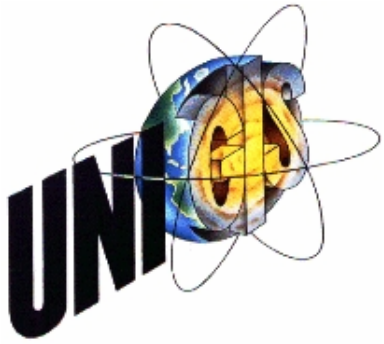

Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„Metamodell zur Konfiguration und Steuerung von Komponenten einer Geodaten-infrastruktur“

vorgelegt von

Dipl. Ing. FH Pascal Imoberdorf
u1217, UNIGIS MSc Jahrgang 2005

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Bern, 27. Juni 2007

Danksagung

Diese Master Thesis bildet den Abschluss des zweijährigen Universitätslehrgangs ”Geographical Information Science & Systems“ (UNIGIS MSc) am Zentrum für Geoinformatik (Z_GIS) der Paris Lodron-Universität in Salzburg.

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meines UNIGIS-Studiums und dem Schreiben dieser Master Thesis unterstützt haben.

Beim Team unter der Leitung von Prof. Dr. Josef Strobl bedanke ich mich für die vielseitigen und interessanten Einblicke in die GIS-Welt sowie für die stets prompte und konstruktive Unterstützung.

Ein ganz besonderer Dank gilt meinem Betreuer Dr. Martin Huber von der Condesys Consulting GmbH, der diese Master Thesis möglich gemacht hat. An zahlreichen Besprechungen hat er sein fachliches Wissen bereitwillig an mich weitergegeben und geduldig meine Fragen beantwortet. Persönlich bedanken möchte ich mich auch bei Tobias Henz und Christine Najar, mit denen ich spannende und fruchtbare Diskussionen über Metadaten und deren Modellierung führen konnte.

Zum Schluss gilt mein Dank meiner Familie und meiner lieben Frau Ditte. Sie haben mich in den vergangenen zweieinhalb Jahren verständnisvoll von vielen Pflichten entbehrt und waren mir immer eine wichtige Stütze und Motivationshilfe.

Eigenständigkeitserklärung

Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit die wörtlich oder sinngemäss übernommen wurden sind entsprechend gekennzeichnet.

Bern, Juni 2007

……

Pascal Imoberdorf

Kurzfassung

Eine Geodateninfrastruktur (GDI) ist ein komplexes System für den netzwerkübergreifenden Zugang auf räumliche Ressourcen. Die Integration der einzelnen Komponenten und Daten erfordert ein hohes Mass an fachlichem und technischem Know-how. In dieser Thesis wurde daher die Idee verfolgt, die Konfiguration und Steuerung der an einer GDI beteiligten Softwarekomponenten durch ein flexibles, wiederverwendbares Metadatenframework zu vereinfachen. Dabei galt es, soweit als möglich auf bestehende Ansätze und Standards aus der Informations- und Kommunikationstechnologie (ICT) zurückzugreifen. Auf Basis des Common Warehouse Metamodels (CWM) entstand auf diese Weise ein konzeptionelles Metamodell, welches als Grundlage für die Realisierung eines zentralen Repositories für GDI-Metadaten dient.

Stichwörter:

Geodateninfrastruktur, Metadatenframework, Metamodell, Common Warehouse Metamodel

Abstract

A spatial data infrastructure (SDI) is a complex system for the cross-network access to spatial resources. The integration of the individual components and data requires a high measure of professional skills and technical know-how. That is why in this thesis the idea was pursued to simplify the configuration and controlling of SDI software components by a flexible and reusable metadata framework. Thereby, it was valid to refer as much as possible to existing information and communication technology (ICT) approaches and standards. Based on the Common Warehouse Metamodel (CWM) a conceptual metamodel was thus developed, which serves as a basis for the realization of a central meta data repository.

Keywords:

Spatial Data Infrastructure, Metadata Framework, Metamodel, Common Warehouse Metamodel

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CASE	Computer-Aided Software Engineering
CGI	Common Gateway Interface
CIM	Computation Independent Model
CITE	Compliance & Interoperability Testing & Evaluation Initiative
CSDGM	Content Standard for Digital Geospatial Metadata
CSW	Catalog Service for Web
CWM	Common Warehouse Metamodel
DB	Datenbank / Database
DBMS	Database Management System
DCES	Dublin Core Metadata Element Set
DCMI	Dublin Core Metadata Initiative
DTD	Document Type Definition
DWH	Data Warehouse
ERM	Entity Relationship Model
ESDI	European Spatial Data Infrastructure
FE	OpenGIS® Filter Encoding Specification
FGDC	U.S. Federal Geographic Data Committee
FOSS4G	Free and Open Source Software for Geoinformatics
FPS	Feature Portrayal Service
GDI	Geodateninfrastruktur
GI	Geoinformation
GIS	Geographisches Informationssystem
GM03	Schweizer Metadatenmodell
GML	Geography Markup Language
GSDI	Global Spatial Data Infrastructure
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
INTERLIS	Inter-Landinformationssystem
ISO	International Organization for Standardization

IT	Informationstechnologie
ITC	Informations- und Kommunikationstechnologie
JDBC	Java Database Connectivity
KOGIS	Koordination, Geo-Information und Services (Bereich swisstopo)
MDA	Model Driven Architecture
MIME	Multipurpose Internet Mail Extensions
MME	Managed Meta Data Environment
MOF	Meta Object Facility
NGDI	Nationale Geodateninfrastruktur
OAI-PMH	Open Archives Initiative - Protocol for Metadata Harversting
OASIS	Organization for the Advancement of Structured Information Standards
OCL	Object Constraint Language
ODBC	Open Database Connectivity
OGC	Open Geospatial Consortium
OLAP	Online Analytical Processing
OMG	Object Management Group
OSF	OWS Service Framework
OWS	OpenGIS® Web Services
PEP	Policy Enforcement Point
PHP	Hypertext Preprocessor
PIM	Platform Independent Model
PSM	Platform Specific Model
RDBMS	Relational Database Management System
SDI	Spatial Data Infrastructure
SE	OpenGIS® Symbology Encoding Specification
SFS	OpenGIS® Simple Feature Specification for SQL
SLD	OpenGIS® Styled Layer Descriptors
SMS	Symbology Management System
SNV	Schweizerische Normen-Vereinigung
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SOGI	Schweizerische Organisation für Geo-Information
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TC	Technical Committee

UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WFS	Web Feature Service
WMS	Web Map Service
WSDL	Web Service Description Language
WSF	Web Service Framework
WWW	World Wide Web
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XPath	XML Path Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

Teil A – Einführung und Grundlagen

1. Einführung

Es wird allgemein geschätzt, dass etwa 80% aller politischen, wirtschaftlichen und privaten Entscheidungen einen direkten oder indirekten Raumbezug aufweisen. Immer mehr Laien kennen und nutzen die Möglichkeiten, um auf intuitive Art und Weise Adressen auf digitalen Karten zu lokalisieren und gleichzeitig nach umliegenden *Points of Interest*[1] zu suchen. Unzählige Anwendungen, die mit geographischen Daten und Diensten verknüpft sind, entwickeln sich in der heutigen Informationsgesellschaft zum alltäglichen Konsumgut (WILLIAMSON, 2003).

Informationen über Phänomene, die mit einer räumlichen Position verbunden sind, werden als Geoinformation (GI) bezeichnet. Im Wissenschaftsmagazin Nature[2] wurde im Jahr 2004 auf eine Studie des US Department of Labor Bezug genommen, welche die Geoinformationstechnologie - neben Nano- und Biotechnologie - als eines der drei wichtigsten Zukunftsgebiete identifiziert.

Gemäss AMSTEIN (2004) ist die Verfügbarkeit von Geoinformation eine wesentliche Voraussetzung für eine gut funktionierende direkte Demokratie und Geoinformation

- eine der bedeutendsten Ressourcen des 21. Jahrhunderts,
- die Grundlage für transparente und nachvollziehbare Entscheidungen in Politik, Wirtschaft und Gesellschaft,
- ein Wirtschaftsfaktor mit zunehmender Bedeutung und ein Wirtschaftsgut ersten Ranges.

Das Umsetzungskonzept zur Strategie für Geoinformation beim Bund (KOGIS, 2003) sowie NOGUERAS-ISO et al. (2005) definieren Geoinformation als wesentliches Element zur Entwicklung eines Landes. Eine Geodateninfrastruktur (GDI) ist vergleichbar mit klassischen Infrastrukturen wie etwa einem Transport-, Versorgungs- oder Telekommunikationsnetz. Geoinformation ist demnach ein Gut, welches bereitgestellt, verwaltet und gepflegt werden muss. Metadaten sind dabei eine zentrale Komponente einer Geodateninfrastruktur, welche den Zugang zu Geoinformation (KELLER, 2006a) und die Kopplung der technischen Komponenten ermöglichen (NOGUERAS-ISO et al., 2005).

[1] *deutsch: interessierender Ort (z.B. umliegende Restaurants, Tankstellen, Geldautomaten)*
[2] *Mapping Opportunities, Nature, Vol 427, S. 376-377, 22. Januar 2004*

1.1 Problemstellung und Motivation

Die Verfügbarkeit und Aktualität digitaler Geoinformation ist eine massgebende Grundlage für die Qualität und Transparenz von Entscheidungen in der Verwaltung, Politik und Wirtschaft. Als Voraussetzungen für eine stabile politische und wirtschaftliche Entwicklung stehen die Sicherung des Grundeigentums und die Planbarkeit der Versorgungsinfrastruktur im Vordergrund.

In den letzten Jahren sind zahlreiche Initiativen für den Aufbau von Geodateninfrastrukturen lanciert worden, welche den einfachen Austausch und die effiziente Nutzung von geographischen Informationen beabsichtigen. Bei öffentlichen und privaten Institutionen mit geringen finanziellen und personellen Möglichkeiten werden Vorhaben zum Aufbau von Geodateninfrastrukturen jedoch oft durch die hohen Beschaffungskosten kommerzieller Software und die begrenzten Mittel für Betrieb, Unterhalt und Nachführung verhindert.

Gleichzeitig sind leistungsfähige GIS-Komponenten, welche die Realisierung von umfangreichen Geodateninfrastrukturen ermöglichen, in zunehmender Anzahl frei verfügbar. Der Aufbau von Geodateninfrastrukturen auf der Basis von frei erhältlicher Software und offenen GIS-Standards wurde in der Schweiz von den Kantonen Solothurn und Neuenburg bereits erfolgreich praktiziert. Die Realisierung einer anforderungsgerechten Systemlösung erfordert ein hohes Mass an Know-how zur Konfiguration und spezifischen Anpassung bzw. Erweiterung der einzelnen Systemmodule.

Der Aufbau und Betrieb von Geodateninfrastrukturen bei begrenzten finanziellen und personellen Ressourcen bedarf folglich kostengünstiger Systemlösungen, welche sich insbesondere durch einen reduzierten Konfigurations- und Entwicklungsaufwand auszeichnen. Diese Arbeit hat zum Ziel, die Konfiguration und Steuerung der an einer GDI beteiligten Softwarekomponenten durch die Bereitstellung eines flexiblen, wiederverwendbaren Metadatenframeworks zu vereinfachen. Basierend auf dieser Grundlage soll der Einsatz von Geoinformation bei lokalen, regionalen, nationalen oder internationalen Organisationen oder Körperschaften mit reduziertem Aufwand möglich sein.

Der Begriff Metadaten ist im Kontext dieser Arbeit vorwiegend in technischer Hinsicht zu verstehen. Über das in der GIS-Gemeinschaft vorherrschende Metadaten-Verständnis zum Zweck der Datenbeschreibung und -katalogisierung für das Auffinden von Geo-Ressourcen hinaus liegt der Fokus dieser Arbeit darin, Metadaten für den physischen und logischen Datenzugriff sowie für die Konfiguration und Steuerung von Komponenten einer GDI zentral zu verwalten und bereit zu stellen.

1.2 Thesen

Basierend auf den oben geschilderten Problemstellungen resultieren folgende Thesen:

- Durch eine zentrale Verwaltung der Informationen über die Daten des Kernsystems kann der Konfigurationsaufwand von GDI-Komponenten und der Integrationsaufwand für Geodaten reduziert werden.
- Ein Metadaten-Repository eignet sich als zentrale Auskunftsstelle für diejenigen GDI-Komponenten, welche konfigurierbar sind.
- Den bestehenden Standards im Geoinformationsbereich (OGC und ISO) mangelt es an Möglichkeiten zur Beschreibung und Verwaltung von technischen Metadaten.
- Im GDI-Umfeld ist eine standardisierte Beschreibung von System-Metadaten analog zu den Ansätzen im Data Warehouse-Bereich[3] notwendig.

1.3 Zielsetzung

Das Ziel dieser Arbeit besteht in der Konzeption eines zentralen, wiederverwendbaren Metadatenframeworks, welches die zur Konfiguration und Steuerung der verschiedenen GDI-Komponenten notwendigen Informationen vorhält und in geeigneter Form bereitstellen kann. Durch eine zentrale Verwaltung der Informationen über die Daten des Kernsystems soll der Konfigurationsaufwand von GDI-Komponenten und der Integrationsaufwand für Geodaten reduziert werden. Die Datensammlungen des Kernsystems werden sowohl für menschliche Benutzer als auch für die automatisierte Auswertung hinsichtlich des physischen Datenzugriffs und der graphischen Darstellung beschrieben. Dabei sind allgemeine System-Metadaten zu identifizieren, die zur Beschreibung von Daten und zur Konfiguration von Diensten benötigt werden. Der physische Zugriff wird generisch implementiert und dem Endbenutzer in einer logischen Datensicht präsentiert. Die Integration neuer Datenbestände soll ohne zusätzliche Programmierung, sondern allein durch Konfiguration möglich sein.

Das in dieser Arbeit beschriebene Metadatenframework zur Steuerung von Komponenten und Diensten einer Geodateninfrastruktur ist hauptsächlich auf operative Bedürfnisse beschränkt. Die Arbeit fokussiert sich vorwiegend auf den Bereich der technischen Metadaten, während semantische Metadaten (z.B. ISO 19115) eine untergeordnete Rolle spielen. Das Vorgehen soll sich dabei besonders an bestehenden Ansätzen aus der allgemeinen Informationstechnologie (IT) orientieren.

[3] *vgl. Universal Meta Data Model (Kap. 2.7) und Common Warehouse Metamodel (Kap. 2.8)*

Die Frontend[4]-Steuerung kann weitgehend ausgeblendet werden, da sich Geo-Webdienste über ihre *Capabilities* selber beschreiben. Damit können die Benutzeroberflächen von Clients, sofern sie die entsprechenden Dienst-Schnittstellen unterstützen, dynamisch gesteuert werden. Der Fokus dieser Arbeit liegt daher auf der Daten- und Middleware[5]-Ebene. Dienste und serverseitige Applikationen sollen ihre Konfigurations-Informationen für den Zugriff auf Geodaten in einem zentralen Repository abholen.

1.3.1 Teilziele

- Analyse von Anforderungen an Metadaten zur Konfiguration und Steuerung von GDI-Komponenten anhand ausgewählter Softwaretools
- Untersuchung von bestehenden IT-Standards hinsichtlich deren Eignung für eine Standardisierung von System-Metadaten im GDI-Umfeld
- Konzeptioneller Entwurf eines Metadatenframeworks für den generischen Datenzugriff zur modulübergreifenden Applikationssteuerung in einem GDI-Framework

Das Metadatenframework wird in UML auf konzeptioneller Ebene modelliert. Es soll den Komponenten einer GDI (vgl. Kap. 2.2) die notwendigen Konfigurationsinformationen (u.a. Datenstruktur, Datenquellen, graphische Darstellung) in einheitlicher Form zur Verfügung stellen. Die Konfiguration einzelner GDI-Komponenten wird hierzu beispielhaft anhand ausgewählter Open Source-Applikationen (vgl. Kap. 3.6 und Anhang) untersucht.

1.3.2 Thematische Abgrenzung

Diese Arbeit wurde im Rahmen einer Studienarbeit bei Dr. Martin Huber von der Condesys Consulting GmbH erstellt. Sie weist einen indirekten Bezug zur Master Thesis von Rolf Mühlemann (u1219) auf, welche sich schwerpunktmässig mit Sicherheitsaspekten (Authentifizierung, Autorisierung und Transportsicherheit) im GDI-Umfeld befasst. Diese Perspektive wird deshalb in dieser Arbeit nicht beleuchtet.

Im Weiteren seien folgende Nichtziele genannt:

- wissenschaftliche Auseinandersetzung mit organisatorischen, institutionellen und wirtschaftlichen Rahmenbedingungen im GDI-Umfeld
- systematische Evaluation und Bewertung der in dieser Arbeit zwecks Anforderungsanalyse verwendeten GDI-Softwarekomponenten
- Implementierung von Schnittstellen für den ad hoc Zugriff auf ein Metadaten-Repository
- Implementierung von Skripten zur dynamischen Konfiguration von GDI-Komponenten

[4] *Computerprogramm zur interaktiven Eingabe sowie Anzeige von Daten (z.B. WebGIS- oder Desktop-Client)*
[5] *Programme, die den Datenverkehr zwischen der Serverplattform und den Client-Anwendungen ermöglichen*

1.4 Lösungsansatz

Soweit als möglich sollen bestehende IT-Methoden und -Konzepte berücksichtigt werden. GIS-spezifische Lösungen stellen im Idealfall eine Spezialisierung bzw. Erweiterung von bereits existierenden IT-Grundlagen dar. Wird diesem Grundsatz zu wenig entsprochen, dann ist die GIS-IT-Integration zur durchgängigen Abwicklung von Geschäftsprozessen oft aufwendig und mühsam. Die bisher fehlende Unterstützung von WSDL und SOAP im Rahmen der OpenGIS® Web Services oder die unzureichenden Möglichkeiten zur Dienstbeschreibung in ISO 19115 ((KELLER, 2006c), (NOGUERAS-ISO et al., 2005)) seien hier beispielhaft erwähnt.

NOGUERAS-ISO et al. (2005) weisen darauf hin, dass Ansätze aus verschiedenen Disziplinen bei der Entwicklung von Geodateninfrastrukturen berücksichtigt werden müssen. Auch die Geo-Community greift im Bedarfsfall auf bestehende IT-Konzepte und -Standards zurück. Diese werden typischerweise auf die spezifischen Anforderungen im Geo-Umfeld reduziert und gegebenenfalls um räumliche Charakteristiken erweitert (z.B. UML -> GeoUML).

Bei der Konzeption des Metadatenframeworks in dieser Thesis wird auf dieselbe Art und Weise vorgegangen. Die Konzeption soll sich in erster Linie an bewährten IT-Grundlagen und bestehenden IT-Standards orientieren. Dabei sind folgende Fragen zu berücksichtigen:

- Welcher Standard wird durch Hersteller, Tools, Middleware, etc. am besten unterstützt? Grundsatz: Nutzung von Bestehendem, möglichst wenig selber entwickeln
- Wie können IT-Standards im GI-Umfeld genutzt und falls notwendig erweitert werden? Grundsatz: nichts Neues erfinden, vermeiden von Insellösungen

Der gewählte Lösungsansatz gründet auf der Idee, dass system- und datenrelevante Informationen in einem Metadatenframework einmalig beschrieben und verwaltet werden. Alle GDI-Komponenten holen die für ihre Konfiguration wichtigen Informationen an dieser zentralen Stelle ab.

1.4.1 Theorie und Technik

In Bezug auf die vorliegende Arbeit sind folgende Grundlagen von Bedeutung:

- Geo- und IT-Standards: Normenserie ISO 19100, OpenGIS® Spezifikationen (u.a. WMS, WFS, CSW, GML, SE), OMG Spezifikationen (u.a. MDA, UML, CWM, XMI)
- OpenGIS® Abstract Specifications: OpenGIS® Reference Model, OpenGIS® Service Architecture (Topic 12), OpenGIS® Catalog Services (Topic 13)
- OpenGIS® Discussion Papers: Symbology Management, Feature Portrayal Services
- GIS-Frontends (Open Source): OpenJUMP, Quantum GIS, uDig, MapBender
- Map Server (Open Source): UMN MapServer, deegree WMS / iGeoPortal
- Geodatenbanken (Open Source): PostgreSQL, PostGIS

1.4.2 Methodik

Das methodische Vorgehen mit einem Verweis auf die entsprechenden Kapitel wird durch den folgenden Ablauf illustriert:

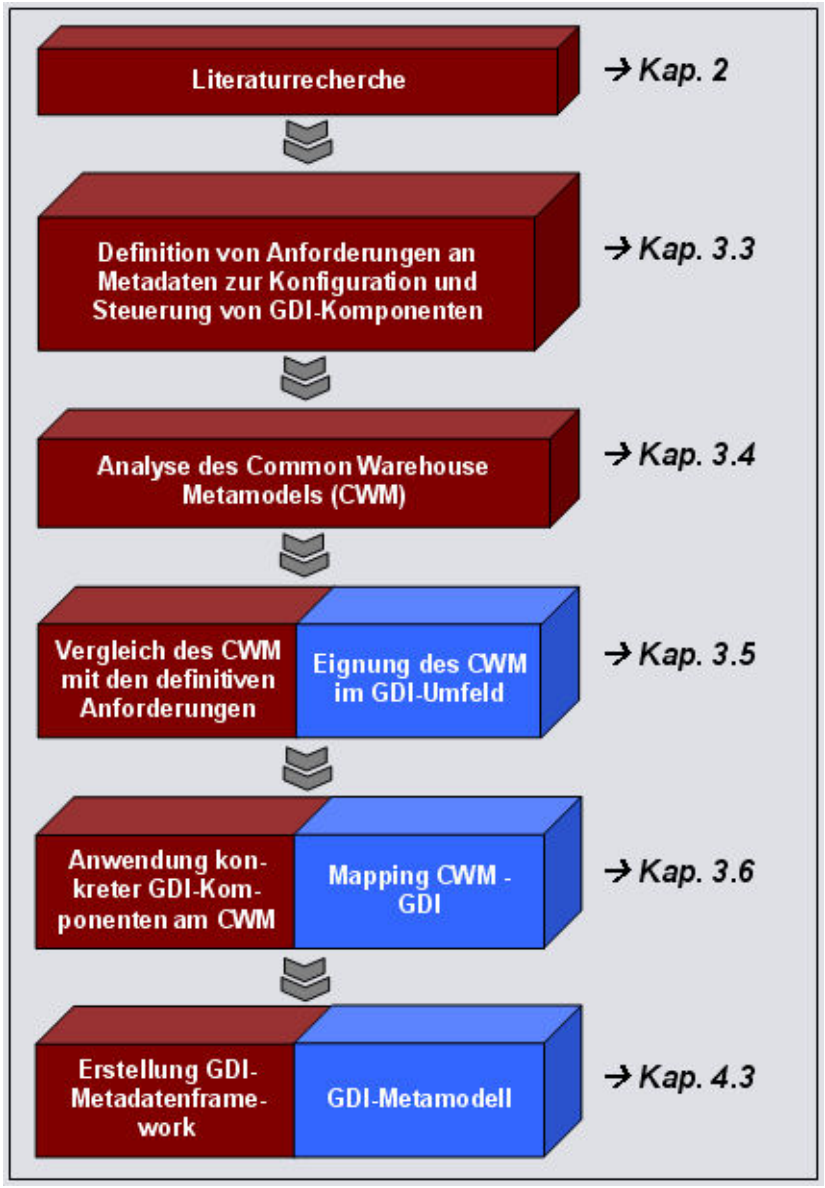

Abb. 1: Methodisches Vorgehen in dieser Thesis (rot: Methodik, blau: Ergebnisse)

Anhand einer einführenden Literaturrecherche werden zunächst die Anforderungen an Metadaten zur Konfiguration von GDI-Komponenten untersucht und festgelegt. Im nächsten Schritt wird ein Ansatz zur Standardisierung von Metadatenmodellen anhand des Common Warehouse Metamodels (CWM) analysiert und anschliessend mit den festgelegten Anforderungen verglichen. Schliesslich folgen die Anwendung konkreter GDI-Komponenten am CWM und die Modellierung eines GDI-Metadatenframeworks. Die konzeptuelle Modellierung erfolgt dabei in der Unified Modeling Language (UML).

Durch ein Metadatenmodell werden Ressourcen an einer zentralen Stelle im GDI-Framework beschrieben. Im Idealfall ist dieses Repository mit der Berechtigungsverwaltung gekoppelt, so dass der Zugriff auf Daten und Dienste durch Authentifizierung und Autorisierung gesteuert wird.

1.5 Struktur der Thesis

Diese Arbeit ist in drei Teile (A, B, C) gegliedert:

Teil A gibt eine allgemeine Einführung in diese Master Thesis (Kap. 1), gefolgt von einem Überblick über grundlegende Begriffe und Konzepte, welche im Rahmen der Literaturrecherche als relevant identifiziert wurden (Kap. 2).

Teil B beinhaltet die Kernthemen dieser Master Thesis. Darin wird zuerst der Lösungsansatz detailliert erläutert (Kap. 3). Hier werden Anforderungen an die Metadaten für den Datenzugriff und zur Applikationssteuerung definiert. Anschliessend folgen Erläuterungen zur gewählten Modellierungsmethode und eine Übersicht über die unterschiedlichen Modellierungsebenen. Zum Abschluss dieses Kapitels wird das Common Warehouse Metamodel genauer analysiert. Das resultierende Metamodell sowie weitere Ergebnisse werden in Kapitel 4 vorgestellt.

In Teil C werden die erzielten Ergebnisse anhand der festgelegten Ziele diskutiert (Kap. 5.1). Dabei wird der gewählte Modellierungsansatz (CWM) reflektiert, wobei auch Vergleiche zu bestehenden OpenGIS® Metadaten-Standards angestellt werden. Es folgt anschliessend die Beurteilung der eingangs aufgestellten Thesen (Kap. 5.2). Am Ende dieser Arbeit folgen eine Zusammenfassung und ein Ausblick (Kap. 6).

2. Grundlagen und Literatur

In diesem Kapitel erfolgt eine Einführung in die für diese Master Thesis relevanten Grundlagen. Es werden Konzepte und Begriffe erläutert, die einen Bezug zu den Themen Geodateninfrastruktur und Metadaten aufweisen und im Hinblick auf diese Arbeit von grundlegender Bedeutung sind.

2.1 Grundlegende Begriffe

Die nachfolgend aufgeführten Begriffe und Definitionen werden in dieser Thesis wiederholt verwendet. Sie sollen daher für ein gemeinsames Verständnis in Form eines Glossars[6] vorab festgelegt werden. Dort wo eine explizite Quellenangabe fehlt, stammen die Definitionen sinngemäss aus den Online-Enzyklopädien WIKIPEDIA[7] bzw. WORDNET[8]. Ausführlichere Beschreibungen zu den einzelnen Themen folgen in den anschliessenden Kapiteln.

Catalog	Gemäss OGC[9] ist ein *Catalog* ein Register von Einträgen, welches (geographische) Datensammlungen (engl. feature collection) einer spezifischen Anwendergruppe (engl. information community) anhand von Metadaten beschreibt. Ein einzelner Eintrag enthält im Minimum den Namen des zugehörigen Datensatzes und eine Angabe darüber, wo bzw. wie dieser zugänglich ist. Im Datenbank-Umfeld hat der Begriff *Catalog* eine anderweitige Bedeutung. Er umfasst alle Tabellen einer Datenbank, auf die ein Benutzer via SQL-Statements zugreifen kann. Ein *Catalog* ist diejenige Verwaltungseinheit, welche einer Datenquelle gleichkommt.
Data Dictionary / Database Catalog	Ein *Data Dictionary* oder *Database Catalog* ist ein Verzeichnis von (→ System-Metadaten), welches Informationen über die Daten bzw. Objekte einer Datenbank enthält. Es umfasst Beschreibungen aller existierenden Tabellen und Sichten mit dem Ziel einer redundanzfreien und zentralen Definition der Datenstrukturen. Das *Data Dictionary* wird hierzu als separate Datenbank aufgebaut, auf welche via SQL lesend zugegriffen werden kann.

[6] *Begriffe in alphabetischer Reihenfolge*
[7] *WIKIPEDIA: http://de.wikipedia.org (25.06.2007)*
[8] *WORDNET: http://wordnet.princeton.edu (25.06.2007)*
[9] *OGC Glossary of Terms: http://www.opengeospatial.org/ogc/glossary (25.06.2007)*

Data Warehouse (DWH)

Ein *Data Warehouse* (deutsch: Datenlager) ermöglicht eine globale Sicht auf eine Vielzahl von heterogenen und verteilten Datenbeständen. Hierzu werden die relevanten Daten aus den originären Datenquellen extrahiert und in einem konsistenten Datenbestand zusammengeführt. Vom analytischen Werkzeug zur Unterstützung von Entscheidungsprozessen hat sich das *Data Warehouse* zu einer zentralen Plattform für die integrierte Informationsversorgung in einem Unternehmen entwickelt.

Datenmodell / Datenschema / Datenbankschema

Ein *Datenmodell* beschreibt die fachlichen Anforderungen des jeweiligen Problemumfelds in einem fachlogischen Konzept. Dazu wird ein Teilaspekt der komplexen Realwelt durch Abstraktion formal in einem Modell abgebildet. Es definiert die innere Struktur der gespeicherten Daten und ihre Beziehungen untereinander. Aus Sicht der Softwareentwicklung ist ein *Datenmodell* eine abstrakte Beschreibung einer technischen Implementierung.

Die Datenmodellierung ist die konkrete und präzise Beschreibung der Daten eines Anwendungsbereichs in einem *Datenschema*. Die konkrete Beschreibung des Datenmodells in einer relationalen Datenbank wird als relationales *Datenbankschema* bezeichnet. Ein *konzeptionelles Datenschema* ist eine systemunabhängige Datenbeschreibung, während ein *logisches Datenschema* die Daten in der Sprache eines bestimmten Datenbank-Verwaltungssystems beschreibt (ZEHNDER, 1998).

Framework

Der Begriff *Framework* (deutsch: Rahmenwerk, Gerüst) bezeichnet eine fundamentale Struktur, welche Prozesse und Technologien zur Lösung eines komplexen Problems umfasst und dabei verschiedene Komponenten zu einer Gesamtlösung integriert. Es definiert die Architektur eines Systems, welches wiederverwendbare, konfigurierbare Komponenten bereitstellt.

Im Hinblick auf diese Arbeit definiert sich der Begriff *Framework*, in Anlehnung an die Definition von SCHUSTER (2003), als „*ein System, welches auf Basis eines Grundgerüsts verschiedene Komponenten für ein bestimmtes Anwendungsgebiet bereitstellt, die zur individuellen Anpassung in das System eingebunden werden können.*“

Geodateninfrastruktur (GDI)

Eine *Geodateninfrastruktur* umfasst einerseits vernetzte Geodatenbanken und Funktionalitäten zum Umgang mit Geodaten, andererseits aber auch den Bereich der institutionellen, organisatorischen, technologischen und wirtschaftlichen Ressourcen, die Entwicklung und Pflege der GDI sowie den verantwortungsvollen Umgang mit den darin zur Verfügung stehenden Geoinformationen (GROOT und McLAUGHLIN, 2000).

Im Rahmen dieser Arbeit wird eine *Geodateninfrastruktur* vorwiegend als technologisches Rahmenwerk verstanden, welches um eine Beschreibung von System-Metadaten zur Steuerung der beteiligten Komponenten erweitert wird.

Geo-Metadaten / Benutzer-Metadaten

Metadaten beschreiben im GIS-Bereich auf unterschiedlichen Aggregationsebenen Eigenschaften von Datensätzen wie Herkunft, Verwendungszweck, Vollständigkeit, Aktualität, Inhalt, Format, Struktur, Genauigkeit, Qualität, Gültigkeit, Einsatz- und Nutzungsbedingungen. *Geo-Metadaten* sind unentbehrlich für das Auffinden, die Evaluation, den Zugriff und die längerfristige Wertsicherung von Geodaten.

IT-Metadaten

Die Bezeichnung *IT-Metadaten* wird in dieser Arbeit als Oberbegriff für Metadaten verwendet, die für den Entwurf, die Konstruktion und Pflege von Informationssystemen benötigt werden. Metadaten werden im IT-Bereich in sehr unterschiedlichem Kontext angewendet. Im Bereich der Softwareentwicklung lässt sich beispielsweise der Applikations-Code auf Basis von Modellen, welche in dieser Domäne als Metadaten betrachtet werden, automatisch generieren.

Managed Meta Data Environment (MME)

Eine *Managed Meta Data Environment* (MME) ist eine Umgebung um Informationen über die Geschäftsdaten und die zugehörigen IT-Anwendungen auf Systemebene zu beschreiben und effektiv zu verwalten. Sie umfasst Architekturkomponenten, Personen und Prozesse, welche für die systematische Sammlung, Speicherung und Verbreitung von Metadaten innerhalb einer Organisation notwendig sind (MARCO und JENNINGS, 2004).

Metadaten	Als *Metadaten* bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten. Gemäss BAUER und GÜNZEL (2001) umfasst der Begriff *Metadaten* „*gemeinhin jede Art von Information, die für den Entwurf, die Konstruktion und die Benutzung eines Informationssystems benötigt wird.*“ *Metadaten* sind strukturierte Daten, mit deren Hilfe eine Informationsressource beschrieben und dadurch besser auffindbar gemacht wird.
Metadatenmodell	Datenmodell zur Beschreibung von Metadaten (→ Datenmodell)
Metadaten-Repository / Metadaten-Dictionary	Metadaten werden meist in einem separaten Bereich, dem so genannten *Metadaten-Repository* gespeichert. Dabei handelt es sich um eine Struktur zum Speichern und Verwalten von Metadaten, welche teilweise auch als *Metadaten-Dictionary* bezeichnet wird. Ein *Repository* besteht aus Datenbanktabellen, die alle notwendigen Beschreibungen zum System und zur Umwelt enthalten.
Metamodell	Ein *Metamodell* definiert eine abstrakte Syntax, welche ein Modell formal beschreibt. Das *Metamodell* legt demnach fest, nach welchen Regeln und Bedingungen ein konkretes Datenmodell erstellt wird. Im IT-Bereich sind *Metamodelle* die Grundlage zur Beschreibung von Modellierungssprachen.
	Im Rahmen dieser Arbeit ist ein *Metamodell* das konzeptuelle Schema eines Metadaten-Repositories, in dem die Metadaten-Elemente und die Beziehungen zwischen diesen festgelegt werden.
Objektmodell	Ein *Objektmodell* beschreibt die Struktur der Objekte in einem System, einschliesslich ihrer Identität, Relationen zu anderen Objekten, Attribute und Operationen.
Portrayal	Im Kontext des World Wide Web steht der englische Begriff *Portrayal* für die Art und Weise, wie Informationen graphisch aufbereitet und präsentiert werden. Bei der Kartendarstellung (engl. map portrayal) muss festgelegt werden, wie geographische Features symbolisiert und visualisiert werden.

Rendern

Rendern (engl. to render: wiedergeben) bezeichnet den Abbildungsprozess, um aus einem internen Modell durch Anwendung geeigneter Transformationen ein zweidimensionales Bild zu generieren.

System-Metadaten

Der Begriff *System-Metadaten* bezeichnet in Datenbanken und ähnlichen Systemen zum Management von gespeicherten Nutzdaten die systeminternen Daten, die zur Verwaltung der eigentlichen Nutzdaten verwendet werden (→ Data Dictionary). *System-Metadaten* sind in der Regel nicht editierbar.

Service-Metadaten

Jeder Web Service (→ Webdienst) beinhaltet Metadaten, welche die Art und Funktionalität eines Methodenaufrufs beschreiben. *Service-Metadaten* bestehen üblicherweise aus drei Teilen:

- Metadaten, die den Servicetyp beschreiben
- Metadaten, die die Operationen des Dienstes beschreiben
- Metadaten, die die Daten des Dienstes beschreiben

In MORALES (2004) werden *Service-Metadaten* auch als (→ System-Metadaten) bezeichnet.

Semantische Metadaten

Semantische Metadaten stellen eine Erweiterung der klassischen Metadaten dar, indem Zusammenhänge zwischen Metadatenelementen hergestellt werden. *Semantische Metadaten* lassen sich hinsichtlich Komplexität, Formalismus und Ausdrucksstärke zwischen elementbasierten Metadaten und Ontologien[10] einordnen.

Im Data Warehousing oder im GIS-Umfeld werden *semantische Metadaten* von Endbenutzern benötigt, die sich für den Inhalt eines Informationssystems interessieren.

Technische Metadaten

Technische Metadaten repräsentieren einen Teilbereich der in dieser Arbeit als IT-Metadaten bezeichneten Metainformationen. Sie umfassen Systeminformationen wie die Strukur der physikalisch gespeicherten Daten oder die technische Beschreibung der zugehörigen Datenhaltungssysteme.

[10] *Onthologie: Formal definiertes System von Begriffen und/oder Konzepten und Relationen zwischen diesen Begriffen*

Technische Metadaten sind beispielsweise eine wichtige Grundlage eines Data Warehouse-Systems, wo sie zur Steuerung und Automatisierung von zahlreichen Prozessen verwendet werden. Metadaten werden hier nicht nur für den Datenbeschaffungsprozess (Extraktion, Transformation, Laden), sondern auch für die Entwicklung und den Betrieb eines Data Warehouse-Systems verwendet. Unter anderem gelten auch Konfigurationsparameter von Softwarekomponenten als *technische Metadaten*.

In der vorliegenden Arbeit liegt der Fokus bei den Metadaten hauptsächlich auf den *technischen Metadaten*.

Webdienste / Web Services

Ein *Webdienst* (engl. web service) ist eine Software-Anwendung, die durch einen Uniform Resource Identifier (URI) eindeutig identifizierbar ist. Es handelt sich dabei konkret um Schnittstellen, die als XML-Artefakte definiert, beschrieben und gefunden werden können. Ein *Webdienst* unterstützt die direkte Interaktion mit anderen Software-Agenten unter Verwendung XML-basierter Nachrichten, welche auf der Basis von internetbasierten Protokollen ausgetauscht werden. Einzelne *Webdienste* stellen lose Softwarekomponenten dar, die über ein verteiltes Netzwerk gekoppelt werden können.

2.2 Geodateninfrastrukturen

Die Ressource Geoinformation (GI) liegt oft noch brach. Die effiziente Nutzung wird erschwert, da transparente Nutzungsrechte fehlen und die Entwicklung von Standards noch stark im Fluss ist. BERNARD und STREIT (2002) verweisen auf die Micus-Studie von FORNEFELD et al. (2004) und sehen eine Hauptschwierigkeit im Fehlen von erfolgreichen Geschäftsmodellen zum Erschliessen von neuen Anwendungsfeldern. Erst wenn Geoinformation im Rahmen von Geschäftsprozessen in Wert gesetzt werden kann und deren Qualität und Kosten für den Kunden transparent sind, erschliessen sich Marktpotenziale. Um die Nutzung von Geoinformation zu erleichtern wurden deshalb auf privater, nationaler, regionaler und internationaler Ebene die Bemühungen zum Aufbau von Geodateninfrastrukturen (z.B. NGDI Schweiz, GDI-DE, ESDI, GSDI) verstärkt.

Eine vollständige Konsolidierung des Begriffs ***Geodateninfrastruktur (GDI)*** hat bisher noch nicht stattgefunden. Es existieren zahlreiche Definitionen, welche unterschiedliche Aspekte hervorheben:

- Die infrastrukturorientierte Sichtweise stellt technische Komponenten wie Netzwerke, Dienste und Standards in den Vordergrund
- Die organisatorisch-institutionelle Sichtweise sieht die Verwaltung als massgeblichen Datenlieferant mit der Wirtschaft als Nutzer. Dabei müssen politische, wirtschaftliche und rechtliche Rahmenbedingungen erfüllt sein.
- Die anwendungs- und datenorientierte Sichtweise stellt Geodatenproduzenten, Geodatenveredler sowie Geoinformationnutzer in den Mittelpunkt.

Die Schweizer Koordinationsstelle für Geoinformationen[11] (KOGIS) definiert eine Geodateninfrastruktur im Rahmen des Impulsprogramms e-geo[12] als „*ein allgemein verfügbares System von Verfahren, institutionellen Einrichtungen, Technologien, Daten und Personen*“ um den Austausch und die effiziente Nutzung geographischer Daten zu ermöglichen. Die Verfügbarkeit von Geoinformation steht hier im Mittelpunkt, wobei die Definition auch den Datenaustausch und organisatorische Aspekte einschliesst.

Der Interministerielle Ausschuss für Geoinformationswesen[13] (IMAGI) definiert die Geodateninfrastruktur Deutschland (GDI-DE) aus einer vorwiegend datenorientierten Sichtweise. Die Kernkomponente einer GDI ist hiernach die Nationale Geodatenbasis, die aus Geobasisdaten, Geofachdaten und den zugehörigen Metadaten besteht.

[11] *KOGIS: http://www.swisstopo.ch/de/about/domains/kogis (25.06.2007)*
[12] *e-geo.ch: http://www.e-geo.ch (25.06.2007)*
[13] *IMAGI: http://www.gdi-de.org/de/imagi/f_imagi.html (25.06.2007)*

Eine stärker prozessorientierte Sichtweise ist in RAJABIFARD et al. (2002) beschrieben. Eine GDI besteht gemäss dieser Definition aus Daten und Nutzern, aus einem Netzwerk, sowie aus Regeln und technischen Standards. Die dynamische Beziehung zwischen den technischen Komponenten bestehend aus Netzwerk, Regeln und Standards wird hier ins Zentrum gestellt. Es wird argumentiert, dass die sich ändernden Bedürfnisse der Nutzer den Zugriff auf neue Daten in einem ständig wechselnden Technologieumfeld erfordern.

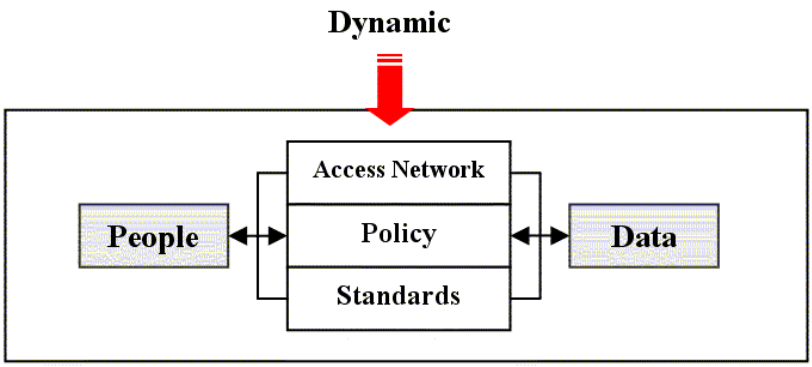

Abb. 2: Komponenten einer GDI nach RAJABIFARD et al.(2002)

Geodateninfrastrukturen stehen nach MENGE (2006) für den nahtlosen Übergang der Verarbeitungszyklen von Geodaten zu Geoinformationen zwischen Erfassung, Aufbereitung, Haltung, Präsentation und Weiterverarbeitung, ohne dabei die Anforderungen der Nutzer und den Aufbau neuer Geschäftsmodelle zu vernachlässigen. Die Betonung liegt hier auf der Anwenderrolle, da der Erfolg und die Mehrwertgenerierung im Wesentlich von den Geschäftsmodellen der Anwendungen abhängen.

In dieser Arbeit wird eine Geodateninfrastruktur vorwiegend als technologisches Rahmenwerk verstanden, welches sich aus Komponenten der verschiedenen Systemebenen zusammensetzt. Die nachfolgende Abbildung ist nicht als Referenzarchitektur einer GDI zu verstehen. Sie dient vielmehr der Klärung, welche Komponenten im Rahmen dieser Arbeit den Umfang einer Geodateninfrastruktur hinsichtlich der Erstellung eines Metadatenframeworks ausmachen. Die wichtigste Komponente in diesem Framework im Hinblick auf die folgenden Kapitel stellt das Repository für System-Metadaten dar.

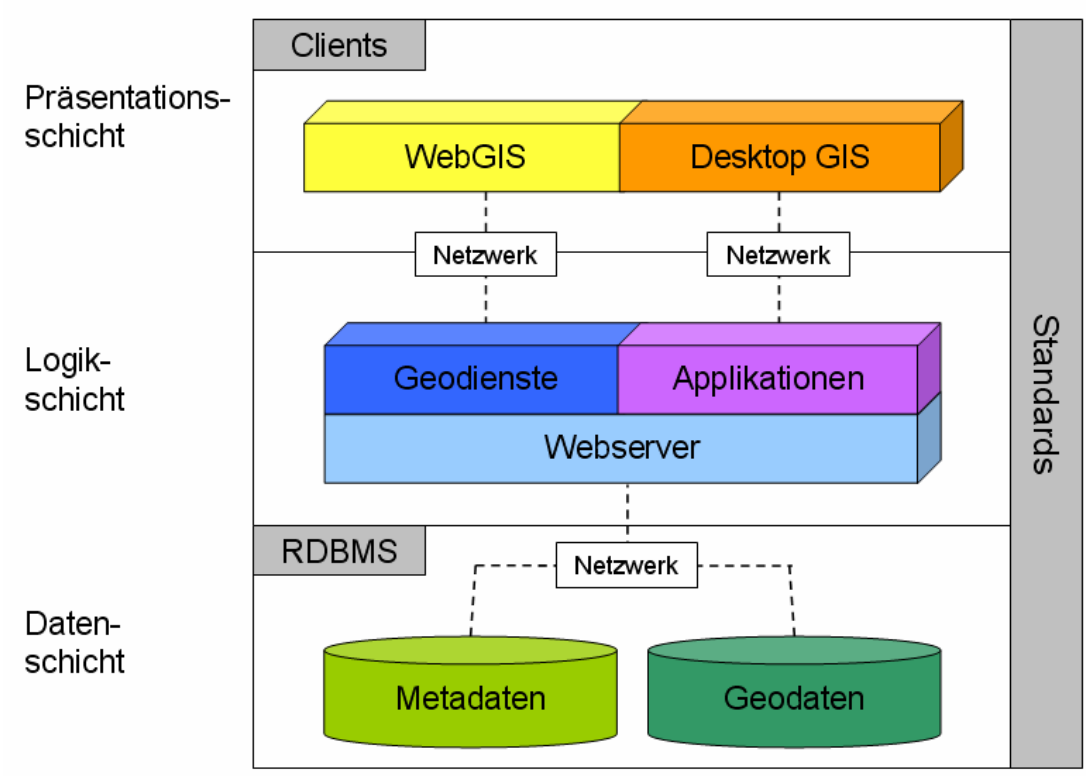

Abb. 3: Komponenten einer GDI hinsichtlich der vorliegenden Arbeit

Als ***Ziele einer Geodateninfrastruktur*** gelten gemeinhin die Erfassung, Harmonisierung und Organisation der Verbreitung und Nutzung von Raumdaten zur Aktivierung des Geoinformationsmarktes und zur Entwicklung einer lebendigen Geoinformationswirtschaft. Auf dieser Grundlage sollen einerseits transparente Entscheidungsprozesse möglich sein und andererseits die Effektivität und Effizienz des Verwaltungshandelns im Rahmen des E-Government gestärkt werden. Nicht zuletzt ist es auch ein erklärtes Ziel, Bürger mit Geodaten und -diensten zu versorgen (BRÜGGEMANN / KLEEMANN, 2004) und die angebotene Qualität von Geoinformation zu erhöhen.

Der ***Nutzen einer Geodateninfrastuktur*** liegt im einfachen Austausch und kostengünstigen Zugang zu verlässlichen Geoinformationen für die Verwaltung, für die Wirtschaft und für die Bürger. Durch nachhaltiges Vorgehen bei der Erfassung, Nachführung und Dokumentation werden Doppelspurigkeiten und Mehrfacherhebungen vermieden. Durch die Verwendung anerkannter und offener Standards im Bereich der Geodaten können die Hindernisse bezüglich Mehrfachnutzung und Austausch von Geoinformation abgebaut werden. Zudem können durch die Nutzung von homogenen Daten und existierenden Standards Anwendungen schneller und einfacher entwickelt werden ((BERNARD et al. (2004)), (RAJABIFARD et al. (2003)).

2.3 Interoperabilität und Standards

Standards sind ein elementarer Bestandteil einer GDI. SCHNEIDER und WEISSHAAR (2003) definieren einen ***Standard*** *„als eine breit akzeptierte und angewandte Regel oder Norm, entweder als offizielle Norm (de-jure-Standard) aus Normungsarbeit hervorgegangen oder als de-facto-Standard (herstellerspezifische Industrie-Standards) durch seine weite Verbreitung gesetzt.*“ De-facto Standards gehen häufig aus Industrie-Standards hervor, die durch jahrelange praktische Umsetzung breite Anwendung über Herstellergrenzen hinweg gefunden haben. Ihre Anwendung ist nicht verbindlich. Standards ermöglichen die Kommunikation zwischen verschiedenen Informationssystemen und erhöhen so deren Flexibilität und Produktivität.

Die Entwicklung von GI-Standards wird vom Open Geospatial Consortium (OGC) geleitet. Statt die Vielzahl der Daten- und Datenaustauschformate zu standardisieren definiert das OGC anhand festgelegter Konsensprozesse Spezifikationen zur Beschreibung von Funktionen, die durch Dienst-Schnittstellen bereitgestellt werden müssen (PICHLER / KLOPFER, 2004). Die Interoperabilität in der geographischen Informationsverarbeitung und die Integration von raumbezogener Informationstechnologie in Standard-IT-Verfahren soll auf der Basis der OGC Web Services (OWS) verbessert werden. Die konkrete Umsetzung der Software wird dabei nicht vorgeschrieben. Das OGC leistet einen wichtigen Beitrag in der GIS-Entwicklung von monolithischen Systemen hin zu verteilten interoperablen Dienstarchitekturen (vgl. Kap. 2.4).

Standards im Sinne des OGC gelten als offen, wenn:

- die Erarbeitung und Verabschiedung in einem konsensorientierten Verfahren erfolgt,
- die Nutzung der Standards frei von jeglichen Lizenzgebühren ist,
- der Standard öffentlich und für den Download zugänglich und deren Verbreitung nicht eingeschränkt ist,
- Standards herstellerunabhängig sind und die Nutzung unabhängig von der Wahl der Basistechnologie oder der Benutzerschnittstelle erfolgen kann.

BERNARD (2004) hat indessen mehrere Hemmnisse zur Verbreitung und zum Einsatz von GI-Standards identifiziert: Einerseits fehlen klare Richtlinien zur einheitlichen Interpretation existierender Spezifikationen. Deren Vollständigkeit und Dauerhauftigkeit sind wichtige Voraussetzungen, damit die Standards von Herstellern getragen werden. Zudem fehlt es an einer semantischen Harmonisierung (geometrische und topologische Referenzmodelle, Mehrsprachigkeit, etc.) und an Qualitätsbeschreibungen für verteilte Dienste. Das OGC bemüht sich im Rahmen der Compliance & Interoperability Testing & Evaluation Initiative (CITE[14]) um die Zertifizierung von Diensten.

[14] *CITE: http://cite.opengeospatial.org (25.06.2007)*

Geodaten stammen meist aus heterogenen Quellen, so dass eine übergreifende Nutzung nur unter der Einhaltung von Standards möglich ist. Standards sind demnach eine zentrale Voraussetzung für die Interoperabilität von heterogenen Anwendungssystemen. Nach der Definition des IEEE (GERACI, 1991) ist ***Interoperabilität*** die Fähigkeit von zwei oder mehreren Systemen oder Komponenten, Informationen auszutauschen und diese zu nutzen. Nach ISO 19118 ist Interoperabilität „*die Fähigkeit zur Kommunikation, zur Ausführung von Programmen und zum Austausch von Daten zwischen verschiedenen funktionalen Einheiten in einer Art und Weise, die von Anwendern wenige oder gar keine Kenntnisse über die Besonderheiten dieser Einheiten erfordert.*“ Interoperabilität schliesst gemäss WIKIPEDIA[15] nicht nur die System- und Anwendungsebene ein, sondern umfasst auch die organisatorische Ebene. Im Software-Kontext bedeutet Interoperabilität, dass Programme dasselbe Dateiformat oder Protokoll verwenden können. Wenn zwei Systeme miteinander vereinbar sind, nennt man sie zueinander kompatibel. Das Ziel von Interoperabilität besteht darin, Informationen auf effiziente und verwertbare Art und Weise auszutauschen bzw. dem Benutzer zur Verfügung zu stellen.

Zur Überwindung von Systemgrenzen im Umfeld einer GDI müssen gemäss PICHLER und KLOPFER (2004)

- Speziallösungen vermieden werden, indem verfügbare Standards (OGC und ISO/TC 211) konsequent angewendet werden,
- für alle Dienste und Informationsprodukte innerhalb einer GDI als Voraussetzung für die Interoperabilität eindeutige Spezifikationen festgelegt werden,
- Spezifikationen auf der Implementierungsebene festgelegt werden (Interoperabilität der Systeme, nicht nur der zugrunde liegenden Konzepte).

Gemäss HUBER (2006a) benötigt eine interoperable GIS-Lösung folgende Elemente:

- Ein Metadatensystem, das ähnlich dem Data Dictionary einer Datenbank die Metadatenstrukturen beschreibt.
- Ein Benutzerrechtesystem, das den Zugriff auf die Daten regelt.
- Eine Sprache für die Formulierung der Abfrage- und Transaktionsbefehle.
- Eine Struktur für die Übermittlung der Daten (Encoding).
- Ein Protokoll für den Transport von Befehlen und Daten.

[15] *WIKIPEDIA: http://de.wikipedia.org/wiki/Interoperabilit%C3%A4t (25.06.2007)*

Eine Geodateninfrastruktur baut in erheblichem Masse auf Normen und Standards. Sie sind eine wesentliche Voraussetzung, damit Daten über Systemgrenzen hinweg ausgetauscht und verteilte Dienste miteinander kommunizieren können. Von grosser Bedeutung für den Aufbau von Geodateninfrastrukturen sind die Standards „ISO 19115 Geographic Information - Metadata“ und „ISO 19119 Geographic Information - Services“. In diesen beiden Spezifikationen werden Metadaten beschrieben, die zur Beschreibung von Geodaten und -diensten notwendig sind. Eine Standardisierung von Metadaten ist von zentraler Bedeutung für die interoperable Geoinformationsverarbeitung.

Das Open Geospatial Consortium (OGC) hat in den vergangen Jahren eine beträchtliche Anzahl an geographischen Spezifikationen hervorgebracht, welche im GDI-Umfeld entsprechend verbreitet sind. Eine enge Zusammenarbeit besteht mit dem Technical Committee (TC 211) der International Organization for Standardization (ISO). Inzwischen wurden bereits diverse OGC-Spezifikationen in die ISO-19100er Normenfamilie integriert. Eine gute Kurzübersicht zu den ISO-Standards im Geoinformationsbereich ist bei der Geodaten-Infrastruktur Brandenburg (GIB)[16] zu finden. Darüber hinaus betätigen sich nationale Organisationen mit der länderspezifischen, geographischen Standardisierung. In der Schweiz ist dies die Schweizerische Normen-Vereinigung (SNV), welche Standards mit rechtlich verbindlichem Charakter festlegt (u.a. „INTERLIS - Modellierung und Austausch von Geodaten“ oder „GM03 - Metadatenmodell für Geodaten“). Parallel dazu werden von Organisationen wie dem World Wide Web Consortium (W3C) oder der Organization for the Advancement of Structured Information Standards (OASIS) allgemeine IT-Standards erarbeitet, welche häufig die Basis von geographischen Standards darstellen. Als prominentes Beispiel sei an dieser Stelle die Extensible Markup Language (XML) als Grundlage der Geography Markup Language (GML) genannt.

Die technologische Entwicklung im GIS-Umfeld lehnt sich wesentlich an bestehende Standards aus der Informationstechnologie (IT) an. Für die breite Nutzung der GIS-Technologien muss sich die Geoinformations-Gemeinschaft in Zukunft noch stärker bemühen, ihre Leistungen im allgemeinen IT-Kontext verfügbar zu machen (HUBER, 2006d). Wenn immer möglich sollen keine Speziallösungen hervorgebracht werden, sondern es soll auf Bestehendem und Bewährtem aufgebaut werden. Die geographischen Webdienste einer modernen Geodateninfrastruktur werden in ein Rahmenwerk von grundlegenden IT-Technologien eingebettet. Den engen Zusammenhang zwischen GIS- und allgemeinen IT-Standards zeigt die folgende Graphik am Beispiel von Katalogdiensten für Metadaten:

[16] *Übersicht zu ISO 19100: http://www.gib-portal.de/papers/GIB_Uebersicht_ISO_Standards.pdf (14.04.2007)*

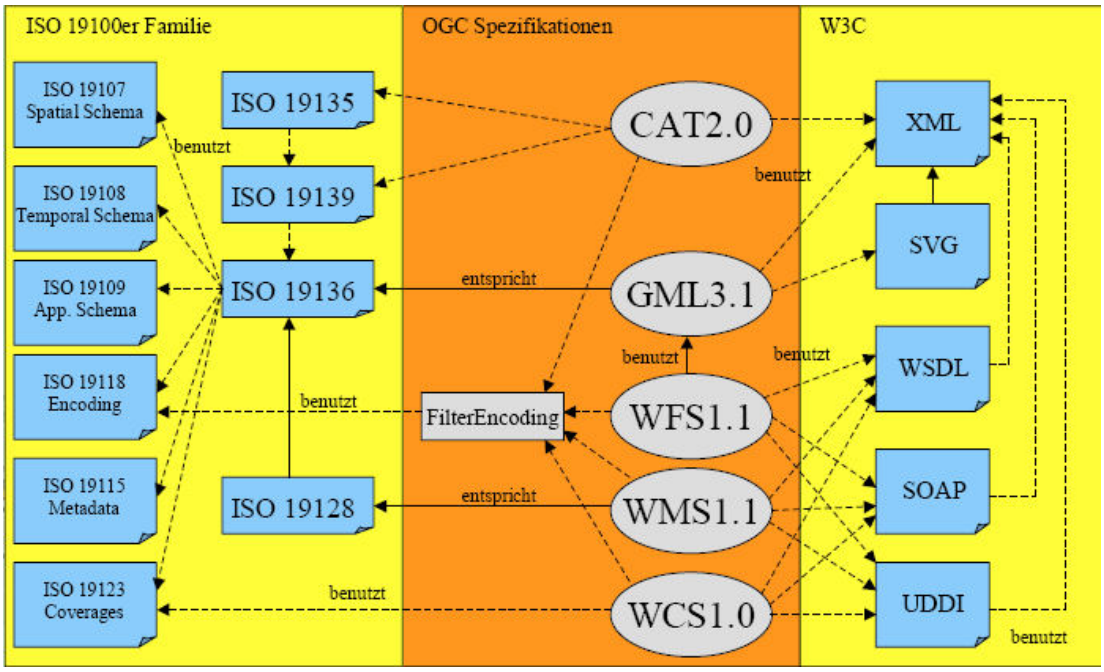

Abb. 4: Standards bei Katalogdiensten für Metadaten (Quelle: THIELE, 2006)

2.4 Interoperable Dienstarchitekturen

Das W3C (HAAS / BROWN, 2004) versteht unter einem ***Webdienst*** (engl. web service) ein Softwaresystem, welches die interoperable Interaktion zwischen lose gekoppelten Softwarekomponenten über ein verteiltes Netzwerk ermöglicht. Es handelt sich dabei konkret um Schnittstellen, die in einem maschinen-lesbaren Format beschrieben sind und die Sprache zur Kommunikation zwischen Server und Client in standardisierter Form festlegen. Gemäss HUBER (2006b) stellen Webdienste einen reinen Transportmechanismus dar, wobei der zugreifende Client (Desktop-GIS, Workflow-System, etc.) beliebig sein kann.

Das Zusammenspiel von Webdiensten erfolgt nach dem Prinzip „*Publish-Find-Bind-Chain*“. Informationen über einzelne Dienste werden vom Dienstanbieter (engl. service provider) über die Veröffentlichung und Registrierung ihrer Metadaten („publish“) bereitgestellt. Der Konsument (engl. service consumer) will einen bestimmten Dienst beziehen und findet („find“) diesen über einen Verzeichnisdienst. Anschliessend erhält der Konsument eine Beschreibung des Dienstes, womit dieser direkt in Clients eingebunden („bind“) werden kann. Mehrere Dienste können auf diese Weise miteinander gekoppelt („chain“) werden.

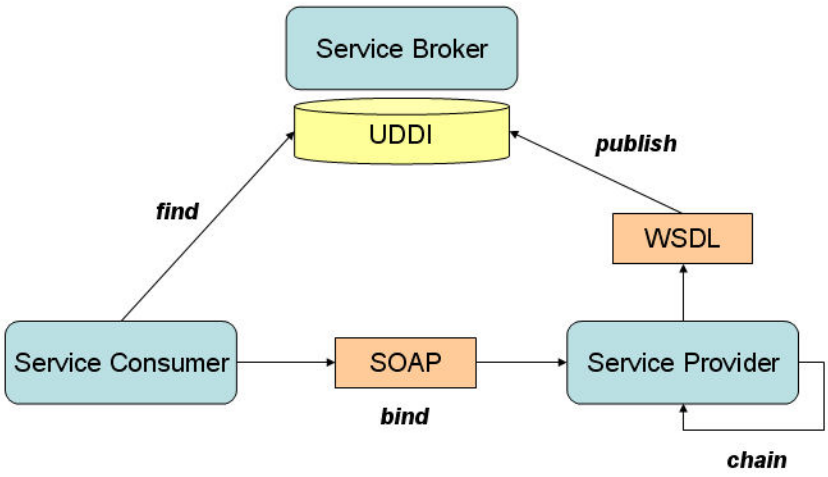

Abb. 5: Publish/Find/Bind/Chain

Die wesentlichen Komponenten einer allgemeinen Webdienst-Architektur sind Universal Description, Discovery and Integration (UDDI)[17] als Verzeichnisdienst zur Suche und Lokalisierung von Webdiensten, die Web Service Description Language (WSDL)[18] zur Beschreibung von Diensten und SOAP[19] (ursprünglich Simple Object Access Protokoll) für den Austausch von Aufrufen zwischen Konsument und Anbieter.

Durch die standardisierte Beschreibung von Webdiensten können diese miteinander kooperieren und zu Anwendungssystemen zur durchgängigen Unterstützung von Geschäftsprozessen aneinander gereiht werden. Die Programmlogik ist dabei nicht in einem einzigen Programm enthalten, sondern über mehrere unabhängige Dienste verteilt (WIKIPEDIA[20]). ISO/TC 211 und OGC (PERCIVALL, 2002) verwenden hierzu den Begriff *Service Chaining*. Darunter zu verstehen ist eine Sequenz von mehreren verketteten Dienstepaaren, welche das Ergebnis von Dienst A als Eingang für Dienst B verwendet. Ein Spezialfall des *Service Chaining* ist das Kaskadieren, bei dem Dienste des gleichen Typs hintereinander geschachtelt werden.

Unter diesen Voraussetzungen entstehen im Gegensatz zu monolithischen GIS ***serviceorientierte Architekturen*** (engl. service-oriented architecture, kurz: SOA) für Daten mit Raumbezug bestehend aus lose gekoppelten, verteilten Verarbeitungseinheiten. Diese sind atomar und operieren unabhängig von anderen Diensten. Im Gegensatz zu einem GIS stellt eine GDI funktionale Einheiten in Form von Webdiensten bereit. Da GI-Dienste von überall und immer erreichbar und unabhängig von spezifischer Hardware nutzbar sein müssen, bietet sich das Internet als zugrunde liegende Infrastruktur mit HTTP (Hyper Text Transfer Protokoll) als Kommunikationsprotokoll an. Damit die Nutzung und Kombination unterschiedlicher GI-Dienste und Geo-

[17] *UDDI: http://www.uddi.org (25.06.2007)*
[18] *WSDL: http://www.w3.org/TR/wsdl20 (25.06.2007)*
[19] *SOAP: http://www.w3.org/TR/soap12-part1 (25.06.2007)*
[20] *WIKIPEDIA: http://de.wikipedia.org/wiki/SOA (25.06.2007)*

daten ohne besonderes Vorwissen über einen spezifischen Dienst oder Datenbestand möglich ist, müssen die Dienste über standardisierte Schnittstellen und Metadatenmodelle zur Beschreibung von Geodaten und Diensten verfügen (DREWNAK, 2003). Durch die Verwendung von bereits bestehenden und weit verbreiteten Internet-Standards (HTTP, XML etc.) entsteht eine offene, erweiterbare und sehr flexible Architektur, die unabhängig von den verwendeten Plattformen, Programmiersprachen und Protokollen ist (SOGI, 2005).

Im GDI-Umfeld stehen ***Geo-Webdienste*** im Vordergrund, bei welchen im Gegensatz zu den allgemeinen IT-Webdiensten zusätzlich zur rein syntaktischen auch eine semantische Standardisierung verfolgt wird. Alle *OGC Web Services (OWS)* basieren auf demselben Funktionsschema (STARK et al., 2006), wobei ein Client mit einem Server über das HTTP-Protokoll (Hyper Text Transfer Protocol) kommuniziert.

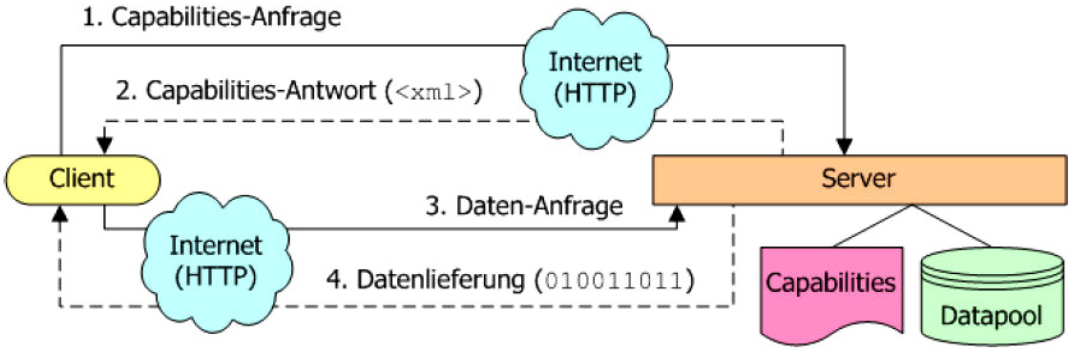

Abb. 6: Funktionsschema von OGC Web Services (Quelle: STARK et al., 2006, S.43)

Die OGC Spezifikationen definieren die Kommunikation zwischen Web Service Consumer (Client) und Web Service Provider (Server), machen allerdings keine Vorgaben wie eine Anfrage abgearbeitet wird. Geo-Webdienste verfügen über einen definierten Raumbezug und beschreiben sich gegenüber Clients über einen Mindestsatz an Metadaten selber. Damit ein Client einen Aufruf für einen bestimmten Dienst erstellen kann, muss er vorgängig gewisse Kenntnisse über dessen Fähigkeiten erlangen. Diese so genannten *Capabilities* müssen von jedem Dienst in Form eines XML-Dokuments bereitgestellt werden können. Neben Informationen zum Dienst selber (Kontaktperson, Zugriffsadresse, Dateiformate etc.) enthalten diese Service- bzw. System-Metadaten auch Informationen zu den verfügbaren Daten. Ein Client, welcher über die Schnittstelle kommuniziert, kann dank diesen Angaben dynamisch gesteuert werden.

2.4.1 OWS Service Framework

Das ***OWS Service Framework (OSF)*** des OGC definiert eine Basisarchitektur für die Implementierung und Integration von OpenGIS® Web Services als Komponenten einer Geodateninfrastruktur (PERCIVALL, 2003). Das OSF besteht aus fünf abstrakten Kategorien:

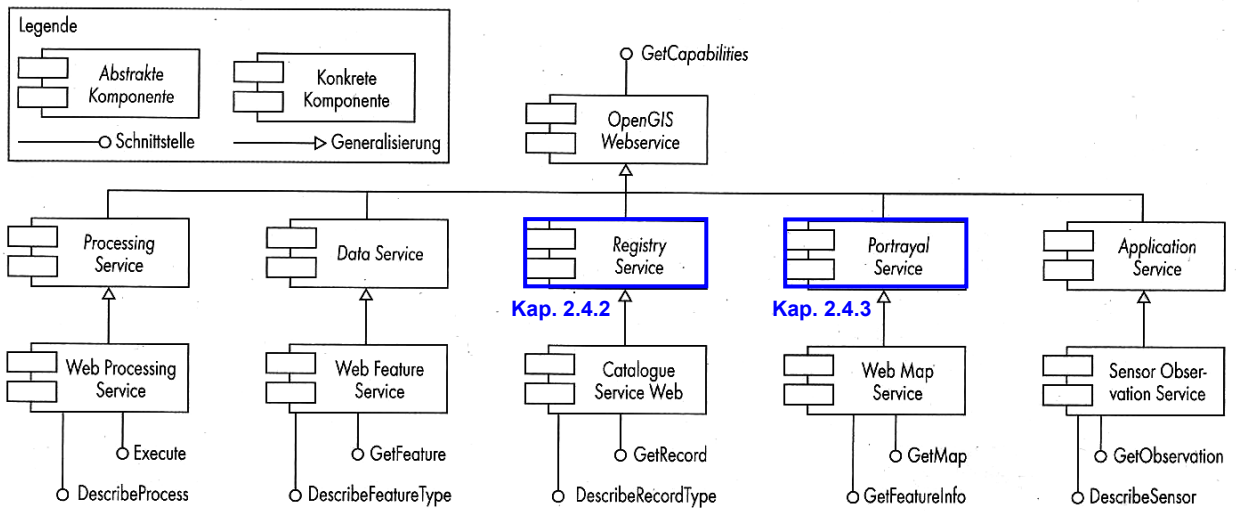

Abb. 7: OWS Service Framework (Quelle: KIEHLE, 2006)

Registrierungsdienste (*Registry Services*) sind der Einstiegspunkt für die Suche nach Daten und Diensten. Datendienste (*Data Services*) dienen dem einheitlichen Zugriff auf verteilte Datensammlungen. Darstellungsdienste (*Portrayal Services*) sorgen für die graphische Präsentation raumbezogener Daten. Verarbeitungsdienste (*Processing Services*) stellen Funktionen zur Datenverarbeitung und Informationsgenerierung bereit. Über Anwendungsdienste (*Application Services*) können Dienste der ersten vier Kategorien angesprochen werden ((PERCIVALL, 2003), (NEBERT, 2004)).

Für die Systemimplementierung ist eine detaillierte Beschreibung bis auf die Schnittstellenebene notwendig. Das OGC hat hierzu zahlreiche Implementierungs-Spezifikationen veröffentlicht, auf deren Basis eine komplette Geodateninfrastruktur aufgebaut werden kann. Anwendungen und Dienste können ohne Beeinträchtigung der Gesamtarchitektur hinzugenommen, modifiziert oder ersetzt werden. Daten- und Darstellungsdienste stellen eine standardisierte Abstraktionsschicht zwischen der Daten- und Geschäftslogikschicht dar, indem der Zugriff auf die Daten gekapselt wird und die Details der konkreten Implementierung verborgen bleiben. Somit lassen sich Daten aus unterschiedlichen Quellen miteinander kombinieren. Die Geschäftslogik wird zukünftig durch geodatenverarbeitende Prozesse und die Anbindung von Sensoren in Echtzeit zusätzlich angereichert. Mittelfristig wird die Herstellung von semantischer Interoperabilität (neben der bisher erreichten syntaktischen Interoperabilität) beabsichtigt, wobei Verfahren des Semantic Web mit Techniken der OpenGIS® Web Services verknüpft werden (KIEHLE, 2006).

2.4.2 Registry Services

Die Registry Services bieten gemäss OGC[21] einen allgemeinen Mechanismus zum Registrieren, Klassifizieren, Beschreiben, Suchen, Pflegen und zum Zugreifen auf Informationen über Netzwerkressourcen. Ein ***Catalog Service*** als konkrete Form eines Registry Services ist eine Schnittstelle zum Durchführen von Operationen zum Auffinden und Abfragen von verteilten Katalogen über Geodaten und -dienste. Metadaten werden benötigt, um alle Bestandteile dieser Informationsnetzwerke so zu beschreiben, dass die vorhandenen Geoinformationsressourcen gefunden und von Anwendern und Systemen genutzt werden können. Die Speicherung von suchmaschinenkonformen Metadaten wird in der Regel in einer Datenbank realisiert.

Ein Katalog ist nach einer genau festgelegten Struktur organisiert. Die durch einen Katalog beschriebenen Geo-Ressourcen können sowohl Daten (*GeodataCollection*) als auch Dienste (*GeoService*) sein. Ein Katalog (*Catalog*) ist dabei selber ein Subtyp eines Geodienstes. Er umfasst Metadaten-Einträge (*CatalogEntry*), welche jeweils eine bestimmte Geo-Ressource beschreiben und referenzieren. Die Ressource kann dabei auch wieder ein Katalog sein, welcher wiederum Einträge besitzt, die auf Geo-Ressourcen verweisen[22]. Der Inhalt und die Struktur von Katalogeinträgen werden durch Metadaten-Standards detailliert vorgegeben.

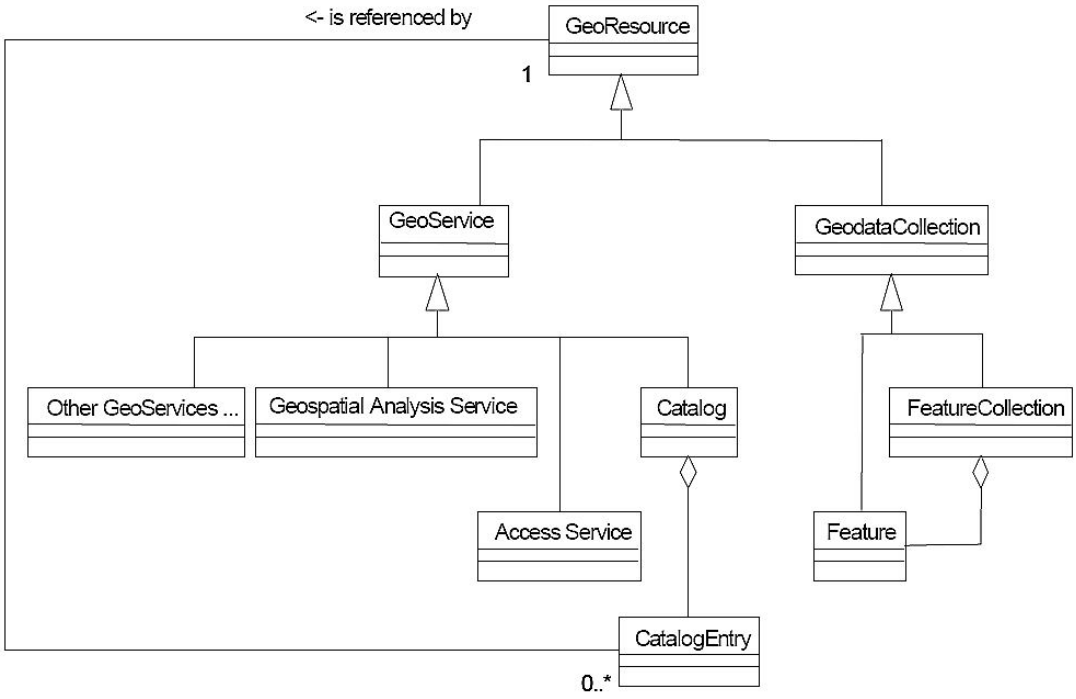

Abb. 8: Organisation und Aufbau von Katalogdiensten (Quelle: OGC, 1999)

[21] *OGC Glossary of Terms: http://www.opengeospatial.org/ogc/glossary (25.06.2007)*
[22] *Ein alternatives Konzept ist bekannt unter dem Begriff „Harvesting". Dabei werden Metadaten aus anderen Katalogen gesammelt und in einem zentralen Repository zusammengeführt.*

In den meisten Fällen amtieren Catalog Services als Vermittler (engl. broker) zwischen den Daten und den darauf zugreifenden Anwendungen (engl. clients).

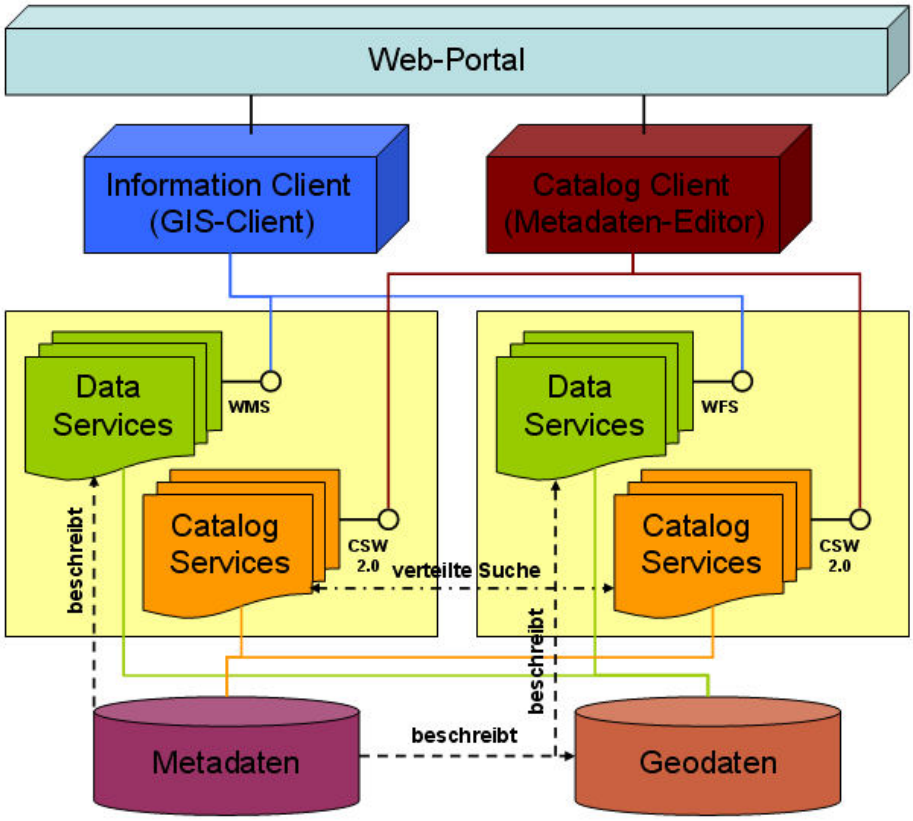

Abb. 9: Katalogdienste in einem verteilten Informationssystem

Wie *Abb. 9* zeigt, nehmen Catalog Services Suchanfragen entgegen, führen Auswertungen des Metadatenbestandes durch und geben die Suchergebnisse an den Client zurück. Eine Catalog Service-Schnittstelle verfügt zudem über Funktionen zum Einpflegen von Metadatensätzen (Metadaten-Editor). Die Interaktion mit einem Catalog Service erfolgt über ein Anwendungsprogramm (z.B. Web-Client, GIS, Fachinformationssystem), das dem Benutzer die interaktive Gestaltung der Suchanfrage ermöglicht und diese in der Folge an den Catalog Service absetzt. Das Ergebnis einer Abfrageoperation sind Metadaten im XML-Format mit Beschreibungen der gefundenen Informationen. Die zurück gelieferten Resultate werden schliesslich in der Benutzeroberfläche präsentiert.

Damit ein Client auf Catalog Services zugreifen kann, benötigt er Kenntnisse über die Schnittstelle und das zugrunde liegende Metainformationsmodell. Auf Implementierungsebene sind in Bezug auf Catalog Services folgende OGC-Spezifikationen relevant:

- OpenGIS® Catalogue Service Implementation Specification (CSW 2.0.2): Implementierungsspezifikation für webbasierte Katalogdienste. Definiert Schnittstellen, Bindings und Abfragesprachen sowie ein Framework für Applikationsprofile, die benötigt werden um Metadatenkataloge abzufragen und zu pflegen.
- CSW 2.0 ISO 19115/19119 Application profile: Applikationsprofil für ISO-basierte Metadatenkataloge

Der Metadatenstandard ISO 19115 legt eine schmale Mindestmenge an Informationen fest, die zu jeder Geoinformationsressource angeboten werden müssen. Neben diesem Minimalsatz sieht der Standard allerdings eine Vielzahl weiterer Elemente vor, die eine sehr detaillierte Beschreibung, beispielsweise der Struktur und Qualität von Datensätzen, ermöglichen. Mit der Norm ISO 19139 existiert zudem eine XML-basierte Realisierung der Metadatenspezifikation ISO 19115. Auf syntaktischer Ebene gibt es Werkzeuge (z.B. XSLT), um dieses Schema auf unterschiedliche fachbezogene Schemata zu übertragen. Die Beschreibung von Diensten erfolgt anhand der ISO-Norm 19119 „Geographic Information - Services“. Diese definiert eine Reihe von Metadatenelementen, die eine automatisierte Nutzung dieser Ressourcen unterstützen.

2.4.3 Portrayal Services

Karten ermöglichen die visuelle Interaktion und Kommunikation basierend auf geographischen Informationen. Durch visuelle Aufbereitung werden Daten in verständliche Information transformiert. Die graphische Darstellung (engl. portrayal) ist im Umgang mit Geodaten eine essentielle Aufgabe. Darstellungsinformationen stellen im Umgang mit Geodaten eine grundlegende Kategorie von Metadaten dar, die es zu verwalten und zu pflegen gilt. Die Generierung von Darstellungselementen erfordert zwei Inputs: Features und Styles. Gemäss OGC (PERCIVALL, 2003) sind Features (~Objekte) als Gruppe von räumlichen Elementen zu verstehen, die zusammen eine Einheit der realen Welt repräsentieren. Zusätzlich zu Sachinformationen besitzt ein Geoobjekt geometrische und topologische Eigenschaften. Durch Klassifikation von Instanzen nach gemeinsamen Charakteristika werden Feature Types (deutsch: Objektklassen) gebildet. Ein Style definiert die Symbolisierung von geographischen Features und deren Eigenschaften beim Erstellen von gezeichneten Karten durch Festlegung von graphischen Parametern. Der Style gibt demnach Eigenschaften und Regeln vor, wie geographische Features in einer Karte dargestellt werden. Ein Symbol ist eine graphische Entität. Ein Style verweist auf ein Symbol. Dieses Konzept macht es möglich, denselben Datensatz unterschiedlich darzustellen. Das Darstellungsmodell von PERCIVALL (2003) unterscheidet 4 Ebenen, welche die für die Visualisierung geographischer Daten erforderlichen Transformationsprozesse zeigen. Die unterste Ebene stellt

eine beliebige Geodatenquelle dar. Daraus werden Features durch die Definition von Abfragebedingungen extrahiert. Durch Zuweisung eines Styles werden diese Features in graphische Elemente transformiert, welche anschliessend von einer Rendering-Komponente unter Berücksichtigung der geforderten Darstellungseigenschaften (Auflösung, Farbtiefe, etc.) in ein zweidimensionales Raster in einem gewünschten Format umgewandelt werden. Die Darstellung dieses Rasters obliegt den Charakteristika des vom Benutzer verwendeten Software-Werkzeugs.

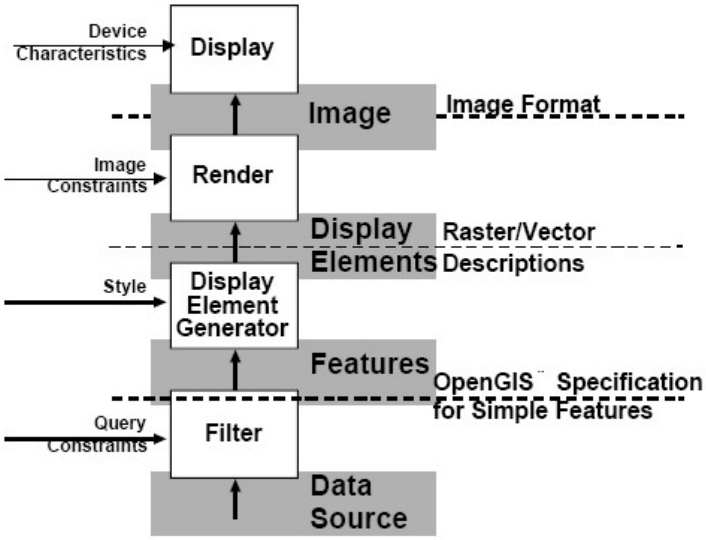

Abb. 10: OGC Portrayal Model (Quelle: PERCIVALL, 2003)

Ein ***Portrayal Service*** ist eine Schnittstelle zur Visualisierung von geographischen Features (Feature Portrayal Service) oder räumlichen Oberflächen (Coverage Portrayal Service). Die nachfolgenden Inhalte beziehen sich insbesondere auf so genannte Feature Portrayal Services (FPS) zur Symbolisierung von geographischen Objekten (engl. features) beim Rendern von digitalen Kartenbildern (WOODWARD / WHITESIDE, 2005). Den vom Benutzer (engl. client) ausgewählten und über einen Web Feature Service (WFS) bezogenen Featureklassen wird dynamisch ein graphischer Stil (engl. style) zugewiesen. Der Benutzer legt das Aussehen der von ihm bezogenen Features durch Definition eines spezifischen Styles selber fest. Die Zuweisung von graphischen Styles erfolgt beim FPS über einen Verweis auf ein Symbology Encoding-Dokument, in welchem Styledefinitionen zum Rendern einer digitalen Karte enthalten sind. Ein Feature Portrayal Service muss neben der üblichen *GetCapabilities*-Anfrage einzig die *GetPortrayal*-Operation unterstützen. Als Antwort auf eine gültige *GetPortrayal*-Operation wird eine gerenderte Karte zurückgegeben, in welcher die von einem Web Feature Service (WFS) angeforderten Features gemäss der im Symbology Encoding enthaltenen Definitionen graphisch in einem Bild dargestellt werden.

Der FPS ist einem Web Map Service (WMS) in seiner prinzipiellen Funktionsweise sehr ähnlich. Bei beiden Diensten werden geographische Quelldaten in ein digitales Bild gerendert. Im Gegensatz zum WMS verwendet der FPS beim Rendern keine layerbezogen Informationen. Der FPS ist daher als low-level Portrayal-Engine zu sehen, welche bei komplexen Szenarien eingesetzt wird. Die mit einem FPS referenzierten Styledefinitionen enthalten ausschliesslich Beschreibungen von Feature Styles, während das mit einem WMS verknüpfte SLD-Dokument zusätzliche Kontextinformationen enthalten kann (URL der Quelle, Benennung der resultierenden Layer, etc.). Der Vorteil des FPS gegenüber einem WMS mit SLD-Funktionalität liegt in der Möglichkeit, dynamische Anfragen zu erstellen, welche eine beliebige GML-Datenquelle zusammen mit einem spezifischen Style referenzieren.

Ein Portrayal Service ist ein zentraler Bestandteil eines Symbology Management Systems (SMS), welches darstellungsrelevante Informationen wie Styles und Symbole verwaltet und deren Verwendung im Kontext der Kartengenerierung festlegt. Der Catalog Service (vgl. Kap. 2.4.2) dient hier der Katalogisierung von Symbolen und Styles. Er ermöglicht dabei die Registrierung, Verwaltung, Suche und den Zugriff auf Styledefinitionen. Die Codierung der Styles im Rahmen des Portrayal-Prozesses erfolgt anhand der Symbology Encoding Language (SE). Der Feature Portrayal Service bezieht die darzustellenden GML-Daten von einem Web Feature Service (WFS) und generiert daraus ein entsprechendes Kartenbild.

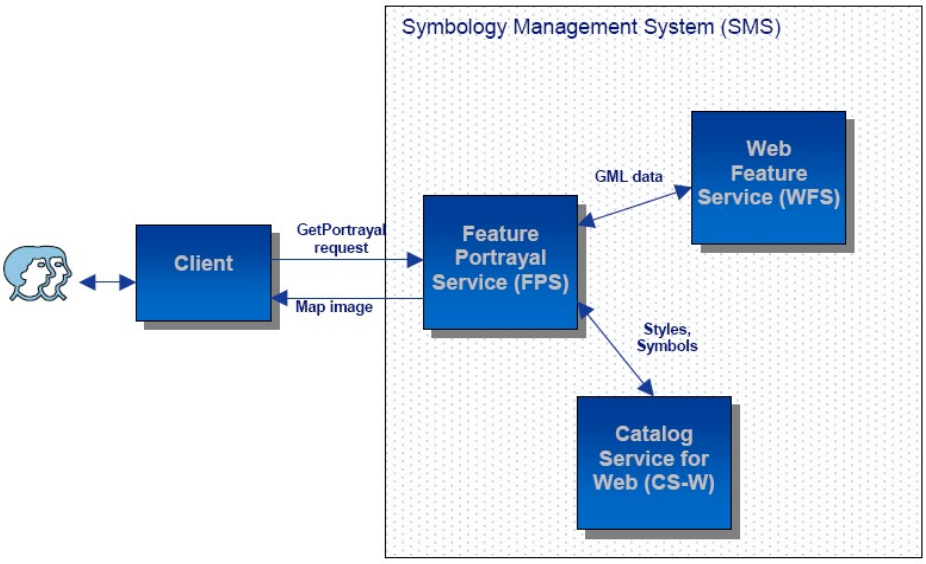

Abb. 11: Symbology Management System (Quelle: TRNINIC, 2006)

Die Kodierung von Darstellungsinformationen wird beim OGC durch das „Styled Layer Descriptor Application Profile of the Web Map Service“ (MÜLLER, MacGILL, 2005) und die „Symbology Encoding Implementation Specification“ (MÜLLER, 2006) abgedeckt. Beide Standards ermöglichen es, die Darstellung der angefragten Daten durch den Anwender zu steuern und thematische Karten durch die Integration von Filterregeln zu erstellen.

Ein ***Styled Layer Descriptor (SLD)*** erweitert herkömmliche Web Map Services (WMS) um die Möglichkeit der benutzerdefinierten Symbolisierung von Geodaten (Features, Coverages). SLD ist eine XML-basierte Sprache zur Beschreibung von Darstellungsvorschriften. Mit Hilfe von SLD kann das Ergebnis eines WMS graphisch durch den Benutzer/Client gesteuert werden. Dazu wird im *GetMap*-Request des WMS ein zusätzlicher Parameter integriert, welcher eine Referenz auf das SLD-Dokument in Form einer URL beinhaltet. In diesem Fall werden die Parameter STYLES und LAYER im ursprünglichen Aufruf weggelassen. Das Ergebnis der benutzergesteuerten Symbolisierung wird auf dem Server gerendert.

Bei Web Map Services erfolgt die Darstellung einer Karte auf Basis von so genannten „styled layers“. Ein solcher Darstellungslayer repräsentiert eine bestimmte Kombination eines Datenlayers und einem zugehörigen Style. Es ist dabei zu beachten, dass ein WMS keine Metadaten hinsichtlich der möglichen Kombinationen von Layer und Styles enthält. Eine Karte setzt sich aus transparenten Layerebenen zusammen, die einander in einer festgelegten Reihenfolge überlagert werden. Die Symbolisierung der Features kann pro Layer festgelegt werden. Die Definition eines Darstellungslayers in einer *GetMap*-Anfrage erfolgt über die Namen der zu kombinierenden Layer und Styles. Dies setzt voraus, dass diese Namen dem WMS bekannt sein müssen.

Ein WMS hat a priori keine Kenntnis über die Inhalte einer referenzierten SLD-Datei. Die Art und Weise, wie ein Client die Darstellung der geographischen Daten über SLD steuern kann, ist vielfältig. Neben dem Zugriff auf eine vordefinierte Anzahl Karten, deren graphische Erscheinung durch jeweils ein statisches SLD definiert ist, besteht auch die Möglichkeit, dass ein Benutzer das Aussehen einer Karte interaktiv festlegt und dabei „on-the-fly“ ein SLD generiert. Das statische Vorgehen erfordert die Möglichkeit, dass ein Client eine SLD-Datei durch eine entsprechende Schnittstellen-Operation (*PutStyles*) auf einem Webserver ablegen kann. Umgekehrt erfolgt die Abfrage von vorhandenen Styles über die Operation *GetStyles*.

Symbology Encoding (SE) ist eine weitere Sprache, um die Ausgabe von Web Map-, Web Feature- und Web Coverage Services graphisch festzulegen. Über das *FeatureStyle*-Element kann die Darstellung von unterschiedlichen Featuretypen definiert werden, wobei es dazu nicht erforderlich ist die Attribute der einzelnen Featuretypen zu kennen. Beispielsweise kann das Innere von Polygonen mit einer Farbe und die zugehörige Begrenzungslinie mit einer anderen Farbe symbolisiert werden. Die Darstellung von Geodaten in Abhängigkeit von spezifischen Attributausprägungen setzt hingegen die Kenntnis der Datenstruktur eines Featuretypen voraus. Eine derartige Abfrage kann beim WFS über die Operation *DescribeFeatureType* erfolgen. Symbology Encoding (SE) - ein Subset des Styled Layer Descriptors - ist eine umfassende XML-Sprache zur clientseitigen Definition von serverseitig benötigten Darstellungsregeln hinsichtlich der Visualisierung von geographischen Features.

Im Gegensatz zu SLD erfolgt die graphische Festlegung von Darstellungseigenschaften nicht auf Layerebene, sondern in Bezug auf spezifische Feature- bzw. Coveragetypen. SE dient der Speicherung von Darstellungsinformationen, die durch beliebige Dienstschnittstellen genutzt werden können. Die graphische Auszeichnung eines einzelnen Featuretypen wird durch ein *FeatureTypeStyle*-Element festgelegt. Darin können Regeln hinsichtlich einer massstabsabhängigen und attributbasierten Darstellung definiert werden. Eine wichtige Rolle spielen hierbei Filterbedingungen auf Basis der Filter Encoding[23] (FE) Spezifikation, welche die gezielte Selektion von Features aufgrund räumlicher und attributiver Kriterien ermöglicht. Ebenfalls in diese Regeln eingebettet sind so genannte „Symbolizers“ zur Beschreibung von graphischen Darstellungseigenschaften für Linien, Polygone, Punkte, Beschriftungen und Raster. Darüber hinaus stellt SE Funktionen zur Klassifizierung, Interpolation und Formatierung von Beschriftungen bereit. Um Daten aus OGC Web Services symbolisieren zu können, ist die Symbology Encoding Syntax insgesamt sehr leistungsfähig und vielfältig.

[23] *OpenGIS® Filter Encoding: http://www.opengeospatial.org/standards/filter (25.06.2007)*

2.5 Metadaten

Das Wort „meta“ stammt laut deutschem DUDEN aus dem Griechischen und ist eine Vorsilbe für „zwischen“, „um“, „nach“, „über“ und „mit“. Metadaten gelten im allgemeinen Sprachgebrauch als „Daten über Daten“ zur Beschreibung von Ressourcen (z.B. Inhalt, Format, Raum- und Zeitbezug, Herkunft, Zugriffsrechte und Speicherort). Diese zwar einprägsame Definition ist jedoch nicht hinreichend genau (HAHNE, 2005). Anders ausgedrückt handelt es sich bei Metadaten um eine Art von Dokumentation, wobei die Frage entscheidend ist, was im jeweiligen Kontext dokumentiert wird (WIENER, 2000). Die Bedeutung des Begriffs Metadaten ist also geprägt vom jeweiligen Anwendungskontext, so dass in den verschiedenen Fachdisziplinen ein unterschiedliches und oft unklares Verständnis vorherrscht. Diese Ansicht widerspiegelt sich deutlich in den folgenden Definitionen aus unterschiedlichen Anwendungsdomänen:

„*Unter dem Begriff Metadaten versteht man gemeinhin jede Art von Information, die für den Entwurf, die Konstruktion und die Benutzung eines Informationssystems benötigt wird.*“ (BAUER und GÜNZEL, 2001).

„*In Datenbanken und ähnlichen Systemen zum Management von gespeicherten Nutzdaten beschreiben Metadaten die systeminternen Daten, die zur Verwaltung der eigentlichen Nutzdaten verwendet werden.*“ (Quelle: Geoinformatik-Service der Universität Rostock[24])

„*Metadaten in GIS beschreiben Eigenschaften, Definition, Herkunft, Gültigkeit, Genauigkeit, Einsatz- und Nutzungsmöglichkeiten von Datensätzen auf unterschiedlichen Aggregationsebenen. Sie sind unentbehrlich für Dokumentation, Transfer und längerfristige Wertsicherung von räumlichen Daten.*“ (Quelle: Geoinformatik-Service der Universität Rostock[25])

„*Data that describes the meaning and structure of business data, as well as how it is created, accessed and used*.“ (DEVLIN, 1997)

Metadaten sind strukturierte Daten zur Dokumentation und formalen Beschreibung von Informationsressourcen. Metadaten werden daher vorwiegend im Zusammenhang mit elektronischen Informationssystemen angewendet. Sie erlauben es, den Inhalt der beschriebenen Ressource zu verstehen, zu vergleichen und auszutauschen.

[24] http://www.geoinformatik.uni-rostock.de/lexikon.asp *(25.06.2007)*
[25] http://www.geoinformatik.uni-rostock.de/lexikon.asp *(25.06.2007)*

Wie sich in diesen Definitionen zeigt, ist die Bedeutung von Metadaten stark von der Anwendungsdomäne geprägt. Nachfolgend wird zwischen Geo-Metadaten, IT-Metadaten und System-Metadaten sowie semantischen und technischen Metadaten (vgl. Kap. 2.1) unterschieden. Eine klare Abgrenzung der einzelnen Kategorien ist aufgrund der unterschiedlichen Interpretationen und Definitionen schwierig. Die Grenzen sind zum Teil fliessend (dargestellt durch die Überschneidung der Ellipsen).

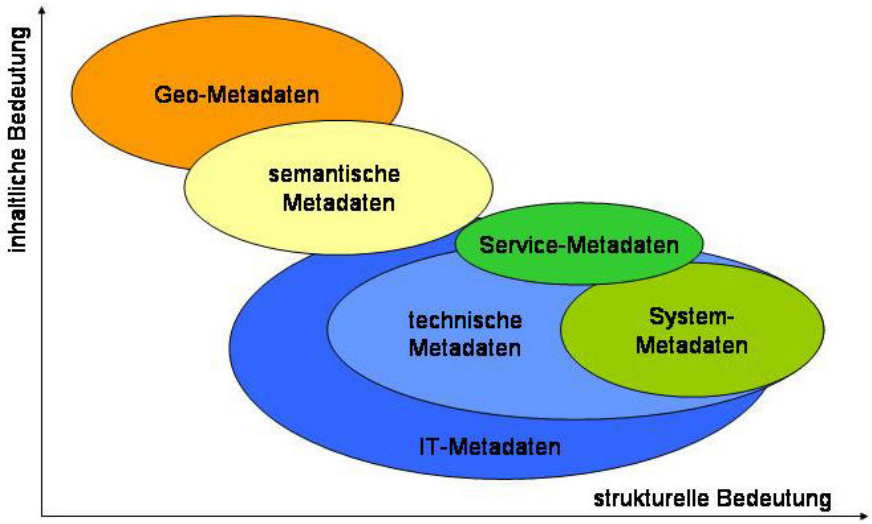

Abb. 12: Begriffliche Einordnung von Metadaten

2.5.1 Geo-Metadaten

Zur Wertsicherung von Geodaten müssen diese hinsichtlich ihrer Inhalte und Qualität beschrieben werden. Metadaten sind die Grundlage für Suchdienste zum Auffinden von Geoinformationsressourcen. Hat ein potenzieller Nutzer einmal Kenntnis über vorhandene Geodatenbestände, so benötigt er Informationen über deren Herkunft, Verwendungszweck, Vollständigkeit, Aktualität, Inhalt, Format, Struktur, Auflösung, Genauigkeit, Qualität, Gültigkeit, Georeferenzierung und Nutzungsbedingungen. Ein potentieller Bezüger kann so die Eignung von Geodaten für einen bestimmten Verwendungszweck beurteilen. Wurde ein geeigneter Datensatz gefunden, werden Metadaten benötigt, die den Zugriff auf eine Ressource ermöglichen. Geo-Metadaten sind also das Mittel um Geoinformationsressourcen zu finden, zu evaluieren und auf sie zuzugreifen (SENKLER, 2006). Sie beschreiben formal die Merkmale von Geodaten.

Undokumentierte Geodaten sind wertlos, da diese nicht nutzbar und wiederverwendbar sind. NOGUERAS-ISO et al. (2005) sehen für die Probleme hinsichtlich der Wiederverwendbarkeit von GI-Ressourcen mehrere Ursachen. Durch die hohe Dynamik von Geoinformation ist es aufwendig und schwierig, Geodaten und deren Metadaten zu synchronisieren und gleichzeitig verschiedene Versionen zu dokumentieren. Der Austausch von Geodaten wird zudem häufig durch rechtliche und politische Einschränkungen behindert. Das häufige Fehlen von geeigneten Infrastrukturen und Kompatibilitätsprobleme bei der Verwendung von Geoinformation über System- und Fachbereichsgrenzen hinweg werden ebenfalls als Ursachen genannt.

Je nach Nutzung werden gemäss MORALES (2004) und NOGUERAS-ISO et al. (2005) folgende Metadaten-Levels unterschieden:

- „*Discovery Metadata*“ werden vom Datenerzeuger bereitgestellt und dienen der Suche nach verfügbaren Geodaten.
- „*Exploration Metadata*“ liefern detaillierte Informationen über die gefundenen Geodaten und erlauben somit die Auswahl eines geeigneten Datensatzes.
- “*Exploitation Metadata*” dienen dem Zugriff und der Nutzung von geeigneten Datensätzen.

Metadaten sollen für Menschen interpretierbar, für Experten aussagekräftig und für Computer prozessierbar sein. Dazu müssen sie basierend auf Standards einheitlich beschrieben (Inhalt) und modelliert (Struktur) werden (NAJAR, 2006). In den vergangenen Jahren sind im Bereich der Geo-Metadaten mehrere nationale und internationale Dokumentationsstandards für Geodaten hervorgegangen. Darin werden Elemente definiert, welche durch Metadaten zu beschreiben sind. Bereits 1994 wurde in den USA per Gesetz festgelegt, dass alle Geodaten nach dem Content Standard for Digital Geospatial Metadata (CSDGM) des Federal Geographic Data Committee (FGDC) beschrieben werden müssen. Entsprechend gross ist die Verbreitung dieses Standards. Dublin Core (DC) ist aus der Dublin Core Metadata Initiative (DCMI) hervorgegangen und dient der domänenübergreifenden Beschreibung von Dokumenten und Objekten im Internet. Das Dublin Core Metadata Element Set (im Jahr 2003 als ISO 15836 verabschiedet) umfasst lediglich 15 Kernelemente. Als „kleinster gemeinsamer Nenner“ wird Dublin Core in der Geoinformations-Domäne oft als Austauschformat zwischen verschiedenen Metadatenstandards und als Basis für den Zugriff auf verteilte Metadatenkataloge (engl. harvesting) genutzt. ISO 19115 ist eine internationale Norm, welche vom ISO Technical Committee (ISO/TC 211) in enger Zusammenarbeit mit dem OGC speziell für Geo-Metadaten entwickelt wurde und ebenfalls 2003 in Kraft getreten ist. Der Standard umfasst über 400 Metadatenelemente, von denen 22 („Core metadata for geographic datasets“) verpflichtend sind. Die genannten Standards unterscheiden sich nicht nur in der Anzahl der zu dokumentierenden Elemente, sondern auch in ihrer Struktur und ihren Inhalten oft beträchtlich (NAJAR, 2006). Durch die hohe Komplexität gestaltet sich die Ablösung von bestehenden Metadatenstandards durch ISO 19115 in der Praxis oft mühsam. ISO 19115 erlaubt die Festlegung von so genannten Profilen (z.B. das Schweizerische GM03), die den Gesamtumfang auf das Wesentliche einschränken. Ein Profil muss mindestens die Core-Elemente des ISO-Metadatenmodells enthalten. Bei Bedarf kann ein Profil aber auch um eigene Metadatenelementen erweitert werden. Eine wichtige Regel legt dabei fest, dass vorhandene Elemente in eigenen Profilen nicht abgeändert werden dürfen (SCHNEEBERGER, 2005).

Metadaten spielen auch im Umfeld von Webdiensten eine wesentliche Rolle. Die Spezifikation der OpenGIS® Catalogue Services (vgl. Kap. 2.4.2) definiert Schnittstellen, Bindings, Encodings und Abfragesprachen sowie ein Framework für Applikationsprofile zur Abfrage und Pflege von Metadatenkatalogen. Das Inkrafttreten von ISO 19115 hat das OGC dazu veranlasst, für den Web Catalogue Service (CSW) ebenfalls ein Profil von ISO 19115 („ISO19115/ISO19119 Application Profile for CSW 2.0“) zu publizieren. Aufgrund der unzureichenden Möglichkeiten von ISO 19115 zur Beschreibung von Geo-Webdiensten hat das OGC mittlerweile einen Entwurf einer Service Catalogue Spezifikation basierend auf ISO 19119 hervorgebracht. Trotz dieser Bemühungen geht die Tendenz derzeit eher in Richtung von generischen, nicht geospezifischen Standards wie WSDL ((STARK et al., 2006), (LIEBERMAN, 2003)). Auf technische Details wie Abfragesprachen sowie Encodings und Protokolle für den Austausch von Metadaten wird an dieser Stelle nicht eingegangen.

Metadaten sind der Schlüssel zur Zugänglichkeit und Auffindbarkeit von Daten und somit eine der wichtigsten und ersten Komponenten einer Geodateninfrastruktur (KELLER, 2006b). Metadaten sind bei verteilten GIS und bei vernetzten GDIs eine zentrale Voraussetzung für Interoperabilität. Metadaten fördern gemeinsam mit räumlichen Datenmodellen die Verknüpfbarkeit von Geodaten. Aus NAJAR (2006) geht hervor, dass verbessertes Metadatenmanagement auch die Interoperabilität von Geodaten verbessert. Umgekehrt sind Geodaten ohne Metadaten im Kontext von GDIs unbrauchbar und verlieren ohne effektives Metadatenmanagement an Wert. NAJAR (2006) schlägt daher eine integrierte Verwaltung von Metadaten und Geodaten vor, so dass Geodaten ihre eigene Metadatenbeschreibung beinhalten. Der Vorteil dieses Ansatzes liegt darin, dass die Konsistenz zwischen Geo- und zugehörigen Metadaten erhöht wird.

2.5.2 IT-Metadaten

Der Begriff Metadaten wird im IT-Bereich deutlich weiter gefasst als im Geo-Umfeld, was hauptsächlich auf die vielseitige Anwendung von Metadaten zurückzuführen ist. Analog zum Geo-Umfeld sind auch in klassischen IT-Disziplinen Daten ohne Dokumentation völlig wertlos. Neben Informationen über die Herkunft und Erzeugung ist vor allem das Wissen über die Bedeutung der Daten von Interesse. In diesem Zusammenhang hat sich der Begriff Metadaten Management etabliert. KRCMAR (2004) spricht dabei vom Management von Wissenssammlungen, wobei Wissenseinheiten um zugehörige Metadaten erweitert werden. Ein in der Literatur umfassend diskutiertes Anwendungsgebiet ist das Management von Wissen innerhalb grosser Unternehmen. Dabei sind so genannte Data Warehouses fester Bestandteil der IT-Infrastruktur, mit deren Hilfe die Versorgung von Entscheidungsträgern mit analytischen Informationen erreicht wird. Eine Vielzahl unterschiedlichster Metadaten bildet die Basis für das Funktionieren derartiger Informationssysteme. Diese werden „*im Rahmen von Erstellung, Betrieb und Nutzung eines Data Warehouses von allen beteiligten Systemkomponenten fortlaufend produziert und konsumiert.*“ (MELCHERT et al., 2002)

Metadaten werden im Data Warehousing nicht nur für den Datenbeschaffungsprozess (Extraktion, Transformation, Laden), sondern auch für die Entwicklung und den Betrieb des Systems verwendet. Sie erlauben die Wiederverwendung von Entwicklungsbausteinen und darüber hinaus auch die konsistente und integrierte Dokumentation des Systems (MELCHERT et al., 2002). Bei der Erfassung von Metadaten im Data Warehouse-Bereich werden im Wesentlichen zwei Ziele verfolgt: „*Zum einen soll mittels technischen Metadaten der Aufwand für den Aufbau und den laufenden Betrieb des Data Warehouse-Systems minimiert werden, zum anderen sollen durch Business-Metadaten eine optimale Auswertung der Daten ermöglicht werden*“ (BAUER / GÜNZEL, 2001). In der Data Warehouse-Domäne verschiebt sich das Gewicht der Informationsbedürfnisse mit der wachsenden Komplexität und Vielfalt der Unternehmensdaten immer stärker weg von rein technischen Metadaten hin zu einer umfassenden Verwaltung aller Metadaten, was insbesondere auch geschäftliche Metadaten einschliesst (JOSSEN, 2004).

Gemäss STAAB (2004) lassen sich Warehouse-Metadaten in drei Kategorien einteilen:

- Semantische Metadaten: Terminologie, Transformations- und Integrationsregeln zur Abbildung der operativen Daten auf die Data Warehouse-Strukturen, Aggregationsregeln für das Zusammenfassen der Daten auf verschiedenen Aggregationsstufen
- Verwaltungstechnische Metadaten: Benutzer und zugehörige Zugriffsrechte, statische Daten und Tabellen
- Schematische Metadaten: Logisches Schema, Abbildung zwischen logischem und physischem Schema, Datenquellen

Aufgrund der Verwendung im Data Warehouse unterscheiden BAUER und GÜNZEL (2001) zwischen der passiven Nutzung von Metadaten zur reinen Dokumentation der Struktur eines Warehouse-Systems und der aktiven Nutzung von Metadaten zur Beschreibung von Prozessen sowie zur Steuerung von Werkzeugen zur Datenerfassung, -speicherung und -analyse. Bei der aktiven Nutzung werden Metadaten dynamisch in die Softwareausführung eingebunden. Werden Metadaten explizit ausserhalb von Anwendungsprogrammen gespeichert, so kann die Flexibilität hinsichtlich deren Wiederverwendbarkeit und Wartbarkeit in Entwicklungsprozessen gesteigert werden. Eine gute Übersicht über verschiedene Ansätze zur Klassifikation von Metadaten im Warehouse-Bereich ist bei LEGLER und ZUBOV (2002) zu finden.

Wie eingangs bereits erwähnt wurde, sind die Anwendungsmöglichkeiten von Metadaten in der IT-Domäne vielfältig. Auch in der Software-Entwicklung sind Metadaten seit dem Aufkommen von so genannten CASE-Tools (Computer Aided Software Engineering) von wachsender Bedeutung. Ausgehend von einer objektorientierten Beschreibung in einer Entwurfssprache wie UML (Unified Modelling Language) lässt sich der Applikations-Code basierend auf Metadaten zu einem gewissen Teil automatisch generieren.

Dieselbe Stossrichtung wird mit der ***Model Driven Architecture (MDA)*** verfolgt, deren Ziel die modellgetriebene, generative Softwareentwicklung ist. Dieser Standard der Object Management Group (OMG) ermöglicht es, aus plattformunabhängigen Modellen (engl. platform-independent models, kurz: PIM) Software für spezifische Technologien und Plattformen (platform-specific models, kurz: PSM) automatisch zu generieren.

"*MDA provides an open, vendor-neutral approach to the challenge of interoperability, building upon and leveraging the value of OMG's established modeling standards: Unified Modeling Language (UML); Meta-Object Facility (MOF); and Common Warehouse Metamodel (CWM).*" (Quelle: OMG[26])

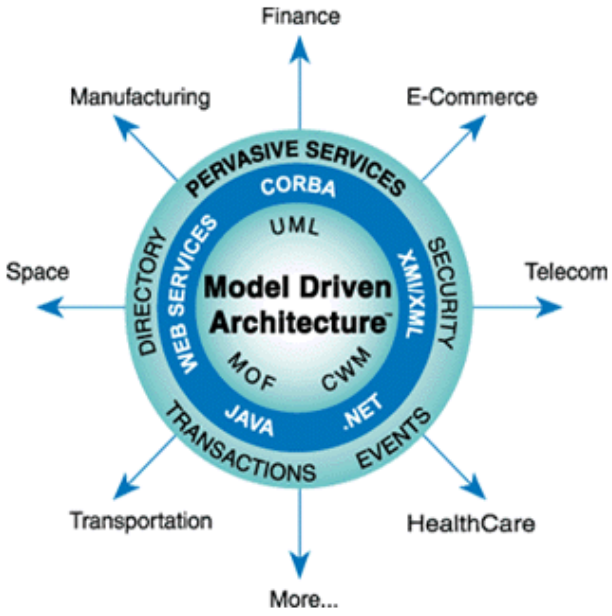

Abb. 13: Model Driven Architecture (Quelle: OMG[27])

Der Vorteil der Model Driven Architecture liegt darin, dass die Langlebigkeit eines Software-Systems durch Wiederverwendung der Modelle erhöht wird. Durch Trennung der Fachlogik von den technischen Aspekten der Realisierung können Änderungen in der Laufzeitumgebung effizient gehandhabt werden. Die Model Driven Architecture definiert Modelle auf verschiedenen Abstraktionsebenen. Das Computation Independent Model (CIM) beschreibt den Problemraum ohne dabei technische Details zu beinhalten. Mit dem Platform Independent Model (PIM) werden unter Zuhilfenahme von UML Funktionalitäten und konkrete Geschäftsaspekte technologieunabhängig modelliert. Durch Transformation können aus einem PIM mehrere Platform Specific Models (PSM) abgeleitet werden. Ein PSM ist die Grundlage zur Code-Generierung auf einer spezifischen Plattform.

[26] http://www.omg.org/mda *(25.06.2007)*
[27] http://www.omg.org/mda *(25.06.2007)*

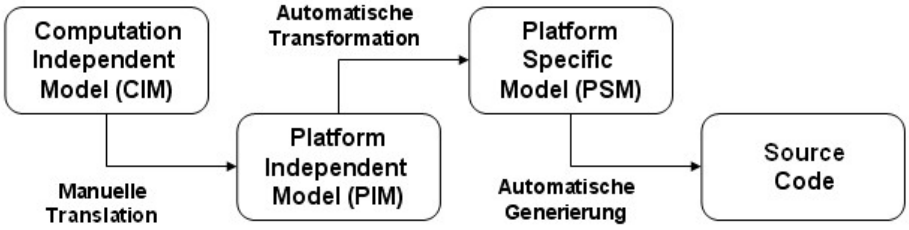

Abb. 14: Funktionsschema der Model Driven Architecture

Die MDA basiert auf vier unterschiedlichen Abstraktionsebenen, die in FRANKEL (2003) detailliert erläutert werden. Generell gilt, dass die nachfolgenden Levels jeweils Instanzen der übergeordneten Levels sind.

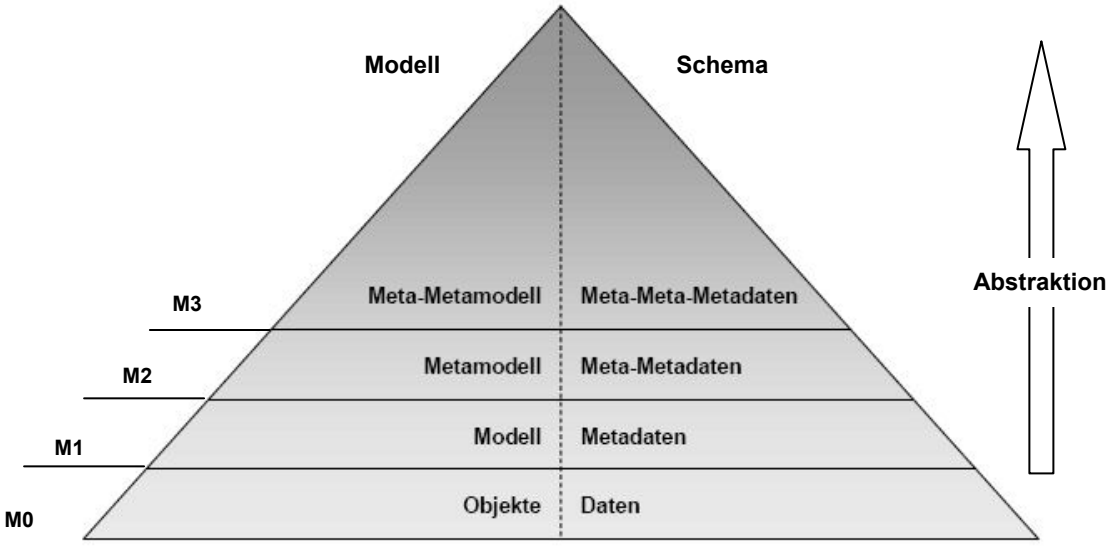

Abb. 15: Abstraktionsebenen der Model Driven Architecture (nach HAHNE, 2005)

M3: Meta-Metamodell mit Konstrukten zur abstrakten Beschreibung von Metamodellen. Die abstrakte Syntax der Modellierungskonstrukte von Metamodellen wird mittels der Meta Object Facility (MOF) formal definiert.

M2: Metamodelle, deren Konstrukte (bei UML z.B. Klassen, Attribute, Operationen, Parameter, Assoziationen) mittels MOF beschrieben werden. Man spricht in diesem Zusammenhang von abstrakter Syntax (engl. abstract syntax), welche ein Modell formal beschreibt. Metamodelle sind konkrete Instanzen von MOF-Konstrukten.

M1: Auf dieser Ebene erfolgt die Beschreibung bzw. Instanzierung von konkreten Modellen. In diesem Kontext wird von konkreter Syntax (engl. concrete syntax) gesprochen. Modelle sind Instanzen von Metamodell-Konstrukten.

M0: Repräsentation von realen Entitäten/Objekten. Objekte und Daten sind Instanzen von Modell-Konstrukten.

Da sich die früheren Ideen einer universellen Modellierungssprache nicht durchgesetzt haben, existieren heute für unterschiedliche Zwecke verschiedene Arten von Metamodellen mit unterschiedlichen Konstrukten (Relationale DB -> Tabellen, Attribute, Schlüssel etc. / UML -> Klassen, Operationen, etc.). Auf einer höheren Abstraktionsebene hat die MDA daher MOF eingeführt, um die Modellierungskonstrukte der verschiedenen Sprachen konsistent, nach einem universellen Verfahren beschreiben zu können. MOF verwendet UML-Konstrukte, um die abstrakte Syntax eines Metamodells zu beschreiben. MOF-Metamodelle haben daher das gleiche Aussehen wie UML-Klassendiagramme. Somit können herkömmliche UML-Tools verwendet werden, um MOF-Metamodelle zu erstellen. Mittels der Object Constraint Language (OCL) werden Metamodelle weiter präzisiert. Wie alle anderen Metamodelle werden auch die Konstrukte von MOF mittels MOF beschrieben. Dieses Metamodell wird häufig als "das" MOF-Modell bezeichnet. Da UML-Modelle auf dem MOF Meta-Metamodell basieren, können diese unter Zuhilfenahme von XML Metadata Interchange (XMI) zwischen UML-Softwaretools frei ausgetauscht werden. Ohne ein gemeinsames Meta-Metamodell wäre dies nicht praktikabel. MOF kann als Meta-Metamodell für beliebige objektorientierte Systeme dienen.

Im Zusammenhang mit MDA wird der Begriff Metadaten auf sämtliche oben genannten Abstraktionsebenen, mit Ausnahme von M0, erweitert.

Er umfasst dabei auch Transformationsregeln (XMI) und die daraus resultierenden Schemas (XML Schema, API). Weiter gelten auch Konfigurationsparameter von Softwareprodukten als Metadaten. Auf die Unified Modeling Language (UML) und das Common Warehouse Metamodel (CWM) wird im Lösungsansatz (vgl. Kap. 3) näher eingegangen.

2.5.3 System-Metadaten

„*Geographische Metadaten wurden lange Zeit als menschen-lesbare Beschreibung von Geodaten verstanden. Diese Sicht ist leider noch sehr verbreitet und zeigt nur einen Bruchteil des möglichen Nutzens von Metadaten.*“ (HUBER, 2006c)

Demnach besteht auch ein Bedarf an Metadaten, die über die herkömmliche Beschreibung von Datenbeständen hinausgehen. DROZ et al. (2006) zeigen anhand der Metadatenbank des Kantons Bern, dass der Einsatz von Metadaten im Rahmen von Geodateninfrastrukturen zweiteilig ist: Von allgemeiner Relevanz sind jene Elemente, die dem Benutzer als Geodatendokumentation zur Verfügung stehen und nach aussen publiziert werden. Daneben gibt es aber auch Metadaten, welche zur Steuerung von spezifischen Applikationen verwendet werden. Diese Art von Metadaten wird beispielsweise dazu verwendet um die Darstellung einer „Internet-Karte“ zu steuern. Darüber hinaus wird auf einer logischen Ebene festgelegt, welche Geoprodukte, Ebenen, Karten etc. in der Geodatenbank vorhanden sind.

Auf die Rolle von Metadaten im Zusammenhang mit Webdiensten wurde bei den Geo-Webdiensten bereits eingegangen. Jeder Webdienst besitzt seine eigenen Metadaten, die die Art und Funktionalität eines XML-Methodenaufrufs beschreiben. Durch die Bereitstellung von Metadaten ergibt sich die Möglichkeit kaskadierender Aufrufe von Web Services, bzw. die Möglichkeit zur dynamischen Auswahl des aufzurufenden Web Services (anstelle einer statischen Programmierung). Damit gewinnt die Verwaltung, Verteilung und Standardisierung von Metadaten in diesem Anwendungsbereich zusätzlich an Bedeutung. MORALES (2004) bezeichnet diese Art von Beschreibungen als System-Metadaten; Sie werden in einem Service-Repository gespeichert und extern verfügbar gemacht.

2.5.4 Aktueller Stand / Ausblick

Die Diskussion um Metadaten wird trotz bestehenden Standards teilweise kritisch geführt. Die Entwicklung ist im Fluss und noch lange nicht abgeschlossen. Die Unzufriedenheit mit der derzeitigen Situation kommt in folgendem Zitat aus dem Artikel von WALSH (2006) zum Ausdruck: "*Standards are useless partly because they overfocus on production of metadata, not consumption.*" Vor allem im Bereich der geographischen Metadaten zeigen sich einige Unzulänglichkeiten, die von KELLER (2006c) adressiert werden: ISO 19115 ist einerseits sehr komplex und es mangelt an der Möglichkeit zur Beschreibungen von Webdiensten. Zudem besteht nach wie vor Uneinigkeit darüber, über welches Protokoll (CSW, WFS, OAI-PMH[28], Z39.50[29]) ein Zugriff auf ein Metadatenmodell zu realisieren ist. Ein gemeinsames Verständnis in der Standardisierung ist durch die Vielfältigkeit und Heterogenität von bestehenden Metadatenmodellen und deren Inhalten gemäss NOGUERAS-ISO et al. (2005) nicht erreicht. Die aktuelle Forschung um Geo-Metadaten beschäftigt sich inzwischen ebenfalls intensiv mit den Entwicklungen des Semantic Web[30]. NOGUERAS-ISO et al. (2005) schlagen diesbezüglich vor, dass zukünftige Geosuchdienste die Informationsgewinnung bei weniger strukturierten Benutzeranfragen unter Einbezug von Onthologien ermöglichen sollten. Auch hier steht die Frage im Raum, ob es Sinn macht, für das Geo-Umfeld eigene Wege durch die Realisierung von Insellösungen zu beschreiten. Ob all der technischen Diskussionen ist entscheidend, dass der nach Geoinformation suchende Endanwender im Mittelpunkt der zukünftigen Entwicklungsbestrebungen steht.

[28] *Open Archives Initiative Protocol for Metadata Harvesting*
[29] *Standard des American National Standards Institute (ANSI)*
[30] *Erweiterung des Internets, um die im WWW verfügbaren Informationen mit für Maschinen verarbeitbarer Semantik zu erweitern / Standards: Resource Description Framework (RDF) / Web Ontology Language (OWL)*

2.6 Data Warehouse Systeme

Nach der Definition von INMON (1996) ist ein ***Data Warehouse (DWH)*** „*eine themenorientierte, integrierte, statische und zeitlich variable Sammlung von Daten, die so organisiert ist, dass sie die Bedürfnisse des Managements unterstützt.*"[31] Das Ziel eines solchen Systems liegt darin, aus einer Fülle von Daten Informationen zu gewinnen, die wirtschaftlich genutzt werden können. Das Data Warehouse ist mehr als eine Momentaufnahme der Unternehmenssituation, da es Informationen für verschiedene Zeitpunkte vorhält. Im Unterschied zu operativen, transaktionsorientierten Systemen werden Daten in einem Data Warehouse dauerhaft gespeichert, ohne dass diese durch Benutzer geändert werden können. Die Daten der operativen Transaktionen sollen von den statischen Warehouse-Daten getrennt werden. Ein Data Warehouse ist demnach eine Infrastruktur, die eine globale Sichtweise auf eine Vielzahl von heterogenen und verteilten Datenbeständen in einem Unternehmen ermöglicht.

Eine Data Warehouse-Architektur besteht aus drei Schichten:

Transformations- / Extraktionsschicht:

Zuerst werden die relevanten Daten im Rahmen eines periodisch durchgeführten ETL-Prozesses (Extract, Transform, Load) aus verschiedenen operativen Quellen extrahiert, durch Transformation in einen konsistenten, strukturierten und verdichteten Zustand überführt und danach in das Data Warehouse geladen.

Applikationsschicht (OLAP[32]-Schicht):

Hier werden Anfragen entgegen genommen und entsprechende Ergebnisse in Form von betriebswirtschaftlichen Kennzahlen auf verschiedenen Aggregationsstufen berechnet. Ebenfalls auf dieser Schicht anzusiedeln ist das so genannte *Data Mining*, ein Datenanalyseverfahren um aus grossen historischen Datenmengen verborgene Zusammenhänge zu ermitteln.

Präsentationsschicht (Analyse-Schicht):

Zu dieser Schicht gehören Clients für unterschiedliche Aufgaben, wie die Formulierung von Benutzer-Anfragen, die Generierung von Reports und die visuelle Aufbereitung von Ergebnissen.

[31] *Übersetzung der englischen Originaldefinition*
[32] *Online Analytical Processing (OLAP)*

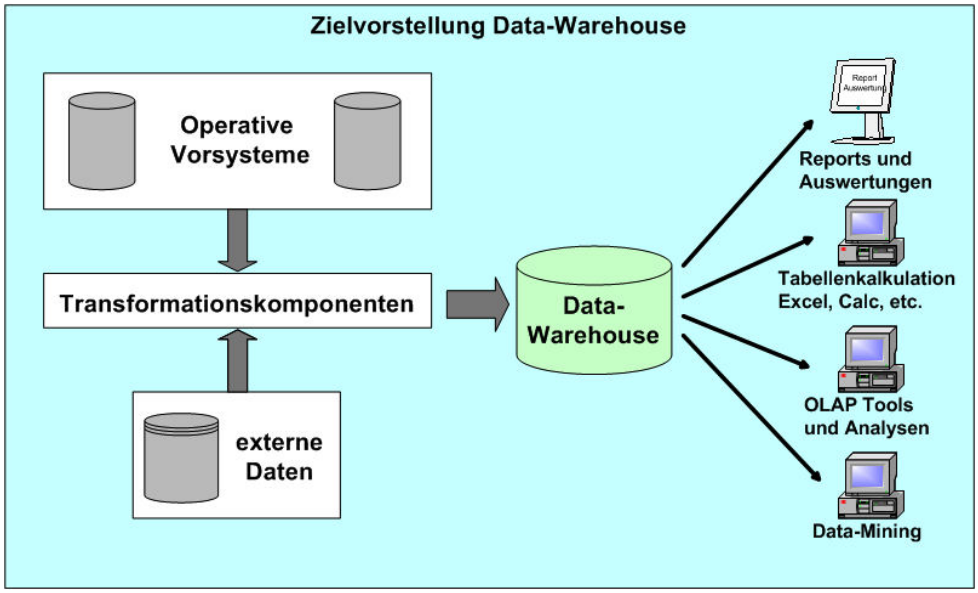

Abb. 16: Data Warehouse Systeme (Quelle: WIKIPEDIA[33] nach ANAHORY/MURRAY, 1997)

In allen drei Schichten werden Metadaten benötigt, die die gespeicherten Daten hinsichtlich ihrer Semantik beschreiben und zudem dazu beitragen, Softwarekomponenten zu einem funktionierenden Gesamtsystem verknüpfen zu können. Im Idealfall besteht eine zentrale Metadatenbasis, die zur Steuerung von allen zum Data Warehouse gehörenden Komponenten genutzt wird. Das Managment von Metadaten bietet einen vielversprechenden Ansatz zur effektiven und effizienten Integration von Softwarekomponenten in einer komplexen, heterogenen Data Warehouse-Architektur ((MELCHERT, 2003), (DO / RAHM, 2000)).

JOSSEN (2004) nennt u.a. folgende Gründe, weshalb das Management von Metadaten in Unternehmen notwendig ist: Es ist Voraussetzung für die inhaltliche Verknüpfung von Daten sowie für eine erweiterte automatische maschinelle Datenverarbeitung. Zudem werden Daten für Software wie auch für den Endbenutzer interpretierbar. Ein Problem ist dabei häufig, dass Metadaten der eingesetzten Werkzeuge zueinander nicht kompatibel sind und deren Austausch somit nur mühsam mögich ist. Ein standardisiertes Austauschformat für Metadaten soll die Interoperabilität von DWH-Systemkomponenten verbessern. Die Object Management Group (OMG) hat sich dieser Aufgabe angenommen und stellt seit 2001 mit dem Common Warehouse Metamodel (CWM) eine Möglichkeit zum Metadatenaustausch zwischen verschiedenen Softwarekomponenten einer Data Warehouse-Architektur bereit (vgl. Kap. 2.8). Ziel ist die einheitliche und verbindliche Festlegung von Syntax und Semantik der Metadaten. Inzwischen wurde dieser Standard von einigen Herstellern (u.a. im Oracle Warehouse Builder) implementiert.

[33] http://de.wikipedia.org/wiki/Data_Warehouse *(25.06.2007)*

Die essentielle Bedeutung von Metadaten in einem Data Warehouse ist unbestritten. Es besteht weitgehend Einigkeit darüber, dass eine physikalische Abtrennung der Metadaten von den Anwendungsdaten sinnvoll ist. Zur Verwaltung von Metadaten wird ein so genanntes Metadaten Repository[34] angelegt und gepflegt. Hierbei handelt es sich üblicherweise um eine rechnergestützte Datenbank, die als Basis für hoch integrative Systeme dient. Ein Repository enthält Metadaten zur Unterstützung von Entwicklung, Wartung und Betrieb eines Informationssystems, so dass diese Systeme ohne Programmieraufwand flexibel auf Änderungen reagieren können. Im Umfeld von relationalen Datenbanken wird im Allgemeinen der Begriff Data Dictionary[35] verwendet. Ein Data Dictionary enthält nicht direkt die Anwendungsdaten, sondern Metadaten, welche die Struktur (nicht den Inhalt) der Anwendungsdaten beschreiben (WIKIPEDIA[36]).

Metadaten können über eine spezielle, durch die Interface Definition Language (IDL) definierte, Schnittstelle ausgetauscht werden. Eine am Informationssystem beteiligte Software greift auf die zentral verwalteten Metadaten des Metadata Repositories über eine Programmier-Schnittstelle zu. Andererseits besteht auch die Möglichkeit, Metadaten in Form von XML-Dateien auszutauschen. Mit dem so genannten XML Metadata Interchange (XMI) hat die Object Management Group (OMG) ein spezielles XML-basiertes Format zum Austausch von Metadaten zwischen Software-Systemen entwickelt. Dieses Format ist in entsprechenden Document Type Definitions (DTD) bzw. XML-Schemata festgelegt.

Ein ***Geodaten-Warehouse***[37] (engl. spatial data warehouse) ist eine zentrale Informationsbasis eines Unternehmens, in welchem Geodaten aus verteilten Produktionssystemen zusammengeführt und zu Abfrage- und Analysezwecken bereitgestellt werden. Die Daten aus den unterschiedlich strukturierten Quellen sind dabei nach einheitlichen Regeln in einer zentralen Datenbank zu integrieren, so dass eine konsistente und normalisierte Datenbasis bereit steht. Das Ziel besteht darin, Raum- und Sachdaten logisch und redundanzfrei zu vereinen und somit Geschäfts- und Entscheidungsprozesse zu vereinfachen. Metadaten sind eine wichtige Grundlage für den Aufbau von Geodaten-Warehouse-Lösungen. Metadatenbanken stellen Ansammlungen von beschreibenden Daten dar, die im Allgemeinen über die Beschreibung der Datenbestände hinausgehen.

[34] *englisch für Lager oder Depot*
[35] *englisch für Datenkatalog oder Datenverzeichnis*
[36] *WIKIPEDIA: http://de.wikipedia.org/wiki/Data_Dictionary (25.06.2007)*
[37] *GIS-Glossar der GIS-Fachstelle des Kantons Basel-Landschaft: http://www.geo.bl.ch/glossar (25.06.2007)*

2.7 Universal Meta Data Model

Angesichts der stetig steigenden Datenvielfalt und schnell wachsenden IT-Infrastrukturen stellt sich ein nachhaltiger Nutzen nur ein, wenn auch Informationen über die Geschäftsdaten und die zugehörigen IT-Anwendungen auf Systemebene beschrieben und effektiv verwaltet werden. MARCO und JENNINGS (2004) bezeichnen eine dafür geeignete Umgebung als ***Managed Meta Data Environment (MME)***. Diese umfasst Architekturkomponenten, Personen und Prozesse, welche für die systematische Sammlung, Speicherung und Verbreitung von Metadaten innerhalb einer Organisation notwendig sind.

Qualitativ minderwertige, falsch interpretierte und schlecht verwaltete Daten sowie redundante, unausgereifte Anwendungen behindern Organisationen in der effektiven Erfüllung ihrer Aufgaben. Während Systeme zur Bewältigung von beinahe allen geschäftsrelevanten Aufgaben konzipiert und realisiert werden, ist ein übergeordnetes Konzept zur Verwaltung all dieser Systeme meist nicht vorhanden. Viele IT-Entwicklungen wachsen vom Kleinen ins Grosse, so dass auch bei einer erfolgreichen Umsetzung die Überführung bzw. Wiederverwendbarkeit dieser Lösungen zum Scheitern verurteilt ist. Eine MME ist in der Lage, durch Katalogisierung aller technischen (Anwendungen, Daten, Prozesse, Hard- und Software) und geschäftlichen Metadaten (Business-Knowledge) die eingesetzten Systeme methodisch zu verwalten. Dadurch wird einerseits die Qualität und das Verständnis über die vorhandenen Daten verbessert und andererseits sind Applikationen vollständig beschrieben, so dass IT-Prozesse wiederverwendbar und überführbar sind. Eine MME ist jedoch kein Data Warehouse für Metadaten, sondern eine operative Umgebung, welche aus den folgenden sechs Komponenten bzw. Ebenen besteht:

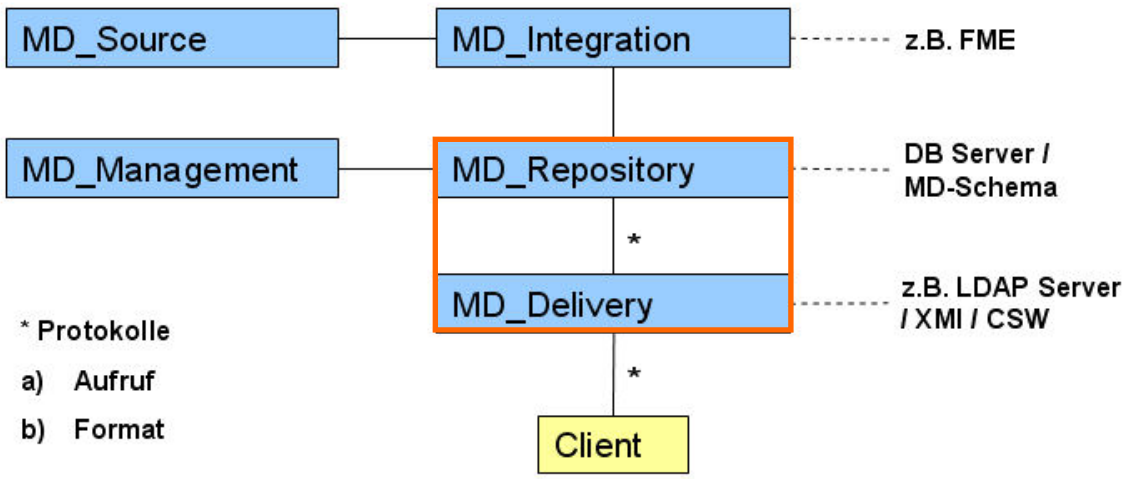

Abb. 17: Komponenten einer Managed Meta Data Environment (MME)

Meta Data Sourcing Layer

Der *Meta Data Sourcing Layer* extrahiert Metadaten aus einer Quelle und überführt diese entweder in die Integrationsschicht oder direkt in ein Metadaten-Repository. Der *Sourcing Layer* muss vom *Integration Layer* getrennt sein.

Die Quellen zur Extraktion von Metadaten sind vielfältig (unvollständige Aufzählung):

Software-Tools: Mögliche Quellen sind relationale Datenbanken und Modellierungswerkzeuge. Die MME liest die Systemtabellen der Datenbank aus, um Metadaten über Spalten- und Tabellennamen, Entitäten, Beziehungen, Indizes und Datenzugriffe zu extrahieren.

Endbenutzer: Grundsätzlich ist zwischen technischen und geschäftlichen Nutzern zu unterscheiden. Über Weboberflächen können Endbenutzer Metadaten (Geschäftsregeln, Qualitätsregeln, Formeln, Prozessregeln, Datenherkunft, Datenzugriff, etc.) direkt erfassen.

Meta Data Integration Layer

Der *Meta Data Integration Layer*, welcher der physischen Speicherung vorgelagert ist, empfängt vom *Sourcing Layer* unterschiedliche Metadaten. Auf dieser Ebene werden die Daten integriert und ins Metadaten-Repository geladen. Während im Data Warehouse der Transformationsprozess vom Ladeprozess getrennt ist, werden diese beiden Arbeitsschritte in einer MME aufgrund des relativ geringen Datenvolumens kombiniert.

Meta Data Repository

Das *Meta Data Repository* ist die Kernkomponente einer MME. Es basiert auf einer relationalen Datenbank zur Katalogisierung und physischen Speicherung von Metadaten. Es sollte soweit als möglich generisch, d.h. frei von applikationsspezifischen Eigenschaften sein. Um den langfristigen Nutzen einer MME sicherzustellen, muss ein Metadaten-Repository durch wiederkehrende Updates aktuell gehalten werden.

Meta Data Delivery Layer

Der *Meta Data Delivery Layer* repräsentiert die Schnittstelle zwischen einem Repository und Endbenutzer-Clients bzw. Anwendungsprogrammen. Häufig generiert ein Repository eine Datei, auf die eine Anwendung lesend zugreift. Eine weitere Zugriffsmöglichkeit besteht darin, direkt via Frontend auf die relationalen Tabellen mittels SQL zuzugreifen.

Meta Data Mart

Ein *Meta Data Mart* ist eine spezielle Zwischenschicht, welche Daten aus dem Repository extrahiert und einer grossen, homogenen Gruppe von Anwendern zur Verfügung stellt. Die Einrichtung von *Meta Data Marts* in einer MME ist optional. Im Rahmen einer GDI spielt dieser Layer eine untergeordnete Rolle.

Meta Data Management Layer

Der *Meta Data Management Layer* dient der systematischen Verwaltung und Pflege des Repositories und anderer MME-Komponenten. Typische Funktionen dieser Schicht sind u.a. die Administration von Benutzern sowie Archivierung, Backup, Modifikation, Synchronisation, Tuning, Statistik (Query und Load), Abfrage, Reporting, Recovery, Sicherheit und Versionierung (mittels Timestamps oder Versionsnummern). Dieser Layer verwaltet und wartet auch die Benutzeroberflächen zum Zugriff von Endbenutzern auf die MME. Die zugänglichen Inhalte sind dabei von den Rechten und Profilen der jeweiligen Nutzer abhängig.

Data Stewardship

Daten sind ein wichtiges Kapital einer Unternehmung. Um diese effektiv in Wert setzen zu können, müssen sie rasch zugänglich, sauber strukturiert, akkurat und verständlich sein. Ebenso wichtig wie technologische Aspekte zum Aufbau einer Metadaten Management-Lösung ist die Regelung der Verantwortlichkeiten zu deren Pflege. Im Umfeld einer MME wird dazu der Begriff *Data Stewardship* verwendet. Data Stewards bilden die Schnittstelle zwischen der IT-Infrastruktur und den Geschäftsanwendern.

Von zentraler Bedeutung ist die Qualität von Metadaten im Umfeld einer MME. In einem System zur Wissensgenerierung und Entscheidungsunterstützung muss der Spielraum für Interpretationen minimal sein. Eine wichtige Aufgabe besteht in diesem Zusammenhang darin, physische Strukturen in einer für den Endanwender verständlichen Weise abzubilden. Die Einführung einer logischen Datensicht ist eine zentrale Anforderung, wobei Tabellen und Spalten aus dem physischen Schema durch logische Entitäten und Attribute ersetzt werden. Zu diesem Zweck ist es erforderlich, Endbenutzerabfragen direkt mit den Metadaten in der MME zu verknüpfen. Auf diese Weise liefert eine Abfrage anstelle eines kryptischen Codes eine für den Benutzer verständliche logische Bezeichnung.

2.8 Das Common Warehouse Metamodel

Alle Software-Komponenten eines Informationssystems sollten zunächst auf der Ebene der Metadaten integriert werden, bevor eine Integration auf Datenebene stattfindet. Metadaten sind in diesem Fall jede Art von Information, die für den Entwurf, die Konstruktion, die Wartung und die Benutzung eines Informationssystems benötigt wird (BAUER / GÜNZEL, 2001). Hierzu gehören Angaben über die Datenquellen, Regeln für Transformations- und Konsolidierungsschritte sowie zur Verbesserung der Datenqualität, Zuordnungsinformationen zwischen den Datenquellen und den Data Warehouse-Modellen sowie die Metadaten der Datenmodelle des Data Warehouse selbst (HUFFORD, 1997).

Ein Metadatenbanksystem ist ein wesentlicher Bestandteil in einer Data Warehouse-Architektur. Darin werden Informationen über die verfügbaren Daten und die beteiligten Systemkomponenten abgelegt und verwaltet. Neben der Möglichkeit zum schnellen und sicheren Auffinden benötigter Informationen besteht der Nutzen von Metadaten darin, eine Interpretation der Data Warehouse-Inhalte zu ermöglichen. Um das volle Nutzenpotenzial von Metadaten erschliessen zu können, müssen diese über den werkzeugbezogenen Einsatz hinaus verfügbar gemacht werden. Der Austausch von Metadaten zwischen den einzelnen Werkzeugen ist ein vielversprechender Ansatz zur Verbesserung der Interoperabilität in heterogenen Systemlandschaften (DO und RAHM, 2000). Um eine Standardisierung der Metadaten und deren effektive Verwaltung zu erreichen, muss zunächst die Modellierung von Metadaten vereinheitlicht werden, so dass für deren Austausch ein anwendungsübergreifendes Konstruktionsschema vorgegeben wird (GOEBEL, 2005).

Das ***Common Warehouse Metamodel (CWM)*** ist eine von der OMG (2003) entwickelte Spezifikation für den formalen und herstellerunabhängigen Zugriff und Austausch von Metadaten in der Domäne des Data Warehousing. Der CWM-Standard dient dazu, den Metadatenaustausch zwischen verschiedenen Softwarekomponenten einer Data Warehouse-Architektur sowie die Interoperabilität zwischen heterogenen Data Warehouse-Umgebungen zu erleichtern. Dazu wurde ein Metamodell einer generischen Warehouse-Architektur definiert, mit dessen Hilfe Metadaten aus betrieblicher und technischer Sicht in abstrakter Form einheitlich beschrieben werden können. Das CWM liefert damit ein Rahmenwerk zur Abbildung von Metadaten über Datenquellen und -ziele, Transformationen, Analysen sowie über Prozesse und Operationen zur Erzeugung und Verwaltung von Warehouse-Daten. Als abstraktes Modell ist das CWM vollkommen plattformunabhängig und eignet sich dadurch als Übersetzungsreferenz für plattform-spezifische Metadatenmodelle. Auf den Aufbau und Umfang des CWM wird beim Lösungsansatz in Kap. 3.4 näher eingegangen.

Zusätzlich zur Spezifikation wird CWM mit folgenden vier Komponenten ausgeliefert:

1. Das komplette CWM in Form von UML-Diagrammen als Rational Rose[38]-Modell
2. Eine MOF-konforme Version des Common Warehouse Metamodells als XML-Datei
3. Eine CWM DTD-Datei zur Prüfung der ausgetauschten XML-Dokumente
4. Eine IDL-Repräsentation (CORBA Interface Definition Language) des CWM

CWM wird von namhaften Herstellern wie Oracle (Oracle Warehouse Builder), IBM (DB2 Warehouse Manager), IKAN (CWD4ALL), Informatica (SuperGlue), SAS (METADATA SERVER), Meta Integration oder Hyperion Solutions unterstützt. Diese Aufzählung erhebt keinen Anspruch auf Vollständigkeit.

Das Common Warehouse Metamodel ist im Kontext der Model Driven Architecture (vgl. Kap. 2.5.2) auf der Metamodellebene M2 einzuordnen. Von Metamodellen wird gesprochen, wenn Modelle und die Modellbildung selbst zum Gegenstand der Modellierung werden. Sie definieren eine abstrakte Syntax, mit welcher Modelle formal beschrieben werden können. Modelle beschreiben die Struktur von Objekten und sind daher als Metadaten zu verstehen. Das CWM ist Bestandteil der Model Driven Architecture (MDA) und weist daher eine enge Beziehung zu Unified Modeling Language (UML), Meta Object Facility (MOF) und XML Metadata Interchange (XMI) auf. Zusammen bilden diese Sandards den Kern der OMG Metadata Architecture. CWM ist eine domänenspezifische Erweiterung dieser Architektur.

MOF ist als eine abstrakte Syntax zur Definition von Metamodellen selber ein Modell für Metamodelle (daher Meta-Metamodell). Die Hauptkonstrukte von MOF sind Klassen, Assoziationen, Pakete, Datentypen und Bedingungen (engl. constraints). Die Unified Modeling Language bildet die Grundlage sowohl von MOF wie auch von CWM. MOF verwendet die UML-Notation zur Beschreibung von Metamodellen, so dass das UML-Metamodell rekursiv durch UML definiert wird. CWM beinhaltet ein Objektmodell, welches auf dem UML-Metamodell basiert. Es nutzt einen Ausschnitt des UML-Standards (Klassendiagramme, Packages, Object Constraint Language) und fügt gleichzeitig Data Warehouse-spezifische Erweiterungen hinzu. Aufgrund dieser gemeinsamen Basis besteht zwischen UML und CWM eine starke strukturelle Ähnlichkeit, so dass UML wiederum als Notation für CWM-Diagramme eingesetzt wird. Da das UML-Metamodell ebenso wie das CWM eine Instanz der MOF darstellt, lassen sich CWM- und UML-konforme Metadaten über so genannte *MOF Reflective Interfaces* austauschen. Metadaten können direkt über IDL[39]-Schnittstellen einem funktionsbasierten Zugriff zugänglich gemacht werden. In diesem Fall greift die beteiligte Software auf die zentral verwalteten Metadaten des CWM zu. Andererseits besteht auch die Möglichkeit, Metadaten über XML Metadata Interchange (XMI) in Form von XML-Dokumenten auszutauschen, wobei dieser Mechanismus

[38] *UML-Werkzeug von IBM*
[39] *Interface Definition Language der OMG*

von der jeweiligen Middleware-Technologie unabhängig ist. Eine datenbankbasierte Lösung zum Austausch von Metadaten ergibt sich, wenn das CWM als konzeptionelles Datenmodell eines Metadaten Repositories genutzt wird, in dem CWM-konforme Metadaten verschiedener Herkunft verwaltet werden. Die Implementierung erfolgt via UML (MELCHERT et al., 2003).

Mittels der *XMI DTD Production Rules* können aus CWM entsprechende Document Type Definitions (DTD) abgeleitet werden, welche die Syntax für Metadaten enthaltende XML-Dokumente definieren. Eine Standard-DTD wird als Teil des CWM-Standards mitgeliefert. Mittels *XMI Document Production Rules* können Metadaten in XMI-Dokumente kodiert bzw. aus XMI-Dateien die enthaltenen Metadaten extrahiert werden (MELCHERT, 2003).

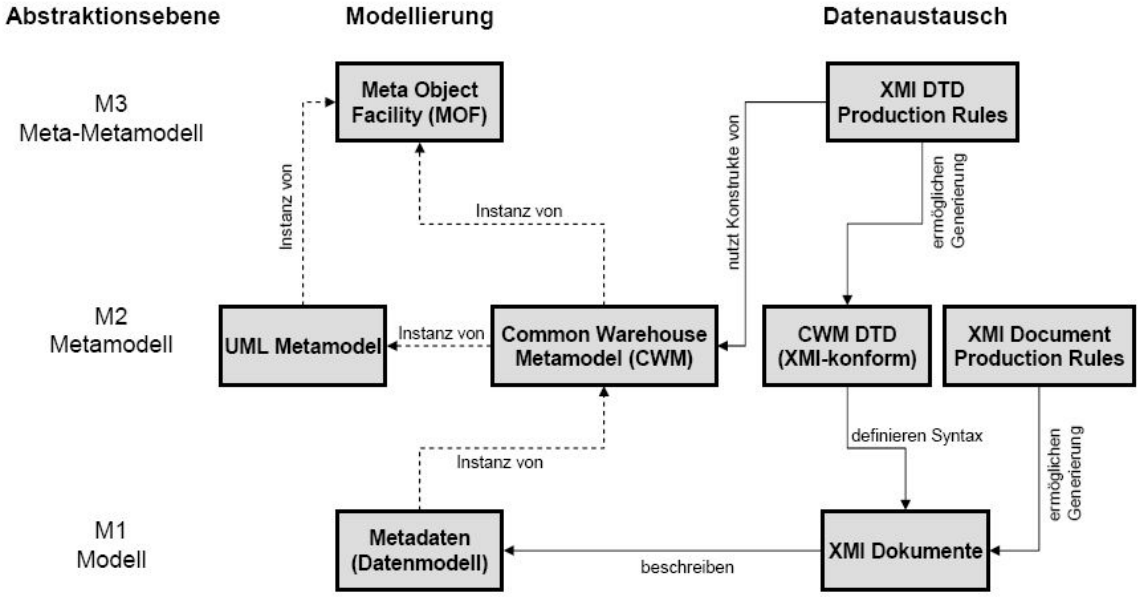

Abb. 18: CWM in der OMG Metamodell-Architektur (Quelle: HAHNE, 2005)

Das Common Warehouse Metamodel bietet eine umfangreiche Palette von generischen Klassen, die für die Beschreibung von Data Warehouse-Metadaten notwendig sind. Trotz dieser breiten Basis an generischen Klassen reicht der Standardumfang von CWM oft nicht aus, um ein Modell eines spezifischen Anwendungssystems genau zu beschreiben. Es wird eine erweiterte Informationsmenge benötigt, die über den Inhalt von CWM in seiner standardisierten Form hinausgeht. In diesen Fällen ist das CWM so zu erweitern, dass sämtliche von einem Anwendungssystem benötigte Metainformationen zur Verfügung stehen. Das OMG bietet in Ergänzung zum normativen Teil des CWM-Standards einige vordefinierte Erweiterungs-Pakete in Form des CWMX-Metamodells an. Darüber hinaus ist CWM auch spezifisch erweiterbar, wofür es grundsätzlich drei Erweiterungsmöglichkeiten gibt:

Tagged Values

Einem Modellierungselement (engl. model element) können beliebige Attributpaare, bestehend aus einem so genannten Tag und dem zugehörigen Wert, hinzugefügt werden. Dabei ist eine Erweiterung des normativen Metamodells nicht notwendig, da das Konstrukt der Tagged Values in CWM bereits explizit vorgesehen wurde. Die Interpretation erfolgt allerdings nicht im CWM, sondern muss zwischen den beteiligten Anwendungen definiert werden. Als Nachteil ist zu nennen, dass dieser Erweiterungstyp auf den Datentyp String beschränkt ist und dass keine Beziehungen zu anderen Modellelementen hergestellt werden können.

Stereotypen

Dieses Konzept wurde aus UML übernommen und ebenfalls fest in CWM integriert. Über Stereotypen können mehrere Tagged Values inhaltlich gruppiert werden. Der Nutzen von Stereotypen ist die bessere semantische Verständlichkeit einer Information für den Benutzer. Die Anzahl der zu einem Stereotypen gehörenden Tagged Values ist beliebig. Die bei den Tagged Values genannten Vor- und Nachteile gelten auch hier.

Erweiterungsklassen

Die mächtigste Möglichkeit um das CWM an die eigenen Bedürfnisse anzupassen ist die modellierte Erweiterung. Unter Anwendung von objektorientierten Techniken, wie zum Beispiel der Vererbung und dem Hinzufügen von Assoziationen, können die Klassen des Metamodells um neue Konstrukte erweitert werden.

Grundsätzlich stellt jede Erweiterung eine Abweichung vom anerkannten Standard dar, über welche sich die austauschenden Parteien gesondert verständigen müssen. Bei sämtlichen Erweiterungen des CWM muss gewährleistet sein, dass die teilhabenden Module Kenntnis über diese Zusatzinformationen erhalten. Nur so können die Daten, welche mit der Erweiterung beschrieben werden, verstanden und weiterverarbeitet werden (HAAG, 2004).

Teil B – Framework für Metadaten

3. Lösungsansatz

Die Idee des Lösungsansatzes besteht darin, dass die Komponenten einer Geodateninfrastruktur (vgl. *Abb. 3*) die für ihre Konfiguration benötigten Informationen an einer zentralen Stelle beziehen. Der Nutzen dieses Ansatzes liegt in der Wiederverwendbarkeit von System-Metadaten, welche redundanzfrei und konsistent an einer einzigen Stelle vorgehalten werden. Änderungen im Datenbestand werden einmalig vorgenommen und stehen in der Folge allen beteiligten GDI-Komponenten zur Verfügung. Dabei stellt sich vordergründig die Frage, ob es bereits Ansätze einer standardisierten Beschreibung von Metadaten zur Steuerung von Komponenten eines Informationssystems gibt. Im Idealfall sind alle beteiligten GDI-Komponenten in der Lage, eine standardisierte Beschreibung eines Metadatenmodells zu verstehen. Im Rahmen dieser Arbeit wird der Versuch unternommen, auf der Basis des Common Warehouse Metamodels (CWM) ein konzeptionelles Metamodell für die Integration von Metadaten als Grundlage für integrierte Anwendungssysteme zu erarbeiten.

3.1 Szenarien zur Integration von Geobasisdaten

Beim Aufbau der Datenbasis einer Geodateninfrastruktur sind unterschiedliche Integrations-Szenarien möglich; KELLER (2003) bezeichnet den ***Top-Down-Ansatz*** als Gesamtmodell und den ***Bottom-Up-Ansatz*** als Kernmodell, wobei nachträgliche Erweiterungen bei beiden Vorgehensweisen möglich sind.

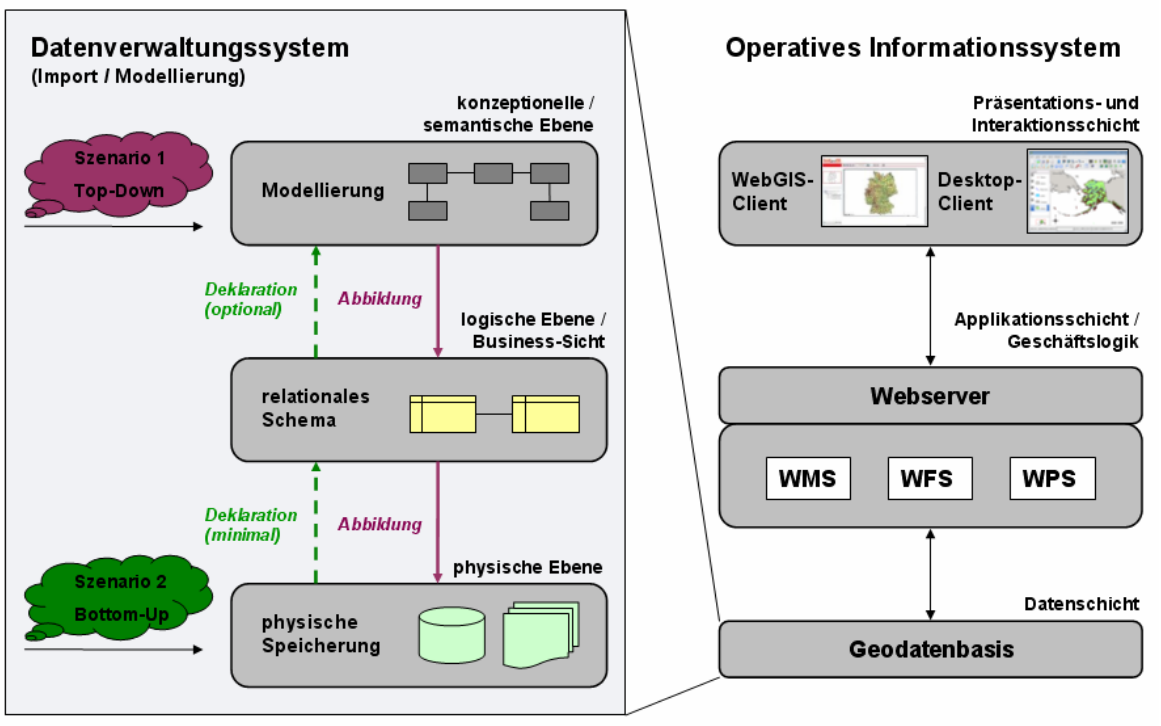

Abb. 19: Szenarien zur Integration von Geobasisdaten

Top-Down-Szenario zum Aufbau einer neuen Datensammlung

Dieses Vorgehen entspricht der Modellierung von Geodaten im herkömmlichen Sinne. Oft ist ein Problemumfeld zu beschreiben, für welches zu Beginn noch keine Datenbasis existiert. Die Modellierung erfolgt zunächst auf einer konzeptuellen bzw. semantischen Ebene, wobei immer von einem spezifischen Anwendungsfall bzw. Geschäftsmodell ausgegangen wird. Dieses abstrakte, anwendungsspezifische Modell der Realwelt wird schliesslich in ein physikalisches Datenschema abgebildet. Die dazwischen liegende logische Ebene repräsentiert die Benutzersicht. Nachdem die Daten physikalisch in einer Datenbank vorliegen, können Anwendungen entweder direkt oder via Geodienste darauf zugreifen. Da das Datenmodell globalen Charakter hat und in der Regel nur noch selten ändert, sind Anpassungen bei der Konfiguration der GDI-Komponenten relativ selten. Änderungen an den Daten selber haben auf die Konfiguration keinen Einfluss.

Bottom-Up-Szenario zur Inwertsetzung von bestehenden Geodaten

In der Praxis liegen umfangreiche Geodatensammlungen aus unterschiedlichen Quellen meist schon vor. Diese Datensätze weisen oftmals unterschiedliche Datenstrukturen auf, haben teilweise einen ungleichen Raumbezug und sind häufig nur schlecht oder gar nicht dokumentiert. Die Zielsetzung besteht vordergründig darin, die Daten von aussen direkt zugänglich zu machen und in Wert zu setzen. Ausgehend von der physikalischen Speicherung muss die Datenbasis in einer für den Benutzer verständlichen, logischen Datensicht aufbereitet und präsentiert werden. Die zugrunde liegende Datenstruktur ändert bei der Integration neuer Basisdaten laufend, was gleichzeitige Anpassungen in der Konfiguration der GDI-Komponenten erforderlich macht. Um den Aufwand hierbei möglichst gering zu halten und den effektiven Betrieb gewährleisten zu können, sollen alle Systemkomponenten die benötigten Metainformationen bei einer zentralen Stelle beziehen.

Der in dieser Thesis gewählte Lösungsansatz orientiert sich am Bottom-Up-Szenario. Dies erfordert folglich die Errichtung einer ***zentralen Metainformationskomponente*** innerhalb der Geodateninfrastruktur, bei der sich die anderen Komponenten zwecks ihrer Konfiguration über die Struktur der Geobasisdaten informieren können. Die Konzeption dieser Metainformationskomponente erfolgt hierbei auf einem übergeordneten Metalevel, der die Beschreibung konkreter Metadatenmodelle ermöglichen soll.

3.2 Metainformationsbasierte Systemintegration

Integration ist eines, wenn nicht sogar das zentrale Thema im Bereich der Informationslogistik. Mit dem Ziel einer möglichst vollständigen, unternehmensweiten Modellierung von Daten und Prozessen sind ***verschiedene Integrationsansätze*** entstanden. Die anfänglich monolithischen Integrationsmodelle für Gesamtarchitekturen (u.a. ZACHMANN, 1987) waren auf die Verknüpfung eng gekoppelter Komponenten zu einer logischen Einheit ausgerichtet. Mit dem Trend zur dezentralen Informationsverarbeitung fand jedoch ein grundlegendes Umdenken hin zur Entkopplung von Systemkomponenten bei gleichzeitiger Sicherstellung konsistenter Datenversorgung statt.

Der ***Data Warehouse-Ansatz*** hat sich im IT-Bereich zur Errichtung von logisch zentralen, dedizierten Informationssystemen etabliert. Dabei werden Applikationen über eine minimale Anzahl von Schnittstellen mit dem Integrationssystem verbunden. Oft werden bei der Integration von Softwaresystemen die Besonderheiten der Konzeption und der eingesetzten Instrumente herausgestellt, so dass mitunter das Verständnis für die konzeptionelle Ähnlichkeit der verschiedenen Integrationsszenarien und der angewandten Lösungsansätze verloren geht (VON MAUR et al., 2003).

KUTSCHE und RÖTTGERS (1999) schlagen ein modell- und ***metainformationsbasiertes Vorgehen*** in der Entwicklung von heterogenen verteilten Informationssystemen vor. Basierend auf der Verwendung von generischen Metainformationskomponenten können aufwendige Programmierarbeiten durch die einfache Spezifikation von Modellkorrespondenzen[40] und deren Eintragung in die entsprechende Metainformationskomponente ersetzt werden. Metainformation spielt laut BUSSE (1998) im Kontext föderierter Informationssysteme eine zentrale Rolle. So kann durch den expliziten Einsatz von Metainformation und der Dokumentation der Beziehungen zwischen autonomen Informationsstrukturen die Evolution heterogener, verteilter Informationssysteme in flexibler Art und Weise unterstützt werden. Zum einen entfällt eine aufwendige Schemaintegration aller existierenden Informationssysteme, zum anderen kann die Anzahl und Art der Einbindung der Basissysteme ohne grossen Aufwand geändert werden.

3.3 Anforderungen an System-Metadaten

Wie im Top-Down-Ansatz (vgl. Kap. 3.1) dargestellt wurde, kann im Idealfall für alle vorkommenden Daten ein globales Datenschema erstellt werden. Dabei erfolgt die Modellierung zunächst ausschliesslich auf einer konzeptionellen bzw. semantischen Ebene. Erst später resultiert eine plattformspezifische Umsetzung. Dieser Top-Down-Ansatz eignet sich insbesondere dann, wenn vorgängig noch keine Datenbasis existiert.

---

[40] *Beziehungen zwischen Informationsmodellen (in BUSSE, 2002)*

Beim Aufbau von Geodateninfrastrukturen bestehen meist zahlreiche Datenquellen, denen jeweils eine eigene Datenstruktur oder ein eigenes Datenschema zugrunde liegt. Dieses Bottom-Up-Vorgehen führt zwangsweise zu einem Data Warehouse-Ansatz; die für den Endanwender notwendigen Daten werden aus heterogenen Datenquellen extrahiert, zu einem gemeinsamen konsistenten Datenbestand zusammengeführt und in einer logischen Datensicht präsentiert. Die Komponenten einer GDI sollten dabei Kenntnis über die physische und logische Datenstruktur besitzen. Metadaten sind der Schlüssel zur Integration von Daten und Diensten und zur Steuerung von Komponenten in einer verteilten, heterogenen Systemlandschaft. Die in einem Metadaten-Repository explizit enthaltenen Informationen müssen möglichst allgemein definiert werden. Gemäss EBERLE (2006) ist ein generisches Modell zu erstellen, welches mit geringem Aufwand an die ständig ändernden Gegebenheiten im GDI-Umfeld angepasst werden kann. Es beschreibt Datensammlungen in einer Art und Weise, so dass diese sowohl für den menschlichen Benutzer wie auch für die automatisierte Auswertung erschlossen werden können. Neue Datenbestände sollen dabei ohne Programmierung, sondern allein durch Konfiguration verfügbar gemacht werden. Die Parameter für den physischen Datenzugriff beziehen die einzelnen Komponenten aus der zentralen Metadatenbank, wobei der Endbenutzer vom Anwendungssystem eine in der Metadatenbank ausgestaltete logische Datensicht präsentiert bekommt.

EBERLE (2006) gliedert die Metadaten-Anforderungen im GDI-Umfeld in fünf Bereiche:

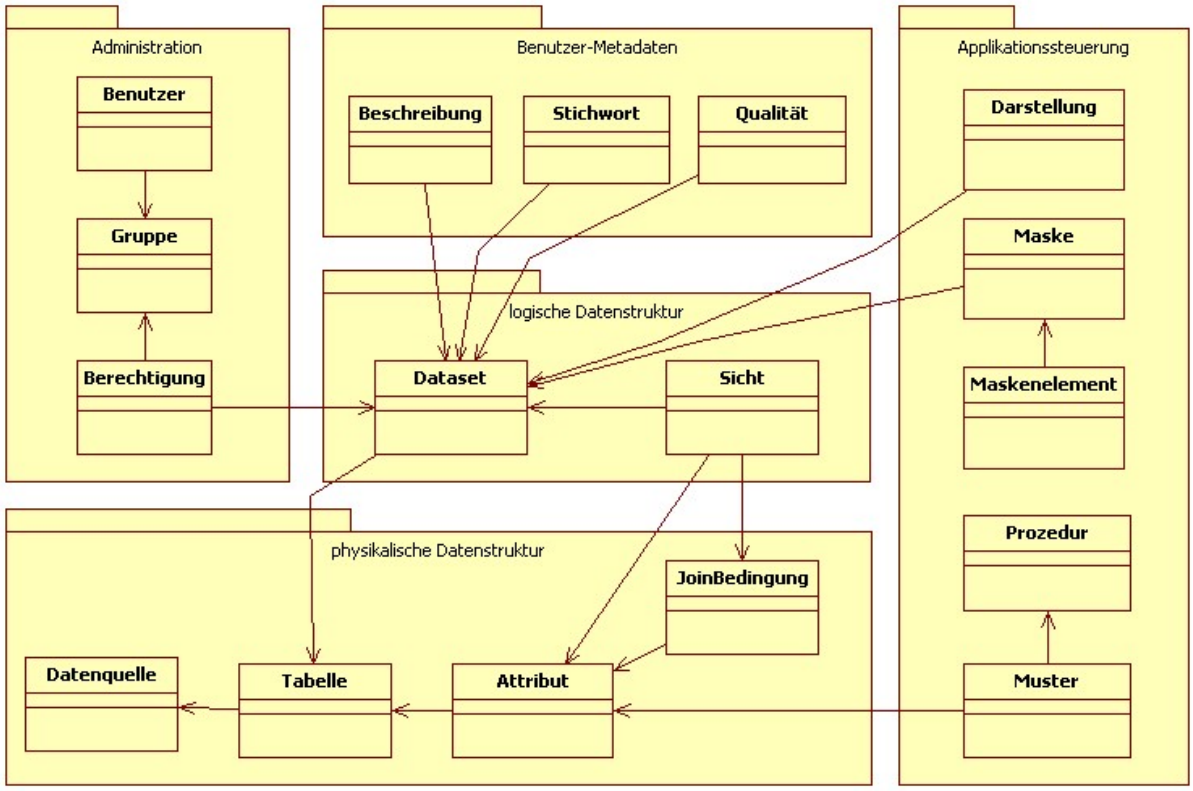

Abb. 20: Objektmodell für Metadaten nach EBERLE (2006)

Administration:	Verwaltung von Benutzern und Berechtigungen
Benutzer-Metadaten:	Beschreibung von Geodaten (vgl. Geo-Metadaten)
Physikalische Datenstruktur:	Beschreibung der physischen Datenressourcen
Logische Datenstruktur:	Logische Benutzersicht auf die physische Datenstruktur
Applikationssteuerung:	Informationen zur Automatisierung von applikationsspezifischen Vorgängen wie die Kartendarstellung, die Erstellung von Editiermasken oder die Anwendung von Auswertungsprozeduren auf bestimmte Muster in den Datenstrukturen.

Obwohl der Fokus dieser Arbeit auf technischen System-Metadaten zur Steuerung von Komponenten und Applikationen liegt, wird der Bereich „Benutzer-Metadaten“ (=Geo-Metadaten) der Vollständigkeit halber in die Konzeption einbezogen. Gemäss den Zielen dieser Master Thesis wird jedoch vorwiegend auf den Bereich der System-Metadaten (physikalische und logische Datenstruktur, Klassifikation und Kartendarstellung) Bezug genommen.

3.4 Analyse des Common Warehouse Metamodels

Die Konzeption eines Metadaten-Repositories soll sich nach Möglichkeit auf bestehende IT-Standards stützen. Da sich die Integration von Geodaten in der Praxis oft am Data Warehouse-Ansatz orientiert, liegt die Anlehnung an das Common Warehouse Metamodel (CWM) nahe. Der Schwerpunkt dieser Arbeit besteht darin, dessen Eignung zur Beschreibung und Verwaltung von System-Metadaten in einer Geodateninfrastruktur zu untersuchen. In diesem Abschnitt folgt deshalb eine Analyse über den Aufbau und den Inhalt des Common Warehouse Metamodels.

Das Referenzmodell basiert auf einem UML-Modell mit über 200 Klassen und mehr als 150 Assoziationen. Der Fokus liegt dabei auf technischen Data Warehouse-Metadaten und dem Austausch von Metadaten zwischen verschiedenen Softwarekomponenten einer Data Warehouse-Architektur. Zwecks einer besseren Übersichtlichkeit sind die Modellelemente thematisch in 21 Pakete unterteilt. Jedes dieser Pakete ist wiederum auf einer von fünf hierarchischen Ebenen angeordnet. Sämtliche Pakete sind vom *Core*-Paket des *Object Models* abhängig. Das CWM wurde ausgelegt um die Wiederverwendbarkeit des Objektmodells und gebräuchlicher Modellierungskonstrukte zu maximieren. Als anschauliches Beispiel sei hier erwähnt, dass das *Object Model* zur Beschreibung von objektorientierten Ressourcen direkt wiederverwendet wird. In den zahlreichen Paketen sind im Sinne einer gemeinsamen Basis für allgemeine Metamodelle verschiedenster Data Warehouse-Systeme nahezu alle allgemein relevanten Aspekte abgedeckt. Pakete, welche für einen spezifischen Anwendungsfall nicht relevant sind, können weggelassen werden. So benötigt ein Anwender, welcher ein relationales Schema beschreiben möchte, aus dem unten gezeigten Schichtenmodell nur die Pakete *Core* und *Relationships* aus der Objektmodell-Ebene sowie *Relational* aus der Ressourcen-Schicht.

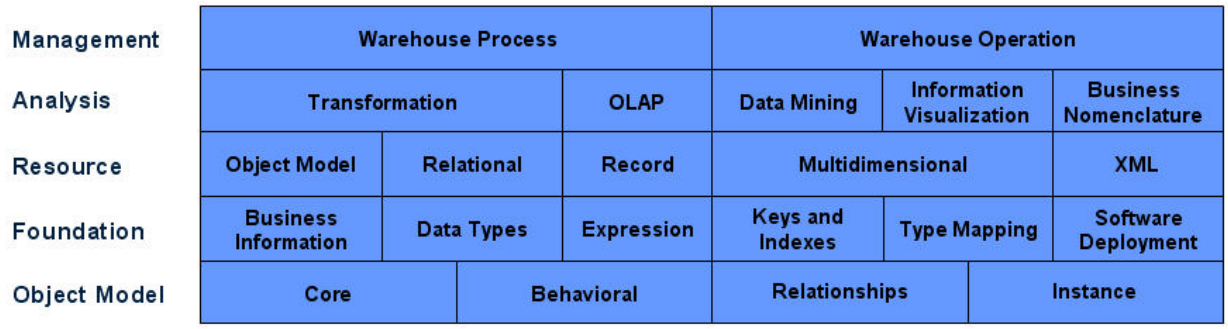

Abb. 21: Pakete zur Strukturierung der Metamodelle im CWM

Das *Object Model* bildet die Basis aller darüber liegenden Schichten und Metamodelle. Es besteht aus den Paketen *Core*, *Behavioral*, *Relationships* und *Instance*. Das *Object Model* definiert eine Sammlung von grundlegenden Konstrukten zur Erstellung und Beschreibung von Metamodell-Klassen der anderen CWM-Pakete. Es umfasst einen Ausschnitt der UML, wobei es sich auf Merkmale beschränkt, die für die Erstellung und Beschreibung des CWM notwendig sind. Das *Core*-Paket ist von sämtlichen CWM-Paketen unabhängig. Die Pakete *Behavioral*, *Relationships* und *Instance* sind ausschliesslich von *Core* abhängig; es bestehen keine gegenseitigen Abhängigkeiten. Das *Object Model* als eine Verallgemeinerung der physischen Datenstruktur repräsentiert die logische Ebene einer Data Warehouse-Architektur.

Die Grundlagen-Ebene (*Foundation*) enthält Metamodelle für die Spezifikation von Grundbausteinen wie Datentypen, Ausdrücke, Schlüssel und Indexierungs-Konstrukte ebenso wie Elemente zur Modellierung von Geschäftsinformationen oder von Softwaresystem-Architekturen. Ziel dieser Modellebene ist die Bereitstellung einer gemeinsamen Grundlage, welche durch darüber liegende Pakete genutzt und bei Bedarf spezialisiert bzw. erweitert werden kann. Alle Metamodelle dieser Schicht sind voneinander unabhängig und können in beliebiger Weise kombiniert bzw. weggelassen werden.

Die Metamodelle der Ressourcen-Ebene (*Resources*) eignen sich zur Beschreibung von Ressourcen in konkreten physischen Datenstrukturen. Es existieren konkrete Pakete zur Repräsentation von recordbasierten, objektorientierten, relationalen, multidimensionalen oder XML-Datenstrukturen. Das Paket *Relational* dient der Beschreibung von relationalen Datenbankschemata, wobei SQL99-konforme relationale Datenbanken inklusive der objektorientierten Erweiterungen unterstützt werden. *Relational* ist innerhalb des CWM abhängig von den Paketen *DataTypes* und *KeyIndexes* aus der *Foundation*-Ebene.

Die Analyse-Schicht (*Analysis*) befasst sich mit der Analyse und Bewertung von Daten eines Data Warehouses. Sie stellt Konstrukte für Datentransformationsprozesse (*Transformation*), die multidimensionale Datenanalyse (*OLAP* und *Data Mining*) und die Visualisierung (*Information Visualization*) bereit. Auf der *Management*-Schicht werden auf hoher Ebene die Abläufe im Data Warehouse beschrieben. Sie enthält Elemente, die zur Steuerung und Überwachung der technischen Transformationsprozesse benötigt werden und erlaubt damit die Festlegung, wann welche Transformationen durchgeführt werden müssen, um bestimmte Analyseszenarien zu erzeugen.

Jede CWM-Klasse fügt sich in eine Vererbungshierarchie ein, durch die das Referenzmodell einen hohen Integrationsgrad erreicht. Abhängigkeiten bestehen nur zwischen Paketen derselben oder darüber liegenden Schichten. Jede CWM-Klasse geht aus einer Spezialisierung der Klasse *Element* des *Core*-Pakets hervor. Sämtliche Metadaten aus dem CWM werden in einem Metadaten-Repository zentral gespeichert und verwaltet. Mit dieser zentralen Speicherung soll die Mehrfachverwendung von Metadaten in den verschiedenen Teilen des Data Warehouse-Prozesses erreicht werden. Erweiterungen des CWM werden ebenfalls im Repository abgelegt.

3.5 Eignung des CWM im GDI-Umfeld

Das Common Warehouse Metamodel ist ein umfangreiches und mächtiges Werkzeug zur Modellierung und -verwaltung von Metadaten. Wesentliche Schwachpunkte des CWM sind gemäss HAHNE (2005) das Fehlen von Metamodellen in den Bereichen der Versionierung und Berechtigungsverwaltung, der Semantik von Data Warehouse-Inhalten, dem Qualitätsmanagement, des Berichtswesens und der Unterstützung organisatorischer Prozesse. Das CWM ist konsequent auf grundlegende Attribute beschränkt und weist eine niedrige Definitionstiefe des Referenzmodells aus (MELCHERT et al., 2003).

In diesem Abschnitt wird hinsichtlich ihrer Eignung auf diejenigen Pakete des CWM Bezug genommen, welche für die Abbildung der Anforderungen an ein Metadatenmodell einer Geodateninfrastruktur als geeignet identifiziert wurden.

Physikalische und logische Datenstruktur

Zur Beschreibung der logischen Datenstruktur eignen sich die Pakete *Core* und *Relationships* aus dem *Object Model*. Darin werden auf abstrakter Ebene Konstrukte zur Modellierung objektorientierter Systeme definiert. Das Paket *Core* spezifiziert abstrakte Klassen an der Spitze der Vererbungshierarchie und stellt neben primitiven Datentypen wie Boolean, String oder Integer auch grundlegende Strukturierungskonstrukte bereit. Das Paket *Relationships* definiert die Arten von Beziehungen, die zwischen Modellierungselementen bestehen können.

Das Erstellen eines Metadatenframeworks als Grundlage eines Metadaten-Repositories ist in dieser Master Thesis auf relationale Datenquellen ausgerichtet. Das Paket *Relational* wird diesem Anspruch gerecht und unterstützt dabei den SQL99-Standard. Mit den Klassen *Catalog* und *Schema* können Datenbanksysteme, bestehend aus Tabellen, Prozeduren und Triggern, abgebildet werden. Die im *Core*-Paket abstrakt definierten Beziehungen nehmen hier konkrete Gestalt an. Im Weiteren werden Klassen aus *Foundation* spezialisiert, um Primär- und Fremdschlüssel entsprechend den Anforderungen der relationalen Modellierung verwenden zu können.

Benutzer-Metadaten und Administration

Mit den Paketen *BusinessInformation* (aus *Foundation*) und *BusinessNomenclature* (aus *Analysis*) stellt das CWM ansatzweise Konstrukte bereit, die auf den Bereich der Benutzer-Metadaten anwendbar sind (siehe BERNARD et al., 2001). *BusinessInformation* enthält beschreibende Informationen zu verantwortlichen Stellen, Kontaktadressen (z.B. Email oder Telefon), Kurzbeschreibungen sowie Verweise auf Dokumente oder Ressourcen. Mit dem Paket *BusinessNomenclature* können Taxonomien und Glossare zur Erläuterung von Begriffen (Nomenklatur / Semantik / Ontologien) angelegt werden. Um Metadaten nach ISO 19115 oder Dublin Core zu beschreiben, sind Erweiterungen notwendig. Die Verwaltung von Benutzerrechten ist in CWM nicht vorgesehen.

Visualisierung und Klassifikation

Mit dem Paket *InformationVisualization* wird im CWM ein äusserst generischer Ansatz zur Beschreibung der graphischen Darstellung von Daten verfolgt. Da die zu visualisierenden Informationen sehr unterschiedlich sind, beschränkt sich dieses Paket auf die Definition einiger containerartiger Konstrukte, welche komplexe Visualisierungsmechanismen enthalten bzw. referenzieren. Einzig die Klasse *XSLRendering*[41] nimmt auf eine konkrete Technologie Bezug.

[41] *XSL (Extensible Stylesheet Language): Sprache zur Erzeugung von Layouts für XML-Dokumente*

3.6 Anwendung konkreter GDI-Komponenten am CWM

Leistungsfähige GIS-Komponenten als Bausteine von Geodateninfrastrukturen sind in zunehmender Anzahl frei verfügbar. Es ist nicht das Ziel dieser Arbeit, eine umfassende Übersicht über verfügbare Werkzeuge und deren Leistungsmerkmale zu vermitteln. Im Rahmen dieser Master Thesis wurden einige repräsentative FOSS4G[42]-Komponenten aufgrund ihrer Eignung und Verbreitung ausgewählt, installiert und konfiguriert (vgl. Anhang).

Installierte GDI-Komponenten (unter Windows):

- PostgreSQL 8.1.5 mit PostGIS 1.1.6
- UMN MapServer 1.6.0 unter ms4w 1.6.0 mit PHP 4
- deegree WMS 2 unter Apache Tomcat 5.5.20
- MapBender 2.4 unter ms4w 1.6.0 mit PHP 4
- Quantum GIS 0.8.0 (pre-release), OpenJUMP 1.0.1 und uDig 1.0.6

Verwendete CWM-Pakete:

Als Resultat der CWM-Analyse wurden diejenigen Pakete identifiziert, welche hinsichtlich der spezifischen GDI-Anforderungen als wesentlich angesehen werden:

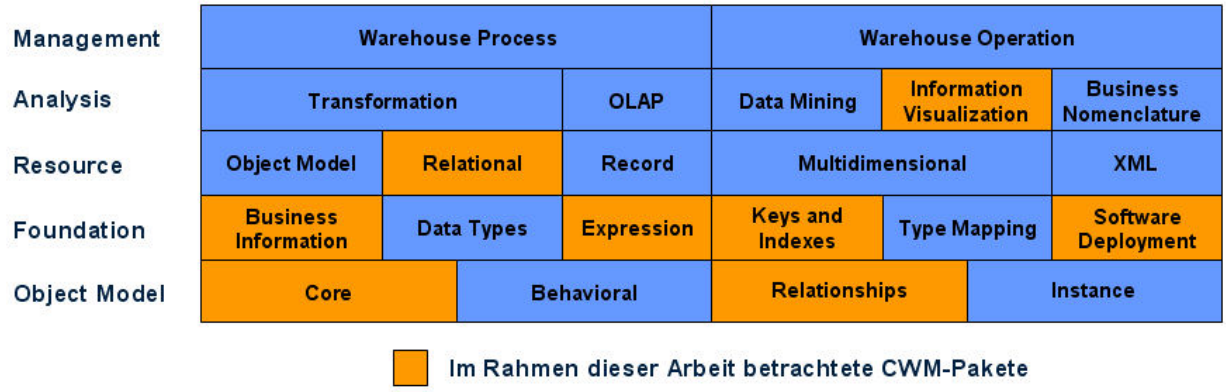

Abb. 22: Identifikation geeigneter Pakete als Ergebnis der CWM-Analyse

Mapping der installierten GDI-Komponenten auf die verwendeten CWM-Pakete:

Die Frontend-Steuerung kann für die anschliessenden Betrachtungen vernachlässigt werden. Es wird davon ausgegangen, dass sich alle Services der Middleware-Ebene über ihre *Capabilities* selber beschreiben. Benutzeroberflächen von Clients können dank diesen Service-Metadaten dynamisch gesteuert werden. Dies setzt voraus, dass die entsprechenden Dienstschnittstellen durch das Frontend unterstützt werden. Im CWM werden sämtliche Clients im Paket *SoftwareDeployment* durch die Klasse *DataProvider* abgebildet.

[42] *FOSS4G: Free and Open Source Software for Geoinformatics*

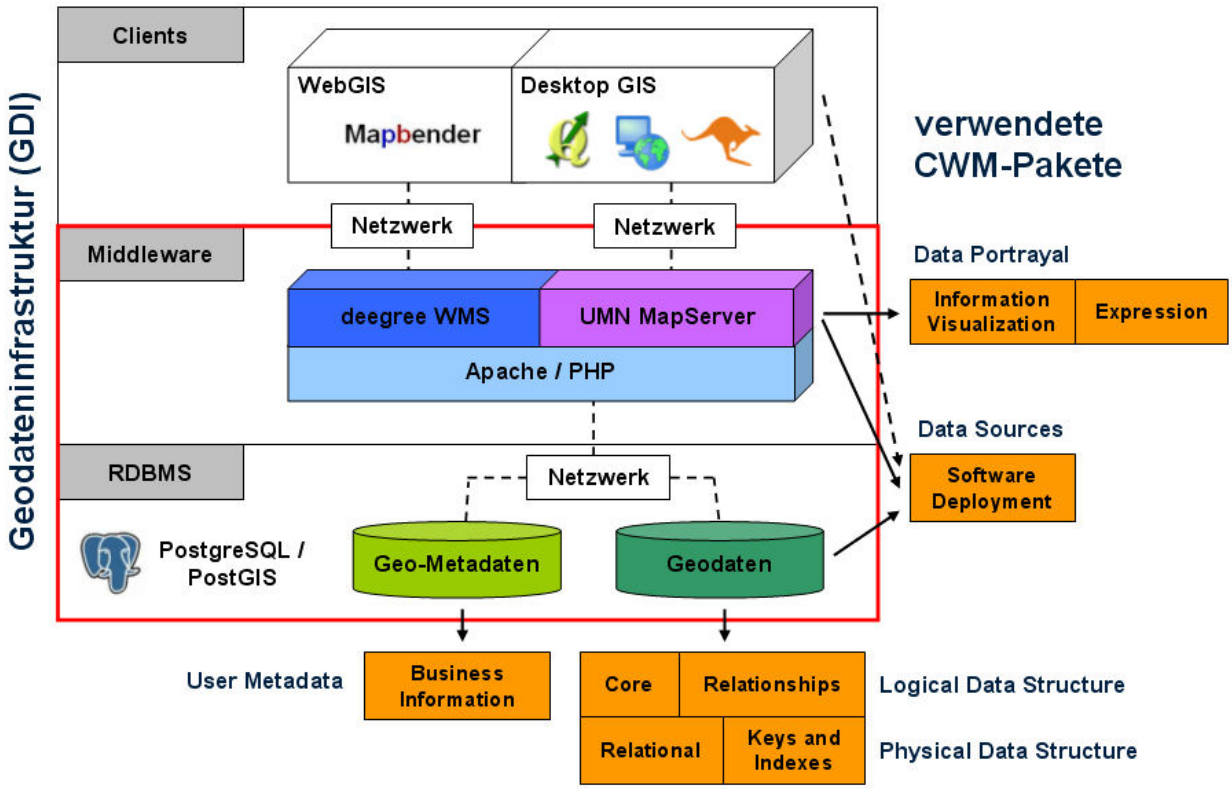

Abb. 23: Anwendung konkreter GDI-Komponenten am CWM

Auf der Middleware-Ebene befinden sich diejenigen GDI-Komponenten, welche den Zugriff auf die Geobasisdaten ermöglichen und die Darstellung der zurückgelieferten Daten durch Konfiguration festlegen. Die Selektion und Klassifikation der abgefragten Daten wird im CWM über Ausdrücke (*Expression*) gesteuert. Die Darstellung der gerenderten Objekte ist auf generische Art und Weise im Paket *InformationVisualization* definiert. Die Services der Middleware-Ebene können zugleich als Clients betrachtet werden, die in ihrer Funktion als *DataProvider* den Zugriff auf einen Server herstellen.

Das darunter liegende Datenbank-Verwaltungssystem wird im Paket *SoftwareDeployment* durch die Klasse *DataManager* repräsentiert. Ein *DataManager* fungiert als Server und verwaltet somit den Zugriff auf die in der Datenbank gespeicherten Daten. Ein *DataManager* kann aber auch ein Fileverwaltungssystem darstellen, durch welches der Zugriff auf lokale oder im Netzwerk verteilte Dateien erfolgen kann. Die physische Struktur (Tabellen, Views, etc.) der Datenbank kann im CWM mit den Paketen *Relational* und *KeyIndexes* abgebildet werden. Die Abbildung der logischen Struktur ist mit den Paketen *Core* und *Relationships* möglich. Das Paket *BusinessInformation* von CWM beschreibt Geo-Metadaten im klassischen Sinne. Diese helfen dem Anwender beim Auffinden von geeigneten Ressourcen.

3.7 Modellierung von Metadaten

In der Informatik werden Daten in Modellen abgebildet, welche unabhängig von der späteren Implementierung sein sollen. Ein Datenmodell stellt gespeicherte Daten, ihre innere Struktur und ihre Beziehungen untereinander dar. Es ist die vereinfachte, abstrahierte Abbildung eines für die jeweilige Fragestellung bedeutsamen Teilaspekts der komplexen Realwelt. Es bezieht sich immer auf einen konkreten Anwendungszweck (wozu, für wen) in einem spezifischen Problemumfeld (wovon).

Im Kontext der Datenmodellierung dient das konzeptionelle (auch konzeptuelle, semantische) Schema der formalen Beschreibung der Daten und deren Beziehungen untereinander. Es ist logisch von einer bestimmten Anwendung und physisch von einem bestimmten Datenbanksystem unabhängig. Aus einem konzeptionellen Modell als formale Beschreibung der Realwelt und ihrer Objekte resultieren ein datenbankorientiertes, logisches Schema und schliesslich die physische Speicherorganisation. Das logische Schema ist nach KELLER (2003) die Implementierung des konzeptionellen Datenmodells in einer Datenbank oder einem Informationssystem mittels einer konkreten Methode (z.B. relational) und einer produktspezifischen Sprache (z.B. Oracle-SQL).

Die konzeptionelle Ebene der Modellierung ist von besonderer Bedeutung, da sie das Glied zwischen einer informellen Problembeschreibung und der informatikbetonten Systementwicklung bildet. Graphische Beschreibungen erleichtern das Verständnis über komplexe Zusammenhänge des jeweiligen Problemumfelds. Aus der konzeptionellen Modellierung sind verschiedene Paradigmen und Ausdrucksmittel hervorgegangen. In der objektorientierten Modellierung hat sich die Unified Modeling Language (UML) durchgesetzt, während sich Entity-Relationship-Diagramme (ERM) in der relationalen Modellierung etabliert haben. Wie die folgende Tabelle zeigt, bestehen zwischen den unterschiedlichen Modellierungsansätzen grundsätzliche Begriffs-Parallelen:

Datenbank	***ERM***	***UML***
Tabelle (table)	Entitätstyp (entity type)	Klasse (class)
Datensatz, Zeile (record, row)	Entität (entity)	Objekt (object)
Feld, Spalte (field, column)	Attribut (attribute)	Attribut (attribute)
Fremdschlüssel (foreign key)	Relation (relationship)	Assoziation (association)

Tab. 1: Gegenüberstellung von Begriffen der unterschiedlichen Modellierungsansätze

3.7.1 Objektorientierte Modellierung mit UML

Eine objektorientierte Herangehensweise ist für den Entwurf föderierter Informationssysteme besonders geeignet (BUSSE, 1998), da sie die notwendige Diskussion zwischen Anwender und Softwareentwickler fördert. Damit unterstützt sie die Verknüpfung struktureller und dynamischer Aspekte sowie fachspezifischer und generischer Anteile. Beim objektorientierten Entwurf werden Objekte gleicher Ausprägung durch Klassen abgebildet. Eine Klasse beschreibt die Struktur und das Verhalten der zugehörigen Objektinstanzen anhand von Attributen und Operationen. Da im Gegensatz zum Datenmodell der relationalen Modellierung auch das Verhalten definiert werden kann, spricht man bei der objektorientierten Modellierung von Objektmodellen.

Die Unified Modeling Language (UML) als graphische Sprache zur Modellierung verteilter Objektsysteme wurde im November 1997 von der Object Management Group (OMG) offiziell als Standard verabschiedet. Sie findet heute breite Anwendung bei der Modellierung von objektorientierten Softwaresystemen, Geschäftsprozessen und Geodaten. UML bietet Konstrukte zur Modellierung sowohl statischer als auch dynamischer Systemaspekte. Die UML gilt in der konzeptionellen Modellierung teilweise als umstritten, da sich ihre Konzepte semantisch auf Komponenten der Softwareentwicklung beziehen. Gerade im Bereich der Standards entspricht die objektorientierte Modellierung jedoch dem meist verwendeten konzeptuellen Formalismus. So definiert beispielsweise die ISO TS 19103:2005 („Geographic information - Conceptual schema language“) ein Profil von UML mit Regeln und Richtlinien zur Anwendung einer konzeptuellen Schemasprache für Geoinformation. Für die Verwendung von UML spricht insbesondere auch die gute Unterstützung durch eine Vielzahl von Software-Werkzeugen. Im Rahmen dieser Arbeit wurde der Enterprise Architect von Sparx Systems verwendet. Dieses Hilfsmittels erlaubt die Erstellung von konzeptionellen, logischen und physischen Modellen, ermöglicht die Generierung von DDL-Statements und bietet Schnittstellen für den XMI Import und Export.

3.7.2 UML-Klassendiagramme

Das im Rahmen dieser Arbeit erarbeitete Metamodell wird auf der konzeptuellen Ebene als Objektmodell in Form von UML-Klassendiagrammen abgebildet. Ein Klassendiagramm ermöglicht eine statische Sicht auf ein System. Es enthält Klassen und deren Beziehungen, welche in der UML-Terminologie als Assoziation bezeichnet werden. Von einer generischen Superklasse können durch Vererbung Subklassen abgeleitet und einmal definierte Strukturen und Operationen wiederverwendet werden. In einer Subklasse können zusätzliche Attribute und Operationen hinzugefügt werden. Dieses Prinzip der Vererbung entspricht je nach Sichtweise einer Generalisierung oder Spezialisierung. Mittels Aggregation bzw. Komposition können zudem Teile zu einem Ganzen zusammengefügt werden. Bei der Komposition ist ein Teil abhängig von der Existenz des Ganzen. Auf die weiteren Konzepte der objektorientierten Modellierung wie Kapselung und Polymorphismus wird an dieser Stelle nicht eingegangen.

Übersicht UML-Notation

Abb. 24 liefert eine Übersicht zu den in dieser Master Thesis verwendeten UML-Konstrukten:

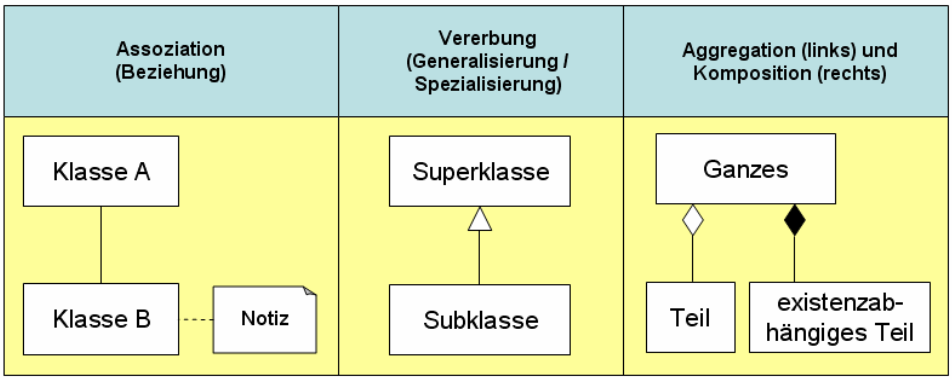

Abb. 24: Grundkonstrukte von UML-Klassendiagrammen

Wie *Abb. 25* zeigt, wird eine Assoziation durch Rollen und Multiplizitäten genauer beschrieben:

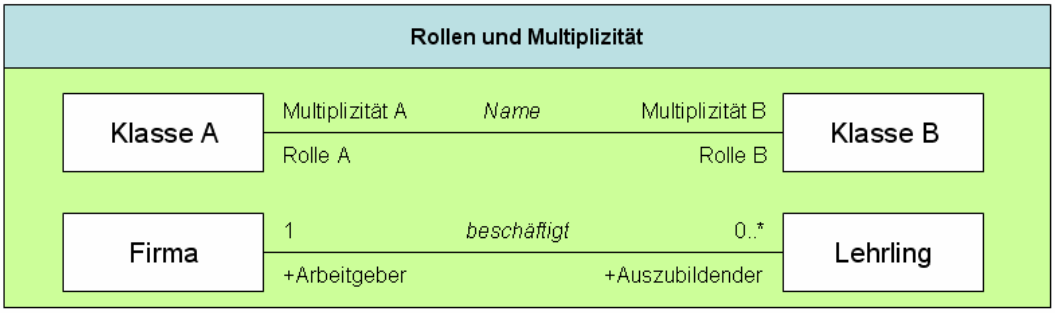

Abb. 25: Assoziation mit Namen, Rollen und Multiplizitäten

Der Name einer Assoziation wird durch ein Verb - in Leserichtung von links nach rechts oder von oben nach unten - ausgedrückt. Die Rolle ist eine Präzisierung der Beziehung zwischen den beiden Klassen. Einer Klasse wird eine Multiplizität mit einer unteren und einer oberen Grenze zugeordnet.

Lesebeispiel:

Eine Firma beschäftigt Lehrlinge. Die Firma ist Arbeitgeber von keinem, einem oder mehreren Lehrlingen. Ein Lehrling ist Auszubildender bei genau einer Firma.

Multiplizitäten (Kardinalität):

Die Multiplizität einer Assoziation macht Angaben über die Anzahl gleichartiger Objekte, zwischen denen ein Zusammenhang besteht. Gemeint ist damit die Anzahl Objekte der Klasse B (bzw. A), die einem Objekt der Klasse A (bzw. B) durch die Beziehung zwischen den Klassen A und B zugeordnet werden können[43].

UML-Multiplizität	***Bedeutung***
1	genau ein (default)
2	genau zwei (usw.)
1..3	von 1 bis und mit 3
3,5	3 oder 5
1..*	ein oder mehrere
0..* oder kurz *	0, 1 oder mehrere
0..1	ein oder kein (optional)

Tab. 2: Multiplizitäten in UML

[43] *Quelle: http://www.integis.ch/glossary_oo_uml_ili2_de.html (25.06.2007)*

4. GDI-Metamodell

In diesem Kapitel wird die Konzeption des Metamodells zur Konfiguration und Steuerung von Komponenten einer Geodateninfrastruktur detailliert beschrieben. Die Arbeit orientiert sich am Common Warehouse Metamodel (CWM). Das resultierende UML-Objektmodell stellt eine konzeptionelle Grundlage zur Realisierung eines zentralen Metadaten-Repositories einer Geodateninfrastruktur dar. Als Einführung in die Beschreibung der verschiedenen Teilmodelle folgen zunächst einige Hinweise zur angewandten Methodik und Notation. In Kapitel 4.3 wird schliesslich das CWM-basierte Objektmodell für GDI-Metadaten vorgestellt, wobei hierzu die von der OMG (2003) herausgegebene Spezifikation herangezogen wird. Um die spezifischen GDI-Anforderungen zu befriedigen, greift das Konzept auf die im CWM enthaltenen Zugangspunkte für modellierte Erweiterungen zurück.

4.1 Hinweise bezüglich Methodik und Notation

Es werden nur die für diese Arbeit relevanten Teile des Standardumfangs des CWM berücksichtigt. Die Einhaltung der Standardkonformität ist nicht oberste Maxime. Dort wo es sinnvoll erscheint, werden Abweichungen zum CWM in Kauf genommen und notwendige Erweiterungen vorgenommen. Zur Reduktion der Komplexität und zugunsten einer besseren Verständlichkeit werden flache Vererbungshierarchien angestrebt. Dies setzt voraus, dass eine Superklasse nicht von zu vielen Subklassen benötigt wird.

Um die Übersichtlichkeit des GDI-Metamodells zu verbessern, erfolgt eine Gliederung in mehrere Teilmodelle. Bei der graphischen Darstellung werden die einzelnen Klassen farblich aufgrund ihrer Zugehörigkeit zu den Teilmodellen unterschieden. Die Zuordnung ist in der folgenden Tabelle ersichtlich:

Farbe	***Metamodell***	***Inhalt***
	User Metadata	Benutzer-Metadaten
	Physical Data Structure	Physische Datenstruktur
	Logical Data Structure	Logische Datenstruktur
	Data Sources	Zugriff auf Datenquellen
	Data Portrayal	Visualisierung und Klassifizierung
	Key Relationships	Supportklassen (relationale Beziehungen)
	Object Relationships	Supportklassen (Objektbeziehungen)

Tab. 3: Inhalt der einzelnen GDI-Metamodelle

Modellierte Erweiterungen zu CWM werden in rot dargestellt. In den nachfolgenden Modellbeschreibungen sind CWM-Pakete durch eine Umrandung und CWM-Klassen bzw. CWM-Attribute durch *Kursivschrift* gekennzeichnet.

In Anlehnung an das CWM werden im GDI-Metamodell englische Bezeichnungen verwendet. Die in dieser Arbeit verwendete Notation folgt den Regeln und Empfehlungen von CHONOLES et al. (2003):

- ***Klassennamen*** sind zusammengesetzte Nomen beginnend mit einem Grossbuchstaben (z.B. *DataManager*).
- ***Attribute*** sind zusammengesetzte Nomen bzw. Adjektive, beginnend mit einem Kleinbuchstaben (z.B. *isCaseSensitive*).
- ***Rollen*** werden durch zusammengesetzte Nomen, beginnend mit einem Kleinbuchstaben (z.B. *referencedTable*) gebildet.
- Bei einem zusammengesetzten Bezeichner sind keine Leerschläge enthalten. Die nachfolgenden Nomen bzw. Adjektive beginnen immer mit einem Grossbuchstaben

Einige Assoziationen sind durch ein vorangestelltes Slash („/“) gekennzeichnet. Damit wird ausgedrückt, dass eine Beziehung von einer übergeordneten Assoziation abgeleitet bzw. geerbt wurde. Die explizite Darstellung dieser Assoziationen dient ausschliesslich einer besseren Verständlichkeit der bestehenden Beziehungen.

4.2 Abgrenzung bezüglich Benutzerrechten

Die Verwaltung von Benutzergruppen und Berechtigungen wurde bisher in CWM nicht berücksichtigt. Auch in dieser Arbeit wird der Aspekt der Authentifizierung und Autorisierung von Systembenutzern in der Modellierung bewusst ausgeklammert.

Über Metadaten werden grundsätzlich sämtliche Inhalte eines Systems beschrieben. Ob jemand Zugriff auf diese Informationen erhält ist durch vorgängige Authentifizierung zu regeln. Es wird davon ausgegangen, dass ein Benutzer hierzu einen Berechtigungsnachweis durch die Angabe seiner Credentials selber erbringen muss. An einem der GDI vorgelagerten Policy Enforcement Point (PEP) werden die Credentials für den Systemzugriff geprüft. Eine Erweiterung um Authentifizierungsinformationen darf bei einer Geodateninfrastruktur nur dann erfolgen, wenn dieser Teil des Metadatenmodells nicht nach aussen gezeigt wird.

Wie der Zugriff auf die Daten aussieht (nur lesend oder auch schreibend) wird durch die Autorisierung festgelegt. Es ist zu beachten, dass ein Metadaten-Repository durchaus Daten enthalten könnte, die gar nicht zugänglich sind. Daher sollte ein System so konzipiert sein, dass Metadaten immer erst nach der Authentifizierung übermittelt werden. Bei einem Metadatensystem zur Steuerung von GDI-Komponenten muss jedoch jedes System alles sehen und auf alles zugreifen können und erst ein übergeordneter Security Layer sorgt sich um die Autorisierung der Enduser.

4.3 CWM-basiertes Objektmodell für GDI-Metadaten

In diesem Kapitel wird das GDI-Metamodell anhand der einzelnen Teilmodelle beschrieben. Die Art der Modellierung richtet sich nach den in Kapitel 4.1 festgelegten Konventionen. Inhaltliche und konzeptionelle Grundlage dieser Arbeit ist die Spezifikation des Common Warehouse Metamodels (OMG, 2003). Das gesamte GDI-Metamodell umfasst 8 Pakete, wovon 7 den gleichnamigen CWM-Paketen inhaltlich vollumfänglich entsprechen, sowie ein weiteres Paket (ModeledExtensions), welches die explizit modellierten Erweiterungen enthält.

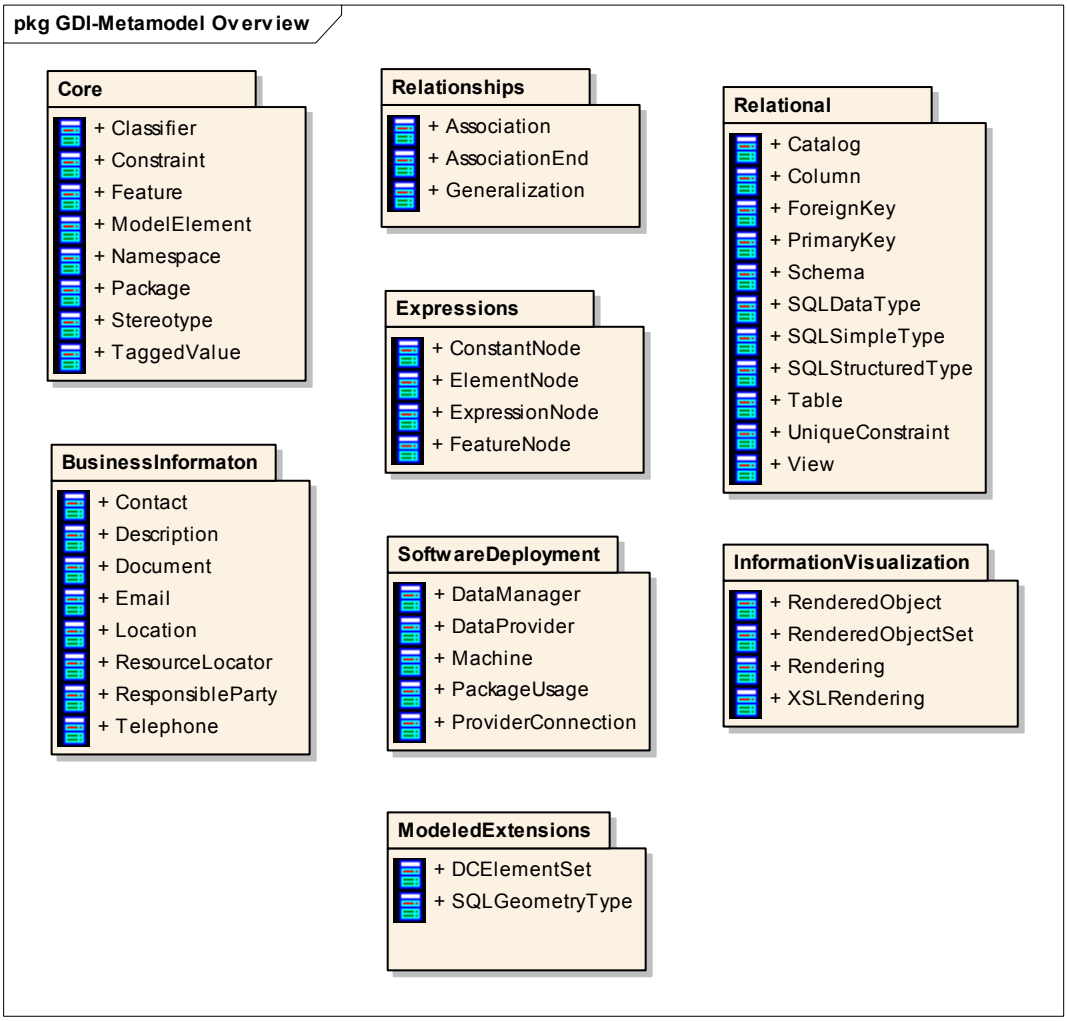

Abb. 26: Übersicht der Pakete und Klassen des GDI-Metamodells

4.3.1 Metamodell: Logische Datenstruktur

Als Grundlage der Logical Data Structure wird das Core-Metamodell des CWM-Object Models herangezogen. Das Core-Paket enthält grundlegende Konstrukte zur Erstellung und Beschreibung von Metamodell-Klassen. Es eignet sich aber ebenso zur Beschreibung der logischen Datenstruktur, welche hier als Verallgemeinerung der physischen Modellaspekte verstanden wird. Die Supportklassen zur Abbildung von Objektbeziehungen sind im GDI-Metamodell Object Relationships enthalten. Dieses basiert auf dem Relationships-Paket der Object Model-Ebene des CWM.

Bezeichnung im GDI-Metamodell:	Logical Data Structure
Verwendete CWM-Basis:	*org.omg::CWM::ObjectModel::Core*
Bestehende Abhängigkeiten:	keine

Bezeichnung im GDI-Metamodell:	Object Relationships
Verwendete CWM-Basis:	*org.omg::CWM::ObjectModel::Relationships*
Bestehende Abhängigkeiten:	*org.omg::CWM::ObjectModel::Core*

Tab. 4: CWM-Basis der logischen Datenstruktur

Das Core-Paket beinhaltet einige grundlegende Datentypen wie *Any*, *Boolean*, *Float*, *Integer*, *UnlimitedInteger*, *String* und *Time*. Der Typ *Name* ist eine Erweiterung von *String*. Er wird im CWM zur Benennung von Modellelementen verwendet. Die Endung „Kind“ lässt auf eine Aufzählung (engl. enumeration) schliessen. In Core existieren folgende Aufzählungstypen:

Typ	***Beschreibung***
ChangeableKind	Legt fest, wie Eigenschaften verändert werden dürfen (*ck_changeable*, *ck_frozen*, *ck_addOnly*). Ersteres erlaubt beliebige Modifikationen, während bei Letzterem das Löschen eines bestehenden Werts untersagt ist.
OrderingKind	Bestimmt die Anordnung der Elemente einer Menge (*ok_ordered*, *ok_unordered*). Die korrekte Reihenfolge ist durch entsprechende Operationen sicherzustellen.
ScopeKind	Bezeichnet, ob eine Eigenschaft einzelnen Instanzen oder einer Klasse angehört (*sk_instance, sk_classifier).*
VisibilityKind	Kennzeichnet die Sichtbarkeit eines Elements ausserhalb seines Namensraums (*vk_public*, *vk_protected*, *vk_private*, *vk_package*, *vk_notapplicable*). Letzteres wird verwendet, wenn Namensräume das Konzept der Sichtbarkeit nicht unterstützen.

Tab. 5: Aufzählungstypen im Core-Paket des CWM

Logical Data Structure

Gegenüber dem CWM wurde die Anzahl zugehöriger Klassen zugunsten flacher Vererbungshierarchien wesentlich reduziert. Auf der obersten Stufe des Metamodells befindet sich die abstrakte Klasse *ModelElement*. Sie repräsentiert ein durch einen Namen bezeichnetes Modellierungsartefakt und ist die Basis aller hier verwendeten Modellierungsklassen. Die Klasse *Namespace* ist eine Subklasse von *ModelElement*, kann aber gleichzeitig Eigentümerin eines solchen sein. Aus dieser rekursiven Vererbung lässt sich ableiten, dass ein *Namespace* zugleich wiederum *Namespaces* bzw. davon abgeleitete Subklassen enthalten kann.

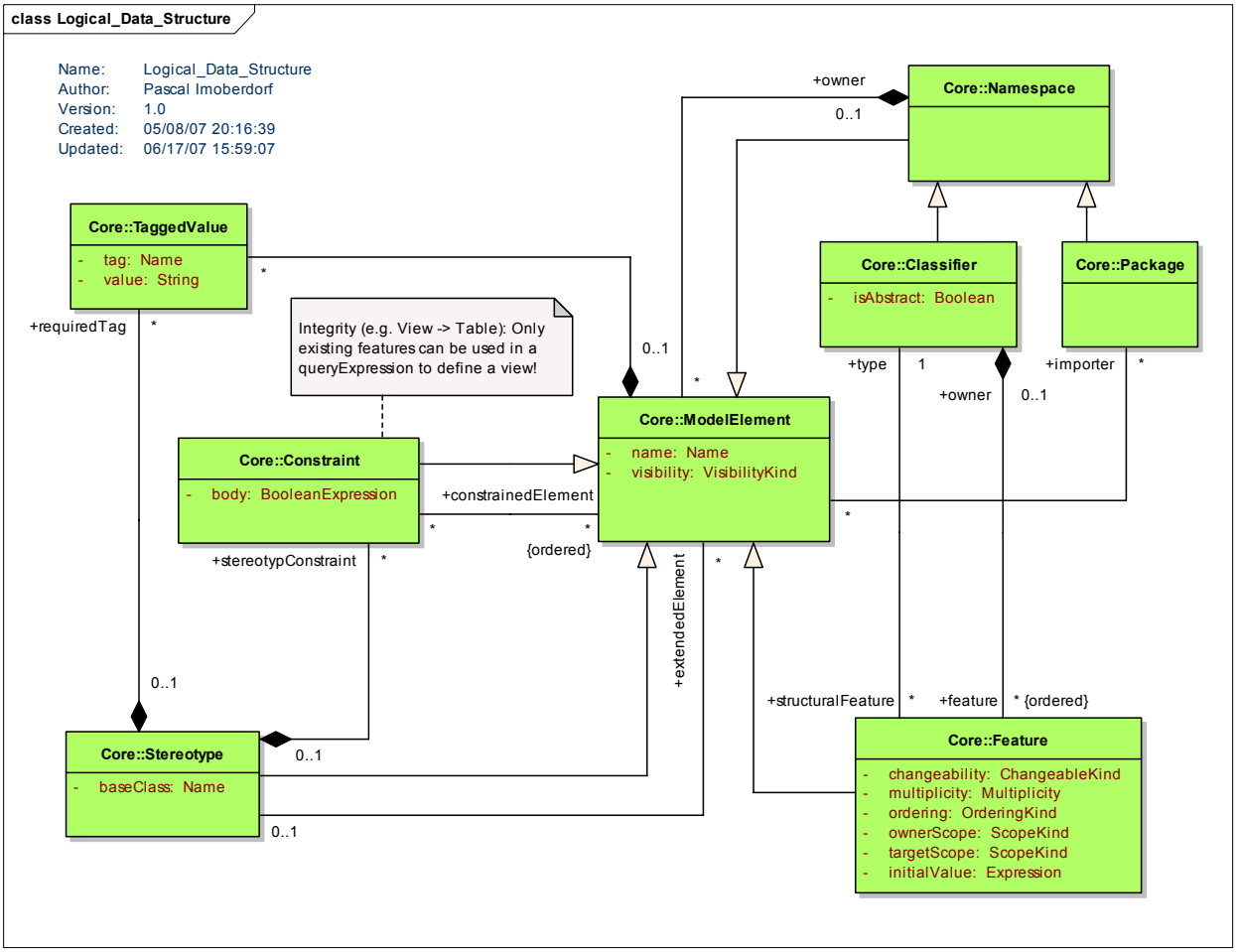

Abb. 27: Metamodell "Logical Data Structure"

Package ist eine Spezialisierung der Klasse *Namespace*. Es stellt einen generellen Mechanismus bereit, um Modellelemente wie *Classifier* oder sogar *Packages* gruppieren zu können, indem es diesen Inhalten als Namensraum dient. Voraussetzung dazu ist es, dass Elemente innerhalb eines *Packages* eindeutig bezeichnet werden. Sobald ein *Package* aus einem Modell entfernt wird, werden auch die darin enthaltenen Elemente beseitigt. Letztlich gilt es zu erwähnen, dass ein *Package* neben seinen eigenen *ModelElements* weitere Elemente aus anderen Paketen importieren kann. Der Namensraum, der durch das *Package* definiert wird, wird dabei um die importierten Elemente erweitert.

Classifier ist eine abstrakte Subklasse von *Namespace*. Ein *Classifier* ist ein Modellierungselement, welches innerhalb eines Modells in unterschiedlicher Form, beispielsweise als Klasse oder Datentyp, auftreten kann. Innerhalb eines bestimmten *Namespaces* muss ein *Classifier* einen eindeutigen Namen besitzen. *Classifier* können, analog zu *Packages*, ineinander geschachtelt werden. Der Zugriff auf verschachtelte *Classifier* hängt von der in der Superklasse *ModelElement* festgelegten Sichtbarkeit (*visibility*) ab. Ein *Feature* als Subklasse von *ModelElement* ist eine Eigenschaft (z.B. ein Attribut oder eine Operation), die in einem *Classifier* gekapselt ist. Ein *Classifier* kann mehrere geordnete *Features* enthalten. Umgekehrt wird jedem strukturellen *Feature* als Datentyp genau ein *Classifier* zugewiesen. Ein *Feature* ist durch seine *multiplicity* gekennzeichnet. Es ist im Weiteren auch möglich, einem *Feature* über das Attribut *initialValue* einen Anfangswert zuzuweisen. Wie alle von *ModelElement* abgeleiteten Klassen besitzen auch *Classifier* und *Feature* ein Attribut *name* zur eindeutigen Identifikation.

Die Klassen *TaggedValue* und *Stereotype* bieten die Möglichkeit zur Erweiterung von bestehenden Modellelementen (vgl. Kap. 2.8). Jedem *ModelElement* können zusätzliche Informationen über *TaggedValues* in der Form „tag=value“ angefügt werden. Die Bedeutung von solchen Name-Wert-Paaren liegt ausserhalb des Einflussbereiches des CWM und muss durch Benutzer- oder Werkzeugkonventionen festgelegt werden. *Stereotypes* dienen in erster Linien der Gruppierung von *TaggedValues* und *Contraints*. *Stereotypes* sind vom Typ *ModelElement* und weisen folglich die gleiche Struktur wie herkömmliche Modellelemente auf. Über das Attribut *baseClass* wird der Name desjenigen *ModelElements* spezifiziert, auf welches der *Stereotype* angewendet werden soll. Ein *Constraint* ist eine semantische Bedingung bzw. Einschränkung, die zutreffen muss damit ein *ModelElement* syntaktisch korrekt ist. Durch *Constraints* kann die Semantik von *ModelElements* im Attribut *body* sprachlich anhand von bool’schen Ausdrücken (*booleanExpression*) spezifiziert werden. Zur Formulierung von *Constraints* ist beispielsweise die Object Constraint Language (OCL) geeignet. Über *Constraints* kann im GDI-Metamodell sichergestellt werden, dass bei der Definition eines *ModelElements* vom Typ *View* (vgl. Kap. 4.3.2) nur existierende Attribute verwendet werden dürfen.

Object Relationships

Das CWM unterscheidet bei Objektbeziehungen zwischen *Generalization* und *Association*. Eine *Association* definiert eine semantische Beziehung zwischen *Classifiers*, welche in Form einer gewöhnlichen Assoziation, einer Aggregation oder einer Komposition auftreten kann. Eine *Association* ist ein *Classifier* und besitzt daher via *Classifier-Feature*-Beziehung zwei (→ binary) oder mehrere (→ n-ary) *AssociationEnds*. Über dieselbe Beziehung (in umgekehrter Richtung) wird festgelegt, dass jedes *AssociationEnd* als Spezialisierung der Klasse *Feature* genau einer *Association* zugehört. Dabei verbindet jedes Ende die zugehörige Assoziation mit Instanzen von *Classifier*.

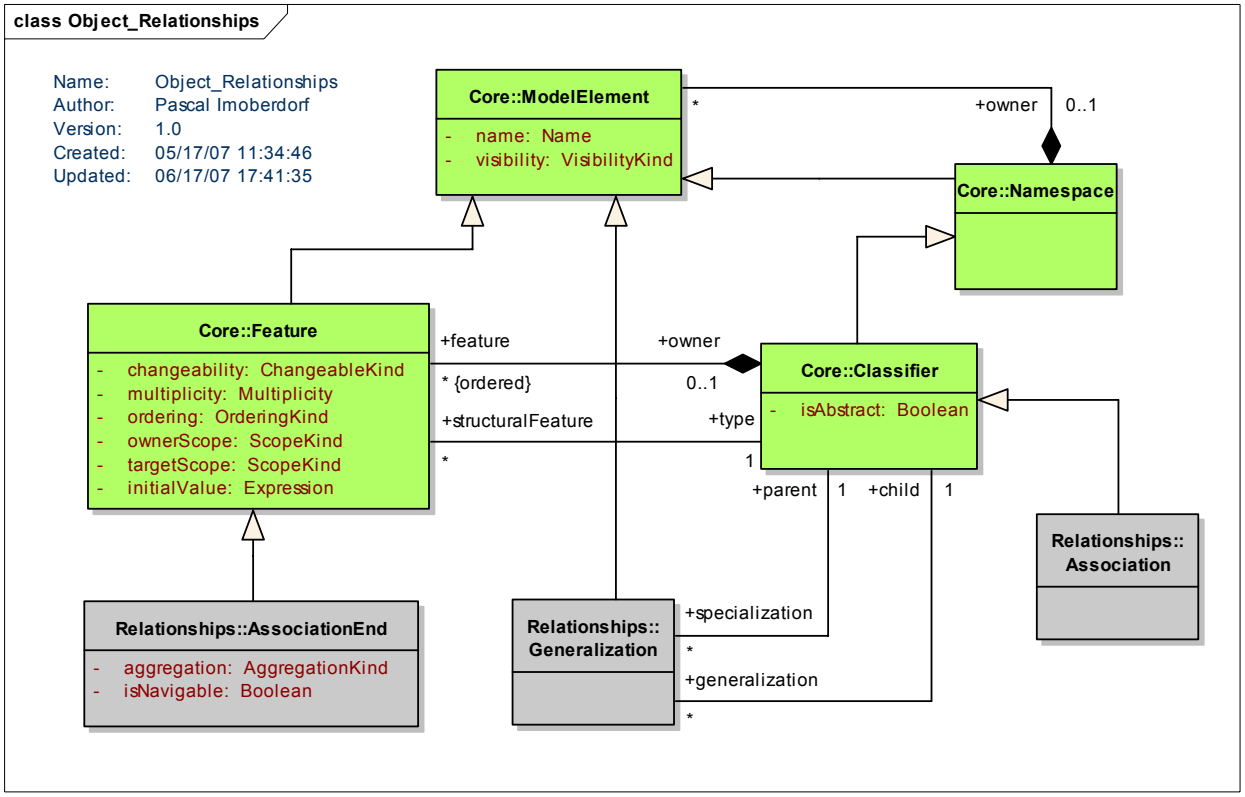

Abb. 28: Metamodell "Object Relationships"

Das Attribut *aggregation* der Klasse *AssociationEnd* ist eine Aufzählung vom Typ *AggregationKind*, welche die Werte „*ak_none*“, „*ak_aggregate*“ oder „*ak_composite*“ annehmen kann. Aggregation und Komposition sind Beziehungen zwischen Teilen und dem zugehörigen Ganzen. Im Unterschied zur Aggregation existieren die Teile einer Komposition nur während der Lebensdauer des Ganzen. Sowohl die Aggregation als auch die Komposition dürfen nur bei binären Beziehungen auftreten. Innerhalb einer *Association* darf per Definition höchstens ein Ende eine Aggregation oder Komposition sein. Diese beiden Bedingungen sind in Form von *Constraints* festzuhalten. Die Klasse *Generalization* repräsentiert hierarchische Objektbeziehungen zwischen *Classifiers* in Form von Eltern-Kind-Beziehungen. Abhängig von der Betrachtungsrichtung stellt eine solche Beziehung eine Spezialisierung oder eine Generalisierung dar. Die Subklasse (*child*) erbt dabei immer die Eigenschaften und Operationen ihrer Superklasse (*parent*). *Generalization* ist eine Subklasse von *ModelElement*.

4.3.2 Metamodell: Physische Datenstruktur

Das Teilmodell Physical Data Structure basiert auf dem Relational-Metamodell der CWM-Ressourcen-Schicht. Das Relational-Paket beschreibt Daten, die durch eine relationale Schnittstelle wie RDBMS, ODBC oder JDBC zugänglich sind. Es adressiert analog zur logischen Datenstruktur auch die Thematik von Beziehungen (Primär- und Fremdschlüssel), wobei im Konkreten das Paket KeysIndexes aus der Foundation-Ebene erweitert wird. Dieser Aspekt wird im Teilmodell Key Relationships modelliert. Mit den Metamodellen Physical Data Structure und Key Relationships lässt sich die physische Datenstruktur einer Datenbankumgebung auf einem Meta-Level vollständig beschreiben. Es gilt hier anzumerken, dass auf die Abbildung von Indizes, Triggern und Procedures abweichend zu CWM vorerst verzichtet wird. Die entsprechenden Klassen können bei Bedarf nachträglich ohne grossen Aufwand integriert werden.

Bezeichnung im GDI-Metamodell:	Physical Data Structure
Verwendete CWM-Basis:	*org.omg::CWM::Resource::Relational*
Bestehende Abhängigkeiten:	*org.omg::CWM::ObjectModel::Core*

Bezeichnung im GDI-Metamodell:	Key Relationships
Verwendete CWM-Basis:	*org.omg::CWM::Resource::Relational*
Bestehende Abhängigkeiten:	*org.omg::CWM::Foundation::KeysIndexes*

Tab. 6: CWM-Basis der physischen Datenstruktur

Physical Data Structure

Die Klassen *Catalog* und *Schema* sind Spezialisierungen der Klasse *Package*. Ein *Catalog* definiert den Namensraum für ein oder mehrere Schemas. Über das Attribut *defaultCharacterSetName* wird der Standard-Zeichensatz für String-Typen bestimmt. Ein *Schema* wiederum bildet den Namensraum für Tabellen (*Table*) und/oder Sichten (*View*), welche beide von der Klasse *Classifier* abgeleitet sind. Das Attribut *isSystem* von *Table* weist auf Systemtabellen hin, während *isTemporary* anzeigt, ob der Inhalt einer Tabelle temporär oder dauerhaft ist. *Views* werden über so genannte *QueryExpressions* definiert. Durch die Implementierung ist die referenzielle Integrität sicherzustellen, d.h. dass Ausdrücke zur Definition von Sichten nur Tabellen und Spalten enthalten dürfen, die im Metadaten-Repository tatsächlich existieren. Das Attribut *isReadOnly* legt fest, ob die referenzierte Tabellenspalte über einen *View* geändert werden kann. Tabellen und Sichten enthalten eine beliebige Anzahl an Spalten (*Columns*). Die Klasse *Column* ist eine Spezialisierung der Klasse *Feature*. Die Festlegung der Beziehung zwischen Spalten und zugehöriger Tabelle bzw. Sicht erfolgt implizit auf der logischen Ebene in Form einer Komposition zwischen den Klassen *Classifier* und *Feature*. Jede *Column* verweist ihrerseits auf genau einen Datentypen (*SQLDataType*), wobei auch diese Assoziation über die übergeordneten Klassen *Feature* und *Classifier* hergestellt und durch Vererbung an die Subklassen weitergereicht wird.

Zur Klasse *Column* gehören die Attribute *characterSetName* für den zulässigen Zeichensatz und *collationName* für die anzuwendende Sortiersequenz. Das Attribut *isNullable* zeigt an, ob eine Spalte einer Tabelle auch NULL-Werte enthalten darf. Das Attribut *length* definiert die Länge eines Byte-Strings, während *precision* die Gesamtanzahl der möglichen numerischen Zeichen in einem Feld und *scale* die Anzahl der Fliesskommastellen festlegt.

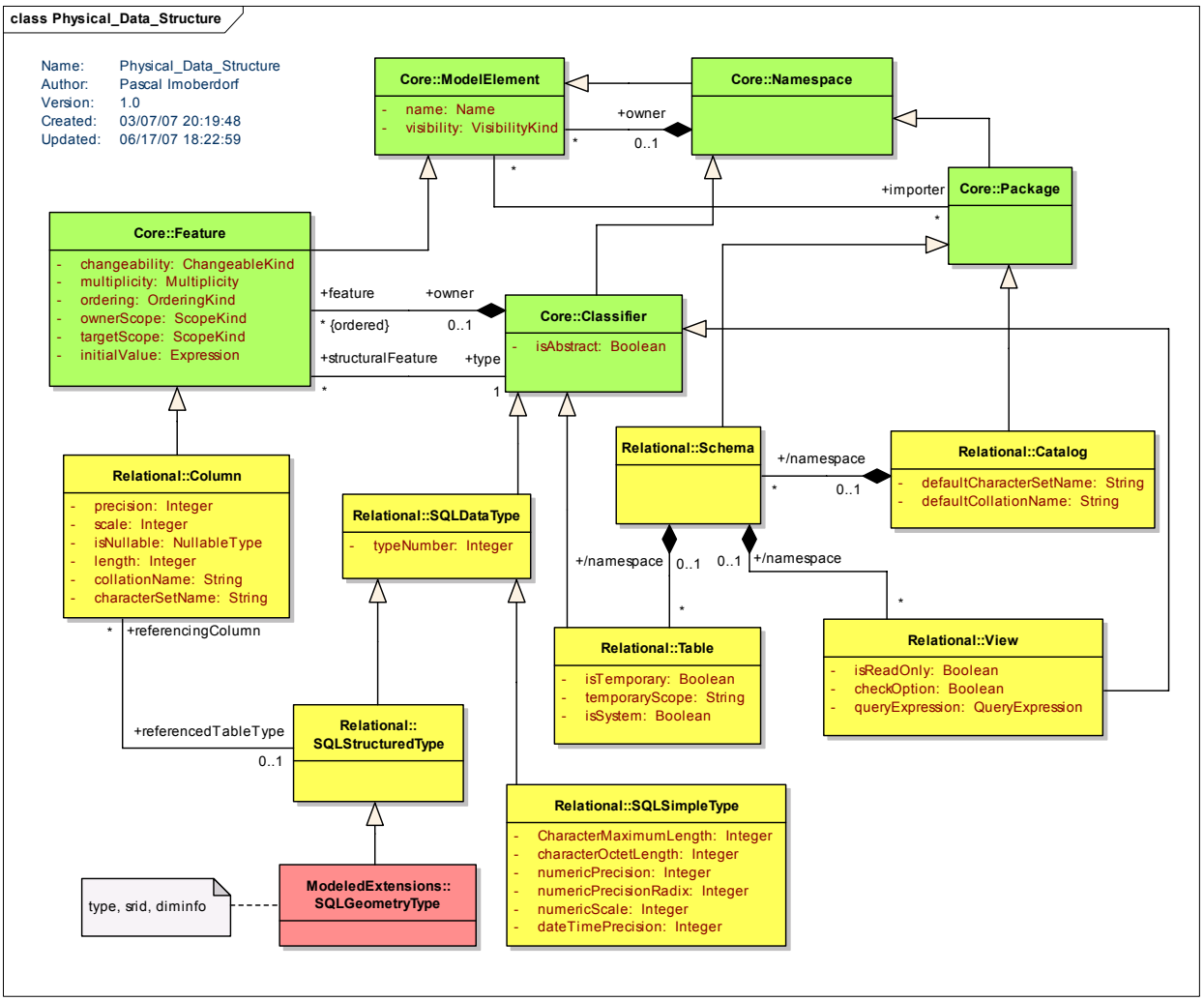

Abb. 29: Metamodell "Physical Data Structure"

Ein *SQLDataType* ist ebenfalls eine Subklasse von *Classifier*. Bei den Datentypen wird zwischen einfachen (*SQLSimpleTypes*) und zusammengesetzten (*SQLStructuredTypes*) Datentypen unterschieden. Als einfache Datentypen gelten u.a. Integer, Float, String oder Boolean. Der Umfang an verfügbaren Basistypen ist je nach Datenbanksystem unterschiedlich. Ein *SQLStructuredType*, welcher eine objekt-relationale Erweiterung darstellt, eignet sich insbesondere auch für die Beschreibung von geometrischen Datentypen (*SQLGeometryType*). Ein geometrischer Datentyp stellt eine durch Vererbung modellierte Erweiterung von CWM dar.

Jeder *SQLStructuredType* ist wiederum eine gewöhnliche Tabelle mit beliebigen Eigenschaften, wobei eine *Column* sowohl *SQLSimpleTypes* als auch *SQLStructuredTypes* enthalten kann. Dies wird im Modell durch die Assoziation zwischen *Column* und *SQLStructuredType* explizit ausgedrückt. Bei geometrischen Datentypen kann/sollte mittels Fremdschlüssel immer ein Verweis auf ein räumliches Bezugssystem hergestellt werden (siehe Key Relationships).

Key Relationships

Die Klassen *KeyRelationship* und *UniqueKey* der Foundation-Ebene wurden abweichend zu CWM zugunsten flacher Vererbungshierarchien weggelassen. *ForeignKey* und *UniqueConstraint* sind somit direkte Erweiterungen der Klasse *ModelElement*. Ein *UniqueConstraint* stellt eine Bedingung dar, um die Eindeutigkeit der Tupel innerhalb einer Relation sicherzustellen. Das Attribut *deferrability* legt hierbei fest, wann diese Bedingung vom System geprüft werden soll (z.B. am Ende einer Transaktion).

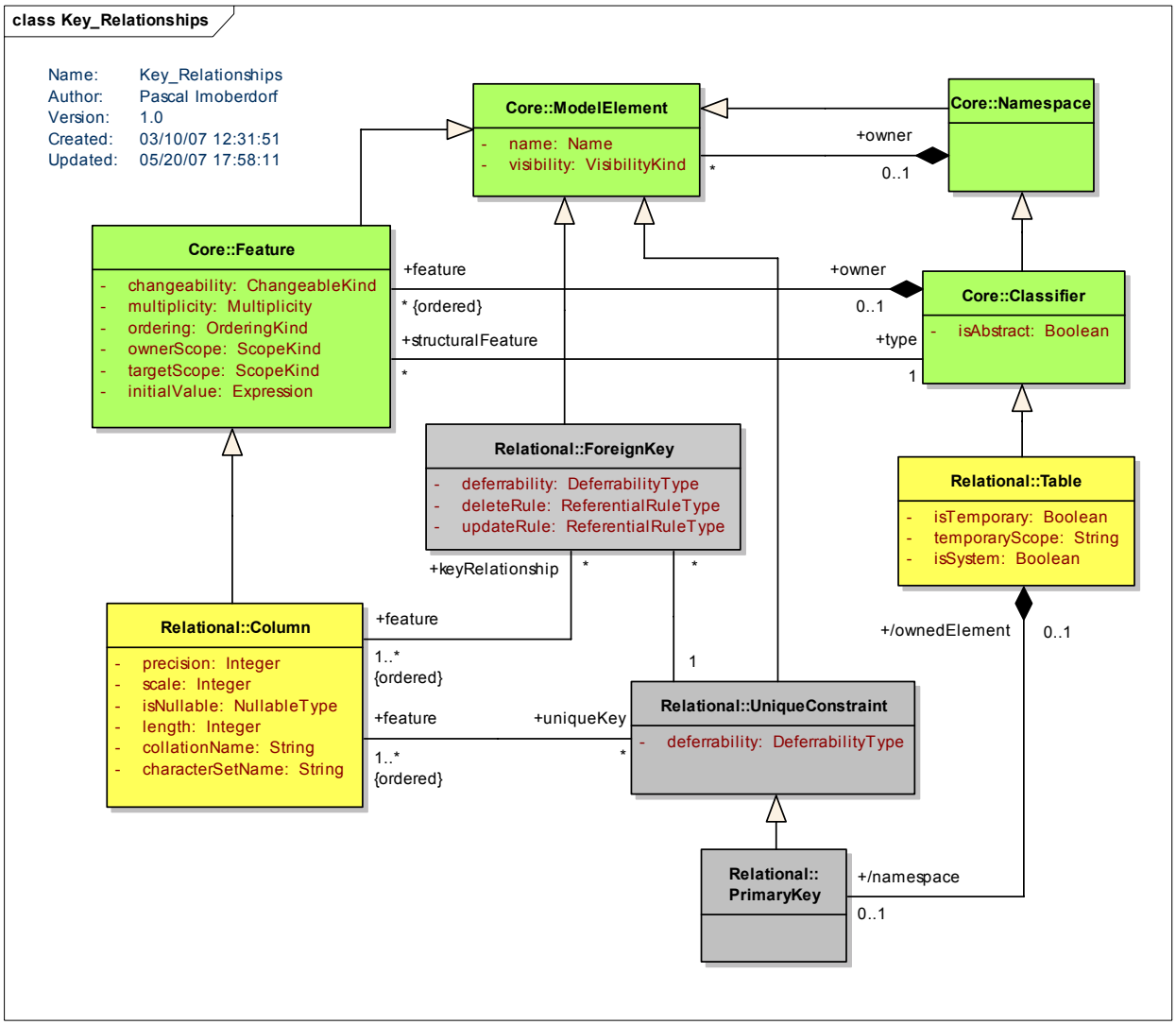

Abb. 30: Metamodell "Key Relationships"

Die Klasse *PrimaryKey* ist eine Spezialisierung der Klasse *UniqueConstraint*. Ein Primärschlüssel setzt sich in geordneter Reihenfolge aus einer oder mehreren Spalten (*Column*) einer Relation zusammen. *Table* wiederum ist der Namensraum eines *UniqueContraints* bzw. eines *PrimaryKeys*. Die Beziehung zwischen *Table* und *UniqueConstraint* wird auf übergeordneter Ebene durch eine Komposition zwischen *Namespace* und *ModelElement* hergestellt. Da pro Tabelle (*Table*) nur genau ein *UniqueConstraint* vom Typ *PrimaryKey* existieren darf, wurde diese Beziehung mit geänderter Multiplizität zwischen *Table* und *PrimaryKey* explizit nochmals eingefügt. Ein Fremdschlüssel (*ForeignKey*) setzt sich ebenfalls aus einer oder mehreren Spalten (*Column*) einer Tabelle zusammen. Diese Verbindung ist hier erneut als Komposition zwischen *Namespace* und *ModelElement* modelliert. Ein Fremdschlüssel verweist auf einen *PrimaryKey* einer anderen Relation und stellt implizit eine Beziehung zu einer *Table* her. Die Attribute *deleteRule* und *updateRule* der Klasse *ForeignKey* legen fest, wie das System sich verhält, sobald ein Tupel des zugehörigen Primärschlüssels gelöscht oder geändert wird.

4.3.3 Metamodell: Datenquellen

Das Metamodell Data Sources wurde vom Paket SoftwareDeployment der CWM-Foundation-Schicht abgeleitet. SoftwareDeployment enthält Klassen um die Softwarekomponenten eines Data Warehouse-Systems im Sinne eines IT-Portfolios zu verwalten. Dabei wird erfasst, welche Softwarepakete aus welchen Einzelkomponenten bestehen, auf welchen Rechnern diese installiert wurden und welche Abhängigkeiten zwischen den installierten Komponenten bestehen. Darüber hinaus bietet das SoftwareDeployment-Metamodell Möglichkeiten, um den Zugriff von Softwaresystemen auf Datenressourcen durch Schnittstellen zu beschreiben. Auf die dafür vorgesehenen Klassen wird im Data Sources-Metamodell zurückgegriffen.

Bezeichnung im GDI-Metamodell:	Data Sources
Verwendete CWM-Basis:	*org.omg::CWM::Foundation::SoftwareDeployment*
Bestehende Abhängigkeiten:	*org.omg::CWM::ObjectModel::Core*

Tab. 7: CWM-Basis von Datenquellen

Die CWM-Klasse *DeployedComponent* wird zugunsten flacher Vererbungshierarchien weggelassen. Das Attribut *pathname*, unter dem die Speicherung des Installationspfads einer *DeployedComponent* möglich ist, wird deshalb in die Klasse *DataManager* integriert. Der *DataManager*, eine Subklasse von *Package*, ist eine System- bzw. Serverkomponente, die den Zugriff auf lokale oder verteilte Datenquellen verwaltet. Konkret kann dies ein File- oder Datenbankverwaltungssystem (DBMS) sein, welches mit einem oder mehreren *dataPackages* assoziiert ist. Ein *dataPackage* ist dabei eine Datenquelle, wie z.B. ein relationales *Schema*, ein *Catalog* oder ein gewöhnliches Datenfile. Das Attribut *isCaseSensitive* der Klasse *DataManager* zeigt an, ob die jeweilige Serverkomponente Gross- und Kleinschreibung in Bezug auf Objektnamen unterschiedlich behandelt.

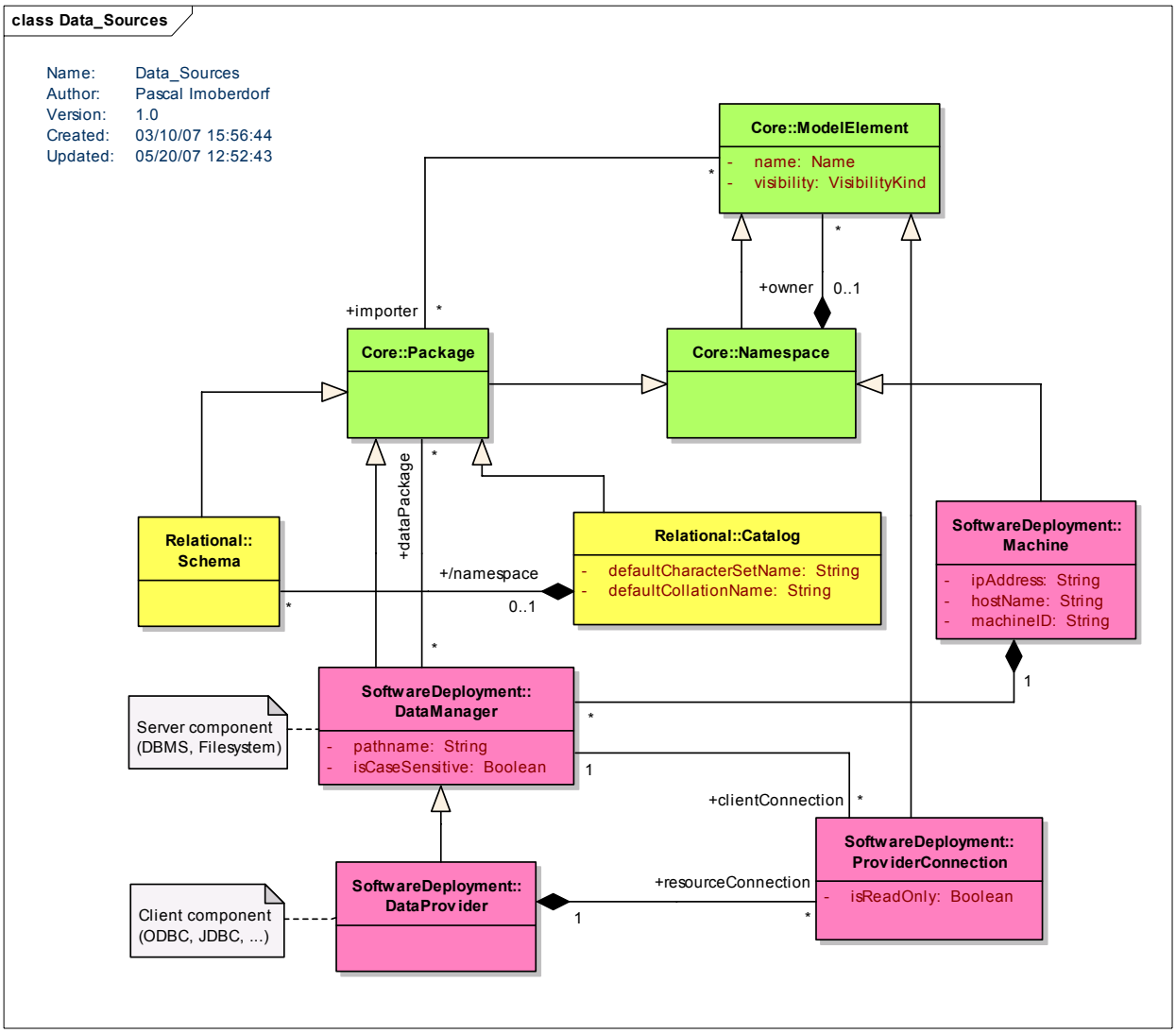

Abb. 31: Metamodell „Data Sources“

Ein *DataProvider* repräsentiert Clients für den Zugriff auf Datenressourcen, welche von einem *DataManager* verwaltet werden. Beispielsweise kann ein ODBC- oder JDBC-Client durch die Klasse *DataProvider* abgebildet werden. Ein *DataProvider* darf mehrere *ProviderConnections* besitzen, wobei eine solche jeweils genau einen *DataManager* referenziert. Die Klasse *ProviderConnection* ist eine Subklasse von *ModelElement*. Durch die Vererbung zwischen *DataProvider* und *DataManager* wird ausgedrückt, dass ein Server, der eine Anfrage eines Clients entgegennimmt, in der Folge möglicherweise selber als Client auftritt und Daten von einem anderen Server bezieht. Die Klasse *Machine* ist eine Subklasse von *Namespace*. Jede Manager- und Provider-Komponente ist physisch auf einem bestimmten Host-Rechner (*Machine*) installiert, der durch seine *ipAddress*, den *hostName* oder seine *machineID* eindeutig identifizierbar und auffindbar ist. Server und Client können dabei auf unterschiedlichen oder auf ein und demselben Rechner installiert sein. Mit diesen Angaben ist es möglich, dass ein System die Verbindung zu den Daten findet. Die Frage der Authentifizierung und Autorisierung wird aus Sicherheitsgründen gänzlich dem Daten bereitstellenden System überlassen.

4.3.4 Metamodell: Benutzer-Metadaten

Das Teilmodell User Metadata basiert auf dem BusinessInformation-Metamodell der CWM-Foundation-Ebene. Seine Modellierungselemente stehen somit sämtlichen Paketen der darüber liegenden CWM-Schichten zur Verfügung. User Metadata enthält generelle Klassen zur Abbildung von beschreibender Metainformation bezogen auf beliebige Modellelemente.

Bezeichnung im GDI-Metamodell:	User Metadata
Verwendete CWM-Basis:	*org.omg::CWM::Foundation::BusinessInformation*
Bestehende Abhängigkeiten:	*org.omg::CWM::ObjectModel::Core*

Tab. 8: CWM-Basis von Benutzer-Metadaten

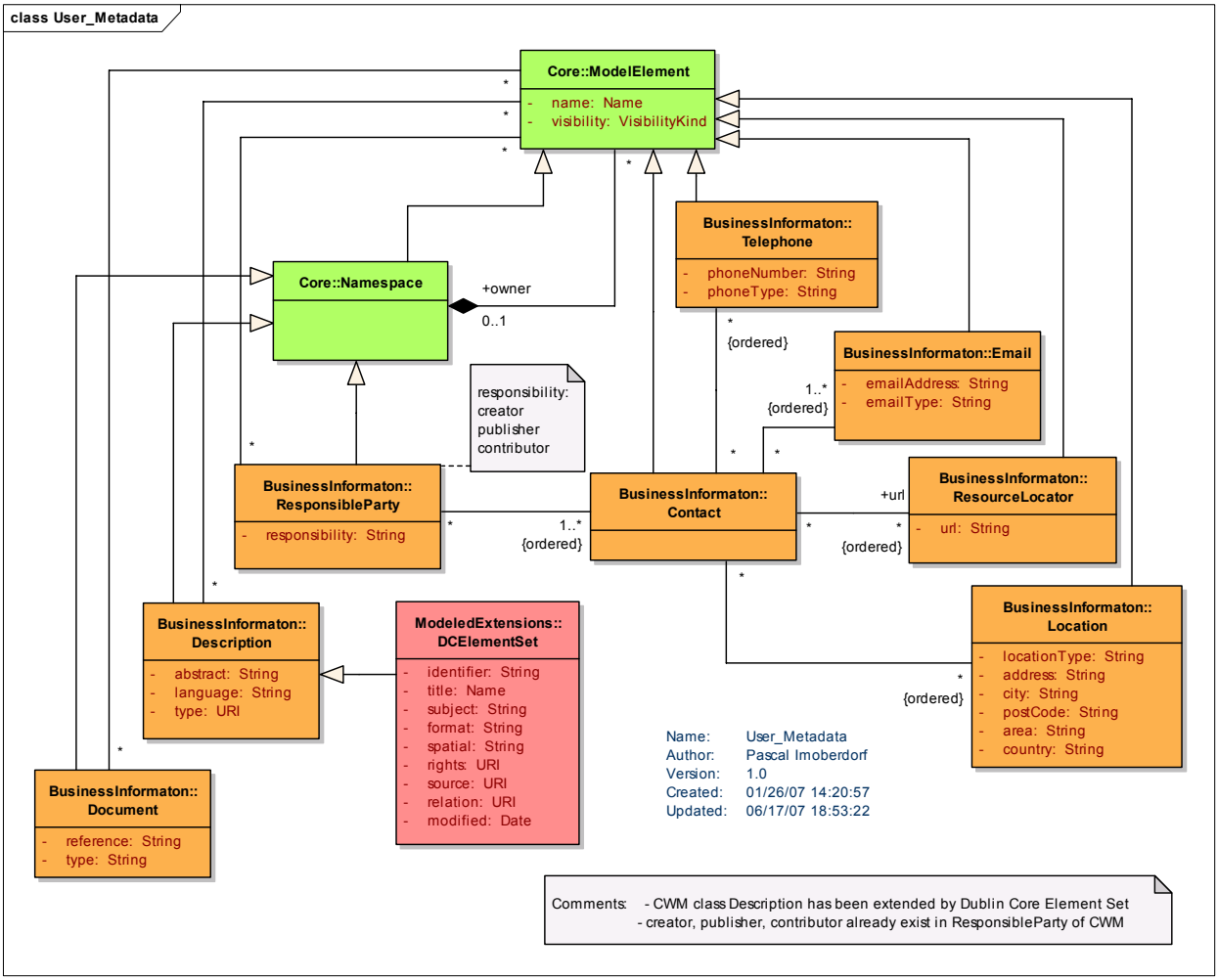

Abb. 32: Metamodell "User Metadata"

Das Metamodell zur Beschreibung von Benutzer-Metadaten stützt sich im Wesentlichen auf die CWM-Klassen *ResponsibleParty*, *Document* und *Description.* Diese drei Hauptklassen sind Subtypen von *Namespace* und weisen Beziehungen zur Klasse *ModelElement* auf. Damit kann jedes *ModelElement* mit einer beliebigen Anzahl dieser beschreibenden Klassen verknüpft werden. Umgekehrt kann dieselbe Beschreibung mehreren Modellelementen gleichzeitig zugewie-

sen werden. Da diese drei Klassen selber ebenfalls Spezialisierungen von *ModelElement* sind, kann beispielsweise die Klasse *ResponsibleParty* mit einem oder mehreren *Documents* verbunden sein. Da diese Klassen jeweils einen *Namespace* darstellen, können sie darüber hinaus hierarchisch strukturiert werden. So kann zum Beispiel eine aus mehreren Kapiteln bestehende Dokumentenstruktur erstellt werden. Dazu enthält ein *Document* vom Typ *Namespace* mehrere Dokumente vom Typ *ModelElement*.

Für jede *ResponsibleParty* ist mindestens ein[44] *Contact* anzugeben. Mehrere Kontaktstellen werden in einer bestimmten Reihenfolge geordnet. Ein Kontakt verweist in beliebiger Anzahl auf die Klassen *Telephone*, *Email*, *ResourceLocator* und *Location*. CWM verwendet hierzu statt einer Komposition jeweils normale Assoziationen. Durch den Ordering-Constraint können die Kontaktinformationen priorisiert werden. Da jede dieser vier Klassen selber vom Typ *ModelElement* ist, sind sie an jeder anderen Stelle in einem Modell verwendbar. Die Bedeutung dieser Klassen ist weitgehend selbsterklärend. Die Attribute *phoneType*, *emailType* und *locationType* können mit einer frei definierbaren Textinformation versehen werden (z.B. mit „home“ oder „work“). Die Klasse *ResourceLocator* ist als Ergänzung zu den anderen drei Klassen zu sehen. Das Attribut *url* kann eine Web-URL referenzieren oder einen Text mit Beschreibung eines Ortes enthalten (z.B. „3.Stock, Raum 115“).

Mit der Klasse *ResponsibleParty* können nicht nur Personen, sondern auch Rollen oder Organisationen abgebildet werden. Das zugehorige Attribut *responsibility* kann einen beliebigen Text enthalten, der die Art und Weise der Zuständigkeit beschreibt. Im GDI-Kontext ist etwa die Verwendung der Werte „*creator*“, „*publisher*„ oder „*contributor*“ denkbar. Ein *Document* enthält über das Attribut *reference* einen Verweis auf extern gespeicherte, beschreibende Informationen. Der Verweis kann direkt über die Angabe des physischen Speicherorts oder über einen Identifikator vom Typ *String* festgelegt werden. Den Namen des Dokuments erbt diese Klasse von der Superklasse *ModelElement*. Im Unterschied dazu werden die beschreibenden Informationen der Klasse *Description* direkt im Modell gespeichert. Diese enthält die Attribute *abstract*[45] zur textuellen Beschreibung, *language* mit Angabe der verwendeten Sprache sowie *type* für die Festlegung eines kontextbezogenen Typs.

Gegenüber CWM wird das Modell durch die Klasse *DCElementSet* als eine Spezialisierung von *Description* spezifisch erweitert. Sie dient der Speicherung von Informationen gemäss dem Dublin Core Metadata Element Set[46]. Sie enthält in Erweiterung zur Klasse *Description* zusätzliche Elemente zur Beschreibung von Ressourcen. Die Dublin Core-Attribute *creator*, *contributor* und *publisher* werden bewusst weggelassen, da diese Informationen bereits in der Klasse *ResponsibleParty* durch das Attribut *responsibility* abgebildet sind.

---

[44] *Abweichung: CWM sieht hier eine n:n-Beziehung vor.*
[45] *Abweichung: CWM verwendet hierzu das Attribut „body“*
[46] *Dublin Core Metadata Element Set: http://dublincore.org/documents/dces (25.06.2007)*

Die Bedeutung der 12 Attribute des Dublin Core Metadata Element Sets (ohne creator, publisher und contributor) wird in der folgenden Tabelle erläutert:

Attributname	***Attributtyp***	***Beschreibung***
dc: identifier	String	Eindeutiger Verweis auf eine Ressource innerhalb eines gegebenen Kontexts
dc: title	String	Name einer Ressource
dc: abstract	URI, String	Kurzbeschreibung einer Ressource (Subtyp von dc:description)
dc: subject	String	Thema der Ressource → z.B. Keywords
dc: language	String	Sprache der Ressource, die Verwendung eines kontrollierten Wortschatzes wie RFC 3066[47] wird empfohlen
dc: type	String	Art der Ressource (z.B. Text, Dataset, Service) → DCMI Type Vocabulary[48]
dc: format	URI	Format der Ressource → Internet Media Types [MIME][49]
dc: spatial	String	Geographischer Bezug (Subtyp von dc:coverage): Bounding Box, Ortsbezeichnung[50], Punktkoordinaten
dc: modified	Date	Zeitpunkt der letzten Änderung (Subtyp von dc:date)
dc: source	URI, String	Quelle, von welcher die Ressource abgeleitet wurde
dc: rights	URI, String	Nutzungs- und Lizenzbedingungen
dc: relation	URI	Eine in Verbindung stehende Ressource

Tab. 9: Erweiterte Attribute nach dem Dublin Core Metadata Element Set

Das Dublin Core Metadata Element Set (DCES) wurde hier beispielhaft gewählt, weil sich die darin enthaltenen Metadatenelemente zur Beschreibung von Ressourcen aus unterschiedlichen Domänen eignen. Da sich die Klasse *DCElementSet* auf die Klasse *ModelElement* bezieht, können theoretisch sämtliche Modellelemente mittels der DCES-Attribute beschrieben werden. Im GDI-Kontext bezieht sich die semantische Beschreibung explizit auf Geodatensätze, so dass sich die Verwendung des *DCElementSets* durch einen *Contraint* (vgl. Logical Data Structure) auf die Klasse *Package* (für *dataPackage*) beschränken sollte.

[47] *RFC 3066: http://www.ietf.org/rfc/rfc3066.txt (25.06.2007)*
[48] *DCMI Type Vocabulary: http://dublincore.org/documents/dcmi-type-vocabulary (25.06.2007)*
[49] *MIME Media Types: http://www.iana.org/assignments/media-types (25.06.2007)*
[50] *Thesaurus of Geographic Names: http://www.getty.edu/research/conducting_research/vocabularies/tgn/index.html (25.06.2007)*

4.3.5 Metamodell: Visualisierung und Klassifizierung

Das Data Portrayal-Metamodell orientiert sich am InformationVisualization-Paket der Analysis-Ebene von CWM. Es enthält Metainformation, die zur Visualisierung der gespeicherten Ressourcen benötigt wird. Das CWM benutzt hierzu einen äusserst generischen Ansatz, bei dem die wenigen Metamodellklassen Informationen zu komplexen Visualisierungsmechanismen enthalten oder auf solche verweisen.

Bezeichnung im GDI-Metamodell:	Data Portrayal
Verwendete CWM-Basis:	*org.omg::CWM::Analysis::InformationVisualization*
Bestehende Abhängigkeiten:	*org.omg::CWM::ObjectModel::Core* *org.omg::CWM::Foundation::Expressions*

Tab. 10: CWM-Basis von Visualisierung und Klassifikation

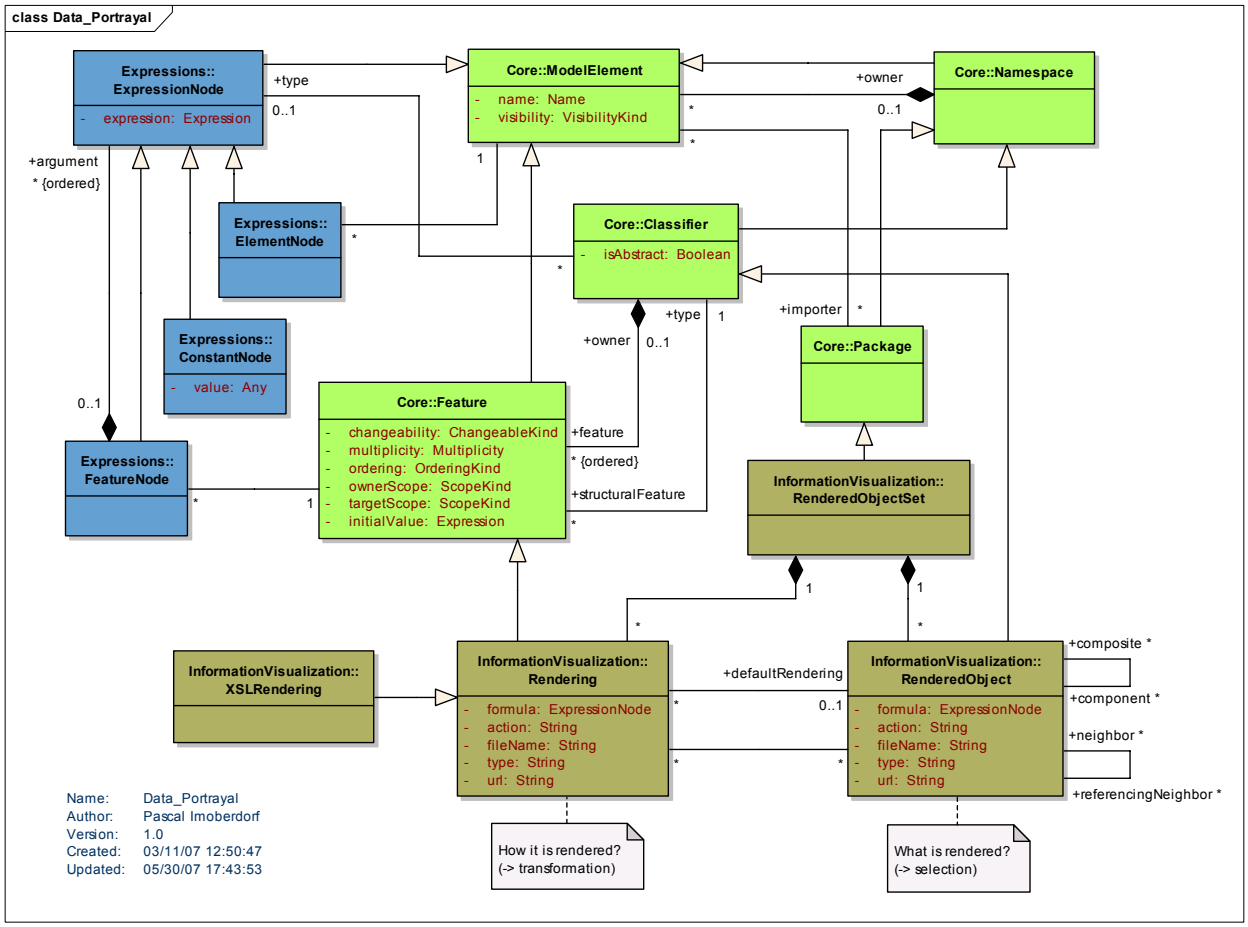

Abb. 33: Metamodell "Data Portrayal"

Ein *RenderedObject* repräsentiert einen logischen Stellvertreter eines zu visualisierenden *Classifiers*. Es kann aus einer beliebigen Anzahl weiterer *RenderedObjects* (*composite → component*) zusammengesetzt sein und gleichzeitig topologische Beziehungen zu anderen *RenderedObjects* (*referencingNeighbor → neighbor*) aufweisen. Über das Attribut *formula* wird

in Form eines *ExpressionNodes* eine Selektion definiert, welche die darzustellenden Objekte festlegt. Zur Veranschaulichung kann man sich beispielsweise eine Abfrage vorstellen, bei der sämtliche Gebäude vom Typ „öffentlich“ selektiert werden. Ein anderer Ausdruck könnte zum Beispiel alle an eine Hauptstrasse angrenzenden Strassen selektieren, welche gleichzeitig nur in eine Richtung befahren werden dürfen. Eine Expression kann in einer beliebigen Form, zum Beispiel durch SQL oder XPath[51], definiert werden.

Jedes *RenderedObject* verweist auf ein oder mehrere *Renderings*. Ein *Rendering* ist eine Subklasse von *Feature* und stellt demnach ein gewöhnliches Merkmal dar, welches seinerseits das Aussehen der ausgewählten Objekte definiert. Es besteht hierzu auch die Möglichkeit, ein *defaultRendering* festzulegen. Ein *Rendering* ist semantisch mit einer Transformation vergleichbar, die ein Quellobjekt in eine graphische oder beliebige andere Form (z.B. Audio) überführt. Diese Transformation wird ebenfalls wieder über das Attribut *formula*, welches eine spezifische Expression enthält, definiert. CWM legt auch hier die Form, wie diese Expression auszugestalten ist, nicht fest. Bezogen auf das obige Beispiel könnte dies nun bedeuten, dass alle öffentlichen Gebäude in einer spezifischen Farbe oder alle angrenzenden Einbahnstrassen durch eine durchgehende Linie mit bestimmter Strichdicke dargestellt werden. Mögliche Arten des *Renderings* könnten beispielsweise HTML, SVG[52] oder XSL[53] sein, welches das CWM durch die Klasse *XSLRendering* als Spezialisierung der Klasse *Rendering* bereits vorsieht.

Schliesslich umfasst dieses Metamodell noch die Klasse *RenderedObjectSet*. Diese ist ein Suptyp von *Package* und repräsentiert einen einfachen Container für *RenderedObjects* und *Renderings*. Anstelle der Verwendung des Attributs *formula* kann sowohl bei *RenderedObjects* wie auch bei *Rendering* über das Attribut *fileName* ein Verweis auf eine Datei festgelegt werden. Beim *Rendering* könnte die Darstellung damit beispielsweise durch ein Symbology Encoding Document (vgl. Kap. 2.4.3) definiert werden. Eine weitere Möglichkeit besteht in der Angabe einer URL, welche auf eine Darstellungs-Ressource (z.B. ein Styled Layer Descriptor) im Internet verweist.

ExpressionNodes sind vom Typ *ModelElement*. Ein *ExpressionNode* bildet einen Baum aus Ausdrücken, bei dem *ConstantNodes*, *FeatureNodes* und *ElementNodes* beliebig kombiniert und geschachtelt werden können. Die Interpretation ist nicht standardisiert, so dass nur durch Konvention festgelegt werden kann, ob sich hinter einem „Node“ komplexe Rechenoperationen wie zum Beispiel JavaScript für aktive Elemente auf HTML-Seiten verbergen. Erst nach dieser gegenseitigen Verständigung können die Instruktionen in der entsprechenden Laufzeitumgebung ausgeführt werden.

---

[51] *XML Path Language: http://www.w3.org/TR/xpath (25.06.2007)*
[52] *Scalable Vector Graphics: http://www.w3.org/Graphics/SVG (25.06.2007)*
[53] *Extensible Stylesheet Language: http://www.w3schools.com/xsl/xsl_languages.asp (25.06.2007)*

Teil C – Diskussion und Ausblick

5. Beurteilung der Ergebnisse

Dieses Kapitel enthält die Beurteilung der erzielten Resultate und der vorgängig aufgestellten Thesen. Die Eignung von CWM wird a posteriori anhand des GDI-Metamodells nochmals diskutiert und kritisch beleuchtet. Im zweiten Teil folgt eine Beurteilung der a priori aufgestellten Thesen.

5.1 Diskussion der Resultate

Die Verwendung des Common Warehouse Metamodels als Basis eines Metadatenframeworks einer Geodateninfrastruktur hat den Vorteil, dass neben der standardisierten Speicherung von Metadaten über XML Metadata Interchange (XMI) auch der standardisierte Austausch möglich ist. Letztlich ist jedoch in erster Linie der Inhalt dafür entscheidend, dass die benötigten Informationen für die Konfiguration und Steuerung der GDI-Komponenten bereitgestellt werden. Es ist daher sinnvoll, bei den nachfolgenden Betrachtungen nach inhaltlichen und strukturellen Aspekten zu differenzieren.

5.1.1 Inhaltliche Aspekte

Das Common Warehouse Metamodel ist eine domänenspezifische Erweiterung der OMG Metadata Architecture. Es wurde speziell für die Verwaltung von Metadaten im Data Warehouse-Bereich vorgesehen und ist daher konsequent auf diese Bedürfnisse ausgerichtet. Die vollständige Berücksichtigung aller Konzepte im Data Warehousing ist schwierig, da die Anforderungen in diesem Bereich einerseits stark variieren und andererseits eine hohe Dynamik aufweisen. Die Festlegung eines einheitlichen Standards zum Management und Austausch von Metadaten ist aus diesen Gründen nur durch eine sehr generische Sichtweise der Modellierung zu erreichen. Demzufolge erstaunt es nicht, dass einige Bereiche wie die Berechtigungsverwaltung oder das Qualitätsmanagement ausgespart wurden. Grundsätzlich stellt das CWM ein wiederverwendbares und umfangreiches Framework zum Management von Metadaten bereit. Der Hauptfokus ist dabei auf die technischen Metadaten gerichtet.

Mit Hilfe des *Relational*-Pakets der CWM-Foundation-Ebene lässt sich ein relationales Schema wie dasjenige einer Geodatenbank vollständig und korrekt modellieren. Das hier beschriebene GDI-Metamodell beschränkt sich auf die Abbildung der Tabellen- bzw. Sichtenstruktur mit den zugehörigen Beziehungen. Relationale Beziehungen zwischen Entitäten lassen sich über Fremdschlüsselattribute festlegen. Die CWM-Elemente zum Modellieren von Indizes, Prozeduren und Triggern wurden bewusst weggelassen. Diese können bei Bedarf nachträglich mit geringem Aufwand hinzugefügt werden.

Ein ganz wesentlicher Aspekt ist die Möglichkeit zur Abbildung räumlicher Datentypen. Räumliche Objekte zeichnen sich dadurch aus, dass sie ein oder mehrere Attribute vom Typ *Geometry* besitzen. In einer objekt-relationalen Datenbank enthalten Tabellenspalten mit Geometrie-Datentypen einen Fremdschlüssel-Verweis auf eine spezielle Geometrie-Tabelle. Die eigentliche Geometrie ist in dieser separaten Geometrie-Tabelle gespeichert. Gemäss OGC Simple Feature Specification (SFS) enthalten Geometrie-Tabellen Kolonnen mit numerischen oder binären SQL-Datentypen. Auf der Metadatenebene reicht es im Gegensatz zur Implementierungsebene grundsätzlich aus, wenn die Existenz einer Geometriespalte innerhalb einer Tabelle nach aussen bekannt gemacht wird. Die interne Speicherung der Geometrie geht über den Metadaten-Bereich hinaus.

Aus dieser Überlegung heraus bieten sich zwei Ansätze an:

1. Unabhängig von der konkreten Implementierung bzw. Speicherung einer Geometrie wird die Tabelle der einfachen Datentypen (*SQLSimpleType*) um den Typ *Geometry* erweitert. Es muss dabei allerdings umschrieben werden, welche Arten von Geometrien (Punkt, Linie, Fläche) für diesen Typ zulässig sind. Alle zusätzlichen Eigenschaften wie beispielsweise das zugehörige Referenzsystem (engl. spatial reference system) müssten dann in der Tabelle, in welcher die Geometrie verwendet wird als zusätzliche Attribute definiert werden.
2. Durch die Definition eines zusammengesetzten Geometrietyps (*SQLStructuredType*) werden alle notwendigen Informationen gekapselt (Art der Geometrie, SRID[54], etc.). Im Hinblick auf das Modell bedeutet dies, dass dazu eine neue Klasse hinzugefügt werden muss.

Im GDI-Metamodell wurde der zweite Ansatz gewählt. Das CWM sieht die dazu notwendigen Konstrukte in Form der Klasse *SQLStructuredType* bereits vor. Auf der Basis dieses zusammengesetzten Typs wurde durch Vererbung die Klasse *SQLGeometryType* hinzugefügt. Ein *SQLGeometryType* stellt in physischer Hinsicht eine herkömmliche Tabelle dar, welche eigene Attribute wie beispielsweise einen Identifikator für ein räumliches Bezugssystem (SRID) enthalten kann.

Eine Geodatenbank kann mehrere relationale Schemas besitzen. Eine einzelne Datenbankinstanz entspricht im CWM der Klasse *Catalog* und stellt somit eine Datenquelle (*dataPackage*) dar. Ein *dataPackage* ist aber nicht ausschliesslich auf relationale Quellen beschränkt. Es kann auch herkömmliche Dateien in einem beliebigen Format repräsentieren. Der Zugriff auf diese Datenquellen wird im CWM durch das Paket *SoftwareDeployment* abgebildet, wovon das GDI-Metamodell nur einen relativ kleinen Teil verwendet.

[54] *SRID: Spatial Reference System Identifier*

Durch einige wenige Klassen (*DataManager*, *DataProvider*, *ProviderConnection*) lassen sich Client-Server-Beziehungen modellieren, wobei sowohl Datenbank- wie auch Fileverwaltungssysteme als Server in Frage kommen. Die physische Identifikation eines Servers erfolgt über die Klasse *Machine*. Mit Hilfe dieser Informationen sind Schnittstellen für den Zugriff auf Datenquellen vollständig beschrieben. An dieser Stelle wird noch einmal darauf aufmerksam gemacht, dass die Verwaltung von Benutzerrechten bewusst weggelassen wurde. Die an der GDI beteiligten Komponenten sollen uneingeschränkt auf ein zentrales Metadaten-Repository zugreifen können. Der Zugriff von aussen ist über die explizite Angabe eines Berechtigungsnachweises durch einen übergeordneten Sicherheitslayer zu regeln.

Neben der physischen Datenorganisation sollte ein Metadatenframework auch eine für den Benutzer verständliche logische Datenstruktur beinhalten. Im Rahmen dieser Arbeit wurde hierzu auf das *Core*-Paket des CWM zurückgegriffen. Dieses Vorgehen erfordert jedoch gegenüber dem in der GIS-Welt verbreiteten Layer-Konzept ein grundlegendes Umdecken. Ein Layer versteht sich als eine logische Unterscheidung eines darzustellenden Themas, wobei in der Regel jede Informationsschicht einer physischen Datei entspricht. Vektordaten können bei der Kopplung an ein Datenbanksystem zur Speicherung von Attributinformationen Objektcharakter aufweisen. Im Rahmen des CWM-basierten GDI-Metamodells beziehen sich sämtliche Metadaten auf geschlossene Objekte mit Attribut- und Lageinformation. Im CWM entsprechen Objektklassen auf logischer Ebene einem *Namespace* und auf physischer Ebene einer Tabelle. Ein *Namespace* kann durch Komposition und rekursive Vererbung wiederum *Namespaces* enthalten, wodurch sich eine hierarchische Struktur aufbauen lässt. Die Beziehungen zwischen Objekten und ihren Merkmalen werden auf logischer Ebene festgelegt.

Die Rendering-Klassen des GDI-Metamodells beschreiben die graphische Darstellung der in der Geodatenbank gespeicherten Daten. Die attributabhängige Klassifikation und Darstellung basiert auf einem oder mehreren Attributen. Die Darstellung kann für jeden vorkommenden Wert unterschiedlich sein oder für einen numerischen Bereich (mit Ober- und Untergrenze) von Attributwerten gelten (Klassengrenzen). Es ist also über Metadaten festzulegen, auf welchem Attribut die Darstellung bzw. Klassifikation beruht. Das CWM-Paket *InformationVisualization* stellt diese Möglichkeit auf eine sehr generische Art und Weise zur Verfügung. Es besteht keine explizite Verbindung zu den Spalten (*Column*) einer Tabelle. Stattdessen wird die Auswahl und Darstellung der gespeicherten Daten über Ausdrücke (*ExpressionNodes)* festgelegt. CWM schreibt die Form dieser Ausdrücke nicht vor, sondern überlässt dies der jeweiligen Implementierung. Im GDI-Kontext wäre es beispielsweise denkbar, die Auswahl der zu symbolisierenden Objekte über OGC Filter Encoding zu definieren. Die Darstellung könnte dann mittels OGC Symbology Encoding oder OGC Styled Layer Descriptor festgelegt (vgl. Kap. 2.4.3) werden. In jedem Fall ist bei Ausdrücken immer gleichzeitig die referentielle Integrität sicherzustellen. Demnach darf ein *ExpressionNode* nur *FeatureNodes* beinhalten, die in der Metadatenbank tatsächlich existieren. Diese Prüfung erfordert von Seiten der Implementierung einen erheblichen

Mehraufwand. Die Eignung von CWM wird daher für diesen Anwendungsbereich grundsätzlich in Frage gestellt. Der CWM-Klasse *XSLRendering* kommt im GIS-Kontext eine besondere Bedeutung zu. Sie stellt eine Spezialisierung der Klasse *Rendering* für XML-basierte Datenquellen dar. Im Umgang mit Web Features Services (WFS), bei welchen die auszutauschenden Daten in der Geography Markup Language (GML) codiert sind, ist eine XSL-basierte Transformation durchaus sinnvoll.

Das Paket *BusinessInformation* ermöglicht die Abbildung von Metadaten im klassischen Sinne. Jedes Modellierungselement innerhalb von CWM kann mit beschreibender Metainformation versehen werden. Das *BusinessInformation*-Metamodell sieht hierzu die textuelle Beschreibung und die Angabe von Kontaktstellen vor. Zudem können Modellierungselemente mit einer beliebigen Anzahl externer Dokumente verknüpft werden. Für den GIS-Bereich ungewöhnlich ist sicherlich die Tatsache, dass jedes Modellelement des CWM Träger von Metainformationen sein kann. Dies erhöht zwar die Flexibilität, verringert aber im Gegenzug die Übersichtlichkeit und Einfachheit. Das im Rahmen dieser Arbeit erstellte GDI-Metamodell wurde zudem um typische Metadatenelemente aus dem GIS-Bereich angereichert. Beispielhaft wurden hierzu die Elemente des Dublin Core Metadata Element Set (DCES) in Form einer Subklasse von *Description* hinzugefügt. Eine Erweiterung um vergleichbare Metadatenmodelle wie etwa ISO 19115 wäre auf diese Art ebenso möglich. Die Erweiterung des Modells gemäss offiziellem GIS-Standard macht durchaus Sinn. Die explizite Verständigung über den erweiterten Inhalt ist nicht notwendig, da die zugreifenden Catalog Services die Struktur dieser standardisierten Inhalte bereits kennen. Problematisch ist hier allerdings, dass sich die Klasse *DCElementSet* auf sämtliche Modellierungselemente bezieht, was semantisch nicht in jedem Fall zutreffend ist. So ist es beispielsweise nicht korrekt, wenn zu einem *SQLDataType* Dublin Core-Metadaten hinterlegt werden.

5.1.2 Strukturelle Aspekte

Aus struktureller Sicht erweist sich das Common Warehouse Metamodel als sehr generisch. Dies zeigt sich hinsichtlich des CWM-basierten GDI-Metamodells am Beispiel des Pakets *InformationVisualization* sehr deutlich. Mittels drei Basisklassen wird hier ein vielseitiger und komplexer Themenbereich sehr allgemeingültig abgedeckt. Ebenfalls auffallend ist die Tatsache, dass sich das gesamte Metamodell stark um die Klasse *ModelElement* zentriert. Dieser Aspekt ist am Beispiel des *BusinessInformation* Metamodels besonders deutlich erkennbar. Alle drei Basisklassen weisen hier eine Assoziation zur Klasse *ModelElement* auf. Da die Basisklassen selber ebenfalls Spezialisierungen von *ModelElement* sind, können die Basisklassen untereinander wiederum beliebige Beziehungen aufweisen. Auf der Attributebene zeigt sich zudem, dass meist nur grundlegende Attribute wie der Objektname oder etwaige Beschreibungen bzw. Kommentare in den Konstrukten des CWM vorgesehen sind.

Die Erweiterungstypen *Stereotypes* und *TaggedValues* sind einerseits auf den Datentyp String beschränkt und andererseits können keine Beziehungen zu anderen Modellelementen hergestellt werden. Diese Art der Erweiterung ist folglich nur sehr beschränkt einsetzbar. Die objektorientierte Erweiterung entspricht zwar dem Modularitätsgedanken des CWM, sie muss aber von allen am Metadatenaustausch beteiligten Anwendungen implementiert werden. Dennoch bot sich im Rahmen des GDI-Metamodells die Integration von Erweiterungsklassen an. Nur damit sind die Modellierung zusätzlicher, nicht-textueller Attribute und die Festlegung von Objektbeziehungen möglich. Darüber hinaus können die konstruierten Erweiterungsklassen inhaltlich zu Erweiterungs-Paketen zusammengefasst und veröffentlicht werden.

Da Beziehungen im CWM weitgehend auf übergeordneter Modellebene festgelegt wurden, sind bei Spezialisierungen zum Teil Einschränkungen auf Assoziationsebene notwendig. Die Spezialisierung führt hierzu spezifische Beziehungen ein, die auf der spezifischen Ebene zusätzlich gelten (vgl. *Abb. 34*).

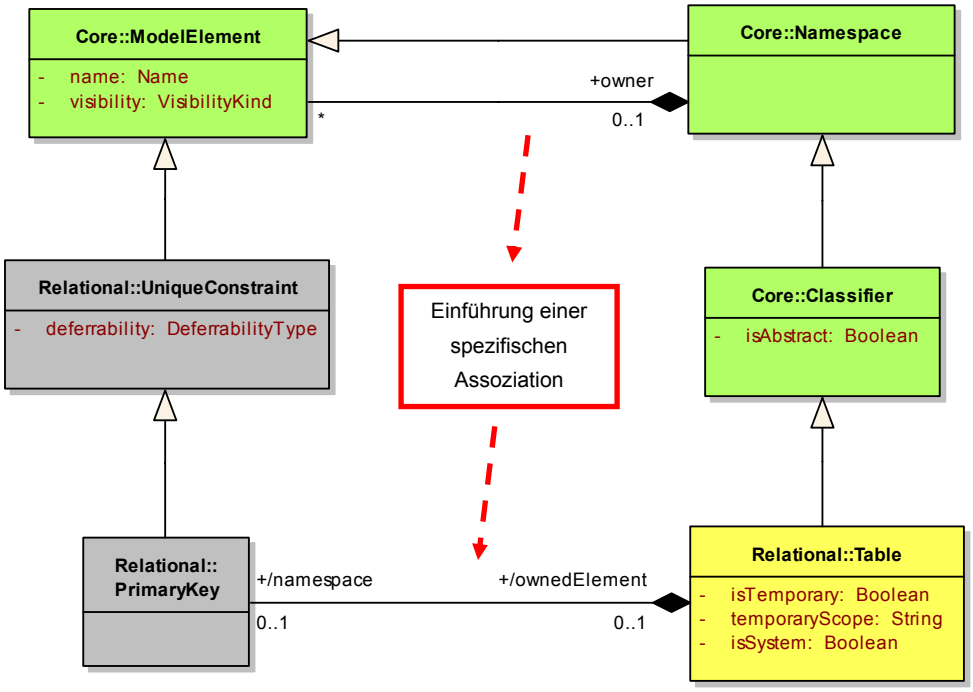

Abb. 34: Definition spezifischer Beziehungen bei Spezialisierungen

Die Beziehung zwischen den Klassen *Table* und *PrimaryKey* entspricht der Komposition zwischen *Namespace* und *ModelElement*. Ein *Namespace* darf demnach mehrere *ModelElements* besitzen. Eine *Table* als Spezialisierung der Klasse *Namespace* hingegen besitzt höchstens einen *PrimaryKey*, der seinerseits eine Spezialisierung der Klasse *ModelElement* darstellt. Die Komposition, welche durch das CWM auf der übergeordneten *Core*-Ebene festgelegt wurde, ist zwischen den Klassen *Table* und *PrimaryKey* durch eine zusätzliche Assoziation weiter zu präzisieren.

Die abstrakte Aussage, dass eine *Table* als *Classifier* beliebige *ModelElements* enthalten kann, gilt daneben aber nach wie vor. Daraus lässt sich schliessen, dass bei einer Spezialisierung neben den allgemeinen auch die speziellen Beziehungen berücksichtigt werden sollen. Dasselbe Prinzip ist auch bei Modellerweiterungen zu beachten. Wenn also aufgrund der Semantik spezifischere Beziehungen nötig werden, sind diese zusätzlich zu den allgemeinen anzugeben.

5.2 Beurteilung der Thesen

1. These: ***Durch eine zentrale Verwaltung der Informationen über die Daten des Kernsystems kann der Konfigurationsaufwand von GDI-Komponenten und der Integrationsaufwand für Geodaten reduziert werden.***

Konfigurationsaufwand entsteht bei einer Geodateninfrastruktur sowohl bei der gelegentlichen Hinzunahme neuer Softwarekomponenten als auch bei der laufenden Integration zusätzlicher Dateninhalte. Mit dem Ziel einer raschen Inwertsetzung von Geodaten wird bei der Datenintegration aufgrund der unterschiedlich strukturierten Quelldaten häufig ein Warehouse-ähnlicher Ansatz (vgl. Geodaten-Warehouse, S. 43) praktiziert. Während sich die zentrale Verwaltung daten- und prozessbezogener Metainformation im IT- und Warehousing-Bereich seit langem etabliert hat und der daraus resultierende Nutzen in zahlreichen Publikationen belegt wurde, ist der Fokus im Geo-Umfeld vorwiegend auf die Haltung von beschreibenden Metadaten zum Auffinden von Geo-Ressourcen beschränkt. Es ist somit naheliegend, den bewährten IT-Ansatz auch im GDI-Kontext anzuwenden und das Management von systemtechnischen Metadaten zu zentralisieren, so dass alle Komponenten die für ihre Konfiguration benötigten Informationen bei einer zentralen Stelle beziehen können. Statt alle beteiligten GDI-Komponenten einzeln konfigurieren zu müssen, reduzieren sich somit die notwendigen Anpassungen auf eine zentrale Metadatenkomponente. Wenn alle GDI-Komponenten auf die gleichen Metadaten zugreifen, können nicht zuletzt auch Redundanzen und Inkonsistenzen vermieden werden.

2. These: ***Ein Metadaten-Repository eignet sich als zentrale Auskunftsstelle besonders für diejenigen GDI-Komponenten, welche konfigurierbar sind.***

Ein Metadaten-Repository stellt die notwenigen Konfigurationsinformationen (Datenstruktur, Datenzugriff, graphische Darstellung, etc.) in Form von technischen Metadaten bereit. Einer flexibel konfigurierbaren Anwendung können die benötigten Metadaten ohne Programmierung, sondern alleine durch technische Anweisungen (i.d.R. Konfigurations-Skripte) bereitgestellt werden. Da GDI-Komponenten in der Regel vielfältig konfigurierbar sind, eignet sich ein Metadaten-Repository folglich als zentrale Auskunftsstelle.

3. These: Den bestehenden Standards im Geoinformationsbereich (OGC und ISO) mangelt es an Möglichkeiten zur Beschreibung und Verwaltung von technischen Metadaten.

Im GIS- bzw. GDI-Umfeld existiert eine stetig wachsende Anzahl an Normen und Standards. Das Open Geospatial Consortium und das ISO/TC 211 haben die Standardisierung in den vergangenen Jahren erheblich vorangetrieben. Mittlerweile haben sich viele dieser Standards etabliert und sind in die Entwicklung von Softwareprodukten eingeflossen. Im GDI-Bereich haben insbesondere ISO 19115 sowie ISO 19119 eine hohe Relevanz. Beide Standards liefern abstrakte Beschreibungen von Metadaten, die im Wesentlichen die semantischen und zum Teil auch die technischen Aspekte der Metainformation für eine externe Bereitstellung umfassen. Dennoch eignen sich die bestehenden Metadatenstandards nicht zur vollständigen Abbildung der internen Struktur der zugrunde liegenden Geo-Ressourcen, da die Beziehungen unter den bestehenden Objektklassen nicht ausreichend modelliert werden können. Die explizite Verwendung technischer Metadaten von bestehenden GI-Standards zur Steuerung und Konfiguration von GDI-Komponenten wurde in der konsultierten Literatur nicht oder nur ansatzweise erwähnt. Der Fokus von bestehenden GI-Standards liegt auf der inhaltlichen Beschreibung von Geo-Ressourcen.

4. These: Im GDI-Umfeld ist eine standardisierte Beschreibung von System-Metadaten analog zu den Ansätzen im Data Warehouse-Bereich notwendig.

Eine Geodateninfrastruktur ist ein Informationssystem bestehend aus technischen Komponenten und Daten. Das Management von Metadaten bietet einen in der IT erprobten Ansatz zur effektiven und effizienten Integration von Softwarekomponenten und Daten. Metadaten erlauben die Wiederverwendung von Entwicklungsbausteinen, so dass Softwarekomponenten zu einem funktionierenden Gesamtsystem verknüpft werden können. Im Weiteren werden Metadaten für die Steuerung und Automatisierung von Betriebs-Prozessen genutzt. Namentlich im Data Warehouse-Bereich wurde dieser Nutzen erkannt. Mit dem Common Warehouse Metamodel ist ein Standard hervorgegangen, der die Handhabung von System-Metadaten werkzeugübergreifend vereinheitlichen soll. Eine standardisierte Beschreibung von System-Metadaten hat auch im GDI-Kontext den Vorteil, dass alle beteiligten Komponenten über eine einheitliche Sprache miteinander kommunizieren können. Da sich in der Folge auch die systemtechnischen Prozesse aller beteiligten Komponenten angleichen, können neu hinzukommende Komponenten die bereits bestehenden Prozesse ebenfalls nutzen. Das im Rahmen dieser Arbeit erstellte GDI-Metamodell stellt einen Entwurf einer standardisierten Beschreibung von technischen GDI-Metadaten dar.

5.3 Fazit der Beurteilung

Die Wahl des CWM als Framework für die Verwaltung von Metadaten bietet hinsichtlich der Erstellung eines GDI-Metamodells eine weitgehend zuverlässige und solide Grundlage. Die thematische Gliederung des Standards in Ebenen und Pakete erlaubt die gezielte Auswahl der benötigten Modellgrundlagen, womit Umfang und Komplexität bereits erheblich reduziert werden. Die physische Datenstruktur kann mittels der bereitgestellten Metamodelle inhaltlich vollständig und strukturell sauber abgebildet werden. Dies schliesst auch die Beschreibungen der Schnittstellen für den Datenzugriff mit ein.

In anderen Bereichen wie beispielsweise der Beschreibung von Benutzer-Metadaten oder der graphischen Darstellung ist das CWM inhaltlich zu wenig spezifisch. Im Portrayal-Bereich liefert die CWM-Spezifikation keine konkreten Anhaltspunkte, wie die praktische Umsetzung auszusehen hat. Die Definition von Ausdrücken (*ExpressionNodes*) in den Rendering-Klassen des *InformationVisualization*-Metamodells ist implementierungsabhängig, so dass sich die beteiligten GDI-Komponenten per Konvention über deren Struktur und Inhalt verständigen müssen. Dieser Teil des CWM ist streng genommen nicht standardisiert, womit bei der praktischen Umsetzung Raum für Interpretationen offen bleibt.

Das Paket *BusinessInformation* bietet lediglich grundlegende Klassen zur Verwaltung von Benutzer-Metadaten. Diese sind stark auf geschäftliche Bedürfnisse ausgerichtet und enthalten nur einen unwesentlichen Bruchteil der Informationen aus ISO 19115 oder Dublin Core. In Ergänzung zu den CWM-Klassen *ResponsibleParty*, *Document* und *Description* wird die Möglichkeit zur qualitativen Beschreibung von Geo-Ressourcen im Rahmen eines GDI-Metamodells als notwendig erachtet. Dieser Aspekt ist durch das CWM bisher nicht abgedeckt. Der Bereich des Qualitätsmanagements kann im *BusinessInformation*-Metamodell des CWM durch die sehr generische Modellierung strukturell nicht sinnvoll integriert werden. Qualität ist kein Modellierungsartefakt im eigentlichen Sinne, sondern beschreibt - je nach Interpretation und Anwendung - Modellierungselemente oder sogar Datenelemente. Wenn in einer Klasse Objekte unterschiedlicher Qualität vorkommen, dann muss die Qualität so modelliert werden, dass zu jeder bestehenden Klasse eine Verbindung zu einer immer vorhandenen Klasse *Quality* besteht. Qualität wird so zum Datenelement und liegt dann nicht mehr auf der Metaebene.

Diese Problematik lässt erkennen, dass im CWM die Meta-Meta-, Meta-, und Datenebene im Sinne eines möglichst allgemeingültigen Modells teilweise nicht immer sauber getrennt sind. Während das *Object Model* zwecks Beschreibung der Artefakte der Modellierung auf der Modellierungsebene (Meta-Meta) einzuordnen ist, befinden sich die Klassen der *Resource*-Schicht auf der Modellebene (Meta). Hierbei handelt es sich um konkrete Klassen, durch die ein konkretes Metadatenmodell erstellt werden kann. Als Beispiel einer Klasse, welche auf der Datenebene einzuordnen ist, sei hier stellvertretend die Klasse *Location* aus dem *BusinessInformation*-Metamodell genannt, durch welche sich Adressen von Kontaktpersonen abbilden lassen.

Insgesamt lassen sich die GDI-spezifischen Anforderungen durch die niedrige Definitionstiefe des Referenzmodells nur zum Teil durch das CWM abbilden. In den einzelnen Metamodellen werden mit dem Hintergrund einer möglichst breiten Unterstützung nur die allgemein benötigten Elemente und Beziehungen definiert. Die Konstruktion eines konzeptionellen Metamodells einer Geodateninfrastruktur auf Basis des CWM erscheint aus folgenden Gründen dennoch sinnvoll:

- Mittels CWM kann zumindest ein wesentlicher Teil der technischen GDI-Metadaten, nämlich die physische Datenstruktur, korrekt und vollständig abgebildet werden.
- Neben der standardisierten Speicherung von technischen Metadaten ist über XML Metadata Interchange (XMI) auch deren standardisierter Austausch möglich.
- Unter Anwendung der bereitgestellten Erweiterungsmechanismen können eigene Modellteile ergänzt und später als Extension Packages zur Weitergabe veröffentlicht werden.

6. Zusammenfassung und Ausblick

Diese Arbeit hat einen Einblick in die Thematik von Geodateninfrastrukturen und Metadaten vermittelt. Dabei wurde ein Verständnis für Metadaten geschaffen, welches sich stark am Data Warehouse-Konzept orientiert. Metadaten sind in diesem Kontext mehr in technischer und weniger in semantischer Hinsicht zu betrachten. Durch das Management von Metadaten sollten GDI-Komponenten und Geo-Ressourcen effektiv und effizient integriert und zu einem gut funktionierenden Gesamtsystem verknüpft werden können.

Der gewählte Lösungsansatz orientierte sich insbesondere an bestehenden Konzepten und Standards aus der Informationstechnologie (IT). Er gründete auf der Idee, dass alle an einer GDI beteiligten Komponenten zwecks Konfiguration aus einem zentralen Repository mit Metadaten versorgt werden können. Die Konfiguration einzelner GDI-Komponenten wurde beispielhaft anhand ausgewählter Open Source-Applikationen untersucht. Nach einer vertieften Analyse wurde das Common Warehouse Metamodel (CWM) als konzeptionelle Grundlage für die Erstellung eines GDI-Metadatenframeworks ausgewählt. Der Hauptteil dieser Arbeit bestand in der Erstellung eines konzeptionellen Metamodells für GDI-Metadaten in Form eines UML-Objektmodells. Hierzu wurde das CWM auf die GDI-relevanten Pakete und Klassen begrenzt und durch räumliche Aspekte erweitert. Das CWM bot hinsichtlich der Erstellung des GDI-Metamodells eine solide Ausgangsbasis, mit welcher die benötigten Konfigurations-Metadaten zumindest in den wichtigen Kernbereichen sauber abgebildet werden konnten. Es hat sich aber auch gezeigt, dass der CWM-Ansatz zur Abdeckung einiger Bedürfnisse inhaltlich zu generisch ist. Insbesondere wird in der aktuellen Version des CWM ein Qualitätsmanagement-Metamodell vermisst. Im Rahmen einer Weiterentwicklung des vorliegenden GDI-Metamodells wäre eine Erweiterung um Qualitäts-Metadaten unter Bezugnahme auf die ISO-Normen 19115/19119 sicherlich sinnvoll.

Die nächsten Schritte zur Realisierung eines integrierten Repositories für GDI-Metadaten würden die Erstellung des konzeptionellen Metadatenmodells sowie dessen Implementierung in einer relationalen Datenbank beinhalten. Damit würde eine essentielle Grundlage für ein universelles Metadatenmanagement geschaffen, dass die Verwaltung, Pflege sowie Nutzung von Metadaten umfasst. Das vorliegende GDI-Metamodell ist dazu in ein relationales Datenbankschema zu transformieren. Ein mögliches Vorgehen in Bezug auf eine praktische Umsetzung der in dieser Arbeit gewonnenen Erkenntnisse ist in der nachfolgenden Abbildung dargestellt. Ein alternativer Lösungsweg zur Abbildung des GDI-Metamodell auf ein relationales Datenschema könnte der von POOLE et al. (2003) entwickelte CWM Metastore[55] darstellen.

[55] *Demo-Download: http://www.wiley.com/legacy/compbooks/poole/CWM_Guide/software.html (25.06.2007)*

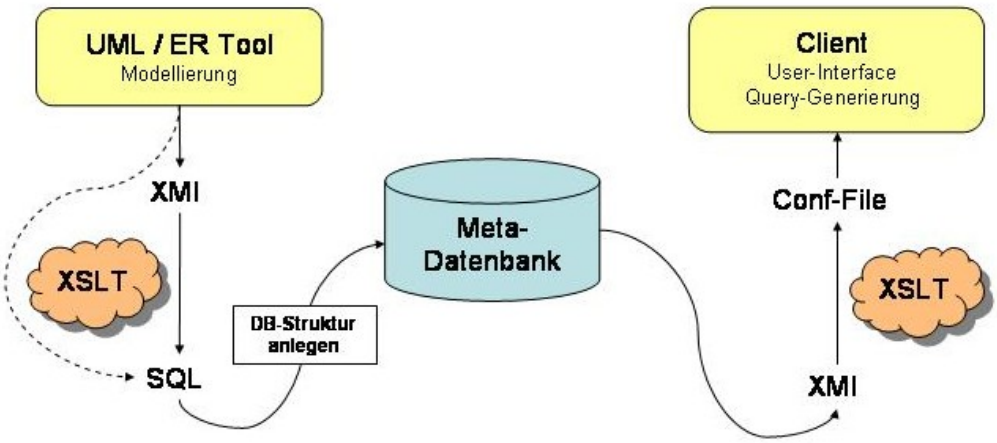

Abb. 35: Mögliches Vorgehen bei der praktischen Umsetzung

Das hier schematisch aufgezeigte Vorgehen lehnt sich an die in MARCO und JENNINGS (2004) vorgestellte Managed Meta Data Environment (MME) an:

1. Sourcing Layer → UML-Tool
2. Integration Layer → XMI / SQL
3. Metadata Repository → Meta-Datenbank
4. Delivery Layer → XMI / MAP

Das CWM-basierte GDI-Metamodell kann hierzu mittels UML-Werkzeugen in eine XMI-Datei exportiert und anschliessend über XSLT in ein logisches SQL-Schema transformiert werden. Nachdem das logische Schema erstellt wurde, ist das Repository-Schema physisch in einer Meta-Datenbank abzubilden. Der Export der zur Konfiguration benötigten Metadaten erfolgt über geeignete XMI-Schnittstellen (IDL, CORBA) wiederum auf standardisierte Weise. Abschliessend wird der Inhalt der XMI-Datei unter Verwendung von XSLT auf ein werkzeugbezogenes Konfigurationsfile (z.B. MAP-File des UMN MapServer) abgebildet. Auf diese Weise kann die Konfiguration von Middleware-Komponenten einer GDI weitgehend automatisiert werden. Der Aufwand bei der Hinzunahme neuer Daten oder Dienste würde sich dadurch erheblich reduzieren. Die in dieser Arbeit erarbeiteten Grundlagen und die daraus gewonnenen Erkenntnisse sind als initialer Beitrag zur Standardisierung von technischen Metadaten im GDI-Umfeld zu sehen. Deren Tauglichkeit für die Praxis ist in weiteren Arbeiten und Studien durch eine prototypische Umsetzung noch ausführlich zu belegen.

Literaturverzeichnis

AMSTEIN J-P. (2004): Aus der Schweiz: Von der Bundesstrategie für Geoinformation zur Nationalen Geodaten-Infrastruktur. Intergeo 2004, Stuttgart, http://www.intergeo.de/deutsch/page/kongress/downloads/archiv/2004/Amstein.pdf (25.06.2007).

BAUER A., GÜNZEL H. (2001): Data Warehouse Systeme - Architektur, Entwicklung, Anwendung. dpunkt-Verlag, Heidelberg.

BERNARD L., EINSPANIER U., STREIT U. (2001): Developing OpenGIS Catalog Services for a GDI - Lessons learned. 4. AGILE Conference, Brno, Czech Republic, http://www.agile-secretariat.org/Conference/brno2001/proceedings/41.pdf (25.06.2007).

BERNARD L., STREIT U. (2002): Geodateninfrastrukturen und Geoinformationsdienste: Aktueller Stand und Forschungsprobleme. Publikation der Deutschen Gesellschaft für Photogrammetrie und Fernerkundung (Band 11).

BERNARD L. (2004): Geodateninfrastrukturen in der Europäischen Informationsgesellschaft. Intergeo 2004, Stuttgart, http://www.intergeo.de/deutsch/page/kongress/downloads/archiv/2004/Bernard.pdf (25.06.2007).

BERNARD L., CROMPVOETS J., FITZKE J. (2004): Geodateninfrastrukturen - ein Überblick. In: BERNARD L., FITZKE J., WAGNER R. M. (Hrsg.) (2004): Geodateninfrastruktur: Grundlagen und Anwendungen. Herbert Wichmann Verlag, Heidelberg.

BRÜGGEMANN H., KLEEMANN S. (2004): GDI-Initiativen in Deutschland. In: BERNARD L., FITZKE J., WAGNER R. M. (Hrsg.) (2004): Geodateninfrastruktur: Grundlagen und Anwendungen. Herbert Wichmann Verlag, Heidelberg.

BUSSE S. (1998): Modellbasierter Entwurf föderierter Informationssysteme. Fraunhofer-Institut für Software- und Systemtechnik ISST, Berlin, http://cis.cs.tu-berlin.de/~sbusse/publications/1998/gi-db.pdf (25.06.2007). In: 10. Workshop "Grundlagen von Datenbanken". Arbeitskreis "Grundlagen von Informationssystemen" im GI-Fachausschuss 2.5, Konstanz. Konstanzer Schriften in Mathematik und Informatik, Nr. 63, Mai 1998, Universität Konstanz, S. 2-6.

BUSSE S. (2002): Modellkorrespondenzen für die kontinuierliche Entwicklung mediatorbasierter Informationssysteme. Dissertation am Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, Logos Verlag, Berlin, http://cis.cs.tu-berlin.de/~sbusse/publications/2002/dissertationBusse.pdf (25.06.2007).

CHONOLES M. J., SCHARDT J. A. (2003): UML 2 for Dummies. Wiley Publishing Inc., New York.

DEVLIN B. (1997): Data Warehouse: From architecture to implementation. Addison Wesley, New York.

DO H. H., RAHM E. (2000): On Metadata Interoperability in Data Warehouses. Technical Report Nr. 01, Institut für Informatik, Universität Leipzig, http://lips.informatik.uni-leipzig.de:80/pub/2000-13 (25.06.2007).

DREWNAK J. (2003): Authentifizierung und Autorisierung in Geodateninfrastrukturen am Beispiel der GDI NRW. Diplomarbeit am Institut für Geoinformatik der Westfälischen Wilhelms-Universität Münster, http://ifgi.uni-muenster.de/downloads/diplomarbeiten_intern/Drewnak/sjfioewtr6t446etr46tr64trerw.pdf (25.06.2007).

DROZ M., DIESBACH M., FANGER L., HAUSER S. P. (2006): GeoDBmeta - Metadatenbank des Kantons Bern: Situationsanalyse. Studienarbeit, Universitätslehrgang „Geographical Information Science & Systems“ (UNIGIS MSc), Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

EBERLE H. (2006): Realisierung eines Umweltinformationssystems mit IT-Standard Methoden. Master Thesis, Universitätslehrgang „Geographical Information Science & Systems“ (UNIGIS MSc), Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

FORNEFELD M., OEFINGER P., JAENICKE K. (2004): Nutzen von Geodateninfrastrukturen. MICUS Management Consulting GmbH, Düsseldorf, http://www.micus.de/pdf/micus_gdi_studie_12_10_2004.pdf (25.06.2007).

FRANKEL D. S. (2003): Model Driven Architecture - Applying MDA to Enterprise Computing. Wiley Publishing Inc., Indianapolis.

GERACI A. (1991): IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. Institute of Electrical and Electronics Engineers Inc., New York.

GIGER C. (2005): Raumbezogene Informationssysteme III - Nationale Geodateninfrastruktur, Teil 1 und 2. Institut für Geodäsie und Photogrammetrie, Eidgenössische Technische Hochschule Zürich, http://www.geoit.ethz.ch/education/vorlesung_ris3_de.html (17.10.2006).

GOEBEL C. (2005): Common Warehouse Metamodel und Imperfektion. Seminararbeit an der Fakultät für Informatik, Universität Karlsruhe, http://www.ipd.uka.de/~ovid/Seminare/DWSS05/Ausarbeitungen/Seminar-DWSS05%20-%2005.pdf (25.06.2007).

GROOT R., McLAUGHLIN J. (2000): Geospatial Data Infrastructure: Concepts, Cases, and Good Practice. Oxford University Press, Oxford.

HAAG A. (2004): Konzeption und Umsetzung einer Erweiterung des Common Warehouse Metamodel (CWM) zur Beschreibung von imperfekten Daten. Studienarbeit an der Fakultät für Informatik, Universität Karlsruhe, http://www.ipd.uka.de/~ovid/Public/Ausarbeitung%20-%20Studienarbeit%2007%20-%20Erweiterung%20CWM.pdf (26.06.2007).

HAAS H., BROWN A. (2004): Web Services Glossary. W3C Working Group, http://www.w3.org/TR/ws-gloss (25.06.2007).

HAHNE M. (2005): Das Common Warehouse Metamodel als Referenzmodell für Metadaten im Data Warehouse und dessen Erweiterung im SAP® Business Information Warehouse. cundus AG, Bretzenheim, http://www.hahneonline.com/paper/BTW2005_Hahne_paper.pdf (25.06.2007).

HUBER M. (2006a): Bisherige Ansätze der Interoperabilität - Modul „OpenGIS und verteilte GeoInformationsverarbeitung“, Lektion 3. Universitätslehrgang „Geographical Information Science & Systems“ (UNIGIS MSc), Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

HUBER M. (2006b): Von der Internet-Applikation zum Web-Dienst - Modul „OpenGIS und verteilte GeoInformationsverarbeitung“, Lektion 9. Universitätslehrgang „Geographical Information Science & Systems“ (UNIGIS MSc), Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

HUBER M. (2006c). Der OGC Catalogue Service - Modul „OpenGIS und verteilte GeoInformationsverarbeitung“, Lektion 12. Universitätslehrgang „Geographical Information Science & Systems“ (UNIGIS MSc), Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

HUBER M. (2006d): Weshalb ist GIS-Interoperabilität wichtig? – Modul „OpenGIS und verteilte GeoInformationsverarbeitung“, Lektion 1. Universitätslehrgang „Geographical Information Science & Systems“ (UNIGIS MSc), Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

HUFFORD D. (1997): Metadata Repositories: The Key to Unlocking Information in Data Warehouses. In BARQUIN R. C., EDELSTEIN H. (Hrsg.) (1997): Planning and Designing the Data Warehouse. Prentice Hall International, New Jersey, S.225-238.

INMON W. H. (1996): Building the Data Warehouse, 2. Auflage. John Wiley & Sons Inc., New York et al.

JOSSEN C. (2004): Metadaten Management Architektur in einer heterogenen Data Warehouse Umgebung. Diplomarbeit, Institut für Informatik der Universität Zürich, http://www.ifi.unizh.ch/archive/mastertheses/DA_Arbeiten_2004/Jossen_Claudio.pdf (25.06.2007).

KELLER S. F. (2003): Modellarten. Dokument der Serie „Model-driven Method Memos“, Center int>e>gis, Institut ITA, Hochschule für Technik, Rapperswil, http://www.integis.ch/documents/MM_Modellarten_v09de.pdf (25.06.2007).

KELLER S. F. (2006a): Have a nice metadata. UNIGIS_Offline, Nr. 29 4/06, S. 1, http://www.unigis.at/club/offline/archiv/UNIGISoffline_4_06.pdf (25.06.2007).

KELLER S. F. (2006b): Geo-Metadaten… Normierung, Auffindbarkeit und Diensteverknüpfung. UNIGIS-Update 10/06, Zentrum für GeoInformatik (Z_GIS), Paris Lodron-Universität Salzburg.

KELLER S. F. (2006c): OSGeodata metadata exchange model. Wiki, GISpunkt, Hochschule für Technik, Rapperswil, http://gis.hsr.ch/wiki/OSGeodata_metadata_exchange_model (25.06.2007).

KIEHLE C. (2006): Geodaten standardisiert integrieren: Lokales abstrakt. iX Magazin für professionelle Informationstechnik, Ausgabe Dezember.

KOGIS (2003): Umsetzungskonzept zur Strategie für Geoinformation beim Bund. Bundesamt für Landestopographie, Wabern, http://www.swisstopo.ch/pub/down/about/kogis/NGDI/KOGIS_BR_Juni03_Konzept_de.pdf (25.06.2007).

KRCMAR H. (2004): Informationsmanagement, 4. Auflage. Springer-Verlag, Berlin, Heidelberg.

KUTSCHE R-D., RÖTTGERS J. (1999): Praxisgerechte Entwicklungsmethoden für Umweltinformationssysteme: Perspektiven für UIS 2000++. 13. Internationales Symposium Informatik für den Umweltschutz der Gesellschaft für Informatik (GI), Metropolis, Marburg, http://enviroinfo.isep.at/UI%2099/UI99-Kutsche.pdf (30.01.2007).

LEGLER F., ZUBOV A. (2002): Metadaten im Data Warehouse. Term Paper, Wirtschaftswissenschaftliche Fakultät, Humboldt-Universität zu Berlin. http://samy.informatik.hu-berlin.de/~legler/studium/studienarbeit/legler_studienarbeit.pdf (24.11.2006).

LIEBERMAN J. (2003): OpenGIS Web Services Architecture. OpenGIS Discussion Paper, Version 0.3, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=1320 (25.06.2007).

MARCO D., JENNINGS M. (2004): Universal Meta Data Model, Wiley Publishing Inc., Indianapolis.

MELCHERT F., AUTH G., HERRMANN C. (2002): Integriertes Metadatenmanagement für das Data Warehousing - Grundlagen, Nutzenpotenziale, Architektur. Institut für Wirtschaftsinformatik, Universität St. Gallen, http://web.iwi.unisg.ch/org/iwi/iwi_pub.nsf/wwwPublAuthorGer/1CFBC6C033C6CC28C12570A30051DFDC/$file/CCDW2%20Arbeitsbericht%2005%20(FME_GAT_CHN).pdf (25.06.2007).

MELCHERT F. (2003): Das Common Warehouse Metamodel als Standard für Metadaten im Data Warehousing. In: VON MAUR E., WINTER R. (Hrsg.) (2003): Data Warehouse Management: Das St. Galler Konzept zur ganzheitlichen Gestaltung der Informationslogistik. Springer Verlag, Berlin, Heidelberg.

MELCHERT F., SCHWINN A., HERRMANN C. (2003): Das Common Warehouse Metamodel - ein Referenzmodell für Data-Warehouse-Metadaten. Institut für Wirtschaftsinformatik, Universität St. Gallen, http://web.iwi.unisg.ch/org/iwi/iwi_pub.nsf/wwwPublAuthorEng/D41F374672070979C12570A3005EDE36 (25.06.2007).

MENGE F. (2006): Geodateninfrastrukturen - „Geo-Disziplinen“ und ihre Ergebnisse wachsen zusammen. Fachbereich Stadtplanung und Umweltschutz, Stadt Braunschweig, http://www.ife.uni-hannover.de/mitarbeiter/seeber/seeber_65/pdf_65/meng12.pdf (25.06.2007).

MORALES J. (2004): Model-driven Design of Geo-Information Services. Dissertation, International Institute for Geo-Information Science and Earth Observation, Enschede, http://asna.ewi.utwente.nl/research/Ph.D.%20Theses/morales-phd-thesis.pdf (25.06.2007).

MORENI C., RIEDO M., GOLAY F., GIGER C., NAJAR C. (2003): Vorstudie zum Projekt e-geo.ch - Organisatorische und technische Aspekte, http://www.kogis.ch/docs/egeo_d.zip (25.06.2007).

MÜLLER M., MacGILL J. (2005): Styled Layer Descriptor Application Profile of the Web Map Service. Draft Implementation Specification, Version 1.1.0, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=12637 (25.06.2007).

MÜLLER M. (2006): Symbology Encoding. OpenGIS Implementation Specification, Version 1.1.0, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=16700 (25.06.2007).

NAJAR C. (2006): A model-driven approach to management of integrated metadata - spatial data in the context of spatial data infrastructures. Dissertation Nr. 16474, Institute of Geodesy and Photogrammetry, Eidgenössische Technische Hochschule Zürich, http://e-collection.ethbib.ethz.ch/show?type=diss&nr=16474 (12.04.2007).

NEBERT D. (2004): Developing Spatial Data Infrastructures: The SDI Cookbook. Version 2.0, Global Spatial Data Infrastructure, http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf (25.06.2007).

NEBERT D., REED C., WAGNER R.M. (2006): Proposal for a Compatible SDI Standards Suite, “SDI 1.0”, GSDI-9 Conference Proceedings, Santiago, Chile, http://www.gsdi9.cl/english/papers/TS19.1paper.pdf (25.06.2007).

NOGUERAS-ISO J., ZARAZAGA-SORIA F. J., MURO-MEDRANO P. R. (2005): Geographic Information Metadata for Spatial Data Infrastructures: Resources, Interoperability and Information Retrieval. Springer Verlag, Berlin, Heidelberg.

OGC (1999): The OpenGIS Abstract Specification - Topic 13: Catalog Services. Version 4, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=901 (25.06.2007).

OMG (2003): Common Warehouse Metamodel (CWM) Specification. Version 1.1, Volume 1, http://www.omg.org/docs/formal/03-03-02.pdf (25.06.2007).

PERCIVALL G. (2002): The OpenGIS Abstract Specification - Topic 12: OpenGIS Service Architecture. Version 4.3, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=1221 (25.06.2007).

PERCIVALL G. (2003). OGC Reference Model. Version 0.1.3, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=3836 (25.06.2007).

PICHLER G., KLOPFER M. (2004): Spezifikation und Standardisierung - OGC, OGC Europe und ISO. In: BERNARD L., FITZKE J., WAGNER R. M. (Hrsg.) (2004): Geodateninfrastruktur, Grundlagen und Anwendungen. Herbert Wichmann Verlag, Heidelberg.

POOLE J., CHANG D., TOLBERT D., MELLOR D. (2003): Common Warehouse Metamodel Developer's Guide. Wiley Publishing Inc., Indianapolis.

RAJABIFARD A., FEENEY M-E. F., WILLIAMSON I. P. (2002): Future directions for SDI development. International Journal of Applied Earth Observation and Geoinformation, Volume 4, Number 1, Elsevier, S. 11-22.

RAJABIFARD A., FEENEY M-E. F., WILLIAMSON I. P., MASSER I. (2003): National SDI Initiatives. In: WILLIAMSON I. P., RAJABIFARD A., FEENEY M-E. F. (Hrsg.) (2003): Developing Spatial Data Infrastructures: From concept to reality. Taylor and Francis, London, New York.

SCHNEEBERGER R. (2005): Nationale Profile der internationalen Standards am Beispiel Metadaten. Tagung zum Thema „Interoperabilität für die breite Nutzung von Geoinformation“, Institute of Geodesy and Photogrammetry, Eidgenössische Technische Hochschule Zürich, http://www.gis.ethz.ch/Interoperability2005/Text/Interop_15_DE.pdf (25.06.2007).

SCHNEIDER A., WEISSHAAR D. (2003): IT-Kompendium - IT-Know-how von A-Z, Transtec AG, Tübingen.

SCHUSTER G. (2003): Entwicklung von Komponenten eines Geoportal-Frameworks - Integration von Internet-Mapserver-Funktionalität in ein Content Management System. Diplomarbeit, Institut für Umweltwissenschaften, Hochschule Vechta, http://www.geographie.uni-freiburg.de/ipg/personen/schuster_grit/dipl/node5.html (25.06.2007).

SENKLER K. (2006): Standardisierung von Metadaten. 10. Workshop KGIS, TU Darmstadt, http://www.ikgis.de/Web/Veranstaltungen/KGIS_Workshop/KGIS_10/Vortraege/Senkler1/vortrag_senkler1.pdf (25.06.2007).

SOGI (2005): Geo-Webdienste. Bericht der Fachgruppe GIS-Technologie SOGI, http://www.sogi.ch/sogi/Geo_Webdienste.pdf (25.06.2007).

STAAB S. (2004): Moderne betriebliche Anwendungen von Datenbanksystemen. Vorlesungsunterlagen, Universität Koblenz, http://www.uni-koblenz.de/~staab/lehre/ws0405/db1/Kapitel15-Teil1-neu.pdf (25.06.2007).

STARK H-J., ANNEN A., STIERLI C., SCHÜTZ S., WIEDMER H. U. (2006): eCH 0056 - Anwendungsprofil Geodienste. Version 1.0, http://www.ech.ch/index.php?option=com_docman&task=doc_download&gid=768&Itemid=181&lang=de (25.06.2007).

THIELE R. (2006): Entwicklung ISO/OGC-konformer Metadaten und Katalogdienste. 43. Sitzung der Arbeitsgruppe „Automation in der Kartographie“, Potsdam, http://www.ikg.uni-hannover.de/aga/pdf-files-aga2006/thiele_AgA2006.pdf (25.06.2007).

TRNINIC M. (2006): Symbology Management. OGC Discussion Paper 05-112, Version 0.2.1, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=13285 (25.06.2007).

VON MAUR E., SCHELP J., WINTER R. (2003): Integrierte Informationslogistik - Stand der Entwicklungstendenzen. In: VON MAUR E., WINTER R. (Hrsg.) (2003): Data Warehouse Management: Das St. Galler Konzept zur ganzheitlichen Gestaltung der Informationslogistik. Springer Verlag, Berlin, Heidelberg,

WALSH J. (2006): Have a nice metadata. http://mappinghacks.com/2006/09/18/have-a-nice-metadata (25.06.2007).

WIENER J. (2000): Meta Data in Context. Artikel in DM Direct Newsletter, http://www.dmreview.com/article_sub.cfm?articleId=1926 (25.06.2007).

WILLIAMSON I. P. (2003): SDIs - Setting the Scene. In: WILLIAMSON I. P., RAJABIFARD A., FEENEY M-E. F. (Hrsg.) (2003): Developing Spatial Data Infrastructures: From concept to reality. Taylor and Francis, London, New York.

WOODWARD B., WHITESIDE A. (2005): Feature Portrayal Service. OGC Discussion Paper 05-110, Version 0.0.30, Open Geospatial Consortium Inc., http://portal.opengeospatial.org/files/?artifact_id=13186 (25.06.2007).

ZACHMANN A. J. (1978): A framework for information system architecture. In: IBM Systems Journal 26, neu aufgelegt in: IBM Systems Journal 38 (1999).

ZEHNDER, C. A. (1998): Informationssysteme und Datenbanken. vdf Hochschulverlag AG, Zürich.

Anhang

Konfiguration von Open Source Middleware-Komponenten

Die Konfiguration der im Rahmen dieser Arbeit ausgewählten GDI-Komponenten wird hier nicht detailliert und vollständig erläutert. Zudem ist darauf hinzuweisen, dass bei den abklärenden Vorarbeiten nicht vom gesamten Konfigurationsumfang Gebrauch gemacht wurde. Für sämtliche hier verwendete Software-Komponenten sind vergleichbare Alternativen verfügbar, auf welche an dieser Stelle jedoch nicht eingegangen wird.

UMN MapServer

Quellen: http://www.umn-mapserver.de/doc44/mapfile-reference44.html (25.06.2007)
http://mapserver.gis.umn.edu/docs/reference/mapfile (25.06.2007)

Die UMN MapServer Software ist eine Open Source Middleware-Komponente für die serverseitige Erstellung von dynamischen Karten. Durch seine hohe Skalierbarkeit und Stabilität bietet der UMN MapServer umfassende Möglichkeiten, um GIS-Funktionalität auf der Basis von Webtechnologien einer breiten Anwendergemeinschaft verfügbar zu machen.

Der UMN MapServer unterstützt eine Reihe von Standards des Open Geospatial Consortiums (OGC). In der aktuellen Version werden WMS (Client/Server) und Non-Transactional WFS (Client/Server) unterstützt. Durch die Integration dieser OGC Standards kann der UMN MapServer als zentrales Bindeglied in heterogenen Architekturen, z.B. als Kernkomponente einer Geodateninfrastruktur, eingesetzt werden.

Die klassische UMN MapServer CGI-Anwendung bietet zahlreiche "out-of-the-box" Features wie z.B. die vorlagengesteuerte Ausgabe, die massstabsabhängige Kartendarstellung, die „on-the-fly“ Projektion zwischen verschiedenen Referenzsystemen, die Einbindung von kartographischen Elementen (Massstab, Übersichtskarte, und Legende) und die Erstellung thematischer Karten mit Klassifikationen durch logische und reguläre Ausdrücke. Über die MapScript-API (Application Programming Interface) können Funktionen und Methoden via Skriptsprachen angestossen werden. Dies erlaubt die Entwicklung von eigenständigen Anwendungen und Fachschalen zur Integration von räumlichen Daten. Weitere Informationen sind den oben angeführten Quellen zu entnehmen.

Beim UMN MapServer beschränkt sich die Konfiguration auf ein so genanntes MAP-File. Alles was mit einer bestimmten Applikation verbunden wird, ist hier definiert. Die meisten Optionen können zur Laufzeit über ein Webformular (CGI Variables) geändert werden. Die MAP-Datei hat eine hierarchische Struktur mit dem MAP-Objekt als Basis-Element. Alle anderen Objekte werden hier untergeordnet.

Beispiel: Konfiguration des UMN MapServers als WMS

```
MAP
  SIZE 950 600
  EXTENT -165 -120 165 120
  UNITS DD
  IMAGECOLOR 200 240 255
  IMAGETYPE PNG

  SCALEBAR
    STATUS EMBED
    UNITS KILOMETERS
    INTERVALS 3
    TRANSPARENT TRUE
    OUTLINECOLOR 0 0 0
  END

  #Global CRS
  #EPSG WGS84
  PROJECTION
    "init=epsg:4326"
  END

  SYMBOL
    NAME 'circle'
    TYPE ELLIPSE
    POINTS 1 1 END
    FILLED TRUE
  END

  WEB
    TEMPLATE global.html
    IMAGEPATH "/ms4w/tmp/ms_tmp/"
    IMAGEURL "/ms_tmp/"
    METADATA
       "WMS_TITLE"              "World and Africa Map"
       "WMS_ABSTRACT"          "African Countries and various hotspots"
       "WMS_SRS"               "EPSG:4326"
       "WMS_SERVER_VERSION"    "1.1.1"
       "WMS_ONLINERESOURCE"    "http://localhost/pgwms.php"
       "WMS_FORMATLIST"  "image/jpeg,image/gif,image/png,image/png;256,image/png;ni"
    END
  END

LAYER
  NAME african_countries
  TYPE POLYGON
  STATUS DEFAULT
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=postgres password=pg4scuba dbname=mysdi host=localhost port=5432"
  DATA "the_geom from african_countries"
  CLASS
    NAME "African_countries"
    # TEMPLATE
    COLOR 235 205 88
    OUTLINECOLOR 80 80 80
  END
  METADATA
    "wms_title" "african_countries"
    "wms_srs" "EPSG: 4326"
  END
END
```

```
LAYER
  NAME african_rivers
  TYPE LINE
  STATUS DEFAULT
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=postgres password=pg4scuba dbname=mysdi host=localhost port=5432"
  DATA "wkb_geometry from african_rivers"
  MINSCALE 1000
  MAXSCALE 100000000
  CLASS
    NAME "African_rivers"
    # TEMPLATE
    OUTLINECOLOR 0 128 255
  END
  METADATA
    "wms_title" "african_rivers"
    "wms_srs" "EPSG: 4326"
  END
END

LAYER
  NAME african_animals
  TYPE POINT
  STATUS DEFAULT
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=postgres password=pg4scuba dbname=mysdi host=localhost port=5432"
  DATA "wkb_geometry from african_animals"
  MINSCALE 1000
  MAXSCALE 50000000
  CLASS
    NAME "African_animals"
    # TEMPLATE
    STYLE
       SYMBOL 'circle'
       SIZE 6
       OUTLINECOLOR 100 100 100
       COLOR 15 160 15
    END
  END
  METADATA
    "wms_title" "african_animals"
    "wms_srs" "EPSG: 4326"
  END
END

LAYER
  NAME african_cities
  TYPE POINT
  STATUS DEFAULT
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=postgres password=pg4scuba dbname=mysdi host=localhost port=5432"
  DATA "wkb_geometry from african_cities"
  MINSCALE 1000
  MAXSCALE 50000000
  CLASS
    NAME "African_cities "
    # TEMPLATE
    STYLE
       SYMBOL 'circle'
       SIZE 6
       OUTLINECOLOR 100 100 100
       COLOR 128 0 128
    END
```

```
  END
  METADATA
    "wms_title" "african_cities"
    "wms_srs" "EPSG: 4326"
  END
END

LAYER
  NAME african_mines
  TYPE POINT
  STATUS DEFAULT
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=postgres password=pg4scuba dbname=mysdi host=localhost port=5432"
  DATA "wkb_geometry from african_mines"
  MINSCALE 1000
  MAXSCALE 50000000
  CLASS
    NAME "African_mines "
    # TEMPLATE
    STYLE
       SYMBOL 'circle'
       SIZE 6
       OUTLINECOLOR 100 100 100
       COLOR 255 0 0
    END
  END
  METADATA
    "wms_title" "african_mines"
    "wms_srs" "EPSG: 4326"
  END
END

LAYER
  NAME country_labels
  TYPE ANNOTATION
  STATUS DEFAULT
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=postgres password=pg4scuba dbname=mysdi host=localhost port=5432"
  DATA "the_geom from african_countries"
  LABELITEM "name"
  CLASS
    LABEL
      COLOR 255 255 255
      OUTLINECOLOR 0 0 0
      POSITION CC
      MINFEATURESIZE 45
    END
  END
  METADATA
    "wms_title" "country_labels"
    "wms_srs" "EPSG: 4326"
  END
END

END # Map File
```

deegree WMS

Quelle: http://www.deegree.org/docs/wms/deegree_wms_configuration_2006-07-31.html (25.06.2007)

Das Open Source Java-Framework deegree stellt verschiedene Komponenten für Geodateninfrastrukturen bereit. Seine Web Service Architektur verwendet konsequent die Standards des Open Geospatial Consortiums (OGC) und des ISO/TC 211 (ISO Technical Committee 211). Der deegree WMS ist die offizielle Referenzimplementierung des WMS-Standards in der Version 1.1.1. Das deegree-Framework umfasst Transactional Web Feature Services (WFS) 1.1.0, Web Coverage Services (WCS) 1.0.0 und Catalogue Services (CSW) 2.0.0. CSW unterstützt dabei auch das Applikationsprofil von ISO19115/ISO19119.

Die Konfiguration von deegree WMS erstreckt sich über drei verschiedene Ebenen bzw. XML-Dateien. Im Gegensatz zum UMN MapServer ist die Konfiguration daher vergleichsweise komplex. Die Informationen für den physischen Zugriff sowie die entsprechende Datenstruktur sind für jeden Datensatz in einem separaten XML-Schema beschrieben. Anschliessend müssen die hier definierten Daten als Layer in einer XML-Konfigurationsdatei eingebunden werden. Die graphische Darstellung der verschiedenen Layer wird schliesslich pro Layer in einer separaten XML-Datei definiert. Der Verweis auf diese Dateien ist in der Konfigurationsdatei enthalten. Wie der UMN MapServer unterstützt auch deegree als Datenquellen sowohl lokale Dateien wie auch Datenbanktabellen.

Die Konfiguration eines deegree WMS wurde im Rahmen der Vorabklärungen auf der Basis eines Demobeispiels der lat/lon GmbH nachvollzogen. Das komplette Beispiel ist in der oben aufgeführten Quelle ausführlich dokumentiert und wird daher an dieser Stelle nicht aufgeführt.