Dokumentation
zur
Modellierung der Geoinformationen des amtlichen Vermessungswesens
(GeoInfoDok)

AAA-Signaturenkatalog

Beschreibung des
AAA-SK-Objektmodells
Version 1.1.0

Stand: 06.12.2017

Arbeitsgemeinschaft der Vermessungsverwaltungen
der Länder der Bundesrepublik Deutschland (AdV)

INHALTSVERZEICHNIS

Dokumentenhistorie

Version	Stand	Bemerkung	Beteiligte
1.0	01.07.2016	Erstellung des Hauptdokumentes	I. Fibinger, K. Rothe, U. Schmitz, B. Sender
1.0	16.11.2016	Revision 2016: Kapitel 6: Ergänzung Abkürzungsverzeichnis Alle Kapitel: Revision aus Rückmeldungen der Länder Kapitel 2: Anpassungen an Abschlussbericht Version 1.10 (#3586, #3587) Kapitel 2: Anpassungen an Abschlussbericht Version 1.10.1 Kapitel 3: Anpassungen an geänderte Dokumentstruktur (PG SK)	I. Fibinger
1.1.0	06.12.2017	Revision 2017: Kapitel 2: Anpassungen an Abschlussbericht Version 1.11: #3861, #4103, #4104, #4105, #4106, #4107 + Dokumentationsänderungen in Emit und RuleSet zum ATKIS-Implizitverhalten Kapitel 2: Anpassungen an Abschlussbericht Version 1.12: #4158, #4209 + Dokumentationsergänzungen für Halo (mehrfache Definition), Layer und Styles (qualifizierte Assoziationen), Modell-Diagramme aktualisiert. Kapitel 3: Ergänzung des Geometrietyps im ALKIS-SK und der Funktion NUMERISCH_NACH_ROEMISCH() sowie redaktionelle Änderungen	I. Fibinger, M. Giesecke, K. Rothe, U. Schmitz

1 Aufbau, Inhalt und Ziel

Mit der Umsetzung der formalisierten AAA-Signaturenkataloge wurden neben technischen auch grundsätzliche Ansätze der bisherigen Signaturenkataloge überdacht. Die wesentliche Veränderung liegt in der Harmonisierung sämtlicher Signaturenkataloge von AAA, so dass die Belange aller Modelle und Präsentationsformen (u. a. auch über AAA hinausgehende Anwendungen wie den WebAtlasDE [Web-SK]) berücksichtigt wurden. Für eine einheitliche Umsetzung wurde für die Signaturenkataloge ein eigenes sprachenunabhängiges Modell entwickelt, das sog. AAA-SK-Objektmodell, welches als UML-Modell vorliegt (Kapitel 2). Dieses ist Basis für die Umsetzung der eigentlichen Dokumentenform der SKs: dem SK-XML (Kapitel 5). Ferner wurden die bisherigen SKs in die Pflegesprache für die Projektgruppen der Signaturenkataloge überführt: die „Map Definition Language“ (MDL). Diese wird hier nur in Ansätzen beschrieben (Kapitel 4) und tritt nur im Zusammenhang mit der Beschreibung der für den allgemeinen Nutzer lesbaren Dokumentation der AAA-SKs in Erscheinung (Kapitel 3).

1.1 Ausgangssituation

In dem AFIS-ALKIS-ATKIS-Referenzmodell[1] sind die Beziehungen zwischen Regelungs-, Produktions- und Kommunikationsebene beschrieben. Das hier vorliegende Dokument beschreibt ausschließlich die Definition, Beschreibung und Weitergabe der Signaturierungsregeln für AFIS, ALKIS und ATKIS. Gerade in puncto Weitergabe wurde für die AAA-SK auch die Nutzung der Dienste berücksichtigt (insbesondere sei hier der WebAtlasDE genannt).

1.2 Umsetzung der Formalisierung

Das AAA-SK-Objektmodell wurde als Teil des AAA-Modells konzipiert:

Abbildung 1.1: Das AAA-SK-Objektmodell im Kontext des AAA-Modells

[1] Hauptdokument der GeoInfoDok: Das gemeinsame AFIS-ALKIS-ATKIS-Referenzmodell, 1996, Kapitel 2

Das unten stehende Schaubild (Abb. 1.2) zeigt die Zusammenhänge der Komponenten

- Objektmodell
- Quellcode-Repository
- MDL-Compiler und -Doku
- Ausgabeformate

auf. Dabei ist das von interactive instruments in Zusammenarbeit mit der AdV entwickelte SK-Objektmodell (blauer Kasten) die Basis für den strukturellen Aufbau des SK-XML (Schema) sowie von MDL. Die von den SK-Gruppen erzeugten SKs sind in MDL überführt. Diese *.mdl-Dateien werden in einem Repository gehalten und mittels Subversion gepflegt (gelber Kasten).

Das zentrale Modul für die Pflege der SKs ist der sog. MDL-Compiler (grüner Kasten). Dieser hat neben dem SK-Objektmodell auch das AAA-Objektmodell implementiert, so dass die SKs an beiden Modellen validiert werden können. Darüber hinaus erzeugt er aus den erfolgreich validierten MDL-Dokumenten das für die offizielle Ausgabe konzipierte SK-XML. MDL-Doku ist ein zusätzliches Programm, das aus den MDL-Dokumenten die menschenlesbare und durch eine grafische Umsetzung der Signaturen schnell interpretierbare Dokumentation (Kapitel 2) erstellt. Das derzeitige Ausgabeformat dieser Dokumentation ist HTML.

Für die Implementierung der SKs in Softwareprodukte wird SK-XML ausgegeben. Dies wird den Softwarefirmen zur Verfügung gestellt (orangener Kasten). Deren Ausgabe kann beliebig sein.

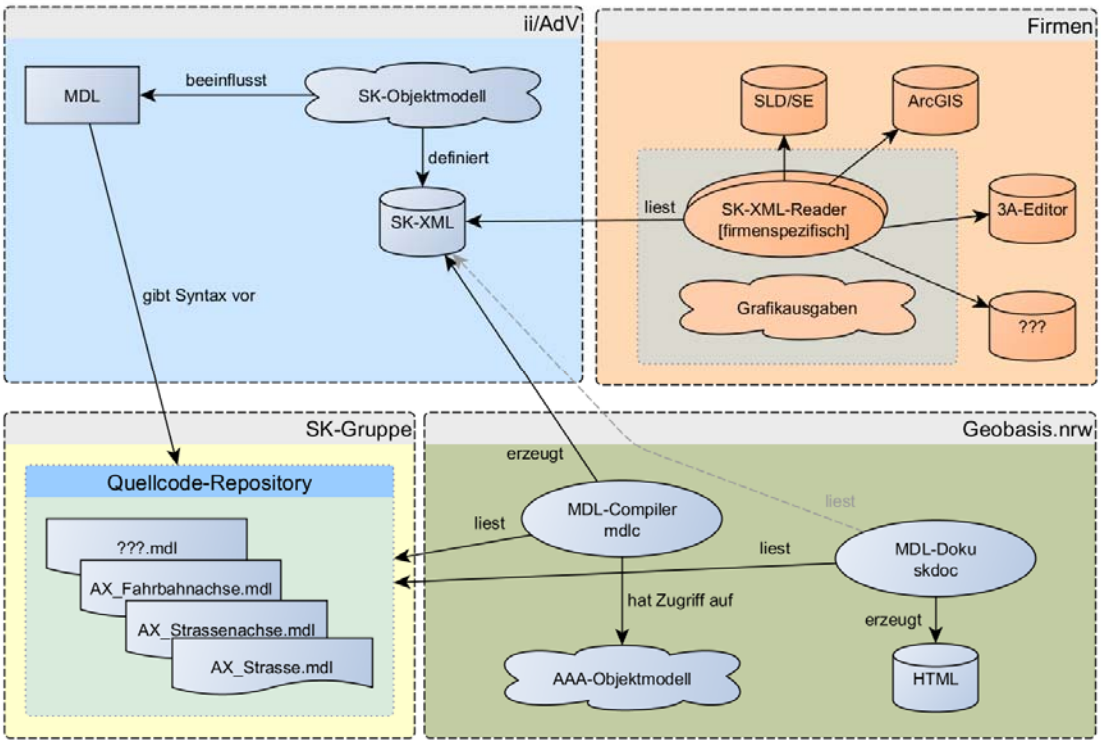


Abbildung 1.2: Komponenten der formalisierten AAA-SK (Schmitz, 2016)

1.3 Ziele und Nutzer

Im Sinne einer automationsfreundlichen Umsetzung zur Erlangung eines höheren Aktualitätsgrads der verschiedenen amtlichen Präsentationen ist eine Vereinheitlichung und gute Strukturierung der Signaturenkataloge sowohl für die SK-Pflegestellen (zur Qualitätssicherung) als auch für die SK-implementierenden Firmen von großem Vorteil. Ziel ist es, eine vollautomatische Implementierung der Signaturenkataloge in die Softwareprodukte zu erreichen. Dies gilt im Moment noch nicht für Positionierungsregeln (DesignRules), die erst nach und nach in sog. PlacementRules (formale Beschreibungen der Positionierung von Signaturen) umgesetzt werden.

Allen Nutzern gerecht soll die in Kapitel 3 beschriebene Dokumentation werden.

2 Das Modell[2]

In diesem Kapitel wird das SK-Modell im Groben vorgestellt. Eine ausführliche Dokumentation der Klassen, Attribute und Rollen befindet sich im Modell selbst. Sie wurden in den von Enterprise Architect® (EA)[3] dafür vorgesehenen Dokumentationsfeldern für die Klassen, Attribute und Rollen abgelegt.

2.1 Vorbemerkungen

An die Neufassung des SK wurden einige Erwartungen gestellt:

- Das zu entwickelnde SK-Modell und die damit verbundenen Formate MDL und SK-XML sollen die SKs von AFIS, ALKIS und ATKIS in gleicher Syntax und harmonisiert darstellen.
- Es soll gleichzeitig geeignet sein für den Web-SK.
- Die genannten SKs sollten vollständig übernommen werden können.
- Das Modell soll so gemacht sein, dass daraus automatisch gebräuchliche Signaturierungsformate abgeleitet werden können. Besonders hervorgehoben ist SLD/SE.

Besonders durch die geforderte Unterstützung für den Web-SK kamen einige Zusatzanforderungen ins Spiel, die die Kataloge „Web-fähig“ machen sollen. Das bedeutet insbesondere die Gestaltbarkeit von SKs, die über weite Maßstabsbereiche hinweg eingesetzt werden können. Die klassischen SK sind ja tatsächlich nur für einen bestimmten Maßstab gedacht. Dem wurde durch Konzepte wie Layers und RuleSets Rechnung getragen, die für die klassischen SKs nicht erforderlich wären. Auch die Forderung nach Styles kommt aus dem Web-SK.

Die offiziellen Vorgaben des OGC-Standards SLD/SE wurden ebenfalls berücksicht. Zwar konnte dieser Standard nicht 1:1 umgesetzt werden, dennoch wurde eine Angleichung in dem Sinne vorgenommen, dass die Signaturbeschreibungen im neuen SK mit denen von SLD/SE vergleichbar sind.

Das bestehende System aus komplexen, schwer automatisierbaren Positionierungsregeln und vergleichsweise einfachen Signaturierungsregeln wurde im Modell dahingehend optimiert, dass die Signaturierungsvorschriften künftig leichter eine automatisch abgeleitete Präsentation ermöglichen.

Modellierung ist ein nicht eindeutig definiertes Fachgebiet. Meist sind neben den eingesetzten Lösungen auch andere denkbar. Kriterien in dieser Modellierung sind (neben der Erfüllung der oben beschriebenen Anforderungen) die Benutzung von Standards, Verständlichkeit des Entwurfs, Erweiterbarkeit und ästhetische Gesichtspunkte.

[2] Erstling, 2016, Abschlussbericht, Version 1.10.1, Kapitel 5
[3] http://www.sparxsystems.de/

2.2 Allgemeine Festlegungen

Das Modell wurde in UML 2.0 als Klassenmodell in Enterprise Architect® (EA) verfasst. Der Modellierungsstil orientiert sich an ISO 19109, macht aber nicht von allen angebotenen Möglichkeiten Gebrauch (z.B. gibt es keine FeatureTypes im Modell). Soweit möglich wurden die Basistypen aus ISO 19103 und die Geometriedefinitionen aus ISO 19107 verwendet.

Die Namen von Klassen, Attributen und Rollen wurden in englischer Sprache formuliert und nach den üblichen Regeln der Groß-/Kleinschreibung benannt („CamelCase“). Klassennamen, einschließlich „Enums“, beginnen mit Großbuchstaben, Attribut- und Rollennamen mit Kleinbuchstaben.

Klassen, Attribute und Rollen sind im EA-Modell im Einzelnen beschrieben.

Rollennamen sind grundsätzlich nur in der navigierbaren Richtung von Assoziationen vergeben. Die meisten Assoziationen sind als Aggregationen und Kompositionen formuliert, wobei die Navigierbarkeit immer vom „Behälter“ zum „Bestandteil“ geht.

Folgende Basistypen wurden für die Attribute vorausgesetzt:

Typbezeichnung	**Beschreibung**
Boolean	Wahrheitswert (ISO 19103)
CharacterString	Text beliebiger Länge (ISO 19103)
Real	Gleitkommazahl (ISO 19103)
Integer	Ganzzahl (ISO 19103)
Date	Datumsangabe (ISO 19103)
GenericName	Namespace-behafteter Name (ISO 19103)
ID	ID (entsprechend XML Schema)

Viele Klassen im Modell sind aus der abstrakten Klasse *Element* abgeleitet, erkennbar im Diagramm an der so lautenden Eintragung jeweils rechts über dem Klassennamen.

Abbildung 2.1: Referenzierbare Objekte

Die Ableitung aus *Element* versorgt die Klassen mit einer Reihe von identifizierenden und beschreibenden Attributen. Das Attribut *id* ist dabei als eindeutiger Identifikator der Objekte gedacht. Liegt ein solcher an einem Objekt vor, so kann das Objekt an einer Stelle eines Instanzendokuments (also als MDL-Datenbasis oder als SK-XML-Dokument) bestehen und mehrfach durch Referenzierung benutzt werden.

In praktischer Nutzung des Objektmodells via SK-XML kann das optionale Attribut *vendorSpecific* verwendet werden, um eine SK-Instanz bei Bedarf mit implementierungsspezifischen Informationen (*Tags*) anzureichern. Diese bestehen aus Schlüssel/Werte-Paaren (*key* und *value*) und werden in dem Behältertyp *TagSet* zusammengefasst. Die Verwendung von *vendorSpecific* und Bedeutung von *Tags* wird im SK-Modell nicht festgelegt, sie sind anwendungsspezifisch.

2.3 Signaturenkatalog, Layers und Regeln

Das folgende Diagramm in Abbildung 2.2 zeigt die Gesamtsicht auf einen Signaturenkatalog und dessen grobe Strukturierung. Ein Gesamtkatalog, z.B. der gesamte ALKIS-SK, wird durch ein *SymbologyCatalog*-Objekt vertreten. Das Objekt enthält diverse Metadaten zum Katalog. Dazu zählen auch die aus *Element* ererbten Attribute wie *name*, *synopsis* und *description*.

Ein *SymbologyCatalog* ist aus einer oder mehreren *Layer*-Definitionen aufgebaut, die ihrerseits mit *Styles* verknüpft sein können. *Layers* sind ein den klassischen AAA-SKs eher fremdes Konstrukt. Sie sind wegen der Einbeziehung des Web-SKs vorhanden, welcher eine Differenzierung in thematische Bildebenen vorsieht.

Die Layers können einen Behälter für „freie Symbolizer“ beinhalten: Klasse *Additional-Symbolizers*. „Freie Symbolizer“ sind solche, die nicht in Emits eingebunden sind, z. B. weil sie im ATKIS-Kontext von „freien Präsentationsobjekten“ referiert werden. Sie werden direkt durch eine konkrete Ausprägung der *PureSymbolizers* angegeben.

Bei den *Styles* handelt es sich um Objekte, die ein allgemeines Mittel zur Definition mehrerer Varianten ein und desselben *Layers* ermöglichen, etwa Ausgestaltung in Farbe und in Graustufen. Das Konstrukt ist primär vorhanden, weil es vom Web-SK benötigt wird. Deshalb ist seine Nutzung im Modell auch zunächst nur auf die vom Web-SK für *Styles* vorgesehenen Konstrukte beschränkt. Diesen entsprechen im Modell *Color* (siehe Abschnitt 2.7) und *Emit*. Eine Erweiterung auf andere benötigte Konstrukte kann aber leicht vorgenommen werden, falls der Bedarf auftaucht.

Wird einem *Layer* mindestens ein *Style* zugewiesen, ist bei Präsentation des Layers die Vorgabe eines (der zugewiesenen) Styles erforderlich.

Gleichfalls wird für den Web-SK auch die Klasse *RuleSet* benötigt, welche die für einen Maßstabsbereich benötigten Regeln (*Rules*) zusammenfasst. Über die Attribute minScaleDenominator und maxScaleDenominator kann ein Maßstabbereich für die

Anwendung der im RuleSet enthaltenen Regeln definiert werden. Normalerweise werden für den Maßstabsbereich des *RuleSets* alle Längen- und Größenangaben der Signaturierung so eingesetzt, wie sie in den *Symbolizers* der *Rules* spezifiziert sind, d.h. für alle Maßstäbe gleich. Davon kann aber durch Verwendung der Eigenschaft *targetScaleDenominator* abgewichen werden. In diesem Falle gelten die Längen- und Größenangaben nur für diesen so angegebenen Maßstab. In allen anderen Maßstäben wird entsprechend verkleinert oder vergrößert.

Ein Objekt der Klasse *Rule* ist die Entsprechung einer Ableitungsregel im ATKIS-SK. Bei ALKIS liegen die Dinge komplizierter. Hier ist die Entsprechung jeweils eine zusammengesetzte Selektionsbedingung, die dann einer oder mehreren Signaturnummern zugeordnet ist.

Ein *Rule*-Objekt ist aus einem *Filter*- und einem oder mehreren *Emit*-Objekten zusammengesetzt. Das *Filter*-Objekt entspricht dem selektierenden Anteil der Regeln, also Objektart, Geometrietyp, Objekttyp und attributive Einschränkungen. Jedes *Emit* entspricht in ATKIS der rechten, die Darstellung beschreibenden, Halbzeile einer Ableitungsregel.

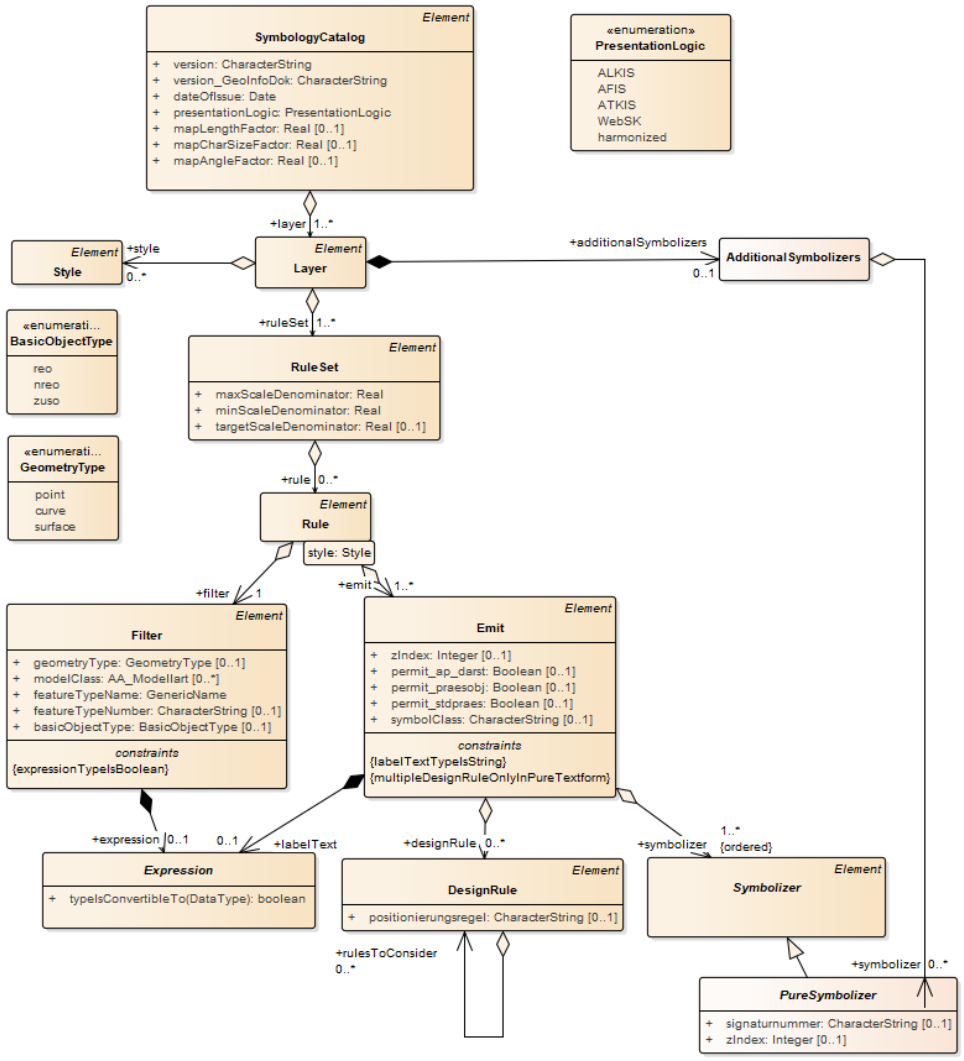


Abbildung 2.2: Signaturenkatalog, Layers und Regeln

Die *Emit*-Objekte, die zu einer *Rule* gehören, werden qualifiziert über *Style*-Objekte referiert. Zu jedem *Style* können mehrere *Emit*-Objekte angegeben werden. Die Qualifizierung kann auch fehlen. Das bedeutet, es können qualifizierte und unqualifizierte *Style*s an *Emit*s gemischt vorkommen. Die qualifizierten *Emit*s werden dann nach *Style* selektiert, und die nicht qualifizierten werden alle ohne *Style*-Auswahl übernommen.

Emit enthält meist die Darstellungspriorität (*zIndex*)[4], Art der Signatur (*symbolClass*) und SK-spezifische logische Attribute. Die wichtigste Information ist jedoch der Verweis auf die Symbolizer und die DesignRules. Im Normalfall[5] liegt nur ein Symbolizer vor, der den Definitionen unter einer Signaturnummer entspricht. Die *DesignRule*-Objekte entsprechen den textlich formulierten Positionierungsregeln, die ebenfalls über Nummern identifiziert werden. Es können mehrere *DesignRules* angegeben sein, da oft mehrere Positionierungsregeln zusammenwirken. Nach Abschluss der Formalisierung des SK soll nur höchstens eine *DesignRule* am *Emit* vorliegen. Diese enthält dann die *PlacementRules*, welche die formale Definition der Platzierung ausdrücken.

Am *Emit*-Objekt kann auch ein labelText als Ausdruck angegeben werden (das bisherige SIT-Attribut). Es wirkt dann als Vorgabewert für alle im *Symbolizer* angesprochenen *Label*-Objekte. Im *Label*-Objekt kann der vorgegebene *labelText* überschrieben werden.

2.4 Ausdrücke

Ausdrücke werden im Modell an wenigen Stellen eingesetzt:

- Im *Filter*-Objekt zur Selektion von Modelldaten.
- Im *Emit* als Vorgabewert für den Textinhalt, ergänzt um das *Label*-Objekt, wo diese Angabe überschrieben werden kann.
- In diversen komplexen *PlacementRules*.
- Als *symbolizerPredicate* beim Vorliegen mehrerer *FilteredSymbolizer* am *Emit*.

Der Einsatz von Ausdrücken zur Selektion von Modelldaten für die Präsentation ist auch im bisherigen SK bereits in beschränktem Umfang vorhanden, sowohl bei ALKIS als auch bei ATKIS. Der vorliegende Entwurf setzt die vorhandenen Ansätze fort und verstärkt sie um die Mittel, die erforderlich sind, um objektübergreifende Selektionen durchführen zu können.

Die Realisierung der *Expression*-Objekte wird in MDL und SK-XML sehr verschieden vorgenommen. Während in SK-XML eine dem UML-Diagramm entsprechende Darstellung in XML gewählt wurde, um die automatische Umsetzbarkeit der Ausdrücke in verschiedene andere Sprachumgebungen zu erleichtern, wurde in MDL eine Textnotation

[4] Wenn *zIndex* angegeben wird, so überschreibt dieser den *zIndex* in den zum *Emit* gehörenden *Symbolizers*. Dort ist die Angabe des *zIndex* verpflichtend.

[5] Mehrere *Symbolizer* werden eingesetzt, wenn die Symbolisierung von der Geometrie abhängt, die durch die *DesignRules* bestimmt wird. Es kommt dann ein *FilteredSymbolizer* zum Einsatz.

gewählt, die dem durch die UML-Klassen definierten Subset[6] der Xpath-Sprache [XPATH] entspricht.

Der Ausgangspunkt für die Ausdrücke ist die abstrakte Klasse *Expression*. Sie steht für einen beliebigen Ausdruck, der aus Dateninhalten, Konstanten und Verknüpfungen zusammengesetzt werden kann. Als zusätzliche Form ist die Existenzquantifizierung (*Some*) zugelassen, sowie das *If*-Konstrukt und die *For*-Schleife.

Das folgende Diagramm beschreibt das Modell für Ausdrücke:

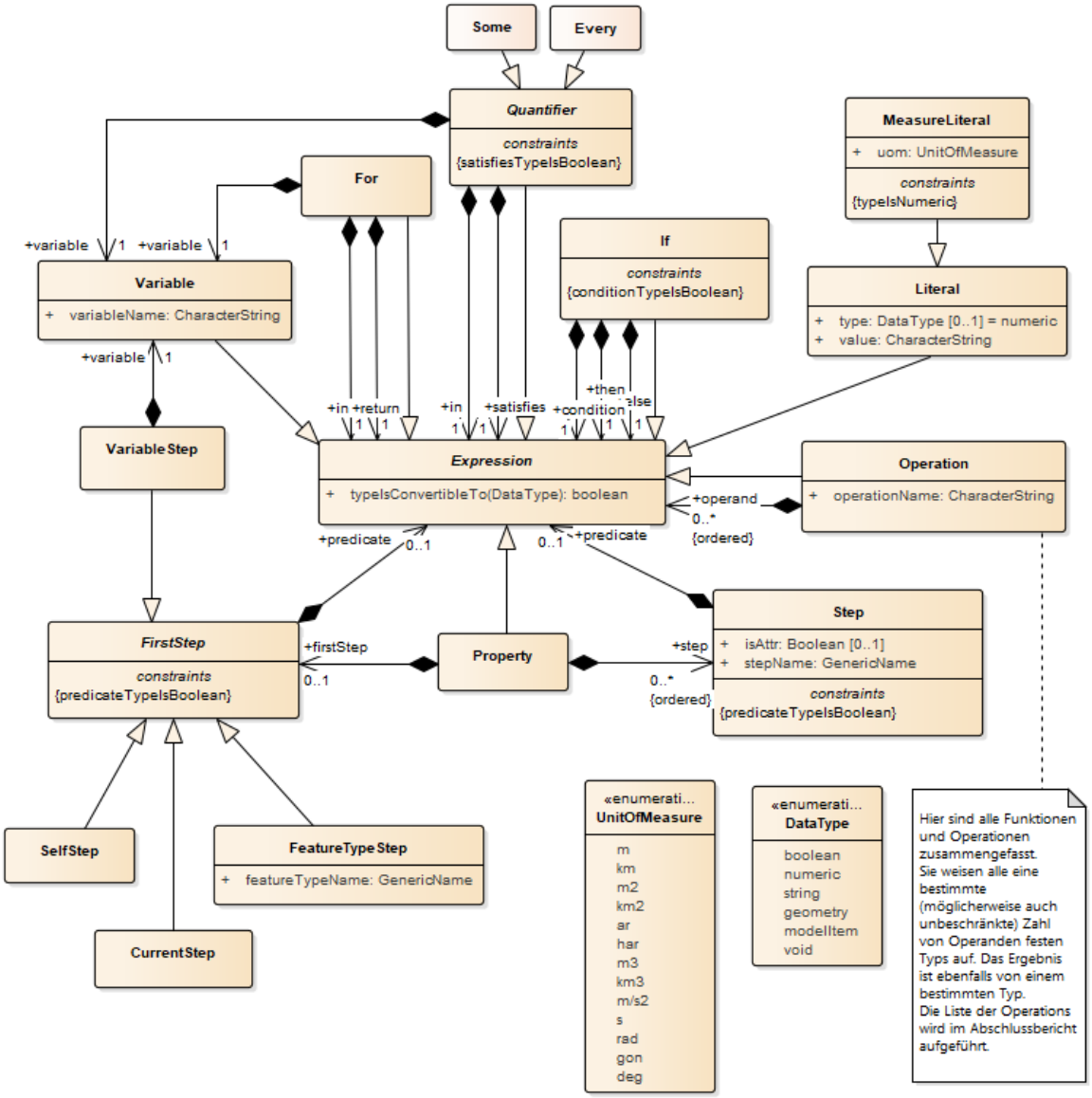


Abbildung 2.3: Ausdrücke

[6] Es ist nicht ganz ein Subset. Hinzugefügt wurden mögliche Maßeinheiten bei numerischen Konstanten und eine Infix-Operation „in“ zum Test ob ein Element in einer Menge liegt.

2.4.1 Dateninhalte

Die Adressierung von Dateninhalten erfolgt durch die Klasse *Property*. *Property* ist einer Untermenge der Xpath-Syntax für die Adressierung von XML-Elementen nachgebildet und navigiert wie diese durch ein XML-Infoset. Als Datenhaltung wird somit ein GML-Dokument der vollständigen Datenhaltung entsprechend dem NAS-Schema angenommen.

Die Adressierung erfolgt durch Pfade, die aus einzelnen „Steps“ zusammengesetzt sind. Dabei werden im Modell ein erster „Step“ – *FirstStep* – und alle weiteren – Step – unterschieden. Die Navigation startet normalerweise von einem Kontext, der durch das jeweils zu präsentierende Feature gegeben ist. Durch jeden Step wird dieser Kontext auf das jeweils adressierte Element weitergeschaltet.

Es ist zu beachten, dass durch die Navigation im (angenommenen) GML-Dokument oft nicht einzelne Werte entstehen sondern Mengen. Der Umgang mit diesen erfolgt nach den Regeln von Xpath. Solche Mengen sind in Wirklichkeit angeordnete Listen (Sequenzen) von Werten, d.h. sie können mehrfach dieselben Werte enthalten. Zugelassen sind sie nur an ganz speziell bezeichneten Positionen des Ausdruckskalküls.

Die Elemente von Listen müssen im definierten Subset der Xpath-Sprache immer denselben Datentyp aufweisen[7].

In jedem Step (auch in *FirstStep*) kann optional eine Selektion durch einen beliebigen boolschen Ausdruck (ein Prädikat) erfolgen.

Beispiel in Xpath:

//AX_Flurstueck[amtlicheFlaeche>1000m2]/flurnummer

FirstStep ist abstrakt und kann mehrere konkrete Formen annehmen:

SelfStep steht für den aktuellen Kontext. Äquivalent in Xpath: „self::“ oder „.“. *SelfStep* ist auch der Default, wenn nichts angegeben ist.

CurrentStep steht für das aktuelle, jeweils zu präsentierende Entity. Die Entsprechung in Xpath ist „current()“.

VariableStep nimmt Bezug auf ein Variable-Objekt. In manchen Fällen (z.B. bei komplexen Positionierungsregeln, im *Some*-Konstrukt oder im *For*-Konstrukt) werden alternative Kontexte über Variablen vorgegeben. Die Entsprechung in Xpath ist „$name“, wobei „name“ der Name der Variablen ist.

[7] Operationen, die „gefährlich“ für einen Verstoß gegen diese Einschränkung wären, wurden deshalb im Subset weggelassen, beispielsweise die Vereinigungsoperation von XPath („|“). Bei anderen wurde die „Typreinheit“ explizit gefordert, z.B. beim *for*-Konstrukt.

FeatureTypeStep setzt den Kontext auf alle Features des angegebenen *featureTypeName*. In Xpath entspricht dieser Klasse das Konstrukt „//name“, wobei name für den Namen des Feature-Types steht.

Die Step-Objekte führen den anfänglich durch *FirstStep* vorgegebenen Kontext schrittweise fort. Aufgrund der Struktur von GML wechseln sich dabei stets Property-Namen und Feature-Type-Namen ab. In Xpath entspricht dies z.B.:

//AX_Flurstueck/istGebucht/AX_Buchungsstelle/buchungsart

Durch den Schalter *isAttr* kann angezeigt werden, dass es sich um ein XML-Attribut handelt. In Xpath:

//AX_Flurstueck/istGebucht/AX_Buchungsstelle/@gml:id

Die adressierten Dateninhalte sind grundsätzlich so, wie im AAA-Modell beschrieben und wie sie in der Datenhaltung vorliegen. Geometrien liegen in Weltkoordinaten vor.

2.4.2 Konstanten

Für Konstanten stehen Objekte der Klasse *Literal* als mögliche Ausdrücke zur Verfügung. Mögliche Datentypen von Konstanten sind: *boolean*, *numeric*, *string*. Numerische Konstanten können mithilfe der Klasse *MeasureLiteral* auch mit einer Einheit (Attribut *uom*) aus einer begrenzten Menge von Längen- und Flächeneinheiten (siehe Enum *UnitOfMeasure*) versehen werden.[8]

In Xpath werden die Einheiten in Erweiterung der Standard-Xpath-Syntax [XPATH] bei Zahlen einfach angehängt.

25km oder *25 km*

sind zwei mögliche Schreibweisen.

Einheiten sollen bei Vergleichen und numerischen Berechnungen automatisch berücksichtigt werden. Dies geschieht im Kontext des MDL-Compilers. Dabei tragen mit Maßen versehene Dateninhalte (Klasse *Property*) aufgrund der NAS-Schemadefinition selbst immer eindeutige Einheiten. Bei Vergleichen und Addition bzw. Subtraktion werden die Einheiten automatisch angeglichen. Bei Produktbildungen und Quotienten werden die Einheiten entsprechend verrechnet und vereinheitlicht. Das Resultat einer Verrechnung muss immer eines der zulässigen Maßeinheiten sein, siehe Enum *UnitOfMeasure*.

In SK-XML darf davon ausgegangen werden, dass diese Angleichung bereits erfolgt ist. Eine Berücksichtigung von Einheiten ist dort deshalb nicht mehr erforderlich, sofern die Modelldaten den Vorgaben des Objektartenkatalogs folgen. Die Angabe im *uom*-Attribut dient nur zur Dokumentation.

2.4.3 Verknüpfungen und Funktionen

Die Klasse *Operation* steht für alle 0 bis n-stelligen Operationen und Funktionen. Nach dem Stand der inzwischen erfolgten Überarbeitung dieses Abschlussberichts handelt es sich dabei um die in der folgenden Tabelle angegebenen.

Es ist davon auszugehen, dass die Liste im Zuge des weiteren Ausbaus und der Harmonisierung der SKs weiter ergänzt werden muss. Der Ausbau soll so vorgenommen werden, dass primär im Xpath-Standard bereits vorhandene Konstrukte verwendet werden.

Hinweis: Die Argumente dürfen im Allgemeinen keine Mengen/Listen[9] sein. Ausnahmen sind besonders gekennzeichnet. An den Stellen, an denen Listen erlaubt sind, sind auch

[8] Zusätzlich wurden alle Maßeinheiten von Properties im AAA-Modell aufgenommen, auch wenn diese für den SK vermutlich keine Bedeutung haben werden.

[9] Obwohl im Zusammenhang von multipel auftretenden Konstrukten der Begriff „Mengen“ sehr anschaulich ist, wird er vermieden, da es sich in allen Fällen um angeordnete Listen handelt, die auch die Wiederholung von Elementen zulassen.

immer einfache Argumente gestattet. Sie wirken dann wie Listen, die nur aus einem einzigen Element bestehen.

Hinweis: Die angegebenen Datentypen in den Argumenten bedeuten nicht, dass als Argument unbedingt ein Ausdruck des betreffenden Typs geschrieben werden muss. Vielmehr sollen immer die impliziten Konversionsvorschriften von Xpath zum Tragen kommen.

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
fn:true	0	Wahrheitswert true	boolean	fn:true()
fn:false	0	Wahrheitswert false	boolean	fn:false()
fn:unary-	1	Vorzeichen - 1. arg: numeric	numeric	- arg
fn:unary+	1	Vorzeichen + 1. arg: numeric	numeric	+ arg
fn:+	2	Addition 1. arg1: numeric 2. arg2: numeric	numeric	arg1 + arg2
fn:-	2	Subtraktion 1. arg1: numeric 2. arg2: numeric	numeric	arg1 — arg2
fn:*	2	Multiplikation 1. arg1: numeric 2. arg2: numeric	numeric	arg1 * arg2
fn:/	2	Division 1. arg1: numeric 2. arg2: numeric	numeric	arg1 div arg2
fn:mod	2	Modulo 1. arg1: numeric 2. arg2: numeric	numeric	arg1 mod arg2
fn:=	2	Gleich 1. arg1: numeric, string, 17oolean 2. arg2: numeric, string, boolean	boolean	arg1 = arg2
fn:!=	2	Ungleich 1. arg1: numeric, string, 17oolean	boolean	arg1 != arg2

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
		2. arg2: numeric, string, boolean		
fn:is	2	Identität von Objekten 1. arg1: modelItem 2. arg2: modelItem	boolean	arg1 is arg2
fn:<	2	Kleiner 1. arg1: numeric, string 2. arg2: numeric, string	boolean	arg1 < arg2
fn:>	2	Größer 1. arg1: numeric, string 2. arg2: numeric, string	boolean	arg1 > arg2
fn:<=	2	Kleiner oder gleich 1. arg1: numeric, string 2. arg2: numeric, string	boolean	arg1 <= arg2
fn:>=	2	Größer oder gleich 1. arg1: numeric, string 2. arg2: numeric, string	boolean	arg1 >= arg2
fn:in	2	Erstes Argument kommt in der durch *in* eingeleiteten Liste wertmäßig vor. 1. arg1: numeric, string 2. arg2: Listennoperand des einheitlichen Typs numeric oder string	boolean	Erweiterung der Xpath-Sprache: arg1 in arg2
fn:sequence	0..*	Listenkonstruktor 1. arg1: Element(e) 2. arg2: Element(e) … Alle Argumente müssen denselben Typ bezüglich des Schemas aufweisen, also z.B. einen bestimmten FeatureType oder einen einfachen Typ, z.B. numeric oder string. Die Elemente können selbst auch wieder multipel sein. Es entstehen dadurch jedoch keine geschachtelten Listen sondern eine solche auf	Liste einheitlichen Typs Die leere Liste ist mit allen Typen verträglich.	(arg1,arg2,..) Leere Liste: ()

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
		einer Ebene. Der Listenkonstruktor ist überall gestattet, wo multiple Werte zugelassen sind. Wenn der Listenkonstruktor ohne Argument aufgerufen wird, entsteht die leere Liste.		
fn:not	1	Nicht 1. arg: boolean	boolean	fn:not (arg)
fn:and	2..*	Und 1. arg1: 19oolean 2. arg2: 19oolean …	boolean	arg1 and arg2 and …
fn:or	2..*	Oder 1. arg1: 19oolean 2. arg2: 19oolean …	boolean	arg1 or arg2 or …
fn:boolean	1	Konversion zu boolean 1. arg: beliebiger Typ, Listen sind zulässig	boolean	fn:boolean(arg)
fn:number	1	Konversion zu numeric 1. arg: string oder anderer geeigneter Typ	numeric	fn:number(arg)
fn:string	1	Konversion zu string D. arg: anderer geeig-neter Typ In Einschränkung von Xpath wird der String-Value eines modellItems nur bei Multiplizität 1 oder 0 unter-stützt.	String	fn:string(arg)
fn:abs	1	Absolutbetrag 1. arg: numeric	numeric	fn:abs(arg)
fn:ceiling	1	Kleinste ganze Zahl größer oder gleich 1. arg: numeric	numeric	fn:ceiling(arg)
fn:floor	1	Größte ganze Zahl kleiner	numeric	fn:floor(arg)

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
		oder gleich 1. arg: numeric		
fn:round	1	5-er Rundung 1. arg: numeric	numeric	fn:round(arg)
fn:stringlength	1	Stringlänge 1. arg: string	numeric	fn:string-length(arg)
fn:substring	2..3	Teilstring 1. arg1: string 2. arg2: numeric – Position 3. arg3: numeric – Länge Das dritte Argument kann ausgelassen werden und defaultiert dann zur verbleibenden Restlänge.	String	fn:substring(arg1, arg2, arg3)
fn:concat	2..*	Konkatenation 1. arg1: string 2. arg2: string …	string	fn:concat(arg1, arg2, …)
fn:uppercase	1	Übersetzung in Großbuch-staben 1. arg: string	string	fn:upper-case(arg)
fn:lowercase	1	Übersetzung in Kleinbuch-staben 1. arg: string	string	fn:lower-case(arg)
fn:translate	3	Zeichenweise Übersetzung 1. arg1: Zu übersetzender string 2. arg2: Zu übersetzende Zeichen 3. arg3: Korrespondierende Zielzeichen	string	fn:translate(arg1, arg2, arg3)
fn:matches	2..3	Regex-Prädikat 1. arg1: Zu prüfender string 2. arg2: Regex-Muster 3. arg3: Regex-Flags Das dritte Argument kann ausgelassen werden (Defaultverhalten)	boolean	fn:matches(arg1, arg2, arg3)

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
fn:replace	3..4	Ersetzung durch Regex-Test 1. arg1: Zu testender String 2. arg2: Regex-Muster 3. arg2: Ersetzungs-String 4. arg3: Regex-Flags Das vierte Argument kann ausgelassen werden (Defaultverhalten)	string	fn:replace(arg1, arg2, arg3, arg4)
fn:tokenize	2..3	Auftrennen einen Strings in eine Liste 1. arg1: Zu zerlegender string 2. arg2: Regex-Ausdruck 3. arg3: Regex-Flags Das dritte Argument kann ausgelassen werden (Defaultverhalten)	Liste von string	fn:tokenize(arg1, arg2, arg3)
fn:stringjoin	2	Zusammenfügen der Inhalte einer Liste zu einem String mit Trenner 1. arg1: Zusammenzufügen-de Liste 2. arg2: String: Trenner	string	fn:string-join(arg1, arg2)
fn:count	1	Zahl der Elemente D. arg: Zu zählende Liste Das Argument darf multipel sein. Ist es das nicht, ist das Ergebnis 1, bzw. 0 bei der leeren Liste.	Numeric	fn:count(arg)
fn:sum	1	Summe der Elemente D. arg: Zu summieren-de Liste Die Liste muss numerische Elemente aufweisen. Ist das Argument nicht multipel, so ist dessen Wert das Resultat. Ist die Liste leer, so kommt 0 zurück.	Numeric	fn:sum(arg)
fn:avg	1	Arithmetisches Mittel der Elemente	Numeric	fn:avg(arg)

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
		D. arg: Liste deren Mittelwert zu bestimmen ist. Die Liste muss numerische Elemente aufweisen. Ist das Argument nicht multipel, so ist dessen Wert das Resultat. Ist die Liste leer, so liegt eine Fehlersituation vor.		
fn:min	1	Minimum der Elemente D. arg: Liste deren Minimum zu bestimmen ist. Die Liste muss numerische Elemente aufweisen. Ist das Argument nicht multipel, so ist dessen Wert das Resultat. Ist die Liste leer, so liegt eine Fehlersituation vor.	Numeric	fn:min(arg)
fn:max	1	Maximum der Elemente D. arg: Liste deren Maximum zu bestimmen ist. Die Liste muss numerische Elemente aufweisen. Ist das Argument nicht multipel, so ist dessen Wert das Resultat. Ist die Liste leer, so liegt eine Fehlersituation vor.	Numeric	fn:max(arg)
mdl:formatnumber	2	Formatierung einer Zahl zu einem String 1. arg1: Zahl 2. arg2: Zahlenmuster wie in XSLT format-number, wobei nur die Bedeutung von Punkt und Komma vertauscht ist. Beispiel für arg2: „###.##0,000“	string	mdl:format-number(arg1, arg2)
mdl:sort	2..3	Sortieren eine Liste von Objekten 1. arg1: Zu sortierende Liste 2. arg2: Ausdruck zur	Liste beliebiger, aber gleichar-	mdl:sort(arg1, arg2, arg3)

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
		Angabe des Sortier-Schlüssels im Kontext von arg1 3. arg3: String: Richtung (‚a' für ascending oder ‚d' für descending) Das dritte Argument kann mit Wirkung ‚a' entfallen.	tiger Objekte	Achtung: In Xpath-Darstellung als Funktionsaufruf muss arg2 als String angegeben werden!
mdl:geo_equals	2	Geometrische Gleichheit 1. arg1: geometry 2. arg2: geometry	boolean	mdl:geo_equals (arg1, arg2)
mdl:geo_contains	2	Geometrisches Enthalten-sein 1. arg1: geometry 2. arg2: geometry	boolean	mdl:geo_contains (arg1, arg2)
mdl:geo_intersects	2	Geometrischer Schnitt 1. arg1: geometry 2. arg2: geometry	boolean	mdl:geo_intersects (arg1, arg2)
mdl:geo_crosses	2	Geometrische Durchdrin-gung 1. arg1: geometry 2. arg2: geometry	boolean	mdl:geo_crosses (arg1, arg2)
mdl:geo_touches	2	Geometrische Berührung 1. arg1: geometry 2. arg2: geometry	boolean	mdl:geo_touches (arg1, arg2)
mdl:geo_isArea	1	Prädikat: Geometrie ist eine Fläche oder eine Liste von Flächen. 1. arg: geometry	boolean	mdl:geo_isArea(arg)
mdl:geo_isCurve	1	Prädikat: Geometrie ist eine Linie oder eine Liste von Linien. 1. arg: geometry	boolean	mdl:geo_isCurve(arg)
mdl:geo_isPoint	1	Prädikat: Geometrie ist ein Punkt oder eine Liste von Punkten. 1. arg: geometry	boolean	mdl:geo_isPoint(arg)
mdl:geo_area	1	Flächenberechnung, Ergebnis in m2	numeric	mdl:geo_area(arg)

operation-Name	Anzahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
		1. arg: geometry		
mdl:geo_length	1	Längenberechnung, Ergebnis in m 1. arg: geometry	numeric	mdl:geo_length(arg)
mdl:geo_crsid	0..1	Ermittlung eines Identifikators für das CRS einer Geometrie. D. arg: geometry Wenn das Argument fehlt, so wird „position“ des aktuellen Features herangezogen, sofern dieses ein solches Property besitzt. Ansonsten liegt eine Fehlersituation vor. Die Rückgabe erfolgt im GID-Format, also z.B. ETRS89_UTM32.	String	mdl:geo_crsid(arg)
mdl:geo_coord	2	Ermittlung der ersten, zweiten oder dritten Koordinate einer Punktgeometrie 1. arg1: geometry (Punkt) 2. arg2: Ganzzahl, 1|2|3	numeric	mdl:geo_coord(arg1, arg2)

Das Feld „operationName“ in der obigen Tabelle steht für das Attribut *Operation.operationName* im Modell. Die Werte identifizieren in dieser Form also auch die Operations im SK-XML.

Die „Xpath-Darstellung“ definiert das Äquivalent in der MDL-Darstellung.

Der Namespace
http://www.w3.org/2005/xpath-functions
für die Standardfunktionen von Xpath, wurde wie üblich mit dem Präfix „fn“ assoziiert.
Für den verwendeten Präfix
mdl — für allgemeine MDL-bezogene Erweiterungen des Xpath-Funktionsvorrats
wurde der Namespace
http://www.adv-online.de/namespaces/adv/mdl/6.0
gewählt.

Der Funktionsvorrat wird SK-spezifisch erweitert um weitere Funktionen, die komplexe Sachverhalte des jeweiligen SK ausdrücken. Bisher sind solche Funktionen nur für ATKIS bekannt.

Für die beispielhaft untersuchten Funktionen und Prädikate des ATKIS-SK50 könnte dies wie im Folgenden angedeutet aussehen:

operation-Name	Zahl Argu-mente	Beschreibung	Typ	Xpath-Darstellung
LGO	0	Maximal zusammenhän-gende Länge	numeric	atk:LGO()
FLB	0	Maximal zusammenhän-gende Fläche	numeric	atk:FLB()
LIEGT_IM_TUNNEL_ODER_SCHUTZGALERIE	0	Abkürzendes Prädikat aus dem ATKIS-SK	boolean	atk:LIEGT_IM_TUNNEL_ODER_SCHUTZGALERIE()
…	…	…		…
LIEGT_NICHT_IM_SIEDLUNGSGEBIET	0	Abkürzendes Prädikat aus dem ATKIS-SK	boolean	atk:LIEGT_NICHT_IM_SIEDLUNGSGEBIET()

Der für die Xpath-Darstellung verwendete Präfix

atk — für ATKIS-SK-spezifische Erweiterungen

wurde für die folgenden Namespaces verwendet:

http://www.adv-online.de/namespaces/adv/mdl-atk10/6.0
http://www.adv-online.de/namespaces/adv/mdl-atk25/6.0
http://www.adv-online.de/namespaces/adv/mdl-atk50/6.0
http://www.adv-online.de/namespaces/adv/mdl-atk100/6.0

Es ist anzustreben, die für die verschiedenen Maßstäbe prinzipiell unterschiedlichen Funktionen zu harmonisieren, um sie am Ende einem einzigen Namespace zuordnen zu können.

Dasselbe soll im Prinzip auch für die Angaben in „operationName“ gelten. Hier ist allen Angaben ein Präfix „atk:“ vorangestellt. Sollte es nicht möglich sein, die ATKIS-Prädikate und Funktionen für die verschiedenen Maßstäbe zu harmonisieren, so ist „atk:“ zu ersetzen durch „sk10:“, „sk25:“, usw.

2.4.4 Quantifizierung

Die Objekte der konkreten Spezialisierungen der Klasse *Quantifier* stehen für quantifizierte Aussagen über einen Ausdruck, der eine Liste bezeichnet.

Die beiden Spezialisierungen sind *Some* und *Every* und stehen entsprechend für Existenzquantifizierung und Allquantifizierung.

Die Aussage bei *Some* ist: Es gibt mindestens ein Element in der durch *in* bezeichneten Liste, welches den Ausdruck, der unter *satisfies* angegeben ist, erfüllt. Dabei wird im *satisfies*-Ausdruck das laufende Element durch die Variable an der Rolle *variable* bezeichnet.

Entsprechend bei *Every*: Alle Element der durch *in* bezeichneten Liste erfüllen den Ausdruck, der unter *satisfies* angegeben ist. Auch hier wird im *satisfies*-Ausdruck das laufende Element durch die *Variable* an der Rolle *variable* bezeichnet.

Die Rolle *in* ist eine der wenigen Stellen in der Ausdruckssprache, die Listen erlaubt. Ein Skalar an dieser Stelle wirkt wie eine 1-elementige Liste. Listen können z.B. durch den Listenkonstruktor[10] erzeugt werden. Die Ausdruckssprache ermöglicht es nicht, Listen zu bilden, die aus verschiedenen Datentypen zusammengesetzt sind. Daher kann bei der Interpretation von *satisfies* von einem definierten Schemakontext ausgegangen werden.

Der Kontext wird durch das *Some*- oder *Every*-Objekt nicht verändert, d.h. *SelfStep* hat zu Beginn beider Ausdrücke dieselbe Bedeutung wie außerhalb des *Some/Every*.

In Xpath entsprechen *Some* und Every den Konstrukten:

some $var in expr1 satisfies expr2

every $var in expr1 satisfies expr2

Some/Every werden nur bei sehr komplexen Filtern anzuwenden sein.

2.4.5 Schleifenkonstrukt

Das *For*-Konstrukt modelliert eine Schleife über die Elemente einer Liste. Der Zweck ist die Umformung der Elemente der Liste durch Ausdrücke auf den Elementen.

Die durch in bezeichnete Liste wird dabei durchlaufen, wobei das laufende Element durch die Variable *variable* bezeichnet wird. In der Rolle *return* wird der Ausdruck angegeben, der die Umformung definiert. Alle Ergebnisse in *return* werden zu einer neuen Liste zusammengefügt, die am Ende das Resultat des *For* ist.

Der Kontext wird durch das *For*-Objekt nicht verändert, d.h. *SelfStep* hat zu Beginn beider Ausdrücke dieselbe Bedeutung wie außerhalb des *For*.

In Xpath entspricht *For* dem Konstrukt:

for $var in expr1 return expr2

[10] Operation sequence(x1,x2,…) im Modell, realisiert durch (x1,x2,...) in XPath.

Im SK-Modell gilt über die Semantik von Xpath hinaus die Einschränkung, dass die Elemente der erzeugten Liste alle einen einheitlichen Typ (bezüglich des Schemas) aufweisen müssen.

2.4.6 If-Konstrukt

Das *If*-Konstrukt entscheidet anhand einer Bedingung zwischen zwei Ausdrücken.

Die Bedingung wird durch die Rolle *condition* angegeben. Hat *condition* den Wahrheitswert *true*, so wird das Resultat des *If* der unter *then* bezeichnete Ausdruck, ansonsten der unter *else* bezeichnete.

In Xpath entspricht *If* dem Konstrukt:

if E1 then E2 else E3

Im SK-Modell gilt über die Semantik von Xpath hinaus die Einschränkung, dass die Ausdrücke E2 und E3 denselben Typ (bezüglich des Schemas) aufweisen müssen. E2 und E3 dürfen Listen sein. Die Gleichheitsbedingung bezüglich der Typen gilt dann für deren Elemente.

2.5 Positionierungsregeln

Eine der wesentlichen Neuerungen des vorliegenden SK-Modells ist die Neufassung der Positionierungsregeln. Sie sind im folgenden Diagramm dargestellt:

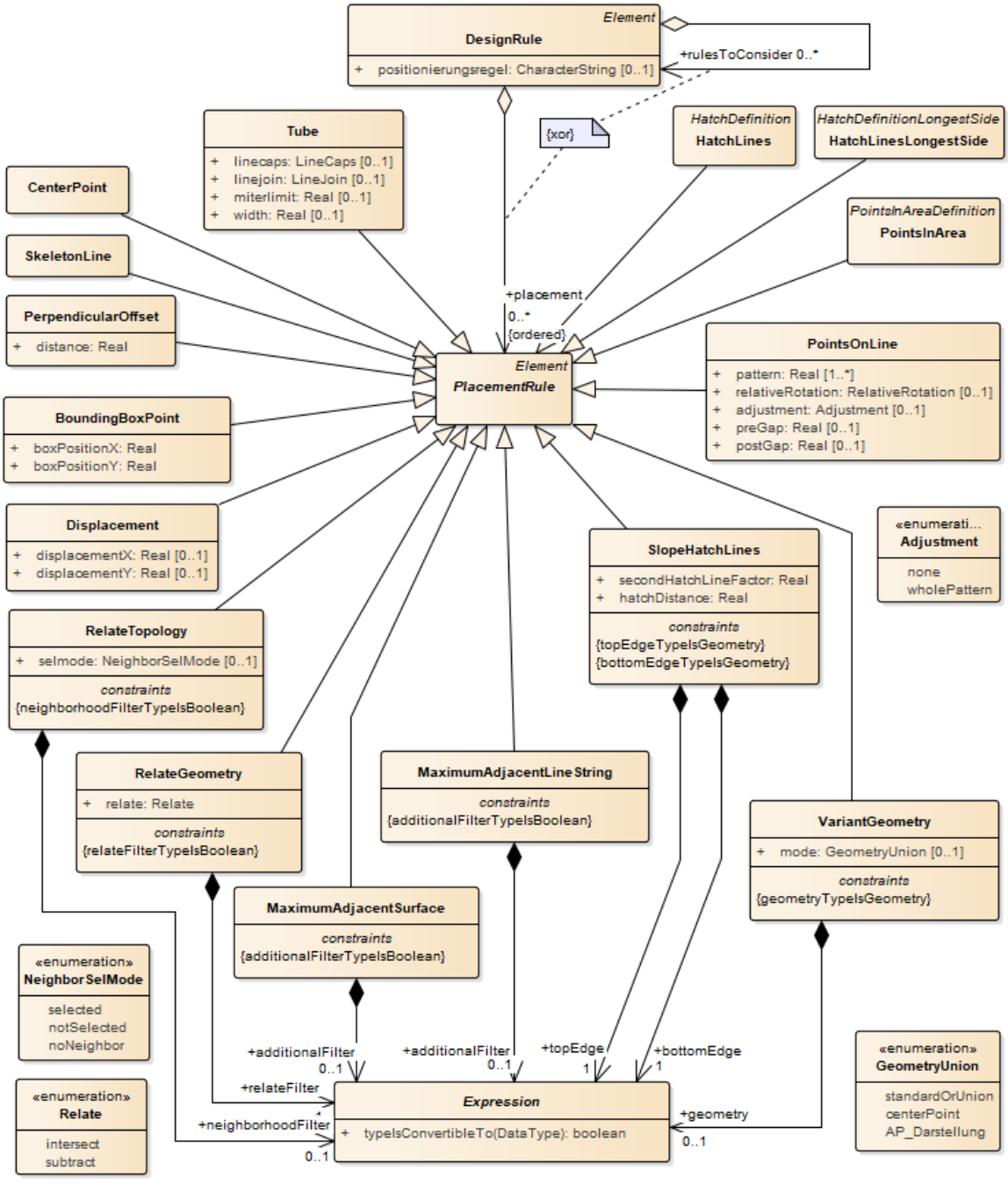


Abbildung 2.4: Positionierungsregeln

Im bisherigen ATKIS-SK (und auch im ALKIS-SK) sind die Positionierungsregeln nur als Texte formuliert. Die Texte nehmen dabei in freier Gestaltung alle Regeln zur Darstellung auf, die sich nicht mit dem formal (und recht eng) gefassten Signaturenkatalog und den ebenfalls (mehr oder weniger) formal verfassten Ableitungsregeln ausdrücken lassen.

Diesen herkömmlichen Texten, welche über eine Positionierungsregelnummer identifiziert werden, entspricht die Klasse *DesignRule* im Modell. Nur Objekte der Klasse *DesignRule* können aus den bisherigen AAA-SKs gewonnen werden. Für die Nachbildung der „zusätzlich zu beachtenden Positionierungsregeln" steht die Rolle *rulesToConsider* an *DesignRule* zur Verfügung. Die Nutzung dieser Rolle soll verschwinden, wenn später eine exakte Beschreibung über *PlacementRules* vorliegt.

Alle anderen im Diagramm aufgeführten Klassen, die alle aus *PlacementRule* abgeleitet sind, sind ein Vorgriff auf eine zukünftige Überarbeitung der SKs mit dem Ziel, auch für Positionierungsregeln eine formale Definition zu ermöglichen. Dies gilt im Übrigen nicht für den Web-SK. Hier könnte durchaus eine völlig automatische Umsetzung der bestehenden maschinenlesbaren SKs vorgenommen werden.

Die Idee bei der neuartigen Formalisierung der Positionierungsregeln ist die der „sukzessiven Geometrietransformation". Jeder formal zu definierenden *DesignRule* kann eine Folge von konkreten *PlacementRules* zugeordnet werden, welche die ursprünglich dem Fachobjekt entnommene Geometrie (bei REOs *position*) in jeweils definierter Weise verändern. Mehrere *PlacementRules* können dabei hintereinander geschaltet werden, um die zu beschreibende Geometrieumformung zu konstruieren. Wenn keine *PlacementRule* angegeben wird, so wird die Geometrie aus dem Fachobjekt unverändert signaturiert. Die jeweiligen Geometrien in der Kette sind über die Variable *$geometry* zugreifbar. Sie liegen grundsätzlich im Weltsystem vor.

Die einzelnen konkreten *PlacementRules* sind dabei so konstruiert, dass sie bei entsprechender Wahl ihrer Attribute eine möglichst große Zahl definierter Geometrieumformungen beschreiben können. Sie sollen also möglichst allgemeingültig angelegt sein. Ziel ist es, eine überschaubare Menge klar beschreibbarer und realisierbarer Muster für die Positionierung zu erhalten. Beispiele:

- *BoundingBoxPoint* konstruiert aus einer beliebigen Geometrie einen Beschriftungspunkt, indem sie das achsenparallele einschließende Rechteck zur Geometrie bildet und aus diesem einen speziellen Punkt auswählt. Die Auswahl des Punkts erfolgt mittels zweier „Koordinaten", die von 0 bis 1 laufen. Die Angabe *X=1*; *Y=1* bezeichnet z.B. den rechten, oberen Punkt. Mit nachgeschalteter Verwendung von *Displacement* kann eine zusätzliche Verschiebung um einen festen Vektor angegeben werden. Damit können z.B. die Positionierungsregeln 103 bis 109 beschrieben werden.
- *MaximumAdjacentSurface* ist eine deutlich komplexere Positionierungsregel. Die *PlacementRule* fordert die Bestimmung eines als Fläche maximal zusammenhängenden Sets von Features. Das Ergebnis des Placements ist die Vereinigung dieser zusammenhängenden Geometrien. Die Bestimmung erfolgt implizit unter dem Filter des *Emit* und zusätzlich unter dem über *additionalFilter* anzugebenden Prädikat. In diesem erscheint das auszugebende REO in der Variablen *$emit* (Variable-Objekt mit Namen „emit", *$emit* in Xpath) und das zu vergleichende REO als lokaler Kontext (Objekt *CurrentNodeStep*, entsprechend *current()* in Xpath). Mit *MaximumAdjacentSurface* kann z.B. die Positionierungsregel 110 auf eine neue Grundlage gestellt werden.

Nicht jede der definierten *PlacementRules* ist an jeder Position der Folge von *PlacementRules* sinnvoll. Die aggregierenden (wie z.B. *MaximumAdjacentSurface*) oder ersetzenden (*VariantGeometry*) sind nur am Anfang der Kette sinnvoll einsetzbar. Auch ist zu beachten, dass verschiedene *PlacementRules* nicht für alle Geometriedimensionen geeignet sind.

Bei den aggregierenden Rules ist hervorzuheben, dass die Definition sich ausschließlich auf den sichtbaren Effekt der Präsentation richtet und nicht auf das Verfahren, wie dieser erzielt wird. Die Implementierung muss also darauf achten, dass nicht für jeden Teil einer Aggregation das Aggregations- und Ausgabeverfahren wiederholt wird.

Es ist an dieser Stelle vielleicht noch der Hinweis angebracht, dass das erarbeitete Modell entsprechend der Beauftragung nicht auf einer vollständigen Analyse aller bestehenden SKs und evtl. zukünftig bestehender Wünsche zu Darstellungsoptionen beruht. Es ist also ziemlich wahrscheinlich, dass bei der Überarbeitung der SKs unter Einsatz von MDL weitere erforderliche PlacementRules entdeckt werden. Ihre Gestaltung soll im Sinne der Realisierbarkeit und größtmöglicher Allgemeingültigkeit vorgenommen werden.

2.6 Signaturen

Die abstrakte Klasse *Symbolizer* fasst die Definitionen zusammen, die in den bisherigen SKs in den Signaturenkatalogen stehen.

Die konkreten Signaturdefinitionen sind in der Klasse *PureSymbolizer* zusammengefasst. Es handelt sich um *PointSymbolizer*, *LineSymbolizer*, *AreaSymbolizer* und *TextSymbolizer*. Zusätzlich wird noch ein *NullSymbolizer* bereitgestellt, um Darstellungen dokumentiert unterdrücken zu können. Alle *PureSymbolizer* außer dem *NullSymbolizer* müssen eine Darstellungspriorität (*zIndex*) aufweisen. Diese kann durch die gleichnamige Eigenschaft im *Emit* überschrieben werden.

Die konkreten Symbolizer (*PureSymbolizer*) gehen in ihren Gestaltungsmöglichkeiten weit über die recht engen Definitionen aus den bisherigen Signaturenkatalogen hinaus. Sie orientieren sich dabei an den Gestaltungsmöglichkeiten des OGC SLD/SE Standards und an den während des Designs geäußerten fachlichen Anforderungen.

Das neue Design der Signaturierung wird deshalb – ebenso wie bei den Positionierungsregeln – bei der Erstübernahme der bestehenden SKs nur zu einem Bruchteil genutzt werden. Eine weitere Nutzung kann erst durch manuelle Fortführung des SK erschlossen werden, wobei die Kompatibilität zu den vorhandenen Datenbeständen (Präsentationsobjekte!) zu beachten ist.

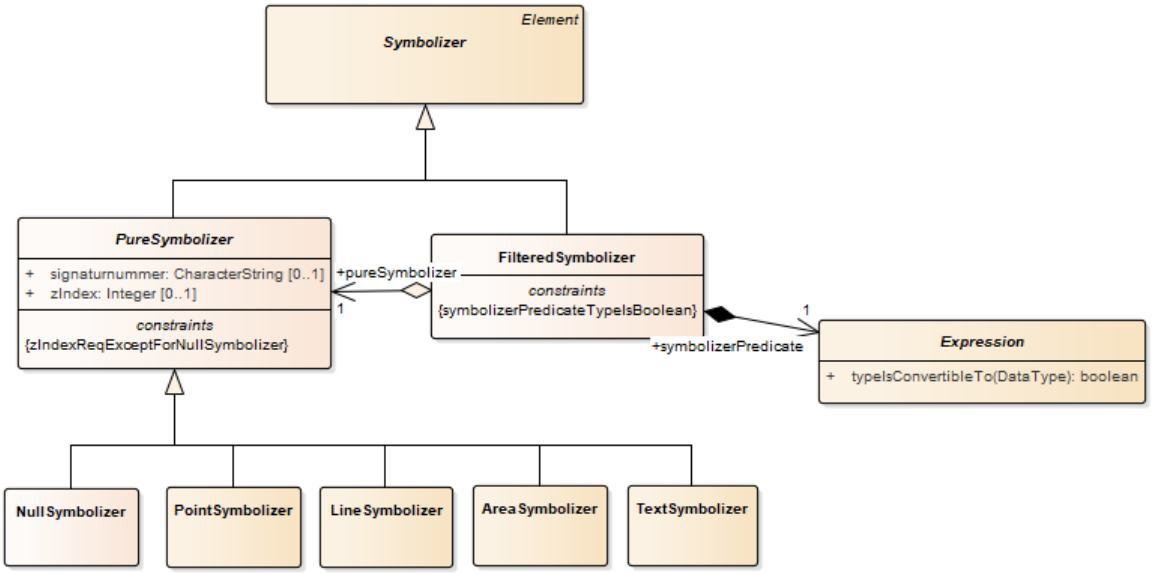


Abbildung 2.5: Symbolizer

Die Anforderungen an die Signaturierung beim Web-SK bleiben im Prinzip im Rahmen der klassischen SKs. Allerdings stecken in den klassischen SKs die Gestaltungsoptionen zur Hauptsache in den Positionierungsregeln, die im Web-SK kaum zur Anwendung kommen, so dass die Darstellungen dort eher sehr einfach sind. Die einzige beobachtete Positionierungsregel im Web-SK ist die Parallelenbildung bei Linien. Es ist allerdings zu erwarten, dass bei der zukünftigen Fortentwicklung des Web-SK komplexere, an SLD/SE orientierte Signaturierungen zum Einsatz kommen werden.

Neben den *PureSymbolizer*-Objekten enthält das Modell eine weitere Klasse zur Angabe eines Prädikats, die Klasse *FilteredSymbolizer*. Mit deren Hilfe kann bei Angabe mehrerer solcher Symbolizer eine ausdrucksgesteuerte Auswahl getroffen werden. Auf diese Weise können z.B. in Abhängigkeit von der durch die Positionierungsregeln definierten Geometrie (etwa in Abhängigkeit von deren Größe) verschiedene Symbolizer wirksam werden. Auf die Geometrie kann im Ausdruck über die Variable *$geometry* zugegriffen werden.

2.6.1 Punktsignaturen

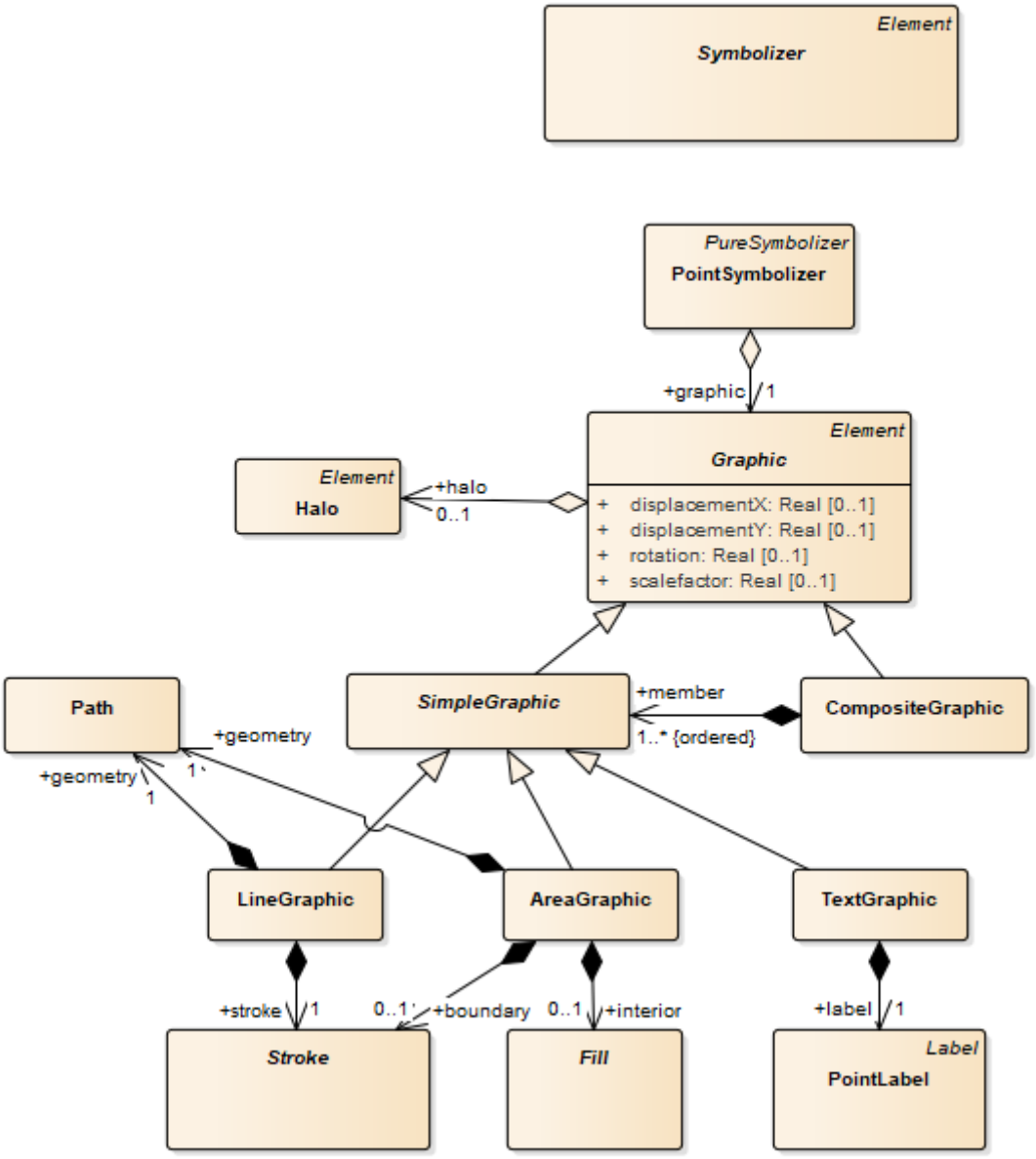


Abbildung 2.6: Punktsignaturen

Die Signaturen für Punkte liegen im bisherigen AAA-SK bereits präzise ausgearbeitet vor. Ihre Umwandlung in ein maschinenlesbares Format macht sie der Verwendung durch Kartierungssoftware verfügbar. Die Signaturen für Punkte werden durch relative Bildkoordinaten im 1/100 mm-System als einzelne Figurenteile bestehend aus Flächen und Linien beschrieben. Diese ihrerseits sind den auch für sonstige Flächen und Linien verfügbaren Darstellungsoptionen ausgesetzt, also Farbfüllung, Strichlierung, Gestaltung der Linienenden, usw.

Das neue SK-Modell für *PointSymbolizer* bildet weitgehend das bisherige Modell nach. Wegen der Wiederverwendbarkeit in anderen Kontexten – etwa für die Platzierung von Punktsymbolen an Linien oder zur Flächenfüllung – ist dabei die eigentliche Definition der Punktsignatur im Objekt Graphic versammelt.

Ein *Graphic* kann ein Behälter namens *CompositeGraphic* sein, wodurch zusammengesetzte Punktsignaturen ausgedrückt werden, oder eine einzelne Signatur dargestellt durch die Klasse *SimpleGraphic*. Als einzelnes *Graphic* oder Teil eines *CompositeGraphic* nimmt ein *SimpleGraphic* eine Form als Linie, Fläche oder Text an in Gestalt der Klassen *LineGraphic*, *AreaGraphic* und *TextGraphic*.

Bei der Zusammenfassung von *SimpleGraphic*-Objekten über *CompositeGraphic* zu einem *Graphic* sind ein paar Besonderheiten zu beachten.

Die Zusammenfassung erfolgt in einer festen Reihenfolge, die nach dem "Painters-Model" umgesetzt wird. Die beschreibenden Attribute der Teile werden vor der Zusammenfassung im Gesamtsymbol ausgeführt, etwa eine Verschiebung, Drehung oder Skalierung. Das resultierende Symbol kann dann durch Eigenschaften aus *CompositeGraphic* in seiner Gesamtheit nochmals verschoben, rotiert oder skaliert werden.

Dies gilt grundsätzlich auch für *Halo*. Das bedeutet: Bei Angabe eines Halos am *CompositeGraphic* wird im ersten Durchgang diese Halodefinition zur Freistellung aller enthaltenen Members eingesetzt (wobei Halodefinitionen an diesen zunächst ignoriert werden). Danach werden die Members in der angegebenen Reihenfolge ausgegeben. Sofern die Members eigene Halodefinition tragen, wird diese vor der Ausgabe der eigentlichen Grafik zur Freistellung des Members benutzt. Im Falle des *TextGraphic* wird dessen Halodefinition vor der Halodefinition im *PointLabel* eingesetzt.

In der Praxis dürfte es jedoch ausreichen, *Halo* entweder nur an *CompositeGraphics* oder nur an *SimpleGraphics* zu definieren.

Allen *Graphic*-Ableitungen gemeinsam sind einige Attribute. Dabei sind die Versatzwerte (*displacementX*/ *displacementY*) streng genommen nicht nötig, um den SK zu übernehmen, da der Versatz hier bereits in die Symboldefinitionen eingearbeitet ist. In der Zukunft können sie aber praktisch sein, um Definitionsaufwand zu sparen. *Rotation* und *scalefactor* werden benötigt, um die Wirkung des punktförmigen Präsentationsobjekts richtig beschreiben zu können.

Die konkreten Graphics *LineGraphic* und *AreaGraphic* werden durch *Path*-Objekte mit Geometrie versorgt, siehe hierzu das Geometrie-Diagramm. Die Ausstattung mit Ausgestaltungsmerkmalen zu Flächeninhalten und Linienmerkmalen erfolgt durch *Fill* und *Stroke* genau wie bei sonstigen Flächen und Linien.

TextGraphic ist eine Erweiterung, die die Kombination von Flächen- und Linienteilen mit Texten erlaubt. Auf diese Weise können Texte aus den Modelldaten in Punktsymbole eingebettet werden. Die Versorgung mit Texten und dessen Ausgestaltung wird durch die Klasse *PointLabel* beschrieben.

Eine nähere Erläuterung zu *Halo* (Freisteller) siehe Abschnitt 2.6.4.

2.6.2 Liniensignaturen

Liniensignaturen werden im bestehenden AAA-SK ähnlich wie die Punktsignaturen präzise beschrieben. Es wurde daher im neuen Modell ein Teilbereich angelegt, der es ermöglicht, diese Definitionen eins-zu-eins und platzsparend zu übernehmen. Zusammengesetzte Definitionen im AAA-SK müssen bereits unter Benutzung der neuen, verbesserten Konstrukte übernommen werden, da sonst in Bezug auf den Musterausgleich falsche Resultate erzielt werden.

Die Defizite der bisherigen Definition liegen im Musterausgleich – dies wird rein textlich durch eine Positionierungsregel (111) gefordert – und an der Einbeziehung weiterer Möglichkeiten zur Liniensignaturierung, z.B. der Positionierung von Punktsignaturen an der Linie. Auch diese müssen übergreifend mit den eigentlichem Linienmuster dem Musterausgleich unterzogen werden, was nicht berücksichtig worden war. Die fehlende übergreifende Natur des Musterausgleichs führt im bisherigen SK schon zu Problemen, wenn mehrere Strichlierungsmuster gleichzeitig auszugleichen sind.

Das neue SK-Modell für *LineSymbolizer* versammelt ähnlich wie bei den Punktsignaturen wegen der Wiederverwendbarkeit in anderen Kontexten die eigentliche Definition in der Klasse *Stroke*. Abgeleitet aus *Stroke* ist die bisherige Definition vorgesehen, ausgehend von der abstrakten Klasse *SolidOrDashedStroke*, und eine völlig neue Definition, *CompoundStroke*, welche die bisherigen Defizite beseitigt und neue Möglichkeiten eröffnet.

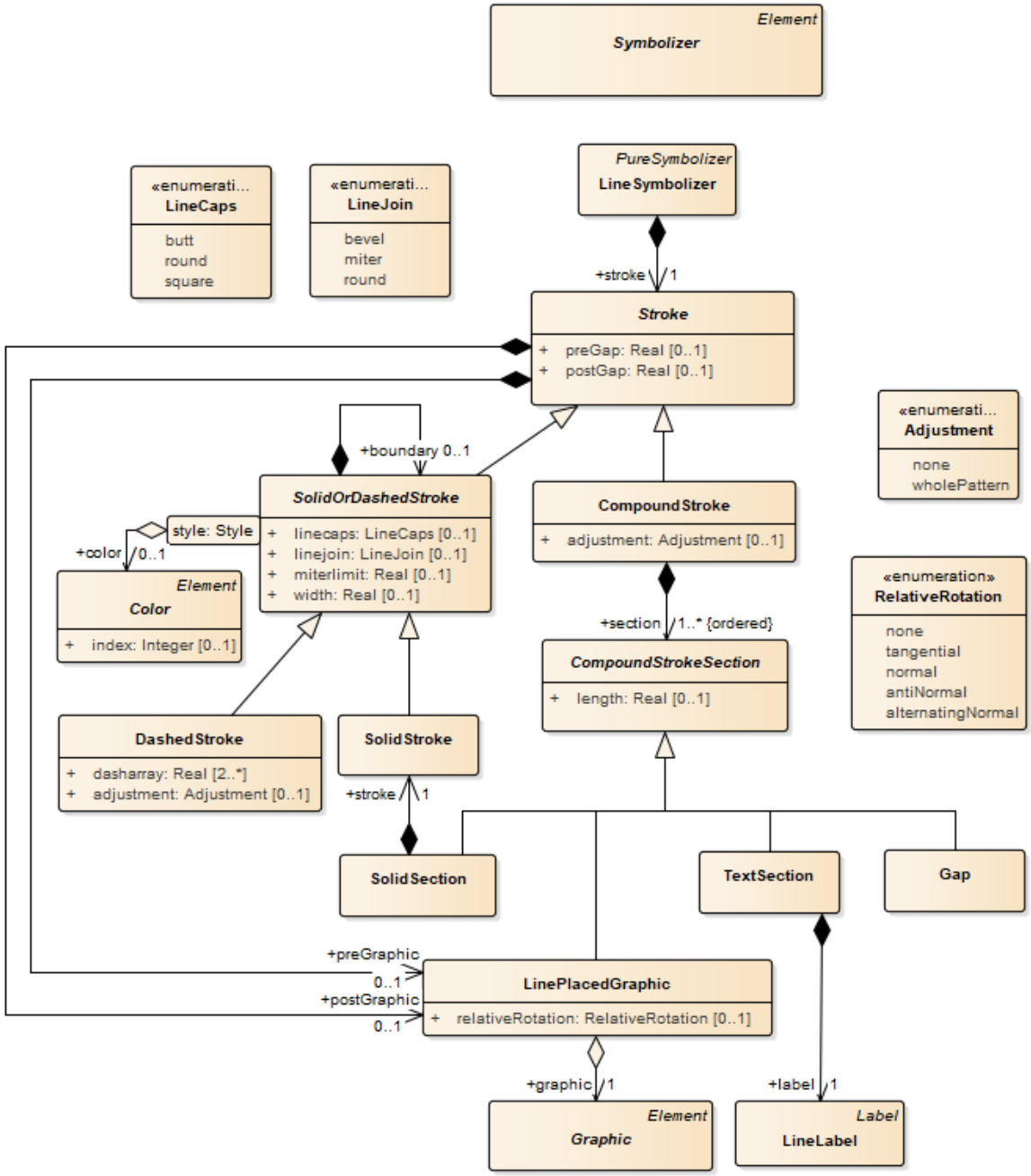


Abbildung 2.7: Liniensignaturen

Der Teil der Modellierung, welcher die bisherige Darstellung im SK nachbildet, unterscheidet die Klassen *SolidStroke* und *DashedStroke*. Ihre gemeinsamen Attribute sind *linecaps*, *linejoin*, *miterlimit* und *width* und *color*. Außerdem erben beide Angaben für einen zusätzlichen Einzug am Anfang und am Ende (*preGap*, *postGap*). Für die besonderen Liniensignaturen, die eine besondere Gestaltung des Rands der Linien aufweisen, wurde die Rolle *boundary* eingeführt, die einer Liniensignatur eine zweite zur Randgestaltung beistellt. Bei *DashedStroke* tritt für die Definition des Musters *dasharray* hinzu. Ein Kreis anstelle eines Strichs, wie es bei manchen Definitionen nötig ist, entsteht durch einen Strich der Länge 0 bei *linecaps=round*. Das Attribut *adjustment* in dieser Klasse ist schon eine Anleihe aus den Positionierungsregeln und entspricht nicht dem bisherigen Stand.

Die neue Definition unter *CompoundStroke* beschreibt die Teile der Liniensignatur als abstrakte *CompoundStrokeSections*-Objekte, welche entweder Linienstücke sind – *SolidSection* – oder Punktsymbole – *LinePlacedGraphic* – oder Texte – *TextSection* – oder Lücken – *Gap*. Der Ausgleich – *adjustment* – erfolgt dabei über die gesamte Definition gleichmäßig.

Zusätzlich erlaubt die neue Definition noch die Angabe einer Punktsignatur am Anfang und am Ende (*preGraphic*, *postGraphic*). Dieses kann wie auch die Punktsignatur im Linienverlauf mit verschiedenen Optionen an den Verlauf der Linie angeglichen werden.

Durch *relativeRotation* kann gesteuert werden, wie die Punktsymbole zu drehen sind:

- *none*: Ungedreht, so wie in der Definition angegeben.
- *tangential*: Das Symbol wird in Richtung der Tangente im Punkt gedreht.
- *normal*: Das Symbol wird in Richtung der Normalen im Punkt gedreht. Die Normale weist nach links vom Linienverlauf.
- *antiNormal*: Wie normal, nur entgegengesetzte Richtung.
- *alternatingNormal*: normal und antiNormal im Wechsel

2.6.3 Flächensignaturen

Flächensignaturen werden im bestehenden SK zwar präzise beschrieben, sind aber auch ausgesprochen arm an Darstellungsmöglichkeiten ausgestattet. Die Möglichkeiten zur Darstellung umfassen nur die Füllung der Fläche mit einer festen Farbe. Beim Einsatz von Flächen für die Konstruktion von Punktsymbolen kann zusätzlich noch die Randlinie beschrieben werden.

Nicht darstellbar im bestehenden SK sind Schraffuren aller Art und die Flächenfüllung mit regelmäßig angeordneten Punktsignaturen. Derartige Signaturen müssen mit Positionierungsregeln textlich beschrieben werden, die wiederum für Schraffen auf Liniensignaturen zugreifen und für Flächenfüllung mit Punktsignaturen auf eben solche. Typischerweise werden dann die Schraffen und Orte für die Punktsignaturen durch linienförmige oder punktförmige Präsentationsobjekte dargestellt.

Die Einschränkung von Flächensignaturen auf einfache Farbfüllung trägt den Möglichkeiten zur automatisierten Signaturierung nicht Rechnung, ist unüblich in Mappingsprachen (siehe z.B. SLD/SE) und würde auch nicht ausreichen, um für die Zukunft den Ansprüchen des Web-SK zu genügen. Der Web-SK kommt fast ohne – Ausnahme: Parallelen zu Linien – Positionierungsregeln aus und benötigt daher (nicht heute, aber wahrscheinlich in Zukunft) ein umfangreicheres Repertoire für die Signaturierung – darunter Flächenfüllung mit Punktsignaturen.

In das SK-Modell wurden deshalb neben der einfachen Füllung mit einer Farbe, die durch die Klasse *SolidFill* gegeben ist, auch Möglichkeiten zur Schraffur und zur Füllung mit

Punktsignaturen eingefügt. Diese sind durch die Klassen *HatchFill* (mit der Abwandlung *HatchFillLongestSide*) und *GraphicFill* gegeben.

Bei der Übernahme der AAA-SKs werden zunächst nur *AreaSymbolizer*-Objekte mit *SolidFill* und ggs. *SolidStroke* oder *DashedStroke* entstehen.

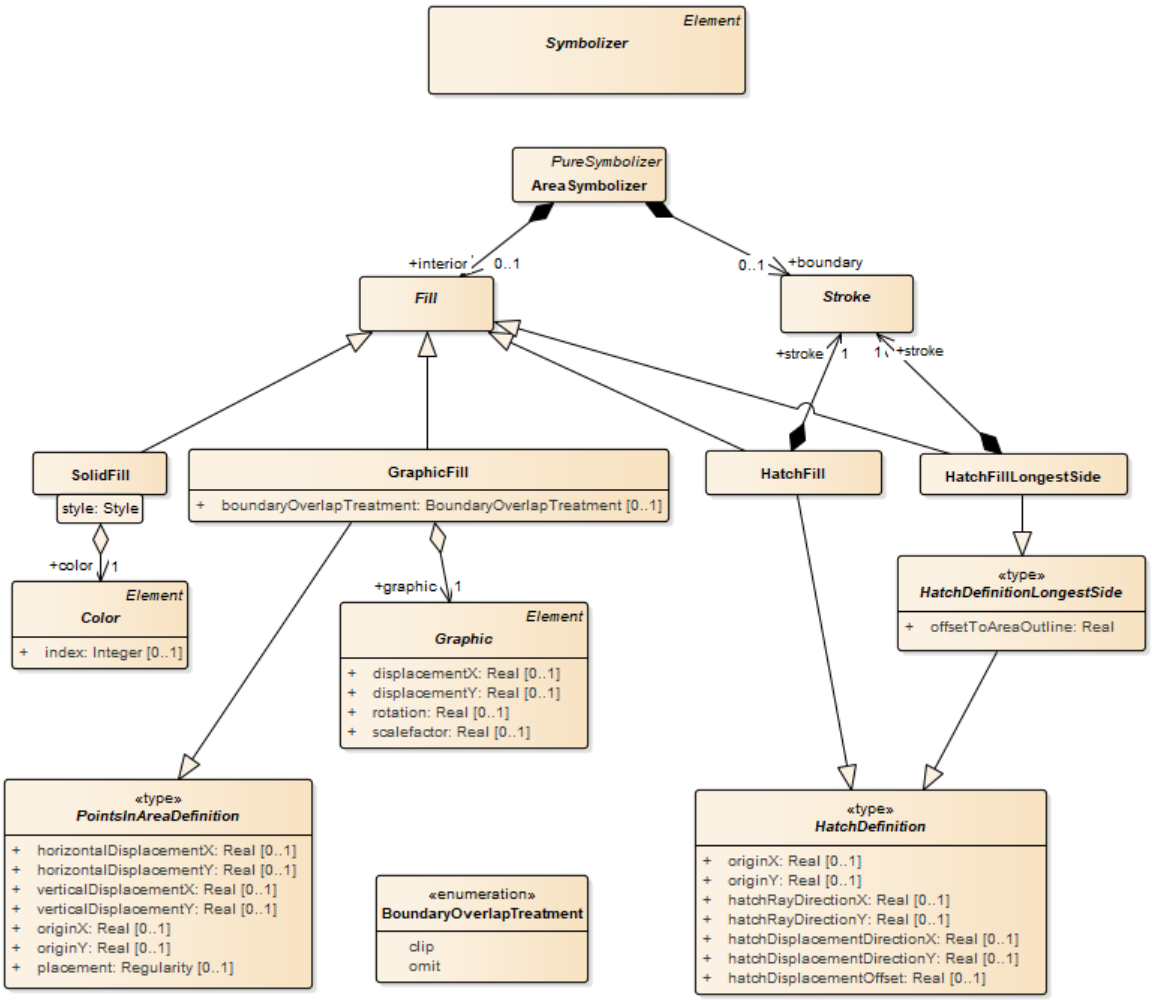


Abbildung 2.8: Flächensignaturen

Will man die bestehenden DTK-Datenbestände durch den weiterentwickelten SK bedienen können, so müssen die alten Definitionen weiterhin Bestand haben. Es muss also weiterhin möglich sein, eine Schraffur durch linienförmige Präsentationsobjekte und einen *LineSymbolizer* auszudrücken. Eine Nutzung der automatisch aus der Fläche erzeugten Schraffen, ist zusätzlich möglich.

Der Zwang, die alte Logik beibehalten zu müssen, ist im Übrigen auch der Grund dafür, dass im Modell der geometrische Anteil der Schraffur und Punktverteilungsdefinition durch Mixin-Klassen (Stereotype <<*type*>>) definiert ist. *HatchDefinition*, *HatchDefinitionLongestSide* und *PointsInAreaDefinition* werden gleichzeitig auch noch bei den neu zu schaffenden präzisen Positionierungsregeln benötigt (sog. *PlacementRules*, siehe Abschnitt

2.5), die beschreiben, nach welchen Regeln die einzelnen Schraffen und Signaturpunkte zu berechnen sind.

2.6.4 Textsignaturen

Im bestehenden SK bezieht sich die Signaturbeschreibung für Texte ausschließlich auf Angaben zur Schrift und zur Farbe. Alle anderen Angaben, etwa zur Position, Ausrichtung, Verschiebung, ja sogar Umbildungen des Texts selbst, sind in textlich ausgeführten Positionierungsregeln vorgegeben.

Um in der Zukunft eine automatisierte Interpretation des SK für Texte zu ermöglichen, wurden im neuen Modell die geometrischen Parameter zur Position und Ausrichtung in die Signaturierungsbeschreibung übernommen. Das Ziel war dabei, Texte im Regelfall ohne weitere Positionierungsinformation direkt mittels der Fachobjektgeometrien platzieren zu können. Auch hier kommen wieder die Anforderungen des Web-SK zum Tragen, welcher fast ohne explizite Positionierungsregeln auskommt.

Ähnlich wie bei den anderen *Symbolizer*-Klassen bezieht auch der *TextSymbolizer* seine wesentliche Information aus einem weiteren Objekt namens *Label*, welches auch in anderen Signaturierungszusammenhängen eingesetzt wird, beispielsweise bei Punkt- und Liniensignaturen. Die Klasse *Label* bündelt bereits die meisten textbezogenen Präsentationsattribute. Spezielle Klassen zur Platzierung von Text am Verlauf von Linien (*LineLabel*) und an Punkten (*PointLabel*) ergänzen diese um spezielle weitere Attribute.

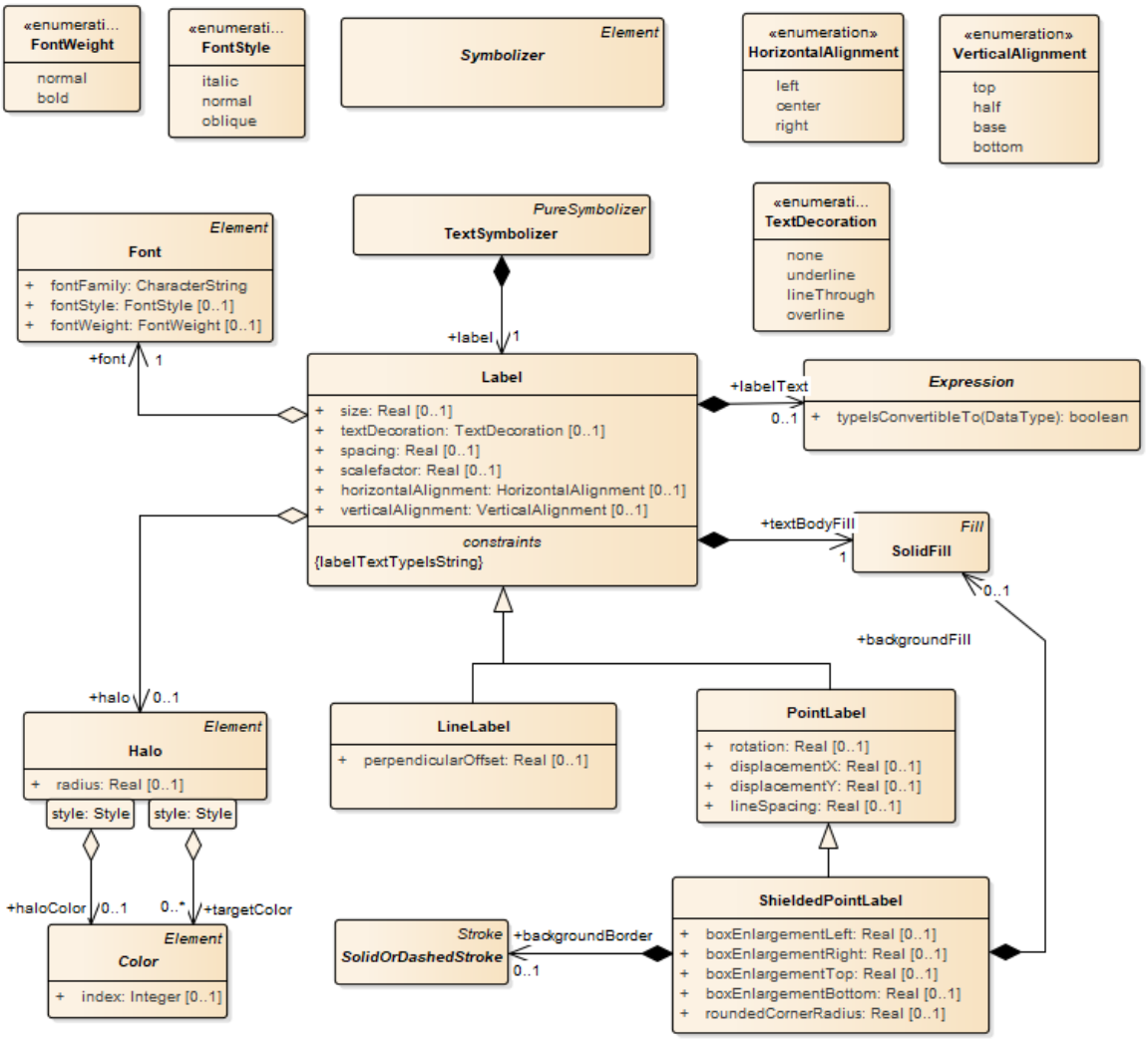


Abbildung 2.9: Textsignaturen

Die Steuerung der Textausgabe ermöglicht die Angabe des Texts selbst als *Expression*, seine Größe, mögliche Unterstreichung, die Sperrung des Texts, seine Skalierung und seine Ausrichtung. Die Farbe wird über *SolidFill* ausgewählt. Die typischen Font-Parameter sind in einem angeschlossenen *Font*-Objekt zusammengefasst. Der Textinhalt als *Expression* (Vorgabe über *Emit*, überschreibbar durch *Label* mit Rolle *labelText*) ermöglicht es auf einfache Weise, sowohl Gestaltungsmerkmale wie Großschreibung (Funktion *fn:upper-case*) als auch „Positionierungsregeln“ wie 611 bis 616 auszudrücken.

Label ist deshalb als konkrete Klasse angelegt, um Kompatibilität mit dem bisherigen AAA-SK zu ermöglichen. Ursprünglich sollte Label abstrakt sein und nur durch die speziellen Klassen LineLabel und PointLabel vertreten werden.

Eine Erweiterung im Hinblick auf den Web-SK ist *ShieldedPointLabel*, welches dazu eingesetzt werden kann, Texte mit hinterlegtem rechteckigem Umriss auszugeben. Dabei kann die Ausgestaltung runder Ecken verlangt werden.

Auch wurde bei Texten (gleichfalls auch bei Graphics) die Ausgabe eines Freistellungsgebiets (*Halo*) steuerbar gemacht.

Die genaue Funktionsweise der Definition im *Halo*-Objekt hängt davon ab, ob Farben für *haloColor* und *targetColor* angegeben werden und welche.

Die Prinzipien der Wirkungsweise der beiden Angaben sind wie folgt:

- Über *haloColor* kann eine Farbe angegeben werden, welche dann tatsächlich als Saum (mit dem gewünschten radius) erscheint. Wird keine *haloColor* spezifiziert, so wird ein durchsichtiger Saum erzeugt.
- Wenn *targetColor* nicht angegeben wird, so wirkt *haloColor* gegen alle Farben der Karte. Werden dagegen für das multiple Property *targetColor* Farben angegeben, so wirkt die Saumbildung nur gegen diese angegebenen Farben.

Insgesamt ergeben sich folgende Möglichkeiten:

haloColor gesetzt, aber keine *targetColor*: Dies ist die normale Halobildung, bei der einfach ein Saum um die Texte/Graphics gelegt wird.

Weder *haloColor*, noch *targetColor*: Durchsichtiger Saum, der den Bildhintergrund durchlässt.

Restliche Fälle: Wenn für *targetColor* spezifische Farben gesetzt sind, so erfolgt die Saumbildung nur gegen diese Farben. Gegen Bildteile mit anderen Farben erfolgt keine Freistellung. Die Saumbildung erfolgt mit der *haloColor*, wenn diese vorliegt. Liegt sie nicht vor, so werden die *targetColors* durchsichtig gestellt.

Die Anbindung der Farben (*haloColor*, *targetColor*) unterliegt der „qualifizierten Assoziation“ nach Style. Halo kann bei entsprechender Definition in *SimpleGraphic* und *CompositeGraphic* kumulativ wirken (siehe Beschreibung zu *CompositeGraphic*).

2.7 Farbdefinitionen

Farbe wird im Modell entweder durch die Klasse *ColorCMYK* (für Farbdefinitionen im CMYK-System) oder durch die Klasse *ColorRGB* (für RGB-Farben) vertreten. Beide Klassen werden durch abstrakte Klasse *Color* zusammengefasst, welche im Modell referiert wird. Zusätzlich wird noch die Klasse *NullColor* bereitgestellt, um explizit dokumentieren zu können, dass bestimmte Signaturteile nicht dargestellt werden sollen.

Als einziges Attribut weist die Color *index* auf, welches für einen eindeutigen Farbindex in einer Farbtabelle steht. Er wird als zusätzliches Identifizierungsmerkmal vergeben und kann als Vorgabe für eine Kanalzuordnung bei einer automatisch abzuleitenden Pseudofarbtabelle für Rasterdateien verwendet werden.

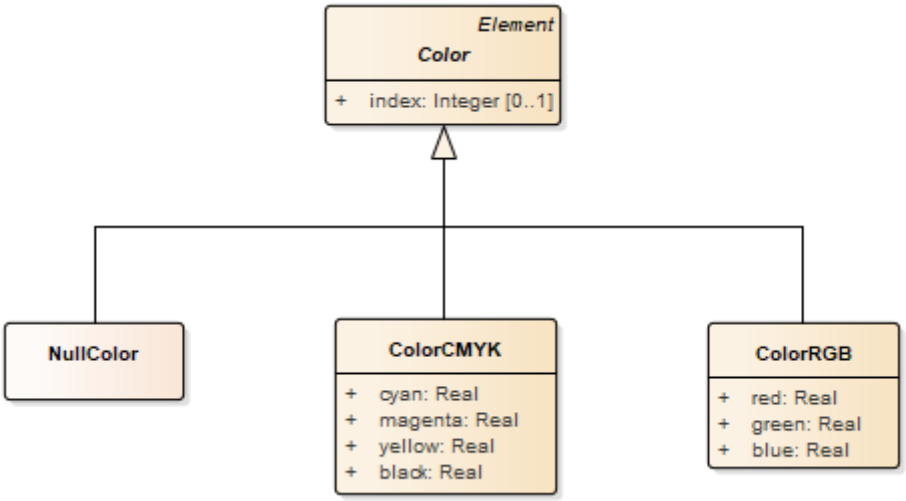


Abbildung 2.10: Farbe

Die *Color*-Objekte unterliegen grundsätzlich dem Einfluss der *Style*-Objekte. Sie sind alle über eine mit *Style*-Objekten qualifizierte Assoziation angebunden, welche die Auswahl der für den betreffenden Style gewünschten Farbe vornimmt.

Beispiel: Nutzung von *Color* durch das Objekt *SolidFill* …

Das *Color*-Objekt ist über die Rolle *color* der Kardinalität [1] qualifiziert an *SolidFill* angebunden. Das bedeutet, dass unter einem gewählten Style jeweils genau ein *Color*-Objekt gültig ist. Die Qualifizierung ist optional, d.h. sie kann auch entfallen. Entfällt sie, so ist nur **ein** *Color*-Objekt überhaupt zulässig.

Das bedeutet folgende Alternativen:

- Ein *SolidFill* besitzt genau ein *Color*-Objekt, wenn die *Style*-Qualifizierung entfällt, oder
- ein *SolidFill* besitzt mehrere *Color*-Objekte, die alle über eine eindeutige Qualifizierung durch *Styles* angebunden sind, die sich unterscheiden und mit einem der *Styles* im Layer übereinstimmen.

2.8 Geometrien für Punktsignaturen

Die Signaturen für Punkte liegen im bisherigen SK bereits ausgearbeitet vor. Sie werden durch relative Bildkoordinaten im 1/100 mm-System als einzelne Figurenteile bestehend aus Flächen und Linien beschrieben. Das Modell sieht eine strukturell identische Übernahme dieser Strukturen vor.

Da nach der Erstübernahme eine weitere Pflege der Punktsignaturen über SVG-Hilfsmittel geplant ist, wurde das Geometriemodell an SVG angelehnt. Um die Umsetzung zu erleichtern, wurde jedoch nur ein kleiner Ausschnitt der SVG-Path-Sprache verwendet.

Das Geometriemodell für die Punktsignaturen ist im folgenden Diagramm dargestellt.

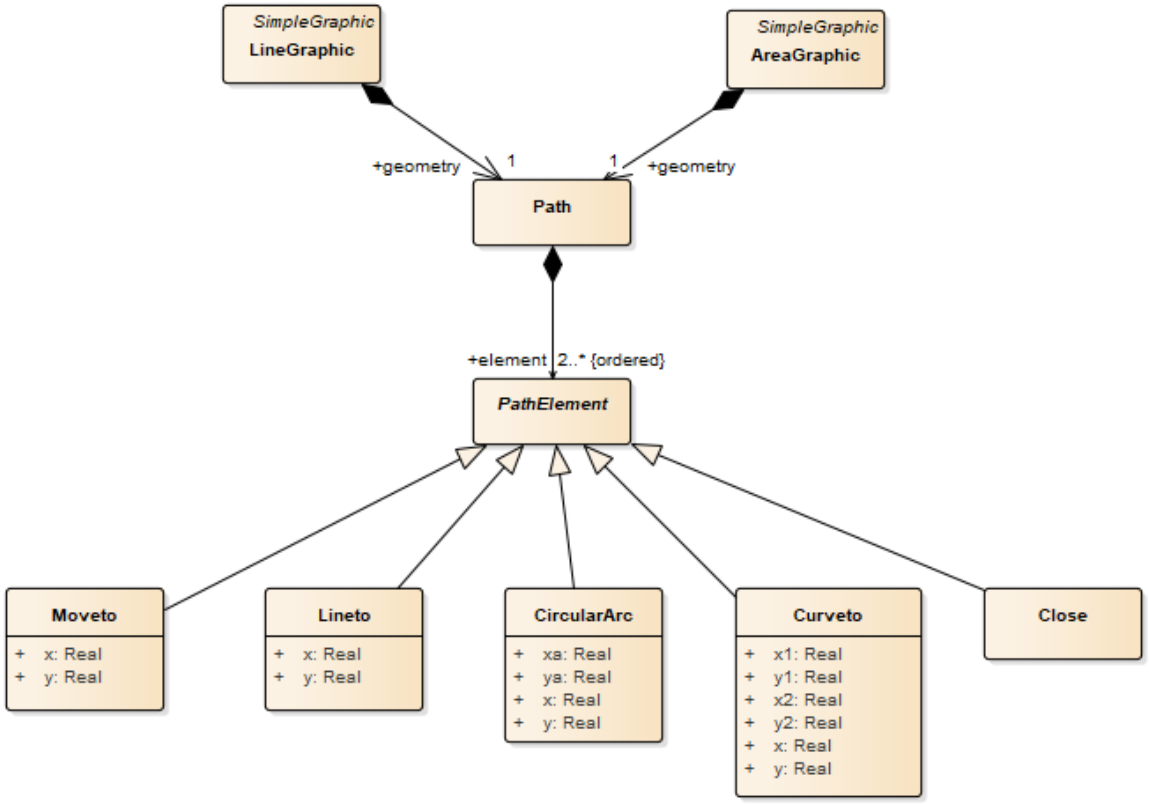


Abbildung 2.11: Geometrien für Punktsignaturen

Die Geometrien sind in einem *Path*-Objekt als Container zusammengefasst, dessen Elemente (mit einer Ausnahme) die SVG-Path-Sprache nachbilden.

Die abstrakte Klasse *PathElement* bündelt die der SVG-Spezifikation nachgebildeten Beschreibungselemente für Pfade. Pfade bestehen aus einer oder mehreren Pfadkomponenten, deren letztgesetzter Punkt, der laufende Punkt, immer Ausgangspunkt des nächsten anzuhängenden Geometriestücks ist.

Dabei setzt *Moveto* den gegebenen Punkt als den Beginn der laufenden Pfadkomponente und darin als laufenden Punkt. *Lineto* schließt an den laufenden Punkt ein Geradenstück an, welches bis zum Endpunkt (*x*,*y*) verläuft. Dieser Punkt wird nach Anschluss des Geradenstücks zum laufenden Punkt. *Close* erzeugt eine gerade Linie vom laufenden Punkt zum Beginn der laufenden Pfadkomponente. Der Beginn der laufenden Pfadkomponente wird dadurch wieder laufender Punkt.

CircularArc ist nicht dem SVG-Standard nachgebildet und stellt eine einfache Beschreibungsmöglichkeit für Kreisbögen dar. Der Kreisbogen verläuft über den Punkt (*xa*,*ya*) bis zum Endpunkt (*x*,*y*). Dieser Punkt wird nach Abschluss des Bogens zum laufenden Punkt.

Curveto ist wieder gemäß dem SVG-Standard modelliert und entspricht kubischen Beziers. (*x1*,*y1*) ist der Kontrollpunkt am laufenden Punkt und (*x2*,*y2*) derjenige am Endpunkt, welcher durch (*x*,*y*) gegeben ist.

Hinweis: Ellipsenbögen wurden wegen ihrer unanschaulichen Definition absichtlich ausgelassen. Sie sollten, soweit sie auftreten, durch Beziers approximiert werden.

Die *Path*-Objekte versorgen die Objekte *LineGraphics* und *AreaGraphics* mit Geometrie. Für *AreaGraphics* müssen alle Pfadkomponenten durch *Close* abgeschlossen werden.

Soweit die Modellierung im SK-Modell.

Die Umsetzung in die beiden Repräsentationen MDL und SK-XML erfolgt ähnlich, aber im Detail etwas unterschiedlich.

In MDL wird unmittelbar der Anforderung Rechnung getragen, dass die Erfassung der Geometrien in SVG [SVG] erfolgen soll. Deshalb werden die Objekte des Modells in MDL-Objekte übersetzt, die direkt eine Geometrierepräsentation in SVG-<*path d*="…"/>-Syntax enthalten.

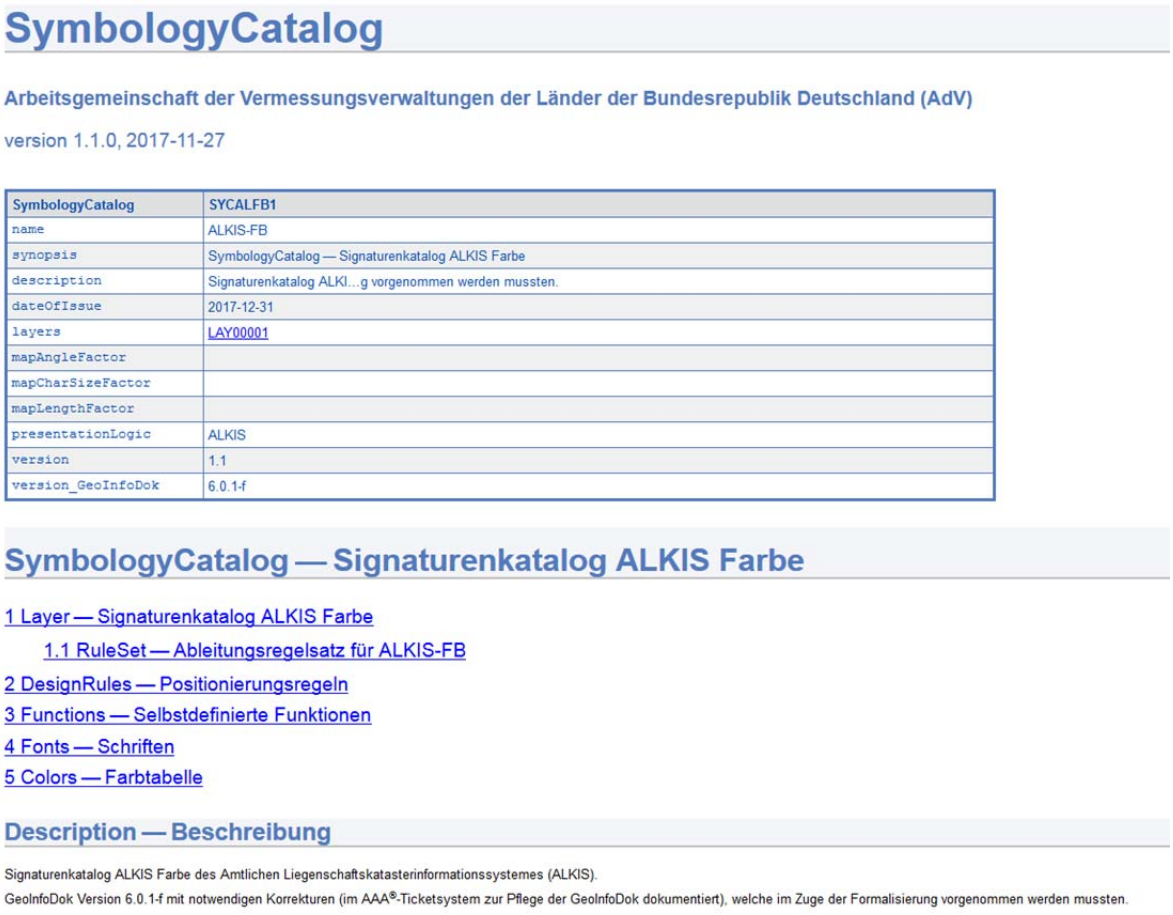
In SK-XML werden die im Modell vorgegebenen Klassen entsprechend den in Kapitel 4 dargestellten Abbildungsregeln umgesetzt und erscheinen als XML-Elemente.

3 Beschreibung der Dokumentation der AAA-Signaturenkataloge

Die SK-Dokumentation wird automatisch aus den in MDL beschriebenen Signaturenkatalogen abgeleitet. Das Dokument liegt aktuell im HTML-Format vor und gliedert sich in mehrere Abschnitte:

- Kataloginformationen
- Layerinformationen
- Regelsatzbeschreibungen mit Ableitungsregeln, die sowohl den Filterausdruck für die Ableitung aus dem Landschaftsmodell enthält, als auch die Überführung in die Signatur mit deren grafischer Ausprägung
- Beschreibungen der Positionierungsregeln, selbstdefinierten Funktionen, verwendeten Schriftarten sowie der Farbtabelle(n)

Über die Startseite (*SymbologyCatalog*) gelangen Sie zu den o. g. Beschreibungen:



SymbologyCatalog

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

version 1.1.0, 2017-11-27

SymbologyCatalog	SYCALFB1
name	ALKIS-FB
synopsis	SymbologyCatalog — Signaturenkatalog ALKIS Farbe
description	Signaturenkatalog ALKI…g vorgenommen werden mussten.
dateOfIssue	2017-12-31
layers	LAY00001
mapAngleFactor	
mapCharSizeFactor	
mapLengthFactor	
presentationLogic	ALKIS
version	1.1
version_GeoInfoDok	6.0.1-f

SymbologyCatalog — Signaturenkatalog ALKIS Farbe

1 Layer — Signaturenkatalog ALKIS Farbe
1.1 RuleSet — Ableitungsregelsatz für ALKIS-FB
2 DesignRules — Positionierungsregeln
3 Functions — Selbstdefinierte Funktionen
4 Fonts — Schriften
5 Colors — Farbtabelle

Description — Beschreibung

Signaturenkatalog ALKIS Farbe des Amtlichen Liegenschaftskatasterinformationssystemes (ALKIS).
GeoInfoDok Version 6.0.1-f mit notwendigen Korrekturen (im AAA®-Ticketsystem zur Pflege der GeoInfoDok dokumentiert), welche im Zuge der Formalisierung vorgenommen werden mussten.

Abbildung 3.1: Startseite des ALKIS-SK Farbe

3.1 Katalogsbeschreibungen (SymbologyCatalog)

SymbologyCatalog:

Die Katalogsbeschreibung fasst die Metadaten für ein gesamtes Katalogwerk zusammen.
Inhalte sind u. a.:

name	Name des Signaturenkatalogs, z. B. „Signaturenkatalog DTK 1:50000“.
Description	Beschreibung des Signaturenkatalogs.
Layers	Verweis auf die Ids der eingebundenen Layer (Kartenebenen), z. B. „LAY00001“.
mapAngleFactor (optional)	Auf alle Winkelgrößen im SK anzuwendender Faktor bezogen auf das Bogenmaß (bei Altgrad beträgt dieser Faktor 0,017 [π/180], bei Neugrad 0,0157 [π/200]).
mapCharSizeFactor (optional)	Auf alle Textgrößen im SK anzuwendender Faktor bezogen auf die Einheit 1mm.
mapLengthFactor (optional)	Auf alle Längenmaße im SK anzuwendender Faktor bezogen auf die Einheit 1mm.
presentationLogic	Anzuwendende Logik für die Behandlung der Präsentationsobjekte. Die SKs von ALKIS, ATKIS und Web-SK unterscheiden sich in dieser Logik (implizite und explizite Ableitung von Präsentationsobjekten, siehe Abschnitt 3.3.3 [Art der Präsentation]). Wert z. B. „ATKIS“.
Version	Versionierung der Ableitung des SK, z. B. „1.0“.
version_GeoInfoDok	Version der GeoInfoDok des, z. B. „6.0.1“.

3.2 Layerbeschreibungen (Layer)

In die Übersicht der Layer gelangen Sie über die Startseite (*SymbologyCatalog*) und den darauf befindlichen Link „Layer – Signaturenkatalog A…“.

Layer

Ein Layer steht für eine Kombination von Regeln für definierte Maßstäbe oder Maßstabsbereiche, welche eine fachliche Einheit bilden.
Inhalte sind u. a.:

name	Name des Layers, z. B. „ALKIS-FB“.
Description	Beschreibung des Layers.
ruleSets	Verweis auf die Ids der eingebundenen Regelsatzes, z. B. „RST00001“.
Styles	Ein Style steht für eine bestimmte Ausprägung einer Layerdefiniton, z. B. die Ausprägungen eines Layers als Farb- oder Graustufenkarte.

3.3 Ableitungsregelsatz (RuleSet) und Ableitungsregeln (Rules)

In die Übersicht des RuleSet und der Ableitungsregeln gelangen Sie über die Startseite (*SymbologyCatalog*) und den darauf befindlichen Link „RuleSet – Ableitungsregelsatz für A…“ oder über die Seite *Layer*.

RuleSet:

Ein Regelsatz fasst Ableitungsregeln (Rules) zusammen, die für einen gemeinsamen Maßstabsbereich vorgesehen sind. Dieser wird durch die Properties minScaleDenominator und maxScaleDenominator definiert. Die enthaltenen Rules werden ebenfalls gelistet.

minScaleDenominator	Unterer Wert für die Maßstabsgrenze, für welche die enthaltenen Rules gelten sollen, z. B. „50000“ (für die DTK50).
maxScaleDenominator	Oberer Wert für die Maßstabsgrenze, für welche die enthaltenen Rules gelten sollen, z. B. „50000“ (für die DTK50).
Rules	RUL00010 RUL00020 RUL00021 …

Rules

Anders als in früheren Versionen der Signaturenkataloge bieten die Rules (Ableitungsregeln) in der vorliegenden Version eine leicht zu lesende und durch die Umsetzung der grafischen Ausgabe untermalte Nutzeransicht, die durch Verlinkung der entsprechenden Elemente für Entwickler relevante Inhalte offenlegt.
Z. B. **RUL00001** (Festgesetztes Überschwemmungsgebiet):

Abbildung 3.2: Beispiel einer Rule aus dem ALKIS-SK Farbe (Ableitungsregel)

Neu ist in diesem Zusammenhang die Aufnahme der sog. freien Symbolizer in formale Ableitungsregeln. Freie Symbolizer sind die Signaturen, die bisher in den ATKIS-Dokumenten „Signaturen“ (Abschnitt 8.2.x.3) und „Kartenrahmen, Titelei, Falzung“

(Abschnitt 8.2.x.8) aufgeführt waren, jedoch keine Ableitungsregel besaßen (freie Präsentationsobjekte). Im ALKIS-SK sind auf diese Weise die Signaturnummern des Teil C („Präsentationen“) eingebunden worden.

Diese Symbolizer erhalten nun einfache Regeln, damit sie später in die offizielle Ausgabesprache SK-XML überführt werden können. Die Regeln für freie Symbolizer finden sich am Ende der Ableitungsregeln. Sie sind gekennzeichnet durch das Regelpräfix „RUS“ mit angehängter Signaturnummer des Symbolizers:

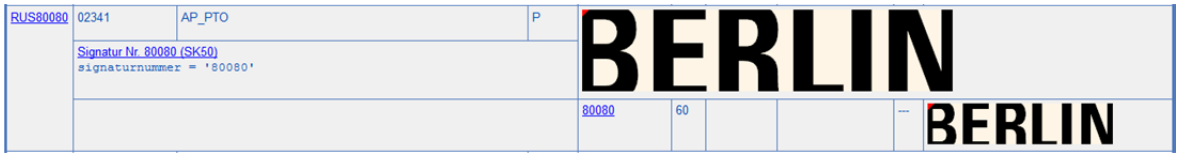


Abbildung 3.3: Beispiel einer Rule aus dem ATKIS-SK50 mit freiem Symbolizer

3.3.1 Spalte „RuleID“

Die Spalte „RuleID“ enthält die eindeutige Nummer der Ableitungsregel. Diese entspricht der bisherigen Nummerierung der Ableitungsregeln in den ATKIS-SKs (ohne die Unternummer [z. B. „.0“], da diese durch die Neuformulierung der Filterausdrücke entfallen konnte [ZUSO-REO-Beziehungen oder REO-REO-Beziehungen werden über den an Xpath angelehnten Filterausdruck ausgegeben]). In den ALKIS-SKs wurde diese analog dazu ergänzt.

Die *RuleID* ist verlinkt. Beim Klick auf **RUL00001** öffnet sich die Ansicht auf die der Regel zugrundeliegende UML-Struktur:

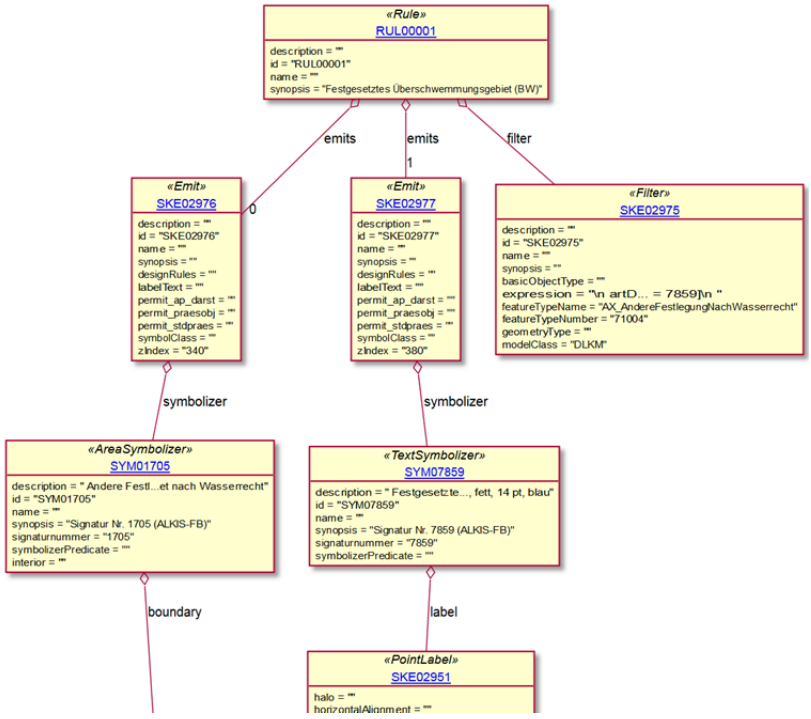


Abbildung 3.4: UML-Instanz einer Rule

Die baumartige Struktur verzweigt sich zunächst in die beiden Hauptbereiche einer Ableitungsregel:

- den *Filter* (die Modelleingabe)
- den oder die *Emit(s)* (die Signaturausgabe)

Letztere verzweigt sich weiter in die verschiedenen Elemente für die Signaturbeschreibung.

Ausgehend von diesem UML-Modell können Sie für jedes Element die Informationen über die MDL-Struktur einsehen.
Z. B. erhalten Sie beim Klick auf den Filter **SKE02975** die Details zum Filter:

```
Filter \
   -id                SKE02975 \
   -expression        {
      artDerFestlegung and artDerFestlegung = 1441 and inversZu_dientZurDarstellungVon_AP_PPO/AP_PPO
   } \
   -featureTypeName   AX_AndereFestlegungNachWasserrecht \
   -featureTypeNumber 71004 \
   -modelClass        DLKM
```

Abbildung 3.5: MDL-Syntax des Elements Filter

3.3.2 Filter-Bereich der Rule

Der linke Spalten-Bereich (rechts von der RuleID) enthält den eigentlichen Filter zur Ableitung der Selektionsmenge aus dem Ausgangsdatenmodell.

	RuleID	OAKennung	Objektart	GTyp
		Filterausdruck		
		Filterausdruck für abfrageabhängige Signaturierung		
[1]	RUL00001	71004	AX_AndereFestlegungNachWasserrecht	
[2]		Festgesetztes Überschwemmungsgebiet (BW) artDerFestlegung and artDerFestlegung = 1441 and inversZu_dientZurDarstellungVon_AP_PPO/AP_PPO and inversZu_dientZurDarstellungVon_AP_PPO/AP_PPO[art = 'ADF'] and inversZu_dientZurDarstellungVon_AP_PTO/AP_PTO[signaturnummer = 7859]		
[3]				
[4]				

Abbildung 3.6: Filter-Bereich der Rule

Er enthält

- Zeile [1] (*OAKennung, Objektart, GTyp)*: die Objektartenkennung (5-stellige Nummer der Objektart), den Objektartennamen sowie deren Geometry-Typ. In ALKIS und ATKIS: *P* für punktförmig, *L* für Linienförmig, *F* für Flächenförmig. Im Falle eines ZUSO oder NREO bleibt diese Zelle leer. In AFIS bleibt diese Zelle ebenfalls leer.
- Zeile [2] (*Filterausdruck*): den eigentlichen Filterausdruck. Dieser umfasst eine Zusammenfassung (*Synopsis*) der Inhalte des Filterausdrucks in Prosatext. Dabei dient der Pipe („|“) als Verkettung von Abfragen unterschiedlicher Attribute und Sachverhalte und das einfache Komma („,“) als Abfrage aus einer Liste von Werten eines Attributs. Ist kein Filterausdruck vorhanden und es wird die Objektart generell abgeleitet, erscheint nur die ID des Filter-Elements (SKE…). Direkt unter der Synopsis wird der eigentliche MDL-Filterausdruck aufgeführt.
- Zeilen [3]ff (*Filterausdruck für abfrageabhängige Signaturierung*): Dieser Bereich ist für eine spätere SK-Version reserviert und deshalb derzeit leer. Künftig sollen die Abfragen von Flächengrößen oder Längen abhängigen Signaturen (i. d. R. Schriften, die mittels der Funktionen atk:FLB und atk:LGO abgeleitet werden) im Symbolizer aufgerufen werden und nicht mehr über die Ableitungsregel selbst (vgl. Abschnitt 2.4, *symbolizerPredicate*). Dann steht in diesen Feldern der jeweilige Ausdruck zur Signaturnummer.

Auf die Angabe des Objekttyps (REO) wurde in der Dokumentation bewusst verzichtet, da er sich aus dem Inhalt des Geometrietyps ableiten lässt (*P*, *L* und *F* sind immer ein REO). Im Falle eines ZUSOs oder NREOs bleibt die Zelle leer.

3.3.3 Emit-Bereich der Rule

Der Rechte Bereich der Rule, der sog. *Emit*, gibt an, wie die über den *Filter* selektierten Objekte signaturiert werden.

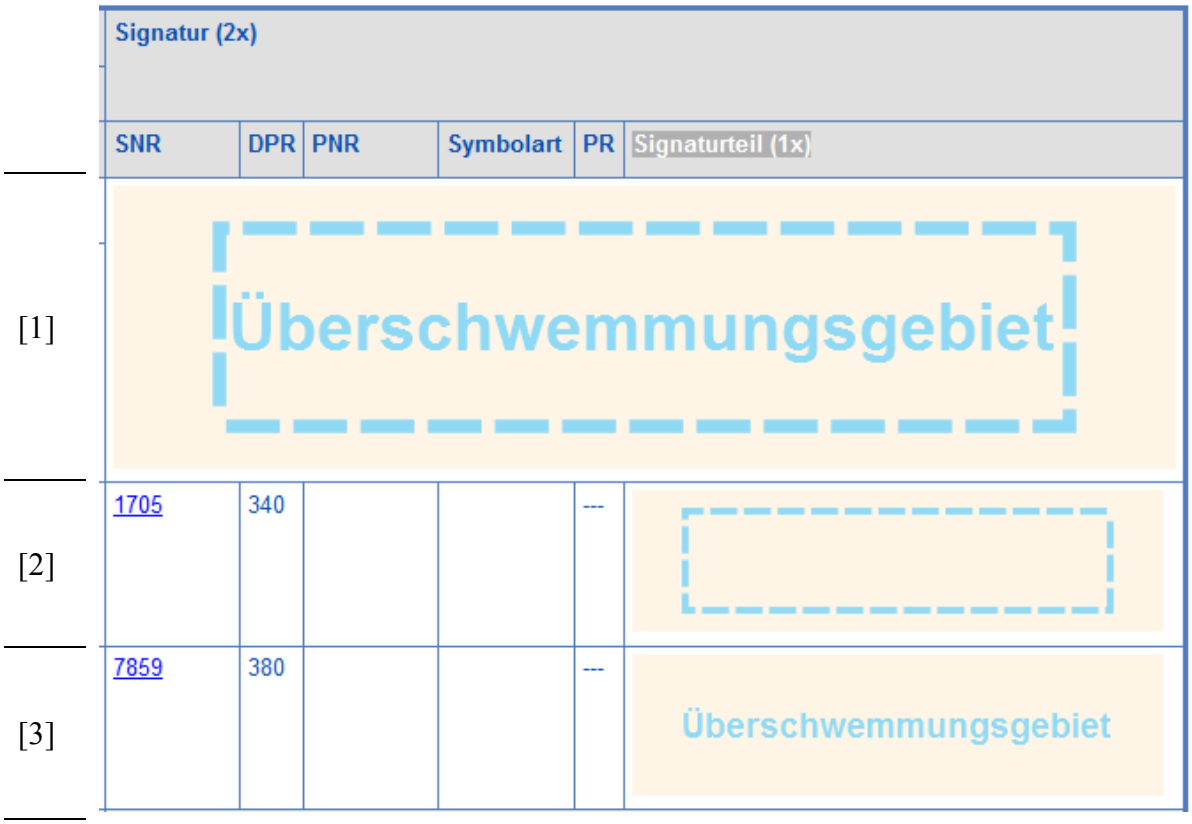


Abbildung 3.7: Emit-Bereich der Rule

Er enthält:

Zeile [1] (Signatur [x-fach]): Die Abbildung der zusammenesetzten Signatur aus der Rule in x-facher Vergrößerung. Diese Spalte löst gleichzeitig die frühere Spalte „SIT“ im ATKIS-SK ab, die den Schriftinhalt enthielt. Dieser wird jetzt direkt in der Darstellung umgesetzt.Was die Umsetzung der Signaturen betrifft, befinden wir uns in der aktuellen Version noch im Entwurfsstadium. Nicht umgesetzt wurde bisher die Darstellung mittels Positionierungsregeln. Letztere werden erst umgesetzt, wenn die Formalisierung über Platzierungsattribute (PlacementRules) stattgefunden hat.

Hinweis zum Schriftinhalt:

Bei der Signaturierung von Schriften wird entweder ein fester Schriftinhalt angegeben (z. B. „Deponie“) oder der Inhalt des Attributs ausgewertet (z. B. der Inhalt der Attributart name, zweitname, bezeichnung).

Für Datenproduzenten gilt: Hat ein Objekt mehrere Namen, Zweitnamen oder Bezeichnungen, so ist zwingend beim textförmigen Präsentationsobjekt der Schriftinhalt

bei der Attributart SIT zu speichern. Ansonsten wäre der Schriftinhalt des textförmigen Präsentationsobjektes nicht mehr eindeutig.

In der grafischen Umsetzung unterscheidet sich der feste von einem Attribut-abhängigen Schriftinhalt durch ein kleines rotes Dreieck in der linken oberen Ecke:

Fester Schriftinhalt *Attribut-abhängiger Schriftinhalt*

Abbildung 3.8: Fester vs. Attribut-abhängiger Schriftinhalt

- Zeile [2]ff (*SNR*, *DPR*, *PNR*, *Symbolart*, *PR*, *Signaturteil [x-fach]*): Die Beschreibung der Ausgabe-Parameter der Signatur. Objekte, welche die Bedingungen in den vorgenannten Spalten erfüllen, erhalten die Signatur (*Symbolizer*) mit den nachfolgend beschriebenen Parametern:

Signaturnummer (*SNR*)

Die Signatur ist als Symbolizer in MDL genau beschrieben. Die Signaturnummer dient somit als Verknüpfungsschlüssel zwischen den Ableitungsregeln und dem Symbolizer. Jeder Signaturenteil erhält eine eigene Signaturnummer.

Beim Klick im UML-Modell auf die Signaturnummer **1705** (*AreaSymbolizer*) erhalten Sie folgende MDL-Ansicht:

```
      AreaSymbolizer \
         -id             SYM01705 \
         -synopsis       "Signatur Nr. 1705 (ALKIS-FB)" \
         -description    {
Andere Festlegung nach Wasserrecht
Schutzgebiet nach Wasserrecht
         } \
         -signaturnummer 1705 \
         -boundary       {
            DashedStroke \
               -postGap    50.00 \
               -preGap     50.00 \
               -color      {
                  ColorRGB \
                     -id    COL00024 \
                     -name  Blau2 \
                     -blue  96.00 \
                     -green 85.00 \
                     -red   56.00
               } \
               -linecaps   butt \
               -linejoin   miter \
               -width      100.00 \
               -adjustment wholePattern \
               -dasharray  "400 100"
         }
   } \
```

Abbildung 3.9: MDL-Definition eines Symbolizers

Sie enthält alle Beschreibungen für die Ausprägung der Signatur 1705, z. B. *width* (Strichbreite).

Darstellungspriorät (*DPR*)

In dieser Spalte wird die Darstellungspriorität (DPR oder in MDL: *zIndex*) der in der Spalte „SNR“ angegebenen Signaturnummer angegeben. Signaturen mit hoher Darstellungspriorität liegen über Signaturen mit niedriger Darstellungspriorität (vgl. auch Abschnitt 3.9). Sind mehrere Darstellungsprioritäten angegeben, so bezieht sich die DPR-Zahl jeweils auf die Signaturnummer der Spalte „SNR“, die in derselben Zeile steht.

Positionierungsregel (*PNR*)

In dieser Spalte können Nummern von Positionierungsregeln (*DesignRules*) angegeben sein, wenn dies für die Signaturierung erforderlich ist.

Positionierungsregeln beschreiben genaue Angaben z. B.

- zur Bemusterung von Linien oder Flächen
- Orientierung von Schriften oder Symbolen
- Angabe des geometrischen Ortes zur Platzierung von Signaturen (z. B. in den Schwerpunkt einer Fläche)
- Angaben über den Schriftinhalt usw.

Sind mehrere Positionierungsregeln angegeben, so bezieht sich die Positionierungsregel jeweils auf die Signaturnummer der Spalte „SNR“, die in derselben Zeile steht.

Die richtige Signaturierung ergibt sich erst durch Anwendung der Signatur selbst, der Darstellungspriorität und, falls notwendig, der Positionierungsregel.

Symbolart

Derzeit nur in ATKIS geführt! Die Angabe der Symbolart (Attribut *ART* bei Präsentationsobjekten) gibt die Kennung des Attributs an, das mit dem Präsentationsobjekt dargestellt werden soll. Wenn mehrere Eigenschaften eines Objekts in einem Präsentationsobjekt dargestellt werden sollen, beschreibt der Wert des Attributs *ART*, um welche Darstellungsanteile es sich bei dem Präsentationsobjekt handelt (z. B. *Name_FKT*).

Art der Präsentation (*PR*)

Diese gibt die Ausgabeform als Standardpräsentation oder Präsentationsobjekt an bzw. lässt die Möglichkeit zu, die Standardpräsentation überzudefinieren mittels des NREO AP_Darstellung).

Dabei steht das erste „x/-“ für *AP_Darstellung*, das zweite für „Präsentationsobjekt“ (PO) und das dritte für die „Standardpräsentation“, z. B.:

„x-x“ bedeutet „wird als Standardpräsentation umgesetzt, *AP_Darstellung* ist erlaubt“.

„-x-“ bedeutet „wird als Präsentationsobjekt umgesetzt“.

„—x“ bedeutet „wird nur als Standardpräsentation umgesetzt“.

„xxx“ bedeutet „kann sowohl als Standardpräsentation mit *AP_Darstellung* als auch als Präsentationsobjekt umgesetzt werden“.

Diese Spalte ist ATKIS-spezifisch, da sich ATKIS von AFIS und ALKIS dahingehend unterscheidet, dass die Erzeugung von Präsentationsobjekten (PO) implizit passiert. D. h. die POs werden nicht explizit abgeleitet (analog einem Fachobjekt wie AX_AndereFestlegungNachWasserrecht), sondern es wird bei Belegung des Feldwerts für Präsentationsobjekte mit einem „x“ impliziert, dass hier eine PO erzeugt wird. Da bei AFIS und ALKIS die POs explizit abgeleitet werden, wird die Spalte immer mit „---“ belegt.

Erläuterungen zu AP_Darstellung, Präsentationsobjekt, Standardpräsentation

AP_Darstellung

Durch das Erzeugen eines NREO *AP_Darstellung* kann

- die Standardsignatur
- die Darstellungspriorität
- die Positionierungsregel

explizit gespeichert werden. Es ist aber auch möglich, die Standardwerte, die in den Ableitungsregeln vorgesehen sind, zu ändern und die Attributwerte beim NREO *AP_Darstellung* mit anderen Werten zu belegen.

Präsentationsobjekt

Präsentationsobjekte sind ausschließlich für die Präsentation erzeugte Objekte, z. B. Punkte oder Linien für Kartenschriften oder Punkte, Linien, Flächen für Symbole im Sinne eines einen räumlichen Sachverhalt beschreibenden grafischen Platzhalters. Sie tragen das Präfix „AP_“ (*AP_Darstellung* ist also ebenfalls ein Präsentationsobjekt). Präsentationsobjekte enthalten eine Liste an Attributen, die das Objekt für seine Umsetzung in eine Kartensignatur näher beschreiben:

- Signaturnummer
- Darstellungspriorität
- Art der Signatur (*SymbolClass*)
- Drehwinkel

- Skalierung (bei *AP_PPO*)

Darüber hinaus bei textförmigen Präsentationsobjekten:

- Schriftinhalt
- Fontsperrung
- Skalierung
- horizontale und vertikale Ausrichtung

Eine detaillierte Beschreibung der Präsentationsobjekte einschließlich *AP_Darstellung* finden Sie im „Hauptdokument“ der GeoInfoDok (/GeoInfoDok-7.0/Hauptdokument).

Standardpräsentation

Als Standardpräsentation wird die direkte Umsetzung der Signaturierung eines über den Filter selektierten REOs bezeichnet. Die Ableitung eines ZUSO oder NREO kann somit nie zu einer Standardpräsentation führen.

Standardpräsentationen können mit *AP_Darstellung* überdefiniert werden (s. o.).

Bei den ATKIS-Signaturenkatalogen sind grundsätzlich in folgenden Fällen keine Standardpräsentationen in den Ableitungsregeln angegeben, d. h. der Nutzer bekommt alle notwendigen Schriften, Symbole usw. als Präsentationsobjekte:

- bei allen Schriften
- bei allen Tunnelportalen
- bei allen Brückenkonturen
- bei allen Symbolen außer bei Symbolen, die auf ein punktförmig modelliertes REO platziert und nach geographisch Nord orientiert sind.

Die im Anhang gezeigten Ablaufpläne sollen das unterschiedliche Vorgehen der Signaturierung beim Datenerzeuger und beim Nutzer veranschaulichen (siehe Anhang 7 und 7.2).

Grafische Ausgabe des *Signaturenteils*

Dieses Feld enthält die grafische Ausgabe des Signaturenteils (= einzelner *Symbolizer*) aus der Ableitungsregel.

3.4 Positionierungsregeln (DesignRules)

In die Übersicht der Positionierungsregeln gelangen Sie über die Startseite (*SymbologyCatalog*) und den darauf befindlichen Link „DesignRules – Positionierungsregeln“ oder über die entsprechende Positionierungsregelnummer in den Ableitungsregeln.

Die Positionierungsregeln enthalten die derzeit noch langtextlichen Beschreibungen der bisherigen Regeln. Diese werden künftig durch eine formalisierte Beschreibung in Platzierungsattributen (*PlacementRules*) beschrieben.

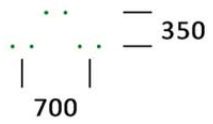
DesignRules

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

version 1.1.0, 2017-11-27

>> SymbologyCatalog ALKIS-FB >> DesignRules

DesignRules — Positionierungsregeln

ID	PNR	zusätzlich zu beachtende PR	Anwendungsregel	Anwendung bei Signaturen
DRU01100	1100		**Gras; Grünanlage; Grünfläche; Grünland; Rain; Garten; Gartenland; Gehölz, Latschenkiefer; Gebüsch** • Untertyp: regelmäßig • Anordnung: (Abstand: 700, Zeilenabstand: 350, Versatz: 350) 	3413, 3421, 3472, 3601
DRU01101	1101		**Gras; Grünanlage; Grünfläche; Grünland; Rain; Garten; Gartenland; Gehölz, Latschenkiefer; Gebüsch** • Untertyp: zufällig • Anordnung: (Abstand: 700, Zeilenabstand: 350, Versatz: 350, Dichte: 20)	3413, 3421, 3472, 3601

DesignRules — Positionierungsregeln

ID	PNR	zusätzlich zu beachtende PR	Anwendungsregel	Anwendung bei Signaturen
DRU00100	100	DRU00110	Das Symbol ist in den Schwerpunkt der Fläche zu setzen.	Symbol in Fläche platzieren
DRU00101	101	DRU00140	Die Schrift ist zentrisch in den Schwerpunkt der Fläche zu setzen. Sind bei einem Objekt NAM und ZNM belegt, ist ZNM unter NAM zu präsentieren.	Schrift in Fläche platzieren
DRU00102	102		Die Schrift ist vom punktförmigen REO aus um 2mm nach rechts und um 2mm nach oben mit dem Schriftbezugspunkt links unten zu platzieren.	Schriften zu punktförmig modellierten Objekten
DRU00103	103	DRU00140	Die Schrift ist von der rechten oberen Ecke der Boundingbox des REO um 1mm nach rechts und um 1mm nach oben mit dem Bezugspunkt links unten zu platzieren.	Namen oder Schriftzusätze zu linien- oder flächenförmig modellierten Objekten (meist geringer geometrischer Ausdehnung)
DRU00104	104	DRU00110	Das Symbol ist von der rechten oberen Ecke der Boundingbox des REO um 3mm nach rechts und um 3mm nach oben zu platzieren.	Symbol für die Nummer des Autobahnknotens (flächenförmig modelliert)
DRU00105	105		Die Schrift ist vom punktförmigen REO aus um 6mm nach rechts und um 6mm nach oben mit dem Schriftbezugspunkt Mitte-Mitte zu platzieren.	Nummer des Autobahnknotens (punktförmig modelliert)

Abbildung 3.10: Liste der DesignRules (Positionierungsregeln) in ALKIS vs. ATKIS

3.5 Funktionen: berechnete Werte, Relationen, geometrische Verschneidungen oder Auswertung von Zeichenketten und Attributwerten (Functions)

In die Übersicht der Funktionen gelangen Sie über die Startseite (*SymbologyCatalog*) und den darauf befindlichen Link „Functions – Selbstdefinierte Funktionen“.

Für die Auswertung von Flächengrößen, Längen, Relationen oder geometrischen Verschneidungen werden im Signaturenkatalog sog. Funktionen eingesetzt.

Functions

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

version 1.1.0, 2017-11-27

>> SymbologyCatalog ALKIS-FB >> Functions

Functions — Selbstdefinierte Funktionen

Name	Argumente	Rückgabe	Erläuterung
HOEHE_AUS_POSITION	(geometry)	string	Ein AP_PTO der Objektart 14003 AX_PunktortAU, das die Eigenschaft "HOEHE_AUS_POSITION" hat und mit Abstand 500 neben einem Objekt der Objektart 61009 AX_BesondererTopographischerPunkt mit seiner verorteten Höhe präsentiert werden soll.
NUMERISCH_NACH_ROEMISCH	(numeric)	string	Ein AP_PTO der Objektart 31001 AX_Gebaeude, das die Eigenschaft numerisch hat aber mit Römischen Zahlen präsentiert werden soll.

Abbildung 3.11: Liste der Functions (selbstdefinierte Funktionen)

Über den Link des jeweiligen Funktionsnamens (Spalte *Name*) gelangen Sie in die MDL-Definition (Befehl *defineFunction*):

```
#-------------------------------------------------------------
# Function Definitions
#-------------------------------------------------------------
defineFunction \
   -name        atk:LGO \
   -synopsis    "LGO: XPath -> nicht möglich" \
   -description {
LGO = Länge des Objektes: Berechnung der Gesamtlänge entlang des Linienzuges
eines REOs.

Haben benachbarte REOs dieselbe Signatur (d.h. dieselbe Signaturnummer,
Darstellungspriorität und ggf. Positionierungsregel), so sind die Längen aller
REOs zu addieren und mit dem/den angegebenen Schwellenwert/en zu vergleichen.

Linienhafte REOs sind benachbart, wenn der Anfangs- oder Endpunkt des einen
REOs mit dem Anfangs- oder Endpunkt des anderen REOs identisch ist.
   } \
   -returntype  numeric

defineFunction \
   -name        atk:FLB \
   -synopsis    "FLB: XPath -> nicht möglich" \
   -description {
FLB = Fläche berechnet: Berechnung der Fläche eines REOs.

Haben benachbarte REOs dieselbe Signatur (d.h. dieselbe Signaturnummer,
Darstellungspriorität und ggf. Positionierungsregel), so sind die Flächen
aller REOs zu addieren und mit dem/den angegebenen Schwellenwert/en zu
vergleichen.

Flächenhafte REOs sind benachbart, wenn deren Kontur in Teilbereichen, die
aus mindestens zwei aufeinanderfolgenden Stützpunkten bestehen, identische
Koordinaten haben.
   } \
   -returntype  numeric
```

Abbildung 3.12: Definition der Funktionen in MDL

In MDL erhalten diese Funktionen ein Präfix:

für AFIS: *afs*

für ALKIS: *alk*

für ATKIS: *atk* und für SK-spezifische Funktionen: *sk10*, *sk25*, *sk50*, *sk100*, *sk250*, *sk1000*

Das Präfix wird getrennt durch einen Doppelpunkt („:“) dem eigentlichen Funktionsnamen vorangestellt. Im MDL-Filterausdruck wird sie dann mit der Parameterklammer („()“) am Ende aufgerufen, z. B. *atk:LGO()* und *atk:LIEGT_IM_TUNNEL_ODER_SCHUTZ-GALERIE()*:

```
Filter \
   -id                SKE03462 \
   -basicObjectType   reo \
   -expression        {
      art = 3000 and atk:BEZ_IST_MAX_ZWEISTELLIG()
   } \
   -featureTypeName   AX_Strassenverkehrsanlage \
   -featureTypeNumber 53002 \
   -geometryType      point \
   -modelClass        DLM50
```

Abbildung 3.13: Aufruf einer Funktion im Filterausdruck einer Rule (expression)

3.5.1 Berechnete Werte

In ATKIS reichen die explizit im DLM enthaltenen Informationen (Geometrietyp, Attributarten, Relationen) für die Signaturierung eines Objektes oft nicht aus. In diesen Fällen ist als zusätzliches Kriterium für die Signaturbildung die Berechnung von Flächengrößen oder Längen von REOs des DLM erforderlich. Folgende Werte können auftreten:

in AFIS:
-

in ALKIS:
-

in ATKIS:

- *BRG_BER* (Berechnete Breite eines speziell im DLM50 flächenhaft modellierten Gewässers) sowie die Angabe eines Wertes oder eines Wertebereiches,
- *FLB* (Fläche berechnet) sowie die Angabe eines Wertes oder eines Wertebereiches,
- *LGO* (Länge des Objektes) sowie die Angabe eines Wertes oder eines Wertebereiches.

3.5.2 Auswertung von Relationen

DLM-Objekte können in vielfältiger gegenseitiger Beziehung (Relation) zueinander stehen. Beispielsweise kann die Angabe einer Unterführungsrelation durch den Eintrag einer Relation „hatDirektUnten“ bei einem REO erfolgen. Der Inhalt der Relation „hatDirektUnten“ ist die Objekt-ID des überführten REOs.

Mögliche Angaben der Relationsbedingungen sind

in AFIS:
-

in ALKIS:

- *HOEHE_AUS_POSITION*

in ATKIS:

- *BRUECKE_ALLEINSTEHEND*
- *BRUECKE_UNTER_TN*
- *IST_DURCHLASS*
- *IST_NICHT_TEIL_VON_BOESCHUNGSFLAECHE*
- *KEIN_DURCHLASS*
- *LIEGT_IM_TUNNEL_ODER_SCHUTZGALERIE*
- *LIEGT_IN_MITTE*
- *LIEGT_NICHT_UNTER_BRUECKE*
- *LIEGT_OBEN*
- *LIEGT_UNTEN*
- *LIEGT_UNTER_BRUECKE*
- *MIT_BOESCHUNGSUNTERKANTE*
- *OHNE_BOESCHUNGSUNTERKANTE*

3.5.3 Geometrische Verschneidungen

Zur richtigen Signaturierung können auch geometrische Verschneidungen verschiedener REOs des DLM notwendig sein.

Mögliche Werte für die Angaben der geometrischen Verschneidungen sind

in AFIS:

-

in ALKIS:

-

in ATKIS:

- *IST_DURCHFAHRT_DURCH_GEBAEUDE*
- *IST_DURCHFAHRT_DURCH_MAUER*
- *IST_KAMMERSCHLEUSE*
- *IST_OEFFENTLICHES_GEBAEUDE*
- *IST_SCHIFFSHEBEWERK*
- *IST_S_BAHN_FLAECHENFOERMIG*
- *IST_S_BAHN_PUNKTFOERMIG*
- *IST_STRASSEN_BAHN_FLAECHENFOERMIG*
- *IST_STRASSEN_BAHN_PUNKTFOERMIG*
- *IST_U_BAHN_FLAECHENFOERMIG*
- *IST_U_BAHN_PUNKTFOERMIG*

- *KEINE_DURCHFAHRT_DURCH_GEBAEUDE*
- *KEINE_DURCHFAHRT_DURCH_MAUER*
- *KEIN_FRIEDHOF*
- *KEINE_S_U_STRASSEN_BAHN_FLAECHENFOERMIG*
- *KEINE_S_U_STRASSEN_BAHN_PUNKTFOERMIG*
- *KEINE_TREPPE*
- *LIEGT_AUF_DAMM_WALL_DEICH*
- *LIEGT_AUF_FELS*
- *LIEGT_AUF_GLETSCHER_ODER_GEWAESSER*
- *LIEGT_AUF_STRASSE*
- *LIEGT_IM_FLACHLAND*
- *LIEGT_IM_GEWAESSER*
- *LIEGT_IM_HOCHGEBIRGE*
- *LIEGT_IM_HUEGELLAND_ODER_MITTELGEBIRGE*
- *LIEGT_NICHT_AUF_FELS_GLETSCHER_GEWAESSER*
- *LIEGT_NICHT_AUF_STRASSE_WEG_BAHN*
- *LIEGT_NICHT_IM_SIEDLUNGSGEBIET*
- *LIEGT_NICHT_AUF_STASSE*
- *LIEGT_NICHT_AUF_STASSE_WEG_BAHN*
- *LIEGT_NICHT_UEBER_GEWAESSER*
- *MIT_WALL*
- *OHNE_WALL*
- *WEG_LIEGT_DARUNTER*

3.5.4 Auswertung von Zeichenketten und Attributwerten

Zur richtigen Signaturierung kann auch die Auswertung von Attributinhalten, die beispielsweise aus Zeichenketten bestehen, notwendig sein. Mögliche Funktionen für die Auswertung solcher Werte sind

in AFIS:
-

in ALKIS:
NUMERISCH_NACH_ROEMISCH

in ATKIS:

- *BEZ_IST_MIN_DREISTELLIG*
- *BEZ_IST_MAX_ZWEISTELLIG*

- *HAT_GWK_XNULL*
- *KEIN_GWK_XNULL*

3.6 Schriften (Fonts)

In die Übersicht der verwendeten Schriften gelangen Sie über die Startseite (*SymbologyCatalog*) und den darauf befindlichen Link „Fonts – Schriften“.

Die verwendeten Schriftarten werden herstellerneutral aufgelistet:

Fonts

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

version 1.1.0, 2017-11-27

>> SymbologyCatalog ALKIS-FB >> Fonts

Fonts — Schriften

ID	Name	Stil	Schnitt	Muster
FON00001	Arial	normal	normal	Hamburgeron
FON00002	Arial	normal	bold	**Hamburgeron**
FON00003	Times New Roman	normal	normal	Hamburgeron
FON00004	Arial	italic	normal	*Hamburgeron*
FON00005	Arial	italic	bold	***Hamburgeron***

Abbildung 3.14: Liste der verwendeten Fonts

3.7 Farbtabelle (Colors)

In die Übersicht der verwendeten Farben gelangen Sie über die Startseite (*SymbologyCatalog*) und den darauf befindlichen Link „Colors – Farbtabelle“.

Die Farbtabelle enthält die aufgeführten Farbwerte. Dabei können abhängig vom Aufbau des Regelwerks alle Farbwerte des gesamten Signaturenkatalogs gelistet werden oder nur die eines bestimmten Layers oder RuleSets.

Colors

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

version 1.1.0, 2017-11-27

>> SymbologyCatalog ALKIS-FB >> Colors

Colors — Farbtabelle

ID	Name	Farbbeispiel	RGB			CMYK				Web
			rot%	grün%	blau%	cyan%	magenta%	yellow%	schwarz%	
COL00001	Weiß		100.00	100.00	100.00	0.00	0.00	0.00	0.00	#ffffff
COL00002	Grau5		70.00	70.00	70.00	30.00	30.00	30.00	0.00	#b3b3b3
COL00003	Grau3		90.00	90.00	90.00	10.00	10.00	10.00	0.00	#e6e6e6
COL00004	Schwarz		0.00	0.00	0.00	0.00	0.00	0.00	100.00	#000000
COL00005	Rot2		96.00	52.00	50.00	4.00	48.00	50.00	0.00	#f58580
COL00006	Rot		99.00	88.00	88.00	1.00	12.00	12.00	0.00	#fde1e1
COL00007	Grau2		96.00	96.00	96.00	4.00	4.00	4.00	0.00	#f5f5f5
COL00008	Braun		95.00	89.00	79.00	5.00	11.00	21.00	0.00	#f3e3ca
COL00009	Grün2		86.00	90.00	76.00	14.00	10.00	24.00	0.00	#dce6c2
COL00010	Grün		95.00	96.00	80.00	5.00	4.00	20.00	0.00	#f3f5cc
COL00011	Ocker		100.00	97.00	86.00	0.00	3.00	14.00	0.00	#fff8dc
COL00012	Blau		75.00	91.00	98.00	25.00	9.00	2.00	0.00	#c0e8fa
COL00013	Grün3		81.00	91.00	85.00	19.00	9.00	15.00	0.00	#cfe8d9

Abbildung 3.15: Farbtabelle

3.8 Beschreibung der Signaturen (Symbolizer)

3.8.1 Signatureigenschaften

Die Signatureigenschaften sind vom jeweiligen Signaturtyp abhängig.

Fläche

Flächen werden neu in MDL als sog. *AreaSymbolizer* definiert. Das wichtigste Gestaltungselement ist dabei die Farbe (für Fläche und Umrisslinie). Sie wird referenziert aus einer Farbtabelle.

```
        AreaSymbolizer \
           -id             SYM40200 \
           -synopsis       "Signatur Nr. 40200 (SK50)" \
           -description    {
Landwirtschaft (Grünland)
Unland, vegetationslose Fläche (Gewässerbegleitfläche, naturnahe Fläche)
(Fläche)
           } \
           -signaturnummer 40200 \
           -zIndex         2 \
           -interior       {
              SolidFill \
                 -color COL00005
           }
     } \
     -zIndex          2
```

Abbildung 3.16: MDL-Definition eines AreaSymbolizer

COL00005 verweist auf folgende im ATKIS-SK50 als CMYK-Wert angegebene Farbdefinition:

```
-color {
   ColorCMYK \
      -id      COL00005 \
      -name    Wiesengrün \
      -black   0.00 \
      -cyan    10.00 \
      -magenta 0.00 \
      -yellow  20.00
}
```

Abbildung 3.17: MDL-Definition eines CMYK-Farbwerts

Die Farbe wird mit dem Farbgrundton und den jeweiligen Farbanteilen in % angegeben (dies gilt analog auch für RGB-Werte, wie sie beispielsweise ALKIS verwendet).

Für Umrisslinien von Flächen gelten alle Eigenschaften die nachfolgend für Linien beschrieben sind.

Linie

Linieneigenschaften sind komplexer als Flächeneigenschaften. Sie unterscheiden sich neben der Farbe nach

- Strichstärke
- Linienabschluss
- Linienscheitel
- Strichart

Strichstärke (*width*)

Angabe der Strichstärke in 1/100 mm.

Linienabschluss (*linecap*)

Legt fest, wie die Linie an den Endpunkten zu zeichnen ist.

Abgeschnitten (*butt*):

Rund (*round*):

Quadratisch (*square*):

Abbildung 3.18: Definition von Linienenden

Pfeile

Abbildung 3.19: Definition von Pfeilspitzen

Pfeilspitzen werden über ein *Graphic* mittels der Properties *preGap*/*postGap* der Klasse *Stroke* im *LineSymbolizer* definiert (vgl. 2.6.2).

Linienscheitel (*linejoin*)

Legt fest, wie die Verbindung an den Scheitelpunkten zu zeichnen ist.

Spitz (*miter*):

Rund (*round*)

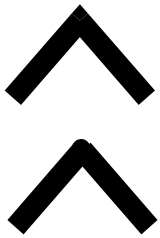


Abbildung 3.20: Definition von Linienverbindungen

Linienfarbe (*color*)

Die Farbe wird mit dem Farbgrundton und den jeweiligen Farbanteilen in % angegeben (s. o.).

Strichart (*DashedStroke*):

Legt fest, wie gestrichelte Linien gezeichnet werden.

```
DashedStroke \
    -postGap    0.00 \
    -preGap     0.00 \
    -color      COL00006 \
    -linecaps   butt \
    -width      15.00 \
    -adjustment wholePattern \
    -dasharray  "150 50"
```

Abbildung 3.21: MDL-Definition einer gestrichelten Linie

Die Werte für Einzug (*preGap* am Anfang, *postGap* am Ende), Linienlänge (1. Wert von *dasharray*) und Lücke (2. Wert von *dasharray*) zwischen den Linien sind in Einheiten von 1/100 mm angegeben. Die Ausrichtung des Musters wird zusätzlich über adjustment geregelt.

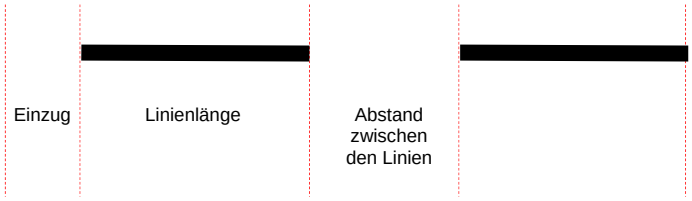


Abbildung 3.22: Definition eines Linieneinzugs

Sonderform punktierte Linie:

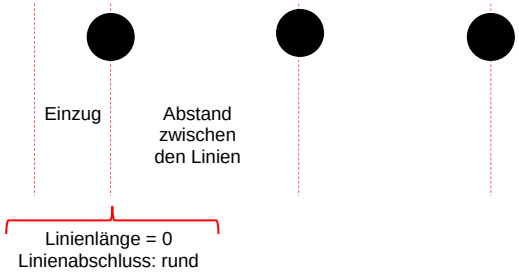


Abbildung 3.23: Definition einer punktierten Linie

Sonderform strich-punktierte Linie:

Strich-punktierte Linien werden in MDL als Abrollsignaturen definiert.

```
CompoundStroke \
   -postGap 0.00 \
   -preGap 0.00 \
   -adjustment wholePattern \
   -sections {
      SolidSection \
         -length 500.00 \
         -stroke {
            SolidStroke \
               -color COL00022 \
               -linecaps butt \
               -linejoin round \
               -width 60.00
         }
      Gap \
         -length 125.00
      SolidSection \
         -length 0.00 \
         -stroke {
            SolidStroke \
               -color COL00022 \
               -linecaps round \
               -linejoin round \
               -width 70.00
         }
      Gap \
         -length 125.00
   }
```

Linienlänge: 500/100 mm
Linienende: Abgeschnitten
Länge der Lücke: 125/100 mm
Linienlänge: 0 mm
Linienende: Rund
Länge der Lücke: 125/100 mm

Abbildung 3.24: MDL-Definition einer strich-punktierten Linie

Symbol

Symbole werden aus Flächen, Linien und Texten zusammengesetzt. In MDL werden die Formen über die SVG-Pfadbeschreibung beschrieben. Dabei handelt es sich um den sog. *PointSymbolizer*, z. B. die Signaturnummer 24000 für das Kirchensymbol:

```
      PointSymbolizer \
         -id             SYM24000 \
         -synopsis       "Signatur Nr. 24000 (SK50)" \
         -description    {
Gebäude (Kirche)
(Einzelsignatur)
         } \
         -signaturnummer 24000 \
         -graphic        {
            CompositeGraphic \
               -id            SKE00159 \
               -displacementX 0.00 \
               -displacementY 0.00 \
               -members       {
                  LineGraphic \
                     -id       SKE00160 \
                     -geometry {
                        Path \
                           -data "M -60.00,130.00 L 60.00,130.00"
                     } \
                     -stroke   {
                        SolidStroke \
                           -color    COL00030 \
                           -linecaps butt \
                           -linejoin miter \
                           -width    20.00
                     }
                  LineGraphic \
                     -id       SKE00163 \
                     -geometry {
                        Path \
                           -data "M 0.00,190.00 L 0.00,50.00"
                     } \
                     -stroke   {
                        SolidStroke \
                           -color    COL00030 \
                           -linecaps butt \
                           -linejoin miter \
                           -width    20.00
                     }
                  AreaGraphic \
                     -id       SKE00166 \
                     -geometry {
                        Path \
                           -data "M 0.00,60.00 A 60.00,60.00 0 1,0 0.00,-60.00 A 60.00,60.00 0 1,0 0.00,60.00 Z"
                     } \
                     -interior {
                        SolidFill \
                           -color COL00030
                     }
                  AreaGraphic \
                     -id       SKE00169 \
                     -geometry {
                        Path \
                           -data "M 0.00,40.00 A 40.00,40.00 0 1,0 0.00,-40.00 A 40.00,40.00 0 1,0 0.00,40.00 Z"
                     } \
                     -interior {
                        SolidFill \
                           -color COL00002
                     }
               }
         }
   } \
```

Abbildung 3.25: MDL-Definition eines Symbols

Die Einheit des Koordinatensystems ist 1/100 mm. Die Reihenfolge der Zeichnung der einzelnen Elemente ist fortlaufend nummeriert. Der Bezugspunkt des Symbols entspricht in der Regel dem Ursprung des lokalen Koordinatensystems.

Schrift

Der Schriftgrad ist in der Einheit Punkt (pt) angegeben. Ein pt entspricht 0,3527 mm. In diesen Schriftgrad einbezogen sind von der Schriftart abhängige Unter-, Mittel- und Oberlängen. Um die Großbuchstabenhöhe (Versalhöhe) einer Schriftart in mm annähernd zu ermitteln, muss der Wert des Schriftgrads mit dem Faktor 0,25 multipliziert werden.

Beispiel: Der Schriftgrad der Signatur 80080 ist mit 26.8 pt (Schriftart Univers schmal, Schriftstil halbfett) angegeben. In mm umgerechnet ist der Großbuchstabe „H“ 6,7 mm hoch.

Abbildung 3.26: Definition des Schriftgrads

3.9 Reihenfolge der Zeichnung (Painters Model vs. zIndex)

Die Reihenfolge der Zeichnung innerhalb eines Symbolizers wird in MDL gemäß dem „Painters Model“ automatisch festgelegt (was oben steht, wird zuerst gezeichnet, was unten steht, zuletzt). Die Darstellungspriorität in der Karte wird in MDL innerhalb der Ableitungsregeln über den zIndex definiert. Dabei liegt eine hohe Zahl in der Zeichenreihenfolge oben und eine niedrige Zahl unten.

Auf das Dokument „Darstellungsprioritäten“ wurde verzichtet.

3.10 Beschreibung von Signaturen über Positionierungsregeln (DesignRules)

Positionierungsregeln beschreiben die Positionierung von Präsentationsobjekten.

```
      DesignRule \
         -id              DRU00100 \
         -synopsis        "Positionierungsregel Nr. 100 (SK50)" \
         -description     {
Das Symbol ist in den Schwerpunkt der Fläche zu setzen.
Anwendung bei Signaturen: Symbol in Fläche platzieren
         } \
         -rulesToConsider {
            DesignRule \
               -id          DRU00110 \
               -synopsis    "Positionierungsregel Nr. 110 (SK50)" \
               -description {
Alle benachbarten REOs mit derselben Signaturierungsregel verbinden.
Anwendung bei Signaturen:
               }
         }
```

Abbildung 3.27: MDL-Definition einer DesignRule

Allgemeine Anmerkungen:

Standarmäßig sind (Muster-) Symbole und Texte, soweit nichts anderes angegeben wurde, nach geographisch Nord ausgerichtet.

Anfangsrichtung = Richtungswinkel am Anfang: RA= Richtungswinkel in gon der ersten Seite des Linienzuges im Anfangspunkt; Nullrichtung nach Norden, Zählung im Uhrzeigersinn.

Endrichtung = Richtungswinkel am Ende: RE= Richtungswinkel in gon der letzten Seite des Linienzuges im Endpunkt – 200 gon

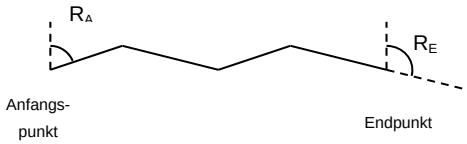


Abbildung 3.28: Richtungswinkel eines Anfangs- und Endpunkts im Linienverlauf

Beispiel Linie punktieren: z. B. PNR 310 (Linienmuster Pipeline)

- Punktabstand Mitte-Mitte: 600/100 mm
- Lücke am Anfang: 0 mm
- Lücke am Ende: -600/100 mm (d. h. auf das Linienende ist ein Punkt zu setzen)

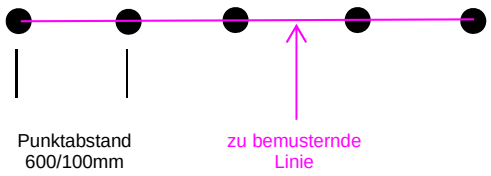


Abbildung 3.29: Definition einer punktierten Linie

Beispiel Linie mit Querstrichen bemustern: z. B. PNR 619 (Linienmuster von Schmalspurbahn, Breitspurbahn, Magnetschwebebahn, Zahnradbahn, Standseilbahn)

- Die Achse des REO mit Linien, symmetrisch und senkrecht zur Achse des REO, bemustern (Querstriche):
- Strichlänge: 80/100 mm
- Strichabstand: 200/100 mm (von Linienmitte zu Linienmitte)
- Lücke am Anfang (Anfang des REOs bis zur Linienmitte des ersten Querstriches): 100/100 mm
- Lücke am Ende (Ende des REOs bis zu Linienmitte des letzten Querstriches): -100/100 mm (das Minuszeichen bedeutet, dass die zu bemusternde Linie um 100/100 mm (virtuell) zu verlängern ist. Mit der neuen Länge wird die Anzahl der Muster berechnet, dann Musterausgleich wie in PNR 111 beschrieben)

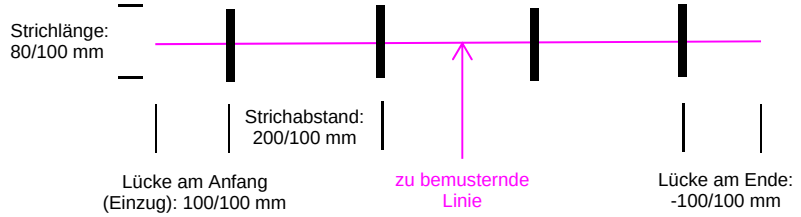


Abbildung 3.30: Definition eines Linienmusters

Beispiel Fläche bemustern: z. B. PNR 412 (Flächenmuster Hopfen)

- Abstand Muster horizontal: 220/100 mm
- Abstand Muster vertikal: 110/100 mm
- Versatz Muster untereinander: 110/100 mm

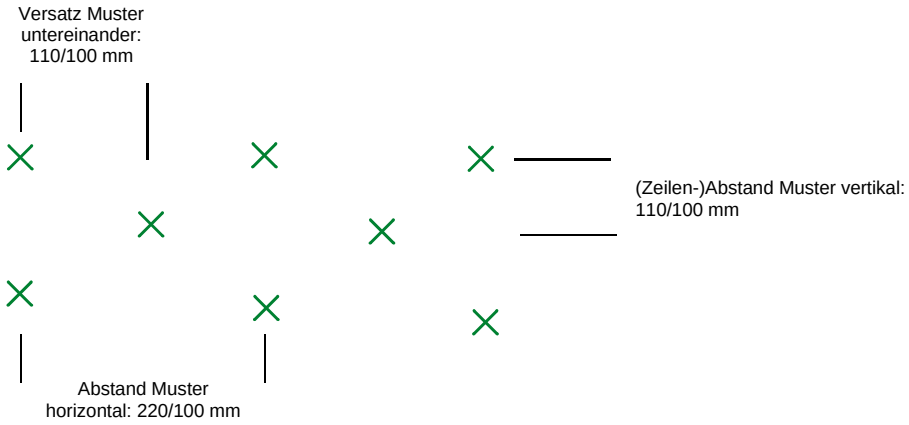


Abbildung 3.31: Definition eines Flächenmusters

4 Map Definition Language (MDL)[11]

Die Map-Definition-Language, kurz MDL, ist ein Hilfsmittel zur Definition des SK-Modells durch Textdateien. MDL ist die Definitionsform des AAA-Signaturenkatalogs für die gestaltenden und pflegenden Verantwortlichen der AdV. Ein Umsetzprogramm, der MDL-Compiler *mdlc*, gestattet die Überführung der in MDL verfassten Definition nach SK-XML.

Die Syntax für MDL wurde an die Regeln der Programmiersprache Tcl (Tool Command Language, [TCL]) angelehnt, da die erste Umsetzung in eben dieser Sprache erfolgt. Dies ermöglicht den Einsatz des Tcl-Interpreters zum Parsen der Quellen. Für die interne objektorientierte Modellierung wurde die Spracherweiterung Itcl verwendet, die ein fest definiertes Klassensystem, ähnlich dem von C++ und Java, zur Verfügung stellt.

Grundsätzlich wurde darauf geachtet, den Sprachumfang zunächst gering zu halten und die Kommandos auf die für den Aufbau des Objektmodells nötigen Konstruktoren zu beschränken.

Die Ableitung von MDL aus dem UML-Modell erfolgte nach folgenden Regeln:

1. Alle konkreten UML-Klassen (mit Ausnahme der unter Punkt 4 und 5 beschriebenen Teilbäume) werden zu Konstruktoren, deren Variable sich aus den Modellklassen unter Berücksichtigung der Vererbungshierarchie ergeben. Alle Attribute und Rollen werden zu Variablen der Klassen.
2. Assoziationen können über zwei verschiedene Wege gebildet werden:
   a. Durch Angabe der katalogweit eindeutigen id (Klasse Element) eines Objekts.
   b. Durch die direkte Erzeugung des Objekts mithilfe seines Konstruktoraufrufs.
3. Enumerationen werden über die jeweilige Class-Set-ter Methode (configure) eingeschränkt.
4. Die Klasse Expression und alle ihre Ableitungen werden in MDL als Xpath-Dialekt spezifiziert, siehe [XPATH]. Der MDL-Compiler prüft die Syntax, eventuelle fachliche Abhängigkeiten und setzt diesen Ausdruck in einen entsprechenden Objektbaum um.
5. Die Klasse Path und deren enthaltene Geometrieelemente werden in ein Geometrie-Objekt umgesetzt, welches eine Variable path aufweist. Diese enthält die Geometrie direkt in eine Untermenge der SVG-Path-Syntax [SVG].

Neben diesen Befehlen verfügt MDL über zusätzliche Kommandos, die z.B. die Organisation des Quellcodes ermöglichen oder vereinfachte Konstruktionsmöglichkeiten anbieten.

[11] Erstling, 2016, Abschlussbericht, Version 1.10.1, Kapitel 6

5 SK-XML[12]

SK-XML ist ein XML-Encoding des oben beschriebenen SK-Modells.

SK-XML kann aus der primären Repräsentierung des SKs in MDL durch einen zugehörigen Compiler abgeleitet werden und dient als Schnittstelle für externe Nutzungen des neuen, formalisierten SK.

Für die Ableitung des SK-XML aus dem UML-Modell wurden im Grundsatz die Regeln verwendet, welche die Ableitung von GML-Applikationsschemas aus ISO 19109-konformem UML steuern. Allerdings handelt es sich weder beim SK-Modell um ein ISO 19109-konformes UML-Modell, noch bei SK-XML um ein GML-Applikationsschema. Trotzdem ist die Übereinstimmung groß genug, dass ShapeChange für eine automatische Umsetzung einsetzt werden konnte.

Die Ableitung des XML Schemas zu SK-XML aus dem UML-Modell erfolgt auf Basis der generellen Idee, dass die Klassendefinitionen im UML-Modell jeweils abgebildet werden auf Deklarationen für complexType und element in XML Schema. Damit werden die Objekte der Klassen gleichnamige Elementstrukturen im XML-Dokument.

5.1 Ableitungsregeln

Die Abbildung erfolgt nach folgenden Regeln:

1. Das gesamte Paket wird in ein XML Schema Dokument überführt.
2. Bis auf die unten aufgeführten Ausnahmen in Regel 10, 11 und 12 werden UML-Klassen zu gleichnamigen globalen Deklarationen für *element* und *complexType*. Der Name des *complexType* wird aus dem Klassennamen durch Anfügen von „Type“ gebildet.
3. Abstrakte Klassen führen zu Deklarationen von *element* und *complexType* mit dem Attribute *abstract="true"*.
4. Bis auf die unten aufgeführten Ausnahmen in Regel 10 und 11 nennen die Deklarationen von *element* abgeleiteter Klassen die Basisklasse im Attribut *substitutionGroup* und deren *complexType*-Deklarationen werden per extension aus dem *complexType* der Basisklasse hergeleitet.
5. Bis auf die unten aufgeführten Ausnahmen in Regel 11 und 12 werden Attribute von Klassen zu lokalen *element*-Deklarationen mit *simpleType* gemäß den Regeln 7 und 8 oder „PropertyType“ gemäß Regel 9 überführt. Kardinalitäten werden in *minOccurs* und *maxOccurs* übernommen (Ausnahmen Regel 13, 14 und 15).
6. Rollen von Klassen werden zu lokalen *element*-Deklarationen mit „PropertyType“ gemäß Regel 9 überführt.
7. Attribute mit einfachen Typen im UML-Schema werden wie folgt auf XML Schema Typen abgebildet: Boolean→boolean, CharacerString→string, Real→double, Inte-

[12] Erstling, 2016, Abschlussbericht, Version 1.10.1, Kapitel 7

ger→integer, Date→date, ID→ID, GenericName→Qname. Der union-Typ AA_Modellart aus dem AAA-Basisschema wird in string abgebildet.

8. Attribute mit Enumeration-Typ im UML-Schema werden auf den *simpleType* abgebildet, der sich aus der Enumeration ergibt. Siehe Regel 11.
9. Für Attribute, die in einem *complexType* umgesetzt werden, oder für Rollen wird ein globaler „PropertyType“ erzeugt, der den Zugriff auf die referierte Klasse beschreibt. Dessen Name ist der Name der Klasse mit dem Postfix „PropertyType“. Bei Klassen, die aus der Klasse *Element* direkt oder indirekt abgeleitet sind, wird der „PropertyType“ so angelegt, dass er wahlweise eine Referenz per *xlink:href* oder eine Einbettung ermöglicht. Bei allen anderen Klassen ist nur eine Einbettung möglich.
10. Klassen mit dem Stereotype <<*type*>> werden nicht in element und *complexType* umgesetzt. Vielmehr werden ihre Inhalte an alle daraus abgeleitete Klassen ohne Stereotype weitergegeben.
11. Klassen mit dem Stereotype <<*enumeration*>> werden in einen SimpleType umgesetzt, der die Werte der Enumeration als Facetten von string definiert.
12. Folgende UML-Attribute mit Kardinalität 1 oder 0..1 werden nicht zu element-Deklarationen sondern zu *attribute*-Deklarationen:
    a. *Element/id*
    b. *Element/name*
    c. *Literal/type*
    d. *Literal/value*
    e. *MeasureLiteral/uom*
    f. *Operation/operationName*
    g. *Variable/variableName*
    h. *featureTypeStep/featureTypeName*
    i. *Step/isAttr*
    j. *Step/stepName*
    k. *Moveto/x*
    l. *Moveto/y*
    m. *Lineto/x*
    n. *Lineto/y*
    o. *CircularArc/xa*
    p. *CircularArc/ya*
    q. *CircularArc/x*
    r. *CircularArc/y*
    s. *Curveto/x1*
    t. *Curveto/y1*
    u. *Curveto/x2*
    v. *Curveto/y2*
    w. *Curveto/x*
    x. *Curveto/y*

    D.h. insgesamt werden XML-Attribute wie folgt eingesetzt: *id* und *name* bei *Element* und alle einfachen Attribute innerhalb von *Expression* und *Path*.

13. Die „qualifizierten Assoziationen“, welche die Auswahl nach *Style* darstellen, werden in ein XML-Attribut *style* mit Typ *anyURI* umgesetzt, welcher es ermöglicht, auf die *id* des betreffenden *Style*-Elements zu verweisen. Die Kardinalität wird in *maxOccurs* auf *unbounded* gesetzt.
14. Die multiplen Attribute *DashedStroke/dasharray* und *PointsOnLine/pattern* werden nicht in multiple Elemente sondern in Listen von double-Werten umgesetzt.
15. Das multipleAttribut *Path/element* wurde in Array-Notation dargestellt, also als einfaches Property mit multiplem Inhalt.

5.2 Praktische Ausführung

Die Umsetzung gemäß den oben genannten Regeln wird mittels ShapeChange 2.0 ab Snapshot-Version ShapeChange-2.0.0-20140829.120120-16 und ab Modellversion 1.4 automatisch durchgeführt.

Zur automatischen Umsetzung wird folgende ShapeChange-Konfiguration eingesetzt:

```
<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeConfiguration xmlns:xi="http://www.w3.org/2001/XInclude"
        xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
        xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
        <input>
            <parameter name="inputModelType" value="EA7"/>
            <parameter name="inputFile" value="AAA.eap"/>
            <parameter name="appSchemaName" value="AAA_Signaturenkatalog"/>
            <parameter name="publicOnly" value="true"/>
            <parameter name="checkingConstraints" value="disabled"/>
            <parameter name="sortedSchemaOutput" value="true"/>
            <xi:include href="http://shapechange.net/resources/config/StandardAliases.xml"/>
        </input>
        <log>
            <parameter name="reportLevel" value="INFO"/>
            <parameter name="logFile" value="log.xml"/>
        </log>
        <targets>
            <TargetXmlSchema class="de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema"
                mode="enabled">
                <xi:include href="http://shapechange.net/resources/config/StandardNamespaces.xml"/>
                <xi:include href="http://shapechange.net/resources/config/StandardMapEntries.xml"/>
                <targetParameter name="defaultEncodingRule" value="sk-xml"/>
                <rules>
                    <EncodingRule name="sk-xml" extends="*">
                        <rule name="rule-xsd-cls-no-gml-types"/>
                        <rule name="rule-xsd-all-naming-gml"/>
                        <rule name="rule-xsd-cls-global-enumeration"/>
                        <rule name="rule-xsd-cls-standard-gml-property-types"/>
                        <rule name="rule-xsd-prop-xsdAsAttribute"/>
                        <rule name="rule-xsd-cls-mixin-classes"/>
                        <rule name="rule-xsd-prop-inlineOrByReference"/>
                        <rule name="rule-xsd-prop-qualified-associations"/>
                        <rule name="rule-xsd-prop-gmlListProperty"/>
                        <rule name="rule-xsd-prop-gmlArrayProperty"/>
                    </EncodingRule>
                    <EncodingRule name="notEncoded" extends="*">
                        <rule name="rule-xsd-all-notEncoded"/>
                    </EncodingRule>
                </rules>
                <xsdMapEntries>
```

```
                <XsdMapEntry type="CharacterString" xsdEncodingRules="*" xmlPropertyType="string"
                    xmlType="string" xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="Integer" xsdEncodingRules="*" xmlPropertyType="integer"
                    xmlType="integer" xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="Real" xsdEncodingRules="*" xmlPropertyType="double"
                    xmlType="double" xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="Boolean" xsdEncodingRules="*" xmlPropertyType="76oolean"
                    xmlType="76oolean" xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="Date" xsdEncodingRules="*" xmlPropertyType="date" xmlType="date"
                    xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="GenericName" xsdEncodingRules="*" xmlPropertyType="Qname"
                    xmlType="Qname" xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="AA_Modellart" xsdEncodingRules="*" xmlPropertyType="string"
                    xmlType="string" xmlTypeType="simple" xmlTypeContent="simple"/>
                <XsdMapEntry type="ID" xsdEncodingRules="*" xmlPropertyType="ID" xmlType="ID"
                    xmlTypeType="simple" xmlTypeContent="simple"/>
            </xsdMapEntries>
        </TargetXmlSchema>
    </targets>
</ShapeChangeConfiguration>
```

5.3 Schema-Dokument und Namespace

Das XML Schema hat den Namen:

sk-xml.xsd

Es enthält alle erforderlichen Vereinbarungen bis auf *xlink*, welches über ein stabiles URL referiert wird.

Als Namespace für das SK-Schema wurde

http://www.adv-online.de/namespaces/adv/sk-xml/6.0

gewählt.

5.4 Handhabung von Objektreferenzen

Das Modell wurde so entworfen, dass es für bestimmte Objekte, nämlich die, welche aus dem Schemaobjekt *Element* abgeleitet wurden, ein einheitliches Vorgehen bei Objektbeziehungen festlegt. Alle aus *Element* abgeleiteten Schemaelemente erhalten einen einheitlichen Satz von Attributen und beschreibenden Elementen.

Die Attribute sind *id* und *name*. Sie dienen beide der Bezeichung der Elemente, wobei *id* für die Bezeichung bei Verknüpfungen unter den Elementen dient und *name* für die Identifikation durch Menschen gedacht ist. Im Gegensatz zu *id* braucht *name* deshalb auch nicht eindeutig zu sein.

5.4.1 Das id-Attribut

id muss innerhalb eines XML-Dokuments eindeutig sein. Der XML-Datentyp ist ID, was den Zeichenvorrat für dieses Attribut festlegt.

5.4.2 Das xlink:href-Attribut

Alle Properties, welche auf aus *Element* hergeleitete Schemaelemente referieren, bieten die Wahl, entweder das referierte Objekt unmittelbar in das Property-Element einzubetten – das referierte Objekt braucht dann keine *id*, kann aber eine haben – oder auf das Objekt wird referiert. Dies geschieht mittels des *xlink:href*-Attributs am Property.

Beispiel für Einbettung am Property SolidFill/color:

```
<skx:SolidFill>
 <skx:color>
  <skx:ColorCMYK id="col_01_c" name="gelb">
   <skx:cyan>0</skx:cyan>
   <skx:magenta>0</skx:magenta>
   <skx:yellow>50</skx:yellow>
   <skx:black>0</skx:black>
  </skx:ColorCMYK>
 </skx:color>
</skx:SolidFill>
```

Eine entsprechende Referenz auf ein externes *Color*-Objekt könnte lauten:

```
<skx:SolidFill>
 <skx:color xlink:href="#col_01_c"/>
</skx:SolidFill>
```

Im Beispiel wurde das referierte *Color*-Schemaelement im selben XML-Dokument verortet. Dazu ist anzumerken, dass an dieser Stelle auch beliebige URIs gestattet sind. Um z.B. das Objekt aus einem separaten XML-Dokument namens „Colors.xml“ zu beziehen, wäre auch

```
<skx:SolidFill>
 <skx:color xlink:href="Colors.xml#col_01_c"/>
</skx:SolidFill>
```

möglich.

5.4.3 Das style-Attribut

Werden verschiedene Styles verwendet (z. B. für eine Farb- oder Graustufenvariante) sind *Style*-Zuordnungen zu schreiben.

Folgendes ist ein Beispiel für einen *AreaSymbolizer*, in dem für den einen *Style* eine referierende Schreibweise benutzt wurde und für den anderen eine einbettende. Normalerweise wird man das eher gleichartig organisieren.

```
<skx:Layer >
 <skx:style><skx:Style id="Farbe"/></skx:style>
 <skx:style><skx:Style id="Graustufen"/></skx:style>
 <skx:ruleSet>
  …
    <skx:AreaSymbolizer >
     <skx:interior>
      <skx:SolidFill>
       <skx:color xlink:href="Colors.xml#col_01_c" style="#Farbe"/>
        <skx:color style="#Graustufen">
         <skx:ColorCMYK >
          <skx:cyan>0</skx:cyan>
          <skx:magenta>0</skx:magenta>
          <skx:yellow>0</skx:yellow>
          <skx:black>20</skx:black>
         </skx:ColorCMYK>
        </skx:color>
      </skx:SolidFill>
     </skx:interior>
    </skx:AreaSymbolizer>
  …
 </skx:ruleSet>
</skx:Layer>
…
```

Die Wirkung soll so sein, dass bei Wahl des Styles mit *id=Farbe* die referierte Farbdefinition wirken soll und bei *Style* mit *id=Graustufen* die zweite, eingebettet vorgegebene. Verzeichnisse und Quellen

6 Verzeichnisse

6.1 Abkürzungsverzeichnis

Abkürzung	Langtext
AAA	AFIS-ALKIS-ATKIS
AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland
AFIS	Amtliches Festpunktinformationssystem
ALKIS	Amtliches Liegenschaftskataster Informationssystem
ATKIS	Amtliches Topographisch-Kartographisches Informationssystem
CRS	Coordinate Reference System
DLM	Digitales Landschaftsmodell
DTK	Digitale Topographische Karte
EQK	Erhebungs- und Qualifizierungskomponente
GeoInfoDok	Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens
GIS	Geoinformationssystem
GML	Geography Markup Language
ID	Identifikator / Identifier
ISO	International Organization for Standardization
MDL	Map Definition Language
NAS	Normbasierte Austauschschnittstelle
NREO	Nicht raumbezogenes Elementarobjekt
OGC	Open Geospatial Consortium
REO	Raumbezogenes Elementarobjekt
SE	Symbology Encoding
SK	Signaturenkatalog
SK-OM	UML-basiertes Objektmodell zum AAA-Signaturenkatalog
SLD	Styled Layer Descriptor
TCL	Tool command language
TFIS	Touristik- und Freizeitinformationssystem
TFIS-SK	Signaturenkatalog zu TFIS
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WebAtlasDE	Internet-Kartendienst der AdV

WEB-SK	Signaturenkatalog des WebAtlasDE
XML	Extensible Markup Language
XPath	XML Path Language
ZUSO	Zusammengesetztes Objekt

6.2 Abbildungsverzeichnis

6.3 Quellen

[AAA-SK-OM und SK-XML] Formalisierung des AAA-SK – SK-Objektmodell, SK-XML und MDL-Erzeugung (Abschlussbericht) / Version 1.10.1/ Stand: 03.03.2016 / Interactive Instruments AG (Autor: Reinhard Erstling)

[ALKIS-SK] Dokumentation zur Modellierung der Geoinformation des amtlichen Vermessungswesens (GeoInfoDok) / ALKIS-Signaturenkatalog / <diverse Teile> / Version 6.0.1 / Stand 31.05.2009 / Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

[ATKIS-SK50] Dokumentation zur Modellierung der Geoinformation des amtlichen Vermessungswesens (GeoInfoDok) / Kapitel 8 ATKIS-Katalogwerke / Abschnitt 8.2.3 ATKIS-Signaturenkatalog 1:50000 / <diverse Teile> / Version 6.0.1 / Stand 15.05.2012 / Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)

[OGC-SLD] Styled Layer Descriptor profile of the Web Map Service Implementation Specification / OGC 05-078r4 / Open Geospatial Consortium Inc. / 29.06.2007

[OGC-SE] Symbology Encoding Imlementation Specification / OGC 05-077r4 / Open Geospatial Consortium Inc. / 21.07.2006

[SLD-SK] Machbarkeit der Anwendung von SLD und SVG für den AAA SK / Diskussionspapier der AdV / 2004

[SVG] Überblicksseite des W3C zu SVG
http://www.w3.org/Graphics/SVG/

[TCL] Entwicklerseite zu Tcl
http://www.tcl.tk/

[TFIS-SK] Schema und XML-Dokumente im Archiv
TFIS-SK_Abgabe_AEDSICAD_130516.zip
bereitgestellt durch LGL BW

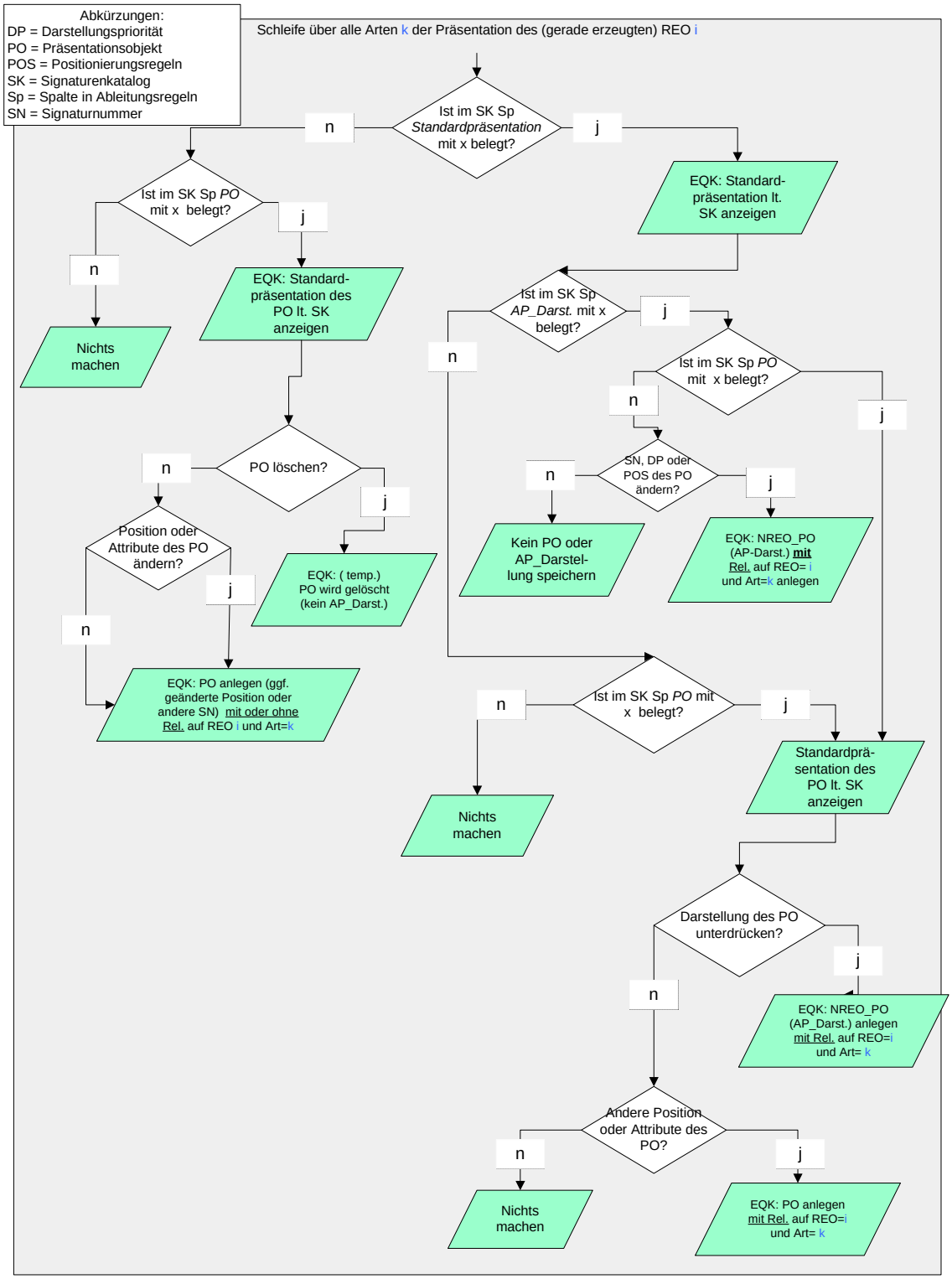
[WEB-SK] Web-SK Version 1.2 /
1073_TOP4.4.1_WebSK_Version_1.2_Stand 20130128.xls
PG ATKIS-Geodienste / AdV

[XPATH] Die XML Path Language W3C Recommendation
http://www.w3.org/TR/xpath20/

7 Anhang

7.1 Anhang 1: Ablaufdiagramm Erzeugung von ATKIS-Präsentationsdaten

Ablauf der Erzeugung der Präsentationsdaten (PO, NREO_PO) in der ATKIS-EQK (Erhebungs- und Qualifizierungskomponente = GIS, in dem die Daten des DLM erfasst und aktualisiert werden):



7.2 Anhang 2: Ablaufdiagramm ATKIS-Daten beim Kunden

Ablauf der Präsentation beim Kunden: