Praxis

4 Beleg Reengineering

Harry M. Sneed
Institut für Wirtschaftsinformatik an der Universität Regensburg
Harry.Sneed@t-online.de

Zusammenfassung

Dieser Beitrag zum Thema Software Reengineering befaßt sich mit dem Reengineering von alten Druckdateien. Das Ziel ist es, die Inhalte der bestehenden Druckdateien in XML Dokumente zwecks der weiteren Verarbeitung sowie zum Abgleich mit den neuen Druckausgaben zu vergleichen. Dabei werden die XML Schema Sprache und die XSLT Transformationssprache herangezogen. Das Tool zur Transformation der alten Belege in XML wurde im Rahmen eines Migrationsprojektes bei der Österreichischen Wirtschaftskammer entwickelt und eingesetzt, um die neuen Grundumlagen Aussendungen gegen die der alten Anwendung auf dem Host zu validieren.

Keywords: Reengineering, Belege, XML, XSD, XSLT

4.1 Hintergrund

Legacy-Software-Systeme produzieren bekanntlich viele Berichte bzw. Listen, die an einzelne Fachdienststellen versendet werden. Gerade im Öffentlichen Dienst sind Papierausgaben bzw. Belege sehr weit verbreitet. Viele dieser Belege, z.B. Steuerbescheide und Zahlungsaufforderungen, gelangen direkt an die Kunden. Es ist also keineswegs einfach, diese Art Benachrichtigung abzulösen.

Andererseits stehen Betriebe und Behörden unter Druck, ihre Informationstechnologie zu modernisieren. Auch die Papierbelege, die sie an ihre Anwender versenden, sollten durch andere Medien, z.B. E-Mails, ersetzt werden. Auch wenn die Belege weiterhin ausgedruckt werden, empfiehlt es sich, dies an einem lokalen Ort zu tun. D.h., die Beleginhalte werden elektronisch verschickt und erst an einer dezentralen Stelle ausgedruckt. Dies hat den zusätzlichen Vorteil, daß das Belegformat lokalisiert werden kann. Es bekommt nicht jeder den gleichen Beleg, sondern einen für ihn angepaßten Beleg.

Um zu diesem höheren Grad an Flexibilität zu gelangen, müssen die bisherigen Druckausgaben in eine andere Form versetzt werden. Dies kann auf zweierlei Weise geschehen. Entweder werden die alten Programme umgeschrieben, um das neue Ausgabeformat zu produzieren, oder die alten Ausgaben werden abgefangen und in das neue Format umgesetzt. Im ersten Fall handelt es sich um Program-Reengineering, im zweiten Fall um Data-Reengineering.[1]

Es besteht demzufolge ein dringender Bedarf, alte Druckbelege in neue Formate zu versetzen und zwar möglichst vollautomatisch. [2]

4.2 Problematik

Die Schwierigkeit beim Reengineering von Computerausdrucken liegt in deren Unregelmäßigkeit. Im Gegensatz zu Programmen, die eine fest definierte Grammatik haben, haben Belege keine Standardgrammatik. Sie können jede beliebige Form annehmen. Physikalisch gesehen gibt es Seiten, Zeilen und Spalten. Logisch gesehen gibt es Literale, Variable und Füllfelder.

Eine physikalische Seite besteht aus n Zeilen mal m Spalten. Die Anzahl Spalten bzw. die Seitenbreite ist in der Regel fix. Alte Mainframe-Berichte hatten oft eine Zeilenlänge von 133 Zeichen, weil dies der Druckerzeilenlänge von 132 plus einem Drucksteuerungszeichen entsprach. Neuere Berichte haben Zeilen mit einer Länge bis zu 160 Zeichen. Die Länge einer Seite hängt ebenfalls vom Zieldrucker ab. Je nach Druckerart könnte sie von 60 bis 100 Zeilen lang sein. Endlose Listen haben keine Seiteneinteilung, sondern nur eine endlose Anzahl Zeilen. Die Zeilen haben aber dann ein festes Format.

Wenn es darauf ankommt, Belege zu beschreiben, muß als erstes das physikalische Format festgelegt werden, d.h. die Zeilenlänge und die Anzahl Zeilen pro Blatt. Damit wird der Rahmen für die weitere Verarbeitung des Beleges gesetzt.

Ein Beleg, bzw. seine Zeilen, enthält außer Leerraum (Spaces) Titelfelder, Datenfelder und Füllfelder. Füllfelder mit irgendwelchen Sonderzeichen, die das Listenlayout ansehnlicher gestalten sollen, tragen zum Reverse Engineering eines Beleges nicht bei, es sei denn, um davor- oder danach stehende Datenwerte zu lokalisieren. Titel sind wichtig, um die Datenwerte zu identifizieren. Sie können als Tags verwendet werden. In Listen mit festen Datenspalten stehen die Spalten Titel in einer oder mehreren übergeordneten Zeilen, die leicht erkennbar sind. Sie können dort entnommen werden, um die Spalten zu identifizieren. In Listen mit festem Format stehen die Titel an bestimmten Stellen auf jedem Blatt. Sie können anhand ihrer Position identifiziert werden.

In frei formatierten Belegen können die Titel an einer beliebigen Stelle stehen. Weder die Zeile noch die Spalte ist bestimmt. In diesem Falle können die Titel nur anhand

ihrer Bezeichnung erkannt werden. Dies setzt allerdings voraus, daß die Bezeichnung ein einheitliches Muster hat. Da das fast immer der Fall ist, gibt es kaum Belege, die sich nicht interpretieren lassen.

Was für die Titel zutrifft, trifft ebenso für die Datenwerte zu. In Listen mit festen Spalten werden sie immer eine feste Position haben, z.B. die Spalte 21. In fest formatierten Belegen werden sie zwar immer an einer bestimmten Spalte beginnen, aber die Zeile ist variabel. D.h., es wird nötig sein, die Zeile anhand gewisser Titel oder Füllfelder zu erkennen, was auch ohne weiteres möglich ist.

In frei formatierten Belegen ist die Identifizierung der Datenwerte nur relativ zum Titel und den Füllfeldern möglich. D.h., es werden zuerst bestimmte Titeltexte und/oder Füllfelder identifiziert. Dann läßt sich, ausgehend von dieser Position, entweder davor, danach oder darunter der Datenwert erkennen. Diese relative Erkennung ist zwar schwierig zu beschreiben, dennoch durchaus möglich.

Schließlich gibt es Werte, die in Vektoren vorkommen. Hier wird eine unbestimmte Anzahl Werte des gleichen Datentyps hintereinander aufgelistet. Das Problem hier ist, zu erkennen, welcher der letzte Datenwert in der Reihe ist. Dies kann nur über eine Typ Überprüfung geschehen. Das soll verhindern, daß der nächste Datenwert, der eine völlig andere Bedeutung hat, als letzter Datenwert des Vektors angenommen wird. Hierfür braucht man ein Unterscheidungskriterium.

4.3 Lösung

4.3.1 XML-Schema als Beschreibungsformat

Entschieden wurde, die XML Schemasprache zu verwenden, um die Belege zu beschreiben. Die Schemasprache hat schon eingebaute Attribute, die auch zu den Belegfeldern passen z.B. der Feldtyp “String” , das Attribut Name für den Feldbezeichner, und das “occurs” Attribut für wiederholte Felder und Feldgruppen. Es fehlen allerdings Attribute, um die genauen Koordinaten eines Belegfeldes zu spezifizieren. Deshalb wurde die Schemasprache hier um zwei zusätzliche Attribute ergänzt

Line und
Col.

Line identifiziert die Zeile, in der ein Feld vorkommt, und Col identifiziert die Spalte, in der das Druckfeld beginnt. Falls das Feld in einer beliebigen Zeile vorkommt, dann heißt das row- Attribut “any”. [3]

Mit der XML-Schemasprache, ergänzt durch die zusätzlichen Attribute, ist es möglich fast alle bestehenden Belege zu spezifizieren und somit auch alle Felder in den Belegen zu identifizieren. Die XML-Schemabeschreibung ist die Grundlage für die Transformation der Belege in XMLDokumente.

4.3.2 XSLT Transformationssprache zur Identifikation der Felder

Für die Identifikation der Druckfelder wird die Style Sheet Sprache XSLT herangezogen. [4] Normalerweise ist XSLT gedacht, XML Dokumente oder XSL Style Sheets in andere Zieldokumente umzusetzen, z,.B. in HTML, CSS oder XHTML. Das heißt, XSLT ist konstruiert um XML, bzw XSL zu interpretieren. Hier wurde die Sprache ergänzt um einfache Text Eingaben zu verarbeiten. Das wesentliche dabei ist die Einführung einer String Verarbeitung. Strings im Text werden an Hand ihrer Anfangsposition “Col” und ihrer Länge erkannt. Die Test Anweisung

```
Test = '(col[30:7] == "Gebuehr")'
```

bedeutet, die Zeichenfolge ab Spalte 30 in der Länge 7 wird mit dem Literal “Gebuehr” verglichen. Bei Gleichheit ist der Test `true`, sonst ist er `false`.

Die Werte werden ebenfalls anhand ihrer Position und Länge herausgeschnitten. Hierzu wird die XSLT copy Anweisung verwendet:

```
<xsl:copy Zahlung = 'col[40:10]'>
```

Dadurch wird die Zeichenfolge in der jeweiligen Zeile ab Spalte 40 in der Länge in das XML Datenelement namens Zahlung heraus kopiert.

Diese String Operationen werden durch zwei weitere nützliche Attribute ergänzt:

Delimited by <Zeichen> und
Prelimited by <Zeichen>

Die delimited by Klausis beended eine String Copy an dem genannten Zeichen, z.B. hier an dem ersten Leerzeichen.

```
<xls:copy
Payment = 'col[40:8] delimited by " "'>
</xsl:copy>
```

Mit dieser Anweisung wird die Zeichenfolge ab Spalte 40 in der Länge 8 heraus kopiert bis zum ersten Leerzeichen.

Die prelimited by Klausis startet eine String Copy nach dem genannten Zeichen, z.B. hier nach dem ersten Komma.

```
<xls:copy
Payment = 'col[40:16] prelimited by ","'>
</xsl:copy>
```

Mit dieser Anweisung wird die Zeichenfolge ab Spalte 40 bis zur Länge 16 nach dem ersten Comma heraus kopiert.

In der Regel können Felder in Druckzeilen nur über eine Abfrage bestimmter Textelemente erkannt werden, d.h., die eigentlichen Werte sind nur über ihre Titel erkennbar. Zur Auswahl von Felder in einer Textdatei werden drei XSLT Anweisungen benutzt:

xsl:if,
xsl:choose und
xsl:for-each

Wenn es um eine wahr/falsch-Entscheidung geht, wird eine *if*-Abfrage verwendet. Wenn es um eine Auswahl nach alternativen Textfragmenten geht, wird eine *choose*-Anweisung benutzt. Die *for each*-Schleifenanweisung

wird dann herangezogen, wenn Felder bzw. Feldgruppen mehrfach, aber in einer unbestimmten Anzahl, auf der gleichen Zeile vorkommen, z.B. wenn ein Kunde mehrere Adressen hat.

4.3.3 Belegtransformationsprozeß

Sind die Belege einmal mit einem XML-Schema beschrieben und deren Transformation mit einer XSLT Spezifikation spezifiziert, ist der Rest voll automatisiert. Das Werkzeug "Repo2XML" kann in einem Lauf mehrere Belege umsetzen. Für jeden Beleg wird zuerst das XML-Schema zu diesem Beleg geholt und geparst. Daraus wird eine interne Datenbeschreibungstabelle für den Beleg generiert. Als zweites wird die XSLT Spezifikation gelesen und eine Umsetzungsprozedur daraus generiert. Dann wird die entsprechende Druckdatei von der generierten Prozedur gelesen und Zeile für Zeile, Feld für Feld in eine XML-Datei umgesetzt.

Die Feldgruppen bilden die XML-Objekte, die auch im Schema beschrieben sind, z.B. Mitglieder, Forderungen und Kammer. Jedes Objekt wird zunächst in einer eigenen Zwischendatei gespeichert, in der pro Instanz eine XML-Datengruppe gebildet wird. Anschließend werden alle Objekte zusammen mit dem Schema in einer einzigen XML-Datei mit dem neuen Schema am Anfang zusammengeführt und sortiert. Das Ergebnis ist eine XML-Datei mit einem Schema als Kopf und n mal m Objekten als Rumpf. Das Schema wird von einer Standardschemadatei entnommen.

4.4 Test der XML Dateien

Ein Test ist stets ein Test gegen etwas. [5] Bei Entwicklungsprojekten wird gegen die Anforderungsspezifikation getestet. Bei Migrationsprojekten wird gegen das alte System getestet. Mit der Belegtransformation durch Repo2Xml wird die Möglichkeit geschaffen, die neuen XML-Dateien mit den alten Druckdateien abzugleichen, denn beide sind jetzt im gleichen Format. Das Werkzeug TestComp liest die alte XML-Datei und speichert die Feldinhalte in einer SQL-Datenbank. Anschließend liest es die neue XML-Datei und paart die Instanzen aufgrund der spezifizierten Suchbegriffe. Entspricht eine Instanz der neuen XML-Datei einer Instanz in der alten XML-Datei, werden die Werte aller Attribute verglichen und die nicht passenden Attribute ausgewiesen. Ist eine neue Instanz in der alten XML-Datei nicht zu finden, gilt dies als Ergänzung.

4.5 Erfahrung mit Beleg Reengineering

Die Methode des Beleg-Reengineering, die in diesem Beitrag beschrieben wurde, ist in einem Migrationsprojekt für die Österreichische Wirtschaftskammer entstanden. Dort war es erforderlich, ein neues .NET-Anwendungssystem gegen das alte CICS-System auf dem Host zu testen. Dazu gehörte neben dem Abgleich der SQLServer- Datenbanktabellen mit den VSAM-Dateien auch der Abgleich der neuen XML-Ausgabedateien mit den alten Druckdateien vom Host. Dies war der Anlaß, die bisherigen Druckdateien in XML umzusetzen.

Insgesamt wurden 5 verschiedene Belegtypen in XML umgesetzt

- die Aussendungen,
- die Mahnungen,
- die Exekutionen,
- die Lohndaten und
- die Buchungsbelege.

Dies ist in allen Fällen mit dem Tool Repo2XML gelungen trotz einzelner Unterschiede zwischen den Ländern. Die generierten XML-Dokumente konnten ohne weiteres mit den neuen XMLDokumenten verglichen werden. Auf diese Weise wurden einige Fehler in den neuen Ausgaben aufgedeckt, die sonst unbemerkt geblieben wären. Die Umsetzung hat sich also gelohnt.

Literaturhinweise

[1] Aiken, P.: Data Reverse Engineering, Addison- Wesley, Reading, MA., 1998.

[2] Sneed, H.: Objektorientierte Softwaremigration, Addison-Wesley, Bonn, 1999.

[3] Fitzgerald, M.: Building B2B Applications with XML, John Wiley & Sons, New York, 2001.

[4] Box, D./Skonnard, A./Lam, J.: Essential XML, Addison-Wesley, München, 2001, S. 179.

[5] Sneed, H./Winter, M.: Test objektorientierter Software, Hanser Verlag, München, 2001.