Metaautomation der Liegenschaftskarte [1]

(mit 8 Bildern)

Von Karl Neumann, Markus Grutza, Torsten Nordmann,
Florian Schlutow und Christian Wolf, Braunschweig

ZUSAMMENFASSUNG: Im ALKIS-Projekt (Amtliches Liegenschaftskataster Informationssystem) werden u.a eine Signaturenbibliothek und ein Werk von Ableitungsregeln bereitgestellt. Die Signaturenbibliothek enthält sämtliche ca. 670 Einzelsignaturen, die in einer Liegenschaftskarte verwendet werden können, und die Ableitungsregeln spezifizieren, welche Klasse von Liegenschaftsobjekten in welchem Kontext mit welcher Signatur dargestellt werden. Obwohl beide Dokumente eigentlich nur für menschliche Leser bestimmt sind und daher als PDF-Dateien zur Verfügung stehen, haben wir die Dokumente dennoch durch Programme umgesetzt: Die Signaturen wurden in Befehle der Graphiksprache SVG transformiert und die Ableitungsregeln in ein Java-Programm. Dieses generierte Programm kann nun Bestandsdatenauszüge einlesen, die im neuen NAS-Format (Normbasierte Austauschschnittstelle) vorliegen, und generiert daraus mit Hilfe der umgesetzten Signaturen Graphiken, die Liegenschaftskarten zumindest recht ähnlich sind.

ABSTRACT: The ALKIS project (official real estate cadastre information system) provides a package of signatures and a collection of derivation rules. The signature package contains all approx. 670 single signatures, which can be used in a real estate map. The derivation rules specify which classes of real estate objects are to be represented by which signatures in a given context. Although both documents are intended for human readers only and therefore are provided as PDF files, we have converted the documents by programs: The signatures were translated automatically into statements of the graphical language SVG, and the derivation rules were transformed also automatically into a JAVA program. This program now reads as input so-called inventory data extracts, given in the new NAS format (norm based exchange interface), and generates as output some map-like graphics. Because of using the translated original signatures, we obtain graphics which are at least somewhat similar to the corresponding real estate maps provided by the authorities.

1 Einleitung

In den Arbeiten *Mathiak/Kupfer/Neumann* 2005, *Neumann/Kupfer/Mathiak* 2005 sowie *Neumann/Petri/Wolf* 2006 hatten wir gezeigt, wie man XML-basierte Sprachen einsetzen kann und so mit relativ geringem Aufwand Präsentationsgraphiken generieren kann, die der Topographischen Karte 1:25.000 (TK25) zumindest recht ähnlich sind. Der Kern dieser Umsetzung von Daten des Digitalen Landschaftsmodells 1:25.000 (DLM25), die im Format der "Einheitlichen Datenbankschnittstelle" vorlagen, in Graphiken des SVG-Formates, musste dabei allerdings von Hand programmiert werden: Sowohl die Ablei-

[1] Erscheint in *Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Frankfurt M., 2007.*

tungsregeln als auch die Spezifikationen der Signaturen lagen hauptsächlich in tabellenorientierten Dokumenten vor, die bei realistischem Aufwand nur durch Menschen aber nicht durch Programme interpretierbar waren. So werden die Signaturen des Signaturenkataloges SK25 etwa durch bemaßte Zeichnungen mit erläuternden Texten definiert (vgl. auch Abschnitt 2). Wir hatten daher nach einer zuvor erarbeiteten Methodik für die wichtigsten Objektarten des DLM25 jeweils Prozeduren (Templates der Programmiersprache XSLT) implementiert und auch dazu passende Style-Klassen. Die Templates durchmusterten dann zur Laufzeit umgesetzte DLM25-Daten, benutzten die jeweiligen Style-Klassen und erzeugten so automatisch die Präsentationsgraphiken.

Im Bereich der Liegenschaftskarte ist die Ausgangssituation durch aktuelle Fortschritte im ALKIS-Projekt nun "maschinen-freundlicher": Es gibt bereits Ausgangsdaten (Bestandsdatenauszüge), die im neuen XML-basierten Format der normbasierten Austauschschnittstelle (NAS-Format) vorliegen. Allerdings sind auch hier sowohl das Werk der Ableitungsregeln als auch die Bibliothek der Signaturen weiterhin an sich als Dokumente konzipiert, die nur für menschliche Leser bestimmt sind. Die Struktur und der Inhalt dieser Dokumente erscheinen jedoch größtenteils formalisiert. Daher lag der Gedanke nahe, die ALKIS-Signaturen und -Ableitungsregeln so per Programm, also automatisch, umzusetzen, um damit ebenfalls per Programm, also nochmals automatisch, aus Bestandsdatenauszügen Graphiken zu erzeugen, die möglichst ähnlich zu Liegenschaftskarten sind. Wir versuchen somit, die automatische Ableitung der Liegenschaftskarte zu automatisieren, daher die Projektbezeichnung "Metaautomation der Liegenschaftskarte".

Im nächsten Abschnitt rekapitulieren wir kurz unsere Erfahrungen, die wir bei der Umsetzung von ATKIS-Daten in TK25-ähnliche Graphiken gemacht haben, da wir die Methode, Signaturen durch SVG-Befehle darzustellen, hier erneut benutzen. Dann gehen wir auf das neue NAS-Format ein, in dem die Bestandsdatenauszüge vorliegen. In Abschnitt 4 skizzieren wir, wie die ALKIS-Signaturen in zwei Schritten durch von uns entwickelte Programme aus der Original-Datei im PDF-Format in eine Bibliothek von SVG-Befehlen umgesetzt werden. Abschnitt 5 behandelt dann die Transformation der Ableitungsregeln, die ebenfalls in zwei Schritten durch spezielle Programme realisiert wird: Zuerst werden die Regeln nach XML umgesetzt, dann von XML in ein Java-Programm, das schließlich Bestandsdatenauszüge einliest, die Bibliothek der SVG-Befehle benutzt und Präsentationsgraphiken generiert. Im letzten Abschnitt fassen wir die Ergebnisse unserer Arbeit kurz zusammen und geben einen Ausblick auf weitere mögliche Aktivitäten.

Unsere Ausführungen in den Abschnitten 4 und 5 stellen im Wesentlichen eine stark komprimierte Sicht auf die Arbeiten von *Grutza* 2005, *Wolf* 2005, *Nordmann* 2006 und *Schlutow* 2006 dar, auf die wir deshalb an dieser Stelle ausdrücklich verweisen wollen.

2 XML-basierte Erzeugung kartenähnlicher Graphiken

Bei der Generierung TK25-ähnlicher Präsentationsgraphiken hatten wir, wie bereits kurz erwähnt, u.a. für jede darzustellende Objektklasse des DLM25 ein spezialisiertes Template implementiert. Dieses Template liest dann z.B. ein Straßenobjekt ein, nimmt eine Koor-

dinatentransformation vor und wählt abhängig von konkreten Attributen des jeweiligen Objektes die passenden zuvor implementierten SVG-Befehle zur Darstellung aus. So ist etwa die Darstellung einer Gemeindestraße in der TK25 natürlich eine andere als die einer Bundesstraße. Auf die Methodik der Implementierung der zahlreichen Templates wollen wir hier nicht näher eingehen, da sie im Kontext der Metaautomation der Liegenschaftskarte nicht von Interesse ist.

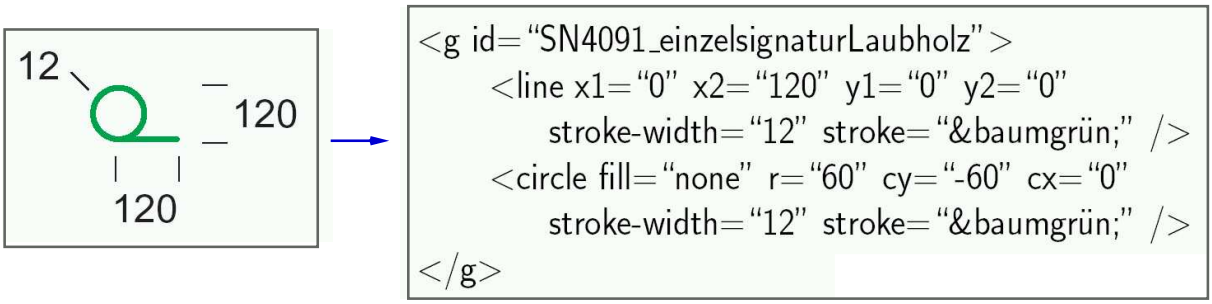


Bild 1 - Umsetzung einer SK25-Signatur nach SVG

Im Prinzip waren die einzelnen Signaturen des SK25 gut auf SVG-Befehle abbildbar. Für jede Symbolsignatur musste z.B. ein Gruppierungsbefehl angelegt werden. In Bild 1 (links) ist dazu ein sehr kleiner Ausschnitt des SK25 dargestellt: die Spezifikation der Laubholz-Einzelsignatur. Die beiden geometrischen Grundformen dieser Signatur, Linie und Kreis, werden in die SVG-Befehle "line" und "circle" umgesetzt. Man erkennt die fast direkte Übernahme der Koordinaten (Linie von (0, 120) nach (0, 0) und Kreis mit Radius 60 sowie Mittelpunkt (0, -60)). Ebenso wurde die Linienbreite (12 [Hunderstelmillimeter]) direkt übernommen. Allerdings mussten auch einige weniger offensichtliche Gegebenheiten berücksichtigt werden: So hat etwa die Y-Achse in SVG eine negative Richtung und es wird das RGB-Farbmodell benutzt; nicht das CMYK-Farbmodell wie in der Farbtabelle des SK25. Durch die Benennung einer so konstruierten Signatur mit einem identifizierenden Namen (im Beispiel "SN4091_einzelsignaturLaubholz"), kann die Signatur leicht an vielen Stellen benutzt werden, indem einfach der jeweilige Name angegeben wird und die Koordinaten, wo das Symbol platziert werden soll, z.B.:

```
<use transform="translate(45153.5,84957.9)"
     xlink:href="#SN4091_einzelsignaturLaubholz"/>
```

Dabei sind die nun verwendeten Koordinaten (im Beispiel (45153.5, 84957.9)) solche, die sich auf das Koordinatensystem der zu erstellenden Präsentationsgraphik beziehen, und sie müssen natürlich für den jeweiligen Anwendungsfall passend berechnet werden. Diese Berechnungen übernahmen – wie erwähnt – auf verschiedene Klassen spezialisierte Templates. Der Kern der Umsetzung der SK25-Signaturen ist jedoch sehr direkt, und man kann sich gut vorstellen, dass diese Umsetzung automatisierbar wäre, lägen die Spezifikationen der Signaturen in maschinen-interpretierbarer Form vor.

3 Format der Bestandsdatenauszüge

Das Amtliche Liegenschaftskataster Informationssystem (ALKIS) ermöglicht bekanntlich u.a. eine redundanz- und blattschnittfreie Speicherung von Liegenschaftsdaten, also z.B. von Flurstücken, Gebäuden, Grenzpunkten oder Gehölzen. Mit ALKIS wird ein bundeseinheitlicher Standard zur Führung der amtlichen Geobasisdaten geschaffen, und es wurde eine Datenaustauschschnittstelle vorgesehen, die auf mehreren internationalen Standards beruht: die so genannte Normbasierte Austauschschnittstelle (NAS, vgl. *ADV* 2005a). In der Definition der NAS-Schnittstelle werden Konzepte der Sprachen Geography Markup Language (GML), Web Feature Service (WFS), XML, XML-Schema und XLinks verwendet. GML ist eine XML-basierte Sprache des Open Geospatial Consortiums (OGC) und legt einen Rahmen zur Modellierung geowissenschaftlicher Welten fest; insbesondere werden hier z.B. Geometrietypen definiert. Der Web Feature Service, ebenfalls vom OGC, spezifiziert Methoden für den Austausch von Geoobjekten über das Web.

Im Kontext der vorliegenden Arbeit ist lediglich das durch die NAS-Schnittstelle festgelegte Format von Bestandsdatenauszügen relevant, also wie eine begrenzte Menge von Liegenschaftsobjekten an einen daran interessierten Benutzer geliefert wird. Zur Illustration eines solchen NAS-konformen Bestandsdatenauszuges dient uns das folgende sehr stark gekürzte Beispiel. Es handelt sich dabei um einen Ausschnitt eines Testdatenauszuges aus Niedersachsen (Quelle: *Niedersächsische Vermessungs- und Katasterverwaltung* 2006). Solche Testdatenauszüge enthalten typischerweise ca. 10.000 Liegenschaftsobjekte, während wir hier im Wesentlichen nur ein Liegenschaftsobjekt, einen Grenzpunkt, aufführen:

```
01 <AX_Bestandsdatenauszug
02   ...
03   xmlns:gml="http://www.opengis.net/gml"
04   xmlns:ogc="http://www.opengis.net/ogc"
05   ...>
06   <erlaeuterung> ...  </erlaeuterung>
07     <allgemeineAngaben> ... </allgemeineAngaben>
08     <koordinatenangaben> ...  </koordinatenangaben>
09   <wfs:FeatureCollection>
10     <gml:boundedBy> ... </gml:boundedBy>
11     <gml:featureMember>
12       <AX_Grenzpunkt gml:id="DENI3100000000a0">
13         <lebenszeitintervall>
14           <AA_Lebenszeitintervall>
15             <beginnt>20050524T12:17:56Z</beginnt>
16           </AA_Lebenszeitintervall>
17         </lebenszeitintervall>
18         <modellart>
19           <AA_Modellart>
20             <advStandardModell>DLKM</advStandardModell>
21           </AA_Modellart>
22         </modellart>
23         <anlass>000000</anlass>
24         <bestehtAus xlink:href="urn:adv:oid:DENI3100000000HI"/>
25         <abmarkung_Marke>1000</abmarkung_Marke>
26       </AX_Grenzpunkt>
27     </gml:featureMember>
28     ...
29   </wfs:FeatureCollection>
30 </AX_Bestandsdatenauszug>
```

Ein konkreter Bestandsdatenauszug wird als ein XML-konformes Objekt gekapselt (Zeile 1: Beginn, Zeile 30: Ende des Objektes). In den hier verkürzt dargestellten Zeilen 2–5 werden die importierten Namensräume deklariert, also Präfixe wie "gml", "ogc" oder "wfs". Die folgenden Erläuterungen wurden aus Platzgründen weggelassen, ebenso die allgemeinen Angaben und die Angaben zum benutzten Koordinatensystem (Zeilen 6–8); alle diese Informationen sind für die graphische Darstellung der Nutzdaten nicht von Bedeutung. Die Liegenschaftsobjekte werden als eine so genannte Feature-Collection zusammengefasst (Zeilen 9–29), für die ein geometrisches Fenster angegeben wird, innerhalb dessen alle Geometrien aller betrachteten Objekte liegen (Zeile 10, "boundedBy"). Danach folgt dann die Menge der einzelnen Liegenschaftsobjekte, hier Feature-Member genannt. Wir haben im Beispiel nur ein solches Objekt angegeben, einen Grenzpunkt (Zeilen 12–26). Dieser hat, wie jedes Liegenschaftsobjekt, einen Identifikator, hier "DENI3100000000a0" (Zeile 12). Dabei bedeutet "DE" Deutschland und "NI" Niedersachsen. Es folgen einige Attribute des Grenzpunktes, die aber hier nicht von Interesse sind, bis auf die Beziehung "besteht aus" (Zeile 24) und das Attribut "Abmarkung-Marke" (Zeile 25). Die Beziehung "besteht aus" verweist auf ein anderes Objekt, einen so genannten Punktort, dort findet man die Koordinaten des Grenzpunktes. Und der Wert 1000 von "Abmarkung-Marke" bedeutet, dass dieser Grenzpunkt keine besondere Abmarkung hat. Der Wert 1100 würde z.B. bedeuten, dass der Grenzpunkt mit einem Grenzstein markiert ist. Der Wert dieses Attributs ist also für die kartographische Darstellung eines Grenzpunktes relevant.

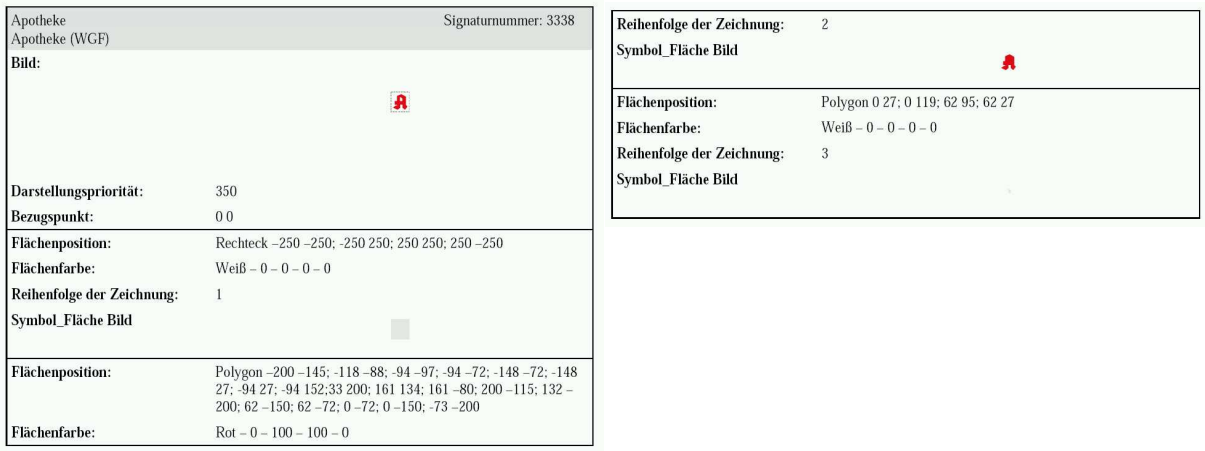
Wie erwähnt, enthalten realistische Bestandsdatenauszüge zahlreiche Objekte der unterschiedlichsten Klassen, z.B. Hunderte Flurstücke, Gebäude, Gehölze, Gewässer, Grenzpunkte, Industrie- und Gewerbeflächen, Siedlungsflächen, Wälder, Wege, Wohnbauflächen. Alle Objektarten, die hier auftreten können, sowie deren Attribute und mögliche Beziehungen zwischen den Objektarten sind detailliert im ALKIS-Objektartenkatalog (*ADV* 2005b) aufgeführt. Abschnitt 5 enthält als weiteres Beispiel eines Liegenschaftsobjektes im NAS-Format ein konkretes Gebäude.

4 Automatische Umsetzung der Signaturen

Der ALKIS-Signaturenkatalog (*ADV* 2005c, Teil B) umfasst ca. 670 einzelne Signaturen. Diese sind in die Klassen Flächen, Linien, Symbole sowie Schriften aufgeteilt, und jede Signatur wird mit allen ihren Eigenschaften nur einmal beschrieben, auch wenn sie für die Darstellung unterschiedlicher Objektklassen verwendet wird. Flächen-, Linien- und Schriftsignaturen werden mit relativ wenigen Attributen beschrieben: So hat eine Linie neben der Signaturnummer, dem Namen sowie der Darstellungspriorität häufig nur noch zwei weitere Attribute, nämlich die Strichbreite und die Farbe. Sehr viel komplexer sind dagegen die Symbolsignaturen, die sich meist aus mehreren Polygonen oder anderen Grundgeometrien zusammensetzen.

Bild 2 enthält zur Illustration einen aus Platzgründen stark verkleinerten – aber sonst im Original belassenen – Ausschnitt aus dem ALKIS-Signaturenkatalog, der als PDF-Datei vorliegt: Es handelt sich um die Beschreibung der Symbolsignatur "Apotheke". Links oben stehen zwei Bezeichnungen (Apotheke, Apotheke (WGF)), rechts davon die Signaturnummer (3338), darunter die Symbolsignatur als Bild und die Darstellungspriorität

(350). Es folgen die drei Polygone, aus denen sich die Signatur zusammensetzt: Zuerst ein weißes Rechteck, dann ein größeres rotes Polygon und schließlich ein kleines weißes Polygon. Alle Polygone sind jeweils als Vektoren angegeben und auch als Bild dargestellt. Das für die Vektoren benutzte Koordinatensystem bezieht sich auf den angegebenen Bezugspunkt (0, 0), der sich in der Mitte des weißen Rechtecks befindet. Alle Koordinaten sind Angaben in Hunderstelmillimeter, und die sich ergebende Größe der Signaturen ist für die Liegenschaftskarte mit dem Maßstab 1:1000 ausgelegt. Daher hat die hier betrachtete Symbolsignatur "Apotheke" eine Größe von 5 mal 5 Millimetern. Alle Farbangaben beziehen sich auf das CMYK-Farbmodell, so bedeutet z.B. die Farbangabe für das rote Polygon "0 - 100 - 100 - 0": 0 Prozent Cyan, 100 Prozent Magenta, 100 Prozent Gelb (Yellow) und 0 Prozent Schwarz (Key, Kontrast).



Apotheke Apotheke (WGF)	Signaturnummer: 3338
Bild:	
Darstellungspriorität:	350
Bezugspunkt:	0 0
Flächenposition:	Rechteck –250 –250; -250 250; 250 250; 250 –250
Flächenfarbe:	Weiß – 0 – 0 – 0 – 0
Reihenfolge der Zeichnung:	1
Symbol_Fläche Bild	
Flächenposition:	Polygon –200 –145; -118 –88; -94 –97; -94 –72; -148 –72; -148 27; -94 27; -94 152;33 200; 161 134; 161 –80; 200 –115; 132 – 200; 62 –150; 62 –72; 0 –72; 0 –150; -73 –200
Flächenfarbe:	Rot – 0 – 100 – 100 – 0

Reihenfolge der Zeichnung:	2
Symbol_Fläche Bild	
Flächenposition:	Polygon 0 27; 0 119; 62 95; 62 27
Flächenfarbe:	Weiß – 0 – 0 – 0 – 0
Reihenfolge der Zeichnung:	3
Symbol_Fläche Bild	

Bild 2 - Auszug aus dem ALKIS-Signaturenkatalog

Zur automatischen Umsetzung der so gegebenen ca. 670 Signaturen in die XML-basierte Vektorgraphiksprache SVG bietet es sich an, die Signaturen in einem ersten Schritt aus dem PDF-Format in ein für die weitere Verarbeitung besser geeignetes XML-Format zu konvertieren. Dazu haben wir zunächst die Struktur der Signaturen formal beschrieben, indem wir ein XML-Schema entwickelt haben, das alle Signaturklassen repräsentiert. Im Wesentlichen haben wir dazu die XML-Klasse "Signatur" definiert, die aus den Unterklassen "Fläche", "Linie", "Symbol" sowie "Schrift" besteht. Für die Unterklassen wurden dann zahlreiche weitere Klassen eingerichtet, wie z.B. "Farbe", "Polygon" oder "Koordinaten". Schließlich wurde ein Programm (in Java) realisiert, das den reinen Textanteil des gegebenen Signaturenkataloges einliest und ihn als XML-Dokument ausgibt, das dem entworfenen XML-Schema folgt. Dabei wurden sämtliche in der ursprünglichen PDF-Datei vorhandenen Abbildungen und Seitenlayout-Informationen entfernt. In Bild 3 ist ein sehr kleiner Ausschnitt des resultierenden XML-Dokumentes wiedergegeben: Die XML-Repräsentation der oben diskutierten Symbolsignatur "Apotheke". Aus Platzgründen musste der dargestellte Ausschnitt leider sehr verkleinert werden. Dennoch ist die prinzipielle Struktur zu erkennen: In der ersten Zeile stehen die Signaturnummer, der Typ und die Priorität. Dann folgen die zwei Bezeichnungen und die drei Polygone (weißes Rechteck, großes rotes Polygon, kleines weißes Polygon). Während die XML-Repräsentation der Symbolsigna-

tur “Apotheke” 93 Zeilen lang ist, umfassen alle nach XML umgesetzten Signaturen ca. 44.000 Zeilen.

```
<Signatur Nr="3338" Typ="Symbol" Darstellungsprioritaet="350">
 <SymbolSignatur>
  <Bezeichnung>Apotheke</Bezeichnung>
  <Bezeichnung>Apotheke (WGF)</Bezeichnung>
  <Bezugspunkt>
   <Koordinate X="0.0" Y="0.0" />
  </Bezugspunkt>
  <Details>
   <Komplex>
    <Zeichnung>
     <Flaeche>
      <Flaechenposition>
       <Rechteck>
        <Koordinaten>
         <Koordinate X="-250.0" Y="-250.0" />
         <Koordinate X="-250.0" Y="250.0" />
         <Koordinate X="250.0" Y="250.0" />
         <Koordinate X="250.0" Y="-250.0" />
        </Koordinaten>
       </Rechteck>
      </Flaechenposition>
      <Flaechenfarbe>
       <Farbgrundton>Weiss</Farbgrundton>
       <Cyan>0</Cyan>
       <Magenta>0</Magenta>
       <Yellow>0</Yellow>
       <Black>0</Black>
      </Flaechenfarbe>
      <ReihenfolgeDerZeichnung>1</ReihenfolgeDerZeichnung>
     </Flaeche>
    </Zeichnung>
```

```
<Zeichnung>
 <Flaeche>
  <Flaechenposition>
   <Polygon>
    <Koordinaten>
     <Koordinate X="-200.0" Y="-145.0" />
     <Koordinate X="-118.0" Y="-88.0" />
     <Koordinate X="-94.0" Y="-97.0" />
     <Koordinate X="-94.0" Y="-72.0" />
     <Koordinate X="-148.0" Y="-72.0" />
     <Koordinate X="-148.0" Y="27.0" />
     <Koordinate X="-94.0" Y="27.0" />
     <Koordinate X="-94.0" Y="152.0" />
     <Koordinate X="33.0" Y="200.0" />
     <Koordinate X="161.0" Y="134.0" />
     <Koordinate X="161.0" Y="-80.0" />
     <Koordinate X="200.0" Y="-115.0" />
     <Koordinate X="132.0" Y="-200.0" />
     <Koordinate X="62.0" Y="-150.0" />
     <Koordinate X="62.0" Y="-72.0" />
     <Koordinate X="0.0" Y="-72.0" />
     <Koordinate X="0.0" Y="-150.0" />
     <Koordinate X="-73.0" Y="-200.0" />
    </Koordinaten>
   </Polygon>
  </Flaechenposition>
  <Flaechenfarbe>
   <Farbgrundton>Rot</Farbgrundton>
   <Cyan>0</Cyan>
   <Magenta>100</Magenta>
   <Yellow>100</Yellow>
   <Black>0</Black>
  </Flaechenfarbe>
  <ReihenfolgeDerZeichnung>2</ReihenfolgeDerZeichnung>
 </Flaeche>
</Zeichnung>
```

```
<Zeichnung>
 <Flaeche>
  <Flaechenposition>
   <Polygon>
    <Koordinaten>
     <Koordinate X="0.0" Y="27.0" />
     <Koordinate X="0.0" Y="119.0" />
     <Koordinate X="62.0" Y="95.0" />
     <Koordinate X="62.0" Y="27.0" />
    </Koordinaten>
   </Polygon>
  </Flaechenposition>
  <Flaechenfarbe>
   <Farbgrundton>Weiss</Farbgrundton>
   <Cyan>0</Cyan>
   <Magenta>0</Magenta>
   <Yellow>0</Yellow>
   <Black>0</Black>
  </Flaechenfarbe>
  <ReihenfolgeDerZeichnung>3</ReihenfolgeDerZeichnung>
 </Flaeche>
</Zeichnung>
</Komplex>
</Details>
</SymbolSignatur>
</Signatur>
```

Bild 3 - XML-Repräsentation der Symbolsignatur “Apotheke”

Der nächste Schritt, die Umsetzung der nunmehr XML-basierten Signaturen in SVG-Befehle, kann jetzt recht direkt durchgeführt werden: Flächen-, Linien- sowie Schriftsignaturen werden auf Style-Klassen und Symbolsignaturen werden auf Gruppierungsbefehle abgebildet, ähnlich wie bereits in Abschnitt 2 skizziert. Dabei haben annähernd alle Attribute der Signaturen, wie etwa Linienbreite, Farbe oder Schriftart, direkte Entsprechungen auf der Seite der SVG-Befehle. Lediglich für die Darstellungspriorität gibt es kein äquivalentes SVG-Sprachmittel.

Wir zeigen am Beispiel der Symbolsignatur “Apotheke” wie eine XML-basierte Signatur nach SVG umgesetzt wird: Da sie eine Symbolsignatur ist, wird sie auf einen Gruppierungsbefehl abgebildet (SVG-Code weiter unten, Zeilen 1–12). Als Identifikator wählen wir “SN” (für Signaturnummer), gefolgt von der Signaturnummer, hier 3338, gefolgt vom Signaturtyp, hier “Symbol”, gefolgt von der ersten Signaturbezeichnung und “Etc”, falls es weitere Bezeichnungen gibt; hier also insgesamt “SN3338SymbolApothekeEtc” (Zeile 1). Da es keine SVG-Entsprechung für die Darstellungspriorität gibt, diese wichtige Information aber für die Generierung von Liegenschaftskarten später noch benötigt wird, übernehmen wir die Priorität stets als zweite Zeile in Form eines SVG-Kommentars. Es folgen jetzt noch die drei Polygone, die jeweils als Path-Befehl umgesetzt werden: Weißes Rechteck (Zeilen 3 und 4), großes rotes Polygon (Zeilen 5–9) sowie kleines weißes Polygon (Zeilen 10 und 11). Bei der Umsetzung der Koordinaten werden die X-Werte einfach unverändert übernommen, während wegen der negativen Richtung der Y-Achse in SVG das Vorzeichen der Y-Werte umgekehrt wird. Außerdem werden die Farbwerte vom CMYK-Farbmodell in das RGB-Farbmodell umgerechnet, so wird z.B. aus “0, 100, 100, 0” (rot) der Wert “FF0000” (Zeile 9).

```
01 <g id="SN3338SymbolApothekeEtc">
02  <!-- Prioritaet 350 -->
```

```
03  <path d="M -250.0 250.0 L -250.0 -250.0 250.0 -250.0 250.0 250.0 Z"
04        fill= "#FFFFFF"/>
05  <path d="M -200.0 145.0 L -118.0 88.0 -94.0 97.0 -94.0 72.0 -148.0 72.0
06           -148.0 -27.0 -94.0 -27.0 -94.0  -152.0 33.0 -200.0 161.0 -134.0
07           161.0 80.0 200.0 115.0 132.0 200.0 62.0 150.0 62.0 72.0 0.0
08           72.0 0.0 150.0 -73.0 200.0 Z"
09        fill= "#FF0000"/>
10  <path d="M 0.0 -27.0 L 0.0 -119.0 62.0 -95.0 62.0 -27.0 Z"
11        fill= "#FFFFFF"/>
12 </g>
13
14 .SN1304FlaecheGebaeudeEtc
15  <!-- Prioritaet 290 -->
16  {fill: #CCCCCC;}
17
18 .SN2505LinieGebaeudeEtc
19  <!-- Prioritaet 300 -->
20  {fill: none;
21   stroke-width: 18;
22   stroke-linecap: butt;
23   stroke-linejoin: miter;
24   stroke: #000000;}
```

Die Zeilen 14–16 enthalten ein Beispiel für eine sehr einfache Style-Klasse, die durch die Umsetzung einer ebenfalls sehr einfachen Flächensignatur entstanden ist: Die Signatur "SN1304" spezifiziert lediglich eine graue gefüllte Fläche, daher besteht die erzeugte Style-Klasse auch nur aus dem Identifikator, der Darstellungspriorität als Kommentar und dem Füllbefehl mit dem Farbwert "CCCCCC" (grau). Die Umsetzung einer Liniensignatur stellt die nächste Style-Klasse dar (Zeilen 18–24): Hier wurden Attribute wie "Linienbreite", "Linienabschluss" und "Linienscheitel" direkt in die SVG-Elemente "stroke-width", "stroke-linecap" und "stroke-linejoin" umgesetzt, wobei z.B. dem Originalwert "abgeschnitten" jetzt der Wert "butt" entspricht.

```
01 <path class="SN1304FlaecheGebaeudeEtc"
02       d="M 2925 2475 L 2925 2700 3600 2700 3600 2475 4500 2475 4500 4275
03          3150 4275 3150 3600 2025 3600 2025 2925 2475 2925 2475 2475
04          2925 2475"/>
05
06 <path class="SN2505LinieGebaeudeEtc"
07       d="M 2925 2475 L 2925 2700 3600 2700 3600 2475 4500 2475 4500 4275
08          3150 4275 3150 3600 2025 3600 2025 2925 2475 2925 2475 2475
09          2925 2475"/>
10
11 <use transform="translate(3600,3375)"
12      xlink:href="#SN3338SymbolApothekeEtc"/>
```

Die so umgesetzten Signaturen können nun in SVG-Dokumenten sehr leicht benutzt werden, wie bereits in Abschnitt 2 kurz dargestellt wurde. Die obigen Zeilen illustrieren dies am Beispiel der Darstellung eines Gebäudes mit Apotheken-Signatur: Zuerst wird eine graue Fläche gezeichnet, wobei jetzt konkrete Karten-Koordinaten benutzt werden müssen (Zeilen 1–4). Dazu wird der Path-Befehl verwendet, der als Parameter die passende Style-Klasse erhält. Danach wird, mit denselben Koordinaten aber einer anderen Style-Klasse, eine schwarze Umrandung generiert (Zeilen 6-9) und schließlich das bekannte Apothekensymbol platziert (Zeilen 11 und 12). In Bild 4 ist die Ausführung der drei SVG-Befehle dargestellt.

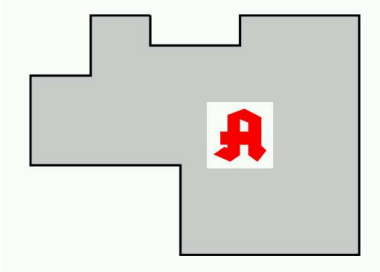


Bild 4 - Gebäude mit Symbolsignatur “Apotheke” als Anwendung dreier SVG-Befehle

Diese Benutzung der in zwei Schritten vom PDF-Format nach SVG-Befehlen umgesetzten Signaturen, die für das hier präsentierte Beispiel “per Hand” durchgeführt wurde, muss bei realistischen Datenbeständen und deren Visualisierung natürlich automatisch geschehen. Dazu werden die ALKIS-Ableitungsregeln entsprechend in ein Programm transformiert, das dann diese Aufgabe übernimmt.

5 Automatische Umsetzung der Ableitungsregeln

Die ALKIS-Ableitungsregeln (*ADV* 2005c, Teil C) legen fest, welches Liegenschaftsobjekt in welchem Kontext mit welcher Signatur in einer Liegenschaftskarte darzustellen ist. So wird etwa ein Gebäude, das allgemein eine Gewerbe- oder Wirtschaftsfunktion hat, durch die Flächensignatur Nr. 1304 (hellgraue gefüllte Fläche) und die Liniensignatur Nr. 2505 (schmale durchgezogene schwarze Linie) dargestellt. Die ca. 850 Regeln setzen sich meist aus mehreren Bedingungen zusammen, die durch Und bzw. Oder verknüpft sind. Die einzelnen Bedingungen können die Existenz eines Objektes oder Attributes, das Vorliegen eines bestimmten Attributwertes oder das Vorhandensein einer Beziehung zu anderen Objekten überprüfen. Bild 5 zeigt als Beispiel eine solche Ableitungsregel, wie sie so im Regelkatalog, der als PDF-Datei zur Verfügung steht, aufgeführt wird.

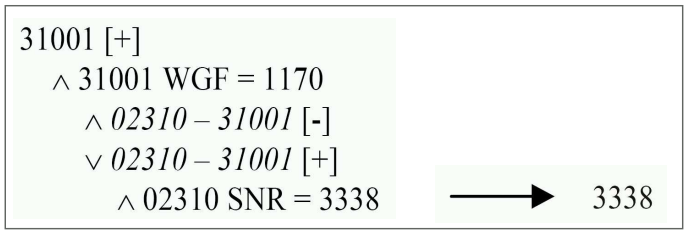


Bild 5 - Beispiel einer Ableitungsregel

Die Bedeutung dieser Regel ist folgende: Wenn das aktuell betrachtete Objekt aus der Klasse “Gebäude” ist (Gebäudeklasse-Kennung: 31001), und wenn das Gebäude-Attribut “weitere Gebäudefunktion” (WGF) denn Wert 1170 (Apotheke) hat, und wenn keine Beziehung zwischen dem aktuellen Objekt – also dem Gebäude – und einem punktförmigen Präsentationsobjekt besteht (Präsentationsobjekt-Kennung: 02310), dann wird eine Symbolsignatur mit der Nummer 3338 (Apotheke) erzeugt, und zwar auf einer Position, die sich aus den Koordinaten des Gebäudes ergibt (die konkrete Rechenvorschrift dafür ist

nicht näher festgelegt). Existiert dagegen eine Beziehung zu einem punktförmigen Präsentationsobjekt, so wird dessen Positionswert zur Platzierung der Symbolsignatur benutzt, falls das Attribut “Signaturnummer” (SNR) des Präsentationsobjektes den Wert 3338 hat.

Das Beispiel zeigt, dass die Syntax der Ableitungsregeln zwar sehr stark an logische Ausdrücke und Formeln erinnert, ihre Interpretation jedoch von der aus der Logik gewohnten abweicht. Tatsächlich gibt die Einrückungsebene sogar die syntaktische Schachtelung der durch Und und Oder verknüpften Bedingungen wieder, die explizit aufgeführten Und- und Oder-Symbole sind daher redundant: Bedingungen auf gleicher Ebene sind durch Oder verknüpft, im Sinne eines “IF...ELSE IF...”-Konstruktes, und eingerückte Bedingungen sind mit der darüber stehenden nicht eingerückten Bedingung durch Und verknüpft, im Sinne eines “IF...THEN IF...”-Konstruktes. Berücksichtigt man nun noch die (implizite) Rahmenvorschrift, nach der bei nicht vorhandener Beziehung zu einem passenden Präsentationsobjekt die Platzierungskoordinaten für eine zu generierende Symbolsignatur aus der Geometrie des aktuell darzustellenden Liegenschaftsobjektes zu berechnen sind, so lassen sich die Ableitungsregeln sinnvoll interpretieren.

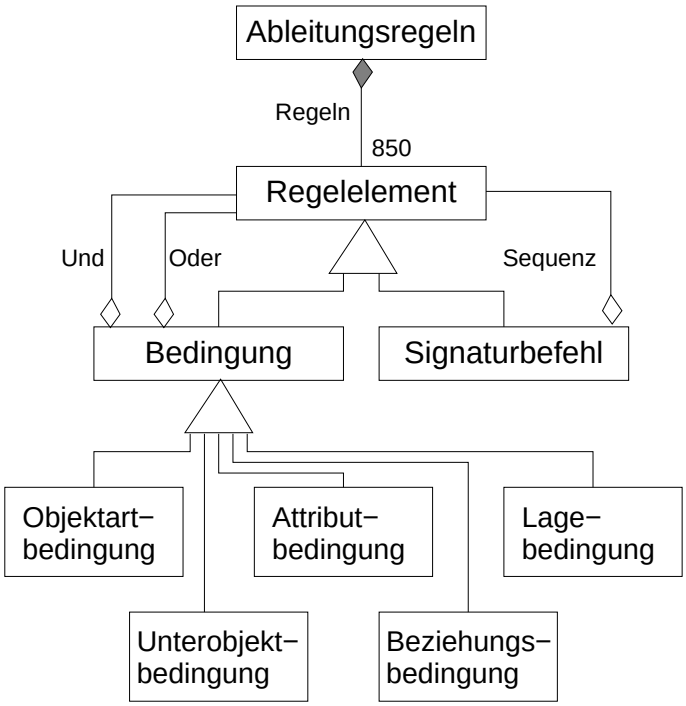


Bild 6 - vereinfachtes Klassendiagramm des Schemas der Ableitungsregeln

Zur Umsetzung der Ableitungsregeln in ein Programm, das die Regeln schließlich ausführt, haben wir zunächst erneut ein XML-Schema erarbeitet. Dieses beschreibt nun die Struktur der Ableitungsregeln und ist in vereinfachter Form in Bild 6 als Klassendiagramm dargestellt: Die ca. 850 ALKIS-Ableitungsregeln bestehen aus Regelelementen, die entweder Bedingungen oder Signaturbefehle sein können. Mehrere Bedingungen können durch Und oder Oder verknüpft werden, und mehrere Signaturbefehle können Sequenzen bilden. Die Bedingungen sind entweder Objektartbedingungen, Attributbedingungen, Unterobjektbedingungen, Beziehungsbedingungen oder – recht selten – Lagebedingungen. So enthält

etwa die Ableitungsregel aus Bild 5 u.a. eine Objektartbedingung und zwei Attributbedingungen. Die Transformation der als PDF-Datei gegebenen Ableitungsregeln übernimmt ein von uns entwickeltes Programm, wobei hier die Skriptsprache Python eingesetzt wurde, die sich besonders gut zur Verarbeitung von Zeichenketten eignet. Das Programm liest den Textanteil der Quelldatei ein, ignoriert wieder Seitenlayout-Informationen und gibt die Regeln als XML-Dokument aus, dessen Struktur dem diskutierten Schema folgt.

```
01 <ObjektartBedingung wert="AX_Gebaeude">
02   <AttributBedingung ausdruck="=1170"
03                      name="weitereGebaeudefunktion">
04     <BeziehungsBedingung null="true" ziel="AP_PPO">
05       <Signatur nummer="3338"/>
06     </BeziehungsBedingung>
07     <BeziehungsBedingung null="false" ziel="AP_PPO">
08       <AttributBedingung ausdruck="=3338"
09                          name="signaturnummer"
10                          objektart="AP_PPO">
11         <Signatur nummer="3338" positionVon="AP_PPO"/>
12       </AttributBedingung>
13     </BeziehungsBedingung>
14   </AttributBedingung>
15 </ObjektartBedingung>
```

Zur Illustration dieser Transformation dient der oben angeführte XML-Code, der die Umsetzung der Ableitungsregel aus Bild 5 darstellt: In Zeile 1 beginnt die umgesetzte Objektartbedingung mit der Überprüfung, ob das aktuelle Liegenschaftsobjekt der Klasse "AX_Gebaeude" angehört. Da Objektartbedingungen immer einen Vergleich auf Gleichheit bedeuten, wird hier kein Vergleichsoperator angegeben sondern nur der jeweilige Wert der Objektart. In dieser Phase der Umsetzung werden auch die verschiedenen Kennungen und Abkürzungen umgesetzt. Beispiele dafür sind "AX_Gebaeude" statt "31001" oder "AP_PPO" (punktförmiges Präsentationsobjekt) statt "02310" und "weitereGebaeudefunktion" statt "WGF". Diese Kennungen und Abkürzungen werden zwar in den originalen Ableitungsregeln benutzt, nicht aber im NAS-Format der Bestandsdatenauszüge. Falls das aktuelle Objekt also ein Gebäude ist, wird die Attributbedingung (Zeilen 2 und 3) betrachtet. Hier wird überprüft, ob das Gebäude-Attribut "weitereGebaeudefunktion" gleich dem Wert 1170 ist (ausdruck="=1170"). Im Gegensatz zu den Objektartbedingungen können in Attributbedingungen verschiedene Vergleiche auftreten, etwa $=$, $\neq$, $<$ oder $>$. Daher wird der jeweilige Vergleichsoperator hier mit dem Vergleichswert angegeben. An dieser Stelle wird deutlich, dass wir das explizite Und und Oder jetzt implizit durch die Schachtelung unserer XML-Struktur ausdrücken: Der Übergang zur nächsten Ebene (Einrückung) bedeutet eine Und-Verknüpfung während Elemente auf der gleichen Ebene durch Oder verknüpft sind. Im Beispiel folgt eine Beziehungsbedingung (Zeile 4), die feststellt, ob keine Beziehung (null="true") zwischen dem betrachteten Gebäude und einem punktförmigen Präsentationsobjekt besteht. Besteht keine solche Beziehung, wird eine Symbolsignatur mit der Nummer 3338 (Apotheke) erzeugt, wobei keine expliziten Koordinaten für deren Position angegeben werden (Zeile 5). Mit Oder ist die nächste Beziehungsbedingung (Zeile 7) verknüpft, die Überprüfung, ob es eine Beziehung gibt (null="false"). Im positiven Fall wird durch die Attributbedingung (Zeilen 8–10) ermittelt, ob der Wert des Attributes "Signaturnummer" des Präsentationsobjektes gleich 3338 ist. Falls das zutrifft, wird nun eine Symbolsignatur "Apotheke" erzeugt (Zeile 11, nummer="3338"), deren Position vom Präsentationsobjekt übernommen wird (positionVon="AP_PPO").

Wie hier nur kurz an einem Beispiel skizziert, werden sämtliche ALKIS-Ableitungsregeln in unser XML-Format umgesetzt. Das resultierende XML-Dokument hat schließlich einen Umfang von ca. 10.000 Zeilen, und es ist für die weitere Verarbeitung der Ableitungsregeln sehr viel besser geeignet als das ursprüngliche PDF-Dokument.

Im nächsten Schritt werden die Ableitungsregeln in ein Programm umgesetzt, das dann seinerseits die Bibliothek der SVG-Befehle benutzt, die die transformierten ALKIS-Signaturen darstellen, und das Bestandsdatenauszüge einliest, um daraus Präsentationsgraphiken zu generieren. Dieses Programm wird zum großen Teil selbst generiert, und zwar durch ein Programm, das die XML-basierten Ableitungsregeln einliest und als Resultat das "regel-ausführende" Programm ausgibt. Am Beispiel der bekannten Ableitungsregel aus Bild 5, deren Umsetzung in unser XML-Format wir weiter oben skizzierten, soll kurz erläutert werden, wie Ableitungsregeln in Java-Code transformiert werden. Für jede Objektklasse, die in einer Liegenschaftskarte dargestellt werden soll, wird eine Methode generiert, die alle Regeln für die jeweilige Klasse umsetzt. Im Folgenden wird daher lediglich ein kleiner Ausschnitt der Methode "AX_Gebaeude_methode" diskutiert, wobei von einigen Implementierungsdetails abstrahiert wird:

```
01 void AX_Gebaeude_methode(){
02  ...
03  if (verarbeiteAttributBedingung("weitereGebaeudefunktion","=1170")){
04     if (sucheBeziehung("true","AP_PPO")){
05        sig=new Signatur();
06        sig.setNummer(3338);
07        signaturen.add(sig);
08     }
09     if (sucheBeziehung("false","AP_PPO")){
10        if (verarbeiteAttributBedingung("AP_PPO","signaturnummer","=3338")){
11           sig=new Signatur();
12           sig.setNummer(3338);
13           p=sucheObjekt("AP_PPO");
14           sig.setPositionVon(positionsberechnung(p));
15           signaturen.add(sig);
16        }
17     }
18  }
19  ...
20 }
```

Die Methode wird aufgerufen, wenn im gerade betrachteten Bestandsdatenauszug ein Gebäudeobjekt gefunden wurde. Mit dem Aufruf der booleschen Methode "verarbeiteAttributBedingung" (Zeile 3) wird festgestellt, ob das Attribut "weitereGebaeudefunktion" des aktuellen Gebäudeobjektes den Wert 1170 hat. Man erkennt, dass die ursprüngliche Attributbedingung einfach in den Methodenaufruf mit den entsprechenden Parametern umgesetzt wurde. Im positiven Fall wird eine weitere boolesche Methode aufgerufen, die ganz analog die Beziehungsbedingung umsetzt (Zeile 4), also ob es keine Beziehung vom Gebäudeobjekt zu einem punktförmigen Präsentationsobjekt gibt. Falls das der Fall ist, wird ein neues Signaturobjekt mit der Nummer 3338 erzeugt (Zeilen 5 und 6), das der Menge der Signaturobjekte des aktuellen Liegenschaftsobjektes hinzugefügt wird (Zeile 7). Danach (Oder-Implementierung) wird auf eine vorhandene Beziehung getestet (Zeile 9) und schließlich nach der Überprüfung des Attributes "Signaturnummer" des Präsentationsobjektes (Zeile 10) im positiven Fall ein Signaturobjekt mit der vom Präsentationsobjekt abgeleiteten Position erzeugt (Zeilen 11–15).

Die Umsetzung der XML-basierten Regeln ist also recht direkt. Allerdings mussten die im resultierenden Java-Code aufgerufenen Methoden natürlich auch implementiert werden, also z.B. die Methoden “verarbeiteAttributBedingung” oder “positionsberechnung”. Des Weiteren waren einige Java-Strukturen zu realisieren, die etwa die erzeugten Signaturen aufnehmen können. Hinter der Erzeugung der Signaturen verbirgt sich auch eine – allerdings recht einfache – Koordinatentransformation der Objekt-Koordinaten, die im NAS-Format gegeben sind, in Karten-Koordinaten. Diese Karten-Koordinaten werden benötigt, um die nach SVG umgesetzten Signaturen als konkrete Ausprägungen benutzen zur können, wie in Abschnitt 4 dargestellt. Das abschließende sehr stark verkürzte Beispiel soll die Anwendung der umgesetzten Ableitungsregeln an einem Gebäudeobjekt illustrieren, das im NAS-Format als Ausschnitt eines Bestandsdatenauszuges gegeben ist:

```
01 <gml:featureMember>
02   <AX_Gebaeude gml:id="DEHHSERV00001FN1">
03     ...
04      <position>
05       <gml:Polygon>
06        <gml:exterior>
07         <gml:Ring>
08          ...
09           <gml:pos>3567807.047 5930017.550</gml:pos>
10           <gml:pos>3567810.850 5930024.755</gml:pos>
11           <gml:pos>3567810.850 5930024.755</gml:pos>
12           <gml:pos>3567814.476 5930022.841</gml:pos>
13            ...
14           <gml:pos>3567817.675 5930011.936</gml:pos>
15           <gml:pos>3567807.047 5930017.550</gml:pos>
16          ...
17         </gml:Ring>
18        </gml:exterior>
19       </gml:Polygon>
20      </position>
21      <gebaeudefunktion>2000</gebaeudefunktion>
22      <weitereGebaeudefunktion>1170</weitereGebaeudefunktion>
23      <bauweise>2100</bauweise>
24      <anzahlDerOberirdischenGeschosse>1</anzahlDerOberirdischenGeschosse>
25      <anzahlDerUnterirdischenGeschosse>1</anzahlDerUnterirdischenGeschosse>
26      <dachform>3100</dachform>
27   </AX_Gebaeude>
28 </gml:featureMember>
```

Das Gebäude ist – wie in Abschnitt 3 ausgeführt – ein Feature-Member und damit Teil einer übergeordneten Feature-Collection, die hier wie zahlreiche weitere Details aus Platzgründen weggelassen wurde. Ein Rahmenprogramm sorgt dafür, dass ein gegebener Bestandsdatenauszug nach auftretenden Liegenschaftsobjekten durchsucht wird. Im Beispiel wird also in Zeile 2 das Gebäudeobjekt mit dem Identifikator “DEHHSERV00001FN1” gefunden. Daraufhin wird die Java-Methode “AX_Gebaeude_methode” aufgerufen und die in ihr enthaltenen umgesetzten Gebäude-Ableitungsregeln durchmustern die Zeilen 3–27 nach zutreffenden Bedingungen. Für das angegebene Gebäudeobjekt werden zwei Bedingungen zu wahr ausgewertet: Einmal (Zeile 21) handelt es sich um ein Gebäude mit der Gebäudefunktion 2000 (Gewerbe- oder Wirtschaftsfunktion allgemein) und zum anderen (Zeile 22) um ein Gebäude mit der weiteren Gebäudefunktion 1170 (Apotheke). Die erste umgesetzte Ableitungsregel, die wir bislang in Abschnitt 3 nur kurz erwähnt haben, wird daher als Ergebnis zwei Java-Signaturobjekte zurückliefern mit den Signaturnummern 1304 (hellgraue gefüllte Fläche) und 2505 (schmale durchgezogene schwarze

Linie). Die zweite Regel, deren Umsetzung wir oben diskutiert haben, sucht nach einem punktförmigen Präsentationsobjekt, das mit dem gerade betrachteten Gebäudeobjekt in Beziehung steht. Wir nehmen an, dass ein solches Präsentationsobjekt nicht existiert. Deshalb wird hier ein Java-Signaturobjekt mit der Signaturnummer 3338 (Apothekensymbol) und mit nicht gesetzter Positionsangabe zurückgeliefert. Danach erhält das Rahmenprogramm wieder die Kontrolle und generiert aus den gelieferten drei Signaturobjekten die passenden SVG-Befehle. Dazu erzeugt es einen Path-Befehl mit der Style-Klasse "SN1304FlaecheGebaeudeEtc" und einen mit der Style-Klasse "SN2505LinieGebaeudeEtc", außerdem einen Use-Befehl mit der Marke "SN3338SymbolApothekeEtc". Die Koordinaten für die beiden Path-Befehle resultieren aus der Positionsangabe des betrachteten Gebäudes (Zeilen 4–20), wobei die erwähnte Koordinatentransformation durchgeführt wird. Die Koordinaten zur Platzierung des Apothekensymbols müssen ebenfalls aus der Positionsangabe des Gebäudes berechnet werden, da die Positionsangabe des zurückgelieferten Signaturobjektes leer ist (es wurde kein zugeordnetes Präsentationsobjekt gefunden). Als sehr einfache Lösung für diesen Fall wird hier stets der Mittelpunkt des umschließenden Rechtecks gewählt.

Auf diese hier an einem Beispiel skizzierte Weise durchläuft das Rahmenprogramm alle Objekte eines gegebenen Bestandsdatenauszuges und ruft für jedes gefundene Objekt die generierten Java-Methoden auf. Diese durchmustern das Objekt mit der jeweils zur Objektklasse passenden Methode und liefern gemäß den umgesetzten Ableitungsregeln Signaturobjekte zurück. Die Signaturobjekte werden dann vom Rahmenprogramm in SVG-Befehle transformiert, wobei die zuvor umgesetzten Signatur-Spezifikationen benutzt werden. An dieser Stelle werden die als Kommentare notierten Darstellungsprioritäten benötigt, um die erzeugten SVG-Befehle in der richtigen Reihenfolge – nach aufsteigender Priorität geordnet – in die SVG-Ausgabedatei zu schreiben. Die Ausgabedatei ist insgesamt das Resultat der automatischen Visualisierung eines Bestandsdatenauszuges.

6 Ergebnisse und Ausblick

Im vorliegenden Text haben wir unser Vorhaben skizziert, die gegebenen Bibliotheken der ALKIS-Signaturen und -Ableitungsregeln so per Programm umzusetzen, um damit ebenfalls per Programm aus Bestandsdatenauszügen Graphiken zu erzeugen, die möglichst ähnlich zu Liegenschaftskarten sind. Dazu haben wir u.a. das noch recht neue NAS-Format vorgestellt, in dem die Bestandsdatenauszüge vorliegen. Außerdem haben wir angerissen, wie die ALKIS-Signaturen in zwei Schritten durch von uns entwickelte Programme aus der Original-Datei im PDF-Format in eine Bibliothek von SVG-Befehlen umgesetzt werden. Schließlich haben wir die beiden Schritte zur Transformation der Ableitungsregeln erläutert: Zunächst werden die Regeln nach XML umgesetzt, dann von XML in eine Sammlung von Java-Methoden. Ein Rahmenprogramm benutzt diese Methoden und die umgesetzten ALKIS-Signaturen, um Bestandsdatenauszüge einzulesen und Präsentationsgraphiken zu generieren.

Wir haben mit unserem Programmsystem Testdatenauszüge aus Baden-Württemberg, Niedersachsen und Rheinland-Pfalz visualisiert (Quellen der Daten: *Landesvermessungsamt Baden-Württemberg* 2006, *Niedersächsische Vermessungs- und Katasterverwaltung*

2006, *Landesamt für Vermessung und Geobasisinformation Rheinland-Pfalz* 2006). Bild 7 zeigt einen Ausschnitt der aus dem Niedersächsischen Testdatenauszug generierten Graphik. Leider musste die Abbildung von einer Bildschirmdarstellung gewonnen werden. Die Auflösung ist deshalb recht gering und auch der Maßstab von 1:1000 wurde nicht genau getroffen. Dennoch kann man erkennen, dass die generierte Graphik in der Tat ähnlich zu einer Liegenschaftskarte ist. So sieht man links oben die Darstellung eines Gebäudes mit der allgemeinen Funktion “Gemeinwesen” und der speziellen Gebäudefunktion “Polizei”. Ferner gibt es einige weitere Gebäude, die meisten davon mit einer Hausnummer, Flurstücke mit Flurstücknummern, zahlreiche Grenzpunkte, usw.

Bild 7 - Ausschnitt aus einer generierten Graphik

Unser Versuch, die automatische Ableitung der Liegenschaftskarte zu automatisieren, ist insgesamt recht erfolgreich verlaufen, war jedoch nicht völlig ohne Probleme. So stellte sich erst nach der zweistufigen Umsetzung des Signaturenkataloges heraus, dass Informationen, die für die Transformation einiger Signaturen wichtig waren, bereits in der ersten Umsetzung nicht berücksichtigt werden konnten. Die betroffenen Symbolsignaturen, z.B. die Haltestellen- und die Parkplatzsignatur, enthalten neben ihrer geometrischen Formbeschreibung jeweils einen Buchstaben; für die Beispiele: ein “H” bzw. ein “P”. Und diese Buchstaben waren im Ausgangsdokument zwar als Abbildungen aber nicht explizit als Zeichen enthalten. Da beim Übergang vom originalen PDF-Dokument zur XML-Repräsentation der Signaturen alle Abbildungen entfernt wurden, ging die entsprechende Information verloren und musste später in den generierten SVG-Befehlen nachgetragen werden. Außerdem gab es einen Fehler bei der Umsetzung der Ableitungsregeln: Es stellte sich heraus, dass die Syntax der Ableitungsregeln für Hausnummern einen kleinen Unterschied zu allen anderen Regeln aufweist: Es wird dort eine Beziehung in inverser Richtung benutzt. Daher fehlten auf den ersten generierten Graphiken sämtliche Hausnummern, was aber inzwischen durch die Arbeit *Melching* 2006 geändert wurde.

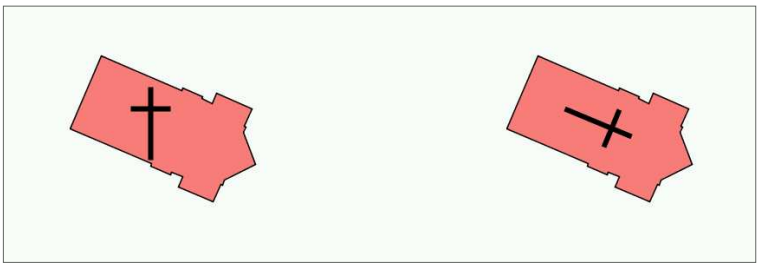


Bild 8 - alternative Platzierung von Symbolsignaturen

Wie zu erwarten war, ist die Platzierung von Symbolsignaturen, für die es in den Bestandsdatenauszügen keine entsprechenden Präsentationsobjekte gibt, nicht optimal. Das wird auch in Bild 7 an der Platzierung der Polizei-Signatur deutlich: Der Mittelpunkt eines Rechtecks, das ein Gebäudepolygon umschließt, führt häufig zu unbefriedigenden Symbolplatzierungen. Noch weniger zufriedenstellend ist die Signaturierung von Kirchen und Kapellen: Bei fehlenden Präsentationsobjekten ist zum einen wieder die Position der Symbolsignatur – eines Kreuzes – nicht optimal, und zum anderen fehlt der jeweilige Drehwinkel, ohne dessen Angabe die Kreuze stets in Nord-Südrichtung ausgerichtet werden. Bild 8 illustriert die Situation: Links ist die Symbolplatzierung dargestellt, die sich ohne ein Präsentationsobjekt ergibt, und rechts wird die wünschenswerte Symbolplatzierung gezeigt. Zur allgemeinen Verbesserung der Symbolplatzierung planen wir daher eine Erweiterung unseres Systems, die bei fehlenden Präsentationsobjekten diese automatisch generiert und in den jeweiligen Bestandsdatenauszug einfügt. Dabei sollen Algorithmen zur optimalen Platzierung realisiert werden. Nachdem die – bislang fehlenden – Präsentationsobjekte ergänzt worden sind, könnten die so angereicherten Bestandsdatenauszüge wie bisher durch unser System visualisiert werden.

Literatur

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV): Erläuterungen zu ALKIS (Version 4.0). http://www.adv-online.de/, 2005a.

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV): ALKIS-Objektartenkatalog (Version 4.0). http://www.adv-online.de/, 2005b.

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV): ALKIS-Signaturenkatalog (Version 4.0): Vorbemerkungen (Teil A), Signaturenbibliothek (Teil B), Präsentationen (Teil C). http://www.adv-online.de/, 2005c.

Grutza, M.: Umsetzung der Signaturenbibliothek des Amtlichen Liegenschaftskataster-Systems nach XML. Studienarbeit, TU Braunschweig, 2005.

Landesamt für Vermessung und Geobasisinformation Rheinland-Pfalz: ALKIS-Beispieldaten (NAS-Testdaten). http://www.lvermgeo.rlp.de/, 2006.

Landesvermessungsamt Baden-Württemberg: ALKIS-Beispieldaten (NAS-Testdaten). http://www.lv-bw.de/, 2006.

Mathiak, B.; Kupfer, A.; Neumann, K.: Modellierung und kartographische Visualisierung von Geodaten mit XML-basierten Sprachen. In Informatik – Forschung und Entwicklung, Band 20, Heft 1–2, pp. 24–32, 2005.

Melching, I.: Optimierung XML-basierter Ableitungsregeln für prototypische Liegenschaftskarten. Studienarbeit, TU Braunschweig, 2006.

Niedersächsische Vermessungs- und Katasterverwaltung: ALKIS-Beispieldaten (NAS-Testdaten). http://www.gll.niedersachsen.de/, 2006.

Neumann, K.; Kupfer, A.; Mathiak, B.: Umsetzung des Signaturenkataloges SK25 bei der XML-basierten Erzeugung kartenähnlicher Graphiken. In Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 34, Frankfurt M. 2005, pp. 107–118.

Neumann, K.; Petri, J.; Wolf, C.: Erzeugung kartenähnlicher Graphiken: XML-basierte Verdrängung und Platzierung von Punktsignaturen. In Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 36, Frankfurt M. 2006, pp. 89–98.

Nordmann, T.: Implementierung von XML-basierten Ableitungsregeln zur Generierung von Liegenschaftskarten. Diplomarbeit, TU Braunschweig, 2006.

Schlutow, F.: Abbildung einer XML-basierten kartographischen Signaturenbibliothek auf SVG-Befehle. Studienarbeit, TU Braunschweig, 2006.

Wolf, C.: Transformation der Ableitungsregeln des Amtlichen Liegenschaftskataster- Systems in XML-Syntaxbäume. Diplomarbeit, TU Braunschweig, 2005.