Universität Paderborn

Diplomarbeit

Interaktive Karten auf Basis Geographischer Informationssysteme

Konzeption und Implementierung eines Web Map Services unter Verwendung von Open-Source-Komponenten und SVG

Prof. Dr. Ludwig Nastansky

Betreuer

Dipl.-Inform. Stefan Smolnik

vorgelegt von

Jan Woltering

Matrikelnummer 6120675

Studiengang Wirtschaftsinformatik

Inhaltsverzeichnis

Inhaltsverzeichnis II
Abkürzungsverzeichnis VI
Abbildungsverzeichnis IX
Tabellenverzeichnis IX
Angaben zur Notation IX
1 Einleitung 1
1.1 Szenario 1
1.2 Zielsetzung 2
1.3 Aufbau der Arbeit 2
2 Grundlagen 4
2.1 Das OpenGIS Reference Model 4
2.1.1 Das OpenGIS Consortium 4
2.1.2 Begriffe und Definitionen 4
2.1.2.1 Geographic Features 4
2.1.2.2 Geometric Object 5
2.1.2.3 Topology 6
2.1.2.4 Coordinate Reference Systems 6
2.1.2.5 Portrayal Model 7
2.1.3 Service Framework 9
2.1.3.1 Überblick 9
2.1.3.2 Services 10
2.1.3.3 Encoding 11
2.1.4 Datenformate 12
2.1.4.1 Simple Feature 12
2.1.4.2 Well-Known Binary / Text Representation for Geometry 13
2.1.4.3 Geography Markup Language 14
2.2 SVG und Kartographie 15
2.2.1 Transformation der kartesischen Systeme 15
2.2.2 Transformation der geometrischen Formen 15
3 Konzepte und Anforderungen 17
3.1 Skizzierung des Gesamtsystems 17
3.1.1 Einordnung in das OpenGIS Reference Model 17
3.1.2 Web Map Service 18

3.1.2.1 GetCapabilities 19
3.1.2.2 GetMap 21
3.1.2.3 GetFeatureInfo 23
3.1.3 Datenbasis 23
3.1.3.1 Diskrete Daten 23
3.1.3.1.1 Simple Feature for SQL 24
3.1.3.1.2 Web Feature Service 25
3.1.3.2 Kontinuierliche Daten 26
3.1.3.2.1 Raumbezogene Rasterbilder 27
3.1.3.2.2 Kaskadierender Web Map Service 27
3.1.4 Thin -, medium - und thick clients 28
3.2 Anforderungen an den Server 30
3.2.1 Modularisierung und Erweiterbarkeit 30
3.2.2 Performance und Skalierbarkeit 31
3.2.2.1 „on the fly“-Generierung und Echtzeit-Karten 31
3.2.2.2 Multi-Tier-Architektur 32
3.2.2.3 Multithreading 33
3.2.2.4 Caching und Pooling 34
3.2.2.4.1 Ressourcen-Pools 34
3.2.2.4.2 Content-Caching 35
3.2.2.5 Komprimierung und Generalisierung 37
3.2.2.5.1 Komprimierung der Koordinaten 38
3.2.2.5.2 Generalisierung 39
3.2.2.5.3 Format-Kompression 40
3.2.3 Sicherheit 42
3.3 Anforderungen an den Client 44
3.3.1 Paradigmen der visuellen Datenexploration 45
3.3.2 Legende und Bildschirmaufbau 46
3.3.3 Adaptives Zoomen und dynamisches Nachladen 47
4 Systementwurf und Implementierung 49
4.1 Die Konzeption im Überblick 49
4.2 Der Server – die Schnittstelle zum GIS 52
4.2.1 Der Servlet-Container - Tomcat 52
4.2.1.1 Sicherheit 52
4.2.1.2 Performance und Multithreading 53

4.2.1.3 Konnektivität 54
4.2.2 Der Apache Web-Server 56
4.2.3 Das SAX-Konzept 56
4.2.4 Das WMS-Servlet 59
4.2.4.1 GetMap-Operation 59
4.2.4.1.1 Die Serialisierungskomponenten 59
4.2.4.1.2 Die Layerkomponenten 61
4.2.4.1.3 Die Struktur und der Aufbau der Karte 63
4.2.4.1.4 Einbinden von externen Signaturen 64
4.2.4.2 Die GetCapabilities-Operation 66
4.2.4.3 Ausnahmebehandlung 67
4.2.4.4 Caching 68
4.2.5 Die Serialisierung 70
4.2.6 Der Layer – die Schnittstelle zur Datenbasis 71
4.2.6.1 Von der Datasource zum LayerControler 72
4.2.6.2 Abstract-Layer 74
4.2.6.3 Simple Feature-Layer 75
4.2.6.4 Label-Layer 78
4.2.6.5 Tiled Raster-Layer 78
4.2.6.6 External Raster-Layer 79
4.2.6.7 WMS-Layer 79
4.2.6.8 WFS-Layer 80
4.3 Der Client – die Schnittstelle zum Benutzer 82
4.3.1 Der Client aus Endanwendersicht 82
4.3.1.1 Graphischer Aufbau und Symbolik 82
4.3.1.2 Funktionalitäten 84
4.3.2 Der Client aus Sicht des Kartenautors 85
4.3.2.1 Aufbau der eXmap 85
4.3.2.2 Interaktivität 88
4.3.3 Implementierung der Kernfunktionalitäten 88
4.3.3.1 Zoom & Pan und die History 88
4.3.3.2 Dynamisches Nachladen 90
4.3.3.3 Kompression 91
4.3.3.4 Die generische Legende 92
5 Ausblick 93

5.1 Implementierung weiterer Layer-Arten 93
5.2 Erweiterung des Gestaltungsspielraums des Benutzers 94
5.3 Implementierung weiterer Serialisierungskomponenten 94
5.4 Entwicklung weiterer Clients 95
5.5 Praktischer Einsatz 96
6 Fazit 98
7 Literaturverzeichnis 100
Eidesstattliche Erklärung 107
Anhang A – Hinweise zur Begleit-CD 108

Abkürzungsverzeichnis

AJP	Apache Jserv Protocol
API	Application Programming Interface
ASP	Active Server Pages
AWT	Abstract Window Toolkits
BAS	Borland Application Server
CAS	Coverage Access Services
CAT	Catalog Interface
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
CTS	Coordinate Transformation Services
DBMS	Database Management Systems
COM	Component Object Model
DOM	Document Object Model
DPI	Dots Per Inch
DTD	Document Type Definition
ECMA	European Computer Manufacturers Association
EPSG	European Petroleum Survey Group
FAS	Feature Access Services
Gaz	Gazetteer Service Profile of a WFS
GeoC	Geocoder
GIF	Graphics Interchange Format
GIS	Geographisches Informationssystem
GML	Geography Markup Language
GNU	GNU's Not UNIX
GPRS	General Packet Radio Service
GUI	Graphical User Interface
gzip	GNU zip
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
IIS	Internet Information Service
IP	Internet Protocol

ISO	International Standards Organization
ISO/OSI	ISO / Open Systems Interconnection
J2EE	Java 2, Enterprise Edition
J2ME	Java 2, Micro Edition
J2SE	Java 2, Standard Edition
JDBC	Java Database Connectivity
JNDI	Java Naming and Directory Interfaces
JNI	Java Native Interface
JPEG	Joint Photographic Experts Group
JSP	Java Server Pages
JVM	Java Virtual Machine
LBS	Location-based Services
LDAP	Lightweight Directory Access Protocol
MIME	Multipurpose Internet Mail Extension
OLE	Object Linking and Embedding
ORM	OGC Reference Model
PDA	Personal Digital Assistant
PDF	Portable Document Format
GeoP	Geoparser
PHP	PHP: Hypertext Preprocessor
PNG	Portable Network Graphics
RSA	Rivest - Shamir - Adleman
SAX	Simple API for XML
SFS	Simple Feature for SQL
SLD WMS	Styled Layer Descriptors Web Map Service
SQL	Structured Query Language
SRS	Spatial Reference System
SSL	Secure Socket Layer
STX	Streaming Transformations for XML
SVG	Scalable Vector Graphics
SVGT	SVG Tiny
TCP	Transmission Control Protocol
Tiff	Tagged Image File Format
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

UTM	Universales Transversales Mercator
W3C	World Wide Web Consortiums
WCS	Web Coverage Service
WCTS	Web Coordinate Transformation Service
WebCGM	Web Computer Graphics Metafile
WFS	Web Feature Service
WKBGeometry	Well-known Binary Representation for Geometry
WKTGeometry	Well-known Text Representation of Geometry
WMS	Web Map Service
XML	Extensible Markup Language
XPath	XML Path Language specification
XSLT	Extensible Stylesheet Language Transformation

Abbildungsverzeichnis

Abbildung 1 - Portrayal Model 8
Abbildung 2 - The OpenGIS Web Service Framework (OSF) 9
Abbildung 3 - Services Interoperability Stack 11
Abbildung 4 - Geometry Class Hierarchy 13
Abbildung 5 - Thin vs. thick clients for portraying features over the Internet 29
Abbildung 6 - Aufbau des Systems 51
Abbildung 7 - Interfacediagramm der Serialisierungskomponenten 60
Abbildung 8 - Interfacediagramm der Layerkomponenten 62
Abbildung 9 - Sequenzdiagramm der GetMap-Operation 64
Abbildung 10 - Screenshot des Clients 83

Tabellenverzeichnis

Tabelle 1 - Parameter der GetCapabilities-Operation 20
Tabelle 2 - Parameter der GetMap-Operation 22

Angaben zur Notation

Sämtliche *Schlüsselwörter* werden in dieser Diplomarbeit bei erstmaliger Nennung kursiv hervorgehoben. Wörter mit direktem Bezug auf den Quellcode, vornehmlich Klassennamen und Methoden sowie XML-Elemente und -Attribute bzw. deren Werte, werden in einer separaten, serifenlosen `Schriftart` dargestellt. Klassennamen werden bei der ersten Anführung inklusive des Package-Namens angegeben.

Zur Steigerung des Leseflusses werden alle Anglizismen, mit Ausnahme der erstmaligen Nennung der Schlüsselwörter, gemäß den Regeln der deutschen Sprache dekliniert. Bei Akronymen wird auf die Deklinationsendung aufgrund der inhärenten Eigenschaften einer Abkürzung verzichtet. Vom Aspekt der Deklination abgesehen werden die Schlüsselwörter in ihrer ursprünglichen Notation angeführt.

1 Einleitung

1.1 Szenario

> *„… The hard part of taking advantage of this flood of geo-spatial information will be making sense of it – tuning raw data into understandable information…“*[1]

Karten sind das Resultat dieses Umwandlungsprozesses. Sie haben die Aufgabe, Informationen jeglicher Art in eine Form zu transformieren, die eine verständliche, visuelle Kommunikation ermöglicht.[2] Sie nehmen innerhalb der visuellen Kommunikation eine herausragende Stellung ein. Schätzungsweise wurden bereits im Jahr 2001 40 Mio. Karten täglich von Nutzern des Internets abgerufen[3].

Wenn man bedenkt, dass ca. 80% aller Geschäfts- und Verwaltungsdaten geographischen Bezug haben[4], wird deutlich, wie notwendig interoperable Systeme zur Distribution und Visualisierung von geographischen Informationen sind. Dabei stehen nicht allein die Karten in Form von Bildern oder Grafiken im Vordergrund, sondern auch die „rohen“ Daten, aus denen mittels geeigneter Analysetechniken verständliche Informationen extrahiert werden.

Die Definition der erforderlichen Schnittstellen, die die Basis der interoperablen, verteilten Systeme bilden, hat sich das *OpenGIS Consortium (OGC)* zur Aufgabe gemacht. Diese Schnittstellen sollen es ermöglichen, dass jeder über alle netzwerk-, applikations- oder plattformbasierten Grenzen hinweg die Vorteile geographischer Informationen und der angebotenen Dienste nutzen kann.

Neben der Definition der Schnittstellen für den Zugriff auf Informationsanbieter und der Standardisierung der für die Informationsübermittlung erforderlichen Datenbeschreibungen verfolgt das OpenGIS Consortium das Ziel, ein Metadatenmodell zu entwickeln, mit dessen Hilfe es möglich ist, sich in der Informationsflut zurechtzufinden. So sollen Katalogsysteme aufgebaut werden, die das Auffinden der relevanten Informationen erleichtern.

---

[1] Al Gore, ehem. Vize-Präsident USA, 1998 Zitiert aus Dickmann (2001) S. 11

[2] vgl. Dässler (2002)

[3] Dickmann (2001) S. 6

[4] OpenGIS Consortium (o. J.)

1.2 Zielsetzung

Die vorliegende Arbeit beschäftigt sich mit dem dynamischen Transformationsprozess von geographischen Daten in geographische Karten. Dieser Transformationsprozess besteht aus einer Kette von Einzeltransformationen; so werden die Daten und geographischen Objekte selektiert, in graphische Objekte umgewandelt, gerendert und anschließend dargestellt[5]. Ziel ist es, ein verteiltes System zu entwickeln, bei dem die einzelnen Transformationsschritte flexibel auf die beteiligten Komponenten verteilt werden können, um ein Höchstmaß an Interaktivität und Dynamik in Abhängigkeit der technischen Umgebung des Systems zu ermöglichen.

Die Serverkomponente dieses Systems soll der Spezifikation eines OpenGIS-konformen Web Map Services genügen und als Schnittstelle zu verschiedenen Datenbasen dienen. Diese Datenbasen stellen die geographischen Informationssysteme dar. Neben Karten im Rasterdatenformat soll der Server Karten als Vektorgraphiken im *Scalable Vector Graphics (SVG)* Format übermitteln können, um ein breites Spektrum an Verwendungsmöglichkeiten dieses Servers zu gewährleisten.

Darüber hinaus soll ein Client konzipiert und implementiert werden, der die Anforderungen der visuellen Datenexploration erfüllt und die Grundlage für adaptives Zoomen zur Verfügung stellt. Der Client selbst soll auf SVG basieren und der Evaluierung dieses relativ neuen Standards bezüglich Interaktivität und Dynamik dienen.

1.3 Aufbau der Arbeit

Im anschließenden Kapitel soll zunächst das *Reference Model* des OpenGIS Consortiums vorgestellt werden, dessen Elemente und Konzeptionen die Grundlage für das entwickelte verteilte System bilden. Abgeschlossen wird das Kapitel mit der Analyse des kartographischen Potentials von SVG. Besondere Berücksichtigung findet dabei die Transformation der geographischen Objekte und die Umwandlung der Koordinatensysteme. Diese Analyse bildet die Voraussetzung für die Verwendung von SVG als Mittel zur Darstellung von geographischen Karten.

Das dritte Kapitel beginnt mit der Skizzierung des Gesamtsystems. Diese umfasst neben der Spezifikation des *Web Map Services (WMS)* die Darstellung der möglichen Datenbasen und eine Klassifizierung der denkbaren Clients des Systems. Folgend

[5] vgl. OpenGIS Consortium (2003) a

werden die sich daraus ergebenden Anforderungen an den Server beschrieben. Darüber hinaus werden in diesem Teil des Kapitels allgemeine Realisierungsanforderungen an den Server wie Erweiterbarkeit und Performanz erläutert. Das Kapitel endet mit der Ausarbeitung der Anforderungen an den Client. Diese gliedert sich hauptsächlich in eine Zusammenfassung der Paradigmen der visuellen Datenexploration und eine Darstellung des Konzepts des adaptiven Zoomens und dynamischen Nachladens auf.

Den umfangreichsten Teil der Diplomarbeit bildet das vierte Kapitel. Es vermittelt einen detaillierten Einblick der Implementierung der im dritten Kapitel vor- und aufgestellten Konzepte und Anforderungen, sowohl des Servers als auch des Clients. Eingeleitet wird das Kapitel mit einer Übersicht der Gesamtarchitektur des Systems. Diese dient als Rahmen für die sich anschließenden Betrachtungen der im Einzelnen verwendeten Konzepte, Techniken und Module.

Die auf Schnittstellen aufbauende Architektur des Systems bietet zahlreiche Möglichkeiten der Erweiterung und Verbesserung. Ein Ausblick darauf wird in Kapitel fünf vorgestellt.

Das Fazit bildet den Abschluss der vorliegenden Diplomarbeit.

2 Grundlagen

2.1 Das OpenGIS Reference Model

2.1.1 Das OpenGIS Consortium

Das OpenGIS Consortium wurde im Jahr 1994 gegründet und stellt eine Vereinigung aller relevanten Organisationen wie GIS- und Datenbankherstellern, Datenprovidern, Dienstleistern, Anwendern und Universitäten dar. Die vom OGC entwickelten Spezifikationen haben das Ziel, die Interoperabilität aller raumbezogenen Komponenten und Dienste zu steigern oder erst zu ermöglichen. Diese Spezifikationen bauen selbst wieder auf bestehenden IT-Spezifikationen wie denen des *World Wide Web Consortiums (W3C)* und der *International Standards Organisation (ISO)* auf.

Analog zu den einzelnen Spezifikationen, die zum Teil schon verabschiedet sind, wurde vom OGC ein *Reference Model (ORM)*[6] veröffentlicht, das die grundlegenden Konzepte und das Zusammenspiel der einzelnen Spezifikationen verdeutlicht. Es bildet ein Framework für die zu implementierenden, interoperablen Komponenten zur Auffindung, Distribution, Transformation und Visualisierung von geographischen Informationen.

2.1.2 Begriffe und Definitionen

2.1.2.1 Geographic Features

Am Anfang jeglicher Modellierung geographischer Information steht das *feature*.

> *„A feature is an abstraction of a real world phenomenon. A geographic feature is a feature associated with a location relative to the Earth. A digital representation of the real world can be thought of as a set of features.“*[7]

Ein *geographic feature* umfasst zwei Betrachtungsebenen; zum einen die konkrete Ausprägung eines solchen, die *feature instance*, und zum anderen den *feature type*, der durch Klassifikation verschiedener Instanzen nach gemeinsamen Charakteristika gebildet wird. Nach dem *General Feature Model* haben feature types spezifische Eigenschaften; diese sind Attribute, Operationen und Beziehungen. Attribute können

---

[6] OpenGIS Consortium (2003) a

[7] vgl. OpenGIS Consortium (2003) a S. 7

dabei räumlichen Bezug aufweisen. Bei den räumlichen Attributen eines features kann es sich sowohl um kontinuierliche als auch um diskrete Daten handeln. Diskrete Daten werden meist als Vektordaten, kontinuierliche als Rasterdaten bezeichnet.

In der deutschsprachigen Literatur wird anstelle von features von *räumlichen Bezugseinheiten*, *räumlichen Objekten*, oder kürzer von *Geoobjekten* gesprochen. Diese werden von de Lang wie folgt definiert:

> *„Geoobjekte sind räumliche Elemente, die zusätzlich zu Sachinformationen geometrische und topologische Eigenschaften besitzen und zeitlichen Veränderungen unterliegen können. Kennzeichnend für Geoobjekte sind somit Geometrie, Topologie, Thematik und Dynamik.“*[8]

Mit Hilfe der objektorientierten Modellierung lassen sich über Geoobjekte *Objektklassen* bilden, die eine Analogie zu den vom OGC eingeführten feature types darstellen.

Um den Zugriff auf geographische Informationen, die features oder Geoobjekte, zu erleichtern, spezifiziert das OGC konzeptuelle Schemata[9], die abstrakte feature types[10] definieren. Auf deren Basis können spezielle, anwendungsbezogene Schemata entwickelt werden. Dementsprechend wird die anwendungsübergreifende Informationsverarbeitung gewährleistet.

2.1.2.2 Geometric Object

Die Geometrie eines features wird durch seine räumlichen Attribute beschrieben. Diese Attribute werden *geometric objects* genannt und umfassen die Lagekoordinaten basierend auf einem eindeutigen räumlichen Bezugssystem (engl. *coordinate system*). Dabei werden metrische Bezugskoordinatensysteme zugrunde gelegt, die eine quantifizierbare und objektivierbare Standortbestimmung zulassen.[11] Wird ein feature von einem Bezugssystem in ein anderes transformiert, so ändern sich ausschließlich die räumlichen Attribute.

---

[8] de Lang (2002) S 157

[9] OpenGIS Consortium (1999) a

[10] OpenGIS Consortium (2001) a

[11] vgl. de Lang (2002) S 158

2.1.2.3 Topology

Die *Topologie* (engl. *topology*) eines features ist bei einer Transformation des Bezugssystems eine Invariante. Sie kennzeichnet die räumliche Beziehung von features untereinander. Abstrahiert von der Geometrie stellt die Topologie die relative Lage eines features dar und spiegelt die Nachbarschafts-, Umgebungs-, Teilmengen- und Überlagerungsbeziehungen eines features zu anderen wider. Topologische Informationen können aus den darunter liegenden Geometrien gewonnen werden und bilden einen n-dimensionalen Graphen, der bei wiederkehrenden, rechenintensiven Beziehungsanalysen einen großen Geschwindigkeitsvorteil bietet. So basieren zum Beispiel Algorithmen zur Bestimmung der kürzesten Wege auf entsprechenden Graphen, die das Straßennetz symbolisieren, und nicht auf den konkreten Geometrien der einzelnen Straßen.

2.1.2.4 Coordinate Reference Systems

Die Lagekoordinaten eines geometric objects sind ausschließlich eindeutig, wenn sie auf einem vollständig definierten räumlichen Bezugssystem (Koordinatensystem) basieren. Bezugssysteme, die auf die Erde referenzieren, werden *geodätische Bezugssysteme* oder in der Sprache des OGC *geographic coordinate reference systems* oder *spatial reference systems (SRS)* genannt und bestehen aus einem Koordinatensystem und einem *geodätischen Datum*. Aus der Erdbezogenheit ergibt sich eine prinzipielle Dreidimensionalität der Koordinatensysteme.

Im Allgemeinen sind Koordinatensysteme als *metrische Räume* definiert, die aus einer Menge M und einer Metrik bestehen. Dabei ist die Metrik (Abstand) eine Abbildung zweier Elemente der Menge M auf die Menge der reellen Zahlen, die die Bedingungen der Definitheit, der Symmetrie und der Dreiecksungleichung erfüllt.

Das geodätische Datum besteht aus mehreren Parametern, die das Koordinatensystem hinsichtlich Ursprung, Orientierung und Skalierung definieren, so dass jedem Punkt der Erde eine eindeutige Koordinate zugeordnet werden kann.

Im Wesentlichen werden zwei Arten von Koordinatensystemen verwendet. In rechnergestützten Informationssystemen bilden kartesische Koordinatensysteme die Grundlage der Darstellung und Verarbeitung von Geometrien, sowohl im Vektor- als auch im Rasterformat. Im Gegensatz dazu verwendet die klassische Kartographie vornehmlich Polarkoordinaten, die einen Punkt des Koordinatensystems durch den

Abstand vom Ursprung und den Winkeln bezüglich der Polarachse bzw. des Äquators darstellen.

Punkte der Erde werden häufig in Polarkoordinaten angegeben. Dazu wird die Erde vereinfacht als Kugel betrachtet, deren Mittelpunkt und Radius die Parameter des geodätischen Datums bestimmen. Der Winkel bezüglich des Äquators wird dabei als *Geographische Breite* (φ) bezeichnet, der Polarachsenwinkel als *Geographische Länge* (λ). Da die Erde ausschließlich stark vereinfacht als Kugel zu betrachten ist und auch nur näherungsweise einem Ellipsoid gleicht, ergeben sich verschiedene regionale geodätische Daten, die die Abflachung an den Polen und die Ausbuchtung am Äquator, sowie kleinere, regionale Dellen und Ausbuchtungen berücksichtigen.

Eine integrierte Verarbeitung der geographischen Daten erfordert die Transformation oder Konversion der Koordinaten von einem Bezugssystem in ein anderes. Dabei wird zwischen Transformationen, die eine Änderung des geodätischen Datums (Transformation) einschließen, und solchen ohne Datumswechsel (Konversion) unterschieden. Die bekannteste Form der Konversion ist die Projektion (engl. *map projection*). Diese Umwandlung bildet eine „Kugeloberfläche“ eines polaren Koordinatensystems auf ein zweidimensionales (kartesisches) Koordinatensystem ab.

2.1.2.5 Portrayal Model

Projizierte Koordinaten bilden wiederum die Grundlage für das *Darstellungsmodell* (engl. *Portrayal Model*) des OGCs:

> *„**Portrayal** is the presentation of information to humans, e.g., a map. A **map** is a two-dimensional visual portrayal of geospatial data; a map is not the data itself.“*

Karten sind eine der bedeutsamsten Methoden für visuelle Interaktion und Kommunikation mit und von geographischen Informationen. Allerdings erfordern viele Aspekte in diesem Kommunikationsprozess weitere Entwicklungen. Nach Ansicht des OGC sollten die folgenden Herausforderungen von zukünftigen Entwicklungen in Angriff genommen werden:

- Nutzung von geographischen Informationen in all ihrem Umfang und ihrer Komplexität; dreidimensionale Darstellungen und vertiefende, wissenschaftliche Visualisierungen sollen dabei nicht ausgeschlossen sein

- universeller Zugriff und uneingeschränkte Nutzung von geographischen Informationen für jedermann an jedem Ort
- teamorientiertes Arbeiten mit geographischen Informationen

Eine Basis zur Bewerkstelligung dieser Aufgaben bildet das Portrayal Model. Es fußt auf einer stufenweisen Einteilung der für die Visualisierung geographischer Daten erforderlichen Transformationsprozesse.

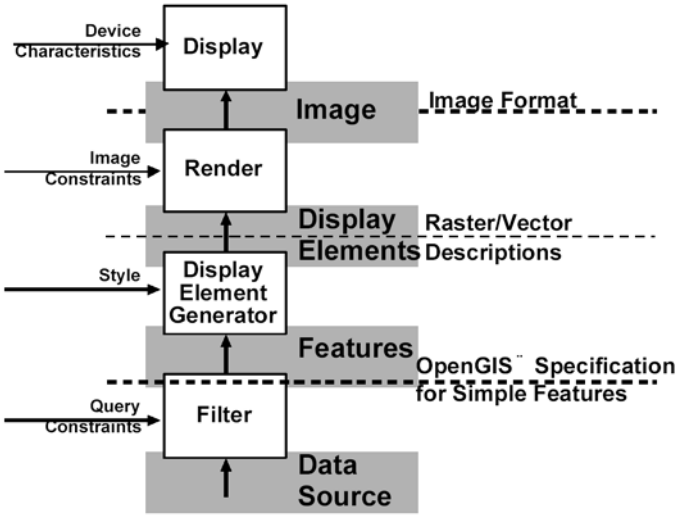

Abbildung 1 - Portrayal Model[12]

Kennzeichnend für jeden Transformationsprozess ist ein möglichst standardisiertes Ausgabeformat kombiniert mit standardisierten Transformationsparametern. So sollen aus einer entsprechenden Datenbasis mittels spezifizierter Anfragesprachen OpenGIS-konforme *Simple Features* selektiert werden. Bei den Simple Features handelt es sich um Objekte, die von abstrakten feature types des konzeptuellen Schemas des OGC erben. Diese features werden unter Verwendung der übergebenen Signaturen in graphische Elemente transformiert. Ein möglicher Standard zur Beschreibung dieser graphischen Elemente ist SVG, allerdings sind auch systeminterne Darstellungsmöglichkeiten wie die graphischen Objekte des Abstract Window Toolkits (AWT) denkbar, je nach Granulierung und Verteilung der Transformations-komponenten. Die graphischen Elemente werden anschließend von einer Renderkomponente unter Berücksichtigung der geforderten Auflösung, Farbtiefe und sonstiger bildbeeinflussender Parameter in ein Rasterformat umgewandelt. Die abschließende Darstellung des Rasters erfolgt gemäß den Charakteristika des vom User verwendeten Gerätes.

[12] OpenGIS Consortium (2003) a S. 23

Das Portrayal Model bildet die Grundlage für Systeme, bei denen die einzelnen Transformationsschritte flexibel auf die beteiligten Komponenten verteilt werden können, um ein Höchstmaß an Interaktivität und Dynamik in Abhängigkeit der technischen Umgebung der Systeme zu ermöglichen.

2.1.3 **Service Framework**

2.1.3.1 Überblick

Die Visualisierung von geographischen Daten ist allerdings nur eine Komponente innerhalb der komplexen Architektur des OGC zur geographischen Informationsverarbeitung. Die Visualisierungskomponente wird eingebettet in ein Netzwerk von *Services*, die spezifische *Operations* mittels *Interface* anbieten[13].

Diese Architektur ist ausgerichtet an dem *publish/find/bind* Modell[14]. Nach diesem Modell sind die Rollen des *Service Providers*, des *Service Requestor*s und des *Service Broker*s unverzichtbar. Durch den Vermittler ist es dem Nachfrager möglich, die dort veröffentlichten Dienste des Anbieters aufzufinden und an sich zu binden.

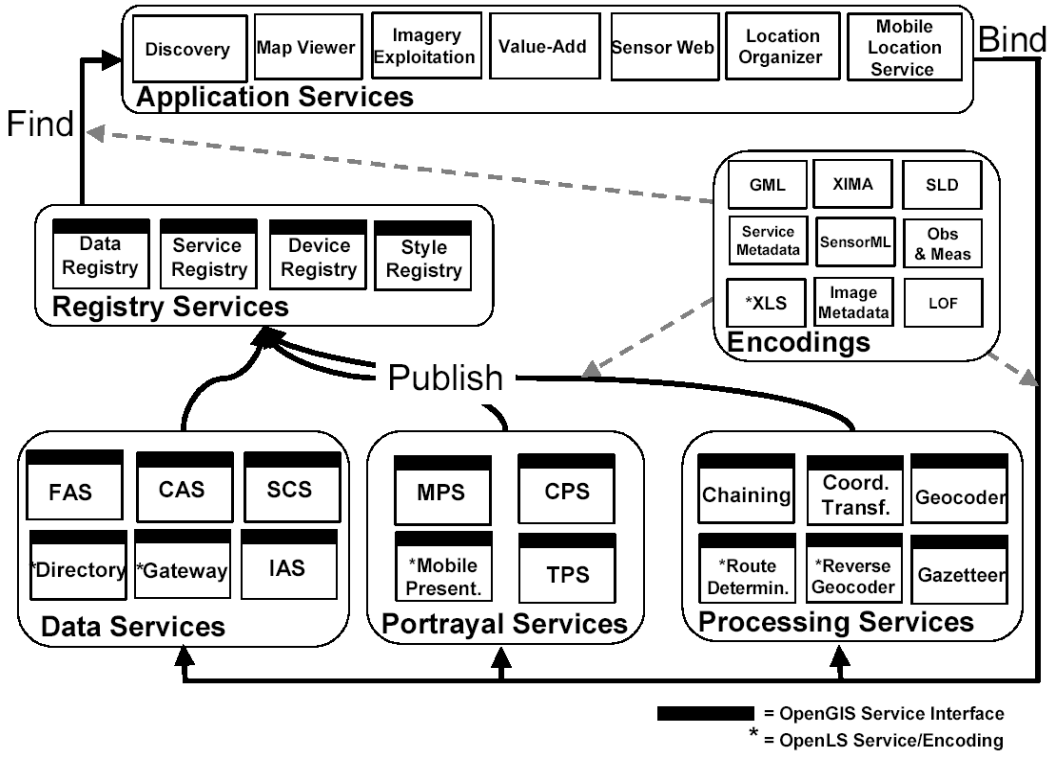

Abbildung 2 - The OpenGIS Web Service Framework (OSF)[15]

[13] OpenGIS Consortium (2003) a S. 28

[14] vgl. OpenGIS Consortium (2003) a S. 30

[15] OpenGIS Consortium (2003) a S. 34

2.1.3.2 Services

Das OpenGIS Consortium unterscheidet zwischen fünf verschiedenen Service-Arten. Im Folgenden sollen diese inklusive der momentan bereits bestehenden Schnittstellenspezifikationen kurz vorgestellt werden:

- Die *Data Services* sind die Grundlage für jegliche Verarbeitung von geographischen Informationen. Sie stellen neben dem Zugriff auf die Daten selbst auch geometrische und sachdatenbezogene Filteroperationen zur Verfügung. Je nach Ausprägung der Daten unterscheidet das OGC hauptsächlich zwischen *Feature Access Services (FAS)*, die diskrete Simple Features in einer Reihe von unterschiedlichen Kodierungen anbieten können, und *Coverage Access Services (CAS)*, die den Zugriff auf zum Teil mehrdimensionale Rasterdaten, wie multispektrale Fernerkundungsdaten, ermöglichen. Das OpenGIS Consortium definierte bislang neben den Spezifikationen von Simple Feature for SQL, -CORBA, -OLE/DCOM die Schnittstellen für einen *Web Feature Service (WFS)*, der lesenden und schreibenden Zugriff auf die im *Geography Markup Language (GML)* Format kodierten (Vektor-)Daten erlaubt, und einen *Web Coverage Service (WCS)*, der webbasierten Zugang zu den sich noch zum Teil in „Rohform" befindlichen Rasterdaten gestattet.
- Die *Processing Services* operieren auf geographischen Daten und können zur Weiterverarbeitung dieser dienen. Sie werden meist in eine Kette von Verarbeitungsschritten integriert und übernehmen dabei zum Beispiel Kodierungs- oder Transformationsaufgaben. Gegenwärtig ist das Interface für *Coordinate Transformation Services (CTS)* erschienen. Als so genannte „Discussion Papers" stehen beim OGC der *Web Coordinate Transformation Service (WCTS)*, der *Geocoder (GeoC)*, der *Geoparser (GeoP)* und das *Gazetteer Service Profile of a WFS (Gaz)* als weitere mögliche Processing Services zur Disposition.
- Die *Portrayal Services* verfolgen das bereits vorgestellte Darstellungsmodell des OGC. Die vom OGC veröffentlichte Spezifikation des *Web Map Services (WMS)* bildet die Grundlage für diese Diplomarbeit und soll im dritten Kapital detailliert vorgestellt werden.
- Die *Registry Services* bieten einen einheitlichen Mechanismus zur Klassifizierung, Registrierung und Beschreibung von Daten und Services

innerhalb eines Netzwerkes. Für den Benutzer sind diese Registry Services ein effektives Hilfsmittel zur Auffindung der von ihm benötigten Informationen und Dienste. Spezifiziert hat das OGC einen *Catalog Interface (CAT)* für den Zugriff und die Verwaltung der Metadaten der registrierten Daten und Dienste.

- Die *Application Services* sind für den Benutzer die Schnittstelle für den Zugriff auf die vorangehenden Dienste. Inwieweit die verschiedenen Dienste genutzt werden richtet sich nach der Struktur des Gesamtsystems und den spezifischen Anforderungen und Möglichkeiten der benutzten Anwendung. Das Spektrum reicht von rudimentären *Location-based Services (LBS)*, die auf Smartphones laufen, bis hin zu anspruchvollsten GIS-Applikationen auf leistungsstarken Workstations. Insofern werden Application Services vom OGC nicht weiter spezifiziert.

2.1.3.3 Encoding

Um die vorgestellten Services zu einem mächtigen multi-tier System zu vernetzen, was dem Benutzer sowohl einen flexiblen Zugriff auf die einzeln Komponenten, als auch eine wertsteigernde Verkettung mehrerer Dienste gestattet, sind Standards zur Kodierung der Kommunikation zwingend erforderlich.

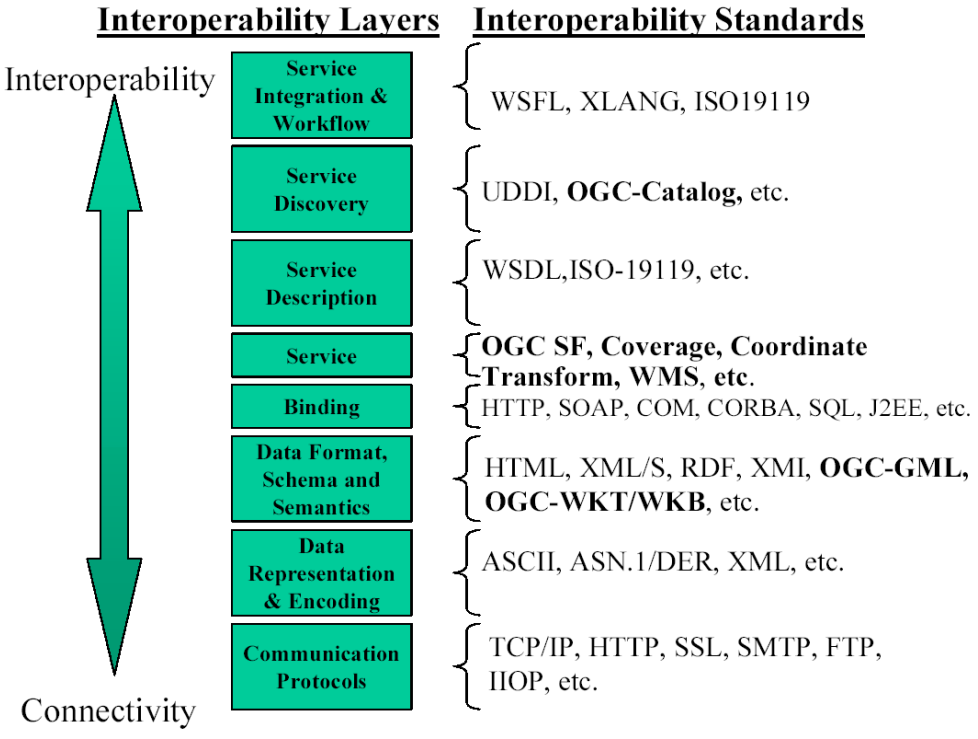

Abbildung 3 - Services Interoperability Stack[16]

[16] OpenGIS Consortium (2003) a S. 56

Abbildung 3 verdeutlicht, inwieweit das OGC bei der Konzeption des Service-Frameworks bestehende Standards benutzt, auf bestehenden Standards aufbaut und an welchen Stellen **eigene** Standards definiert werden.

2.1.4 **Datenformate**

2.1.4.1 Simple Feature

Die elektronische Verarbeitung von features verlangt vor allem spezielle Kodierungen zur Repräsentation der geometrischen Objekte. Zu diesem Zweck hat das OGC auf Basis des konzeptuellen Schemas für features so genannte *Simple Features* definiert, deren Geometrien prinzipiell zweidimensional sind und über ausschließlich linear-interpolierte Kanten verfügen. Für die nichträumlichen Attribute der features werden dabei einfache, native Datentypen verwendetet.

Das Geometrie-Objektmodell, dargestellt in Abbildung 4, basiert auf einer abstrakten Geometry-Klasse, die für alle Unterklassen den Bezug zu einem coordinate reference system festlegt. Beerbt wird die Geometry-Klasse von Klassen zur Repräsentation von Punkten, Kurven und Flächen, und von Klassen zur Aggregation dieser primitiven Geometrie-Typen. Bis auf die Point-Klasse sind alle anderen Klassen wiederum abstrakte Klassen, für die die Linearität der Kanten noch nicht zwingend ist. Diese Linearität wird erst für die instanziierbaren Polygon- und LineString-Klassen festgelegt. Nach dieser Struktur besteht ein Point aus einer zweidimensionalen Koordinate, die entweder auf einem projizierten, kartesischen Koordinatensystem oder einem Polarkoordinatensystem mit festem Umfang basiert. Ein LineString besteht wiederum aus mindestens zwei Points und ein Polygon aus mindestens einem LinearRing, wobei dieser einem einfachen, geschlossenen LineString gleicht. Der erste LinearRing eines Polygons stellt die äußere Grenzen dar, die weiteren repräsentieren die inneren Ausschlüsse. Analog zu primitiven Objektklassen sind alleinig die Aggregationsklassen MultiPolygon, MultiLineString und MultiPoint instanziierbar. Eine Ausnahme bildet die GeometryCollection-Klasse, die als Containerklasse für alle anderen Klassen fungiert.

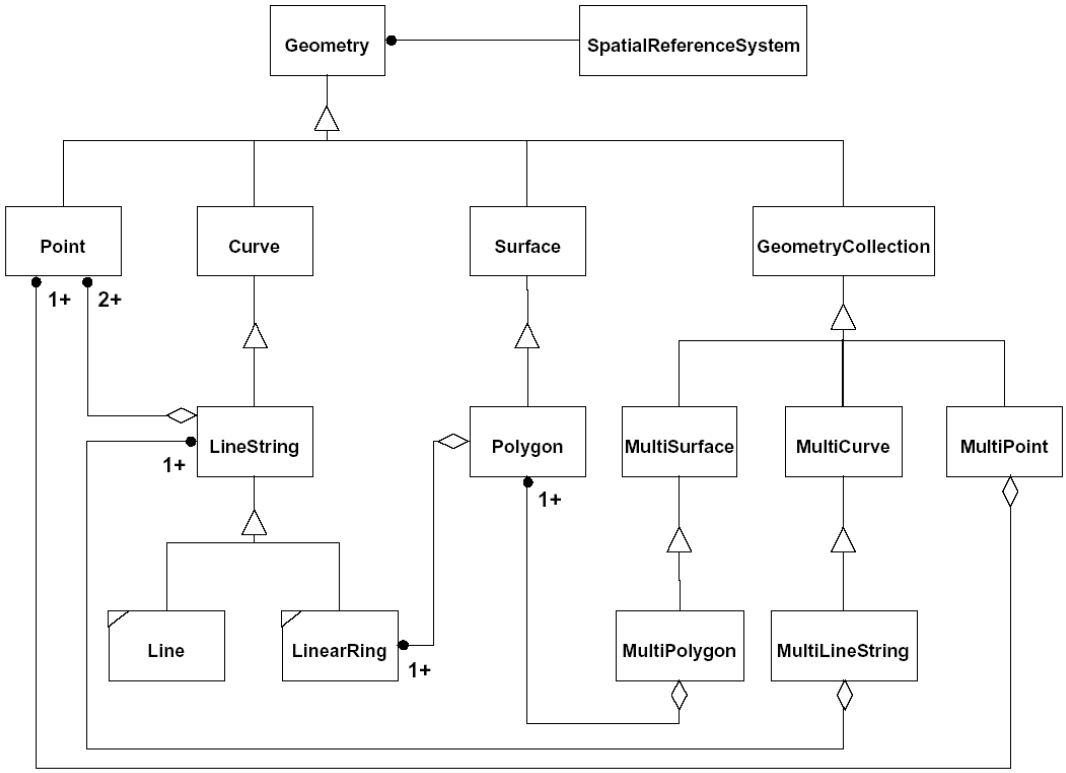

Abbildung 4 - Geometry Class Hierarchy[17]

Zur Kodierung dieser Geometrien hat das OGC drei Standards verabschiedet.

2.1.4.2 Well-Known Binary / Text Representation for Geometry

Die *Well-known Binary Representation for Geometry (WKBGeometry)*[18] bildet die Geometrien der geometric objects sequentiell auf einem Byte-Strom ab. Dieser Strom besteht aus einer Folge von 32-Bit langen natürlichen Zahlen (Unsigned Integer) und 64-Bit langen Dezimalzahlen (Double). Der Geometrie-Typ und die Anzahl der folgenden Kindelemente werden durch Integer dargestellt, eine Koordinate wird mit zwei Double-Werten abgebildet. Zur Kodierung der Zahlen sind sowohl Big Endian als auch Little Endian zugelassen; die Byte-Order selbst wird mit einem Byte dargestellt.

Neben der rein binären Kodierung hat das OGC eine alpha-numerische Repräsentation unter dem Namen *Well-known Text Representation of Geometry (WKTGeometry)*[19] veröffentlicht. Diese Repräsentation bildet ein einzelnes Geometrie-Objekt analog zur binären Darstellung auf einer einzigen Zeichenkette ab. Zur Behandlung der variablen

[17] OpenGIS Consortium (1999) a S. 2-1

[18] OpenGIS Consortium (1999) a

[19] OpenGIS Consortium (1999) a

Längen der alpha-numerischen Darstellung der Geometrie-Typen und Koordinaten werden spezielle Begrenzungszeichen verwendet.

Vornehmlich die WKBGeometry-Kodierung ist eine portable und vor allem performante Lösung und findet in den Spezifikationen für Simple Feature for SQL, -CORBA und -OLE/DCOM Anwendung. Innerhalb dieser Spezifikationen wird auch die Bindung an das geographische Bezugssystem und der Zugriff auf die Simple Features selbst geregelt.
Im weiteren Verlauf wird sich diese Diplomarbeit allerdings ausschließlich und gemäß dem bereits vorgestellten Portrayal Model mit der Verarbeitung von Simple Features und ihren Geometrien befassen.

2.1.4.3 Geography Markup Language

Die *Geography Markup Language (GML)*[20] ist eine auf XML fußende Kodierung von features, die sowohl die Darstellung der geographic objects, also der Geometrien und der zugehörigen Bezugssysteme, als auch die der nichträumlichen Attribute ermöglicht. GML besteht aus einer Reihe von Schemata, die variabel von anwendungsspezifischen XML-Schemata verwendet werden können, um vollständige XML-Dokument-Schemata zu definieren. Somit ist GML vom Prinzip her weiter gefasst als WKBGeometry, das nur zur Repräsentation von Geometrien dient. Wie bei WKBGeometry ermöglichen die Versionen 2.x von GML allerdings auch nur die Kodierung der von Simple Features verwendeten Geometrien. Die GML der dritten Generation soll die Beschränkungen aufheben und eine vollständige Implementierung der abstrakten Spezifikation des OGC geometry models bieten[21].

Um den Makel des rechenintensiven Parsens der alpha-numerischen Repräsentation der Geometriekoordinaten zu verhindern, arbeitet das OGC an einer *Binary-XML Encoding Specification*[22]. Dieses so genannte *B-XML*-Format soll die Vorzüge der hierarchischen Strukturierung und Objektorientierung von XML mit der Performanz der binären Kodierung vereinigen, um hoch komplexe feature-instances, mit all ihren, nicht mehr zwingend zweidimensionalen, linear-interpolierten Geometrien und weit verzweigten Topologien auf effiziente Art und Weise speichern, transportieren und verarbeiten zu können.

---

[20] OpenGIS Consortium (2001) b

[21] vgl. OpenGIS Consortium (2003) c

[22] OpenGIS Consortium (2003) b

2.2 SVG und Kartographie

Die Vorzüge von SVG mittels des offenen, XML-basierten, vollständig EMCA-Script-gebundenen Standards[23], interaktive Karten für das World Wide Web zu entwickeln, wurden von A. Neuman und A. M. Winter[24,25] detailliert dargelegt. Auf dieser Arbeit wird im Folgenden aufgebaut, wobei sie um die Potentiale des SVG-Standards als Beschreibungssprache der vom Portrayal Model spezifizierten Graphic Elements in einem wesentlichen Punkt erweitert werden soll. Um SVG zur Beschreibung der Graphic Elements verwenden zu können ist die Möglichkeit der Transformation der kartesischen Systeme und der Geometrien der Simple Features von fundamentaler Bedeutung.

2.2.1 Transformation der kartesischen Systeme

Das Koordinatensystem von SVG basiert auf einem zweidimensionalen, euklidischen Raum. Die Menge M dieses metrischen Raumes basiert ebenso wie die der Bezugssysteme, die Simple Features benutzen, auf der Menge der rationalen Zahlen (mit hinreichender Genauigkeit). Dieses ermöglicht quasi die direkte Verwendung der Koordinaten der Simple Features, sofern diese auf ein projiziertes Koordinatensystem Bezug nehmen. Werden dagegen unprojizierte Polarkoordinaten verwendet, können je nach Ausdehnung der darzustellenden Geometrien bzw. des Gesamtbereichs nicht mehr zu tolerierende Verzerrungen auftreten.

Der Ursprung des Koordinatensystems befindet sich, im Gegensatz zu projizierten kartesischen Bezugssystemen, bei SVG in der oberen linken Ecke des Darstellungsbereiches. Dieses erfordert eine Invertierung der Ordinate, die durch einfache Negation erreicht werden kann.

2.2.2 Transformation der geometrischen Formen

Den Geometrie-Typen der Simple Features stehen unter SVG neben den primitiven Typen wie Rechteck, Kreis, Ellipse, Linie, Polylinie und Polygon, das Pfad-Element, das Gruppierungselement sowie das `use`-Element gegenüber.

Die LineString- und Polygon-Geometrie können am effektivsten durch Pfade repräsentiert werden. Pfade bieten die Möglichkeit, alle erdenklichen Linienformen

---

[23] W3C (2003)

[24] Winter (2000)

[25] Dahinden, Neuman und Winter (2003)

(offene Pfade) oder Objektformen (geschlossene Pfade) zu erzeugen. Die Fähigkeit von Pfaden über beliebig viele Start- und, falls nötig, Endpunkte zu verfügen, stellt insbesondere eine korrekte graphische Repräsentation der inneren Ausschlüsse der Polygon-Geometrie sicher. Syntaktisch ist die Repräsentation der Koordinaten bzw. der einzelnen Formen eines Pfades der Kodierung von WKT-/WKBGeometry sehr ähnlich, was eine einfache Transformation ermöglicht.

Punkte besitzen unter SVG keine direkte Repräsentation; die Dimensionslosigkeit eines Punktes schreibt zwingend die Verwendung eines Symbols zu seiner Repräsentation vor. Symbole wiederum können unter SVG mit Hilfe des `use`-Elementes variabel referenziert werden. Das `use`-Element legt durch seine `x`- und `y`-Attribute den Zeichenort des extern definierten, referenzierten Symbols fest und dient somit der Repräsentation eines Punktes.

Das Gruppierungselement gestattet es, beliebige SVG-Elemente zusammenzufassen, und kann somit Aggregationsklassen wie GeometryCollection, MultiPoint, MultiPolygon und MultiLineString semantisch korrekt darstellen. Die MultiPolygon- und MultiLineString-Klasse können zwar graphisch korrekt durch ein einzelnes `path`-Element repräsentiert werden, allerdings können dabei, gerade beim Multi-Polygon, die semantischen Informationen über die Kindelemente verloren gehen. Insofern sollten die Polygone eines Multi-Polygons durch einzelne Pfade dargestellt und durch ein Gruppierungselement zur Repräsentation des Multi-Polygons zusammengefasst werden.

Die möglichen nichträumlichen Attribute der features können einfach als weitere XML-Attribute des durch den Transformationsprozess entstandenen SVG-Elementes angebunden werden. Weiterhin gestattet die Verwendung von *Cascading Style Sheets (CSS)* eine flexible Einbindung von Signaturen, wie Füllfarben und -mustern, Liniengestaltung und Symbolen.

3 Konzepte und Anforderungen

3.1 Skizzierung des Gesamtsystems

3.1.1 Einordnung in das OpenGIS Reference Model

Das zu entwickelnde System soll auf den vorgestellten Konzepten und Spezifikationen des OGC aufbauen. Dabei stehen die Entwicklung eines SVG- und browserbasierten Application Services, der dem User einen einfachen, interaktiven und dynamischen Zugang zu geographischen Informationen ermöglicht, und eines integrierten Web Map Services, der als Schnittstelle zu verschieden Datenbasen dient, im Vordergrund.

Die Kommunikation zwischen diesen beiden Services soll ausschließlich über das vom OGC vorgegebene Interface eines WMS erfolgen, um die Wiederverwendbarkeit der einzelnen Services und die Interoperabilität mit anderen Services zu ermöglichen. Es ist wichtig an dieser Stelle hervorzuheben, dass die Serverkomponente nur die Aufgaben eines WMS übernimmt, die aus der Transformation der Simple Features in graphische Elemente und dem gegebenenfalls nötigen Rendern dieser bestehen. Die Erzeugung von interaktiven und dynamischen Karten soll die Client-Applikation unter Verwendung dieser graphischen Elemente, die je nach Ausprägung der features um Sachdaten-Attribute ergänzt wurden, übernehmen.

Diese standardisierte und flexible Kopplung der beiden zu entwickelnden Komponenten soll es ermöglichen, auf der einen Seite einer Vielzahl von verschiedenen Clients den Zugriff auf die an einen Server gebundenen geographischen Informationen zu gestatten und auf der anderen Seite mittels eines Clients Karten basierend auf den Datenbasen mehrerer Server zu erzeugen.

Der Focus dieser Entwicklungen ist auf eine performante Transformation von diskreten Simple Features in SVG-Elemente gerichtet. Eine Verarbeitung von kontinuierlichen Daten soll nur in unmittelbar vom SVG-Standard unterstützten Formaten behandelt werden. Die Simple Features soll der WMS mittels spezifischer Abfragen aus den mit ihm verbundenen Datenbasen selektieren und somit dem Konzept eines *Integrated WMS*[26] entsprechen. Jede dieser Abfragen soll dabei einen *Layer* darstellen. Der WMS

[26] OpenGIS Consortium (2002) a S.12

bereitet somit die Flut von Informationen, die die Datenbasen zur Verfügung stellen, auf und erleichtert folglich die Exploration der geographischen und sachlichen Daten.

Als Datenbasen dieses Systems sollen sowohl interne Informationssysteme wie Simple Feature for SQL-konforme Datenbanken oder Rasterdateien, als auch externe Web Services wie ein kaskadierend angebundener Web Map Service oder ein Web Feature Service dienen können. Darüber hinaus soll der Zugriff auf proprietäre Systeme, wie die verschiedenen geographischen Erweiterungen der Datenbanksystemhersteller, oder auf eine J2EE-Umgebung nicht ausgeschlossen sein.

3.1.2 **Web Map Service**

Nach dem OGC produziert ein Web Map Service Karten als visuelle Repräsentation von geographischen Daten. Diese Karten sollten generell in einem browserkonformen Rasterformat wie PNG, GIF oder JPEG gerendert werden können, aber auch Darstellungen in Vektorformaten wie SVG oder Web Computer Graphics Metafile (WebCGM) bzw. in applikationsspezifischen Rasterformaten sind möglich. Die Spezifikation für den im Folgenden behandelten Web Map Service 1.1.1[27] sieht allerdings keine konkrete Festlegung auf einzelne Formate vor. Sie beschreibt ausschließlich, mit welchen Operationen Karten, Sekundärdaten und Metadaten des Services angefragt werden können.

Das OpenGIS Consortium unterscheidet dabei zwischen zwei Ausprägungen eines WMS. Zum einen wird ein *basic WMS*[28] definiert, der über die beiden Standardoperationen *GetMap* und *GetCapabilities* und die optionale Operation *GetFeatureInfo* verfügt. Zum anderen spezifiziert das OGC einen *Styled Layer Descriptors Web Map Service (SLD WMS)*, der einen basic WMS um die zusätzlichen, optionalen Operationen *DescribeLayer*, *GetLegendGraphic*, *GetStyles* und *PutStyles* erweitert.

Die Aufgabe eines basic WMS besteht hauptsächlich aus der Bereitstellung der geographischen Daten in visueller Form auf Basis so genannter *Named Layer*. Jeder dieser Named Layer bildet eine spezifische Zusammenstellung einer endlichen Anzahl von features. Zur graphischen Repräsentation eines Layers bietet ein basic WMS mindestens einen vordefinierten Style an.

---

[27] OpenGIS Consortium (2001)

[28] OpenGIS Consortium (2001) c S. 1 ff.

Diese Funktionalität eines basic WMS wird bei einem SLD WMS dadurch erweitert, dass es dem User möglich wird, über einen so genannten *Styled Layer Descriptor* dem WMS mitzuteilen, wie ein Layer darzustellen ist und, sofern dies vom WMS unterstützt wird, welche Daten von welchem WFS diesen Layer bilden. Im Rahmen dieser Diplomarbeit haben die Konzepte eines SLD WMS nur eine periphere Bedeutung. Es soll aber das Konzept der benutzerdefinierten Signaturen bei speziellen Architekturentscheidungen Berücksichtigung finden, um eine spätere Erweiterung zu gestatten.

Fokussiert wird die Konzeption und Implementierung eines basic WMS mit den beiden Standardoperationen GetMap und GetCapabilities. Das OGC definiert für diese Operationen ausschließlich, in welcher Form diese aufgerufen bzw. beantwortet werden. Auf welche Weise die Implementierung erfolgt ist nicht Teil der Spezifikation.

Zur Kommunikation mit einem WMS schreibt das OGC die Nutzung des *Hypertext Transfer Protocol (HTTP)* vor. Der Aufruf der einzelnen Operationen erfolgt über die (ggf. operationsspezifische) *Uniform Resource Locator (URL)* mittels der *GET*-Methode des Hypertext Transfer Protocols. Die Kodierung der Parameter dieser Anfragen erfolgt gemäß der üblichen Verwendung[29]. Bei den Parameternamen soll die Schreibweise unberücksichtigt sein, im Gegensatz dazu ist die Verarbeitung der Parameterwerte case-sensitive. Sollte der Wert eines Parameters aus einer Liste von Einzelwerten bestehen, werden diese durch Kommata getrennt. Die Darstellung von Zahlen erfolgt nach angelsächsischer Notation.

3.1.2.1 GetCapabilities

Die GetCapabilities-Operation liefert die Metadaten des WMS. Sie umfassen sowohl die Beschreibung der unterstützten Operationen und Parameter, als auch detaillierte Angaben über die angebotenen Layer des WMS. Neben der reinen Beschreibung des Inhalts und der Dienste dienen diese Metadaten der automatischen Registrierung des WMS in Katalogsystemen.

Die Parameter der GetCapabilities-Anfrage-URL sind in der nachstehenden Tabelle aufgelistet.

[29] The Internet Society (1999)

Parametername	Parameterwert	Erforderlich(E)/ Optional(O)	Beschreibung
VERSION	version	O	Geforderte Version des WMS
SERVICES	„WMS“	E	Service-Art
REQUEST	„GetCapabilities“	E	Operation
UPDATESEQUENCE	string	O	Sequenznummer zur clientseitigen Cache-Verwaltung

Tabelle 1 - Parameter der GetCapabilities-Operation

Der optionale Parameter UPDATESEQUENCE sollen von dem zu implementierenden WMS nicht unterstützt werden. Für ein tiefer gehendes Verständnis dieses Parameters sollte die Spezifikation des WMS 1.1.1 herangezogen werden.

Die Antwort des WMS auf eine GetCapabilities-Anfrage erfolgt in Form eines gültigen XML-Dokumentes gemäß der vom OpenGIS Consortium veröffentlichten *Document Type Definition (DTD)*[30]. Der MIME-Type dieser HTTP-Response ist nach den Vorgaben des OGC „`application/vnd.ogc.wms_xml`“.

Das Capabilities-Dokument gliedert sich in drei Teile auf:

- Der erste Teil besteht aus den Metadaten des Services als solchem. Diese können Angaben über Namen und Titel, eine Stichwortliste und eine Beschreibung des Services, Informationen über Gebühren und Zugriffseinschränkungen zur Nutzung und Kontaktinformationen des Betreibers umfassen.
- Der zweite Teil des Dokumentes beschreibt die unterstützten Operationen und deren Parameter. Angeführt werden die URL der einzelnen Operationen, die unterstützten Formate zur Generierung der Karten bzw. die der Ausnahmebeschreibungen und mögliche proprietäre Parameter.
- Der abschließende Teil beschreibt den angebotenen Inhalt, also die Layer des WMS. Dabei wird zwischen Layern und Named Layern unterschieden. Ausschließlich Named Layer können per GetMap-Operation abgefragt werden und beinhalten direkt geographische Daten. Layer ohne Namen dienen der hierarchischen Strukturierung und können Attribute an die untergeordneten Layer vererben. Wichtige Attribute eines Named Layers sind neben dem Namen

[30] OpenGIS Consortium (2001) d

und dem Titel die vordefinierten Signaturen (Style), mit denen der Layer dargestellt werden kann, sowie die unterstützten räumlichen Bezugssysteme (SRS), die Lage und Begrenzung des Layers im EPSG:4326[31] geographic coordinate reference system (LatLonBoundingBox) und die Lage und Begrenzung angegeben in den jeweilig unterstützten räumlichen Bezugssystemen (BoundingBox). Weitere Attribute umfassen Angaben über die Dimension, die Metadaten und die skalierungsabhängige Sichtbarkeit des Layers.

3.1.2.2 GetMap

Die GepMap-Operation eines basic WMS hat die Aufgabe, in Abhängigkeit der in Tabelle 2 aufgelisteten Parameter digitale Karten zu erzeugen. Diese Karten können entweder durch Rasterbilder oder Vektorgrafiken repräsentiert werden. Sollte eine Anfrage aus verschiedenen Gründen nicht korrekt verarbeitet werden können, muss der Service eine Fehlermeldung im angefragten Format zurückliefern.

Parametername	**Parameterwert**	**Erforderlich(E)/ Optional(O)**	**Beschreibung**
VERSION	version	O	Geforderte Version des WMS
REQUEST	„GetMap“	E	Operation
LAYERS	layer_list	E	Liste mit minimal einem Layer
STYLES	style_list	E	Liste von Darstellungs-stilen analog zu den angefragten Layern
SRS	namespace:identifier	E	räumliches Bezugsystem
BBOX	minx,miny,maxx,maxy	E	Begrenzung des Darstellungsbereichs in Einheiten des SRS
WIDTH	output_width	E	Breite in Pixel
HEIGHT	output_height	E	Höhe in Pixel
FORMAT	MIME TYPE	E	Format der Karten als MIME-Type

[31] Das weltweit gebräuchlichste Polarkoordinatensystem, mit dem unter anderen die GPS-Satelliten arbeiten.

TRANSPARENT	„TRUE/FALSE“	O	Transparenter Hintergrund
BGCOLOR	color_value	O	Hintergrundfarbe
EXCEPTIONS	exception_format	O	Format der Fehlermeldung
TIME	time	O	Datumsangabe für sich ändernde Layer
ELEVATION	elevation	O	Höhenangabe
Vendor-specific parameters		O	Optionale Parameter des spezifischen Services

Tabelle 2 - Parameter der GetMap-Operation

Die nach dem Portrayal Model aufgebaute GetMap-Operation eines WMS besteht aus einer Komponente zur Anfragebearbeitung und verschiedenen Komponenten zur Generierung der Display Elements der angebotenen Layer sowie Renderkomponenten für das entsprechende Format.

Die GetMap-Operation selbst stellt die übergeordnete Anfragebearbeitungskomponente dar, die durch den obligatorischen REQUEST-Parameter festgelegt wurde. Sie entscheidet mittels des LAYERS-Parameters, für welche Layer die Display Elements generiert werden sollen und wählt abhängig vom FORMAT-Parameter die gegebenenfalls benötigte Renderkomponente aus. Der EXCEPTIONS-Parameter gibt an, in welchem Format eventuell auftretende Fehler ausgegeben werden sollen. Darüber hinaus ermittelt die Anfragebearbeitungskomponente für die einzelnen Display Elements Generator-Komponenten aus dem STYLES-Parameter den Style des jeweiligen Layers und ordnet gemäß der Layerlistenreihenfolge die Aufrufe der Display Elements Generator-Komponenten und somit die Darstellungsreihenfolge der einzelnen Layer in der resultierenden Karte.

Die Display Elements Generator-Komponenten haben die Aufgabe aus der mit ihnen verbundenen Datenbasis die features zu selektieren und unter Verwendung der BBOX- und SRS-Parameter in Display Elements zu transformieren. Der BBOX-Parameter dient dabei sowohl der Bestimmung des karteneigenen Koordinatensystems, als auch der Filterung der features aus der Datenbasis, sofern dieses von der Datenbasis unterstützt wird. Zur weiteren Spezifikation der geforderten features dienen der TIME-Parameter, der bei sich temporal ändernden features wie zum Beispiel bei Wetterdaten benutzt werden kann, und der ELEVATION-Parameter, der zur Festlegung der dritten

Koordinate wie beispielsweise einer bestimmten Atmosphärenschicht verwendet werden kann. Die Signaturen der graphischen Elemente werden durch den übergebenen Style bestimmt.

Mittels der WIDTH- und HEIGHT-Parameter wird für die Renderkomponenten, sofern das geforderte Format ein Rendern der Display Elements vorsieht, die Auflösung der Rasterkarte bestimmt. Der BGCOLOR-Parameter kann zur Festlegung einer anderen als der standardmäßigen weißen Hintergrundfarbe benutzt werden. Sofern es das angefragte Format vorsieht, wie beim PNG- und GIF-Format, kann über das TRANSPARENT-Flag der Hindergrund als transparent markiert werden. Als weitere das Ausgabeformat beeinflussende Parameter wären Angaben über die Qualität, die Farbtiefe, die Interpolation und die Druckauflösung denkbar.

3.1.2.3 GetFeatureInfo

Von der Konzeption der optionalen GetFeatureInfo-Operation wird vorerst abgesehen, da der zu unterstützende SVG-Standard einen wesentlich interaktiveren und vor allem direkteren Zugriff auf die Sachdaten eines features zulässt. Diese Sachdaten gehen natürlich beim serverseitigen Rendern der SVG-Karten zu Karten im Rasterdaten-Format verloren, so dass den auf Rasterdaten basierenden Clients keine Möglichkeit zur Sekundärdatenexploration gegeben ist.

3.1.3 Datenbasis

3.1.3.1 Diskrete Daten

Die Datenbasis bildet das Herz jeglicher Informationsverarbeitung. Häufig werden Datenbasen in entsprechende Managementsysteme eingebunden, die eine standardisierte und effiziente Speicherung, Änderung und Abfrage der Daten zulassen. Besonderes für diskrete Daten nehmen solche *Database Management Systems (DBMS)* eine herausragende Stellung ein. Neben einer standardisierten Sprache zur Erstellung, Abfrage und Änderung der Daten verfügen DBMS meist über eine Transaktionskontrolle für den Multi-User Betrieb und ein Ausfallmanagement. Je nach Strukturierung der Daten wird zwischen (objekt-)relationalen, hierarchischen und objektorientierten Datenbasen unterschieden.

Für die Konzeption eines WMS steht allerdings die effiziente Selektion der Simple Features aus den zum Teil sehr großen Datenbeständen unter Verwendung von Filtern auf den geographischen und sekundären Attributen im Vordergrund.

3.1.3.1.1 **Simple Feature for SQL**

Das OpenGIS Consortium hat für den Zugriff auf geographische Daten, die in einer (objekt-)relationalen Datenbank gespeichert sind, die Spezifikation *Simple Feature for SQL (SFS)* verabschiedet. Jedes feature wird dabei durch eine Zeile einer so genannten *feature table* repräsentiert. Diese Tabelle besteht aus *geometry columns* in denen die Geometrien der features abgelegt sind, und weiteren nativen Spalten, die die nichträumlichen Attribute eines features beinhalten.

Das OGC unterscheidet drei verschiedene Realisierungsformen für den Zugriff und die Speicherung der Geometrien. Die ersten beiden verwenden ausschließlich den SQL92-Standard und ermöglichen entweder numerisch-relationale oder binäre Anfragen der Geometrien. Die dritte Realisierungsform erweitert den SQL92-Standard und wird *SQL92 with Geometry Types* genannt. Sie baut auf dem objekt-relationalen Konzept auf und spezifiziert neben dem Zugriff auf die Geometrie im WKBGeometry- und WKTGeometry-Format auch Funktionen für diese Geometrie-Typen. Weiterhin werden in dieser SQL92-Erweiterung Tabellenstrukturen zur Referenzierung und somit zur Konsistenzkontrolle der räumlichen Bezugssysteme der Geometrien festgelegt.

Die durch SQL92 with Geometry Types definierten Funktionen lassen sich in drei Kategorien einteilen:

- Die Funktionen der ersten Kategorie beziehen sich auf die Geometrie als solche und liefern zum Beispiel für alle Geometrie-Typen den jeweiligen Type, die ID des räumlichen Bezugssystems, die WKBGeometry-/WKTGeometry-Repräsentation, die Dimension oder auch die minimale, rechteckige Begrenzung. Darüber hinaus sind für die einzelnen Geometrie-Typen spezifische Funktionen definiert. So lassen sich durch diese Funktionen, um nur einige Beispiele zu nennen, die Länge und die Geschlossenheit eines LineStrings, die Anzahl der inneren Ringe und die Fläche eines Polygons oder die Anzahl der Geometrien einer GeometryCollection ermitteln. Weiterhin ermöglichen Funktionen dieser Kategorie das Erzeugen von Geometrien aus der WKBGeometry-/WKTGeometry-Repräsentation, was für die Speicherung von Geometrien eine notwendige Bedingung darstellt.
- Die Funktionen der zweiten Kategorie sind mengentheoretisch und konstruktiv orientiert. So sind Funktionen zur Bestimmung der Schnitt-, Vereinigungs- oder Differenzmenge zweier Geometrien gegeben. Ebenso existieren Funktionen zur

Berechnung eines Puffers um eine Geometrie oder der konvexen Hülle einer Geometrie.

- Die dritte Kategorie wird durch Funktionen bestimmt, die die unterschiedlichen räumlichen Beziehungen von zwei Geometrien zueinander überprüfen. Dabei sind die Funktionen zur Überprüfung einer nichtleeren Schnittmenge zweier Geometrien (intersects) und zur Ermittlung, ob eine Geometrie vollständig von einer anderen eingeschlossen wird (contains), von elementarer Bedeutung. Diese Funktionen ermöglichen in Abhängigkeit der gewünschten räumlichen Begrenzung (BBOX) eine effiziente Selektion der Simple Features aus der Datenbasis.

SQL92 with Geometry Types-konforme Datenbanksysteme bieten aufgrund dieser erweiterten relationalen Algebra weitreichende Möglichkeiten zur Selektion, Filterung, Verdichtung und Analyse von geographischen Daten. Besonders im Intranet gewährleisten sie mittels standardisierter Protokolle eine performante Abfrage der Simple Features und sind somit eine bedeutende Datenbasis für den zu entwickelnden Integrated Web Map Service.

3.1.3.1.2 **Web Feature Service**

Analog zu Simple Feature for SQL wurde vom OpenGIS Consortium die Spezifikation für einen *Web Feature Service (WFS) 1.0* [32] verabschiedet. Neben den Operationen *GetCapabilities* und *DescribeFeatureType* und den optionalen *Transaction*- bzw. *LockFeature*-Operationen, die zum transaktionssicheren Einfügen, Ändern und Löschen von Simple Features dienen, bietet ein WFS vor allem die *GetFeature*-Operation, die die Funktionen eines SELECT-Statements an eine SFS-konforme Datenbank übernimmt.

Die Anfrage der GetFeature-Operation erfolgt mittels spezifischer[33] XML-Dokumente, die per HTTP-POST-Request an den Service gesendet werden. Das `GetFeature`-Wurzelelement beinhaltet dabei mindestens ein `Query`-Element, das den gewünschten feature type als Attribut und die zu selektierenden Attribute des features als `PropertyName`-Kindelemente enthält. Zusätzlich ist ausschließlich noch die Angabe eines

[32] OpenGIS Consortium (2002) b

[33] OpenGIS Consortium (2003) c

speziellen `Filter`-Elementes möglich, das nach der *Filter Encoding Implementation Specification*[34] definiert ist.

Die Addressierung der zu filternden Attribute basiert auf einer Teilmenge der *XML Path Language specification (XPath)*[35]. Zur Filterung der sekundären Attribute dienen einfache Vergleichsoperatoren wie gleich, größer und kleiner. Die geometrischen Attribute können mit Hilfe von Operatoren zur Bestimmung von räumlichen Beziehungen gefiltert werden. Weiterhin stehen die logischen Operatoren `and`, `or` und `not` zur Verfügung, die den Aufbau von komplexen Filterausdrücken gestatten.

Die DescribeFeatureType-Operation hat die Funktion, die jeweiligen feature types, die ein WFS anbietet, durch ein XML-Schema zu beschreiben. Dieses so genannte *GML application schema* erbt von den abstrakten GML2-Schemata des OGC und dient somit zur vollständigen Beschreibung des Antwortdokumentes der GetFeature-Operation. Das `FeatureCollection`-Element bildet die Wurzel dieses GML 2-konformen Antwortdokumentes. Das `FeatureMember`-Element, als einziges und unbegrenztes Kindelelement der Wurzel, fungiert als Container der einzelnen GML application schema-konformen feature instances.

Diese einfache und zugleich flexible Struktur ist die Grundlage für eine breite Akzeptanz der Web Feature Services. Diese Services ermöglichen eine standardisierte und von der eigentlichen Datenhaltung abstrahierte Selektion und Filterung der gespeicherten features. Die persistente Datenhaltung kann nach diesem Konzept sowohl mittels einer relationalen Datenbank, als auch durch XML- oder objektorientierte Datenbanken erfolgen, da die flache Struktur des Ausgabedokumentes eine Transformation aus jeder dieser Speicherarten ermöglicht. Weiterhin stellt die Kommunikation über das Hypertext Tansfer Protocol einen internetweiten, plattformunabhängigen, „proxy- und firewallfreundlichen“ Zugriff sicher und macht somit die Unterstützung von Web Feature Services als Datenbasis zu einem integralen Bestandteil des zu entwickelnden Integrated Web Map Services.

3.1.3.2 Kontinuierliche Daten

Die Verarbeitung von kontinuierlichen Daten ist im Gegensatz zur Verarbeitung der oben beschriebenen diskreten Simple Features weitaus weniger standardisiert. Diese

---

[34] OpenGIS Consortium (2001) e

[35] W3C (1999)

Rasterdaten umfassen zum Teil eine Vielzahl von Bändern, wie zum Beispiel die verschiedenen Spektren einer fernerkundlichen Aufnahme. Die Formate dieser Daten sind meist für eine wissenschaftliche Analyse und Bearbeitung ausgelegt und nur mit entsprechenden Applikationen zu verarbeiten. Eine Selektion des relevanten Bildausschnittes aus den zum Teil großflächigen, hochauflösenden Rasterdaten und die Berechnung der gewünschten Auflösung verlangen zudem effiziente Indexstrukturen und entsprechende Tansformationsalgorithmen. Diese Aufgaben sollen nach den Konzepten des OGC die Coverage Access Services übernehmen.

Da im Rahmen dieser Diplomarbeit der SVG-Standard als Beschreibungssprache der Display Elements dienen soll, werden die möglichen Rasterdatenformate auf die von SVG unterstützten Formate JPEG und PNG beschränkt.

3.1.3.2.1 **Raumbezogene Rasterbilder**

Rasterbilder werden bei SVG durch das `image`-Element als Referenz eingebunden und erst von der Renderkomponente separat geladen. Die Display Elements Generator-Komponenten eines WMS generieren für einen Rasterdaten beinhaltenden Layer also ausschließlich die `image`-Elemente mit den Referenzen auf die konformen Rasterbilder und ihren Positionskoordinaten innerhalb der resultierenden Karte. Als Datenbasis für einen solchen Rasterdaten-Layer werden entsprechend nur die Referenz bzw. im konkreten Fall die URI auf die Bilddatei und die Begrenzungskoordinaten im angefragten räumlichen Bezugssystem benötigt. Für große Rasterbilder wie zum Beispiel Luftbilder, die leicht Größen von mehreren hundert Megabytes annehmen können, ist es sinnvoll, diese in viele kleinere zu zerteilen, deren jeweiliges Begrenzungsrechteck und entsprechende URI die Attribute eines diskreten Simple Features bilden und in einer feature table abgelegt werden können. Dies ermöglicht eine effiziente Selektion von Ausschnitten eines ursprünglich großen Rasterbildes.

3.1.3.2.2 **Kaskadierender Web Map Service**

Auf ähnliche Art und Weise können nachgeordnete Web Map Services eingebunden werden. Im Gegensatz zu statischen Rasterbildern kann hierbei die URL entsprechend den Parametern der primären GetMap-Operation moduliert werden. Zum einen können somit geographische Daten verarbeitet werden, die nicht zwingend kontinuierlich, aber nur mittels WMS verfügbar sind. Zum anderen kann ein solcher kaskadierend angebundener Web Map Service als Client eines nachgelagerten Web Coverage

Services fungieren. Eine solche Konstruktion ermöglicht eine transparente Einbindung von ursprünglich nicht SVG-konformen Rasterdatenformaten.

3.1.4 **Thin -, medium - und thick clients**

Nach dem Portrayal Model wird zwischen drei möglichen Web Mapping Architekturen unterschieden, je nachdem welche Teile des Darstellungsprozesses vom Client übernommen werden bzw. welche Art von Daten der Client verarbeitet.[36]

Der *thick client* greift direkt auf die Datenbasis zu und stellt dem User umfassende Funktionen zur Selektion, Bearbeitung und Darstellung der Simple Features zur Verfügung. Der thick client übernimmt die gesamten, zum Teil rechenaufwändigen, Transformationsschritte zur Darstellung der geographischen Daten. Realisiert werden kann ein solcher Client durch eine eigenständige Applikation oder eingebettet in den Browser durch ein Applet oder Plugin. Ein thick client verlangt genügend Ressourcen seitens der Hardware und detaillierte Kenntnisse der Datenbasis seitens des Users. In gewisser Weise übernimmt ein Web Map Service die Funktion eines thick clients hinsichtlich der Transformation der geographischen Daten inklusive des gegebenenfalls nötigen Renderns. Der Layer-Autor übernimmt dahingehend die Funktion des Users bezüglich der Definition der einzelnen Layer.

Konzipiert werden soll im Rahmen der Diplomarbeit ein WMS, der einerseits für die Kommunikation mit *thin clients*, die ausschließlich die Anzeige der gerenderten Karten übernehmen, ausgelegt ist und andererseits *medium clients* den Zugriff auf die erzeugten Display Elements ermöglicht, was somit eine vom Benutzer kontrollierte Renderung gestattet.

[36] vgl. OpenGIS Consortium (2003) a S. 39 ff.

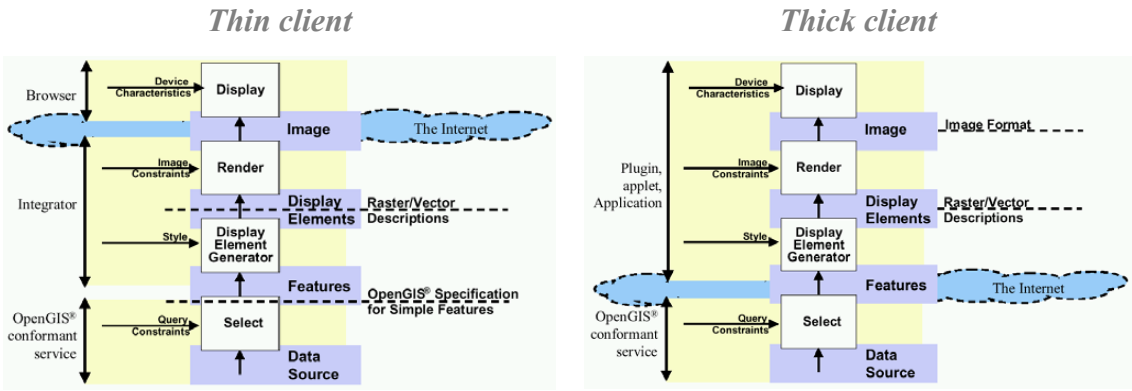

Abbildung 5 - Thin vs. thick clients for portraying features over the Internet[37]

Thin clients stellen die geringsten Anforderungen an die Hardware und Software des vom Benutzer verwendeten Gerätes. Jegliche Interaktion mit der Karte wie zoom oder pan führt allerdings zu einer serverseitigen Neuberechnung und dem wiederholten Transport der Karte über das Netz. Weiterhin ist die auf Bildkoordinaten basierende Rückermittlung des gewünschten features bei einer GetFeatureInfo-Operation nur mit hohem serverseitigen Rechen- und Speicheraufwand möglich. Thin clients sollten daher nur auf Geräten eingesetzt werden, die entweder nicht über die nötigen Ressourcen zum Rendern der Display Elements seitens der Hardware verfügen, wie zum Beispiel Smartphones oder kleine PDAs, oder deren Softwareausstattung eine Unterstützung der Display Elements (hier in SVG) verhindert.

Neuere Entwicklungen in diesem Bereich versprechen allerdings sowohl einen browserweiten SVG-Support[38] als auch die Unterstützung von SVG auf Smartphones und Handhelds. Dazu wurde eigens ein vereinfachter SVG-Standard mit Namen *SVG Tiny (SVGT)*[39] verabschiedet, der aufwändige graphische Filtereffekte zwar nicht unterstützt, aber für die Darstellung der Simple Features und der nötigen Signaturen vollkommen ausreichend ist.

Medium clients stellen einen flexiblen Mittelweg zwischen diesen beiden oberen Extremen dar. Sie verhindern, dass der Benutzer detaillierte Kenntnisse über die zugrunde liegenden Datenbasen benötigt, und ermöglichen stattdessen mittels des Layer-Konzeptes einen transparenten Zugang zu den aufbereiteten geographischen Informationen der Datenbasen. Auf der Darstellungsseite bieten sie dem Benutzer durch die eigene Renderkomponente die Möglichkeit, auf Basis der übermittelten Display

[37] vgl. OpenGIS Consortium (2003) a S. 40

[38] W3C (2004)

[39] W3C (2003) a

Elements die Skalierung und den Darstellungsausschnitt der Karte frei und ohne weitere Kommunikation mit dem WMS bestimmen zu können. Dadurch, dass die semantische Struktur der Karte dem medium client bekannt ist, können dem Endanwender auch Mechanismen zur Exploration der Sekundärdaten angeboten werden. Die GetFeatureInfo-Operation wird quasi in den Client integriert. Dieses Konzept des medium clients führt somit zur Realisierung eines *thin servers*, der ausschließlich für die Selektion und Transformation der Simple Features zuständig ist und sinnvollerweise die rechenaufwendige Renderung der Karten an den Client überträgt.

3.2 Anforderungen an den Server

3.2.1 Modularisierung und Erweiterbarkeit

Die Generierung einer Karte besteht im Wesentlichen aus zwei variablen Komponenten. Zum einen haben die Display Elements Generator-Komponenten, im Folgenden *Layerkomponenten* genannt, die Aufgabe, aus einer Vielzahl von möglichen Datenbasen Layer zu selektieren und diese in eine SVG-konforme Repräsentation zu transformieren. Jede Layerkomponente entspricht dabei einem Named Layer der Capabilities des WMS. Diese SVG-konformen Layer werden durch eine Gruppe von transformierten Simple Features dargestellt. Dieses Layer-bildende Gruppierungselement beinhaltet auch den Style des Layers. Ergänzt um das umschließende `svg`-Element, das auch das karteneigene Koordinatensystem festlegt, bilden diese Layer die angefragte Karte. Die Aufgabe der Renderkomponenten ist es, auf der anderen Seite diese Karte in das angefragte Ausgabeformat zu transformieren. Da das Ausgabeformat sowohl verschiedene Rasterformate als auch die alpha-numerische Repräsentation von SVG sowie weitere denkbare Formate umfassen kann, sollen die Renderkomponenten im Folgenden allgemeiner *Serialisierungskomponenten* genannt werden.

Die einzelnen Layer werden nach den Vorgaben der GetMap-Operation durch die Parameter BBOX, SRS, TIME und ELEVATION sowie über den jeweiligen Style spezifiziert. Diese Parameter bilden die Attribute eines Interfaces, dem, erweitert um eine Methode zur Generierung der SVG-Repräsentation, jede Layerkomponente genügen muss. Die Selektion und Transformation der jeweiligen geographischen Daten wird durch dieses Interface gekapselt, was eine breite Unterstützung von verschiedenen Datenbasen wie SFS-konformen Datenbanken, Web Feature Services, georeferenzierten Rasterbildern usw. ermöglicht. Weiterhin sollen somit Implementierungen von Layerkomponenten realisiert werden können, die die Fähigkeiten der Datenbasen

erweitern und eine eigenständige Aufarbeitung der geographischen Daten übernehmen. Ein Beispiel dafür wäre die Visualisierung von topologischen Strukturen in einer geographischen Karte, wie beispielsweise U-Bahn-Verbindungen in einem Stadtplan. Grundsätzlich sollen die einzelnen Layer dynamisch mittels des Namens geladen und verwendetet werden können. So soll der WMS nicht nur prinzipiell sondern auch zur Laufzeit erweiterbar sein. Ebenso soll zur Laufzeit eine Änderung der nötigen Konfiguration eines Layers, die unter anderem den Anfrageausdruck an die Datenbasis umfassen kann, möglich sein.

Für die Serialisierungskomponenten sollen ähnliche Anforderungen bezüglich der Modularisierung und Erweiterbarkeit gelten wie bei den Layerkomponenten. Analog lässt sich durch die vorgegebenen Anfrageparameter und Methoden zur Festlegung des Ausgabestroms bzw. zur Übergabe der SVG-Karte ein für alle Serialisierungskomponenten verbindliches Interface spezifizieren. Inwieweit die übergebenen Parameter von der jeweiligen Serialisierungskomponente genutzt werden ist vom angefragten Ausgabeformat abhängig. Weiterhin sollten die Serialisierungskomponenten abhängig vom FORMAT-Parameter geladen werden können. Ein Mapping der MIME-Types und der jeweiligen Serialisierungskomponenten soll variabel konfiguriert werden können, um die prinzipielle Erweiterung des Web Map Services ohne Änderung der Quelle zu ermöglichen.

Darüber hinaus sollten alle weiteren Komponenten des Services über Interfaces angesprochen werden, um eine prinzipielle Austauschbarkeit dieser Komponenten unter Beibehaltung aller anderen zu gewährleisten. Die Konfiguration der einzelnen Komponenten und die Referenzierung anderer Komponenten sollen flexibel über externe Konfigurationsdaten erfolgen, die je nach Art auch zur Laufzeit änderbar sein sollen.

3.2.2 **Performance und Skalierbarkeit**

3.2.2.1 „on the fly“-Generierung und Echtzeit-Karten

Die Architektur des Systems soll es gestatten, den Service in einem breiten Spektrum von Anwendungsszenarien einsetzen zu können. So soll auch der worst-case behandelt werden können, in dem eine Vielzahl von Usern (> 100) gleichzeitig auf sich temporal rasch ändernde Daten (< 5 min) wie zum Beispiel Verkehrs- oder Wetterdaten zugreifen. Darüber hinaus soll es sogar möglich sein, dass ein Benutzer mittels entsprechenden Web-Interfaces die Datenbasis selbst, beziehungsweise vornehmlich die

sekundären Attribute eines features, ändern kann. Die so veränderte Karte soll in Echtzeit, quasi „on the fly“, entsprechend der geforderten Parameter wie Kartenausschnitt, Style und Auflösung generiert werden können.

Weiterhin stellt die Verarbeitung von weiträumigen, zum Teil sehr detaillierten geographischen Informationen einen gewichtigen Punkt im Anforderungsspektrum des Servers dar. Zusätzlich sollen generierte Karten in für den Benutzer akzeptabler Geschwindigkeit und unter Verwendung der heutigen Bandbreiten (min. 7 KB/s [Modem/GPRS]) übermittelt werden können.

Im diametralen Verhältnis zu den artikulierten Anforderungen an das System stehen die umfangreichen Datenmengen von geographischen Informationen. So umfasst schon eine SVG-Karte der aus einzelnen Flurstücken des Liegenschaftskatasters zusammen gesetzten Straßen des Stadtgebietes einer Mittelstadt (Nordhorn ca. 52.000 Einwohner, 48 km² Fläche) ungefähr 1,7 Megabyte. Diese hohen Datenmengen sind nicht nur für den schnellen Transport der Karten über das Netz nur bedingt geeignet, sondern stellen auch an die Verarbeitung der Daten seitens des Servers bezüglich Speichernutzung und Geschwindigkeit massive Anforderungen.

Diese Anforderungen und Gegebenheiten verlangen eine Softwarearchitektur, die entsprechend der spezifischen Anforderungen der einzelnen Anwendungsszenarien eine lineare Skalierung der Hardware gestattet und in allen Bereichen so ressourcenschonend wie möglich konzipiert ist, aber die oben genannte Flexibilität bezüglich Modularisierung und Erweiterbarkeit nicht einschränkt.

In den folgenden Abschnitten werden die Konzepte und Methoden vorgestellt, mittels derer die an das System gestellten Anforderungen erfüllt werden können.

3.2.2.2 Multi-Tier-Architektur

Eine Möglichkeit zur besseren Skalierungsfähigkeit des Systems ist die Verteilung der verschiedenen Komponenten. Diese verteilten Komponenten, auch Tiers genannt, können dabei sowohl auf einem einzelnen Computer residieren als auch über mehrere Computer verteilt sein. Auch die Verteilung eines einzelnen Tiers auf mehrere Rechner, Cluster genannt, ist denkbar. Entscheidend für ein verteiltes System ist ein Kommunikationsprotokoll, das einen transparenten Zugriff auf die jeweiligen vernetzten Tiers gestattet.

Das zu entwickelnde System baut auf der Multi-Tier-Architektur des OGC Service-Frameworks auf. Alle an den WMS angebundenen Data-, Portrayal- und Processing Services werden aus Sicht des WMS zu einem Tier zusammengefasst. So besteht das zu entwickelnde System aus mindestens vier Tiers wie den verschiedenen Datenbasen, dem WMS-Application Server, dem Web-Server und den Clients. Hierbei spaltet sich der WMS, vereinfacht ausgedrückt, wieder in die Layerkomponenten und die Serialisierungskomponenten auf. Ziel ist es, jedem dieser Tiers klar definierte Aufgaben zuzuordnen, die jeweils den Stärken der verwendeten Komponenten entsprechen und darüber hinaus einen transparenten Austausch der einzelnen Tiers gestatten.

Weiterhin soll der Input eines Tiers so festgelegt werden, dass jede Komponente ausschließlich die Menge an Informationen und deren Strukturierung verarbeitet, die für den jeweiligen Transformationsschritt benötigt wird. So ist die gesamte semantische Struktur der Karte, also das gesamte, hierarchisch aufgebaute, intern verknüpfte SVG-Dokument, erst für die Renderung dieser Karte nötig. Alle vorherigen Transformationsschritte finden auf der Ebene der Simple Features statt und können somit sequenziell verarbeitet werden, was eine geringe Speicherbelastung zur Folge hat, da jedes verarbeitete Simple Feature direkt an die nachgelagerte Transformationskomponente weitergereicht werden kann. Insbesondere führt die Verlagerung der Renderkomponente auf den Client zu einer erheblichen Entlastung des Servers, da diese eine ausschließlich sequentielle und dadurch ressourcenschonende Abarbeitung der Anfrage ermöglicht.

3.2.2.3 Multithreading

Der hohe Grad an verlangter Nebenläufigkeit des Systems erfordert Multithreading-Technologien zur Behandlung der einzelnen Anfragen an den Server. Neben den Geschwindigkeitsvorteilen bei der Instanziierung eines Anfragebearbeitungsthreads im Vergleich zum Aufruf eines neuen Prozesses zur Anfragebearbeitung, ermöglicht die Multithreading-Technologie die Nutzung von gemeinsamen Ressourcen innerhalb des gemeinsamen Speichers des umgebenden Betriebssystemprozesses.

Sollte es sich bei diesen gemeinsamen Ressourcen, auch Betriebsmittel genannt, um exklusiv benutzbare Betriebsmittel handeln, wie zum Beispiel eine Verbindung zu einer Datenbank, sind Synchronisationsmechanismen notwendig, so dass diese Ressource sicher von den konkurrierenden Threads genutzt werden kann. Meist führt ein solcher

Synchronisationsmechanismus zu einer Serialisierung der Zugriffsversuche und somit zu einer zeitlichen Verzögerung der konkurrierenden Threads.

3.2.2.4 Caching und Pooling

3.2.2.4.1 **Ressourcen-Pools**

Eine Möglichkeit die zeitliche Verzögerung zu verhindern bzw. zu minimieren ist die redundante Vervielfältigung der exklusiven Ressourcen. Die duplizierten Betriebsmittel werden in so genannten Pools verwaltet. Der Pool sorgt für einen threadsicheren Zugriff auf die einzelnen Ressourcen. Sollte ein Betriebsmittel wiederverwendbar sein, wird es nach der Benutzung von dem benutzenden Thread an den Pool zurückgegeben. Meist wird ein solcher Pool von einem eigenen Kontrollthread verwaltet, der die Mindest- und Höchstzahl der Betriebsmittel im Pool überwacht und gegebenenfalls ein weiteres Betriebsmittel erzeugt oder ein nicht mehr benötigtes entfernt.

Anwendung finden solche Pools besonders bei Datenbankverbindungen, da der Aufbau der Verbindung und die Anmeldung des User gerade bei DBMS mit ausgefeilten Rechtesystemen eine Latenzzeit vom mehreren Sekunden verursachen können. Aber auch komplexere Objekte, deren Instanziierung nicht unerhebliche Kosten verursacht und die prinzipiell wiederverwendbar sind, sollten in solchen Pools verwaltet werden.

So ist die Verwaltung der einzelnen Layerkomponenten in Pools vorzunehmen. Gerade bei Layern, deren Datenbasis eine Datenbank ist, bieten diese Konzepte große Geschwindigkeitsvorteile, da jede Layerkomponente somit über eine eigene, quasi persistente Verbindung zur Datenbank verfügt. Über diese Verbindung kann bei der Instanziierung der Layerkomponente das jeweilige Statement zur Selektion der features präkompiliert werden. Bei der Generierung der Display Elements wird dieses vorbereitete Statement nur noch um die relevanten BBOX-, TIME- und ELEVATION-Parameter ergänzt und kann direkt ohne weiteren Verbinbungsaufbau und Interpretations- bzw. Optimierungsaufwand seitens der Datenbank ausgeführt werden.

Der Nachteil dieses Konzeptes ist, dass jede dieser Verbindungen durch zwei Threads, jeweils einen innerhalb der Datenbank und einen innerhalb der Clientapplikation, gehalten wird. Bei sehr vielen Layern und großen Pools kann es so zu einem nicht zu vernachlässigenden Verwaltungs-Overhead und einem Konflikt mit der maximalen

Anzahl von Usern der Datenbank und Threads des Betriebsystems[40] kommen. Somit ist eine flexible Konfiguration der Pools bezüglich Mindest- und Höchstzahl der Mitglieder und deren Abbaugeschwindigkeit nötig.

3.2.2.4.2 **Content-Caching**

Caching stellt eine der effektivsten Möglichkeiten dar, die Antwortzeit einer Anfrage signifikant zu verringern. Statt einer Neuberechnung der dynamischen Daten werden diese einfach aus dem Speicher gelesen. Für die Validierung der Gültigkeit der gecachten Daten sind zwei Verfahren denkbar. Zum einen kann die vorgelagerte Cachingkomponente explizit den Erzeuger der Daten die Gültigkeit dieser in Abhängigkeit der Anfrage prüfen lassen. Zum anderen kann der Erzeuger der Cachingkomponente implizit mitteilen, wie lange die Daten gültig sind.

In der Umgebung eines WMS sind mehrere Caching-Möglichkeiten vorhanden. So könnten an jeder Schnittstelle zwischen den einzelnen Komponenten die Daten gepuffert werden. Allerdings ergeben sich dabei verschiedene Probleme.

Grundsätzlich hat der WMS keinen Einfluss auf die an ihn gebundenen Datenbasen. Die Selektion der Daten sollte ausschließlich über die vorgegebenen Protokolle erfolgen und spezifische Implementierungen zur expliziten Validierung der Gültigkeit vormals selektierter Daten ausschließen. Inwieweit die angebundenen Datenbasen ihrerseits für ein Caching der Anfragen und somit zur Steigerung der Anfragebearbeitungs-geschwindigkeit sorgen, ist von der jeweiligen Datenbasis und ihrer Konfiguration abhängig. Dieses Faktum ist für die Konzeption des gesamten Multi-Tier-Systems wichtig, spielt aber für die konkrete Implementierung des WMS keine Rolle.

Somit bleibt nur die Möglichkeit der Festlegung eines Gültigkeitsintervalls, in dem die Daten gecacht werden können. Nach Ablauf dieses Intervalls müssen die Daten aus den Datenbasen neu abgefragt werden. Um eine höchstmögliche Granularität zur Bestimmung dieses Intervalls zu erreichen, sollte es für jeden Layer separat festgelegt werden können, da bestimmte Daten wie beispielsweise die eines Straßenplans ein sehr großes Intervall ermöglichen und somit lange nicht neu abgefragt werden müssen. Andere Daten wie die der Staus auf diesen Straßen können sich aber minütlich ändern.

---

[40] So sind beispielsweise unter den gängigen Linux-Versionen global nur 1024 Threads bzw. durch das 1:1 – Modell Prozesse möglich [vgl. Heiss (2003)]

Die Transformationsprozesskette eines Web Map Services bietet zwei Stellen an denen eine Cachingkomponente realisiert werden kann; entweder auf der Ebene der einzelnen Layer oder auf der Ebene der gesamten Karte.

Sollten die generierten Display Elements der einzelnen Layerkomponenten in Abhängigkeit ihrer Parameter gecacht werden, hätte das den Vorteil, von der Kombination der Layer und ihrer Reihenfolge in der angefragten Karte unabhängig zu sein. Allerdings hätte diese Art des Cachings zur Folge, dass eine eigene Cachingkomponente implementiert werden muss und die Display Elements in einem „cachebaren“, serialisierten Format generiert werden müssten. Darüber hinaus sollte auch weiterhin ein Caching der resultierenden Karte erfolgen.

Stattdessen sollte nur die gesamte Karte in Form der HTTP-Response implizit von einem vorgelagerten Web-Proxy-Server gecacht werden. Das `Last-Modified`- und das `Expires`-Datum[41] des HTTP-Headers sollen sich dabei aus dem kleinsten Gültigkeitsintervall der angefragten Layer errechnen. Dieses Verfahren hat mehrere Vorteile. Zum einen kann dabei auf bestehende Implementierungen eines solchen Proxy-Servers zurückgegriffen werden, die meist in bestehende Web-Server integriert sind, was als weiteren Vorteil zur Folge hat, dass der Kommunikationsaufwand zwischen dem Web-Server und WMS-Application-Server auf die Requests reduziert wird, die eine Neuberechnung, also die Selektion und die Serialisierung, der Karte verlangen. Weiterhin ist so die Möglichkeit zum clientseitigen Caching gegeben.

Die Chance bei dieser Form des Cachings ein Cache-Hit zu erzielen, also Daten abzufragen, die vormals erzeugt wurden und sich im Cache befinden, hängt unter Vernachlässigung der Gültigkeit maßgeblich von der Variabilität aller Anfrage-URL, also von deren Parametern in jeweiliger Ausprägung und Reihenfolge, ab und ist somit rein clientseitig bestimmt. Der Server hat nur die Möglichkeit, mittels verschiedener Verdrängungsalgorithmen die Größe des Caches zu überwachen und häufig angefragte URL zu bevorzugen.

Zur Minimierung der unterschiedlichen Anfragen können clientseitig verschiedene Methoden eingesetzt werden, die gleichzeitig ein akzeptables Maß an Flexibilität sicherstellen oder dieses sogar steigern können:

[41] The Internet Society (1999)

- Karten sollten soweit möglich nur einen Layer darstellen. Das Ausgabeformat sollte dabei Transparenz unterstützen (PNG, GIF, SVG), so dass die einzelnen Ein-Layer-Karten selbst unter HTML mittels der Layer-Technologie von CSS 2 zu mehrschichtigen Karten überlagert werden können. Mittels JavaScript ist dabei sogar eine Änderung der Darstellungsreihenfolge möglich.
- Die gesamte räumliche Ausdehnung eines Layers sollte in kleinere Kacheln (engl. Tiles) gerastert werden, deren räumliche Begrenzung die BBOX-Parameter der einzelnen Requests bestimmen. So kann der rasterbasierte Client zwar nicht jegliche Skalierung und den Darstellungsausschnitt der Karten frei bestimmen, aber die Anzahl der unterschiedlichen Requests an den Server wird stark reduziert. Darüber hinaus müssen clientseitig beim Schwenken (pan) nur die Tiles nachgeladen werden, die sich noch nicht im Cache der Applikation oder des Browsers befinden.
- Die Auflösung sollte nur stufenweise entsprechend der nutzbaren Bandbreite und der Größe des Displays auswählbar sein.

Inwieweit unterschiedliche Styles und räumliche Bezugssysteme angeboten werden, richtet sich stark nach der Ausrichtung des Systems und kann nur schwer verallgemeinert werden. Allerdings gilt auch hierbei, dass eine geringere Auswahl den Rechenaufwand des WMS und den Speicherverbrauch des Caches vermindert.

Bei der Realisierung von Caches wird im Allgemeinen zwischen persistenten und transienten, also rein speicherbasierten, Formen unterschieden. Allerdings sind auch Hybridformen denkbar. Die Art und Weise der Realisierung kann je nach verwendeter Hardware gerade im Bereich der Festplatten einen starken Einfluss auf die Performance des Caches und somit auf die Latenzzeit einer Anfrage haben.

3.2.2.5 Komprimierung und Generalisierung

Die Größe der Daten spielt sowohl bei der Verarbeitungsgeschwindigkeit als auch beim Transport dieser Daten eine entscheidende Rolle. Zur Verringerung der Datenmengen können sowohl verlustlose als auch verlustbehaftete Komprimierungsverfahren eingesetzt werden. Außerdem sind bei geographischen Daten vier Betrachtungsebenen zur Komprimierung denkbar; zum einen können sich die Verfahren auf das Koordinatensystem einer Geometrie und dessen Genauigkeit beziehen, dann auf die Anzahl der Koordinaten und deren Verbindungsart zur Repräsentation dieser

Geometrie, weiterhin auf die Ausprägung der features und abschließend auf das Datenformat der Karte.

An dem bereits erwähnten Beispiel der Straßenkarten aus den Flurstücken des Liegenschaftskatasters sollen die verschiedenen Verfahren illustriert werden.

3.2.2.5.1 **Komprimierung der Koordinaten**

Die geographischen Informationen des Liegenschaftskatasters sind im *UTM-Koordinaten-System (Universales Transversales Mercator Koordinatensystem)*, also in einem zylindrisch projizierten, kartesischen Koordinatensystem abgelegt und die einzelnen Koordinaten sind im Double-Format in der Datenbank gespeichert. Die Maßeinheit dieses Koordinatensystems wird in Metern angeben, womit die Koordinaten maximal acht Vorkommastellen haben. Die räumliche Begrenzung des relevanten Bereiches misst aber nur 6 km in der Breite und 8 km in der Höhe. Die Normalisierung der Koordinaten bzgl. des minimalen x-Wertes bzw. durch die Negation der y-Achse des maximalen y-Wertes ergibt in der alpha-numerischen Repräsentation der Koordinaten im SVG-Format eine Reduktion von bis zu vier Vorkommastellen. Da in Deutschland die Koordinaten nur siebenstellig sind, reduzieren sie sich nur um drei Stellen. Die der Normalisierung zugrunde liegende BoundingBox kann als Attribut der SVG-Karte abgelegt werden, um die Originalkoordinaten wiederherstellen zu können.

Durch eine zusätzliche zentimetergenaue Rundung der Koordinaten, die zwar eine verlustbehaftete aber in diesem Bereich hinreichend genaue Komprimierung darstellt, wird die ursprüngliche Datenmenge der SVG-Karte von ca. 1,7 MB auf 0,7 MB reduziert und entspricht damit ziemlich genau der Größe der WKBGeometry-Repräsentation der Daten. Dieses nur aus einer Subtraktion und der anschließenden Rundung bestehende und dadurch performante Verfahren führt zu einer deutlichen Geschwindigkeitssteigerung sowohl beim Transport der Karte als auch beim Parsen der alpha-numerischen Repräsentation der Koordinaten durch die Renderkomponente. Der Rundungs- oder auch Genauigkeitsfaktor sollte vom jeweiligen Verwendungszweck der Daten abhängig sein und somit für jeden Layer einzeln gesetzt werden können.

Von einer weiteren Komprimierung durch eine unter SVG mögliche relative Adressierung der Koordinaten eines Pfad-Ausdruckes wird Abstand genommen, da es durch die Rundung zu einer Summierung der Fehler kommen würde. Allerdings kann durch eine SVG-eigene Skalierung um den Genauigkeitsfaktor und die Multiplikation der Koordinaten mit dem reziproken Wert dieses Skalierungsfaktors für eine rein

ganzzahlige Repräsentation der Koordinaten gesorgt werden, was einerseits den Dezimalstellenpunkt verhindert und darüber hinaus, je nach Wertebereich der schon normalisierten Koordinaten und abhängig vom Genauigkeitsfaktor, die führenden oder „endenden“ Nullen der Koordinaten vermeiden kann. Dieses Verfahren verlangt aber eine zusätzliche reziproke Skalierung der verwendeten Signaturen und wird somit vorerst nicht weiter verfolgt, um eine strikte Trennung von Daten und Layout zu gewährleisten.

3.2.2.5.2 **Generalisierung**

Eine weitere allerdings nicht verlustlose Reduktion der Datenmenge kann durch Generalisierung erfolgen. Dabei werden zwei Betrachtungsebenen unterschieden. Die eine Ebene bezieht sich auf die Geometrie eines features, die andere betrachtet die topologischen Beziehungen mehrerer features.

Durch so genannte Lienenglättung kann die Anzahl der Koordinaten einer Geometrie stark vermindert werden. Meist werden dazu approximative Verfahren eingesetzt, die entweder global den gesamten Pfad oder nur dessen einzelne Sektion betrachten. Die Approximation erfolgt dabei durch komplexere mathematische Funktionen wie Polynome oder auch durch einfache lineare Regression. Der *Douglas/Peuker Algorithmus* ist ein besonders effizientes und genaues Verfahren zur linearen Regression und kann somit zur Lienenglättung von Simple Features angewendet werden. [42]

Andere Verfahren, die einen Linienzug durch Bezierkurven kubisch oder bi-kubisch approximieren, können zwar mittels der `path`-Elemente von SVG dargestellt werden, müssen aber zur Laufzeit durchgeführt werden, da eine persistente Datenhaltung dieser approximierten Formen mittels Simple Features nicht möglich ist. Insofern ist eine Behandlung dieser Formen und Verfahren im Rahmen dieser Diplomarbeit nicht vorgesehen. Allerdings sollte eine dahingehende Erweiterung des WMS bzgl. der Verarbeitung von solchen Formen möglich sein, da sowohl GML3 als auch beispielsweise die *Spatial Extension* des DBMS der Firma Oracle[43] nicht-linearinterpolierte Formen behandeln.

---

[42] vgl. Taylor (1995)

[43] Oracle (2002)

Da nur die Datenbasen über explizite Cachingverfahren verfügen können, sollte ihnen eine rechnenaufwändige Generalisierung oder Transformation[44] zur Laufzeit vorbehalten bleiben; sei es zur Darstellung eines linear approximierten und damit Simple Feature-konformen Kreises, oder auch umgekehrt zur liniengeglätteten Darstellung einer ursprünglich Simple Feature-konformen Geometrie mittels polynomineller Approximation.

Die zweite Ebene der Generalisierung bezieht sich auf die Topologie der features. Wie bereits beschrieben sind die Straßen des Beispiels aus einzelnen Flurstücken zusammengesetzt. Somit sind die innen liegenden Kanten der Flurstücke einer Straße zur Repräsentation der Straße überflüssig. Eine Verschmelzung der einzelnen äußeren Ringe der Straßenflurstückspolygone zu einem einzigen Straßenpolygon führt je nach Struktur zu einer starken Verminderung der benötigten Datenmenge zur Repräsentation einer Straße. Dieser aufwändige und feature-erzeugende Generalisierungsprozess sollte allerdings nicht oder nur von einer dahingehend optimierten Datenbasis zur Laufzeit durchgeführt werden. Er ist aber ein wichtiges bzw. notwendiges Hilfsmittel zur abstrahierten Darstellung von detaillierten und kleinräumigen features in großflächigen Bereichen.[45]

3.2.2.5.3 **Format-Kompression**

Die abschließende Ebene der Komprimierung bezieht sich direkt auf das Kartenformat und ist somit von einer speziell geographischen Betrachtung unabhängig.

Für das SVG-Format spezifiziert das W3C die Kompression mit dem *gzip*-Verfahren[46], womit Kompressionsraten von 80 – 85 Prozent erzielt werden[47]. So lässt sich die bereits optimierte Straßenkarte nochmals auf eine Größe von 0,23 MB reduzieren. Der beim gzip-Verfahren verwendete *deflate*-Algorithmus basiert hauptsächlich auf der Reduzierung von Redundanzen innerhalb des verarbeiteten Datenstroms und ist deshalb für das strukturierte und textbasierte XML-Format als solches gut geeignet. Die Pfad-Ausdrücke der Flurstückspolygone lassen sich damit allerdings schwerer komprimieren, so dass die Kompressionsrate der Straßenkarte mit 70% signifikant niedriger ausfällt als die der Referenzangaben des W3C. Allerdings lässt sich durch die Kombination der

---

[44] dazu gehört auch die Transformation der Koordinatensysteme

[45] vgl. oben und im Folgenden Peter; Weibel (1999) und Cecconi; Galanda (2002)

[46] The Internet Society (1996)

[47] W3C (2003) b S. /minimize.html

beschriebenen bis auf die Rundung verlustlosen Verfahren insgesamt für diese durchaus repräsentativen Daten eine Kompressionsrate von 87% erzielen.

Zur Komprimierung der Rasterbilder stehen vornehmlich zwei Formate zur Auswahl; das verlustfreie PNG-Format auf der einen und das verlustbehaftete JPEG-Format auf der anderen Seite.

Das PNG-Format[48] zeichnet sich hauptsächlich durch die Möglichkeit für transparente Hintergründe und einer Indizierung der Farbpalette aus. Somit sind Bittiefen von 2, 4, 8, 16 und 32 Bit pro Pixel möglich, was gerade bei Karten, die nur aus einem Layer und möglichst wenig Farben bestehen, wie zum Beispiel bei der Straßenkarte mit nur einer Füll- und Lienenfarbe, sehr starke Komprimierungen zulässt. So ist auch die Straßenkarte mit einer Genauigkeit von einem Meter pro Pixel, also einer Auflösung von 8000 x 6000 Pixel, auch nur 0,7 MB groß. Diese Genauigkeit ist allerdings um den Faktor 10 schlechter als die der ursprünglichen SVG-Koordinaten. Da der Speicherverbrauch zum Rendern linear mit der Auflösung korreliert und schon für die Erzeugung dieser Auflösung bei dem entwickelten System ca. 480 MB Hauptspeicher benötigt werden, wird deutlich, wo die Grenzen von rasterbasierten Karten liegen. Eine Renderung von geringeren Auflösungen, die entweder mit einer weiteren Verringerung der Genauigkeit oder mit der Verkleinerung des Kartenausschnitts verbunden ist, ist allerdings ohne weiteres möglich. So ist sowohl eine clientseitige Renderung, die mit den Vorzügen der direkten Skalierungs- und Ausschnittsänderung einhergeht, bei den momentan maximalen Displaygrößen von 1600 x 1200 Pixel mit unter 20 MB Hauptspeicherverbrauch[49] zu realisieren, als auch eine serverseitige Renderung, die bei dem angeführten Beispiel mit einer Auflösung von 800 x 600 Pixel nur noch ca. 5 MB Arbeitsspeicher verbraucht. Die resultierende PNG-Datei hat bei dieser Auflösung und unter Beibehaltung aller weiteren Parameter nur noch eine Größe von 38 KB und ist damit deutlich kleiner als die mit den vorgestellten, quasi verlustlosen Verfahren komprimierten SVG-Daten. Die Genauigkeit der PNG-Datei liegt mit 10 Metern pro Pixel auch weit niedriger.

Mit Hilfe dieser serverseitigen Renderung der einzelnen Ein-Layer-Karten kann somit aber auch eine rasterbasierte Generalisierung zur Darstellung großflächiger Bereiche durchgeführt werden, die in dieser Form mit SVG nicht möglich ist. Durch eine quasi

[48] W3C (1996)

[49] Bei Übertragung der severseitigen Verhältnisse auf den Client

kaskadierende Anbindung des WMS an sich selbst kann eine Kombination von direkten SVG-Simple Feature-Daten und mittels Rasterung generalisierten Simple Feature-Daten ermöglicht werden. Die Kombinationstechnik kann gerade bei medium clients zu weiteren Geschwindigkeitssteigerungen führen, die durch die erzielte Datenmengenreduktion erreicht werden, was aber auch mit einem Informationsverlust über die Identität der einzelnen features einhergeht.

Das JPEG-Format ist vornehmlich zur Komprimierung von deckenden Karten vorgesehen, da der Hindergrund nicht als transparent markiert werden kann. Darüber hinaus können farbige Bilder im JPEG-Format nur mit einer Farbtiefe von 24 Bit pro Pixel dargestellt werden, was bei höchster Qualität analog zum verlustfreien PNG-Format in der 800 x 600 Pixel-Auflösung des behandelten Beispiels zu einer mehr als fünfmal größeren Datei führt. Eingesetzt werden sollte das JPEG-Format also schon deshalb hauptsächlich zur Darstellung vielfarbiger Karten, also Karten, die multi-spektrale Rasterdaten wie beispielsweise Luftbilder enthalten. Dabei kommen auch die großen Vorzüge des JPEG-Formats, die sich durch die flexible Wahl der Ausgabequalität und des Interpolationsalgorithmus ergeben, zur Geltung. Die Interpolation eignet sich prinzipiell besser für kontinuierliche Daten, da das menschliche Auge zwar auf Mustererkennung unter widrigen Bedingungen ausgelegt ist, aber diskrete Daten mit starkem Kontrast zum Hintergrund zu sehr „verschwimmen“ können. Nichtsdestotrotz sind die Dateigrößen des Beispiels in den noch akzeptablen Qualitäten 0,9; 0,6 und 0,3 mit entsprechend 64 KB; 27 KB und 13 KB zum Teil geringer als die des entsprechenden PNG-Formats. So kann die angesprochene Kombinationstechnik auch und unter Umständen sogar besser mit dem JPEG-Format angewendet werden, allerdings mit der zusätzlichen Einschränkung, dass ein entsprechender JPEG-Layer immer deckend ist.

3.2.3 **Sicherheit**

Das Thema Sicherheit wird in der WMS 1.1.1 Spezifikation nur indirekt angesprochen. Es ist zwar in dem Capabilities-Dokument eines WMS die Angabe von Gebühren zur Nutzung des WMS vorgesehen, aber in welcher Form die Abrechnung dieser und einer dafür nötigen Authentifizierung des Benutzers erfolgen soll, wird nicht weiter behandelt. Des Weiteren ist ausschließlich die Nutzung des HTTP zur Kommunikation mit dem WMS festgelegt. Trotzdem ist für viele geographische Daten der Schutz vor dem Zugriff von nicht autorisierten Benutzern unerlässlich.

Die Vertraulichkeit und Integrität der Kommunikation kann mit Hilfe des *Secure Socket Layer (SSL) over HTTP (HTTPS)*, sofern es vom Client unterstützt wird, transparent für den WMS und den Benutzer durch den vorgelagerten Web-Server erfolgen. Die Verwendung dieses Protokolls ist vom Sicherheitsniveau des Gesamtsystems abhängig und geht mit erhöhtem Rechenaufwand für die Kodierung der Daten mittels der auf dem RSA-Verfahren basierenden Kryptographiemethode einher.

Für die Implementierung des WMS ist wesentlich entscheidender, ob die Daten auf Ebene des Services oder auf Ebene der einzelnen Layer geschützt werden sollen. Sollte nur der Service als solcher geschützt werden, ließe sich dieses durch einfache Konfiguration des vorgelagerten Web-Servers realisieren und müsste nicht weiter behandelt werden. Allerdings hat diese Methode erhebliche Nachteile, sofern für den Schutz der Daten ein ausgefeilteres, rollenbasiertes Rechtesystem genutzt werden soll, da bei einem solchen System für jede einzelne Rolle ein eigener Map Service für den Zugriff auf die jeweils autorisierten Daten eingerichtet werden müsste.

Stattdessen sollte es möglich sein, für jeden Layer einzeln konfigurieren zu können, für welche Rollen der Zugriff auf diesen Layer gestattet sein soll. Die Verwaltung des User/Rollen-Rechtesystems sollte dabei nicht durch den Web Map Service selbst erfolgen, sondern durch den umgebenden Application-Server zur Verfügung gestellt werden, ebenso wie die Authentifizierung des Benutzers. Die Aufgabe dieses um den Sicherheitsaspekt erweiterten WMS soll aus der Überprüfung der Rollen des anfragenden Users im Hinblick auf die zugriffsberechtigenden Rollen für jeden einzelnen der angefragten Layer bestehen. Der User sollte für den WMS transparent durch den Application-Server unter Verwendung eines geeigneten Sitzungsverfolgungsverfahrens ermittelt werden.

Welche Verfahren zur Sitzungsverfolgung verwendet werden können hängt stark von der Client-Applikation ab. Sollte sie innerhalb eines Browsers laufen, steht sowohl die Methode des URL-Rewritings als auch die des Sessioncookies zur Auswahl. Bei einer eigenständigen Applikation ist zur Unterstützung der Sicherheitsfunktionen neben der obligatorischen HTTP-GET-Methode sowohl der Anmeldevorgang als auch die Sitzungsverfolgung separat zu implementieren, was ein Grund für die Vernachlässigung des Sicherheitsaspektes seitens des OGC sein könnte. Prinzipiell können aber auch dabei beide Methoden umgesetzt werden.

Abschließend bleibt festzustellen, dass geschützte Daten nicht durch den vorgelagerten Web-Server gecacht werden dürfen und ein URL-Rewriting zur Sitzungsverfolgung auch ein Caching von öffentlich zugänglichen Daten verhindert. Der Sicherheitsaspekt ist also mit einem erheblichen Verwaltungs- und Rechenaufwand verbunden und eine Verwendung hängt maßgeblich von den Anforderungen und der Konzeption des konkreten Anwendungsszenarios ab.

3.3 Anforderungen an den Client

Der zu entwickelnde SVG-basierte Application Service, im folgenden Client genannt, soll im Wesentlichen zwei Aufgaben erfüllen.

Aus Sicht des Endanwenders soll der Client einen einfachen, intuitiven und interaktiven Zugang zu aufbereiteten, geographischen Karten ermöglichen. Die Anwendung sollte die Anforderungen der visuellen Datenexploration erfüllen, dem Benutzer eine Legende der angebotenen Informationen (Layer) bereitstellen und adaptives Zoomen ermöglichen. Weiterhin sollte der Client nur die geographischen Daten vom WMS abfragen, die gemäß dem Darstellungsbereich und dem Abstraktionsniveau erforderlich sind, um das Transportaufkommen weitestgehend zu minimieren und so die Ladezeiten der geforderten Daten deutlich zu verringern.

Aus Sicht des Kartenautors soll die Client-Applikation so strukturiert sein, dass sie die für die Umsetzung der Anforderungen nötigen Implementierungen vom Aufbau der eigentlichen Karte kapselt. Das gestattet zum einen die Wiederverwendbarkeit der Implementierungen und soll zum anderen dem Kartenautor eine standardisierte Umgebung zum eigenständigen Design der Karten bieten. Zur Gestaltung der Karten sollen vom Kartenautor sämtliche von SVG angebotenen Darstellungsmittel inklusive des Scriptings, zur Erzeugung von interaktiven Layern nutzbar sein.

Der zu entwickelnde medium client soll eine Basistechnologie zur Gestaltung von interaktiven, dynamischen Karten bilden.

3.3.1 **Paradigmen der visuellen Datenexploration**

> *„Dynamische und interaktive Karten eignen sich [in] besonders gut als visuelle Kommunikationsschnittstellen für Informationssysteme, da sie wie keine andere visuelle Metapher eine Reihe von grundlegenden Anforderungen an die Gestaltung von grafischen Nutzerschnittstellen erfüllen.“* [50]

Wichtige Designkriterien für graphische Nutzerschnittstellen oder GUI wurden von Ben Shneiderman beschrieben. Diese Kriterien werden heute vor allem zur Qualitätsprüfung (Usability) von Kommunikationsoberflächen benutzt. Gute GUI erlauben dem Benutzer, interaktive Karten mit geringem Lernaufwand und weitgehend ohne elektronische Hilfestellungen kognitiv zu erfassen und zu bedienen. Aus den Erfahrungen mit graphischen Oberflächen ergeben sich nach Shneiderman im Wesentlichen sechs Anforderungen an GUI, die eine visuelle Exploration des Informationsraums, bzw. im geographischen Fall der Karte, ermöglichen sollen:

Zur Phase der Interaktion mit der Karte soll dem User ein *Overview* über die gesamte Karte möglich sein. Zur Umsetzung der gleichzeitigen Anzeige von Überblick und Detail stehen zwei Methoden zur Auswahl. Zum einen kann durch einen Lupen- oder Fischaugeneffekt ein Teil der Karte unter Beibehaltung der Gesamtkarte lokal vergrößert werden. Realisiert werden soll diese Anforderung aber mit der zweiten Methode, die auf einer separaten Übersichtskarte basiert, die die gesamte Karte anzeigt und in der der jeweilig sichtbare Bildausschnitt durch ein Rechteck dargestellt wird.

Die Selektion des Detailbereichs soll mittels *Zoom*-Funktion erfolgen. Dabei wird ein bestimmter Interessenbereich der Detailkarte vergrößert. Die Selektion dieses Bereichs soll dabei sowohl durch Festlegung des neuen Mittelpunktes und entsprechendem Skalierungsfaktor als auch durch die direkte Auswahl mittels eines Auswahlrechtecks möglich sein. Ebenso sollen die Verkleinerung und die Änderung der Beobachtungsposition, also die Verschiebung des Darstellungsbereichs (pan) unter Beibehaltung der Skalierung, möglich sein.

Unter der *Filter*-Funktion versteht Shneiderman die Möglichkeit des Herausfilterns von uninteressanten Bereichen des Informationsraums. Die Karte als Informationsraum wird durch die enthaltenen Layer somit in Bereiche eingeteilt. Eine (De-)

[50] vgl. im Folgenden Dässler (2002)

Aktivierungsfunktion der einzelnen Layer entspricht daher der Funktionalität und stellt eine weitere Anforderung an den Client.

Details-on-Demand bedeutet dem User eine Funktionalität anzubieten, die die Anzeige von Detailinformationen eines selektierten Informationsobjektes gestattet. Auf den WMS bezogen bedeutet es die Umsetzung der GetFeatureInfo-Operation. Clientseitig kann diese Anforderung auf zwei sich nicht ausschließenden Wegen erfolgen. Dem Benutzer kann einerseits beim Überfahren des features mit der Maus eine kurze Beschreibung dessen in die Karte projiziert werden, andererseits können detaillierte Informationen bei direkter Selektion durch „Anklicken“ in einem weiteren Bereich (Popup) angezeigt werden. Beide Methoden sollen vom Client umsetzbar sein. Die eigentliche Umsetzung kann aber erst bei einer konkreten Karte erfolgen und ist somit vom Kartenautor abhängig.

Die von Shneiderman formulierte *Relate*-Funktion dient der Visualisierung der Informationsstruktur und bezieht sich so hauptsächlich auf die Topologie der angezeigten features. Aus dem Layer-orientierten Blickwickel soll es aber möglich sein, die Transparenz von flächigen Layern und die Darstellungsreihenfolge aller Layer ändern zu können, um so einen Einblick in die „versteckten“ Strukturen der Karte zu bekommen.

Mit Hilfe der *History*-Funktion soll es dem Endanwender möglich sein durch die Abfolge seiner Aktivitäten navigieren zu können. Die Rückverfolgung der zoom & pan Aktionen als weiteres Orientierungshilfsmittel steht dabei im Vordergrund.

Ergänzt werden sollen die Funktionalitäten um ein Werkzeug zur Distanzmessung, das einerseits der Bestimmung des jeweiligen Maßstabs und andererseits der Orientierung im Raum dient. Eine Druckfunktion und die Anzeige des ursprünglichen Kartenausschnitts sollen auch zum Funktionsumfang des Clients gehören.

3.3.2 **Legende und Bildschirmaufbau**

Jede Karte sollte über eine Legende verfügen, die die in der Karte enthaltenen Layer mit ihrem Titel und der verwendeten Signatur anzeigt. Über die reine Darstellung als Hilfsmittel zur Identifizierung und Zuordnung der Signaturen zu den entsprechenden Layern hinaus, soll die Legende zur Selektion der einzelnen Layer bei den Layer-spezifischen Funktionalitäten wie Änderung der Darstellungsreihenfolge, Transparenzänderung der flächigen Layer und (De-)Aktivierung der Layer dienen.

Weiterhin sollen die Einträge der Legende eine Abstraktion von den Named Layers des WMS darstellen und eine hierarchische und gruppierende Strukturierung der Layer einer Karte ermöglich, ähnlich zu der Organisation der Layer in dem Capablitities-Dokument eines Web Map Services.

Aus Ergonomiegründen sollten die Schaltflächen der Layer-spezifischen Funktionen in der Legende integriert sein. Die übrigen auf die gesamte Karte bezogenen Tools wie Zoom & Pan oder History-Back bzw. -Forward sollen in einer allgemeinen Toolbar zusammengefasst werden.

Um eine Maximierung des Anzeigebereichs zu erzielen, soll eine Karte im Vollbild des Browserfensters angezeigt werden. Aus diesem Grund ist die Integration der Toolbar, der Legende und der Übersichtskarte in die Detailkarte zwingend notwendig. Um dem Benutzer den gesamten Darstellungsbereich zur eingehenden Betrachtung der Detailkarte nicht zu verwehren, sollen sowohl die je nach Layeranzahl umfangreiche Legende als auch die Übersichtskarte versenkbar sein. Da das Browserfenster selbst in seiner Größe veränderbar ist, wird somit eine flexible benutzerseitige Einstellung der gesamten Darstellungsgröße und folglich auch der zu rendernden Auflösung, die mit dem Rechenaufwand und dem Speicherverbrauch korreliert, möglich.

3.3.3 **Adaptives Zoomen und dynamisches Nachladen**

Über die reine Funktionalität des ausschnittsweisen Vergrößerns hinaus, sollte der Client dem Kartenautor einen Mechanismus zur Gestaltung von adaptiven Karten zur Verfügung stellen, die dann dem User adaptives Zoomen ermöglichen.

Ein Zoomprozess wird als adaptiv bezeichnet, wenn die Informationsdichte der Karte sich dem Kartenmaßstab respektive dem Skalierungsfaktor anpasst, so dass die Karte bei jeglicher Skalierung eine konstante Informationsdichte enthält.[51]

Für jeden Layer der Karte soll ein Intervall festgelegt werden können, das den minimal- und den maximalmöglichen Skalierungsfaktor angibt, in dem dieser Layer sichtbar sein soll. Durch die Bereitstellung von mehreren Layern, die den jeweiligen Generalisierungsstufen ein und derselben feature-Zusammenstellung entsprechen, kann so durch die der Informationsdichte der jeweiligen Layer entsprechenden Einstellungen der Sichtbarkeitsbereiche quasi ein adaptiver Layer erzeugt werden. Der Client soll bei

[51] vgl. Brühlmeier (2002) S. 45

jeder Skalierungsänderung die skalierungsabhängige Sichtbarkeit jedes Layers prüfen und gegebenenfalls (de-)aktivieren.

Von einer maßstabsabhängigen Skalierung der Signaturen wird abgesehen, um die Verwendung von maßstabsgetreuen Signaturen nicht zu verhindern, wie beispielsweise bei Häusern, deren Außenwandstärken durch die Strichbreiten dargestellt werden.

Das Konzept des dynamischen Nachladens soll es gestatten, auch große Datenmengen in einer Karte darzustellen. Es sollen nur die Daten der Layer geladen werden, die sich noch nicht in der Karte befinden und aktuell sichtbar sind. Das heißt, dass der Nachladeprozess von der Sichtbarkeit der einzelnen Layer, demnach von der nutzerseitigen bzw. skalierungsabhängigen Sichtbarkeit und von dem gewählten Kartenausschnitt abhängig ist. Um eine einfache Berechnung der bereits geladenen Daten zu ermöglichen, sollen die Layer analog zu großen Rasterbildern gekachelt werden können. Dieses korrespondiert gemäß Abschnitt 3.2.2.4.2 darüber hinaus positiv mit dem Cache des Web-Servers, da durch die Rasterung die räumlichen Begrenzungen für alle Instanzen des Clients gleich festgesetzt werden. So wird die Gesamtperformance des Systems erhöht, da nur die Daten nachgeladen werden, die auch benötigt werden und zusätzlich leicht gecacht werden können.

4 Systementwurf und Implementierung

Über die inhaltlichen Anforderungen an das System hinaus, soll das System so plattformunabhängig sein, dass es sowohl auf allen gängigen Betriebssystemen und Hardwarearchitekturen lauffähig ist, als auch alle verwendeten Komponenten oder Softwareprodukte austauschbar sind, so dass neben kommerziellen Lösungen auch frei verfügbare oder sogar Open-Source Softwarekomponenten eingesetzt werden können.

4.1 Die Konzeption im Überblick

Die Architektur des Gesamtsystems baut auf vier Tiers auf. Jedem dieser Tiers werden klar abgegrenzte und spezifizierte Aufgaben zugeordnet. Im Vordergrund der Konzeption steht dabei die bestmögliche Verteilung dieser Aufgaben auf die jeweiligen Tiers. Zusätzlich sollen weitestgehend bestehende Standardkomponenten zur Erfüllung der verschiedenen Aufgaben verwendet werden. Die Eigenimplementierung soll sich dabei auf die Entwicklung des WMS und eines SVG-basierten Clients beschränken.

Der WMS dient zur Integration der in Abschnitt 3.1.3 beschriebenen Datenbasen. Die Aufgaben des WMS sollen sich dabei auf die Transformation aller gelieferten Daten in SVG-Elemente, den Aufbau eines SVG-Karten-Dokumentes, dessen eventuell geforderter Renderung und die Serialisierung zu einer HTTP-Response beschränken. Um die in Kapitel 3.2 aufgestellten Anforderungen sowohl an Modularisierung und Erweiterbarkeit als auch an Performance zu erfüllen, wird der Service auf Basis der *Java Servlet API*[52] realisiert. Andere Lösungen zur Erzeugung von dynamischen Webinhalten wie PHP oder ASP sind eher zur Gestaltung von dynamischen Webseiten gedacht und verfügen nicht über die nötige Komplexität zum modularen Aufbau eines WMS oder sind proprietäre Produkte, die momentan nur für bestimmte Plattformen ausgelegt sind, wie Microsofts .net-Framework mit der Programmiersprache C#. Konzeptuell ist C# sehr stark an Java angelehnt, verspricht allerdings Vorteile hinsichtlich Stabilität und Performance[53].

Als Servlet-Container wurde die Open-Source Referenzimplentierung *Tomcat* der *Apache Software Foundation* gewählt. Der Servlet-Container lässt sich direkt mittels verschiedener Protokolle an alle gängigen Web-Server wie den Lotus Domino Server von IBM, den Internet Information Service (IIS) von Microsoft, den Netscape

[52] Java Community Process (2003)

[53] vgl. Vasters et al. (2001) S. 34

Enterprise Server und den Open-Source Web Server Apache binden. So dient der Web-Server als Eintrittspunkt in das System und bildet einen Puffer zwischen den Clients und dem WMS. Er nimmt alle Anfragen der Clients entgegen und leitet nur diejenigen Anfragen an den WMS weiter, deren Antworten sich noch nicht oder nicht mehr im Cache des Web-Servers befinden.

Der Client selbst besteht aus einem interaktiven SVG-Dokument, das sowohl die vom Kartenautor entworfene Struktur, Interaktivität und Gestaltung der Karte als auch die Implementierungen zur Erfüllung der in Kapitel 3.3 beschriebenen Anforderungen enthält. Die gesamte Umsetzung sollte ausschließlich auf dem SVG-Standard basieren. Da allerdings bei jeder verfügbaren SVG-Umgebung verschiedene Einschränkungen bei der Erfüllung der Vorgaben des W3C, der Plattformunabhängigkeit und der Performance bestehen, wird der Client nur für die Umgebung des *SVG-Viewer-Plugins* von Adobe entwickelt. Dieses Plugin hat die Vorteile, dass es auf der einen Seite für die meisten gängigen Betriebssysteme frei verfügbar ist und auf der anderen Seite die fast vollständige Umsetzung des SVG 1.1 Standards bietet. Des Weiteren ist es nativ in C++ implementiert, was im Gegensatz zu Java-Lösungen für eine hohe Renderleistung sorgt.

Ebenso wie beim Client wird das serverseitige Rendern nicht von einer Eigenimplementierung übernommen. Stattdessen wird auf das bestehende SVG-Framework *Batik* der Apache Software Foundation zurückgegriffen. Ebenso wie alle anderen Komponenten der Apache Software Foundation ist auch diese Open-Source. Die Batik-Komponente basiert gleich dem Servlet-Container auf der Programmiersprache Java und überzeugt durch die, vom Aspekt der Animation abgesehen, vollständige Umsetzung des SVG-Standards, wodurch sich diese Komponente gerade für die Renderung der SVG-Dokumente eignet.

Der Transport der SVG-Elemente innerhalb des WMS von den Layerkomponenten zu den Serialisierungskomponenten erfolgt mit Hilfe des *SAX*-Protokolls, was eine standardisierte und performante Verarbeitung dieser Elemente zulässt. Zur reinen Serialisierung der SAX-Ereignisse in eine alpha-numerische XML-Repräsentation zum Transport über das HTTP wird der *Xalan-Transformer* der Apache Software Foundation eingesetzt.

Der Zugriff auf die in Abschnitt 3.1.3 festgelegten Datenbasen wird durch die Layerkomponenten gekapselt, die die verschiedenen Daten in eine SVG-konforme Repräsentation transformieren und diese mittels SAX-Events an die

Serialisierungskomponenten weiterleiten. Das WMS-Servlet hat die Anfragebearbeitung und die Auswahl bzw. Verbindung der jeweiligen Layerkomponenten mit der entsprechenden Serialisierungskomponente zur Aufgabe. Darüber hinaus werden die einzelnen Layer durch das WMS-Servlet zu einem konformen SVG-Dokument zusammengefügt und mit zusätzlichen Informationen wie externen Signaturen und solchen zur geographischen Einordnung versehen.

Abbildung 6 zeigt schematisch das Zusammenwirken der an der GetMap-Operation beteiligten Komponenten. Die Darstellung ist an die des Portrayal Models in Abbildung 1 und Abbildung 5 angelehnt.

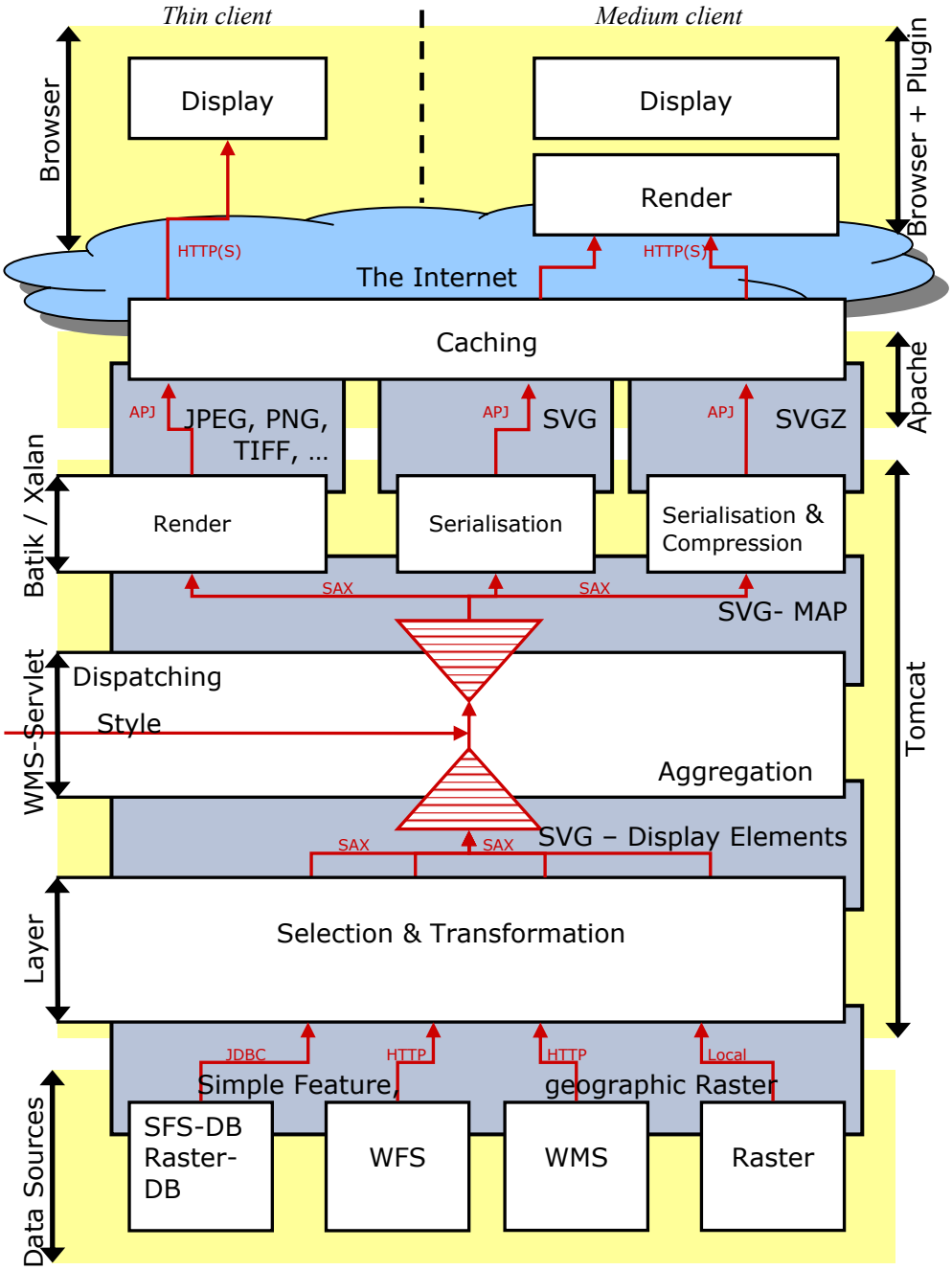

Abbildung 6 - Aufbau des Systems

4.2 Der Server – die Schnittstelle zum GIS

4.2.1 Der Servlet-Container - Tomcat

Der Apache Tomcat Server ist die offizielle Referenzimplementierung eines Servlet-Containers für die Java Servlet- und *Java Server Pages (JSP)* Technologien. Der Tomcat *„dient heute vielen J2EE-Plattformen wie JBoss, Borland Application Server (BAS), IBM WebSpehre und Sun J2EE 1.3.x als Referenzimplementierung oder als Basis der Web-Service-Initiative von Sun als Servlet-Container.“*[54] Die aktuelle Version 5.x implementiert die Unterstützung der Servlet API 2.4 und JSP API 2.0.

Die Java Servlet API kapselt die eigentliche HTTP-Anfrage und -Antwort und stellt dem Entwickler eine einheitliche Schnittstelle zur Bearbeitung dieser Anfrage zur Verfügung. Durch die Verwendung der Programmiersprache Java ist der Einsatz des Tomcat wie auch der Servlets in jeder Umgebung lauffähig, die von der *Java Virtual Machine (JVM)* unterstützt wird.

Darüber hinaus bietet der Tomcat die Unterstützung eines *Java Naming and Directory Interface (JNDI)*-`InitialContext`, der den namenbasierten Zugriff auf gemeinsame Ressourcen der Web-Application zulässt.

4.2.1.1 Sicherheit

Der Tomcat verfügt über ein ausgefeiltes Sicherheitsmanagement, das zum Tragen kommt, falls die Festlegung der Zugriffsberechtigungen nicht für den gesamten WMS mittels des vorgelagerten Web-Servers, sondern rollenbasiert auf der Ebene der einzelnen Layer erfolgt. Die dazu nötige Ermittlung der jeweiligen Rollen des anfragenden Benutzers kann beim Einsatz eines Web-Application-Servers standardisiert durch eine von der Servlet API spezifizierten Methode erfolgen.

Für die vorangehend nötige Authentifizierung des Benutzers bietet der Tomcat in der momentanen Version drei unterschiedliche Methoden an. Die einfachste Art ist der vom HTTP standardisierte Basic Authentification-Mechanismus, des Weiteren sind sowohl Web-Formulare als auch Zertifikate nach dem *X.509*-Standard zur Authentifizierung nutzbar. Die Sitzungsverfolgung erfolgt beim Tomcat vorgabegemäß per Sessioncookie, der nach dem erstmaligen clientseitigen Zugriff permanent die jeweilige Session-Id im HTTP-Header überträgt. Nach erfolgreicher Anmeldung wird der Session-Id der

[54] Roßbach (2002) S.32

entsprechende Eintrag aus der Benutzerdatenbank zugeordnet, so dass einem Servlet die Rollen des jeweiligen Benutzers angegeben werden können.

Die Benutzerdatenbank, beim Tomcat Realm genannt, kann ein spezielles XML-Dokument oder eine bestimmte Regeln erfüllende SQL-Datenbank sein und ebenso durch einen *Lightweight Directory Access Protocol (LDAP)*-Directory-Server realisiert werden. Die Festlegung von sicherheitsrelevanten Bereichen im URL-Raum einer Web-Application kann beim Tomcat und gemäß der Servlet API in der web.xml-Datei durch entsprechende Muster festgelegt werden.

Wichtig für den um den Sicherheitsaspekt erweiterten WMS ist die Möglichkeit des durch die Servlet API standardisierten Zugriffs auf die Rollen des jeweiligen Benutzers. Welche Benutzerdatenbank verwendet wird und in welcher Form die Authentifizierung erfolgt, bzw. inwieweit die Integrität der Daten durch das HTTPS sichergestellt wird, wird auf diese Weise von der Implementierung des WMS gekapselt und ist für die spezifischen Anforderungen einer konkreten Systemumgebung zu konfigurieren bzw. separat zu implementieren.

4.2.1.2 Performance und Multithreading

Die Aufgabe des Servlet-Containers, beim Tomcat Catalina genannt, besteht in der Bereitstellung einer Ablaufumgebung, in der Servlets mit Anfragen versorgt werden. Der Container basiert auf einer Serverinstanz in einer JVM, die mehrere Services enthalten kann. Jeder Service wiederum kann aus mehreren Netzwerkprotokoll-endpunkten bestehen. Ein Service erzeugt mit Hilfe des jeweiligen Netzwerkprotokoll-endpunktes bei einer Anfrage ein einheitliches Request/Response-Paar, welches an das entsprechende Bearbeitungsservlet weitergeleitet wird.

Die interne Architektur baut dabei auf der Multithreading-Umgebung der JVM auf. Eine Anfrage wird dabei jeweils von einem Java-Thread verarbeitet. Die hohe Nebenläufigkeit des Containers ist so gewährleistet, was gerade bei einem Mehrprozessorsystem eine performante Bearbeitung von gleichzeitigen Anfragen ermöglicht. Dieses bedeutet allerdings auch, dass ein Servlet gleichzeitig von mehreren Threads aufgerufen werden kann, um die Bearbeitung einer spezifischen Anfrage vorzunehmen. Zur Gewährleistung der Threadsicherheit eines Servlets existieren verschiedene Möglichkeiten. Zum einen ist ein Servlet als solches threadsicher, wenn auf die Verwendung von unsicheren Instanzvariablen verzichtet wird. Sofern dieses nicht möglich ist, kann ein Servlet durch die Implementierung des `SingleThreadModel`-

Interfaces dem Servlet-Container den exklusiven Aufruf durch einen Thread vorschreiben. Beim Catalina-Container werden dazu mehrere Servlet-Instanzen auf einem Stack verwaltet. Die Entfernung und Zuordnung einer Instanz zu einem Anfragebearbeitungsthread erfolgt dann jeweils exklusiv.

Das Konzept der Java-Threads baut allgemein auf leichtgewichtigen User-Level-Threads auf, welche durch den schwergewichtigen JVM-Prozess verwaltet werden. Wie und inwieweit leichtgewichtige User-Level-Threads, also Threads die ohne einen Wechsel des Adressraums aktiviert werden können, implementiert werden, hängt stark vom Betriebssystem ab. Nichtsdestotrotz ist prinzipiell die Verwendung von User-Level-Threads dem Einsatz von prozessbasierten Techniken wie der des *Common Gateway Interface (CGI)* zur Generierung von dynamischen Webinhalten vorzuziehen.

4.2.1.3 Konnektivität

Der Tomcat kann über zwei verschiedene Protokolle angesprochen werden. Zu jedem dieser Protokolle existiert ein entsprechendes Verbindungselement.

Mit dem HTTP-Connector bietet der Tomcat die Funktionalitäten eines eigenständigen Web-Servers. Der HTTP-Connector unterstützt das HTTP/1.1-Protokoll, so dass der Tomcat mittels eines spezifisch festgelegten TCP-Ports direkt angesprochen werden kann. Weiterhin kann der Tomcat so mit einem HTTP-Proxy-Server verbunden werden[55].

Eine direkte Anbindung des Tomcats an einen vorgelagerten Web-Server bietet allerdings diverse Vorteile, so dass dafür das separate *Apache Jserv Protocol (AJP)*[56] entwickelt wurde, welches in Version 1.3 von allen gängigen Web-Servern mittels entsprechendem Konnektor unterstützt wird. Für die direkte Verbindung mit dem Web-Servers bestehen mehrere Gründe. Im Folgenden werden die für die Entwicklung des WMS relevanten näher dargelegt:[57]

- Die rechenaufwändige (De-)Kodierung des SSL kann für den Tomcat transparent durch den Web-Server erfolgen, was meist mit Geschwindigkeitsvorteilen einhergeht, da eine native Umsetzung des Ver- und Entschlüsselungsalgorithmus in der Regel performanter ist.

---

[55] vgl. Apache Jakarta Project (o. J.) a S. http.html

[56] Apache Jakarta Project (o. J.) b

[57] vgl. Apache Jakarta Project (o. J.) c

- Der Transport von statischen Inhalten, im Falle des SVG-basierten WMS hauptsächlich die raumbezogenen Rasterbilder, erfolgt durch den Web-Server schneller.
- Die Stabilität des Systems wird erhöht, da der native Web-Server über eine der verwendeten Plattform besser angepassten Socket-Behandlung und -Verwaltung verfügt als der Tomcat, der auf die Socket-Funktionalitäten der JVM zurückgreift. Die Fehleranfälligkeit des Systems wird auf diese Weise stark vermindert.
- Spezifische Funktionalitäten des Web-Servers können genutzt werden, wie zum Beispiel Caching, welches allerdings auch durch die Kopplung mittels HTTP-Proxy-Server möglich ist.
- Die Authentifizierung und Autorisierung der Benutzer kann transparent für den Tomcat bzw. WMS durch den Web-Server erfolgen.
- Der Schutz des Systems vor böswilligen Attacken wird vom vorgelagerten Web-Server übernommen. Der Tomcat ist so nicht direkt über das Internet und das TCP/IP-Protokoll erreichbar.
- Der Web-Server kann als Loadbalancer fungieren, der es ermöglicht, die Anfragelast auf ein Cluster von Tomcats respektive WMS zu verteilen und der somit außerdem den Aufbau einer redundanten, ausfallsicheren Architektur gestattet, um die Java-abhängigen Stabilitätsengpässe zu minimieren.

Das AJP basiert auf einer binären Modifizierung des HTTP/1.1[58]. Als Transport Layer, gemäß dem *ISO/OSI-Modell*, können allerdings im Gegensatz zum HTTP und abhängig von den entsprechenden Verbindungselementen des Web-Servers nicht nur das TCP/IP-Protokoll, sondern auch, je nach physischer Verteilung von Web-Server und Tomcat(-Cluster) und dem verwendeten Betriebssystem, Protokolle eingesetzt werden, die weitere Performancesteigerungen durch die Nutzung eines gemeinsamen Speichers oder spezieller Unix-Sockets versprechen. Das bei physisch verteilten Komponenten eingesetzte Verfahren wird auf der Basis von persistenten und in einem Pool gehaltenen TCP/IP-Sockets realisiert.

[58] Roßbach (2002) S. 132 und Apache Jakarta Project (o. J.) b

4.2.2 **Der Apache Web-Server**

Um den vollständigen Einsatz von Open-Source-Software zumindest serverseitig zu ermöglichen, wird in der prototypischen Implementierung die Verbindung des Tomcats mit dem Apache HTTP Server evaluiert.

Der Apache HTTP Server, im Folgenden kurz Apache genannt, war das Initialprojekt der Apache Software Foundation. Der Apache zeichnet sich vor allem durch Robustheit, Sicherheit und Effizienz aus, die sich in seiner großen Verbreitung widerspiegeln.[59]

Die Verbindung mit dem Tomcat erfolgt über das AJP durch den JK2-Konnektor[60]. Dieser Konnektor, der neben der Implementierung für den Apache auch über Versionen für die weiteren bereits genannten Web-Server verfügbar ist, gestattet nicht nur die Verbindung durch die verschiedenen Transport-Layer wie Speicherteilung, Unix-Socket, TCP/IP-Socket und nativer *Java Native Interface (JNI)*-Einbindung, sondern beinhaltet auch einen Loadbalancer, der die sitzungstreue Verteilung der Anfragen auf ein Cluster von Tomcats ohne den sonst nötigen Hardware-Loadbalancer ermöglicht.

Für das konzeptuell notwendige Caching der Anfrageergebnisse seitens des Web-Servers bietet der Apache sowohl ein speicherbasiertes als auch persistentes Verfahren an. Ein hybrides Verfahren lässt sich auf diese Weise nur durch Hardware-Festplatten-Puffer realisieren. Die Verweildauer der Daten im Cache wird durch die im HTTP-Header übermittelten `last-modified`- und `expires`-Felder bestimmt. Weiterhin lässt sich in der Konfiguration des Caches bestimmen, inwieweit beim Fehlen dieser Felder die Werte per Standardwert festlegt oder durch spezielle Algorithmen interpoliert werden sollen.

4.2.3 **Das SAX-Konzept**

Die Anforderungen an die Modularität und Erweiterbarkeit des WMS in besonderem Hinblick auf die Erweiterung der Layer- und Serialisierungskomponenten verlangen nach einem möglichst standardisierten Übergabemechanismus der Display Elements von den verschiedenen Layerkomponenten an die jeweilige Serialisierungskomponente.

Da zur Kodierung der Display Elements der SVG-Standard verwendet wird und SVG eine konkrete Implementierung der Metasprache XML ist, bieten sich drei verschiedene Methoden zur Realisierung des Übergabeprotokolls an:

---

[59] Apache (o. J.)

[60] Apache Jakarta Project (o. J.) b

Die von der übergeordneten Anfragebearbeitungskomponente und den jeweiligen Layern zu generierende SVG-Karte könnte nach dem *Pull*-Prinzip auf Basis eines alpha-numerischen Datenstroms von der jeweiligen Serialisierungskomponente angefordert werden. Eine strikte Umsetzung dieser Methode ist allerdings schwer zu realisieren. Eine Kombination der Pull-Methode mit der *Push*-Variante, bei der der Erzeuger die textuelle Repräsentation der SVG-Elemente in einen Puffer schreibt, aus dem die Serialisierungskomponente lesen kann, ist unter Java entweder umständlich und inperformant durch eine temporäre Datei, oder wesentlich eleganter durch so genannte Piped-Streams umzusetzen. Die Methode erfordert allerdings beim serverseitigen Rendern der SVG-Karte oder einer sonstigen Umwandlung ein Parsen des Datenstroms. Clientseitig ist dieses Parsen, also die syntaktische Analyse des Zeichenstroms, durch die Verwendung des HTTP zwangsläufig nötig.

Das Rendern der SVG-Karte erfordert bedingt durch die Möglichkeiten der Verknüpfung von verschiedenen SVG-Elementen, sei es durch das `use`-Element oder durch die Benutzung der CSS-Technologie, die Kenntnis der gesamten Struktur des SVG-Karten-Dokumentes. Zu einer speicherbasierten Beschreibung eines XML-Dokumentes wurde vom W3C das *Document Object Model (DOM)* entworfen. Demnach kann die SVG-Karte mittels des DOM komplett im Speicher aufgebaut, als Referenz an die Serialisierungskomponente übergeben und anschließend von dieser weiterverarbeitet werden. Diese speicherteilende Methode hat durchaus Geschwindigkeitsvorteile, wenn das SVG-Dokument von der Serialisierungs-komponente entweder gerendert oder mittels baumbasierten Techniken wie *Extensible Stylesheet Language Transformation (XSLT)* weiterverarbeitet werden soll.

Der Einsatz des DOM als speicherbasierter Übergabemechanismus ist aus zweierlei Gründen problematisch. Erstens ist der Aufbau einer auf dem DOM fußenden Repräsentation der SVG-Karte gerade unter Java ausgesprochen speicherintensiv, so beansprucht ein 100 KB großes Dokument wenigstens 1 MB Speicher[61]. Eine objektorientierte Repräsentation verlangt demnach einen um den Faktor 10 größeren Speicher als die alpha-numerische Repräsentation, was auf das Beispiel aus Abschnitt 3.2.2 bezogen einen Speicherverbrauch von ca. 17 MB pro Anfrage bedeutet. Zweitens führt der Aufbau des DOM bei Anfragen, die ausschließlich eine alpha-numerische Serialisierung der SVG-Karte verlangen, zum einen zu unnötigem Speicherverbrauch,

[61] Anderson (2000) S.199

der zur Serialisierung der Gesamtstruktur des Dokumentes nicht erforderlich ist und des weiteren zu einer Verzögerung der Beantwortung dieser Anfragen führt, da die rekursive Abarbeitung des XML-Baums zur Erzeugung des textuellen Ausgabestroms erst nach dessen vollständigem Aufbau erfolgen kann.

Einen Kompromiss zwischen diesen beiden Extremen, dem zeichenbasierten Streaming-Modell und dem speicherbasierten DOM, bietet der de facto Standard *Simple API for XML (SAX)*. Bei SAX, das ausschließlich für Java konzipiert wurde, handelt es sich um Public Domain Software, die allerdings von so gut wie allen namhaften Softwareherstellern unterstützt wird und mittlerweile direkt in die J2SE von Sun aufgenommen wurde. Darüber hinaus existieren einige Implementierungen, die das SAX-Konzept in andere Sprachen portieren[62].

Das SAX-Prinzip fußt auf dem Push-Modell, nach dem dem Empfänger das Auftreten eines SAX-Ereignisses vom Erzeuger mitgeteilt wird. Diese Ereignisse können unter anderem das Auftreten eines XML-Elementes mit seinen Attributen, der Text eines Elementes und das Ende eines Elementes sein. Die Übermittlung der Ereignisse, auch SAX-Events genannt, erfolgt direkt vom Erzeuger durch den Aufruf spezifischer Methoden des an ihn übergebenen Empfänger-Objektes. Das Empfänger-Objekt muss dazu entsprechende SAX-Interfaces implementieren.

Im Gegensatz zum DOM wird aber immer nur ein Ereignis an den Empfänger weitergereicht, so dass sich die Speicherbelastung auf dieses eine Ereignis beschränkt. Entgegen dem Streaming-Modell wird beim SAX durch den Methodenaufruf die syntaktische Identität der einzelnen XML-Einheiten, die sich im DOM wieder finden, übermittelt, so dass das sonst nötige Parsen entfällt und ein DOM wesentlich schneller aufgebaut werden kann.

Aus programmiertechnischer Sicht ist SAX wesentlich leichter und sicherer zu handhaben als die direkte Erzeugung der textuellen Repräsentation, da durch den Aufruf der spezifischen Methoden eine Kapselung von der XML-Syntax vorgenommen wird. Weiterhin ist aber die Einhaltung der Wohlgeformtheit von XML durch den Entwickler zu tragen.

Über den reinen Transport der einzelnen XML-Einheiten hinaus eignet sich SAX zur performanten Transformation oder Filterung eines XML-Dokumentes. So können durch

[62] Megginson (o. J.) S. /?selected=langs

den Einsatz dieser SAX-Filter und deren Hintereinanderreihung so genannte SAX-Pipelines konstruiert werden, die bei einem einmaligen Durchlauf das ursprüngliche XML-Dokument in ein anderes transformieren. Allerdings werden durch den einmaligen Durchlauf des Dokumentes die Transformationsmöglichkeiten ohne Aufbau eines DOM erheblich eingeschränkt, womit eine vollständige Umsetzung der XSL durch SAX kaum möglich ist. Nichtsdestotrotz eignet sich die SAX-Filter-Technologie hervorragend zur strukturähnlichen Transformation von XML-Dokumenten[63].

Innerhalb der Implementierung des WMS spielt SAX eine herausragende Rolle, da es sowohl als Übermittlungsmechanismus der SVG-Elemente an die Serialisierungskomponenten als auch zur Transformation von GML in SVG eingesetzt wird.

4.2.4 **Das WMS-Servlet**

Das WMS-Servlet (`com.gfs.gis.exmap.services.SVGWebMapService`) bildet den Einstiegspunkt für die Verarbeitung einer Web Map Service-Anfrage. Es nimmt die HTTP-GET-Anfrage entgegen, sorgt für eine von der Schreibweise unabhängige Aufbereitung der übergebenen Parameter, validiert durch die VERSION- und SERVICE-Parameter die Übereinkunft mit der WMS 1.1.1-Spezifikation, sorgt für eine einheitliche Ausnahmebehandlung und leitet gemäß dem REQUEST-Parameter die Anfragen an spezifische Methoden zur Anfragebehandlung der GetMap- und GetCapabilities-Operationen weiter. Bei nicht übereinstimmenden VERSION- und SERVICE-Parametern bzw. bei der Anfrage einer nicht unterstützten Operation wird direkt eine Ausnahmenverarbeitung veranlasst.

4.2.4.1 GetMap-Operation

Die `DoGetMap`-Methode verarbeitet die vom OGC spezifizierte GetMap-Operation. Sie stellt die in Abschnitt 3.1.2.2 beschriebene übergeordnete Anfragebearbeitungskomponente dar und sorgt für eine inhaltliche Aufbereitung der Anfrage-Parameter; so werden alle erforderlichen Parameter wie LAYERS, STYLES, WIDTH, HEIGHT, BBOX, SRS und FORMAT analysiert und zur Weiterverarbeitung transformiert.

4.2.4.1.1 **Die Serialisierungskomponenten**

In Abhängigkeit des Formats der resultierenden Karte wird eine entsprechende Serialisierungskomponente ermittelt. Die Erzeugung der Serialisierungskomponente wird von einer so genannten SAX-Serializer-Factory übernommen. Jede dieser

[63] vgl. Becker (2003) a und b

Factories implementiert das `com.gfs.gis.exmap.serialization.SAXSerializerFactory`-Interface, dessen einzige Methode `newSAXSerializer` die Erzeugung einer `com.gfs.gis.exmap.serialization.SAXSerializer`-Interface implementierenden Klasseninstanz festlegt[64]. Die einzelnen SAX-Serializer-Factories sind mit dem entsprechenden Ausgabe-MIME/TYPE der erzeugten `SAXSerializer` an den JNDI-Kontext der Web-Application gebunden. Das `SAXSerializer`-Interface, das die jeweiligen Serialisierungskomponenten implementieren müssen, erweitert die Schnittstellen `org.xml.sax.ContentHandler`, `org.xml.sax.DTDHandler` und `org.xml.sax.ext.LexicalHandler` der SAX-Spezifikation um Methoden zur Übergabe des Ausgabestroms und für die Serialisierung verwendbarer Parameter.

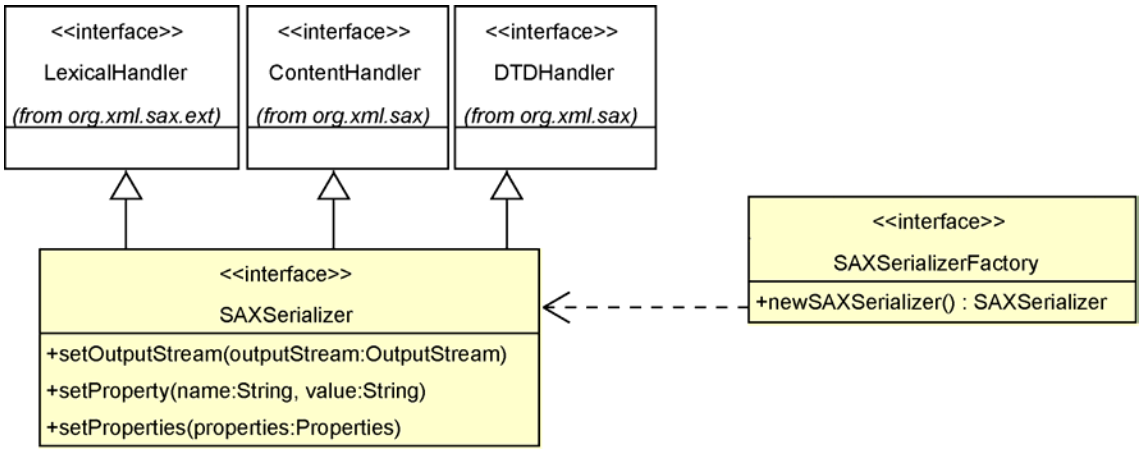

Abbildung 7 - Interfacediagramm der Serialisierungskomponenten

Die Kapselung der Serialisierungskomponenten durch das `SAXSerializer`-Interface und der standardisierte Zugriff auf diese Komponenten mittels JNDI und entsprechendem Factory-Interface ermöglichen die in Abschnitt 3.2.1 postulierte Modularisierung und Erweiterbarkeit der Serialisierungskomponenten des WMS. Das WMS-Servlet greift ohne Kenntnis der konkreten Implementierung nur durch den FORMAT-Parameter auf die Serialisierungs- bzw. Renderkomponenten zu, so dass weitere bislang nicht unterstützte Ausgabeformate wie GIF, Flash oder GEOTiff ohne Änderung des Servlets oder sonstiger Komponenten hinzugefügt werden können. Zudem gestattet die Verwendung des Factory-Konzeptes eine flexible und parametrisierbare Erzeugung der Serialisierungskomponenten.

Durch die Implementierung der SAX-Interfaces `ContentHandler`, `DTDHandler` und `LexicalHandler` bilden die `SAXSerializer` die Empfänger der die SVG-Karte

[64] Im Folgenden werden Serialisierungskomponenten und SAXSerializer synonym verwendet.

darstellenden SAX-Events. Die für die Renderung nötigen Attribute der `viewBox` und der Breite bzw. Höhe werden dem svg-Wurzelelement der Karte angefügt. Die Übergabe aller Anfrageparameter des WMS-Servlets an die Serialisierungskomponenten durch die `setProperties`-Methode räumt diesen eine autonome Verarbeitung der optionalen OGC-Parameter und der möglichen proprietären Parameter ein. Die Threadsicherheit der Serialisierungskomponenten wird durch eine synchronisierte Neu-Instanziierung einer solchen durch die entsprechende Factory sichergestellt. Somit muss bei der Implementierung eines `SAXSerializers` keine Rücksicht auf die Multithreading-Umgebung des Tomcats genommen werden.

4.2.4.1.2 **Die Layerkomponenten**

Das WMS-Servlet greift durch die einzelnen im LAYERS-Parameter angeführten Named Layer nach ähnlichen Prinzipien wie bei der Selektion der Serialiserungs-komponenten durch den FORMAT-Parameter auf die entsprechenden Layerkomponenten zu. Bezüglich der Implementierung unterscheiden sich die Mechanismen dabei allerdings maßgeblich, da für die Verwaltung der Layerkomponenten gemäß den Abschnitten 3.2.1 und 3.2.2.4.1 wesentlich höhere Anforderungen gelten. Zum einen sollen die Layerkomponenten zur Laufzeit flexibel erweitert und geändert werden können, was für die Serialisierungskomponente als vernachlässigbar erachtet wurde. Zum anderen soll die Handhabung von ausschließlich exklusiv nutzbaren Layerkomponenten in Ressourcen-Pools möglich sein, um die Latenzzeiten beim Zugriff auf eine konkrete Layerkomponenten-Instanz zu vermeiden, die bei deren Erzeugung beispielsweise durch den Aufbau einer Datenbankverbindung verursacht werden.

Analog zum `SAXSerializerFactory`-Interface wurde das `com.gfs.gis.exmap.-layer.LayerControler`-Interface entwickelt. Diese Schnittstelle kapselt die Verwaltung der Layerkomponente und stellt mit den Methoden `getLayer` und `release` den notwendigen threadsicheren Mechanismus zum Erhalt bzw. zur Rückgabe einer Layerkomponenten-Instanz, im folgenden kurz Layer genannt, zur Verfügung. Darüber hinaus ermöglicht der `LayerControler` den Zugriff auf die instanzunabhängigen Eigenschaften der Layerkomponente. Besonders sind dabei die Länge des Aktualisierungsintervalls und die zugriffsberechtigenden Rollen hervorzuheben. Somit ist die poolbasierte Verwaltung einer Layerkomponente grundsätzlich möglich, eine konkrete Implementierung aber nicht bei jeder Layerkomponentenart notwendig.

Da der vom Tomcat bereitgestellte `InitialContext`, über den der Zugriff auf die `SAXSerializerFactory` erfolgt, ausschließlich beim Starten des Servers initialisiert wird und zur Laufzeit dessen Änderung auf Basis des Konfigurationsdokumentes nicht möglich ist, wurde statt einer Bindung der `LayerControler` an diesen Kontext ein eigener Mechanismus entwickelt. Das zu diesem Zweck entworfene `com.gfs.gis.exmap.layer.LayerControlerHandler`-Interface bietet ähnlich zum JNDI-Kontext den namenbasierten Zugriff auf die an eine konkrete Implementierung dieser Schnittstelle gebundenen `LayerControler`. Weiterhin bietet diese Schnittstelle eine Methode zur Ermittlung aller Namen der gebundenen `LayerControler`. Der konkrete `LayerControlerHandler` wird wiederum an den JNDI-Kontext des Tomcats gebunden, so dass das WMS-Servlet mittels eines als Initialisierungsparameter abgelegten Namens darauf zugreifen kann und so vollkommen unabhängig von der konkreten Implementierung ist.

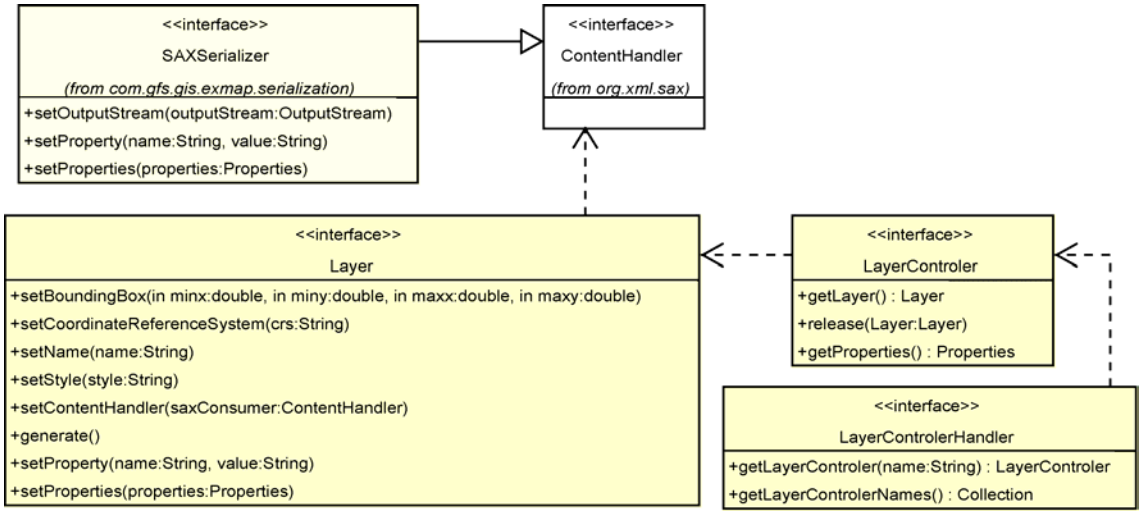

Abbildung 8 - Interfacediagramm der Layerkomponenten

Das `com.gfs.gis.exmap.layer.Layer`-Interface ist analog zum `SAXSerializer`-Interface aufgebaut und verfügt ebenso über dieselben Methoden zur Übergabe aller nicht standardmäßig erforderlichen Parameter. Alle Pflichtparameter einer GetMap-Anfrage, die einen Layer konkret betreffen wie die Werte des BBOX-Parameters, der SRS-Parameter und der Style des Layers werden aber explizit durch entsprechende Methoden gesetzt. Analog zum Ausgabestrom wird dem `Layer` der `ContentHandler`, also der ermittelte `SAXSerializer`, übergeben. Der Verarbeitungsprozess wird beim Layer nicht implizit durch den Erhalt von SAX-Ereignissen wie beim `SAXSerializer`, sondern explizit durch den Aufruf der `generate`-Methode vom WMS-Servlet ausgelöst.

4.2.4.1.3 **Die Struktur und der Aufbau der Karte**

Nach der Anfrageparameteranalyse und der Selektion der Serialisierungskomponente beginnt der Aufbau der SVG-Karte. Ebenso wie der Start des Dokumentes wird das `svg`-Wurzelelement vom WMS-Servlet direkt an den jeweiligen `SAXSerializer` übermittelt. Als Attribute werden dem Wurzelelement die im Dokument enthaltenen XML-Namensräume, die aus der Anfrage ermittelte Breite und Höhe der Karte in Pixel und der ursprüngliche BBOX-Parameter übergeben. Da die in Abschnitt 3.2.2.5.1 ausgearbeitete Komprimierung der Koordinatensysteme standardmäßig durchgeführt wird, errechnet sich das zur Festlegung des Koordinatensystems der Karte nötige `viewBox`-Attribut aus der Normalisierung der räumlichen Begrenzung und wird als solches ebenfalls an das Wurzelelement angefügt. Anschließend werden die nötigen Signaturen in Form einer so genannten *Extended-Style-Definition* an dieser Stelle in das Dokument eingearbeitet. Eine genaue Beschreibung der Funktionalität erfolgt im anschließenden Abschnitt. Nachfolgend werden die einzelnen Layer in ihrer vorgegebenen Reihenfolge in das Dokument eingefügt. Dazu werden nacheinander die `LayerControler` über den jeweiligen Namen beim konfigurierten `LayerControlerHandler` ermittelt. Sollte die optional einstellbare, rollenbasierte Berechtigungsüberprüfung positiv ausfallen, wird exklusiv eine `Layer`-Instanz an den aktuellen Bearbeitungsthread, also an das Servlet, gebunden. Der `Layer`-Instanz werden wiederum die standardmäßigen und optionalen Anfrageparameter und der ermittelte `SAXSerializer` übergeben. Nach der Generierung der Display Elements wird die `Layer`-Instanz an den `LayerControler` zurückgegeben.

Der Ablauf soll in dem nachstehenden Sequenzdiagramm nochmals verdeutlicht werden.

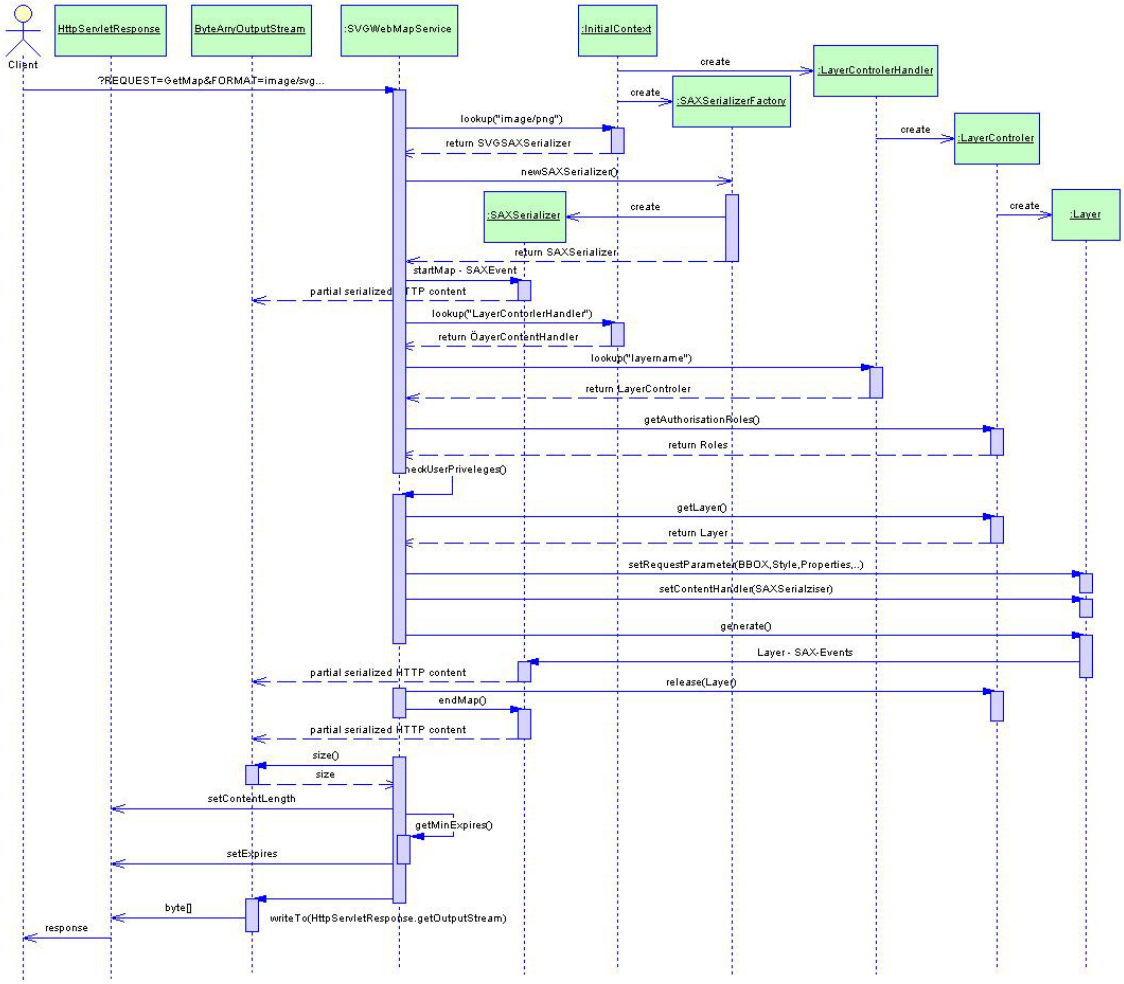

Abbildung 9 - Sequenzdiagramm der GetMap-Operation

4.2.4.1.4 Einbinden von externen Signaturen

Das Konzept der Extended-Style-Definition ähnelt dem Prinzip des Style Layer Discriptors eines erweiterten WMS. Es bezieht sich allerdings ausschließlich auf die Gestaltung der Layer und sieht keine Verknüpfung von externen, nicht direkt an den WMS gebundenen WFS vor.

Da der Gestaltungsspielraum der Layer in der Spezifikation des SLD zwar auf SVG basiert, allerdings nur eine Teilmenge der gesamten gestalterischen Möglichkeiten von SVG beinhaltet und durch die direkte Verarbeitung von SVG, also dessen Verwendung als Beschreibungssprache der Display Elements, der gesamte Sprachschatz von SVG verarbeitet werden kann, wurde das Konzept des SLD aufgegriffen aber entschieden modifiziert. Die Möglichkeit der freien Gestaltung der angefragten Karten in allen möglichen Formaten durch den Anfragesteller mittels übergebenen Verweises auf die Gestaltungsrichtlinien wurde vom SLD WMS übernommen. Die Beschreibungssprache weicht allerdings stark von der des SLD ab und basiert rein auf SVG.

Die Gestaltung eines Elementes[65] kann bei SVG durch drei verschiedene, nicht disjunkte Methoden erfolgen. Zum einen können alle Gestaltungsattribute als einzelne XML-Attribute an das jeweilige Element angefügt werden, weiterhin ist die Zusammenfassung der Gestaltungsattribute zu einem XML-Attribut (`style`) möglich. Darüber hinaus unterstützt SVG die CSS-Technologie, welche die externe Definition des `style`-Attributes und dessen Verknüpfung durch das so genannte `class`-Attribut ermöglicht. Generell wird die Gestaltung und Formatierung eines Elternelementes an seine Nachkommen so lange vererbt, bis es von einem dieser überschrieben wird.

Den Möglichkeiten von CSS und SVG bedient sich die Konzeption der Extended-Style-Definition. Bei einer Extended-Style-Definition-Datei handelt es sich um ein SVG-Fragment, dessen Wurzel das `defs`-Element bildet. Davon eingeschlossen werden ein Cascading Style Sheet mit der Definition des Styles eines Layers und eventueller weiterer Definitionen zur spezielleren Gestaltung eines Layers durch Füllmuster, Filtereffekte oder Symbole. Die Referenzierung eines Eintrags im CSS durch einen Layer bzw. dessen umschließendes Gruppierungselement erfolgt über das Setzen des `class`-Attributes, das dem jeweiligen Wert des aufgelösten STYLES-Parameters entspricht. Zur Definition von Symbolen als Darstellung von Punkten setzt sich der Verweis des `use`-Elementes aus dem Prefix „`symbol`“ und dem jeweiligen STYLE/`class`-Attribut zusammen.

Diese Technik gestattet eine vollkommen freie, ausschließlich durch die Möglichkeiten von SVG beschränkte Gestaltung der Karten durch den Anfragesteller. So sind auch Karten bzw. die gestalterische Ausarbeitung dieser Karten für spezielle Zielgruppen, wie beispielsweise kindgerechte Darstellung, ohne größeren Aufwand und mit der Unterstützung von SVG-verarbeitenden Grafikprogrammen wie Adobe Illustrator oder Corel Draw umzusetzen.

Sollte die Angabe des Verweises auf die Extended-Style-Definition in der Anfrage fehlen, wird der in der Servlet-Konfiguration vorhandene Standardverweis verwendet. Dabei erfolgt die Verknüpfung von CSS-Eintrag und Layer über den Namen des Layers. Die Zusammenfassung der Standarddarstellung einer Extended-Style-Definition-Datei bedingt jedoch bei einer (komprimierten) alpha-numerischen Serialisierung die unnötige Übermittlung der Darstellungsrichtlinien der in der jeweiligen Karte nicht angefragten Layer des WMS.

---

[65] Alle Elemente, die das Interface `styleable` implementieren.

Das Einfügen der Extended-Style-Definition in das SVG-Karten-Dokument wird durch die Verwendung eines SAX-Parsers mit vorgelagertem SAX-Filter zur Verhinderung der `startDocument`- und `endDocument`-Ereignisse realisiert. Dabei ist jede konforme, für den WMS per URL erreichbare Extended-Style-Definition-Datei verarbeitbar.

4.2.4.2 Die GetCapabilities-Operation

Die GetCapabilities-Operation hat die Aufgabe, die Metadaten des WMS in einer spezifischen, XML-konformen Repräsentation zu liefern. Da diese Metadaten für jeden Web Map Service von unterschiedlich hoher Bedeutung sind und viele Einträge Daten des jeweiligen Anwendungskontextes umfassen, wie zum Beispiel den Titel, die Beschreibung, etwaiger Schlüsselwörter, die entsprechenden Service-URL und die unterstützten Ausgabeformate bzw. die so genannten Vendor-specific-Parameter, die wiederum von den implementierten Serialisierungs- und Layerkomponenten abhängen, wird zumindest für die in Abschnitt 3.1.2.1 beschriebenen ersten beiden Teile des resultierenden Capabilities-Dokumentes von einer generischen Erzeugung abgesehen.

Stattdessen wird dem Administrator des WMS die Möglichkeit gegeben, die Eigenschaften dieses Services flexibel und den Anforderungen des Anwendungskontextes gemäß in einer entsprechenden XML-Datei zu editieren. Hinsichtlich der Anführung der einzelnen Named Layer des Services ist allerdings eine automatische, rudimentäre Erzeugung möglich, die ausschließlich die minimalen Daten wie Titel, Namen, Angaben der räumlichen Ausdehnung in Breiten- und Längengraden und im unterstützen Koordinatensystem umfasst. Detailliertere Angaben zu den Named Layern erfordern aber weiterhin die manuelle Pflege dieser und somit aller Daten. Eine automatische Generierung in Kombination mit einer manuellen Eingabe ist auch möglich, führt aber zu Redundanzen.

Realisiert wird die flexible Kombination aus manueller Eingabe und automatischer Generierung durch die Benutzung der XML-Processing-Instruction, die an geeigneter Stelle beispielsweise umschlossen von einem Wurzel-Layer-Element in das Capabilities-Dokument eingefügt wird. Der in der Konfiguration des WMS-Servlets angegebene Verweis auf ein solches Dokument wird aufgelöst und das entsprechende Dokument wird mit einem SAX-Parser mit vorgelagertem SAX-Filter (`com.gfs.gis.exmap.util.sax.GenericLayerCapabilitiesXMLFilterImpl`) verarbeitet. Der SAX-Filter leitet alle SAX-Ereignisse an den entsprechenden textuellen `SAXSerializer` weiter. Sollte allerdings ein SAX-`processingInstruction`-Ereignis mit

dem Namen „ParserBreak“ erfolgen, greift der SAX-Filter auf den bei der Initialisierung übergebenen LayerControlerHandler zu, ermittelt alle von ihm verwalteten Layer des WMS und löst die entsprechenden SAX-Ereignisse zur Generierung der Layer-Capabilities aus. Die jeweiligen Layer-spezifischen Daten ermittelt der SAX-Filter aus den instanzunabhängigen Eigenschaften des Layers, also durch die getProperties-Methode der LayerControler. Dazu müssen die entsprechenden Eigenschaften der Layer die benötigten Daten enthalten; ansonsten werden die Standardwerte verwendet.

4.2.4.3 Ausnahmebehandlung

Die gesamte Anfragebehandlung mit ihren nachgelagerten Komponenten verfügt über eine ausgefeilte Ausnahmebehandlung. Alle Methoden, die einen Fehler verursachen können, leiten diesen an die doGet-Methode des WMS-Servlets weiter.

Realisiert wird diese Ausnahmebehandlung wie unter Java vorgesehen durch das Exception-Konzept. Zur Unterstützung der vom OGC verabschiedeten Form der Ausnahmen, die einen speziellen Kodex und dessen Beschreibung enthalten, wurde eine eigene Exception-Klasse implementiert, um diesem Konzept Rechnung zu tragen.

Die in dieser Form des basic WMS realisierten Kodizes sind:

- InvalidFormat wird ausgelöst, wenn kein dem FORMAT-Parameter entsprechender SAXSerializer gefunden wird.
- InvalidSRS wird von den einzelnen Layerkomponenten ausgelöst und ist somit von deren Implementierung abhängig.
- LayerNotDefined wird vom LayerControlerHandler ausgelöst, sofern kein LayerControler an den übergebenen Layernamen gebunden ist.

Darüber hinaus sind weitere, eigene Kodizes eingeführt worden:

- OperationNotSupported wird produziert, wenn die Anfrage Parameter enthält, die als solche oder deren Werte bzw. eine Teilmenge der möglichen Werte nicht unterstützt werden.
- InvalidParameter gibt an, wenn ein standardmäßig benötigter, von den übrigen Kodizes nicht behandelter Parameter nicht vorhanden oder aber nicht behandelbar ist, zum Beispiel durch fehlgeschlagene Typkonvertierung.
- InternalError kapselt alle anderen nicht weiter spezifizierten Fehler.

Die im Falle eines Fehlers auftretenden Exceptions werden einer dafür konzipierten Methode übergeben. Diese setzt einerseits die eventuell bereits geschriebene Teilantwort zurück und erzeugt anderseits in Abhängigkeit vom EXCEPTIONS-Parameter (`application/vnd.ogc.se_xml`, `application/vnd.ogc.se_inimage`, `application/vnd.ogc.se_blank`) die Ausgabe der Ausnahme. Die Ausnahmebehandlungsmethode unterstützt dabei jeden der drei vom OGC verabschiedeten Ausgabemodi.

Der erste Modus sieht die Ausgabe des Ausnahmekodex und dessen Beschreibung in einem mittels veröffentlichter DTD[66] spezifiziertem XML-Dokument vor. Dementsprechend wird von der Ausnahmebehandlungsmethode der alpha-numerische `SAXSerializer` ausgewählt und das XML-Dokument durch Aufruf der nötigen SAX-Ereignisse erzeugt.

Weiterhin sind in der WMS-Spezifikation die optionalen Ausgaben in dem im FORMAT-Parameter vorgegeben Format vorgesehen. Dabei ist sowohl die Ausgabe des aufgetretenen Kodex als auch die Rückgabe eines leeren Bildes möglich. Zu diesem Zweck wird der dem Format entsprechende `SAXSerializer` gewählt. Sollte die Angabe des Kodex in der Grafik durch den Wert `application/vnd.ogc.se_inimage` des EXCEPTION-Parameters vorgegeben sein, wird dieser Kodex als Text in die resultierende SVG-Grafik eingefügt. Diese Fähigkeit ermöglicht einem Client, der das entsprechende Format als Rückgabe der `GetMap`-Operation erwartet, dem Benutzer das Auftreten des Fehlers im eigentlichen Anzeigebereich der Karten quasi indirekt mitzuteilen. Löst hingegen eine der optionalen Arten der Ausnahmeausgabe einen Fehler aus, da beispielsweise das geforderte Format nicht unterstützt wird, erfolgt die Ausgabe des Fehlers durch das standardmäßige XML-Dokument. Weiterhin ist zu administrativen Zwecken optional eine Archivierung der Fehler in einer Log-Datei möglich. Ein solcher Archiveintrag enthält nicht nur die Angabe des Kodex sondern auch Angaben über den eventuell auslösenden internen Systemfehler.

4.2.4.4 Caching

Ein Caching der erzeugten Daten durch einen vorgelagerten Web-Server verlangt meist die Angabe des letzten Modifikationszeitpunkts, des Zerfallszeitpunkts und der Größe

[66] OpenGIS Consortium (2002) d

des generierten Inhalts. Übergeben werden diese Angaben durch spezielle Felder im HTTP-Response-Header.

Da die dynamischen Daten aufgrund der fehlenden Kenntnis des Änderungszeitpunkts nur implizit gepuffert werden, wird zur Berechnung der beiden Zeitpunkte auf ein Gültigkeitsintervall zurückgegriffen. Der Wert des `last-modified`-Feldes errechnet sich aus der Differenz der aktuellen Systemzeit und dieser Zeit modulo der Länge des Gültigkeitsintervalls. Analog dazu wird der Wert des `expires`-Feldes aus der Summe des letzten Modifikationszeitpunkts und der Länge des Gültigkeitsintervalls berechnet. Dieses Verfahren führt bei einem einstündigen Gültigkeitsintervall zu einer Neuabfrage der Daten nach Ablauf jeder vollen Stunde.

Die Länge des Gültigkeitsintervalls wird bei der `GetMap`-Operation aus dem Minimum aller Intervalllängen der angefragten Layer errechnet. Sollte ein Layer eine nicht öffentliche Zugriffsberechtigung erfordern, wird statt der zum Caching nötigen Felder im HTTP-Header die private Behandlung der Daten vermerkt. Die Länge des Gültigkeitsintervalls für die Beantwortung der GetCapabilities-Anfragen und bei eventuell auftretenden Fehlern wird separat in den Initialisierungsparametern des WMS-Servlets abgelegt.

Die Ermittlung der Größe der erzeugten Daten erfordert eine servletseitige Pufferung dieser Daten, da die HTTP-Header-Felder vor den eigentlichen Daten in den Ausgabestrom geschrieben werden müssen. Zu diesem Zweck wird der Serialisierungskomponente statt des direkten `javax.servlet.ServeltOutputStream` der `javax.servlet.HttpServletResponse` ein `java.io.ByteArrayOutputStream` übergeben. Dieser Ausgabestrom ermöglicht die Ermittlung der in diesen Strom geschriebenen Bytes, deren Anzahl den Wert des `content-length`-HTTP-Header-Feldes ergibt.

Da dieses Konzept die Fortführung des stromartigen Ansatzes der Transformation von Simple Features in SVG-Display-Elements bis zum medium client verhindert, bei der Verarbeitung von zugriffsbeschränkten Daten unnötig ist und weiterhin nicht jeder Web-Server zum Caching der Daten die Angabe über deren Größe im HTTP-Header verlangt, sollte die servletseitige Pufferung in den Konfigurationsdaten des WMS-Servlets (de-)aktiviert werden können. Eine Implementierung dessen steht aber noch aus.

4.2.5 **Die Serialisierung**

Zur Serialisierung der SAX-Ereignisse wird auf zwei bereits bestehende Softwarekomponenten zurückgegriffen.

Der eingesetzte Xalan Transformer der Apache Software Foundation bietet mit seiner `org.apache.xalan.transformer.TransformerIdentityImpl`-Klasse die Basisklasse zur Implementierung der reinen alpha-numerischen Serialisierung der SAX-Ereignisse. Diese Klasse erfüllt bereits die zur Implementierung des `SAXSerializer`-Interfaces nötigen SAX-Interfaces `ContentHandler`, `DTDHandler` und `LexicalHandler`. Zur Implementierung des `SAXSerializer`-Interfaces wird diese Klasse von der Klasse `com.gfs.gis.exmap.serialization.impl.XMLSerializer` beerbt und um die notwendigen Methoden ergänzt. Die `setOutputStream`-Methode kapselt die Benutzung der `setResult`-Methode und sorgt für die Zuweisung eines SAX-`StreamResult`. Die Methoden `setProperty` und `setProperties` werden implementiert, erfüllen aber keine Funktion.

Zur Komprimierung des Ausgabestroms mit dem von SVG unterstützten gzip-Verfahren wurde ein weiterer `SAXSerializer` implementiert. Diese `com.gfs.gis.exmap.serialization.impl.XMLZSerializer` genannte Klasse erbt wiederum vom `XMLSerializer` und erzeugt beim Setzen des Ausgabestroms einen auf dem übergebenen Ausgabestrom basierenden `java.util.zip.GZIPOutputStream`. Wichtig bei dieser Implementierung ist das Abschließen des gzip-Stroms vor dessen Leerung und Übertragung. Der Abschluss des Stroms wird bei der Behandlung des SAX-`endDocument`- Ereignisses durchgeführt.

Das zum Rendern der SVG-Karten eingesetzte Batik-Framework der Apache Software Foundation stellt zur Umwandlung der SAX-Ereignisse in ein zum Rendern benötigtes DOM die Klasse `org.apache.batik.dom.svg.SAXSVGDocumentFactory` zur Verfügung. Diese Klasse implementiert analog zur Xalan-Klasse bereits die nötigen SAX-Interfaces und sorgt für den Aufbau des DOM. Beerbt wird diese Klasse von der Klasse `com.gfs.gis.exmap.serialization.impl.BatikImageSerializer`, die für die Implementierung des `SAXSerializer`-Interfaces und den eigentlichen Aufruf zum Rendern sorgt. Das DOM wird bei der Behandlung des SAX-`startDocument`-Ereignisses initialisiert und durch die Behandlung der SAX-Ereignisse durch die Oberklasse aufgebaut. Bei der Verarbeitung der `endDocument`-Methode wird das DOM zusammen mit dem Ausgabestrom an den eigentlichen, von Batik so genannten Transcoder übergeben. Der Transcoder wurde dem `BatikImageSerializer` von der entsprechenden

`SAXSerializerFactory` übergeben und basiert auf der Batik-Klasse `org.apache.batik.transcoder.image.ImageTranscoder`, welche das `org.apache.batik.transcoder.Transcoder`-Interface implementiert.

Die Klasse `ImageTranscoder` ist eine abstrakte Klasse, die die Umwandlung von SVG in ein `java.awt.image.BufferedImage` des AWT vornimmt. Das Schreiben dieses `BufferedImage`-Objektes im entsprechenden Format in den Ausgabestrom wird von den die `ImageTranscoder`-Klasse beerbenden Klassen übernommen. So ist die Unterstützung von einer Vielzahl von unterschiedlichen Formaten ohne großen Implementierungsaufwand denkbar. Realisiert sind seitens der Batik-Komponente bislang Image-Transcoder für die Formate JPEG, PNG und Tiff.

Die durch die Methoden `setProperty` und `setProperties` unterstützte Übergabe der Anfrageparameter gestattet das clientseitige Setzen von Render-Parametern. Diese Parameter werden von dem `BatikImageSerializer` analysiert und als so genannte Transformer-Hinweise an den konkreten `ImageTranscoder` weitergeben. Jeder der drei genannten `Transcoder` kann den Parameter DPI zur Angabe der Druckauflösung des gerenderten Bildes, den BGCOLOR-Parameter, der die Hintergrundfarbe angibt und auch als optionaler Parameter vom OGC spezifiziert wurde, und den LANGUAGE-Parameter verarbeiten. Der LANGUAGE-Parameter dient zur Festlegung der Sprache, falls ein `switch`-Element die Verarbeitung von mehreren Sprachen vorsieht. Die Angabe dieses Parameters ermöglicht zum Beispiel die Schreibweise von Ländernamen oder anderen Bezeichnungen, die direkt in der Karte angegeben sind, abhängig von der jeweiligen Sprache anzuzeigen. Voraussetzung dafür ist allerdings die Generierung eines solchen `switch`-Elementes durch die jeweiligen Layerkomponenten.

Weiterhin bietet der `Transcoder` zur Erzeugung des JPEG-Formats die Möglichkeit zur Angabe der verwendeten Qualität. Diese kann durch den QUALITY-Parameter gesetzt werden. Der PNG-Format-Transcoder verwendet die Parameter GAMMA, BPP und TRANSPARENT zur plattformunabhängigen Einstellung der Gamma-Korrektur, zur Indizierung und Ermittlung der Größe der Farbpalette und zur Festlegung eines nicht-transparenten Hintergrundes.

4.2.6 **Der Layer – die Schnittstelle zur Datenbasis**

Die Layer bilden das Herzstück bei der Verarbeitung von geographischen Informationen. Sie erfüllen die Aufgabe der Transformation der unterschiedlichen Daten in eine einheitliche, SVG-konforme Repräsentation und bilden so die

Schnittstellen zu den angebundenen Datenbasen und Geographischen Informationssystemen.

4.2.6.1 Von der Datasource zum LayerControler

Da die Konzeption des WMS auf den hohen Anforderungen an Performance und Skalierbarkeit beruht und die effektive Bereitstellung der geographischen Informationen hauptsächlich durch eine SFS-konforme Datenbank vorgenommen wird, verwaltet die Implementierung des in Abschnitt 4.2.4.1.2 spezifizierten `LayerControlers` gemäß Abschnitt 3.2.2.4.1 einen Pool von Layer-Instanzen.

Die Pool-Funktionalitäten selbst wurden dabei nicht eigenständig entwickelt. Stattdessen wird auf die im *Apache Jakarta Commons Project* entstandenen Implementierungen allgemeiner Java-Object-Pools zurückgegriffen. Drei verschiedene das Interface `org.apache.commons.pool.ObjectPool` implementierende Pools werden zur Verfügung gestellt.

- Die Pool-Implementierung der Klasse `org.apache.commons.pool.impl.-GenericObjectPool` sorgt ebenso wie alle weiteren Implementierungen für eine threadsichere Verwaltung von Objekten. Die Verwaltung umfasst eine flexible Konfiguration des Pools bezüglich Mindest- und Höchstzahl der enthaltenen `Layer`-Instanzen und deren Abbaugeschwindigkeit, benötigt aber zur Kontrolle der verwalteten Objekte einen eigenen Kontrollthread. Unter anderem nutzt auch die vom Tomcat zur Verfügung gestellte Implementierung einer `javax.sql.DataSource` zur Verwaltung der einzelnen Verbindungen diese Implementierung eines Pools.
- Die `org.apache.commons.pool.impl.StackObjectPool`-Klasse benutzt zur Verwaltung der eingelagerten Objekte einen limitierten Stack. Diese Klasse benötigt keinen eigenen Kontrollthread, sondern überprüft ausschließlich bei der Rückgabe von erzeugten Objekten die Obergrenze des Stacks. Diese Implementierung eignet sich besonders gut für solche Layer-Instanzen, die keine oder nur während des Generierungsprozesses Verbindungen zu anderen Prozessen aufbauen.
- Die `org.apache.commons.pool.impl.SoftReferenceObjectPool`-Klasse bedient sich zur Verwaltung der Objekte der Klasse `java.lang.ref.SoftReference`. Objekte dieser Klasse werden von der JVM separat behandelt. Abhängig vom verfügbaren Hauptspeicher berechnet die JVM die Lebensdauer der Instanzen der `SoftReference`-Klasse. Demnach ist für einen `SoftReferenceObjectPool`

lediglich eine initiale Größe anzugeben. Ein solcher Pool benötigt weder einen Kontrollthread noch eine Obergrenze. Durch die unkontrollierbare Lebensdauer und Anzahl der Instanzen sollte diese Pool-Implementierung allerdings mit Bedacht verwendet werden.

Die Verwaltung der einzelnen eingelagerten Objekte eines Pools wird von einer die Schnittstelle `org.apache.commons.pool.PoolableObjectFactory` implementierenden Klasse vorgenommen. Diese Klasse wird einem `ObjectPool` mitsamt seiner Konfiguration bei der Initialisierung übergeben. Die `PoolableObjectFactory` übernimmt die Erzeugung, die (De-)Aktivierung, die Validierung und die Finalisierung eines Objektes des Pools.

Implementiert wird diese `PoolableObjectFactory` ebenso wie das `LayerControler`-Interface von der Klasse `com.gfs.gis.exmap.layer.controler.pool.properties.-LayerControlerImpl`. Diese Klasse bildet die Schnittstelle zwischen der vorgegebenen Pool-Konstruktion und den Konzepten des WMS. Die Erzeugung der jeweiligen `Layer`-Instanzen wird allerdings von einer separaten `com.gfs.gis.exmap.layer.factories.-pool.PoolableLayerFactory` analog zur `SAXSerializerFactory` erledigt. Ein `com.gfs.gis.exmap.layer.factories.pool.PoolableLayer` implementiert das `Layer`-Interface und bietet darüber hinaus Methoden zur Verwaltung eines Erzeugungszeitstempels, die eine zeitabhängige Validierung dieses Objektes gestatten. Jede Instanz der `LayerControlerImpl`-Klasse verfügt über einen eigenen `ObjectPool`.

Der Aufbau und die Instanziierung des `LayerControlerHandlers` und der verwalteten `LayerControler` vollziehen sich nach dem folgenden Schema. Die Implementierung des `LayerControlerHandler` verwaltet die `LayerControler` in einer `java.util.HashMap`, die einen effizienten, namenbasierten Zugriff auf die enthaltenen `LayerControler` ermöglicht. Sollte kein `LayerControler` an den mit der `getLayerControler`-Methode übergebenen Namen gebunden sein, wird eine neue Instanz der Implementierung des `LayerControler`-Interfaces erzeugt. Zur Initialisierung werden der jeweilige Name und das Wurzelverzeichnis der Konfigurationsdateien als Parameter übergeben. Zur Konfiguration werden zwei Java-Properties-Dateien verwendet, deren Name sich aus dem Layernamen und den Suffixen „`.properties`“ bzw. „`.poolconfig.properties`“ zusammensetzen. Mit Hilfe von „`default`“-Dateien mit dem entsprechenden Suffix können sowohl für die Eigenschaften der Layer als auch für deren Pools Standardwerte gesetzt werden.

In der jeweiligen „.poolconfig.properties“-Datei werden die Konfiguration des verwaltenden Pools und dessen erzeugende Factory abgelegt. Jede dieser das Interface com.gfs.gis.exmap.layer.controler.pool.ObjectPoolFactory implementierenden Klassen hat alleinig die Aufgabe, analog zur SAXSerializerFactory einen ObjectPool zu erzeugen und wird von der LayerControlerImpl-Klasse benutzt. Die zur Initialisierung einer ObjectPool-Implementierung benötigte PoolableObjectFactory bildet die LayerControlerImpl-Klasse selbst.

Aus der „.properties“-Datei werden die Properties des LayerControlers erzeugt. Aus den Konfigurationsdaten wird über den festgelegten Schlüssel „factory“ analog zur ObjectPoolFactory die konkrete PoolableLayerFactory ermittelt.

Nach jeder Erzeugung einer Layer-Instanz wird dieser über die entsprechende Methode durch den LayerControler die aktuelle Systemzeit übergeben. Sollte während der Laufzeit die Konfigurationsdatei eines Layers modifiziert werden, kann vor der Rückgabe einer seiner Layer-Instanzen an den Pool deren Gültigkeit mittels des Vergleichs der jeweiligen Zeiten geprüft und die Instanz gegebenenfalls verworfen werden. Diese Überprüfung kann durch Setzen einer reloadable-Property deaktiviert werden, um die Zugriffe auf die Festplatte, die wegen der Ermittlung des letzten Modifikationszeitpunkts der Konfigurationsdatei nötig sind, und die Anzahl der offenen Datei-Deskriptoren zu minimieren. Solange diese Eigenschaft noch nicht den Wert „false“ hat, werden die aktuellen Konfigurationsdaten von der Festplatte geladen.

Durch diese Konstruktion ist sowohl die Erweiterung des WMS zur Laufzeit um einen weiteren Layer, als auch eine Änderung eines bestehenden trotz der Bereitstellung von mehreren exklusiven Layer-Instanzen möglich. Die Verwendung von einzelnen Dateien für jeden Layer ermöglicht - im Gegensatz zur Verwaltung der Konfigurationsdaten aller Layer in einer Datei - die Änderung eines Layers unter Beibehaltung der Gültigkeit aller weiteren Layer und deren Instanzen.

4.2.6.2 Abstract-Layer

Jede Layer-Klasse hat das in Abschnitt 4.2.4.1.2 beschriebene Layer-Interface zu implementieren. Um die Implementierungen zu vereinfachen, wurde die abstrakte Klasse com.gfs.gis.exmap.layer.impl.AbstractLayer entwickelt, die die notwendigen „set“-Methoden des Layer- und PoolableLayer-Interfaces implementiert und lediglich die Umsetzung der abstrakten generate-Methode an die konkreten erbenden Layer-Klassen überträgt. Weiterhin verfügt diese Klasse über Hilfsmethoden zur Übergabe der

Genauigkeit der transformierten Koordinaten und zum Vergleich des unterstützten Koordinatensystem mit dem des SRS-Parameters. Darüber hinaus enthält diese Klasse eine Methode zur Erzeugung des die Layer-Daten umgebenden Gruppen-Elementes, dessen Attribute sich aus der Angabe der konkreten räumlichen Begrenzung des Layers, des Layernamens, des Styles des Layers angegeben im `class`-Attribut und dem Typus der konkreten Implementierung zusammensetzen. Die Erweiterung des WMS um eine weitere Datenbasenart kann durch die Benutzung dieser abstrakten Klasse auf einfache Art und Weise durchgeführt werden und verhindert die redundante Implementierung von bestehenden Funktionalitäten.

Im Folgenden werden die konkreten Implementierungen des `Layer`-Interfaces und somit die Umsetzung der Anbindung der verschiedenen Datenbasen an den WMS beschrieben. Die Methoden, die zur Übergabe von bestimmten zur Durchführung der Transformationsaufgabe benötigten Objekte und Werte verwendet werden und nicht im `Layer`-Interface angeführt sind, werden analog zum Factory-Konzept der `SAXSerializer` von der jeweiligen `PoolableLayerFactory` genutzt, die die Erzeugung der entsprechenden, konkreten `Layer`-Instanz übernimmt.

4.2.6.3 Simple Feature-Layer

Eine Simple Feature for SQL-konforme Datenbank angebunden über ein breitbandiges Netzwerk oder residierend auf demselben Computer wie der WMS bildet momentan eine der performantesten Lösung zur Verarbeitung von Simple Features.

Die Verbindung mit einer dieser Datenbanken erfolgt bei dieser `com.gfs.gis.exmap.layer.impl.SimpleFeatureStatementLayer` genannten Klasse mittels *JDBC*. Genauer gesagt wird dieser Klasse ein präkompiliertes SQL-Statement, also ein `java.sql.PreparedStatement`-Objekt, zur Abfrage der Simple Features übergeben und bleibt bis zur Finalisierung einer Instanz dieser Klasse an diese gebunden. Erzeugt wird dieses `PreparedStatement` von der entsprechenden `PoolableLayerFactory` unter Verwendung der Datenbank-URL und dem Anfrageausdruck, angegeben in der „`.properties`“-Datei des jeweiligen Layers.

Die Ausprägung dieser Anfrage ist für die Klasse irrelevant, so dass die gesamte relationale Algebra für die Abfrage der Daten eingesetzt werden kann. Die Klasse benötigt darüber hinaus ausschließlich den Namen der Spalte mit dem Identifikationsattribut des jeweiligen Simple Features, den Namen der WKBGeometry-Spalte und optional die Position des Parameters zur Übergabe der aktuell angefragten

räumlichen Begrenzung durch die Werte des BBOX-Parameters. Die Unterstützung der TIME- und ELEVATION-Parameter ist in dieser Klasse nicht vorgesehen.

Beim Setzen des BBOX-Parameters wird aus dessen Werten, sofern die Angabe der entsprechenden Parameterposition erfolgt ist, die WKTGeometery-Repräsentation eines Polygons mit entsprechenden Begrenzungen erzeugt und somit als String an der übergebenen Stelle im `PreparedStatement` eingefügt. Welche Funktion auf diesen Parameter angewendet wird ist ebenso wie der gesamte Aufbau des SQL-Ausdrucks für diese Layer-Art nicht von Bedeutung. So ist sowohl die auf der konkreten Form der Geometrien fußende Abfrage des Enthaltenseins möglich, als auch die Überprüfung von Überschneidungen des übergebenen Rechtecks und der Begrenzungsrechtecke der Geometrien mittels so genannter R-Tree-Indizes performant durchführbar. Ausschließlich die verwendete Datenbank beschränkt die Mächtigkeit des sprachlichen Ausdrucksvermögens. Auf der einen Seite erfüllt nicht jede Datenbank die gesamte SFS-Spezifikation, auf der anderen Seite können die Datenbanken über eigene Funktionen, Ausdrücke und Konstruktionen verfügen, die nicht in der SFS-Spezifikation vorgesehen sind, aber eine Geschwindigkeitssteigerung oder einen andersartigen Mehrwert bieten.

Bei der Ausgabe der Geometrien verhält es sich ähnlich. Für die Verarbeitung der Geometrien ist lediglich entscheidend, dass diese in der WKBGeometry-Repräsentation vorliegen. Inwieweit die persistent gehaltenen Daten vor der Ausgabe durch SFS-konforme oder auch Datenbank-inhärente Funktionen verändert, nur teilweise selektiert oder auch generalisiert wurden, ist für die Weiterverarbeitung dieser Daten unerheblich, solange sie WKBGeometry-konform repräsentiert werden. Weiterhin ist die `SimpleFeatureStatementLayer`-Klasse in der Lage, sämtliche Simple Features zu verarbeiten.

Die Transformation der Simple Features wird allerdings von einer Hilfsklasse übernommen. Nach dem Aufruf der `generate`-Methode durch das WMS-Servlet wird das umgebende Gruppierungselement erzeugt, das `PreparedStatement` ausgeführt und die `com.gfs.gis.exmap.util.svg.WKBSVGSAXTransformer` genannte Hilfsklasse erzeugt. Zur Initialisierung wird dem `WKBSVGSAXTransformer` der aktuelle Style, also das jeweilige `class`-Attribut, der minimale x- und der maximale y- Wert, die Genauigkeit für diesen Layer und der aktuelle `ContentHandler`, also der `SAXSerializer`, übergeben. Bei der sequentiellen Abarbeitung des Anfrageergebnisses werden alle Eigenschaften

der Simple Features, die nicht der Geometrie oder der Id entsprechen, in eine textuelle Repräsentation transformiert und unter einem speziellen Namespace in einer `SAX-Attributes`-Liste abgelegt. Erweitert wird diese Liste um das `id`-Attribut bestehend aus dem Layernamen und dem Identifikationsattribut des jeweiligen Simple Features, um eine dokumentweite Eindeutigkeit dieser Attribute zu erreichen. Anschließend werden dem `WKBSVGSAXTransformer` die Liste der Attribute und die Referenz auf den WKBGeometry-Binärstrom übergeben. Die Transformation der Simple Features in die SVG-konforme Repräsentation vollzieht sich dabei nach den in Abschnitt 2.2.2 artikulierten Regeln und der Kompression der Koordinaten gemäß Abschnitt 3.2.2.5.1. Dem jeweilig ersten erzeugten SVG-Element bzw. dem ersten SAX-`startElement`-Ereignis werden die übergebenen Attribute angefügt, also bei einem Polygon direkt dem `path`-Element und bei einer Aggregationsklasse dem umgebenden Gruppierungselement.

Die Auslagerung der eigentlichen Transformationsaufgabe gestattet eine einfache Erweiterung des WMS um die Unterstützung von Datenbasen, die die Simple Features mittels COM oder CORBA anbieten.

Für eine effiziente Transformation und Analyse des Binärstroms ist wenn möglich auf eine Verwendung der Big-Endian-Byteorder zu achten, da Java bedingt durch seine Plattformunabhängigkeit nur diese Ordnung vorsieht und eine Umwandlung zwar in der Transformationsklasse vorgenommen werden kann, aber das „shiften“ durch die meist nativen Datenbasen performanter vollzogen wird[67].

Momentan wird die SFS-Spezifikation von zwei Open-Source-DBMS implementiert. Das Datenbank Management System *MySQL* unterstützt seit der Version 4.1 die Spezifikation rudimentär. Die für die Verwendung als Datenbasis des WMS wesentlichen Merkmale wie Speicherung der Simple Features in der WKBGeometry-Repräsentation und deren indizierte Selektion durch systemeigene Funktionen werden allerdings von diesem DBMS angeboten.[68]

Die *PostGIS*-Erweiterung des *PostgreSQL*-DBMS implementiert hingegen die vollständige Unterstützung von SFS. Über die SFS-Spezifikation hinaus bietet dieses DBMS Funktionalitäten zur Transformation der Koordinaten in alle von der EPSG

---

[67] Eine Java-seitige Änderung der Byte-Order führt auf dem Testsystem bei dem in Abschnitt 3.2.2.5 angeführten Beispiel zu einer um den Faktor 3 langsameren Gesamtanfragebeantwortungszeit.

[68] vgl. MySQL (2004)

veröffentlichten Koordinatensysteme und zur Generalisierung der Simple Features mit dem Douglas/Peuker-Algorithmus an.[69]

Weiterhin wird auch von einigen kommerziellen Datenbanksystemherstellern wie beispielsweise beim *Spatial Extender* der Datenbank *DB2* von IBM schon momentan die WKTGeometry- bzw. WKBGeometry-Repräsentation der geographischen Objekte unterstützt[70].

4.2.6.4 Label-Layer

Die `com.gfs.gis.exmap.layer.impl.LabelLayer`-Klasse funktioniert nach ähnlichen Mechanismen wie die `SimpleFeatureStatementLayer`-Klasse, nur werden bei dieser Layer-Art nicht Geometrien in die Karte eingefügt, sondern Beschriftungen an vorgegebenen Koordinaten. Statt der Angabe über die WKBGeometry-Spalte benötigt die `LabelLayer`-Klasse die Namen der entsprechenden Beschriftungstext-Spalte und der X- bzw. der Y-Koordinaten-Spalte. Direkt von der `LabelLayer`-Klasse wird ein `text`-Element mit den transformierten Koordinaten und allen weiteren nach dem oben beschriebenen Verfahren erzeugten Attributen angelegt und der Beschriftungstext wird mittels SAX-`startCharacter`-Ereignis übergeben. Bislang werden weder verschiedene Sprachen, noch die Ausrichtung des Textes an Pfaden oder die Rotation des Textes durch die Übergabe eines entsprechenden Winkels unterstützt. Eine dahingehende Erweiterung ist allerdings problemlos möglich.

4.2.6.5 Tiled Raster-Layer

Analog zu den Simple Feature- und Beschriftungslayer-Klassen bietet die `com.gfs.gis.exmap.layer.impl.TiledRasterStatementLayer`-Klasse die Geometrie-Datenbank-gestützte Verarbeitung von großen, gekachelten Rasterbildern. Zur Erzeugung der `image`-Elemente benötigt diese Klasse die Namen der Spalten, die den URI-Verweis auf das eigentliche Bild und die räumliche Begrenzung, angegeben durch die minimalen und maximalen x- und y-Koordinaten, enthalten. Die nach dem obigen Verfahren ermittelten weiteren feature-Eigenschaften werden um die aus den Koordinaten berechneten `x`- und `y`- bzw. `width`- und `height`-Attribute und um das nötige `xlink:href`-Attribut ergänzt.

---

[69] vgl. PostGIS (o. J)

[70] vgl. IBM (2002)

Zur Selektion der jeweiligen Raster-Kacheln sollte die räumliche Begrenzung der einzelnen Kacheln als Simple Feature in der entsprechenden Relation abgelegt und mit einem Geometrie-Index versehen sein, so dass abhängig vom BBOX-Parameter die erforderlichen Kacheln auf effiziente Art und Weise von der Datenbank ermittelt werden können.

4.2.6.6 External Raster-Layer

Die `com.gfs.gis.exmap.layer.impl.ExternalRasterResourceLayer`-Klasse dient zum einfachen Einbinden von einzelnen räumlich referenzierten Rasterbildern, die ebenso wie beim oben beschriebenen `TiledRasterStatementLayer` nur in den von SVG unterstützen Formaten PNG und JPEG vorliegen dürfen. Bei der Initialisierung dieser `Layer`-Klasse werden ihr, analog zur jeweiligen Zeile einer SQL-Anfrage, der URI-Verweis auf das eigentliche Bild und die räumliche Begrenzung, angegeben durch die minimalen und maximalen x- und y-Koordinaten, übergeben. Bei jedem `generate`-Aufruf werden aus diesen Daten und abhängig vom übergebenen BBOX-Parameter die jeweiligen Kartenkoordinaten berechnet und mit den Attributen der Höhe und Breite bzw. des Verweises auf das Bild ein `image`-Element generiert.

4.2.6.7 WMS-Layer

Die `com.gfs.gis.exmap.layer.impl.WMSLayer`-Klasse dient zur kaskadierenden Verknüpfung von nachgelagerten Web Map Services, bei denen es sich sowohl zur rasterbasierten Generalisierung um den WMS selbst als auch um externe Web Map Services handeln kann. Die `WMSLayer`-Klasse funktioniert an sich nach ähnlichen Prinzipien wie die `ExternalRasterResourceLayer`-Klasse. Bei jedem Aufruf der `generate`-Methode wird jeweils ein Layer mit nur einem darin enthaltenen `image`-Element generiert.

Hinsichtlich der Attribute dieses `image`-Elementes verhält es sich aber grundsätzlich anders. Der Zugriff auf einen WMS ermöglicht im Gegensatz zum statischen Rasterbild die dynamische Übergabe der BBOX-, SRS-, HEIGHT- und WIDTH-Parameter. Der nötigen Basis-URL des nachgelagerten WMS mit den statischen, frei konfigurierbaren Parametern wie LAYERS, STYLES usw. werden die dynamischen Parameter der Anfrage angefügt. Durch einen Initialisierungsparameter kann die Auflösung der resultierenden Karte entweder direkt durch Weitergabe der ursprünglichen WIDTH- und HEIGHT-Parameter oder durch die Berechnung der WIDTH- und HEIGHT-

Parameter auf Basis des BBOX-Parameters mit entsprechendem Genauigkeitsfaktor bestimmt werden.

Da das `xlink:href`-Attribut unter SVG nur eine URI als Verweis gestattet, also die WMS-Anfrage-URL mit den nötigen Parametern nicht unterstützt wird und das Auftreten der Parametertupelbegrenzungszeichen Fehler verursacht, muss die WMS-Anfrage-URL kodiert und als URI angefügt werden, deren entsprechender Service die angefügte WMS-Anfrage-URL dekodiert und als so genanntes HTTP-Redirect zurückschickt. Da weiterhin nicht jeder SVG-Client, so auch das verwendete Adobe SVG-Viewer-Plugin im Gegensatz zur Batik-Komponente, ein solches HTTP-Redirect verarbeitet, wurde ein separates Servlet (`com.gfs.util.servlets.io.Proxy`) implementiert, das die WMS-Anfrage-URL dekodiert, seinerseits die Anfrage ausführt, das Ergebnis in den ursprünglichen Ausgabestrom schreibt und somit als Proxy-Servlet fungiert. Diese Konstruktion ermöglicht nicht nur die Verarbeitung einer jeden URL als `xlink:href`-Attribut, sondern gestattet darüber hinaus das Caching der generierten Karten des angebundenen WMS durch den vorgelagerten Web-Server. Hinsichtlich der Pufferung der Daten wird dasselbe Konzept wie beim WMS-Servlet angewendet. Sollte die Antwort des angebundenen WMS keine Angaben über den letzten Modifikationszeitpunkt und den Verfallszeitpunkt enthalten, werden diese Angaben aus den Konfigurationsdaten des Servlets ermittelt.

4.2.6.8 WFS-Layer

Die `com.gfs.gis.exmap.layer.impl.WFSLayer`-Klasse dient zur universellen Anbindung eines Web Feature Services an den WMS. Hinsichtlich der Verarbeitung der WFS-Anfragen erfüllt diese Klasse die gleichen Anforderungen an Flexibilität und Unabhängigkeit wie die `SimpleFeatureStatement`-Klasse, so dass die Gestaltung der Anfragen nur von den Richtlinien des OGC abhängt. Somit kann bei einer Erweiterung des basic WMS zu einem SLD WMS auf die bestehenden Implementierungen zurückgegriffen werden.

Die nötige Flexibilität wird durch die Verknüpfung mit einem externen WFS-Anfrage-XML-Dokument erreicht. Der Verweis auf dieses Dokument wird aufgelöst und an einen SAX-Parser mit vorgelagertem SAX-Filter übergeben, analog zur Verarbeitung der GetCapabilities-Operation. Als `ContentHandler` des Filters fungiert der `XMLSerializer`, dessen Ausgabestrom die `java.net.HttpURLConnection` zum

entsprechenden WFS stellt. Gekapselt wird die `HttpURLConnection` durch eine Hilfsklasse (`com.gfs.util.http.HTTPPostRequest`).

An die Filter-Klasse, `com.gfs.gis.exmap.util.sax.WFSRequestFilter` genannt, werden die Koordinaten des BBOX-Parameters übergeben, so dass dieser Filter beim Auftreten einer SAX-`processingInstruction` mit der Bezeichnung „`BBOX`“ auf Basis dieser Koordinaten eine GML-konforme Repräsentation des Begrenzungsrechtecks statt der Processing-Instruction erzeugt. Weiterhin ermittelt dieser Filter während des Durchlaufs den Namen des jeweiligen Simple Features und dessen in den `PropertyName`-Elementen abgelegten Namen der Sekundärdatenelemente.

Diese Konstruktion ermöglicht sowohl die dynamische Übergabe des jeweiligen BBOX-Parameters an eine in der Anfrage dafür vorgesehene Funktion, als auch die Ermittlung der für die Umwandlung der entsprechenden Antwort nötigen Namen des Simple Features und der Eigenschaften dieses Simple Features in einem Durchlauf und ohne redundante Pflege der benötigten Daten. Dies macht die Unterstützung eines SLD WMS inklusive der Angabe der Sekundärattribute in der SVG-Karte erst möglich, da sowohl bei dieser Konstruktion als auch bei einem SLD WMS ausschließlich der Verweis auf das jeweilige Anfrage-Dokument benötigt wird.

In der `generate`-Methode der `WFSLayer`-Klasse wird nach erfolgreichem Verbindungsaufbau zum Web Feature Service und der Verarbeitung der Anfrage die Antwort des WFS wieder an einen SAX-Parser mit vorgelagertem Filter übergeben. Der `com.gfs.gis.exmap.util.svg.GMLSVGSAXFilter` genannte Filter erzeugt analog zum `WKBSVGSAXTransformer` die SVG-Elemente. Anders als der WKBGeometry-Transformer verarbeitet der Filter allerdings die gesamte Antwort und löst beim jeweiligen Auftreten von GML-Elementen entsprechende Verarbeitungsroutinen aus, die von der Struktur der GML-Elemente abhängig das Auslösen der resultierenden SVG-SAX-Ereignisse beim übergebenen `SAXSerializer` veranlassen. Die ermittelten sekundären Eigenschaften werden ebenso wie bei der WKBGeometry-Transformation dem jeweilig ersten erzeugten SVG-Element eines Simple Features angefügt. Zur Erläuterung der genauen Funktionsweise dieser SAX-Filter-basierten Transformation von GML zu SVG sei auf den Quellcode verwiesen.

Trotz der effizienten, stromartigen und speicherschonenden Transformation von GML in SVG, besonders im Vergleich zum XSLT-Verfahren, ist diese Verarbeitung von Simple Features durch das rechenintensive Parsen der alpha-numerischen

Repräsentation der Koordinaten signifikant langsamer als bei deren Kodierung im WKBGeometry-Format. Eine Anbindung von Web Feature Services sollte somit nur vorgenommen werden, sofern ein direkter Zugriff auf die WBKGeometry-Repräsentation dieser Daten nicht möglich ist.

Von dem Aufbau einer persistenten Verbindung zum WFS wird abgesehen, da einerseits der Verbindungsaufbau im Vergleich zur Übertragung der Daten vernachlässigbare Kosten verursacht und anderseits jede Verbindung einen zusätzlichen Thread bzw. Socket erfordert.

4.3 Der Client – die Schnittstelle zum Benutzer

Der Client als OGC Application Service bildet die Schnittstelle zwischen dem WMS, dem Kartenautor und dem Benutzer. Bei der Implementierung des Clients steht daher die Entwicklung einer Umgebung im Vordergrund, die sowohl dem Kartenautor eine standardisierte Basis zum Design beliebiger Karten stellt, als auch dem Endanwender einen intuitiven Zugang zu den aufbereiteten geographischen Informationen gewährt.

4.3.1 Der Client aus Endanwendersicht

4.3.1.1 Graphischer Aufbau und Symbolik

Der graphische Aufbau der Client-Applikation erfolgt nach der in Abschnitt 3.3.2 beschriebenen Konzeption. Die allgemeine Toolbar ist in der oberen linken Ecke der Applikation platziert. Die Legende mit der entsprechend kontextabhängigen Toolbar schließt sich nach unten verlaufend daran an. Diagonal gegenüber befindet sich die Übersichtskarte.

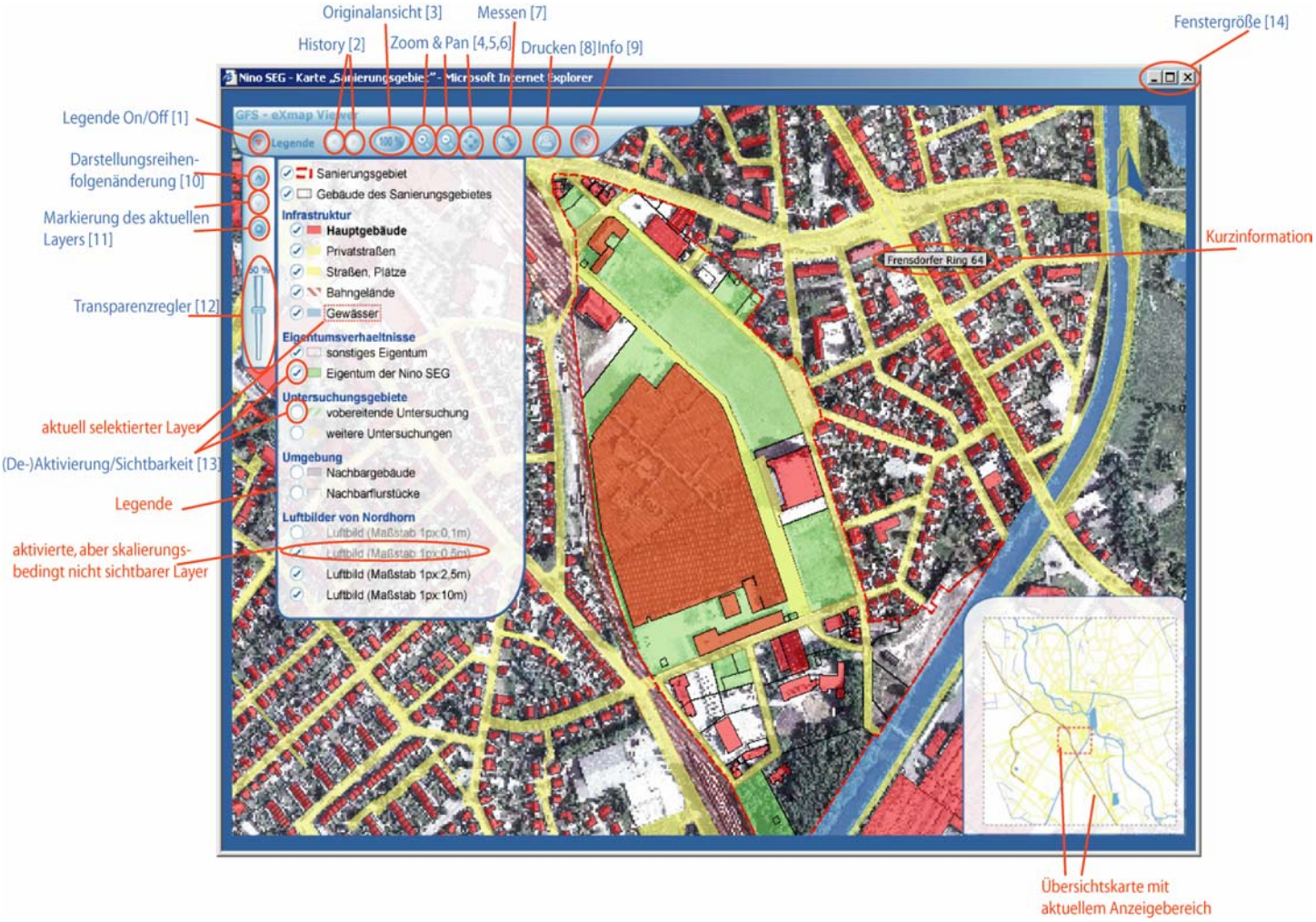

Abbildung 10 - Screenshot des Clients[71]

Die Client-Applikation füllt das gesamte Browser-Fenster aus. Bei dessen Größenänderung bleibt die relative Positionierung der Toolbar bzw. der Legende und der Übersichtskarte zu den jeweiligen Ecken des Fensters ebenso wie die Größe dieser Komponenten erhalten. Die eigentliche Detailkarte hingegen wird mit der Größe des Browser-Fensters skaliert.

Der jeweilig sichtbare Kartenausschnitt wird in der Übersichtskarte durch das rote, gestrichelte Rechteck dargestellt. In der Legende markiert ein ebenso gestaltetes Rechteck den jeweils selektierten Eintrag, der den Kontext für die Toolbar der Legende bildet. Ein ausgegrauter Legendeneintrag zeigt dem Endanwender, dass die Daten dieses Eintrags im aktuellen Maßstab der Karte nicht sichtbar sind. Jeweils fett geschrieben wird der Eintrag in der Legende, der das oberste in der Detailkarte befindliche feature

[71] Das Design der Toolbars und der Button ist nicht Teil dieser Diplomarbeit, sondern wurde extern entworfen und als SVG-Fragment in den Client integriert. Dieser Aspekt stellt weiterhin die Vielseitigkeit von SVG unter Beweis.

unterhalb des Cursors beinhaltet. Ebenso können zu diesem feature je nach Konfiguration Detailinformationen in die Karte projiziert werden.

Der Legendenbereich ist nicht vollständig deckend, damit bei einer umfangreichen Legende der Benutzer auch weiterhin die groben Strukturen der Detailkarte, die zum Teil durch die Legende verdeckt werden, erkennen kann.

4.3.1.2 Funktionalitäten

Zur vollständigen Maximierung des Darstellungsbereichs kann die Legende mitsamt der Übersichtskarte aus- und eingeblendet werden [1]. Die von Shneiderman geforderte Zoom-Funktionalität wird durch drei Werkzeuge umgesetzt. Das Vergrößerungswerkzeug [4] gestattet dem Anwender sowohl durch Klicken in die Detailkarte die Zentrierung des Klickpunktes und die Vergrößerung der Umgebung durch einen fixen Skalierungsfaktor, als auch die freie Auswahl des zu vergrößernden Bereiches durch das „Ziehen“ eines Auswahlrechtecks bei gedrückter linker Maustaste. Das Verkleinerungswerkzeug [5] arbeitet analog zum Vergrößerungswerkzeug, wobei aber keine Bereichsselektion möglich ist. Das Verschieben des Kartenausschnitts wird durch das Schwenkwerkzeug [6] ermöglicht. Bei gedrückter linker Maustaste ist ein freies Schwenken der Karte möglich. Erst nach dem jeweiligen Abschluss der Benutzeraktionen wird der aktuelle Kartenausschnitt analysiert und somit gegebenenfalls das Nachladen eines Layers oder die skalierungsabhängige (De-) Aktivierung veranlasst. Mit Hilfe der History-Funktion [2] kann der Anwender durch die Abfolge dieser Benutzeraktionen navigieren. Die separate Funktion zur Darstellung des ursprünglichen Kartenausschnitts [3] erspart dem Anwender die vollständige Rückverfolgung seiner Aktionen.

Weiterhin stehen dem Anwender in der allgemeinen Toolbar ein Werkzeug zur Distanzmessung [7] und die Möglichkeit zum Drucken der aktuellen Ausprägung des Clients [8] zur Verfügung. Ein verhältnistreuer Ausdruck erfordert allerdings eine entsprechende Einstellung der Seitengröße, da der Client ebenso wie im Browser-Fenster beim Druck maximiert wird, eine verhältnistreue Ausrichtung aber entfällt.

Die Zoom- und Pan- bzw. die Mess-Funktionen werden direkt in der Detailkarte angewendet, so dass die details-on-demand-Funktionen der Detailkarte während der Verwendung dieser Funktionen deaktiviert sind, was unter anderem durch den jeweiligen Cursor symbolisiert wird. Die Deaktivierung der jeweiligen Funktion kann

sowohl direkt durch die Aktivierung der Info-Funktion [9] als auch indirekt und automatisch durch die Durchführung einer beliebigen anderen Funktion erfolgen.

Als Layer-spezifische Funktionen stehen dem Anwender die Änderung der Darstellungsreihenfolge [10], das Markieren eines Layers [11] und die stufenlose Veränderung der Deckkraft eines Layers mittels Schieberegler [13] zur Verfügung. Die Layer werden abstrahiert von den Named Layers des WMS durch einen Eintrag in der Legende repräsentiert, und können durch Gruppierungseinträge hierarchisch strukturiert sein. Durch Anklicken des jeweiligen (Gruppierungs-)Eintrags wird dieser selektiert und bildet den Kontext für die Toolbar der Legende. Je nach Eintragstypus sind nur bestimmte Funktionen der Toolbar möglich. Die Änderung der Darstellungsreihenfolge ist bei jedem Eintragstypus möglich. Dabei wird der jeweilige Eintrag innerhalb der Hierarchieebene je nach Richtung mit seinem Vorgänger oder Nachfolger vertauscht. Dahingegen können nur Einträge bzw. Layer markiert werden, die auf einer vektoriellen Datenbasis beruhen. Die Modifikation der Transparenz ist bei flächigen Layern sowohl des vektoriellen als auch des rasterbasierten Typus möglich.

Zur Steigerung der Übersichtlichkeit wurde die (De-)Aktivierungsfunktion [10] für jeden Layer einzeln direkt vor dem entsprechenden Eintrag in der Legende platziert.

4.3.2 **Der Client aus Sicht des Kartenautors**

Aus Sicht des Kartenautors steht die Entwicklung einer Umgebung im Vordergrund, die auf der einen Seite eine einfache, redundanzlose Erzeugung der Karten mit allen im obigen Abschnitt dargestellten und im Abschnitt 3.3 geforderten Funktionalitäten ermöglicht, und auf der anderen Seite so flexibel ist, dass der Gestaltungsspielraum ausschließlich durch den SVG-Standard beschränkt wird.

4.3.2.1 Aufbau der eXmap

Zur Erfüllung dieser Anforderungen basiert das *eXmap* getaufte Konfigurationsdokument der Karten auf einem erweiterten SVG-Dokument. Diese eXmap wird clientseitig beim Laden analysiert. Diese Analyse erfordert entsprechende Strukturen, Elemente und Attribute in dem SVG-Dokument. Das Dokument enthält soweit möglich keine eigenen Daten, sondern bildet ausschließlich die strukturelle Hülle für die eigentliche Karte. Alle Erweiterungsattribute werden in dem Namensraum `exmap` deklariert.

Das `svg`-Wurzelelement einer eXmap wird vornehmlich um solche Attribute erweitert, die den verwendeten WMS mit den jeweilig nötigen, aber variablen Parametern beschreiben und Angaben über das zugrunde liegende geographische Koordinatensystem bzw. den initialen Kartenausschnitt enthalten. Der minimale x- bzw. y-Wert des anzuführenden `BBOX`-Attributes entspricht, analog zum transformierten Koordinatensystem der Karten des WMS, dem Nullpunkt des Koordinatensystems der Detailkarte. Die Ausrichtung dieses `svg`-Elementes muss links oben sein. Weiterhin muss beim Laden die entsprechende Initialisierungsfunktion aufgerufen werden.

Als Kinder beinhaltet das Wurzelelement neben den nötigen Verweisen auf die Clientprogrammdateien fünf Elemente. Angeführt von einem Gruppenelement, das ausschließlich bis zum Abschluss des Initialisierungsprozesses sichtbar ist, folgen ein Gruppierungselement als Containerelement für eine Extended-Style-Definition-Datei, sowie ein weiteres `svg`-Element als Wurzel der Detailkarte, ein Gruppierungselement als Containerelement für das SVG-Fragment, das die graphischen Elemente der Toolbar und der Legende enthält und ein Gruppierungselement als Wurzel zur Definition der Übersichtskarte.

Konzeptuell fußt die Übersichtskarte auf der Verwendung einer separaten Karte im Rasterdatenformat, was aufgrund der geringen Auflösung und gemäß den in Abschnitt 3.2.2.5.2 dargelegten Generalisierungskonzepten ein wesentlich niedrigeres Datenaufkommen erwarten lässt als bei einer SVG-basierten Repräsentation. In dem die Übersichtskarte beinhaltenden `image`-Element werden daher die nötigen variablen WMS-Parameter zur Anfrage der Karte wie LAYERS, STYLES, BBOX, FORMAT, WIDTH und HEIGHT definiert. Zusätzlich ist wegen der im Abschnitt des WMS-Layers beschriebenen Probleme die Angabe einer URI des zu verwendenden Proxys von Nöten. Das Wurzelelement enthält neben dem nötigen `image`-Element noch einige weitere Gestaltungselemente. Die Integration der Extended-Style-Definition-Datei und des SVG-Fragments der Toolbar und der Legende erfolgt hingegen parametrisiert während des Initialisierungsprozesses.

Zur Strukturierung der Detailkarte und somit zur Gestaltung der Legende stehen dem Kartenautor zwei spezielle Gruppierungselemente zur Verfügung. Deklariert wird der Typus dieser Elemente durch ein spezielles `exmap:type`-Attribut. Der Wert `MenuGroupEntry` beschreibt einen Legendengruppierungseintrag, dessen Beschriftung durch das `exmap:title`-Attribut gesetzt wird. Als Kindelemente dieser

Gruppierungselemente sind sowohl weitere Gruppierungselemente desselben Typus als auch Gruppierungselemente des Typs `MenuEntry` möglich. Elemente dieses Typs repräsentieren die Layereinträge der Legende. Neben dem `exmap:type`- und dem `exmap:title`-Attribut können diese Elemente ein weiteres `exmap:GeometryType`-Attribut enthalten, über dessen Werte die Art des Legendensymbols dieses Layers bestimmt wird. Bei den Werten `Image` und `Point` kann durch ein zusätzliches `exmap:symbol`-Attribut auf die Id des zu verwendenden Symbols referenziert werden. Bei allen weiteren Werten des `exmap:GeometryType`-Attributs, vornehmlich (`Multi`-)`Polygon` und (`Multi`-)`Polyline`, wird einem entsprechenden allgemeinen Symbol für diesen Typus der Wert des `style`-Attributs dieses Elementes übertragen. Auf diese Art und Weise wird die redundanzlose Gestaltung der Karte vollzogen, da sich dieses `style`-Attribut nicht nur direkt auf die Kindelemente dieses Gruppierungselementes und somit auf die Darstellung der transformierten Simple Features auswirkt, sondern auch die Gestaltung des Symbols des Legendeneintrags indirekt bestimmt. Weiterhin wirken sich die Funktionen zur Variation der Transparenz und die Markierung der Layer ebenso auf die jeweiligen `style`-Attribute aus. Die initiale (De-)Aktivierung des Eintrags wird über das SVG-eigene `visibility`-Attibut geregelt. Zur einwandfreien Identifikation muss jedes der beschriebenen Elemente mit einem eindeutigen `id`-Attribut versehen werden.

Die Verknüpfung des WMS mit dem Client erfolgt mittels eines weiteren spezifischen Gruppierungselementes des Typs `LayerContainer`, im Folgenden *Layercontainer* genannt. Jedes dieser Elemente verweist direkt mit seinem `id`-Attribut auf einen Named Layer des WMS. Mindestens eins dieser Elemente sollte als Kind- oder Kindeskindelement eines Gruppierungselementes des Typs `MenuEntry` angeführt werden. Die Strukturierung unterhalb eines Legendeneintragselementes ist vollkommen variabel und losgelöst von der Legendenstruktur. So können mehrere Layercontainer unter einem Legendeneintrag zusammengefasst werden, um einerseits einen adaptiven Layer erzeugen zu können, und um andererseits eine inhaltliche Aggregation verschiedener Named Layer des WMS, die jeweils nur auf einem Anfrageausdruck beruhen, vornehmen zu können. Weiterhin kann durch diese Flexibilität beispielsweise ein Layer des WMS als Maske für einen Weiteren definiert werden. Daher bildet ein Legendeneintrag eine Abstraktion von den Layern des WMS.

Konfiguriert wird ein Layercontainer neben den obligatorischen Attributen `id` und `type` durch die Attribute `exmap:loading`, `exmap:BBOX`, `exmap:minScale`, `exmap:maxScale`, `exmap:horizontal-tiles` und `exmap:vertical-tiles`. In der vereinfachten Variante mit

dem `exmap:loading`-Attributwert `full` werden die Daten des Layers abhängig vom `exmap:BBOX`-Attributwert geladen, sobald der Layer im jeweiligen Maßstab sichtbar und durch den übergeordneten Legendeneintrag aktiviert ist. Dieses Konzept wird bei der Layercontainerart mit dem `exmap:loading`-Attributwert `tiled` um das Konzept des dynamischen Nachladens von ausschließlich im aktuellen Kartenausschnitt noch nicht geladenen Kacheln eines gerasterten Layers erweitert. In diesem Fall bestimmt das `exmap:BBOX`-Attribut die gesamte Ausdehnung der zu rasternden Fläche und das `exmap:horizontal-tiles`-Attribut bzw. das `exmap:vertical-tiles`-Attribut gibt die Anzahl der horizontalen bzw. vertikalen Kacheln an.

4.3.2.2 Interaktivität

Die Programmierung der Interaktivität der Detailkarte, also die Reaktion auf die SVG-Ereignisse, meist beschränkt auf die GUI-Ereignisse `onclick`, `onmouseover` und `onmouseout`, bleibt dem Kartenautor überlassen. Durch die Ablage der feature-Eigenschaften als XML-Attribute der transformierten Simple Features und die vollständige Nutzbarkeit der Scripting-Umgebung, sowohl des Plug-Ins als auch des Browsers, sind der interaktiven Gestaltung der Karten analog zur client- und serverseitigen HTML-Programmierung keine Grenzen gesetzt.

Nichtsdestotrotz stehen dem Kartenautor jeweils eine Standardmethode zur Behandlung des `onmouseover`- bzw. des `onmouseout`-Ereignisses zur Verfügung. Diese beiden quasi miteinander gekoppelten Methoden zur Behandlung des Überfahrens eines features mit der Maus erfüllen drei Funktionen. Erstens wird der jeweilig übergeordnete Legendeneintrag währenddessen fett geschrieben. Zweitens ist mittels Parametern das Setzen eines speziellen temporären Styles dieses features möglich. Drittens kann durch die Übergabe eines Attributnamens der Wert der jeweiligen feature-Eigenschaft in die Karte projiziert werden. Welche Sekundärattribute die features eines Layers enthalten, richtet sich nach der jeweiligen Konfiguration des Named Layers des WMS.

4.3.3 Implementierung der Kernfunktionalitäten

4.3.3.1 Zoom & Pan und die History

Die Zoom und Pan-Funktionalitäten des Client sind, statt auf die entsprechenden Funktionalitäten des SVG-Viewer-Plugins zurückzugreifen, aus verschiedenen Gründen eigenständig implementiert worden:

- Die History-Funktion des Clients und das Konzept des dynamischen Nachladens verlangen eine vollständige Kontrolle der Benutzeraktionen.
- Die Zoom-Funktionalität bezieht sich ausschließlich auf die Detailkarte. Aufgrund der Kombination von Legende, Übersichtskarte sowie Toolbar mit der Detailkarte in einem SVG-Dokument ist die differenzierte Behandlung der Zoom-Funktion von Nöten.
- Das SVG-Viewer-Plugin limitiert den maximalen Vergrößerungs- bzw. Verkleinerungsfaktor. So sind jeweils nur vier Schnitte möglich, was für großräumige und zugleich sehr detaillierte Karten eindeutig zu wenig ist.
- Die Aktivierung der Plugin-eigenen Funktionalitäten kann ergonomisch nicht optimal und wenig intuitiv mittels Tastenkombination, oder ausschließlich eingeschränkt per Kontextmenü erfolgen.

Die Eigenimplementierung wirkt sich allerdings nachteilig auf die Performanz aus, da die Umsetzung im von SVG unterstützten EMCA-Script im Vergleich zu der nativen und direkt im Plugin implementierten Umsetzung langsamer ist. Weiterhin muss für jedes der Werkzeuge zur besseren Orientierung ein separates Cursorsymbol als SVG-Grafik neben der Stelle des eigentlichen Cursors eingeblendet werden, da das SVG-Viewer-Plugin in der Version 3 die direkte Modifikation des Cursors noch nicht unterstützt.

Die Realisierungen der Zoom-In-, Zoom-Out- und Pan-Funktionen fußen auf der Modifikation des `viewBox`-Attributs des `svg`-Wurzelelementes der Detailkarte. Die für die Errechnung der entsprechenden Werte des neuen `viewBox`-Attributs benötigte Umrechnungsfunktion der x- und y-Werte der GUI-Ereignisse im Bildschirmkoordinatensystem in das Koordinatensystem der Detailkarte basiert auf von Kevin Lindsey[72] veröffentlichten Konzepten. Der Transformationsprozess beruht auf der Transformation der um die aktuelle Ausrichtung der Detailkarte gemäß ihrem `preserveAspectRatio`-Attribut modifizierten Ereigniskoordinaten mit der invertierten Transformationsmatrix des `svg`-Wurzelelementes der Detailkarte. Für ein eingehendes Verständnis dieses Prozesses sei auf den Quellcode verwiesen[73].

[72] Lindsey (2002)

[73] Datei: mappingtools.js; Funktion: getMapPoint(x,y)

Nach dem Abschluss der jeweiligen Benutzeraktion, also beim `onmouseup`-Ereignis, wird der aktuelle Wert des `viewBox`-Attributs für die Realisierung der History-Funktion auf einem Stack abgelegt.

4.3.3.2 Dynamisches Nachladen

Die oben angeführte Umrechnungsfunktion ermöglicht auch unter Verwendung der minimalen und maximalen Plugin-Koordinaten[74] und des minimalen x- bzw. y-Wertes des `exmap:BBOX`-Attributs angegeben im `svg`-Wurzelelement die Ermittlung der räumlichen Begrenzung des aktuell sichtbaren Detailkartenausschnitts.

Nach jeder der Benutzeraktionen Layeraktivierung, zoom, pan und der Verwendung der History werden die Elemente unterhalb des Wurzelelementes der Detailkarte rekursiv analysiert und bei jedem Auftreten eines durch den aktuellen Maßstab und die Aktivierung des übergeordneten Legendeneintrags sichtbaren Layercontainers überprüft, welche der in der aktuellen räumlichen Begrenzung befindlichen Kacheln noch nicht geladen wurden. Die Überprüfung einer Kachel erfolgt dabei nicht geometrisch und linear über die bereits geladenen Kacheln, sondern mittels der spezifizierten `getElementById`-Funktion des DOM-ECMAScript-Bindings[75], was je nach Plugin-interner Indizierung der Elemente sogar mit Laufzeit von O(1) realisiert sein kann. Die Verknüpfung des `id`-Attributs des Layercontainers mit den jeweiligen Werten der räumlichen Begrenzung einer Kachel bildet das `id`-Attribut des jeweilig resultierenden Layers und ermöglicht diese Art der Überprüfung.

Für jede noch nicht geladene Kachel wird eine GetMap-Anfrage-URL mit dem `id`-Attribut des Layercontainers als LAYER-Parameterwert, den jeweiligen Werten der räumlichen Begrenzung einer Kachel als BBOX-Parameterwert und den im Wurzelelement der eXmap abgelegten sonstigen Parameterwerten erzeugt und auf einem Stack abgelegt. Nach Beendigung der Analyse wird dieser Stack abgearbeitet und jede Anfrage einzeln an den WMS gerichtet. Das jeweilige Ergebnis wird in ein DOM-Fragment transformiert. Falls es sich dabei um ein Ausnahmedokument des WMS handelt, wird dieses ausgegeben. Andernfalls wird das Gruppierungselement des jeweilig angefragten Layers mit dem entsprechenden `id`-Attribut versehen und als Kind des Layercontainers eingefügt, dessen Id dem `exmap:name`-Attribut dieses Gruppierungselementes entspricht. Aus den jeweiligen Werten der räumlichen

---

[74] (0,0) und `([window.width],[window.height])`

[75] W3C (2003) c S. ecma-script-binding.html

Begrenzung der Karte, angegeben im `exmap:BBOX`-Attribut des Wurzelelementes der eXmap und dem `exmap:BBOX`-Attribut des Layers, wird die Transformation seines Gruppierungselementes errechnet. Nötig wird diese Transformation durch die Normalisierung der Koordinaten bezüglich des minimalen x- bzw. y-Wertes des BBOX-Parameters. Diese Technik kann bei großflächigen, stark gekachelten Layern zu einer zusätzlichen Datenmengenreduktion führen, da weitere Vorkommastellen eingespart werden können. Während des Einfügeprozesses wird für jedes feature dessen schon bestehende Existenz innerhalb der Karte performant über das jeweilige `id`-Attribut überprüft. Die Möglichkeit der redundanten Abfrage eines features, also dessen Beinhaltung in mehreren Kacheln, ist durch die geographische Ausprägung eines features und die Art der Selektion der features des entsprechenden Anfrageausdrucks begründet. Zur Steigerung der Nebenläufigkeit des Clients und somit zur Gewährleistung der Reaktionsfähigkeit auf weitere Benutzeraktionen wird jede Anfrage und deren Ergebnisbearbeitung durch einen eigenen Thread ausgeführt.

Die Umsetzung ist nicht konform zum SVG-Standard. Sowohl die für die Anfrage der Daten nötige `GetURL`-Funktion als auch die zur Erzeugung des DOM-Fragments aus einem Text nötige `parseXML`-Funktion sind nicht vom W3C spezifiziert, sondern werden vom Adobe SVG-Viewer-Plugin zur Verfügung gestellt. Eine SVG-konforme Umsetzung des Konzeptes des dynamischen Nachladens müsste unter Verwendung des `use`-Elementes mit entsprechend gesetzter URI im `xlink:href`-Attribut erfolgen. Allerdings unterstützt das Adobe SVG-Viewer-Plugin beim `use`-Element lediglich Verweise auf Elemente im selben Dokument[76], was folglich die dargelegte Art der Umsetzung erfordert. Weiterhin limitiert das Plugin aus Sicherheitsgründen bei der `GetURL`-Funktion den Zugriff auf URL derselben Domain wie die der eXmap selbst. So ist die Verarbeitung von verschiedenen WMS als Datenquellen einer eXmap nur über den Umweg eines Proxy-Servers oder bei der Installation mehrerer unter der gleichen Domain erreichbarer Web Map Services möglich.

4.3.3.3 Kompression

Wie auch bei der Verarbeitung von SVG-Dokumenten und Programmdateien unterstützt das Adobe SVG-Viewer-Plugin bei der `GetURL`-Funktion die automatische Dekompression der empfangenen Daten, so dass alle Anfragen an den Server komprimierte Daten verlangen können. Für die Integration der Toolbar und der Legende

[76] Adobe (2001)

gelten die gleichen Bedingungen wie für das dynamische Nachladen, da bei ihr auf dieselben Funktionen des Plugins zurückgegriffen wird. Durch die konsequente Kompression aller vom Client benötigten Daten lässt sich die Größe des gesamten statischen Teils der Applikation auf rund 40 KB beschränken.

Die Kompression der SVG-Daten, also der graphischen Elemente von Toolbar und Legende und der eXmap selbst, erfolgt durch ein weiteres Servlet. Dieses Servlet sorgt sowohl für die Kompression der Dateien nach dem gzip-Verfahren als auch für die Entfernung so genannter whitespaces bei den SVG-Dateien, was besonders im Hinblick auf die generische Erzeugung der Legende aus programmiertechnischen Gründen Vorteile bietet.

4.3.3.4 Die generische Legende

Die Legende der Karte, also die hierarchische Aufstellung der Struktur und der Gestaltung der Karte, wird während des Initialisierungsprozesses des Clients generiert. Dazu wird die Struktur der Detailkarte, demnach der Aufbau der Elemente der Typen `MenuEntry` und `GroupMenuEntry` unterhalb des Wurzelelementes, rekursiv analysiert. Die Legende wird gemäß den jeweiligen Attributen der auftretenden Elemente aufgebaut. Hinsichtlich der Elementenreihenfolge der jeweiligen Ebene wird die Struktur allerdings geändert, so dass das jeweils letzte Element der Ebene den ersten Eintrag der Legendenebene bildet. Diese Strukturänderung bedingt sich durch die Art der Zeichenreihenfolge von SVG-Elementen. Zuerst auftretende Elemente werden unter SVG auch zuerst auf der Zeichenfläche dargestellt. Für den Endanwender gestaltet sich allerdings eine Legende, deren oberster Eintrag dem obersten dargestellten Layer entspricht, wesentlich intuitiver.

Dieser Sachverhalt führt dazu, dass bei der Änderung der Darstellungsreihenfolge der Layer innerhalb der Legende ein Eintrag beispielsweise mit seinem Vorgänger vertauscht wird, allerdings in der Detailkarte bzw. deren DOM das jeweilige Element mit seinem Nachfolger vertauscht wird. Innerhalb der Karte reicht für die Änderung der Darstellungsreihenfolge der Tausch der jeweiligen DOM-Knoten aus. Bei der Legende hingegen müssen zusätzlich alle Koordinaten der Einträge neu berechnet werden.

5 Ausblick

Der modulare Aufbau des WMS und die Kopplung der verwendeten Komponenten des gesamten Systems mit Hilfe verschiedener Schnittstellen und Protokolle schaffen die Basis für ein breites Spektrum von Erweiterungs- und Verbesserungsmöglichkeiten des Systems.

5.1 Implementierung weiterer Layer-Arten

Gerade im Bereich der zu unterstützenden Layer-Arten und Datenbasen sind durch die Vielzahl von unterschiedlichen Formaten und Darstellungsarten Erweiterungen des WMS denkbar.

Von besonderem Interesse ist die Unterstützung von Datenbasen, die das Konzept der Simple Features von der restriktiven Verwendung von linearinterpolierten Kanten befreien. Sowohl GML3 als auch die Spatial Extension für das Datenbanksystem der Firma Oracle sehen die Verarbeitung elliptischer Formen oder auch elliptischer Abschnitte eines Koordinatenzuges vor. Die Transformation dieser Formen in eine SVG-konforme Repräsentation wird durch die Ausdrucksmächtigkeit von SVG nicht verhindert, so dass deren Implementierung die Basis für vielseitige Verbesserungen bilden kann:

- Die Genauigkeit zur Beschreibung eines features wird durch erweiterte Ausdrucksmöglichkeiten gesteigert, da eine lineare Approximation der jeweiligen geometrischen Formen nicht von Nöten ist.
- Durch nicht-lineare Approximation generalisierte features mit ursprünglich linearinterpolierten Kanten können verarbeitet werden, was zu einer erheblichen Datenmengenreduktion führen kann[77].

Neben der Erweiterung des WMS um die Unterstützung der verschiedenen Datenbasen im Allgemeinen sind zusätzliche Layer-Arten zur Darstellung spezieller Simple Features denkbar. Dazu gehören unter anderem die Erweiterung der `LabelLayer`-Klasse durch die Möglichkeit zur Ausrichtung der Beschriftungstexte an Pfaden, sowie die Entwicklung eines Symbol-Layers, der die darzustellenden Symbole einer punktuellen feature-Zusammenstellung anhand spezifizierter Sekundärattribute skaliert oder rotiert.

[77] Siehe Abschnitt 3.2.2.5.2 Generalisierung

Über die reine Unterstützung von weiteren Datenbasen hinaus gestattet die Konzeption des WMS durch die variable Übergabe aller Anfrageparameter an die jeweiligen Layerkomponenten auch die Implementierung von speziellen Layer-Arten, die auf Basis der zusätzlichen Parameter die features des Layers selektieren oder berechnen können. Beispielsweise lassen sich so unter Verwendung von Wegfindungsalgorithmen und dementsprechend aufbereiteten Datenbasen die berechneten Routen als Ergebnis einer speziellen Anfrage zur Routenberechnung direkt in die Gesamtkarte projizieren.

5.2 Erweiterung des Gestaltungsspielraums des Benutzers

Hinsichtlich der Erweiterung des benutzerseitigen Gestaltungsspielraums bieten sich zwei Konzepte an:

Auf der einen Seite bewirkt die Unterstützung der SLD WMS-Spezifikation eine standardisierte Erweiterung des WMS, die sowohl, wenn auch weniger mächtig als bei der Verwendung einer Extended-Style-Definition-Datei, die benutzerseitige Gestaltung der Karte als auch die individuelle Abfrage der Simple Features eines Layers durch die Referenzierung eines separaten WFS zulässt.

Auf der anderen Seite kann das Konzept der Extended-Style-Definition um die Möglichkeit zur Transformation der SVG-Karte durch einen als Parameter übergebenen *Streaming Transformations for XML (STX)*-Stylesheet erweitert werden, der dem jeweiligen `SAXSerializer` quasi als SAXFilter vorgelagert ist[78]. Dieses Transformationsverfahren entspricht im Gegensatz zur alternativen Verwendung eines XSLT-Stylesheets konzeptuell der konsequenten Verwendung von SAX und ermöglicht die performante, serverseitige Aufbereitung der Karte um interaktive oder graphische Elemente. So kann beispielsweise die SVG-Karte unmittelbar während des Generierungsprozesses um eine Legende oder Funktionalitäten zur Sekundärdatenexploration erweitert werden. Vornehmlich bei der serverseitigen Renderung der Karten und bei Clientumgebungen, die nicht die beim eXmap-Client evaluierten Verfahren nutzen können, bietet dieses Verfahren erhebliche Vorteile.

5.3 Implementierung weiterer Serialisierungskomponenten

Bei der Erweiterung der Serialisierungskomponenten des WMS stehen zwei Aspekte im Vordergrund.

---

[78] vgl. Becker (2003) a und b

Zum einen ist die Unterstützung weiterer Rasterdatenformate für einen vielfältigen Einsatz des WMS erstrebenswert. Unter den zahlreichen Rasterdatenformaten gilt dem im World Wide Web weiterhin häufig verwendeten GIF-Format und dem im Bereich der Geographischen Informationssysteme verwendeten Format *GeoTiff* das Hauptaugenmerk bei der Erweiterung des WMS. Das GIF-Format kann als adäquates Substitut bei Clients verwendet werden, die das PNG-Format nicht unterstützen. Das GeoTiff-Format ist eine Modifikation des Tiff-Formats, bei der im Kopfbereich der Bilddatei im Wesentlichen Informationen über das verwendete Koordinatensystem und die räumliche Ausdehnung der Rasterdaten abgelegt sind[79], und die so direkt und ohne weitere Beschreibungsdateien von Geographischen Informationssystemen verarbeitet werden kann. Für beide Formate lässt sich die auf der Batik-Komponente beruhende Implementierung zur Renderung der SVG-Karte nach dem in Abschnitt 4.2.5 angeführten Prinzip nutzen.

Zum anderen bildet die Unterstützung zusätzlicher Vektordatenformate je nach Anwendungskontext eine nützliche Erweiterung des WMS. Zu nennen sind in diesem Fall hauptsächlich der weit verbreitete de facto Standard *Portable Document Format (PDF)* der Firma Adobe und das ebenso proprietäre Format *Flash* der Firma Macromedia.

5.4 Entwicklung weiterer Clients

Bei der Entwicklung weiterer Clients sollte die Evaluierung der Nutzungspotentiale des entwickelten WMS in Verbindung mit SVG Tiny und mobilen Endgeräten, die in die Kategorie der thin clients einzuordnen sind, im Vordergrund stehen. Der WMS kann dabei als Datenbasis für verschiedene Location-based Services zum Tragen kommen, durch die Verwendung von SVG zu einer erheblichen Verringerung des Datenaufkommens beitragen und gleichzeitig die clientseitige Interaktivität der Dienste auf eine standardisierte Art und Weise erhöhen. Insbesondere das Open-Source-Projekt *TinyLine*[80], das an einer *Java 2, Micro Edition (J2ME)*-Implementierung einer SVGT-Umgebung arbeitet, stellt eine vielversprechende Basis für weitere Entwicklungen in diesem Bereich.

---

[79] vgl. Ritter und Ruth (2000)

[80] Nähere Informationen zu diesem Projekt sind unter http://www.tinyline.com zu finden

Weiterhin existieren bereits WMS 1.1.1-konforme, auf Rasterdaten basierende Clients unter anderem als frei verfügbare Web-Clients[81] oder auch als Add-On für professionelle GIS-Software wie zum Beispiel für das weit verbreitete *ArcGIS*[82] der Firma Esri. Gegebenenfalls ist eine Erweiterung der bestehenden Implementierungen um die in Abschnitt 3.2.2.4.2 angeführten Konzepte notwendig.

5.5 **Praktischer Einsatz**

Die technische Umsetzung der aufgestellten Konzepte und Anforderungen führt zu einem breit gefächerten Anwendungsspektrum des WMS im Allgemeinen und des eXmap-Clients im Speziellen. Die Stärken der Implementierung kommen insbesondere bei sich in kurzer zeitlicher Abfolge ändernden, weiträumigen und zugleich sehr detaillierten Daten mit geographischem Bezug zum Tragen. Die Verwendung von SVG ermöglicht dem Kartenautor eine standardisierte und zugleich sehr variable Gestaltung der Karten, die dem Endanwender ein Höchstmaß an Flexibilität und Interaktivität in einer performanten Umgebung bieten.

Ein Beispiel für den praktischen Einsatz der gesamten Implementierung ist das Internetprojekt der Nino Sanierungs- und Entwicklungsgesellschaft mbH[83]. Im Rahmen der Sanierung und der städtebaulichen Entwicklung des Geländes der ehemaligen Textilfabrik Nino AG wurde vom Sanierungstreuhänder der Stadt Nordhorn ein Geographisches Informationssystem aufgebaut. In dieses GIS wurden sämtliche geographiebezogenen Projekt- und Planungsdaten der beteiligten Unternehmen und Behörden integriert und dessen öffentlichkeitsrelevanter Teil wurde einem breiten Publikum durch die Referenzimplementierung des WMS zur Verfügung gestellt. Zukünftig sollen in einem geschützten Bereich sämtliche geographischen Daten zur Aufrechterhaltung eines einheitlichen Informationsstandes allen am Projekt beteiligten und autorisierten Personen angeboten werden. Im weiteren Verlauf plant die Nino SEG, den Internetauftritt um ein Facility-Management der bestehenden Gebäudesubstanz und der neu festgelegten Grundstücke auf Basis des WMS zu erweitern.

Die Fähigkeiten des WMS sind allerdings nicht auf den kommunalen oder regionalen Bereich beschränkt, sondern richten sich an alle Organisationen, Behörden und

---

[81] Ein freier WMS-Client ist unter http://sourceforge.net/projects/inlinewms/ zu finden

[82] Der Downdoad des WMS-Add-On der Firma Esri befindet sich unter http://www.esri.com/software/opengis/interopdownload.html#ogc

[83] Die Internetpräsenz des Projekts ist unter http://www.nino-seg.de zu finden

Unternehmen, die geographische Daten aktuell und an jedem Ort der Erde einem großen Publikum, ganz nach der Philosophie des OpenGIS Consortiums, zugänglich und erfahrbar machen wollen.

6 Fazit

Das Ziel der vorliegenden Diplomarbeit war die Konzeption und Implementierung eines flexiblen, offenen und verteilten Systems zur Erzeugung von dynamischen und interaktiven Karten. Konzeptuell fußt das System auf dem Reference Model des OGC. Dabei bildet zum einen das Portrayal Model die Grundlage für den Transformationsprozess der Simple Features, und das vorgestellte Service-Framework mitsamt den verabschiedeten Spezifikationen die Basis zur Verknüpfung der verschiedenen Komponenten. Besondere Beachtung finden in diesem Zusammenhang die Spezifikationen zur Beschreibung der Simple Features. Erweitert werden diese Konzepte um die Verwendung des SVG-Standards als graphische Repräsentation der geographischen Objekte, was die einheitliche Übermittlung von geographischen Informationen innerhalb der verteilten Transformationskette erst ermöglicht.

Die Spezifikation eines Web Map Services ist der Ausgangspunkt für die Konzeption und Realisierung der Serverkomponente des Systems. Über die reine Umsetzung der Vorgaben des OGC hinaus standen beim Entwurf des WMS die Erweiterbarkeit und Performance der Architektur im Vordergrund. Der modulare Aufbau des WMS ermöglicht sowohl die vielfältige Integration der heterogenen Datenbasen als auch die Ausgabe von verschiedenartigen Formaten und somit die Anbindung von thin- und medium clients. Hinsichtlich der Performance und Skalierbarkeit ist die Architektur des WMS für den produktiven Einsatz ausgerichtet und für Anwendungsszenarien mit sehr großen, sich temporal rasch ändernden Datenmengen, geringen Bandbreiten und hohem Anfrageaufkommen ausgelegt. Darüber hinaus wurde der WMS um Sicherheitsaspekte erweitert, die den Aufbau eines rollenbasierten Rechtesystems für den Zugriff auf die einzelnen Layer gestatten. Bei der Implementierung des WMS ist insbesondere die konsequente Umsetzung des SAX-Konzeptes hervorzuheben, die für einen standardisierten, performanten und speicherschonenden Transport der im SVG-Format repräsentierten geographischen Informationen sorgt. Die Konzeption der Extended-Style-Definition erlaubt weiterhin die flexible, benutzerseitige Gestaltung der resultierenden Karten.

Der entwickelte medium client erfüllt die Anforderungen der visuellen Datenexploration und bietet in Kombination mit den Konzepten des adaptiven Zoomens und dynamischen Nachladens dem Anwender einen schnellen, übersichtlichen und intuitiven Zugang zu den aufbereiteten geographischen Informationen. Aus Sicht des

Kartenautors bildet die Implementierung eine Basistechnologie zur einfachen und vollkommen flexiblen Gestaltung von dynamischen und interaktiven Karten.

Die aus der bestehenden Referenzimplementierung gewonnenen Erfahrungen versprechen einen vielfältigen Einsatz dieses stabilen, performanten, erweiterbaren und innovativen Systems.

7 Literaturverzeichnis

Adobe (2001):

Adobe: ***Current Support for SVG***, Version 3.0 (Build 76), 2001, http://www.adobe.com/svg/indepth/pdfs/CurrentSupport.pdf, am 02.02.2004

Anderson (2000):

Anderson, Richard; Birbeck, Mark; Kay, Michael; u.a.: ***XML professionell***, MITP-Verlag, Bonn, 2000

Apache (o. J.):

Apache: ***Apache HTTP SERVER PROJECT***, ohne Jahr, http://httpd.apache.org/, am 02.02.2004

Apache Jakarta Project (o. J.) a:

Apache Jakarta Project: ***Tomcat - Server Configuration Reference***, Version 5.0, ohne Jahr, http://jakarta.apache.org/tomcat/tomcat-5.0-doc/config/, am 02.02.2004

Apache Jakarta Project (o. J.) b:

Apache Jakarta Project: ***JK2***, Version 2.0, ohne Jahr, http://jakarta.apache.org/tomcat/tomcat-4.1-doc/jk2/index.html, am 02.02.2004

Apache Jakarta Project (o. J.) c:

Apache Jakarta Project: ***Tomcat FAQ - Connectors***, Version 2.0, ohne Jahr, http://jakarta.apache.org/tomcat/faq/connectors.html, am 02.02.2004

Becker (2003) a:

Becker, Oliver: ***Extended SAX Filter Processing with STX***, 2003, http://www.informatik.hu-berlin.de/~obecker/Docs/EML2003/script.html, am 02.02.2004

Becker (2003) b:

Becker, Oliver: ***Transforming XML on the Fly***, 2003, http://www.idealliance.org/papers/dx_xmle03/papers/04-02-02/04-02-02.pdf, am 02.02.2004

Brühlmeier (2002):

Brühlmeier, Tobias: ***Interaktive Karten – adaptives Zoomen mit Scalable Vector Graphics,*** 2002, http://www.carto.net/papers/tobias_bruehlmeier/2000_10_tobias_bruehlmeier_diplomarbeit_adaptives_zoomen.pdf, am 02.02.2004

Coulouris; Dollimore; Kindberg (2002):

Coulouris, George F.; Dollimore, Jean; Kindberg, Tim: ***Verteilte Systeme : Konzepte und Design***, Pearson Studium, München, 2002

Dahinden, Neumann und Winter (2003):

Dahinden, T.; Neuman, A und Winter, A. M.: ***Internet-Kartengraphik mit SVG – Werkzeuge, Techniken, Anwendungen***, In: Web.Mapping 2, Hrsg.: Asch, H. und Heerman C., Wichman, Heidelberg, 2002, Seiten. 190 - 206

Dässler (2002):

Dässler, Rolf: ***Visuelle Kommunikation mit Karten***, 2002 http://fabdp.fh-potsdam.de/daessler/paper/karten.pdf, am 02.02.2004

Cecconi und Galanda (2002):

Cecconi, A.; Galanda, M.: ***ADAPTIVE ZOOMING IN WEB CARTOGRAPHY***, 2002 http://www.svgopen.org/2002/papers/cecconi_galanda__adaptive_zooming/, am 02.02.2004

de Lange (2002):

de Lange, Norbert: ***Geoinformatik in Theorie und Praxis***, Springer, Heidelberg, 2002

Dickmann (2001):

Dickmann, Frank: ***Web-Mapping und Web-GIS***, Westermann, Braunschweig, 2001

Guttman (1984):

Guttman, Antonin: ***R-Trees: A Dynamic Index Structure for Spatial Searching,*** Proceedings, ACM SIGMOD'84, Boston, MA, June 18-21, 1984, Seiten. 47-57, http://www.es.ucsc.edu/~tonig/rtrees/rtrees.pdf, am 02.02.2004

Heiss (2003):

Heiss, Janice J.: ***Red Hat Linux 9 and Java 2 Platform, Standard Edition 1.4.2: A Winning Combination***, Sun Microsystems, 2003, http://java.sun.com/developer/technicalArticles/JavaTechandLinux/RedHat, am 02.02.2004

IBM (2002):

IBM: ***IBM DB2 Spatial Extender - User's Guide and Reference***, Version 8.0, 2002, ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/pdf/letter/db2sbe80.pdf, am 02.02.2004

Java Community Process (2003):

Java Community Process: ***JSR-000154 JavaTM Servlet 2.4 Specification***, Sun Microsystems, 2003, http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html, am 02.02.2004

Langer (2002):

Langner, Torsten: ***Verteilte Anwendungen mit Java***, Markt-und-Technik, München, 2002

Lindsey (2002):

Lindsey, Kevin: ***Transformations***, Version 1.0, 2002, http://www.kevlindev.com/tutorials/basics/transformations/toUserSpace/index.htm, am 02.02.2004

Megginson (o. J.):

Megginson, David: ***Simple API for XML***, Version 2.0, ohne Jahr, http://www.saxproject.org, am 02.02.2004

OpenGIS Consortium (o. J.):

OpenGIS Consortium: ***Vision & Mission***, ohne Jahr, http://www.opengis.org/about/?page=vision, am 02.02.2004

OpenGIS Consortium (1999) a:

OpenGIS Consortium: ***Topic 5 – Feature***, Version 4.0, 1999, http://www.opengis.org/doc/99-105r2.pdf, am 02.02.2004

OpenGIS Consortium (1999) b:

OpenGIS Consortium: ***Simple Features - SQL***, Version 1.1, 1999, http://www.opengis.org/doc/99-049.pdf, am 02.02.2004

OpenGIS Consortium (2001) a:

OpenGIS Consortium: ***Topic 1 – Feature Geometry***, Version 5.0, 2001, http://www.opengis.org/doc/01-101.pdf, am 02.02.2004

OpenGIS Consortium (2001) b:

OpenGIS Consortium: ***Geography Markup Language***, Version 2.0, 2001, http://www.opengis.org/doc/01-029.pdf, am 02.02.2004

OpenGIS Consortium (2001) c:

OpenGIS Consortium: ***Web Map Service***, Version 1.1.1, 2001, http://www.opengis.org/doc/01-068r2.pdf, am 02.02.2004

OpenGIS Consortium (2001) d:

OpenGIS Consortium: ***WMS 1.1.1 Capabilities DTD***, Version 1.1.1, 2001, http://schemas.opengis.net/wms/1.1.1/WMS_MS_Capabilities.dtd, am 02.02.2004

OpenGIS Consortium (2001) e:

OpenGIS Consortium: ***Filter Encoding***, Version 1.0, 2001, http://www.opengis.org/doc/02-059.pdf, am 02.02.2004

OpenGIS Consortium (2002) a:

OpenGIS Consortium: ***Styled Layer Descriptor***, Version 1.0, 2002, http://www.opengis.org/doc/02-070.pdf, am 02.02.2004

OpenGIS Consortium (2002) b:

OpenGIS Consortium: ***Web Feature Service***, Version 1.0, 2002, http://www.opengis.org/doc/02-058.pdf, am 02.02.2004

OpenGIS Consortium (2002) c:

OpenGIS Consortium: ***Geography Markup Language***, Version 3.1, 2002, http://www.opengis.org/doc/02-023r4.pdf, am 02.02.2004

OpenGIS Consortium (2002) d:

OpenGIS Consortium: ***WMS 1.1.1 Exceptions DTD***, Version 1.1.1, 2001, http://schemas.opengis.net/wms/1.1.1/WMS_exception_1_1_1.dtd, am 02.02.2004

OpenGIS Consortium (2003) a:

OpenGIS Consortium: ***OpenGIS Reference Model***, Version 0.1.2, 2003, http://www.opengis.org/doc/03-040.pdf, am 02.02.2004

OpenGIS Consortium (2003) b:

OpenGIS Consortium: ***Binary-XML Encoding Specification***, Version 0.08, 2003, http://www.opengis.org/doc/03-002r8.pdf, am 02.02.2004

OpenGIS Consortium (2003) c:

OpenGIS Consortium: ***WFS Schema***, Version 1.0, 2003, http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd, am 02.02.2004

Oracle (2002):

Oracle: ***Oracle Spatial***, Version 9.2, 2002, http://download-uk.oracle.com/docs/pdf/A96630_01.pdf, am 02.02.2004

Peter und Weibel (1999):

Peter, Beat; Weibel, Robert:
Using Vector and Raster-Based Techniques in Categorical Map Generalization, Third ICA Workshop on Progress in Automated Map Generalization, Ottawa, 1999, http://www.geo.unizh.ch/~beatp/ICA99_Working_Group_Paper.pdf, am 02.02.2004

MySQL (2004):

MySQL: ***MySQL Reference Manual***, Version 4.1, 2004, http://netmirror.org/mirror/mysql.com/Downloads/Manual/manual-a4.pdf, am 02.02.2004

PostGIS (o. J):

PostGIS: ***PostGIS Manual***, Version 0.8.2, ohne Jahr, http://www.postgis.org/docs/postgis.pdf, am 02.02.2004

Ritter und Ruth (2000):

Ritter, Niles; Ruth, Mike: ***GeoTIFF Format Specification***, Version 1.0, 2000, http://remotesensing.org/geotiff/spec/geotiff1.html, am 02.02.2004

Rossbach (2002):

Rossbach, Peter: ***Java Servlets und JSP mit Tomcat 4x***, Software & Support, Frankfurt, 2002

Shneiderman (1998):

Shneiderman, Ben: ***Designing the User Interface***, Third Edition Addison Wesley, Reading, Mass., 1998

Steyer (1999):

Steyer, Ralph: ***Java 2 Kompendium***, Markt-und-Technik, München, 1999,

Taylor (1995):

Taylor, George: ***LINE SIMPLIFICATION ALGORITHMS***, 1995, http://www.comp.glam.ac.uk/pages/staff/getaylor/papers/lcwin.PDF, am 02.02.2004

The Internet Society (1996):

The Internet Society: ***GZIP file format specification***, Version 4.3, 1996, http://www.ietf.org/rfc/rfc1952.txt, am 02.02.2004

The Internet Society (1999):

The Internet Society: ***Hypertext Transfer Protocol,*** Version 1.1, 2003, http://www.ietf.org/rfc/rfc2616.txt, am 02.02.2004

Vasters et al. (2001):

Vasters; Oellers; Javidi; Jung; Freiberger; DePetrillo: ***.NET-Crashkurs***, Microsoft Press, Unterschleißheim, 2001

W3C (1996):

W3C: ***PNG (Portable Network Graphics) Specification***, Version 1.0, 1996, http://www.w3.org/TR/REC-png.html, am 02.02.2004

W3C (1999):

W3C: ***XML Path Language (XPath)***, Version 1.0, 1999, http://www.w3.org/TR/xpath, am 02.02.2004

W3C (2003) a:

W3C: ***Mobile SVG Profiles: SVG Tiny and SVG Basic***, Version 1.1, 2003, http://www.w3.org/TR/SVGMobile, am 02.02.2004

W3C (2003) b:

W3C: ***Scalable Vector Graphics (SVG) 1.1 Specification***, Version 1.1, 2003, http://www.w3.org/TR/SVG11, am 02.02.2004

W3C (2003) c:

W3C: ***Document Object Model (DOM) Level 3 Core Specification***, Version 1.0, 2003, http://www.w3.org/TR/2003/CR-DOM-Level-3-Core-20031107, am 02.02.2004

W3C (2004):

W3C: ***SVG Implementations***, 2004, http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8, am 02.02.2004

Winter (2000):

Winter, Andréas. M.: ***Internetkartographie mit SVG***, 2000 http://www.carto.net/andre.mw/projects/atlas/internetkarto_svg_atlas_001206.pdf, am 02.02.2004

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Paderborn, den 28.05.2004 .....

(Jan Woltering)

Anhang A – Hinweise zur Begleit-CD

Die Begleit-CD zur vorliegenden Arbeit enthält die schriftliche Ausarbeitung als PDF-Dokument und als Microsoft-Word-Dokument, sowie ein Archiv der verwendeten Online-Quellen. Des Weiteren wurden der Quellcode der Implementierung, die Dokumentation des Quellcodes, eine Beispiel-Web-Applikation sowie eine HTML-basierte Oberfläche auf der CD abgelegt.

Die digitalen Versionen dieser Diplomarbeit befinden sich ebenso wie die index.html-Datei der HTML-Oberfläche im Wurzelverzeichnis der CD. Unter dem Verzeichnis /implementierung sind die Dokumentation des Quellcodes im docs-Verzeichnis, der Quellcode der Serverkomponente im src-Verzeichnis und das webapp-Verzeichnis zu finden. Der Quellcode des Clients liegt innerhalb des webapp-Verzeichnisses unter exmaps/client. Im Archiv der Online-Quelle unter dem Verzeichnis /quellen sind die jeweiligen Ressourcen entsprechend ihrer URI abgelegt, dabei wird der Host der URI durch das erste Verzeichnis repräsentiert. Weiterhin befinden sich im Verzeichnis /abbildungen die Originaldateien der in dieser Arbeit enthaltenen Abbildungen und im Verzeichnis /kompression alle Formate des in Kapitel 3 angeführten Beispiels.

Die HTML-Oberfläche umfasst neben den Verweisen auf die digitalen Versionen der vorliegenden Arbeit und der Quellcode-Dokumentation vornehmlich Verweise auf die verschiedenen Instanzen der bestehenden Implementierungen des Systems und die Anbieter der externen Softwarekomponenten. Bei den Instanzen des WMS sind hauptsächlich die Referenzimplementierung der Nino SEG und die Instanz für den OGC WMS 1.1.1 Conformance Test zu nennen. Die Instanz des Conformance Tests ist ebenso wie die Testergebnisse auf der CD abgelegt. Eine Online-Überprüfung ist weiterhin unter Verwendung der ebenfalls aufgeführten URL möglich.