Modellierung und Generierung eines Produkthandbuches in XML zur Wiederverwendung für unterschiedliche Ausgabemedien – Praxisbericht

Marcus Popp
Fakultät für Informatik und Automatisierung
TU Ilmenau

Dipl.-Ing. Gerd Körber
Innovative Medien & Systeme GmbH
Suhl

Abstract. Der vorliegende Praxisbericht beschreibt exemplarisch die Vorgehensweise bei der Implementierung einer vorhandenen Datenstruktur in XML am Beispiel eines Produkthandbuches. Besonderes Augenmerk wurde dabei auf die Medienneutralität der Rohdaten, den Export der Daten im XML–Format und die Möglichkeiten der Anreicherung der Rohdaten mit Darstellungsinformationen gelegt. Der Bericht fasst die gesammelten Erfahrungen zusammen und beleuchtet die eingesetzten Technologien.

1 Motivation

Die Arbeit entstand im Rahmen einer Studienarbeit an der TU Ilmenau in Zusammenarbeit mit der Firma Innovative Medien & Systeme GmbH in Suhl. Zielstellung war der Entwurf einer XML–basierten Datenstruktur für ein umfangreiches INSTABUS–Produkthandbuch der Firma Merten. Die entworfene Datenstruktur soll als medienneutraler Ausgangspunkt für eine möglichst vielgestaltige Wiederverwendung der Produktdaten, u.a. zur Ausgabe im Internet oder zum Druck des Handbuches dienen. Im weiteren wurde untersucht, inwieweit sich die entworfene Datenstruktur zusammen mit Visualisierungsinformationen in einem relationalen Datenmodell abbilden lässt. Die notwendigen Verwaltungstools waren prototypisch zu implementieren, die Vor- und Nachteile einer solchen Verfahrensweise herauszuarbeiten und Vorschläge für eine optimale und kostengünstige Lösung des Problems zu unterbreiten.

2 Vorgehensweise

2.1 Dokumentenanalyse

Vor dem Entwurf der Datenstruktur wurde eine eingehende Dokumentenanalyse anhand eines vorliegenden Produkthandbuches in gedruckter Form durchgeführt, was zu folgenden Ergebnissen führte:

1. Aufgrund der bisherigen Verwendung von Textverarbeitungsprogrammen zur Erstellung des Handbuches sind viele Inkonsistenzen in der Dokumentenstruktur zu finden. Vor einer endgültigen Projektumsetzung ist deshalb in Rücksprache mit dem Auftraggeber zu klären, ob diese gewünscht sind oder nicht.
2. Die Produkte und deren Eigenschaften sind so heterogen und zugleich hochkomplex, dass eine vollständige Abbildung des Handbuchinhaltes in eine Datenstruktur im Rahmen der geplanten Projektzeit nicht möglich war, weil der verfügbare Zeitrahmen gesprengt werden würde.
3. Zur Vereinfachung sollen zunächst in der Datenstruktur nur Attribute Verwendung finden, die bei vielen Produktbeschreibungen vorkommen.

Aus der Dokumentenanalyse ergibt sich direkt die Konsequenz, dass man mit einer reinen HTML-Lösung an dieser Stelle nicht weiter kommt, da es sich um ein äußerst komplexes und zudem lebendiges Dokument handelt, welches in regelmäßigen Abständen inhaltlich aktualisiert wird und auch visuell dem Zeitgeist folgen soll. Es bietet sich also an einen Schritt weiter zu gehen und eine XML – Lösung zu entwickeln, die Daten, Struktur und Layout voneinander trennt.

2.1.1 Analyseergebnisse

Die Beschreibung eines Produktes besteht aus 4 großen Blöcken:

1. Allgemeine Produktdaten
2. Funktionsbeschreibung
3. Montagebeschreibung
4. Technische Daten

Zu den allgemeinen Produktdaten werden einfache einwertige Attribute gerechnet, die bei allen Produkten vorkommen. Dies sind z.B. Produktname, Produktnummer, Bestellnummer, Produktbild, Produktsymbol, usw..
Auch Funktions- und Montagebeschreibung kommen bei allen Produkten vor, haben aber eine komplexere Zusammensetzung als die allgemeinen Produktdaten. Sie bestehen aus einer endlichen, sich wiederholenden Abfolge von Text-, Bild- und Tabellenelementen mit unterschiedlichen Inhalten.
Die technischen Daten sind sehr vielgestaltig und abhängig von der jeweiligen Produktgruppe und Variante des Produktes. Ein Schaltaktor hat andere technische Attribute als ein Jalousieaktor oder eine Baugruppe zur Spannungsversorgung des Bussystems.

2.2 Vorhandene Softwarelösungen am Markt

Da ein Aspekt des Projektes die Suche nach einer möglichst günstigen Lösung für das Problem war, mußte nach der Analyse der Dokumentenstruktur eine Marktanalyse folgen. Das Ergebnis war ernüchternd. Im großen und ganzen gab es 2 Kategorien von Softwarelösungen für die Problemstellung:

1. kleine, proprietäre Tools die zu Preisen von unter 3000,- € angeboten werden, die aber mit der abzubildenden Datenstruktur völlig überfordert waren
2. komplexe, verteilte Softwaresysteme die kaum einen Wunsch offen gelassen haben, für die aber mindestens 35.000,- € zu bezahlen waren

Als Ergebnis läßt sich somit festhalten, dass für klein- und mittelständische Unternehmen keine adäquaten Lösungen existieren, die nicht das begrenzte Budget einer solchen Firma sprengen. Die Entwicklung eines Prototypen mit Perspektive auf ein komplettes Produkt zum Verwalten von Produktdaten mit Ausgabemöglichkeiten für verschiedene Nutzungskontexte läßt sich also rechtfertigen.

2.3 Entwurf der XML–Datenstruktur

DTD vs. XML–Schema

Wenn man mit dem Entwurf einer XML–Datenstruktur beginnt, stehen mit DTDs und XML–Schemata zwei Möglichkeiten zur Verfügung, um die logische Struktur einer Klasse von XML–Dokumenten zu definieren.
Prinzipiell erlaubt die XML–Spezifikation zwar auch Instanzen ohne DTD oder Schema zu erstellen. Das ist aber nur sinnvoll, wenn eine Dokumenteninstanz nur einmal erstellt wird und deren Struktur für weitergehende Aufgaben, wie z.B. den Datenaustausch, nicht von Belang ist. Sobald man aber, wie z.B. bei den Produktspezifikationen des INSTABUS–Handbuches, zahlreiche XML–Dokumente des selben Formates erstellen möchte, ist eine Definition einer DTD oder eines Schemas sehr zu empfehlen. Strukturelle Fehler im Dokument werden dann durch Validierung des XML–Dokumentes gegen DTD oder Schema sichtbar. Außerdem kann man sie als Vorlage anderen Mitarbeitern zur Verfügung stellen, die dann auf dieser Basis eigene XML–Dokumente erstellen können.

Doch worin besteht der Unterschied zwischen einer DTD und einem XML – Schema?

Im Gegensatz zu DTDs wird in XML–Schemata die selbe Syntax wie in XML verwandt, was die Konsistenz des gesamten Projektes erhöht und den Einarbeitungsaufwand verringert. Außerdem verbessern sich durch den Einsatz von Schemata die Möglichkeiten zur Deklaration und Definition von Datentypen massiv. Während man in DTDs keine Datentypen kennt, kann man in Schemata aus 44 vordefinierten wählen und zusätzlich noch selbst neue Datentypen definieren. Aufgrund der Anlehnung an Konzepte der Objektorientierung besteht die

Möglichkeit, einmal definierte Datentypen wiederzuverwenden, zu erweitern oder einzuschränken.
Unter Beachtung dieser Fakten ließ sich die Frage, ob für die Entwicklung der Datenstruktur des Produkthandbuches eine DTD oder ein XML–Schema verwendet werden soll, letztendlich leicht beantworten. Die Entscheidung fiel zu Gunsten des XML–Schemas aus.
Auf Basis der Dokumentenanalyse wurde ein Schema erstellt, welches – wie bereits erwähnt – zunächst nur Attribute verwendete, die bei vielen Produkten vorhanden waren. Wenn eine Erweiterung des Schemas durchgeführt werden soll ist das problemlos, z.B. durch einbinden weiterer externer Schemata über `include`–Tags, möglich.

2.4 Datenspeicherung relational oder im XML–Format?

Unter Kostengesichtspunkten hat XML einen großen Vorteil: Es ist weder durch Patente, Copyrights oder sonstige Restriktionen eingeschränkt. Demzufolge entstehen dem Benutzer keine zusätzlichen Kosten durch die Verwendung von XML. Ein nicht zu vernachlässigender Punkt ist allerdings, dass im Vergleich zu einem relationalen System höhere Entwicklungskosten anfallen, wenn man die Daten selbst auf Basis des Dateisystems mit XML verwalten oder eine XML–Datenbank verwenden will. Diese Tatsache resultiert zum Einen aus fehlendem Wissen mit solchen Systemen und der damit einhergehenden kostenintensiven Weiterbildung und Einarbeitung, und zum Anderen aus dem höheren Entwicklungsaufwand für ein reines XML–System. Außerdem gibt es bereits einige Arbeiten, die auf die schlechte Performanz reiner XML–Datenbanksysteme aufmerksam gemacht haben. Exemplarisch sei hier [SSS01] genannt.
Nicht zu vernachlässigen ist auch die im Vergleich zu rein relationalen Systemen datenintensivere Speicherung von XML–Daten aufgrund des schlechten Verhältnisses zwischen Nutzdaten und Metadaten. In besonderen Fällen, wie z.B. bei der Messdatenerfassung oder der Speicherung von technischen Daten, kann es 1:9 und schlechter sein. Im Ergebnis können dann relativ schnell große Datenmengen entstehen. Zur Verdeutlichung folgendes kleines Beispiel mit 4 Byte Nutzdaten und 37 Byte Metadaten:

```
<Eingangsspannung>24 V</Eingangsspannung>
```

Man könnte das Volumen der Metadaten im obigen Beispiel durch geschicktes verwenden von Abkürzungen verringern, was allerdings die „human readability“ negativ beeinflussen würde.

	Access	XML
Fixes Volumen für Datenschema	370.000 Byte	3.700 Byte
Zusätzliches durchschnittliches Volumen je Datensatz	1.000 Byte	4000 Byte
Gesamtvolumen für 123 Datensätze	**493.000 Byte**	**495.700 Byte**
Pro – forma Gesamtvolumen für 2000 Datensätze	2.370.000 Byte	8.003.700 Byte

Tabelle 1: Berechnung des Datenvolumen–Breakevens zwischen XML– und MS Access–Version für die entworfenen prototypischen Datenstrukturen des INSTABUS–Handbuches

Die Tabelle zeigt, dass die Speicherung der Daten in einer Access–Datenbank ab etwa 123 Datensätzen aus Volumengesichtspunkten günstiger wäre, und das bei einer größeren Anzahl von Datensätzen das Gesamtvolumen für die XML–Version wesentlich schneller steigt. Jedoch muß angemerkt werden, dass wir uns in beiden Fällen noch nicht in kritischen Größenordnungen bewegen. Eine Tendenz ist dennoch erkennbar.
Da man aber trotz dieser Nachteile nicht auf die höhere Zukunftssicherheit und die wesentlich größere Flexibilität in der Interaktion mit anderen XML – Anwendungen verzichten kann, muß man nach einer Alternative suchen.
Die Verwendung der relationalen Technologie in Verbindung mit einer selbst erstellten XML–Exportschnittstelle scheint an diesem Punkt mehr als logisch. Auf diese Weise kann man kostengünstig und gleichzeitig flexibel agieren.
Das relationale Datenbankschema wurde auf Basis des bereits erstellten XML–Schemas entwickelt und ist deshalb sehr stark an das XML–Schema angelehnt. Als Konsequenz aus dieser Vorgehensweise erhält man eine Datenbank, die das Mapping nach XML optimal unterstützt.

2.5 Entwurf eines Layouts für das INSTABUS–Handbuch

Aufgrund der vielfältigeren Möglichkeiten von XSLT im Vergleich zu XSL oder CSS, wurde diese Variante den anderen beiden zur Gestaltung eines Layouts vorgezogen. Von den angebotenen Möglichkeiten konnten aber im Prototypen zunächst nur wenige genutzt werden. So wurden zum Beispiel Schriftart, -größe und -farbe des Produktnamens geändert und einige Schrifteffekte, wie Kursiv- und Fettdruck, verwendet. Des weiteren wurde ein Großteil der Daten in einer unsichtbaren Tabelle angeordnet. Wie das Layout im Browser angezeigt wird, kann man in Abb. 1 sehen.

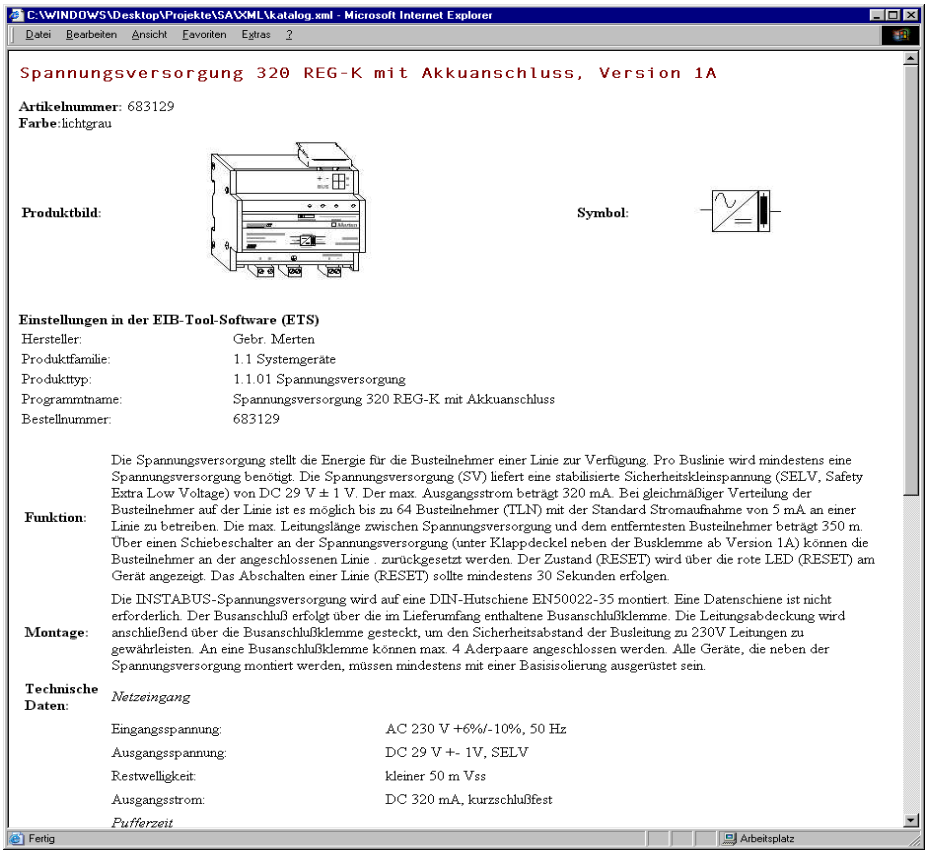


Abb. 1. XML-Dokument formatiert mit XSLT

2.6 Entwurf eines Prototyps zur Verwaltung der Daten

Zuletzt wurde der Prototyp eines kleinen Content–Management–Systems programmiert. Dieses greift über ODBC auf die erstellte Datenbank zu. Über die Oberfläche können alle Daten gepflegt und jedem Produkt ein individuelles Layout zugewiesen werden. Bei der Konvertierung der Datenbank nach XML wird dann das jeweilige Layout in das XML–File eingebunden.

2.7 Verwendete Tools

Für sämtliche Entwicklungen rund um XML wurde XMLSpy 4.0 der Firma Altova verwendet. Der Grund war anfangs folgender: XMLSpy war das erste Tool, welches die im März 2001 verabschiedete Spezifikation des XML–Schemas vollständing unterstützte.

Während der Arbeit am Projekt konnte XMLSpy aber auch in Sachen Bedienkomfort und Stabilität überzeugen. Einziger Knackpunkt ist der XSLT–Designer, der recht häufig abstürzte. Wenn dieses Manko in der Zwischenzeit behoben ist, kann man XMLSpy uneingeschränkt für die Entwicklung von XML–Dokumenten empfehlen.
Als Datenbank war Microsoft Access 2000 und als Programmierumgebung Borland Delphi 5.0 vorgegeben.

3 Zusammenfassung und Ausblick

Das entwickelte Datenmodell für den Prototypen ist wenig flexibel, da es exakt auf das vorliegenden INSTABUS–Handbuch zugeschnitten ist. Eine Flexibilisierung ist durch die Verwendung von generischen Attribut–Tabellen denkbar. Dies würde dazu führen, dass die verschiedenen Tabellen zur Speicherung der einzelnen Produkt– oder Artikelattribute zusammengelegt werden. Der Vorteil: Man muss nicht bei jeder Veränderung in der Struktur des Handbuches das Datenbankschema anpassen. Man eröffnet sich mit einer solchen Lösung die Möglichkeit individualisierte Dokumente nach Inhalt und Layout, genau auf die Bedürfnisse des Kunden zugeschnitten, zu erstellen und verwalten.

4 Literatur

[H&S1995] Andreas Heuer, Gunter Saake: *Datenbanken – Konzepte und Sprachen*, Thomson Publishing, 1995

[Hart2001] Falk Hartung: *Portierbare Daten und Code: XML und Java*, Studienjahresarbeit 2001, TU Ilmenau

[K&K01] Manfred Knobloch, Matthias Kopp: *Web-Design mit XML,* Dpunkt Verlag, 2001

[MerEIB] Merten GmbH & Co. KG: *INSTABUS Produkthandbuch* http://www.merten.de/deutsch/download/eib_handbuch.html

[N&H2000] Simon North, Paul Hermans: *XML in 21 Tagen (Übersetzung der Originalausgabe: Teach Yourself XML in 21 Days),* Markt & Technik Verlag, 2000

[Popp01] Marcus Popp: *Entwurf einer XML – basierten Datenstruktur für ein Produkthandbuch,* Studienjahresarbeit 2001, TU Ilmenau

[SSS01] Rita Schindler, Günther Specht, Andre Schmatloch: *Performanz von XML – Datenbanksystemen;* Beitrag zum 7. Workshop MIK 2001, TU Ilmenau 2001

[XMLSpec] The World Wide Web Consortium: *Extensible Markup Language (XML) 1.0 (Second Edition), 06.10.200* http://www.w3.org/TR/2000/REC-xml-20001006