XML-basierte Kodierung von Geodaten mittels der Geography Markup Language

Franz-Josef Behr, Hochschule für Technik, Stuttgart

Zusammenfassung

Im Umfeld der Geoinformatik sind unterschiedliche Datenformate für den Austausch von Geodaten seit langem von großer Bedeutung für die Praxis. Neben den dort eingesetzten offiziellen und De-facto-Standards spielen im Bereich der Informationstechnologie im allgemeinen sowie zunehmend in der Geoinformatik XML-basierte Verfahren des Datenaustausches und der Kommunikation eine bedeutende Rolle. Dabei kommt der Geography Markup Language (GML) als umfassendes Format und künftiger ISO-Standard für die Modellierung und Übertragung von Geodaten besondere Bedeutung zu.

Dieser Beitrag behandelt die notwendigen Grundlagen der eXtensible Mark-up Language (XML) sowie relevanter Standards im XML-Umfeld. Darauf aufbauend werden die Möglichkeit der Datendefinition mittels GML aufgezeigt. Der Schwerpunkt liegt dabei auf den Prinzipien der Deklaration von Elementen, der Definition von Datentypen sowie von Geo-Objekten (Features).

1 Einleitung

Bei der eXtensible Mark-up Language (XML) handelt es sich um eine *Auszeichnungssprache* (engl. markup language), die es als Metasprache erlaubt, weitere Sprachen zu definieren, zu denen auch die Geography Markup Language (GML) zählt. Derartige Sprachen und Konzepte haben in den vergangenen Jahren rasante Verbreitung im Umfeld der Geoinformatik und der Informationsverarbeitung insgesamt gefunden. Wesentliche Erfolgsfaktoren für XML sind zum einen die textbasierte Kodierung der Information (d. h. keine Binärdaten), zum anderen der für den Nutzer zunächst leichte Zugang zur Interpretation der Dateninhalte[1].

GML wurde als XML-basierte Sprache mit der Zielsetzung entwickelt, Modellierung, Transport und Speicherung von Objekten mit Raumbezug zu ermöglichen (Cox 2004, Lake 2004). Dabei sollten Geometrie- wie Fachattribute gleichermaßen unterstützt werden. Nach der ersten Version im Jahr 2000 wurden alle ein bis zwei Jahre neue Versionen publiziert (siehe Abbildung 1). Die Fortschreibungen dieser Empfehlung sind folgendermaßen charakterisiert:

- Nutzung jeweils aktueller XML-Standards,
- stetige Vervollkommnung bezüglich der zu modellierenden Geometrien und Objektzusammenhänge,
- zunehmende Komplexität (vgl. Seitenanzahl in Abbildung 1).

Gerade die Mächtigkeit der Sprache scheint einer breiten Nutzung bisher entgegenzustehen. Durch Profile besteht jedoch die Möglichkeit, durch die Bildung einer Untermenge von möglichen GML-

[1] Als Beispiel für die Lesbarkeit können ALKIS-Beispieldaten herangezogen werden, die unter http://www.lv-bw.de/lvshop2/produktinfo/wir-ueber-uns/links/advprojekt/ALKIS-Beispieldaten.zip zum Download angeboten werden.

Konstrukten die Komplexität und notwendige Einarbeitung zu verringern (Vretanos 2005).

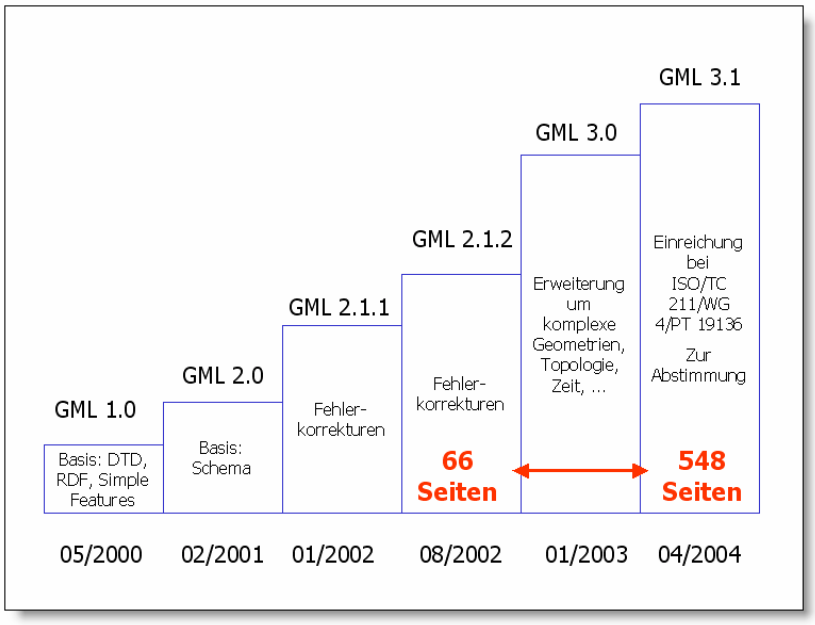

Abbildung 1: Entwicklung von GML hin zum ISO-Standard (nach Strobel 2004).

2 XML – die Grundlage

Wesentliche Stuktureinheit zur Gliederung von Dateninhalten in XML ist das *Element*. Es besitzt einen Elementnamen und weist den in Abbildung 2 dargestellten Aufbau auf, besteht also aus einem einleitenden Start-Tag[2], dem Elementinhalt und dem schließenden End-Tag. Der Elementname taucht in beiden Tags auf, beim End-Tag mit vorangestelltem Schrägstrich.

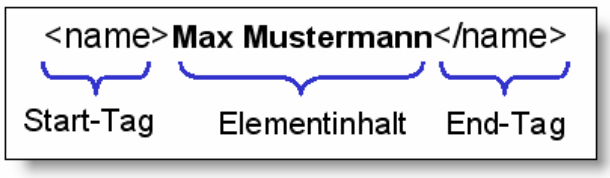

Abbildung 2: Aufbau eines XML-Elements.

Für den Elementinhalt gilt, dass er Text (wie in Abbildung 2) oder, wie in GML häufig der Fall, wiederum andere Elemente enthalten kann (siehe nachfolgendes `punkt`-Beispiel); er kann aber auch leer sein und dann in verkürzter Schreibweise in der Form

```
<name />
```

notiert werden. Textinhalte unterliegen einer Einschränkung: Sie dürfen nicht Zeichen enthalten, die für das Markup, d. h. die Gliederung des XML-Dokuments, vorbehalten sind, wie < oder &. Diese müssen maskiert als *Entitäten* (engl. entities) kodiert werden (z. B. `<` anstelle von <).

[2] tag (engl.): Schildchen, Etikett

Elementnamen sind weitgehend frei wählbar und können beispielsweise auch deutsche Sonderzeichen enthalten.

Elemente können neben ihrem Inhalt weitere Informationen in Form von Attributen tragen, die im Start-Tag eingetragen werden:

```
<text x="15" y="135">AbcDef</text>
```

Von dieser Möglichkeit wird insbesondere bei der XML-definierten Sprache SVG sowie bei der Entwicklung von GML-Fachschemata Gebrauch gemacht. Das Start-Tag des `text`-Elements zeigt, dass Attributangaben paarweise aus Attributnamen und Wert bestehen. Der Wert muss in einfachen oder doppelten Anführungszeichen stehen; jedes Attribut darf in einem Element nur ein Mal erscheinen.

Aus der Tatsache, dass ein Element andere Elemente enthalten kann, ergibt sich die Forderung, dass diese „Schachtelung" korrekt vorgenommen werden muss[3].

Die äußerste „Schachtel" – das Element, das alle anderen umschließt – heißt *Wurzelelement* (engl. root element).

Folgt ein XML-Dokument diesen Regeln, so wird es als *wohlgeformt* (engl. well formed) bezeichnet.

Jedes XML-Dokument besteht aus:

- einem Prolog, der XML-Deklaration,
- einem Abschnitt oder Verweis auf eine Dokumententyp-Deklaration oder ein XML-Schema, in denen die nachfolgenden Elemente deklariert werden,
- einem Wurzelelement und darin hierarchisch eingebetteten, weiteren Elementen,
- Verarbeitungsanweisungen (optional),
- Kommentaren (optional),
- optionalen `CDATA`-Abschnitten
- optionalen Entitäten, d. h. Kürzel oder Ersetzungszeichen für Einzelzeichen oder Zeichenketten.

Im nachfolgenden Beispiel spezifiziert die XML-Deklaration in Zeile 1 die zugrundegelegte XML-Version, den für das Kodieren verwendete Zeichensatz (hier ISO-8859-1) und legt fest, dass das Dokument „standalone", ohne weitere Deklarationen, gültig ist.

```
 1: <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
 2: <!DOCTYPE punkt [
 3: <!ELEMENT punkt (x, y)>
 4: <!ELEMENT x (#PCDATA)>
 5: <!ELEMENT y (#PCDATA)>
 6: ]>
 7: <punkt>
 8:  <x>3500000.0</x>
 9:  <y>5400000.0</y>
10: </punkt>
```

In den Zeilen 2 bis 6 erfolgt die Dokumenttyp-Deklaration (engl. document type declaration), die den Aufbau des Dokumentes vorgibt. In unserem einfachen Fall beschreibt sie in den Zeile 2 – 6 die

[3] Eine reale Schachtel kann nur in *einer* anderen enthalten sein, und Schachteln können sich nicht überlappen.

Zugehörigkeit zum *Wurzelelement* `punkt`, das als Inhalt die Elemente `x` und `y` enthält. Diese bestehen aus Zeichendaten[4].

Die Zeilen 7 bis 10 enthalten ein `punkt`-Element, das zugleich Wurzelelement ist. Elementinhalt sind, wie in Zeile 3 deklariert, die `x`- und `y`-Elemente mit den Koordinatenwerten.

Verarbeitungsanweisungen (im Beispiel nicht enthalten) sind für weiter verarbeitende Programme vorgesehen, beispielsweise zur Berücksichtigung von Stilangaben oder zur Durchführung einer XSL-Transformation. In *`CDATA`-Abschnitten* können beispielsweise ECMAScript-Anweisungen in das XML-Dokument integriert werden, ohne dass sie von einem *XML-Parser* ausgewertet werden, der das Prüfen, Zerlegen und Weiterverarbeiten der Inhalte vornimmt.

Wichtiger Bestandteil von Programmier- und Auszeichnungssprachen sind Kommentare, eine Dokumentbeschreibung im Source-Code, die vom Parser ignoriert wird. Für XML-Sprachen sieht ein *Kommentar* folgendermaßen aus:

```
<!-- Dies ist ein Kommentar -->
```

Zwischen den Kommentarzeichen `<!--` und `-->` können alle beliebigen Zeichen stehen, mit Ausnahme von `--`, da dies das Ende eines Kommentars markiert.

2.1 Die XML-Sprachfamilie

Unter dem Oberbegriff XML verbergen sich eine große Reihe unterschiedlicher Sprachen und Konzepte, die weitgehend vom World Wide Web Consortium (W3C, http://w3.org/) koordiniert und verantwortet werden. Im Zusammenhang mit GML sind unter anderem zu nennen:

- XML-Namensräume,
- Dokumententyp-Deklaration und XML Schema,
- XLink – XML Linking Language und XML Pointer,
- CSS2 – Cascading Stylesheets,
- Scalable Vector Graphics (SVG),
- Extensible Stylesheet Language Family mit XSL Transformations, XML Path Language und XSL Formatting Objects.

Weitere XML-Technologien finden sich in unterschiedlichsten Anwendungsbereichen. Für die Kodierung mathematischer Formeln beispielsweise steht MathML zur Verfügung (http://www.w3.org/Math/). Selbst im Musikbereich eröffnen sich neue Möglichkeiten des Datenaustausch zwischen Programmen dank eines XML-basierten Formats (http://www.capella.de/).

Im Umfeld der Geoinformatik bildet, wie bereits erwähnt, XML die Grundlage der *Geography Markup Language* sowie der darauf aufbauenden künftigen deutschen Normbasierten Austauschschnittstelle (NAS) der AdV (Düren 2005, http://www.adv-online.de/). Weitere Einsatzgebiete sind Kommunikationsschnittstellen bei Web-Diensten im Umfeld des Internet-Mapping wie dem Web Map Service (OGC 2004), dem Web Feature Service (OGC 2002), dem Web Coordinate Transformation Service (Whiteside et al. 2005) sowie bei Internet-GIS-Servern.

[4] Das Schlüsselwort #PCDATA leitet sich historisch von dem Ausdruck »parsed character data« ab: Zeichendaten, die von einem XML-Programm (Parser) weiter zerlegt werden.

2.2 XML-Namensräume

Wie zuvor erwähnt besteht bei der Wahl der Elementnamen große Freiheit, so dass die mehrfache Nutzung identischer Namen nicht ausgeschlossen werden kann. Beispielsweise kann ein `titel`-Element im Umfeld einer Personenverwaltung und einer CD-Datensammlung auftauchen!

XML-Namensräume (engl. namespaces) bieten eine einfache Möglichkeit, um Element- und Attributnamen in XML-Dokumenten eindeutig zu benennen: Sie werden mit Namensräumen verknüpft, die durch URI-Verweise[5] identifiziert werden. Durch die Angabe und Nutzung von Kürzeln für Namensräume können Elemente aus mehreren Schemata gemeinsam benutzt werden.

Im folgenden Beispiel werden für das `TKFD`-Wurzelelement die Namensräume `tkfd`, `gml` und `xsi` deklariert. Aus dem Namensraum `tkfd`, der über den URI-Verweis `http://www.lv-bw.de/tkfd` identifiziert wird, werden die Elemente `Bahnhof` und `objektArt` sowie das Attribut `id` verwendet. Aus dem `gml`-Namensraum werden die Elemente `centerOf`, `Point` und `pos` verwendet.

```
<?xml version="1.0" encoding="UTF-8"?>
<TKFD xmlns:tkfd="http://www.lv-bw.de/tkfd"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.lv-bw.de/tkfd">
<tkfd:Bahnhof>
  <tkfd:objektArt tkfd:id="EZ00VPK">9201</tkfd:objektArt>
  <gml:centerOf>
    <gml:Point>
      <gml:pos>3515955.37 5409276.28</gml:pos>
    </gml:Point>
  </gml:centerOf>
</tkfd:Bahnhof>
```

2.3 Dokumententyp-Deklaration und XML Schema

Die Dokumententyp-Deklaration (document type declaration, DTD) erlaubt es, vorkommende Elemente, ihre Attribute und ihren hierarchischen Aufbau zu definieren. Traditionell u. a. im Bereich von HTML und SVG sowie bei GML 1.0 angewandt wird diese Form der Elementdeklaration heutzutage als unzureichend angesehen und zunehmend durch die Nutzung von XML-Schema abgelöst.

Ein XML-Schema (http://www.w3.org/XML/Schema) erlaubt es, detailliert und exakt eine Klasse von XML-Dokumenten zu spezifizieren (vgl. Abschnitt 3). Im Vergleich zu DTD wächst die Komplexität, u. a. auch wegen der Möglichkeit der Vererbung. Für ein auf einem Schema basierendes konkretes „XML-Objekt“ wird der Begriff „Instanzdokument“ oder kurz *Instanz* verwendet. Eine XML-Instanz kann als XML-Datei existieren, kann aber genauso gut ein Datenstrom oder Feldinhalt in einem Datenbanksatz sein. Schemadefinitionen, die auch in mehreren Dokumenten verteilt vorliegen können, spielen bei GML sowie bei NAS eine bedeutende Rolle.

Ist ein XML-Dokument wohlgeformt und folgt es einer als DTD oder Schema vorgegebenen Deklaration, heißt es *gültig*.

[5] URI: Abkürzung für Unified Resource Identifier zur eindeutigen Identifizierung einer Internet-Ressource.

2.4 XML Linking Language (XLink) und XML Pointer Language (XPointer)

Die XML Linking Sprache (*XLink*, http://www.w3.org/XML/Linking) erlaubt es, Elemente in XML-Dokumente einzufügen sowie Verknüpfungen zwischen Ressourcen herzustellen. XLink geht damit über das Verlinken von Informationen, wie es in HTML verwendet wird, wesentlich hinaus. In SVG-Dokumenten gestattet es XLink, separat definierte Symbole, wie z. B. kartographische Signaturen oder Rasterdaten, an beliebiger Stelle zu referenzieren und in eine Karte zu integrieren.

Ein Schema-Dokument *xlinks.xsd* ist Teil der GML 3 zugrunde liegenden Schemadefinitionen. Elemente und Datentypen können somit durch Verweise auf andere Deklarationen beschrieben werden (vgl. Beispiele in Abschnitt 3).

Unterstützt wird XLink durch XPointer (http://www.w3.org/TR/xptr-framework/) zur gezielten Adressierung von Dokumentteilen (Fragmenten). Dabei leitet das Zeichen # (wie in nachfolgenden Beispielen teilweise verwendet) den Namen des Fragments ein.

2.5 CSS2 - Cascading Stylesheets

Kaskadierende Stilangaben (*Cascading Style Sheets*, CSS, http://www.w3.org/Style/CSS/) spezifizieren auf einfache Weise die Darstellung von XML-Dokumenten auf Ausgabegeräten unterschiedlicher Art (Bildschirm, Drucker, ...). Mit der Trennung von Präsentation und Inhalt wird ein wesentliches Konzept der raumbezogenen Informationsverarbeitung umgesetzt. In SVG können z. B. mittels CSS Darstellungsstile für einzelne Layer zentral definiert werden. Auch GML sieht die Möglichkeit vor, Stilvorgaben für die Visualisierung von Geo-Daten zu spezifizieren.

2.6 SVG

SVG als XML-basierte, textorientierte Auszeichnungssprache erlaubt es, zweidimensionale, skalierbare Grafiken zu beschreiben, die Vektorelemente, Rasterdaten und Text als grafische Objekte in ein XML-Dokument integrieren. Die Darstellung erfolgt durch sogenannte *User Agents, d. h.* Browser, Browser-Plugins oder eigenständige SVG-Viewer. Eine Übersicht – leider nicht immer aktualisiert – gibt http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8#viewer.

Als Vektorgrafikformat hat SVG viele Vorteile gegenüber den herkömmlichen, im Web verwendeten Grafikformaten wie GIF, JPG und PNG (Watt 2001, Fibinger 2001):

- Nicht-proprietäres, offengelegtes Format,
- Hochauflösende Grafiken sind möglich, die auch bei Skalierung nicht an Qualität verlieren.
- SVG unterstützt eine hohe Farbtiefe.
- Animationen sind möglich, die im Gegensatz zu animierten GIF-Grafiken nicht zu größeren Dateien führen.
- SVG-Grafiken untersützen das Document Object Model (DOM, W3C 2005). Dies ermöglicht es, SVG-Elemente per ECMAScript oder einer anderen objektorientierten Programmier-sprache zu manipulieren.
- Filtereffekte, wie z. B. Schatten, sowie Farbverläufe sind möglich.

Aufgrund dieser besonderen Eigenschaften hat SVG insbesondere im Bereich der Geoinformatik und Kartographie sehr hohe Bedeutung gewonnen. So ist SVG beispielsweise für den Web Map Service als Ausgabeformat vorgesehen (Kettemann 2005a, 2005b). Einige GI-Systeme und -Werkzeuge bieten direkte SVG-Erzeugung, für andere stehen Konvertierungswerkzeuge als Zusatz zur Verfügung (Behr 2005). Ebenfalls existieren Ansätze für die Online-Digitalisierung von Geodaten (Neumann 2004). Am Landesvermessungsamt Baden-Württemberg wird SVG für die Herstellung analoger touristischer Karten genutzt (Graf 2004).

Bezüglich GML ist die Erzeugung von SVG mittels XSL für die Visualisierung von besonderer Bedeutung.

2.7 Extensible Stylesheet Language

Mit der *Extensible Stylesheet Language Family* (XSL, http://www.w3.org/Style/XSL/) können XML-Dokumente in andere XML-Dokumente oder in Datenströme transformiert werden können. Zur XSL-Sprachfamilie gehören XSL Formatting Objects (XSL-FO, hier nicht weiter diskutiert), XSLT und die XML Path Language (XPath).

XSL Transformations (XSLT) dienen der Überführung von Inhalt und Struktur eines XML-Dokuments in eine andere Struktur. Wichtig für den Bereich der Geoinformatik ist die Transformation von GML-Instanzen in SVG-Dokumente (Abbildung 3), ein Ansatz, der z. B. von der britischen Ordnance Survey zur Visualisierung von Katasterdaten verwendet wird (Ordnance Survey 2005). Eine praktische Umsetzung zeigt Barth (2004) mit der Überführung bayerischer DFK-Daten nach SVG mittels XSLT.

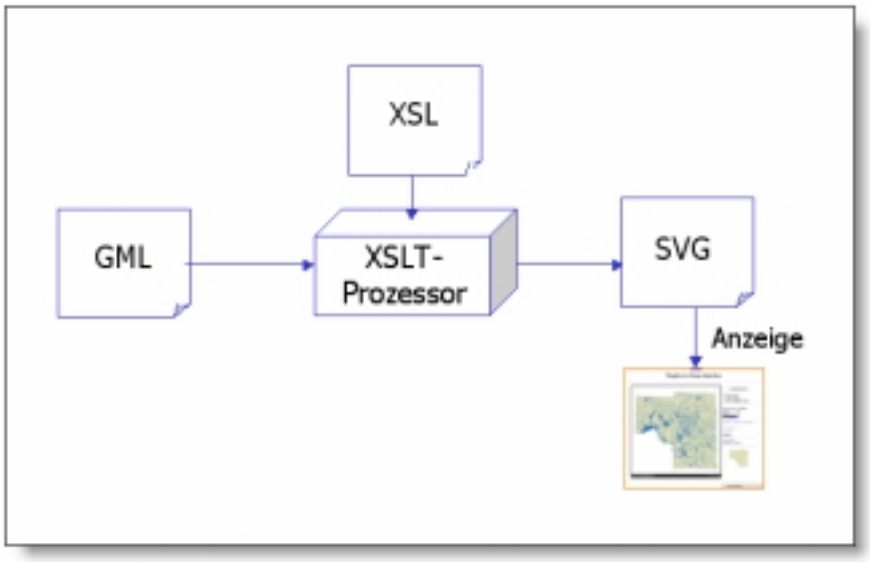

Abbildung 3: Prinzip der Visualisierung von GML mittels XSL-Transformation. GML-Daten werden durch in XSL formulierte Vorgaben nach SVG transformiert und zur Anzeige gebracht.

Mit *XPath* (http://www.w3.org/TR/xpath) werden bei der Transformation Teile eines XML-Dokuments adressiert (W3C 2002). Für diese Selektion wird keine XML-Syntax verwendet, sondern eine Art Pfadangabe. So wird beispielsweise aus dem `tkfd:Bahnhof`-Element des zuvor gezeigten XML-Beispiel mittels

```
<xsl:value-of select="substring-
before(gml:centerOf/gml:Point/gml:pos,' ')"/>
```

der Rechtswert selektiert.

3 Grundlagen der GML-Schemadefinition

Die grundlegenden Elemente und Datentypen von GML 3 sind in mehr als 30 XML-Schema-Dokumenten (Basisschemata) definiert. Für eine konkrete Anwendung muss davon ein anwendungsspezifisches Schema abgeleitet werden, d. h. eine formale Beschreibung aller für das Fachgebiet oder eine bestimmte Aufgabe relevanten Inhalte und Zusammenhänge erstellt werden (Düren / Riekert 2005). In einem solchen Anwendungsschema können, wie nachfolgend gezeigt, für eigene Geo-Objekte (Features) entsprechende Elemente und Eigenschaften definiert werden. Ein Beispiel für eine derartige Modellierung ist das AFIS-ALKIS-ATKIS-Anwendungsschema (Seifert 2005). Einen Leitfaden für die Modellierung von Fachinformationen im Kontext von ALKIS (vgl. Constantin 2005) bietet das AFIS-ALKIS-ATKIS-Koordinierungsgremium (AdV 2004); die Ausführungen des vorliegenden Beitrags beschränken sich jedoch auf GML.

Was verbirgt sich hinter einer Schemadefinition? Unterhalb des `schema`-Wurzelelements, das als Attribut den vorgesehenen Namensraum enthält (`targetNamespace`), wird die Struktur von Geo-Objekten in Form von Elementen deklariert. Dabei können vordefinierte, einfache Datentypen oder komplexe, selbst definierte Datentypen verwendet werden, Konzepte, wie sie seit Jahren bei objektorientierten Programmiersprachen bekannt sind.

3.1 Datentypen

Zu den einfachen, atomaren Datentypen in XML-Schema gehören unter anderem Zeichenketten (`string`), ganze und reelle Zahlen, Datums- und Zeitangaben (http://www.w3.org/TR/xmlschema-0/#CreatDt). Das folgende Element wird zum Beispiel mit dem Namen `RW_OL` deklariert und ist vom Datentyp `double`.

```
<xs:element name="RW_UR" type="xsd:double"/>
```

Durch *Facetten* können Wertebereiche einfacher Datentypen eingeschränkt werden („`restriction`“). Für den selbstdefinierten, ganzzahligen Typ `objektArtType` sind beispielsweise nur Werte 9401 – 9403 zulässig:

```
<xs:simpleType name="objektArtType">
  <xs:restriction base="xs:integer">
    <xs:enumeration value="9401"/>
    <xs:enumeration value="9402"/>
    <xs:enumeration value="9403"/>
  </xs:restriction>
</xs:simpleType>
```

Der Datentyp `geometryType` ist eigentlich vom Typ `string`, darf aber nur die drei definierten Werte annehmen:

```
<xs:simpleType name="geometryType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Punkt"/>
    <xs:enumeration value="Linie"/>
    <xs:enumeration value="Flaeche"/>
  </xs:restriction>
</xs:simpleType>
```

Aus einfachen Typen können komplexe Typen zusammengestellt werden, die ebenfalls wieder in anderen, komplexen Datentypen enthalten sein könnten. Der Datentyp `BlattschnittType`

beispielweise verfügt über vier Eigenschaften, die die linke untere sowie die rechte obere Ecke eines Kartenblatts beschreiben; das fünfte Attribut `FK_Name` ist für die Kartenblattbezeichnung vorgesehen.

```
<xs:complexType name="BlattschnittType">
  <xs:sequence>
    <xs:element name="RW_UL" type="xsd:double"/>
    <xs:element name="HW_UL" type="xsd:double"/>
    <xs:element name="RW_OR" type="xsd:double"/>
    <xs:element name="HW_OR" type="xsd:double"/>
    <xs:element name="FK_Name" type="xsd:string"/>
  </xs:sequence>
</xs:complexType>
```

3.2 Abgeleitete Typen

Eigene Typen können von vordefinierten Typen durch Erweiterung oder sogenannte Äquivalenzklassen abgeleitet werden, eine Vorgehensweise, die bei GML sehr häufig zum Einsatz kommt.

Im folgenden Beispiel erweitert („`extension`“) der Datentyp `BahnhofType` den abstrakten Typ `AbstractFeatureType` des `gml`-Namensraums um die Eigenschaften `tkn` und `centerOf`.

```
<xs:complexType name="BahnhofType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="tkn" type="xs:string">
        <xs:element ref="gml:centerOf"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Bei Äquivalenzklassen können Elemente der im `substitutionGroup`-Attribut spezifizierten Gruppe im abgeleiteten Typ ersetzt werden. Dies entspricht der Polymorphie-Eigenschaft objektorientierter Programmiersprachen (Kolbe 2004). Im nachfolgenden Beispiel wird ein Element `TouristischeRouten` vom Typ `TouristischeRoutenType` deklariert. Somit kann das Element aufgrund der `substitutionGroup`-Angabe immer auch dort auftreten, wo `_FeatureCollection` genutzt werden kann.

```
<xs:element name="TouristischeRouten"
type="tkfd:TouristischeRoutenType"
substitutionGroup="gml:_FeatureCollection"/>
```

3.3 Elemente

Für die Deklaration von Elementen sind drei Varianten möglich (vgl. Kolbe 2004):

- Deklaration durch Angabe eines Datentyps, der gesondert definiert werden muss:

```
<xs:element name="SportFreizeit" type="tkfd:SportFreizeitType" />
```

- Deklaration durch Verweis auf ein globales Element, das an anderer Stelle deklariert sein muss:

```
<xs:element ref="tkfd:Bahnhof" />
```

- Deklaration durch Definition eines sogenannten anonymen Typs, der nur lokal innerhalb dieses Elements zur Verfügung steht:

```
<xs:element name="Blattschnitt">
    <xsd:sequence>
      <xsd:element name="RW_OL" type="xsd:double"/>
      <xsd:element name="HW_OL" type="xsd:double"/>
      <xsd:element name="RW_UR" type="xsd:double"/>
      <xsd:element name="HW_UR" type="xsd:double"/>
      <xsd:element name="FK_Name" type="xsd:string"/>
  </xsd:sequence>
</element>
```

Mittels `sequence` deklarierte Inhaltselemente müssen in der vorgegebenen Reihenfolge im Instanzdokument vorkommen.

Daneben stehen noch zwei weitere Methoden zur Verfügung: Bei einer Deklaration mittels `choice` muss genau eine der Alternativen im Instanzdokument auftreten. Mittels `group` werden Gruppen deklariert, die über ihren Namen in Deklarationen eingefügt werden können.

Die Häufigkeit des Auftretens eines Elements kann durch die beiden Attribute `minOccurs` und `maxOccurs` festgelegt werden:

```
<xs:element ref="tkfd:Bahnhof" minOccurs="0" maxOccurs="unbounded"/>
```

3.4 Ein Beispiel

Die Struktur eines Features `BadeseeBadeplatz` wird mittels XML-Schema festgelegt. Es ergibt sich folgende Schemadefinition:

```
<xs:element name="BadeseeBadeplatz" type="tkfd:BadeseeBadeplatzType"
substitutionGroup="gml:_Feature"/>

<xs:complexType name="BadeseeBadeplatzType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="objektArt" type="tkfd:objektArtType">
        </xs:element>
        <xs:element name="tkn" type="xs:string">
        </xs:element>
        <xs:element ref="gml:centerOf"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Diese Definition umfasst zwei Teile:

- die Elementdeklaration, die den Namen (`BadeseeBadeplatz`) und den Typ des Elements (`BadeseeBadeplatzType`) beinhaltet;
- die Typdefinition (auch Inhaltsmodell genannt), die den inneren Aufbau eines `BadeseeBadeplatz`-Features beschreibt.

In der Elementdeklaration wird durch den Wert `gml:_Feature` des Attributs `substitutionGroup` deutlich gemacht, dass es sich um die Deklaration eines Features handelt. Mit anderen Worten: Unser Datentyp `BadeseeBadeplatzType` leitet sich vom abstrakten Datentyp `AbstractFeatureType` ab, der allgemeine Eigenschaften für Features in GML beinhaltet. `gml:_Feature` ist ein Element dieses Datentyps.

4 Definition von Objekten in GML

4.1 Geo-Objekte und Eigenschaften

Kern des GML-Ansatzes sind *Geo-Objekte* (*Features*) mit ihren geometrischen und nicht-geometrischen Eigenschaften; in ihnen werden die Geodaten abgelegt. Features dienen der Repräsentation von Objekten der realen Welt, wie z. B. Straßen, Gebäuden, Flüssen, Vermessungspunkten, jeweils bezogen auf einen bestimmten Anwendungskontext (OGC 2002).

Im folgenden Beispiel wird die Instanz eines Geo-Objekts `Radweg` in GML modellliert:

```
<Radweg gml:id="EZ02LLB"> ... </Radweg>
```

Der Instanz ist das Attribut `id` zugeordnet, das dem `gml`-Namensraum angehört. Eine solche eindeutige Kennung ist innerhalb eines GML-Dokuments für jede Objektinstanz zwingend notwendig.

Eigenschaften werden zu Kindelemente, die die Objektinstanz näher beschreiben. Für unseren Radweg werden beispielweise Angaben zu `objektArt`, `kategorie` und `kartenBlattNummer` spezifiziert:

```
<Radweg gml:id="EZ02LLB">
  <objektArt>9102</objektArt>
  <kategorie>1470</kategorie>
  <kartenBlattNummer>l7120</kartenBlattNummer>
</Radweg>
```

Während die Namen von Features mit Großbuchstaben anfangen, beginnen die Namen von Eigenschaftselemente üblicherweise mit einem Kleinbuchstaben. Nachfolgende Wortbestandteile werden jeweils mit einem Großbuchstaben begonnen.

Eigenschaftswerte können, wie im Beispiel gezeigt, innerhalb der Objektinstanz deklariert werden. Unter Nutzung von XLink können Eigenschaften auch an anderer Stelle in diesem Dokument oder in einem beliebig anderen, über einen URI identifizierbaren Dokument referenziert werden:

```
<objektArt xlink:href="http://www.anyserver.de/objArt.xml#9401" />
```

Das `objektArt`-Element wird hierbei zu einem leeren XML-Element; der Verweis wird als Attribut dem Start-Tag beigefügt. Die eigentliche Information findet sich auf dem Server `www.anyserver.de` im Fragment `9401` der Datei `objArt.xml`.

4.2 Beziehungen zwischen Features

Features können Eigenschaften besitzen, die wiederum Features sind. Liegt ein Badesee an einem Wanderweg, kann diese Beziehung beispielsweise folgendermaßen notiert werden:

```
<BadeseeBadeplatz>
  <objektArt>9401</objektArt>
```

```
  <liegtAn>
    <tkfd:WanderWeg gml:id="wanderweg001" />
  </liegtAn>
</BadeseeBadeplatz>
```

Als Verweis auf ein referenziertes Feature (hier im selben Dokument) wird die Beziehung folgendermaßen beschrieben:

```
<BadeseeBadeplatz>
  <liegtAn xlink:href="#wanderweg001" />
</BadeseeBadeplatz>
```

4.3 Geometrische Eigenschaften

Während in GML 2 nur wenige Grundformen wie Punkt, Linie, Fläche und Multipolygon vorgesehen waren, verfügt die aktuelle Version GML 3 über eine deutlich erweiterte Vielzahl geometrischer Elemente. Dazu zählen (Kolbe 2004):

- Linienhafte Geometrien mit verschiedenen Interpolationsverfahren wie linear, kreisbogenförmig, splineförmig, als Klothoide usw., die auch gerichtet sein können,
- flächenhafte Geometrien, auch gerichtet, mit verschiedenen Interpolationsverfahren (planar, sphärisch, ellipsoidal, ...),
- volumenhafte Geometrien.

Dazu kommen noch komplexe Geometrien, räumliche und zeitliche Referenzsysteme, Topologie, Maßeinheiten, Metadaten, Coverages; Beobachtungen (Observations) und an SVG angelehnte Stilvorgaben (Default Styles) für die Visualisierung der Geodaten.

Features dürfen beliebig viele geometrische Eigenschaften besitzen. Jede geometrische Eigenschaft wird in einem `geometryProperty`-Kindelement eingeschlossen, das den Datentyp bzw. die Rolle des Geometrie-Objekts (z.B. `centerLineOf`, `curveProperty`, `surfaceProperty`) bezeichnet

Kindelement dieses Geometrie-Eigenschaftselements sind ein oder mehrere Geometrie-Objekte (z.B. `Point`, `LineString`, `Polygon` usw.). Im ersten Beispiel wird der Verlauf eines linienförmigen Objekts durch fünf Stützpunktkoordinaten festgelegt:

```
<gml:centerLineOf>
  <gml:LineString>
    <gml:pos>3512126.69 5411713.94</gml:pos>
    <gml:pos>3512219.35 5411712.56</gml:pos>
    <gml:pos>3512462.85 5411720.61</gml:pos>
    <gml:pos>3512627.57 5411729.57</gml:pos>
    <gml:pos>3512748.51 5411740.24</gml:pos>
  </gml:LineString>
</gml:centerLineOf>
```

Neben dem `pos`-Element können Koordinatenangabe beispielsweise auch über `gml:coordinates` erfolgen. Das zweite Beispiel zeigt die lagemäßige Beschreibung eines punktförmigen Features:

```
<gml:centerOf>
  <gml:Point>
    <gml:pos>3512280.93 5410246.16</gml:pos>
  </gml:Point>
```

```
</gml:centerOf>
```

Auch für geometrische Eigenschaften gilt, dass der Elementinhalt als Referenz auf eine anderes XML-Element definiert werden kann:

```
<gml:centerLineOf xlink:href=¬
"#0000002CFV9019101R00000002CFU9019101R0"></gml:centerLineOf>
```

4.4 GML und räumliche Bezugssysteme

Zur Geometrieinformation muss das räumliche Bezugssystem (engl. spatial reference system, SRS, bzw. coordinate reference system, CRS, vgl. Whiteside 2005, Lake 2004:192ff) benannt werden, in dem die Koordinaten vorliegen. Bei zusammengesetzten Geometrien reicht es, wenn das Bezugssystem bei der Angabe des umschließenden Rechtecks (`boundedBy` bzw. `Envelope`) benannt ist (Kolbe 2004). Für die Referenzierung eines Bezugssystems wird durch Angabe eines URI auf die Klassifikation geodätischer Bezugssysteme der European Petrol Survey Group EPSG (http://www.epsg.org) zurückgegriffen. Jedem Bezugssystem entspricht darin eine eindeutige Codenummer (z. B. 31467 für Gauß-Krüger-Projektion, 3. System):

```
<tkfd:Bahnhof gml:id=“b00214“>
  <gml:centerOf>
    <gml:Point
srsName="http://www.opengis.net/gml/srs/epsg.xml#31467">
      <gml:pos>3514623.36 5411115.90</gml:pos>
    </gml:Point>
  </gml:centerOf>
</tkfd:Bahnhof>
```

4.5 Sammlungen von Features

Geo-Objekte können zu Sammlungen von Features (FeatureCollections) zusammen gefasst werden, die Ähnlichkeit mit einem Layer in einem Geo-Informationssystem aufweisen. Eine solche Sammlung kann leer sein oder beliebig viele Mitglieder (FeatureMembers) besitzen.

Eine Sammlung kann selbst eigene räumliche und nichträumliche Eigenschaften besitzen. Sie ist *per definitionem* selbst ein Feature; somit sind auch FeatureCollections von FeatureCollections möglich!

Im nachfolgenden Beispiel sind Features aufgeführt, die Teil einer Sammlung `SportFreizeit` sind:

```
<tkfd:SportFreizeit gml:id="ID000001">
  <gml:featureMember>
    <tkfd:BadeseeBadeplatz gml:id="ID000002">
      ...
    </tkfd:BadeseeBadeplatz>
  </gml:featureMember>
  <gml:featureMember>
    <tkfd:Bootsvermietung gml:id="ID000004">
      ...
    </tkfd:Bootsvermietung>
  </gml:featureMember>
...
```

4.6 Beispiel

Im abschließenden, umfangreicheren Beispiel wird ein Wanderweg modelliert, der sich aus einzelnen Abschnitten zusammensetzt (Abitew 2005). Der Wanderweg besitzt Eigenschaften, die für ihn in seiner Gesamtheit gültig sind. Jeder Wegabschnitt besitzt zusätzliche Eigenschaften, die nur für ihn gelten.

Das Element `WanderWeg` ist vom Typ `WanderWegType` und wird indirekt abgeleitet von der abstrakten Klasse `gml:AbstractFeatureCollectionType`.

```
<element name="WanderWeg"
substitutionGroup="gml:_FeatureCollection">
  <complexType>
    <complexContent>
      <extension base="tkfd:WanderWegType"/>
    </complexContent>
  </complexType>
</element>
```

`AbstractFeatureCollectionType` ist in der Datei features.xsd als abstrakter Typ definiert, der unter anderem die Eigenschaften Name, Beschreibung, `id`-Attribut, das `boundedBy`-Attribut sowie eine beliebige Anzahl von Members vorsieht.

Der komplexe Typ `WanderWegType` sieht über die Eigenschaften des `AbstractFeatureCollectionType` hinaus die Eigenschaften `objektArt`, `kartenBlattNummer` und `WanderWegTeil` vor:

```
<complexType name="WanderWegType">
  <complexContent>
    <extension base="gml:AbstractFeatureCollectionType">
      <sequence>
        <element ref="tkfd:objektArt"/>
        <element name="kartenBlattNummer" type="string">
        </element>
        <element ref="tkfd:WanderWegTeil" minOccurs="1"
maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Ein Geo-Objekt `WanderWeg` besteht also aus mindestens einem Geo-Objekt `WanderWegTeil`. Die Maximalanzahl ist unbegrenzt.

Das Element `WanderWegTeil` wird wie folgt definiert:

```
<element name="WanderWegTeil" substitutionGroup="gml:_Feature">
  <complexType>
    <complexContent>
      <extension base="tkfd:WanderWegTeilType"/>
    </complexContent>
  </complexType>
</element>
```

Der zugehörige Datentyp besitzt folgenden Aufbau:

```
<complexType name="WanderWegTeilType">
```

```
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element ref="gml:centerLineOf"/>
        <element name="beschaffenheitDesWeges" type="tkfd:bwgType">
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Neben den über `AbstractFeatureType` geerbten Eigenschaften wird hier zunächst als Verweis die Geometrieeigenschaft `gml:centerLineOf` definiert. Die weitere Eigenschaft `beschaffenheitDesWeges` ist durch einen Sub-Typ des Datentyps `string` bestimmt, der als zulässige Werte `11110`, `11120`, `1130` und `9998` enthält:

```
<simpleType name="bwgType">
  <restriction base="string">
    <enumeration value="1110 "/>
    <enumeration value="1120 "/>
    <enumeration value="1130 "/>
    <enumeration value="9998 "/>
  </restriction>
</simpleType>
```

In ähnlicher Weise erfolgt die Definition des Elements `objektArt` bzw. seines Typs `tkfd:objektArtType`:

```
<element name=" objektArt" type="tkfd:objektArtType"/>
<simpleType name="objektArtType">
  <restriction base="string">
    <enumeration value="9501"/>
    <enumeration value="9502"/>
    ...
  </restriction>
</simpleType>
```

Die Definition einer GML-Instanz hat schließlich folgenden Aufbau:

```
<tkfd:Wanderweg gml:id="w00001B">
  <tkfd:objektArt>9101</tkfd:objektArt>
  <tkfd:kartenBlattNummer>l7120</tkfd:kartenBlattNummer>
  <gml:featureMember>
    <tkfd:WanderwegTeil gml:id="0000002BZX">
      <gml:centerLineOf">
        <gml:LineString>
           <gml:pos>3516104.74 5409978.21</gml:pos>
           ...
        </gml:LineString>
      </gml:centerLineOf>
      <tkfd:beschaffenheitDesWeges>1110
      </tkfd:beschaffenheitDesWeges>
    </tkfd:WanderwegTeil>
    <tkfd:WanderwegTeil gml:id="0000005WQR">
      <gml:centerLineOf>
        <gml:LineString>
            <gml:pos>3515899.76 5409993.11</gml:pos>
            ...
```

```
          </gml:LineString>
        </gml:centerLineOf>
        <tkfd:beschaffenheitDesWeges>1110
        </tkfd:beschaffenheitDesWeges>
  </tkfd:WanderwegTeil>
</tkfd:Wanderweg>
```

5 Zusammenfassung

XML hat sich weltweit als Standard für den Austausch von Daten und für die Interoperabilität von Anwendungen etabliert. Für Geoinformatik und Kartographie sind XML-basierte Sprachen wie GML SVG und andere von großer Bedeutung. Für den Geodatenaustauschs ist GML 3 der derzeit umfassendste Standard zur Repräsentation von Geodaten, der sich als ISO/CD 19136-Standard in Bearbeitung befindet. Aufbauend auf vordefinierten Basis-Schemata lassen sich eigene, anwendungsspezifische Schemata aufbauen und 0D-, 1D-, 2D- und 3D-Geometrien modellieren und übertragen. Dabei werden die notwendigen Elemente und Datentypen unter Nutzung einfacher Typen sowie durch Ableitung von abstrakten GML-Datentypen deklariert. In Instanzdokumenten werden dann Geo-Objekte mit ihren Geometrie- und Fachattributen definiert bzw. durch geeignete Programme erzeugt.

Kennzeichnend für die aktuelle Version von GML ist sicher die hohe Komplexität, die eine entsprechende Einarbeitung voraussetzt. Für die Entwicklung eigener Anwendungsschemata existieren IT-Werkzeuge, so dass diese aus UML-Modellen automatisiert abgeleitet werden können (Düren 2005) oder mittels XML-Editoren wie XML-Spy direkt erstellt werden können (Abitew 2005).

Die Entwicklung von GML-Profilen kann die weitere Verbreitung begünstigen.

Das Landesvermessungsamt Baden-Württemberg hat die Arbeit zum Thema GML-Modellierung durch die Bereitstellung von Beispieldaten sowie der notwendigen Dokumentation wirksam unterstützt. Herzlichen Dank, insbesondere an Herrn Gerald Graf für die Zusammenarbeit!

6 Literaturverzeichnis

Abitew, Dejen (2005): *Developing a GML Model for Thematic Mapping Data of Baden-Württemberg Land Survey Authority*. Unveröffentliche Masterarbeit, Hochschule für Technik, Stuttgart

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV) – AFIS-ALKIS-ATKIS-Koordinierungsgremium (2004): *Modellierung von Fachinformationen unter Verwendung der GeoInfoDok*. Stand: 01.10.2004, http://www.adv-online.de/ [03.08.2005]

Watt, A., Lilley C. et al. (2001): *SVG Unleashed*. SAMS, 1117 S., ISBN 0-672-32429-6

Barth, Yvonne (2004): *Nutzung von Servlets, JavaServer Pages, XML und XSL zur SVG–basierten Visualisierung raumbezogener Daten*. Unveröff. Diplomarbeit, Hochschule für Technik Stuttgart, http://www.carto.net/papers/yvonne_barth/2004_yvonne_barth_dfkviewer.pdf [12.01.2005]

Behr, F.-J. (2005): XML-basierte Visualisierung von Geodaten mittels SVG. . in: Kettemann, Rainer, Coors, Volker (2005): *Aktuelle Entwicklungen in der Geoinformatik*. Tagungsband 5. Vermessungsingenieurtag, Hochschule für Technik, Stuttgart

Constantin P. (2005): Die Einführung des Amtlichen Liegenschaftskataster-Informationssystem ALKIS in Baden-Württemberg. *Mitteilungsheft des DVW-Landesvereins Baden-Württemberg*, Heft 1, März 2005

Cox, S., Daisey, P., Lake, R., Portele, C., Whiteside, A. (2004): *Geography Markup Language (GML)* 3.1. http://portal.opengeospatial.org/files/?artifact_id=4700 [25.07.2005]

Düren, Ulrich, Riekert, Jens (2005): *Einführung / Normen und Standards für Geoinformation für AAA*. http://www.lverma.nrw.de/produkte/liegenschaftsinformation/katasterinfo/alkis/ [03.08.2005]

Düren, Ulrich (2005): XML, GML, NAS. http://www.lverma.nrw.de/produkte/-liegenschaftsinformation/katasterinfo/alkis/images/aktuell/-5_Normen_Standards_XML_GML_NAS.pdf [03.08.2005]

Fibinger, Iris, 2001: *SVG: Untersuchung des XMLSstandards für zweidimensionale Vektorgraphiken und Erstellung eines SVG-Tutorials*, http://www.carto.net/papers/iris_fibinger-/2001_03_iris_fibinger_svg_diplomarbeit.pdf [23.03.2001]

Graf, Gerald (2004): *Vektordatenausgabe im Format SVG am Beispiel der Ausgabe von Thematischen Kartenfachdaten (TKFD)*. http://www.lv-bw.de/LVShop2/produktinfo/wir-ueber-uns/links/svg/-VektordatenAusgabeImFormatSVG_110105.pdf [07.02.2005]

Lake, Ron, Burggraf, D.S., Trninic, M., Rae, L. (2004). *Geography Markup Language (GML): Foundation for the Geo-web*. John Wiley & Sons, Chichester, England, 388 S.

Kettemann, Rainer (2005a): Geodaten in Baden-Württemberg werden interoperabel. *Mitteilungsheft des DVW-Landesvereins Baden-Württemberg*, Heft 1, März 2005

Kettemann, Rainer (2005b): GIS im Intra-/Internet und Web-Dienste für Geoinformationssysteme. in: Kettemann, R., Coors, V. (2005): *Aktuelle Entwicklungen in der Geoinformatik*. Tagungsband 5. Vermessungsingenieurtag, Hochschule für Technik, Stuttgart

Kolbe, Thomas (2004): *Repräsentation von Geodaten mit der Geography Markup Language GML 3*. Seminar D100-007-12-04, Zentrum für graphische Datenverarbeitung, Darmstadt

Neumann, Andreas (2004): Using SVG for Online Digitizing and Editing of Geographic Data. Proceedings *SVG Open Conference*, Tokyo, 2004, http://www.svgopen.org/2004/papers/-abstract_neumann_svgopen_2004_svg_as_editing_environment/ [10.02.2005]

Neumann, Andreas, Winter, André (2003): *Kartographie im Internet auf Vektorbasis, mit Hilfe von SVG nun möglich*. http://www.carto.net/papers/svg/index_d.shtml [10.02.2005]

Open Geospatial Consortium (2002): *Web Feature Service Implementation Specification*. https://portal.opengeospatial.org/files/?artifact_id=7176 [10.02.2005]

Open Geospatial Consortium (2004): *Web Map Service*. http://portal.opengis.org/-files/?artifact_id=5316 [10.02.2005]

Ordnance Survey (2005). *Transforming GML (XSLT)*. http://www.ordnancesurvey.co.uk/oswebsite/-products/osmastermap/xml/gml.html [07.02.2005]

Seifert, Markus (2005): Das AFIS-ALKIS-ATKIS-Anwendungsschema als Komponente einer Geodateninfrastruktur. *Zeitschrift für Vermessungswesen*, 2/2005

Strobel, S. (2004). Anwendung von GML als herstellerübergreifende Geodatenschnittstelle. http://www.agis.unibw-muenchen.de/HTML/Weiterbildung/13_Strobel.pdf [03.08.2005]

Vretanos, P. (2005): *GML simple features profile (GMLsf)*. http://portal.opengeospatial.org/-files/?artifact_id=11266 [03.08.2005]

W3C (2005): *Document Object Model*. http://www.w3.org/DOM/ [07.02.2005]

W3C (2002): XML *Path Language (XPath) Version 1.0, Deutsche, kommentierte Übersetzung*. http://www.obqo.de/w3c-trans/xpath-de [07.02.2005]

Whiteside, Arliss (2005): *Recommended XML/GML 3.1.1 encoding of common CRS definitions (XML for CRS)*. https://portal.opengeospatial.org/files/?artifact_id=8837 [25.07.2005]

Whiteside, Arliss, Müller, Markus U., Fellah, Stephane, Warmerdam, Frank (2005): *Web Coordinate Transformation Service (WCTS)*. https://portal.opengeospatial.org/files/?artifact_id=8847 [03.08.2005]