Umsetzung des Signaturenkataloges SK25 bei der XML-basierten Erzeugung kartenähnlicher Graphiken [1]

(mit 4 Bildern)

Von Karl Neumann, Andreas Kupfer und Brigitte Mathiak, Braunschweig

ZUSAMMENFASSUNG: Mit der XML-basierten Transformationssprache XSLT können in GML kodierte Geodaten so in Dokumente der Vektorgraphiksprache SVG abgebildet werden, dass recht kartenähnliche Graphiken entstehen. Im vorliegenden Text skizzieren wir, wie dabei die durch den Signaturenkatalog SK25 gegebenen Spezifikationen systematisch berücksichtigt werden können, indem die Signaturen in SVG- und die Ableitungsregeln in XSLT-Konstrukte umgesetzt werden.

ABSTRACT: By means of the XML based transformation language XSLT, GML-encoded geo data can be transformed into documents of the vector graphics language SVG, resulting in graphics that bear close resemblance to maps. In this paper we outline, how the specifications given by the signature catalog SK25 can be systematically incorporated into that transformation process. The signatures from the catalog can be represented in form of SVG statements, while the transformation rules can be reformulated as XSLT templates.

1 Einleitung

Auch im Bereich der Geodaten und der rechnerbasierten Kartennutzung werden Techniken und spezielle Sprachen immer interessanter, die auf der Trägersprache XML aufbauen (*Herrmann/Asche* 2000, *Kraak/Brown* 2000). Eine solche Sprache ist z.B. die XML-Anwendung GML (Geography Markup Language), die inzwischen bereits in der Version 3 vom Open GIS Consortium herausgegeben wurde (*OGC* 2004) und die einen offenen Standard zum Austausch von Geodaten darstellt. Dieser Standard legt mit XML-Sprachmitteln im Wesentlichen ein Geometrieschema und ein Featureschema fest, in deren Rahmen eigene Geoanwendungswelten modelliert werden können. Mit einer weiteren XML-basierten Sprache, der Extensible Stylesheet Language for Transformation (XSLT) lassen sich alle Arten von XML-Dokumenten – und damit auch GML-Dokumente – in anders strukturierte XML-Dokumente überführen. Es liegt daher der Gedanke nahe, Geoobjekte mittels XSLT auf graphische Objekte so abzubilden, wie man es aus kartographischen Anwendungen gewohnt ist, wobei sich als Zielsprache dazu wiederum eine XML-basierte Sprache anbietet, nämlich Scalable Vector Graphics (SVG). Insgesamt ergibt sich so ein Ablauf, in dem Geodaten als GML-Dokumente kodiert werden, dann, von XSLT-Dokumenten gesteuert, so transformiert werden, dass SVG-Dokumente entstehen, die möglichst kartenähnlich aussehen. Wir haben diesen Ablauf unter Verwendung

[1] Erscheint in *Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Frankfurt M., 2005.*

von Daten des Digitalen Landschaftsmodells implementiert und in *Neumann et al.* 2004, *Neumann/Mathiak/Kupfer* 2004 beschrieben. Darauf aufbauend skizzieren wir nun im vorliegenden Text, wie man den Signaturenkatalog SK25 für die Digitale Topographische Karte 1:25000 (DTK25) bei einer solchen XML-basierten Erzeugung kartenähnlicher Graphiken geeignet berücksichtigen kann.

Dazu gehen wir im nächsten Abschnitt auf den Signaturenkatalog SK25 ein und geben dann in Abschnitt 3 einen kurzgefassten Überblick über das Verfahren, nur mit XML-basierten Sprachen DLM25-Daten sowohl zu modellieren als auch kartenähnlich zu visualisieren (eine Langfassung dieser Beschreibung ist in *Mathiak/Kupfer/Neumann* 2004 zu finden). Abschnitt 4 skizziert anschließend die Umsetzung des Signaturenkataloges SK25 in XML-Konstrukte. Bei dieser Umsetzung werden u.a. Einzelsignaturen, wie z.B. Laub- oder Nadelbaumsymbole, durch SVG-Anweisungen und Ableitungsregeln durch XSLT-Sprachkonstrukte dargestellt. Im Ausblick diskutieren wir kurz, welche weiteren Arbeitsschritte zur Generierung kartenähnlicher Graphiken ebenfalls durch die XML-basierte Technik realisiert werden könnten.

2 Signaturenkatalog SK25

Der Signaturenkatalog SK25 (vgl. *AdV* 2002) beschreibt in textueller und graphischer Form die Ableitungsinformationen der Digitalen Topographischen Karte 1:25000 (DTK25) aus dem Digitalen Geländemodell (DLM25). Etwas vereinfacht gesehen, besteht der Katalog aus den Ableitungsregeln (Teil A) und den Kartensignaturen (Teil B), wobei man diese noch in die eigentlichen Signaturen und eine Farbtabelle aufteilen kann. Beide Teile sind als Sammlung von insgesamt ca. 300 Formblättern angelegt.

ATKIS-Signaturenkatalog 1 : 25 000 (ATKIS-SK25), Teil A: Ableitungsregeln **Objektbe**

Objektart	Obj-Typ	Für Kartensignatur relevante Attribute/Attributwerte, berechnete Werte oder Referenzen	SN	DP	
				L	F
4101_Ackerland	F		4010	---	4
4101_Ackerland	F	*VEG 8000 Streuobst*	4010 *4043*	--- *10*	4 ---
4101_Ackerland	F	*VEG 9997 Attribut trifft nicht zu*	4010	---	4

Bild 1 - Ausschnitte aus den Ableitungsregeln (Teil A des SK25)

In Bild 1 sind als Beispiel für Ableitungsregeln zwei kleine Ausschnitte aus Teil A zu sehen: Für jede in der Spalte “Objektart” aufgeführte DLM-Objektart wird angegeben, wie diese Objektart signaturiert wird. Dabei kann die Signaturierung vom zugehörigen Objekttyp abhängig sein; es bedeuten: Objekttyp K komplexes, P punktförmiges, L linienförmiges, F flächenförmiges Objekt. Die im Beispiel auftretende Objektart “Ackerland” ist vom Objekttyp “flächenförmig” und wird auch als Fläche signaturiert. In der nächsten Spalte werden eventuelle Attribute oder auch Attributwerte aufgeführt, die zur weiteren Auswahl der Kartensignatur dienen. Dabei sind Attributeinträge gegliedert in Attributtyp und Attributwert; im Beispiel etwa: “VEG 8000” (Attributtyp: Vegetation, Attributwert: Streuobst). Schließlich wird in der Spalte “SN” angegeben, mit welcher Signaturnummer die Objekte konkret dargestellt werden, und in der Spalte “DP” ist eine Darstellungspriorität angegeben. Diese Zahl spezifiziert, in welcher Reihenfolge sich eventuell überlagernde Signaturen gezeichnet werden sollen: Signaturen mit hohen Werten liegen über Signaturen mit niedrigeren Werten. Die Darstellungsprioritäten werden getrennt nach linienhaften (Spalte “L”) und flächenhaften (Spalte “F”) Elementen der Signatur angegeben. Spalte “L” enthält Werte für Linien, Straßenkonturen, Symbole oder Flächenkonturen und Spalte “F” Werte für Flächen, Straßendecker oder Symbolinnenflächen. Für das erste Beispiel aus Bild 1 ergibt sich damit die Regel, dass flächenförmige Objekte der Art “Ackerland” mit der Darstellungspriorität 4 und der Flächensignatur Nr. 4010 (Fläche: ackerocker) signaturiert werden. Objekte der Art “Ackerland”, deren Attribut “Vegetation” den Wert “Streuobst” aufweist, werden zusätzlich noch mit der Darstellungspriorität 10 und der Flächenmustersignatur Nr. 4043 (Streuobstmuster: baumgrün) dargestellt.

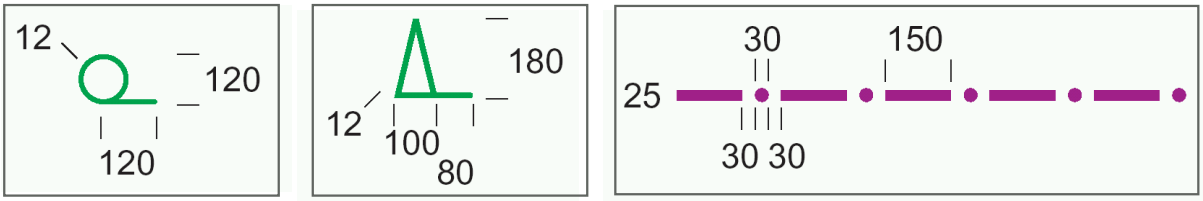

Bild 2 - Beispiele für Kartensignaturen (Ausschnitte aus Teil B des SK25)

Im Signaturenteil wird für jede Signaturnummer, die auch als Verknüpfungsmerkmal zu Teil A des SK25 dient, eine vergrößerte Signaturdarstellung angegeben. Sie enthält Maßangaben in 1/100 mm und stellt die einzelnen Signaturen in der jeweiligen charakteristischen Farbe dar. In Bild 2 sind als Beispiele drei Ausschnitte aus Teil B angegeben: Die Signaturen für Laubbäume, Nadelbäume und Gemeindegrenzen.

Die ebenfalls in Teil B angesiedelte Farbtabelle stellt den Zusammenhang zwischen den Farbnamen und den Farbanteilen der so genannten Euroskala her, die auf den Druckfarben Cyan, Magenta, Gelb und Schwarz beruht (CMYK-Farbmodell). Die SK25-Farbtabelle führt 6 Konturenfarben und 19 Flächenfarben auf. So wird die Konturenfarbe “Baumgrün” z.B. durch die Farbanteile C = 100 %, M = 0 %, Y = 100 %, K = 0 % und die Flächenfarbe “Ackerocker” durch C = 0 %, M = 0 %, Y = 10 %, K = 0 % spezifiziert. Der Signaturenkatalog enthält noch weitere Informationen, z.B. zur Gestaltung des Kartenrahmens, auf die wir hier jedoch nicht näher eingehen, da wir sie bei unserer Umsetzung nicht berücksichtigen.

3 XML-basierte Erzeugung kartenähnlicher Graphiken

Ausgangspunkt für unsere XML-basierte Erzeugung kartenähnlicher Graphiken sind in GML modellierte und dann auch kodierte Objekte des DLM25. Zur Modellierung des Digitalen Landschaftsmodells mit GML haben wir einen komplexen Typ eingeführt, genannt "AtkisMemberType", der als Exemplare "AtkisMember" enthalten kann. Wie vom GML-Rahmen vorgegeben, wurde "AtkisMemberType" als Featuretyp definiert, also vom GML-Typ "AbstractFeatureType" abgeleitet. Als Geometrie-Elemente passen hier gut die von GML vorgegebenen: "location" für Punkte, "centerLineOf" für Linien und "polygonMember" für Polygone. Wir haben zahlreiche weitere Klassen deklariert: So sind die einzelnen Objektarten des DLM25 wie "Straße" oder "Weg" in unserer Modellierung Subtypen des "AtkisMemberType". Dieser Struktur folgend können Objektexemplare angegeben werden, z.B. konkrete Straßen, Grenzen oder Verwaltungsbezirke mit ihren jeweiligen Koordinaten. Die weiter unten folgenden GML-Zeilen enthalten ein Beispiel dafür: Die Angabe einer (sehr kurzen) Straße namens "Badstraße", die auch in der Realität tatsächlich nur drei Stützpunkte hat. Auf ähnliche Weise können weitere Objekte der verschiedenen Objektarten als "AtkisMember" innerhalb eines übergeordneten "AtkisModells" aufgeführt werden, die zu einem ausgewählten Ausschnitt des DML25 gehören.

```
<AtkisMember>
  <Strasse>
    <gml:name> Badstrasse </gml:name>
    <AtkisOID> 86118065 </AtkisOID>
    <gml:centerLineOf>
      <gml:coord> <gml:X> 4437952.980 </gml:X>
                  <gml:Y> 5331812.550 </gml:Y> </gml:coord>
      <gml:coord> <gml:X> 4437960.070 </gml:X>
                  <gml:Y> 5331818.450 </gml:Y> </gml:coord>
      <gml:coord> <gml:X> 4437967.200 </gml:X>
                  <gml:Y> 5331825.410 </gml:Y> </gml:coord>
    </gml:centerLineOf>
    <Attribute>
      <Zustand> in Betrieb </Zustand>
      <AnzahlDerFahrstreifen Bedeutung="tatsaechliche Anzahl"> 2
      </AnzahlDerFahrstreifen>
      <Funktion> Strassenverkehr </Funktion>
      <VerkehrsbedeutungInneroertlich> Anliegerverkehr
      </VerkehrsbedeutungInneroertlich>
      <Widmung> Gemeindestrasse </Widmung>
    </Attribute>
  </Strasse>
</AtkisMember>
```

Um solche Sammlungen von GML-kodierten Objekten des DLM25 auf Graphikobjekte abzubilden, benutzen wir die XML-Sprache XSLT. Sie ähnelt einer funktionalen Programmiersprache und überführt XML-Dokumente – und damit auch GML-Dokumente – in XML-Dokumente anderer Struktur (*W3C* 1999, *Kay* 2001). Bei der Transformation wird nach dem Prinzip des "Pattern Matching" vorgegangen, d.h. das Quelldokument wird auf das Vorkommen von Mustern hin untersucht. Diese Muster werden in Form von XSLT-Templates vorgegeben. Trifft ein Muster zu, steht ebenfalls im jeweiligen Template, wie ein Stück des Zieldokumentes konstruiert werden soll. Das weiter unten aufgeführte

XSLT-Template sucht z.B. innerhalb eines gegebenen GML-Quelldokuments nach Knoten des Typs “AtkisModell”, dort nach Knoten des Typs “AtkisMember”, dort nach Straßen-Knoten. Wenn ein Straßen-Knoten gefunden wurde, wird überprüft, ob es in der Menge der Attribute einen Widmung-Knoten gibt und ob dieser den Wert “Gemeindestraße” oder “Sonstiges” aufweist. Falls das zutrifft, wird ein weiteres Template aufgerufen, das den Namen “DrawPath” hat. Dabei wird diesem Template der Parameter “linieNebenstrasse-NahverkehrVordergrund” mitgegeben. Letztlich generiert das aufgerufene Template einen SVG-Zeichenbefehl (genauer: einen Path-Befehl) für verschiedene Arten von Linien, wobei das Aussehen der jeweiligen Linie durch den Parameter “styleclass” gesteuert wird.

```
<xsl:template
     match="/dlm:AtkisModell/dlm:AtkisMember/dlm:Strasse">
  <xsl:if test="contains(dlm:Attribute/dlm:Widmung,‘Gemeindestrasse’)
                or contains(dlm:Attribute/dlm:Widmung,‘Sonstiges’)">
    <xsl:call-template name="DrawPath">
      <xsl:with-param name="styleclass"
                select="linieNebenstrasseNahverkehrVordergrund">
    </xsl:call-template>
  </xsl:if>
</xsl:template>
```

Andere Templates suchen nach den weiteren DLM25-Objekten im übergeodneten GML-Dokument: Es gibt Templates für Flüsse, Seen, Wälder, Wohnbauflächen usw. Und die Templates für Objekte mit punktförmiger Geometrie rufen dann ein Template auf, das alle Arten von Punktsignaturen zeichnet, während die linienförmigen Objekte durch das bereits angesprochene Template “DrawPath” behandelt werden. “DrawPath” liest im Wesentlichen die Koordinaten der linienhaften DLM-Objekte, führt eine Koordinatentransformation durch und generiert einen SVG-Path-Befehl mit der für die jeweils bearbeitete Objektklasse passenden Style-Klasse. Diese verschiedenen Style-Definitionen haben wir in Form von Cascading Style Sheets (*Meyer* 2002) vorher geeignet implementiert. Das folgende Beispiel zeigt die sehr einfache Style-Klasse “linieNebenstrasseNahverkehrVordergrund”, die das Aussehen der Linien festlegt, die Nebenstraßen des Nahverkehrs darstellen. Darunter ist ein generierter Path-Befehl aufgeführt, der diese Style-Klasse benutzt und durch Umsetzung der oben aufgeführten “Badstraße” entstanden ist.

```
.linieNebenstrasseNahverkehrVordergrund
  {fill: none; stroke-width: 8.5px;
   stroke: snow; stroke-linejoin: round}

<path class="linieNebenstrasseNahverkehrVordergrund"
      d="M 8245.97 -2142.98
         L 8253.65 -2146.15 8259.83 -2151.12"/>
```

Tatsächlich müssen zur Darstellung von Straßen meist jeweils zwei Style-Klassen implementiert werden: Eine für eine breitere Linie im Hintergrund – bei Nebenstraßen typischerweise in einem rötlichen Farbton – und eine schmalere Linie im Vordergrund. Insgesamt ergeben diese beiden übereinander gezeichneten Linien dann die typische Straßensignatur, z.B. eine weiße Linie mit dünnen rötlichen Rändern.

4 Umsetzung des Signaturenkataloges

Die Berücksichtigung des Signaturenkataloges bei der im letzten Abschnitt skizzierten XML-basierten Erzeugung kartenähnlicher Graphiken ist im Falle der Farbtabelle recht einfach: Hier müssen nur die Farbwerte aus dem CMYK-Farbmodell in das von SVG benutzte RGB-Farbmodell umgerechnet und dann mit XML-Konstrukten geeignet repräsentiert werden. Farben können in SVG durch 6-stellige Hexadezimalzahlen angegeben werden, dabei stehen jeweils 2 Stellen für Rot, Grün, Blau (*Cagle* 2002, *Eisenberg* 2002). Mit folgenden Formeln sind daher diese Farbanteile aus einem CMYK-Prozentanteiltupel zu berechnen:

$$R=(100-(C+K))\cdot 2.56 \quad G=(100-(M+K))\cdot 2.56 \quad B=(100-(C+Y))\cdot 2.56$$

Die so umgerechneten Farben lassen sich in XML gut durch so genannte Entities darstellen, da XML-Entities gerade für benannte Konstanten aller Art vorgesehen sind (*Ray* 2001, *Harold/Means* 2002, *Eckstein/Eckstein* 2004). Aus Gründen der Übersichtlichkeit bietet es sich an, als Namen dieser Entities die ursprünglichen Farbnamen aus der TK25-Farbtabelle zu übernehmen. Damit ergeben sich 25 benannte Entities, die insgesamt die Farbtabelle umsetzen und von denen zur Illustration hier einige aufgeführt werden:

```
<!ENTITY schwarz          "#000000">;   <!ENTITY grundrissbraun "#650000">;
<!ENTITY reliefbraun      "#cc6565">;   <!ENTITY bachblau       "#00ffff">;
<!ENTITY baumgruen        "#00ff00">;   <!ENTITY grenzviolett   "#9900ff">;
  ...
<!ENTITY TK25-mittelgruen "#bfff99">;   <!ENTITY TK25-hellgruen "#e5ffd8">;
```

Diese Farben können nun bei der Umsetzung der Signaturen benutzt werden, einer Teilaufgabe, die relativ einfach aber recht umfangreich ist, da alle einzelnen im SK25 aufgeführten Signaturen durch Style-Klassen und SVG-Zeichenbefehle implementiert werden müssen. So wird z.B. die Signatur SN4100 (Einzelsignatur Nadelholz) durch einen Path-Befehl und die Signatur SN4091 (Einzelsignatur Laubholz) durch eine Linie und einen Kreis dargestellt:

```
<path id="SN4100_einzelsignaturNadelholz"
    d="M 130 0 L -50 0 0 -180 50 0"
    fill="none" stroke-miterlimit="20" stroke-linejoin="miter"
    stroke-width="12" stroke="&baumgruen;" />

<g id="SN4091_einzelsignaturLaubholz">
    <line x1="0" x2="120" y1="0" y2="0"
      stroke-width="12" stroke="&baumgruen;" />
    <circle fill="none" r="60" cy="-60" cx="0"
      stroke-width="12" stroke="&baumgruen;" />
</g>
```

Bei der Wahl der Identifikatoren bietet es sich wieder an, die vom Katalog vorgegebenen Originalnummern und -namen zu verwenden (im Beispiel: "SN4100_einzelsignaturNadelholz" und "SN4091_einzelsignaturLaubholz"). Die Koordinaten für die einzelnen SVG-Zeichenbefehle können fast direkt aus den Abbildungen des Signaturenkataloges übernommen werden, wobei allerdings zu beachten ist, dass die Y-Achse in SVG eine andere

Richtung hat als im klassischen Kartesischen Koordinatensystem. In Bild 3 sind neben den schon aus Abschnitt 2 bekannten Signaturabbildungen stark vergrößerte Darstellungen der ausgeführten obigen SVG-Befehle zu sehen, wobei die Koordinatensysteme nachträglich eingefügt wurden. Vergleicht man die Bemaßungsangaben aus dem Signaturenkatalog mit den angegebenen Hilfskoordinaten einerseits und den expliziten Koordinaten in den SVG-Befehlen andererseits, sieht man die direkte Umsetzung. So wird etwa die Nadelholzeinzelsignatur durch drei Linien der Stärke 12 (in 1/100 mm) erzeugt. Die erste Linie hat eine Länge von 180 Einheiten und geht vom Punkt (130, 0) zum Punkt (-50, 0), die zweite Linie von (-50, 0) nach (0,-180) usw. Als Farbe wird “baumgrün” gewählt, und außerdem wird angegeben, dass die Verbindung zwischen zwei Linien als Gehrung (“miter”), also spitz, erfolgen soll. Entsprechend wurden die Koordinaten für Linie und Kreis der SVG-Befehle zur Darstellung der Laubholzsignatur aus den Originalbemaßungen abgeleitet.

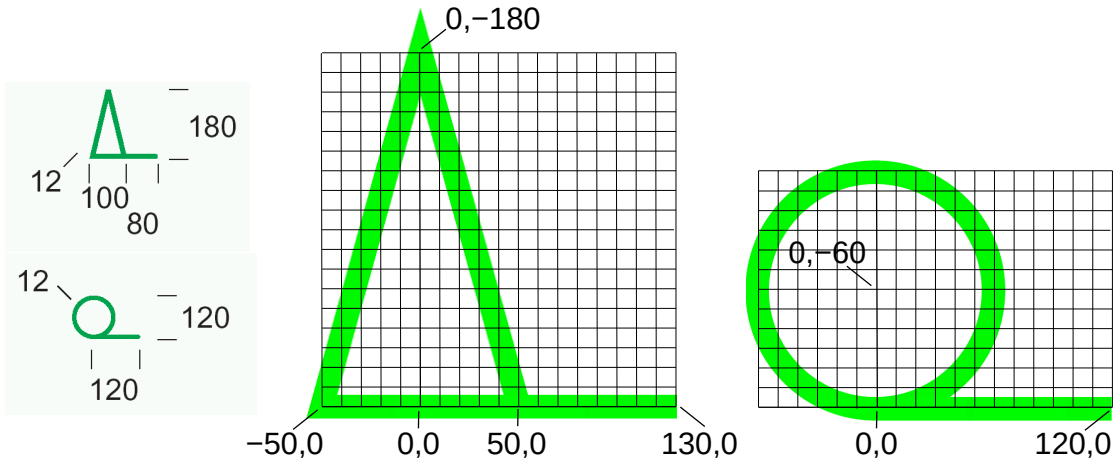

Bild 3 - Konstruktion von Einzelsignaturen

So umgesetzte Einzelsignaturen können nun an einzelnen Punktkoordinaten gesetzt werden aber auch als Elemente von Flächenmustern benutzt werden. Weiter unten wird als Beispiel das Muster (“pattern”) zur Darstellung von Mischwald konstruiert. Dazu werden jeweils zwei Einzelsignaturen für Laubholz und Nadelholz verwendet und im Muster geeignet platziert (“translate...”). Die Style-Klasse für die Signatur “SN4110” (Fläche mit Mischholz) ist dann sehr einfach, da über eine URL nur auf das entsprechende Muster verwiesen werden muss. Aber auch die Definitionen von Style-Klassen für homogen eingefärbte Flächen sind recht kurz, wie die Beispiele “SN4090” (Waldfläche) und “SN4170” (Brachland) zeigen: Hier muss nur jeweils die im Katalog spezifizierte Farbe angegeben werden. Zur Vervollständigung der Beispiele zur Umsetzung von Signaturen haben wird noch die linienhaften Signaturen für Straßen (SN3100, SN3150, SN3180) angegeben: Hier werden, wie schon im Abschnitt 3 angesprochen, immer jeweils zwei Style-Klassen für die “hintere” breite und die “vordere” schmalere Linie benötigt, wobei Farben und Breiten wieder direkt aus dem Katalog übernommen werden können.

Zum Abschluss der Diskussion über die Umsetzung der Signaturen soll noch einmal darauf hingewiesen werden, dass diese Umsetzung zwar konzeptionell recht einfach ist, dass der zu bewältigende Arbeitsaufwand aber nicht zu unterschätzen ist, da ja mehrere Hundert Signaturen umzusetzen sind.

```
.SN4090flaecheWald {fill: &waldgruen;}
.SN4170flaecheBrachland {fill: &brachbraun;}

<pattern id="Mischwald"
  patternUnits="userSpaceOnUse" x="0" y="0" width="300" height="375">
  <use xlink:href="#SN4091_einzelsignaturLaubholz"
                                          transform="translate(7,17)" />
  <use xlink:href="#SN4100_einzelsiganturNadelholz"
                                         transform="translate(82,204)" />
  <use xlink:href="#SN4100_einzelsiganturNadelholz"
                                         transform="translate(157,17)" />
  <use xlink:href="#SN4091_einzelsignaturLaubholz"
                                        transform="translate(232,204)" />
</pattern>

.SN4110flaechenmusterMischholz {fill: url(#Mischwald)}

.SN3100linieNebenstrasseBreitHintergrund
             {fill: none; stroke-width: 140px; stroke: &grundrissbraun;}
.SN3100linieNebenstrasseBreitVordergrund
                        {fill: none; stroke-width: 100px; stroke: &weiss;}
.SN3150linieNebenstrasseMittelHintergrund
             {fill: none; stroke-width: 120px; stroke: &grundrissbraun;}
.SN3150linieNebenstrasseMittelVordergrund
                         {fill: none; stroke-width: 80px; stroke: &weiss;}
.SN3180linieNebenstrasseSchmalHintergrund
              {fill: none; stroke-width: 90px; stroke: &grundrissbraun;}
.SN3180linieNebenstrasseSchmalVordergrund
                         {fill: none; stroke-width: 60px; stroke: &weiss;}
```

Die Umsetzung der Ableitungsregeln (Teil A des Signaturenkataloges) ist anspruchsvoller: Hier werden alle zu signaturierenden Objektarten betrachtet, und für jede einzelne Objektart wird für alle ihre Darstellungsprioritäten im Prinzip ein XSLT-Template implementiert. Ein typisches Beispiel für Objekte mit mehreren Darstellungsprioritäten sind die Straßen mit zwei Prioritäten: eine (niedrigere) für die Straßenkontur und eine (höhere) für den Straßendecker. Jedes Template muss nun im GML-Quelldokument nach der jeweils zu behandelnden Objektart suchen und auch die spezifische Darstellungspriorität berücksichtigen. Die Suche nach der Objektart wird (wie in Abschnitt 3) mit Hilfe des XSLT-Befehls "match" realisiert, und die Darstellungsprioritäten steuern wir mit dem XSLT-Attribut "mode": Das Quelldokument wird mit jeder Priorität einmal durchmustert, beginnend mit der niedrigsten (mode = 1, mode = 3, ..., mode = 34). Die auf eine Objektart und eine Darstellungspriorität spezialisierten Templates ermitteln nun anhand von Attributwerten der zu betrachtenden Objektart die auszuwählende Signatur, die dann als passende Style-Klasse in Form eines Parameters einem weiteren Template mitgegeben wird. Dieses Template ist eines, das punkt-, linien- oder flächenförmige Graphikobjekte erzeugt ("DrawPoint", "DrawPath", "DrawPolygon") und das abhängig vom Typ der jeweiligen Objektklasse gewählt wird.

Das folgende Beispiel-Template behandelt für die Objektklasse "Straße" die Darstellungspriorität 15 (match = "... Strasse" mode = "DP_15") und setzt damit die Ableitungsregeln für den Straßendecker von Kreis- und Gemeindestraßen um, die in Betrieb sind und nicht von einer Brücke überführt werden. Dazu werden erst die Werte einiger Attribute überprüft (z.B. "Widmung = Gemeindestrasse", "Zustand = in Betrieb"), und im Erfolgsfall wird dann anhand der Breite der Fahrbahn entschieden, welche Liniensignatur (Decker von SN3100, SN3150 oder SN3180) ausgewählt wird, d.h. welche Style-Klasse dem Tem-

plate “DrawPath” als Parameter übergeben wird. Das Template für die Konturen der hier betrachteten Straßen sieht ganz ähnlich aus: Lediglich der Wert für “mode” ist dann “DP_14” und die übergebenen Style-Klassen sind die für die Konturen, also die weiter oben definierten Klassen “SN3100linieNebenstrasseBreitHintergrund”, “SN3150linieNebenstrasseMittelHintergrund” und “SN3180linieNebenstrasseSchmalHintergrund”.

```
<xsl:template match="/dlm:AtkisModell/dlm:atkisMember/dlm:Strasse"
                                                              mode="DP_15">
 <xsl:choose>
  <xsl:when
    test="(contains(dlm:Attribute/dlm:VerkehrsbedeutungInneroertlich,
                                                 'Durchgangsverkehr') or
           contains(dlm:Attribute/dlm:VerkehrsbedeutungInneroertlich,
                                                   'Ortsverkehr') or
           contains(dlm:Attribute/dlm:VerkehrsbedeutungInneroertlich,
                                                 'Sammelverkehr') or
           contains(dlm:Attribute/dlm:VerkehrsbedeutungInneroertlich,
                                           'Attribut trifft nicht zu'))
       and (contains(dlm:Attribute/dlm:Funktion, 'Strassenverkehr'))
       and (contains(dlm:Attribute/dlm:Widmung, 'Kreisstrasse') or
            contains(dlm:Attribute/dlm:Widmung, 'Gemeindestrasse') or
            contains(dlm:Attribute/dlm:Widmung,
                                        'Attribut trifft nicht zu') or
                contains(dlm:Attribute/dlm:Widmung, 'Sonstige'))
       and (contains(dlm:Attribute/dlm:Zustand, 'in Betrieb'))
       and not(dlm:WirdVonBrueckenUeberfuehrt)">
    <xsl:choose>
      <xsl:when test="number(dlm:Attribute/dlm:BreiteDerFahrbahn)
                                                        div 10 &gt; 12">
        <xsl:call-template name="DrawPath">
          <xsl:with-param name="styleclass"
                    select="'SN3100linieNebenstrasseBreitVordergrund'" />
        </xsl:call-template>
      </xsl:when>
      <xsl:when test="number(dlm:Attribute/dlm:BreiteDerFahrbahn)
                                                         div 10 &gt; 6">
        <xsl:call-template name="DrawPath">
          <xsl:with-param name="styleclass"
                   select="'SN3150linieNebenstrasseMittelVordergrund'" />
        </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
        <xsl:call-template name="DrawPath">
          <xsl:with-param name="styleclass"
                   select="'SN3180linieNebenstrasseSchmalVordergrund"' />
        </xsl:call-template>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:when>
 </xsl:choose>
</xsl:template>
```

Das hier vorgestellte Beispiel zur Umsetzung der Ableitungsregeln für Straßen-Objekte mit der Darstellungspriorität 15 zeigt, dass die benötigten XSLT-Templates schon eine gewisse Komplexität erreichen können. Das gilt aber nicht für alle im Katalog auftretenden Kombinationen von DLM-Objektklassen und Darstellungsprioritäten. Als Beleg dazu dient das Template weiter unten, das die Objektklasse “Wald/Forst” mit der Darstellungspriorität 4 umsetzt: Hier muss nur das Attribut “Vegetationsmerkmal” überprüft werden, um dann die entsprechende Flächensignatur mit dem Template “DrawPolygon” und der Style-Klasse “SN4090flaecheWald” zu generieren.

```
<xsl:template match="/dlm:AtkisModell/dlm:atkisMember/dlm:WaldForst"
                                                              mode="DP_04">
 <xsl:choose>
  <xsl:when
    test="contains(./dlm:Attribute/dlm:Vegetationsmerkmal, 'Laubholz')
       or contains(./dlm:Attribute/dlm:Vegetationsmerkmal, 'Nadelholz')
       or contains(./dlm:Attribute/dlm:Vegetationsmerkmal,
                                                 'Laub- und Nadelholz')">
   <xsl:call-template name="DrawPolygon">
    <xsl:with-param name="styleclass" select="'SN4090flaecheWald'" />
   </xsl:call-template>
  </xsl:when>
 </xsl:choose>
</xsl:template>
```

Die hier skizzierte Methodik, die Signaturen und die Ableitungsregeln des Signaturenkataloges SK25 in SVG- und XSLT-Konstrukte umzusetzen, lässt sich recht direkt in das Verfahren der XML-basierten Erzeugung kartenähnlicher Graphiken integrieren, das wir in Abschnitt 3 sehr kurz und in *Neumann/Mathiak/Kupfer* 2004 ausführlicher beschrieben haben. Allerdings ist der dabei zu leistende Implementierungsaufwand wegen der zahlreichen zu erstellenden Style-Klassen und Templates sehr hoch. Wir haben uns daher bislang auf die Umsetzung weniger Ableitungsregeln beschränkt und so lediglich die tatsächliche Anwendbarkeit der Methodik demonstriert.

5 Ausblick

Ein Schwachpunkt unserer XML-basierten Erzeugung kartenähnlicher Graphiken ist, dass sämtliche Koordinaten der Graphikobjekte nur durch direkte Transformation der ursprünglichen Koordinaten der DLM-Objekte entstehen. Die resultierenden Graphikobjekte werden zwar genau nach den im SK25 angegebenen Darstellungsprioritäten und auch mit der jeweils umgesetzten Signatur sowie Farbe generiert, es wird jedoch kein weitergehender kartographischer Kontext nebeneinanderliegender Objekte beachtet. So kann es in unseren Darstellungen geschehen, dass etwa die Signaturen von Baumreihen durch eine linienhafte Straßensignatur teilweise verdeckt werden (vgl. Bild 4, links). Hier müssten daher die Koordinaten der Baumreihen-Objekte nicht nur einfach transformiert werden, sondern sie müssten abhängig vom Abstand zur Liniengeometrie der benachbarten Straße vorher von Fall zu Fall etwas verschoben werden (vgl. Bild 4, rechts).

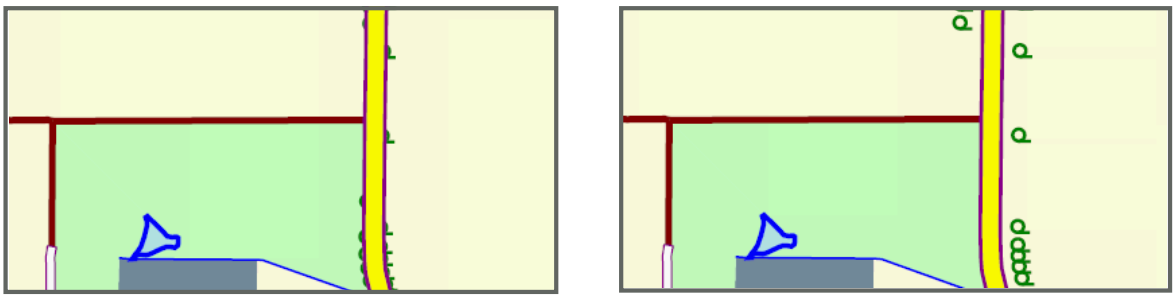

Bild 4 - Erst verdeckte, dann verdrängte Signaturen von Baumreihen

Diese bekannte Operation der kartographischen Verdrängung (*Bobrich* 2002, *Hake/Grünreich/Meng* 2002), die üblicherweise durch den Einsatz von konventionellen imperativen Programmiersprachen realisiert wird, kann nun im Prinzip auch mit XSLT implementiert werden, was durch erste Experimente bestätigt wurde (*Petri* 2005). Damit wäre dann ein weiterer Teil des kartographischen Abbildungsprozess von DLM25-Daten zur DTK25 rein XML-basiert durchführbar.

Ähnlich wie im vorliegenden Text skizziert, könnte auch der ALKIS-Signaturenkatalog (*AdV* 2004) in XML-basierte Konstrukte umgesetzt werden, und auch die XSLT-gesteuerte Generierung etwa der Liegenschafts-Karte 1:1000 als SVG-Dokument wäre analog zum Verfahren hier möglich. Dabei wäre die Umsetzung des ALKIS-Signaturenkataloges sogar einfacher als die des SK25, da die Spezifikationen der Signaturen im ALKIS-Katalog bereits vektorbasiert sind und nicht als Bemaßungen wie im SK25 angegeben werden. Auch die Ableitungsregeln scheinen direkter umsetzbar zu sein: Der neue ALKIS-Signaturenkatalog gibt die Regeln bereits in einer formalisierten Notation an. Als Beispiel hier eine Ableitungsregel für Gebäude, so wie sie im Ableitungsteil des ALKIS-Kataloges angegeben ist (links), und deren Umformulierung in logik-orientierten Pseudocode (rechts):

31001 [+] ∧ 31001 GFK = 1XXX ∧ 31001 HOH [-] ∨ 31001 HOH = FALSE ∧ 31001 ZUS [-] ∨ 31001 ZUS [+] ∧ 31001 ZUS ≠ 2200 ∨ 2300 ∨ 3000 ∨ 4000 ∧ 31001 OFL [-] → 2505 + 1301	if o.objektart = 31001 and o.GFK[1] = 1 and (o.HOH = null or not o.HOH) and (o.ZUS = null or (o.ZUS ≠ null and o.ZUS ∉ {2200, 2300, 3000, 4000})) and o.OFL = null then signatur(o, 2505); signatur(o, 1301) endif;

Man sieht, dass sich die logik-orientierte Formulierung dieser Ableitungsregel ziemlich direkt aus der Original-Regel ergibt. Und die logik-orientierte Formulierung ihrerseits ist von der Struktur her nicht weit vom entsprechenden Bedingungsteil eines XSLT-Templates zur Darstellung von Ableitungsregeln entfernt (vgl. Beispiele in Abschnitt 4). Als realisierbare Vision ist es daher durchaus vorstellbar, die Ableitungsregeln des ALKIS-Kataloges vollautomatisch durch einen passenden Übersetzer als XSLT-Templates erzeugen zu lassen. Die Konstruktion eines solchen Übersetzers wäre allerdings schon eine herausfordernde Aufgabe.

Literatur

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV): ATKIS-Signaturenkatalog 1:25000 (ATKIS-SK25). Version 4.1, Bonn, 2002.

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV): Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok). Version 3.0, Bonn, 2004.

Bobrich, J.: Kartographische Verdrängung mit MOVE. In Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Nr. 22, 2002, pp. 19–29.

Cagle, K.: SVG Programming. Apress, 2002.

Eckstein, R.; Eckstein, S.: XML und Datenmodellierung. dpunkt.verlag, 2004.

Eisenberg, J.D.: SVG Essentials. O'Reilly, 2002.

Harold, E.R.; Means, W.S.: XML in a Nutshell. O'Reilly, 2002.

Hake, G.; Grünreich, D.; Meng, L.: Kartographie – Visualisierung raum-zeitlicher Informationen. de Gruyter, 8. Aufl., 2002.

Herrmann, C.; Asche, H.: Web.Mapping 1: Raumbezogene Information und Kommunikation im Internet. Wichmann, 2000.

Kay, M.: XSLT Programmer's Reference. 2. Auflage, Wrox Press, 2001.

Kraak, M.J.; Brown, A.: Web Cartography: Developments and Prospects. Taylor & Francis, 2000.

Mathiak, B.; Kupfer, A.; Neumann, K.: Using XML Languages for Modeling and Web-Visualization of Geographical Legacy Data. In Proc. "VI Brazilian Symposium on GeoInformatics: GeoInfo 2004", Iochpe, C.; Camara, G. (Hrsg.), Instituto Nacional de Pesquisas Espaciais, Campos do Jordao 2004, pp. 265–280.

Meyer, E.A.: On CSS. Mastering the Language of Web Design with Cascading Style Sheets. New Riders, 2002.

Neumann, K.; Ahlbrecht, P.; Eckstein, E.; Mathiak, B.; Kupfer, A.: Visualization of Landscape Data in Digital Maps by Exclusive Use of XML-Based Languages. In Proc. "12th Int. Conf. on Geoinformatics", Brand, S.A. (Hrsg.), Gävle 2004, pp. 370–374.

Neumann, K.; Mathiak, B.; Kupfer, A.: Modellierung und kartographische Visualisierung von Geodaten mit XML-basierten Sprachen. In Proc. "Modellierung 2004", Rumpe, B.; Hesse, W. (Hrsg.), Lecture Notes in Informatics P-45, 2004, pp. 93–107.

Neumann, K.; Mathiak, B.; Kupfer, A.: Der Einsatz von GML, XSLT und SVG am Beispiel von ATKIS-DLM-Daten. In Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 31, Frankfurt M. 2004, pp. 97–107.

Open GIS Consortium (OGC): Geography Markup Language (GML) 3.1. 2004, http://schemas.opengis.net/gml/3.1.0/

Petri, J.: Realisierung eines Beispiels der kartographischen Verdrängung mit XSLT. Studienarbeit, TU Braunschweig, erscheint 2005.

Ray, E.T.: Einführung in XML. O'Reilly, 2001.

World Wide Web Consortium (W3C): XSL Transformations (XSLT). Version 1.0, 1999, http://www.w3.org/TR/1999/REC-xslt-19991116.

World Wide Web Consortium (W3C): Scalable Vector Graphics (SVG) 1.2 Specification. 2004, http://www.w3.org/TR/2004/WD-SVG12-20040510/