Technische Universität München

Vergleichende Untersuchungen zur Modellierung und Modelltransformation in der Region Bodensee im Kontext von INSPIRE

Claude Eisenhut

Tatjana Kutzner

Technische Universität München

Vergleichende Untersuchungen zur Modellierung und Modelltransformation in der Region Bodensee im Kontext von INSPIRE

Autoren:	Claude Eisenhut (Eisenhut Informatik AG) Tatjana Kutzner (TUM)
Mitwirkende:	Dieter Heß (LGL BW) DI Stefan Klotz (BEV) Dr. Markus Seifert (LVG BY) Dr. Peter Staub (swisstopo)
Berater:	Dr.-Ing. Andreas Donaubauer (ETH) Dr.-Ing. Andreas Illert (BKG)
Leitung:	Univ.-Prof. Dr.-Ing. Matthäus Schilcher (TUM)

Tatjana Kutzner
Technische Universität München
Fachgebiet Geoinformationssysteme
Arcisstraße 21
80333 München

September 2010
ISBN 978-3-935049-74-0

Inhaltsverzeichnis

1 Motivation und Problemstellung **1**
- 1.1 Geodaten und Interoperabilität . . . 1
- 1.2 Vorteile von Datenmodellen und Modelltransformation . . . 3
- 1.3 Grenzen der Modelltransformation . . . 4
- 1.4 Ziele und Aufbau dieser Studie . . . 5
- 1.5 Dank . . . 7

2 Begriffsbestimmung **9**
- 2.1 Begriffe aus der Modellierungswelt . . . 9
  - 2.1.1 Zum Begriff Modell . . . 10
  - 2.1.2 Modelle und deren Sprache . . . 11
  - 2.1.3 Verwendung von Modellen . . . 15
  - 2.1.4 Modellierung . . . 15
  - 2.1.5 Modellarten . . . 22
- 2.2 Begriffe zu Kodierung und Datentransfer . . . 26
- 2.3 Begriffe aus der Transformationswelt . . . 26
  - 2.3.1 Zum Begriff Modelltransformation . . . 26
  - 2.3.2 Modelltransformation und Transformationssprachen . . . 31
  - 2.3.3 Implementierung der modellbasierten Transformation von Geodaten . 31
- 2.4 Einordnung der Begriffe in OMG, ISO und INSPIRE . . . 32
  - 2.4.1 OMG – MOF und UML . . . 32
  - 2.4.2 ISO 19109 General Feature Model und Anwendungsschema . . . 34
  - 2.4.3 ISO 19103 UML-Profil . . . 37
  - 2.4.4 INSPIRE Generic Conceptual Model . . . 40

3 IST-Situation **43**
- 3.1 Anwendungsfälle . . . 43
  - 3.1.1 Transformation nach INSPIRE . . . 43
  - 3.1.2 Transformation in Fachanwendung . . . 44
- 3.2 UML-Profile . . . 45
  - 3.2.1 AAA-UML-Profil . . . 46
  - 3.2.2 INTERLIS-UML-Profil . . . 50
  - 3.2.3 INSPIRE-UML-Profil . . . 52
  - 3.2.4 Gegenüberstellung der UML-Profile . . . 55

4 Problemstellung **59**
- 4.1 Modellierungssprache . . . 59

4.1.1 UML-Abänderung . . . 59
4.1.2 Unterschiedliche UML-Versionen . . . 60
4.1.3 Ein UML-Profil pro Projekt . . . 60
4.2 Kodierungsregeln . . . 61
4.2.1 Kodierung von Referenzen . . . 61
4.2.2 Kodierung von Identifikatoren . . . 62
4.2.3 Formale Kodierungsregeln . . . 63
4.2.4 Nicht dokumentierte Kodierungsregeln . . . 63
4.2.5 Formale Regeln zur Herleitung von Implementierungsschemata . . . 63
4.2.6 Nicht dokumentierte Herleitung von Implementierungsschemata . . . 63
4.3 Modelle . . . 64
4.3.1 Fehlende konzeptuelle Modelle . . . 64
4.3.2 Einbindung räumlicher Attribute . . . 64
4.3.3 Komplexität der Modelle . . . 65

5 Anforderungen an Transformationssprachen 67
5.1 Sprachparadigma von Transformations- und Modellierungssprache . . . 67
5.2 Komplexe Transformationen . . . 67
5.3 Fehlerbehandlung während der Transformation . . . 67
5.4 Unterstützung unterschiedlicher Modellierungssprachen . . . 68
5.5 Berücksichtigung unterschiedlicher Versionen einer Modellierungssprache . . . 68
5.6 Effizienz bei der Ausführung von Transformationen . . . 68

6 Bestandsaufnahme geeigneter Transformationssprachen 71
6.1 XSL Transformations (XSLT) . . . 71
6.2 MOF 2.0 Query/View/Transformation . . . 72
6.3 Rule Interchange Format (RIF) . . . 72
6.4 FME . . . 73

7 Fazit 75
7.1 Zusammenfassung . . . 75
7.2 Lösungsansätze . . . 76
7.2.1 Lösungsansätze für die Problematik unterschiedlicher UML-Profile . . . 76
7.2.2 Lösungsansätze für andere festgestellte Probleme . . . 78

A Projektvorschlag 79

Abbildungsverzeichnis

1.1 Testregion Bodensee 2
1.2 UML-Profile in der Bodenseeregion 4
1.3 Ziele 6

2.1 Original, Modell und Modellierer 11
2.2 Von der realen Welt zum konzeptuellen Schema 12
2.3 Visuelle und textuelle Darstellung formaler und informeller Sprachen 13
2.4 UML-Modellelement Klasse 14
2.5 Gegenüberstellung der Modellierungsebenen von MDA und Datenbanken . 17
2.6 Profil 20
2.7 Profil 21
2.8 OMG 4-Schichten-Architektur 24
2.9 Implementierungsschema 25
2.10 Modelltransformation – Grundlegende Konzepte 27
2.11 Transformation von Quell- nach Zielmodell 28
2.12 Modellbasierte Transformation von Geodaten 29
2.13 Datentyp, Primitiver Datentyp und Aufzählungstyp 34
2.14 Ausschnitt aus dem General Feature Model 35
2.15 Wertsemantik bei räumlichen Attributen 36
2.16 Referenzsemantik bei räumlichen Attributen 37
2.17 Zusammenhang zwischen GFM, UML, Anwendungsschema und GML . . . 38
2.18 Stereotype «CodeList» und «Union» 40

3.1 Transformation nach INSPIRE 44
3.2 Transformation in Fachanwendung 45
3.3 Erweiterung des General Feature Models im AAA-Basisschema 46
3.4 Erzeugung von Featuretypen mit Geometrie im AAA-Modell 47
3.5 Implementierungsschema 49
3.6 Zusammenhang zwischen INSPIRE-„UML-Profil“, UML, ISO 19103, ISO 19107 und ISO 19109 55

4.1 UML-Abänderung 59
4.2 Kodierung von Referenzen 61
4.3 Kodierung von Identifikatoren 62
4.4 Problematik Wert- und Referenzsemantik 65

Tabellenverzeichnis

2.1 In ISO 19103 definierte Stereotypen 39

3.1 2D-Raumbezugsgrundformen für das AAA-Anwendungsschema 48
3.2 3D-Raumbezugsgrundformen für das AAA-Anwendungsschema 48
3.3 Primitive Datentypen in INTERLIS 51
3.4 Stereotypen in INTERLIS 51
3.5 Geometrische Entsprechungen zwischen der OGC-Simple-Feature-Spezifikation und ISO 19107 53
3.6 Stereotypen des INSPIRE-UML-Profils 54
3.7 Gegenüberstellung der UML-Profile aus Sicht eines Modellierers 57

1 Motivation und Problemstellung

1.1 Geodaten und Interoperabilität

Früher wurden Geodaten von einem Nutzer bzw. einer Organisation selbst erstellt und meist auch nur von diesem Nutzer bzw. innerhalb dieser Organisation verwendet. Wurden Geodaten weiterverbreitet, dann als Dateien auf Speichermedien. Mit der Ausbreitung des Internets ist es möglich geworden, komfortabel auf externe, weltweite Geodatenbestände zuzugreifen und diese in eigene Anwendungen zu integrieren. Geodaten können heute über das Internet bestellt und direkt heruntergeladen bzw. mittels Web-Diensten auch direkt über das Internet visualisiert, analysiert und verändert werden und sind oftmals sogar frei verfügbar. Dieser umfangreiche verteilte und heterogene Datenbestand führt stetig zu neuen Anwendungsbereichen, in denen Geodaten eingesetzt werden können, und umgekehrt wächst der Datenbestand mit den Einsatzmöglichkeiten.

Dabei ist es i. d. R. so, dass jede Organisation bzw. jeder Anwendungsbereich seine Geodaten in unterschiedlichen Systemen (z. B. Geodatenbanken oder GIS) vorhält, unterschiedliche räumliche Bezugssysteme und Datentransferformate verwendet und insbesondere die Geodaten in voneinander abweichenden Datenmodellen beschreibt. Datenmodelle legen fest, welche Objekte der realen Welt mit den Geodaten erfasst werden sollen (wie z. B. Flurstücke, Straßen oder Flüsse), durch welche geometrischen und topologischen Eigenschaften sowie Sachinformationen (z. B. Flurstücksnummer, Straßenname oder Flussname) diese Objekte charakterisiert sind und ob bzw. welche Beziehungen zwischen den Objekten bestehen.

Dadurch, dass Geodaten spezifisch auf eine Organisation bzw. einen Anwendungsbereich zugeschnitten sind, ist ein schneller und problemloser Einsatz der Geodaten in anderen Organisationen und Anwendungsbereichen oftmals nicht möglich. Angenommen, ein Mitarbeiter einer EU-Umweltbehörde möchte länderübergreifende Informationen über den Bodenseeraum aus den Geoinformationssystemen Bayerns, Baden-Württembergs, Österreichs und der Schweiz abrufen, wie in Abbildung 1.1 zu sehen ist. Er wird wohl annehmen, dass die Daten aufgrund der unmittelbar nebeneinander liegenden Gebiete problemlos miteinander kombiniert werden können. In der Realität ist dem jedoch nicht so. Jedes Land hält die Daten in unterschiedlichen Systemen vor, verwendet unterschiedliche räumliche Bezugssysteme und Datentransferformate und nutzt insbesondere auch voneinander abweichende Datenmodelle.

Zwar werden für die Vernetzung heterogener Datenbestände über das Internet heute vielfach Geo Web Services, also Web Services für die Verarbeitung und Bereitstellung von Geodaten, eingesetzt. Insbesondere das Open Geospatial Consortium (OGC), ein Zusammenschluss aus GIS-Anbietern, IT-Firmen, Datenanbietern, Anwendern und Hochschulen,

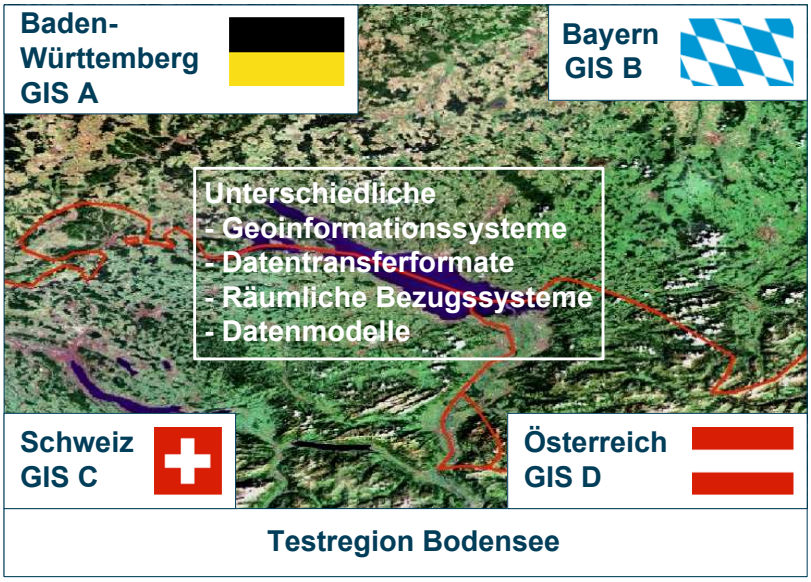


Abbildung 1.1: Testregion Bodensee

widmet sich dem Ziel, Standards für den Zugriff auf Geo Web Services zu erarbeiten. Damit werden jedoch nur der Zugriff auf unterschiedliche Systeme zur Datenhaltung, die räumlichen Bezugssysteme und Datentransferformate standardisiert, das Problem der unterschiedlichen Datenmodelle bleibt weiterhin bestehen.

In diesem Zusammenhang spielt der Begriff Interoperabilität eine große Rolle. Interoperabilität bezeichnet die Fähigkeit heterogener Systeme zur Zusammenarbeit. Interoperabilität bietet die Möglichkeit, transparent auf verschiedenartige Geodaten zuzugreifen und diese in einen einzelnen Arbeitslauf zu integrieren, ohne sie dabei in den eigenen Datenbestand zu überführen [Bill, 1999]. Eine wichtige Entwicklung stellen diesbezüglich so genannte Geodateninfrastrukturen (GDI) dar. Mittels der in einer GDI eingesetzten Standards wird das Zusammenführen heterogener Geodaten auf einfache Weise möglich.

Von besonderer Bedeutung ist dabei der zur Zeit stattfindende Aufbau einer Europäischen Geodateninfrastruktur, welche in der EU-Richtlinie INSPIRE (Infrastructure for Spatial Information in the European Community) festgelegt und definiert wurde [EP, 2007]. Die Richtlinie verlangt von allen Mitgliedstaaten der Europäischen Union, Geodaten über interoperable Geo Web Services und mittels einheitlicher Datentransferformate bereitzustellen. Darüber hinaus werden für bestimmte Themen wie z. B. Verwaltungsgrenzen oder Gewässernetz europaweit einheitliche Datenmodelle, INSPIRE Data Specifications genannt, erstellt.

Für die Anbieter von Geodaten (z. B. Umweltbehörden oder Vermessungsverwaltungen) bedeutet dies nun jedoch nicht, dass sie ihre originären Datenbestände ändern müssen, damit sie den INSPIRE-Datenmodellen entsprechen. Die INSPIRE-Richtlinie sieht hierfür vielmehr so genannte Modelltransformationsdienste vor. Hiermit sollen Daten, die ein Anwender über einen Geo Web Service anfordert, direkt vor der Ausgabe an den Nutzer in die von der EU vorgegebenen Datenmodelle transformiert werden. Die Daten werden dem Nutzer INSPIRE-konform bereit gestellt, ohne dass sich die originären Daten ändern.

Die Notwendigkeit der Transformation zwischen Datenmodellen lässt sich insbesondere an der Bodenseeregion aufzeigen, da hier gleich drei Länder mit verschiedenen Datenmodellen (Deutschland, Österreich und die Schweiz) aneinander grenzen und selbst die Datenmodelle der Bundesländer Baden-Württembergs und Bayerns – wenngleich beide auf ATKIS basieren – kleine Unterschiede aufweisen. Mittels eines Modelltransformationsdienstes soll sich der oben erwähnte Mitarbeiter einer EU-Umweltbehörde die benötigten länderübergreifenden Informationen aus den Geoinformationssystemen Bayerns, Baden-Württembergs, Österreichs und der Schweiz problemlos im INSPIRE-Datenmodell ausgegeben lassen und miteinander kombinieren können.

1.2 Vorteile von Datenmodellen und Modelltransformation

Die Modellierung von Geodaten kann auf unterschiedliche Weise erfolgen. Insbesondere zu unterscheiden ist zwischen der Modellierung auf Ebene der Geodatenformate und zwischen der Modellierung auf konzeptueller Ebene, welche unabhängig von bestimmten Datenformaten erfolgt. Wie bereits aus dem vorherigen Abschnitt hervorgeht, beschäftigt sich die Studie vorrangig mit der Modellierung auf konzeptueller Ebene. Diese Art der Modellierung bringt eine Reihe von Vorteilen mit sich [RTG, 2010]:

- Daten sind langlebiger als Systeme und Formate, durch die Modelle ist Nachhaltigkeit gegeben
- Die automatische Ableitung unterschiedlicher Formate aus ein und demselben Datenmodell ist dadurch möglich
- Gesetzliche Anforderungen sehen die Modellierung auf konzeptueller Ebene vor, wie z. B. bei INSPIRE
- Bei den Überlegungen über das Vertriebspotential von Daten wird geplant, was verkauft werden kann (Inhalt), es wird also ein konzeptuelles Modell entworfen

Auf die gleiche Weise kann auch bei der Transformation unterschieden werden zwischen dem Umformatieren der Daten (z. B. von XML nach Shape) und der semantischen Transformation (z. B. vom deutschen Datenmodell nach INSPIRE). Auch bei der semantischen Transformation können eine Reihe von Vorteilen angeführt werden [RTG, 2010]:

- Systemunabhängigkeit
- Nachhaltigkeit durch Nachnutzung der Geodaten
- Nebenläufige Qualitätssicherung
- Transformationsregeln können wiederverwendet werden
- Fachexperten können mitgestalten ohne durch die technischen (Format-)Details geblendet zu werden

- Visuelle Kommunikationsbasis durch graphische Darstellung der Datenmodelle bzw. Transformation ist gegeben

1.3 Grenzen der Modelltransformation

Wie zu Beginn des Kapitels dargelegt wurde, können Daten aus unterschiedlichen Quellen nur dann vollständig miteinander kombiniert werden, wenn die dazugehörigen Datenmodelle identisch sind. Ansonsten ist es notwendig, die Daten semantisch zu transformieren, wodurch ihnen ein gemeinsames Datenmodell zugrunde gelegt wird.

Für die Beschreibung von Datenmodellen werden so genannte Modellierungssprachen eingesetzt. Inbesondere die Modellierungssprache UML (Unified Modeling Language) hat eine große Verbreitung gefunden. So wurden z. B. auch die Datenmodelle der von den Vermessungsverwaltungen geführten Geobasisdaten und auch die INSPIRE-Datenmodelle mit UML erstellt.

Eine Modellierungssprache besteht aus einer Menge von Funktionalitäten und Konstrukten, welche für die Beschreibung der Datenmodelle verwendet werden können. Der Umfang an Funktionalitäten und Konstrukten ist sehr groß, was je nach Aufgabe dazu führen kann, dass zu viele Freiheiten bei der Modellierung bestehen. Aus diesem Grund ist es möglich, die Menge der Funktionalitäten und Konstrukte einzuschränken. Darüber hinaus kann eine Modellierungssprache auch erweitert werden. Dies ist von Vorteil, wenn bestimmte Modellierungskonstrukte wiederholt verwendet werden. Mittels einer Erweiterung können diese Konstrukte auf einfachere Weise beschrieben und eingesetzt werden. In beiden Fällen spricht man von einem Profil der Modellierungssprache.

Bezogen auf den Bodenseeraum stellt sich die Situation wie in Abbildung 1.2 gezeigt dar. Sowohl Deutschland als auch die Schweiz und die EU verwenden für die Erstellung der

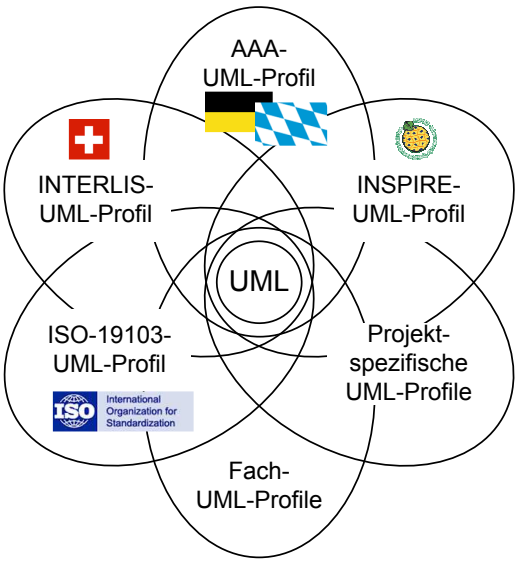


Abbildung 1.2: UML-Profile in der Bodenseeregion

Datenmodelle die Modellierungssprache UML, jedoch liegen den Datenmodellen jeweils unterschiedliche UML-Profile zugrunde. Deutschland und INSPIRE verwenden ein UML-Profil, welches in der Norm ISO 19103 definiert ist, die Schweiz dagegen nutzt ein UML-Profil, welches auf dem Schweizer Standard INTERLIS basiert. Zudem wird UML kontinuierlich weiterentwickelt, so dass zusätzlich auch unterschiedliche Versionen der UML zum Einsatz kommen können, was in Deutschland (UML-Version 1.4.2) und bei INSPIRE (UML-Version 2.1) der Fall ist.

Wie in Abbildung 1.2 zu sehen ist, existieren Datenmodelle nicht nur auf Länderebene, sondern sie können auch in Geobasisdaten-nutzenden Stellen (Fachverwaltungen) vorkommen. Hier werden teilweise wieder andere Profile verwendet, die speziell auf bestimmte Fachanwendungen angepasst sind. Darüber hinaus zeigt die Abbildung, dass im Prinzip jedes Projekt ein eigenes Profil besitzen kann. Es existiert somit eine große Anzahl an projektspezifischen Profilen, jedes davon mit eigenen Einschränkungen und Erweiterungen.

Damit kann festgehalten werden, dass zu einer Modellierungssprache sehr unterschiedliche Profile existieren können. Darüber hinaus kann auch der Fall eintreten, dass gänzlich andere Modellierungssprachen für die Beschreibung von Datenmodellen eingesetzt werden.

An dieser Stelle sei zudem darauf hingewiesen, dass auch das beste Modell die Semantik der modellierten Objekte nicht vollständig beschreiben kann. Jeder Maschine fehlt die Intelligenz bzw. die Vorstellungskraft, wie sie der Mensch besitzt. Hört bzw. liest ein Mensch den Begriff Haus, so kann er diesen Begriff in seiner Vorstellung konkret umsetzen. Eine Maschine dagegen kann sich keine Vorstellung von dem Begriff machen. Auch eine ausführlichere Beschreibung des Begriffs durch Attribute stellt für die Maschine keine Hilfe dar.

1.4 Ziele und Aufbau dieser Studie

Diese Studie hat seinen Ausgangspunkt im Forschungs- und Entwicklungsprojekt „Modellbasierter Ansatz für den Web-Zugriff auf verteilte Geodaten am Beispiel grenzübergreifender GIS-Anwendungen (mdWFS)“, welches seit 2006 im Auftrag des Bundesamts für Kartographie und Geodäsie an der Technischen Universität München bearbeitet wird. In den ersten beiden Jahren waren zudem die Eidgenössische Technische Hochschule Zürich und deren Auftraggeber swisstopo als weitere Projektpartner beteiligt.

Ziel des mdWFS-Projekts ist es, einen Lösungsansatz für die Transformation von Geodatenmodellen zu erarbeiten, welcher auf der Modelltransformation basiert und in eine webbasierte Umgebung eingebunden ist. Derzeit ist eine Transformation jedoch nur mit einheitlichem UML-Profil möglich, weshalb die AAA- und INSPIRE-Modelle in INTERLIS nachmodelliert werden. Diese Lösung ist jedoch unbefriedigend und stellt im produktiven Betrieb einen nicht realistischen Aufwand dar. Aus diesem Grund wurde seitens der Mitarbeiter des mdWFS-Projekts eine Studie angeregt, welche sich ausführlicher mit der Problematik der unterschiedlichen UML-Profile beschäftigt. Finanziert wurde die Studie vom Landesamt für Geoinformation und Landentwicklung Baden-Württemberg (LGL BW), der Bayerischen Vermessungsverwaltung (LVG BY), dem Österreichischen Bundesamt für Eich-

und Vermessungswesen (BEV) sowie dem Schweizer Bundesamt für Landestopografie (swisstopo).

Abbildung 1.3 stellt anschaulich dar, mit welchen Fragestellungen sich diese Studie beschäftigt. Ein Hauptziel der Studie ist es, generell ein Bewusstsein zu schaffen für die Problematik, die sich durch das Vorhandensein verschiedener UML-Profile ergibt. Es soll aufgezeigt werden, welche Auswirkungen dies auf die Transformation von Datenmodellen hat. Ein weiteres Problem ergibt sich bei der Überführung der Datenmodelle in Transferformate, was als Kodierung bezeichnet wird. Die Datenmodelle Deutschlands, der Schweiz und der EU werden anhand unterschiedlicher Kodierungsregeln in das Transferformat GML überführt. Dies kann dazu führen, dass die GML-Dateien – obwohl es sich um das gleiche Transferformat handelt – nicht miteinander kombinierbar sind.

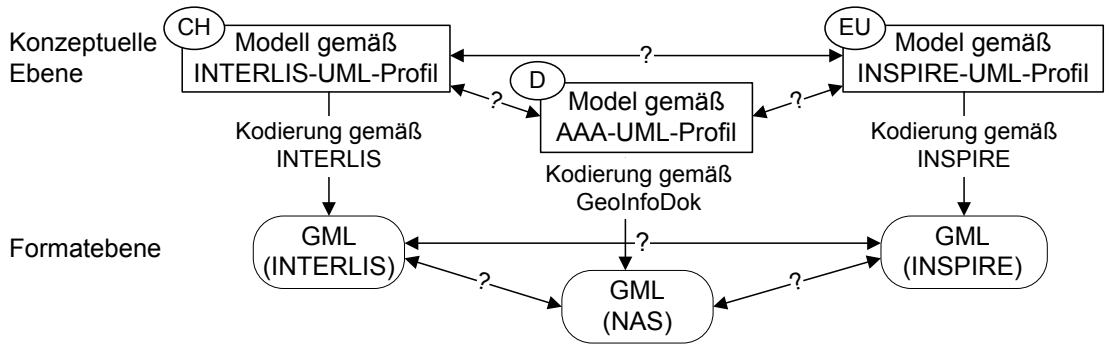


Abbildung 1.3: Ziele

Die Studie folgt dabei dem Leitgedanken, dass das Modell maschinenlesbar und maschineninterpretierbar sein muss. Andernfalls kann das Potenzial der semantischen Transformation nicht vollständig ausgeschöpft werden.

Kapitel 1 gibt eine allgemeine Einführung in die Problematik in der Sprache des Anwenders. Kapitel 2 definiert eine Reihe von Begriffen aus den Bereichen Modellierung, Kodierung und Transformation. Mit dieser Begriffsbestimmung soll sowohl für den Leser der Studie wie auch für die nachfolgenden Kapitel eine einheitliche Grundlage geschaffen werden. Darüber hinaus wird eine Einordnung der Begriffe in für die Studie relevante Standards und Normen aus dem GIS-Bereich vorgenommen. Kapitel 3 befasst sich basierend auf zwei Anwendungsfällen zur Modelltransformation mit der IST-Situation in Deutschland, in der Schweiz und in der EU, d. h., wie sind die Datenmodelle definiert und welche Unterschiede existieren zwischen ihnen. Darauf aufbauend werden in Kapitel 4 anschaulich konkrete Probleme beschrieben. Kapitel 5 definiert anschließend allgemeine Anforderungen an Transformationssprachen, mit denen sich die beschriebenen Probleme lösen lassen. In Kapitel 6 werden ansatzweise einzelne Sprachen für die Transformation von Datenmodellen vorgestellt. Kapitel 7 schließlich präsentiert mögliche Lösungsansätze.

1.5 Dank

Die Verfasser der Studie danken den Vermessungsverwaltungen Baden-Württembergs, Bayerns, Österreichs und der Schweiz für die finanzielle Förderung der Studie. Ein besonderer Dank gilt auch den Mitwirkenden der Vermessungsverwaltungen und den externen Beratern für ihre wertvollen Anregungen und Diskussionen.

Der Projektvorschlag für diese Studie wurde am 09. September 2009 bei den Vermessungsverwaltungen eingereicht und von diesen am 18. September 2009 im Rahmen der Informationstagung der Vermessungsverwaltungen der Bodensee-Anrainerländer positiv entschieden. Die Durchführung der Studie fand vom 01. Januar 2010 bis 31. Juli 2010 statt.

2 Begriffsbestimmung

Nachfolgend werden eine Reihe von Begriffen, die in der Modellierung wie auch in der Transformation von Geodaten gebräuchlich sind, näher erläutert. Diese Begriffe werden in der entsprechenden Literatur wie auch in der Praxis häufig verwendet, jedoch manchmal mit einer unterschiedlichen Bedeutung. Mit dieser Begriffsbestimmung soll sowohl für den Leser der Studie wie auch für die nachfolgenden Kapitel eine einheitliche Grundlage geschaffen werden.

Um eine bessere Übersichtlichkeit zu gewährleisten, werden die Begriffe in drei Gruppen unterteilt, in Begriffe aus der Modellierungswelt, in Begriffe zu Kodierung und Datentransfer und in Begriffe aus der Transformationswelt. Mittels „(→ Abschnitt xxx)“ wird dabei im Text auf andere Begriffe der Begriffsbestimmung verwiesen.

2.1 Begriffe aus der Modellierungswelt

konzeptuell – konzeptionell In deutschen Texten zum Thema Modellierung und Modelltransformation sind häufig die Begriffe *konzeptuell* und *konzeptionell* anzutreffen. Beide Begriffe leiten sich zwar vom englischen Begriff *conceptual* ab, sie besitzen jedoch eine leicht voneinander abweichende Bedeutung:

- *konzeptuell* = ein Konzept aufweisend [Hesse et al., 2008]
- *konzeptionell* = ein Konzept betreffend [Hesse et al., 2008]
  = die Konzeption betreffend, in Bezug auf die Konzeption [Duden, 2006]

Hesse und Mayr weisen in [Hesse et al., 2008] darauf hin, dass der Begriff *konzeptionell* fälschlicherweise oft synonym zum Begriff *konzeptuell* verwendet wird. Diese Studie wird deshalb ausschließlich den Begriff *konzeptuell* verwenden, welcher in Zusammenhang mit dem hier behandelten Thema als der Richtigere erscheint.

Maschineninterpretierbarkeit Unter *Maschineninterpretierbarkeit* versteht man in der Informatik ganz allgemein, dass ein Text von einem Computerprogramm gelesen und ausgeführt werden kann. Dies bedeutet, dass der Text bzw. im Kontext dieser Studie Modelle (→ Abschnitt 2.1.1) so genau strukturiert sein müssen, dass sie von dem Programm vollständig verstanden und abgearbeitet werden können. Es dürfen für das Programm keine Widersprüche bzw. Entscheidungsspielräume mehr existieren. Die Modelle dienen damit in gewisser Weise zur Steuerung von Laufzeitsystemen (→ Abschnitt 2.1.3). Der Begriff Maschineninterpretierbarkeit bezieht sich dagegen nicht darauf, ob bzw. inwieweit Modelle von bestimmten Werkzeugen gelesen werden können.

2.1.1 Zum Begriff Modell

Modellbegriff der Informatik Je nach Anwendungsbereich, in dem Modelle eingesetzt werden, existieren unterschiedliche Modellbegriffe. So versteht man unter einem Architekturmodell die maßstäbliche Darstellung eines Entwurfs, während mathematische Modelle versuchen, die wesentlichen Parameter von Phänomenen mathematisch zu erfassen, wodurch z. B. Prognosen zum Klimawandel abgegeben werden können oder die Statik eines Gebäudes errechnet werden kann [Wikipedia, 2010a]. Insbesondere interessant für diese Studie ist der Modellbegriff der Informatik. Modelle sind hier meist *sprachliche Repräsentationen* in Form von geschriebenen oder gesprochenen Texten, Bildern oder Grafiken. Sie verweisen auf das, was repräsentiert werden soll und stellen somit eine *Abstraktion des Modellierten* dar [Hesse et al., 2008].

Die Verwendung von Modellen in der Informatik ist dabei keineswegs eine neue Entwicklung, vielmehr sind Modelle im Datenbankbereich schon seit Mitte der 1970er verbreitet. So wurde der Begriff *konzeptuelles Schema* (→ gleicher Abschnitt weiter unten) bereits 1975 in einer Veröffentlichung des Standards Planning and Requirements Committee (SPARC) des American National Standards Institute (ANSI) verwendet [Steel, 1975] und 1977 wurde von E. Falkenberg festgehalten, dass ein konzeptuelles Schema alle relevanten semantischen Aspekte und sonst nichts beinhalten soll [Falkenberg, 1977]. Auch spricht Falkenberg bereits davon, dass eine Transformation (→ Abschnitt 2.3) zwischen Daten semantisch äquivalent durchzuführen ist.

Merkmale von Modellen Bereits 1973 hat H. Stachowiak drei wichtige Merkmale definiert, die ein Modell kennzeichnen [Hesse et al., 2008]:

1. *Abbildungs-Merkmal*: Jedes Modell steht für *sein* Original. Mit Original kann alles gemeint sein, was in der realen Welt vorkommt, also Gegenstände, aber auch Nicht-Materielles oder Zusammenhänge. Beispiele für Originale aus dem Bereich der Geoinformation sind Gebäude, Grundstücke und Flüsse, komplette Infrastrukturen wie Leitungsnetze und Verkehrswege, oder auch Geländeformen der Erdoberfläche. Die Beziehung zwischen Original und Modell kann vielschichtig sein und ist immer von der Sichtweise des jeweiligen Modellierers bzw. Modellnutzers geprägt, wie Abbildung 2.1 zeigt.

2. *Reduktionsmerkmal*: Ein Modell weist nicht alle Eigenschaften des Originals auf, sondern nur einige und selbst diese können verändert sein. Modelle sind somit oft Vereinfachungen der realen Welt, die für einen bestimmten Anwendungszweck erstellt wurden. Dabei können Informationen, die als nicht relevant gelten, vereinfacht oder sogar gänzlich weggelassen worden sein, um die Beschreibung verständlicher zu machen [Devillers et al., 2006] oder um sie an ein bestimmtes Anwendungsgebiet anzupassen.

3. *Pragmatisches Merkmal*: Ein Modell hat den Zweck, unter bestimmten Bedingungen und bezüglich bestimmter Fragestellungen das Original zu ersetzen. Die Modellierung findet somit immer zweck-, kultur- und umfeldbedingt statt.

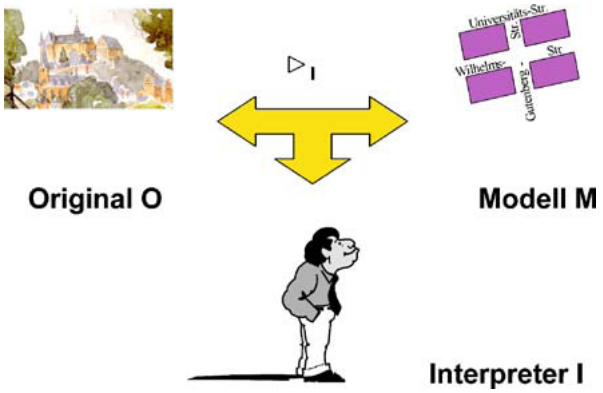


Abbildung 2.1: Original, Modell und Modellierer (Quelle: [Hesse et al., 2008])

Universe of Discourse – Konzeptuelles Modell – Konzeptuelles Schema Das Modell der realen Welt erhält man durch Abstrahierung einer bestimmten, meist vom jeweiligen Anwendungsgebiet geprägten Sicht der Wirklichkeit. In diesem Zusammenhang wird auch der Begriff *Universe of Discourse* bzw. *Realweltausschnitt* verwendet. Dabei handelt es sich um eine Sicht auf die reale oder gedachte Welt, die alles enthält, was von Interesse ist. Dieser Realweltausschnitt ist jedoch immer ein gedachtes Modell und nicht niedergeschrieben.

Wird dieser Realweltausschnitt informell (→ Abschnitt 2.1.2) niedergeschrieben, so spricht die Norm *ISO 19101 Geographic information – Reference model* von einem *conceptual model* (dt. konzeptuelles Modell), wohingegen das *conceptual schema* (dt. konzeptuelles Schema) ein formal (→ Abschnitt 2.1.2) niedergeschriebenes Modell darstellt [ISO, 2002a]. Abbildung 2.2 stellt diesen Sachverhalt anschaulich dar. Die Beschreibung der konzeptuellen Modelle und der Schemata erfolgt dabei mithilfe von *konzeptuellen Schemasprachen* (engl. conceptual schema language), welche auf einem oder mehreren bestimmten *Sprachparadigmen* (conceptual formalism) basieren (→ Abschnitt 2.1.4).

Des Weiteren findet sich in der Norm ISO 19101 folgender Grundsatz: „The Conceptualisation principle states [...] that a conceptual schema should contain only those structural and behavioural aspects, that are relevant to the universe of discourse. All aspects of physical exernal or internal data representation should be excluded. This requires the production of a conceptual schema, which is independent with respect to physical implementation technologies and platforms“ [ISO, 2002a]. Das konzeptuelle Schema umfasst somit nur solche Informationen, die auch im Universe of Discourse enthalten sind. Informationen zur physischen Implementierung des Modells darf das konzeptuelle Schema dagegen auf keinen Fall beinhalten.

2.1.2 Modelle und deren Sprache

formal – informell, visuell – textuell Die Erstellung von Modellen erfolgt mittels einer Sprache. Es wird dabei zwischen *formalen* und *informellen Sprachen* unterschieden. Formale Sprachen sind maschinenlesbar und darüber hinaus maschineninterpretierbar (→ Abschnitt 2.1) und sie besitzen genaue Regeln. Zu den formalen Sprachen zählen Program-

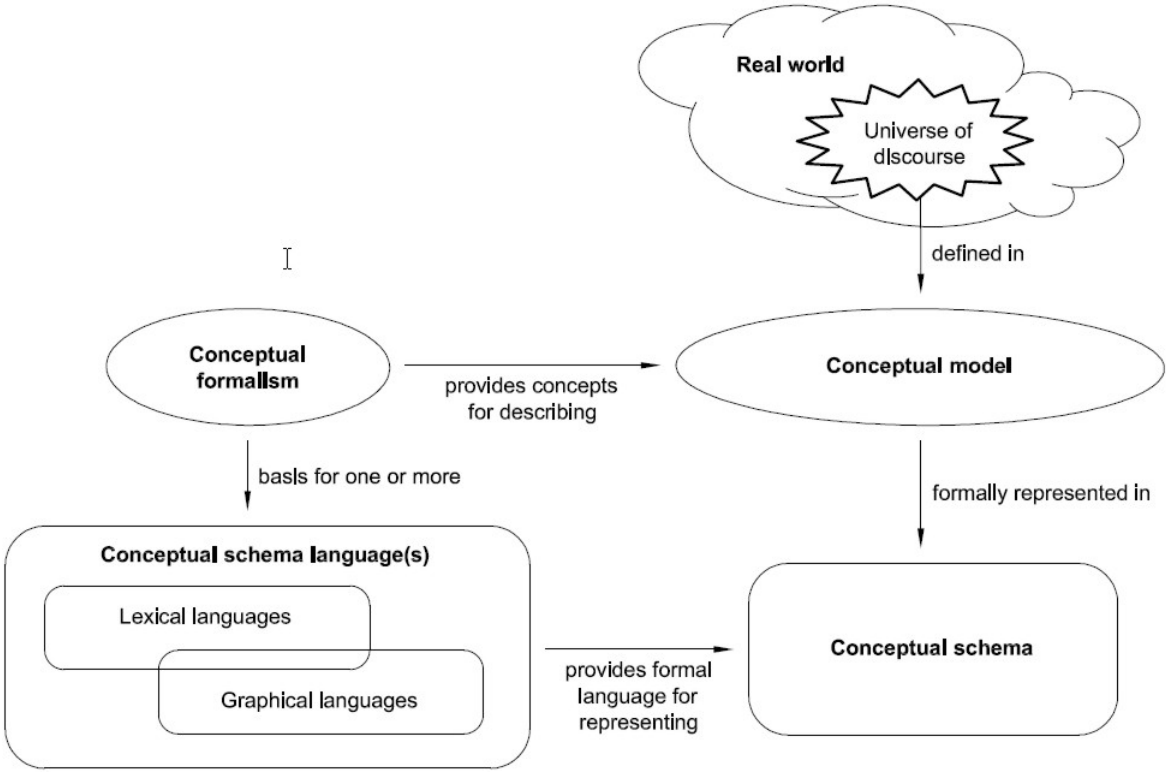
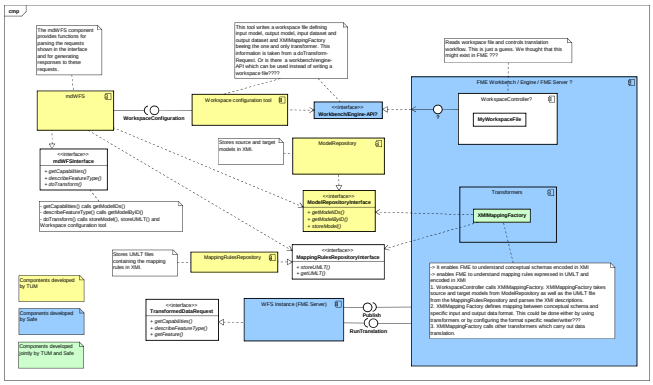


Abbildung 2.2: Von der realen Welt zum konzeptuellen Schema (Quelle: [ISO, 2002a])

miersprachen und Modellierungssprachen. Natürliche Sprachen dagegen, wie Deutsch oder Englisch, sind informell. Beide Arten von Sprachen können sowohl in *visueller* wie auch in *textueller* Form vorkommen, was in Abbildung 2.3 beispielhaft dargestellt ist. Zu den formalen, visuellen Sprachen zählt die Modellierungssprache Unified Modeling Language (UML) (→ Abschnitt 2.4.1). Abbildung 2.3(a) zeigt ein entsprechendes Modell, das mit dieser Sprache erstellt wurde. Formal und textuell dagegen ist die Extensible Markup Language (XML) in Abbildung 2.3(b), welche für die formale Strukturierung von Text eingesetzt wird. Auch die Geography Markup Language (GML) ist damit eine formale, textuelle Sprache, da sie auf XML basiert. Darüber hinaus zählen auch Programmiersprachen wie Java oder C++ zu den formalen, textuellen Sprachen. Als Beispiel für eine informelle, visuelle Sprache ist in Abbildung 2.3(c) eine protoelamische Bilderschrift dargestellt, welche vor 5000 Jahren zur Kommunikation diente. In der Gegenwart kommen dagegen informelle, textuelle Sprachen zum Einsatz, wie z. B. die deutsche Sprache, mit der die vorliegende Studie erstellt wurde.

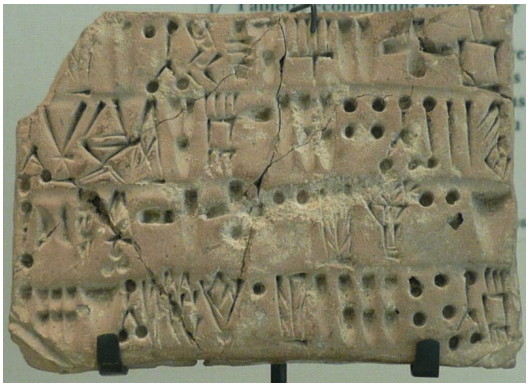
Alle vier Sprachvarianten können für die Beschreibung von Modellen eingesetzt werden. Bezogen auf Abbildung 2.2 eignen sich die informellen Sprachen somit insbesondere für die Beschreibung der konzeptuellen Modelle (→ Abschnitt 2.1.1), welche den Realweltausschnitt in informeller Weise wiedergeben. Die konzeptuellen Schemata (→ Abschnitt 2.1.1) dagegen beschreiben den Realweltausschnitt auf formale Weise, so dass hierfür vorrangig formale Sprachen einzusetzen sind.



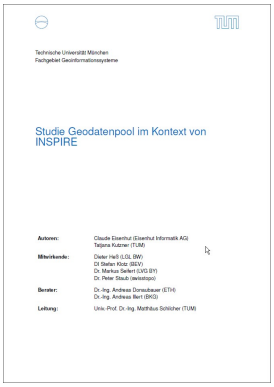
(a) Formal + visuell = UML-Modell

```
<gml:surfaceProperty>
  <gml:Surface>
    <gml:patches>
      <gml:PolygonPatch>
        <gml:exterior>
          <gml:LinearRing>
            <gml:posList>4450119.82 5
                4450099.55 5339213.15
            </gml:LinearRing>
          </gml:exterior>
        </gml:PolygonPatch>
    </gml:patches>
  </gml:Surface>
</gml:surfaceProperty>
```

(b) Formal + textuell = XML-Datei



(c) Informell + visuell = Bilderschrift (Quelle: [Wikipedia, 2010b])



(d) Informell + textuell = Die vorliegende Studie

Abbildung 2.3: Visuelle und textuelle Darstellung formaler und informeller Sprachen

Syntax Jede formale und informelle Sprache besitzt eine *Syntax*. Beim Text in Abbildung 2.3(c) ist davon auszugehen, dass er, abgesehen von Archäologen, von keinem heute lebenden Menschen verstanden wird. Offensichtlich wurden aber beim Schreiben des Textes Regeln eingehalten, wie z. B. dass die Zeichen in Zeilen stehen. Solche Regeln sind syntaktische Regeln, sie können überprüft werden, ohne dass der Inhalt verstanden werden muss. Die Syntax steht somit für die Form des Textes, d.h. für Regeln, nach denen Texte strukturiert werden müssen [Tantau, 2006].

Die *Syntax einer informellen Sprache* umfasst die Menge an Regeln, nach denen Sätze gebildet werden dürfen. Bedeutung oder Sinn der gebildeten Sätze sind dabei unwichtig. Jede Sprache hat ihre eigene Syntax, wobei sich aber die Syntaxen verschiedener Sprachen durchaus ähneln können wie es beispielsweise bei Deutsch und Englisch der Fall ist.

Die *Syntax einer formalen Sprache* dagegen umfasst die Menge an Regeln, nach denen Programmtexte bzw. Modelle gebildet werden dürfen. Auch hier sind Bedeutung oder Sinn der Texte und Modelle egal und die Syntaxen verschiedener Programmiersprachen bzw. Modellierungssprachen können sich ebenfalls ähneln [Tantau, 2006].

Semantik Neben der Syntax ist jeder formalen und informellen Sprache auch eine *Semantik* zu eigen. Der Satz „Heute ist schönes Wetter“ hat eine Bedeutung. Die Semantik legt solche Bedeutungen fest, wobei syntaktisch falschen Sätzen im Allgemeinen keine Bedeutung zugewiesen wird. Ein Satz kann aber auch mehrere Semantiken besitzen, wodurch sich unterschiedliche Bedeutungen ergeben. Der Satz „Steter Tropfen höhlt den Stein“ sagt in der wortwörtlichen Semantik aus, dass Steine ausgehöhlt werden, wenn lange Zeit Wasser auf sie tropft. In der übertragenen Semantik dagegen besagt der Satz, dass sich Beharrlichkeit auszahlt [Tantau, 2006].

Auf die gleiche Weise haben auch Programmtexte und Modelle eine Bedeutung. Die Semantik von Programmiersprachen bzw. Modellierungssprachen legt dabei fest, was mit einem Programmtext bzw. einem Modell gemeint ist. Die Semantik steht somit für die Zuordnung von Bedeutung zu Syntax (Text bzw. Bildsymbol).

Syntax und Semantik am Beispiel UML Anhand der Modellierungssprache UML soll der Unterschied zwischen Syntax und Semantik noch einmal exemplarisch aufgezeigt werden. In UML existiert das *Modellelement Klasse*. Die Syntax legt die Form des Modellelements Klasse fest. Da UML eine visuelle Modellierungssprache ist, existiert ein Bildsymbol für Klasse, welches in Abbildung 2.4 dargestellt ist. Mit der Semantik wird dagegen die Bedeutung des Bildsymbols festgelegt. Eine Klasse beschreibt demnach Objekte gleicher Bedeutung und Struktur und kann einen Klassennamen, Attribute, Methoden und Beziehungen besitzen.

Objekte sind die individuellen Exemplare einer Klasse. Die Klasse aus Abbildung 2.4 beschreibt beispielsweise Flurstücke. Jedes einzelne Flurstück, das durch die Klasse repräsentiert wird, besitzt die Attribute nutzung, flaeche und extentOf sowie die Methoden setNutzung und getNutzung, mit denen der Wert des Attributs nutzung gesetzt und abgefragt werden kann.

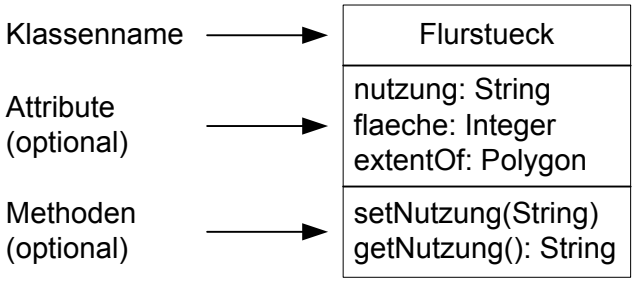


Abbildung 2.4: UML-Modellelement Klasse

Domänenspezifische Sprache Darüber hinaus existiert der Begriff der *domänenspezifischen Sprache* (engl. domain-specific language, *DSL*). Bei einer DSL handelt es sich um eine formale Sprache, die für einen bestimmten Einsatzbereich (Domäne) entwickelt wurde, um Probleme zu lösen, die speziell in dieser Domäne auftreten [Wikipedia, 2010c]. Hierzu gehören z. B. Sprachen wie Prolog und Mathematica, aber auch die Datenbankabfragesprache SQL. Domänenspezifische Sprachen sind somit das Gegenteil von universell

einsetzbaren Programmiersprachen (engl. general-purpose language) wie Java bzw. von universell einsetzbaren Modellierungssprachen wie UML.

Der Bereich der Geoinformation ist auch eine Domäne, in welcher der Einsatz von DSLs für die Modellierung von Geodaten vorstellbar wäre. In der Praxis wird jedoch bisher meist die universelle Modellierungssprache UML verwendet. Einzig die Schweiz verwendet mit dem Schweizer Standard *Interlis* [INTERLIS, 2010] eine domänenspezifische Modellierungssprache für Geodaten.

2.1.3 Verwendung von Modellen

Mit Modellen können unterschiedliche Ziele verfolgt werden, weshalb Modelle je nach Verwendungszweck unterschiedlichen Ansprüchen gerecht werden müssen. Folgende Möglichkeiten der Verwendung von Modellen sind im Rahmen dieser Studie relevant:

- *Modelle zur Kommunikation zwischen Menschen*: Möchten sich z. B. Entwickler und Anwender über bestimmte Ideen oder Vorstellungen austauschen, kann es von Vorteil sein, wenn diese Ideen schriftlich vorliegen. Insbesondere Modelle stellen dabei eine gute Hilfe dar und können im Sinne einer Ideenskizze verwendet werden. Das Modell dient hier in erster Linie dazu, ein gemeinsames Verständnis zwischen Menschen zu schaffen. Modelle müssen in diesem Stadium nicht maschineninterpretierbar (→ Abschnitt 2.1) sein.

- *Modelle für die Softwareentwicklung*: In der Softwareentwicklung dienen Modelle häufig als Ausgangsmodell, woraus andere Modelle generiert werden. Dies ist beispielsweise der Fall beim AAA-Modell und den INSPIRE Data Specifications, wo aus den konzeptuellen UML-Modellen Implementierungsschemata (→ Abschnitt 2.1.5) generiert werden. Bei diesen Modellen muss Maschineninterpretierbarkeit (→ Abschnitt 2.1) gegeben sein.

- *Modelle zur Steuerung von Laufzeitsystemen*: Bei diesem Verwendungszweck besitzen Modelle die Aufgabe, ein System zu steuern. Modelle dieser Art werden z. B. im Rahmen des in Abschnitt 3.1.1 vorgestellten Forschungsprojekts mdWFS eingesetzt. In diesem Projekt werden die Transformationsregeln (→ Abschnitt 2.3.2) zur Transformation von Modellen mittels einer auf UML basierenden Transformationssprache beschrieben. Anschließend können die Transformationsregeln von einer Software interpretiert werden und es kann schließlich automatisch die Transformation durchgeführt werden. Damit Modelle zur Steuerung von Laufzeitsystemen eingesetzt werden können, muss das Modell vollständig maschineninterpretierbar (→ Abschnitt 2.1) sein.

2.1.4 Modellierung

Die Modellierung von Geodaten kann auf unterschiedlichen Ebenen durchgeführt werden. So kann die Modellierung zum einen abhängig von dem System, auf dem die Daten eingesetzt werden, und dem Format, in dem die Daten verwendet werden, erfolgen. Zum

anderen können Datenmodelle aber auch unabhängig von spezifischen Systemen und Formaten erstellt werden – oft verbunden mit dem Ziel, aus den Modellen automatisch mittels Kodierungsregeln (→ Abschnitt 2.2) ein formatspezifisches Modell abzuleiten. Diese unterschiedlichen Ebenen sind auch bei der objektorientierten Softwareentwicklung und bei der Modellierung von Datenbankanwendungen zu finden und werden nachfolgend näher betrachtet.

Damit Modellierung jedoch überhaupt stattfinden kann, werden Modellierungssprachen benötigt. Der Einsatz einer bestimmten Modellierungssprache hängt dabei auch von dem gewählten Sprachparadigma ab. Auch hierauf geht dieser Abschnitt detaillierter ein.

Modellierungsebenen der Model-Driven Architecture Die *Model-Driven Architecture* (MDA) definiert einen modellbasierten Ansatz für die Softwareentwicklung. Der MDA-Ansatz wurde von der Object Management Group (OMG) (→ Abschnitt 2.4.1) entwickelt. In einem Modell sind die fachlichen Geschäftsprozesse oft mit technischen Informationen vermischt. MDA dagegen nimmt hier eine Trennung vor und definiert Modelle, die das zu entwickelnde System aus den folgenden unterschiedlichen Gesichtspunkten beschreiben:

- *Computation Independent Model* (CIM) für die Beschreibung der fachlichen Anforderungen unabhängig von technischen Aspekten
- *Platform Independent Model* (PIM) für die *plattformunabhängige* Modellierung
- *Platform Specific Model* (PSM) für die *plattformabhängige* Modellierung
- *Platform Model* (PM) mit dem Programmcode

Im PIM wird das fachliche Wissen (Fachlogik) des Softwaresystems, wie z. B. Geschäftsprozesse oder Fachverfahren, technologieunabhängig und abstrakt erfasst und modelliert (*plattformunabhängig*). Im Gegensatz dazu wird im PSM die Implementierungstechnologie berücksichtigt, d. h. die technischen Aspekte bezogen auf eine konkrete Plattform, welche normalerweise weniger abstrakt ist (*plattformabhängig*). Die explizite Trennung der Fachlogik von der Implementierungstechnologie gewährleistet, dass beide Teile unabhängig voneinander wiederverwendet werden können. Auch der Programmcode wird bei der MDA als Modell betrachtet. Programmcode kann nämlich als eine Abstraktion des Maschinencodes angesehen werden, der vom Compiler aus dem Programmcode generiert wird.

Was die Abstraktion bei den Modellierungsebenen der MDA insgesamt betrifft, so ist der Grad der Abstraktion auf CIM-Ebene am größten und nimmt in Richtung PM-Ebene schrittweise ab.

Modellierungsebenen bei Datenbanksystemen Beim Entwurf einer Datenbankanwendung lassen sich drei Abstraktionsebenen unterscheiden [Kemper et al., 2009]:

- *Konzeptuelle Ebene*: Diese Ebene dient dazu, den Universe of Discourse (→ Abschnitt 2.1.1), welcher mit der Datenbankanwendung erfasst werden soll, zu strukturieren. Die Modellierung erfolgt dabei unabhängig vom verwendeten Datenbanksystem. Das

Ergebnis dieses konzeptuellen Entwurfs ist ein konzeptuelles Schema (→ Abschnitt 2.1.1), das den Universe of Discourse formal wiedergibt.

- *Implementierungsebene / logische Ebene*: Hier erfolgt die Modellierung in den Konzepten, d. h. im Datenmodell (→ Abschnitt 2.1.5) des zum Einsatz kommenden Datenbanksystems. Mögliche Datenmodelle sind z. B. das relationale Modell (→ gleicher Abschnitt weiter unten), das objektorientierte Modell (→ gleicher Abschnitt weiter unten) oder das objektrelationale Modell (→ gleicher Abschnitt weiter unten).

- *Physische Ebene*: Auf dieser Ebene geht es darum, die Performanz einer Datenbankanwendung zu erhöhen, indem Strukturen wie z. B. Datenblöcke, Zeiger und Indexstrukturen für die Speicherung der Daten eingesetzt werden.

Der Begriff konzeptuelle Ebene wird in der Datenbankwelt jedoch auch unterschiedlich verwendet. Zum einen steht er für ein Teilmodul in der Architektur eines Datenbankmanagementsystems (DBMS), welche 1975 von ANSI-SPARC definiert wurde (ANSI-SPARC-Architektur) [Steel, 1975]. Zum anderen repräsentiert der Begriff, so wie oben beschrieben, einen Zwischenschritt in der Entwicklung eines DBMS und kann dabei als ungenaues bzw. vages Zwischenprodukt bei der Konkretisierung hin zu einem konzeptuellen Modell im Sinne des ANSI-SPARC-Architekturmoduls betrachtet werden.

Gegenüberstellung MDA und Datenbankanwendung Abbildung 2.5 stellt die soeben beschriebenen Modellierungsebenen einander gegenüber. Dabei ist zu erkennen, dass das Platform Independent Model (PIM) und die konzeptuelle Ebene auf der gleichen Stufe stehen. Beide modellieren den Universe of Discourse unabhängig von der zum Einsatz kommenden Plattform. Dagegen erfolgt beim Platform Specific Model (PSM) wie auch bei der Implementierungsebene die Modellierung in den Konzepten der verwendeten Plattform, womit diese beiden ebenfalls als äquivalent zu betrachten sind.

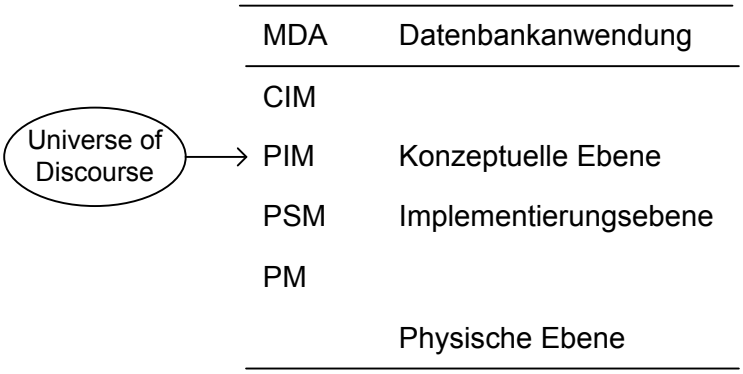


Abbildung 2.5: Gegenüberstellung der Modellierungsebenen von MDA und Datenbanken

Sprachparadigmen Jede Modellierungssprache weist ein bestimmtes Sprachparadigma auf. Der Einsatzbereich einer Modellierungssprache hängt deshalb immer auch von dem

dieser Sprache zugrunde liegenden Sprachparadigma ab. Folgende Sprachparadigmen sind für diese Studie und die hier behandelte Thematik relevant:

- *Relationales Paradigma*: Dieses Paradigma wurde bereits 1970 von E. F. Codd vorgestellt [Codd, 1970] und ist bis heute beim relationalen Datenbankmodell im Einsatz. Grundlage dieses Paradigmas ist die *Relation*. Hierbei handelt es sich um eine Tabelle, die in mathematischer Form beschrieben ist. Jede Tabelle besteht aus Spalten und Zeilen, wobei die Spalten der Tabelle als *Attribute* bezeichnet werden und die Zeilen als *Tupel*. Für jedes Attribut wird zudem festgelegt, welche Werte das Attribut annehmen darf. Jedes Tupel besteht somit aus so vielen Attributwerten wie die Tabelle Spalten besitzt, was im Datenbankbereich auch als ein *Datensatz* bezeichnet wird.

- *Entity-Relationship-Paradigma* (ER-Paradigma): Nur wenige Jahre später, 1976, wurde von P. Chen das Entity-Relationship-Modell (ER-Modell) präsentiert [Chen, 1976]. Grundlagen des ER-Modells sind die *Entität* (engl. entity) und die *Beziehung* (engl. relationship). Entitäten stehen für Realweltobjekte und Beziehungen für Verknüpfungen zwischen den Entitäten. Das ER-Modell wird in erster Linie für die Definition konzeptueller Datenmodelle eingesetzt. Es findet eine rein statische Modellierung statt, das dynamische Verhalten von Entitäten und Beziehungen wird nicht beschrieben.

- *Objektorientiertes Paradigma* (OO-Paradigma): Bei diesem Paradigma stehen die Fachkonzepte und deren dynamisches Zusammenspiel im Mittelpunkt der Modellierung. Das Paradigma basiert auf der gleichen Idee wie das ER-Paradigma. Es wurde jedoch um das Konzept der *Methoden* ergänzt, womit die dynamischen Aspekte von Software modelliert werden können. Zudem werden die Entitäten hier als *Klasse* bezeichnet und die Beziehungen als *Assoziation*. Die Attribute können beim OO-Paradigma komplexe und benutzerdefinierte Datentypen als Wert besitzen. Dies ist insbesondere für Geodaten wichtig, z. B. zur Modellierung von Geometrien. Darüber hinaus enthält das objektorientierte Paradigma die Konzepte Objektidentität, Vererbung, Polymorphie sowie Datenkapselung.

- *Objektrelationales Paradigma* (OR-Paradigma): Dieses Paradigma ist überwiegend im Datenbankbereich in so genannten objektrelationalen Datenbanken zu finden [Kemper et al., 2009]. Das OR-Paradigma stellt, wie der Name bereits vermuten lässt, eine Kombination des ER-Paradigmas mit Konzepten aus dem OO-Paradigma dar. Insbesondere wird dabei das ER-Paradigma um die Konzepte Vererbung, komplexe und benutzerdefinierte Datentypen sowie Objektidentität erweitert.

- *XML-Paradigma*: XML (Extensible Markup Language) ist eine Auszeichnungssprache (engl. Markup Language) zur Beschreibung und Strukturierung von Daten. Grundlage von XML bilden *Elemente*, welche die Daten enthalten und diese auch näher beschreiben. Dadurch, dass Elemente weitere Elemente als Subelemente enthalten dürfen, ergibt sich eine hierarchische Datenstruktur.

- *RDF-Paradigma*: RDF (Resource Description Framework) ist eine Sprache, mit der Aussagen über Ressourcen im Internet getroffen werden können. Jede Ressource wird

dabei durch einen eindeutigen Bezeichner (URI) identifiziert. Die Aussagen werden anhand von RDF-Modellen modelliert, welche auf Graphen basieren. Grundlagen eines RDF-Modells sind *Subjekt* (die Ressource, über die eine Aussage getroffen wird), *Prädikat* (eine Eigenschaft des Subjekts) und *Objekt* (das Argument des Prädikats).

- *Paradigma der regelbasierten Strukturbeschreibung*: Unter diesem Paradigma werden Programme zusammengefasst, welche die Struktur von Daten beschreiben. Ein Beispiel hierfür ist Schematron, eine Schemasprache zur Validierung von Inhalt und Struktur von XML-Dokumenten.

Konzeptuelle Schemasprache Ein Begriff, der im Zusammenhang mit Modellierungssprachen häufig gebraucht wird, ist *CSL* (Conceptual Schema Language) (dt. konzeptuelle Schemasprache). Dieser Begriff wurde erstmals 1979 von B. Breutmann, E. Falkenberg und R. Mauer in der wissenschaftlichen Veröffentlichung „CSL: A Language for Defining Conceptual Schemas“ eingeführt [Falkenberg et al., 1979]. In diesem Beitrag wurde eine Datendefinitionssprache beschrieben, mit der konzeptuelle Schemata (→ Abschnitt 2.1.1) definiert werden können. Diese Datendefinitionssprache wurde CSL genannt, d.h., der Begriff CSL ist in gewisser Weise ein „Produktname“ für diese eine bestimmte Sprache.

Heute jedoch wird der Begriff CSL – insbesondere im Kontext von Geoinformation und im Zusammenhang mit ISO-Normen – losgelöst von seiner ursprünglichen Bedeutung verwendet. Zum einen ist der Begriff ein Oberbegriff für konzeptuelle Modellierungssprachen im Allgemeinen. Zum anderen verwendet ihn die Norm ISO 19103 (→ Abschnitt 2.4.3) als Synonym für ein dort definiertes UML-Profil (→ gleicher Abschnitt weiter unten).

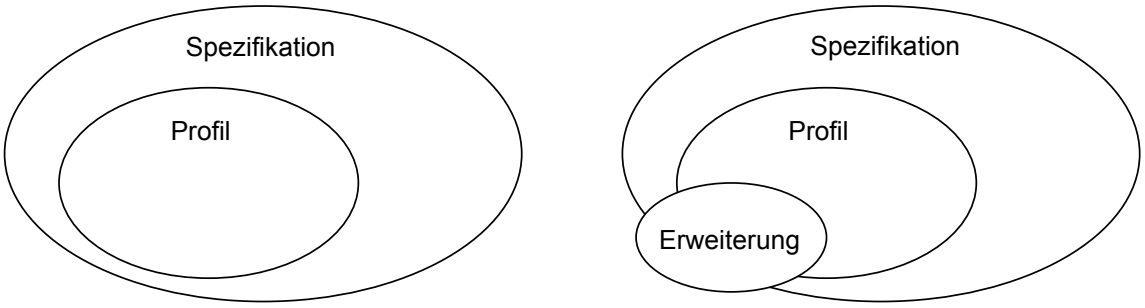
Profil gemäß ISO 19106 Die Norm *ISO 19106 Geographic Information – Profiles* definiert Richtlinien für die Erstellung von Profilen zu den Normen der Normenserie ISO 19100 [ISO, 2004]. Die Definition basiert auf einer allgemeineren Definition, welche in Teil 1 des Technical Report *ISO/IEC TR 10000-1 Information technology – Framework and taxonomy of International Standardized Profiles* [ISO, 1998] beschrieben ist.

Die Konformität eines Profils zu der Norm ISO 19106 kann auf zwei verschiedene Arten sichergestellt werden [ISO, 2004]:

- „Conformance class 1 is satisfied when a profile is established as a pure subset of the ISO geographic information standards.“

- „Conformance class 2 allows profiles to include extensions within the context permitted in the ISO geographic information standard and permits the profiling of non-ISO geographic information standards as parts of profiles.“

Dies bedeutet, dass ein Profil entweder eine Einschränkung (Conformance class 1) oder eine Erweiterung (Conformance class 2) einer Spezifikation darstellt. Bei einer Einschränkung besteht das Profil nur aus einer Teilmenge der von einer Spezifikation angebotenen Konstrukte (Abbildung 2.6(a)), bei einer Erweiterung dagegen enthält das Profil Konstrukte, die in der Spezifikation selbst nicht existieren, jedoch gemäß den Vorgaben der Spezifikation bezüglich Erweiterungen erstellt werden dürfen (Abbildung 2.6(b)).



(a) Profil: Einschränkung (b) Profil: Erweiterung

Abbildung 2.6: Profil

UML-Profil Für Profile der Modellierungssprache UML (→ Abschnitt 2.4.1) liefert die UML-Spezifikation des OMG eine eigene Definition. Wichtige Punkte hierbei sind [OMG, 2007b]:

- „A profile must provide mechanisms for specializing a reference metamodel (such as a set of UML packages) in such a way that the specialized semantics do not contradict the semantics of the reference metamodel. That is, profile constraints may typically define well-formedness rules that are more constraining (but consistent with) those specified by the reference metamodel."

- „The profiles mechanism is not a first-class extension mechanism (i.e., it does not allow for modifying existing metamodels). Rather, the intention of profiles is to give a straightforward mechanism for adapting an existing metamodel with constructs that are specific to a particular domain, platform, or method."

Ein *UML-Profil* gestattet somit nur Einschränkungen der Modellierungssprache UML. Erweiterungen sind gemäß der UML-Definition von Profilen nicht erlaubt.

Für die Definition eines entsprechenden Profils stellt UML die Konzepte *Stereotypen*, *Tagged Values* (Eigenschaften) und *Constraints* (Einschränkungen) zur Verfügung. Das UML-Metamodell (→ Abschnitt 2.1.5) bleibt durch die Verwendung der Konzepte unverändert. Dadurch sollen UML-Modelle, die ein Profil verwenden, weiterhin problemlos mit jeder UML-Software verarbeitbar sein.

Stereotype legen fest, wie vorhandene Modellelemente der UML so an bestimmte Anwendungsbereiche angepasst werden können, dass damit bestimmte Konzepte dieses Anwendungsbereichs ausgedrückt werden können. Stereotype werden immer zwischen Guillemets (« ») angegeben. Stereotype können Eigenschaftsdefinitionen (engl. tag definitions) besitzen. Bei Anwendung des Stereotyps können diese Definitionen durch Eigenschaftswerte (engl. tagged values) ergänzt werden. Die Eigenschaftswerte werden dem UML-Modellelement, auf welches der Stereotyp angewendet wurde, als Kommentar hinzugefügt. Des Weiteren können die Eigenschaften eingeschränkt werden. Für die Formulierung entsprechender Einschränkungen stellt OMG die Sprache OCL (Object Constraint Language) zur Verfügung.

Am Beispiel des UML-Modellelements *Klasse* aus Abbildung 2.4 soll dies im Folgenden anschaulich erläutert werden: Abgebildet ist eine UML-Klasse, die Flurstücke repräsentiert.

Die einzelnen, konkreten Flurstücke sind die individuellen Exemplare dieser Klasse und werden als Objekte bezeichnet. Im Bereich der Geoinformation wird für solche Objekte oft auch der Begriff *Feature* verwendet; jedes individuelle Flurstück stellt somit ein Feature dar. Features gleicher Bedeutung und Struktur werden durch so genannte *Featuretypen* repräsentiert, was dem UML-Modellelement Klasse entspricht. Damit nun bei Verwendung des UML-Modellelements Klasse erkennbar ist, dass es sich um einen Featuretyp im Sinne der Geoinformation handelt, wird dem Klassensymbol einfach ein entsprechender Vermerk in Form eines Stereotyps hinzugefügt. Wie in Abschnitt 3.2 noch erläutert werden wird, wird hierfür beim AAA-Modell und bei INSPIRE ein Stereotyp namens «featureType» eingesetzt. Abbildung 2.7 zeigt die Klasse ergänzt um den Stereotypen. Dadurch erkennt der Nutzer des Modells sofort, dass es sich bei der modellierten Klasse um den Featuretyp Flurstück handelt.

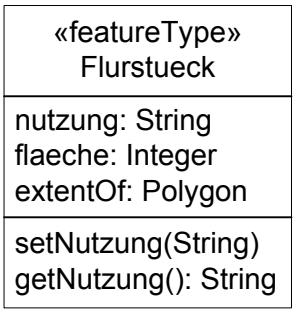


Abbildung 2.7: Profil

Die Definition von UML-Profilen stellt somit, insbesondere da UML eine universale Modellierungssprache ist, eine einfache Möglichkeit dar, UML an bestimmte Einsatzbereiche bzw. Domänen anzupassen und im Sinne einer DSL (→ Abschnitt 2.1.2) zu verwenden. UML-Profile sollten insbesondere eingesetzt werden, wenn [Born et al., 2004]:

- Das Anwendungsgebiet eine spezifische Terminologie verwendet, die auch bei den Elementen des Modells zum Einsatz kommen soll, wie z. B. oben am Featuretyp Flurstück dargestellt.
- Sprachkonstrukte eingesetzt werden sollen, für die in UML keine Notation vorgegeben ist
- die Semantik bestimmter Sprachkonstrukte eingeschränkt werden soll

Im Gegensatz dazu existiert auch die Möglichkeit, das UML-Metamodell selbst zu erweitern. Hierbei werden jedoch Änderungen am UML-Metamodell vorgenommen, wodurch die UML-Modelle nicht mehr problemlos mit jeder UML-Software verarbeitbar sind. Dieses Vorgehen wird in der Modellierung von Geodaten in Deutschland, Österreich und der Schweiz bisher jedoch nicht in Betracht gezogen und ist deshalb auch nicht Gegenstand dieser Studie.

Ontologie *Ontologien* stellen in der Geoinformatik gegenwärtig ein großes Forschungsthema dar. Nach Gruber [Gruber, 1995] kann eine Ontologie definiert werden als „explicit specification of a conceptualization“. Eine Ontologie definiert Vokabular und Konzepte für die Beschreibung und Repräsentation eines bestimmten Wissensgebietes. Mittels Ontologien wird versucht, die Welt maschineninterpretierbar (→ Abschnitt 2.1) zu beschreiben, so dass die Maschine diese Interpretation verarbeiten kann. Eine Ontologie stellt somit ein Modell zur Steuerung von Laufzeitsystemen (→ Abschnitt 2.1.3) dar, welches von Softwareprogrammen ausgewertet wird.

Ontologien sollen so u.a. den Austausch von Informationen über bestimmte Domänen unterstützen. Gemäß Obrst [Obrst, 2003] basieren Ontologien auf den Konzepten Klasse, Instanz, Beziehung, Eigenschaft, Funktion und Prozess sowie Einschränkung und Regel. Dies ist äquivalent zu den Konzepten, die von UML (→ Abschnitt 2.4.1) einschließlich OCL zur Verfügung gestellt werden. UML und OCL stellen somit ein Mittel unter vielen anderen dar, um Ontologien – in Form von UML-Modellen – formal zu beschreiben. UML-Modelle können aus diesem Grund auch als Ontologien angesehen werden.

2.1.5 Modellarten

Bei der Modellierung kann zwischen verschiedenen Modellarten unterschieden werden, wovon die für diese Studie relevanten Modellarten im Nachfolgenden aufgeführt werden.

Datenmodell Ein *Datenmodell* bezeichnet ein Modell, welches speziell zur Beschreibung von Daten dient. Datenmodelle entsprechen dem conceptual schema in Abbildung 2.2. Beispiele sind das AAA-Modell sowie die INSPIRE Data Specifications, welche beide als konzeptuelles Datenmodell vorliegen und mittels UML definiert sind.

Produktionsmodell Beim *Produktionsmodell* handelt es sich um ein Datenmodell für die Produktion bzw. Erfassung von Daten, welches vorrangig in der Schweiz zum Einsatz kommt. So kann beispielsweise die Datenerfassung auf dem Feld (z. B. Erfassung von Geometrie und Art der Objekte), die Erfassung weiterer Daten im Büro (z. B. Alter der Objekte) sowie die Nachführung der Daten (z. B. neue Erfassung der Geometrien) gemäß je einem unterschiedlichen Produktionsmodell erfolgen. Das Produktionsmodell kann dabei auch Informationen enthalten, die später u. U. nicht veröffentlicht werden. Auch können aus dem Produktionsmodell je nach Kontext verschiedene Publikationsmodelle abgeleitet und befüllt werden.

Publikationsmodell Ein *Publikationsmodell* beschreibt Daten, die z. B. im WebGIS publiziert werden. Dieses Modell kann gegenüber dem Produktionsmodell stark vereinfacht sein. Publikationsmodelle werden auch als *Produkt-* oder *Nutzermodelle* bezeichnet.

Eine spezielle Form hiervon ist das *Interface-/Austausch-Modell*. Beispielsweise werden in der Schweiz bei der amtlichen Vermessung (AV) zwar Adresse, Grundstück und Bodenbedeckungsfläche erfasst, jedoch keine Gebäude. Damit diese Daten mit dem Grundbuchsystem (Nicht-GIS) genutzt werden können, wo Gebäude in Bezug zu einem Grundstück benötigt

werden, wird eine Verschneidung von Grundstück und Bodenbedeckungsfläche durchgeführt, woraus auf der AV-Seite schließlich Gebäude abgeleitet und ans Grundbuch übermittelt werden können. Hieraus ergibt sich das Austausch-Modell. Dieses Modell kann nicht nur weniger Informationen enthalten, sondern auch komplett andere Strukturen besitzen.

Ereignismodell Beim *Ereignismodell* werden nicht Daten bzw. der Datenbestand an sich modelliert, sondern die Datenstruktur der Ereignisse und Meldungen, die zu einer Änderung am Datenbestand führen können, wie z. B. ein Umzug oder eine Parzellierung.

Statik- bzw. Dynamikmodell Modelle können danach unterschieden werden, ob eher statische oder dynamische Aspekte modelliert werden sollen. *Statische Modelle* beziehen sich auf Konstellationen von Gegenständen und Beziehungen, die zu einem bestimmten Zeitpunkt auftreten. *Dynamische Modelle* dagegen beschreiben einen Vorgang, eine Aktion oder einen Prozess. Zum Beispiel ist ein Auftrag statisch, wohingegen eine Auftragsänderung eher dynamisch modelliert wird [Hesse et al., 2008].

Ein Beispiel für ein dynamisches Modell ist das *Prozessmodell*. Prozessmodelle können sowohl organisationsübergreifend wie auch innerhalb einer Organisation zum Einsatz kommen.

Metamodell Ein *Metamodell* ist ein Modell, das Konstrukte enthält, mit denen ein anderes Modell beschrieben werden kann. Das Konzept des Metamodells lässt sich sehr gut anhand der *4-Schichten-Architektur* beschreiben, die von der OMG (→ Abschnitt 2.4.1) definiert wurde. Mit dieser Architektur ist es möglich, alle Arten von Daten, deren Strukturen und deren Datenflüsse als Modell zu beschreiben (= mit einer Modellierungssprache zu modellieren) sowie diese Beschreibungen (= Modellierungssprachen) selbst wieder zu beschreiben (= mit einer Modellierungssprache zu modellieren).

Die 4-Schichten-Architektur der OMG ist in Abbildung 2.8 dargestellt. Die einzelnen Schichten haben folgende Bedeutung:

- *Ebene M0* (Info Layer): Beinhaltet die Geodaten selbst.
- *Ebene M1* (Model Layer): Beinhaltet das Modell, mit dem die Daten aus Ebene M0 beschrieben werden. Hierzu zählt z. B. das AAA-Modell.
- *Ebene M2* (Metamodel Layer): Beinhaltet die Beschreibung der Struktur und Semantik des Modells aus Ebene M1. Diese Beschreibung wird als Modellierungssprache bezeichnet. Eine Modellierungssprache ist das *Metamodell* für Modelle aus der Ebene M1. Ein Beispiel für ein solches Metamodell ist die Modellierungssprache UML, mit der das AAA-Modell aus Ebene M1 modelliert wurde.
- *Ebene M3* (Metametamodel Layer): Beinhaltet die Beschreibung zu Struktur und Semantik der Metamodelle aus Ebene M2. Bei der OMG wurde hierfür der Standard Meta Object Facility (MOF) (→ Abschnitt 2.4.1) entwickelt, mit dem z. B. die Modellierungssprache UML definiert wurde. Modellierungssprachen auf dieser Ebene werden

als *Metametamodell* bezeichnet. Die Ebene M3 ist rekursiv, d.h. MOF beschreibt nicht nur Metamodelle der Ebene M2, sondern auch sich selbst. Andernfalls könnte dieses Prinzip unendlich lange fortgesetzt werden.

Zusammenfassend kann der Begriff Metamodell folgendermaßen definiert werden: Ein Metamodell ist ein Modell, das Konstrukte enthält, mit denen Modelle der darunter liegenden Ebene beschrieben werden können. Somit sind auch Metametamodelle nichts anderes als Metamodelle, welche in diesem Fall die Modelle der Ebene M2 beschreiben.

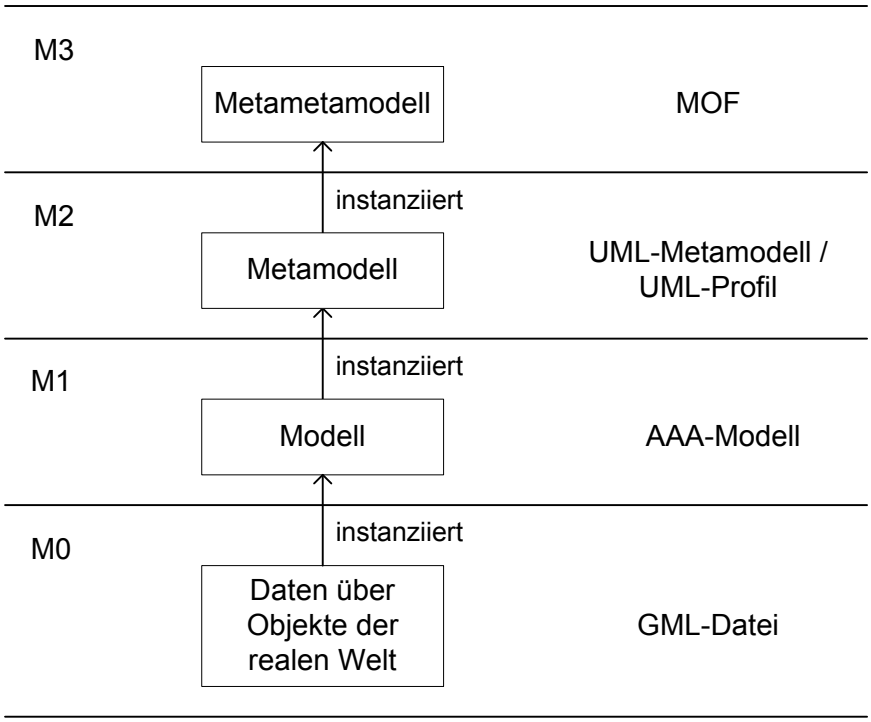


Abbildung 2.8: OMG 4-Schichten-Architektur

Anwendungsschema Unterschiedliche Anwendungsbereiche bedingen unterschiedliche Geodaten. Ein *Anwendungsschema* ist ein konzeptuelles Schema (→ Abschnitt 2.1.1), das die semantische Struktur von Geodaten beschreibt, die in einem oder mehreren bestimmten Anwendungsbereichen zum Einsatz kommen. Die Beschreibung erfolgt mittels einer formalen Sprache (→ Abschnitt 2.1.2). Ein anderer Begriff für Anwendungsschema ist *Applikationsschema*. Beispielsweise stellt die INSPIRE Data Specification für Protected Sites ein Anwendungsschema dar, das definiert, welche Informationen zum Thema Schutzgebiete relevant sind und in welcher Struktur diese Informationen durch Geodaten repräsentiert werden sollen.

Gemäß der Norm ISO 19101 ist ein Anwendungsschema definiert als “conceptual schema for data required by one or more applications“ [ISO, 2002a]. Ein GML-Anwendungsschema (XML-Schema) ist deshalb kein Anwendungsschema im Sinne eines konzeptuellen Schemas, auch wenn es so heißt. Ein GML-Anwendungsschema beschreibt nämlich nicht

nur die semantische Struktur eines Datensatzes, sondern insbesondere auch die Syntax (→ Abschnitt 2.1.2) der entsprechenden GML-Dateien.

Oftmals werden GML-Anwendungsschemata jedoch zuerst mittels UML definiert und anschließend gemäß den Kodierungsregeln (→ Abschnitt 2.2) der Norm *ISO 19136 Geographic information – Geography Markup Language (GML)* Annex E [ISO, 2007a] umgesetzt. Siehe hierzu auch Abbildung 2.17, wo u.a. der Zusammenhang zwischen Anwendungsschema, GML-Anwendungsschema und GML-Datei dargestellt ist.

Implementierungsschema Ein *Implementierungsschema* beschreibt ein Modell in den Konzepten der Plattform, auf der die modellierten Geodaten zur Verfügung stehen sollen. In Abschnitt 2.1.4 wurden die Modellierungsebenen von MDA und Datenbanksystemen einander gegenübergestellt. Das Implementierungsschema fügt sich hier nahtlos ein. Es ist plattformspezifisch und stellt somit ein Platform-Specific Model (PSM) dar bzw. entspricht im Datenbankbereich der Implementierungsebene.

Implementierungsschemata bilden eine Zwischenstufe zwischen den konzeptuellen, plattformunabhängigen Schemata und den Geodaten, siehe Abbildung 2.9. Implementierungsschemata kommen auch in den in dieser Studie untersuchten UML-Profilen zum Einsatz (→ Abschnitt 3.2). Unter Verwendung von Implementierungsschemata erfolgt der Übergang vom konzeptuellen Schema zu den Geodaten in zwei Schritten: Zuerst wird das konzeptuelle Schema (PIM) in ein Implementierungsschema (PSM) überführt und dieses anschließend in ein Datentransferformat (PM) (→ Abschnitt 2.2) wie beispielsweise GML (Geography Markup Language).

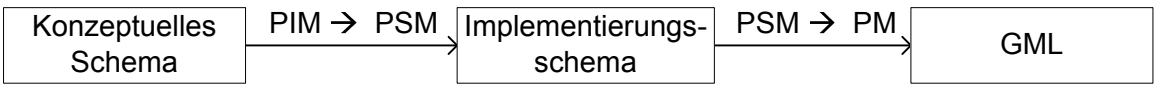


Abbildung 2.9: Implementierungsschema

Modellebene – Formatebene – Datenebene Bei der Modellierung und auch bei der Transformation tauchen häufig die Begriffe *Modellebene*, *Formatebene* und *Datenebene* auf. Die Begriffe lassen sich folgendermaßen in die hier vorgestellten Konzepte einordnen:

- *Modellebene*: Entspricht der Ebene M1 der OMG 4-Schichten-Architektur (→ Abschnitt 2.1.5), d. h., zu dieser Ebene zählen z. B. Datenmodelle und Anwendungsschemata.

- *Formatebene*: Diese Ebene gehört zur Modellebene, jedoch sind die Modelle hier an die physische Darstellungsform des verwendeten Formats gebunden. So gehören z. B. XML-Schemata zur Formatebene.

- *Datenebene*: Entspricht der Ebene M0 der OMG 4-Schichten-Architektur (→ Abschnitt 2.1.5), d. h., auf dieser Ebene befinden sich die Daten. Die Datenebene kann auch als *Instanzebene* bezeichnet werden.

2.2 Begriffe zu Kodierung und Datentransfer

Transferformat Beim *Transferformat* handelt es sich um ein Datenformat, mit dem digitale Daten zwischen verschiedenen Systemen ausgetauscht werden können. Ein im GI-Bereich sehr bekanntes Transferformat ist die Geography Markup Language (GML), mit der Geodaten im Format XML zwischen Systemen übertragen werden können.

Das konzeptuelle Modell legt die Semantik (→ Abschnitt 2.1.2) der Geodaten fest. Werden die im Modell beschriebenen Geodaten in einem Transferformat wiedergegeben (d. h. kodiert), dann ist zu beachten, dass die im Modell festgelegte Semantik weder im Implementierungsschema noch im Transferformat geändert werden darf. Die Daten dürfen ausschließlich um implementierungsspezifische Details ergänzt werden.

Transferformatschema Das *Schema eines Transferformats* legt die Struktur des Transferformats fest. Das Transferformat GML beispielsweise ist eine Sprache, die auf XML basiert. Dementsprechend wird die Struktur von GML-Dateien mittels XML-Schema beschrieben.

Kodierung Der Begriff *Kodierung* (engl. encoding) bezeichnet den Vorgang der Überführung eines konzeptuellen Schemas in ein Transferformat. Das konzeptuelle Schema wird dabei anhand von Kodierungsregeln und gegebenenfalls über ein Implementierungsschema (→ Abschnitt 2.1.5) als Zwischenschritt in das gewünschte Transferformatschema überführt. Zudem werden die Daten, die auf dem konzeptuellen Schema beruhen, anhand von weiteren Kodierungsregeln in das gewählte Transferformat überführt.

Kodierungsregeln *Kodierungsregeln* (engl. encoding rules) definieren die einzelnen Schritte, die bei der Kodierung von einem konzeptuellen Schema in ein Transferformat durchzuführen sind. Aus einem Datenmodell können unter Zuhilfenahme von Kodierungsregeln und geeigneter Compiler praktisch beliebige Transferformate automatisch abgeleitet werden. Gerade hierfür ist die Maschineninterpretierbarkeit (→ Abschnitt 2.1) der Modelle von größter Wichtigkeit.

2.3 Begriffe aus der Transformationswelt

Neben der Modellierung von Modellen ist auch die maschinelle Verarbeitbarkeit der Modelle wichtig. Insbesondere die Modelltransformation spielt dabei in der GIS-Welt eine große Rolle. Modelltransformation ist jedoch noch ein recht junges Betätigungsfeld, welches aus dem Bereich des Software Engineering hervorgegangen ist. Bisher existieren nur wenige Standards und Normen.

2.3.1 Zum Begriff Modelltransformation

Modelltransformation Mittels *Modelltransformation* können aus vorhandenen Modellen neue Modelle erzeugt werden und es können aus vorhandenen Modellen auch Code, Daten

oder Transferformate generiert werden. Die Modelltransformation an sich ist unabhängig von einem bestimmten Paradigma. Ein Haupteinsatzbereich der Modelltransformation liegt darin, konzeptuelle Schemata (→ Abschnitt 2.1.1) aufeinander abzubilden, indem ein oder mehrere *Quellschemata* in ein oder mehrere *Zielschemata* überführt werden. Modelltransformation wird immer dann eingesetzt, wenn sich die Modelle der *Quellsysteme* von den Modellen der *Zielsysteme* unterscheiden.

Abbildung 2.10 stellt die grundlegenden Konzepte der Modelltransformation zwischen einem Quell- und einem Zielmodell dar. Die Transformation wird bezogen auf die Metamodelle (→ Abschnitt 2.1.5) definiert, wofür geeignete, mittels einer Transformationssprache (→ Abschnitt 2.3.2) definierte Transformationsregeln (→ Abschnitt 2.2) eingesetzt werden. Die Transformationsregeln legen fest, wie die Abbildung durchzuführen ist. Das Quellmodell muss deshalb konform zu seinem Quell-Metamodell sein und ebenso das Zielmodell zu seinem Ziel-Metamodell. Ziel- und Quell-Metamodell dürfen sich jedoch voneinander unterscheiden. Durchgeführt wird die Transformation auf den Modellen selbst und nicht auf den Metamodellen. Dazu liest ein Transformationswerkzeug das Modell des Quellsystems, führt die Transformationsregeln auf dem Modell aus und gibt schließlich ein neues Modell aus, welches konform zur Semantik des Zielsystems ist. Ziel sollte dabei immer sein, dass die Transformation der Modelle automatisch durchgeführt wird.

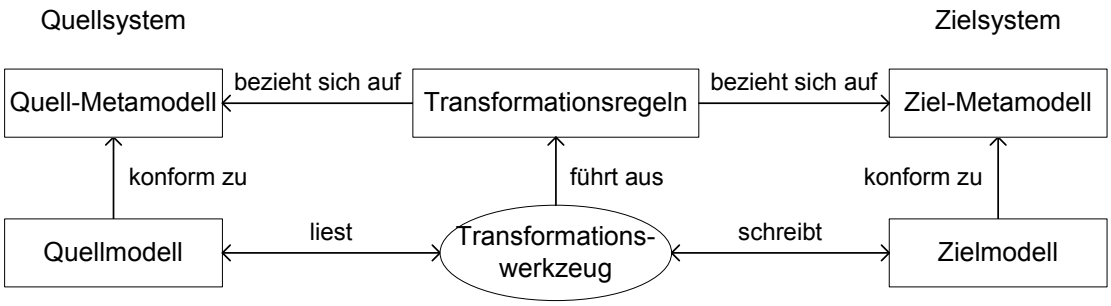


Abbildung 2.10: Modelltransformation – Grundlegende Konzepte (Quelle: [Czarnecki et al., 2006], verändert)

Damit Modelle, die in visueller Form (→ Abschnitt 2.1.2) vorliegen, von einem Transformationswerkzeug gelesen bzw. ausgegeben werden können, wird ein Transferformat (→ Abschnitt 2.2) für die Modelle benötigt. Die Abbildung visueller Modelle in ein Transferformat erfolgt mittels Kodierungsregeln (→ Abschnitt 2.2). So können beispielsweise UML-Modelle mittels des Transferformats XML Metadata Interchange (XMI) in textueller Form (→ Abschnitt 2.1.2) wiedergegeben und durch ein Transformationswerkzeug verarbeitet werden.

Vertikale und horizontale Transformation In Abschnitt 2.1.4 wurden die verschiedenen Modellierungsebenen der MDA vorgestellt. Neben der Modellierung ist auch die Transformation ein wichtiger Bestandteil der MDA. Erfolgt die Transformation innerhalb derselben Ebene (z. B. PIM → PIM), so spricht man von *horizontaler Transformation* oder auch von *Modell-nach-Modell-Transformation* (engl. Model-to-Model bzw. M2M). Findet die Transformation

dagegen zwischen verschiedenen Ebenen, d. h. von einer übergeordneten in eine darunterliegende Ebene statt (z. B. PIM → PSM, PSM → PM), dann handelt es sich um eine *vertikale Transformation*. Bei der vertikalen Transformation ist zu beachten, dass die Semantik, die in der übergeordneten Modellebene definiert wird, bei der Abbildung auf die darunterliegende Ebene nicht verändert werden darf. Mittels der vertikalen Transformation können z. B. Programmcode, Datentransferformate oder Dokumente generiert werden, weshalb sie auch als *Modell-nach-Text-Transformation* (engl. Model-to-Text bzw. M2T) bezeichnet wird. Die Transformation von Modellen in Geodaten bzw. in ein Transferformat (→ Abschnitt 2.2) ist identisch mit dem Begriff Kodierung (→ Abschnitt 2.2). Die Transformationsregeln entsprechen damit den Kodierungsregeln (→ Abschnitt 2.2).

Beispiel einer horizontalen Transformation Ein typisches Beispiel für eine horizontale Transformation ist in Abbildung 2.11 dargestellt. Das Quellschema repräsentiert die Klasse Gebäude, das Zielschema die Klasse Haus. Ziel ist es nun, die Informationen aus der Klasse Gebäude in die Klasse Haus zu überführen. Damit die Transformation korrekt durchgeführt werden kann, muss in einer entsprechenden Transformationsregel definiert werden, dass die Klasse Gebäude auf die Klasse Haus abzubilden ist. Zudem ist die Adresse des Gebäudes in die Attribute PLZ, Straße und Hausnummer aufgeteilt, wohingegen sie beim Haus aus einem Attribut besteht. Hierfür ist ebenfalls eine geeignete Transformationsregel zu definieren, welche die Abbildung zwischen den Attributen durchführt.

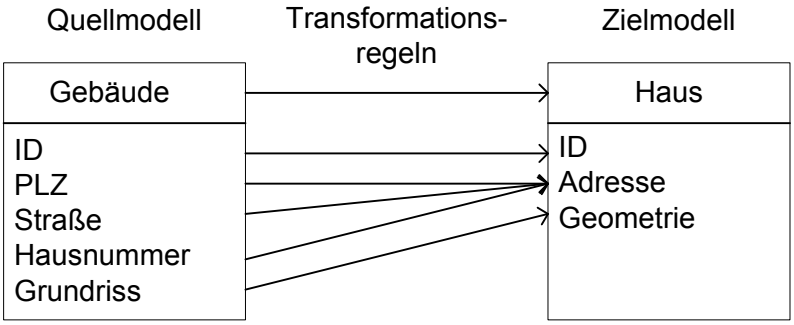


Abbildung 2.11: Transformation von Quell- nach Zielmodell

Modellbasierte Transformation von Geodaten Es bieten sich zwei Möglichkeiten an, Geodaten zu transformieren:

- *Syntaktische Transformation*: Bei einer Transformation auf syntaktischer Ebene wird nur die Syntax der Geodaten umgebaut, d.h. es wird eine Formatumwandlung durchgeführt. Ein Beispiel hierfür wäre die Transformation von Shape-Dateien in GML-Dateien.
- *Semantische Transformation*: Bei dieser Transformation werden die Daten umstrukturiert, so dass sie der Semantik eines anderen Datenmodells entsprechen. Ein Beispiel hierfür ist die Umstrukturierung von Bodenbedeckungsflächen in Gebäude. Auf dieser Transformationsmöglichkeit liegt der Fokus dieser Studie.

Zwischen der oben beschriebenen *Modelltransformation* und der *modellbasierten Transformation von Geodaten* existiert somit der folgende entscheidende Unterschied: Bei der Modelltransformation werden Modelle transformiert. Bei der modellbasierten Transformation von Geodaten dagegen findet die Transformation eine Ebene tiefer statt, d.h. es werden die Geodaten selbst, in Form von Transferformaten, transformiert. Dies ist in Abbildung 2.12 dargestellt. Die Abbildung basiert auf Abbildung 2.10, sie wurde jedoch dahingehend erweitert, dass hier die Transformation der Geodaten dargestellt ist. Ziel sollte es auch hier sein, dass die Transformation automatisch abläuft.

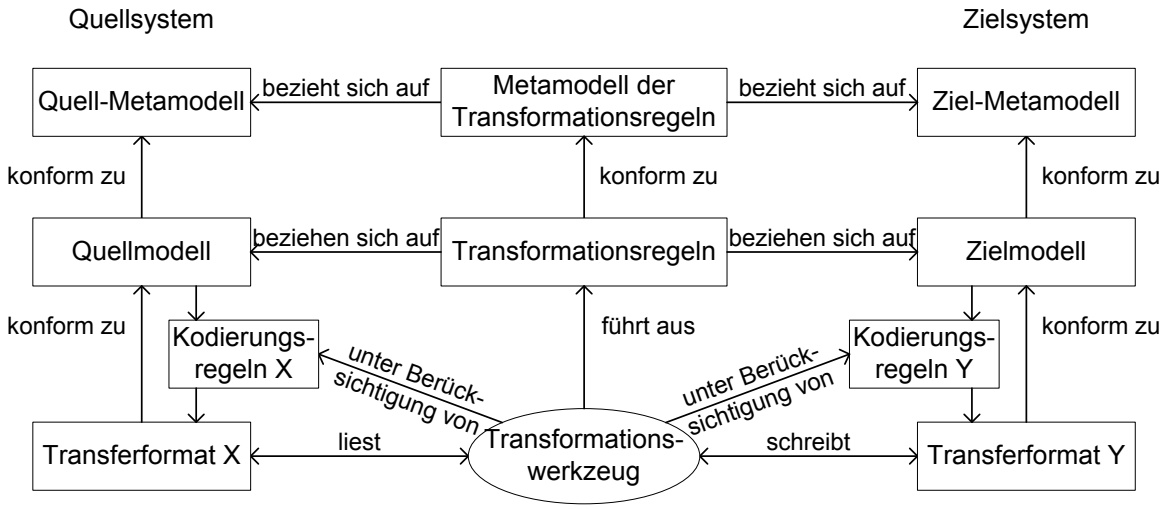


Abbildung 2.12: Modellbasierte Transformation von Geodaten

Wie aus Abbildung 2.12 zu sehen ist, beinhaltet die modellbasierte Transformation von Geodaten einen weiteren wichtigen Aspekt, nämlich Kodierungsregeln (→ Abschnitt 2.2) zur Herleitung von Transferformaten (→ Abschnitt 2.2) für die zu transformierenden Geodaten. Diese Regeln existieren zusätzlich zu den Kodierungsregeln für die Abbildung visueller Modelle in Transferformate, wie oben beim Abschnitt Modelltransformation beschrieben wurde. Jedes Transferformat für Geodaten wird anhand spezifischer Kodierungsregeln aus dem konzeptuellen Schema hergeleitet. Die Geodaten können deshalb nur dann korrekt transformiert werden, wenn die jeweiligen Kodierungsregeln bei der Transformation berücksichtigt werden.

Refactoring Beim *Refactoring* handelt es sich um eine Methode, die ursprünglich eingesetzt wurde, um die Struktur von Programmcode zu verbessern. In [Fowler, 2000] wird Refactoring definiert als ein „Prozess, ein Softwaresystem so zu verändern, dass das externe Verhalten nicht geändert wird, der Code aber eine bessere interne Struktur erhält“. Dies bedeutet, dass die Semantik des Codes beim Refactoring nicht verändert werden darf. Inzwischen wird Refactoring aber auch bei Datenbanksystemen und der konzeptuellen Modellierung angewendet. Refactoring kann der horizontalen Transformation zugeordnet werden.

Schema Mapping Modelltransformationen können in zwei Phasen unterteilt werden, in die Konfigurationsphase und in die Ausführungsphase. Die Konfigurationsphase wird auch als *Schema Mapping* bezeichnet und beschäftigt sich ausschließlich mit der Definition von Transformationen. Die Ausführung von Transformationen ist nicht Bestandteil des Schema Mapping. Beim Schema Mapping werden Quell- und Zielschemata anhand korrespondierender Schemaelemente miteinander in Beziehung gesetzt, d. h., es wird definiert, welche Elemente aus den Quellschemata in Elemente aus den Zielschemata überführt werden können. Anschließend können aus dem Mapping Transformationsregeln für die Transformation von Quellschemata in Zielschemata abgeleitet werden [Lehto, 2007].

Schemaintegration *Schemaintegration* bezeichnet das Vorgehen, bei dem alle relevanten Schemata, *lokale Schemata* genannt, zu einem gemeinsamen, *globalen Schema* kombiniert werden. Schemaintegration beinhaltet auch die Durchführung der Transformation von den lokalen Schemata in das globale Schema [Lehto, 2007]. Die INSPIRE Data Specifications sind ein Beispiel für ein globales Schema, da sie eine Schnittmenge von Informationen aus den nationalen Schemata der EU-Mitgliedstaaten beinhalten.

Eine interessante Fragestellung in diesem Zusammenhang ist, in welche Richtung die Transformation definiert wird. Dementsprechend lässt sich nämlich nach *Originalschema* und *Sicht* unterscheiden. Das Schema, von dem die Transformation ausgeht, stellt i. d. R. das Originalschema dar, während das Schema, in das transformiert wird, eine Sicht auf das Originalschema repräsentiert. Eine Sicht enthält genau die Menge an Informationen des Originalschemas, welche für die jeweiligen Nutzer von Interesse ist [Kemper et al., 2009]. So ist z. B. auch das INSPIRE-Schema eine Sicht.

ETL Die Abkürzung *ETL* steht für *Extract, Transform, Load* und bezeichnet den Prozess der Integration von Daten aus einem oder mehreren Quellsystemen in ein Zielsystem. Der Prozess beinhaltet folgende Schritte:

- *Extract*: Extrahieren der Daten aus den relevanten Quellsystemen
- *Transform*: Transformieren der Daten, so dass sie dem Schema des Zielsystems entsprechen
- *Load*: Speichern der Daten im Zielsystem

ETL-Prozesse entsprechen somit der Schemaintegration. ETL-Prozesse spielen eine große Rolle in Data Warehouses. Sind in einen ETL-Prozess Geodaten involviert, dann spricht man auch von *Spatial ETL*.

Abgrenzung zu anderen Transformationen Neben der semantischen Transformation existieren eine Reihe weiterer Transformationen, die in diesem Kontext zu nennen sind. Nachfolgend werden diese Transformationen kurz aufgelistet und es wird eine Abgrenzung zur semantischen Transformation vorgenommen.

- *Koordinatentransformation*: Ist Bestandteil der semantischen Transformation.

- *Portrayal*: Hierbei handelt es sich um eine semantische Transformation.
- *GIS-Analysen*: Können als Anwendungen der semantischen Transformation betrachtet werden. [Donaubauer et al., 2010] präsentieren einen Ansatz, wie Geoprocessing Workflows als semantische Transformation beschrieben werden können.
- *Datenprüfungen*: Können durch semantische Transformation realisiert werden.
- *Kartographische Generalisierung*: Hierbei handelt es sich um eine Anwendung bzw. einen Spezialfall der semantischen Transformation.

2.3.2 Modelltransformation und Transformationssprachen

Transformationsregeln *Transformationsregeln* dienen dazu, die Abbildung zwischen den Schemata zu beschreiben. Transformationsregeln müssen maschinenlesbar und maschineninterpretierbar sein, damit sie von einem Transformationswerkzeug verarbeitet werden können. Aus diesem Grund werden für die Definition von Transformationsregeln formale Sprachen benötigt, so genannte Transformationssprachen.

Transformationssprache Mittels einer *Transformationssprache* können Transformationsregeln definiert werden. Transformationssprachen sind formale Sprachen (→ Abschnitt 2.1.2) und stellen, wie jede andere Sprache auch, eine entsprechende Syntax und Semantik (→ Abschnitt 2.1.2) bereit.

Es existieren viele Ansätze für Transformationssprachen, jedoch noch kaum Standards. So werden in [Czarnecki et al., 2006] 32 verschiedene Transformationssprachen untersucht und nach bestimmten Merkmalen gruppiert. Dabei werden Sprachen aus vier verschiedenen Quellen herangezogen, Sprachen, die in der Literatur publiziert sind, Sprachen des OMG, Sprachen, die in Open-Source-Werkzeugen implementiert sind und Sprachen, die in kommerziellen Werkzeugen vorkommen.

Direktionalität Erfolgt die Transformation immer nur in eine Richtung, d. h. von den Quellmodellen ins Zielmodell, dann spricht man von *unidirektionaler Transformation*. Ist dagegen die Transformation in beide Richtungen möglich, also vom Zielmodell auch wieder zurück in die Quellmodelle, dann handelt es sich um eine *bidirektionale Transformation*.

2.3.3 Implementierung der modellbasierten Transformation von Geodaten

Es existieren zwei verschiedene Möglichkeiten, Transformationen zu implementieren. Welche der beiden Möglichkeiten eingesetzt wird, ist jedoch immer auch abhängig von der Anwendung, der Komplexität des Datenmodells sowie den Nutzeranforderungen [RTG, 2010]:

- *On-the-fly-Transformation*: Die Transformation wird immer dann durchgeführt, wenn der Nutzer transformierte Daten abrufen möchte. Dies bietet den großen Vorteil, dass

direkt zur Anfragezeit in beliebige Zieldatenmodelle transformiert werden kann. On-the-fly-Transformation eignet sich insbesondere, wenn die Aktualität eine entscheidende Rolle spielt. Die Performanz kann jedoch beeinträchtigt werden, wenn große Datenmengen zu transformieren sind oder die Transformation an sich sehr komplex ist. Handelt es sich dagegen um ähnliche Modelle bzw. weniger komplexe Transformationen oder alternativ um kleinere Datenmengen, so sind keine Performanzeinbußen zu erwarten. Die Anfrage wird hier stets in der Semantik des Zielmodells gestellt. Damit die Transformation entsprechend der Anfrage durchgeführt werden kann, muss diese erst in die Semantik des Quellmodells zurückübersetzt werden.

- *Offline-Transformation*: Die Transformation wird im Voraus durchgeführt und die transformierten Daten werden in einer Datenbank vorgehalten (Sekundärdatenbestand). Der Nutzer stellt seine Anfrage direkt an diese Datenbank und erhält die transformierten Daten, ohne dass zuvor noch eine Transformation durchgeführt werden muss und ohne dass die Anfrage in die Semantik des Quellmodells zurückübersetzt werden muss. Die Transformation ist in diesem Fall ein interner Prozess beim Datenbereitsteller.

2.4 Einordnung der Begriffe in OMG, ISO und INSPIRE

Die folgenden Abschnitte behandeln die Themen Modellierung und Modelltransformation im Kontext der relevanten Standards und Normen der OMG, der ISO und von INSPIRE. Die Einordnung der in den vorherigen Abschnitten erwähnten Begriffe in den nachfolgenden Text wird dabei mittels „(→ Abschnitt xxx)“ hervorgehoben.

2.4.1 OMG – MOF und UML

Die *Object Management Group* (OMG) ist ein 1989 gegründeter internationaler Verband, der aus Mitgliedern der Informationssystem- und Softwareentwicklungsbranche besteht. Die OMG beschäftigt sich mit der Standardisierung und Veröffentlichung von Praktiken und Theorien zur objektorientierten Softwareentwicklung. Die veröffentlichten Spezifikationen können system- und plattformunabhängig eingesetzt werden. Dadurch wird die Wiederverwendbarkeit, Portierbarkeit und Interoperabilität objektorientierter Software im verteilten heterogenen Umfeld unterstützt.

Im Zuge der Begriffsbestimmungen wurden bereits die Model-Driven Architecture des OMG (→ Abschnitt 2.1.4) und die Einordnung von Metamodellen in die 4-Schichten-Architektur des OMG (→ Abschnitt 2.1.5) beschrieben. Dieser Abschnitt geht nun detaillierter auf die darin bereits erwähnten Spezifikationen MOF und UML ein und stellt Konzepte daraus vor, welche insbesondere für die Inhalte dieser Studie von Relevanz sind.

Meta Object Facility (MOF) Die Spezifikation *Meta Object Facility* (MOF) [OMG, 2006] bildet die Grundlage dafür, dass Modelle definiert, transformiert, aus Anwendungen exportiert bzw. in Anwendungen importiert, zwischen verschiedenen Systemen ausgetauscht und in Code überführt werden können. Die MDA (→ Abschnitt 2.1.4) vereint diese Schritte durch

die Modellierung von PIMs über die Generierung von PSMs bis hin zur Generierung von Code und ausführbaren Anwendungen.

Einzige Voraussetzung ist, dass die Modelle auf MOF basieren müssen. Die MOF-Spezifikation stellt hierfür entsprechende Konstrukte bereit, mit denen Modelle definiert werden können. So wurde z. B. das Metamodell der UML mit MOF definiert [OMG, 2007b].

Ein wichtiger Bestandteil der MOF ist das Transferformat (→ Abschnitt 2.2) *XML Metadata Interchange* (XMI) [OMG, 2007c]. Das Format ist XML-basiert und ermöglicht den Austausch jeglicher Metadaten, deren Metamodelle auf MOF basieren. Häufig wird XMI als Austauschformat für UML-Modelle eingesetzt.

Unified Modeling Language (UML) Die Modellierungssprache *Unified Modeling Language* (UML) ist eine Spezifikation der OMG und definiert ein Metamodell, mit dem UML-Modelle erstellt werden können. UML entspricht dem *OO-Paradigma* (→ Abschnitt 2.1.4).

Die Spezifikation der UML ist in die Dokumente *UML Infrastructure* [OMG, 2007a] und *UML Superstructure* [OMG, 2007b] unterteilt. Das Dokument UML Infrastructure definiert Konzepte des Metametamodells, d.h., es stellt nicht nur die Grundlage für die UML-Spezifikation dar, sondern beschreibt gleichzeitig auch das MOF-Modell. Das Dokument UML Superstructure dagegen beschreibt das eigentliche UML-Metamodell. Siehe hierzu auch Abbildung 3.6.

Es existieren unterschiedliche Versionen der UML-Spezifikation. Die *UML-Versionen 1.4.2* und *2.1* sind gegenwärtig im GIS-Bereich am weitesten verbreitet. Die UML-Version 1.4.2 ist zudem als Norm ISO 19501 [ISO, 2005c] verfügbar. Daraus, dass verschiedene UML-Versionen im Gebrauch sind, können sich jedoch auch Probleme ergeben, z. B. bei der Transformation zwischen UML-Modellen, die mittels unterschiedlicher UML-Versionen beschrieben sind.

Datentypen in UML Wichtig im Kontext dieser Studie ist der Begriff *Datentyp*. Attribute besitzen gewöhnlich einen Typ, welcher den Wertebereich des Attributs vorgibt. Bei Attributen kann dieser Typ ein Datentyp oder eine Klasse sein.

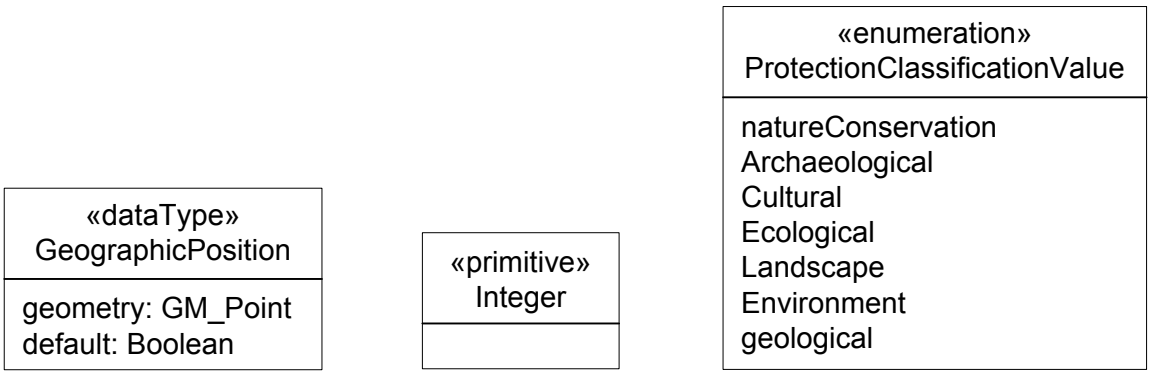
Ein Datentyp unterscheidet sich von einer Klasse darin, dass die Werte des Datentyps keine Identität besitzen. Dies bedeutet, dass die Werte nicht voneinander unterscheidbar sind. So ist beispielsweise der Wert 5 des Datentyps Integer immer derselbe Wert, egal bei welchem Attribut er als Wert auftritt. Die Werte einer Klasse dagegen sind die Objekte der Klasse (Instanzen). Sie besitzen immer eine eigene Identität und sind somit voneinander unterscheidbar [Born et al., 2004]. Datentypen dürfen jedoch ebenfalls wie die Klassen Operationen und Attribute besitzen.

Datentypen werden normalerweise mit dem Stereotyp «dataType» dargestellt, siehe Abbildung 2.13(a). Es existieren jedoch zwei spezielle Datentypen, die eigens gekennzeichnet werden:

- *Primitive Datentypen*: Sie besitzen den Stereotyp «primitive» (Abbildung 2.13(b)). Es handelt sich dabei um Datentypen, die keine interne Struktur aufweisen und aus denen weitere Datentypen zusammengesetzt werden können. UML selbst stellt vier primitive Datentypen bereit: String (Zeichenketten), Integer (ganze Zahlen), UnlimitedNatural

(natürliche Zahlen) und Boolean (Werte true/false). Darüber hinaus können weitere selbstdefinierte primitive Datentypen eingeführt werden. In UML vordefinierte primitive Datentypen existieren erst seit der UML-Version 2.0.

- *Aufzählungstypen*: Bei diesem Datentyp werden die möglichen Werte des Attributs explizit aufgezählt. Aufzählungstypen besitzen in der Regel keine Attribute und Operationen. Der Datentyp erhält den Stereotyp «enumeration» (Abbildung 2.13(c)).



(a) Datentyp (b) Primitiver Datentyp (c) Enumeration

Abbildung 2.13: Datentyp, Primitiver Datentyp und Aufzählungstyp (Quellen: [JRC, 2009d], [OMG, 2007b] und [JRC, 2009a])

2.4.2 ISO 19109 General Feature Model und Anwendungsschema

Die Norm *ISO 19109 Geographic Information – Rules for application schema* [ISO, 2005b] definiert Regeln für die Erstellung von Anwendungsschemata (→ Abschnitt 2.1.5). Damit ist es möglich, Anwendungsschemata auf einheitliche Weise zu erstellen, wodurch die Generierung, Verarbeitung, Visualisierung und der Transfer von Geodaten zwischen verschiedenen Anwendern und Systemen erleichtert werden soll.

General Feature Model Grundlage eines jeden Anwendungsschema bildet das so genannte *General Feature Model* (GFM). Beim GFM handelt es sich um ein Metamodell (→ Abschnitt 2.1.5) speziell für raumbezogene Objekte (Features) und deren Eigenschaften. Das GFM definiert Konzepte, die benötigt werden, um den Universe of Discourse (→ Abschnitt 2.1.1) in ein Anwendungsschema zu überführen. Abbildung 2.14 zeigt einen Ausschnitt aus dem General Feature Model.

Feature Eine zentrale Rolle beim GFM spielt das *Feature*, welches eine Abstraktion eines Phänomens der realen Welt darstellt [ISO, 2002a]. Ein *FeatureType* beschreibt, welche

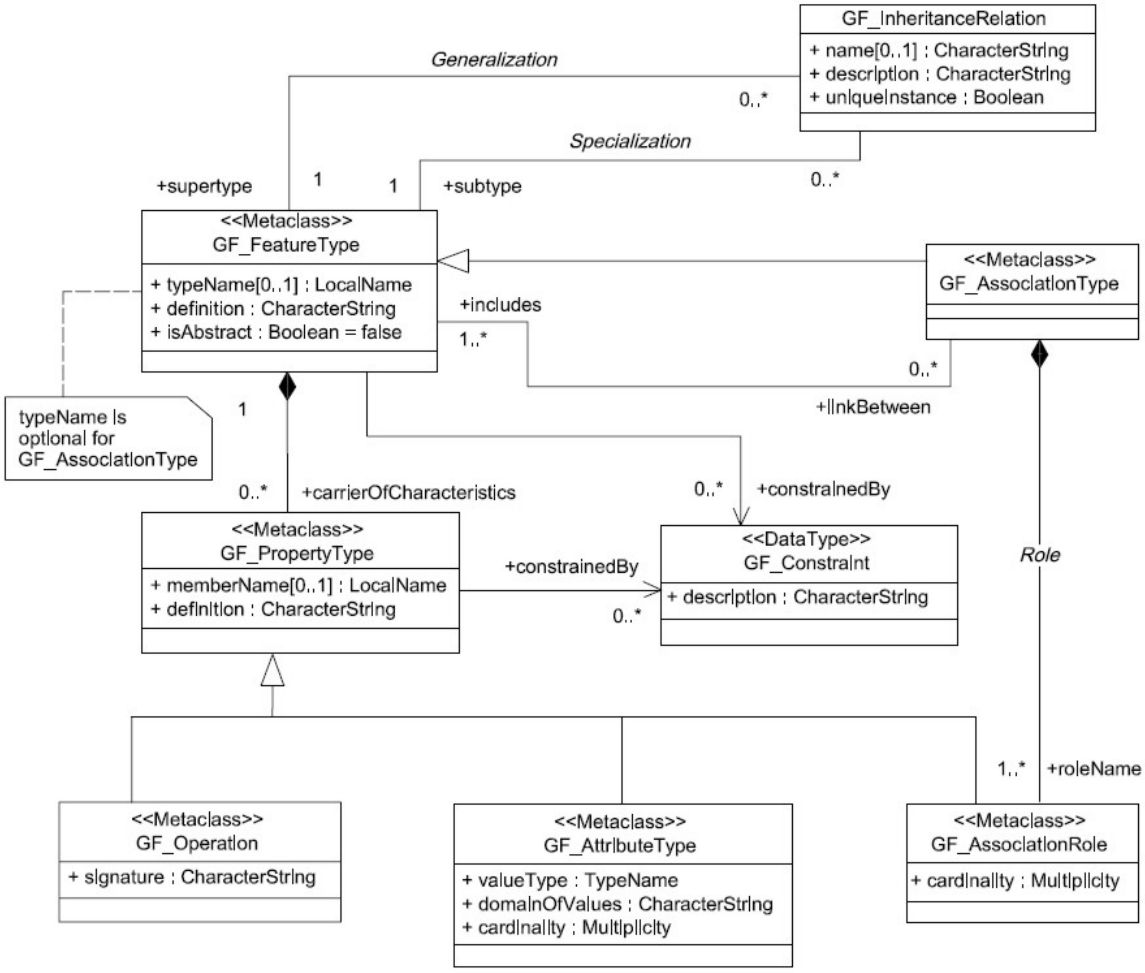


Abbildung 2.14: Ausschnitt aus dem General Feature Model (Quelle: [ISO, 2005b])

Informationen gleichartige Features beinhalten. Das GFM stellt für FeatureTypes die Metaklasse *GF_FeatureType* zur Verfügung. Alle FeatureTypes, die gemäß dem GFM modelliert werden, sind von dieser Metaklasse abzuleiten. Ein FeatureType besteht aus Name, geometrischen und nichtgeometrischen Attributen, Operationen sowie Assoziationen zu anderen FeatureTypes. Die konkreten Features eines FeatureTypes werden als *Feature-Instanzen* bezeichnet. So könnten beispielsweise vom FeatureType „Schloss“ die Instanzen „Schönbrunn“, „Nymphenburg“ oder „Versailles“ gebildet werden. Das GFM entspricht dem *OO-Paradigma* (→ Abschnitt 2.1.4).

Das GFM bietet eine neue Sichtweise auf Datenmodelle in Geoinformationssystemen. Die herkömmliche Sichtweise ist stark auf Geometrien ausgerichtet, d.h. Objekte werden als Punkte, Linien oder Polygone betrachtet, denen Attribute zugeordnet sind, welche oftmals in Datenbanktabellen gespeichert werden. Beim GFM dagegen stehen Features im Mittelpunkt. Die Geometrien des Features werden als räumliche Attribute repräsentiert und ebenso wie die anderen Attribute des Features behandelt. Dieser Ansatz reflektiert die Sprache der Nutzer, welche in der Regel von Straßen, Flüssen, etc. und nicht von Punkten, Linien oder

Polygonen sprechen und auch nicht von Tabellen, Listen oder Tupeln [CSIRO, 2010]. Ein Feature kann dadurch auch durch mehr als eine Geometrie repräsentiert werden. So kann z. B. der FeatureType „Fluss“ ein Attribut „Flussachse“ mit einer Linie als Geometrie besitzen und zudem ein Attribut „Wasserfläche“, dessen Geometrie ein Polygon ist.

Syntax und Semantik Mit dem GFM wird nur die Semantik (→ Abschnitt 2.1.2) des Metamodells definiert. Das GFM stellt keine konkrete Syntax (→ Abschnitt 2.1.2) im Sinne einer Syntax einer CSL (→ Abschnitt 2.1.4) für die Modellierung der Anwendungsschemata bereit. Anwendungsschemata müssen mit der Syntax einer existierenden CSL definiert werden. Hierfür wird von der Norm ISO 19109 die Sprache UML vorgeschlagen, wobei ebenso jede beliebige andere CSL eingesetzt werden könnte. Zudem werden Regeln für die Umsetzung der GFM-Konzepte in UML bereitgestellt.

Wertsemantik und Referenzsemantik bei räumlichen Attributen Räumliche Eigenschaften eines Features werden durch ein oder mehrere räumliche Attribute angegeben. Für die Beschreibung der räumlichen Eigenschaften sollen die geometrischen und topologischen Objekte der Norm ISO 19107 [ISO, 2002b] herangezogen werden. Zudem werden im GFM zwei Möglichkeiten genannt, wie die räumlichen Attribute in einem Anwendungsschema angegeben werden können:

- *Wertsemantik*: Die räumliche Eigenschaft wird in einer UML-Klasse durch ein Attribut repräsentiert, wobei der Datentyp des Attributs ein Objekt der Norm ISO 19107 ist. Beispielsweise hat in Abbildung 2.15 die UML-Klasse AdministrativeUnit das Attribut geometry, dessen Datentyp das ISO-19107-Objekt GM_MultiSurface ist. Diese Art der Darstellung ist identisch für die UML-Versionen 1.4.2 und 2.1.

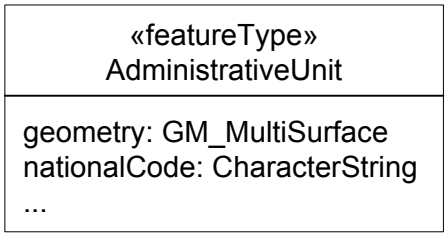
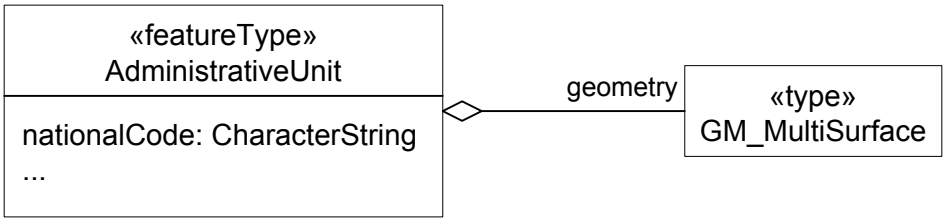


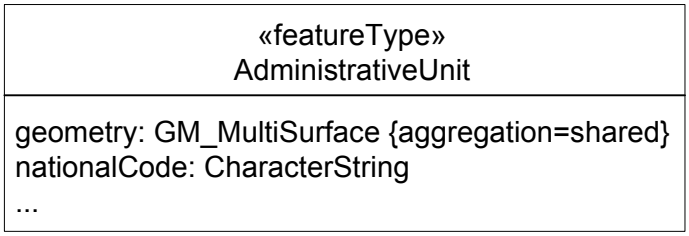
Abbildung 2.15: Wertsemantik bei räumlichen Attributen (Quelle: [JRC, 2010a])

- *Referenzsemantik*: Die räumliche Eigenschaft wird als Assoziation zwischen einer UML-Klasse, die ein Feature repräsentiert, und einer Geometrie-Klasse der Norm ISO 19107 angegeben. Abbildung 2.16(a) zeigt dies wiederum am Beispiel der UML-Klasse AdministrativeUnit. Die Assoziation geometry verbindet diese Klasse mit der Geometrie-Klasse GM_MultiSurface. Um explizit auszudrücken, dass es sich bei den Geometrien um Attribute handelt, wurde in der Abbildung keine normale Assoziation verwendet, sondern eine Aggregation. Aggregationen sind spezielle Assoziationen

und beschreiben eine Ganzes-Teile-Beziehung. In der Abbildung beschreibt die Aggregation somit, dass Objekte der Klasse GM_MultiSurface Bestandteil von Features der Klasse AdministrativeUnit sind. In UML-Version 2.1 existiert neben Abbildung 2.16(a) noch eine zweite Möglichkeit, Aggregationen darzustellen, was in Abbildung 2.16(b) gezeigt wird.
Bei der Referenzsemantik können mehrere FeatureType-Klassen mit einer Geometrie-Klasse in Beziehung stehen. Dabei ist jedoch zu beachten, dass bei einer Änderung eines Geometriewerts sich der Wert für all diejenigen Objekte ändert, die auf diesen Geometriewert verweisen.



(a) UML-Versionen 1.4.2 und 2.1



(b) UML-Version 2.1

Abbildung 2.16: Referenzsemantik bei räumlichen Attributen

Zusammenhang zwischen GFM, Anwendungsschema und UML Abbildung 2.17 veranschaulicht die Beziehung zwischen dem GFM und einem Anwendungsschema. Darüber hinaus wird der Zusammenhang zur Sprache UML dargestellt. Gemäß der Abbildung besteht der Unterschied zwischen einem normalen UML-Modell und einem Anwendungsschema darin, dass bei einem UML-Modell sowohl Syntax als auch Semantik im UML-Metamodell definiert sind, wohingegen beim Anwendungsschema die Semantik im GFM definiert ist, die Syntax dagegen dem UML-Metamodell entnommen wird. Auf Datenebene wird zudem der Bezug zu GML gezeigt. Eingebettet ist die Darstellung in die 4-Schichten-Architektur gemäß OMG (→ Abschnitt 2.1.5).

2.4.3 ISO 19103 UML-Profil

Die Norm *ISO 19103 Geographic information – Conceptual schema language* [ISO, 2005a] definiert Regeln für die Verwendung der Sprache UML innerhalb der ISO 19100 Normenserie

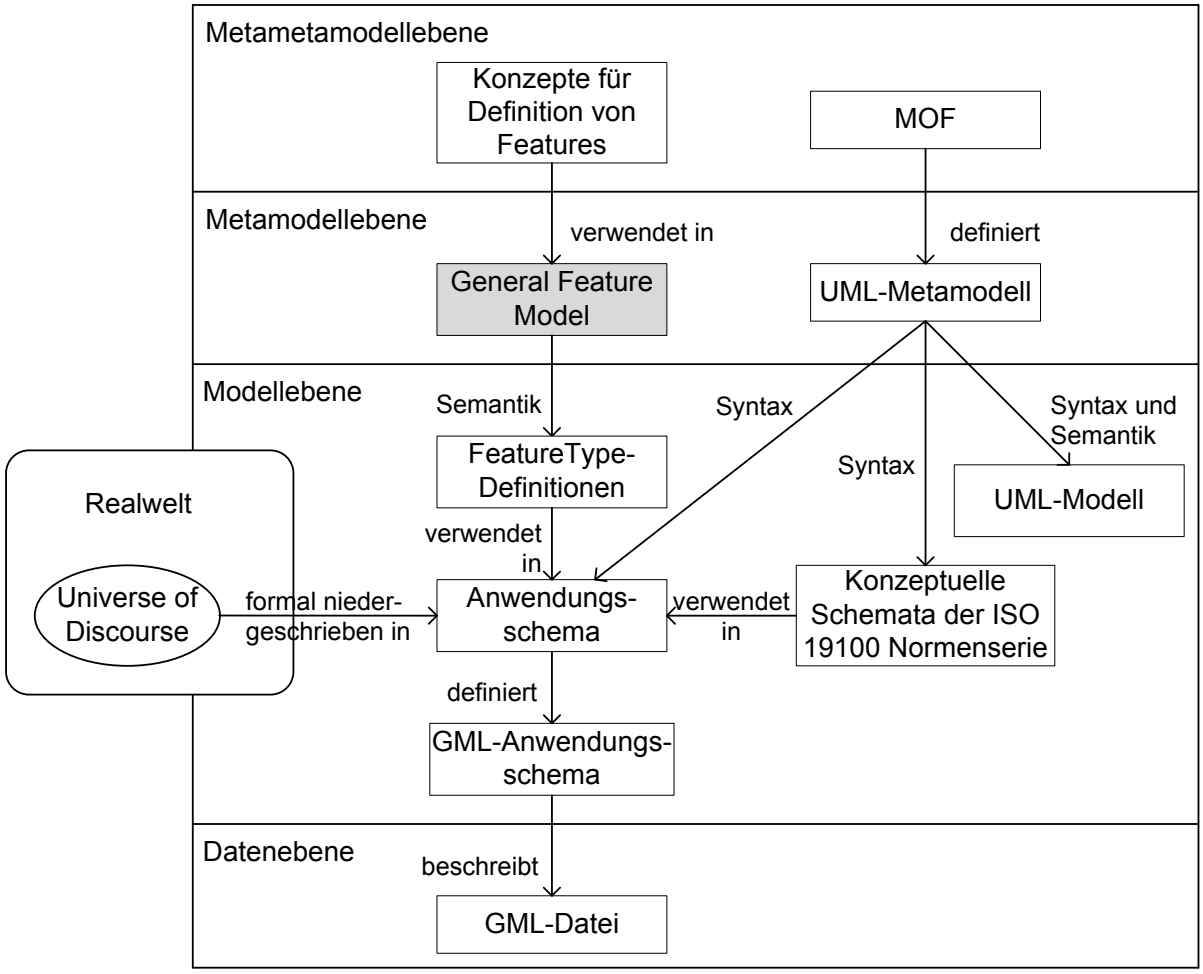


Abbildung 2.17: Zusammenhang zwischen GFM, UML, Anwendungsschema und GML (Quelle: [ISO, 2005b], verändert)

im Sinne eines UML-Profils (→ Abschnitt 2.1.4). UML soll dabei in der Version 1.4.2 gemäß der Norm ISO 19501 eingesetzt werden.

Datentypen Für das UML-Profil werden eine Reihe von Basisdatentypen definiert wie Datum und Zeit, Zahlen (z. B. Integer, Real), Text (z. B. CharacterString), Wahrheitswerte (z. B. Boolean) und Maßeinheiten (z. B. Length, Scale). Die UML-Version 1.4.2 schreibt die Verwendung bestimmter Datentypen nicht vor und stellt auch keine vordefinierten Datentypen bereit. Die UML-Version 2.1 dagegen definiert eine Reihe von primitiven Datentypen (→ Abschnitt 2.4.1). Die Verwendung dieser primitiven Datentypen kann zu Kompatibilitätsproblemen mit UML-Modellen führen, die gemäß der Norm ISO 19103 in der UML-Version 1.4.2 modelliert wurden (→ Abschnitt 4.1.2).

Stereotypen Des Weiteren legt die Norm fest, welche Stereotypen und Tagged Values (→ Abschnitt 2.1.4) zu verwenden sind. Eine Reihe von Stereotypen, die im Rahmen dieses

UML-Profils eingesetzt werden, sind bereits in UML vordefiniert. Darüber hinaus definiert die Norm ISO 19103 zusätzliche Stereotypen, die in Tabelle 2.1 dargestellt sind.

Stereotyp	Erläuterung
«CodeList»	Ähnlich einer «enumeration» in UML (→ Abschnitt 2.4.1), wobei bei einer «enumeration» die Wertemenge fix im Modell definiert ist. Bei der Verwendung von «CodeList» ist dies nicht der Fall, die Wertemenge ist hier durch den Nutzer beliebig erweiterbar.
«Leaf»	Ein Paket, das Definitionen, aber selbst keine Pakete enthält.
«Union»	Mit diesem Stereotyp wird ein Typ definiert, der zur Laufzeit nur genau eine der angegebenen Alternativen als Wert besitzt.

Tabelle 2.1: In ISO 19103 definierte Stereotypen

Zudem werden in der Norm zusätzliche Vorgaben gemacht bezüglich der Angabe von Kardinalität und Rollenname, der Kennzeichnung der Optionalität von Attributen und Rollennamen sowie zu Namenskonventionen und der Verwendung von UML-Paketen.

Unterschied zur UML-Profil-Definition der OMG Das UML-Profil, welches in der Norm ISO 19103 beschrieben ist und in diesem Abschnitt vorgestellt wurde, ist kein UML-Profil im Sinne der UML-Profil-Definition der OMG (→ Abschnitt 2.1.4). Als Begründung hierfür können folgende Punkte angeführt werden:

- Der Stereotyp «CodeList» stellt keine Spezialisierung gemäß der UML-Spezifikation dar, was nachfolgend ausführlicher erläutert wird.
- Der Stereotyp «Union» stellt ebenfalls keine Spezialisierung gemäß der UML-Spezifikation dar, siehe nachfolgende Erläuterung.
- Die Definition von Datentypen, die in der Norm ISO 19103 vorgenommen wird, ist gemäß der UML-Spezifikation nicht Bestandteil eines UML-Profils.

Das in der Norm ISO 19103 definierte UML-Profil kann somit nicht als echtes UML-Profil bezeichnet werden. Von hier an wird deshalb in dieser Studie der Begriff UML-Profil in Anführungsstriche („UML-Profil“) gesetzt, wenn das ISO-19103-UML-Profil gemeint ist.

Gemäß der Definition der OMG dürfen Stereotype nur Spezialisierungen von Modellelementen des UML-Metamodells sein (→ Abschnitt 2.1.4). Das ISO-19103-„UML-Profil“ enthält jedoch Sprachkonstrukte, die keine reinen Spezialisierungen des UML-Metamodells sind. Die Stereotypen «CodeList» und «Union» sind solche Sprachkonstrukte, welche zwar auf UML-Klassen angewendet werden, jedoch keine Spezialisierungen im Sinne der UML-Spezifikation darstellen.

Abbildung 2.18 zeigt Beispiele für die Verwendung der Stereotype «CodeList» und «Union», die dem deutschen AAA-Modell (→ Abschnitt 3.2.1) entnommen sind. Der Datentyp AX_Rechtszustand_Schutzzone enthält drei Werte. Der Stereotyp «CodeList» legt fest, dass

der Datentyp bei Bedarf zur Laufzeit um weitere Werte ergänzt werden kann. Der Datentyp AG_Geometrie stellt drei Alternativen zur Verfügung. Gemäß der «Union»-Definition darf der Datentyp zur Laufzeit nur genau eine der angegebenen Alternativen als Wert besitzen. Bei «CodeList» und «Union» handelt es sich somit nur syntaktisch um UML-Klassen, da sie auf das UML-Klassensymbol angewendet werden. Semantisch sind sie jedoch als eigenständige UML-Modellelemente zu betrachten.

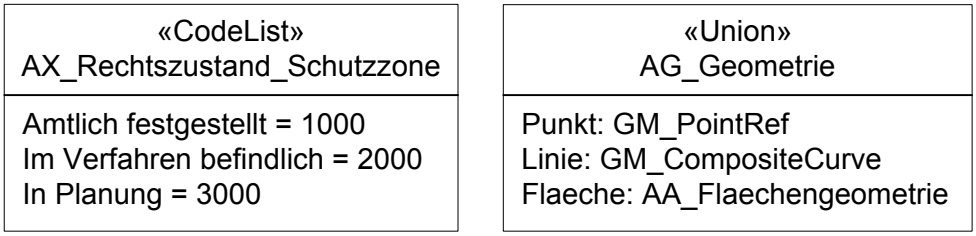


Abbildung 2.18: Stereotype «CodeList» und «Union» (Quelle: [AdV, 2010])

Die Aussage, dass es sich um kein echtes UML-Profil handelt, gilt damit auch für jedes auf der Norm ISO 19103 basierende „UML-Profil“, da diese Profile ebenfalls die Stereotypen «CodeList» und «Union» beinhalten. Zwei solche Profile, das AAA-„UML-Profil“ und das INSPIRE-„UML-Profil“ werden in Abschnitt 3.2.1 bzw. 3.2.3 vorgestellt.

Revision und damit einhergehende Problematik Die Norm ISO 19103 befindet sich zur Zeit in Revision. Es ist jedoch noch offen, wann die revidierte Norm verabschiedet sein wird; dies kann durchaus noch Jahre in Anspruch nehmen. Fest steht allerdings, dass die Revision auf der UML-Version 2.2 basieren wird. Diesbezüglich stellt sich die Frage, wie Rückwärtskompatibilität gewährleistet werden kann, sollte die revidierte Norm einmal verabschiedet sein und sich damit die Basis für die Definition von Anwendungsschemata ändern. Durch die Revision werden zukünftig Anwendungsschemata, die auf der Norm ISO 19103 basieren, in zwei unterschiedlichen Modellierungssprachen vorliegen – in UML 1.4.2 und UML 2.2. Es kann nämlich nicht davon ausgegangen werden, dass bereits definierte Anwendungsschemata sofort bzw. überhaupt in die UML-Version 2.2 überführt werden.

2.4.4 INSPIRE Generic Conceptual Model

INSPIRE Data Specifications Eines der Hauptziele von INSPIRE ist es, Interoperabilität und Harmonisierung von Geodaten und Diensten innerhalb Europas zu gewährleisten [EP, 2007]. Eine große Herausforderung stellen dabei die unterschiedlichen Anwendungsbereiche, *INSPIRE-Themen* genannt, dar, für die Anwendungsschemata (→ Abschnitt 2.1.5), die so genannten *INSPIRE Data Specifications*, zu definieren sind (von Bezugssystemen über Hydrographie und Kataster hin zu atmosphärischen Konditionen). Dies erfordert die Berücksichtigung von spezifischen Anforderungen, woraus sich 20 verschiedene *Interoperabilitätskomponenten* (wie Regeln für Anwendungsschemata, Koordinatenreferenzierung, Mehrsprachigkeit, Datentransformation) ergeben, von denen jede einzelne zur Interoperabilität beiträgt.

INSPIRE Generic Conceptual Model Das INSPIRE-Dokument *INSPIRE Generic Conceptual Model* [JRC, 2009a] beschreibt ein Framework, mit dem zu den einzelnen INSPIRE-Themen entsprechende INSPIRE Data Specifications entwickelt werden können. Dies erfolgt unter Einbeziehung der verschiedenen Interoperabilitätskomponenten, der Schwerpunkt liegt dabei u.a. auf den Komponenten Anwendungsschemata, räumliche und zeitliche Darstellungen von Geoobjekten in verschiedenen Detaillierungsgraden, Objektidentifikatoren und Einschränkungen.

Das Framework basiert auf der Normenserie ISO 19100. Das INSPIRE-„UML-Profil“, welches ebenfalls in dem Dokument beschrieben ist, wird in Abschnitt 3.2.3 ausführlich erläutert. Alle INSPIRE Data Specifications sind gemäß des INSPIRE Generic Conceptual Model zu modellieren. Das eigentliche Vorgehen bei der Modellierung ist jedoch nicht in diesem Dokument beschrieben, sondern befindet sich in dem INSPIRE-Dokument *Methodology for the development of data specifications* [JRC, 2008]. Ebenso die Kodierung (→ Abschnitt 2.2) des Transferformats, diese ist im INSPIRE-Dokument *Guidelines for the encoding of spatial data* [JRC, 2009b] definiert.

3 IST-Situation

Dieses Kapitel befasst sich mit der IST-Situation der modellbasierten Transformation von Geodaten in Deutschland, in der Schweiz und in der EU. Es wird beschrieben, wie die Datenmodelle definiert sind, wie die UML-Profile dazu aussehen und welche Unterschiede existieren. Auch die Kodierung der Datenmodelle in Transferformate wird betrachtet.

3.1 Anwendungsfälle

Die Ausführungen in diesem und in den nachfolgenden Kapiteln basieren dabei auf zwei Anwendungsfällen, die sich mit der modellbasierten Transformation von Geodaten beschäftigen und die deshalb nachfolgend kurz vorgestellt werden.

3.1.1 Transformation nach INSPIRE

Seit 2006 wird an der Technischen Universität München im Auftrag des Bundesamts für Kartographie und Geodäsie das Forschungs- und Entwicklungsprojekt *Modellbasierter Ansatz für den Web-Zugriff auf verteilte Geodaten am Beispiel grenzübergreifender GIS-Anwendungen (mdWFS)* bearbeitet. In den ersten beiden Jahren waren zudem die Eidgenössische Technische Hochschule Zürich und deren Auftraggeber swisstopo als weitere Projektpartner beteiligt.

Ziel des Projekts ist es, einen Lösungsansatz für die modellbasierte Transformation von Geodaten, welcher in eine webbasierte Umgebung eingebunden ist, zu erarbeiten. Dazu wurde eine konzeptuelle Sprache zur Formulierung von Transformationsregeln für die semantische Transformation spezifiziert sowie die OGC Web-Feature-Service-Spezifikation um eine Schnittstelle für einen modellbasierten Web Feature Service, der semantische Transformationen ausführen kann, erweitert.

Im Rahmen des Forschungssprojekts mdWFS wird ein Anwendungsfall bearbeitet, welcher sich mit der Transformation zweier verschiedener Quellmodelle in ein Zielmodell befasst. Als Quelldatenmodelle dienen das deutsche Digitale Landschaftsmodell (ATKIS Basis-DLM) als Teil des AFIS-ALKIS-ATKIS-Referenzmodells (AAA-Modell) sowie das Schweizer Topographische Landschaftsmodell (TLM). Diese Datenmodelle werden im Projekt in die von der EU definierten europaweit einheitlichen INSPIRE-Datenmodelle (INSPIRE Data Specifications) transformiert, was in Abbildung 3.1 dargestellt ist.

Sowohl die Quellmodelle als auch die Zielmodelle liegen in UML vor. Jedes Modell wird jedoch mittels eines anderen UML-Profils definiert. Die einzelnen UML-Profile werden in den nachfolgenden Abschnitten ausführlicher behandelt:

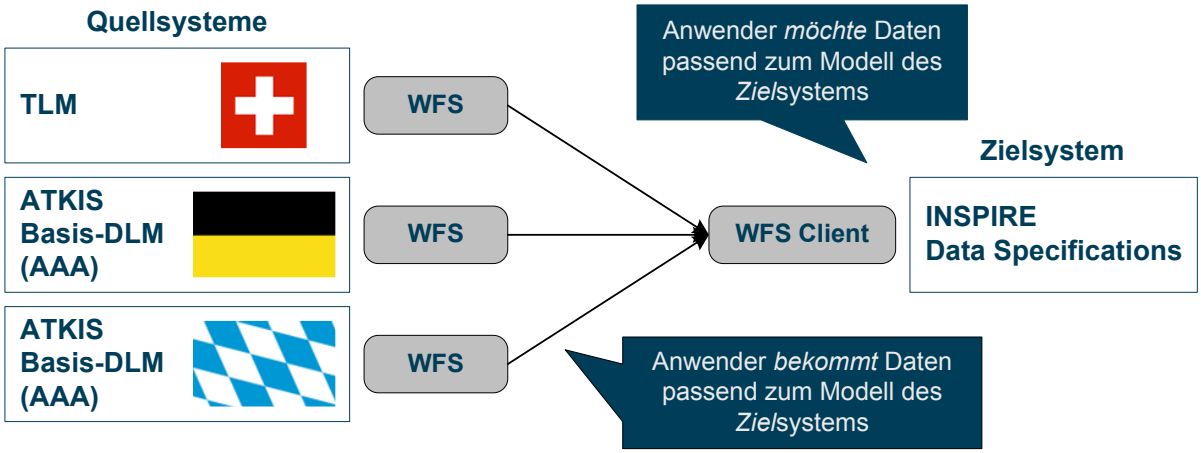


Abbildung 3.1: Transformation nach INSPIRE

- AAA-Modell: AAA-UML-Profil (→ Abschnitt 3.2.1)
- TLM: Interlis-UML-Profil (→ Abschnitt 3.2.2)
- INSPIRE Data Specifications: INSPIRE-UML-Profil (→ Abschnitt 3.2.3)

3.1.2 Transformation in Fachanwendung

Neben der Transformation nach INSPIRE sind auch eine Reihe anderer Zielmodelle denkbar wie z. B. nationale Datenmodelle oder auch Modelle von Fachanwendungen. Aus diesem Grund soll in diese Studie ein entsprechender Anwendungsfall einbezogen werden, welcher sich mit der Transformation aus dem ATKIS Basis-DLM in das Modell für Geobasisdaten einer Fachanwendung beschäftigt. Der Anwendungsfall wurde vom Landesamt für Geoinformation und Landentwicklung Baden-Württemberg (LGL BW) bereit gestellt. Als Fachanwendung dient das GIS-System GISELa (Geographisches InformationsSystem Entwicklung Landwirtschaft), welches zentral alle relevanten flächenbezogenen Informationen über den ländlichen Raum der Landwirtschaftsverwaltung landesweit zur Verfügung stellt. Als zentrale Datenhaltungskomponente für GISELa dient der MLR-Geodatenserver (MLR-GDS), der vom Geodatenzentrum des LGL BW fachlich betrieben wird.

Das Datenmodell des MLR-Geodatenservers repräsentiert das Zielmodell der modellbasierten Transformation. Einen Überblick über den Workflow, welcher momentan ausgeführt wird, um die Ausgangsdaten in den MLR-Geodatenserver zu überführen, liefert Abbildung 3.2.

Gegenwärtig wird als Quellmodell für den MLR-Geodatenserver das ATKIS Basis-DLM in der bisherigen Modellierung verwendet. Für die Untersuchung im Rahmen dieser Studie wird dem Anwendungsfall jedoch das künftige ATKIS Basis-DLM im AAA-Modell zugrunde gelegt. Das entsprechende UML-Profil hierzu wird in Abschnitt 3.2.1 ausführlich beschrieben.

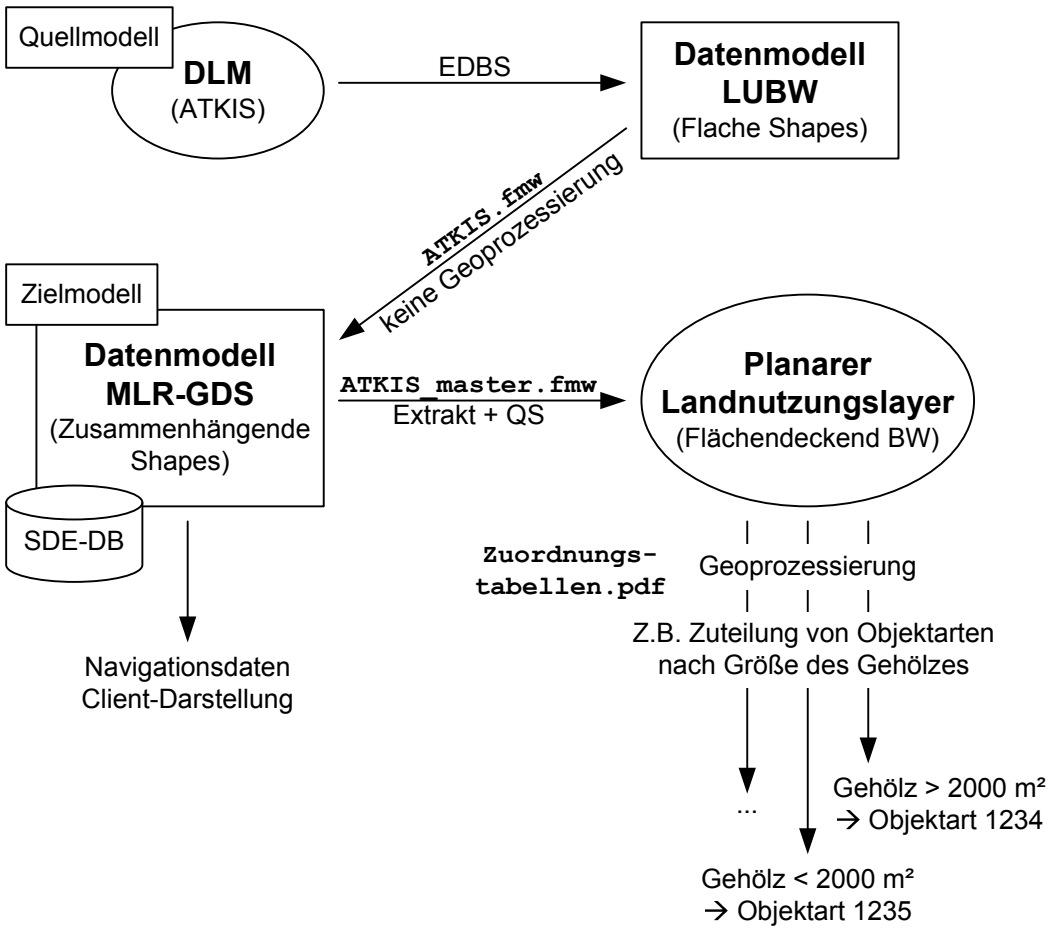


Abbildung 3.2: Transformation in Fachanwendung

Für das Zielmodell ist keine UML-Modellierung und kein UML-Profil vorhanden. Das Zielmodell kann jedoch aus einem FME-Workspace, welcher momentan die Transformation auf Datenebene durchführt (ATKIS.fmw, siehe Abbildung 3.2), abgeleitet werden.

3.2 UML-Profile

Wie in Abschnitt 2.1.4 erläutert wurde, kann die Sprache UML durch die Definition von UML-Profilen an bestimmte Einsatzbereiche angepasst werden. Jedoch kommt nicht in jedem Land dasselbe UML-Profil zum Einsatz und selbst innerhalb einzelner Länder wie z. B. in Deutschland auf Bundeslandebene können Unterschiede auftreten bzw. sogar auf Ebene der Fachverwaltungen, wo teilweise Profile verwendet werden, die speziell auf bestimmte Fachanwendungen angepasst sind. Nachfolgend wird aufgezeigt, wie die UML-Profile Deutschlands, der Schweiz und der EU definiert sind und welche Unterschiede zwischen ihnen existieren.

3.2.1 AAA-UML-Profil

Das deutsche *AFIS-ALKIS-ATKIS-Anwendungsschema* (AAA-Anwendungsschema) setzt sich zusammen aus dem AAA-Basisschema, dem AAA-Versionierungsschema, dem AAA-Fachschema, den NAS(Normbasierte Austausschnittstelle)-Operationen sowie dem AAA-Ausgabekatalog. Das AAA-Basisschema bildet dabei die Grundlage für die Modellierung des AAA-Fachschemas. Das AAA-Fachschema repräsentiert die konzeptuellen Schemata, die den Universe of Discourse (→ Abschnitt 2.1.2) formal wiedergeben.

Metamodell Das AAA-UML-Profil ist im Dokument *Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesen (GeoInfoDok)* [AdV, 2008] beschrieben. Die Modellierung des AAA-Basisschemas basiert auf der Normenserie ISO 19100. Das Metamodell des AAA-Basisschemas ist das General Feature Model der Norm ISO 19109 (→ Abschnitt 2.4.2), welches jedoch um die Metaklasse AA_ObjektOhneRaumbezug erweitert wurde. Die neue Metaklasse wurde dabei von der Metaklasse GF_FeatureType (→ Abschnitt 2.4.2) abgeleitet, was in Abbildung 3.3 dargestellt ist. Mit ihr können FeatureTypes erstellt werden, für die kein Raumbezug zulässig ist.

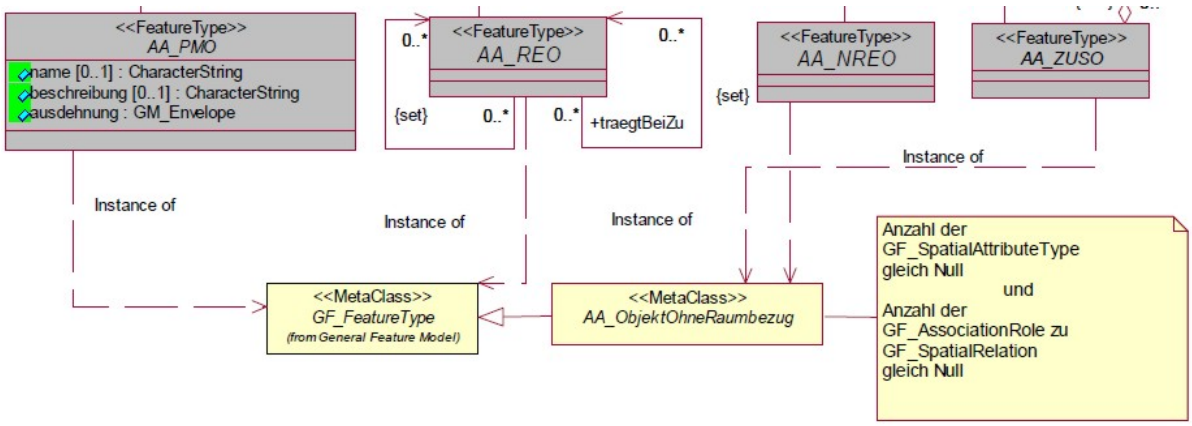


Abbildung 3.3: Erweiterung des General Feature Models im AAA-Basisschema (Quelle: [AdV, 2008])

Datentypen Die Beschreibung des AAA-Anwendungsschemas erfolgt mit UML gemäß der Norm ISO 19103 (→ Abschnitt 2.4.3), d.h., UML wird in der Version 1.4.2 verwendet. Zudem werden für die Erstellung des Anwendungsschemas die entsprechenden Regeln aus der Norm ISO 19109 herangezogen.

Geometrien Im AAA-Anwendungsschema werden die Raumbezugsgrundformen der Norm ISO 19107 verwendet. Im AAA-Basisschema wurden dazu eine Reihe von abstrakten Klassen definiert, die sich auf die ISO-Geometrien beziehen. Bei der Erstellung eines Anwendungsschemas sind von diesen abstrakten Klassen durch Vererbung konkrete Klassen abzuleiten, welche dann im Anwendungsschema eingesetzt werden können.

Folgendes Beispiel in Abbildung 3.4 soll dies veranschaulichen [Schilcher et. al., 2008]: Die Klasse AX_Fliessgewaesser erbt von der abstrakten Klasse AX_TatsaechlicheNutzung. Diese Klasse wiederum erbt von der Klasse TA_SurfaceComponent, welche eine der abstrakten Klassen des AAA-Basisschemas ist. Die Klasse TA_SurfaceComponent ist von der Klasse TS_SurfaceComponent ableitet. TS_SurfaceComponent schließlich ist eine Realisierung der ISO-19107-Geometrie GM_CompositeSurface.

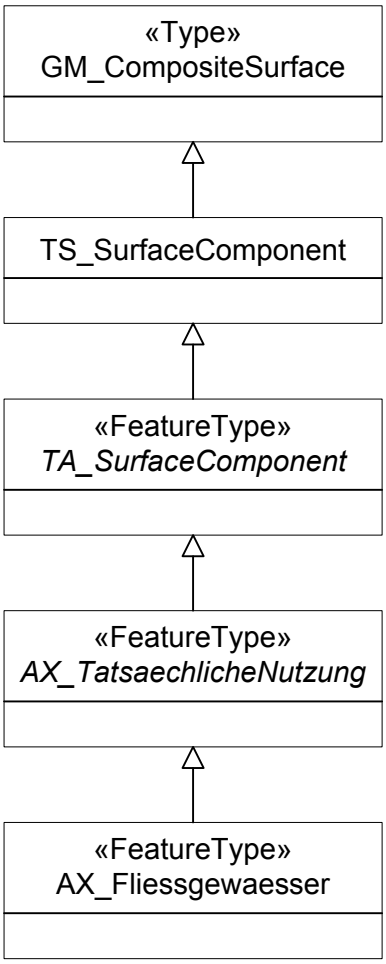


Abbildung 3.4: Erzeugung von Featuretypen mit Geometrie im AAA-Modell

Um die Komplexität zu verringern, sind nur die in Tabelle 3.1 aufgelisteten Raumbezugsgrundformen zugelassen. Für 3D-Geometrien können zudem die in Tabelle 3.2 dargestellten Raumbezugsgrundformen eingesetzt werden.

Die GeoInfoDok legt zudem fest, dass jedes raumbezogene AAA-Fachobjekt nur maximal eine Geometrie aufweisen darf. Andernfalls ist für jede Geometrie ein eigenes Fachobjekt zu erzeugen. Dies stellt eine Einschränkung des General Feature Model dar, welches definiert, dass ein FeatureType mehr als eine Geometrie besitzen darf (→ Abschnitt 2.4.2).

Darüber hinaus wurde in Abschnitt 2.4.2 beschrieben, dass Geometrien gemäß dem General Feature Model mittels Wertsemantik oder Referenzsemantik als Attribute in einem

Raumbezugsgrundformen 2D	
Geometrische Objekte	
Primitive	GM_Point, GM_Curve, GM_PolyhedralSurface, GM_PointRef
Komplexe	GM_CompositeCurve, GM_CompositeSurface
Aggregate	GM_MultiPoint, GM_MultiCurve, GM_MultiSurface
Topologische Objekte	
Primitive	TS_PointComponent, TS_CurveComponent, TS_SurfaceComponent, TS_Face
Komplexe	TP_Complex

Tabelle 3.1: 2D-Raumbezugsgrundformen für das AAA-Anwendungsschema (Quelle: [AdV, 2008])

Raumbezugsgrundformen 3D	
Geometrische Objekte	
Primitive	GM_Solid, GM_SurfaceBoundary, GM_TriangulatedSurface, GM_OrientableSurface
Komplexe	GM_CompositeSolid
Topologische Objekte	
Primitive	TS_Solid, TS_Feature, TS_Theme

Tabelle 3.2: 3D-Raumbezugsgrundformen für das AAA-Anwendungsschema (Quelle: [AdV, 2008])

Anwendungsschema angegeben werden können. Aus Abbildung 3.4 ist jedoch erkennbar, dass, obwohl dem AAA-Anwendungsschema das General Feature Model zugrunde liegt, weder Wertsemantik noch Referenzsemantik eingesetzt werden. Stattdessen bilden die Geometrien das Grundgerüst, woraus alle Feature-Klassen abgeleitet werden.

Stereotypen In den AAA-Anwendungsschemata werden die in der Norm ISO 19103 definierten Stereotypen verwendet (→ Abschnitt 2.4.3). Darüber hinaus wird zur größeren Klarheit für alle FeatureTypes der Stereotyp «FeatureType» der Norm ISO 19136 Annex E [ISO, 2007a] eingesetzt. Da gemäß Abbildung 3.3 alle Objektarten von der Metaklasse GM_FeatureType (→ Abschnitt 2.4.2) abzuleiten sind, können keine Objekte ohne diesen Stereotyp existieren. Zusätzlich werden in den AAA-Anwendungsschemata in der Norm ISO 19136 spezifizierte UML Tagged Values verwendet.

Kodierung Die Kodierung (→ Abschnitt 2.1.5) wird gemäß den Kodierungsregeln (→ Abschnitt 2.1.5) der Normen ISO 19136 Annex E [ISO, 2007a] und ISO 19139 für Metadaten [ISO, 2007b] erstellt. Als Datenaustauschformat wird GML verwendet. Die Kodierung von

UML nach GML wird in zwei Stufen durchgeführt mit einem Implementierungsschema (→ Abschnitt 2.1.5) als Zwischenstufe, was auch in Abbildung 3.5 dargestellt ist [AdV, 2008]:

1. Das AAA-Anwendungsschema enthält einige UML-Konstruktionen, die von den Kodierungsregeln der Normen ISO 19136 Annex E und ISO 19139 nicht unterstützt werden. Das Anwendungsschema wird deshalb in ein Implementierungsschema überführt, das konform zu den Vorgaben der beiden Normen ist. Die Ableitung des Implementierungsschemas erfolgt skriptgestützt mittels der Software Rational Rose.

2. Anschließend wird das Implementierungsschema gemäß den Kodierungsregeln der Normen ISO 19136 Annex E und ISO 19139 in ein GML-Anwendungsschema überführt. Hierfür wird die Software ShapeChange eingesetzt.

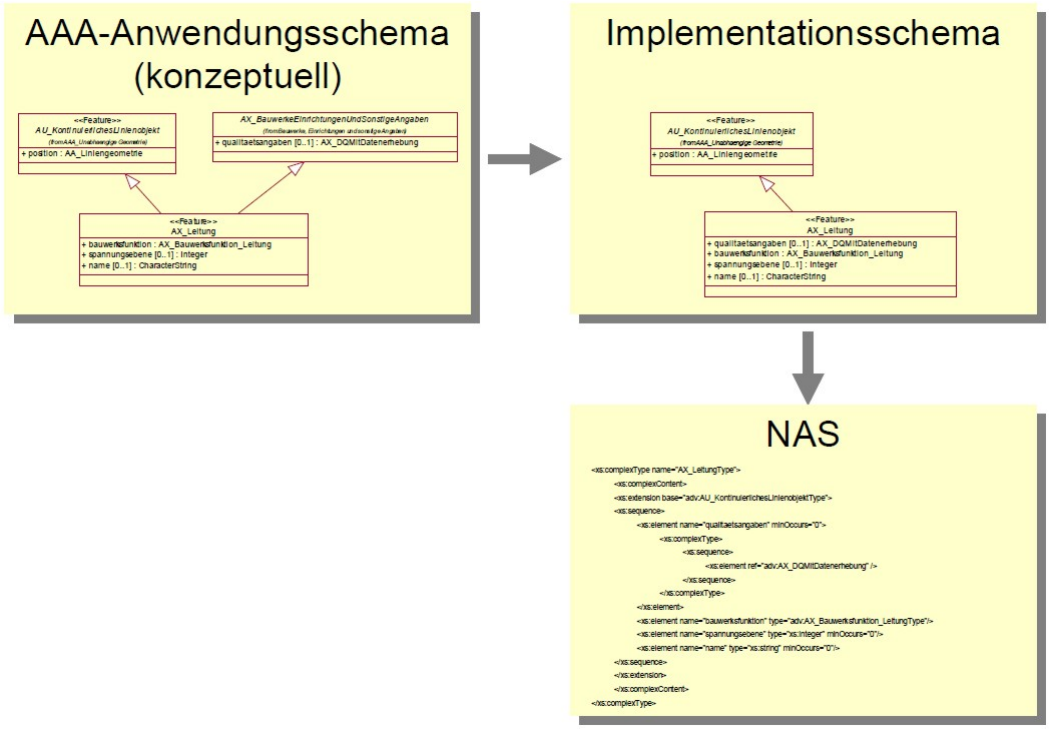


Abbildung 3.5: Implementierungsschema (Quelle: [AdV, 2008])

Fazit Das AAA-UML-Profil ist kein UML-Profil im Sinne der UML-Profil-Definition der OMG (→ Abschnitt 2.1.4). Das AAA-UML-Profil basiert auf dem „UML-Profil“ der Norm ISO 19103 (→ Abschnitt 2.4.3) und kann somit selbst ebenfalls kein echtes UML-Profil darstellen.

Die Regeln zur Herleitung des Implementierungsschemas sind in der GeoInfoDok beschrieben. Die Kodierungsregeln sind den Normen ISO 19136 Annex E und ISO 19139 entnommen. Alle Regeln sind in textueller, informeller Form definiert.

3.2.2 INTERLIS-UML-Profil

Bei dem in Abschnitt 3.1.1 beschriebenen Anwendungsfall wird als eines der Quellmodelle das TLM (Topographisches Landschaftsmodell) eingesetzt. Das TLM ist das Basislandschaftsmodell der Schweiz und wurde mit der Modellierungssprache INTERLIS modelliert. Es existiert kein eigenes UML-Profil speziell für das TLM, sondern die Modellierung basiert allein auf INTERLIS.

Beim TLM handelt es sich um ein Produktionsmodell (→ Abschnitt 2.1.5). Aus diesem Produktionsmodell werden je nach Kontext verschiedene Produkt- oder Nutzermodelle (→ Abschnitt 2.1.5) abgeleitet und befüllt.

Metamodell Es existiert kein normiertes (de-jure) UML-Profil für INTERLIS. Im Rahmen der Software UML/INTERLIS-Editor [KOGIS, 2010] wurde eine UML-Metamodell-Erweiterung basierend auf der UML-Version 1.4.2 definiert, welche üblicherweise (de-facto Standard) benutzt wird. Das ist aber nicht kritisch, weil das INTERLIS-Modell das normative Modell darstellt und nicht die UML-Visualisierung davon.

Zudem existiert aber ein Metamodell, welches nicht auf MOF basiert. Dieses Metamodell ist in INTERLIS selbst beschrieben und stellt keine Erweiterung des UML-Metamodells dar [INTERLIS, 2008].

Geometrien Für die Datenmodellierung sind keine Basismodelle wie z. B. die Modelle aus ISO 19107 notwendig, da das UML-Profil für INTERLIS geometrische Typen als primitive Datentypen (Stereotyp «primitive» (→ Abschnitt 2.4.1)) definiert. Für die vordefinierten Typen existieren keine Multi-Varianten (MultiPoint, MultiCurve, etc.), sondern ein Typ-Generator (*aggregate type generator* gemäß der *Norm ISO 11404 Information technology – General-Purpose Datatypes (GPD)*). Dieser erzeugt z. B. MultiPoints mittels der Codezeile `geom[*] : Point`.

Datentypen Die in INTERLIS definierten primitiven Datentypen sind in Tabelle 3.3 aufgeführt.

Stereotypen Die in INTERLIS definierten Stereotypen sind in Tabelle 3.4 aufgeführt. Daneben werden auch eine Reihe von Tagged Values benötigt, um die genauen Eigenschaften der INTERLIS-spezifischen Modell-Elemente definieren zu können. Diese sind jedoch hier nicht angegeben.

Implementierungsschema Ein mit INTERLIS erstelltes Modell ist so genau, dass kein manuell erstelltes (und nachzuführendes) Zwischenmodell in Form eines Implementierungsschemas erforderlich ist, um das Transferformat automatisch herzuleiten.

Mehrsprachigkeit In INTERLIS ist es möglich, mehrsprachige Modelle zu erstellen (z. B. Klasse „Gebaeude“ / „Batiment“).

Primitiver Datentyp	Erläuterung
TextType	Verschiedene Arten von Text
EnumerationType	Verschiedene Arten von Aufzählungen
EnumTreeValueType	Weitere Art von Aufzählung
AlignmentType	Vordefinierte Aufzählung (für ili1-Kompatibilität)
BooleanType	Vordefinierte Aufzählung (True/False)
NumericType	Ganze Zahlen, Fließkommazahlen, Festkommazahlen (inkl. min/max, Maßeinheit)
FormattedType	Anwenderdefinierbare serialisierte Datentypen
TimeType	Zeit inkl. min/max
DateType	Datum inkl. min/max
DateTimeType	Datum + Zeit inkl. min/max
OIDType	Anwenderdefinierbare Arten von Objekt-Identifikatoren
BlackboxType	Beliebige XML-Fragmente oder binäre Objekte
ClassType	Klassenname (primär für Metamodelle relevant)
AttributePathType	Attributname (primär für Metamodelle relevant)
CoordinateType	Koordinate (mit verschiedenen Konsistenzbedingungen)
PolylineType	Linie (mit verschiedenen Konsistenzbedingungen)
SurfaceType	Einzelfläche (mit verschiedenen Konsistenzbedingungen)
AreaType	Flächennetz (= Einzelfläche + vordefinierte Konsistenzbedingung)

Tabelle 3.3: Primitive Datentypen in INTERLIS

Stereotyp	Erläuterung
ModelDef	Definition eines Applikationsschemas (kann aber auch nur UnitDef oder GraphicDef enthalten)
TopicDef	Definition eines Datenbehälters
MetaDataBasketDef	Definition von CRS für die Verwendung in CoordinateType und von Symbolen für die Verwendung in GraphicDef
UnitDef	Definition von Maßeinheiten
FunctionDef	Definition von Funktionen für die Verwendung in Konsistenzbedingungen, ViewDef oder GraphicDef
ViewDef	Datenumbau (als Basis für komplexe Konsistenzbedingungen, berechnete Attribute/Assoziationen oder GraphicDef)
GraphicDef	Darstellungsbeschreibung
DrawingRule	Teil einer GraphicDef
LineFormTypeDef	Definition von neuen Kurvenstückformen
RunTimeParameterDef	Definition von Daten aus dem Laufzeitsystem, z. B. aktuelles Datum für die Verwendung in Konsistenzbedingungen

Tabelle 3.4: Stereotypen in INTERLIS

Fazit Für INTERLIS kann es kein UML-Profil im Sinne der Profil-Definition von UML aus Abschnitt 2.1.4 geben, da INTERLIS Sprachkonstrukte enthält, die keine reinen Spezialisierungen von Klassen des UML-Metamodells sind (z. B. GraphicDef). Hierbei verhält es sich entsprechend dem „UML-Profil“ der Norm ISO 19103 (→ Abschnitt 2.4), das wegen «CodeList» und «Union» kein echtes UML-Profil ist.

3.2.3 INSPIRE-UML-Profil

Das INSPIRE Generic Conceptual Model, welches bereits in Abschnitt 2.4.4 vorgestellt wurde, enthält auch alle Festlegungen zum INSPIRE-UML-Profil [JRC, 2009a].

Metamodell Als Metamodell liegt den INSPIRE-Anwendungsschemata das General Feature Model (→ Abschnitt 2.4.2) der Norm ISO 19109 zugrunde.

Datentypen Jedes INSPIRE-Anwendungsschema ist mit UML gemäß den Normen ISO 19103 und ISO 19109 zu definieren, mit der Ausnahme, dass die UML-Version 2.1 anstatt der Version 1.4.2 zu verwenden ist.

Geometrien Die geometrischen und topologischen Objekte basieren auf der Norm ISO 19107 und dürfen in INSPIRE-Anwendungsschemata ohne Einschränkungen verwendet werden. Es wird jedoch empfohlen, die Auswahl der Objekte, soweit möglich, auf diejenigen der Spezifikation *Simple feature access – Part 1: Common architecture* des OGC [OGC, 2006] einzuschränken. Die Spezifikation enthält nur 0-, 1-, 2-, und 2,5-dimensionale Geometrien, wobei alle Kurveninterpolationen linear sind. Die Geometrien werden in den INSPIRE-Anwendungsschemata als Datentypen entsprechend der Wertsemantik (→ Abschnitt 2.4.2) verwendet.

Die Geometrien der OGC-Spezifikation sind jedoch nicht identisch mit denen der Norm ISO 19107. Zudem unterstützt die OGC-Spezifikation im Gegensatz zur Norm ISO 19107 keine Topologien. Damit die ISO-19107-Geometrien dennoch problemlos verwendet werden können, schlägt die OGC-Spezifikation mögliche Entsprechungen zwischen den OGC-Geometrien und den ISO-Geometrien vor, welche in der Tabelle 3.5 aufgelistet sind. Dabei sind teilweise sogar mehrfache Zuordnungen zu einer OGC-Geometrie möglich.

Darüber hinaus beziehen sich die beiden Spezifikationen auf unterschiedliche Abstraktionsebenen. Die OGC-Spezifikation *Simple feature access - Part 1: Common architecture* ist eine implementierungsabhängige aber plattformunabhängige Spezifikation[1], wohingegen die Norm ISO 19107 abstrakt und plattformunabhängig ist. Folgendes Beispiel veranschaulicht den Unterschied: Die x- und y-Koordinaten der OGC-Spezifikation sind vom Typ Double. In der Norm ISO 19107 sind sie dagegen vom abstrakten Datentyp Number, welcher durch konkrete Datentypen instanziiert werden muss.

[1] Neben dieser Spezifikation existiert auch der plattformabhängige zweite Teil *Simple feature access – Part 2: SQL option* für die Speicherung von Geometrien in Datenbanken

Simple Feature Access	ISO 19107
Point	GM_Point, DirectPosition
Curve	GM_Curve, GM_GenericCurve, GM_CurveSegment, GM_LineString, GM_LineSegment
LineString	GM_LineString
Polygon	GM_GenericSurface, GM_Surface, GM_SurfacePatch, GM_Polygon
PolyhedralSurface	GM_PolyhedralSurface als Subtyp von GM_Surface
GeometryCollection	GM_Aggregate und sein Subtyp GM_MultiPrimitive
MultiPoint	GM_MultiPoint
MultiLineString	GM_MultiCurve, GM_MultiLineString
MultiPolygon	GM_MultiSurface

Tabelle 3.5: Geometrische Entsprechungen zwischen der OGC-Simple-Feature-Spezifikation und ISO 19107 (Quelle: [OGC, 2006])

Attribute ohne Wert Das INSPIRE Generic Conceptual Model definiert zusätzliche Regeln, falls Daten für bestimmte Attribute keinen Wert aufweisen [JRC, 2009a]:

- *Der Wert ist in der realen Welt nicht vorhanden oder nicht anwendbar*: Das Attribut wird in diesem Fall mit einer Multiplizität, deren untere Schranke 0 ist, modelliert und erhält einen leeren Wert. Beispiel: Eine Klasse „Straße“ besitzt das Attribut „streetName“, jedoch müssen nicht alle Straßen in Europa zwingend einen Straßennamen besitzen.
- *Der Wert ist zwar im Datensatz nicht vorhanden, aber möglicherweise in der realen Welt*: Dann erhält das Attribut den Stereotyp «voidable». Als Wert wird „Unknown“ vergeben, wenn der Wert dem Datenanbieter nicht bekannt ist bzw. von ihm nicht erfasst wurde, und „Unpopulated“, wenn der Wert nicht Teil des Datensatzes ist, den der Datenanbieter bereitstellt.

Stereotypen Das INSPIRE Generic Conceptual Model legt auch fest, welche Stereotypen im INSPIRE-UML-Modell (engl. Consolidated INSPIRE UML Model) eingesetzt werden dürfen. Der Großteil der Stereotypen wurde bereits in anderen Normen und Standards definiert und wird hier wiederverwendet. Tabelle 3.6 listet die Stereotypen auf und nennt die Quellen, wo sie ursprünglich definiert wurden. Für die zugehörigen Tagged Values wird auf [JRC, 2009a] verwiesen.

Kodierung Die Kodierung (→ Abschnitt 2.1.5) wird, wie auch bei den AAA-Anwendungsschemata (→ Abschnitt 3.2.1), gemäß den Kodierungsregeln (→ Abschnitt 2.1.5) der Normen ISO 19136 Annex E [ISO, 2007a] und ISO 19139 für Metadaten [ISO, 2007b] erstellt, wobei noch ein paar zusätzliche Regeln definiert sind. Als Datenaustauschformat wird ebenfalls GML verwendet.

Stereotyp	Herkunft	Erläuterung
«applicationSchema»	ISO 19136 Annex E	Definiert ein UML-Paket, das ein UML-Anwendungsschema repräsentiert
«leaf»	ISO 19103	(→ Abschnitt 2.4.3)
«featureType»	ISO 19136 Annex E	FeatureType gemäß der Norm ISO 19109 (→ Abschnitt 2.4.2)
«placeholder»	INSPIRE Generic Conceptual Model	FeatureType ist Platzhalter für einen FeatureType, der erst ist einem zukünftigen Thema spezifiziert wird
«type»	UML	(→ Abschnitt 2.4.1)
«dataType»	UML	(→ Abschnitt 2.4.1)
«union»	ISO 19107	Typ darf zur Laufzeit nur genau eines der angegebenen Alternativen als Wert besitzen (→ Abschnitt 2.4.3)
«enumeration»	UML	(→ Abschnitt 2.4.1)
«codeList»	ISO 19103	(→ Abschnitt 2.4.3)
«import»	UML	Die Inhalte eines UML-Pakets werden in ein anderes UML-Paket importiert
«voidable»	INSPIRE Generic Conceptual Model	siehe oben
«lifeCycleInfo»	INSPIRE Generic Conceptual Model	Attribut enthält Informationen über den Lebenszyklus eines Objekts
«version»	INSPIRE Generic Conceptual Model	An einer Assoziation ist eine bestimmte Version eines FeatureTypes beteiligt, nicht der FeatureTyp an sich

Tabelle 3.6: Stereotypen des INSPIRE-UML-Profils

Im Gegensatz zu den AAA-Anwendungsschemata muss die Kodierung bei INSPIRE nicht zwingend in zwei Stufen durchgeführt werden mit einem Implementierungsschema (→ Abschnitt 2.1.5) als Zwischenstufe. Die Möglichkeit des Implementierungsschemas wird zwar genannt, jedoch konnten für die Annex-I-Themen keine Anforderungen identifiziert werden, die einen Zwischenschritt über das Implementierungsschema erfordern würden. Zudem wird in den INSPIRE-Dokumenten auch nicht näher auf mögliche für die Kodierung einsetzbare Softwareprodukte eingegangen [JRC, 2009b].

Fazit Wie das AAA-„UML-Profil“ ist auch das INSPIRE-UML-Profil kein UML-Profil im Sinne der UML-Profil-Definition der OMG (→ Abschnitt 2.1.4). Das INSPIRE-UML-Profil basiert auf dem „UML-Profil“ der Norm ISO 19103 (→ Abschnitt 2.4.3) und kann somit selbst ebenfalls kein echtes UML-Profil darstellen.

Da für die Annex-I-Themen kein Zwischenschritt über ein Implementierungsschema erforderlich ist, sind in den INSPIRE-Dokumenten auch keine entsprechenden Regeln für

die Herleitung eines Implementierungsschemas enthalten. Die Kodierungsregeln sind den Normen ISO 19136 Annex E und ISO 19139 entnommen. Die Regeln sind dort in textueller Form definiert.

Abbildung 3.6 zeigt eine Einordnung des INSPIRE-„UML-Profils“ in UML sowie in die Normen ISO 19103, ISO 19107 und ISO 19109. Mit „Modellierungssprache des TC211“ soll verdeutlicht werden, dass die Definition der Stereotypen «codeList» und «union» in der Norm ISO 19103 nicht ein UML-Profil zur Folge hat, sondern eine neue Modellierungssprache. Diese Modellierungssprache besitzt zwar die Syntax von UML, nicht jedoch deren Semantik, weshalb sie eine eigene, neue Sprache darstellt. Das gleiche gilt für „Modellierungssprache von INSPIRE“.

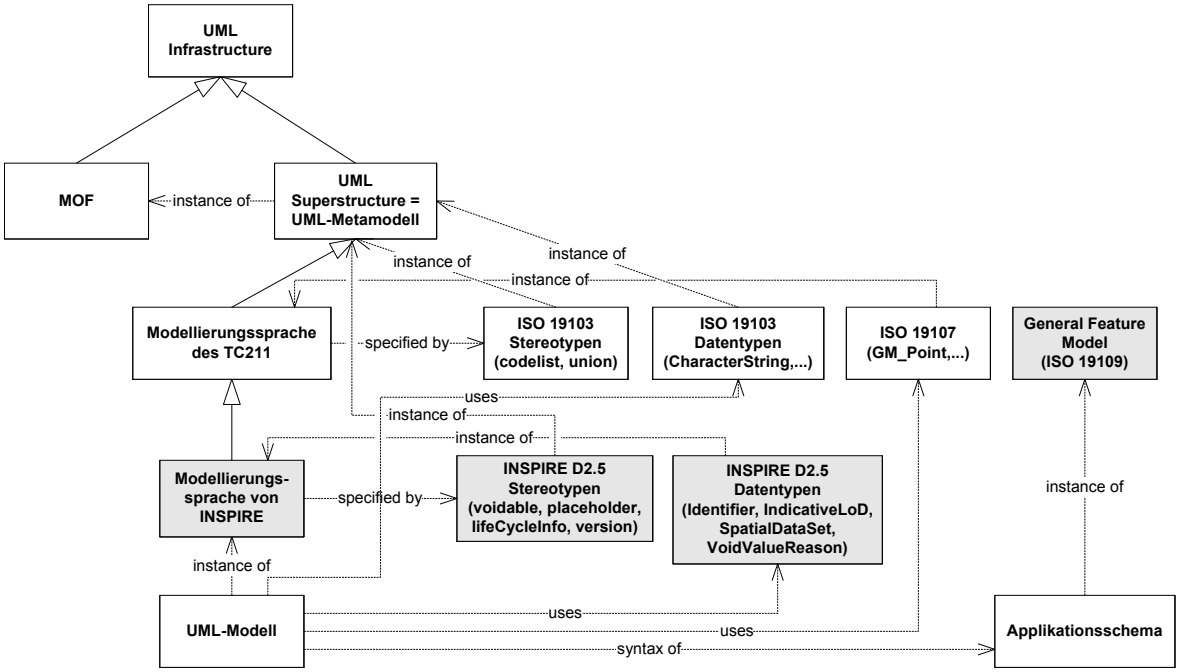


Abbildung 3.6: Zusammenhang zwischen INSPIRE-„UML-Profil“, UML, ISO 19103, ISO 19107 und ISO 19109

3.2.4 Gegenüberstellung der UML-Profile

Tabelle 3.7 enthält abschließend eine Gegenüberstellung der oben vorgestellten Konzepte aus Sicht einer Person, die Anwendungsschemata gemäß diesen Konzepten erstellen soll bzw. Transformationsregeln zwischen Modellen, die diese Konzepte einsetzen, definiert.

Konzept	AAA	INTERLIS	INSPIRE
Metamodell (CSL-Semantik)	General Feature Model mit Erweiterung um Metaklasse AA_ObjektOhneRaumbezug	UML-Metamodell-Erweiterung bzw. alternativ Nicht-MOF-basiertes Metamodell	General Feature Model
CSL-Syntax	ISO 19103 (UML 1.4.2) + ISO 19109	UML 1.4.2 bzw. alternativ textuelle Sprache	ISO 19103 (UML 2.1) + ISO 19109
Transferformat	XMI 1.0		XMI 2.1
Geometrietypen	GM_Point, GM_Curve, GM_PolyhedralSurface, GM_PointRef, GM_CompositeCurve, GM_CompositeSurface, GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, TS_PointComponent, TS_CurveComponent, TS_SurfaceComponent, TS_Face, TP_Complex, GM_Solid, GM_SurfaceBoundary, GM_TriangulatedSurface, GM_OrientableSurface, GM_CompositeSolid, TS_Solid, TS_Feature, TS_Theme	PolylineType, SurfaceType, AreaType	ISO 19107, jedoch Einschränkung auf OGC Simple Feature empfohlen
Einbindung räumlicher Attribute	Geometrietypen bilden Grundgerüst, Featuretypen sind Ableitung davon	Wertsemantik	Wertsemantik

Konzept	AAA	INTERLIS	INSPIRE
Datentypen	ISO 19103	TextType, EnumerationType, EnumTreeValueType, AlignmentType, BooleanType, NumericType, FormattedType, TimeType, DateType, DateTimeType, OIDType, BlackboxType, ClassType, AttributePathType, CoordinateType	ISO 19103
Stereotypen	«CodeList», «Leaf», «FeatureType», «Type», «Enumeration», «DataType», «Union»	ModelDef, TopicDef, MetaDataBasketDef, UnitDef, FunctionDef, ViewDef, GraphicDef, DrawingRule, LineFormTypeDef, RunTimeParameterDef	«applicationSchema», «leaf», «featureType», «placeholder», «type», «dataType», «union», «enumeration», «codeList», «import», «voidable», «lifeCycleInfo», «version»
Kodierungsregeln	ISO 19118 + ISO 19136 Annex E / ISO 19139	1) ISO 19118 + eigene Regeln 2) ISO 19118 + eigene Regeln, die zu ISO 19136 führen, aber nicht gemäß den Regeln von Annex E	ISO 19118 + ISO 19136 Annex E / ISO 19139 + zusätzliche Regeln
Transferformat	ISO 19136	1) eigenes XML-basiertes Format 2) ISO 19136	ISO 19136
Implementierungsschema / -regeln	ja / ja	nein / nein	ja, bei Bedarf / nein
Mehrsprachige Modelle	nein	ja	nein

Tabelle 3.7: Gegenüberstellung der UML-Profile aus Sicht eines Modellierers

4 Problemstellung

In den vorhergehenden Kapiteln wurde die gegenwärtige Situation bezüglich der Datenmodelle Deutschlands, der Schweiz und der Europäischen Union aus allgemeiner Sichtweise beschrieben. Dieses Kapitel stellt nun konkret einzelne Problemfälle vor, die im Rahmen der Bearbeitung der Anwendungsfälle aus Abschnitt 3.1 aufgetreten sind und auf den Ausführungen in den vorhergehenden Kapiteln basieren. Dabei wird eine Unterteilung der Probleme nach Modellierungssprache, Kodierung und Modelle vorgenommen.

4.1 Modellierungssprache

4.1.1 UML-Abänderung

Modellierung in UML	Kodierung in GML

«featureType**»** AdministrativeUnit
beginLifespanVersion: DateTime
	`<AdministrativeUnit>`
	`  <beginLifespanVersion>`
	`  **2010-02-18T10:15:30Z**`
	`  </beginLifespanVersion>`
	`</AdministrativeUnit>`

«featureType**»** AdministrativeUnit
«voidable»
beginLifespanVersion: DateTime
	`<AdministrativeUnit>`
	`  <beginLifespanVersion xsi:nil="true"/>`
	`</AdministrativeUnit>`

Abbildung 4.1: UML-Abänderung

Abbildung 4.1 zeigt die Klasse AdministrativeUnit aus den INSPIRE Data Specifications. Die Klasse enthält das Attribut beginLifespanVersion. Der Datentyp DateTime legt dabei fest, dass das Attribut beginLifespanVersion genau „einen Wert“ hat, der vom Typ „DateTime“ ist.

In den INSPIRE Data Specifications ist es jedoch möglich, den Stereotyp «voidable» anzugeben, der festlegt, was passiert, falls Daten für bestimmte Attribute keinen Wert aufweisen (→ Abschnitt 3.2.3). Mit «voidable» umfasst der Wertebereich des Attributs demnach nicht nur DateTime-Informationen, sondern auch „kein Wert vorhanden“, was dem Datentyp DateTime widerspricht.

Dies stellt eine Änderung der Modellierungssprache UML dar, die fachlich begründbar, jedoch nicht konform zur UML-Spezifikation ist, und die Maschineninterpretierbarkeit des Datenmodells beeinträchtigt. Ebenso können dadurch bei der Transformation zwischen verschiedenen Datenmodellen Probleme auftreten.

4.1.2 Unterschiedliche UML-Versionen

Das AAA-Modell sowie das TLM liegen in UML-Version 1.4.2 vor, das INSPIRE-Modell in UML-Version 2.1. Zudem befindet sich die Norm ISO 19103 derzeit in Revision (→ Abschnitt 3.2.1), wodurch in Zukunft Modelle, die auf dieser Norm basieren, auch in UML-Version 2.2 vorliegen könnten. Es kann aber auch der Fall auftreten, dass mit UML-Modellen gearbeitet wird, die nicht auf der Norm ISO 19103 basieren. Auch diese Modelle sind gemäß einer bestimmten UML-Version definiert.

Ein Beispiel dafür, dass die Verwendung unterschiedlicher UML-Versionen zu Problemen führen kann, ist der Datentyp Boolean, der in der Norm ISO 19103 aber auch in der UML-Version 2.1 definiert ist. Die UML-Version 1.4.2 selbst schreibt die Verwendung bestimmter Datentypen nicht vor und stellt auch keine vordefinierten Datentypen bereit. Das ISO-19103-„UML-Profil“, welches auf der UML-Version 1.4.2 basiert, muss deshalb zwingend eine Reihe von Basisdatentypen selbst definieren (→ Abschnitt 2.4.3). Die UML-Version 2.1 dagegen stellt eine Reihe von primitiven Datentypen bereit (→ Abschnitt 2.4.1). Bei UML-Modellen, die gemäß der Norm ISO 19103 in der UML-Version 1.4.2 modelliert wurden, kann dies zu Kompatibilitätsproblemen führen. Die Norm ISO 19103 definiert z. B. den Datentyp Boolean. Die UML-Version 2.1 dagegen stellt den Typ Boolean selbst bereit. Wird nun ein UML-Modell von UML-Version 1.4.2 nach UML-Version 2.1 migriert, muss zuvor geklärt werden, wie dieser Datentyp in UML-Version 2.1 behandelt wird.

Darüber hinaus bedeutet dies für UML-Modelle, die zwar gemäß der Norm ISO 19103, jedoch mit UML-Version 2.1 modelliert werden, wie es bei INSPIRE der Fall ist, dass zuvor festgelegt werden muss, welcher Datentyp für die Modellierung eingesetzt werden soll.

4.1.3 Ein UML-Profil pro Projekt

Jedes Projekt kann sein eigenes UML-Profil definieren, weshalb man bei einer Modelltransformation unter Umständen mit sehr vielen unterschiedlichen UML-Profilen konfrontiert wird. Dies ist auch beim Anwendungsfall Transformation nach INSPIRE (→ Abschnitt 3.1.1) der Fall, wo die Quellmodelle (AAA, INTERLIS) und das Zielmodell (INSPIRE) voneinander abweichende „UML-Profile“ aufweisen.

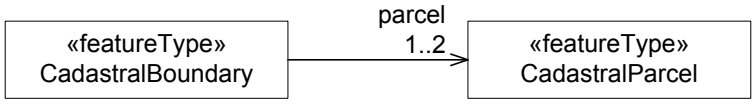
Die Definition von Stereotypen stellt in UML keine Schwierigkeit dar, weshalb diese Möglichkeit, UML an seine Einsatzzwecke anzupassen, komfortabel erscheinen mag. Jedoch sind damit auch Konsequenzen verbunden, über die man sich im Voraus bewusst sein muss. So wurde in Abschnitt 2.4.3 zur Norm ISO 19103 erläutert, dass die Stereotype «codeList» und «union» keine Spezialisierungen von Modellelementen des UML-Metamodells darstellen. Dies bedeutet, dass dadurch eine neue Modellierungssprache definiert wird, die zwar die Syntax von UML verwendet, jedoch eine eigene Semantik aufweist.

Jedes Projekt, welches auf diese Weise Stereotype definiert, definiert dadurch gleichzeitig eine neue Modellierungssprache mit projektspezifischer Spezialsemantik. Dies wiederum stellt ein Problem dar für die Verarbeitung der Modelle mittels Transformationssprachen bzw. Modellierungswerkzeugen.

4.2 Kodierungsregeln

4.2.1 Kodierung von Referenzen

UML-Modell



XML-Schema

```
<complexType name="CadastralBoundaryType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element maxOccurs="2" name="parcel"
                 nillable="true" type="gml:ReferenceType">
          <annotation>
            <appinfo>
              <targetElement xmlns="http://www.opengis.net/
                  gml/3.2">cp:CadastralParcel</targetElement>
            </appinfo>
          </annotation>
          <annotation>
            <appinfo>
              <targetElement xmlns="http://www.opengis.net/
                  gml/3.2">cp:CadastralParcel</targetElement>
            </appinfo>
          </annotation>
        </element>
      ...
```

Kodierung in GML

```
<CadastralParcel gml:id="A00012ab">…</CadastralParcel>
<CadastralBoundary>
  <parcel xlink:href="urn:x-inspire:object:id:DENW1000:A00012ab"/>
  …
</CadastralBoundary>
```

anstelle von: `<parcel xlink:href="#A00012ab"/>`

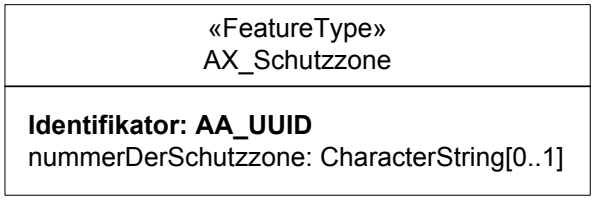
Abbildung 4.2: Kodierung von Referenzen

Der UML-Modell-Ausschnitt aus den INSPIRE Data Specifications in Abbildung 4.2 legt fest, dass eine Flurstücksgrenze an 1 bis 2 Flurstücke angrenzen darf (parcel 1..2). Jedoch ist weder aus dem UML-Modell noch aus dem XML-Schema erkennbar, dass für Referenzen auf die Flurstücksgrenzen nicht einfach der Wert des XML-Fragmentidentifikators gml:id herangezogen werden darf. Stattdessen sind die Referenzen mittels einer URN der Form urn:x-inspire:object:id:<namespace>:<local identifier> anzugeben, welche mit der URN für

Identifikatoren identisch ist (→ Abschnitt 4.1.1). Jede betroffene Referenz muss deshalb im Transformationswerkzeug von Hand programmiert bzw. konfiguriert werden oder es werden spezifische Transformationswerkzeuge für INSPIRE erstellt.

4.2.2 Kodierung von Identifikatoren

Modellierung in UML



Kodierung in GML

```
<AX_Schutzzone gml:id="DE_A00DPQO000MXH">
  <gml:identifier codesSpace ="http://www.adv-online.de/">
    urn:adv:oid:DE_A00DPQO000MXH
  </gml:identifier>
  <nummerDerSchutzzone>017</nummerDerSchutzzone>
</AX_Schutzzone>
```

Abbildung 4.3: Kodierung von Identifikatoren

Dieses Beispiel stammt aus dem AAA-Modell. Die Klasse AX_Schutzzone in Abbildung 4.3 enthält das Attribut Identifikator. Dieses Attribut wird im AAA-Austauschformat NAS nicht als Kindelement <Identifikator> des Elements <AX_Schutzzone> kodiert, sondern als sein Attribut gml:id. Im AAA-„UML-Profil“ (→ Abschnitt 3.2.1) ist jedoch festgelegt, dass der Identifikator zusätzlich auch gemäß der Norm ISO 19136 mittels des dort vordefinierten Elements gml:identifier zu kodieren ist [AdV, 2008]. Dabei fällt auf, dass sich die Art der Kodierung des Identifikators unterscheidet. Beim Attribut gml:id ist nur der aus 16 Zeichen bestehende Identifikator anzugeben, beim Element gml:identifier dagegen ist der Identifikator als URN (Uniform Resource Locator) in der Form „urn:adv:oid:<Identifikator>“ zu kodieren.

Aus dem UML-Modell ist nicht erkennbar, dass der Identifikator zweifach zu kodieren ist. Ebenso ist weder aus dem UML-Modell noch aus dem GML-Anwendungsschema ersichtlich, wo die Informationen zur korrekten Kodierung nachgelesen werden können.

Gleiches gilt für das INSPIRE-„UML-Profil“, außer dass der beim Element gml:identifier anzugebende Identifikator die Form „urn:x-inspire:object:id:<namespace>:<local identifier>“ besitzt.

Auch für die Transformation sind solche Informationen von Bedeutung. Dieses Beispiel zeigt nämlich, dass nicht alle Informationen aus dem Modell abgelesen werden können, sondern auch die unterschiedlichen Sprachparadigmen (→ Abschnitt 2.1.4) berücksichtigt werden müssen, in diesem Fall das XML-Paradigma.

4.2.3 Formale Kodierungsregeln

In den Ausführungen zum AAA-„UML-Profil“ (→ Abschnitt 3.2.1) und zum INSPIRE-„UML-Profil“ (→ Abschnitt 3.2.3) wurde dargelegt, dass die Kodierungsregeln der jeweiligen Spezifikationen in Prosa vorliegen, also in informeller Form. Damit die Kodierung automatisch durchgeführt werden kann, ist es nicht notwendig, dass die Kodierungsregeln in formaler Form basierend auf einer formalen Sprache vorliegen. Die Kodierungsregeln müssen vielmehr vorrangig so genau in Prosa beschrieben sein, dass eine eindeutige Implementierung der Kodierungsregeln gewährleistet ist. Andernfalls besteht die Gefahr, dass die beschriebenen Kodierungsregeln durch den Implementierer auf unterschiedliche Art interpretiert und umgesetzt werden. Durch eindeutig beschriebene Kodierungsregeln ist dagegen sichergestellt, dass die Kodierungsregeln auf jedem System ohne Ausnahme auf die gleiche Weise ausgeführt werden.

4.2.4 Nicht dokumentierte Kodierungsregeln

Sind die Kodierungsregeln nicht dokumentiert, so können diese dennoch erstellt werden, wenn sowohl das konzeptuelle Modell als auch das Transferformatschema bekannt sind. Dieses Vorgehen ist jedoch fehleranfällig und deshalb nicht sehr sinnvoll. In diesem Fall wäre es vorteilhafter, bei der Erstellung von Transformationsregeln zwischen Datenmodellen direkt das Transferformatschema zu lesen. Damit die modellbasierte Transformation jedoch korrekt durchgeführt werden kann, muss die komplette Semantik, die im konzeptuellen Modell vorhanden ist, auch im Transferformatschema vorhanden sein.

4.2.5 Formale Regeln zur Herleitung von Implementierungsschemata

Für die Herleitung von Implementierungsschemata anhand formaler Regeln gelten die gleichen Aussagen, die in Abschnitt 4.2.3 bezüglich der formalen Kodierungsregeln getroffen wurden. D. h., die Regeln zur Herleitung von Implementierungsschemata müssen nicht formal definiert sein, sie müssen jedoch so genau beschrieben sein, dass die Implementierungsschemata eindeutig aus den konzeptuellen Modellen hergeleitet werden können.

4.2.6 Nicht dokumentierte Herleitung von Implementierungsschemata

Hier gelten ebenfalls die gleichen Aussagen, die in Abschnitt 4.2.4 bezüglich der nicht dokumentierten Kodierungsregeln getroffen wurden. D. h., statt der Rekonstruktion von Regeln, aus denen die Implementierungsschemata hergeleitet werden können, wäre es sinnvoller, direkt mit dem Implementierungsschema zu arbeiten, vorausgesetzt, dieses enthält die komplette Semantik des konzeptuellen Schemas. Darüber hinaus ist diese Anforderung nur relevant, wenn überhaupt ein Implementierungsschema vorgesehen ist.

4.3 Modelle

4.3.1 Fehlende konzeptuelle Modelle

Beim Anwendungsfall aus Abschnitt 3.1.2 wird eine Transformation vom AAA-Modell in das Modell des MLR-Geodatenservers durchgeführt. Für das Zielmodell steht jedoch kein konzeptuelles Modell zur Verfügung. Um dennoch eine Transformation auf konzeptueller Ebene durchführen zu können, musste das Zielmodell mit UML nachmodelliert werden. Hierbei ergaben sich einige Probleme:

- Als Grundlage für die Nachmodellierung diente ein Datenbankschema, in dem die Zieldaten gespeichert sind. Diesem Datenbankschema liegt das ER-Paradigma (→ Abschnitt 2.1.4) zugrunde, zudem wird für die Daten als Transferformatschema das Shape-Format verwendet. Shape-Dateien können jeweils nur Geometrien eines Typs enthalten, also z. B. Punkt-, Linien- oder Flächengeometrien. Dies spiegelte sich auch im Datenbankschema wieder und entsprechend in der UML-Modellierung.
- Wird ein konzeptuelles Modell erstellt, so sollten die darin verwendeten Konzepte unabhängig von einer späteren Implementierung sein. Einer Implementierung liegen nämlich immer auch die Konzepte einer Plattform zugrunde (hier ER-Paradigma und Shape-Format), wohingegen die im konzeptuellen Modell definierten Fachkonzepte nicht mehr eindeutig erkennbar sind. Ohne Fachwissen ist deshalb kein fehlerfreies konzeptuelles Zielmodell für diesen Anwendungsfall erstellbar. So können z. B. beim Datenbankschema Optimierungen vorgenommen worden sein, welche ohne Wissen bezüglich der Fachkonzepte, die dem Datenbankschema zugrunde liegen, nicht rückgängig gemacht werden können.
- Die Modellierung des Zielmodells erfolgte in UML. Es muss festgelegt werden, ob die Modellierung basierend auf einem UML-Profil erfolgen soll, und wenn ja, auf welchem UML-Profil.
- Ebenso muss festgelegt werden, welche UML-Version für die Modellierung des Zielmodells eingesetzt wird.

4.3.2 Einbindung räumlicher Attribute

Bei den Ausführungen zum General Feature Model der Norm ISO 19109 (→ Abschnitt 2.4.2) wurde bereits dargelegt, dass Geometrien mittels Wertsemantik und Referenzsemantik in einem Anwendungsschema angegeben werden können. Dies kann bei einer Transformation Probleme verursachen, wenn Quell- und Zielschema unterschiedliche Semantiken aufweisen. Es ist dann eine Transformation zwischen Geometrie-Datentyp (Wertsemantik) und Geometrie-Klasse (Referenzsemantik) durchzuführen, wie in Abbildung 4.4 dargestellt ist.

Darüber hinaus kann aber auch der Fall eintreten, dass die Geometrien auf komplett andere Weise in ein Anwendungsschema integriert sind. Ein Beispiel hierfür ist das AAA-Anwendungsschema, bei dem die Geometrietypen das Grundgerüst bilden und die Featuretypen Ableitungen davon sind (→ Abschnitt 3.2.1).

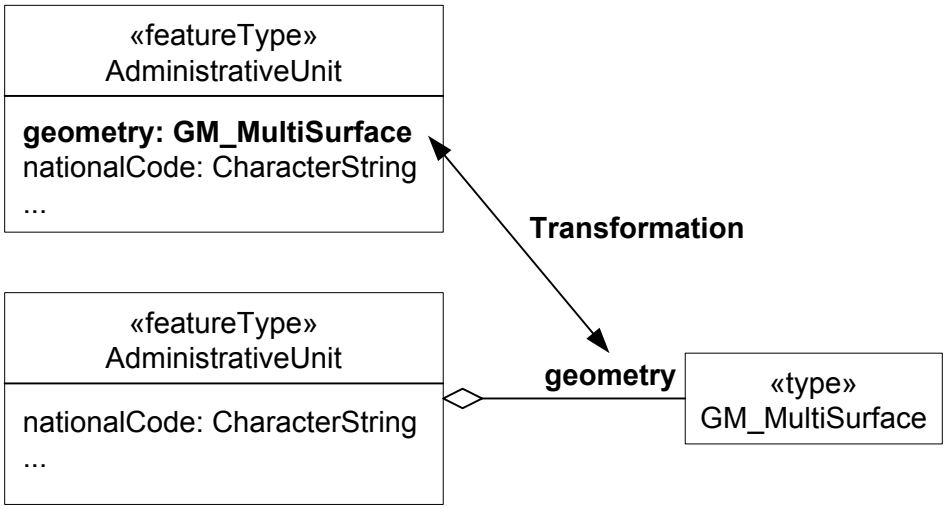


Abbildung 4.4: Problematik Wert- und Referenzsemantik

4.3.3 Komplexität der Modelle

Die Gegenüberstellung der „UML-Profile“ von AAA, INTERLIS und INSPIRE in Abschnitt 3.2.4 zeigte, dass die Profile eine unterschiedliche Komplexität aufweisen und damit auch die darauf basierenden Modelle. Diese Unterschiede in der Komplexität müssen bei der Transformation berücksichtigt werden können. Gelöst werden kann das Problem mit Transformationssprachen jedoch nicht.

5 Anforderungen an Transformationssprachen

Dieses Kapitel definiert allgemeine Anforderungen an Transformationssprachen, die zur Lösung der im vorhergehenden Kapitel beschriebenen Probleme beitragen können. Diese Anforderungen müssen erfüllt sein, damit die Transformation und Kodierung basierend auf maschinenlesbaren und mschineninterpretierbaren Modellen durchgeführt werden kann. Andernfalls können die Potenziale des modellbasierten Ansatzes nicht vollständig ausgeschöpft werden.

5.1 Sprachparadigma von Transformations- und Modellierungssprache

Die Transformationssprache muss zur Modellierungssprache passen, d.h. sie sollte auf dem gleichen Sprachparadigma (→ Abschnitt 2.1.4) beruhen wie die Modellierungssprache mit der die zu transformierenden Modelle erstellt werden. Negativbeispiele sind z. B. XML (XML-Paradigma) als Modellierungssprache mit SQL (ER-Paradigma) als Transformationssprache oder GML (OO-Paradigma) als Modellierungssprache mit XSLT (XML-Paradigma) (→ Abschnitt 6.1) als Transformationssprache.

5.2 Komplexe Transformationen

Ein weiterer wichtiger Punkt ist die Verständlichkeit von Transformationen. Aufgrund der Heterogenität und der großen Unterschiede in der Komplexität der Modelle kann die Gesamtheit der Transformationsregeln ebenfalls sehr komplex und unübersichtlich werden. Eine Transformationssprache muss so definiert sein, dass auch komplexe Transformationen verständlich definiert und ausgeführt werden können, ohne dass hierbei Einschränkungen bezüglich der Einsetzbarkeit umfangreicher Modelle entstehen.

5.3 Fehlerbehandlung während der Transformation

Bei der Erstellung von Transformationsregeln kann es vorkommen, dass seitens des Anwenders einzelne durchzuführende Transformationen nicht durch entsprechende Regeln definiert werden. Ein Grund hierfür könnte z. B. in der Komplexität der Modelle liegen, welche dazu führt, dass der Anwender einzelne notwendige Transformationen einfach übersieht.

Für die Transformation selbst bedeutet dies, dass dadurch bestimmte Werte in den Transformationsregeln nicht behandelt werden und bei der Transformation zu Fehlern führen. Es treten somit Fehler aufgrund unvollständiger Regeln auf. Um jedoch zu vermeiden, dass sich diese Fehler auf das Zielmodell übertragen, muss während der Transformation eine Fehlerbehandlung durchgeführt werden. Die Fehlerbehandlung sollte automatisch erfolgen, ohne dass eine Interaktion seitens des Anwenders erforderlich ist.

5.4 Unterstützung unterschiedlicher Modellierungssprachen

Beim Anwendungsfall des Projekts mdWFS (→ Abschnitt 3.1.1) wurden die AAA- und INSPIRE-Datenmodelle in der Modellierungssprache INTERLIS nachmodelliert, damit die Transformation in einer auf Basis von INTERLIS entwickelten Umgebung durchgeführt werden konnte. Dieses Vorgehen stellt jedoch im produktiven Betrieb einen nicht realistischen Aufwand dar. Eine grundlegende Anforderung an eine in der Praxis einsetzbare Transformationssprache besteht somit darin, dass keine Nachmodellierung der Modelle in einer festgelegten Modellierungssprache durchzuführen ist. Diese Anforderung ist für UML nur erfüllbar, wenn die Modellierungssprachen UML-Profile besitzen die konform zur UML-Spezifikation sind. Ansonsten ist die Transformation nur mühsam durchführbar.

5.5 Berücksichtigung unterschiedlicher Versionen einer Modellierungssprache

Wie in Abschnitt 4.1.2 beschrieben ist, liegen das AAA-Modell und das TLM gemäß UML-Version 1.4.2 vor, das INSPIRE-Modell dagegen in UML-Version 2.1. Zudem kann auch der Fall auftreten, dass UML-Modelle transformiert werden, die nicht auf der Norm ISO 19103 basieren. Auch diese Modelle sind entsprechend einer bestimmten UML-Version definiert. Darüber hinaus können Versionsunterschiede aber auch bei jeder anderen Modellierungssprache existieren. Die sprachlichen Unterschiede, die in den verschiedenen Versionen vorliegen, müssen bei einer Transformation berücksichtigt werden. Es wird deshalb eine Transformationsbeschreibung benötigt, welche die Unterschiede in den Modellversionen berücksichtigt bzw. entsprechend anpassbar ist.

5.6 Effizienz bei der Ausführung von Transformationen

Ein Anwender verwendet eine Transformationssprache, um Transformationen zu definieren. Deshalb ist für ihn zuallererst einmal wichtig, dass er die Transformationen mit der Sprache übersichtlich und komfortabel beschreiben kann. Dies bedeutet jedoch noch lange nicht, dass die Sprache damit auch optimal für die Ausführung der Transformationen geeignet

ist. Bei der Ausführung ist nämlich vielmehr von Bedeutung, dass die definierten Transformationen effizient vom Transformationswerkzeug abgearbeitet werden können. Gerade Geodaten gehen oft mit großen Datenmengen einher, weshalb eine effiziente Ausführung von Transformationen insbesondere im GIS-Bereich gefordert ist.

Man unterscheidet in der Informatik zwischen dem *imperativen* und dem *deklarativen Programmierparadigma*. Alle Programmiersprachen und damit auch Transformationssprachen basieren auf einem dieser Paradigmen. Die Transformationssprache XSLT (→ Abschnitt 6.1) beispielsweise entspricht dem deklarativen Paradigma. Deklarative Sprachen weisen gegenüber imperativen Sprachen keine Nachteile auf. Es ist jedoch aus Effizienzsicht darauf zu achten, dass sie für Transformationen optimiert sind.

6 Bestandsaufnahme geeigneter Transformationssprachen

Es existiert eine Vielzahl von Sprachen für die Modelltransformation. Einen Eindruck davon vermitteln Czarnecki und Helsen in [Czarnecki et al., 2006], die in ihren Untersuchungen zur Modelltransformation 32 verschiedene Transformationssprachen heranziehen. Dazu zählen:

- *Sprachen, die in der Literatur publiziert sind*: z. B. VIATRA2 (VIsual Automated model TRAnsformations) Framework, Tefkat, ATL (Atlas Transformation Language), UML-LX, BOTL (Bidirectional Object-oriented Transformation Language), MOLA (MOdel transformation LAnguage) und Kermeta
- *Sprachen der OMG*: QVT-Core, QVT-Relations und QVT-Operational
- *Sprachen, die in Open-Source-Werkzeugen implementiert sind*: z. B. Andro-MDA, openArchitectureWare, Fujaba (From UML to Java And Back Again) und MTF (Model Transformation Framework)
- *Sprachen, die in kommerziellen Werkzeugen implementiert sind*: z. B. XMF-Mosaic, OptimalJ und MetaEdit+

Es würde den Rahmen dieser Studie bei weitem überschreiten, alle diese Sprachen dahingehend zu analysieren, inwieweit sie sich für die modellbasierte Transformation im Kontext dieser Studie eignen. Nachfolgend werden deshalb nur einzelne ausgewählte Ansätze kurz vorgestellt, die derzeit bei der Transformation von Geodaten eine Rolle spielen.

6.1 XSL Transformations (XSLT)

Auch heute ist es in der Geoinformatik noch Stand der Technik, Transformationsregeln auf Ebene der Transferformate zu definieren und nicht auf Modellebene. Im GIS-Bereich hat insbesondere das Transferformat Geography Markup Language (GML) [ISO, 2007a] eine sehr starke Verbreitung gefunden. GML ist ein XML-basiertes Format, weshalb bei der Verwendung von GML der Fokus bei der Transformation auf XML-Schemata liegt, welche die GML-Anwendungsschemata repräsentieren.

Wie in Abschnitt 2.1.5 beschrieben wurde, sind XML-Schemata nicht Modelle im eigentlichen Sinn, sondern sie sind auf Formatebene anzusiedeln. Somit kann zwar durchaus eine semantische Transformation durchgeführt werden, jedoch wird diese auf der syntaktischen Ebene definiert.

Die Transformation der XML-Schemata wird normalerweise mittels *XSL Transformations* (XSLT) [W3C, 2007] durchgeführt. XSLT ist ein Standard des World Wide Web Consortium (W3C) und definiert eine Sprache zur Transformation von XML-Dateien. XSLT ist sehr stark auf das XML-Paradigma ausgerichtet, weshalb seine Einsatzmöglichkeit mit anderen Formaten möglich, aber nur beschränkt nützlich ist.

Dies trifft auch auf die Verwendung von XSLT für die Transformation von GML-Anwendungsschemata zu, da diese dem OO-Paradigma entsprechen. Die Norm ISO 19136, die das GML-Format definiert, schreibt hierzu: „To ensure proper use of the conceptual modelling framework of the ISO 19100 series of International Standards, all application schemas are expected to be modelled in accordance with the General Feature Model as specified in ISO 19109." D. h., GML ist zwar XML-basiert, da für die Definition der GML-Anwendungsschemata XML-Syntax eingesetzt wird. Die GML-Anwendungsschemata selbst entsprechen jedoch dem General Feature Model der Norm ISO 19109 und das wiederum basiert auf dem OO-Paradigma, wie bereits in Abschnitt 2.4.2 beschrieben wurde.

6.2 MOF 2.0 Query/View/Transformation

Der Standard *MOF 2.0 Query/View/Transformation* (QVT) [OMG, 2008a] ist eine Entwicklung der OMG und spezifiziert eine Sprache für die Beschreibung von Modell-zu-Modell-Transformationen. Dem QVT-Standard liegt das OO-Paradigma zugrunde. Insgesamt beschreibt der Standard drei verschiedene Sprachen: QVT-Core, QVT-Relations und QVT-Operational. QVT findet Anwendung in der Model-Driven Architecture (→ Abschnitt 2.1.4), d. h. QVT ermöglicht die Transformation von PIM → PIM bzw. von PIM → PSM. Voraussetzung ist jedoch, dass die Metamodelle der zu transformierenden Modelle mittels MOF beschrieben sein müssen, wie dies z. B. bei UML inklusive echter UML-Profile der Fall ist. QVT eignet sich damit für die Transformation von UML-Modellen. Für Modell-zu-Text-Transformationen stellt die OMG einen eigenen Standard, die *MOF Model to Text Transformation Language* [OMG, 2008b], zur Verfügung.

6.3 Rule Interchange Format (RIF)

Das *Rule Interchange Format* (RIF) [W3C, 2010a] ist ein Format, das derzeit vom W3C im Rahmen des Semantic Web entwickelt wird. Mit RIF können Regeln zwischen verschiedenen Systemen, welche Regeln verarbeiten, ausgetauscht werden. RIF setzt sich aus drei Dialekten zusammen, RIF Core, RIF Basic Logic Dialect (BLD) und RIF Production Rule Dialect (PRD). Das RIF-Format verwendet XML-Syntax, entspricht aber dem RDF-Paradigma (→ Abschnitt 2.1.4), weshalb es für die Transformation von UML-Modellen, welche dem OO-Paradigma entsprechen, ungeeignet ist.

Im Rahmen der Entwicklung einer Technical Guidance für einen INSPIRE Transformation Network Service [JRC, 2010b] wird diese Sprache jedoch momentan als mögliche geeignete Sprache für die modellbasierte Transformation von Geodaten betrachtet, weshalb sie auch im Rahmen dieser Studie aufgeführt wird.

6.4 FME

Die Software *FME* der kanadischen Firma Safe Software [Safe, 2010] stellt eine geeignete Plattform dar, um Transformationen zwischen verschiedenen Geodatenformaten zu realisieren. Die Benutzeroberfläche von FME bietet die Möglichkeit, komfortabel Mappings zwischen Daten zu definieren und entsprechende Transformationen durchzuführen.

FME arbeitet dabei sowohl auf Formatebene wie auch auf Modellebene. Auf Formatebene erfolgt das Einlesen und Schreiben der Daten. Auf Modellebene können mit FME konzeptuelle Datenmodelle gelesen und geschrieben werden, wie z. B. INTERLIS-Modelle [Eisenhut, 2006] oder in XMI serialisierte UML-Modelle [Schilcher et. al., 2009]. Darüber hinaus wird auch die Transformation auf semantischer Ebene definiert.

FME arbeitet jedoch nach dem Relationalen Paradigma (→ Abschnitt 2.1.4), wohingegen die in dieser Studie beschriebenen konzeptuellen Modelle gemäß dem OO-Paradigma erstellt wurden. Daraus ergeben sich Schwierigkeiten beim Einlesen der Modelle in FME. So muss beim Einlesen der INTERLIS-Modelle und auch der in XMI serialisierten UML-Modelle eine Abbildung vom OO-Paradigma auf das Relationale Paradigma durchgeführt werden. Auch bei der Definition der Transformationsregeln muss das Relationale Paradigma berücksichtigt werden.

FME wird im Projekt mdWFS (→ Abschnitt 3.1.1) eingesetzt. Im Rahmen des Projekts wird derzeit eine Erweiterung für FME entwickelt, womit es möglich sein wird, UML-Modelle aus den Anwendungsfällen in FME einzulesen und eine semantische Transformation auf diesen UML-Modellen durchzuführen. Die Transformation erfolgt mittels der im Projekt entwickelten Transformationssprache UMLT [Schilcher et. al., 2008], welche ebenfalls auf die internen Strukturen von FME abgebildet wird.

7 Fazit

Abschließend werden in diesem Kapitel die wichtigsten Ergebnisse und Erkenntnisse aus den vorhergehenden Kapiteln zusammengefasst und mögliche Lösungsansätze für die in Kapitel 4 beschriebenen Probleme vorgestellt.

7.1 Zusammenfassung

In Kapitel 2 wurde beschrieben, dass mithilfe von Modellen die reale Welt in abstrahierter Form wiedergegeben werden kann. Modelle bieten eine Sicht auf die reale Welt, die alles enthält, was für das jeweilige Anwendungsgebiet von Interesse ist. Werden diese Modelle mit einer formalen Modellierungssprache beschrieben, dann spricht man von einem konzeptuellen Schema. Damit das Modell richtig verstanden werden kann, ist Semantik notwendig. Die Semantik von Modellierungssprachen legt fest, was mit einem Modell gemeint ist.

Der Vorteil von Modellen besteht darin, dass Modelle unabhängig von spezifischen Systemen und Formaten erstellt werden können. Modelle enthalten keine Informationen zu deren physischen Implementierung, sondern es wird allein der gewünschte Realweltausschnitt beschrieben. Es ist somit eine explizite Trennung von Fachlogik und Implementierungstechnologie gewährleistet.

Darüber hinaus wurde der Begriff Profile vorgestellt, welcher für die Studie von zentraler Bedeutung ist. Profile stellen entweder eine Einschränkung oder eine Erweiterung einer Spezifikation dar. Bei einer Einschränkung besteht das Profil nur aus einer Teilmenge der von einer Spezifikation angebotenen Konstrukte. Bei einer Erweiterung dagegen enthält das Profil Konstrukte, die in der Spezifikation selbst nicht existieren, jedoch gemäß den Vorgaben der Spezifikation bezüglich Erweiterungen erstellt werden dürfen. Ein UML-Profil gestattet nur Einschränkungen der Modellierungssprache UML. Erweiterungen sind gemäß der UML-Definition von Profilen nicht erlaubt.

Des Weiteren wurde in der Studie anhand von Anwendungsfällen beschrieben, dass es notwendig sein kann, Modelle in andere Modelle zu überführen. Hierfür wird Modelltransformation benötigt. Mittels Modelltransformation können konzeptuelle Schemata aufeinander abgebildet werden, indem ein oder mehrere Quellschemata in ein oder mehrere Zielschemata überführt werden. Modelltransformation wird immer dann eingesetzt, wenn sich die Modelle der Quellsysteme von den Modellen der Zielsysteme unterscheiden.

Modelltransformation kann darüber hinaus auch eingesetzt werden, um Geodaten zu transformieren, so dass diese in ihrer Semantik dem Modell des Zielsystems entsprechen. Dies wird in der Studie als modellbasierte Transformation von Geodaten bezeichnet. So ist nicht zuletzt durch die INSPIRE-Richtlinie die Notwendigkeit entstanden, nationale Geodaten in das INSPIRE-Datenmodell überführen zu können. Es bieten sich jedoch weit mehr

Einsatzbereiche für die modellbasierte Transformation von Geodaten an, wie beispielsweise die Transformation in Fachmodelle. Einer der Anwendungsfälle beschäftigt sich dementsprechend damit, wie Geodaten, die gemäß dem ATKIS Basis-DLM vorliegen, in das Modell für Geobasisdaten einer Fachanwendung aus Baden-Württemberg überführt werden können.

Anschließend wurden die Themen Modellierung und Modelltransformation im Kontext der relevanten Standards und Normen der OMG, der ISO und von INSPIRE behandelt. Darauf aufbauend konnte dann die IST-Situation bezüglich der Datenmodelle und Modellierungssprachen Deutschlands (AAA-Modell), der Schweiz (INTERLIS) und der Europäischen Union (INSPIRE) untersucht und miteinander verglichen werden. Es konnte gezeigt werden, dass die Modellierungssprachen bzw. UML-Profile eine Reihe von Unterschieden aufweisen, welche sich nachteilig auf die modellbasierte Transformation von Geodaten auswirken. Einzelne konkrete Problemfälle wurden vorgestellt. Hierzu zählen z. B. das Vorhandensein unterschiedlicher UML-Versionen, UML-Abänderungen, fehlende Informationen im UML-Modell bezüglich der Kodierung des Modells durch Geodaten oder auch ein fehlendes konzeptuelles Datenmodell.

Auch für Transformationssprachen wurden eine Reihe von allgemeinen Anforderungen definiert, die zur Lösung der beschriebenen Probleme beitragen können und erfüllt sein müssen, damit die Transformation und Kodierung basierend auf maschinenlesbaren und maschineninterpretierbaren Modellen durchgeführt werden kann. Zudem wurden einzelne ausgewählte Ansätze kurz vorgestellt, die derzeit bei der Transformation von Geodaten eine Rolle spielen.

7.2 Lösungsansätze

7.2.1 Lösungsansätze für die Problematik unterschiedlicher UML-Profile

Gemeinsames Kern-UML-Profil Basierend auf den Ausführungen in den vorhergehenden Kapiteln kann generell festgehalten werden, dass eine Transformation zwischen unterschiedlichen UML-Profilen korrekt durchführbar ist, wenn es sich bei den betroffenen UML-Profilen um echte UML-Profile im Sinne der UML-Spezifikation handelt.

Bei den in Kapitel 3.2 vorgestellten „UML-Profilen“ ist dies jedoch nicht der Fall, da beispielsweise die Stereotypen «codeList» und «union» keine Spezialisierung vorhandener UML-Elemente darstellen. Wie Abbildung 3.6 zeigt, wird auf diese Weise kein UML-Profil gemäß der UML-Definition von Profilen, sondern eine neue Modellierungssprache definiert. Diese Modellierungssprache besitzt zwar die Syntax von UML, nicht jedoch deren Semantik. Eine Transformation sollte zwar auch in diesem Fall durchführbar sein, es ist jedoch davon auszugehen, dass hiermit gewisse Schwierigkeiten verbunden sind.

Ein Lösungsansatz besteht deshalb darin, in allen Projekten ein gemeinsames UML-Profil zu verwenden. Hierfür müsste aus allen vorhandenen UML-Profilen ein so genanntes gemeinsames *Kern-UML-Profil* erstellt werden, indem aus den existierenden UML-Profilen Abbildungen auf das Kern-UML-Profil definiert werden. Dieses Kern-UML-Profil darf nur gemeinsame Elemente aus den verschiedenen UML-Versionen sowie aus den verschie-

denen UML-Profilen enthalten. Dies ist deshalb wichtig, da ein umfassendes UML-Profil, welches auch die nicht gemeinsamen Elemente enthalten würde, die breite Nutzung der Geodaten außerhalb des GI-Bereichs behindern würde. Dieses Kern-UML-Profil stellt fortan die Basis für eine interoperable modellbasierte Transformation dar und kann gemäß der beiden nachfolgend beschriebenen Varianten eingesetzt werden. Das Kern-UML-Profil muss ein echtes UML-Profil darstellen.

Variante 1: Vielfalt an Modellierungssprachen akzeptieren Das Vorhandensein unterschiedlicher Modellierungssprachen könnte bedeuten, dass ganz einfach individuell auf bestimmte Anwendungsbereiche ausgerichtete Modellierungssprachen und damit unterschiedliche Metamodelle benötigt werden, sich bestimmte Anwendungen also nicht in bestimmte Sprachen zwängen lassen. Man könnte somit sagen, die heutige Vielfalt an Modellierungssprachen beweist, dass die Sprachen benötigt werden. Gleichzeitig könnte dies aber auch einen Hinweis auf Mangelkonzepte in heutigen Modellierungssprachen liefern. Beispielsweise wäre denkbar, dass flexibel spezifizierbare Modellierungssprachen wie DSLs einen vorteilhaften Beitrag hierzu leisten könnten.

Dennoch wird, damit zwischen den unterschiedlichen Modellierungssprachen eine Transformation durchgeführt werden kann, ein gemeinsames Kern-UML-Profil benötigt. Das Kern-UML-Profil kann in diesem Fall als Zwischenstufe einer mehrstufigen semantischen Transformation eingesetzt werden.

Die Transformation würde dabei folgendermaßen ablaufen: Das Quellschema mit „UML-Profil“ x wird mittels eines 1:1-Transformationswerkzeuges in ein Quellschema überführt, welches auf dem Kern-UML-Profil basiert. Dieses Quellschema kann nun mittels semantischer Transformation in ein Zielschema überführt werden, das ebenfalls auf dem Kern-UML-Profil basiert. Zum Schluss wird dieses Zielschema mittels eines 1:1-Transformationswerkzeuges in ein Zielschema mit dem dafür vorgesehenen „UML-Profil“ y überführt.

Der Vorteil bei diesem Ansatz liegt darin, dass die semantische Transformation innerhalb eines einheitlichen Kern-UML-Profils definiert werden kann. Die 1:1-Transformationswerkzeuge müssen nur einmal für jedes „UML-Profil“ definiert und implementiert werden und können anschließend automatisch ein beliebiges konzeptuelles Schema, das auf einem bestimmten „UML-Profil“ basiert, in ein konzeptuelles Schema gemäß dem Kern-UML-Profil umwandeln. So müsste beispielsweise für alle INSPIRE-Modelle nur ein 1:1-Transformationswerkzeug entwickelt werden und genauso für alle mit INTERLIS modellierten Datenmodelle.

Variante 2: Vielfalt eliminieren Bei diesem Ansatz würden alle Projekte nur noch das gemeinsame Kern-UML-Profil einsetzen. Dies bedeutet, dass eine Abbildung auf das Kern-UML-Profil nur für eine bestimmte Übergangsphase durchzuführen ist, solange bis das Kern-UML-Profil das einzig verwendete UML-Profil darstellt.

Empfehlung für neue Modelle Es dürfen keine neuen UML-Profile definiert werden, da dies zu den in dieser Studie beschriebenen Nachteilen bei der Transformation der Datenmodelle führt. Stattdessen sollte ein Kern-UML-Profil verwendet werden oder es sollte eine freiwillige Beschränkung auf den Umfang des Kern-UML-Profils erfolgen.

Empfehlung für existierende Modelle Bei Revisionen sollte unbedingt auf den Umfang des Kern-UML-Profils hingearbeitet werden, damit in Zukunft nach und nach die in dieser Studie beschriebenen Nachteile beseitigt werden können.

7.2.2 Lösungsansätze für andere festgestellte Probleme

Transformationssprachen untersuchen Darüber hinaus bedarf es einer weiteren Untersuchung vorhandener Transformationssprachen. Wie Kapitel 6 aufzeigt, existiert eine Vielzahl von Transformationssprachen. Jedoch sind nicht alle diese Sprachen gleich gut geeignet für die Transformation von Datenmodellen. Auch die Bereiche Datenbanken, Data Warehouses und ETL sollten in die Untersuchungen miteinbezogen werden. Diese Bereiche beschäftigen sich schon seit den 1970er Jahren mit der Integration heterogener Daten und Datenmodelle sowie dem Schema Mapping.

Auch müsste untersucht werden, welche Geodaten-spezifischen Funktionalitäten eine Transformationssprache unterstützen sollte, so fehlen beispielsweise bei allen Sprachen geometrische Operationen.

Nachmodellierung nur mit Fachwissen Beim Anwendungsfall aus Abschnitt 3.1.2 wird eine Transformation vaus dem AAA-Modell in das Modell des MLR-Geodatenservers durchgeführt. Für das Zielmodell steht jedoch kein konzeptuelles Modell zur Verfügung, weshalb das Zielmodell mit UML nachmodelliert werden musste, um dennoch eine Transformation auf konzeptueller Ebene durchführen zu können. Daraus ergaben sich eine Reihe von Problemen, die in Abschnitt 4.3.1 beschrieben wurden und zu dem Schluss führen, dass die Nachmodellierung konzeptueller Modelle in einem anderen Paradigma als dem, in welchem die Daten vorliegen, nur dann eindeutig und korrekt durchführbar ist, wenn der Modellierer über entsprechende Kenntnisse des Anwendungsbereichs verfügt.

Revision von Modellierungssprachen Wie in Abschnitt 4.1.2 beschrieben wurde, liegen das AAA-Modell sowie das TLM in UML-Version 1.4.2 vor, das INSPIRE-Modell in UML-Version 2.1. Zudem befindet sich die Norm ISO 19103 derzeit in Revision (→ Abschnitt 3.2.1), wodurch in Zukunft Modelle, die auf dieser Norm basieren, auch in UML-Version 2.2 vorliegen könnten.

Diesbezüglich wurde bereits in Abschnitt 2.4.3 die Frage gestellt, wie Rückwärtskompatibilität gewährleistet werden kann, sollte die revidierte Norm einmal verabschiedet sein und sich damit die Basis für die Definition von Anwendungsschemata ändern. Es kann nämlich nicht davon ausgegangen werden, dass bereits definierte Anwendungsschemata sofort bzw. überhaupt in die UML-Version 2.2 überführt werden. Es müssen Regeln und Richtlinien festgelegt werden, wie mit dieser Versionsänderung umgegangen wird.

Dies gilt auch, wenn an bestehenden Profilen eine Revision durchgeführt wird, also insbesondere für die derzeitige Revision der Norm ISO 19103 (→ Abschnitt 2.4.3) und eine mögliche Revision von INTERLIS.

A Projektvorschlag

Dieser Anhang enthält einen Auszug aus dem Projektvorschlag, der am 09. September 2009 den Vermessungsverwaltungen Baden-Württembergs, Bayerns, Österreichs und der Schweiz übergeben wurde. Der Auszug gibt die für diese Studie geplanten Inhalte und Ziele wieder und stellt die Projektorganisation für die Durchführung der Studie dar.

Definition einer Studie zur Verwendung der Mittel aus dem Projekt „Bodensee-Geodatenpool“

Prof. Matthäus Schilcher (TUM),
Dr. Andreas Donaubauer (ETH), Claude Eisenhut (Eisenhut Informatik),
Dieter Heß (LGL BW), Dr. Andreas Illert (BKG),
DI Stefan Klotz (BEV), Tatjana Kutzner (TUM),
Dr. Markus Seifert (LVG BY), Dr. Peter Staub (swisstopo)

09.09.2009

Inhalte der Studie

1. Begriffsbestimmung, Problemstellung und IST-Situation bzgl. unterschiedlicher CSL-Profile bei der Modellierung von Geodaten
   - Begriffsbestimmung: Was ist ein Modell, Datenmodell, Interface-Modell, CSL, UML-Profil, DSL (Domain Specific Language), ...
   - Begriffsbestimmungen gemäß INSPIRE, INTERLIS und ISO einordnen
   - Bestandsaufnahme der IST-Situation:
     Welche konzeptionellen Schemasprachen (CSL) bzw. UML-Profile werden bei INSPIRE, in Deutschland und der Schweiz zur Modellierung von Geodaten verwendet?
     Welche Unterschiede existieren zwischen den UML-Profilen?
     Wie werden CSL verwendet (Modellierungsparadigmen, wird Datenmodell beschrieben oder Transferformat, Produktionsmodell, Prozessmodell, Sprachparadigmen, etc.)?
   - Herausarbeitung der Problemstellung anhand von Anwendungsfällen aus dem Geodatenpool Bodensee + Problemfälle anhand existierender Modelle aufzeigen
   - Aufzeigen der dadurch entstehenden Probleme und Schaffung eines Bewusstseins für diese Thematik

Fachgebiet Geoinformationssysteme Technische Universität München

Inhalte der Studie

2. Anforderungen an eine CSL-Profil-übergreifende Transformation im Kontext von INSPIRE und Bodensee Geodatenpool bzw. Geobasisdaten der Länder
   - Keine Nachmodellierung in Interlis mehr durch maschinenlesbares und interpretierbares Modell
   - Berücksichtigung der Komplexität der unterschiedlichen CSL-Profile
   - An unterschiedliche Modellversionen anpassbare Transformationsbeschreibung
   - Fehlerbehandlung während der Transformation
   - Berücksichtigung der Dauer einer Transformation
   - Einsetzbarkeit großer Modelle und komplexer Transformationen
   - Komplexe Transformationen verständlich machen
   - Entwicklung einer Tranformationsdefinition im Team
   - Nutzen jenseits von INSPIRE für Transformation zwischen originären Datenmodellen von Geobasisdaten und den Fachanwendungen zugrunde liegenden Datenmodellen von Geobasisdaten (anhand Anwendungsfällen von LGL BW, siehe Folien 8-11, und ggf. swisstopo)
   - Evtl. Nutzeranforderungen ermitteln (z.B. Bidirektionalität)

Inhalte der Studie

3. Bestandsaufnahme von Lösungsansätzen für eine CSL-Profil-übergreifende Transformation aus der Allgemeinen Informatik
   - Modell-zu-Modell-Abbildungen in der Informatik
   - Evaluierung von Mapping-Sprachen und Vergleich zu UMLT
   - Relevanz der Lösungsansätze für mdWFS-Projekt
   - Untersuchung der Bereiche Datenbank, Data Warehouse, XML, OMG, XSLT (SAX-Fähigkeit bei großen Datenmengen)
4. Bewertung und Empfehlungen
   - Mögliche Anregungen für den Umgang mit unterschiedlichen UML-Profilen
   - Aufzeigen grundsätzlicher Lösungsvorschläge zur praktischen Anwendung der Transformation für die am Bodensee-Geodatenpool beteiligten Vermessungsverwaltungen
   - Bewertung einer CSL-Profil-übergreifenden Transformation, u.a. wie können (zu) hohe Freiheitsgrade bei Metamodell-Transformationen gelöst werden?
   - Mögliche Anregungen für Anpassung der verwendeten Mapping-Sprache und deren Einbringung in Standardisierungsgremien

Ziele der Studie

- Leitgedanke der Studie
  „Das Modell muss maschinenlesbar und interpretierbar sein. Andernfalls kann das Potenzial des modellbasierten Ansatzes nicht ausgeschöpft werden.“
- Ein Bewusstsein schaffen für die Problematik verschiedener UML-Profile bezüglich semantischer Interoperabilität.
- Einen Leitfaden für die Auftraggeber und Ersteller von Datenmodellen in den folgenden Punkten liefern:
  - Für Definition von UML-Profilen und Datenmodellen
    - Ergebnisse der Studie können die Definition neuer Profile dahingehend beeinflussen, dass Abänderungen von UML nur dann vorgenommen werden, wenn die Maschineninterpretierbarkeit gewährleistet ist.
    - Ergebnisse der Studie können als Checkliste dienen, um Datenmodelle so zu erstellen, dass unterschiedliche Modellierungen zwischen Quell- und Zielmodell, wie z.B. die Stereotype <<Feature>> und <<featureType>>, von vornherein ausgeschlossen werden können.

Ziele der Studie

- Für Definition von Kodierungsregeln
  - Ergebnisse der Studie können momentane Problematik verdeutlichen. Die Kodierungsregeln müssen eindeutig und maschineninterpretierbar sein, damit der modellbasierte Ansatz ohne Probleme funktionieren kann.
- Für Standardisierung im Bereich Datenmodellierung und Modelltransformation
  - Ergebnisse der Studie Geodatenpool sind von großem Nutzen für die „Technical Guidance for INSPIRE Transformation Service“ wie auch für die Erstellung von Annex II und III und zukünftige Revisionen von Annex I Datenspezifikationen, sowie generell bei der Neuaufstellung und für Revisionen von Datenmodellen der Bodenseeanrainerstaaten (sowohl für Geobasis- als auch für Geofachdaten).

- Synergien zu JRC-Studie für Transformationsdienste und Projekt mdWFS (z.B. UMLT verbessern)

Fachgebiet Geoinformationssysteme Technische Universität München

Organisation

- Projektleitung
  - TUM: Herr Prof. Schilcher
- Projektmitarbeiter
  - TUM: Frau Kutzner
  - Eisenhut Informatik: Herr C. Eisenhut
- Mitwirkende
  - swisstopo: Herr Dr. Staub
  - BEV: Herr DI Klotz
  - LGL BW: Herr Heß
  - LVG BY: Herr Dr. Seifert
- Berater
  - BKG: Herr Dr. Illert
  - ETH: Herr Dr. Donaubauer

Literaturverzeichnis

[AdV, 2008] ARBEITSGEMEINSCHAFT DER VERMESSUNGSVERWALTUNGEN DER LÄNDER DER BUNDESREPUBLIK DEUTSCHLAND (ADV) (Hrsg.): *Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesen (GeoInfoDok)*. 2008

[AdV, 2010] ARBEITSGEMEINSCHAFT DER VERMESSUNGSVERWALTUNGEN DER LÄNDER DER BUNDESREPUBLIK DEUTSCHLAND (ADV) (Hrsg.): *Aktuelle Dokumente der GeoInfoDok*. `http://www.adv-online.de/icc/extdeu/broker.jsp?uCon=68470b36-de06-8a01-e1f3-351ec0023010&uBasVariantCon=11111111-1111-1111-1111-111111111111`, Abruf: 25. 07. 2010

[Bill, 1999] BILL, R.: *Grundlagen der Geo-Informationssysteme, Band 2 - Analysen, Anwendungen und neue Entwicklungen*. Wichmann Verlag, 1999

[Born et al., 2004] BORN, M. ; HOLZ, E. ; KATH, O.: *Softwareentwicklung mit UML 2*. Addison-Wesley, 2004

[Chen, 1976] CHEN, P. P.-S.: The Entity Relationship model: Toward a unified view of data. In: *ACM Transactions on Database Systems*, 1976, S. 9–36

[Codd, 1970] CODD, E. F.: A Relational Model of Data for Large Shared Data Banks. In: *Communications of the ACM 13*, 1970, S. 377–387

[CSIRO, 2010] COMMONWEALTH SCIENTIFIC AND INDUSTRIAL RESEARCH ORGANISATION (Hrsg.): *Overview of some relevant standards from ISO/TC 211*. `https://www.seegrid.csiro.au/twiki/bin/view/AppSchemas/IsoTc211Standards#General_Feature_Model`, Abruf: 27. 05. 2010

[Czarnecki et al., 2006] CZARNECKI, K. ; HELSEN, S.: Feature-based survey of model transformation approaches. In: *IBM Systems Journal* 45 (2006), Nr. 3, S. 621–645

[Devillers et al., 2006] DEVILLERS, R. ; JEANSOULIN, R.: Spatial Data Quality: Concepts. In: *Fundamentals of Spatial Data Quality*. ISTE Ltd., 2006, S. 31–42

[Donaubauer et al., 2010] DONAUBAUER, A. ; KUTZNER, T. ; STRAUB, F.: Accuracy-aware Web-based Geoprocessing Chains. In: TATE , FISHER (Hrsg.): *Accuracy 2010 - Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, 2010

[Duden, 2006] DUDENREDAKTION (Hrsg.): *Duden - Deutsches Universalwörterbuch*. Bibliographisches Institut, 2006

[Eisenhut, 2006] EISENHUT INFORMATIK (Hrsg.): *ili2fme - INTERLIS-plugin for FME*. `http://www.eisenhutinformatik.ch/interlis/ili2fme/`, Abruf: 01. 09. 2010

[EP, 2007] EUROPÄISCHES PARLAMENT UND EUROPÄISCHER RAT (Hrsg.): *Richtlinie 2007/2/EG des Europäischen Parlaments und des Rates vom 14. März 2007 zur Schaffung einer Geodateninfrastruktur in der Europäischen Gemeinschaft (INSPIRE). Amtsblatt der Europäischen Union Nr. 108/1 vom 25.4.2007*. 2007

[Falkenberg, 1977] FALKENBERG, E.: Concepts for the Coexistence approach to Data Base Management. In: MORLET, RIBENS (Hrsg.): *International Computing Symposium*, 1977 (IFIP), S. 377–393

[Falkenberg et al., 1979] BREUTMANN, B. ; FALKENBERG, E. ; MAUER, R.: CSL: A Language for Defining Conceptual Schemas. In: BRACCI, NIJSSEN (Hrsg.): *Data Base Architecture*, 1979 (IFIP), S. 337–356

[Fowler, 2000] FOWLER, M.: *Refactoring. Wie Sie das Design vorhandener Software verbessern*. Addison-Wesley, 2000

[Gruber, 1995] GRUBER, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: *International Journal Human-Computer Studies* 43 (1995), Nr. 5-6, S. 907–928

[Hesse et al., 2008] HESSE, W. ; MAYR, H. C.: Modellierung in der Softwaretechnik: eine Bestandsaufnahme. In: *Informatik-Spektrum* 31 (2008), Nr. 5, S. 377–393

[INTERLIS, 2008] *INTERLIS 2 – Metamodell*. `http://www.interlis.ch/interlis2/metamodele_d.php`, Abruf: 02. 09. 2010

[INTERLIS, 2010] *INTERLIS*. `http://www.interlis.ch/`, Abruf: 20. 04. 2010

[ISO, 1998] *Information technology – Framework and taxonomy of International Standardized Profiles – Part 1: General principles and documentation framework*. 1998

[ISO, 2002a] Norm ISO 19101:2002 2002. *Geographic information – Reference model*

[ISO, 2002b] Norm ISO 19107:2002 2002. *Geographic information – Reference model*

[ISO, 2004] Norm ISO 19106:2004 2004. *Geographic information – Profiles*

[ISO, 2005a] Norm ISO 19103:2005 2005. *Geographic information – Conceptual schema language*

[ISO, 2005b] Norm ISO 19109:2005 2005. *Geographic information – Rules for application schema*

[ISO, 2005c] Norm ISO 19501:2005 2005. *Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2*

[ISO, 2007a] Norm ISO 19136:2007 2007. *Geographic information – Geography Markup Language (GML)*

[ISO, 2007b] Norm ISO 19139:2007 2007. *Geographic information – Metadata – XML schema implementation*

[JRC, 2008] DRAFTING TEAM DATA SPECIFICATIONS: *Methodology for the development of data specifications*. 2008

[JRC, 2009a] DRAFTING TEAM DATA SPECIFICATIONS: *INSPIRE Generic Conceptual Model*. 2009

[JRC, 2009b] DRAFTING TEAM DATA SPECIFICATIONS: *Guidelines for the encoding of spatial data*. 2009

[JRC, 2009d] INSPIRE THEMATIC WORKING GROUP ADDRESSES: *INSPIRE Data Specification on Addresses – Guidelines*. 2009

[JRC, 2010a] INSPIRE THEMATIC WORKING GROUP ADMINISTRATIVE UNITS: *INSPIRE Data Specification on Administrative units – Guidelines*. 2010

[JRC, 2010b] BEARE, M. ; HOWARD, M. ; PAYNE, S. ; WATSON, P.: *Development of Technical Guidance for the INSPIRE Transformation Network Service. State Of The Art Analysis*. 2010

[Kemper et al., 2009] KEMPER, A. ; EICKLER, A.: *Datenbanksysteme - Eine Einführung*. Oldenbourg Verlag, 2009

[KOGIS, 2010] *UML/INTERLIS-Editor*. `http://www.umleditor.org/`, Abruf: 22. 06. 2010

[Lehto, 2007] LEHTO, L.: *Real-time Content Transformations in a Web Service-based Delivery Architecture for Geographic Information*, Helsinki University of Technology, Diss., 2007

[Obrst, 2003] OBRST, L.: Ontologies for Semantically Interoperable Systems. In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, 2003, S. 366–369

[OGC, 2006] OPEN GEOSPATIAL CONSORTIUM INC. (Hrsg.): *OpenGIS Implementation Specification for Geographic Information - Simple feature access - Part 1: Common Architecture*. 2006

[OMG, 2006] OBJECT MANAGEMENT GROUP (Hrsg.): *Meta Object Facility (MOF)*. 2006

[OMG, 2007a] OBJECT MANAGEMENT GROUP (Hrsg.): *OMG Unified Modeling Language (OMG UML), Infrastructure*. 2007

[OMG, 2007b] OBJECT MANAGEMENT GROUP (Hrsg.): *OMG Unified Modeling Language (OMG UML), Superstructure*. 2007

[OMG, 2007c] OBJECT MANAGEMENT GROUP (Hrsg.): *MOF 2.0/XMI Mapping, Version 2.1.1*. 2007

[OMG, 2008a] OBJECT MANAGEMENT GROUP (Hrsg.): *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*. 2008

[OMG, 2008b] OBJECT MANAGEMENT GROUP (Hrsg.): *MOF Model to Text Transformation Language, v1.0*. 2008

[RTG, 2010] EISENHUT, C. ; ILLERT, A. ; KUTZNER, T. ; MÜLLER, M. ; SCHILCHER, M.: *Semantische Datenmodelltransformation im Kontext von INSPIRE*. `http://www.rtg.bv.tum.de/images/stories/downloads/aus-und_weiterbildung/fortbildungsseminare/2010/WS_SemDat/workshop_semtrans_ergebnissearbeitsgruppen.pdf`, Abruf: 27. 04. 2010

[Safe, 2010] SAFE SOFTWARE INC. (Hrsg.): *Homepage*. `http://www.safe.com/`, Abruf: 01. 08. 2010

[Schilcher et. al., 2008] SCHILCHER, M. ; DONAUBAUER, A. ; FICHTINGER, A. ; KUTZNER, T. ; CAROSIO, A. ; HENRICH, S. ; STAUB, P.: *mdWFS Endbericht Phase III*. 2008

[Schilcher et. al., 2009] KUTZNER, M. Schilcher A. Donaubauer T.: *mdWFS Endbericht Phase IV*. 2009

[Seifert, 2008] SEIFERT, M.: *Wissenschaftlicher Beitrag für den Aufbau einer Geodateninfrastruktur zur Lösung von Aufgaben des E-Government*, Eidgenössische Technische Hochschule Zürich, Diss., 2008

[Steel, 1975] STEEL, JR., T. B.: Data base standardization: a status report. In: *SIGMOD '75: Proceedings of the 1975 ACM SIGMOD international conference on Management of data*, ACM, 1975, S. 149–156

[Tantau, 2006] TANTAU, T.: *Syntax versus Semantik – Text und seine Bedeutung*. http://www.tcs.uni-luebeck.de/Lehre/2006-WS/Logik/Vorlesung:A_Syntax_versus_Semantik, Abruf: 21. 05. 2010

[W3C, 2007] WORLD WIDE WEB CONSORTIUM (Hrsg.): *XSL Transformations (XSLT)*. http://www.w3.org/TR/xslt20/, Abruf: 01. 08. 2010

[W3C, 2010a] WORLD WIDE WEB CONSORTIUM (Hrsg.): *RIF Overview*. http://www.w3.org/TR/rif-overview/, Abruf: 01. 08. 2010

[Wikipedia, 2010a] WIKIPEDIA, DIE FREIE ENZYKLOPÄDIE: *Modell (Begriffsklärung)*. http://de.wikipedia.org/w/index.php?title=Modell_(Begriffsklärung)&oldid=75630145, Abruf: 08. 07. 2010

[Wikipedia, 2010b] WIKIPEDIA, DIE FREIE ENZYKLOPÄDIE: *Elamische Sprache*. http://de.wikipedia.org/w/index.php?title=Elamische_Sprache&oldid=75858167, Abruf: 13. 07. 2010

[Wikipedia, 2010c] WIKIPEDIA, DIE FREIE ENZYKLOPÄDIE: *Domain-specific language*. http://en.wikipedia.org/w/index.php?title=Domain-specific_language&oldid=373224047, Abruf: 13. 07. 2010

ISBN 978-3-935049-74-0