Hochschule für
Technik und Wirtschaft
Dresden
University of Applied Sciences

Fakultät Geoinformation
Studiengang Kartographie

Integration von Tiled Map Services in Geodateninfrastrukturen

Diplomarbeit zur Erlangung
des akademischen Grades
Diplom-Ingenieur für Kartographie (FH)

Tag der Einreichung: 27. September 2010

Name: Thomas Jurk
geboren am: 28.06.1982

1. Gutachter: Prof. Dr.-Ing. Frank Schwarzbach
2. Gutachter: Dipl.-Ing. (FH) André Müller

Danksagung

Für die fachliche Anregung und Unterstützung zu dieser Diplomarbeit bedanke ich mich bei meinen Gutachtern, Herrn Prof. Dr.-Ing. Frank Schwarzbach und Dipl.-Ing. (FH) André Müller.

Ein weiterer Dank geht an die Laboringenieure des Bereichs Geoinformatik, die mir in der praktischen Umsetzung dieser Arbeit stets engagiert geholfen haben.

Ganz besonders danke ich meiner Freundin Martina für den Beistand und vor allem auch die Geduld während der gesamten Studienzeit.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit mit dem Titel „Integration von Tiled Map Services in Geodateninfrastrukturen“ selbständig verfasst und ausschließlich die angegebenen Literaturquellen verwendet habe.

Dresden, 27.09.2010

Thomas Jurk

Inhaltsverzeichnis

Danksagung **I**

Eidesstattliche Erklärung **II**

Inhaltsverzeichnis **III**

Einleitung **1**

1 Überblick **3**

2 Grundlagen **4**

2.1 Geodateninfrastrukturen....4

2.1.1 Web Services....4

2.1.2 Standardisierungen....5

2.2 REST - Representational State Transfer....7

2.3 Klassische Kartendienste....10

2.4 Kachelung....12

2.4.1 Technologischer Hintergrund....13

2.5 Standards....20

2.5.1 WMS Tiling Client Recommendation....21

2.5.2 Tile Map Service Specification....24

2.5.3 OpenGis Web Map Tile Service Implementation Standard....26

2.5.4 KML....37

3 Implementierung **39**

3.1 Zielsetzung....39

3.2 Anforderungsanalyse....40

3.2.1 Schnittstellen....40

3.2.2 Plattform....42

3.2.3 Hardwareressourcen....42

3.3 Bestandsanalyse....43

3.3.1 Geowebcache....44

3.3.2 TileCache....47

3.3.3 ArcGIS Server....49

3.3.4 MapProxy....50

3.3.5 mod_tile....52

3.3.6 MapServer....53
3.3.7 Schlussfolgerung....53
3.4 Installation und Konfiguration....54
3.4.1 Java SE Runtime Environment (JRE)....54
3.4.2 Apache Tomcat....54
3.4.3 Geowebcache....56
3.5 Erkenntnisse und Probleme....61
3.5.1 Fehldarstellungen....61
3.5.2 Grafikformate....64
3.5.3 Speichernutzung....65
3.5.4 Einbindung der GeoSN Geodatendienste....66
3.6 Last- und Performancetests....67
3.6.1 Definition der Testbedingungen....67
3.6.2 Durchführung....68
3.6.3 Auswertung....71

4 Integration in die Basiskomponente Geodaten Sachsen 74
4.1 Basiskomponente Geodaten Sachsen....74
4.1.1 Fachliche Sicht....75
4.1.2 Funktionale Sicht....76

5 Zusammenfassung und Ausblick 79
5.1 Darstellung der Ergebnisse....79
5.2 Ausblick....80

Listingverzeichnis 82

Tabellenverzeichnis 83

Abbildungsverzeichnis 84

Literaturverzeichnis 87

Anlagen 91
A Legenden zur UML-Notation....92
B Geowebcache Konfiguration....95
C Tile Service Profil - Sachsen....101
D Digitale Anlage....107

Einleitung

Die Anforderungen an Kartendienste haben sich in den letzten Jahren stark gewandelt. Vorallem Unternehmen wie Google, Yahoo und Microsoft erzeugten mit ihren interaktiven Kartenanwendungen bei dem Nutzer höhere Erwartungen an Geschwindigkeit und Bedienbarkeit. Die einfache Integration der Dienste dieser Anbieter in andere Web-Anwendungen führte zu ihrer starken Verbreitung. So werden heute im Internet die verschiedensten Inhalte mit interaktiven Karten verknüpft. Möglich wurde diese Entwicklung nur durch den Einsatz besonderer technologischer Konzepte.

„Die Welt als Bilderpuzzle“ - so kann die Technologie umschrieben werden, die in einem kachelnden Kartendienst steckt. Dienste solcher Art speichern Karten unterteilt in einzelne gleich große Bilder. Anwendungen, die auf diese Dienste zugreifen, kombinieren die Kacheln wieder zu einem Kartenbild. Nutzerinteraktionen, wie das Verschieben des Kartenausschnittes, erfordern so kein komplettes Neuladen. Zentraler Gedanke hinter dem Kacheln von Karten ist also die *Wiederverwendung*. In Verbindung mit modernen Web-Mapping-Anwendungen ermöglichen kachelnde Kartendienste eine schnelle Darstellung und einen hohen Grad an Interaktivität.

Die Vorteile klassischer Kartendienste liegen vorallem in der individuellen Gestaltung der Karten durch den Nutzer. Die Nutzer kachelnder Kartendienste haben keine Kontrolle über die graphische Präsentation der Karten. Kacheln in einheitlichen Projektionen und Maßstäben sollen das das Konzept der Wiederverwendung erweitern und so die Kartendarstellung für einen großen Nutzerkreis beschleunigen.

Wie sich das technologische Konzept von kachelnden Kartendiensten in vorhandene Infrastrukturen einordnet, ist Teil der Untersuchungen dieser Diplomarbeit. Als Ergebnis soll ein entsprechender Dienst für den produktiven Einsatz im Labor Geoinformatik der Hochschule für Technik und Wirtschaft Dresden implementiert werden.

Hinweis:

Für die Darstellung einiger Diagramme in dieser Arbeit wurde die Unified Modeling Language (UML) verwendet. Die Bedeutung der UML-Notationen wird durch Legenden in Anhang A verdeutlicht.

1 Überblick

Das folgende Kapitel 2 befasst sich mit den Grundlagen von Tiled Map Services. Dabei werden die für die Verständlichkeit notwendigen technologischen Konzepte, Standards und Technologien näher vorgestellt. Darüber hinaus werden grundlegende Begriffe und Sachverhalte, die im Folgenden mehrfach verwendet werden, erklärt.

Das Kapitel 3 dokumentiert den praktischen Teil der Diplomarbeit. Der gesamte Prozess der Integration eines Tiled Map Services im Labor Geoinformatik, von der Anforderungsanalyse bis zur Realisierung wird beschrieben. Last- und Performancetests sollen abschließend Schlussfolgerungen auf die Vorteile von Tiled Map Services zulassen.

In Kapitel 4 wird mit Bezug auf die Basiskomponente Geodaten Sachsen die Integration eines Tiled Map Services in Geodateninfrastrukturen theoretisch untersucht. Dabei werden die zu erwartenden Anpassungen sowie technologischen Anforderungen diskutiert.

Kapitel 5 gibt abschließend eine Zusammenfassung der Diplomarbeit und einen Ausblick auf zukünftige Entwicklungen.

2 Grundlagen

2.1 Geodateninfrastrukturen

Geoinformationssysteme (GIS) galten lange Zeit als typische Monolith-Systeme, deren Planung, Implementierung und Bedienung weitreichende Kenntnisse von Entwickler und Nutzer voraussetzten. Der strukturelle Aufbau des Internets und die dadurch gestellten neuen Anforderungen bedingten ein Aufspalten dieser monolithischen Strukturen in einzelne funktionale Einheiten und die Implementierung dieser als verteiltes System. Die aus diesen Entwicklungen entstandenen Geodateninfrastrukturen (GDI) verarbeiten räumlich verteilt vorliegende Datenbestände und bereiten diese vorwiegend zu Web-optimierten Informationen auf.

Geodateninfrastrukturen beschreiben die Gesamtheit aller Daten und damit verbundenen Systemkomponenten. Im Vergleich zu Systeminfrastrukturen der klassischen Informationstechnik stellen Geodateninfrastrukturen nichts anderes als eine Service-orientierte Architektur (SOA) für Daten mit Raumbezug dar. [KIEHLE, 2006, S. 55]

Neben dieser technologischen Sicht auf Geodateninfrastrukturen spielt der Begriff GDI auch auf organisatorischer und rechtlicher Ebene eine Rolle. Die Betrachtungsweise ändert allerdings nicht das Ziel, das als *Erleichterung des Zugangs zu Geodaten* formuliert werden kann [GDI-DE, 2007]. Relevant für diese Diplomarbeit ist überwiegend die technologische Sicht auf dieses Thema und wird nachfolgend genauer betrachtet.

2.1.1 Web Services

Web Services implementieren das verteilte System in Geodateninfrastrukturen. Sie bilden eine Abstraktionsebene der Anwendungslogik, die ausschließlich auf offenen und hersteller-unabhängigen Standards basiert. Dadurch besteht eine Plattformunabhängigkeit bei der Kommunikation zwischen einzelnen Systemkomponenten. Anders als beim bisherigen Web-Ansatz, bei dem der

Nutzer Daten und Funktionalität vom Webserver abruft, erfolgt hierbei eine direkte Kommunikation, die als Maschine-zu-Maschine-Kommunikation bezeichnet werden kann. Die wichtigsten Web Service-Protokolle und Standards basieren auf XML und verwenden HTTP (Hypertext Transfer Protocol) als Transportprotokoll. Für fast alle Programmiersprachen stehen Implementationen für Web Service-Protokolle zur Verfügung. Darüber hinaus lässt sich der breite Einsatz dieser Technologie in der Software-Industrie auch durch die einfache Handhabung und hohe Verfügbarkeit begründen. [HALÖ, 2004, S. 12F]

Gerade im GIS-Bereich gibt es viele unterschiedliche Hersteller mit proprietären Datenformaten und Technologien, deren raumbezogene Datenbestände mit Hilfe der Web Services unabhängig und flexibel verarbeitet werden können.

2.1.2 Standardisierungen

Geodateninfrastrukturen wurden einleitend mit Service-orientierten Architekturen (SOA) verglichen.

> *„Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenunabhängige Nutzung und Wiederverwendung ermöglicht.“* [MELZER, 2007, S. 11]

Das OGC hat sich die Abbildung der Prinzipien einer SOA auf den GeoIT-Sektor als Aufgabe gestellt. Neben einer Vielzahl von Standards veröffentlicht diese gemeinnützige Organisation vor allem abstrakte Referenzmodelle, Architekturmodelle, Anwendungsstudien und Vorschläge zur Integration von Geodaten in IT-Systeme. Die Mitarbeit von GIS-Herstellern sowie von Universitäten und bedeutenden Unternehmen der Informationstechnik aus geofernen Branchen unterstreicht die Relevanz der Veröffentlichungen durch das OGC.

Das OGC Reference Model (ORM)[1] beschreibt auf welcher konzeptuellen Basis OpenGis[2] Web Services entworfen werden sollen. In OGC-ORM, 2008 werden wichtige grundlegende Merkmale definiert:

1 http://www.opengeospatial.org/standards/orm

2 OpenGIS ist der Markenname unter dem Standards des OGC veröffentlicht werden

- Verwendung offener Standards. Die Nutzung von Diensten unbekannter Anbieter soll dadurch garantiert werden.
- Verzeichnisdienst oder *Repository,* in dem die zur Verfügung stehenden Dienste registriert werden.
- Lose Kupplung (*loose coupling*) der Dienste, d. h. Dienste werden von Anwendungen oder anderen Diensten bei Bedarf dynamisch gesucht, gefunden und eingebunden.

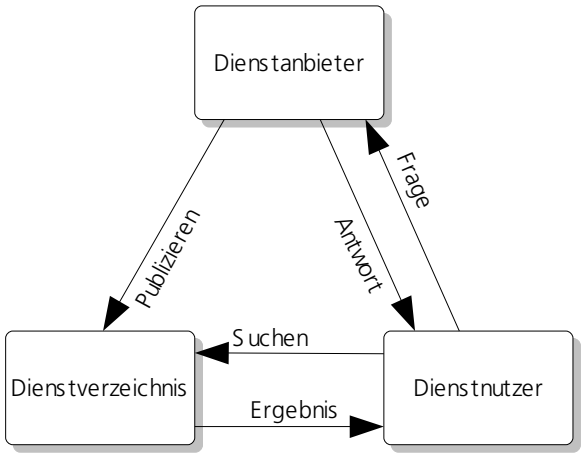

Abbildung 2.1: Die Service-orientierte Architektur

Durch die lose Kupplung der Dienste entstehen die typischen Rollen einer SOA. Abbildung 2.1 zeigt die Rollenverteilung und die Aktionen zwischen den Rollen.

- Der Dienstanbieter veröffentlicht seine sogenannte *Service Description* in maschinenlesbarer Form im Dienstverzeichnis und bedient Serviceanfragen der Dienstnutzer.
- Der Dienstnutzer findet über das Dienstverzeichnis Dienstanbieter und benutzt die als Suchergebnis erhaltene *Service Description* als Anleitung für den Zugriff auf den Dienstanbieter.

Die Aktionen in dieser verteilten Umgebung sind als *publish–find–bind* Paradigma bekannt.

Die in dem OGC Reference Model gemachten konzeptuellen Vorgaben finden derzeit nicht in allen OGC Web Services Anwendung. Jedoch stellen laut MELZER, 2007 die Standards SOAP[3],

3 Protokoll für den Austausch XML-basierter Nachrichten in Computer-Netzwerken

WSDL[4] und UDDI[5] Umsetzungen der drei wichtigsten Komponenten einer SOA dar, werden aber erst in der im April 2010 veröffentlichten OGC Web Service Common Implementation Specification Version 2.0.0[6] empfohlen.

Name	Version	Datum
Web Map Service (WMS)	1.3.0	20.01.2004
Web Feature Service (WFS)	1.1.0	03.05.2005
Web Coverage Service (WCS)	1.1.2	19.03.2008
Web Map Tile Service (WMTS)	1.0.0	06.04.2010

Tabelle 2.1: Raumbezogene OGC Web Services[7]

Tabelle 2.1 gibt einen Überblick über die aktuellen OGC Web Service Standards für raumbezogene Daten. Der OpenGIS Web Map Tile Service (WMTS) Implementation Standard wurde auf Basis der OGC Web Service Common Implementation Specification Version 1.1.0 (OGC-OWS, 2007) entworfen. Im Gegensatz zu den anderen in Tabelle 2.1 aufgeführten Web Services enthält dieser erstmals die Empfehlung für das SOAP-Protokoll für alle Operationen des Dienstes. Für ältere Standards gibt es die Möglichkeit des sogenannten SOAP-Wrapping, welches in DAKALB, 2009 beschrieben wird.

Der OpenGIS Web Map Tile Service Implementation Standard wird im Kapitel 2.5.3 näher vorgestellt.

2.2 REST - Representational State Transfer

Mit REST wird ein Softwarearchitekturstil für verteilte Informationssysteme beschrieben, dessen Ursprünge auf Webstandards, insbesondere HTTP, zurückgehen. Eingeführt wurde der Begriff von Roy T. Fielding, Kernentwickler vieler Webstandards und ehemaliger Vorsitzender der Apache Software Foundation, in seiner Dissertation „Architectural Styles and the Design of Network-based Software Architectures“ (FIELDING, 2000). Aufgrund der besonderen Bedeutung für die hier vorgestellten Standards werden die grundlegenden Prinzipien einer REST-Architektur vorgestellt.

4 Web Services Description Language – Beschreibungssprache für Web Services auf Basis von XML
5 Universal Description, Discovery and Integration – standardisierter Verzeichnisdienst
6 http://www.opengeospatial.org/standards/common
7 Stand: Juli 2010

REST ist kein Web Service Standard, sondern legt Vorschriften in Form von Architektur-Constraints fest. Die wichtigsten Vorgaben dieser Architektur sind:

- Client-Server-Umgebung (lose Kupplung)
- Zustandslosigkeit (der Client verwaltet ggf. den Sitzungszustand)
- Cachefähigkeit

Abstrakt werden durch FIELDING, 2000 Komponenten definiert, die über Konnektoren Datenelemente in einem Netzwerk austauschen. Datenelemente oder Ressourcen sind also der zentrale Gegenstand dieser Architektur.

REST definiert eine Ressource als beliebige Information (Dokument, Bild, Dienst), deren eindeutige Identifizierung über eine URI[8] und Übertragung in unterschiedlichen Repräsentationen (Datenformate, Sprachen) erfolgt. Abfragen und Manipulationen einer Ressource beschränken sich auf einen festgelegten Satz von Methoden (HTTP Methoden, [RFC2616, 1999, SEKTION 9]). Aufgrund der Relevanz von Ressourcen werden REST-Architekturen oft auch als ROA (Resource-oriented Architecture) bezeichnet.

URI	Semantik der Ressource	HTTP Methoden
/pictures	Alle Bilder	GET, POST
/pictures/{pid}	Ein spezielles Bild	GET, PUT, DELETE
/pictures/{pid}/comments	Alle Kommentare eines Bildes	GET, POST
/pictures/{pid}/comments/{cid}	Ein Kommentar zu einem Bild	GET, PUT, DELETE

Tabelle 2.2: Beispiel für eine Bildergallerie nach REST-Architektur

In Tabelle 2.2 werden anhand eines einfachen REST-konformen (RESTful) Dienstes, Ressourcen und zugehörige Methoden aufgelistet. Bei der Implementation eines solchen Dienstes müssen die notwendigen Ressourcen und deren Beziehungen zueinander bestimmt werden. Die Bedeutung der HTTP Methoden sind nach [RFC2616, 1999, SEKTION 9] allgemein:

8 Uniform Resource Identifier (http://tools.ietf.org/rfc/rfc1630.txt)

- **GET**

  - Abfrage der Informationen einer Ressource
  - keine Änderung des Serverzustands

- **POST**

  - Erstellen einer neuen, untergeordneten Ressource

- **PUT**

  - Speicherung einer mitgeschickten Information an einer spezifischen URI

- **DELETE**

  - Löschen der Ressource

Die Repräsentation einer Ressource ist nicht auf eine Darstellungsvariante festgelegt. HTTP ermöglicht über Content Neogation ([RFC2616, 1999, SEKTION 14.1]) die Darstellung ein und der selben Ressource in unterschiedlichen Formaten oder Sprachen. So kann die clientspezifische Ausgabe ermöglichen, dass für einen Browser die Ressource `/pictures` (Tabelle 2.2) als HTML, für einen programmatischen Client als XML übertragen wird.

Die Verbindung von identifizierbaren Ressourcen erfolgt wie im Web über Verknüpfungen (Links). Dieses standardisierte Identifikationsverfahren stellt sicher, dass sich die verknüpften Ressourcen in einem anderen Prozess oder gar auf einem anderen Rechner befinden können. So könnte die HTML-Repräsentation der Ressource `/pictures/{pid}` für das kontrete Bild auf einen anderen Server verlinken. [TILKOV, 2009]

Auch wenn REST hier unabhängig von Geodateninfrastrukturen beschrieben wird, könnte das Architekturprinzip in Zukunft auf diesem Gebiet eine wichtige Rolle spielen. Gerade in Umgebungen die hauptsächlich auf Informationsaustausch orientiert sind, hat REST durch seine Architekturvorgaben entscheidende Vorteile. Auch das Bilden von sogenannten Mashups[9], das durch die generische Schnittstelle vereinfacht wird, kann die Stellung von ROA neben Service-orientierten Architekturen begünstigen.

9 Kombination bereits bestehender Webinhalte (web application hybrid)

2.3 Klassische Kartendienste

Die Darstellung von raumbezogenen Daten in Form von Karten ist wohl eine der häufigsten Verwendungsformen. Mit klassischen (Web-) Kartendiensten, wie z. B. dem WMS, kann jeder Nutzer individuell auf einen beliebigen Kartenausschnitt mit frei wählbaren Maßstab zugreifen. Die vom Endverbraucher (Client), in dem Fall ein Webbrowser oder ein Desktop-GIS, angeforderte Karte wird über das verwendete Netzwerkprotokoll in einem Schritt übertragen. Ein WMS ermöglicht darüber hinaus die Kombination verschiedener Layer und die Gestaltung der Karte mittels Styled Layer Descriptor (SLD)[10]. Für das fertige Kartenbild dienen weit verbreitete Rastergrafikformate als Austauschdatenformat zwischen Server und Client.

Das Ablaufmodell für die Kartenerzeugung nach Adrian Cuthbert (Abbildung 2.2) veranschaulicht den Prozess unter Verwendung der OGC Standards. Bis das individuelle Resultat einer Anfrage an den Web Map Service beim Client angezeigt wird, durchlaufen die Quelldaten mehrere Schritte in einer u. U. verteilten Umgebung. Zu den in diesem Modell beschriebenen Schritten gehören:

- Selektieren der Geometrie- und Sachdaten aus der Datenquelle. Die selektierten Geometrieobjekte werden Features genannt.
- Umwandlung der Geometriedaten unter Anwendung von Darstellungsvorschriften in visuelle Objekte im Display Element Generator.
- Generierung (*rendering*) einer Pixelgrafik aus der visuellen Beschreibung der Karte.
- Anzeige der Grafik auf dem Endgerät (z. B. Webbrowser).

10 http://www.opengeospatial.org/standards/sld

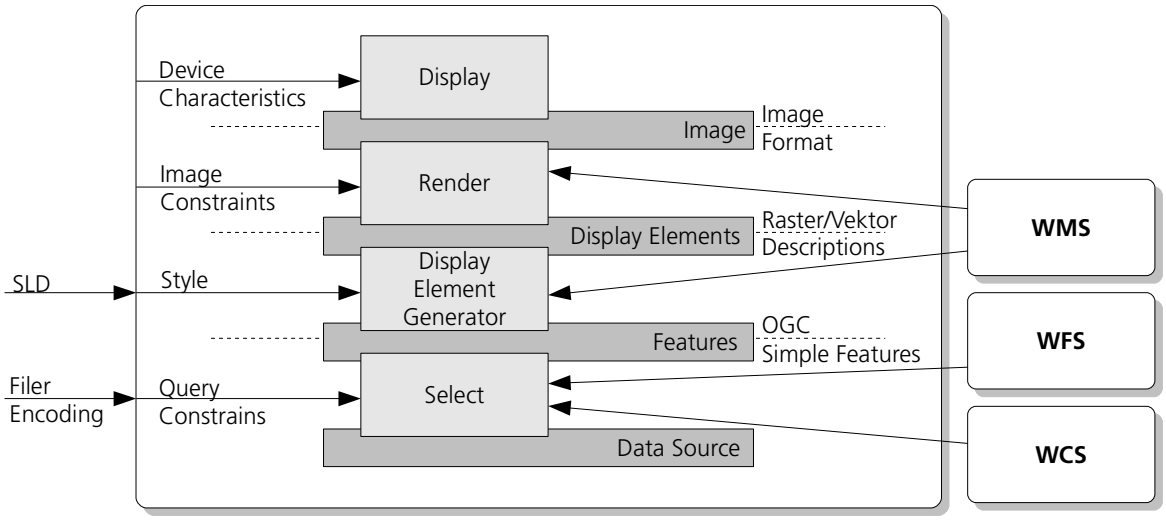

Abbildung 2.2: Portrayal-Modell nach Adrian Cuthbert [DOCUTH, 1998]

Auch wenn sich Quelldaten und Kartendienst auf dem selben physischen Server befinden, beansprucht das dynamische Rendern jedes Kartenausschnittes einen gewissen Anteil an Bildverarbeitungskapazität. Die Möglichkeit des Kaskadieren, d. h. das Verketten mehrerer WMS, erhöht die Anzahl der Schritte bis zum gerenderten Ergebnis. Die Individualität der von klassischen Kartendiensten ausgelieferten Karten und die zum Teil komplexen Prozesse, die bei deren Erzeugung stattfinden, führen an den Diensten zugrunde liegenden Hardwareressourcen zu erhöhten Anforderungen. So skaliert die begrenzte Bildverarbeitungskapazität mit der Anzahl der zugreifenden Clients. Es können dadurch erhöhte Wartezeiten entstehen, die die Nutzung des Dienstes negativ beeinflussen (siehe Kapitel 3.6).

Die Geschwindigkeit, also die Antwortzeit eines Kartendienstes, hat großen Einfluss auf dessen Benutzerfreundlichkeit, welche besonders bei kommerziellen Angeboten im Vordergrund steht. Aber auch die Weiterentwicklung der Web-Mapping-Anwendungen zu Rich Internet Applications (RIA), bei denen Technologien wie *Javascript DOM* oder *Flash* zum Einsatz kommen, erhöht die Anzahl der Zugriffe auf die Dienste. Die Interaktivität, die durch RIA eingeführt wird, mit Funktionen wie dynamisches Verschieben (*panning*) und Zoomen (*zooming*) verstärken diesen Effekt. Durch einfachere Bedienbarkeit wird gleichzeitig einem größeren Publikum Zugang zu Web-Mapping-Anwendungen eröffnet.

Die gestiegende Erwartungshaltung der Nutzer an die Performance und Bedienbarkeit von Web-Anwendungen führte zur Einführung erster kommerzieller Lösungen wie Google Maps, Yahoo Maps oder Microsoft Bing Maps (früher Live Maps). Durch serverseitiges Vorprozessieren des Kartenmaterials, auch als Pre-Rendering, Kachelung, Tiling oder Caching bezeichnet, entsteht der entscheidende Geschwindigkeitsvorteil. [SEIDEL, 2009]

2.4 Kachelung

Das Konzept der Kachelung besteht darin, dass nicht mehr bis auf die Datenquelle zurückgegriffen, sondern ein schon vorher fertig gerendertes Bild ausgeliefert wird. Um dennoch einen möglichst individuellen Zugriff zu ermöglichen, wird ein Gerüst aus einzelnen Kacheln zwischengespeichert. Der Kartenclient hat Zugriff auf jede einzelne Kachel und kann so das entgültige Kartenbild zusammenfügen. Ein Verschieben der Kartenansicht bewirkt nicht ein komplettes Neuladen, sondern nur ein Nachladen der entsprechenden Kacheln.

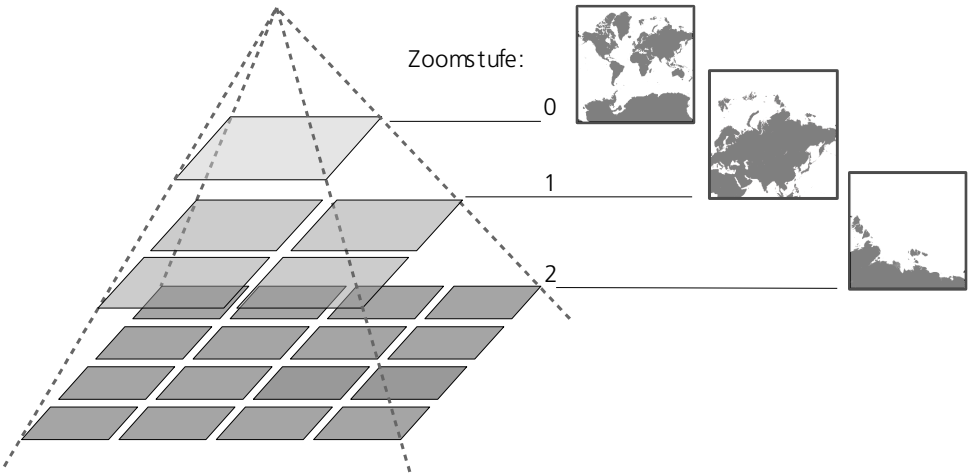

Abbildung 2.3: Möglicher Aufbau eines gekachelten Datenbestandes

Abbildung 2.3 veranschaulicht vereinfacht den möglichen Aufbau der gespeicherten Datenstruktur. Bei steigender Zoomstufe und konstanter Kachelgröße vergrößert sich der Maßstab und dadurch die Anzahl der notwendigen Kacheln. Somit wird die gleiche Fläche einer Kachel in der ersten Zoomstufe durch mehrere Kacheln in der zweiten Zoomstufe dargestellt[11]. In der Abbildung wird der ideale und auch am meisten verwendete Fall dieser Datenstuktur gezeigt. Durch die Erhöhung der Zoomstufe um 1 verdoppelt sich der Maßstab und dem entsprechend vervierfacht sich die Anzahl der Kacheln. Die Organisation der Kacheln kann mit dem aus der Informatik bekannten Quadtree verglichen werden. Dieser beschreibt einen gerichteten Baum, bei dem jeweils 4 Kinder aus einem inneren Knoten entstehen. In Abbildung 2.4 wird dieser Zusammenhang mit der in Abbildung 2.3 gezeigten Datenstruktur verdeutlicht.

11 Einige Implementation erlauben auch unterschiedliche Kachelgrößen

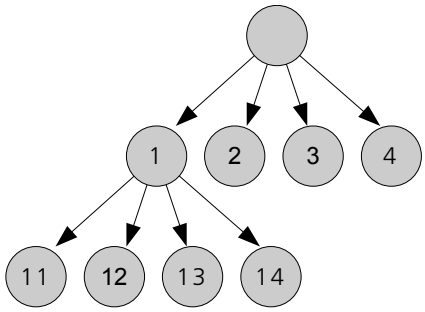

Abbildung 2.4: Quadtree als gerichteter Baum

Die von Google Maps eingeführte Datenstruktur entspricht dem eines Quadtree. Im kleinsten Maßstab deckt eine Kachel mit 256x256 Pixel die Fläche der Erde ab. Die verwendete Projektion ist EPSG:3857[12] (WGS84 / Pseudo-Mercator). Durch diese Festlegungen ergibt sich eine spezifische Maßstabsreihe. Viele kommerzielle und freie Kartendienstanbieter haben ihre Datenstruktur nach den Vorgaben von Google aufgebaut. Ein entsprechendes Profil ist durch OGC-WMTS, 2010, ANNEX E.4 standardisiert.

2.4.1 Technologischer Hintergrund

In der Spezifikation 6.0 des Grafikformats TIFF wird Tiling für den Einsatz bei hochauflösenden Rasterdaten beschrieben. Das Kacheln selbst erfolgt nur auf dem Originalbild bei einer Kachelgröße in Pixel, die ein Vielfaches von 16 betragen soll [ADOBE-TIFF, 1992, SEKTION 15]. Das auf der TIFF-Spezifikation basierende GeoTIFF[13] Format für georeferenzierte Rasterdaten erlaubt das zusätzliche Speichern einer Bildpyramide. Diese Funktion wird z. B. von dem verbreiteten Geospatial Data Abstraction Library[14] (GDAL) implementiert. Die Stufen der Pyramide werden als sogenannte Overviews direkt in der GeoTIFF-Datei gespeichert. GDAL speichert GeoTIFF-Dateien mit einer voreingestellten Kachelgröße von 256 x 256 Pixel [GEOTIFF, 2000; GDAL-GEOTIFF, 2010]. Die Kacheln und Overviews reduzieren beim Zugriff auf hochauflösende Rasterdaten den notwendigen Arbeitsspeicher, da z. B. bei der Anzeige auf Bildschirmen nur ein bestimmter Ausschnitt in einer bestimmten Auflösung geladen werden muss. Moderne Kartendienste wie der UMN MapServer verwenden ebenfalls das GeoTIFF Format für den performanten Zugriff auf Rasterdaten [MAPSRV-RASTER, 2010]. Das Konzept, Kacheln einzeln zu speichern und durch die Kartenclients abzurufen, ist ein nachvollziehbarer Schritt, der vorallem mit den Gegebenheiten des Internets, also dessen Struktur und Protokollen, begründet werden kann. Durch das direkte

12 http://www.epsg-registry.org/export.htm?gml=urn:ogc:def:crs:EPSG::3857
13 http://www.remotesensing.org/geotiff/spec/geotiffhome.html
14 http://www.gdal.org/

Ansprechen der Kacheln können Technologien zum Einsatz kommen, die sich im Internet über Jahre auf bandbreitenintensiven und hochfrequentierten Portalen bewährt haben. Zu den wichtigesten Technologien zählen:

- HTTP Caching
- Server Side Caching
- Lastverteilung

2.4.1.1 HTTP Caching

Wenn HTTP/1.1[15] als Protokoll zur Übertragung der Daten zwischen dem Kartenclient und Server dient, kann HTTP Caching helfen Zugriffszeiten zu verringern. Der HTTP-Standard sieht vor, dass jede Anfrage (*Request*) von einem Client und jede Antwort (*Response*) von einem Server mittels sogenannter Header-Information beschrieben wird. Diese Header können neben der Request-Methode oder dem Status Code auch Informationen und Regeln enthalten, die es dem Client erlauben die Daten lokal zwischenzuspeichern. Darauf aufbauend besteht auch die Möglichkeit zur Überprüfung der Aktualität lokal gespeicherter Daten.

Abbildung 2.5 zeigt das Sequenzdiagramm eines einfachen HTTP-GET-Requests, wie er z. B. beim Zugriff auf Daten eines Webservers erfolgt. Der Server antwortet entsprechend des Requests mit einer HTTP-Response. Sind Header in der HTTP-Response enthalten, die das Zwischenspeichern der Daten erlauben, legt der Client sie lokal im Arbeitsspeicher oder auf der Festplatte ab. Weitere Zugriffe auf die Daten können jetzt ohne weitere Client-Server-Kommunikation durch den Cache bedient werden (Abbildung 2.6). Zusätzliche HTTP-Response-Header können eine Gültigkeitsdauer für die Daten im Cache bestimmen. [RFC2616, 1999, SEKTION 13]

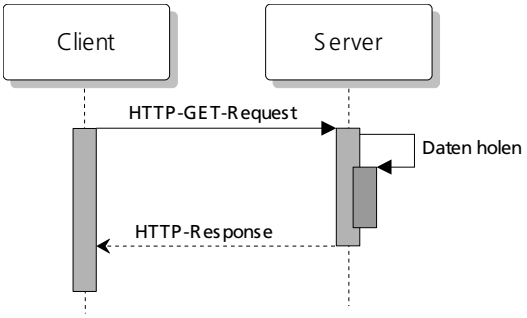

Abbildung 2.5: Einfaches HTTP-Request-Response

15 http://www.w3.org/Protocols/rfc2616/rfc2616.html

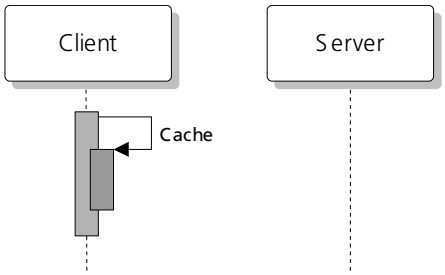

Abbildung 2.6: Daten liegen im Client Cache

Für kachelnde Kartendienste ist die Vorhersage für die Gültigkeitsdauer der Daten relativ schwierig. Eine große Zeitspanne könnte zu veralteten Kartenausschnitten führen, eine Kurze zum Verlust des Geschwindigkeitsvorteils. Mit der Validierung der gecachten Daten beim Webserver bietet HTTP/1.1 ein Verfahren an, das mit geringem Einsatz von Netzwerkkapazitäten die Aktualität der Daten garantiert. Auf noch nicht im Client Cache gespeicherte Daten antwortet der Server immer mit einem Zeitstempel oder einer Prüfsumme der Daten im Response-Header. Der Client setzt bei erneuten Zugriff diese Informationen unverändert in den Request-Header. Dadurch kann ein Abgleich beim Server erfolgen. Sind die Informationen unverändert, anwortet der Server mit einem bestimmten Status-Code und leerer Nachricht. Das Sequenzdiagramm in Abbildung 2.7 zeigt den Ablauf für noch aktuelle Daten im Cache. Die Abbildung 2.5 entspricht dem Ablauf für veraltete Daten. [RFC2616, 1999, SEKTION 14.9]

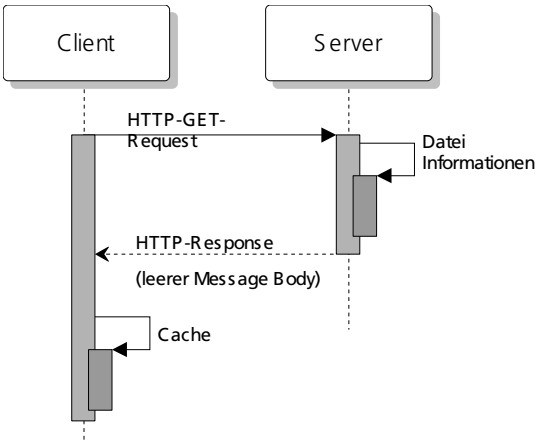

Abbildung 2.7: Validierung der Daten im Client Cache

Die Kombination aus kurzer Gültigkeitsdauer und anschließender Validierung ermöglicht einen Kompromiss zwischen schnellen Zugriffszeiten und Aktualität.

Der Einsatz von HTTP Caching reduziert beim Client die Anzahl der Zugriffe auf die eventuell begrenzten Netzwerkkapazitäten und bewirkt dadurch eine schnellere Anzeige bei mehreren Zugriffen auf die selbe Ressource. Beim Server führt der konsequente Einsatz von HTTP Caching zur effizienteren Nutzung der zur Verfügung stehenden Bandbreite und Hardwareressourcen. [SOUDERS, 2008, S. 27FF]

Die in den Abbildung 2.5 bis 2.7 dargestellte direkte Client-Server-Kommunikation kann auch durch Web Proxies erweitert werden. Web Proxies ermöglichen in geschlossenen Netzwerken den Zugang zum Internet, dienen aber meist auch als Web Caches. Das Cachen in einem Web Proxy erfolgt wie beim Client nach den Regeln der HTTP-Header. Jeder Zugriff von Clients auf Ressourcen eines Servers im Internet wird über den Web Proxy geleitet. Hat der Proxy die Ressource zwischengespeichert, antwortet er stellvertretend für den Server (Abbildung 2.8). Vorallem durch den größeren Nutzerkreis wird der Web Cache gegenüber Client Cache effektiver genutzt. Aber auch die größere räumliche Nähe des Proxies im Vergleich zu der der Server außerhalb des Netzwerkes bringt Geschwindigkeitsvorteile.

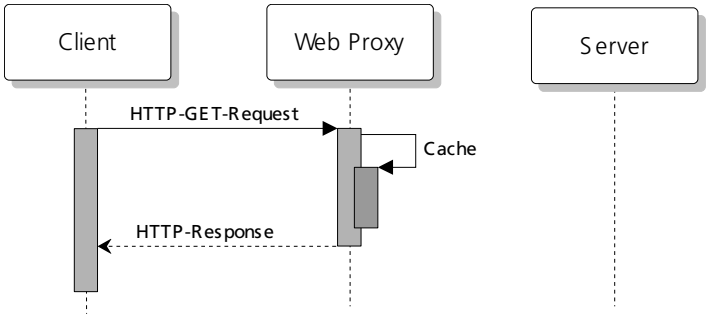

Abbildung 2.8: Web Proxy antwortet stellvertretend für den Server

Web Proxies können noch weitere Funktionen in Netzwerken erfüllen. Auch viele Implementationen von Web Caches gehen weit über das einfache Zwischenspeichern hinaus. Relevant sind vorallem die Vorteile, die Web Caches oder allgemein HTTP Caching mit sich bringen. Klassische Kartendienste mit dynamischen, nicht wiederverwendbaren Bilddaten können von diesen Technologien nicht profitieren.

2.4.1.2 Server Side Caching

Server Side Caching ist ein Verfahren, dynamische Daten über eine bestimmte Zeitspanne auf dem Server zu speichern und diese statischen Daten statt der dynamischen an den Client zu senden. Besonders für nicht individuelle, allgemeine Daten ist Server Side Caching geeignet. Da jeder Zugriff auf eine dynamische Ressource einen bestimmten Anteil an Verarbeitungskapazität

beansprucht, kann das Caching die begrenzten Kapazitäten schonen oder eine höhere Zugriffsrate erlauben. Beim Erstellen und Verwalten eines Caches müssen folgende zentrale Fragen beantwortet werden:

- Wann werden die Cache-Einträge erstellt?
- Wie groß darf der Cache werden?
- Welche Daten werden bei einem vollen Cache gelöscht?
- Wann und wie wird der Cache aktualisiert?

Nachfolgend werden diese Fragen in Bezug auf kachelnde Kartendienste beantwortet.

Die Erstellung der kompletten Bildpyramide kann durch sogenanntes Preseeding vor dem eigentlichen Zugriff der Clients erfolgen. Dieses Vorgehen ist geeignet, wenn die Daten lokal vorliegen, da der Erstellungsprozess (Seeding) bei verteilt liegenden Quellen sehr viel Zeit in Anspruch nehmen kann (Tabelle 2.3). Alternativ ist es möglich, die Bildpyramide stückweise durch den Zugriff der Clients aufzubauen. Dieses Seeding by Request hat den Nachteil, dass beim ersten Zugriff auf die Kachel Wartezeiten beim Client und Rechenlast beim Server entstehen.

Die Größe des Caches ist bei kachelnden Kartendiensten abhängig von dem verwendeten Dateiformat und von der Anzahl der Zoomstufen. Da die Kombination der Layer in der Regel beim Client erfolgt, muss für jeden Layer eine separate Bildpyramide aufgebaut werden. Für weitere Projektionen ist ebenfalls jeweils eine neue Bildpyramide notwendig. Ist der Cache nach dem Prinzip eines Quadtree aufgebaut, lässt sich der Speicherbedarf nach der Formel in Listing 2.1 berechnen.

$$N_T = m \sum_{i=0}^{n-1} 4^{p+i}$$

N_T Kachelanzahl

m Anzahl der Layer (ggf. mit Anzahl der Projektionen multipliziert)

n Anzahl der Zoomstufen

4^p Kachelanzahl in der 1. Zoomstufe (für 2 Kacheln ist p = 0,5)

Listing 2.1: Formel zur Bestimmung der Kachelanzahl nach ZHLIZU, 2008

Ein einfacher Faktor ermöglicht über die in Listing 2.1 eingeführte Formel, den ungefähren Speicher- und Zeitbedarf für die Erstellung eines Kartencaches zu bestimmen. Nach Abschätzen der

durchschnittlichen Dateigröße einer Kachel ergibt das Produkt aus Dateigröße und Kachelanzahl (N_T) den insgesamt benötigten Speicherplatz. Analog lässt sich der Zeitbedarf über den Zeitfaktor einer Kachel bestimmen.

Zoomstufenanzahl	Kachelanzahl	Speicherbedarf	Zeitbedarf
1	2	40 KB	0,22 Sekunden
2	10	200 KB	1,11 Sekunden
3	42	840 KB	4,67 Sekunden
4	170	3 MB	19 Sekunden
...	...	...	...
11	2796202	53 GB	4 Tage
12	11184810	213 GB	14 Tage

Tabelle 2.3: Speicher- und Zeitbedarf für das Erstellen einer Bildpyramide

Exemplarisch zeigt die Tabelle 2.3 wie sich Speichernutzung und Zeitaufwand entsprechend der Zoomstufenzahl erhöht. Die Daten der Tabelle wurden nach der angegebenen Formel für eine Bildpyramide mit zwei Kacheln in der ersten Zoomstufe (OGC-WMTS, 2010, ANNEX E.3), einer Dateigröße pro Kachel von 20 KB und einem Zeitaufwand von einer Sekunde für das Erstellen von neun Kacheln[16] berechnet. Eine Bildpyramide in der Quadtree-Struktur mit 12 Zoomstufen könnte z. B. eine Karte vom Maßstab 1:8 192 000 bis 1:4 000 abbilden.

Ist die Größe des Caches begrenzt, kann der Einsatz bekannter Verdrängungsstrategien für einen optimierten Datenbestand sorgen:

- FIFO (First in First Out) - die älteste Kachel wird gelöscht.
- LRU (*Least Recently Used*) - die Kachel auf die am längsten kein Zugriff erfolgte, wird gelöscht.
- LFU (*Least Frequently Used*) - die am seltensten gelesene Kachel wird verdrängt.

Diese und andere optimierte Verdrängungsstrategien werden auch bei Web Caches eingesetzt. [BONCHI, 11/2001, S. 168]

Für die Aktualisierung des zwischengespeicherten Datenbestandes gibt es zwei allgemeine Lösungen. Das Pulling von Daten beschreibt eine gezielte zyklische Abfrage von Änderungen der

16 Das Zusammenfassen meherer Kacheln beim Cacheprozess, genannt Metatiling, reduziert die Anzahl der Zugriffe auf die Datenquelle und damit deren Auslastung.

Datenquelle. Beim Pushing von Daten werden von der Datenquelle Änderungen an den Cache aktiv übertragen. Bei gekachelten Kartendiensten wirken sich Veränderungen in den Quelldaten auf alle Stufen der Bildpyramide aus. Damit nur ein Teil aller Stufen neu gekachelt werden muss, gibt es Verfahren das betroffene Gebiet räumlich einzugrenzen.

2.4.1.3 Lastverteilung

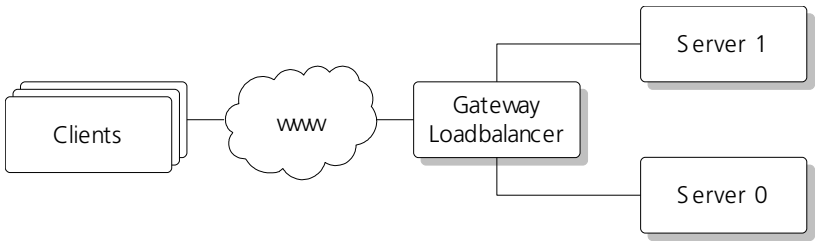

Abbildung 2.9: Einfache Lastverteilung durch einen Gateway

Der Einsatz von Lastverteilung (engl. load balancing) ist in verschiedenen Schichten des ISO-OSI-Referenzmodells[17] möglich. Die hier vorgestellte Variante bezieht sich ausschließlich auf die Anwendungsebene. Sie lässt sich auch mit klassischen Kartendiensten sehr gut realisieren. So kann die notwendige Verarbeitungskapazität für das Rendering der Kartenansichten auf mehrere Rechner in einem geschlossenen Netzwerk verteilt werden. Abbildung 2.9 zeigt einen einfachen Aufbau der Lastverteilung, in der die Clients mit einem Gateway kommunizieren. Server 0 und 1 stellen die gleichen Funktionalitäten bereit, sind aber nicht direkt durch die Clients erreichbar. Der Gateway verteilt die Anfragen gleichmäßig an die Server. Diese Verteilung erfolgt üblicherweise auf Grundlage von ermittelten Lasten an den Servern. [BAUMEISTER, 2005]

17 DIN ISO 7498: Information technology – Open Systems Interconnection – Basic Reference Model: The basic model

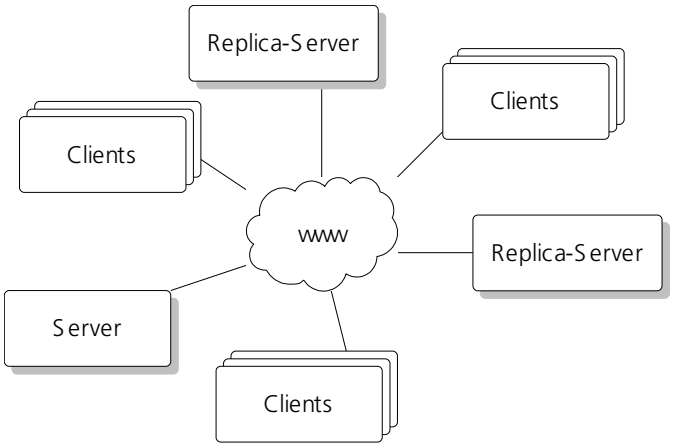

Abbildung 2.10: Content Distribution Network

Im Gegensatz zur einfachen Lastverteilung kommunizieren die Clients in einem Content Distribution Network (CDN) nicht immer mit dem selben Server. Ein CDN besteht aus einer Reihe von Webservern, die über mehrere geografische Orte verteilt sind. Durch Techniken wie DNS-basiertes Request Routing oder einfache HTTP-Redirection (beschrieben in RFC3568, 2003) werden die Anfragen der Clients zu dem Server, der die geringste räumliche Entfernung oder die geringste Last aufweist, weiter geleitet (Abbildung 2.10). Der Ursprungsserver gleicht die zu verteilenden Inhalte mit einer großen Zahl an Reqlica-Servern ab. CDNs werden zur Auslieferung statischer Inhalte verwendet, sind also für klassische Kartendienste nicht geeignet. Der Einsatz eines solchen Netzwerks ist von der Anzahl der zu erwartenden Nutzer und deren geografischer Verteilung abhängig. Auf dem Markt existieren kommerzielle (Akamai, Amazon CloudFront, CacheFly u. a.) und freie (Coral CDN, CoDeeN u. a.) Anbieter, welche den Zugriff auf ihre umfangreiche Infrastruktur als Dienstleistung anbieten. [SOUDERS, 2008, KAPITEL 2; BLEICH, 2008]

2.5 Standards

Mit der Vorstellung von Google Maps 2005 und die darauffolgende Entwicklung von freien Alternativen wie Openstreetmap[18] (2006) begannen die Bestrebungen kachelnde Kartendienste zu standardisieren. Auf der FOSS4G[19] 2006 wurden als Resultat einer informellen Diskussionsrunde (BOF) zum Thema Web Map Tiling Standard erste Ansätze für Spezifikationen veröffentlicht. Im Vordergrund stand die Verbesserung der „cacheability“ von existierenden WMS Servern auf der einen Seite und auf der Anderen die Entwicklung eines Tiling Standards für die Implementation von Web Map Servern und Clients. [OSGEO-TILING, 2006]

18 http://www.openstreetmap.org/
19 http://www.foss4g.org/

Alle Standards basieren auf dem Konzept, Kacheln fester Größe in einem festen geografischen Netz mit einer festen Maßstabsreihe zu definieren. Es wird also die in Kapitel 2.4 vorgestellte Datenstruktur möglichst interoperabel beschrieben. Die Adressierung der Kacheln geht immer von der Überlegung aus, dass die gekachelte Bildpyramide in einzelne lokale, voneinander abhängige Koordinatensysteme unterteilt wird. Die Zoomstufe dient zur Identifikation dieser auch Tile Matrix oder Tile Set genannten Koordinatensysteme. Eine Zuordnung des Koordinatenursprungs zum übergeordneten Koordinatensystem georeferenziert die jeweilige Tile Matrix. Je nach Standard liegt der Ursprung entweder in der Nordwest- oder Südwestecke der Darstellungsebene. Client und Server können so mit den Werten aus Koordinatenursprung, Maßstab (oder Auflösung) und Kachelgröße die Eckwerte der Kacheln im lokalen und übergeordneten Koordinatensystem bestimmen. Eine interaktive Karte auf maptiler.org[20] beschreibt die in Abbilung 2.11 dargestellte Adressierung genauer und zeigt zusätzlich für jede Kachel die Begrenzungen.

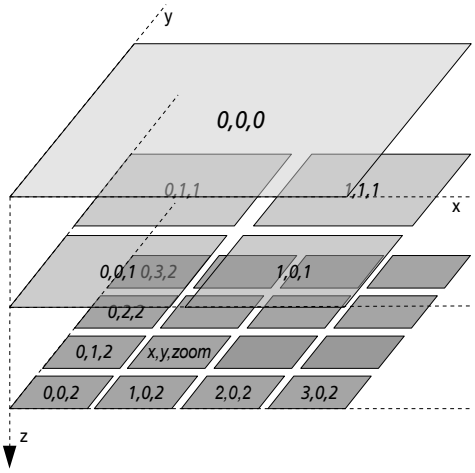

Abbildung 2.11: Adressierung der Kacheln

2.5.1 WMS Tiling Client Recommendation

Die WMS Tiling Client Recommendation (WMS-C) wurde von Mitgliedern der Open Source Geospatial Foundation (OSGeo)[21] 2006 veröffentlicht und basiert auf der OpenGIS WMS Spezifikation 1.1.1. WMS-C beschreibt Erweiterungen zum WMS Standard, die die Qualität und Skalierbarkeit von gekachelten Karten sowie die Cache-Fähigkeit von WMS Requests verbessern sollen. [OSGEO-WMSC, 2006]

20 http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/
21 http://www.osgeo.org/

Da das Hauptanliegen der Recommendation die Kompatibilität zu bestehenden WMS Diensten ist, können die Kacheln nicht direkt adressiert werden. Der Kartenclient hat die Aufgabe auf Basis einer erweiterten Dienstbeschreibung, korrekte `GetMap` Requests für jede Kachel abzusetzen. Die schematische Darstellung (Abbildung 2.12) der `VendorSpecificCapabilities` veranschaulicht, wie durch WMS-C die Anfragen an einen Dienst eingeschränkt werden. Ein Tile Set beschreibt ein sogenanntes Tiling Profile, in dem durch Bekanntgeben der Projektion, der Auflösungen, Höhe und Breite der Kacheln und optional der Begrenzung auf ein bestimmtes Gebiet, eine gekachelte Bildpyramide definiert wird. In dem `Resolution` Element werden die einzelnen Auflösungen in einer mit Komma getrennten Liste angeben. Ein Resolutionwert gibt die Pixelauflösung (Einheit der angegebenen Projektion pro Pixel) an. Darüber hinaus definiert das Tiling Profile ein festes Dateiformat und optional Festlegungen für Layer und Style. Das lokale Koordinatensystem der einzelnen Zoomstufen hat seinen Ursprung in der Südwestecke der Darstellungsebene und wird in einem Tile Set durch die Bounding Box oder die verwendete Projektion definiert. Zusammen bestimmen diese Angaben wie ein WMS-C konformer Client Anfragen an den Dienst zu stellen hat.

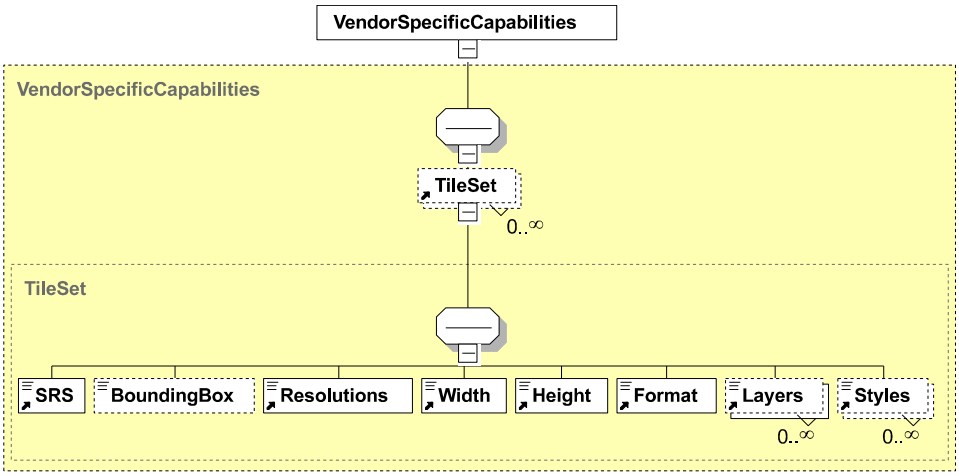

Abbildung 2.12: Schematische Darstellung der Capabilities Erweiterung durch WMS-C

Konkret ordnet sich ein `GetMap` Request anhand der Request Parameter `srs`, `width`, `height`, `format`, `layers`, `styles` einem bestimmten Tile Set zu. Nur durch Angabe der Bounding Box, also der geografischen Begrenzung, kann der `GetMap` Request einer bestimmten Kachel in dem Tile Set zugeordnet werden. Dadurch werden die einzelnen `GetMap` Requests, die zum Aufbau einer Karte notwendig sind, cachefähig. Browser oder Web Caches können sie also für wiederkehrende Anfragen zwischenspeichern.

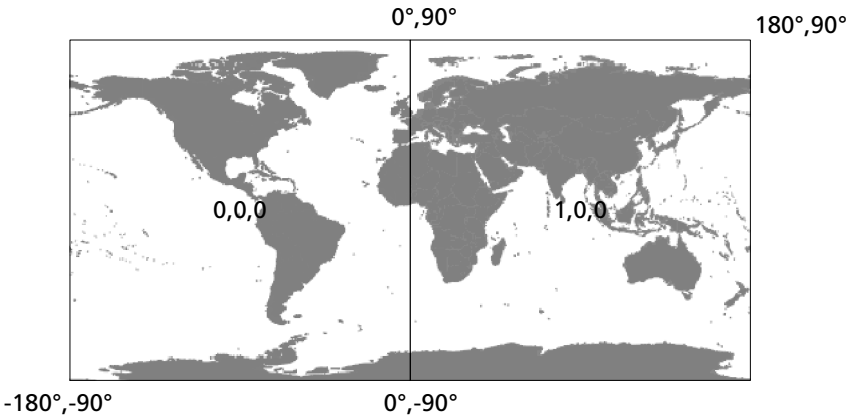

Abbildung 2.13: Bounding Box und Kacheln in der geographischen Projektion

Parameter	Wert Kachel (0,0,0)	Wert Kachel (1,0,0)
service	wms	wms
version	1.1.1	1.1.1
request	GetMap	GetMap
layers	erde	erde
format	image/png	image/png
styles		
exceptions	application/vnd.ogc.se_inimage	application/vnd.ogc.se_inimage
srs	EPSG:4326	EPSG:4326
bbox	-180,-90,0,90	0,-90,180,90
width	256	256
height	256	256

Tabelle 2.4: WMS-C GetMap Parameter entspechend der Abbildung 2.13

Tabelle 2.4 listet die entprechenden Parameter für die Kacheln in der ersten Zoomstufe auf. Die zugrunde liegende Bildpyramide ist in OGC-WMTS, 2010, ANNEX E definiert und wird z. B. von Google Earth verwendet. In der WMS Tiling Client Recommendation selbst werden zwei globale Profile beschrieben, die dem Google Earth (OGC-WMTS, 2010, ANNEX E.3) und Goolge Maps (OGC-WMTS, 2010, ANNEX E.4) Profil entspechen.

Wenn WMS-C mit dynamischen Kartendiensten eingesetzt wird, bietet es nur die Vorteile des HTTP Caching, kann dafür aber bei bestehenden Softwareinstallationen schnell umgesetzt werden und benötigt keinen zusätzlichen Speicherplatz für die Bildpyramiden. In Verbindung mit kachelnden Kartendiensten hat WMS-C den Nachteil, dass der HTTP-Request nicht direkt

die einzelne Kachel anspricht, sondern eine spezielle Logik notwendig ist, die die WMS Anfrage einer bestimmten Datei zuordnet. Um diese Zuordnung zu erleichtern, sieht die Recommendation eine Rundung auf 6 Nachkommastellen für die Werte des `bbox`-Parameters vor. [OSGEO-WMSC, 2006]

2.5.2 Tile Map Service Specification

Die Tile Map Service Specification (OSGEO-TMS, 2006) wurde ebenfalls 2006 von der Open Source Geospatial Foundation veröffentlicht. Die Spezifikation soll nicht nur die Kommunikation zwischen Client und Server regeln, sondern auch Vorschläge für die Implementation der Kartendienste geben.

Ein Tile Map Service (TMS) implementiert ein REST-Interface, mit dem auf die vorgerenderten Kacheln möglichst einfach zugegriffen werden kann. Die Ressource-Orientierung wird auch konsequent bei der Organisation der Dienstbeschreibung eingesetzt. Jeweils ein Dokument beschreibt den Dienst allgemein, die vorhandenen Layer, die Zoomstufen und die dazugehörigen Kacheln. Alle Metadaten sind zueinander verlinkte XML Dokumente. Die Spezifikation gibt zwar Beispiele für den Aufbau dieser Metadatendokumente, beschreibt aber keine Möglichkeit zur Validierung.

Da kein plattformneutrales Modell durch die Spezifikation gegeben ist, wird an dem konkreten Beispiel in Listing 2.2 – 2.5 gezeigt, wie sich die Dokumentenstruktur eines TMS aufbaut. Die eigentliche Dienstbeschreibung erfolgt in der TileMapService Ressource (Listing 2.3), welche mit der Capabilities eines WMS vergleichbar ist. In diesem Dokument werden die zur Verfügung stehenden Layer (TileMaps) linear aufgelistet und mit der zugehörigen TileMap Ressource verlinkt. Die TileMap Ressource (Listing 2.4) beschreibt die Datenstruktur der Bildpyramide und referenziert diese zu einem festgelegten Koordinatensystem. Ein Kartenclient verwendet die in den Tile Sets definierte Basis-URL für den Zugriff auf die Kacheln (Listing 2.5). Die Ordnerstruktur nach dem Muster `http://[Server]/[Dienstversion]/[Layername]/[Zoomstufe]/[X]/[Y].[Dateiendung]` wird von der Spezifikation empfohlen. Der Dienst kann aber, bis auf die x- und y-Koordinaten, selbst seine Ressourcen beschreiben. Daher ist das gegebene und im Listing gezeigte Schema nicht zwingend.

Wie auch bei der WMS-C Recommendation hat das lokale Koordinatensystem der einzelnen Zoomstufen seinen Ursprung in der Südwestecke der Darstellungsebene. Dieser Ursprung wird durch das `Origin`-Element in der Layer beschreibenden XML (Listing 2.4) definiert. Die Auflösung der Zoomstufen wird auch hier mit Karteneinheiten pro Pixel angegeben.

```
http://tms.geodaten.htwdd

<?xml version="1.0" encoding="UTF-8" ?>
<Services>
   <TileMapService
      title="TMS"
      version="1.0.0"
      href="http://tms.geodaten.htwdd/1.0.0" />
</Services>
```

Listing 2.2: Root Ressource - Zentrales XML Dokument mit Verlinkung auf den TMS

```
http://tms.geodaten.htwdd/1.0.0

<?xml version="1.0" encoding="UTF-8" ?>
<TileMapService
   version="1.0.0"
   services="http://tms.geodaten.htwdd/1.0.0">
   ...
   <TileMaps>
      <TileMap
         title="World Map"
         srs="EPSG:4326"
         profile="none"
         href="http://tms.geodaten.htwdd/1.0.0/wmap" />
      ...
   </TileMaps>
</TileMapService>
```

Listing 2.3: TileMapService Ressource - XML Dokument für die eigentliche Dienstbeschreibung

```
http://tms.geodaten.htwdd/1.0.0/wmap

<?xml version="1.0" encoding="UTF-8" ?>
<TileMap
   version="1.0.0"
   tilemapservice="http://tms.geodaten.htwdd/1.0.0">
   ...
   <BoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
   <Origin x="-180" y="-90" />
   <TileFormat
      width="256"
      height="256"
      mime-type="image/png"
      extension="png" />
   <TileSets profile="none">
      <TileSet href="http://tms.geodaten.htwdd/1.0.0/wmap/0"
         units-per-pixel="0.703125" order="0" />
      <TileSet href="http://tms.geodaten.htwdd/1.0.0/wmap/1"
         units-per-pixel="0.3515625" order="1" />
      ...
   </TileSets>
</TileMap>
```

Listing 2.4: TileMap Ressource - XML Dokument für die Beschreibung einer Bildpyramide

```
1 http://tms.geodaten.htwdd/1.0.0/wmap/0/0/0.png
2
3
```

Listing 2.5: Request und Response auf eine TMS Kachel Ressource

Der größte Vorteil der Spezifikation ist, dass die einzelnen im Dateisystem vorliegenden Kacheln direkt über eine URL angesprochen werden können. Die Zugriffe der Clients erfordern dadurch vom Dienst keine weitere Logik und können von einfachen Webservern verarbeitet werden. Die Abbildung der Daten in einer einfachen Ordnerstruktur vereinfacht auch die Implementation von Client und Serveranwendungen, die sowohl HTTP-Caching als auch Server Side Caching und Lastverteilung in einem CDN anwenden. Konkrekte Implementationsvorschläge werden auch direkt in der Spezifikation beschrieben [OSGEO-TMS, 2006, IMPLEMENTATION ADVICE].

2.5.3 OpenGis Web Map Tile Service Implementation Standard

Der am 6. April 2010 vom OGC veröffentlichte Standard beschreibt auf Basis der Web Service Common Implementation Specification (OWS Common, OGC-OWS, 2007) einen kachelnden Kartendienst. Grundlegend flossen bei der Entwicklung vorhandene Ansätze mit ein. So finden sich die Konzepte aus der OSGeo TMS Specification, aber auch aus Google Maps und NASA OnEarth[22] wieder. Durch das Aufgreifen vorhandener Implementationsschemata wurde erstmals REST als Architekturkonzept in einem OGC Standard eingeführt. Neben REST bietet die Spezifikation für die Nachrichtenübermittlung zwischen Client und Server die klassische Prozedur-orientierte Architektur mit Key-Value Pairs (KVP) oder XML über SOAP an. Alle nachfolgenden Angaben beschreiben die Funktionen eines WMTS vorerst plattformneutral mit Bezug auf Version 1.0.0 des Standards (OGC-WMTS, 2010, 07-057R7).

22 http://onearth.jpl.nasa.gov/tiled.html

Das WMTS Interface bietet drei grundlegende Operationen:

- GetCapabilities – liefert eine Beschreibung über Fähigkeiten und Datenstruktur des entprechenden Dienstes (vgl. OGC-WMS, 2004, KAPITEL 7.2).
- GetTile – liefert eine Kachel als Teil einer Karte für einen bestimmten Layer.
- GetFeatureInfo – liefert Informationen über Features an einer bestimmten Pixelposition der gekachelten Karte (vgl. OGC-WMS, 2004, KAPITEL 7.4). Diese Funktion ist optional.

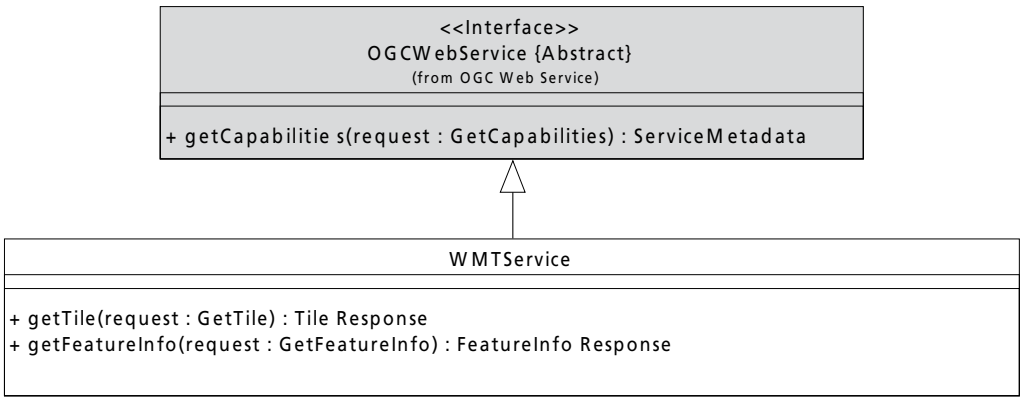

Abbildung 2.14: Funktionen des WMTS Interface als UML Diagramm (vereinfacht) [OGC-WMTS, 2010, FIGURE 1]

Die grau hinterlegte Klasse im Diagramm (Abb. 2.14) wurde aus OWS Common übernommen. Dieses Schema wird auch auf nachstehende Diagramme zum WMTS angewendet. Die Klassen ServiceMetadata und GetTile werden nachfolgend genauer beschrieben, da sie die grundlegenden Strukturen des Standards festlegen. Die komplette Dokumentation des Implementationsmodells ist in OGC-WMTS, 2010, KAPITEL 7 und in den erweiterten UML Diagrammen in OGC-WMTS, 2010, ANNEX C zu finden.

2.5.3.1 Service Metadaten

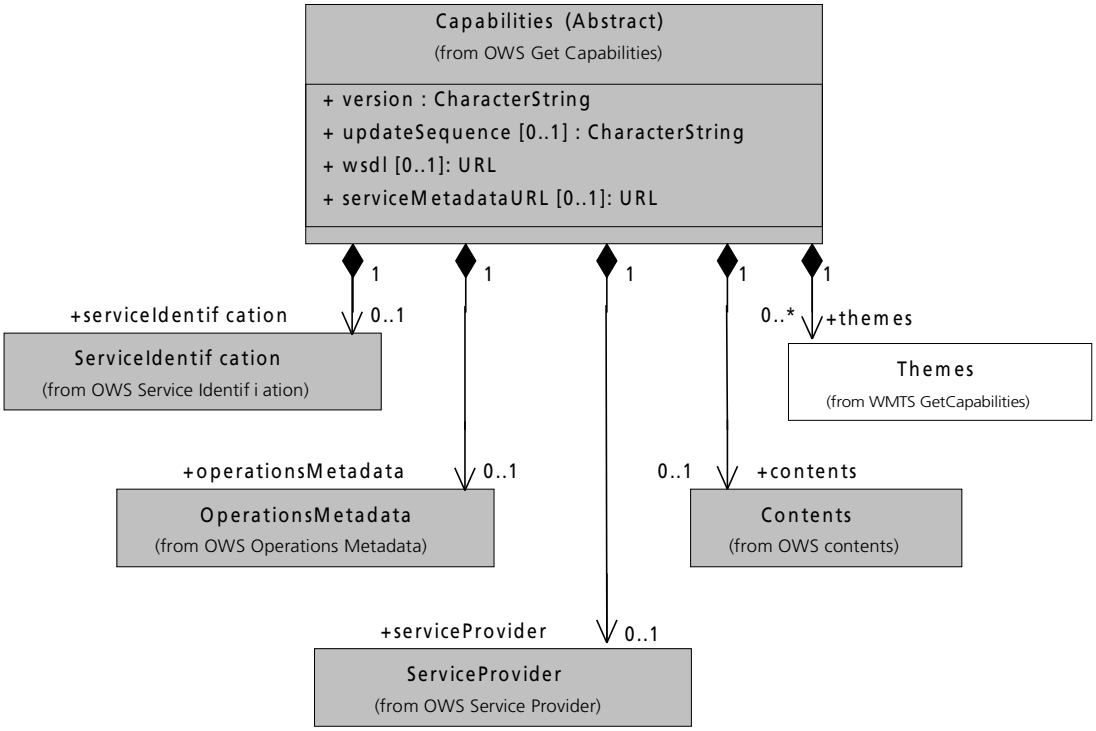

Abbildung 2.15: Service Metadaten UML Diagramm (vereinfacht) [OGC-WMTS, 2010, FIGURE 4]

Die Klassen der Service Metadaten wurden überwiegend aus OWS Common (OGC-OWS, 2007, KAPITEL 7.4.2) übernommen. Erweiterungen, die bei der Standardisierung eingeflossen sind, lassen sich mit den besonderen Gegebenheiten kachelnder Kartendienste begründen. So beschreibt die Themes Klasse (Abbildung 2.15) wie Layer thematisch kombiniert werden können. Eine hierarchische Organisation von Kartenlayern wie es bei WMS möglich ist, lässt sich bei WMTS nicht umsetzen, da die Kombination der Layer auf der Clientseite erfolgt. Die unabhängige Definition von Themen ermöglicht mehrere Layerkombinationen oder sogar das Ignorieren der Vorgaben durch den Client. Die für diesen Implementationsstandard spezifischen Erweiterungen werden weitesgehend in der Contents Klasse definiert, da sie die eigentliche Datenstruktur beschreibt.

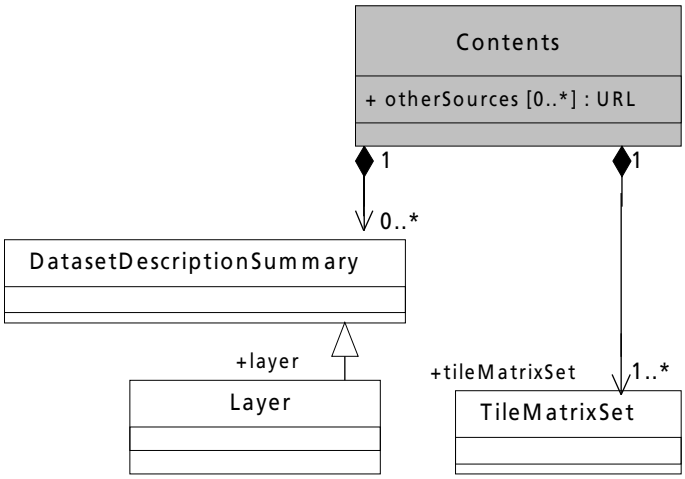

Abbildung 2.16: Service Metadaten für Contents [OGC-WMTS, 2010, FIGURE 4]

Abbildung 2.16 zeigt vereinfacht die Contents Klasse, in der die Definition der Layer sowie der Tile Matrix Sets getrennt erfolgt. Ein Tile Matrix Set beschreibt allgemein die Struktur einer gekachelten Bildpyramide mit festgelegten Zoomstufen (Tile Matrix) unabhängig von den zugrunde liegenden Daten. Im Vergleich zu den schon beschriebenen Spezifikationen WMS-C und TMS ermöglicht WMTS eine flexiblere Organisation der Datenstrukur. So können jeweils pro TileMatrix die Pixelmaße der Kacheln (`tileWidth, tileHeight`) bestimmt werden, sowie deren Zuordnung in das übergeordnete Koordinatensystem (`topLeftPoint`) (Abbildung 2.17). Weiterhin ist auffällig, dass für ein Tile Matrix Set vorerst keine Angaben zum Dateiformat gemacht werden. Diese Festlegung erfolgt wie beim WMS Standard für jeden Layer separat.

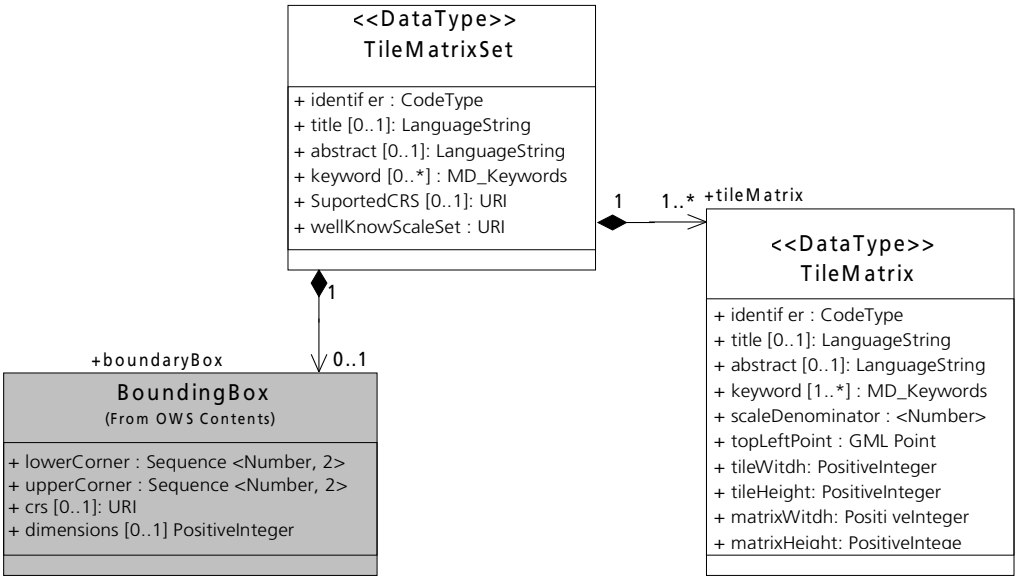

Abbildung 2.17: WMTS Tile Matrix Sets mit Attributen [OGC-WMTS, 2010, FIGURE 8]

Der Maßstab (`scaleDenominator`) einer Tile Matrix wird auf Basis einer standardisierten Pixelauflösung von 0.28 x 0.28 mm angegeben. Diese Festlegung kann allerdings nicht sicherstellen, dass der angegebene Maßstab auch der Anzeige beim Client entspricht. Generell bedeuten verschiedene Anzeigegeräte unterschiedliche Bildschirmauflösungen (Handy, TV, Desktop PC). Allerdings besteht die Möglichkeit den angezeigten Maßstab auf der Clientseite hardwarespezifisch anzupassen.

Das Attribut `identifier` dient zur eindeutigen Identifikation der Instanzen von Tile Matrix sowie Tile Matrix Set. In der Regel wird die Identifikation anhand der Projektion gewählt. Zum Beispiel wäre „EPSG:31468“ für ein Tile Matrix Set und entsprechend „EPSG:31468:0“ für die erste Tile Matrix der Identifikator. Da der Standard beliebige Zeichenketten erlaubt, ist es allerdings auch möglich mehrere Tile Matrix Sets für die gleiche Projektion zu erstellen. Einzige Voraussetzung ist die Eindeutigkeit des `identifier` Attributs innerhalb des Namespaces der Service Metadaten.

Die getrennte Beschreibung von Layer und Tile Matrix erlaubt die Wiederverwendung der Tile Matrix Sets. Ein Layer kann einem oder mehreren Tile Matrix Sets zugeordnet werden. Das in Abbildung 2.18 dargestellte UML Diagramm veranschaulicht die Beziehungen genauer. Ein Layer referenziert mit dem Tile Matrix Set Link ein oder mehrere Tile Matrix Sets. Da eine Tile Matrix für die Wiederverwendung möglichst allgemein definiert werden sollte, ist es mit Tile Matrix Limits möglich den Darstellungsbereich einzuschränken. Pro Layer wird mindestens ein Dateiformat für die Kacheln bestimmt und ein oder mehrere Styles repräsentieren jeweils eine Kartendarstellung.

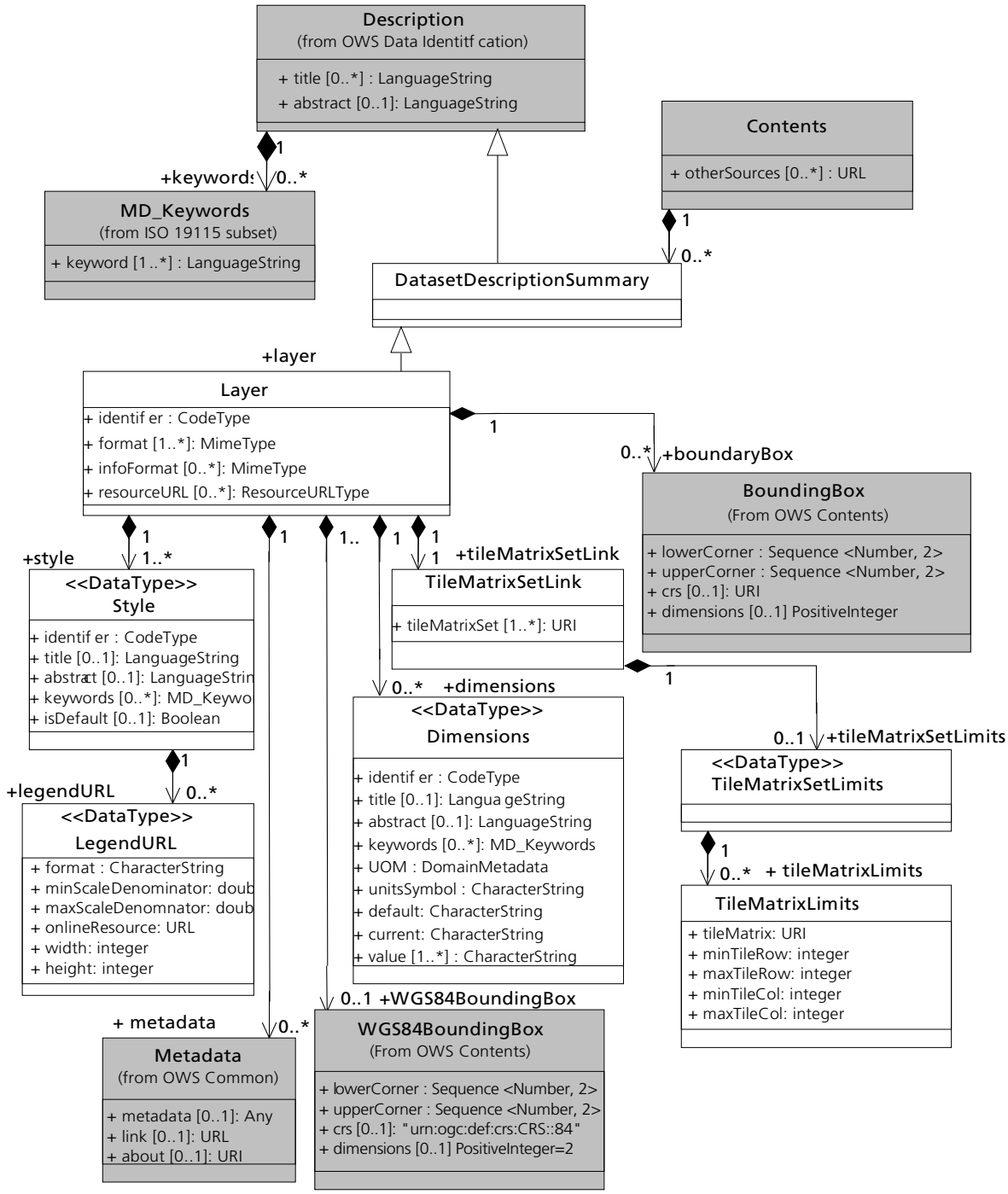

Abbildung 2.18: WMTS Layer mit Attributen [OGC-WMTS, 2010, FIGURE 6]

Aus den beschriebenen Service Metadaten für einen WMTS lässt sich nach der Formel `nTileMatrixSets x nStyles x nFormats` (n – Anzahl) die Zahl der zu kachelnden Bildpyramiden pro Layer bestimmen. Die Service Metadaten können auch multi-dimensionale Daten beschreiben. Beispiele für zusätzliche Dimensionen sind Zeit oder Höhenangaben. Das Hinzu-

fügen von Dimensionen erweitert die angegebene Formel um das Produkt `nDimensions`. Für entsprechende Dienste bedeutet eine zusätzliche Dimension auch eine zusätzlich zu speichernde Bildpyramide für jede Projektion, jeden Style und jedes Format.

Die Zusammenhänge zwischen den in den Diagrammen angegebenen Attributen werden in Abbildung 2.19 nochmals verdeutlicht. Im Gegensatz zu den schon vorgestellten Standards wird eine Tile Matrix durch die Nordwestecke der Darstellungsebene georeferenziert. Dadurch verhält sich die Nummerierung der Kacheln in der y-Richtung invers zu der in TMS und WMS-C. Die Tile Matrix Limits beschreiben eine Teilmenge aus einer referenzierten Tile Matrix mit den Attributen `minTileRow`, `minTileCol` und `maxTileRow`, `maxTileCol`.

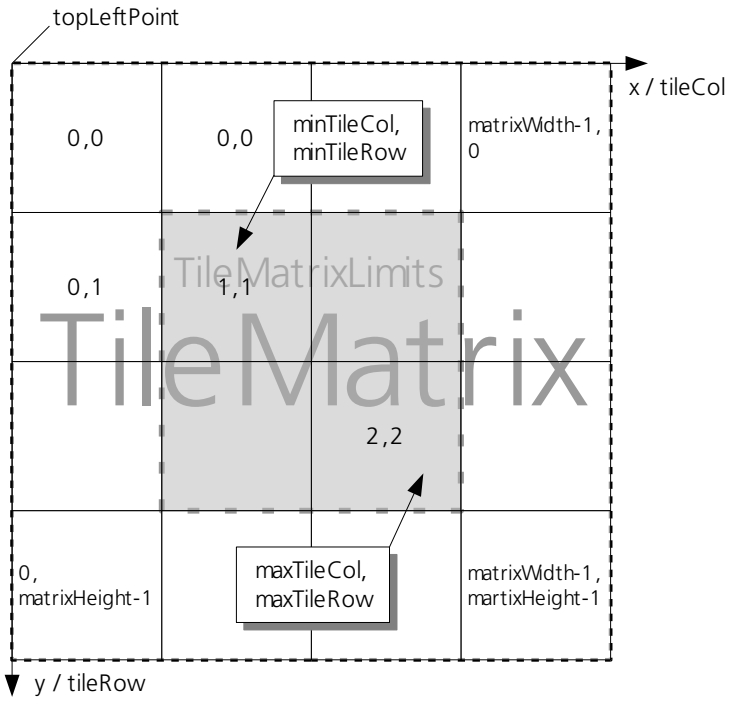

Abbildung 2.19: TileMatrix und TileMatrixLimits Eigenschaften nach OGC-WMTS, 2010, FIGURE 7

2.5.3.2 GetTile Request

Der WMS Standard beschreibt den GetMap Request, bei dem durch Angabe des Layers, der Projektion und der Begrenzung das fertige Kartenbild als Response geliefert wird. Für den WMTS GetTile Request werden keine Projektion und auch keine zu der Projektion passende Begrenzung parametrisiert. Der Kartenclient spricht die in den Service Metadaten beschriebenen Layer, das zugeordnete Tile Matrix Set (Bildpyramide), die entsprechende Tile Matrix (Zoomstufe) und die darin enthaltene Kachel direkt an. Ein komplettes Kartenbild benötigt mehrere

entsprechend zusammengehörende GetTile Requests. Die Berechnung, welche Kacheln für einen bestimmten Ausschnitt benötigt und wie sie in dem Kartenbild positioniert werden, erfolgt beim Kartenclient. Das detailierte UML Diagramm in Abbildung 2.20 beschreibt wie genau ein GetTile Request aufgebaut ist.

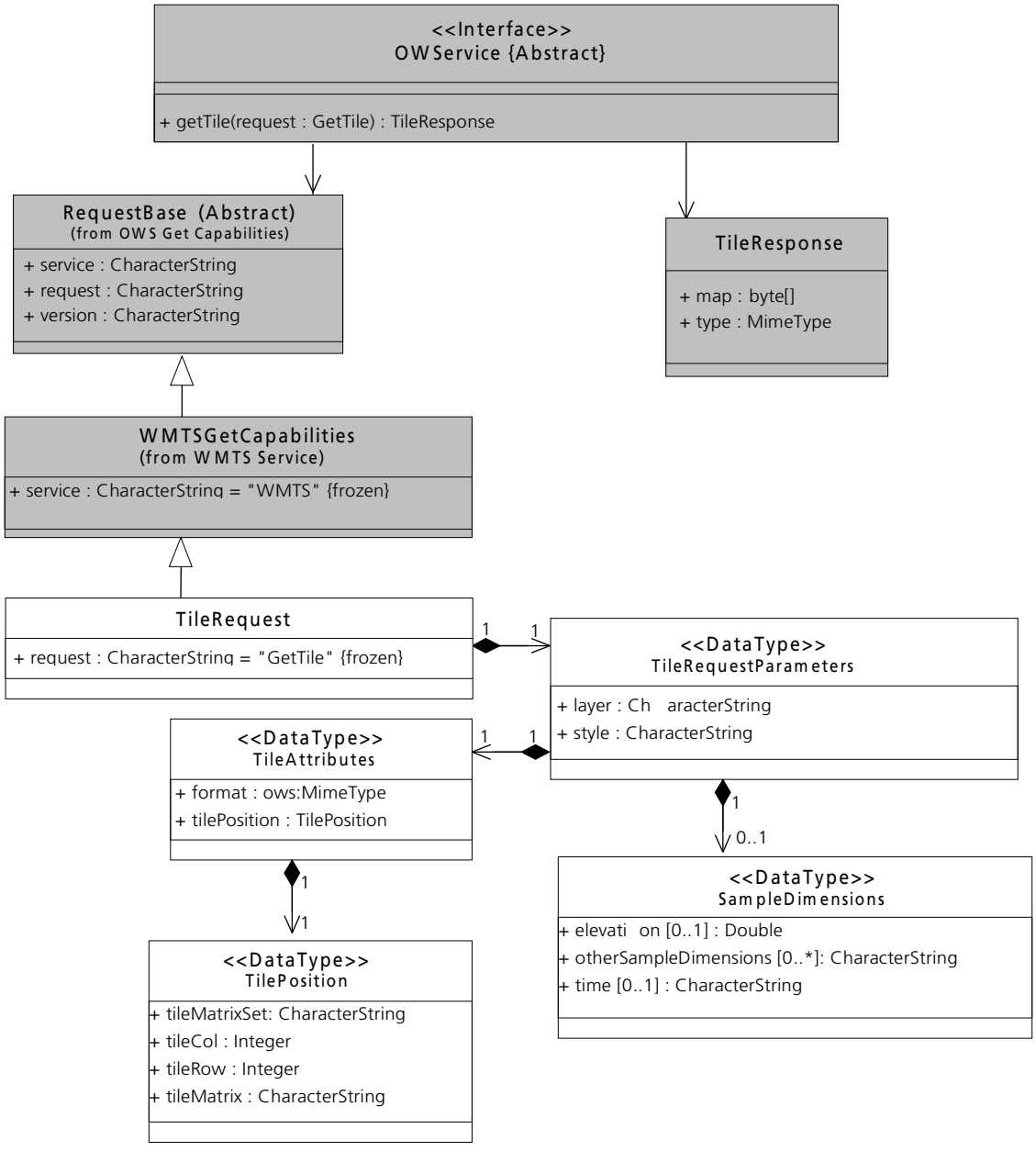

Abbildung 2.20: GetTile Request Parameter [OGC-WMTS, 2010, FIGURE 10]

Die Adressierung der Kacheln in einer Tile Matrix erfolgt mit den Attributen `tileCol` und `tileRow`, welche den x- und y-Koordinaten im schon beschriebenen ganzzahligen Koordinatensystem entsprechen. In Tabelle 2.5 werden beispielhaft die Werte für GetTile Requests aufgelistet, deren Response der in Abbildung 2.13 dargestellten Kacheln entsprechen.

Attribut	Wert (Kachel 0,0)	Wert (Kachel 1,0)
service	WMTS	WMTS
version	1.0.0	1.0.0
request	GetTile	GetTile
layer	erde	erde
style	default	default
format	image/png	image/png
tileMatrixSet	EPSG:31468	EPSG:31468
tileMatrix	EPSG:31468:0	EPSG:31468:0
tileCol	0	1
tileRow	0	0

Tabelle 2.5: GetTile Request entsprechend der Abbildung 2.13

2.5.3.3 Request Encoding

Die vorgestelle plattformneutrale Beschreibung des Dienstes kann für konkrete verteilte Systeme übersetzt werden. Der Standard spezifiziert geeignete Kodierungen für HTTP-GET Funktionsaufrufe mittels KVP oder REST und HTTP-POST Funktionsaufrufe, bei denen XML oder SOAP eingesetzt wird. Entsprechend der Vorgaben aus OWS Common werden die vom Dienst unterstützten Requestkodierungen in den OperationsMetadata (vgl. Abb. 2.15) deklariert.

Key Value Pair

Zum Beispiel würde der KVP GetTile Request für die Kachel 0,0 aus der Tabelle 2.5 folgender URL entsprechen:

```
http://wmts.geodaten.htwdd/service?
service=WMTS&version=1.0.0&request=GetTile&layer=erde&style=default&format=
image/png&tileMatrixSet=EPSG:31468&TileMatrix=EPSG:31468:0&tileCol=0&tileRo
w=0
```

Die Antwort auf diesen Request wäre ein dem `format` Parameter entsprechendes Bild oder bei Fehlschlag ein in OGC-WMTS, 2010, TABLE 24 definierter HTTP Exception Code.

SOAP

Listing 2.6 und 2.7 enthalten die SOAP Kodierung für die gleiche Resquest – Response-Folge. Der Standard schreibt vor, dass die Binärdaten der GetTile Response entsprechend der XML

Schema Part 2 W3C Recommendation Base64[23] kodiert und in den SOAP Body eingebettet werden. Dieses Vorgehen erlaubt im Gegensatz zu verlinkten externen Quellen die Implementation von Sicherheitsmechanismen wie Web Services Security[24] (WSS). Die Fehlerbehandlung mit SOAP ist allgemein in OGC-OWS, 2007, KAPITEL 8.5 beschrieben. Für den GetTile Request werden in OGC-WMTS, 2010, TABLE 23 spezielle Exceptionscode-Werte festgelegt.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <soap:Envelope
3     xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
4     <soap:Body>
5        <GetTile  service="WMTS" version="1.0.0"
                   xmlns="http://www.opengis.net/wmts/1.0">
6           <Layer>erde</Layer>
7           <Style>default</Style>
8           <Format>image/png</Format>
9           <TileMatrixSet>EPSG:31468</TileMatrixSet>
10          <TileMatrix>EPSG:31468:0</TileMatrix>
11          <TileRow>0</TileRow>
12          <TileCol>0</TileCol>
13       </GetTile>
14    </soap:Body>
15 </soap:Envelope>
```

Listing 2.6: SOAP WMTS GetTile Request

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <soap:Envelope
3     xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
4     <soap:Body>
5        <wmts:BinaryPayload
             xmlns:wmts="http://www.opengis.net/wmts/1.0">
6           <wmts:Format>image/png</wmts:Format>
7           <wmts:BinaryContent>
8              <!-- base64-encoded -->
9              <![CDATA[R0lGOD...J3IiYEAADs=]]>
10          </wmts:BinaryContent>
11       </wmts:BinaryPayload>
12    </soap:Body>
13 </soap:Envelope>
```

Listing 2.7: SOAP WMTS GetTile Response

REST

Wichtigster Schritt in einer Ressource-orientierten Architektur ist das Bestimmen der Ressourcen und deren Beziehungen zueinander. WMTS beschreibt drei Ressourceklassen: den Service (ServiceMetadata), die Kacheln (Tile) und optional die Feature Informationen (FeatureInfo). Der Standard definiert nur den Gebrauch von HTTP-GET für den Download der Ressourcen.

23 Die Kodierung nach Base64 beschreibt ein Verfahren zur Kodierung von Binärdaten in einer ASCII Zeichenfolge
24 http://www.ibm.com/developerworks/library/specification/ws-secure

Ein WMTS der REST einsetzt, besteht aus einer Menge eindeutig identifizierbarer URLs (canonical URLs) zum Service Metadatendokument, zu den Kacheln, und zu den FeatureInfo Dokumenten. Dabei ist der Download der Service Metadaten Ausgangspunkt für die Dienstbeschreibung. Die URL für das Dokument kann beliebig gewählt werden. Folgende Syntax wird empfohlen:

```
http://wmts.geodaten.htwdd/rest/1.0.0/WMTSCapabilities.xml
```

Der Teil "http://wmts.geodaten.htwdd/rest/" stellt ein Beispiel für eine Basis URL dar. Vorgeschrieben wird die Repräsentation des Service Metadatenmodells als valides XML. Es wird aber keine Einschränkung auf weitere Formate vorgenommen. So besteht die Möglichkeit die Metadaten in anderen verbreiteten Formaten wie z. B. JSON[25] auszuliefern. Entsprechend der REST Architektur würde sich die URL zum Serviceendpunkt nur an der Dateiendung unterscheiden:

```
http://wmts.geodaten.htwdd/rest/1.0.0/WMTSCapabilities.json
```

Jeder Layer in den Service Metadaten enthält ein oder mehrere ResourceURL Elemente (vgl. Abb. 2.18). Das ResourceURL Element beschreibt mit Hilfe einer Templatesyntax einen eindeutigen URL Endpunkt für jede Kachel und jedes FeatureInfo Dokument. Eine komplette Übersicht zur Templatesyntax ist in OGC-WMTS, 2010, TABLE 32 / 34 zu finden.

Die Repräsentation des GetTile Requests aus den vorhergehenden Beispielen für SOAP und KVP lautet als RESTful HTTP-GET Request:

```
http://wmts.geodaten.htwdd/rest/1.0.0/erde/default/EPSG:31468/EPSG:31468:0/0/0.png,
```

wenn das ResourceURL Element für den Layer „erde“ wie folgt definiert wird:

```
<ResourceURL format="image/png" resourceType="tile" template="http://wmts.geoda-
ten.htwdd/rest/1.0.0/erde/default/{TileMatrixSet}/{TileMatrix}/{TileRow}/
{TileCol}.png"> .
```

25 JavaScript Object Notation

2.5.4 KML

KML (Keyhole Markup Language) ist ein auf XML-basiertes Dateiformat, welches ursprünglich durch die Firma Keyhole Corp.[26] entwickelt wurde. Seit April 2008 ist KML ein offizieller OGC Standard (OGC-KML, 2008). Eingesetzt von den Client-Komponenten der Programme Google Earth und Google Maps dienen KML-Dokumente zur Beschreibung von Geodaten.

Ein Tiled Map Service kann die Möglichkeiten von KML nutzen, um gekachelte Layer für den Earth Viewer Google Earth zu veröffentlichen. In dem Verfahren, welches auch *KML-Superoverlay* genannt wird, werden die KML-Elemente `GroundOverlay` und `NetworkLink` eingesetzt. Mit dem `NetworkLink`-Element wird ein Netzwerk aus untereinander verlinkten KML-Dokumenten, welche je nach Sichthöhe und Region in Google Earth angezeigt werden, erzeugt.

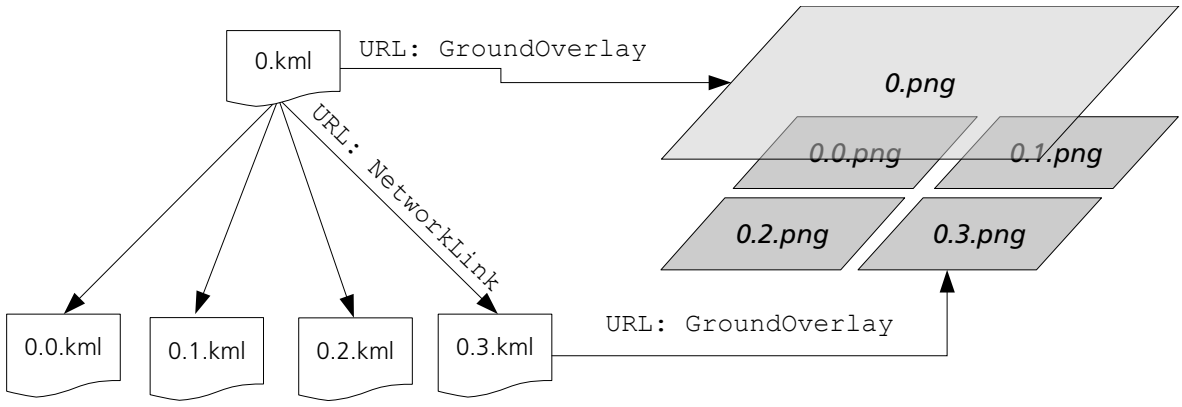

Abbildung 2.21: Dokumentenstruktur eines KML-Superoverlays

In Abbildung 2.21 wird der Aufbau eines Superoverlays veranschaulicht. Das Element `NetworkLink` verlinkt KML-Dokumente, georeferenziert sie durch Angabe einer Bounding Box und definiert deren Anzeige für einem bestimmten Maßstabsbereich. Jeder KML kann so eine Kachel in der Bildpyramide zugeordnet werden. Diese Kachel wird durch das `GroundOverlay`-Element über eine URL eingebunden und im Viewer als Textur auf der Erdoberfläche angezeigt. Stufenloses Zoomen ist durch Interpolation der Kacheln im Viewer möglich.

Kartendienste, die eine *KML-Superoverlay* Schnittstelle anbieten, generieren die KML-Dateien auf Anfrage der Clients. Als Schnittstellenbeschreibung dient ein einfaches KML-Dokument, welches weiterführend die dynamisch erzeugten Dateien verlinkt.

26 Die Firma Keyhole Corp. wurde im Oktober 2004 von Google Inc. übernommen.

Das Einbinden der einzelnen Kacheln kann über jede Schnittstelle erfolgen, die Requests auf eine eindeutige Kachel-URL erlaubt. Da der KML-Standard nur ein gültiges Referenzsystem definiert (OGC-KML, 2008, ANNEX B), muss für die Verwendung von *KML-Superoverlays* eine entsprechende Projektion (z. B. EPSG:4326) unterstützt werden.

3 Implementierung

3.1 Zielsetzung

Die praktische Aufgabe dieser Diplomarbeit umfasst die Implementierung eines kachelnden Kartendienstes im Labor Geoinformatik der Hochschule für Technik und Wirtschaft Dresden unter Verwendung von Open Source Software. Als Datenquellen sollen beliebige OGC WMS eingebunden werden können. Die Bereitstellung von gekachelten Datenquellen erfolgt beispielhaft durch den Meilenblätter WMS der HTW sowie einer Auswahl von WMS der Basiskarte Sachsen[1]. Die Einbindung weiterer WMS soll möglich sein.

Im folgenden Unterkapitel werden die Anforderungen an einen Tiled Map Service mit Sicht auf die Gegebenheiten im Labor Geoinformatik definiert.

1 Geodatendienste des Staatsbetriebs Geobasisinformation und Vermessung Sachsen (GeoSN)

3.2 Anforderungsanalyse

3.2.1 Schnittstellen

Das allgemeine Ziel eines Web Services ist die Interoperabilität. Das kann nur durch Unterstützung und Einhaltung offener Standards gewährleistet werden. Aktuell bietet nur das OGC einen offenen Implementierungsstandard für kachelnde Kartendienste an. Allerdings muss auch die Relevanz anderer Spezifikationen und eventuell nicht offiziell dokumentierter Schnittstellen berücksichtigt werden. Gerade die Entwicklung von Tiled Map Services ist geprägt durch kommerzielle Anbieter, die seit 2005 auf dem Markt sind. Google Maps, Yahoo Maps und Bing Maps setzen auf eine kompatible Datenstruktur, die in ihren Auflösungen und Kachelgrößen übereinstimmen. Diese Anbieter stellen darüber hinaus gut dokumentierte APIs zur Verfügung, die eine Kombination mit externen Kartendiensten über einfache REST Schnittstellen ermöglichen.

Im Gegensatz zu den proprietären APIs ermöglicht die freie JavaScript-Bibliothek Openlayers[2] das Einbinden von gekachelten Kartendiensten über verschiedene Schnittstellen. Die unter der BSD Lizenz[3] stehende Bibliothek bildet die Grundlage vieler freier Web-Mapping-Anwendungen. Aktuell ermöglicht sie in der Version 2.9.1 die Integration von WMS-C, TMS und erlaubt den Zugriff auf die ArcGIS Server REST Schnittstelle[4]. Eine Unterstützung von WMTS ist noch in der Entwicklung, wird aber in der Version 2.10 eingeführt [OPENLAYERS, 2010].

Wichtigste Anforderung für den zu implementierenden Dienst ist eine übereinstimmende Schnittstelle mit dem Tile Cache Client, der im Zuge der Diplomarbeit von Herrn Naumann (DANAUM, 2010) entsteht. Der in dieser Diplomarbeit gewählte Tile Cache Client basiert auf dem Openlayers Framework (Version 2.9.1) und unterstützt damit die Schnittstellen WMS-C und TMS.

Da WMTS als Weiterentwicklung von WMS-C und TMS angesehen werden kann und eine Unterstützung von standardkonformen Clients in Zukunft wünschenswert ist, wird diese Schnittstelle als zusätzliche Anforderung mit aufgenommen.

Kachelnde Kartendienste benötigen für den Aufbau eines Datenbestandes eine Datenquelle. Grundsätzlich gibt es zwei Varianten diese Abhängigkeit zu implementieren. Ist der Dienst unabhängig von einer Datenquelle implementiert, greift er in der Regel über standardisierte Webser-

2 http://openlayers.org/
3 http://www.opensource.org/licenses/bsd-license.php
4 http://resources.esri.com/arcgisserver/index.cfm?fa=services

vice Schnittstellen auf klassische Kartendienste zu. In einer Client-Server Umgebung übernimmt er also beide Rollen. Die Kommunikation über auf das System begrenzte Programmierschnittstellen (API) mit der Datenquelle stellt die zweite Implementierungsvariante dar. Diese Lösung erzeugt eine Abhängigkeit zu existierenden Programmbibliotheken, erweitert jedoch die Zahl der unterstützten Datenformate und Schnittstellen um ein Vielfaches. So ermöglicht die freie Programmbibliothek GDAL die Einbindung von aktuell 114 Datenformaten in andere Anwendungen [GDAL-FORMATS, 2010]. Auch der verbreitete MapServer verfügt mit MapScript über eine Schnittstelle, auf die unter anderem mit Sprachen wie Perl, PHP, Python und Java zugegriffen werden kann [MAPSRV-MAPSCRIPT, 2010]. Durch die Aufgabenstellung dieser Diplomarbeit wird WMS als Clientschnittstelle des Dienstes festgelegt. Diese Entscheidung erlaubt eine von der vorhandenen Infrastruktur unabhängige Implementation.

Die Erzeugung und die Verwaltung eines zu der Datenquelle redundanten Datenbestandes sind grundlegende Bestandteile von Tiled Map Services. Dieses Vorgehen führt allerdings auch zu Problemen (vgl. Kapitel 2.4.1.2), die durch den Dienst möglichst schnell und automatisiert gelöst werden sollten. Neben den eigentlichen Daten benötigt ein kachelnder Kartendienst also Informationen darüber, wie lange eine bestimmte Kachel gültig ist, bevor sie neu generiert werden muss, oder muss es ermöglichen, bestimmte Kacheln, definierte Gebiete und Zoomstufen auf Anfrage zu aktualisieren. Der zu implementierende Dienst sollte über eine plattformunabhängige Schnittstelle verfügen, die es erlaubt die Aktualität bzw. die Gültigkeit der Daten im Cache zu garantieren. Die Anforderung definiert vorerst keinen konkreten Schnittstellenstandard, da momentan keine Spezifikationen für diesen speziellen Anwendungsfall existieren.

In Abbildung 3.1 werden zusammenfassend die erforderlichen Schnittstellen veranschaulicht.

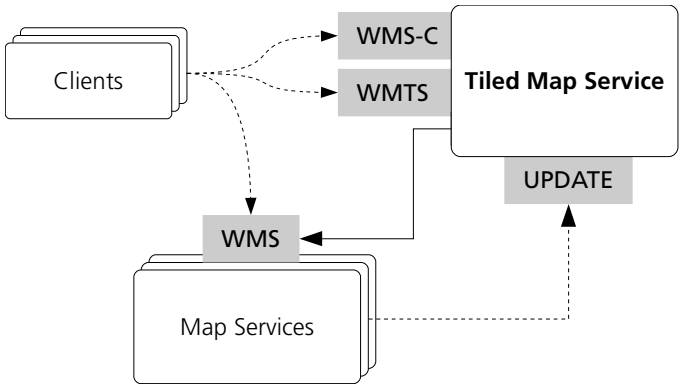

Abbildung 3.1: Schnittstellenanforderung an den Tiled Map Service

3.2.2 Plattform

Das Labor Geoinformatik der HTW setzt grundsätzlich Windows als Betriebssystem ein. Die Anforderung besteht also in erster Linie darin, dass der zu implementierende Dienst unter diesem Betriebssystem lauffähig ist. Trotz der im Labor vorhandenen homogenen Systemstruktur kann Plattformunabhängigkeit ein vorteilhaftes Ziel sein. So ermöglichen systemferne Programmiersprachen wie Java, Perl oder Python das Implementieren von Anwendungen mit wenigen Abhängigkeiten zum Betriebssystem und verringern dadurch den Wartungs- und Konfigurationsaufwand.

3.2.3 Hardwareressourcen

Im Gegensatz zu klassischen Kartendiensten ist Rechengeschwindigkeit nicht ausschlaggebend für die Performance des Dienstes. Da Rastergrafiken aus mehreren Quellen zwischengespeichert werden, konzentrieren sich die Anforderungen auf möglichst viel leistungsfähigen Massenspeicher und eine schnellen Netzwerkanbindung.

Die vom Labor Geoinformatik zur Verfügung gestellte Serverhardware besteht aus einem Dual Prozessor Mainboard mit zwei Intel Xeon E5530 Prozessoren (jeweils 4 Kerne), 16 GB Ram und zwei 450 GB Festplatten im RAID 1 Verbund[5]. Das System ist mit 1 Gbit/s in das Netzwerk der HTW eingebunden.

Eine grobe Abschätzung der zu erwartenden Datenmengen ergab einen Speicherbedarf von 110 GB. Diese Angabe basiert auf der Annahme, dass vier Layer mit jeweils zwei Projektionen und

5 RAID (Redundant Array of Independent Disks), RAID 1 organisiert 2 Festplatten zu einer logischen Festplatte gleicher Größe und verbessert so die Datensicherheit und Performance.

zehn Zoomstufen vollständig auf dem Server gespeichert werden und die Kachelgröße durchschnittlich 20 KB beträgt (vgl. Listing 2.1). Die angegebene Zoomstufenanzahl ermöglicht die Darstellung der Karten in einer Maßstabsreihe von ca. 1:2 000 000 bis 1: 4 000. Der kleinste Maßstab erlaubt eine Übersicht von ganz Sachsen in einem Kartenfenster von 600 mal 450 Pixel. Der größte Maßstab wurde aus den Meilenblätterscans[6] abgeleitet, die mit einer Auflösung von einem Meter pro Pixel vorliegen. Diese Auflösung ergibt bei Umrechnung über die standardisierte Bildschirmauflösung von 0.28 mm einen Maximalmaßstab von 1:3 571. Größere Maßstäbe hätten keinen Informationsgewinn zur Folge und könnten bei der Darstellung Treppeneffekte hervorrufen.

Für die Implementation wurden entsprechend der ermittelten Anforderungen die Ressourcen des Systems in einer virtuellen Maschine mit zwei Prozessoren, 2 GB Arbeitsspeicher und 200 GB Massenspeicher (25 GB System- und 175 GB Datenpartition) bereitgestellt.

3.3 Bestandsanalyse

Ziel dieses Unterkapitels ist es, einen allgemeinen Überblick über die wichtigsten freien, aber auch proprietären Tiled Map Services zu geben. Die Bestandsanalyse soll unter Berücksichtigung der gestellten Anforderungen eine Schlussfolgerung auf die umzusetzende Lösung zulassen.

6 Diese Scans werden vom Meilenblätter WMS als Datenquelle benutzt.

3.3.1 Geowebcache

Geowebcache[7] ist ein Open Source Projekt innerhalb der Open Source Geospatial Foundation, entstanden im Rahmen des Google Summer of Code[8] 2007 unter dem Namen JTilecache. Heute wird Geowebcache als Teil des ebenfalls freien GeoServer[9] unter der GNU LGPL v3 (GNU Lesser General Public License) angeboten. Durch funktionale Abgrenzung kann der in Java Servlet Technologie implementierte Dienst allerdings auch unabhängig vom GeoServer betrieben werden. Die aktuelle stabile Version zum Zeitpunkt dieser Diplomarbeit ist Geowebcache 1.2.2, welche eine umfangreiche und weit entwickelte Implementation eines Tiled Map Services darstellt.

Folgende Serverschnittstellen werden durch Geowebcache (GWC) angeboten:

- WMS (bis Version 1.3) / WMS-C
- TMS
- WMTS (Version 1.0.0, mit dem KVP Interface)
- KML
- Google Maps
- Virtual Earth (Bing Maps)
- REST (für die Konfiguration / Aktualisierung des Caches)

Plattformunabhängige Clientschnittstellen:

- GeoRSS[10] (für die Aktualisierung des Caches)
- WMS (bis Version 1.3) (für die Erstellung des Caches)

Geowebcache ist modular aufgebaut, das heisst, jede einzelne Schnittstelle ist nur vom Kern (gwc-core) des Dienstes abhängig. Die Entkopplung der einzelnen Applikationskomponenten beruht auf dem Architekturkonzept des zugrunde liegenden Java Anwendungsframeworks Spring[11]. Dementsprechend erfolgt die Konfiguration des Dienstes modular durch XML-Dokumente. In der Abbildung 3.2 werden alle Konfigurationselemente einer mit Apache Tomcat realisierten

7 http://geowebcache.org
8 http://code.google.com/intl/de/soc/
9 http://geoserver.org
10 http://www.georss.org
11 http://www.springsource.org

Geowebcache-Installation dargestellt. Allgemeine Einstellungen sowie die Definition einzelner, durch den Dienst angebotener Layer erfolgt in der obersten Schicht. Schnittstellen-spezifische Angaben, aber auch die Voreinstellung von Dateipfaden für das Zwischenspeichern von Kacheln werden nach der Spring Bean Definition in XML-Dokumenten vorgenommen.

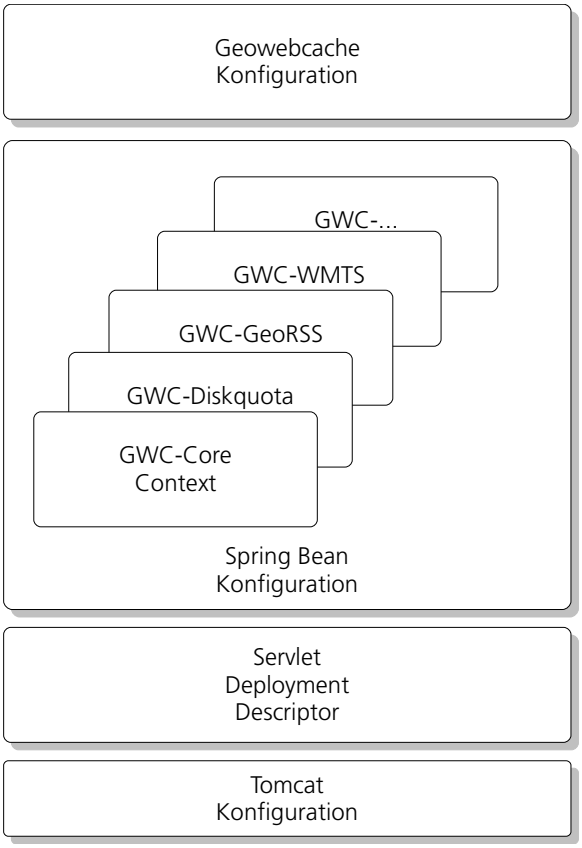

Abbildung 3.2: Konfigurationsdokumente einer Geowebcache-Installation

In der einfachsten Konfigurationsvariante genügt die URL eines WMS. Dessen Service Metadaten werden als Grundlage für die automatisierte Bereitstellung der Layer und Kachelprofile verwendet. Die erweiterten Einstellungen erlauben das Einbinden mehrerer WMS, die Auswahl bestimmter Layer und die Definition von Kachelprofilen.

Geowebcache hebt sich durch den Funktionsumfang von anderen Tiled Maps Services ab. Besonders erwähnenswert sind die Schnittstellen, aber auch Konfigurationsmöglichkeiten, die eine flexible Aktualisierung des gekachelten Datenbestandes erlauben. So ist es möglich, jeweils pro Layer und Zoomstufe die Lebensdauer der Kacheln im Server Cache zu bestimmen. Weiterhin erlaubt GWC zoomstufenspezifische HTTP Header, welche die Validität der Kacheln im Browsercache bestimmen. Die REST-Schnittstelle steuert Seedingprozesse (Erstellen / Erneuern /

Löschen), individuell begrenzt auf eine bestimmte Bounding Box in einer oder mehreren Zoomstufen. Anders funktioniert die GeoRSS-Clientschnittstelle. Durch ein konfiguriertes Pollintervall wird in regelmäßigen Zeitabständen ein GeoRSS-Feed von der Datenquelle abgefragt. GeoRSS beschreibt mit GML eine Menge von Geometrien, über die Geowebcache das zu aktualisierende Gebiet ermittelt und einen Seedingprozess auslöst (Listing 3.1). Die Definition von konkreten Geometrien verhindert zusätzlich das Cachen von Kacheln ohne Inhalt. Dieses Phänomen tritt auf, wenn ein Gebiet, für das nur unregelmäßig Daten existieren, über eine Bounding Box eingeschränkt wird.

```
1  <feed xmlns="http://www.w3.org/2005/Atom"
       xmlns:georss="http://www.georss.org/georss"
       xmlns:gml="http://www.opengis.net/gml">
2    <title>GeoRSS feed samples</title>
3    <link href="http://wms.htwdd/georss" />
4    <updated>2010-08-13T18:30:02Z</updated>
5    <entry>
6      <title>Polygon sample</title>
7      <link href="http://wms.htwdd/georss/123" />
8      <id>123</id>
9      <updated>2010-08-17T07:02:32Z</updated>
10     <content>a sample geometry</content>
11     <georss:where>
12       <gml:Polygon>
13         <gml:exterior>
14           <gml:LinearRing>
15             <gml:posList>45.256 -110.45 46.46 -109.48 43.84
                 -109.86 45.8 -109.2 45.256 -110.45</gml:posList>
16           </gml:LinearRing>
17         </gml:exterior>
18       </gml:Polygon>
19     </georss:where>
20   </entry>
21   <entry>...</entry>
22 </feed>
```

Listing 3.1: Beispiel einer GeoRSS mit einer einfachen Geometrie

Beide Update-Schnittstellen erfordern eine entsprechende Implementierung an der Datenquelle, ermöglichen aber auch die volle Kontrolle über den im Cache gespeicherten Datenbestand. Neben der GeoRSS- und REST-Schnittstelle bietet GWC ein einfaches Servlet an, welches die Cacheverwaltung mittels HTML im Browser ermöglicht.

Über das in der Version 1.2.2 eingeführte Diskquota Modul kann der Speicherbedarf einzelner Layer begrenzt werden. Dies erfolgt konfigurierbar über die Verdrängung gespeicherter Kacheln nach den Regeln *Least Frequently Used* oder *Least Recently Used* (vgl. Kap. 2.4.1.2). Die Optimierung des Datenbestandes nach diesen Regeln kann in frei definierbaren Intervallen vorgenommen werden.

Überwachung von Speicherbedarf und Schnittstellen, die einen Abgleich mit Datenquellen ermöglichen, sind Alleinstellungsmerkmale auf dem Gebiet der Tiled Map Services. Die Implementierung in Java macht den Dienst plattformunabhängig. Aber auch die für den Zugriff auf Datenquellen verwendete standardisierte Webserviceschnittstelle WMS erlaubt einen flexiblen Einsatz und vereinfacht die Integration in vorhandene Infrastrukturen.

3.3.2 TileCache

TileCache[12] wurde im November 2006 von Christopher Schmidt und James Schuyler als einer der ersten Tiled Map Services programmiert [Schmidt, 2006]. In der aktuellen Version 2.10 wird der in Python implementierte Dienst als freie Software unter der BSD Lizenz[13] von der Firma Metacarta[14] angeboten.

Serverschnittstellen, die durch TileCache angeboten werden, sind:

- WMS-C
- TMS
- KML

Implementierte Clientschnittstellen:

- WMS
- ArcXML

ArcXML ist ein proprietärer XML-Standard der Firma ESRI, der in diesem Fall für die Einbindung von ArcIMS Web Map Server in TileCache genutzt werden kann. Neben WMS und ArcXML kann TileCache über API Aufrufe mit MapServer, Mapnik[15] oder GDAL kommunizieren. Diese Programme nehmen die Funktion eines Rendering Backends ein und erweitern so die vom Dienst unterstützten Schnittstellen und Datenformate, erfordern allerdings eine parallele Installation zu TileCache (vgl. Kap. 3.2.1).

Im Unterschied zum Geowebcache erlaubt TileCache das Speichern der Kacheln nicht nur auf dem vom System zur Verfügung gestellten Massenspeicher. Plugins erweitern die Software um sogenannte Cache Backends, welche auch den Arbeitsspeicher oder externe Webdienste als Spei-

12 http://www.tilecache.org
13 http://www.opensource.org/licenses/bsd-license.php
14 http://www.metacarta.com
15 Mapnik rendert OpenStreetMap Daten (http://mapnik.org)

chermedium zugänglich machen. So ist es zum Beispiel möglich, den Amazon Simple Storage Service (Amazon S3)[16] als Speicher für die Bildpyramiden zu verwenden. Auf diese Weise können die Vorteile der von Amazon angebotenen kostenpflichtigen Infrastruktur genutzt werden (vgl. Kap. 2.4.1.3, CDN).

Die Konfiguration von TileCache erfolgt in einer einfachen Textdatei, die nach dem Python Konfigurationsschema[17] formatiert ist. Unterteilt in Sektionen können Einstellungen für den Dienst allgemein, Layer und Cache Backend vorgenommen werden.

Eine Webservice-Schnittstelle für die Cacheverwaltung wird durch den Dienst nicht angeboten. Der Seedingprozess kann entweder sequenziell durch Kachelrequests erfolgen oder über einen Scriptaufruf in der Systemkonsole gesteuert werden. Die Optimierung des gesamten durch den Dienst erstellten Datenbestandes ist nach der Regel *Least Recently Used* ebenfalls nur über einen manuellen Scriptaufruf in der Konsole möglich. Die Scriptaufrufe können allerdings durch Einbinden in den Taskmanager des Systems automatisiert werden.

TileCache kann entweder über einen mitgelieferten Python HTTP-Server betrieben werden oder über eine CGI / FCGI Schnittstelle[18] mit vorhandenen HTTP-Servern kommunizieren. Die Einbindung in einen Apache HTTP-Server kann über das Modul mod_python erfolgen.

16 http://aws.amazon.com/de/s3/
17 http://docs.python.org/library/configparser.html
18 (Fast) Common Gateway Interface, RFC 3875

3.3.3 ArcGIS Server

ArcGIS Server ist ein Teil der umfangreichen kommerziellen Softwaresuite ArcGIS der Firma ESRI[19]. Aktuell in der Version 10 können durch ArcGIS Server Kartendienste, Geoprozessingdienste, Geodatendienste, Netzwerkanalysedienste, Suchdienste sowie Geokodierungs- und Geometriedienste angeboten werden. Überwiegend werden diese Dienste über SOAP und REST Schnittstellen angesprochen. Die Unterstützung von OGC Standards beschränkt sich auf KML, WMS, WCS und WFS. Das Produkt ArcGIS Server enthält neben den Diensten auch ein umfangreiches Sortiment an Webservice APIs zur Integration in Clientanwendnungen. Konkret stellt ArcGIS Server für Java, JavaScript, Flex, .NET und Silverlight APIs zur Verfügung, aber auch Erweiterungen für die API von Bing Maps und Google Maps. [ARCGIS, 2010]

Die Bereitstellung von kachelnden Kartendienste ist in ArcGIS Server nur über eine nicht standardisierte REST Schnittstelle möglich. Das umfangreiche Angebot an APIs reduziert allerdings den Integrationsaufwand für vorhandene Clientanwendungen.

Der Erstellungsprozess eines kachelnden Kartendienstes ist fest in die Struktur der ArcGIS Softwaresuite integriert. Im Autorenwerkzeug ArcMap wird eine Karte mit festen Zoomstufen erstellt. Datenquellen für diese Karte können von ArcMap unterstützte Dateiformate[20] und externe Webdienste sein. Die fertige Karte wird in die Anwendung ArcCatalog geladen und dort auf einen ArcGIS Server publiziert. Als Austauschformat zwischen den Anwendungen dient das proprietäre Dateiformat MXD oder MSD (Map Service Definition).

Der Seedingprozess kann durch Zugriff der Clients auf die Kacheln oder durch manuellen Eingriff erfolgen. Wie auch in Geowebcache können die von der Aktualisierung betroffenen Gebiete über Geometrien, in diesem Fall mit Shape Dateien, eingegrenzt werden. Mechanismen für die automatische Kachelaktualisierung existieren in der aktuellen Version nicht. Die Grundlagen für einen automatischen Abgleich sind aber geschaffen, da ArcGIS über verschiedene Scriptsprachen programmierbar ist. [ARCGIS-HELP, 2010]

19 http://www.esri.com

20 http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html

3.3.4 MapProxy

Die Entwickler von MapProxy[21] bezeichnen ihre Software allgemein als Proxy für geographische Daten. Im grundlegenden Konzept unterscheidet sich der unter GNU AGPL (GNU Affero General Public License) angebotene Dienst nicht von den hier vorgestellten Implementationen. Allerdings versucht MapProxy mit Kachelung auch normale WMS Clients zu beschleunigen. Dies wird erreicht, in dem die Kacheln der gespeicherten Bildpyramide für WMS Anfragen wieder zusammengesetzt und beliebige Zoomstufen durch Interpolation erzeugt werden. Um den enormen Datenumfang, der durch seperate Bildpyramiden entstehen kann, zu reduzieren, werden zusätzliche Bildformate und Projektionen durch integrietes Rendering separat für jeden Clientrequest erzeugt.

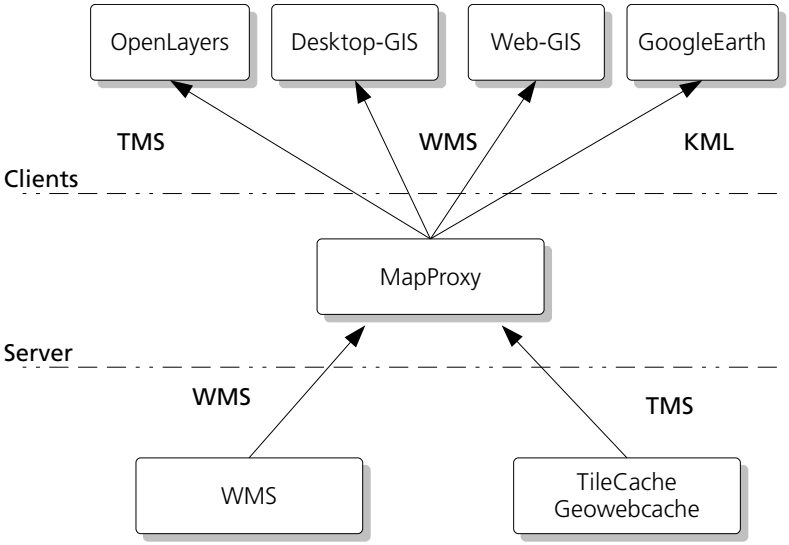

Abbildung 3.3: MapProxy als "man in the middle" zwischen Map Clients und Map Server

Abbildung 3.3 zeigt die vom Dienst unterstützen Server- und Clientschnittstellen. Es wird deutlich, dass MapProxy auch dazu verwendet werden kann, gekachelte Dienste in klassische Kartendienste zu wandeln.

Der Dienst lässt sich sehr einfach konfigurieren. Unterteilt in separate Dateien können Einstellungen für Layer, Datenquellen sowie Seedingprozesse in der Auszeichnungssprache YAML[22] vorgenommen werden.

Der Seedingprozess kann durch Requests der Clients oder durch einen Scriptaufruf in der Betriebsystemkonsole ausgeführt werden. Die Konfiguration ermöglicht, das zu aktualisierende

21 http://mapproxy.org
22 http://www.yaml.org

Gebiet über Geometrien (Shape Datei / Koordinatenliste) einzuschränken. Die Speichernutzung der Bildpyramiden lässt sich nicht direkt einschränken. Nur über eine konfigurierbare Regel lassen sich Kacheln bestimmten Alters aus dem Dateisystem löschen.

Neben den Vorteilen (geringer Speicherverbrauch, variable Verwendung) ergeben sich durch das interne Rendering auch Nachteile. Da das Transformieren der Kacheln in dieser Software eine zeitkritische Aufgabe ist, wird ein einfacher Algorithmus eingesetzt. Dieser überführt die Pixel der Rastergrafiken nur in Abhängigkeit von einer einfachen Gitterstruktur in ein anderes Koordinatensystem (affine Transformation). Das führt zu Abweichungen, die in den Quelldaten nicht vorhanden sind. Diese Ungenauigkeiten entstehen auch, wenn Zoomstufen durch WMS Anfragen interpoliert werden.

Eine Plattformunabhängigkeit besteht nicht direkt. MapProxy ist in Python geschrieben, benötigt allerdings für die Funktion vom Betriebssystem abhängige Bibliotheken. Aus diesem Grund lässt sich eine komplette Installation nicht auf andere Betriebssysteme übertragen.

Wie TileCache kann MapProxy über CGI oder FCGI in vorhandene HTTP-Server eingebunden werden. Für den Apache Webserver kann das Modul mod_wsgi[23] die Kommunikation zwischen Dienst und HTTP-Server übernehmen. Für den schnellen Testbetrieb kann der integrierte Python Webserver Paste[24] verwendet werden.

23 http://code.google.com/p/modwsgi
24 http://pythonpaste.org

3.3.5 mod_tile

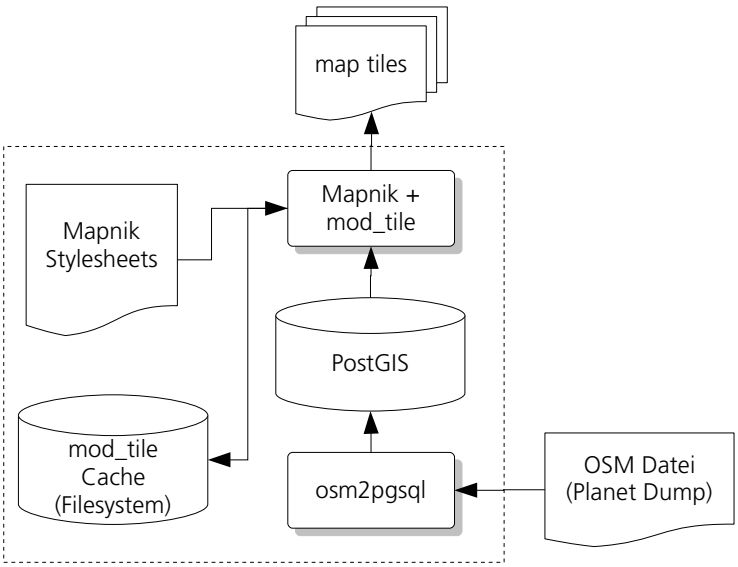

Abbildung 3.4: Auszug aus der OSM Infrastruktur nach [OSM COMPONENTS, 2010]

mod_tile ist ein Modul für den Apache Webserver und wurde speziell für die OpenStreetMap (OSM) Infrastruktur entwickelt. Der Quellcode wird unter GNU GPL v2 (GNU General Public License) veröffentlicht. Da das Modul auf den Einsatz mit OpenStreetMap spezialisiert ist, beschränken sich die unterstützten Schnittstellen auf ein Minimum. Abbildung 3.4 zeigt einen Auschnitt der OSM Infrastruktur und ordnet mod_tile in diese ein. Es wird deutlich, dass das Modul nur in Abhängkeit von Mapnik funktioniert. Mapnik übernimmt die Funktion des Renderers und verwendet in diesem Fall eine PostGIS Datenbank als Quelle. Neben PostGIS kann Mapnik ESRI Shapefiles, TIFF Raster, OSM XML sowie alle von GDAL unterstützten Formate lesen. Die Serverschnittstelle ist an TMS angelehnt, beschreibt sich aber nicht selbst durch XML Dateien, sondern hat eine festgelegte URL Struktur nach dem Schema `http://{server}/{layer}/{z}/{x}/{y}.{format}`.

Erstellung und Aktualisierung von Kacheln in mod_tile kann neben Clientrequests auch durch Scriptaufrufe erfolgen. Diese Scripte sind allerdings nur auf Unix Betriebssystemen lauffähig. Auch der gesamte C Quellcode von mod_tile ist momentan nur auf Unix oder unixähnlichen Betriebssystemen kompilierbar. [MOD_TILE, 2010]

3.3.6 MapServer

Mapserver[25], aufgrund seiner ursprünglichen Entwicklung an der University of Minnesota auch als UMN MapServer bekannt, ist eine in C geschriebene Open Source Rendering-Engine für Geodaten. Zum Zeitpunkt der Untersuchung lag die Software in der Version 5.6.5 vor und unterstützte mit dem *Tile Mode* auch das Kacheln nach dem in OGC-WMTS, 2010, ANNEX E3 beschriebenen Google Maps Kachelprofil. [MAPSRV-TILE MODE, 2010]

Der *Tile Mode* von MapServer implementiert keinen echten kachelnden Kartendienst, sondern ermöglicht nur über eine nicht standardisierte Schnittstelle den Zugriff auf einzelne Kacheln. Jede Kachel wird unabhängig von anderen Requests dynamisch vom MapServer gerendert. So kann der Dienst nicht von den Geschwindigkeitsvorteilen, die durch serverseitiges Cachen entstehen, profitieren.

```
1 mbl = new google.maps.ImageMapType({
2   getTileUrl: function(coord, zoom) {
3     return "http://localhost/mapsrv?"
4            +"mode=tile&layers=Bmbl_vollton&tilemode=gmap&tile="
5            + coord.x +"+"+ coord.y +"+"+ zoom;
6   },
7   tileSize: new google.maps.Size(256, 256),
8   isPng:false,maxZoom:16,minZoom:8,opacity:1,name:"Meilenbl&aumltter"
9 });
```

Listing 3.2: Integration von MapServer in die Google Maps API v3

Die Schnittstelle eignet sich vorallem für die schnelle Integration von Diensten in proprietäre Kartenclients. Es werden speziell für Google Maps und Bing Maps angepasste URL Schemata angeboten, welche mit wenig Aufwand neue Kartenlayer dem Client hinzufügen. Listing 3.2 zeigt den relevanten Ausschnitt für die Integration mit Hilfe der Google Maps JavaScript API. Das URL Template für das gezeigte Beispiel ist: `http://{server}/{mapservercgi}?mode=tile&layers={[layer]}&tilemode=gmap&tile={x}+{y}+{zoom}`. Eine Beispielintegration ist auf der digitalen Anlage abgelgt (Anhang D, Beipiel Clients).

3.3.7 Schlussfolgerung

Die untersuchten Tiled Map Services differenzieren sich stark in den unterstützten Schnittstellen, Funktionen sowie Konfigurationsaufwand. Für die Integration in das Labor Geoinformatik eignen sich vorallem TileCache, MapProxy und Geowebcache. TileCache überzeugt durch die Fähigkeit direkt mit MapServer zu kommunizieren, welcher bereits im Labor eingesetzt wird. Beliebige Projektionen und Grafikformate, geringe Speichernutzung und einfache Konfiguration

25 http://mapserver.org

sprechen für MapProxy. Geowebcache erfüllt allerdings als einziges Produkt alle in den Anforderungen definierten Punkte und überzeugt durch einfache Konfiguration und Plattformunabhängigkeit.

Die Entscheidung für die Implementation eines Tiled Map Services im Labor Geoinformatik fiel aus den genannten Gründen auf Geowebcache.

3.4 Installation und Konfiguration

Die Installation erfolgt auf der in Kapitel 3.2.3 vorgestellten virtuellen Maschine (KS14) unter dem Betriebssystem Windows 2003 Service Pack 2. Im Netzwerk der HTW ist das System unter der IP-Adresse 141.56.4.14 (DNS-Name ks14.htwdd) über eine Remotedesktopverbindung erreichbar. Die für die Installation benötigte Software steht auf der beigelegten CD in der eingesetzten Version zur Verfügung.

3.4.1 Java SE Runtime Environment (JRE)

Die Java Laufzeitumgebung ist Voraussetung für die Installation der weiteren Komponenten Apache Tomcat und Geowebcache. Der Download erfolgt über die Java-Webseite von Oracle (http://www.java.com/download). Im Rahmen dieser Installation wurde mit der Version JRE 6, Update 21 gearbeitet. Lauffähig sind die Komponenten ab der JRE Version 5, welche als freie und offene Software (Java Research License bzw. Sun Community Source License Version 2.8) angeboten wird.

3.4.2 Apache Tomcat

Geowebcache verwendet für die implementierten Webdienste die Java Servlet Technologie. Entsprechend benötigt GWC einen Webserver, der diese Technologie unterstützt. Neben dem freien Apache Tomcat[26] (Apache License Version 2) ist GWC kompatibel mit den ebenfalls unter Opensource Lizenz stehenden Java-Webserver Glassfish[27] (Common Development and Distribution License) und Jetty[28] (Apache License Version 2). Für die Installation auf KS14 wurde Apache Tomcat in der Version 6.0.29 eingesetzt. Der Download erfolgte als Binär Distribution in der 32-bit Windows ZIP-Variante (http://tomcat.apache.org/download-60.cgi).

26 http://tomcat.apache.org
27 https://glassfish.dev.java.net
28 http://jetty.codehaus.org/jetty

Entpackt nach D:\apache-tomcat-6.0.29 kann die Installation des Systemdienstes nach den in Listing 3.3 gezeigten Befehlen in der Systemkonsole erfolgen.

```
1  D:\apache-tomcat-6.0.29\bin>set JAVA_HOME=C:\Programme\Java\jre6

2  D:\apache-tomcat-6.0.29\bin>set PATH=%PATH%;%JAVA_HOME%\bin

3  D:\apache-tomcat-6.0.29\bin>service.bat install
   Installing the service 'Tomcat6'
   Using CATALINA_HOME: "D:\apache-tomcat-6.0.29"
   Using CATALINA_BASE: "D:\apache-tomcat-6.0.29"
   Using JAVA_HOME:     "C:\Programme\Java\jre6"
   Using JVM:   "C:\Programme\Java\jre6\bin\client\jvm.dll"
   The service 'Tomcat6' has been installed.
```

Listing 3.3: Konsolenbefehle für die Apache Tomcat Installation

Der Systemdienst automatisiert das Starten und Beenden des Webservers bei einem Neustart des Betriebssystems. Die manuelle Start/Stop - Kontrolle erfolgt durch das Konfigurationsprogramm `tomcat6w.exe`, welches sich im `bin`-Ordner der Tomcat Installation befindet.

Eine weitere Konfiguration des Webservers ist normalerweise nicht notwendig, da Tomcat das Deployment (Installation und Inbetriebnahme) von Servlets über den Browser ermöglicht. Speziell für die zu erwartenden Aufgaben wurden dennoch Änderungen in der allgemeinen Tomcat Konfiguration (`conf/server.xml`) vorgenommen. Listing 3.4 beinhaltet die relevanten Ausschnitte. Das `Connector`-Element definiert die Eigenschaften der HTTP/1.1 Schnittstelle. Nach den Hinweisen von TOMCAT-CONF, 2010 wurden die Einstellungen auf viele gleichzeitige HTTP Requests optimiert. Da Tomcat als eigenständiger Webserver auf dem System arbeitet, wurde der Standard Port von 8080 auf 80 geändert.

```
1  <Connector port="80" protocol="org.apache.coyote.http11.Http11NioProtocol"
2    acceptorThreadCount="2" useExecutor="false" maxKeepAliveRequests="200"
3    maxHttpHeaderSize="8192" maxThreads="1024" minSpareThreads="25"
4    maxSpareThreads="256" enableLookups="false" acceptCount="1024"
5    connectionTimeout="5000" disableUploadTimeout="true"
6    compression="off" />
```

Listing 3.4: Änderungen in der server.xml der Apache Tomcat Installation

Nach Start des Servers über `tomcat6w.exe` können Anwendungen in einem Web Application Archive (WAR) durch Upload einer lokalen Datei oder Angabe einer öffentlichen URL über den Tomcat Web Application Manager (http://ks14.htwdd/manager/html[29]) installiert werden.

29 Der Application Manager ist passwortgesichert. Der Zugang ist über die conf/tomcat-users.xml konfigurierbar.

3.4.3 Geowebcache

Die Projektseite von Geowebcache (http://sourceforge.net/projects/geowebcache/files) stellt eine aktuelle stabile Version als plattformunabhängige WAR-Datei zum Download zur Verfügung. Für die Installation auf KS14 wurde allerdings eine aktuellere Entwicklerversion (Revision 1186) aus dem SVN Repository (http://geowebcache.org/svn/trunk/geowebcache) verwendet, da seit dem letzten Versionssprung (1.2.2) im März 2010 viele Änderungen in den Quellcode eingegangen sind. In dieser aktuelleren Version werden Fehler in der KML-Darstellung in Google Earth, in den Metadaten für WMTS sowie in der Konfiguration von HTTP Proxy behoben. Für die Installation ist besonders eine funktionierende Proxyeinstellung wichtig, um WMS Server ausserhalb des HTW-Netzes in den Dienst einzubinden.

Ausgangspunkt für die Installation ist der Geowebcache-Servlet Ordner, so wie er in der digitalen Anlage vorhanden ist. Eine Anleitung zum Kompilieren von Geowebcache aus den aktuellen Quellen ist ebenfalls auf der CD vorhanden.

Auf KS14 wurden die bereitgestellten Anwendungen ausserhalb des Tomcat Installationsordners angelegt. Diese Maßnahme soll die Installationen unabhängig von einander halten. Die Trennung der Anwendungsordner erfordert einen entsprechenden Konfigurationseintrag in der server-.xml von Apache Tomcat (`conf/server.xml`). Listing 3.5 zeigt die vorgenommenen Einstellungen für Geowebcache und dem Tilecache Client von Herrn Naumann (DANAUM, 2010). Die `Context`-Elemente definieren für die bereitzustellenden Servlets feste Pfade im Dateisystem, Pfade für die URL der Anwendungen und Namesvorgaben für die Log-Dateien.

```
1  <Context path="" override="true"
2    docBase="D:/geowebcache-1.2-SNAPSHOT">
3      <Valve className="org.apache.catalina.valves.AccessLogValve"
4        prefix="geowebcache_access." suffix=".log" pattern="common"/>
5  </Context>
6
7  <Context path="/client" override="true"
8    docBase="D:/tilecache-client" reloadable="true">
9    <Valve className="org.apache.catalina.valves.AccessLogValve"
10     prefix="tilecache-client_access." suffix=".log" pattern="common"/>
11 </Context>
```

Listing 3.5: Context Konfiguration der Apache Tomcat servlet.xml

3.4.3.1 Dienstkonfiguration

Geowebcache verwaltet in der Standardkonfiguration alle Kacheln im Temp-Ordner des Betriebssystems. Im vorliegenden Fall befindet sich dieser Ordner auf der kleineren Systempartition (vgl. Kap. 3.2.3). Die Einstellungen für den Speicherordner der Kacheln und der zugehö-

rigen Dateidatenbank erfolgt in der Spring Bean Konfiguration `WEB-INF/geowebcache-core-context.xml`. Durch die Angabe der JDBC[30] URL für den MetaStore und dem Pfad zum Cache-Ordner werden alle Daten auf der größeren Partition (D:) abgelegt (Listing 3.6).

```
1  <bean id="gwcMetaStore"
2    class="org.geowebcache.storage.metastore.jdbc.JDBCMetaBackend"
3    destroy-method="destroy">
4    <constructor-arg value="org.h2.Driver" />
5    <constructor-arg value="jdbc:h2:file:D:/cache/h2_metastore" />
6    <constructor-arg value="sa" />
7    <constructor-arg value="" />
8  </bean>
9
10 <bean id="gwcBlobStore"
11   class="org.geowebcache.storage.blobstore.file.FileBlobStore"
12   destroy-method="destroy">
13   <constructor-arg value="D:/cache" />
14 </bean>
```

Listing 3.6: Konfiguration des Geowebcache Cacheordners

Die Konfiguration der durch den Dienst angebotenen Layer wird zentral in der `WEB-INF/classes/geowebcache.xml` vorgenommen (vgl. Abbildung 3.2). Nachfolgend werden nur die für die Einbindung beliebiger WMS relevanten Punkte näher beschrieben. Die Abbildung 3.5 zeigt die grobe Unterteilung in `gridSets` und `layers` innerhalb der XML-Datei. Angelehnt an den WMTS Standard werden durch GridSets, getrennt von der Layerdefinition, Kachelprofile definiert. In diesen Profilen werden für eine bestimmte Projektion Zoomstufen, Bounding Box sowie Kachelgröße festgelegt. Layer können mehrere GridSets über eine eindeutige Kennung referenzieren. Eine weitere Einschränkung auf das für den Layer relevante Gebiet erfolgt durch die Definition von gridSubSets (vgl. Abb. 2.19 TileMatrixLimits).

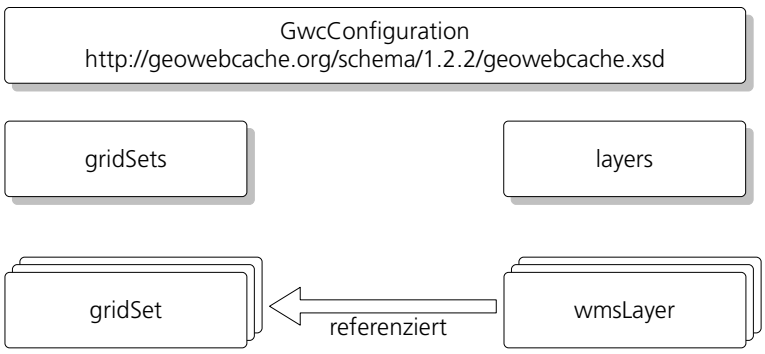

Abbildung 3.5: Struktur der geowebcache.xml

Externe WMS werden über das `wmsLayer`-Element durch Angabe der Service-URL eingebunden. Mit der Layerdefinition können weiterhin die WMS-Parameter, Version, Layers, Format und

30 Java Database Connectivity (JDBC) ist eine Einheitliche Datenbankschnittstelle für die Java-Plattform.

Style angegeben werden. Die Referenzierung eines bestimmten GridSets setzt bei dem angebenen WMS die Unterstützung der entsprechenden Projektion voraus. Geowebcache stellt zwei vordefinierte GridSets mit der Kennung EPSG:3857 und EPSG:4326 zur Verfügung. Diese entsprechen den in OGC-WMTS, 2010, ANNEX E. 3,4 standardisierten Profilen `GoogleCRS84Quad` und `GoogleMaps-Compatible`.

In Listing 3.7 wird exemplarisch der Meilenblätter WMS mit dem Layer „Bmbl_vollton“ in den Dienst eingebunden. Um das Cachen von Gebieten ohne Informationen zu verhindern, wird durch Angabe einer möglichst minimalen Bounding Box[31] das darzustellende Gebiet eingeschränkt (Listing 3.7 Zeile 12 -19). Aufgrund des im Kapitel 3.2.3 ermittelten Maximalmaßstabs von 1:3 571 für die Meilenblätter ist es erforderlich, die Zoomstufen des refenzierten GridSets zu begrenzen. Durch Angabe von „16“ im Element `zoomStop` kachelt der Dienst nur die Stufen 0 – 16 innerhalb der Layerausdehnung. Der Maßstab wird so auf 1: 4 265 begrenzt.

```
1  <wmsLayer>
2    <name>meilenblaetter</name>
3    <mimeFormats><string>image/jpeg</string></mimeFormats>
4    <wmsUrl>
5     <string>http://geoinformatik.htw-dresden.de/cgi-bin/mblsn</string>
6    </wmsUrl>
7    <wmsVersion>1.1.1</wmsVersion>
8    <wmsLayers>Bmbl_vollton</wmsLayers>
9    <gridSubsets>
10     <gridSubset>
11       <gridSetName>EPSG:3857</gridSetName>
12       <extent>
13         <coords>
14           <double>1113194.907932736</double>
15           <double>6446275.841017158</double>
16           <double>1781111.852692377</double>
17           <double>6800125.454397307</double>
18         </coords>
19       </extent>
20       <zoomStop>16</zoomStop>
21     </gridSubset>
22 </wmsLayer>
```

Listing 3.7: Beispiel für Integration des Meilenblätter WMS in GWC

Für das in Zusammenarbeit mit Herrn Naumann (DANAUM, 2010) erstellte Tile Service Profil – Sachsen (Anhang C) ist die Definition von GridSets in den Projektionen EPSG:31468 und EPSG:31469 (Gauss-Krüger 4./5. Streifen) notwending. Listing 3.8 beinhaltet die GridSet-Definition für die Projektion EPSG:31468. Die Ausdehnung des GridSets entspricht der gesamten Fläche, die durch EPSG-REG, 2010 für Gauss-Krüger 3. und 4. Streifen definiert ist (10.5° bis 16.5

31 Die Werte für die Bounding Box können in der Regel aus den Service Metadaten des eingebundenen WMS entnommen werden.

ö.L, 0° bis 84° n. Br.). Nach den Profilvorgaben wurden die Zoomlevels als Maßstabszahl (`scaleDenominators`) festgelegt. Intern erfolgt die Umrechnung von Maßstab nach Resolution über die angegebene Pixelgröße (`pixelSize`) und dem Wert Meter pro Einheit (`metersPerUnit`). Durch diese vorerst allgemeinen Festlegungen kann das GridSet auch anderen Layern für das Gebiet Sachsen zugeordnet werden.

```
<gridSet>
  <name>EPSG:31468</name>
  <srs><number>31468</number></srs>
  <extent>
    <coords>
      <double>4333108</double>
      <double>0</double>
      <double>5001550</double>
      <double>9333388</double>
    </coords>
  </extent>
  <pixelSize>0.00028</pixelSize>
  <metersPerUnit>1</metersPerUnit>
  <scaleDenominators>
    <double>2183915.093862179</double>
    <double>1091957.546931089</double>
    <double>545978.7734655447</double>
    <double>272989.3867327723</double>
    <double>136494.6933663862</double>
    <double>68247.34668319309</double>
    <double>34123.67334159654</double>
    <double>17061.83667079827</double>
    <double>8530.918335399136</double>
    <double>4265.459167699568</double>
    <double>2132.729583849784</double>
    <double>1066.364791924892</double>
  </scaleDenominators>
  <tileHeight>256</tileHeight>
  <tileWidth>256</tileWidth>
</gridSet>
```

Listing 3.8: GWC GridSet-Definition für EPSG:31458

In Tabelle 3.1 sind die im Rahmen dieser Diplomarbeit konfigurieten Layer aufgelistet.

Layer	GridSet	Maximalmaßstab
Meilenblätter[1]	EPSG:31468, EPSG:4326, EPSG:3857	1:4 265
Meilenblätter Blattschnitt[1]	EPSG:31468, EPSG:4326, EPSG:3857	1: 4 265
ATKIS DOP-RGB[2]	EPSG:31468	1 : 2 132
TOP.Sachsen[2]	EPSG:31468	1: 4 265

[1] http://geoinformatik.htw-dresden.de/cgi-bin/mblsn
[2] http://www.landesvermessung.sachsen.de/ias/basiskarte4/service/register

Tabelle 3.1: Konfigurierte Kartenlayer auf KS14

Im Anhang B dieser Diplomarbeit kann die komplette Konfiguration eingesehen werden. Weiterhin befindet sich auf der digitalen Anlage eine Web Application Archive - Datei bzw. der Servlet Ordner, wie er auf KS14 unter `D:/geowebcache-1.2-SNAPSHOT` vorzufinden ist.

3.4.3.2 Verwaltung

Ist Geowebcache nach den gezeigten Einstellungen konfiguiert, kann der Tomcat Server gestartet werden. Die Installation auf KS14 ist direkt unter der URL http://ks14.htwdd im Browser erreichbar. Auf der Startseite sind die Service Metadaten der Schnittstellen WMTS, WMS-C und TMS verlinkt.

Eine Übersicht der durch den Dienst angebotenen Layer wird unter der URL http://ks14.htwdd/demo angeboten. Auf dieser Seite wird auch zu dem integrierten Openlayers Client verlinkt, in dem der jeweilige Layer getestet werden kann. Weiterhin erfolgt die Verlinkung zu den KML-Dateien für Layer in der EPSG:4326 Projektion.

Das Erstellen der Kacheln kann über die REST-Schnittstelle im Browser erfolgen. Die URL für die Seed-Oberfläche wird entweder über die Demo-Seite (http://ks14.htwdd/demo) aufgerufen oder direkt nach folgenden Schema im Browser eingebeben:

```
http://{gwc-server}/rest/seed/{layername}
```

In der HTML-Oberfläche können für den Kartenlayer sogennante Seed / Reseed / Truncate - Tasks erstellt werden. Jeder Task bearbeitet nur ein bestimmtes Format in einer bestimmten Projektion. Um den Prozess zu beschleunigen, können mehrere Tasks parallel ausgeführt werden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <seedRequest>
3    <name>adv_dop</name>
4    <gridSetId>EPSG:31468</gridSetId>
5    <zoomStart>0</zoomStart>
6    <zoomStop>10</zoomStop>
7    <format>image/jpeg</format>
8    <!-- type can be
9      * seed (add tiles)
10     * reseed (replace tiles)
11     * truncate (remove tiles)
12   -->
13   <type>SEED</type>
14   <!-- Number of seeding threads to run in parallel -->
15   <threadCount>1</threadCount>
16 </seedRequest>
```

Listing 3.9: XML zur Steuerung des Seedingprozesses über die REST-Schnittstelle

Listing 3.9 zeigt exemplarisch eine XML, mit der ein Seedingprozess über die REST-Schnittstelle gesteuert werden kann. Ein beliebiger HTTP Client kann so den Cache des Dienstes verwalten. Für das Beispiel in Listing 3.10 wurde der einfache HTTP Konsolen-Client cURL[32] verwendet. Über einen POST-Request an die entsprechende URL wird die in Listing 3.9 gezeigte XML an den Server gesendet und ein Seedingprozess für den Layer `adv_dop` ausgelöst.

```
1  D:\curl-7.21.1\curl.exe -T D:\seed-task\adv_dop.xml -X POST
2  http://user:password@localhost/rest/seed/adv_dop.xml
```

Listing 3.10: Seed Task gesteuert über die REST Schnittstelle durch cURL

Eine erweiterte Beschreibung zur REST-Schnittstelle ist in der Geowebcache Dokumentation vorhanden[33].

3.5 Erkenntnisse und Probleme

In diesem Unterkapitel werden Probleme und entsprechende Lösungen sowie erlangte Erkenntnisse diskutiert, die im Zusammenhang mit der Implementierung des Tiled Map Services im Labor Geoinformatik auftraten.

3.5.1 Fehldarstellungen

Neben den Kartenlayern, die Rasterdaten als Datengrundlage verwenden, wurden in Geowebcache auch WMS eingebunden, die Vektordaten rendern (Tabelle 3.1, TOP.Sachsen, Meilen-

32 http://curl.haxx.se
33 http://geowebcache.org/trac/wiki/GWC_seeder

blätter Blattschnitt). Bei dem Überführen von Vektordaten in Rasterdaten können durch das Rendering Effekte entstehen, die sich negativ auf die Darstellung von Karten über kachelnde Kartendienste auswirken.

3.5.1.1 Metatiling

Geowebcache und auch alle anderen untersuchten Tiled Map Services unterstützen Metatiling. Das in der Clientschnittstelle des Dienstes implementierte Verfahren soll zum einen die Zugriffsrate auf die Datenquelle reduzieren, andererseits aber auch Darstellungsprobleme verhindern. In Abbildung 3.6 wird das Verfahren schematisch kurz vorgestellt.

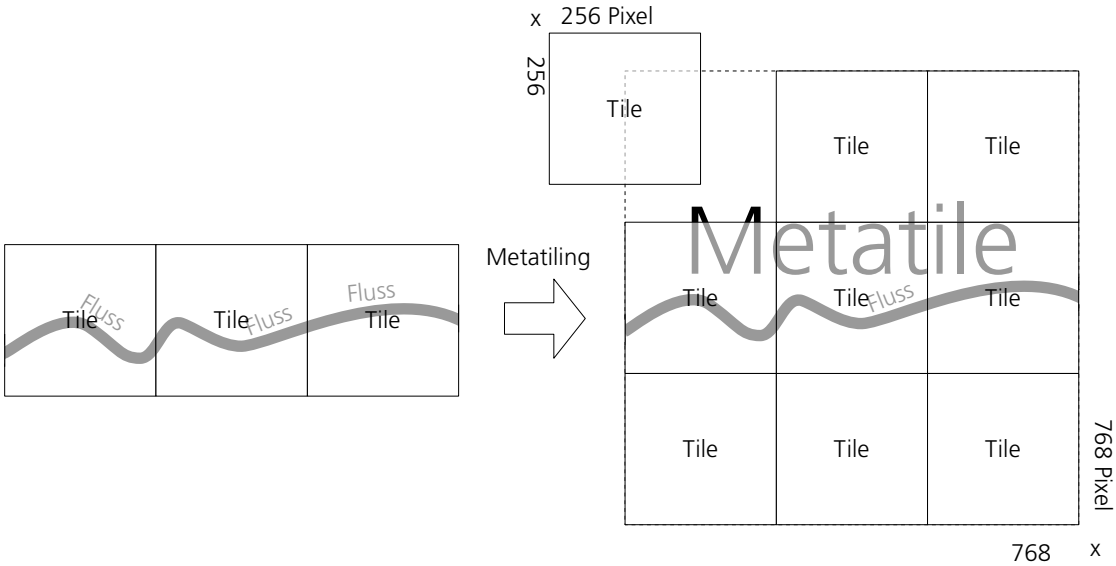

Abbildung 3.6: Das Zusammenfassen mehrerer Kacheln soll Darstellungsprobleme verhindern.

Klassische Kartendienste passen in der Regel die Labelpositionierung für Geometrien auf den Darstellungsausschnitt dynamisch an. Dieses Vorgehen führt beim Cachen kleiner Kartenausschnitte zu dem gezeigten Effekt (Abbildung 3.6). Metatiling kann die Auswirkungen der dynamischen Positionierung nur reduzieren. An den Rändern der zusammengefassten Kacheln treten weiterhin Darstellungsprobleme auf. So können, wie in Abbildung 3.7 gezeigt, Label abgeschnitten werden.

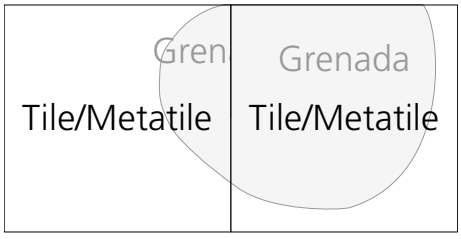

Abbildung 3.7: Abgeschnittene Label bei Verwendung von Tiles/Metatiles

Im Rahmen dieser Diplomarbeit zeigten erste Versuche, dass diese Probleme auch beim Einbinden der Layer Meilenblätter Blattschnitt und TOP.Sachsen auftreten. Besonders auffällig war die Fehldarstellung in den kleineren Maßstäben (Abbildung 3.8).

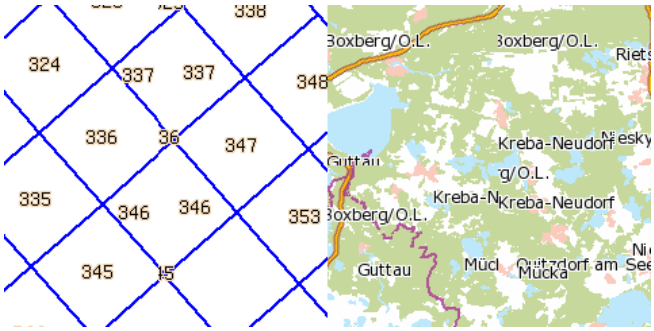

Abbildung 3.8: Abgeschnittene Label im Layer Meilenblätter Blattschnitt und TOP.Sachsen

Dieser Effekt kann nur verhindert werden, wenn direkt auf die Labelpositionierung Einfluss genommen werden kann. Versuche, die Positionierung mit SLD zu ändern, scheiterten an der fehlenden Unterstützung passender Regeln durch die SLD Spezifikation. Nur direkte Änderungen in der MapServer-Konfiguration des Meilenblätter WMS konnten die Probleme beheben. Die Rendering-Regel `PROCESSING "LABEL_NO_CLIP=OFF"` innerhalb der Layerdefinition bewirkt eine feste Positionierung der Label und verhindert das Anpassen auf den angeforderten Kartenausschnitt. Die im Layer TOP.Sachsen auftretenden Fehldarstellungen konnten nicht korrigiert werden. Ein direkter Einfluss auf das Rendering-Verhalten für diesen durch den Staatsbetrieb Geobasisinformation und Vermessung Sachsen angebotenen externen Dienst ist ausgeschlossen.

3.5.1.2 Randeffekte

Ein wieder zusammengesetztes Kartenbild von einem kachelnden Kartendienst sollte so wie in der Datenquelle aussehen. Allerdings kommt es vor, dass durch das Abrunden von Linienenden beim Rendering von Vektordaten Randeffekte entstehen (Abbildung 3.9). Durch den sogenannten Gutter-Parameter ist es mit Geowebcache möglich, diese Darstellungsfehler zu verhindern.

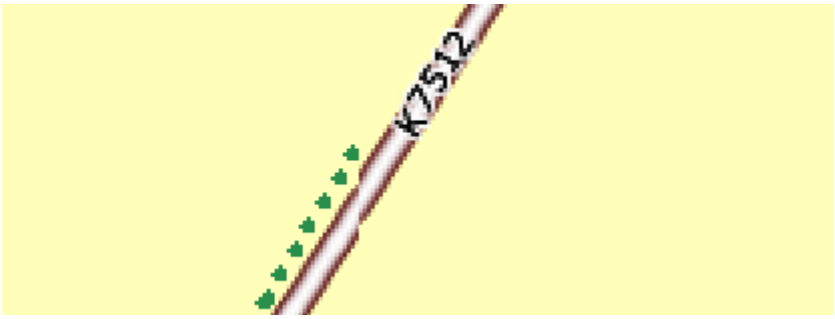

Abbildung 3.9: Beipsiel für einen Randeffekt:Unterbrochene Straße im Layer TOP.Sachsen (vergrößert)

Der Gutter-Parameter bestimmt die Anzahl der Pixel, die vom Rand einer Kachel entfernt werden sollen, bevor sie im Cache abgelegt wird. Das Einsetzen dieser Funktion hat zur Folge, dass die Clientschnittstelle des kachelnden Kartendienstes die Rastergrafiken in größeren Abmessungen von der Datenquelle anfordert, als eigentlich für die Darstellung benötigt werden (Abbildung 3.10).

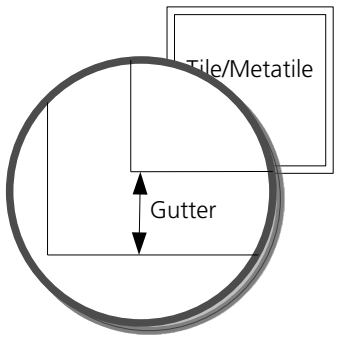

Abbildung 3.10: Gutter-Parameter

Für den durch Geowebcache gekachelten Layer TOP.Sachsen wurde ein Wert von zwei Pixel eingestellt, um den erläuterten Effekt zu reduzieren.

3.5.2 Grafikformate

Klassische Kartendienste unterstützen in der Regel eine große Anzahl an Rastergrafikformaten. Das dynamische Rendern ermöglicht den Kartenclients die Wahl eines optimalen Bildformats. So können verlustfrei komprimierende Formate eine hohe Bildqualität garantieren, verlustbehaftete Formate eine schnelle Anzeige bei geringen Netzwerkkapazitäten. Jedes zusätzliche Grafikformat für einen Layer in kachelnden Kartendiensten hat die Speicherung einer weiteren redundanten Bildpyramide zur Folge[34]. Das Erstellen von Layern in einem Tiled Map Service erfordert also die Vorbetrachtung, welches das optimale Format für die zu kachelnden Daten ist.

Das Format für die im Rahmen dieser Diplomarbeit gekachelten Kartenlayer wurde durch die Unterscheidung zwischen kontinuierlichen und thematischen Daten bestimmt. Die verlustbehaftete Komprimierung durch das JPEG-Format erzeugt für kontinuierliche Daten (Meilenblätter, ATKIS DOP-RGB) die kleinsten Dateien bei guter Qualität. Liegen Quelldaten mit vielen einfarbigen Flächen vor, ist der Einsatz des verlustfrei komprimierenden PNG-Formats vorteilhafter.

34 MapProxy kann von dieser Ausssage ausgenommen werden (vgl. Kapitel 3.3.4).

Um eine mehrfach verlustbehaftete Bildkomprimierung zu verhindern, erlaubt Geowebcache die Definition sogenannter Formatfilter. Diese Regeln bestimmten das zu verwendende Grafikformat an der Clientschnittstelle des Dienstes. Tabelle 3.2 zeigt die für die Konfiguration von Geowebcache verwendeten Formate. Es konnte überwiegend das verlustfrei komprimierende Grafikformat an der Clientschnittstelle eingesetzt werden.

Layer	Format Serverschnittstelle	Format Clientschnittstelle
Meilenblätter	JPEG	PNG
Meilenblätter Blattschnitt	PNG (8Bit)	PNG
ATKIS DOP-RGB	JPEG	JPEG[1]
TOP.Sachsen	PNG (24 Bit)	PNG
[1]Die Kachelung mit PNG Daten als Quelle führte bei diesem Dienst zu Fehlern in der Darstellung.		

Tabelle 3.2: Konfigurierte Formate an Client- und Serverschnittstelle

3.5.3 Speichernutzung

In Kapitel 3.2.3 wurde als Hardwareanforderung ein Speicherbedraf von 110 GB angegeben. Durch die Kachelung der 11. Zoomstufe für den Layer ATKIS DOP-RGB erhöht sich dieser theoretische Wert auf 140 GB. Nach ersten Untersuchungen wird für alle Layer eine Gesamtspeicherbelegung von nur 62 GB durch das Betriebssystem angegeben[35].

Layer	theoretische[1] Größe in GB	Größe auf dem Datenträger in GB	Größe in GB
Meilenblätter	39	22,5	17,5
Meilenblätter Blattschnitt	39	3,25	0,11
ATKIS DOP-RGB	53	19 (32)[2]	15 (26)[2]
TOP.Sachsen	13	4,16	2,25
[1]Berechnet unter der Annahme von einer durchschnittlichen Kachelgröße von 20 KB. [2]Diese Werte sind eine Prognose auf Grundlage des zu 60% gekachelten Layers.			

Tabelle 3.3: Theoretische und reale Größe der gekachelten Layer

Bei genauer Betrachtung der Speichernutzung einzelner gekachelter Kartenlayer in Tabelle 3.3 wird deutlich, dass die errechneten zum Teil stark von den realen Werten abweichen. Hauptgrund ist vorallem die durchschnittliche Dateigröße. Diese ist zum einen spezifisch für das verwendete Dateiformat, überwiegend aber abhängig vom Anteil der Kacheln ohne Inhalt. Die

35 Die 11. Zoomstufe des Layers ATKIS DOP-RGB war zum Zeitpunkt der Untersuchungen noch nicht vollständig gekachelt.

Diskrepanz zwischen den Angaben der Größe auf dem Datenträger und der realen Größe wird durch das Dateisystem des Betriebssystems verursacht. Jede Datei wird in 4 KB große Sektoren abgespeichert. Dadurch belegen Dateien mit 1 KB 4 KB und Dateien mit 21 KB 24 KB auf dem Datenträger. Besonders bei einer großen Menge kleiner Dateien, wie sie von kachelnden Kartendiensten verwaltet werden, wirkt sich dieser Effekt stark aus. Neben dem Mehrverbrauch an Speicherplatz hat das Speichern vieler kleiner Dateien auch Einfluss auf die Zugriffsgeschwindigkeit auf das Dateisystem durch den Dienst [ZHLIZU, 2008]. Der Layer Meilenblätter Blattschnitt enthält nur ein grobes geografisches Netz und wenige Label (Abbildung 3.8). Da Geowebcache leere Kacheln nicht erkennen kann, wurden bei der Kachelung dieses Layers viele leere Kacheln mit einer Dateigröße von unter einem Kilobyte abgelegt. Das Erkennen leerer Kacheln während des Kachelprozesses ist allerdings für spätere Versionen von Geowebcache geplant [GWC-ROADMAP, 2010].

3.5.4 Einbindung der GeoSN Geodatendienste

Der Staatsbetrieb Geobasisinformation und Vermessung Sachsen bietet eine umfangreiche Auswahl an frei zugängigen Geodatendiensten. Eine komplette Übersicht der verfügbaren Dienste ist unter www.landesvermessung.sachsen.de[36] einzusehen. Die Nutzungsbeschränkungen[37] erlauben eine Vervielfältigung der Daten für den nicht gewerblichen Zweck. Allerdings ist das Einbinden der angebotenen Dienste in externe Anwendungen nur durch eine gesonderte Erlaubnis gestattet.

Im Rahmen der Implementation wurde die Kachelung der Layer TOP.Sachsen und ATKIS-DOP-RGB (Tabelle 3.1) über die angebotene OGC WMS Schnittstelle durch Geowebcache geplant. Allerdings ergaben erste Versuche, dass der Seedingprozess nach einer bestimmten Zeitspanne durch den eingebundenen WMS unterbrochen wurde. Nachfolgende Anfragen wurden mit einer HTTP-Response 502 (Bad Gateway) beantwortet. Die Annahme, dass der Cachingprozess aufgrund der stetigen und monotonen WMS-Anfragen eine Art Denial of Service[38] (DoS) am externen Dienst auslöst, konnte nicht bestätigt werden. Allerdings könnte der Fakt, dass der WMS nach einer bestimmten Zeit wieder erreichbar wird, für das Vorhandensein temporärer Sperrlisten sprechen. Diese werden oft eingesetzt, um DoS-Attacken zu verhindern. Eine mögliche Lösung für das Problem wäre das Reduzieren der Anfragefrequenz durch Geowebcache. Eine solche Einstellung wird allerdings nicht angeboten. Aus diesem Grund wurde ein Betriebssystem-Task auf KS14 eingerichtet, der alle 4 Stunden einen Seedingprozess für diese Layer

36 http://www.landesvermessung.sachsen.de/inhalt/geo/basis/basis_dienste.html
37 http://www.landesvermessung.sachsen.de/inhalt/geo/basis/basis_nutz.html
38 Dienstverweigerung als Folge einer Überlastung von Infrastruktursystemen

startet. Dabei wird die REST-Schnittstelle von Geowebcache, wie in Listing 3.10 gezeigt, über einen Konsolenbefehl angesprochen. Das Erstellen der 10-stufigen Bildpyramide für den Layer TOP.Sachsen dauerte auf diese Weise 14 Tage. Die Kachelung des Layers ADV-DOP-RGB war zum Zeitpunkt der Untersuchungen mit 60% nach 29 Tagen noch nicht abgeschlossen.

3.6 Last- und Performancetests

Der Vorteil kachelnder Kartendienste liegt vorallem in der Steigerung der Benutzerfreundlichkeit, die in Zusammenarbeit mit interaktiven Kartenclients erreicht wird. In diesem Unterkapitel soll die Leistungsfähigkeit des implementierten Tiled Map Services mit einem klassischen Kartendienst verglichen werden. Die Untersuchungen sollen weiterhin Schlussfolgerungen über die maximal mögliche Auslastung des Dienstes zulassen.

Nach MELZER, 2007, S. 156 ist Performance

… eine erbrachte Leistung, die anhand von bestimmten Kriterien gemessen und im Vergleich zu Sollwerten oder Referenzsystemen bewertet wird. Performance setzt sich aus dem Erfüllungsgrad quantitativer und qualitativer Kriterien zusammen.

Die Performance des Tiled Map Services soll anhand eines Vergleichs quantitativer Kriterien zu einem Refenzsystem bestimmt werden. In diesem Fall dient ein klassischer Kartendienst als Referenzsystem. Der Durchsatz, definiert als bearbeitete Aufträge pro Zeiteinheit, sowie die Auslastung von Speicher und CPU dienen als quantitative Kriterien.

Um aussagekräftige Vergleiche der beiden Systeme zu ermöglichen, ist es notwendig, einheitliche Bedingungen für die Ermittlung der Referenz- und Testwerte zu schaffen.

3.6.1 Definition der Testbedingungen

Für den Test werden folgende Bedingungen definiert:

- Kachelnder und klassischer Kartendienst werden auf dem gleichen Computersystem getestet.
- Die Tests werden auf den gleichen Inhalten (Layer) in der jeweiligen Technologie durchgeführt.
- Ein Auftrag wird als ein komplettes Kartenbild in einem Clientfenster mit einer Auflösung von 720 mal 500 Pixel definiert.

- Referenz- und Testwerte werden nacheinander bestimmt, um gegenseitigen Einfluss auszuschliessen.

3.6.2 Durchführung

3.6.2.1 Referenz- und Testsystem

Die Tests wurden auf der zur Verfügung gestellten virtuellen Maschine (KS14) ausgeführt, deren genaue Hardwareeigenschaften bereits in Kapitel 3.2.3 näher vorgestellt wurden.

Das Referenzsystem ist eine MapServer-Installation (Version 5.2.1), welche über die CGI-Schnittstelle mit einem Apache HTTP Server (Version 2.2.10) kommuniziert. Da als Testdaten die Meilenblätter verwendet wurden, war es notwendig, den kompletten Grunddatenbestand auf KS14 abzulegen. Die Konfiguration von MapServer erfolgte nach den Vorgaben des originalen Meilenblätter WMS, welcher unter http://geoinformatik.htw-dresden.de/cgi-bin/mblsn erreichbar ist. Dazu zählen auch die vorgenommenen Optimierungen für die Rasterdaten[39], welche eine schnellere Verarbeitung durch MapServer ermöglichen. Das fertig konfigurierte Referenzsystem ist in der digitalen Anlange (Anhang D, Verwendete Software) zu finden.

Als Testsystem dient die in Kapitel 3.4 beschriebene Geowebcache-Installation mit einer vollständig gekachelten Bildpyramide der Meilenblätter in der Gauss-Krüger Projektion, 4. Streifen und Kacheln im JPEG-Format mit 256 mal 256 Pixel.

3.6.2.2 Testclient

Der Testclient hat bei der Bestimmung der Performance die Aufgabe, definierte Requests an den Dienst abzuschicken und die Antwortzeiten statistisch auszuwerten. Für diesen Zweck wurde das Lasttest-Programm JMeter[40] gewählt. Das freie (Apache-Lizenz 2.0) in Java geschriebene Werkzeug ermöglicht über ein einfaches Userinterface das Zusammenstellen von Testabläufen. Neben HTTP(S) können mit JMeter auch Schnittstellen wie SOAP, JDBC, LDAP, POP3(S) und IMAP(S) angesprochen werden.

Um die Systemressourcen während des Testdurchlaufs nicht zu beeinflussen, wurde der Testclient auf einem separaten System ausgeführt. Die Kommunikation zwischen den Systemen

39 http://mapserver.org/input/raster.html#rasters-and-tile-indexing
40 http://jakarta.apache.org/jmeter

erfolgte über eine 1 Gbit/s Netzwerkverbindung. Für diesen Zweck wurde der Rechner mit dem Testclient an den zentralen Netzwerkswitch im Serverraum des Labors Geoinformatik angeschlossen.

In den Testbedingungen wurde ein Auftrag als ein Kartenausschnitt im Clientfenster mit einer Auflösung von 720 mal 500 Pixel definiert. Aufgrund dieser Vorgaben war es vor den eigentlichen Tests notwendig, die Anzahl der Kachel-Requests pro Kartenausschnitt zu bestimmen. Für diese Aufgabe wurde ein Kartenclient auf Basis von Openlayers implementiert, der für jeden gezeigten Kartenausschnitt die entspechenden Requests aufzeichnet. Weiterhin ermöglicht der Client das automatisierte Abspielen einer aufgezeichneten Navigationsfolge in der Karte, um einheitliche Bedingungen für die verwendeten Schnittstellen zu schaffen. Über diese Funktion konnte ein für diesen Client spezifisches Verhältniss von WMS-C zu WMS Requests pro Kartenausschnitt bestimmt werden. In drei Durchgängen wurde die Requestanzahl ermittelt (Tabelle 3.4). Die Abfolgen mit 100 unterschiedlichen Kartenausschnitten wurden unabhängig von einander mit leerem Browsercache durchgeführt.

Durchgang	WMS-C Requests	WMS Requests
1	1505	100
2	1506	100
3	1468	100

Tabelle 3.4: Request-Anzahl für 100 Kartenansichten

Das Request-Verhältnis wird für die weitere Verwendung gerundet mit **15 : 1** (WMS-C : WMS) angegeben. Es muss darauf hingewiesen werden, dass dieser Wert nicht allgemein verwendbar ist, da er spezifisch für den eingesetzen Kartenclient, Browser und Datenstruktur des Tiled Map Service ist.

Die durch den Kartenclient aufgezeichneten Requests wurden für die Erstellung der Testabläufe in JMeter verwendet. JMeter ermöglicht neben dem einfachen Auswerten von Requests auch das Simulieren von mehreren Nutzern. So konnte zusätzlich das Erreichen von Sollwerten getestet werden. Das Userverhalten wurde aus Erfahrungswerten mit 12 Kartenansichten pro Sekunde festgelegt.

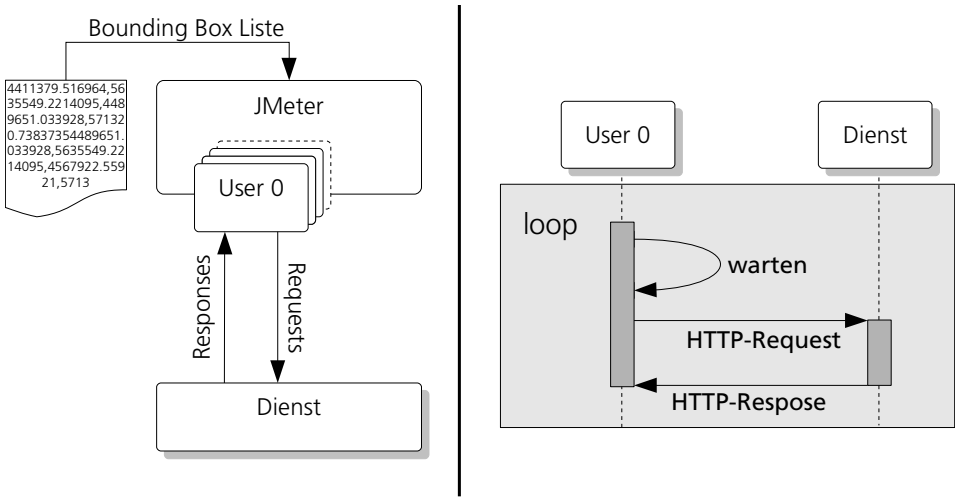

Abbildung 3.11: Schematische Darstellung und Sequenzdiagramm des Testablaufs für die Usersimulation

Abbildung 3.11 zeigt schematisch und in einem Sequenzdiagramm, wie der Testablauf in JMeter definiert wurde. Jeder simulierte User sendet unabhängig von anderen Usern in der festgelegten Frequenz von 12 Aufträgen pro Mintute über einen Testzeitraum von 5 Minuten Anfragen an den Dienst. Für den Tiled Map Service wurde die HTTP-Request Frequenz entsprechend des festgelegten Verhältnisses von 15 : 1 um den Faktor 15 erhöht. Mit diesen Vorgaben konnte ein JMeter Testablauf mit 5, 10, 20, 40, 80, 160, 320, 640 Usern und entstprechende Erwartungswerte definiert werden.

Die Ermittlung des Maximal-Durchsatzes erfolgte ohne eine Pause zwischen den Requests mit 5 (WMS) bzw. 30 (WMS-C) simulierten Usern. Die Nutzeranzahl wurde auf Grundlage der theoretisch möglichen Bandbreite von 1 Gbit/s und einer maximalen Requestfrequenz von 200 HTTP Anfragen pro Sekunde für einen durch JMeter simulierten User bestimmt.

Während einer Testreihe wurden die Werte für Arbeitsspeicherbelegung und CPU-Auslastung auf KS14 notiert und den einzelnen Tests zugeorndet.

Die JMeter Testabläufe, die Ausgangsdaten sowie die Testergebnisse sind auf der digitalen Anlage dieser Diplomarbeit abgelegt.

3.6.3 Auswertung

Die Auswertung des Maximal-Durchsatzes von Referenz- und Testsystem ergaben einen 17 mal höheren Durchsatz für den Tiled Map Service (Abbildung 3.12). In der vorgestellten Konfiguration verarbeitet Geowebcache 130 Kartenansichten (1940 Kachel-Requests) pro Sekunde. Die durch diese Requests verursachte Brutto-Datenmenge beträgt fast 300 Mbits/s. MapServer erreicht maximal 7,6 Requests pro Sekunde und verarbeitet damit nur knapp 7 Mbit/s.

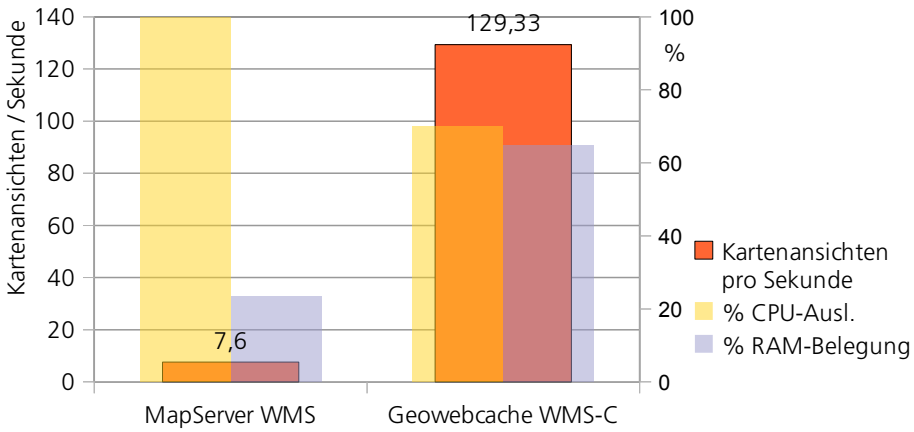

Abbildung 3.12: Durchsatz von MapServer und Geowebcache im Vergleich

Während der Ermittlung des Maximal-Durchsatzes wurde die CPU voll durch den klassischen Kartendienst ausgelastet. Der kachelnde Kartendienst belastete den Prozessor durchschnittlich nur zu 70 %. Im Vergleich zu Geowebcache belegte MapServer mit 20 % überraschend wenig des zur Verfügung stehenden physischen Arbeitsspeichers. Neben den verarbeiteten Anfragen pro Sekunde protokolliert JMeter statistische Angaben über die am Dienst entstehenden Wartezeiten einzelner Anfragen. Mit einer maximalen Wartezeit von 1,3 Sekunden (MapServer) bzw. 0,2 Sekunden (Geowebcache) konnten erstaunlich geringe Werte für beide Dienste ermittelt werden. Die erweiterten Tests zeigen allerdings ein anderes Bild.

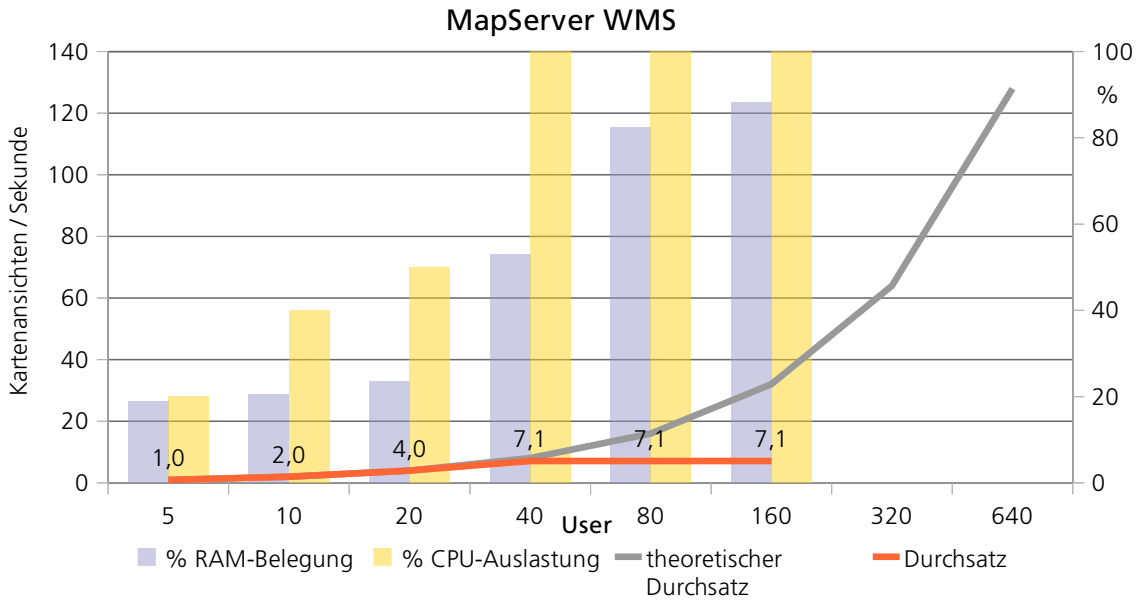

Abbildung 3.13: Durchsatz und Ressourcenverbrauch in Abhängigkeit von der Nutzerzahl (WMS)

Der Durchsatz für MapServer stagniert ab einen Wert von 40 Nutzern. Grund für diese vom theoretischen Durchsatz abweichenden Werte ist die volle CPU Auslastung. Die Verarbeitungskapazität des Systems ist durch die für jeden User individuell zu verarbeitende Kartenerzeugung ausgeschöpft. Die Testreihe konnte für 320 und 640 Nutzer nicht fortgesetzt werden, da der Dienst für diese Werte nur noch eine HTTP-Error-Response lieferte. Allerdings ist eine eindeutige Tendenz abzusehen. Unter der Last von 160 Nutzern betrug die durchschnittliche Wartezeit auf einzelne Anfragen über 20 Sekunden. Einige Clientanfragen wurden sogar erst nach 40 Sekunden durch den Dienst verarbeitet.

Bei 640 Nutzern beträgt die maximale Wartezeit für eine Kachel-Response von Geowebcache 0,2 Sekunden. In Abbildung 3.14 wird dargestellt, dass selbst bei dieser Nutzeranzahl noch Reserven in den Hardwareressourcen des Systems bestehen. Auch der gemessene Durchsatz weicht nicht signifikant von den theoretischen Werten ab.

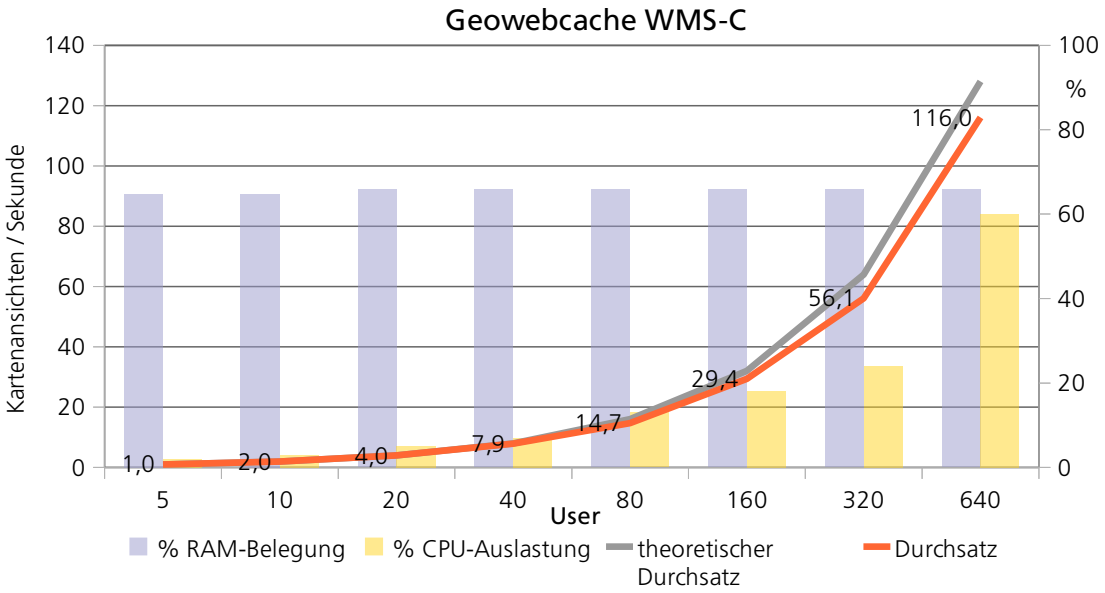

Abbildung 3.14: Durchsatz und Ressourcenverbrauch in Abhängigkeit von der Nutzerzahl (WMS-C)

Die ermittelten Werte zeigen das Resultat für eine ideelle Testumgebung, die in eine abschliessende Betrachtung mit einbezogen werden muss. So kann aus diesen Ergebnissen nicht direkt auf die Anzeigegeschwindigkeit von Kartenansichten in einem Webclient geschlossenen werden. WebClients für kachelnde Kartendienste werden überwiegend mit rechen- und speicherintensiven Rich Internet Applications (RIA) umgesetzt. Die Erzeugung einer Kartenansicht aus einzelnen Kacheln beansprucht Systemressourcen und verlangsamt so die Darstellung. Findet eine clientseitige Kombination unterschiedlicher Layer statt, vergrößert sich dieser Einfluss. Untersuchungen mit dem implementierten Kartenclient bestätigen diese Aussagen. So ergaben die Zeitmessungen für die automatische Navigationsabfolge im Vergleich zur WMS-Schnittstelle eine 10 Prozent längere Durchlaufzeit. Neben den Systemressourcen beeinflussen auch andere Faktoren die Darstellungsgeschwindigkeit. Tile Caching fähige Kartenclients sind auf die parallele Verarbeitung vieler HTTP-Requests für die schnelle Kartendarstellung angewiesen. Doch alle gängigen Browser beschränken die maximale Anzahl gleichzeitig offener Netzwerkverbindungen [AJAXPERF, 2006]. So entstehen auf der Clientseite Wartezeiten, die nicht direkt mit höherer Dienstperformace verhindert werden können.

4 Integration in die Basiskomponente Geodaten Sachsen

In Kapitel 3 wurde die Integration eines kachelnden Kartendienstes in die einfache System- und Dienstestruktur des Labors für Geoinformatik der HTW-Dresden untersucht. Der durchlaufene Prozess erlaubt erste Schlussfolgerungen für die Anforderungen an Tiled Map Services in Geodateninfrastrukturen. Diese Anforderungen werden nachfolgend an der konkreten Infrastrukur Basiskomponente Geodaten des Freistaats Sachsen (GeoBAK)[1] theoretisch betrachtet.

4.1 Basiskomponente Geodaten Sachsen

Als Teil der eGovernment Initiative Sachsen soll GeoBAK eine standardisierte Integration von Geoinformationen aus unterschiedlichen Quellen sicherstellen. Für die Nutzer werden Geoinformationen in Form von Standarddiensten sowie über ein zentrales Geoportal bereitgestellt. Nutzer dieser Infrastruktur sind Bürger, Wirtschaft und Verwaltung. [GEOBAK, 2005, S. 8F]

Das Grobkonzept GEOBAK, 2005 beschreibt die unterschiedlichen Sichten auf dieses System nach dem Reference Model of Open Distributed Processing (RM-ODP)[2]. Die für Integration von Tiled Map Services relevanten Aspekte werden nachfolgend in diese Sichten eingeordnet und diskutiert.

1 http://www.egovernment.sachsen.de/58.htm

2 ISO/IEC 10746 Informationstechnik - Verteilte Verarbeitung in Offenen Systemen - Referenzmodell (http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html)

4.1.1 Fachliche Sicht

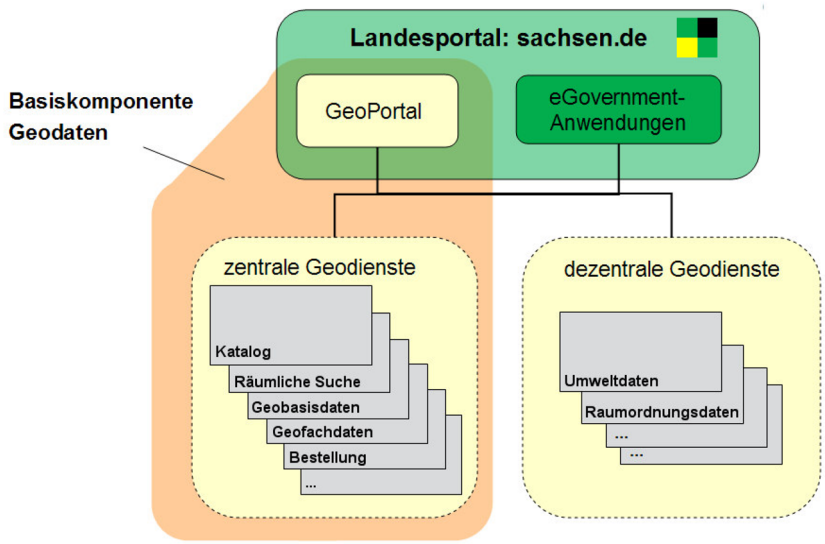

Abbildung 4.1: GeoBAK - Einordnung und Abgrenzung [GEOBAK, 2005, ABB. 4]

Abbildung 4.1 ordnet GeoBAK in die eGovernment-Struktur des Landes Sachsen ein. Durch GEOBAK, 2005 wird als ein fachliches Ziel das Erschliessen und Bereitstellen von Geoinformationen für einen *breiten Nutzerkreis* definiert. Weiterhin soll der Einsatz *webbasierter Geodienste* verstärkt werden. Diese gestellten Ziele können auch allgemein auf Geodateninfrastrukturen bezogen werden. Kachelnde Kartendienste erfüllen diese fachlichen Anforderungen aufgrund ihrer zugrundeliegenden technologischen Konzepte.

Der Einsatz von kachelnden Kartendiensten in Geodateninfrastrukturen ist zum einen abhängig von der Anzahl der zu erwartenden Nutzer, von den genutzen Dateninhalten sowie von deren Nutzungswegen. Kachelnde Kartendienste sind vor dem Hintergrund entstanden möglichst vielen Nutzern performant den Zugang zu Karten im Internet zu ermöglichen. Um dieses Ziel zu erreichen, wurde die individuelle Kartendarstellung für den einzelnen Nutzer eingeschränkt sowie eine zusätzliche redundante Datenhaltung eingeführt. Die Integration dieser Technologie ist nur gerechtfertigt, wenn sie auch entspechend genutzt wird. So eignen sich vorallem Dateninhalte, die eine geringe Änderungsdynamik aufweisen und ohne Zugangsbeschränkungen für viele Nutzer zur Verfügung gestellt werden sollen.

GeoBAK wurde mit dem Ziel konzeptioniert, Geodaten standardisiert zu integrieren und zugänglich zu machen. Die einzelnen Teilkomponenten der Infrastruktur werden in interne zentrale und externe dezentrale Komponenten unterschieden. Die Visualisierung von Geobasisdaten und Geofachdaten ist durch das Konzept als ein zentraler Dienst definiert. Dieser in dem Konzept

auch „Portrayal-Service“ genannte Dienst erstellt das entgültige Kartenbild. Kachelnde Kartendienste besitzen ähnliche Eigenschaften. Sie aggregieren Kartendienste aus unterschiedlichen Quellen. Aufgrund dieser Eigenschaften kann ein kachelnder Kartendienst als zentrale Komponente mit ähnlichen Abhängigkeiten wie der Visualisierungs-Dienst eingeordnet werden.

Das Grobkonzept beschreibt verschiedene Nutzungwege für die angebotenen Dienste. Der Zugang auf die Infrastruktur durch einen Geoportal-Client ermöglicht die Erstellung von Karten in einer Webanwendung. Mit beliebigen externen Anwendungen soll über den Nutzungsweg „OGC-Service“ die Kombination lokaler Daten mit denen aus GeoBAK möglich sein. Die Unterstützung der OGC Schnittstelle WMTS durch den kachelnden Kartendienst kann beide Nutzungswege ermöglichen. Voraussetzung dafür ist die Integration der standardisierten Schnittstelle in den Geoportal-Client.

4.1.2 Funktionale Sicht

Ziel der funktionalen Sicht ist die Identifizierung und Beschreibung der Funktion einzelner Komponenten. Weiterhin sollen die Interaktionen zwischen diesen Komponenten dargestellt werden.

4.1.2.1 Strukturelle Sicht

Ein kachelnder Kartendienst kann als eine Komponente innerhalb einer Geodateninfrastruktur identifiziert werden. Die Funktionen, die ein solcher Dienst implementiert, werden über seine Schnittstellen definiert. Im Rahmen der Anforderungsanalyse in Kapitel 3.2 wurden mögliche Server- und Clientschnittstellen vorgestellt. Vereinfacht kann festgelegt werden, dass ein kachelnder Kartendienst neben einer Schnittstelle, die den Zugriff auf die bereitgestellten Daten ermöglicht, auch über Schnittstellen für die Erstellung sowie Aktualisierung der Daten verfügen muss.

Durch das Grobkonzept werden Dienste bestimmten Servicetypen zugeorndet. Basis-Services realisieren die grundlegenden Funktionalitäten wie den Zugriff auf Geodaten, Metadaten und Benutzerdaten. Mehrwert-/Integrationsservices führen die Basis-Services zusammen und erfüllen fachliche Anforderungen bzw. komplette Anwendungsfälle.

In der Abbildung 4.2 wird eine Variante für die Eingliederung von Tiled Map Services in das GeoBAK Service-Modell dargestellt. Dieser steht gleichwertig neben dem Portrayal-Service und unterstützt die gleichen Clientschnittstellen, greift also auf die Mapping-Services über eine WMS-Schnittstelle zu. Der Service Dispatcher kommuniziert mit dem Tiled Map Service und den OGC-Clients über die WMTS Schnittstelle.

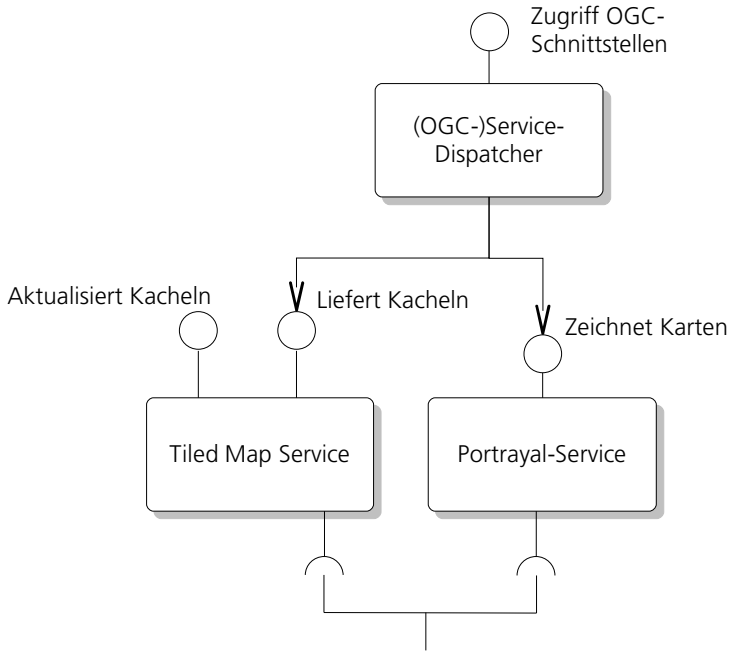

Abbildung 4.2: Möglicher Integrationsweg in des GeoBAK Service-Modell

Die Aktualisierung von Kacheln erfolgt in diesem Vorschlag aktiv durch Zugriff auf eine „Aktualisierungsschnittstelle“. Die Einordnung von „Aktualisierungsclients“ ist für diese Infrastruktur nicht direkt möglich, da die erzeugten Daten der Mapping-Services aus unterschiedlichen Quellen stammen können und es keinen standardisierten Weg der Aktualisierung gibt. Eine manuelle Aktualisierung könnte über den im Grobkonzept beschriebenen Administrationsclient erfolgen.

4.1.2.2 Dynamische Sicht

Die Interaktionen zwischen den Komponenten werden nachfolgend für den Anwendungsfall „Zugriff auf OGC-Services“ in einem UML-Sequenzdiagramm dargestellt.

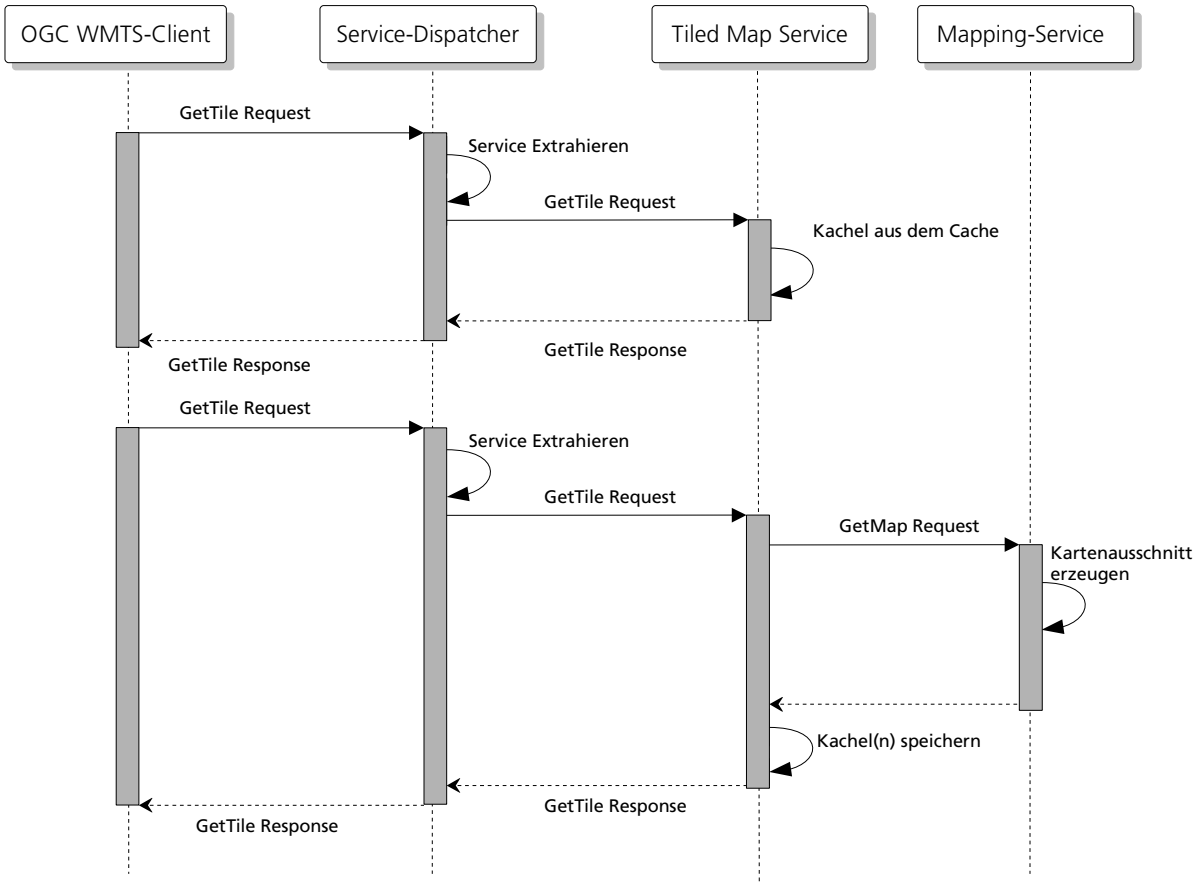

Abbildung 4.3: Interaktionen zwischen den Komponenten bei einem GetTile Request

Der kachelnde Dienst greift bei nicht vorhandenen Kacheln auf die Datenquelle zurück und speichert diese zwischen.

5 Zusammenfassung und Ausblick

5.1 Darstellung der Ergebnisse

Ziel dieser Diplomarbeit war es den Stand der Technik im Bereich Tiled Map Services darzustellen. Durch die Implementation eines solchen Dienstes im Labor Geoinformatik konnten die relevanten Aspekte dieser Technologie näher betrachtet werden. So wurden auf der Grundlage einer Anforderungsanalyse unterschiedliche Lösungen für Tiled Map Services analysiert. Mit Hilfe dieser Bestandsanalyse konnte ein passendes Open Source Produkt bestimmt und in die Infrastruktur des Labors Geoinformatik eingebunden werden. Es wurden unterschiedliche Kartenlayer durch den Dienst gekachelt und für den durch Herrn Naumann implemtentierten Tile Cache fähigen Kartenclient bereit gestellt. Das Einbinden unterschiedlicher Datenquellen ermöglichte Schlussfolgerungen auf generelle Probleme, die durch den Einsatz von kachelnden Kartendiensten entstehen können. So existiert keine funktionierende Server – Client – Lösung für die Aktualisierung des redundanten Datenbestandes. Weiterhin wurden durch den Implementierten Dienst eine große Anzahl leerer Kacheln gespeichert, welche ohne Nutzen viel Festplattenspeicher einnehmen. Die standardisierte WMS-Clientschnittstelle sollte die Unabhängigkeit von vorhandenen und zukünfigen Implementationen sicherstellen. Allerdings konnte dieser Vorteil nicht ausgenutzt werden. Die Konfiguration der eingebundenen Kartendienste war zum Teil nicht für die Kachelung geeignet und musste, wenn möglich, extra für den Dienst angepasst werden.

Die im weiteren Verlauf durchgeführten Last- und Performancetests sollten das Anwendungsgebiet kachelnder Kartendienste nochmals unterstreichen. Mit den Ergebnissen konnte nicht nur die maximale Performance ermittelt werden, sondern auch die Verwendung von Hardwareressourcen bei bestimmten Lasten.

Im letzten Kapitel wurde die Integration eines kachelnden Kartendienstes in die vorhandene Infrastruktur Basiskomponente Geodaten des Freistaats Sachsen untersucht. Die Betrachtungen der fachlichen sowie funktionalen Sicht zeigten, dass dies mit den selben Einschränkungen, wie sie im Labor Geoinformatik der HTW vorliegen, möglich ist.

5.2 Ausblick

Mit Geowebcache 1.2.3 sollen Neuerungen wie das Erkennen von leeren Kacheln sowie in späteren Versionen die Unterstützung von WMTS über SOAP und REST eingeführt werden.

Die populäre Webmapping-API Openlayers kann seit der Version 2.10 auch Karten über WMTS einbinden. Ob sich der Standard behaupten kann steht noch in Frage. Momentan gibt es viele Alternativen, die sich Server und Clientseitig leicht implementieren lassen.

Wie lange kachelnde Kartendienste die Welt der interaktiven Webanwendungen bestimmen kann schlecht vorausgesagt werden. Da mit ihnen immer ein Kompromiss aus schneller Kartendarstellung und Individualität bzw. Flexibilität eingegangen werden muss ist es durchaus möglich, dass sie mit der nächsten Browsertechnologie durch andere Konzepte ersetzt werden.

Listingverzeichnis

Listing 2.1: Formel zur Bestimmung der Kachelanzahl nach ZhLiZu, 2008.....17

Listing 2.2: Root Ressource - Zentrales XML Dokument mit Verlinkung auf den TMS 24

Listing 2.3: TileMapService Ressource - XML Dokument für die eigentliche Dienstbeschreibung 25

Listing 2.4: TileMap Ressource - XML Dokument für die Beschreibung einer Bildpyramide....25

Listing 2.5: Request und Response auf eine TMS Kachel Ressource.....25

Listing 2.6: SOAP WMTS GetTile Request.....35

Listing 2.7: SOAP WMTS GetTile Response.....35

Listing 3.1: Beispiel einer GeoRSS mit einer einfachen Geometrie.....46

Listing 3.2: Integration von MapServer in die Google Maps API v3 53

Listing 3.3: Konsolenbefehle für die Apache Tomcat Installation.....55

Listing 3.4: Änderungen in der server.xml der Apache Tomcat Installation.....55

Listing 3.5: Context Konfiguration der Apache Tomcat servlet.xml.....56

Listing 3.6: Konfiguration des Geowebcache Cacheordners 57

Listing 3.7: Beispiel für Integration des Meilenblätter WMS in GWC.....58

Listing 3.8: GWC GridSet-Definition für EPSG:31458.....59

Listing 3.9: XML zur Steuerung des Seedingprozesses über die REST-Schnittstelle.....61

Listing 3.10: Seed Task gesteuert über die REST Schnittstelle durch cURL.....61

Tabellenverzeichnis

Tabelle 2.1: Raumbezogene OGC Web Services.....7

Tabelle 2.2: Beispiel für eine Bildergallerie nach REST-Architektur.....8

Tabelle 2.3: Speicher- und Zeitbedarf für das Erstellen einer Bildpyramide.....18

Tabelle 2.4: WMS-C GetMap Parameter entspechend der Abbildung 2.13.....23

Tabelle 2.5: GetTile Request entsprechend der Abbildung 2.13.....34

Tabelle 3.1: Konfigurierte Kartenlayer auf KS14.....60

Tabelle 3.2: Konfigurierte Formate an Client- und Serverschnittstelle.....65

Tabelle 3.3: Theoretische und reale Größe der gekachelten Layer.....65

Tabelle 3.4: Request-Anzahl für 100 Kartenansichten.....69

Abbildungsverzeichnis

Abbildung 2.1: Die Service-orientierte Architektur....6

Abbildung 2.2: Portrayal-Modell nach Adrian Cuthbert [DOCUTH, 1998]....11

Abbildung 2.3: Möglicher Aufbau eines gekachelten Datenbestandes....12

Abbildung 2.4: Quadtree als gerichteter Baum....13

Abbildung 2.5: Einfaches HTTP-Request-Response....14

Abbildung 2.6: Daten liegen im Client Cache....15

Abbildung 2.7: Validierung der Daten im Client Cache....15

Abbildung 2.8: Web Proxy antwortet stellvertretend für den Server....16

Abbildung 2.9: Einfache Lastverteilung durch einen Gateway....19

Abbildung 2.10: Content Distribution Network....20

Abbildung 2.11: Adressierung der Kacheln....21

Abbildung 2.12: Schematische Darstellung der Capabilities Erweiterung durch WMS-C....22

Abbildung 2.13: Bounding Box und Kacheln in der geographischen Projektion....23

Abbildung 2.14: Funktionen des WMTS Interface als UML Diagramm (vereinfacht) [OGC-WMTS, 2010, Figure 1]....27

Abbildung 2.15: Service Metadaten UML Diagramm (vereinfacht) [OGC-WMTS, 2010, Figure 4]....28

Abbildung 2.16: Service Metadaten für Contents [OGC-WMTS, 2010, Figure 4]....28

Abbildung 2.17: WMTS Tile Matrix Sets mit Attributen [OGC-WMTS, 2010, Figure 8]....29

Abbildung 2.18: WMTS Layer mit Attributen [OGC-WMTS, 2010, Figure 6]....31

Abbildung 2.19: TileMatrix und TileMatrixLimits Eigenschaften nach OGC-WMTS, 2010, Figure 7....32

Abbildung 2.20: GetTile Request Parameter [OGC-WMTS, 2010, Figure 10].....33

Abbildung 2.21: Dokumentenstruktur eines KML-Superoverlays.....37

Abbildung 3.1: Schnittstellenanforderung an den Tiled Map Service.....41

Abbildung 3.2: Konfigurationsdokumente einer Geowebcache-Installation.....44

Abbildung 3.3: MapProxy als "man in the middle" zwischen Map Clients und Map Server.....49

Abbildung 3.4: Auszug aus der OSM Infrastruktur nach [OSM Components, 2010].....51

Abbildung 3.5: Struktur der geowebcache.xml.....56

Abbildung 3.6: Das Zusammenfassen mehrerer Kacheln soll Darstellungsprobleme verhindern. 61

Abbildung 3.7: Abgeschnittene Label bei Verwendung von Tiles/Metatiles.....61

Abbildung 3.8: Abgeschnittene Label im Layer Meilenblätter Blattschnitt und TOP.Sachsen….62

Abbildung 3.9: Beipsiel für einen Randeffekt:Unterbrochene Straße im Layer TOP.Sachsen (vergrößert).....62

Abbildung 3.10: Gutter-Parameter.....63

Abbildung 3.11: Schematische Darstellung und Sequenzdiagramm des Testablaufs für die Usersimulation.....69

Abbildung 3.12: Durchsatz von MapServer und Geowebcache im Vergleich.....70

Abbildung 3.13: Durchsatz und Ressourcenverbrauch in Abhängigkeit von der Nutzerzahl (WMS).....70

Abbildung 3.14: Durchsatz und Ressourcenverbrauch in Abhängigkeit von der Nutzerzahl (WMS-C).....71

Abbildung 4.1: GeoBAK - Einordnung und Abgrenzung [GeoBAK, 2005, Abb. 4].....74

Abbildung 4.2: Möglicher Integrationsweg in des GeoBAK Service-Modell.....76

Abbildung 4.3: Interaktionen zwischen den Komponenten bei einem GetTile Request.....77

Literaturverzeichnis

ADOBE-TIFF, 1992 — Adobe Developers Association: *TIFF*. Version: 6.0, http://partners.adobe.com/public/developer/tiff/index.html

AJAXPERF, 2006 — Breen, Ryan: *Circumventing browser connection limits for fun and profit*. http://www.ajaxperformance.com/2006/12/18/circumventing-browser-connection-limits-for-fun-and-profit/, Abruf: August 2006

ARCGIS-HELP, 2010 — ESRI: *Creating a cached map service*. http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html#/Tutorial_Creating_a_cached_map_service/009300001575000000/, Abruf: August 2010

ARCGIS, 2010 — ESRI: *ArcGIS Server | ArcGIS Resource Center*. http://resources.arcgis.com/ja/content/arcgisserver/10.0/about, Abruf: August 2010

BAUMEISTER, 2005 — Baumeister, Johann: *Unter dem Joch, Load Balancing: Add-ons, Appliances, Software*. In: iX - Magazin für professionelle Informationstechnik, Ausgabe: August 2005, S. 114ff

BLEICH, 2008 — Bleich, Holger: *Verteile und herrsche*. In: ct - Magazin für Computer und Technik, Ausgabe: August 2008, S. 160-163

BONCHI, 11/2001 — Bonchi F. et al.: *Web log data warehousing and mining for intelligent web caching*. In: Data & Knowledge Engineering, Ausgabe: November 2001, S. 165-189

DAKALB, 2009 — Kallbach, Maria: *Orchestrierung von Geo Web Services*. Diplomarbeit, Hochschule für Technik und Wirtschaft, 2009

DANAUM, 2010 — Naumann, Fabian: *Implementierung Tile Caching fähiger Map Clients*. Diplomarbeit, Hochschule für Technik und Wirtschaft, 2010

DOCUTH, 1998 — A. Doyle, A. Cuthbert: *Essential Model of Interactive Portrayal*. Version: -, OpenGIS Project Document 98-061 http://member.opengis.org/tc/archive/arch98/98-061.pdf

EPSG-REG, 2010 — OGP: *EPSG Geodetic Parameter Registry*. http://www.epsg-registry.org/, Abruf: September 2010

FIELDING, 2000 — Fielding R.: *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation, 2000

GDAL-FORMATS, 2010 — OSGeo: *GDAL Raster Formats*. `http://www.gdal.org/formats_list.html`, Abruf: August 2010

GDAL-GEOTIFF, 2010 — OSGeo: *GeoTIFF File Format*. `http://www.gdal.org/frmt_gtiff.html`, Abruf: August 2010

GDI-DE, 2007 — Arbeitskreis Architektur (GDI-DE): *Architektur der Geodateninfrastruktur Deutschland*. Version: 1.0, 17.08.2007 `http://www.gdi-de.org/de_neu/arbeitskreise/navl_architektur.html`

GEOBAK, 2005 — Auräth T. et. al., Projektgruppe GeoBAK: . Version: 1.0, 25.02.2005

GEOTIFF, 2000 — Ritter, Nils et al.: *GeoTIFF Format Specification*. Version: 1.8.2,

GWC-ROADMAP, 2010 — OSGeo: *Geowebcache Roadmap*. `http://geowebcache.org/trac/wiki/roadmap`, Abruf: September 2010

HALÖ, 2004 — Hauser, Tobias ; Löwer, Ulrich M.: *Web Services - Die Standards*. Gallieo Computing, 2004

KIEHLE, 2006 — Kiehle, Christian: *Geodaten Standardisiert integrieren - Lokales abstrakt*. In: iX - Magazin für professionelle Informationstechnik, Ausgabe: Dezember 2006, S. 55ff

MAPSRV-MAPSCRIPT, 2010 — University of Minnesota: *MapScript Introduction - MapServer 5.6.5 documentation*. `http://mapserver.org/mapscript/introduction.html`, Abruf: August 2010

MAPSRV-RASTER, 2010 — University of Minnesota: *Preprocessing Rasters - MapServer 5.6.5 documentation*. `http://mapserver.org/input/raster.html#preprocessing-rasters`, Abruf: August 2010

MAPSRV-TILE MODE, 2010 — University of Minnesota: *Tile Mode - MapServer 5.6.5 documentation*. `http://mapserver.org/output/tile_mode.html`, Abruf: August 2010

MELZER, 2007 — Melzer, Ingo: *Service-orientierte Architekturen mit Web Services*. Spektrum Verlag, 2007

MOD_TILE, 2010 — OpenStreetMap: *Quellcode und Dokumentation*. http://trac.openstreetmap.org/browser/applications/utils/mod_tile, Abruf: August 2010

OGC-KML, 2008 — Open Geospatial Consortium: *OGC KML*. Version: 2.2.0, 14.04.2008 http://www.opengeospatial.org/standards/kml

OGC-ORM, 2008 — Open Geospatial Consortium: *OGC Reference Model*. Version: 2.0, 11.11.2008 http://www.opengeospatial.org/standards/orm

OGC-OWS, 2007 — Open Geospatial Consortium: *OGC Web Service Common Implementation Specification*. Version: 1.1.0, 02.09.2007 http://www.opengeospatial.org/standards/common

OGC-WMS, 2004 — Open Geospatial Consortium: *OGC Web Map Service Implementation Standard*. Version: 1.3.0, http://www.opengeospatial.org/standards/wms

OGC-WMTS, 2010 — Open Geospatial Consortium: . Version: 1.0.0, 06.04.2010 http://www.opengeospatial.org/standards/wmts

OPENLAYERS, 2010 — Openlayers: *Entwicklungsziele für die Version 2.10*. http://trac.openlayers.org/query?status=closed&milestone=2.10+Release, Abruf: August 2010

OSGEO-TILING, 2006 — OSGeo: *TilingStandard*. http://wiki.osgeo.org/wiki/TilingStandard, Abruf: August 2010

OSGEO-TMS, 2006 — OSGeo: *Tile Map Service Specification*. Version: 1.0, http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification

OSGEO-WMSC, 2006 — OSGeo: *WMS Tiling Client Recommendation*. Version: 1.1.1, http://wiki.osgeo.org/wiki/WMS_Tiling_Client_Recommendation

OSM COMPONENTS, 2010 — OpenStreetMap: *Components of OSM* . http://wiki.openstreetmap.org/wiki/File:OSM_Components.png, Abruf: August 2010

RFC2616, 1999 — Fielding R. et al: *Hypertext Transfer Protocol -- HTTP/1.1* . Version: 1.1, http://www.ietf.org/rfc/rfc2616.txt

RFC3568, 2003 — Barbir, A. et al.: *Known Content Network (CN) Request-Routing Mechanisms*. Version: -, http://tools.ietf.org/rfc/rfc3568.txt

SCHMIDT, 2006 — Christopher Schmidt: *TileCache: Map Tile Caching*. http://crschmidt.net/blog/archives/181/tilecache-map-tile-caching/, Abruf: August 2010

SEIDEL, 2009 — Ole Seidel: *Optimale WebMapping-Anwendungen mit Flex/Flash*. In: Landmarcs - Geo Business News, Ausgabe: März 2009, S. 8

SOUDERS, 2008 — Steve Souders: *High Performance Web Sites*. O'Reilly Verlag, 2005

TILKOV, 2009 — Tilkov, Stefan: *REST - Der bessere Web Service?*. http://it-republik.de/jaxenter/artikel/REST---Der-bessere-Web-Service-2158.html, Abruf: August 2009

TOMCAT-CONF, 2010 — apache.org: *Apache Tomcat Configuration Reference - The HTTP Connector*. http://tomcat.apache.org/tomcat-6.0-doc/config/http.html, Abruf: August 2010

ZHLIZU, 2008 — Zhang, Yansheng; Li, Dancheng; Zhu, Zhiliang: *A Server Side Caching System for Efficient Web Map Services*. In: The 2008 International Conference on Embedded Software and Systems Symposia (ICESS2008), Ausgabe: Juli 2008, S. 32-37

Anlagen

A Legenden zur UML-Notation

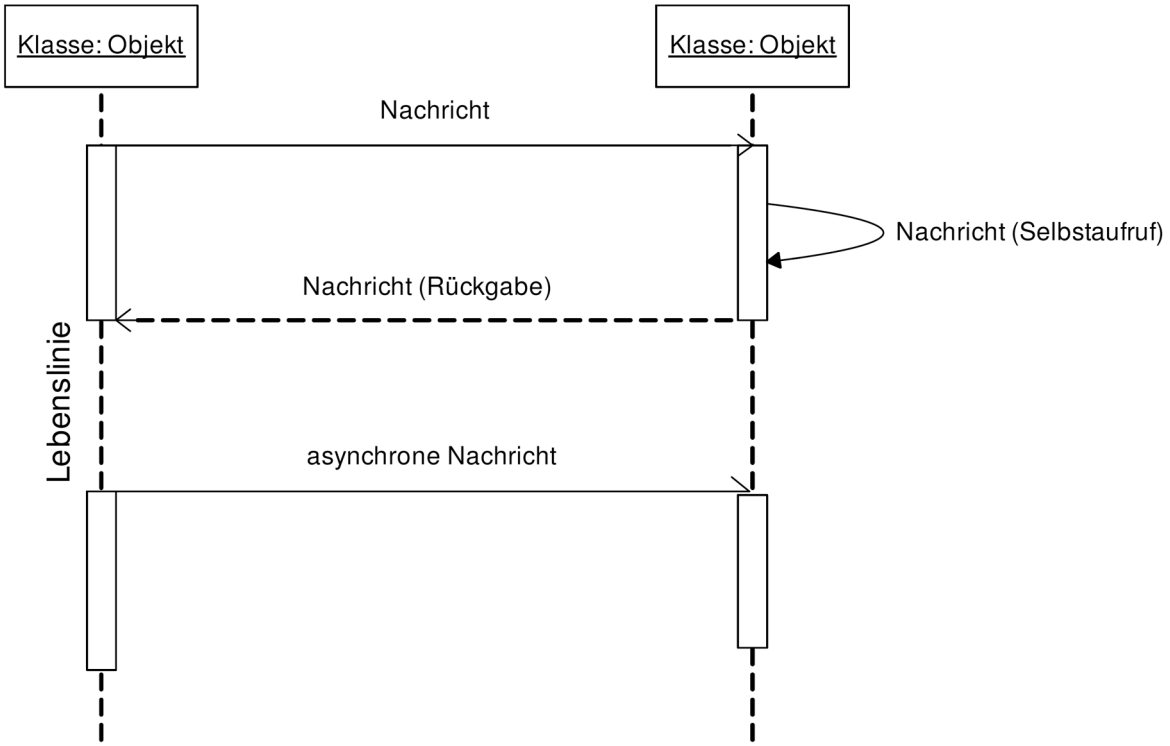

Abbildung A.1: Legende UML Sequenzdiagramm [GEOBAK, 2005, S.15]

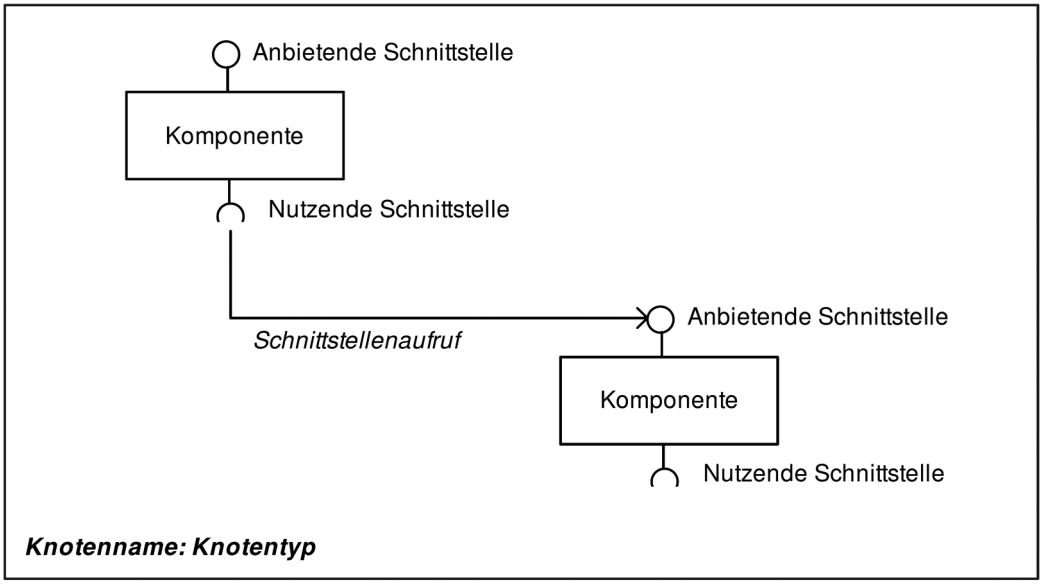

Abbildung A.3: Legende UML-Komponenten Diagramm [GEOBAK, 2005 S. 15]

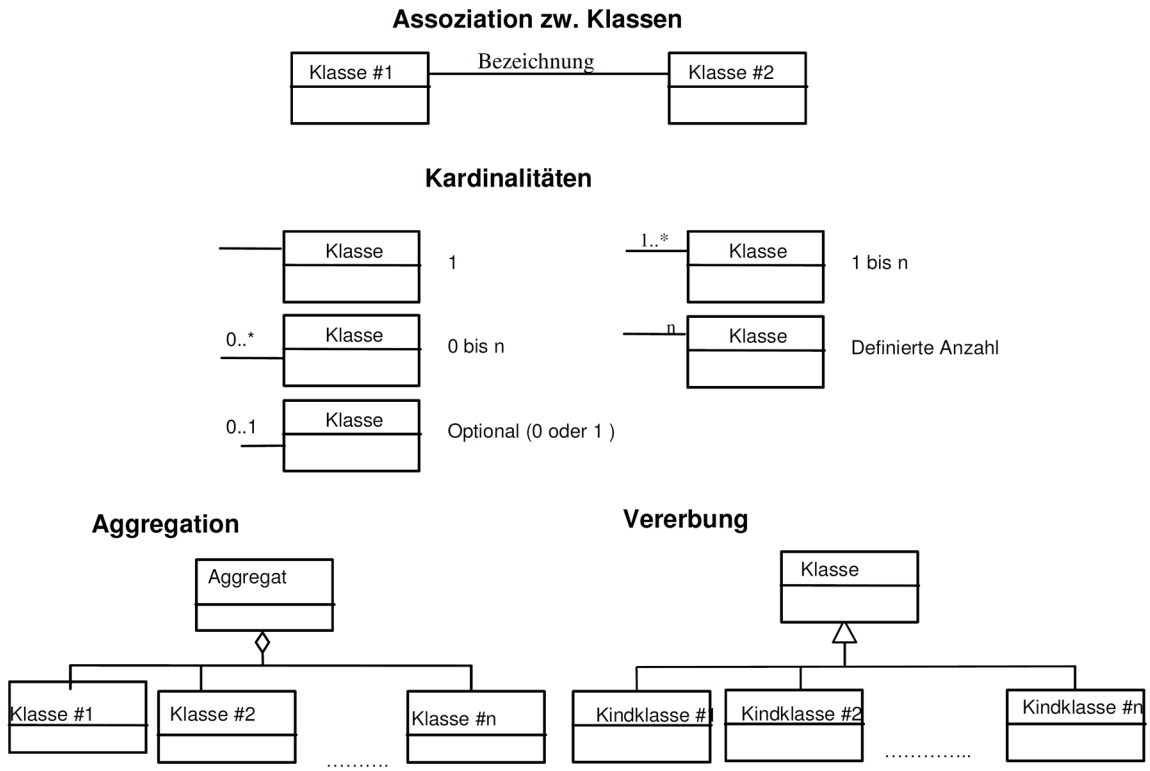

Abbildung A.2: Legende UML-Klassendiagramm [GEOBAK, 2005 S. 14]

B Geowebcache Konfiguration

```
<?xml version="1.0" encoding="utf-8"?>
<gwcConfiguration
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://geowebcache.org/schema/1.2.2/geowebcache.xsd"
xmlns="http://geowebcache.org/schema/1.2.2">

<!--
  http://geowebcache.org/docs/current/configuration/xml/exhaustive.html
-->

  <version>1.2.2</version>

  <backendTimeout>30</backendTimeout>
  <cacheBypassAllowed>false</cacheBypassAllowed>
  <proxyUrl>http://www-cache.htw-dresden.de:3128</proxyUrl>

  <gridSets>

    <gridSet>
      <name>EPSG:31468</name>
      <srs><number>31468</number></srs>
      <extent>
<!--
BBOX GK4 + 5
10.5, 0, 16.5, 84
-->
        <coords>
          <double>4333108</double>
          <double>0</double>
          <double>5001550</double>
          <double>9333388</double>
        </coords>
      </extent>
<!--
OPTIONAL The size of a single pixel in meters. OGC standards such as WMS
1.3.0 and
WMTS assume 0.28mm per pixel, which corresponds to 90.71428571428572 DPI
-->
      <pixelSize>0.00028</pixelSize>
      <metersPerUnit>1</metersPerUnit>
<!--
OGC 07-057r7
urn:ogc:def:wkss:OGC:1.0:GoogleCRS84Quad
Table E.3 — Definition of Well-known scale set GoogleCRS84Quad
-->
      <scaleDenominators>
        <double>2183915.093862179</double>
        <double>1091957.546931089</double>
        <double>545978.7734655447</double>
        <double>272989.3867327723</double>
        <double>136494.6933663862</double>
        <double>68247.34668319309</double>
        <double>34123.67334159654</double>
        <double>17061.83667079827</double>
        <double>8530.918335399136</double>
        <double>4265.459167699568</double>
<!--

Meilenblätter Auflösung: 1 Pixel = 1 Meter
```

```
(96 DPI / 25,4 mm) * 1000 = 3780 Pixel / m -> 1:3780 (größter Maßstab ohne
sichtbare Pixel)

oder :

Pixelgröße MB: 1 Pixel = 1m

Bildschirm (OGC): 1 Pixel = 0.00028 m

Maßstab 0.00028:1

^-1

~ 1:3571
-->
        <double>2132.729583849784</double>
        <double>1066.364791924892</double>

      </scaleDenominators>

      <tileHeight>256</tileHeight>
      <tileWidth>256</tileWidth>
    </gridSet>
  </gridSets>
  <layers>
    <wmsLayer>
      <name>meilenblaetter</name>
      <mimeFormats><string>image/jpeg</string></mimeFormats>
      <wmsUrl>
        <string>http://geoinformatik.htw-dresden.de/cgi-bin/mblsn</string>
      </wmsUrl>
      <wmsVersion>1.1.1</wmsVersion>
      <wmsLayers>Bmbl_vollton</wmsLayers>

      <formatModifiers>

        <formatModifier>
          <responseFormat>image/jpeg</responseFormat>
<!-- images are requested from the backend to avoid double compression -->
          <requestFormat>image/png</requestFormat>
          <transparent>false</transparent>
          <bgColor>0xFFFFFF</bgColor>
<!-- default quality from mapserver width GD/JPEG response: 75 (0-100) -->
          <compressionQuality>0.8</compressionQuality>
        </formatModifier>

      </formatModifiers>

      <gridSubsets>
        <gridSubset>
          <gridSetName>EPSG:3857</gridSetName>
          <extent>
            <coords>
              <double>1113194.907932736</double>
              <double>6446275.841017158</double>
              <double>1781111.852692377</double>
              <double>6800125.454397307</double>
            </coords>
          </extent>
```

```
          <zoomStop>16</zoomStop>
        </gridSubset>
        <gridSubset>
          <gridSetName>EPSG:4326</gridSetName>
          <extent>
            <coords>
              <double>11.3782</double>
              <double>50.0754</double>
              <double>15.0987</double>
              <double>51.6923</double>
            </coords>
          </extent>
          <zoomStop>16</zoomStop>
        </gridSubset>
        <gridSubset>
          <gridSetName>EPSG:31468</gridSetName>
          <extent>
            <coords>
              <double>4457010</double>
              <double>5553510</double>
              <double>4712900</double>
              <double>5728370</double>
            </coords>
          </extent>
          <zoomStop>9</zoomStop>
        </gridSubset>
      </gridSubsets>
      <queryable>true</queryable>
      <useETags>true</useETags>
      <!-- ein halbes jahr -->
      <expireClients>15768000</expireClients>
    </wmsLayer>

    <wmsLayer>
      <name>meilenblaetter_blattschnitt</name>
      <mimeFormats><string>image/png</string></mimeFormats>
      <wmsUrl>
        <string>http://geoinformatik.htw-dresden.de/cgi-bin/mblsn</string>
      </wmsUrl>
      <wmsVersion>1.1.1</wmsVersion>
      <wmsLayers>Blattecken2</wmsLayers>
      <transparent>true</transparent>
      <gutter>2</gutter>
      <metaWidthHeight><int>7</int><int>7</int></metaWidthHeight>
      <gridSubsets>
        <gridSubset>
          <gridSetName>EPSG:3857</gridSetName>
          <extent>
            <coords>
              <double>1113194.907932736</double>
              <double>6446275.841017158</double>
              <double>1781111.852692377</double>
              <double>6800125.454397307</double>
            </coords>
          </extent>
          <zoomStop>16</zoomStop>
        </gridSubset>
        <gridSubset>
          <gridSetName>EPSG:4326</gridSetName>
```

```
          <extent>
            <coords>
              <double>11.3782</double>
              <double>50.0754</double>
              <double>15.0987</double>
              <double>51.6923</double>
            </coords>
          </extent>
          <zoomStop>16</zoomStop>
        </gridSubset>
        <gridSubset>
          <gridSetName>EPSG:31468</gridSetName>
          <extent>
            <coords>
              <double>4457010</double>
              <double>5553510</double>
              <double>4712900</double>
              <double>5728370</double>
            </coords>
          </extent>
          <zoomStop>9</zoomStop>
        </gridSubset>
      </gridSubsets>
      <useETags>true</useETags>
      <queryable>true</queryable>
      <!-- ein halbes jahr -->
      <expireClients>15768000</expireClients>
    </wmsLayer>

    <wmsLayer>
      <name>top_sachsen</name>
      <mimeFormats><string>image/png</string></mimeFormats>
      <wmsUrl>
        <string>http://www.landesvermessung.sachsen.de/ias/basiskarte4/serv
ice/SRV4TOPSN/WMSFREE_TK/WMSFREE_TK/wmsservice</string>
      </wmsUrl>
      <wmsVersion>1.1.1</wmsVersion>
      <gutter>2</gutter>
      <metaWidthHeight><int>3</int><int>3</int></metaWidthHeight>
      <wmsLayers>MS</wmsLayers>
      <gridSubsets>
        <gridSubset>
          <gridSetName>EPSG:31468</gridSetName>
          <extent>
            <coords>
              <double>4483820</double>
              <double>5547910</double>
              <double>4716180</double>
              <double>5747090</double>
            </coords>
          </extent>
          <zoomStop>9</zoomStop>
        </gridSubset>
      </gridSubsets>
      <queryable>true</queryable>
      <useETags>true</useETags>
      <expireClients>600</expireClients>
    </wmsLayer>
```

```
    <wmsLayer>
      <name>adv_dop</name>
      <mimeFormats><string>image/jpeg</string></mimeFormats>
      <wmsUrl>
        <string>http://www.landesvermessung.sachsen.de/ias/basiskarte4/serv
ice/SRV4ADV_P_DOPRGB/WMSFREE_TK/wmsservice</string>
      </wmsUrl>
      <wmsVersion>1.1.1</wmsVersion>
      <metaWidthHeight><int>4</int><int>4</int></metaWidthHeight>
      <wmsLayers>adv_dop</wmsLayers>
      <formatModifiers>
        <formatModifier>
          <responseFormat>image/jpeg</responseFormat>
<!-- images are requested from the backend to avoid double compression -->
          <requestFormat>image/jpeg</requestFormat>
<!-- default quality from mapserver width GD/JPEG response: 75 (0-100) -->
          <compressionQuality>0.9</compressionQuality>
        </formatModifier>
      </formatModifiers>
      <gridSubsets>
        <gridSubset>
          <gridSetName>EPSG:31468</gridSetName>
          <extent>
            <coords>
              <double>4483820</double>
              <double>5547910</double>
              <double>4716180</double>
              <double>5747090</double>
            </coords>
          </extent>
          <zoomStop>10</zoomStop>
        </gridSubset>
      </gridSubsets>
      <queryable>true</queryable>
      <useETags>true</useETags>
      <expireClients>600</expireClients>
    </wmsLayer>
  </layers>
</gwcConfiguration>
```

C Tile Service Profil - Sachsen

Hochschule für
Technik und Wirtschaft
Dresden
University of Applied Sciences

Fakultät Geoinformation

Tile Service Profil - Sachsen

Version: 1.0.0

Thomas Jurk

Fabian Naumann

Dresden, am 09. September 2010

1 Vorbemerkung

Das Tile Service Profil - Sachsen dient der Definition einheitlicher Parameter kachelnderGeodatendienste auf Basis des Open-GIS-Standards WMTS (Version 1.0.0) und der WMS Tiling Client Recommendation der Open Source Geospatial Foundation.

Dieses Dokument soll die Möglichkeit bieten, Tile Services verschiedener Anbieter durch Verwendung einheitlicher Resolutions und Projektionen miteinander zu überlagern.

Das Tile Service Profil - Sachsen wurde im Rahmen der Diplomarbeiten “Integration von Tiled Map Services in Geodateninfrastrukturen” von Herrn Thomas Jurk und “Implementierung Tile Caching fähiger Map Clients” von Herrn Fabian Naumann erstellt und ist als erste Empfehlung zu verstehen.

2 Profil

2.1 Resolutions

Ein Tile Service sollte folgende Resolutions besitzen:

Zoomstufe	Maßstabszahl	Pixelgröße (Grad)	Pixelgröße (Meter)
0	2183915.093862179	$5.49316406250000\ 10^{-3}$	611.4962262814100
1	1091957.546931089	$2.74658203125000\ 10^{-3}$	305.7481131407048
3	545978.7734655447	$1.37329101562500\ 10^{-3}$	152.8740565703525
4	272989.3867327723	$6.86645507812500\ 10^{-4}$	76.43702828517624
5	136494.6933663862	$3.43322753906250\ 10^{-4}$	38.21851414258813
6	68247.34668319309	$1.71661376953125\ 10^{-4}$	19.10925707129406
7	34123.67334159654	$8.58306884765625\ 10^{-5}$	9.554628535647032
8	17061.83667079827	$4.29153442382812\ 10^{-5}$	4.777314267823516
9	8530.918335399136	$2.14576721191406\ 10^{-5}$	2.388657133911758
10	4265.459167699568	$1.07288360595703\ 10^{-5}$	1.194328566955879
11	2132.729583849784	$5.36441802978516\ 10^{-6}$	0.5971642834779395

Um eine möglichst weitreichende Interoperabilität zu gewährleisten, werden die Resolutions aus den „Well-known scale sets“ des OpenGIS WMTS Standards (Annex E.3) für dieses Profil verwendet. Die Berechnung beruht auf der Festlegung, dass Zoomstufe 0 die ganze Erde in einer 256 x 256 großen Kachel darstellt. Das Bildpyramidengerüst wird durch Quadtrees aufgebaut, womit sich Maßstab respektive Resolution um den Faktor 2 ändert.

Die für amtliche Kartenwerke üblichen Maßstäbe sind fürWeb Map Applikationen ungeeignet, da der Skalierungsfaktor nicht einheitlich ist und damit die Usability verschlechtert.

2.2 Format

Ein Tile Service sollte folgende Formate besitzen:

- image/jpeg
- image/png

2.3 Projektionen

Ein Tile Service sollte folgende Projektionen besitzen:

2.3.1 EPSG:4326

Code	4326
Name	WGS84
Unit	degree
Scope	Horizontal component of 3D system. Used by the GPS satellite navigation system and for NATO military geodetic surveying.

http://www.epsg-registry.org/export.htm?gml=urn:ogc:def:crs:EPSG::4326

2.3.2 EPSG:3857

Code	3857
Name	WGS 84 / Pseudo-Mercator
Unit	meter
Scope	Certain Web mapping and visualisation applications.

http://www.epsg-registry.org/export.htm?gml=urn:ogc:def:crs:EPSG::3857

2.3.3 EPSG:31468

Code	31468
Name	DHDN / 3-degree Gauss-Kruger zone 4
Unit	meter
Scope	Large and medium scale topographic mapping and engineering survey, cadastral survey.

http://www.epsg-registry.org/export.htm?gml=urn:ogc:def:crs:EPSG::31468

2.3.4 EPSG:31469

Code	31469
Name	DHDN / 3-degree Gauss-Kruger zone 5
Unit	meter
Scope	Large and medium scale topographic mapping and engineering survey, cadastral survey.

http://www.epsg-registry.org/export.htm?gml=urn:ogc:def:crs:EPSG::31469

D Digitale Anlage

\Diplomarbeit_Jurk_2010.pdf

\Tile Service Profil - Sachsen.pdf

\Beispiel Clients\

    google+mapserver.html

    google+tms.html

    google+wmts.html

    mapproxy_test.html

    meilenblaetter.kmz

\Berechnung Kachelung\

    Berechnung Speicher und Zeitbedarf.xls

\Geowebcache Eclipseprojekt\

\Performance Tests\

    Ausgangsdaten\

    Ergebnisse\

    Testabläufe\

\Seed Task\

\Verwendete Software\