PDF-Dokumente wird anhand von Preflight-Tests dargestellt. Vertiefend dazu soll abschließend erläutert werden, wie die praktische Umsetzung einer individuellen Lösung vollzogen wurde und ein kleiner Ausblick auf weitere Möglichkeiten zur Optimierung des Workflows gegeben werden.

2. Der „pablish® - catalog publishing“-Workflow

Das Produkt „pablish®“ der Wuppertaler Firma @ computational design GmbH ist ein Content-Management-System (im Folgenden CMS), ein System zur Verwaltung von Inhalten. Es basiert auf der objektorientierten Open-Source Engine ZOPE (Z Object Publishing Environment), die als Container für alle Arten von Informationen, die publiziert werden sollen, dient.

Das Zusatzmodul „pablish® – catalog publishing“ ermöglicht die automatisierte Ausgabe von Daten aus dem CMS für eine Online- und Offline-Katalogproduktion. Im Sinn einer Cross-Media-Produktion können unter anderem sowohl HTML-Dateien für das Internet als auch PDF-Dateien für das Internet und den Druck erzeugt werden. Bei dieser dynamischen Ausgabe laufen im Hintergrund Prozesse ab, auf die im Folgenden näher eingegangen werden soll.

2.1 Funktionsweise

2.1.1 Frontend und Content-Management-System

Die Benutzerschnittstelle für den Workflow bildet das browserbasierte Frontend. Ein Frontend wird zur interaktiven Anforderung, Eingabe sowie Anzeige von Daten verwendet. Es ist der Teil eines Programms, welcher die Darstellung der Daten übernimmt. Alles, was der Nutzer sieht, gehört zum Frontend. Es dient als Benutzeroberfläche für das CMS, wobei das Grundprinzip der Trennung von Design und Inhalt gilt. Auf der Basis von ZOPE bildet pablish® eine Plattform für die Realisierung von Internetanwendungen (in diesem Fall das Frontend), das über einen Browser aufgerufen werden kann. Auch hier wird eine strikte Trennung von Layout und Content vollzogen, ideal für eine vielfältige Ausgabe aus einem Datenbestand. Verschiedene Layouts können auf den gleichen Inhalt angewendet werden. Für jeden Kunden kann nicht nur die Benutzeroberfläche angepasst werden, wie z.B. ein Logo, sondern auch die Funktionen, über die er verfügen kann.

Wie schon erwähnt wird pablish® - catalog publishing als CMS für die Datenverwaltung bei der Katalogproduktion eingesetzt. Das bedeutet, dass alle Veränderungen am Content, die über das Frontend vorgenommen werden, hier zusammenlaufen

und anschließend für den User wieder am Bildschirm dargestellt werden. Somit lassen sich Daten nicht nur einfach und schnell eingeben, sondern auch direkt visuell auf ihre Richtigkeit überprüfen.

Für die weitere Verarbeitung der Katalogdaten übernimmt das CMS aber noch eine weitere Aufgabe. Es erzeugt aus den vorhandenen Daten einen Datenstrom in der Extensible Markup Language (im Folgenden XML). Die dabei entstehende Struktur lässt sich nach vorher definierten Regeln festlegen. Die Katalogdaten werden in einer logischen Baumstruktur geordnet, die dem klassischen Printprodukt Katalog nachempfunden ist (Titelseite, Inhaltsverzeichnis, Kapitel, etc.). Um den Sinn und Nutzen einer Verwendung von XML zu verstehen, werden im Folgenden der Grundgedanke und die fundamentalen Eigenschaften näher betrachtet.

2.1.1.1 XML als Informationsträger

XML ist ein Standard, der zur Erstellung von maschinen- und menschenlesbaren Dokumenten in Form einer Baumstruktur entwickelt wurde.[1] Die Norm regelt dabei ausschließlich die Struktur solcher Dokumente. Die Namen der einzelnen Strukturelemente (XML-Elemente) für eine konkrete XML-Anwendung lassen sich frei wählen. Ein XML-Element kann ganz unterschiedliche Daten enthalten bzw. beschreiben, als weit verbreitetes Beispiel Text, aber auch Grafiken oder abstraktes Wissen. Es gibt keine vorgeschriebenen Funktionen oder Befehle, sondern nur eine Festlegung über die Struktur der einzelnen Elemente, die sich flexibel und projektspezifisch wählen lassen. Die Dokumenttyp-Deklaration (im Folgenden DTD) bildet dafür „eine Grammatik für eine Klasse von Dokumenten“[2] und beschreibt unter anderem, welche Elemente und Attribute in einer XML-Datei enthalten sein dürfen bzw. in welchem Zusammenhang sie stehen. Auf Grundlage dieser Grammatik wird das Dokument abgearbeitet, um feststellen zu können, ob es wohlgeformt ist und alle formalen Kriterien erfüllt werden.

---

[1] Vgl. „Herkunft und Ziele“ In: Extensible Markup Language (XML) 1.0, Stand: 2003, Kapitel 1.1; Online im Internet: URL <http://edition-w3c.de/TR/2000/REC-xml-20001006> (abgerufen am 24. August 2005)

[2] „Prolog und Dokumenttyp-Deklaration“ In: Extensible Markup Language (XML) 1.0, Stand: 2003, Kapitel 2.8; Online im Internet: URL <http://edition-w3c.de/TR/2000/REC-xml-20001006> (abgerufen am 24. August 2005)

Ein entscheidender Grundgedanke von XML ist es, Daten und ihre Darstellung voneinander zu trennen. Somit erreicht man mit XML die gewünschte Medienneutralität, um aus einem Datenbestand über verschiedene Konvertierungen mehrere Ausgabeformate zu erzeugen. Dies ist unter anderem Ziel eines Cross-Media-Unternehmens. Der Datenbestand liegt medienneutral vor und kann mit einem spezifischen Ausgabelayout dargestellt werden. Dies passiert bei der @ computational design GmbH für eine Internet-Ausgabe mit der Extensible Stylesheet Language Transformation (im Folgenden XSL-T) und für eine PDF-Ausgabe mit der XSL-T-Erweiterung Extensible Stylesheet Language - Formatting Objects (im Folgenden XSL-FO).

2.1.1.2 XSL-T und XSL-FO - Formatierung für die Darstellung von Informationen

XSL-T stellt wie XML einen Standard dar, der zur Erzeugung von Layouts (Stylesheets) aus XML-Dokumenten definiert wurde. Diese Stylesheets können den medienneutralen XML-Dokumenten zugewiesen werden und für die verschiedensten, speziellen Medien formatiert werden (Internet, Druck, Mobiltelefone; siehe Abbildung 1). Ein XSL-T-Prozessor parst (analysiert) dafür das XML-Dokument und verbindet den Inhalt mit der ihm im Stylesheet zugewiesenen Formatierung.

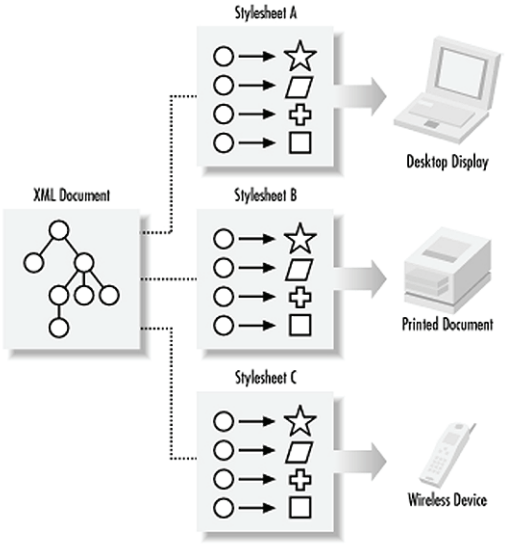

Abbildung 1: Verwendung von verschiedenen Stylesheets für unterschiedliche Ausgaben[3]

[3] Vgl. Ray, Erik T.; „Learning XML“, O'Reilly Verlag, 1. Auflage, 2001, S. 90

Die in der Abbildung dargestellte Transformation muss man sich anhand der Baumstruktur von XML verdeutlichen. Von einem Wurzelelement (root) zweigen einzelne Äste bzw. Knoten (nodes) ab und bilden somit Kindelemente der Wurzel. Eine Verwurzelung/-zweigung kann unendlich weit fortgeführt werden und somit Elemente immer detaillierter beschrieben werden. Bei einer Katalogproduktion kann man sich die Root als den Katalog selbst vorstellen, Kindelemente würden dabei beispielsweise die Titelseite, das Inhaltsverzeichnis oder ein Container für die Kapitel sein. Dies soll noch einmal in der folgenden Abbildung verdeutlicht werden.

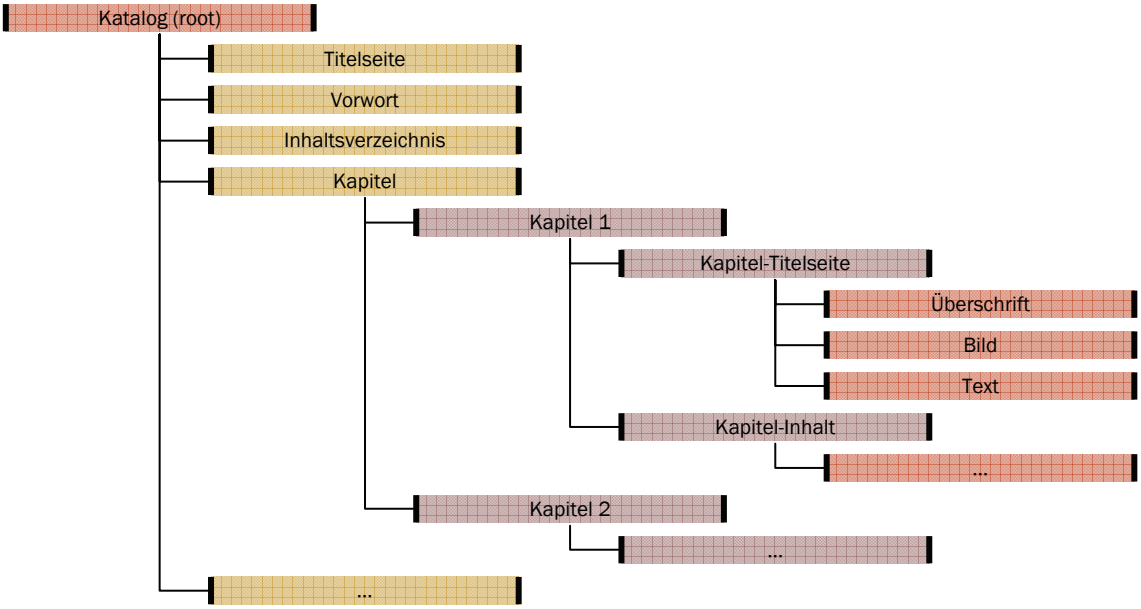

Abbildung 2: XML-Baumstruktur beispielhaft für eine Katalogproduktion

Jedem Element im XML-Baum ist ein Name, idealerweise auch eine ID, die jeweils nur einmal vorkommen darf, zugewiesen. Möchte man im Stylesheet Formatvorgaben für einen bestimmten Knoten/Knotentyp/Element machen, muss dazu der Name/die ID als Referenz angeben werden. Bei einer Ausgabe durchläuft der Parser den XML-Datenstrom bis er auf die angegebene Referenz stößt und liefert die Inhalte eines Knotens in der gewünschten Formatierung zurück.

Typischerweise werden mit XSL-T, im Bezug auf eine Ausgabe von XML-Daten, HTML-Dateien für das Internet gestaltet. Dabei wird auf die spezifischen Formatierungsmöglichkeiten von Text, Grafik und Bildern zurückgegriffen. Durch die Layoutierung nach den Vorgaben des World Wide Web Consortium (W3C) für Cascading Stylesheets (CSS) ist es inzwischen sogar möglich, eine relativ gezielte und vor allem einheitliche Ausgabe mit verschiedenen Browsern zu erreichen.

Die Anforderungen an eine Ausgabe für den Druck sind allerdings viel höher und komplexer, wenn man an den Seitenaufbau von PDF oder die Behandlung von Schrift denkt. Für diese speziellen Anforderungen dient eine Erweiterung von XSL-T.

XSL-FO (Extensible Stylesheet Language - Formatting Objects) ist eine XML-Anwendung, die beschreibt, wie Text, Grafiken, Linien und andere Informationen auf einer Seite angeordnet werden. Mit Hilfe von XSL-FO ist es möglich, PDF-Dateien zu erzeugen, mit denen bei einer entsprechenden Weiterverarbeitung qualitativ hochwertige Druckerzeugnisse auf Papier hergestellt werden können. Optional lassen sich aber auch PDF-Dateien für das Internet generieren, die für den Office-Bereich eingesetzt werden können und die dortigen Anforderungen gänzlich erfüllen.

Die XSL-FO-Dokumentation beschreibt, wie sich Inhalte auf einer Seite durch Abstände vom Seitenrand sehr genau platzieren lassen. Dazu wird die Layoutstruktur einer Seite über Blöcke angelegt, für die dann genauere Kriterien (wie z.B. Typographie, Farbinformationen, etc.) für das spätere Layout spezifiziert werden. Weiterhin können Seitenlayoutvorlagen, wie sie aus Layoutprogrammen bekannt sind, angelegt und z.B. Schriften eingebettet werden. Nachteilig ist allerdings, dass zur Interpretation der XSL-FO Befehle ein kommerzieller PDF-Generator erworben werden muss. Bei diesen Erzeugerprogrammen wird von den Herstellern allerdings größtenteils außer Acht gelassen, dass das erzeugte PDF, neben dem Generieren aus einer Internetanwendung heraus für den Ausdruck am Arbeitsplatz oder Zuhause, auch für eine hochwertige Druckproduktion genutzt werden könnte.

Befehle und Funktionen zur Steuerung eines Color Managements für PDF-Dateien werden z.B. vom PDF-Generator XEP von RenderX, der von @ computational design GmbH eingesetzt wird, auch im Hinblick auf eine PDF/X-3-Ausgabe, nicht bzw. unvollständig unterstützt.

Erst nach und nach wächst das Bewusstsein bei den Herstellern, dass auch diese Tools die für die Druckindustrie spezifischen Parameter erfüllen müssen, denn der Bedarf an solchen Programmen steigt.

2.1.2 Dynamische PDF-Generierung

Der PDF-Generator XEP von der amerikanischen Softwarefirma RenderX ist eine Anwendung zur Erstellung von Seitenlayouts und ist in der Lage XSL-FO-Befehle zu interpretieren. Dazu wird der XML-Baum über ein XSL-Stylesheet geparst und ein FO-Baum erstellt, der jetzt nicht mehr einen logischen, sondern einen layoutorientierten Ablauf beschreibt. Somit ist genau festgelegt, an welcher Stelle Inhalte auf der Seite platziert werden. Die Grundlage von XEP für eine PDF-Generierung bildet die „Adobe PDF Reference“, in der die PDF-Spezifikationen von Adobe dokumentiert und detailliert beschrieben werden. XEP wurde nach den dort gemachten Vorgaben programmiert. Auf diesem Programmiergrundgerüst wurde eine Java basierende Umgebung aufgesetzt und die PDF-spezifischen Eigenschaften können darüber aufgerufen werden. Standardmäßig lässt sich mit der neusten Version von XEP (4.2) ein PDF 1.4 kompatibles PDF erzeugen. Der Schlüssel zu einer zielgerichteten PDF-Erzeugung liegt in der Konfiguration von XEP.

Die @ computational design GmbH verwendet XEP in der Version 3.7. Mit dieser Version lassen sich PDF 1.3 kompatible PDF-Dateien erstellen. Der sonstige Funktionsumfang ist allerdings sehr eingeschränkt und es können nur detaillierte Angaben zu Schriften und zu der Verwendung einer Silbentrennung gemacht werden. Die Möglichkeit weitere Angaben zu Color Management, Multimedia-Funktionalitäten oder sogar einer PDF/X-Ausgabe zu machen ist bei dieser Version noch nicht möglich. Bis zur aktuellen Version 4.2 wurde der Funktionsumfang ausgebaut.

2.1.3 Ablaufbeschreibung

Die folgende Abbildung soll noch einmal den Ablauf des pablish® - catalog publishing-Workflows verdeutlichen. Der User kann über das Frontend Texte und Bilder hinzufügen, editieren oder löschen und sich diese anzeigen lassen. Im Hintergrund verarbeitet pablish® - catalog publishing als CMS die Daten und legt alle eingespeisten Bilder unter neuem Namen zentral auf der Festplatte ab.

Im Hinblick auf eine Druckproduktion werden bei @ computational design GmbH Bilder im TIFF-Format mit entsprechend hoher Auflösung eingebunden. Bei einer Ausgabe wird aus dem CMS eine XML-Datei mit allen relevanten Informationen für die Katalogproduktion in der angesprochenen Baumstruktur erzeugt. Die XML-Datei

enthält unter anderem auch die Verweise zu den aus dem CMS abgespeicherten Bildern.

Die XML-Datei und zur Verfügung stehende Stylesheets werden für eine PDF-Ausgabe anschließend von XEP verarbeitet. Die Layoutinformationen aus den Stylesheets werden mit den Daten aus der XML-Datei verbunden, es werden also Inhalt und Layout zusammengeführt. Die generierte PDF-Datei ist im Hinblick auf den PDF/X-3-Standard nicht druckreif. Die Gründe und Ursachen dafür sollen im weiteren Verlauf der Arbeit aufgezeigt werden.

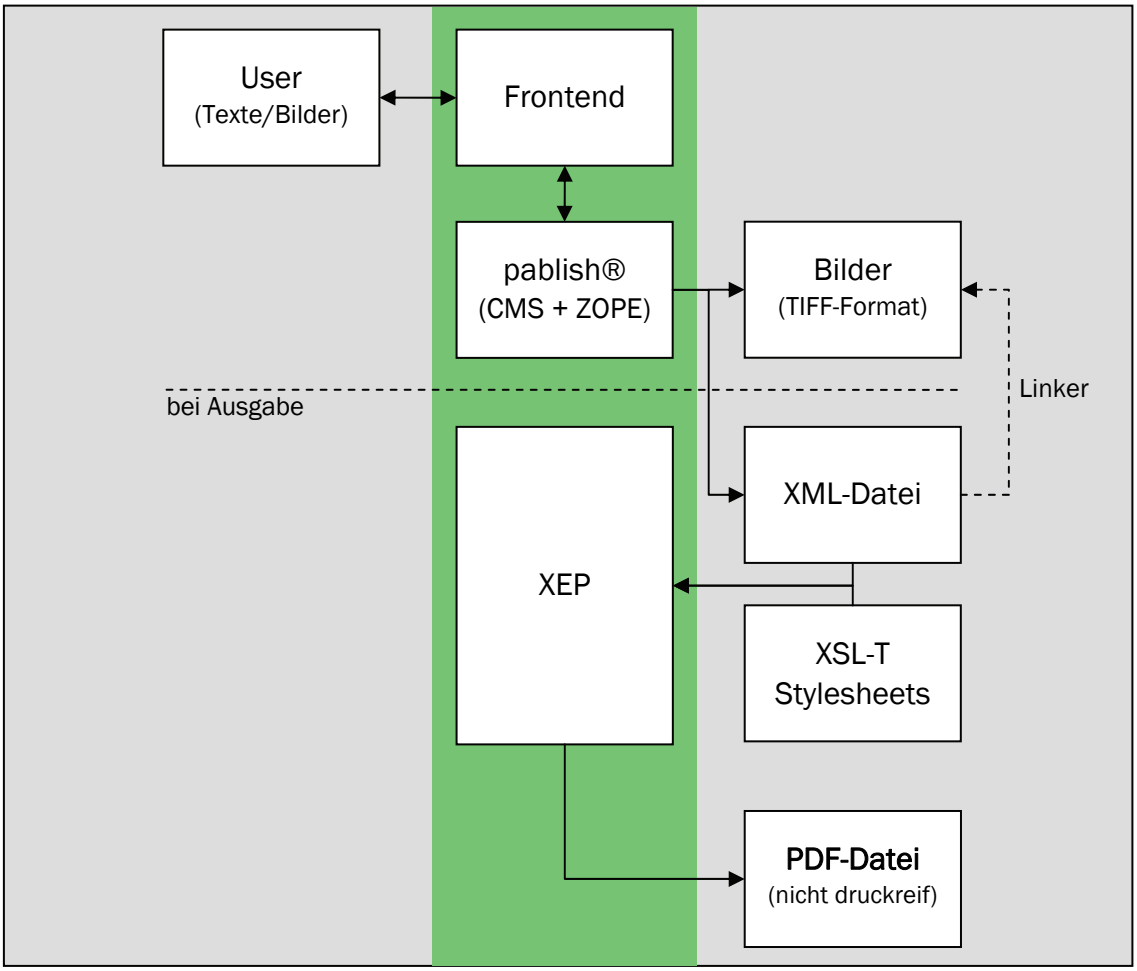

Abbildung 3: Der pablish® - catalog publishing-Workflow

2.2 Schwachstellen bei der PDF-Erzeugung

Das erzeugte PDF liefert hinsichtlich der Bilddaten unerwünschte Ergebnisse. Farbprofile, die in Bilder eingebettet sind, bleiben nach der PDF-Generierung nicht erhalten.

2.2.1 Farbprofile und Bilddatenbehandlung

Farbmanagement mit Farbprofilen hat in der Druckindustrie einen immer größeren Stellenwert eingenommen. Das International Color Consortium (im Folgenden ICC) wurde 1993 mit der Absicht gegründet, eine Vereinheitlichung von Farbmanagementsystemen zu erreichen. Die ICC-Spezifikation beschreibt ein Dateiformat zur farbmetrischen Charakterisierung von Ein- und Ausgabegeräten.[4] Bis heute hat sich der Gebrauch von ICC-Profilen weitestgehend durchgesetzt. ICC-Profile können in Bilddatenformaten und Seitenbeschreibungen eingebettet werden. PDF bietet hierbei den Vorteil, dass auch bei mehreren Bildern jedes Bild durch ein eigenes Profil genau beschrieben sein kann und ausgabespezifisch umgerechnet werden kann.

Den Vorteil und die Notwendigkeit von ICC-Profilen soll ein kleines Beispiel verdeutlichen: Man fotografiert unter Studiobedingungen (konstante Beleuchtungsbedingungen) mit zwei verschiedenen Digitalkameras dieselbe Szene, in der ein Testchart enthalten ist. Das Testchart dient dazu, ein ICC-Profil für die spezifischen, farbmetrischen Eigenschaften des Gerätes zu erstellen und beinhaltet definierte Farbfelder, für die farbmetrische Referenzmesswerte vorliegen. ICC-Profile lassen sich z.B. mit Programmen wie Profile-Maker, Colortune oder Printopen erstellen. Dabei werden die Beziehungen der von der Digitalkamera aufgezeichneten Farbwerte zu Referenzwerten ermittelt und als das Profil abgespeichert.

Von dem aufgenommenen Foto werden zwei Versionen abgespeichert. Nur bei der zweiten Version werden die erstellten ICC-Profile anschließend in die Bilder eingebettet. Zu einer weiteren Verarbeitung werden die Fotos an einem Monitor dargestellt. Dafür findet jeweils eine Transformation der Farbräume vom Eingabe- in den

[4] Vgl. Drümmer, Olaf; Merz, Thomas; „Die PostScript- & PDF-Bibel“, dpunkt-Verlag, Heidelberg; 2. Auflage, 2002, S. 225

Ausgabefarbraum. Bezugssystem dafür ist ein geräteunabhängiger Referenzfarbraum, der Profile Connection Space (im Folgenden PCS).

Bei den zuerst gemachten Aufnahmen (ohne Profile) führt dies zu unterschiedlichen Darstellungen am Monitor, da die Ausgangsdaten ohne Anpassung über den PCS auf dem Farbraum des Ausgabegerätes abgebildet werden.

Ganz anders erscheint die Darstellung der Bilder, die ICC-Profile enthalten. Hierbei sehen die Fotos im Idealfall identisch aus. Bei der Umrechnung in den geräteneutralen Referenzfarbraum werden die ICC-Profile mit den Ausgangsdaten verrechnet und anschließend in den Ausgabefarbraum transformiert.

Bei einer Katalogproduktion können Bilder über das CMS eingebunden werden. Oftmals werden diese von z.B. von Fotografen angefertigt oder aus Bilddatenbanken ausgesucht. Idealerweise sind in diesen Dateien ICC-Profile der Eingabegeräte (z.B. Kamera oder Scanner) enthalten, damit bei einer späteren Ausgabe eine optimale Farbwiedergabe erzielt werden kann. Sind keine ICC-Profile vorhanden bzw. werden sie während der Verarbeitung im Workflow entfernt kann keine gezielte Farbwiedergabe garantiert werden. Mit diesem Problem beschäftigt sich der folgende Absatz.

Bei der Katalogproduktion mit dem pablish® – calatog publishing-Workflow von @ computational design GmbH wurde das erzeugte PDF hinsichtlich der Bilder und einer ICC-Profilbehandlung untersucht. Dazu eignen sich so genannte Preflight-Tools, wie z.B. Enfocus Pitstop, mit denen man sich detaillierte Informationen über Objekte einer PDF-Datei (Texte, Bilder, Grafiken, etc.) komfortabel in Adobe Acrobat anzeigen lassen kann. Dabei stellt man fest, dass alle wichtigen Bildinformationen, wie Auflösung, Größe, etc., korrekt in den PDF-Datenstrom übernommen werden. Allerdings sind Profile, die ursprünglich in Bilder eingebettet waren, nach der PDF-Generierung nicht mehr vorhanden, wurden demnach nicht beachtet und entfernt. Das Preflight-Tool zeigt weiterhin, dass der Farbraum, in dem sich das Bild befindet, nicht verändert wurde und das Bild in einem Device-Farbraum (RGB/CMYK/Grau) vorliegt. Die Ursache für das Entfernen der ICC-Profile kann im Workflow am CMS oder dem PDF-Generator XEP liegen, da nur dort die Bilder verarbeitet werden. Die über das Frontend in das CMS eingelesenen Bilder werden automatisch umbenannt

und auf der Festplatte abgelegt. Dabei bleiben eingebettete ICC-Farbprofile erhalten. Dies bedeutet, dass die Veränderung bei der PDF-Generierung stattfinden muss.

Das PDF-Erzeuger-Tool XEP scheint die ICC-Profile zu ignorieren bzw. zu entfernen, wenn die PDF-Datenstruktur erzeugt wird. Im Weiteren muss untersucht werden, ob die Ursache dafür an der Konfiguration oder an den Möglichkeiten des Programms selbst liegt.

3. Workflow-Optimierung im Hinblick auf PDF/X-3 und Color Management

Der beschriebene Workflow lässt sich hinsichtlich der aufgezeigten Schwachstellen optimieren. Ein Eingriff ist an mehreren Stellen möglich bzw. nötig.

3.1 PDF/X-3 und Color Management

Speziell für die hohen Anforderungen der Druckindustrie wurde der PDF/X-3-Standard entwickelt, durch den eine fehlerfreie Produzierbarkeit sichergestellt wird. Alle Elemente, die PDF flexibel, dynamisch und interaktiv machen (z.B. das Einbetten von Multimedia-Elementen oder Transparenzen), können zu einer unerwünschten Ausgabe führen. Vielmehr wird mit dem PDF/X-3 Standard nicht nur festgelegt, was die PDF-Datei nicht enthalten darf, sondern die Inhalte, die enthalten sein müssen und ein schlankes, aber effizientes PDF geliefert.

Color Management ist ein wichtiges Instrument für eine gezielte Ausgabe von Druckprodukten. Der richtige Umgang mit Farbe spart viel Zeit und Geld. Dies fängt schon bei einem Bewusstsein für die charakteristischen Unterschiede von Ausgabegeräte und Ausgabemethoden an. Die Handhabung von ICC-Farbprofilen oder der Einsatz von Color Management führt oftmals eher zu Abschreckung oder Ablehnung.

Charakteristische Merkmale von Eingabe- und Ausgabegeräten können über Testcharts gemessen und so aus den gewonnenen Daten ICC-Profile erzeugt werden.[5]

Im Hinblick auf ein Color Management müssen für die PDF/X-3-Erzeugung einige Aspekte berücksichtigt werden. Eine ausführliche Beschreibung und Definition für das Color Management in diesem Bereich findet man in der Broschüre „Das Standard-Datenformat für die Druckproduktion PDF/X-3“ von Olaf Drümmer, die vom Bundesverband Druck und Medien e.V., der Ifra und Ugra herausgegeben wurde: „ICC-Profile haben eine besondere Bedeutung für PDF/X. Zum einen können in einer PDF/X-3-Datei ICC-Profile verwendet werden, um ein Bild oder ein anderes Objekt farblich genau zu definieren. Es ist übrigens keinesfalls vorgeschrieben, dass in einer PDF/X-3-Datei ICC-Profile für Bilder oder andere Objekte verwendet werden müssen [...]. Zum anderen wird ein ICC-Ausgabeprofil im so genannten „OutputIntent“ (frei zu übersetzen als „Ausgabeabsicht“) verwendet, um anzugeben, für welche Druckbedingung eine PDF/X-3Datei erstellt wurde. Hierbei ist zu beachten, dass sich ICC-Profile für Objekte auf einer PDF-Seite beim Darstellen oder Ausgeben der PDF/X-3Datei sich immer auf das Ergebnis auswirken. Das ICC-Ausgabeprofil im OutputIntent hingegen dient zunächst nur Informationszwecken – es teilt dem Empfänger einer PDF/X-3-Datei mit, für welchen Druckprozess die Datei erstellt wurde. Weiterhin sollte es verwendet werden, um PDF/X-3-Dateien, die nicht nur CMYK und Schmuckfarben, sondern auch ICC-basierte oder Lab-basierte Farben verwenden, korrekt zu separieren. Schließlich ist vorgeschrieben, dass für das Erstellen eines Proofs das als OutputIntent eingebettete ICC-Ausgabeprofil im Zusammenspiel mit dem ICC-Profil für das Proofsystem zu benutzen ist. Für die Praxis extrem wichtig ist hierbei, dass das ICC-Ausgabeprofil im OutputIntent nicht automatisch zum Einsatz kommt, sondern nur durch entsprechenden Vorgaben bzw. Einstellungen im Verarbeitungsprozess des Empfängers“[6].[7]

Die anderen Bereiche der PDF/X-3-Normforderungen sollen im folgenden Teil der Arbeit am praktischen Beispiel des erzeugten PDF-Katalogs dargestellt werden.

---

[5] Weiterführend zu Color Management: Internetseiten der European Color Initiative (ECI) - http://www.eci.org und des International Color Consortiums (ICC) - http://www.eci.org

[6] Drümmer, Olaf; „Das Standard-Datenformat für die Druckproduktion PDF/X-3“, 2003, S. 16

[7] Weiterführend zu PDF/X-3: Internetseiten http://www.pdfx3.de und http://www.prepress.ch

3.2 Steuerung der PDF-Generierung

3.2.1 Software Development Kit (SDK)

Der kalifornische Softwarehersteller RenderX stellt für XEP kostenpflichtig ein Software Development Kit (SDK) mit den Namen „Connectivity Kit“ zur Verfügung. Damit soll man die Möglichkeit haben, über die Programmierschnittstelle das Programm nach individuellen Bedürfnissen anzupassen bzw. zu erweitern oder eigene darauf basierende Anwendungen zu erstellen.

Für Testzwecke steht eine Trial Version zur Verfügung, mit der man bereits einen Einblick in den Umfang und die Dokumentation des SDK hat. Leider blieben damit alle Versuche, eine Anpassung des Farbmanagements vorzunehmen, erfolglos. Ein Eingriff in den PDF-Produktionsprozess ist nicht weit genug möglich, da die von RenderX programmierten Klassen und Objekte, die zur Erzeugung von PDF dienen, nicht zur Verfügung stehen.

3.2.2 Anpassen der Workflow-Komponenten

Sowohl bei der Dateneingabe über das CMS als auch bei der Konfiguration von XEP selbst können bereits erste Ansätze für eine erfolgreiche PDF/X-Ausgabe gemacht werden. Betrachtet man einmal die grundlegenden Anforderungen, die für eine PDF/X-3-Ausgabe gestellt werden, bedarf es größtenteils nur einiger kleiner Änderungen der Einstellungen.

Die nachfolgende Tabelle liefert eine Aufstellung über die wichtigsten Normforderungen des PDF/X-3-Standards, die für eine dynamische Katalogproduktion interessant sind. Die aus XEP erzeugte PDF-Datei wurde mit einem Preflight-Tool daraufhin untersucht, um festzustellen, ob die Forderung erfüllt werden bzw. was man unternehmen kann, um eine Konformität zu erreichen.

Normforderung	erfüllt	nicht erfüllt	Bemerkung
PDF-Version (1.3)	x		XEP 3.7 erzeugt PDF 1.3
Präzise Angaben zur Seitengeometrie		x	Angaben zur TrimBox fehlen, kann mit PDFlib nachgeholt werden
Einbetten der verwendeten Schriften		x	Kann mit XEP konfiguriert werden
Keine Verwendung von Transparenzen	x		Keine transparenten Grafiken für den Katalog vorhanden
Kompression von Bildern (keine LZW-Komprimierung)		x	LZW-Komprimierung vorhanden, PDFlib ersetzt automatisch LZW durch ZIP
Angaben zum Überfüllungszustand		x	Angabe fehlt, wird mit PDFlib nachgeholt
PDF/X-Identifikation		x	Angabe fehlt, wird mit PDFlib nachgeholt
Ausgabeabsicht definiert		x	Angabe fehlt, wird mit PDFlib nachgeholt
Farbigkeit definiert (nur geräteabhängige Farben – CMYK/Schmuckfarben bzw. geräteabhängige Farben – z.B. ICC-basiertes RGB)		x	Die Aufzählungszeichen im Katalog (•) stammen nicht aus einem Schriftsatz, sondern werden als RGB-Bild eingebettet.

Eine Verwendung von RGB-Bildern ist möglich, allerdings muss der Farbraum genau definiert werden. Dafür bietet sich das RGB-ICC-Profil der European Color Initiative (ECI) an, das als Arbeits- und Austauschfarbraum für Werbeagenturen, Verlage, Reprounternehmen und Druckereien empfohlen wird.[8] Es gilt zu klären, ob es möglich ist, statt Bildern für eine Aufzählung die Aufzählungszeichen aus dem verwendeten Schriftsatz zu benutzen, um eine Qualitätssteigerung und Optimierung zu erreichen.

Im Katalog werden kleine Graustufen- und RGB-Bilder, deren Farbe der Hintergrundfarbe entspricht, als Abstandshalter verwendet. Dies ist bei einer HTML-Programmierung für das Internet durchaus legitim und üblich. Allerdings sollte versucht werden, die Verwendung solcher Bilder zu vermeiden. Die Dateigröße der

[8] Vgl. European Color Initiative, „Arbeitsfarbräume“, Online im Internet: URL <http://www.eci.org/eci/de/044_arbeitsfarbraeume.php> (abgerufen am 20. September 2005)

PDF-Datei steigt unnötig an, vor allem, wenn noch ein ICC-Profil in diese Bilder eingebettet wird. Der OutputIntent, der Überfüllungsschlüssel (Trapped Key), die PDF/X-Identifikation und die fehlenden Angaben über die Seitengeometrie können nachträglich mit der PDFlib-Programmierbibliothek eingetragen werden.

Wegen einer nicht PDF/X-3-konformen Kompression von TIFF-Bildern (die LZW-Komprimierung ist lizenzrechtlich geschützt) kann das Nachbearbeiten der Katalogbilder mit einer Bildbearbeitungssoftware, z.B. Adobe Photoshop, nötig werden. Dazu müsste das Bild mit einem anderen Kompressionsalgorithmus neu abgespeichert werden. Als Alternative kann auch eine automatisierte, programmiertechnische Lösung gewählt werden, indem man einen TIFF-Konverter programmiert bzw. integriert. Mit diesem können Bilder mit LZW-Komprimierung erkannt und mit dem ZIP-Kompressionsalgorithmus umgerechnet werden. Diese Arbeit kann PDFlib zusätzlich übernehmen. Bilder werden bei einer Umwandlung oder dem Einbetten in PDF generell unverändert „durchgeschleust". Im Hinblick auf eine PDF/X-3-Erzeugung wird auch auf vorhandene Kompressionsformate getestet und, wenn diese nicht PDF/X-3-konform sind, durch ZIP ersetzt.

Mit XSL-FO lassen sich grundlegende Angaben über die verwendete Typographie (Schriftart/-größe/-schnitt, etc.) machen. Wie bereits erwähnt, werden Layoutinformationen in XSL-FO über Blöcke definiert. In diesen werden auch Informationen über die verwendete Typographie gemacht. „Für jeden Block werden die entsprechend spezifizierten typographischen Attribute wiederholt"[9].

```
<fo:block color="#000000" font-family="Helvetica" font-size="9.5pt"
text-align="start"> Inhalt des Blockes </fo:block>
```

Das Beispiel soll zeigen, dass die Möglichkeiten für die Textformatierung schon sehr umfangreich sind, neben der Schriftfarbe, Schriftart, Schriftgröße (Maßangaben in Punkt oder Pixel) kann auch die Ausrichtung des Inhaltes in diesem Block angegeben werden. Bei der PDF-Generierung werden die Schriftinformationen nur teilweise aus den Stylesheets extrahiert.

[9] Montero Pineda, Manuel; Krüger, Manfred, „XSL-FO in der Praxis – XML-Verarbeitung für PDF und Druck", dpunkt-Verlag, Heidelberg, 1. Auflage, 2004, S. 13

Die grundlegenden Informationen für die Fonts, die XEP benötigt, lagern allerdings in einer externen XML-Datei (fonts.xml). Dort werden die verwendeten Schriftarten aufgeführt und Angaben zu ihrer Verwendung gemacht.

```
<font name="Courier" embed="false"></font>
```

Der Parameter für das Einbetten (embed) muss hierbei auf „true“ gesetzt werden, damit die Schrift bei der PDF-Generierung auch wirklich mit in das PDF eingebettet wird.

3.3 IMG2PDF – Erzeugung von PDF-Dateien aus Bilddaten

Durch die gemachten Untersuchungen wurde festgestellt, dass über XEP keine angemessene Steuerung der Bilddaten erfolgen kann. Dies führt zu der weiterführenden Überlegung, dass eine Anpassung des Workflows vor der Datenverarbeitung mit XEP passieren muss. Das Konvertieren der TIFF-Bilddaten in ein anderes Grafikformat scheint wegen der XEP-spezifischen Eigenschaften nicht sehr sinnvoll, da der gleiche Fehler wieder auftreten würde. Allerdings lassen sich ICC-Profile nicht nur in Grafikformate, sondern auch in Seitenbeschreibungen einbetten. Da die Ausgabe in PDF erfolgt, muss eine Lösung gefunden werden, die sich ohne weiteres in PDF einbetten lässt. Es macht Sinn, alle Bilder des Kataloges in PDF zu konvertieren, weil sich die anderen Grafikformate nicht eignen. PDF in PDF einzubetten ist ab der PDF-Version 1.2 mit so genannten XObjects möglich. Letztendlich sollen keine TIFF-Bilder mehr in XEP eingelesen werden, sondern PDF-Dateien, die die Bilder mit den eingebetteten Profilen enthalten. Dazu wird das Bild im Grunde genommen von der PDF-Struktur umschlossen und als Objekt eingebettet („Image to PDF“: IMG2PDF). Möglich wäre es auch, das Bild in den Content Stream der PDF-Datei einzulesen, dazu wird aber in der PDF Reference nur bei kleinen Datenmengen geraten, da dies zu einer größeren Datengröße führt. Weiterhin ist mit einem XObject das Referenzieren desselben Bildes innerhalb der PDF-Datei möglich, ohne das Bild selbst mehrmals einbetten zu müssen.[10]

[10] Vgl. Adobe Systems Incorporated, „PDF Reference, fifth Edition, Adobe® Portable Document Format, Version 1.6”, 2004, S. 325

Ab PDF-Version 1.3 ist der Umgang mit ICC-Profilen möglich. Da PDF/X-3 auf der PDF-Version 1.3 basiert, liegt der gewählte Lösungsansatz innerhalb der PDF/X-3-Spezifikationen. Mit der IMG2PDF-Umwandlung ist es nun möglich für die weitere Verarbeitung mit XEP die Bilder als PDF einzulesen. Dafür muss aber auch die XML-Datei, die die Quellinformationen über den Speicherort der Bilder enthält, geändert werden, da die Quellpfadinformationen nun auf die PDF-Bilder verweisen müssen. Diese Anpassung selbst muss im CMS vorgenommen werden.

3.3.1 PDFlib – Einbinden einer PDF-Bibliothek

Bei der PDFlib-Produktfamilie handelt es sich um ein Toolkit zur dynamischen Erzeugung und Verarbeitung von PDF-Dateien. PDFlib selbst stellt kein Programm für Endanwender dar, sondern vielmehr eine Programmierbibliothek, die einem die Interna der PDF-Erzeugung ersparen soll und eine Programmierschnittstelle bietet.[11]

Der Kern von PDFlib ist in ANSI C geschrieben und über ein API (Programmierschnittstelle – Application Programming Interface) auch von einer Vielzahl anderer Programmiersprachen aus verfügbar.

Laut der PDFlib GmbH ist PDFlib in der Version 6 „die erste Software am Markt, mit der PDF-Dateien erzeugt und verarbeitet werden können, die den neuesten Ausgaben der in der Druckvorstufe relevanten PDF/X-Standards entsprechen (PDF/X-1a:2003, PDF/X-2:2003 und PDF/X-3:2003)“[12]. Weiterhin wurde PDFlib, genau wie XEP, nach den Vorgaben der Adobe PDF Reference programmiert. Zusätzlich zu diesen Grundanforderungen wurden sinngemäß nachvollziehbare Befehle eingebettet. Damit lässt sich die PDF-Generierung vereinfachen. Für das Erzeugen eines PDF-Dokuments muss zum Beispiel nur noch der Befehl zum Anlegen einer Seite mit der entsprechenden Größe angegeben werden.

---

[11] Vgl. PDFlib GmbH, „PDFlib, PDFlib+PDI, PPS; Dynamische PDF-Generierung – und mehr!“, Online im Internet: URL <http://www.pdflib.com/pdffiles/PDFlib-6.0-datasheet_D.pdf> (abgerufen am: 10.09.2005)

[12] PDFlib GmbH, „PDFlib-Produktfamilie, Version 6 – Die Highlights“, Online im Internet: URL <http://www.pdflib.com/pdffiles/PDFlib-6-Highlights_D.pdf> (abgerufen am: 10.09.2005)

Im Unterschied zu XEP bietet PDFlib „keine Funktionen zur Berechnung des Seitenlayouts (Formatierung), sondern unterstützt die grafischen Grundelemente von PDF und bietet ausführliche Unterstützung für alle gängigen Fontformate und Zeichensätze“[13].

Auf die IMG2PDF-Programmierung bezogen steht bei PDFlib eine Option „pdfx“ bereit, die noch mit einem Attribut für den gewünschten PDF/X-Standard, belegt werden kann. Wird die Option gewählt, werden automatisch die grundlegenden Parameter für eine PDF/X-Datei geladen (z.B. wird die PDF-Version auf 1.3 und die PDF/X-Identifikation auf den entsprechenden, vorgegebenen Wert gesetzt). Dies erleichtert das Erstellen von PDF-Dateien, wenn man sich nicht mit den PDF-Datei-Spezifikationen beschäftigen möchte. Trotzdem hat man bei Bedarf alle Möglichkeiten, noch einzelne Parameter für die individuellen, spezifischen Gegebenheiten anzupassen und zu verändern.

Bei der Erstellung der PDF-Dateien dient PDFlib für diese spezielle Anwendung als ein „PDF-Wrapper“, indem standardmäßig die Bilder als XObjects in die PDF-Datei eingebettet werden.

Der folgende Ausschnitt aus IMG2PDF (vollständiger Inhalt im Anhang) soll die Funktionsweise der PDFlib verdeutlichen. Die einleitenden Kommentare (Beginn gekennzeichnet mit //) sollen kurz den in der folgenden Zeilen verwendeten Befehl erläutern. In den folgenden Klammern finden sich die Optionen und deren Attribute.

[13] Drümmer, Olaf; Merz, Thomas; „Die PostScript- & PDF-Bibel“, dpunkt-Verlag, Heidelberg; 2. Auflage, 2002, S. 342

PDF- und Color Management Informationen

```
// Das zu erzeugende PDF-Dokument soll vom Typ PDF/X sein und
// nach der Spezifikation PDF/X-3:2002 angelegt werden
p->set_parameter("pdfx", "PDF/X-3:2002");

// In Bilder eingebettete ICC-Profile sollen berücksichtigt werden
p->set_parameter("honoriccprofile", "true");

// Eingebettete ICC-Profile werden überwacht und bei Problemen wird
// eine Fehlermeldung ausgegeben
p->set_parameter("iccwarning", "true");

// Das ICC-ISOcoated-Profil wird als farbmetrische Charakterisierung
// des Ausgabefarbraums (OutputIntent) festgelegt
p->load_iccprofile("ISOcoated.icc", "usage=outputintent");
```

Bild- und Seiteninformationen

```
// Das Bild "bild.tif" wird in den Speicher geladen
// „ignoremask" bewirkt, dass Transparenzen verworfen werden
image = p->load_image("auto", "bild.tif", "ignoremask");

// Seite wird erzeugt, Größe A4
p->begin_page_ext(a4_width, a4_height, "");

// Seitengröße auf die Größe des Bildes anpassen
p->fit_image(image, 0.0, 0.0, "adjustpage");

// Das Bild wird aus dem Speicher entfernt
p->close_image(image);

// Die Seitenerstellung wird beendet und die Seite geschlossen
p->end_page_ext("");
```

Alle Befehle zur PDF-Erzeugung und deren Steuerung können statisch in der C++-Datei enthalten sein. Dies würde aber bedeuten, dass für jede Katalogproduktion der Quellcode neu kompiliert werden müsste, falls sich Einstellungen für die PDF-Generierung ändern. Um dies zu umgehen, werden alle relevanten Informationen für das Projekt bei Programmaufruf aus einer externen Datei zur Verfügung gestellt. Das Einlesen geschieht in Variablen, die dann bei der weiteren Verarbeitung verwendet werden. Als Informationsträger eignet sich dafür XML, zum einen wegen der Vorteile, die bereits angesprochen wurden, aber zum anderen auch wegen der Möglichkeit, Änderungen einfach und schnell vorzunehmen.

Wenn für ein neues Projekt alle Informationen und Daten für den zu produzierenden Katalog bereit stehen, können die kundenspezifischen Daten innerhalb von wenigen Minuten geändert werden. Die Angaben für die PDF/X-3-Erzeugung sollten unverändert bleiben, es sei denn, die Ausgabe erfordert eine Modifikation (z.B. hinsichtlich des OutputIntents oder PDF/X-Standards). Zum Ermitteln der Inhalte der XML-Datei benötigt man einen XML-Parser, wie z.B. TinyXML.

3.3.2 TinyXML

TinyXml wurde von Lee Thomason als OpenSource-Projekt entwickelt und ist ein XML-Parser für C++. Mit TinyXml ist es möglich, XML-Dokumente zu lesen und aus den vorhandenen Inhalten (Elemente, Attribute, etc.) C++-Objekte zu erzeugen, die das XML-Dokument repräsentieren.[14]

Beim Programmaufruf wird die Baumstruktur des XML-Dokuments geparst. Die Suche beginnt bei der Wurzel und erstreckt sich auf alle von da ausgehenden untergeordneten oder gleichrangig gestellten Elemente. Über in der C++-Datei festgelegte Kriterien werden aus bestimmten Elementen Objekte erzeugt. Um z.B. die Informationen für PDFlib verwenden zu können, muss man durch die von dem Benutzer festgelegte Struktur der XML-Datei navigieren. Für den unten abgebildeten Code-Auszug bedeutet das: job / pdflib / pdflib-parameter / info. Die entsprechenden Daten (in "") werden dann in Variablen abgespeichert.

```
<job>
     <pdflib>
          <pdflib-parameter>
               <info ID="pdfx" key="pdfx" value="PDF/X-3:2002"/>
               ...
          </pdflib-parameter>
          ...
     </pdflib>
     ...
</job>
```

[14] Vgl. Thomason, Lee, „TinyXML", Online im Internet: URL <http://sourceforge.net/projects/tinyxml>

3.3.3 Ablaufbeschreibung

Die Erweiterung des Workflows mit der IMG2PDF-Programmierlösung soll in der folgenden Abbildung noch einmal näher erläutert werden. Wird eine Ausgabe des Kataloges durch den Benutzer ausgelöst, wird vom CMS neben der XML-Datei auch eine Text-Datei erzeugt, die die Dateipfade zu allen verwendeten Bildern enthält.

Anschließend wird das kompilierte C++-Programm IMG2PDF aufgerufen. Die Pfadangaben aus der Text-Datei werden zeilenweise ausgelesen und bilden die Eingabe für die PDF-Konvertierung. Die erzeugte PDF-Datei soll letztendlich den gleichen Dateinamen mit der PDF-typischen Endungen „pdf“ erhalten. Damit sollen Verwechselungen ausgeschlossen werden. Die eigentliche Konvertierung findet mit PDFlib statt. Das Programm erhält die erforderlichen Parameter aus einer zusätzlichen XML-Datei. TinyXML parst dazu das Dokument und es werden die gewünschten Informationen in Variablen geschrieben, die bei Bedarf ausgelesen werden.

Die XML-Datei aus dem CMS, die die Katalogdaten enthält, referenziert durch einige Modifikationen nicht mehr auf TIFF-Bilder, sondern auf die erzeugten PDF-Dateien. Dies stellt sicher, dass für die anschließende PDF-Erzeugung mit XEP die PDF-Dateien anstatt der Bilder eingelesen werden. Somit erhalten wir, im Bezug auf Bilder im Katalog, ein druckreifes PDF. Mit den zusätzlichen Modifikationen an der Konfiguration von XEP kann die Ausgabe schon im Hinblick auf eine PDF/X-3-Datei vorbereitet werden. Die eigentlich gewünschte PDF/X-3-Datei kann auf diese Weise allerdings noch nicht erzeugt werden.

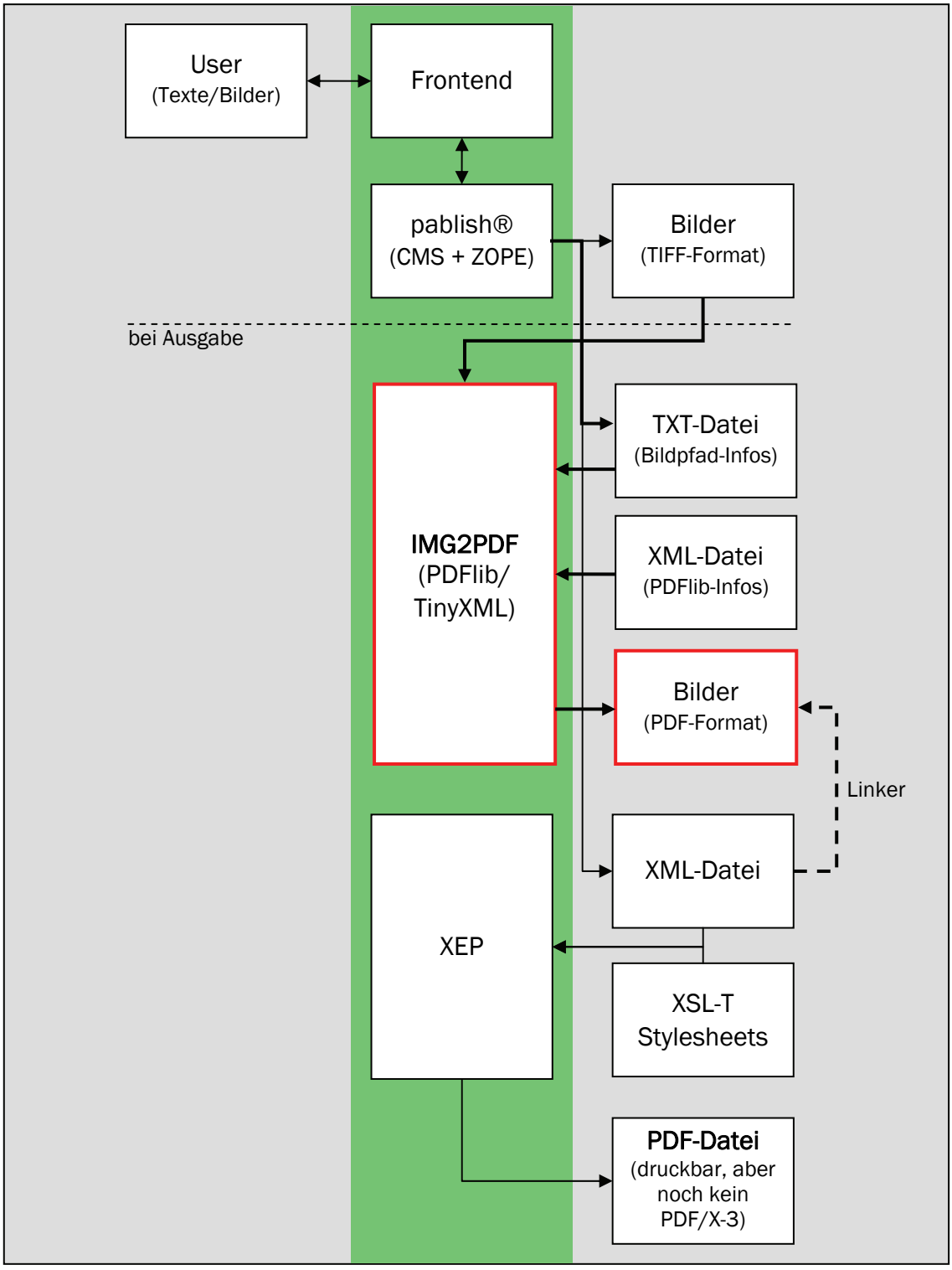

Abbildung 4: Erweiterung des Workflows mit IMG2PDF

4. Fazit

IMG2PDF steht für den Praxisbetrieb bereit. Wie sich diese Workflow-Erweiterung beim Kunden bewährt, muss noch getestet werden. Nach einer Ist-Analyse des Problems der Bildbehandlung wurde mit PDFlib eine zwar nicht kostenfreie, aber individuell anpassbare, flexible und qualitativ hochwertige Lösung gefunden. Mit den zusätzlich vorgenommenen Modifikationen am CMS und dem Versuch eine einfache Handhabung über XML zu garantieren, ist die Qualität und der Nutzen von pablish® - catalog publishing deutlich gestiegen.

Ein Nebeneffekt kann noch bei einer umfangreichen Katalogproduktion auftreten, bei der viele Bilder verarbeitet werden. Ist in jedem Bild ein ICC-Profil eingebettet, steigt die Dateigröße enorm an. Wenn mit der PDFlib vorab die Katalogbilder in PDF/X-3 umgewandelt werden, wird dabei zusätzlich das Ausgabeprofil mit eingebettet. Obwohl heutzutage Dateivolumina immer weniger problematisch sind, ist das Überarbeiten der PDF/X-3-Norm in dieser Hinsicht dringend erforderlich, da bei umfangreichen Produktionen auch jetzt schon sehr große Datenmengen anfallen. Sinnvoll wäre eine Möglichkeit, Ausgabeprofile oder häufig verwendete ICC-Profile für Bilder zentral abzulegen und darauf zugreifen zu können.

Wichtig im Zusammenhang mit dem Thema dieser Arbeit ist es auch, ein Bewusstsein für die Notwendigkeit von Color Management, unter anderem mit ICC-Profilen, den Einsatz von PDF/X-3 und die daraus resultierenden Verbesserungen zu schaffen.

5. Ausblick

Im Rahmen dieser Arbeit konnte leider bei der Katalogproduktion keine druckreife PDF/X-3-Datei erzeugt werden, aber der Weg dorthin soll im Folgenden kurz angedeutet werden.

Ein Ansatz könnte das Nachschalten von PDFlib sein. Der erzeugte Katalog wird noch einmal von PDFlib verarbeitet und die vorhandene PDF-Datei mit den fehlenden Angaben für eine valide PDF/X-3-Datei ergänzt. Dazu muss ein weiteres C++-Programm geschrieben werden, mit dem ein PDF/X-3-Dokument angelegt und die vorhandene Katalog-PDF-Datei importiert wird. Mit TinyXML werden die PDFlib-Parameter aus der vorhandenen XML-Datei ausgelesen und verarbeitet. Im Anschluss sollte man ein druckreifes PDF/X-3-Dokument vorliegen haben, das man auch bedenkenlos an den Druckpartner weitergeben kann. Der beschriebene Ablauf wird auf dem Schaubild auf der nächsten Seite bildlich dargestellt.

Weiterhin wäre zu untersuchen, ob man nicht auch mit einer zukünftigen XEP-Version valide PDF/X-3-Dokumente erzeugen kann. Die in der Dokumentation beschriebenen Möglichkeiten der aktuellen Version 4.2 sind dazu noch sehr beschränkt. Angaben über den PDF/X-Standard und einen OutputIntent lassen sich treffen, müssen aber in jedem Stylesheet angegeben werden. Dies ist Änderungen gegenüber überaus unflexibel. Dazu kommt, dass besonders darauf geachtet werden muss, dass kein Bild mit LZW-Komprimierung vorliegt und es zu einem Abbruch bei der PDF/X-3-Erzeugung kommen würde, wenn das Bild nicht vorher umgewandelt wird. Diese Aufgabe übernimmt im Gegensatz zu PDFlib XEP nämlich nicht.

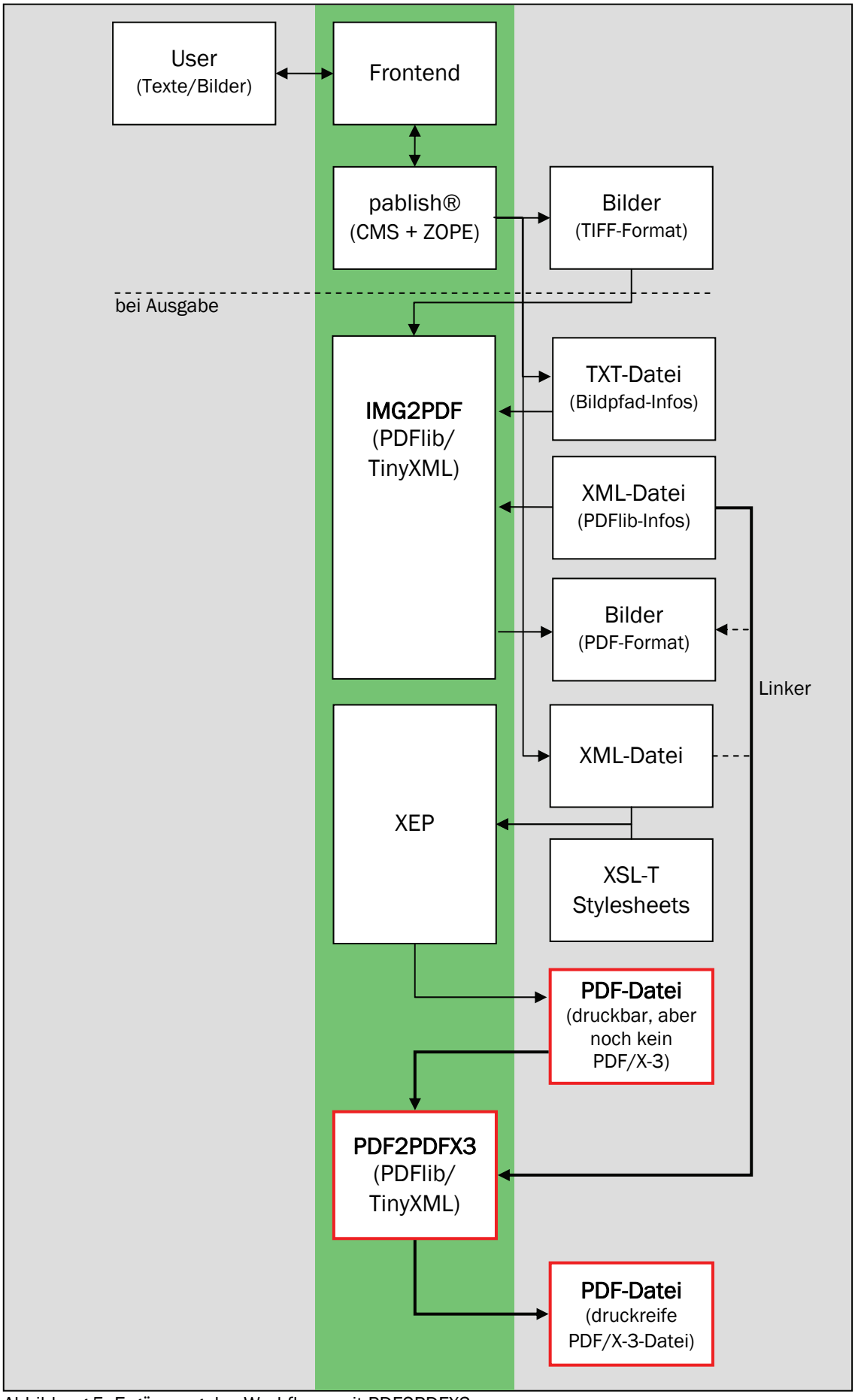

Abbildung 5: Ergänzung des Workflows mit PDF2PDFX3

6. Literaturverzeichnis

1. Drümmer, Olaf; „Das Standard-Datenformat für die Druckproduktion PDF/X-3“, Hrsg. Bundesverband Druck und Medien e.V. (bvdm), Ugra, Ifra; 2003

2. Drümmer, Olaf; Merz, Thomas; „Die PostScript- & PDF-Bibel“, dpunkt-Verlag, Heidelberg; 2. Auflage, 2002

3. Montero Pineda, Manuel; Krüger, Manfred; „XSL-FO in der Praxis – XML-Verarbeitung für PDF und Druck“, dpunkt-Verlag, Heidelberg, 1. Auflage, 2004

4. Ray, Erik T.; „Learning XML“, O’Reilly Verlag, 1. Auflage, 2001

5. Homepage der PDFlib GmbH, http://www.pdflib.com, als Download:
   „PDFlib, PDFlib+PDI, PPS; Dynamische PDF-Generierung – und mehr“,
   http://www.pdflib.com/pdffiles/PDFlib-6.0-datasheet_D.pdf
   „PDFlib-Produktfamilie, Version 6 – Die Highlights“,
   http://www.pdflib.com/pdffiles/PDFlib-6-Highlights_D.pdf

6. Homepage zu TinyXML, http://www.grinninglizard.com/tinyxml und
   http://sourceforge.net/projects/tinyxml

7. Homepage von Adobe Systems Incorporated, http://www.adobe.com,
   als Download:
   „PDF Reference, fifth Edition, Adobe® Portable Document Format, Version 1.6”, 2004, http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf

8. Homepage des World Wide Web Consortiums: http://www.w3c.org, als Download: „Extensible Markup Language (XML) 1.0 (Zweite Auflage), Deutsche Übersetzung“, 2003, http://edition-w3c.de/TR/2000/REC-xml-20001006

9. Als Ansprechpartner:
   @ computational design GmbH
   Robertstraße 5a, 42107 Wuppertal, Germany
   Tel: +49 - (0) 202 - 75 85 0 – 0, E-Mail: co-de@co-de.de
   Dipl.-Ing. Mehmet S. Yilmaz (Geschäftsführung)
   http://www.pablish.de und http://www.co-de.de

7. Abbildungsverzeichnis

Abbildung 1: Verwendung von verschiedenen Stylesheets für unterschiedliche Ausgaben 5
Abbildung 2: XML-Baumstruktur beispielhaft für eine Katalogproduktion 6
Abbildung 3: Der pablish® - catalog publishing-Workflow 9
Abbildung 4: Erweiterung des Workflows mit IMG2PDF 23
Abbildung 5: Ergänzung des Workflows mit PDF2PDFX3 26

8. Anhang

8.1 Programmierung mit C++ - IMG2PDF

img2pdf.cpp

```
#include <iostream>
#include <fstream>
#include "pdflib.hpp"
#include "tinyxml.h"
#include <vector>

using namespace std;

int main(int argc, char* argv[])
{
   TiXmlElement* psRoot;
   TiXmlElement* psPdflib;
   TiXmlElement* psInfo;
   TiXmlElement* psBilder;
   TiXmlElement* psFile;
   const char* pcId;
   const char* pcKey;
   const char* pcValue;
   const char* filepath;
   string    sCreator, sAuthor, sTitle, sPdfx, sIccprofile,
             sHonoriccprofile, sIccwarning, sTrapped,
             sOutputintent, sOutputintentusage, imgpfad;

   TiXmlDocument doc(argv[1]);

   if(argc<2)
   {
    cerr << "Please specify a xml-file" << endl;
    return 1;
   }

   if(!doc.LoadFile())
   {
    cerr << "Error while loading file: " << doc.ErrorDesc() << endl;
    return 2;
   }

   psRoot   = doc.RootElement();
   psPdflib = psRoot->FirstChildElement("pdflib");
   psInfo   = psPdflib->FirstChildElement("pdf-informationen");

   for(
        TiXmlElement* psElement = psInfo->FirstChildElement("info");
        psElement;psElement = psElement->NextSiblingElement("info"))
        {
```

```
        pcId    = psElement->Attribute("ID");
        pcKey   = psElement->Attribute("key");
        pcValue = psElement->Attribute("value");
        if      ( strcmp ( pcKey, "Creator" ) == 0 )
                  sCreator=strdup(pcValue);
        else if ( strcmp ( pcKey, "Author"  ) == 0 )
                  sAuthor=strdup(pcValue);
        else if ( strcmp ( pcKey, "Title"   ) == 0 )
                  sTitle=strdup(pcValue);
        else if ( strcmp ( pcKey, "Trapped" ) == 0 )
                  sTrapped=strdup(pcValue);
      }
  psInfo = psInfo->NextSiblingElement("pdflib-parameter");
  for(
      TiXmlElement* psElement = psInfo->FirstChildElement("info");
      psElement;psElement = psElement->NextSiblingElement("info"))
      {
        pcId    = psElement->Attribute("ID");
        pcKey   = psElement->Attribute("key");
        pcValue = psElement->Attribute("value");
        if      ( strcmp ( pcKey, "pdfx"            ) == 0 )
                  sPdfx=strdup(pcValue);
        else if ( strcmp ( pcKey, "iccprofile"      ) == 0 )
                  sIccprofile=strdup(pcValue);
        else if ( strcmp ( pcKey, "honoriccprofile" ) == 0 )
                  sHonoriccprofile=strdup(pcValue);
        else if ( strcmp ( pcKey, "iccwarning"      ) == 0 )
                  sIccwarning=strdup(pcValue);
        else if ( strcmp ( pcId,  "outputintent"    ) == 0 )
                  sOutputintent=strdup(pcKey);
                  sOutputintentusage=strdup(pcValue);
      }
  psBilder   = psPdflib->NextSiblingElement("bilder");
  psFile     = psBilder->FirstChildElement("file");
  filepath   = psFile->Attribute("src");
  ifstream infile(filepath, ios::out|ios::binary|ios::in);
  while (! infile.eof() )
  {
      infile >> imgpfad;
      try
      {
        cout << imgpfad << " --> ";
        string fname=imgpfad;
        fname.erase(fname.rfind(".tif"));
        fname=fname + ".pdf";
        cout << fname << endl;
```

```
            PDFlib *p;
            int image;
            p = new PDFlib();
            p->set_parameter("pdfx", sPdfx);
            p->set_parameter("compatibility", sCompatibility);
            p->set_parameter("honoriccprofile", sHonoriccprofile);
            p->set_parameter("iccwarning", sIccwarning);

            if (p->begin_document(fname, "") == -1)
            {
               cerr << "Error: " << p->get_errmsg() << endl;
               return 3;
            }

            p->set_info("Creator", sCreator);
            p->set_info("Author", sAuthor);
            p->set_info("Title", sTitle);
            p->set_info("Trapped", sTrapped);
            p->load_iccprofile(sOutputintent, sOutputintentusage);
            image = p->load_image("auto", imgpfad, "ignoremask");

            if (image == -1)
            {
               cerr << "Error: " << p->get_errmsg() << endl;
               return 4;
            }

            p->begin_page_ext(a4_width, a4_height, "");
            p->fit_image(image, 0.0, 0.0, "adjustpage");
            p->close_image(image);
            p->end_page_ext("");
            p->end_document("");
         }

         catch (PDFlib::Exception &ex)
         {
              cerr << "PDFlib exception occurred: " << endl;
              cerr << "[" << ex.get_errnum() << "] "
                   << ex.get_apiname()
                   << ": " << ex.get_errmsg() << endl;
              return 5;
         }
    }
    infile.close();
    return 0;
}
```

8.2 Extensible Markup Language (XML) - Medienneutraler Datenlieferant

img2pdf.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE job SYSTEM "img2pdf-info.dtd">

<job>

    <pdflib>

        <pdf-informationen>
            <info ID="Creator" key="Creator" value="img2pdf.cpp"/>
            <info ID="Author" key="Author"
             value=" Alexander Jacob fuer
             '@ computational design GmbH'"/>
            <info ID="Title" key="Title"
             value="Image-to-PDF-Konverter: IMG2PDF"/>
            <info ID="Trapping" key="Trapped" value="False"/>
        </pdf-informationen>
        <pdflib-parameter>
            <info ID="pdfx" key="pdfx" value="PDF/X-3:2002"/>
            <info ID="iccprofile" key="honoriccprofile"
             value="true"/>
            <info ID="iccwarning" key="iccwarning" value="true"/>
            <info ID="outputintent" key="ISOcoated.icc"
             value="usage=outputintent"/>
        </pdflib-parameter>
    </pdflib>

    <bilder res="300">
        <file ID="bildpfade" src="imgpfad.txt"/>
    </bilder>
</job>
```

8.3 Document Type Definition (DTD)

img2pdf.dtd

```
<!-- <!DOCTYPE img2pdf
[ -->
<!ELEMENT job (pdflib, bilder)+>
     <!ELEMENT pdflib (pdf-informationen, pdflib-parameter)+>

          <!ELEMENT pdf-informationen (info+)>
               <!ELEMENT info EMPTY>
                    <!ATTLIST info
                         ID CDATA #REQUIRED
                         key CDATA #REQUIRED
                         value CDATA #REQUIRED>

          <!ELEMENT pdflib-parameter (info+)>

     <!ELEMENT bilder (file?, bild+)>
          <!ATTLIST bilder res CDATA #REQUIRED>

               <!ELEMENT file EMPTY>
                    <!ATTLIST file
                         ID CDATA #REQUIRED
                         src CDATA #REQUIRED>

               <!ELEMENT bild EMPTY>
                    <!ATTLIST bild
                         ID CDATA #REQUIRED
                         name CDATA #REQUIRED
                         width CDATA #REQUIRED
                         height CDATA #REQUIRED
                         format CDATA #REQUIRED
                         type (pic | logo) "pic">

<!-- ] -->
```