Software Tools zum interoperablen Austausch und zur Visualisierung von Geodatensätzen über das Internet

Martin Schultz, Michael Decker, Sebastian Lührs

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

Schriften des Forschungszentrums Jülich
Reihe Energie & Umwelt / Energy & Environment Band / Volume 117

Forschungszentrum Jülich GmbH
Institut für Energie- und Klimaforschung (IEK)
Troposphäre (IEK-8)

Software Tools zum interoperablen Austausch und zur Visualisierung von Geodatensätzen über das Internet

Martin Schultz, Michael Decker, Sebastian Lührs

Schriften des Forschungszentrums Jülich
Reihe Energie & Umwelt / Energy & Environment Band / Volume 117

ISSN 1866-1793 ISBN 978-3-89336-730-6

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über <http://dnb.d-nb.de> abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag D-52425 Jülich Telefon (02461) 61-5368 · Telefax (02461) 61-6103 E-Mail: zb-publikation@fz-juelich.de Internet: http://www.fz-juelich.de/zb
Umschlaggestaltung:	Grafische Medien, Forschungszentrum Jülich GmbH
Quelle Titelbild:	@Victoria – Fotolia.com
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2011

Schriften des Forschungszentrums Jülich
Reihe Energie & Umwelt / Energy & Environment Band / Volume 117

ISSN 1866-1793
ISBN 978-3-89336-730-6

Vollständig frei verfügbar im Internet auf dem Jülicher Open Access Server (JUWEL)
unter http://www.fz-juelich.de/zb/juwel

Introduction

By Martin G. Schultz

Monitoring of the Earth and its atmosphere is essential for assessing the state of the environment and for measuring success of political measures to curb air pollution or mitigate climate change. The modern concept of monitoring the chemical composition of the atmosphere goes beyond the classical set-up of individual observational sites and involves a variety of measurement platforms (ground-based, ships, aircraft, satellites) and principles (chemical analysis, active and passive remote sensing, etc.). Numerical models play an increasingly important role for the interpretation of observations and their integration into a consistent global picture.

The exchange of data, from simple time series at point locations to complex multi-dimensional data sets from remote sensing instruments and numerical models, is a fundamental aspect for building a system of systems for global earth observations (GEOSS). Different data formats, data protocols and access restrictions limit the availability of earth observation data and create great technical and management challenges for building operational services like the European atmosphere service (http://www.gmes-atmosphere.eu) which shall provide daily forecasts and retrospective analyses to decision makers, environmental agencies and the common user. New technologies and the development of international standards for the description of geospatial data (http://www.opengeospatial.org/) open up new possibilities and bring the vision of truly interoperable data exchange closer to reality. The concept of interoperability means that computer systems can interact autonomously and exchange information about the availability of data sets and their details without direct human intervention. This changes the way how data are accessed, processed and visualized (Figure 1) and can lead to much faster turnaround times for the analysis of specific episodes or scientific studies in general.

The global network of air quality data is still in its infancy. A GEOSS community of practice (http://wiki.esipfed.org/index.php/GEO_AQ_CoP) has been established to foster exchange of ideas around this topic and to establish a set of real data nodes which can form the backbone of global interoperable AQ data exchange. Several challenges remain on both the technical side and on the management and governance level. These issues have been discussed at a recent workshop in Croatia (http://wiki.esipfed.org/index.php/Air_Quality_Data_Network_Solta_2011) which was organized under the title “from virtual to real” with participation of the Forschungszentrum Jülich.

This report collects two academic theses which were recently accepted at the Fachhochschule Aachen and which address two of the core aspects of interoperable data exchange: the master thesis by Michael Decker describes the development of a web coverage service (WCS) server

for atmospheric composition data in the netcdf data format. This development was performed jointly with Kari Hoijarvi and Rudolf Husar from Washington University, Louisiana. This server software forms the essential interface between the actual storage location of the data and the web-based access through automated systems or via human-controllable web interfaces. Such a web interface, which allows for flexible selection of individual data sets, their variables or specific geographical regions, download of the selected data or their visualisation, is described in the bachelor thesis by Sebastian Lührs, which forms part 2 of this report. Both pieces of software are freely available to the community and are put to regular use in the MACC project's global boundary condition service at http://macc.icg.kfa-juelich.de:50080.

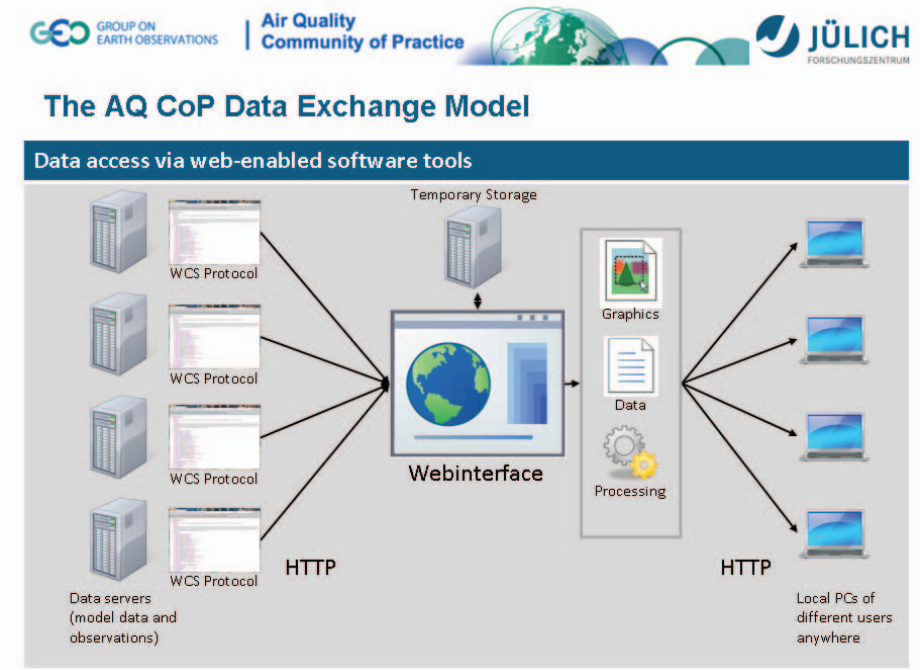

Figure 1: Simplified representation of the flow of earth observation data in a fully interoperable framework. Compared to the current "classical" model, where users must know individual data servers, their access protocols and data formats, the new concept allows for uniform access to all data through community web interfaces or via automated shell scripts. In reality the set-up needs to be more complex because of the need to identify and find data sets and maintain the relations between data servers and web interfaces.

Einleitung

Von Martin G. Schultz

Die kontinuierliche Beobachtung der Erde und ihrer Atmosphäre ist von entscheidender Bedeutung für die Überwachung unserer Umwelt und zur Kontrolle des Erfolges von Maßnahmen zur Luftreinhaltung oder Eindämmung des Klimawandels. Ein modernes Konzept der Überwachung („Monitoring“) der Luftqualität geht weit über das klassische Netzwerk einzelner Messstationen hinaus und beinhaltet eine Reihe von Messplattformen (Bodenstationen, Schiffsbeobachtungen, Flugzeugmessungen und Satelliteninstrumente) und –methoden (chemische Analysen, aktive und passive Fernerkundung, etc.). Numerische Modelle erlangen eine zunehmende Bedeutung für die Interpretation solcher Beobachtungen und ihre Integration in ein zusammenhängendes globales Bild.

Der Austausch von Daten, angefangen von einfachen Zeitserien einzelner Messstationen bis hin zu komplexen, mehrdimensionalen Datensätzen aus der Fernerkundung oder von numerischen Modellen, ist ein fundamentaler Aspekt beim Aufbau eines globalen Systems der Erdbeobachtung (GEOSS). Unterschiedliche Datenformate, Datenprotokolle und Zugangsbeschränkungen limitieren die Verfügbarkeit von Erdbeobachtungsdaten und führen zu großen technischen sowie Managementproblemen bei der Inbetriebnahme von operationellen Datendiensten wie dem europäischen Dienst zur Überwachung der chemischen Zusammensetzung der Atmosphäre (http://www.gmes-atmosphere.eu). Dieser Dienst soll tägliche Vorhersagen und retrospektive Analysen der Luftqualität in Europa und anderswo bereitstellen, damit diese Informationen von Entscheidungsträgern, Umweltbehörden und der Bevölkerung genutzt werden können. Neue Technologien und die Entwicklung internationaler Standards zur Beschreibung georeferenzierter Daten (http://www.opengeospatial.org/) eröffnen neue Möglichkeiten und lassen die Vision eines echt interoperablen Datenaustausches Realität werden. Das Konzept der Interoperabilität bedeutet, dass Computersysteme selbstständig Informationen über das Vorhandensein und den Inhalt von Datensätzen austauschen können, ohne dass der Nutzer über die Details Bescheid wissen muss. Dieses Konzept verändert die Art und Weise, wie Geodaten beschafft, prozessiert und visualisiert werden können (Abb. 1). Interoperabilität hat das Potenzial die Zeit, die für die Analyse eines besonderen Ereignisses oder allgemeine wissenschaftliche Studien benötigt wird, erheblich zu verkürzen.

Das globale Netzwerk von Luftqualitätsdaten befindet sich noch in einem sehr frühen Entwicklungsstadium. Eine GEOSS “community of practice” (http://wiki.esipfed.org/index.php/GEO_AQ_CoP) wurde ins Leben gerufen, um den Austausch von Ideen zu befördern und ein Basisnetzwerk zum interoperablen Austausch echter Daten aufzubauen, welches dann relativ leicht erweitert werden kann. Dabei gibt es jedoch noch einige technische und andere

Hürden zu überwinden. Der Stand und Ausblick dieses Basisnetzwerkes waren Themen eines kürzlich durchgeführten Workshops auf der Insel Solta in Kroatien (http://wiki.esipfed.org/index.php/Air_Quality_Data_Network_Solta_2011), der unter Beteiligung des Forschungszentrums Jülich unter dem Motto „Von virtuell zu real“ organisiert wurde.

Dieser Bericht enthält den Abdruck zweier Abschlussarbeiten, die vor kurzem an der Fachhochschule Aachen angenommen wurden und die sich mit zwei Kernthemen des interoperablen Datennetzwerkes befassen. Die Masterarbeit von Michael Decker beschäftigt sich mit der Entwicklung eines Datenservers, der das „web coverage service“ (WCS) Datenprotokoll und das Datenformat netcdf unterstützt. Diese Softwareentwicklung wurde gemeinsam mit Kari Hoijarvi und Rudolf Husar von der Washington University in Louisiana, USA, durchgeführt. Die Server-Software stellt das entscheidende Interface zwischen dem eigentlichen Datenspeicher und dem Zugriff auf die Daten über automatisierte Systeme oder Web-Schnittstellen dar. Ein solches Web-Interface ist in der Bachelorarbeit von Sebastian Lührs beschrieben, die den zweiten Teil des Berichtes umfasst. Das Interface erlaubt eine flexible Auswahl von Datensätzen inklusive der Auswahl einzelner Variablen oder bestimmter geographischer Regionen, und man kann die ausgewählten Daten herunterladen oder visualisieren. Beide Softwarepakete sind von den Autoren frei erhältlich. Sie werden regelmäßig im Rahmen des Jülicher MACC Datenservers http://macc.icg.kfa-juelich.de:50080 benutzt.

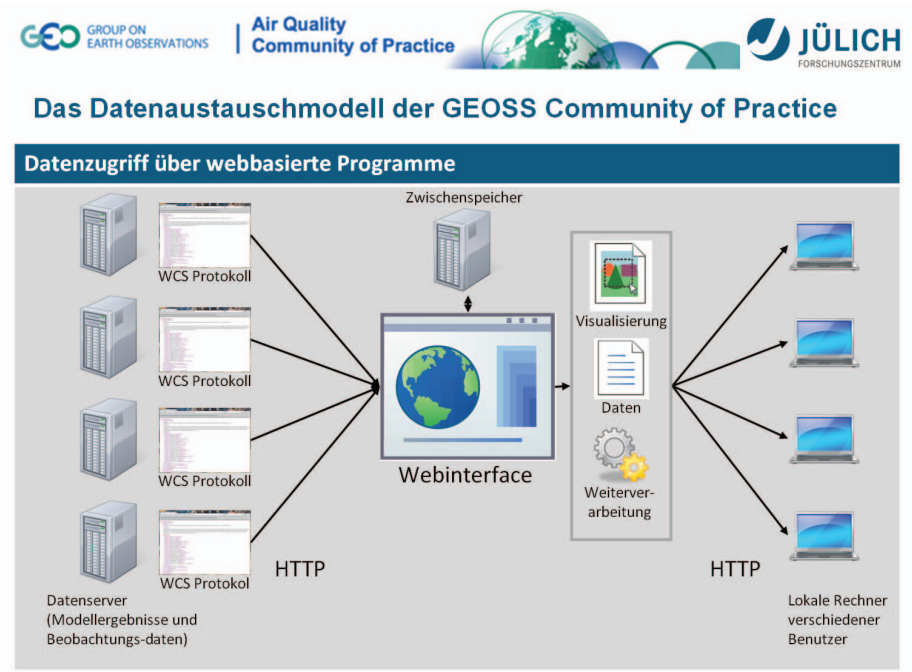

Abbildung 1: Vereinfachte Darstellung des Datenflusses in einem echt interoperablen System. Der Datenzugriff erfolgt über einheitliche Schnittstellen und erfordert nicht mehr das Wissen des einzelnen Nutzers über den Zugriff auf einzelne Datenserver. Zusätzliche Dienste zum Auffinden von Datensätzen machen den Aufbau eines solchen Systems in Wirklichkeit etwas komplizierter.

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I

Web-basierte Bereitstellung, Verarbeitung und Verknüpfung georeferenzierter Atmosphärendaten

Michael Decker

Entstanden im Rahmen einer Masterarbeit
an der Fachhochschule Aachen, Campus Jülich

Jülich, Februar 2011

Zusammenfassung

Im Bereich der Atmosphärenforschung und Erdbeobachtung besteht ein zunehmendes Interesse an der Zusammenführung von Daten aus verschiedenen Quellen. Speziell sollen durch den Vergleich von Modelldaten und Beobachtungsdaten die Computermodelle verbessert werden und neue Erkenntnisse über die in der Atmosphäre ablaufenden Prozesse gewonnen werden.

Die vorliegende Arbeit beschäftigt sich mit der Nutzung von Geodaten in diesem Kontext und speziell mit Datensätzen von numerischen Simulationen der atmosphärischen Zusammensetzung, die bei einem internationalen Modellvergleich gewonnen wurden. Ziel ist die Aufbereitung und Bereitstellung dieser Daten in einer Weise, die eine Verknüpfung und Auswertung durch interessierte Wissenschaftler vereinfacht.

Im Bereich der Aufarbeitung wird zunächst auf das Problem der Standardkonformität von zu verarbeitenden Datensätzen eingegangen. Anschließend wird die Entwicklung einiger häufig benötigter Operatoren beschrieben, die die Vergleichbarkeit von Datensätzen aus verschiedenen Quellen verbessern sollen. Dazu zählen die Interpolation von Gedaten auf einheitliche Gitter sowie weitere (statistische) Operationen zum Vergleich dieser Daten.

Im letzten Teil wird die von der amerikanischen Umweltschutzbehörde EPA teilfinanzierte und in Zusammenarbeit mit der *Washington University in St. Louis* erfolgte Entwicklung eines Datenservers nach dem offenen WCS-Standard zur Bereitstellung von Atmosphärendaten beschrieben. Ziel ist die Vereinfachung des Zugriffs auf Geodaten aus unterschiedlichen Quellen durch Aufbau eines weltweiten Netzwerkes von WCS-kompatiblen Servern im Bereich der Erdbeobachtung.

Glossar

API	Application Programming Interface
CDL	network Common Data form Language
CRS	Coordinate Reference System (Koordinatenreferenzsystem)
GEO	Group on Earth Observations
GEOSS	Global Earth Observation System of Systems
GIS	Geoinformationssystem
GMES	Global Monitoring for Environment and Security
HDF	Hierarchical Data Format
IPCC	Intergovernmental Panel on Climate Change
MMR	Mass Mixing Ratio (massebezogenes Mischungsverhältnis)
NCL	NCAR Command Language
NetCDF	network Common Data Form
OGC	Open Geospatial Consortium
OWS	OGC Web Service
UCAR	University Corporation for Atmospheric Research
UOM	Unit of Measurement (Maßeinheit)
VMR	Volume Mixing Ratio (volumenbezogenes Mischungsverhältnis)
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WSSD	World Summit on Sustainable Development (Weltgipfel für nachhaltige Entwicklung 2002 in Johannesburg)

Kapitel 1

Einleitung

1.1 Georeferenzierte Daten

Geodaten bzw. georeferenzierte Daten zeichnen sich dadurch aus, dass sie einen räumlichen Bezug zur Erdoberfläche besitzen. Sie spielen seit der Entwicklung des ersten modernen Geoinformationssystems (GIS) *Canada Geographic Information System* (CGIS) in den 1960er Jahren eine zunehmend wichtige Rolle in verschiedensten Anwendungsgebieten. Dazu zählen unter anderem: Kartographie, Stadtplanung, Logistik sowie Umwelt- und Klimaforschung.

Die Art der Daten ist dabei je nach Anwendungsgebiet äusserst verschieden. So kann es sich beispielsweise um Fotos handeln, die an einem bestimmten Ort aufgenommen wurden, um Vektordaten, die den Umriss einer geologischen Formation beschreiben oder um Messungen eines Wolkenradars, die den aktuellen Bedeckungsgrad der Erdoberfläche zeigen.

Georeferenzierte Daten werden in den Klima- und Umweltwissenschaften schon seit längerer Zeit genutzt. Einerseits bei regionalen und globalen Computermodellen und andererseits bei Messdaten.

Weltweit arbeiten zahlreiche Forschergruppen an verschiedensten Klima- und Wettermodellen, während andere Gruppen sich mit der Messdatenerfassung beschäftigen. Messdaten kommen dabei aus einer Vielzahl verschiederer Quellen, zum Beispiel: Satelliten, Messflugzeugen, Wetterballons und Bodenmessstationen.

Um komplexe Zusammenhänge besser verstehen zu können, wird die internationale Zusammenarbeit verschiedener Forschergruppen zunehmend wichtiger. Zu dieser Zusammenarbeit gehört neben dem Austausch von Wissen auch der Austausch von wissenschaftlichen Daten. Im Bereich der Atmosphärenforschung besteht derzeit zum Beispiel verstärktes Interesse im Vergleich von Computermodellen und Messdaten. Diese Vergleiche sollen dabei helfen, Com-

putermodelle zu verbessern und physikalische und chemische Prozesse in der Atmosphäre besser zu verstehen.

Mit der *Group on Earth Observations*[1] (GEO) hat sich im Jahr 2005 nach entsprechenden Forderungen auf dem *Weltgipfel für nachhaltige Entwicklung* 2002 (WSSD) und der *acht führenden Industrienationen* (G8) eine internationale Organisation gegründet, die sich die Bündelung und Unterstützung von geowissenschaftlichen Projekten und Bemühungen zum Ziel gesetzt hat. Sie sieht eine bessere internationale Zusammenarbeit im Bereich der Erdbeobachtung als dringende Voraussetzung, um ihr volles Potential ausschöpfen zu können. Zu ihren Mitgliedern gehören derzeit über 80 Regierungen, über 50 nationale und internationale Organisationen sowie die Europäische Kommission.

GEO hat als Ergebnis des WSSD insgesamt neun Kernthemen identifiziert, die sich gegenseitig bedingen und deswegen in einem gemeinsamen Kontext betrachtet werden müssen: *Katastrophen*, *Gesundheit*, *Energie*, *Klima*, *Wasser*, *Wetter*, *Ökosysteme*, *Landwirtschaft* und *Biodiversität*.

Konkret münden die Ziele der GEO in den Aufbau des *Global Earth Observation System of Systems* (GEOSS), das sich dieser Probleme im Rahmen eines 10-Jahres-Plans von 2005 bis 2015 annehmen soll.

Das *Open Geospatial Consortium* (OGC) ist ein seit 1994 bestehendes internationales Konsortium aus derzeit über 400 einzelnen Unternehmen, Regierungsorganisationen und Universitäten, deren Ziel die Erarbeitung von offenen Geodaten-Schnittstellen für das Internet und die Informationstechnik allgemein ist. Solche Schnittstellen bzw. Standards sind ein wichtiger Teil auf dem Weg zu einfacherem Austausch und gemeinsamer Nutzung von Geodaten. Standards werden beim OGC nach dem Konsensprinzip entwickelt und bei Bedarf auch erweitert.

Die Komplexität des Systems Erde (Abbildung 1.1) ist zu groß, um sie mit isolierten Messungen zu erfassen. Ein besseres Verständnis der Zusammenhänge erfordert die Kombination verschiedener Daten aus weltweit verteilten, unterschiedlichen Quellen. Daten, die für einen bestimmten Zweck erhoben wurden, können oft auch zur Beantwortung anderer Fragestellungen hilfreiche Hinweise geben. So können zum Beispiel zur landwirtschaftlichen Planung erhobene Landnutzungsdaten auch im Kontext von extremen Wetterereignissen wie Flutkatastrophen wichtige Informationen liefern.

Die einfache Kombination von Daten aus unterschiedlichen Quellen scheitert in der Praxis jedoch an mehreren Faktoren: Wissenschaftliche Daten wer-

[1] `http://www.earthobservations.org`

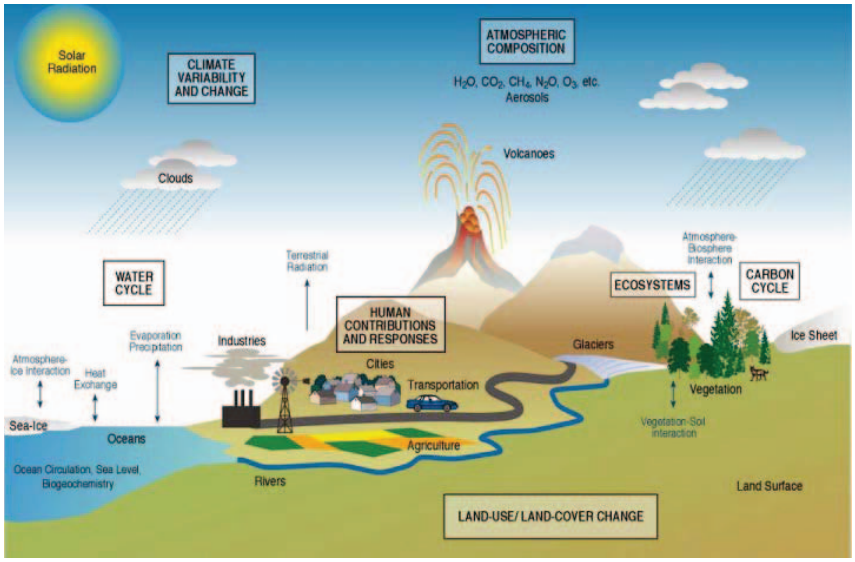

Abbildung 1.1: Schema des komplexen Systems Erde aus [GEO, 2009, S. 6]

den von verschiedenen Einrichtungen und Gruppen erzeugt und vorgehalten und liegen oft in sehr unterschiedlichen Formaten vor. Falls andere Gruppen überhaupt von ihrer Existenz wissen, so müssen sie die fremden Daten zur eigenen Nutzung noch an die lokalen Gegebenheiten anpassen, also beispielsweise gemäß der Anforderungen der eigenen Auswertungssoftware konvertieren. Dies erfordert jedoch entsprechende Einarbeitung in das fremde Format, was Zeit kostet und zusätzliche Fehlerquellen beinhaltet. Aus diesen Gründen werden eigentlich innerhalb der Wissenschaftsgemeinschaft verfügbare Daten nur in geringem Maße von anderen Gruppen genutzt und Synergieeffekte behindert. Hinzu kommen Zugangsbeschränkungen, die relevante Daten (wie zum Beispiel aktuelle Wettervorhersagen in Europa) dem Zugriff durch die Gemeinschaft entziehen.

Ein unkomplizierterer Datenaustausch zwischen verschiedenen Institutionen und Gruppen soll in Zukunft durch verbesserte Vernetzung (*Interoperabilität*) von Datendiensten erreicht werden. Interoperabilität bedeutet in diesem Zusammenhang, dass unterschiedliche und verteilte Datendienste über eine einheitliche Schnittstelle abgefragt werden können. Wie die Daten intern vorgehalten werden spielt dabei keine Rolle. Eine einheitliche Datenschnittstelle ermöglicht die Entwicklung anwendungsspezifischer Software, die mit Daten

aus unterschiedlichen Quellen ohne weitere Eingriffe zusammenarbeitet und diese je nach Bedarf zusammenführen kann.

GEO argumentiert, dass der technische Fortschritt ein System wie GEOSS sowohl möglich als auch nötig macht. Das Problem der Auffindbareit von Daten soll dabei durch die *GEOSS Component and Service Registry* gelöst werden. Dabei handelt es sich um einen Katalogdienst, in den verfügbare Datensätze von beteiligten Institutionen und Gruppen eingetragen werden können. Zur Verbesserung der Interoperabilität dient die *GEOSS Standards and Interoperability Registry*, die Informationen über relevante Standards - wie zum Beispiel die entsprechenden OGC Standards (siehe auch Abschnitt 2.2.1) - und Übereinkünfte sammelt und verfügbar macht.

1.2 TF HTAP

Viele Luftschadstoffe können sich wegen ihrer Langlebigkeit nicht nur lokal und regional rund um ihre Quellen auswirken, sondern können sich auch über die gesamte Hemisphäre oder sogar global verteilen. Dazu zählen zum Beispiel:

- Ozon und seine Vorläufer
- Feinstaub
- Quecksilber
- schwer abbaubare organische Schadstoffe

Das Verständnis solcher Langstreckentransporte von Luftschadstoffen hilft dabei, ihre Auswirkungen auf die regionale Luftqualität besser abzuschätzen.

Bei der *Task Force on Hemispheric Transport of Air Pollution*[2] (TF HTAP) handelt es sich um eine im Jahr 2004 gegründete Arbeitsgruppe, welche den interkontinentalen Transport von Luftschadstoffen auf der nördlichen Hemisphäre untersuchen und so zu einem besseren Verständnis dieser Transportprozesse beitragen soll. Sie dient als Forum für internationale wissenschaftliche Zusammenarbeit und als Vermittler zwischen der wissenschaftlichen Gemeinschaft und der internationalen Politik.

Die in der Literatur vorhandenen Ergebnisse zu Schadstofftransport und Quellen-Senken Beziehungen sind im allgemeinen schwer miteinander zu vergleichen. Gründe dafür sind zum Beispiel die unterschiedliche Methodik und die unterschiedlichen Metriken, die von verschiedenen Forschungsgruppen zur Bewertung benutzt werden. Zur Verbesserung der Vergleichbarkeit von verschiedenen Modellen hat sich das Mittel der kollaborativen Multimodell-Studien

[2] `http://www.htap.org`

und -Vergleiche herausgebildet. Gegenüber einzelnen Modellen haben diese den Vorteil, dass über Durchschnittswerte und (Standard-)Abweichungen robustere Vorhersagen getroffen werden können, die besser mit Beobachtungsdaten übereinstimmen, als die individuellen Modellvorhersagen [Schulz u. a., 2006; Stevenson u. a., 2006; Reichler und Kim, 2008]. Zusätzlich dient die Variablilität der Ergebnisse als Maß für die noch vorhandenen Unsicherheiten in der Modellierung der Prozesse. Zahlreiche solcher Studien wurden in der Vergangenheit bereits durchgeführt, wobei diese sich in Dauer, Komplexität, Ausführung und Beteiligung stark unterschieden.

Unter der Federführung der TF HTAP wurde im Jahr 2006 ein Modellvergleich durchgeführt, der insgesamt 29 verschiedene (Chemie-)Transportmodelle umfasste. Dazu wurden insgesamt vier Regionen definiert, deren Quellen-Senken-Verhalten genauer untersucht werden sollte: Nordamerika (NA), Europa und Nordafrika (EU), Ostasien (EA) und Südasien (SA). Ziel war die Quantifizierung von Unsicherheiten speziell im Bezug auf Ozon und seine Vorläufer oder Aerosole durch die gezielte Untersuchung der Diversität von Modellen in den Bereichen Transport, Chemie, Emissionen und weitere Randbedingungen. Die Ergebnisse dieser Studie wurden zusammen mit einer Analyse von Beobachtungsdaten in einem Zwischenbericht (HTAP2007) veröffentlicht.

Für den HTAP-Modellvergleich wurden insgesamt vier Arbeitspakete definiert. In *ExperimentSet1* (SR) wurden die Quellen-Senken (*source-receptor*) Zusammenhänge in den vier ausgewählten Regionen für verschiedene Emissionsszenarien mit einheitlichen klimatologischen und meteorologischen Parametern des Jahres 2001 simuliert. *ExperimentSet2* (TP) sollte durch Simulation des Transports von passiven Tracersubstanzen die Parametrisierung der verschiedenen Modelle testen und so die Unterschiede zwischen den Modellen im *ExperimentSet1* diagnostizieren helfen. In *ExperimentSet3* (ES) wurden Simulationen des Transports ausgewählter Luftschadstoffe (Quecksilber, Ozon, Aerosole, langlebige organische Schadstoffe) mit Daten aus einer 2004 durchgeführten Messkampagne (ICARTT) verglichen. In *ExperimentSet4* schließlich wurden Zukunftsszenarien zur Beurteilung der Auswirkungen von Emissions- und Klimaveränderungen durchgeführt. Bei den *Future Emissions*-Läufen (FE) wurde der Effekt von Emissionsänderungen untersucht, indem Annahmen für zukünftige Emissionen mit den meteorologischen Randbedingungen des Jahres 2001 simuliert wurden. In den *Future Climate*-Läufen (FC) hingegen wurden meteorologische Daten aus Klimamodellläufen (*Intergovernmental Panel on Climate Change*[3] (IPCC) SRES A2[4] Szenarios für das Jahr 2100) zusammen mit Emissionen von 2001 verwendet.

[3] `http://www.ipcc.ch/`

[4] `http://sedac.ciesin.columbia.edu/ddc/sres/`

Eine Auswertung des *ExperimentSet1* zeigt, dass Modelle und Messwerte für die Ozonkonzentrationen in Europa für das ganze Jahr 2001 im Allgemeinen gut übereinstimmen. In Nordamerika und Japan werden die tatsächlichen Ozonkonzentrationen im Sommer und frühen Herbst jedoch systematisch überschätzt [Fiore u. a., 2009].

Im Rahmen dieser Studie betreibt das *Institut für Energie- und Klimaforschung - Troposphäre (IEK-8)* im Forschungszentrum Jülich seit 2006 einen zentralen Datenserver, auf dem derzeit ca. 9TB Speicherplatz zur Verfügung stehen. Dieser dient zum einen zur Archivierung der durchgeführten Modellvergleiche und als zentrale Plattform für Auswertungen der Nutzer. Zum anderen fungiert er als Enwicklungsplattform für Prozessierungswerkzeuge und eine offene Web-Schnittstelle in Form des WCS-Servers, der in Kapitel 4.2 dieser Arbeit beschrieben wird.

1.3 Ziele

Das Ziel dieser Arbeit ist die Vereinfachung des Umgangs mit und des Zugriffes auf das HTAP-Datenarchiv im Speziellen und auf georeferenzierte Erdbeobachtungsdaten im CF-NetCDF-Format im Allgemeinen.
Die Aufgabe gliedert sich konkret in die folgenden Teilaspekte:

Gewährleistung der Standardkonformität der HTAP-Datensätze

Die für den HTAP-Modellvergleich vorliegenden Datensätze wurden von einer großen Zahl verschiedener Arbeitsgruppen beigesteuert. Die Standardkonformität dieser Datensätze ist für die weitere automatisierte Verarbeitung von entscheidender Bedeutung. Eine manuelle Kontrolle mit den zuvor vereinbarten Regeln hat sich jedoch als sehr arbeitsintensiv herausgestellt, weswegen sie möglichst weitgehend automatisiert werden soll.

Entwicklung von Interpolations- und Vergleichsoperatoren für HTAP-Datensätze

Die am HTAP-Modellvergleich beteiligten Modelle verwenden zahlreiche unterschiedliche Gitterweiten für ihre Koordinatengitter. Um die Vergleichbarkeit der Datensätze miteinander zu verbessern, müssen diese deswegen zunächst sowohl in horizontaler als auch in vertikaler Richtung vereinheitlicht werden. Zu diesem Zweck sollen Interpolationsoperatoren entwickelt werden, die eine solche Vereinheitlichung leisten können. Dabei bestehen insbesondere bei der vertikalen Interpolation noch ungelöste Probleme, die genauer betrachtet

werden müssen. Zusätzlich soll untersucht werden, inwieweit sich die implementierte Interpolation auf die wissenschaftlich relevanten Erhaltungsgrößen *atmosphärische Säulendichte* und *globale Masse* auswirkt.

Zum - durch die Interpolation ermöglichten - Vergleich der Datensätze miteinander sollen zudem weitere Operatoren entwickelt werden, die beispielsweise die Mittelung oder Differenzbildung von verschiedenen Modellergebnissen erlauben.

Entwicklung einer Web-Schnittstelle zur standardisierten und dynamischen Bereitstellung von Klima- und Wetterdaten

Die Standardkonformen HTAP-Datensätze sollen für andere interessierte Institutionen auf einfachem Wege zugänglich gemacht werden. Dies soll auch die Notwendigkeit zur Vergabe von Nutzeraccounts auf dem HTAP-Datenserver mit alleinigem Ziel des Zugriffs auf die Datensätze eliminieren. Zu diesem Zweck soll auf dem HTAP-Datenserver eine auf dem offenen WCS-Standard aufsetztende Web-Schnittstelle geschaffen werden. Konkret soll dazu in Zusammenarbeit mit der *Washington University in St. Louis* ein dort entwickelter Prototyp einer WCS-Server-Software zur Praxistauglichkeit im Umgang mit den HTAP-Daten weiterentwickelt werden.

Kapitel 2

Beschreibung der zu verarbeitenden Daten

2.1 Das NetCDF-Format

Allgemein handelt es sich bei *NetCDF* (network Common Data Form) um eine Kombination von Programmbibliotheken und einem von diesen Bibliotheken definierten Datenformat, das entsprechend als *NetCDF-Format* bezeichnet wird. Federführend in der Entwicklung und Betreuung von NetCDF ist *Unidata*, ein Programm der *University Corporation for Atmospheric Research* (UCAR), welches hauptsächlich von der amerikanischen *National Science Foundation* (NSF) finanziert wird.

Das Einsatzgebiet von NetCDF ist der Austausch von Vektor-orientierten wissenschaftlichen Daten. Dazu bietet die NetCDF-Bibliothek eine einheitliche Schnittstelle (API) für Erstellung, Modifikation von und Zugriff auf NetCDF-Daten. Die Bibliothek steht für zahlreiche verschiedene Programmiersprachen zur Verfügung. Unterstützung für C/C++, Fortran77 und Fortran90 sind in der von Unidata bereit gestellten Distribution bereits fest eingebaut. Für weitere Sprachen wie Ada, IDL, Java, MATLAB, Perl, Python, R, Ruby und Tcl/Tk sind NetCDF-Schnittstellen entweder direkt in die Sprache integriert oder von Drittherstellern verfügbar.

Beim NetCDF-Datenformat selbst handelt es sich um ein Binärformat, das Unidata folgendermaßen charakterisiert[1]:

- *selbstbeschreibend*: eine Datei beinhaltet (Meta-)Informationen über die in ihr enthaltenen Daten
- *portabel*: Daten können auf Systemen mit unterschiedlichen internen Re-

[1] `http://www.unidata.ucar.edu/software/netcdf/docs/faq.html#whatisit`

präsentationen von elementaren Datentypen (integer, float, character) ausgelesen werden

- *skalierbar*: effizienter Zugriff auf Untermengen großer Datensätze
- *erweiterbar*: Daten können einer entsprechend strukturierten Datei hinzugefügt werden, ohne dass Datensätze verschoben oder die Dateistruktur neu definiert werden muss
- *parallel und gemeinsam nutzbar*: ein schreibender und mehrfache lesende Zugriffe auf die gleiche Datei zur gleichen Zeit sind möglich
- *archivierbar*: Kompatibilität aktueller und zukünftiger Versionen zu allen älteren NetCDF-Formaten

Das NetCDF-Format ist das Basisformat der in dieser Arbeit behandelten Daten.

2.1.1 Versionen

Zu unterscheiden ist zwischen der Bibliotheks- und der Dateiformat-Version.

Die derzeit aktuelle Version der NetCDF-Bibliothek trägt die Versionsnummer 4.1 und wurde im April 2010 veröffentlicht.

Das derzeit am häufigsten benutzte Dateiformat ist *NetCDF-3*, das 1996 mit der Bibliotheksversion 3.1a eingeführt wurde. Mit Version 3.6 wurde das Dateiformat durch 64-bit Adressierung (vorher 32-bit) erweitert, um die Speicherung von größeren Datensätzen zu ermöglichen. Zur Unterscheidung zwischen den beiden Varianten wird das ursprüngliche Format seitdem als *classic*- und die Erweiterung als *64-bit offset*- oder *large file*-Format bezeichnet.

Mit Bibliotheksversion 4.0 wurde 2008 weiteres Dateiformat (*NetCDF-4*) eingeführt, das auf dem *Hierarchical Data Format*[2] (HDF) aufbaut. Die Nutzung von HDF als Basis für NetCDF-4 ermöglicht unter anderem größere Flexibilität bei der Speicherung von Daten und eine bessere Speicherplatznutzung durch (optionale) transparente Kompression.

2.1.2 Datentypen

NetCDF unterstützt die gängigen numerischen Datentypen, die aus Programmiersprachen wie C und Fortran bekannt sind. Im Speziellen sind dies die vorzeichenbehafteten Ganzzahltypen *byte* (8-bit), *short* (16-bit) und *int* (32-bit) sowie die Fließkommatypen *float* (32-bit) und *double* (64-bit). Hinzu kommt

[2] `http://www.hdfgroup.org/`

der nicht vorzeichenbehaftete Typ *char* (8-bit), der auch klassisch zur Darstellung von Zeichenketten (Strings) genutzt wird, wie es beispielsweise in C der Fall ist.

Mit NetCDF-4 wurden die Liste der verfügbaren Datentypen erweitert. Die zuvor genannten vorzeichenbehafteten Typen *short* und *int* bekamen die jeweiligen nicht vorzeichenbehafteten Gegenstücke *ushort* und *uint*. Zusätzlich wurden 64-bit Ganzzahltypen in beiden Varianten (*int64* und *uint64*), ein echter Typ für Zeichenketten (*string*) und die Möglichkeit zur Nutzung benutzerdefinierter Datentypen eingeführt.

2.1.3 logischer Aufbau

Eine NetCDF-3-Datei besteht im Regelfall aus **Dimensionen**, **Variablen** und **Attributen**. NetCDF-4 bzw. das darunterliegende HDF-Format bietet zusätzlich noch so genannte **Gruppen**, die zur weiteren Strukturierung eines Datensatzes eingesetzt werden können. Auf diese wird im Folgenden jedoch nicht näher eingegangen.

Dimensionen bestehen aus einem Dimensionsnamen und einer ganzzahligen positiven Länge. Neben konstanten Dimensionslängen lassen sich auch eine (bis einschließlich NetCDF-3) oder mehrere (ab NetCDF-4) Dimensionen als dynamisch (*unlimited*) definieren. Die Länge einer solchen Dimension passt sich dann dem aktuellem Bedarf des Datensatzes an. Eine Dimension kann eine reale physikalische Dimension in Raum und Zeit sein oder zur Indizierung anderer Größen dienen, etwa des Laufindex eines Modelllaufes oder einer Liste von vordefinierten Wellenlängen. Sie bilden die Basis für Variablendefinitionen, indem sie deren Form und Größe bestimmen.

Variablen dienen zur Speicherung der eigentlichen Daten. Allgemein wird eine Variable über ihren Namen, ihren Datentyp und ihre Form (*shape*) definiert. Alle Elemente einer Variablen haben somit auch immer den gleichen Datentyp. Die Form kann entweder skalar oder feldartig sein. Skalare Variablen hängen von keiner Dimension ab und können entsprechend auch nur einen einzelnen Datenwert aufnehmen. Feldartige Variablen werden über ihre Abhängigkeit von einer oder mehreren Dimensionen definiert und bekommen dadurch ihre Form zugewiesen. Entlang einer *unlimited*-Dimension kann eine Variable zu einem späteren Zeitpunkt um neue Einträge erweitert werden. Ein häufiger Anwendungsfall ist beispielsweise die Aufzeichnung einer Zeitreihe. In diesem Fall wäre die Zeitdimension als *unlimited* definiert. Der Variablen können dann nach jeder Messung die neuen Daten hinzugefügt werden.

Die Definition von Variablen über eine Menge von ihnen zugeordneten Dimensionen führt dazu, dass sich NetCDF vor allem zur Speicherung von gitter- bzw. quarderförmig strukturierten Daten eignet. Die Herstellung eines Zusam-

menhangs zwischen den auf Dateiebene definierten Dimensionen und wissenschaftlich bedeutsamen Koordinaten ist nicht Teil des NetCDF-Konzeptes und muss durch geeignete Konventionen geregelt werden (siehe Abschnitt 2.2.2).

Attribute dienen zur Aufnahme von Metadaten, wodurch eine Datei erst im Sinne von NetCDF *selbstbeschreibend* wird. Grundsätzlich können Attribute jeder Variablen sowie global der ganzen Datei zugewiesen werden. Analog zu Variablen besitzen auch Attribute einen Namen sowie einen Datentyp, der aus der Menge aller in NetCDF verfügbaren Typen stammen kann. Die Form eines Attributwertes ist im Gegensatz zu einer Variablen immer eindimensional. Skalare Attributwerte werden als einelementige Felder realisiert.

2.1.4 Metadaten

Durch die große Flexibilität der Attribute lässt sich NetCDF einerseits sehr vielseitig einsetzen, erschwert andererseits aber auch die automatische Verarbeitung von Datensätzen und behindert dadurch Interoperabilität. Unidata selbst schlägt deswegen schon einige grundlegende Konventionen vor, die den Umgang vereinheitlichen sollen. Dazu zählt zum Beispiel die Angabe der physikalischen Einheit einer Variable in ihrem *units*-Attribut sowie ein über das *missing_value*-Attribut angegebener Füllwert für fehlende Datenpunkte. Eine Herausforderung ist die Repräsentation von Zeitachsen in NetCDF, da gebräuchliche Datums- und Zeitdarstellungen sich nicht in einem elementaren NetCDF-Datentyp ausdrücken lassen. Eine von Unidata vorgeschlagene und weit verbreitete Möglichkeit besteht darin, das *units*-Attribut der Zeitachse nach dem Schema „`<Zeiteinheit> since <Referenzdatum>`“ aufzubauen und der Zeitachse selbst dann entsprechende Offsetwerte zuzuweisen. Ein Beispiel dieser Anwendung zeigt Abbildung 2.1, wo *time* somit die Zeit „12 Uhr am 01.01.1996“ enthalten würde.

Zusätzlich gibt es zahlreiche weitergehende NetCDF-Konventionen für verschiedene Einsatzgebiete, die normalerweise die von Unidata vorgeschlagenen Konventionen berücksichtigen. Eine in der Klima- und meteorologischen Forschung weit verbreitete Konvention ist die auch in dieser Arbeit genutzte *Climate and Forecast Metadata Convention* (CF), auf die in Abschnitt 2.2.2 näher eingegangen wird.

2.1.5 CDL-Notation

Da das NetCDF-Format als Binärformat nicht direkt menschenlesbar ist, existiert mit *CDL* (network Common Data form Language) eine menschenlesbare alternative Darstellungsmöglichkeit von NetCDF-Inhalten. Unidata liefert mit

`ncdump` und `ncgen` zwei Programme mit der NetCDF-Distribution aus, mit denen sich Daten in beide Richtungen konvertieren lassen.

Das folgende Listing zeigt in Anlehnung an den NetCDF Users Guide [Unidata Program Center, 2010] eine Beispieldatei in CDL-Darstellung. Die Beispieldatei zeigt dabei auch den Einsatz einiger in den Attribut-Konventionen vorgeschlagenen Standard-Attribute wie *units* und *long_name*.

```
netcdf cdl_beispiel {
 // Beispiel eines NetCDF Datensatzes in CDL-Notation
 dimensions:  // Dimensionsnamen werden zuerst definiert
   lat = 5, lon = 10, level = 4, time = unlimited;
 variables:  // Variablen: Namen, Typen, Form und Attribute
   float  temp(time,level,lat,lon);
              temp:long_name = "temperature";
              temp:units = "celsius";
   float  rh(time,lat,lon);
             rh:long_name = "relative humidity";
             rh:valid_range = 0.0, 1.0; // min und max
   int    lat(lat), lon(lon), level(level), time(time);
            lat:units = "degrees_north";
            lon:units = "degrees_east";
            level:units = "hPa";
            time:units = "hours since 1996-01-01";
 // globale Attribute
           :comment = "Beispieldatensatz";
 data:  // optionale Variablenzuweisungen
   level  = 1000, 850, 700, 500;
   lat    = 20, 30, 40, 50, 60;
   lon    = 0,20,40,60,80,100,120,140,160,180;
   time   = 12;
   rh     =.5,.2,.4,.2,.3,.2,.4,.5,.6,.7,
            .1,.3,.1,.1,.1,.1,.5,.7,.8,.8,
            .1,.2,.2,.2,.2,.5,.7,.8,.9,.9,
            .1,.2,.3,.3,.3,.3,.7,.8,.9,.9,
             0,.1,.2,.4,.4,.4,.4,.7,.9,.9; }
```

Abbildung 2.1: NetCDF-Beispieldatei in CDL-Notation mit Verwendung einiger im NetCDF Users Guide [Unidata Program Center, 2010] vorgeschlagener Attribut-Konventionen (`units`, `long_name`, `valid_range`)

2.2 Standards und Konventionen

2.2.1 OGC-Standards

Das OGC verwaltet und entwickelt aktuell über 30 Standards aus dem Bereich der raumbezogenen Informationsverarbeitung bzw. der Geodaten, die unter anderem der Herstellung von mehr Interoperabilität im Bereich der Geodatendienste dienen.

Einer der bekanntesten vom OGC verwalteten Standards ist die auf XML basierende Geodaten-Auszeichnungssprache *KML*[3], die ursprünglich als *Keyhole Markup Language* entwickelt wurde und im Jahr 2004 durch Kauf der Firma *Keyhole, Inc* in den Besitz von Google überging. Google reichte KML dann 2007 als Standardvorschlag beim OGC ein, wo Version 2.2 im Jahr 2008 zu einem offiziellen OGC Standard erklärt wurde. Verwendung findet KML in erster Linie in den Google-Produkten *Google Earth*[4] und *Google Maps*[5].

WMS

Ein weiterer Standard, der zunehmend Verbreitung findet, ist der *Web Map Service*[6] (WMS). Dieser definiert eine Schnittstelle zum Zugriff auf georeferenzierte Karten im Form von Bildern. Mit Hilfe dieser Schnittstelle lassen sich mit wenig Aufwand interaktive Karten in Webseiten einbetten, die dynamisch die anzuzeigenden Kartenkacheln nachladen können, wie es beispielsweise das freie Kartenframework *OpenLayers*[7] tut.

WFS

Der *Web Feature Service*[8]-Standard (WFS) definiert eine Schnittstelle zur Abfrage auf georeferenzierte Vektordaten, die auch *features*[9] genannt werden. Ein *feature* kann in diesem Zusammenhang jedes Objekt sein, das in Zeit und Raum verortet werden kann. Praktisch sind dies zum Beispiel Gebäude, Strassen oder Wetterfronten. Im Vergleich zu WMS lässt sich WFS als eine Schnitstelle verstehen, die Zugriff auf die Vektordaten ermöglicht, die als Grundlage für eine Kartendarstellung dienen. Neben dem Abruf bietet WFS

[3] `http://www.opengeospatial.org/standards/kml`
[4] `http://www.google.com/earth/`
[5] `http://maps.google.com`
[6] `http://www.opengeospatial.org/standards/wms`
[7] `http://openlayers.org/`
[8] `http://www.opengeospatial.org/standards/wfs`
[9] `http://www.opengeospatial.org/ogc/glossary/f`

zusätzlich auch Möglichkeiten zur Erstellung und Modifikation von Objekten (*WFS-Transactional*, WFS-T).

WCS

In der vorliegenden Arbeit wird der OGC-Standard *WCS*[10] (Web Coverage Service) genutzt. Dieser Standard definiert - wie auch WMS - eine plattformunabhängige Web-Schnittstelle zur Bereitstellung bzw. Abfrage von georeferenzierten Datensätzen, die bei WCS als *Coverages* bezeichnet werden.

Das OGC versteht ein *Coverage* abstrakt als eine Sammlung von Merkmalen (*ranges*), die innerhalb eines räumlich und zeitlich begrenzten Bereiches miteinander verknüpft sind[11]. Im Kontext von Geoinformationssystemen handelt es sich dabei um zwei- oder höherdimensionale Beschreibungen von Phänomenen, die sich im Bezug auf die Erdoberfläche einem Ort zuordnen lassen. WCS beschäftigt sich im Speziellen mit so genannten *grid coverages*, wobei den einzelnen Merkmalsausprägungen Positionen in einem in horizontaler Richtung regulären Koordinatengitter zugeordnet werden. Andere Koordinatenachsen wie vertikale Achsen oder Zeit müssen jedoch nicht regulär sein. Zu solchen *grid coverages* zählen beispielsweise digitale Satellitendaten und Luftbilder der Erde und generell digitale Daten, die als diskrete Werte an allen enthaltenen Gitterpunkten vorliegen. Die an den Gitterpunkten vorliegenden Daten beschreiben dabei oft eine räumliche Umgebung - zum Beispiel den Mittelwert einer Gitterzelle - und nicht den singulären Gitterpunkt.

Das Format der zurückgelieferten Daten wird im Standard offen gelassen. Stattdessen wird auf separate Dokumente verwiesen, die Profile für jedes verwendete Format definieren sollen. Derzeit am weitesten verbreitet ist die Verwendung von Rasterbildformaten wie GeoTIFF, gif oder jpeg. Diese haben den Vorteil, dass sie sich leicht implementieren lassen - bieten andererseits aber auch nur wenig Mehrwert gegenüber der Nutzung des WMS-Standards. Die Nutzung eines aussagekräftigeren Formates ist deswegen wünschenswert.

Für das NetCDF-Format existiert derzeit noch kein vom OGC als Standard akzeptiertes Profil. Im Rahmen des *GALEON*-Netzwerkes[12] (Geo-interface to Atmosphere, Land, Earth, Ocean, NetCDF) gab es jedoch einige Versuche zur Nutzung des WCS-Protokolls mit NetCDF als zugrundeliegendem Datenformat. An diesen Versuchen hat neben anderen Einrichtungen auch Unidata aktiv teilgenommen.[13] Diese Experimente mündeten dann im Mai 2008

[10] `http://www.opengeospatial.org/standards/wcs`
[11] `http://www.opengeospatial.org/ogc/glossary/c`
[12] `http://www.ogcnetwork.net/galeon`
[13] `http://www.unidata.ucar.edu/projects/THREDDS/GALEON/netCDFprofile-short.htm`

in einen ersten Entwurf eines OGC Diskussionspapiers, das derzeit in Version 2.2 von April 2009 vorliegt [Domenico und Nativi, 2009]. Es soll dazu dienen, NetCDF als ein offiziell unterstütztes Format in kommenden Versionen des WCS-Standards zu etablieren.

Ein standardisiertes NetCDF-Profil ist auch deswegen nötig, weil für interoperable Anwendungen nicht nur die Schnittstellen selbst, sondern auch die zurückgelieferten Daten genau definiert sein müssen. Im Gegensatz zu Bildformaten bietet NetCDF als komplexes Format deutlich mehr Interpretationsspielraum, so dass es sich ohne genau definiertes Profil nur in Ausnahmefällen interoperabel nutzen lassen wird.

2.2.2 CF Metadata Conventions

Bei den *Climate and Forecast Metadata Conventions* (CF) handelt es sich um einen Satz von Konventionen, die auf dem NetCDF-Format aufsetzen und ursprünglich für den Einsatz in Klimawissenschaften ausgelegt waren, zunehmend aber auch für Beobachtungsdaten genutzt werden. Die erste Version CF-1.0 wurde im Oktober 2003 veröffentlicht, die derzeit aktuelle Version CF-1.5 im Oktober 2010.

Ziel der Konvention ist die Definition von standardisierten Metadaten und Konsistenzkriterien, mit denen sich der Inhalt einer NetCDF-Datei bzw. der in ihr enthaltenen Variablen und deren räumliche und zeitliche Ausrichtung vollständig beschreiben lassen. Diese Standardisierung erlaubt den einfacheren Austausch von Daten und Programmen zwischen verschiedenen Einrichtungen und Gruppen und lässt sich somit auch als Maßnahme zur Herstellung von besserer Interoperabilität verstehen.

Zu den Grundprinzipien von CF gehören laut eigener Aussage [Lawrence u. a., 2006]:

- selbstbeschreibende Datensätze, die ohne weitere externe Datenquellen interpretiert werden können (im Gegensatz zu z.B. dem GRIB-Format)
- Konventionen werden nur für konkret bekannte Nutzungen definiert und nach Bedarf erweitert, statt über zukünftige Nutzung zu spekulieren
- die Nutzung sollte sowohl für Erzeuger von Daten wie auch für deren Verwender einfach sein
- die Semantik der Metadaten sollte sowohl für Menschen wie auch für Programme verständlich sein
- Redundanzen sollten minimiert werden, damit Inkonsistenzen bei der Datenerzeugung möglichst vermieden werden

Entstehung und Nutzung

Entstanden sind die CF-Konventionen auf Basis der deutlich simpleren *COARDS*-Konventionen (Cooperative Ocean/Atmosphere Research Data Service), die 1995 von zwei Universitäten und der *National Oceanic and Atmospheric Administration* (NOAA) veröffentlicht wurden. Primäres Ziel war die Schaffung eines besseren Metadatenstandards für die Klimamodellierung, der die COARDS-Konventionen zweckmäßig erweitert. Die CF-Konventionen gelten derzeit als de facto Standard für die Speicherung von Simulationsergebnissen und finden zum Beispiel bei den gegenwärtig durchgeführten Klimasimulationen für den fünften Sachstandsbericht (*Fifth Assessment Report*, AR5) des IPCC Verwendung. Eine Anwendung von CF auf Beobachtungsdaten ist ebenfalls sinnvoll, weil durch die einheitlichen Beschreibungskonzepte der Vergleich von Simulation und Beobachtung erheblich vereinfacht wird. Gerade im Kontext von GEO (siehe Abschnitt 1.1) ist eine solche Entwicklung wünschenswert. Inhaltlich sind keine gravierenden Probleme bei der Ausdehnung der CF-Konventionen auf Beobachtungsdaten zu erwarten, da letztendlich unabhängig von der Datenquelle sehr ähnliche Sachverhalte beschrieben werden müssen.

Um den Anforderungen dieser wachsenden Nutzergemeinde besser gerecht werden zu können, wurde die Pflege und Weiterentwicklung der Konventionen von den ursprünglichen fünf Autoren in einen gemeinschaftlichen Entwicklungsprozess überführt. Über eine Mailingliste und ein Bugtracking-System können interessierte Nutzer dort an der zukünftigen Weiterentwicklung mitwirken[14].

Obwohl die CF-Konventionen speziell im Hinblick auf den Einsatz mit NetCDF entwickelt wurden, sind die meisten Konzepte sehr allgemein gehalten und lassen sich somit potentiell auch auf viele andere Dateiformate übertragen. In diesem Sinne können sie auch als allgemeine Konzepte zum Umgang mit Metadaten in den Geowissenschaften verstanden werden - selbst wenn das im Einzelfall genutzte Datenformat nicht NetCDF ist. Bei Verwendung gleicher Metadatenstandards wäre eine Konvertierung zwischen solchen Formaten dann erheblich einfacher zu bewerkstelligen.

Konzept

Im Folgenden werden einige der grundlegenden Ideen von CF kurz erläutert. Die vollständige Definition der aktuellen Version CF-1.5 umfasst über 70 Seiten und ist auf der offiziellen CF-Webseite verfügbar[15].

[14] `http://cf-pcmdi.llnl.gov/discussion`
[15] `http://cf-pcmdi.llnl.gov/documents/cf-conventions`

Grundsätzlich versucht CF, die Menge an verschiedenen Attributnamen überschaubar zu halten und ein Attribut so allgemein wie möglich zu benutzen. Da eine einzelne Datei verschiedenste Arten von Informationen enthalten kann, ordnet CF die Metadaten den Variablen zu. Variablennamen sollen dabei irrelevant sein, die Bedeutung von Variablen wird ausschließlich über Attribute definiert. Einzige Ausnahme sind die Koordinatenvariablen, die laut Unidata-Konvention genauso heißen müssen wie ihre (einzige) Dimension.

Auf Dateiebene sind lediglich einige allgemeine Attribute definiert, die Informationen über die Datei als solche enthalten, zum Beispiel ein beschreibender Titel (`title`), der Name der Institution, die die Daten veröffentlicht hat (`institution`) oder Referenzen auf Dokumentationen und Publikationen (`references`). Die globalen Attribute dienen damit eher zur Kennzeichnung der Herkunft von Daten.

Variablen zugeordnete Attribute hingegen beschreiben die Daten selbst. Eines der wichtigsten Attribute zur Interpretation der Variablenwerte ist das `units`-Attribut. Dieses muss einen zur (von Unidata entwickelten) Software *UDUNITS*[16] konformen String der (physikalischen) Einheiten der Variablenwerte enthalten. Beispiele für solche Attributwerte sind etwa einzelne SI-Einheiten wie „`hPa`“, zusammengesetzte SI-Einheiten wie „`m s-2`“ (entspricht $m \cdot s^{-2}$) oder auch die bereits erwähnte Unidata-Konvention für Zeitachsen wie „`seconds since 2000-01-01`“.

Im Gegensatz zum bereits in COARDS enthaltenen `long_name`-Attribut, das wegen seiner freien Form jedoch nur menschenlesbar ist, wurden für das neue `standard_name`-Attribut mit dem *standard name table*[17] systematische, eindeutige Bezeichner für Größen eingeführt. Die Definitionstabelle enthält neben dem Bezeichner auch ihre physikalischen Einheiten sowie eine kurze menschenlesbare Erklärung der jeweiligen Größe. Wie die CF-Konventionen selbst wird auch diese Tabelle nach Bedarf erweitert. Derzeit enthält die Tabelle über 2000 Einträge aus verschiedenen geowissenschaftlichen Bereichen.

Für einige Analysen ist es wichtig zu wissen, welche Ausdehnung die einem Wert zugrunde liegende Zelle hat, sowie in welchem Verhältnis die Datenwerte mit der Zelle stehen. Die Ausdehnung kann über zusätzliche *Bounds*-Variablen in der Datei hinterlegt werden, die über das `bounds`-Attribut in der entsprechenden Koordinatenvariablen referenziert werden. Diese Möglichkeit ist vor allem bei unregelmäßigen Koordinatengittern oder generell bei Zellen mit unterschiedlichen Abmessungen nützlich. Zur Kennzeichnung der (statistischen) Methode, mit der die Werte gewonnen wurden, dient das `cell_methods`-Attribut. Es wird an die betreffenden Datenvariablen angehängt und informiert

[16] `http://www.unidata.ucar.edu/software/udunits/`
[17] `http://cf-pcmdi.llnl.gov/documents/cf-standard-names/`

für jede Koordinatenachse über die in dieser Richtung benutzte Operation. Derzeit definierte Operationen sind zum Beispiel: `point`, `sum`, `maximum`, `median` oder `mean`. Ein Temperaturwert könnte zum Beispiel als Maxiumum des letzten Messintervalls ermittelt worden sein (`cell_methods = "time: maximum"`), während eine Niederschlagsmenge als Summe der im Messzeitraum gefallenen Niederschläge aufgezeichnet wird (`cell_methods = "time: sum"`).

CF-Beispiel

Abbildung 2.2 zeigt ein einfaches Beispiel einer Zeitreihe von Druck, Maximaltemperatur und Niederschlagsmenge mit CF-konformen `standard_name`-, `units`-, `cell_methods`- und `bounds`-Attributen.

Das `units`-Attribut der Variablen `time` legt die Referenzzeit auf den 19.04.1998 um 6 Uhr und die Einheit des Zeitoffset auf „Stunden“ fest. Die Datenpunkte reichen somit von der Referenzzeit aus gesehen 0 bis 48 Stunden in die Zukunft, also vom 6 Uhr am 19.04.1998 bis 6 Uhr am 21.04.1998. Die Variable `time_bnds`, die von `time:bounds` referenziert wird, enthält die Grenzen des jeweiligen Messzeitraumes. Diese erstrecken sich von Datenpunkt aus gesehen jeweils 12 Stunden in die Vergangenheit. Da für jeden Zeitpunkt Ober- und Untergrenzen benötigt werden, hat `time_bnds` eine zusätzliche Dimension (`nv`) der Länge 2. Die restlichen `units`-Attribute sind einfache physikalische Einheiten der gemessenen Größen.

Die `cell_methods`-Attribute der Variablen `pressure`, `maxtemp` und `ppn` geben an, wie die Daten im jeweiligen Messzeitraum ermittelt wurden: Der Druck (`pressure`) wurde jeweils punktuell zum angegebenen Zeitpunkt erfasst, der Temperaturwert (`maxtemp`) ist die Maximaltemperatur des Zeitraumes und die Niederschlagsmenge (`ppn`) ergibt sich durch Summation über den Messzeitraum.

```
dimensions:
  time = UNLIMITED; // (5 currently)
  station = 10;
  nv = 2;
variables:
  float pressure(time,station);
    pressure:standard_name = "air_pressure";
    pressure:units = "kPa";
    pressure:cell_methods = "time: point";
  float maxtemp(time,station);
    maxtemp:standard_name = "air_temperature";
    maxtemp:units = "K";
    maxtemp:cell_methods = "time: maximum";
  float ppn(time,station);
    ppn:long_name="depth of water-equivalent precipitation";
    ppn:standard_name="lwe_thickness_of_precipitation_amount
       ";
    ppn:units = "mm";
    ppn:cell_methods = "time: sum";
  double time(time);
    time:standard_name = "time";
    time:units = "hours since 1998-04-19 06:00:00";
    time:bounds = "time_bnds";
  double time_bnds(time,nv);
data:
  time = 0., 12., 24., 36., 48.;
  time_bnds = -12.,0., 0.,12., 12.,24., 24.,36., 36.,48.;
```

Abbildung 2.2: Beispiel einer Zeitreihe von Druck, Maximaltemperatur und Niederschlagsmenge in CDL-Notation mit CF-konformen `cell_methods`-, `bounds`-, `standard_name` und `units`-Metadaten, angelehnt an [Eaton u. a., 2010, Beispiel 7.4]

2.3 Modell- und Beobachtungsdaten

Dieser Abschnitt gibt einen kurzen Überblick über die verschiedenen Arten von Geodaten und deren Unterscheidungsmerkmale im Bezug auf die vorliegende Arbeit. Wie eingangs schon beschrieben (Abschnitt 1.1), deckt der generische Begriff „Geodaten“ noch vielfältige andere Einsatzgebiete ab, die hier nicht behandelt werden.

Im Kontext dieser Arbeit kann man die behandelten Daten auf zwei Kategorien einteilen: **Modelldaten** und **Beobachtungsdaten**.

Bei **Modelldaten** handelt es sich im Allgemeinen um Ergebnisse aus Modellrechnungen wie Klima- oder Wettermodellen. Diese sind also von Com-

putern berechnete und in diesem Sinne synthetische Daten, die retrospektive Analysen oder Vorhersagen über das beobachtbare Klima oder Wetter machen sollen. Man unterscheidet weiter zwischen **globalen** und **regionalen** Modellen, wobei die globalen Modelle Vorhersagen für die gesamte Erde machen sollen und regionale Modelle sich auf ein Teilgebiet beschränken. Bei regionalen Modellen dienen Ergebnisse aus globalen Modellläufen zur Definition der Randbedingungen, die für den regionalen Modelllauf benötigt werden. Bei den in dieser Arbeit behandelten HTAP-Daten handelt es sich um Modelldaten aus globalen (Chemie-)Transportmodellen.

Beobachtungsdaten hingegen stammen aus Messungen. Diese können durch einzelne Messstationen, Flugzeuge, Ballons, Satelliten oder andere Aufbauten aufgenommen worden sein. Im Gegensatz zu den Vorhersagen von Modellen können Beobachtungen also immer nur Vergangenheit und Gegenwart widerspiegeln. Während Modelldaten normalerweise ein ganzes Gebiet in Form eines (homogenen) Koordinatengitters repräsentieren, liegen Beobachtungsdaten oft nur für einzelne Punkte oder inhomogene Gitter vor.

2.4 Das HTAP-Datenarchiv

2.4.1 Überblick

Das zentrale HTAP-Archiv des IEK-8 hat derzeit ein Datenvolumen von circa 550 GB und setzt sich je etwa zur Hälfte aus Daten mit monatlicher (*monthly*) und stündlicher (*hourly*) Zeitauflösung zusammen. Während die Monatsdaten - je nach Modell - bis zu 48 vertikale Schichten vom Boden bis zur oberen Atmosphäre enthalten, beschränken sich die stündlichen Datensätze (mit Ausnahme von Stationsdaten, siehe unten) aus Platzgründen auf die Bodenschicht.

Die Mengen gasförmiger Substanzen werden in Form von (einheitenlosen) volumenbezogenen Mischungsverhältnissen (*Volume Mixing Ratio*, *VMR*), Aerosolmengen als massenbezogene Mischungsverhältnisse (*Mass Mixing Ratio*, *MMR*) und meteorologische Größen (Temperatur, Niederschlag, ...) in ihren entsprechenden physikalischen Einheiten ausgedrückt.

Insgesamt enthält das Archiv Ergebnisse von 29 verschiedenen Modellen, die zusammen 85 unterschiedliche Szenarien Simuliert haben, davon 43 in *ExperimentSet1* (SR*), 2 in *ExperimentSet2* (TP*), 6 in *ExperimentSet3* (ES*) und 24 in *ExperimentSet4* (10 FE*, 14 FC*). Zusätzlich wird zwischen 12 „Dateitypen“ unterschieden, die jeweils einen anderen Aspekt der Simulationsergebnisse beinhalten. Dabei sind aus verschiedenen Gründen nicht alle theoretisch möglichen Kombinaionen von Modell, Szenario und Typ erstellt worden. Die folgenden Dateitypen wurden bei der Experimentplanung definiert:

- **aerosolaod**: stündlich gemittelte optische Dicke der modellierten Aerosole bei 550 nm
- **aerosolm**: monatlich gemittelte MMRs von Aerosolen für verschiedene aerodynamische Durchmesser
- **budgetm**: monatlich gemittelte Produktions- und Verlustraten für Ozon sowie Oxidationsraten für Kohlenstoffmonoxid und Methan
- **depm**: monatlich gemittelte Trockendepositionsraten relevanter Substanzen
- **emim**: monatlich gemittelte Emissionsarten der Substanzen
- **jvalm**: monatlich gemittelte Photolyseraten für Stickstoffdioxid und Ozon
- **metm**: monatlich gemittelte meteorologische Daten (Temperatur, Niederschläge, ...)
- **sfc**: stündlich gemittelte VMRs von Ozon und Stickstoffdioxid in der Bodenschicht
- **sfc1**: monatlich gemittelte VMRs von Quecksilber und langlebigen organischen Schadstoffen in der Bodenschicht
- **tracerm**: monatlich gemittelte VMRs von Gasphasenkomponenten
- **vertprof**: täglich gemittelte Vertikalprofile für VMRs von Ozon, Kohlenstoffmonoxid, Stickstoffmonoxid und Stickstoffdioxid an vordefinierten Koordinaten („Stationen“)
- **wetdepm**: monatlich gemittelte Nassdepositionsraten relevanter Substanzen

2.4.2 Interne Koordinatenstruktur der Datensätze

Alle betrachteten Datensätze basieren auf den drei räumlichen Dimensionen *Längengrad*, *Breitengrad* und *Höhe* (meist in Druckeinheiten) sowie einer *Zeitdimension* bzw. einer Untermenge dieser Dimensionen.

Zur Vereinfachung und in Übereinstimmung mit den CF-Konventionen wird die Dimension des Längengrades als *X-Achse*, die des Breitengrades als *Y-Achse*, die der Höhe als *Z-Achse* und die der Zeit als *T-Achse* bezeichnet.

Dieser Abschnitt geht kurz auf die benutzten Darstellungen für die drei räumlichen Dimensionen ein. Die Benutzung der Zeitdimension entspricht den CF-Konventionen und wurde bereits in Abschnitt 2.2.2 erläutert.

Konzept der Gitterzellen

X- und Y-Achse bilden zusammen eine Struktur von horizontalen *Gitterzellen*, die den Ortsbezug der Daten herstellen. Die Schnittpunkte der X- und Y-Koordinaten definieren dabei jeweils den Mittelpunkt einer jeden Gitterzelle, wie in Abbildung 2.3 dargestellt ist.

Die Ausdehnung einer jeden Zelle kann alleine aus diesen Daten nicht zweifelsfrei ermittelt werden, auch wenn die Zellgrenzen bei den vorliegenden Daten im Regelfall auf halber Entfernung zwischen zwei Zellmittelpunkten liegen. Eine eindeutige Bestimmung der Zellausdehnung wird - sofern vorhanden - durch die in den CF-Konventionen definierten *Bounds*-Variablen ermöglicht (siehe Abschnitt 2.2.2).

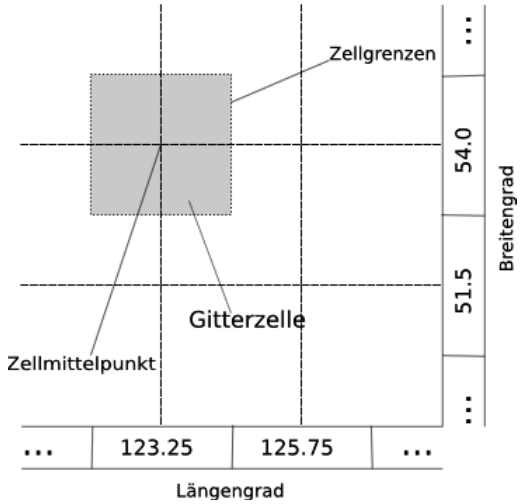

Abbildung 2.3: konzeptueller horizontaler Aufbau der Gitterzellen der HTAP-Daten mit beispielhaften Längen- und Breitengradachsen, die angegebenen Koordinatenwerte bezeichnen die Position des Mittelpunkts in der jeweiligen Dimension

Höhendarstellung

Während die Darstellung eines horizontalen Gitters mit Längen- und Breitengraden - abgesehen von den zahlreichen verschiedenen Koordinatenreferenzsystemen (CRS) - eine weit verbreitete Praxis ist, gibt es für die Darstellung der Höhenachse unterschiedliche Darstellungsformen mit verschiedenen Vor- und Nachteilen. Beispiele für unterschiedliche Darstellungen sind: Höhe über

mittlerem Meeresspiegel bzw. Geoid, Höhe über Grund, barometrische Höhe, etc.

Die Datensätze des HTAP-Datenarchivs verwenden im Regelfall die so genannte *hybrid sigma*-Darstellung, weil diese Vorteile bei der Modellierung von Transportprozessen in der Troposphäre bietet. Bei einigen Arten von Messungen hingegen wird eher die barometrische Höhe bevorzugt. Die Unterschiede zwischen diesen beiden Darstellungsformen werden im Folgenden kurz erläutert.

Bei der **barometrischen Höhendarstellung** nutzt man den mittleren Luftdruck in einer bestimmten atmosphärischen Höhe als Referenz. Dieser Luftdruck lässt sich mithilfe der barometrischen Höhenformel nährungsweise in eine Höhe über dem mittleren Meeresspiegel bzw. dem Geoid umrechnen. Diese Möglichkeit ist ein Vorteil der barometrischen Höhendarstellung. Im Bezug auf Computersimulationen von Nachteil ist, dass die Luftdruckschichten nicht bodenfolgend sind: Die unteren Schichten, die nahe über dem Geoid befinden, liegen auf Landmassen teils schon unter der Erdoberfläche, was zu Singularitäten (oder komplizierten Randbedingungen) bei der Simulation von Transportprozessen führt. Abbildung 2.4 (links) veranschaulicht dieses Problem.

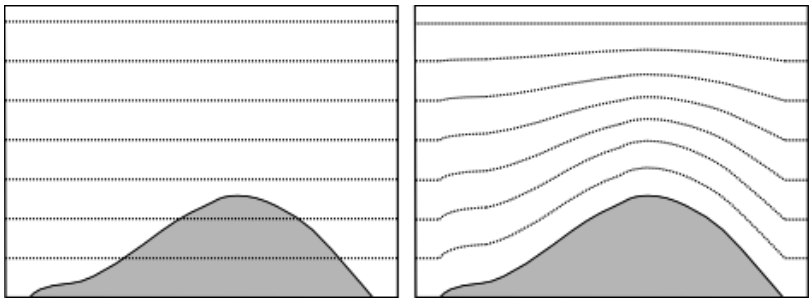

Abbildung 2.4: vereinfachte Illustration der barometrischen Höhendarstellung (links) und der *hybrid sigma*-Höhendarstellung (rechts) anhand eines vertikalen Schnittes der Erdoberfläche und darüberliegender Atmosphäre
links: alle Schichten sind horizontal zum Geoid ausgerichtet, die unteren beiden Druckschichten schneiden die Landmasse
rechts: die unteren Schichten folgen dem Bodenverlauf und schneiden die Landmasse nicht, mit zunehmender Höhe gehen sie in einen rein horizontalen Verlauf der barometrischen Höhendarstellung über
graue Fläche: Landmasse; **schwarz gepunktete Linien**: Schichten des Modells in der jeweiligen Druckdarstellung

Abhife schafft die Verwendung einer an den Höhenverlauf der Erdoberfläche - und somit an die Transportprozesse - angepassten Darstellung. Dazu eignet sich neben anderen Darstellungsformen auch die **hybrid sigma**-Darstellung. Es versucht dem Umstand Rechnung zu tragen, dass bodennahe Luftschichten durch das Profil der Erdoberfläche beeinflusst werden und dass dieser Einfluss mit zunehmender Höhe abnimmt. Die Konsequenz ist, dass eine modellierte Luftschicht an jedem Ort einer anderen barometrischen Höhe entsprechen kann und die Höhenachse des Datensatzes somit nicht auf dem Luftdruck als Koordinate aufbauen kann. Stattdessen wird jede Höhenschicht über zwei Faktoren a und b definiert. Aus diesen lässt sich - zusammen mit den ebenfalls im Datensatz enhaltenen Informationen zum Luftdruck auf der Erdoberfläche (p_s) und einem Referenzdruck (p_0) - der Luftdruck an jedem horizontalen Gitterpunkt (x, y) für jede Schicht (z) nach folgender Formel berechnen:

$$p_{x,y,z} = a_z \cdot p_0 + b_z \cdot p_{s_{x,y}}$$

Die beiden Faktoren a und b bestimmen also das Mischungsverhältnis der beiden Druckkomponenten so, dass in Bodennähe der p_s-Term dominiert, was in diesem Bereich für ein bodenfolgendes Verhalten sorgt, und mit steigender Höhe die Bedeutung des p_0-Terms zunimmt, was eine zunehmende Angleichung an die oben beschriebene Darstellung bewirkt. Abbildung 2.4 (rechts) veranschaulicht diesen Effekt.

Kapitel 3

Aufbereitung und Weiterverarbeitung

3.1 Standardkonformität von Datensätzen

Voraussetzung für eine automatisierte Verarbeitung von Datensätzen und Interoperabilität ist eine hinreichend weit ausgeprägte Standardkonformität. Bei den in dieser Arbeit betrachteten georeferenzierten Datensätzen bedeutet das konkret die Konformität mit den CF-Konventionen.

3.1.1 Problematik

Wie sich bei der Verarbeitung der Datensätze aus dem HTAP-Modellvergleich (siehe Abschnitt 1.2) gezeigt hat, ist die Einhaltung der CF-Konventionen (siehe Abschnitt 2.2.2) trotz vorheriger Absprache zwischen den Teilnehmern nicht trivial zu bewerkstelligen.

Auch wenn seine Nützlichkeit unbestritten bleibt, so ist entgegen aller Bemühungen der CF-Autoren doch ein relativ komplexes Regelwerk entstanden, das nur mit erheblichem Zeitaufwand und Erfahrung zu durchschauen ist. Zudem erwies sich der sonst nützliche restriktive Umgang bei Erweiterungen des *Standard Name Table* als hinderlich, da viele in den HTAP-Daten beschriebene Substanzen bis heute nicht aufgenommen wurden. Hinzu kommt, dass einige Klarstellungen erst nach Abgabe der Simulationsergebnisse in späteren CF-Versionen erfolgten.

Insgesamt wurden die CF-Konventionen in vielfältiger Weise fehlinterpretiert, was sich in falsch plazierten oder fehlenden Attributen an entscheidenden Stellen äußerte. Teilweise verantwortlich für diese Probleme ist vermutlich die erst nachträglich erfolgte Konvertierung in CF-konformes NetCDF, da keines

der benutzten Modelle von sich aus CF-konforme Ausgabedaten produziert. Mit CMOR[1] steht zwar ein Werkzeug bereit, das bei einer solchen Konvertierung helfen soll und bei HTAP auch genutzt wurde, jedoch erfordert es zur richtigen Konfiguration ebenfalls viel Detailwissen über CF.

3.1.2 Automatisierte Konformitätsprüfung

Wegen der zahlreichen Konformitätsprobleme beim HTAP Modellvergleich wurde beschlossen, ein Werkzeug zu entwickeln, mit dem Ergebnisse zukünftiger Modellvergleiche automatisch auf CF-Konformität überprüft werden können und erst bei fehlerfreien Metadaten eine Aufnahme in das Archiv erfolgt. Im Falle der Abweisung sollen dem Benutzer möglichst konkrete Rückmeldungen mit Hinweisen auf Korrekturmöglichkeiten gegeben werden.

existierende Software

Bei Beginn dieser Arbeit existierte bereits ein Prüfwerkzeug (`cfchecks.py`), das am *Hadley Centre for Climate Prediction and Research* des britischen *Met Office* entwickelt wurde und auch auf den offiziellen CF-Webseiten verlinkt ist[2].

Dieses war jedoch aus diversen Gründen nicht für die vorgesehenen Aufgaben geeignet. Zum einen waren die Konformitätstests unvollständig implementiert und hätten so im Bezug auf die HTAP-Datensätze zu viele Fehler übersehen. Zudem fehlten konkrete Testfälle (Unit-Tests) zur Überprüfung der Funktionsfähigkeit und die Fehlerprotokollausgabe war wenig kontrollierbar.

Eigenentwicklung

Aus oben genannten Gründen wurde deswegen beschlossen, ein Prüfwerkzeug zu entwickeln, das den eigenen Bedürfnissen besser Rechnung trägt:

- Benutzung von Python mit der NetCDF-Schnittstelle PyNIO[3] von UCAR/Unidata
- grösstmögliche Abdeckung der CF-Konventionen mit besonderem Fokus auf die HTAP-Datensätze
- besser kontrollierbare Fehlerprotokollstruktur → bessere Integrierbarkeit in verschiedene Anwendungen

[1] `http://www2-pcmdi.llnl.gov/cmor`
[2] `http://cf-pcmdi.llnl.gov/conformance/compliance-checker/`
[3] `http://www.pyngl.ucar.edu/Nio.shtml`

- Testfälle für alle möglichen Fehlermeldungen und Hinweise
- leichte nachträgliche Erweiterbarkeit bei neuen CF-Versionen

Ergebnis dieser Eigenentwicklung ist ein `CFchecker` getauftes Python-Paket, das auch im Quelltext frei erhältlich ist.[4] Allgemein nützliche Basisfunktionen im Zusammenhang mit CF wurden dabei in eine eigene Basisbibliothek (*CommonUtils*[5]) ausgelagert, da sie auch für weitere, CF-konforme Datensätze verarbeitende, Werkzeuge nützlich sind. Beispiele für die ausgelagerten Funktionalitäten sind das Auffinden von Koordinatenvariablen und ihre automatische Klassifizierung nach den CF-Kriterien, die Zuordnung von Koordinatenvariablen zu ihren `bounds`-Variablen und das Parsen von `cell_methods`-Werten.

Den Kern von `CFchecker` bilden Klassen, die die Tests für je eine CF-Version enthalten. Da Versionen aufeinander aufbauen, kann die Prüfklasse für eine neue Version jeweils von der vorherigen Version vererbt werden und es müssen lediglich einige wenige Tests hinzugefügt oder überschrieben werden. Der *Standard Name Table* wird direkt in Form der von der CF-Gemeinschaft zur Verfügung gestellten XML-Datei benutzt.[6] Der Zugriff auf diese wird über eine eigene Klasse abstrahiert, die die benötigte Funktionalität wie das Überprüfen der Existenz eines Namens und das Nachschlagen der mit dem Namen verbundenen Einheiten kapselt. Die Protokollfunktionalität ist ebenfalls in einer eigenen Klasse implementiert, um eine flexiblere Weiterverarbeitung für aufrufende Programme zu erlauben, als dies bei direkter Konsolenausgabe möglich wäre. Abbildung 6.1 illustriert den Aufbau anhand eines vereinfachten Klassendiagrammes.

Mit Hilfe von im Quellcodepaket enthaltenen Unit-Tests und entsprechenden Testdaten wird sichergestellt, dass Fehler auch nach Programmanpassungen korrekt erkannt werden. Dazu existiert mindestens je ein Testfall für jeden erzeugbaren Logeintrag und für CF-konforme Metadaten. Speziell im Hinblick auf spätere Erweiterungen erleichtert diese Vorgehensweise die Entwicklung erheblich, da die korrekte Funkion des bisherigen Codes schnell verifiziert werden kann.

Zur einfacheren Benutzung steht nach Außen hin eine Schnittstelle bereit, die je nach verlangter CF-Version die benötigten Tests aufruft und so eine einfache Einbindung in andere Anwendungen ermöglicht. Im `CFchecker`-Paket mitgeliefert wird ein einfaches Kommandozeilenwerkzeug, das diese Schnittstelle verwendet, um eine alleinstehend nutzbare Anwendung zu realisieren. Eine weitere derzeit in Entwicklung befindliche Anwendung für `CFchecker`

[4] `http://repositories.icg.kfa-juelich.de/hg/CFchecker/`
[5] `http://repositories.icg.kfa-juelich.de/hg/CommonUtils/`
[6] `http://cf-pcmdi.llnl.gov/documents/cf-standard-names`

ist ein Web-Frontend[7], das dem Nutzer die Überprüfung seiner Dateien ohne lokale Installation in Echtzeit erlaubt. Ein anderer Anwendungsfall wäre die Einbindung in Prozessierungswerkzeuge, wodurch sichergestellt werden könnte, dass nur CF-konforme Datensätze genutzt werden bzw. die Ausgabedaten auch wieder CF-konform sind.

Bei der Implementierung stellte sich auch heraus, dass sich - entgegen der Intention der Autoren - nicht alle Vorgaben von CF automatisiert überprüfen lassen, da in einigen Fällen erst aus dem nicht maschinenlesbaren Kontext klar wird, was bestimmte Variablen oder Attribute bedeuten. Somit lassen sich leider nicht alle Vorgaben vollständig automatisiert prüfen. Dennoch werden viele häufig gemachte Fehler aufgedeckt und so die Arbeitslast manueller Überprüfungen stark gesenkt. Auch sind manche Vorgaben ohne konkrete Erfahrung in speziellen Themenfeldern nur schwer vollständig zu durchblicken. `CFchecker` ist daher derzeit besonders auf Datensätze von Chemie-Transportmodellen abgestimmt. Mit der Zeit sind durch Einsatz mit verschiedenartigen Datensätzen durchaus noch Verbesserungen zu erwarten. Da der Quelltext offen zur Verfügung steht, können interessierte Parteien diese Verbesserungen prinzipiell auch selbst einbringen.

3.2 (Statistische) Operationen auf Datensätzen

Statistische Operationen wie die Bildung von Differenzen zwischen zwei Datensätzen oder der Mittelwertbildung über mehrere Datensätze sind ein oft verwendetes Mittel bei der Auswertung von Modellvergleichen, wie sie auch bei den hier behandelten HTAP-Daten vorgenommen wurden, beispielsweise in [Fiore u. a., 2009]. Um eine Vergleichbarkeit herzustellen, müssen die betrachteten Datensätze im Allgemeinen auf einem einheitlichen Gitter vorliegen. Dazu können Interpolationsoperatoren benutzt werden, wie sie in Abschnitt 3.3 beschrieben werden.

Es sind auch Operatoren denkbar, die nur auf einen einzelnen Datensatz angewandt werden. Ein Beispiel dafür ist die Mittelung über eine Dimension des Datensatzes oder die Berechung von atmosphärischen Säulendichten und globalen Massen von Substanzen.

Die hier beschriebenen Operatoren wurden - zusammen mit den nachfolgend in Abschnitt 3.3 beschriebenen Interpolationsoperatoren - zu einem modularen Python-Paket namens `CFdatatools`[8] zusammengefasst, deren Basisfunktionalität über ein enthaltenes einfaches Kommandozeilwerkzeug direkt

[7] `http://htap.icg.kfa-juelich.de:50080/upload`
[8] `http://repositories.icg.kfa-juelich.de/hg/CFdatatools/`

genutzt werden kann. Gegenüber bereits verfügbaren Operatoren mit NetCDF-Unterstützung (NCO[9], CDO[10]) wurde bei den `CFdatatools` besonderer Wert auf möglichst weitgehende CF-Konformität und die Nutzbarkeit als Bibliothek gelegt. Es hat sich jedoch gezeigt, dass CF-Konformität nicht bei allen Operationen unter allen Umständen komplett automatisiert erreicht werden kann.

3.2.1 Differenzen zwischen zwei Datensätzen

Zum Vergleich verschiedener Datensätze kann es hilfreich sein, die Differenz von Datenfeldern zu bilden. Eine solche Funktionalität stellt der Operator `diff` bereit. `diff` erzeugt aus den beiden Eingabedateien im CF-konformen NetCDF-Format eine Ausgabedatei, die im Regelfall ebenfalls CF-konform ist. Diese Konformität kann jedoch nicht unter allen Umständen garantiert werden. Zum Beispiel existiert derzeit in CF keine Möglichkeit, die erfolgte Differenzbildung im `standard_name`-Attribut auszudrücken. Somit kann das `standard_name`-Attribut nicht zur maschinenlesbaren Beschreibung benutzt werden, was eine automatische Erkennung dieser Operation in einem Datensatz schwierig macht.

Wie bereits in der Einleitung angedeutet ist die Grundvoraussetzung für die Anwendbarkeit der Differenzoperation das Vorliegen der zu vergleichenden Datensätze in einem einheitlichen Gitter. Die zu vergleichenden Datenfelder[11] müssen ebenfalls in beiden Datensätzen vorhanden sein. `diff` benutzt das in den CF-Konventionen festgelegte `units`-Attribut bei der Berechnung, um sicherzustellen, dass nur Datenfelder mit gleichen Einheiten miteinander verglichen werden. Dazu werden die in der ersten Datei benutzten Einheiten zu den Referenzeinheiten für die Ausgabedatei erklärt und - wenn möglich - die Datenfelder in der zweiten Datei vor der Berechnung entsprechend konvertiert, inkompatible Einheiten führen zu einem Fehler. Ist nur der Absolutwert der Abweichungen und nicht deren Vorzeichen von Interesse, so kann eine entsprechende Modifikation der Ausgabedaten über die implementierte Absolutwertoption erreicht werden. Eine relative Angabe der Abweichungen anstelle der standardmäßigen absoluten Angabe kann ebenfalls über eine Option gesteuert werden. Die zweite Eingabedatei stellt dabei die Referenzgröße dar.

Der diff-Operator wird im folgenden Abschnitt 3.3 benutzt, um die Auswirkungen von Interpolationsoperationen zu vergleichen. Beispiele sind in den Abbildungen 3.2 und 3.5 zu sehen.

[9] `http://nco.sourceforge.net/`
[10] `https://code.zmaw.de/projects/cdo`
[11] *Datenvariablen* im CF-NetCDF Vokabular

3.2.2 Mittelwerte mehrer Datensätze

Wie beispielsweise in [Fiore u. a., 2009] gezeigt wird, lässt sich durch die Mittelung eines Szenarios über mehrere Modelle oft ein besseres Abbild der Realität herstellen, als dies bei jedem individuellen Modellergebnis der Fall ist.

Das `CFdatatools`-Paket stellt für solche Berechnungen den `mean`-Operator zur Verfügung. Er verhält sich in seiner generellen Arbeitsweise analog zum oben beschriebenen `diff`-Operator: Gemeinsame Datenfelder werden anhand von übereinstimmenden NetCDF-Variablennamen erkannt und abweichende Einheiten zwischen den einzelnen Eingabedateien werden vor der Mittelung immer in die entsprechenden Einheiten der ersten Eingabedatei umgerechnet. Ebenfalls müssen auch für diese Operation alle Datensätze vor Anwendung der Mittelungsoperation auf einem einheitlichen bzw. vereinheitlichten Gitter vorliegen.

Im Normalfall werden alle Eingabedateien bei der Mittelung gleich gewichtet, optional unterstützt der Operator jedoch auch die explizite Verwendung von unterschiedlichen Gewichten für jeden Datensatz.

Darüber hinaus ist zu beachten, dass Mittelwerte nur für solche Gitterzellen bestimmt werden können, an denen alle Eingabefelder auch Werte enthalten: Fehlt der Wert einer Zelle in nur einem einzigen Eingabefeld, so kann an dieser Gitterstelle kein Mittelwert berechnet werden. Zu solchen Situationen kann es beispielsweise bei Benutzung der *barometrischen Höhendarstellung* in den bodennahen Schichten kommen. Eine in Abschnitt 3.3.2 diskutierte Möglichkeit zur Vermeidung dieser Situation ist die vertikale Extrapolation von Datenwerten.

3.2.3 Mittelung über Dimensionen eines Datensatzes

Für einige Anwendungsfälle kann auch die Mittelung entlang einer oder mehrerer Dimensionen eines einzelnen Datensatzes interessant sein. So könnte im Falle der HTAP-Datensätze beispielsweise aus den zwölf gespeicherten Monatswerten ein Jahresmittel berechnet werden. Auch die mittlere Verteilung von Substanzen entlang eines Längen- oder Breitengrades kann möglicherweise von Interesse sein.

Der `dimaverage`-Operator stellt dem Benutzer diese Möglichkeiten zur Verfügung und erlaubt auch die gleichzeitige Mittelung über mehrere Dimensionen.

3.2.4 Berechnung von atmosphärischen Säulendichten

Die atmosphärische Säulendichte ist ein Maß für die in einer (eindimensionalen) Luftsäule senkrecht zur Erdoberfläche vorhandenen Moleküle bzw. deren Masse.

Diese Größe ist beispielsweise für den Vergleich mit durch *Nadirbeobachtung* gewonnenen Satellitendaten relevant. *Nadir* bezeichnet im Kontext der Erdbeobachtung eine Richtung, die senkrecht auf die Erdoberfläche zeigt. Typische Einheiten für die Säulendichte sind Moleküle/cm^2 (Satellitendaten) und kg/m^2 (CF-Konventionen). Dieser Unterschied ergibt sich auch aus der unterschiedlichen Art der Messung: Bei Satellitenmessung lässt sich besser die Teilchenzahl als deren Gesamtmasse bestimmen. Andererseits werden beispielsweise bei der Modellierung Substanzen eher durch ihr massen- oder volumenbezogenes Mischungsverhältnis in der Luft ausgedrückt, woraus sich leichter eine massenbezogene Säulendichte berechnen lässt. Ausgehend von dem volumenbezogenen Mischungsverhältnis des Stoffes gilt für die massenbezogene Säulendichte d_m:

$$d_m = \sum_{i=1}^{N}(\mathrm{mmr}_{s,i} \cdot \frac{p_{diff,i}}{g}) \text{ mit } \mathrm{mmr}_{s,i} = \mathrm{vmr}_{s,i} \cdot \frac{M_s}{M_{air}}$$

mit vmr_s: volumenbezogenes Mischungsverhältnis des Stoffes
mmr_s: massenbezogenes Mischungsverhältnis des Stoffes
M_s: molare Masse des Stoffes
M_{air}: molare Masse der Luft
p_{diff}: Druckdifferenz zwischen oberer und unterer Zellgrenze
g: Erdbeschleunigung
N: Anzahl der Höhenschichten

Zur Berechnung der auf die Molekülanzahl bezogene Säulendichte d_n ist hingegen zusätzlich die allgemeine Gasgleichung ($p \cdot V = n \cdot R \cdot T$) erforderlich, die die Lufttemperatur als weitere Variable in die Berechnung einführt. d_n kann somit nur berechnet werden, wenn neben den Mischungsverhältnissen auch die Temperaturfelder des Modelllaufes vorliegen, was nicht für alle HTAP-Datensätze der Fall ist. Es ergibt sich dann für d_n:

$$d_n = \sum_{i=1}^{N}(\mathrm{vmr}_{s,i} \cdot \frac{N_A \cdot p_{diff,i}}{R \cdot T_i})$$

mit vmr_s: volumenbezogenes Mischungsverhältnis des Stoffes
N_A: Avogadro-Konstante
p_{diff}: Druckdifferenz zwischen oberer und unterer Zellgrenze
R: allgemeine Gaskonstante

T: mittlere Temperatur der Zelle
N: Anzahl der Höhenschichten

Der `colint`-Operator implementiert die Berechnung der Säulendichten in beiden Bezugseinheiten. Da sich im Falle der HTAP-Daten die Temperaturfelder und die Mischungsverhältnisse in verschiedenen NetCDF-Dateien befinden, bietet das Modul die Möglichkeit zur Verwendung von zwei Eingabedateien zur Berechnung. Dabei werden alle benötigten Daten ausser den Mischungsverhältnissen aus der ersten Datei ausgelesen und die Mischungsverhältnisse der betrachteten Stoffe aus der zweiten Datei. Der in die Berechnung einzubeziehende Höhenbereich kann optional nach oben gekappt werden. Dies vereinfacht den Vergleich von Quelldaten mit unterschiedlich weit nach oben ausgeprägten Schichten und erlaubt darüber hinaus beispielsweise die gezielte Berechnung des troposphärischen Anteils von Datensätzen, die auch eine stratosphärische Komponente beinhalten.

3.2.5 Berechnung von globalen Massen

Die globale Masse eines Spurenstoffes wird in der Fachliteratur auch mit *burden* bezeichnet. Wie der Name schon sagt, handelt es sich dabei um die zu einem bestimmten Zeitpunkt insgesamt in der Erdatmosphäre (oder Troposphäre) vorhandenen Masse des betrachteten Spurenstoffes.

Die Berechnung der globalen Masse eines Stoffes erfolgt ähnlich der Berechnung der massenbezogenen Säulendichten. Vor der Summation der einzelnen Zellwerte müssen diese lediglich noch mit der Zellfläche gewichtet werden, um die Gesamtmasse des Stoffes pro Zelle zu erhalten. Diese Massen können dann zur globalen Masse aufsummiert werden. Im Gegensatz zu der Säulendichte wird hierbei auch über die horizontalen Dimensionen integriert, so dass im Regelfall nur noch ein eindimensionales, zeitabhängiges Datenfeld übrig bleibt. Für die globale Masse eines Stoffes m_g ergibt sich:

$$m_g = \sum_{x=1}^{M} \sum_{y=1}^{N} (d_{m_{x,y}} \cdot A_{x,y})$$

mit M: Anzahl der Gitterzellen in Längengrad-Richtung
N: Anzahl der Gitterzellen in Breitengrad-Richtung
d_m: massenbezogene Säulendichte
A: horizontale Zellfläche

Die Zellgrößen können dabei entweder direkt im Datensatz enthalten sein, wie es bei den HTAP-Daten in der Regel der Fall ist, oder sie lassen sich anhand der horizontalen Zellgrenzen berechnen. Der hier implementierte `globmass`-Operator bevorzugt dabei die Berechnung anhand der Zellgrenzen, da so eine

größere Kontrolle über einzelne Parameter möglich ist. So ist in diesem Fall zum Beispiel das benutzte Erdmodell und dessen Abmessungen (z.B. Erdradius) genau bekannt, was bessere Vergleichbarkeit zwischen Datensätzen verschiedener Herkunft bietet. Die Erde wird sowohl im `globmass`-Operator als auch in den HTAP-Modellen durch eine Kugel approximiert, was die Berechnung der Flächen vereinfacht und für den angestrebten Einsatzzweck ausreichende Genauigkeit bietet.

Zusätzlich erlaubt die Nutzung der Zellgrenzen auch eine exaktere Eingrenzung eines zu betrachtenden Ausschnittes, da die Grenzen vor Berechnung der Zellflächen entsprechend an den Ausschnitt angepasst werden können, was bei Verwendung von vorberechneten Zellflächen nur schwer möglich ist.

Eine Eingrenzung des betrachteten Bereiches erlaubt den genaueren Vergleich von Datensätzen, die verschieden große Bereiche der Erde abdecken. Dies gilt auch für den Vergleich von Original- und interpolierten Datensätzen des gleichen Modells, da die Interpolation immer mindestens in Richtung der Breitengrade zu einer Einschränkung des abgedeckten Bereiches führt (siehe auch Abschnitt 3.3.1). Eine Eingrenzung des Höhenbereiches wie beim `colint`-Operator wird ebenfalls unterstützt (siehe Abschnitt 3.2.4).

3.3 Interpolation von Datensätzen

Innerhalb der hier behandelten HTAP-Datensätze unterscheiden sich - vor allem zwischen den unterschiedlichen Modellen - die verwendeten Koordinatengitterabstände sowohl in horizontaler wie auch in vertikaler Richtung. Um jedoch Vergleichbarkeit zwischen verschiedenen Datensätzen bzw. Modellen herzustellen, müssen die Datenfelder auf einheitliche Gitterkoordinaten abgebildet werden. Konkret von Interesse sind beispielsweise Berechnungen von Mittelwerten oder Differenzen verschiedener Modellergebnisse (siehe auch Abschnitt 3.2). Eine solche Vereinheitlichung kann durch Interpolation der Daten erreicht werden. Dabei stellt sich auch die Frage, inwieweit Interpolationen wissenschaftlich interessante Erhaltungsgrößen beeinflussen. Solche Größen sind beispielsweise die globale Masse eines Spurenstoffes und die atmosphärische Säulendichte.

Im Idealfall sollte ein Interpolationsoperator minimalen Einfluss auf solche Erhaltungsgrößen ausüben. Im Folgenden werden die in der vorliegenden Arbeit entwickelten Interpolationsoperatoren vorgestellt und kurz auf ihre Robustheit im Bezug auf erwähnten wissenschaftlichen Größen hin untersucht.

Alle hier vorgestellten Interpolationsoperatoren sind im `CFdatatools`-Paket (siehe auch Abschnitt 3.2) enthalten.

Die praktischen Auswirkungen der implementierten Interpolationsopera-

toren auf die Säulendichte und die globale Masse wurden anhand von Datensätzen aus dem SR1-Szenario des HTAP-Datenarchivs exemplarisch an drei Substanzen aus 13 verschiedenen Modellen untersucht. Als zu untersuchende Substanzen wurden Ozon (O_3), Kohlenstoffmonoxid (CO) und Stickstoffdioxid (NO_2) ausgewählt, da diese von den meisten Modellen berechnet wurden und unterschiedliche charakteristische Verteilungen aufweisen: Das Maximum der Ozonkonzentration liegt in der Stratosphäre, wo die Höhenauflösung bei vielen Modellen nur relativ schlecht ist. CO und NO_2 besitzen Maxima in Bodennähe, wobei NO_2 wegen seiner geringeren chemischen Lebensdauer auch starke horizontale und vertikale Gradienten aufweist. Zusätzlich sind auch beträchtliche Mengen NO_2 in der Stratohphäre vorhanden, wo der Konzentrationsverlauf in etwa dem von O_3 entspricht.

3.3.1 Horizontale Interpolation

In der horizontalen Dimension basieren alle HTAP-Datensätze auf einem Längengrad-Breitengrad-Gitter (siehe Abschnitt 2.4.2). Diese weisen jedoch unterschiedliche Koordinatenursprünge und Abstände auf. Zudem kommen neben regulären Gittern auch so genannte gaußsche Gitter vor, die in Richtung der Breitengrade irregulär sind. Diese irregulären Breitengrade sind ein Resultat der Verwendung von spektralen Koordinaten (Kugelflächenfunktionen, engl. *spherical harmonics*) und des Verzichts auf eine verfrühte Interpolation auf ein reguläres Gitter. Die Abweichungen von einem regulären Gitter sind jedoch eher gering. Detailiertere Informationen zur Benutzung von Kugelflächenfunktionen und Spektraltransformationen in Transportmodellen finden sich in [Stepaniak, 2004].

Bilileare Interpolation

Als einfachster Interpolationsansatz bietet sich eine bilineare Interpolation des horizontalen Gitters auf das gewünschte Zielgitter an. Diese kann unabhängig in X- und Y-Richtung erfolgen und lässt sich in wenigen Schritten implementieren, was im Modul `hintgrid` des `CFdatatools`-Paketes realisiert wurde.

Da bei dieser Methode alle Höhenschichten gleich behandelt werden und diese nicht voneinander abhängig sind, lässt sich die Interpolation auch parallel auf alle Schichten anwenden, was die Bearbeitungsgeschwindigkeit bei entsprechender paralleler Programmierung verbessern kann.

Problematik der Zellgrenzen

Das implementierte Interpolationsmodul geht davon aus, dass die horizontalen Gitterpunkte die Mittelpunkte einer jeden Zelle darstellen, was bei den HTAP-Datensätzen im Allgemeinen der Fall ist. Die Interpolation der Werte an den neuen Gitterpunkten erfolgt dann durch lineare Gewichtung des Abstands dieser Zellmittelpunkte voneinander.

Um den interpolierten Datensatz zu komplettieren und weitere Auswertungen möglich zu machen, müssen auch neue Zellgrenzen generiert werden. Hierbei wird davon ausgegangen, dass die Koordinaten des Zielgitters ebenfalls die Zellmittelpunkte beschreiben. Handelt es sich bei dem Zielgitter um ein reguläres Gitter, wovon hier auszugehen ist, dann können die Zellgrenzen entsprechend so generiert werden, dass sie auch im Zieldatensatz diese Bedingung erfüllen. Dazu werden jeweils die Mittelpunkte zwischen zwei benachbarten Koordinaten zur Zellgrenze erklärt. Bei periodischen Achsen können auf diese Weise auch die Grenzen der sich am „Rand“ befindlichen Zellen bestimmt werden, da diese sich durch die periodische Randbedingung der Achse in gleicher Weise behandeln lassen. Ein solcher Fall liegt beispielsweise vor, wenn der betrachtete Datenausschnitt in Längengradrichtung die gesamte Erde umfasst. In Richtung der Breitengrade ist diese Herangehensweise normalerweise nicht möglich, so dass die äußeren Grenzen der Randzellen anderweitig approximiert werden müssen. Dazu werden diese Grenzen so gewählt, dass der Abstand zwischen gegebenem Koordinatenpunkt und Außenrand identisch mit dem Abstand zum Innenrand ist. Der gegebene Koordinatenpunkt liegt somit auch hier im Mittelpunkt der Zelle. Umfasst die Längengradachse des betrachteten Ausschnittes nicht die gesamte Erde, so findet dieses Verfahren dort ebenfalls Anwendung.

Problematisch wird dieser Ansatz bei stark irregulären Gittern, da die gegebenen Koordinatenpunkte dann nicht mehr im - oder nahe am - durch die Grenzen definierten Mittelpunkt liegen.

Untersuchung der Auswirklungen auf Erhaltungsgrößen

Allgemein lässt sich feststellen, dass die Spannweite der Datenwerte zwischen minimalem und maximalem Wert durch die lineare Interpolation tendentiell reduziert wird. Dies liegt daran, dass durch die lineare Interpolation keine neuen Extremwerte erzeugt werden können, die außerhalb des ursprünglichen Wertebereichs liegen: Wird ein Datenpunkt in der Nähe eines Extremwertes interpoliert, so wird dieser interpolierte Datenpunkt automatisch näher am Mittelwert liegen und somit den Kurvenverlauf abflachen.

Als Zielgitter wurden zur Untersuchung reguläre Gitter mit Gitterweiten von 1, $2,5$, 5, 10 und 15 Grad gewählt. Da die Gitter der HTAP-Datensätze Abstände zwischen $1,8$ und 3 Grad haben, testet diese Auswahl der Zielgitter das Verhalten der betrachteten Größen sowohl bei Verfeinerung wie auch bei Vergröberung des Gitters.

Der Vergleich des Verhaltens der globalen Massen gestaltet sich dabei einfacher, da die zu vergleichenden Datenfelder nur noch eine Zeitdimension besitzen und alle räumlichen Dimensionen durch die Integration entfallen.

Bei Betrachtung der relativen Abweichungen zeigt sich, dass eine starke Vergröberung des Gitters auf 10 oder 15 Grad je nach Modell zu Abweichungen bis zu etwa 5% bei NO_2 führt. Im Gegensatz dazu schwanken die Ozonwerte nur um maximal 0,6% und CO nur unwesentlich stärker. Diese Beobachtung lässt sich durch die vergleichsweise starken Gradienten von NO_2-Feldern erklären, bei denen die lineare Interpolation auf göberen Gittern keinen hinreichend guten Schätzert liefert. Bei Ozon und CO tritt dieser Effekt aufgrund der weniger ausgeprägten Gradienten nur abgeschwächt auf. Bei Vergröberung auf Gitterweiten von 5 Grad liegen die Abweichungen um etwa eine Größenordnung unter denen bei 15 Grad. Abbildung 3.1 zeigt die relativen Abweichungen für alle getesteten Gitterweiten und Substanzen beispielhaft am MOZART-Modell.

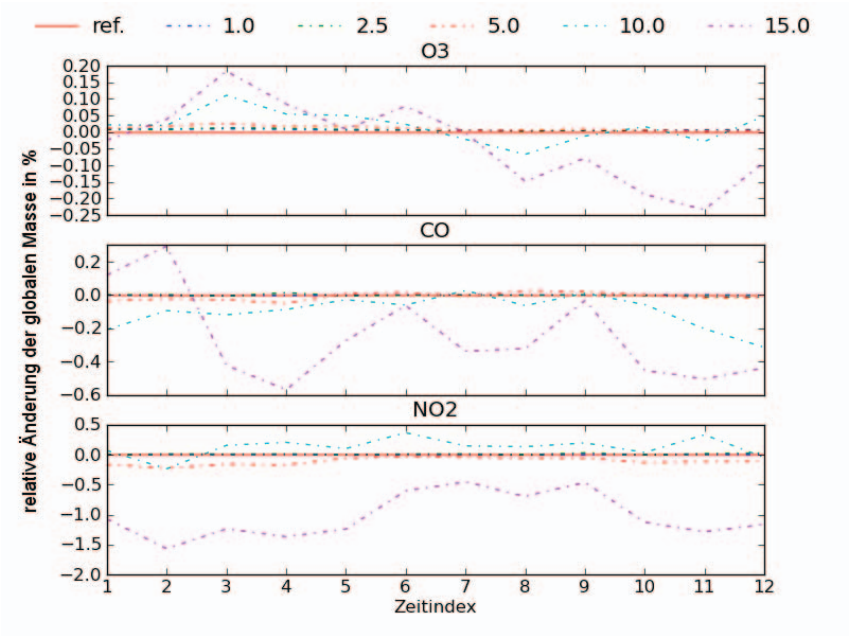

Abbildung 3.1: relative Abweichungen der globalen Masse bei verschiedenen interpolierten Gitterweiten zwischen 1 und 15 Grad am Beispiel des MOZART-Modells: deutlich zu sehen sind die im Verhältnis zu den anderen Gitterweiten großen Abweichungen bei 10 und 15 Grad Gitterabstand
durchgehende Linie: Referenz (0% Abweichung), gestrichelte Linien: Abweichungen bei gegebenem horizontalem Gitterabstand

Im Falle von Verfeinerung (oder nur geringfügiger Vergröberung) des Gitters liegen die Abweichungen deutlich niedriger - in den meisten Fällen unter 0,1% und in vielen Fällen sogar um noch eine Größenordnung darunter. Im Falle des MOZART-Modells ist zu beachten, dass auch die Interpolation auf $2,5$ Grad Gitterweite noch eine Vergröberung darstellt, da das Quellgitter eine Weite von etwa $1,9$ Grad besitzt. Die Abweichungen bei 1 Grad liegen bei allen drei Substanzen unter 0,01% und verdoppeln sich im Extremfall bei Vergröberung auf 2,5 Grad. Dies deutet darauf hin, dass eine Verfeinerung des Gitters - genau wie nur geringfügige Vergröberung - keinen nennenswerten Einfluss auf die globale Masse hat und dass diese Vorgehensweise somit beim Vergleich zwischen Modellen benutzt werden kann.

Ein praktischer Vergleich der Säulendichten ist komplizierter, da das zu

vergleichende Datenfeld weiterhin von beiden horizontale Raumdimenionen abhängt und seine Form der des horizontalen Gitters des jeweiligen Quelldatensatzes entspricht. Die auf zwei verschiedenen horizontalen Gittern berechneten Säulendichten lassen sich deswegen nicht direkt - beispielsweise durch Differenzbildung der Datenfelder - miteinander vergleichen. Somit entsteht bei der Evaluation der Benutzbarkeit der horizontalen Interpolation genau das Problem, das sie lösen soll. Besonders interessant wäre das Verhalten der Interpolation in Bereichen mit starken Gradienten, wie sie beispielsweise an Gebirgen auftreten.

Allgemein lässt sich feststellen, dass eine Verfeinerung des Gitters durch weitere Unterteilung der einzelnen Zellen nicht auf die Säulendichte an den vergleichbaren Gitterpunkten auswirken kann. Dies liegt daran, dass die Interpolation diese Datenwerte unverändert in das Zielgitter übernimmt und keine Abhängigkeit von der veränderten Zellfläche vorliegt. Somit ist die Datenbasis an solchen Gitterstellen in den zu vergleichenden Datensätzen identisch. Bei allen nicht auch im Quellgitter vorhandenen Koordinaten ergibt sich wie bereits oben beschrieben tendentiell eine Verringerung der Amplitude der Datenwerte gegenüber den Ursprungswerten.

Statt einem direkten Vergleich von Quelldatensatz und interpoliertem Datensatz kann über die obigen Überlegungen hinaus zusätzlich zumindest überprüft werden, ob Interpolation und Berechnung der Säulendichte invariant im Bezug auf ihre Anwendungsreihenfolge sind. Dieser Ansatz gleicht dem visuellen Vergleich der Konturplots von Original- und interpoliertem Datensatz insofern, als dass auch beim Konturplot in der Regel zwischen den einzelnen Datenpunkten (linear) interpoliert wird. Damit liegen zwei - zum Beispiel durch Differenzbildung - direkt miteinander vergleichbare Datensätze vor, deren Differenzen hauptsächlich auf den Interpolationsoperator zurückzuführen sind, da die Berechnung der Säulendichten selbst nur sehr geringen (numerischen) Fehlern unterworfen sein sollte. Eine gute Übereinstimmung dieser beiden Datensätze kann zwar nicht abschließend die Benutzbarkeit des Interpolationsansatzes nachweisen, sie kann aber zumindest als guter Anhaltspunkt dienen.

Abbildung 3.2 zeigt eine mit diesem Ansatz berechnete relative Differenz für die massenbezogenen Säulendichten von Ozon im MOZART-Modell bei einem Grad Gitterweite. Dabei wurde das nach der Berechnung der Säulendichten interpolierte Feld von dem zuerst auf das Zielgitter interpolierten Feld subtrahiert. Die Übereinstimmung beider Ansätze ist im Allgemeinen hoch, es gibt jedoch sichtbare Unterschiede im Bereich von großen Gebirgszügen wie beispielsweise dem Himalaya und den Anden sowie in der Antarktis und in Grönland. Ein solches Verhalten entspricht weitgehend den Erwartungen und zeigt sich bei allen betrachteten Gitterweiten in ähnlichem Umfang. Die größten relativen

Abweichungen zwischen den beiden Ansätzen betragen bei allen betrachteten Modellen etwa 1%. Dieses Ergebnis ist auch mit den Ergebnissen aus der Betrachtung des Verhaltens der globalen Massen vereinbar, die als gewichtete Summe der Säulendichten eine etwa zwei bis drei Größenordnungen geringere maximale Abweichung aufweisen.

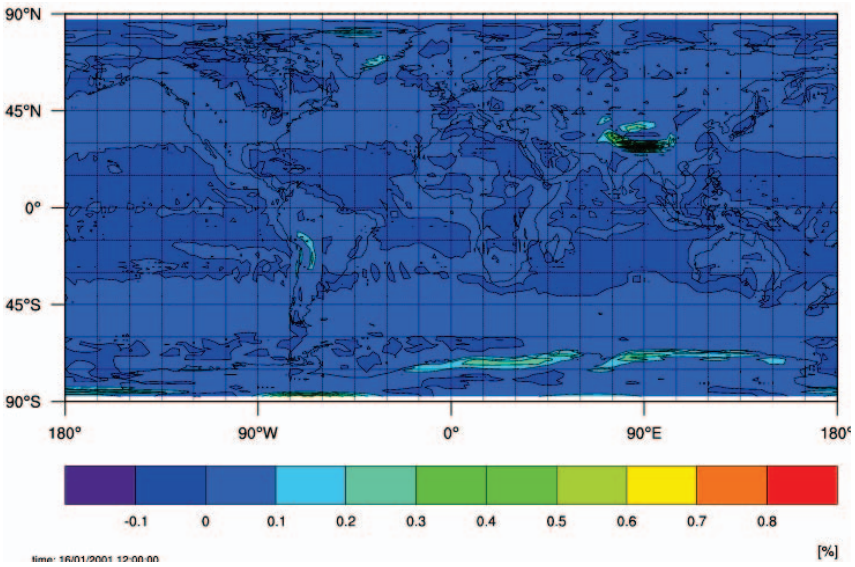

Abbildung 3.2: relative Differenz von auf zwei unterschiedlichen Wegen errechneten massenbezogenen Säulendichten für Ozon im MOZART-Modell mit einem Grad Gitterauflösung
zu erkennen sind die Abweichungen im Bereich von gößeren Gebirgsformationen wie dem Himalaya und den Anden sowie in der Antarktis und in Grönland.

Diese Beobachtung gilt nicht nur für Ozon, sondern erstreckt sich auch auf die beiden anderen Untersuchten Substanzen CO und NO_2. Bei diesen liegen die maximalen Abweichungen der Säulendichten in einem vergleichbaren Bereich. Die bei Berechnung der anzahlbezogenen Säulendichten nötige Interpolation der Temperaturfelder stellt eine zusätzliche potentielle Unsicherheit dar, die sich bei den durchgeführten Betrachtungen jedoch nicht signifikant auswirkten. Somit haben die Aussagen für die massenbezogenen Säulendichten auch in diesem Fall Gültigkeit.

3.3.2 Vertikale Interpolation

Bei der vertikalen Interpolation liegt bei den hier behandelten Datensätzen eine andere Ausgangssituation vor, da die Darstellungsform der Höhenachse unter Umständen gleichzeitig in eine andere Darstellungsform überführt werden soll. Ein konkreter Anwendungsfall ist die Konvertierung von *sigma hybrid*-Darstellung in die *barometrische* Darstellung, wie sie zum Vergleich der HTAP-Daten mit Daten aus anderen Quellen bevorzugt wird. Dieser Abschnitt beschäftigt sich deswegen konkret mit diesem Problem und geht nicht weiter auf die reine Interpolation innerhalb der *barometrischen* Darstellung ein. Das Vorgehen unterscheidet sich dabei jedoch - abgesehen von der bei der Konvertierung der Darstellungsformen nötigen Vorberechnungen - nicht.

Zu lösende Probleme

Im Bezug auf die betrachteten Erhaltungsgrößen ergeben sich im Gegensatz zur horizontalen Interpolation größere Schwierigkeiten bei der Bestimmung von sinnvollen vertikalen Zellgrenzen. Dies liegt vor allem daran, dass die Gitterabstände in vertikaler Richtung in der Regel irregulär sind. Die *sigma hybrid*-Darstellung verschärft diese Problematik eher noch, da für jede horizontale Gitterstelle andere Druckvektoren vorliegen können (vergleiche Abbildung 2.4). Ohne Zellgrenzen könen wegen der dann fehlenden Höheninformation jedoch weder globale Masse noch die atmosphärischen Säulendichten berechnet werden. Dies ist ein bis jetzt nicht allgemein gelöstes Problem, zu dem im Folgenden einige einfache Überlegungen und Abschätzungen angestellt und ausprobiert werden sollen.

Ein weiteres Problem ist das Finden von sinnvollen Datenwerten ober- bzw. unterhalb der im Modell enthaltenen Höhenschichten (Extrapolation). Im einfachsten Fall könnte auch diese linear erfolgen, eine andere Möglichkeit wäre die Benutzung von geeigneten Splines mit sinnvoll gewählten Randbedingungen. Im Folgenden wird nur die einfache lineare Extrapolation auf ihre Benutzbarkeit hin untersucht.

Eine Extrapolation unter die Bodenschicht macht physikalisch in der Regel keinen Sinn, da an diesen Stellen gerade keine Atmosphäre vorhanden ist, die die betrachteten Stoffe enthalten kann. Da jedoch die verwendeten Geländemodelle von Computermodell zu Computermodell variieren können, kann eine Extrapolation - beispielsweise vor einer Mittelwertbildung mehrerer Modelle - trotzdem in bestimmtem Maße hilfreich sein, um eine möglichst gute Abdeckung zu erzielen. Dies gilt jedoch nur unter der Bedingung, dass die Extrapolation nicht zu inakzeptabel großen Verfälschungen von relevanten Größen wie Säulendichte oder globaler Masse führt.

Linearer Interpolationsansatz

Die Python-Implementierung des linearen Interpolationsoperators basiert auf dem Prinzip der Interpolationfunktion `vinth2p`[12] aus der Bibliothek der *NCAR Command Language*[13] (NCL) und wurde entsprechend ebenfalls `vinth2p` genannt.

Da die Quelldaten in *sigma-hybrid*-Darstellung vorliegen, muss zunächst der tatsächliche Druck an jedem horizontalen Gitterpunkt nach der in Abschnitt 2.4.2 beschriebenen Vorschrift berechnet werden. Anschließend lassen sich dann wiederum durch lineare Interpolation die Datenwerte für die gewünschten Drücke bestimmen. Da hierbei jeder horizontale Gitterpunkt potentiell eine eigene Druckachse besitzt, ist die effiziente Berechnung der interpolierten Datenwerte schwieriger zu erreichen als bei der barometrischen Höhendarstellung, bei der alle horizontalen Gitterpunkte eine gemeinsame Druckachse besitzen und Interpolationsgewichte deswegen nur einmal berechnet werden müssen. Drücke über bzw. unter den verfügbaren Schichten werden dabei auf Basis der letzten beiden bekannten Datenpunkte linear extrapoliert. Die Extrapolation nach oben (Drücke kleiner als der kleinste vorhandene Druck) wird dabei in der Version aus der NCL-Bibliothek automatisch durchgeführt, die Extrapolation nach unten kann optional erfolgen. In der hier implementierten Version wurde die Extrapolation nach oben ebenfalls optional gemacht, um das Verhalten genauer steuern zu können.

Eine Bestimmung von vertikalen Zellgrenzen ist in der als Grundlage dienenden NCL-Bibliotheksfunktion nicht vorgesehen. Eine Untersuchung ihrer Auswirkungen auf die relevanten Erhaltungsgrößen ist deswegen alleine aus diesem Ansatz heraus nicht möglich.

Bestimmung von Zellgrenzen

Um zu einer Abschätzung der Auswirkungen der Interpolation auf die Berechnung der globalen Masse und Säulendichten von Spurenstoffen zu gelangen, müssen zusätzlich Zellgrenzen bestimmt werden. Die einfachste Methode besteht in der Anwendung des schon für die Zellgrenzen in horizontaler Richtung (vergleiche Abschnitt 3.3.1) genutzten Ansatzes, bei dem die Grenzen durch das arithmetische Mittel je zweier Druckkoordinaten gegeben sind. Ein Problem besteht dabei wie im horizontalen Fall in der Wahl der obersten und der untersten Grenze, die nicht mehr als Mittelpunkte bestimmt werden können. Hierbei kann ebenfalls die gleiche Vorgehensweise ausprobiert werden, so dass die Koordinatenwerte im Mittelpunkt der neu definierten Zelle liegen.

[12] `http://www.ncl.ucar.edu/Document/Functions/Built-in/vinth2p.shtml`
[13] `http://www.ncl.ucar.edu/index.shtml`

Als Randbedingungen könnten dabei nach oben (vereinfachend) 0 hPa und nach unten der Normaldruck von 1013, 25 hPa angenommen werden.

Untersuchung der Auswirklungen auf Erhaltungsgrößen

Bei der Untersuchung der Auswirklungen der linearen vertikalen Interpolation auf globale Masse und Säulendichte sind zwei verschiedene Fragen von Interesse:

1. Lassen sich die Interpolationsstufen und Zellgrenzen so wählen, dass globale Masse und Säulendichte in hinreichendem Maße erhalten bleiben?

2. Wie stark wirkt sich eine Extrapolation über die im Quelldatensatz enthaltenen Schichten auf die Erhaltungsgrößen aus?

Grundsätzlich liegt bereits die Vermutung nahe, dass eine Extrapolation der Daten sowohl Säulendichte als auch globale Masse erhöhen muss, da sich die zusätzliche Extrapolation nicht auf die interpolierten Datenwerte und noch zusätzliche Gitterzellen in die Summe mit einfließen.

Es ist auch anzumerken, dass selbst eine gute Übereinstimmung von globaler Masse und Säulendichte nur als notwendige Bedingung für die Nutzbarkeit dieses Interpolationsansatzes dienen kann. Desweiteren müsste untersucht werden, inwieweit die höhenabhängige Verteilung der Stoffe erhalten bleibt. Ein solcher Vergleich stellt sich jedoch wiederum als schwierig heraus, da die unterschiedlichen Quell- und Zielfelder sich nicht direkt miteinander vergleichen lassen. Dieses Problem ähnelt dem bei der Evaluation der horizontalen Interpolation (Abschnitt 3.3.1) aufgetretenen Problem beim Vergleich der Säulendichten. Die dort zum Einsatz gekommene Abschätzung über Interpolation der Säulendichten des Quellgitters auf das Zielgitter lässt sich jedoch nicht einfach auf den hier vorliegenden Fall übertragen. Eine Untersuchung dieser Problemstellung ist nicht mehr Teil dieser Arbeit und erfordert weitere Betrachtungen, falls sich der hier vorgestellte Ansatz als aussichtsreich erweist.

Für die weiteren Untersuchungen wurden die untersuchten Modelle auf zwischen 10 und 160 gleichverteilte Druckstufen im Bereich von 0 bis 1000 hPa vertikal interpoliert. Die Zellgrenzen wurden dabei nach dem oben beschriebenen Verfahren bestimmt.

Zur Illustration wurde wiederum das MOZART-Modell ausgewählt. Da sich alle untersuchten Modelle jedoch ähnlich verhalten, können die dargestellten Ergebnisse qualitativ größtenteils auch auf die anderen Modelle übertragen werden.

Auswirkungen auf die globale Masse

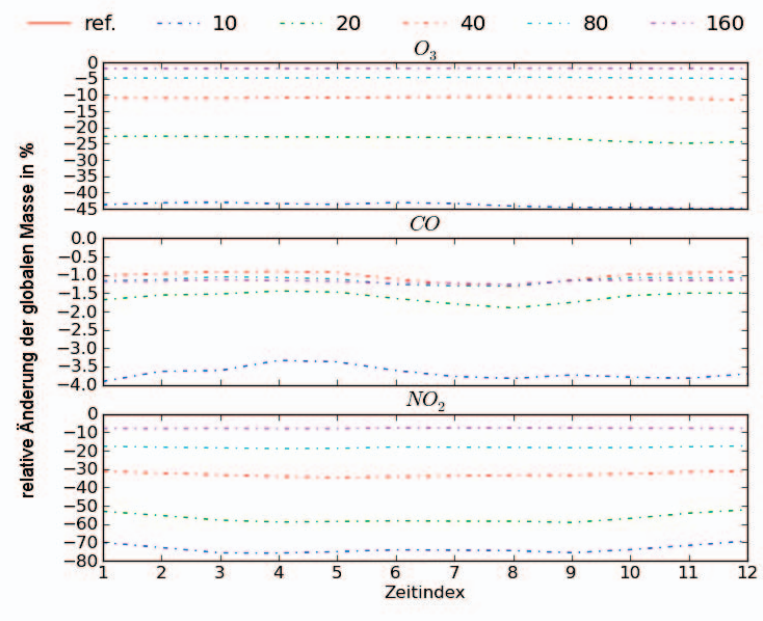

Abbildung 3.3: relative Abweichungen der globalen Masse bei vertikaler Interpolation mit 10 bis 160 gleichverteilten Druckstufen im MOZART-Modell: O_3 und NO_2 verbessern sich mit zunehmender Stufenzahl, bei CO pendeln die Abweichungen bereits ab 20 Stufen in etwa um den gleichen Wert.
durchgehende Linie: Referenz (0% Abweichung), gestrichelte Linien: Abweichungen bei angegebener Anzahl von Druckstufen

Eine erste Abschätzung der Güte der Interpolation liefert ein Vergleich der globalen Massen. Abbildung 3.3 zeigt einen solchen Vergleich für die drei bereits vorher bei der horizontalen Interpolation betrachteten Stoffe O_3, CO und NO_2. Hierbei fällt auf, dass sowohl O_3 als auch NO_2 bei 10 Druckstufen starke Abweichungen von bis zu -45 bzw. -80 % aufweisen und die Abweichungen auch bei 20 Stufen noch erheblich sind. Bei CO hingegen liegen die Abweichungen bereits ab 20 Stufen nahe beieinander und auch die maximale Abweichung ist mit ca. -4 % um mehr als eine Größenordnung kleiner als bei den anderen beiden Stoffen. Insgesamt zeigt sich bei allen Stoffen eine Tendenz zur Abnahme der Abweichungen mit zunehmender Stufenzahl, die speziell bei

O_3 in vielen Modellen eine proportionale Abhängigkeit nahelegt. NO_2 zeigt ebenfalls eine starke Abhängigkeit von der Stufenzahl, die aber weniger strukturiert scheint. CO zeigt hingegen nur eine schwach ausgeprägte Abhängigkeit, die hauptsächlich vom Unterschied zwischen den beiden gröbsten Einteilungen (10 und 20 Stufen) dominiert ist.

Eine zusätzliche Extrapolation bei gleichen Druckstufen zeigt je nach betrachtetem Stoff unterschiedliche Auswirkungen: Während sich die Abschätzung der globalen Masse von Ozon in allen Modellen verbessert oder gleich bleibt, weisen CO und NO_2 teils stärkere Abweichungen als bei reiner Interpolation auf. Bei CO liegen diese bei vielen Modellen weiterhin in der gleichen Größenordnung, es gibt jedoch auch Ausreißer mit bis über 300% Abweichung bei niedrigen Auflösungen. Im Falle von NO_2 schwanken die Abweichungen bei vielen der betrachteten Modelle stärker als bei der reinen Interpolation, teils werden auch dort Abweichungen bis 200% erreicht. Dieses unterschiedliche Verhalten ist wahrscheinlich darauf zurückzuführen, dass die niedrige Bodenkonzentration von Ozon die Abschätzung in diesem Bereich nicht nennenswert beeinflusst, bei den anderen beiden Substanzen mit hohen Bodenkonzentrationen jedoch zu massiven Verfälschungen führt. Bei der Extrapolation nach oben ergibt sich durch Einbeziehung des Datenpunktes bei 0 hPa und wegen des im Allgemeinen geringeren Abstands zur obersten Quellschicht eine bessere Abschätzung der stratosphärischen Ozonkonzentration, wohingegen CO und NO_2 wegen ihrer verfälschten Bodenkonzentrationen hiervon nicht profitieren können. Abbildung 3.4 stellt die Auswirkungen der Extrapolation auf die globale Masse am Beispiel des MOZART-Modells dar.

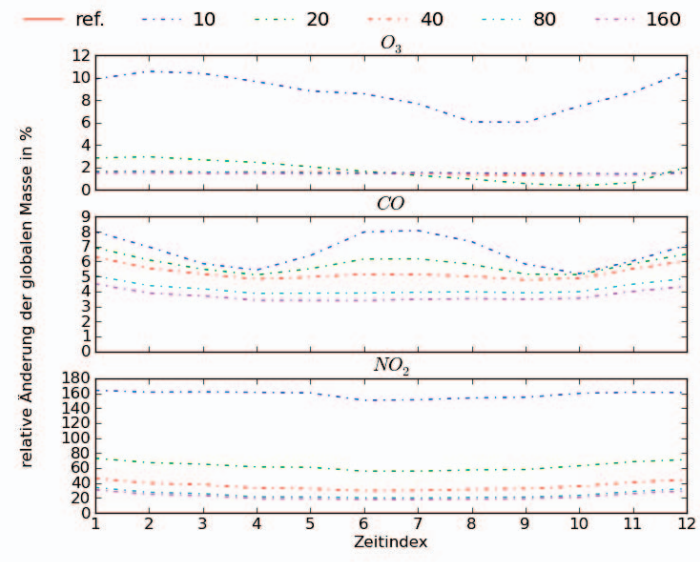

Abbildung 3.4: relative Abweichungen der globalen Masse bei vertikaler Interpolation und Extrapolation mit 10 bis 160 gleichverteilten Druckstufen im MOZART-Modell: Während bei O_3 deutliche Verbesserungen gegenüber der reinen Interpolation zu sehen sind, verschlechtern sich die Abweichungen bei CO leicht und bei NO_2 teils sehr deutlich.
durchgehende Linie: Referenz (0% Abweichung), gestrichelte Linien: Abweichungen bei angegebener Anzahl von Druckstufen

Auswirkungen auf Säulendichten

Bei genauerer Betrachtung der Säulendichten von Ozon zeigen sich die größten Abweichungen bei allen betrachteten Modellen im Äquatorbereich der Sommerhalbkugel mit lokalem Maximum im Bereich zwischen Afrika und Australien sowie einem globalen Maximum nördlich von Australien. Während sich die Amplitude der Abweichungen mit zunehmender Stufenzahl verringert, bleibt dieses Muster weitgehend erhalten, was auf grundsätzliche Schwierigkeiten bei der Interpolation in diesen Bereichen hindeutet. Abbildung 3.5 zeigt die beschriebenen Beobachtungen für 10 Druckstufen im MOZART-Modell.

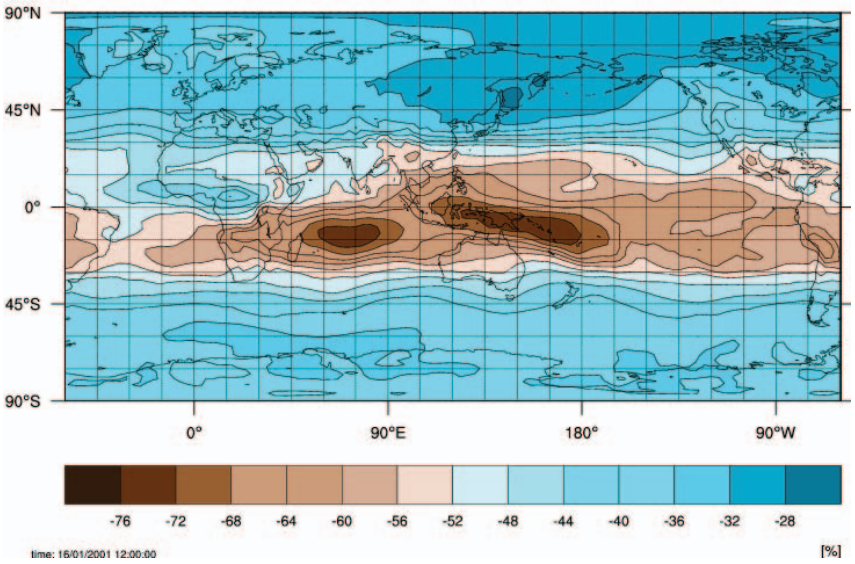

Abbildung 3.5: relative Abweichungen der Säulendichten von Ozon bei vertikaler Interpolation mit 10 gleichverteilten Druckstufen zwischen 0 und 1000 hPa im MOZART-Modell: Die gößten Abweichungen konzentrieren sich im Äquatorbereich mit lokalen Maxima zwischen Afrika und Australien sowie nördlich von Australien.

Dieses Muster lässt sich in den untersuchten Modellen weder für CO noch für NO_2 in dieser Form feststellen, hier sind die Abweichungen in der Regel relativ gleichmäßig über die Erdoberfläche verstreut und variieren stark mit der Anzahl der gewählten Druckstufen und dem konkreten Modell.

Betrachtung der Höhenprofile

Um die Problematik bei der Interpolation der Stoffkonzentrationen von O_3, CO und NO_2 besser zu verstehen und mögliche Erklärungsansätze für das beobachtete Verhalten zu gewinnen, sollen hier exemplarisch einige Höhenprofile der Stoffe kurz betrachtet werden. Zum Vergleich werden die für 10 und 20 Schichten interpolierten Werte (ohne Extrapolation) dargestellt. Pro Stoff wurden jeweils zwei Profile exemplarisch ausgewählt, die jedoch nicht notwendigerweise repräsentativ für andere Gitterpunkte sind.

Abbildung 3.6 stellt zwei Höhenprofile von Ozon dar, wobei das linke Profil einen Punkt aus dem Bereich nordöstlich von Australien zeigt, an dem die Säulendichten bei allen betrachteten im globalen Vergleich am stärksten unterschätzt werden (-73% bei 10 Stufen). Das rechte Profil stammt aus dem Bereich über dem europäischen Kontinent, wo die Säulendichte nur etwa halb so stark unterschätzt wird (-32% bei 10 Stufen). Um den Konzentrationsverlauf in den unteren und mittleren Schichten besser zu veranschaulichen, sind die Profile zusätzlich um den Faktor 10 vergrößert aufgetragen. Im linken Profil ist bei etwa 100 hPa eine deutliche Änderung des Gradienten zu beobachten, die sich mit den gewählten gleichverteilten Interpolationsstufen im oberen Bereich nicht gut abbilden lässt. Da keine Extrapolation vorgenommen wurde, liegt der höchste interpolierte Punkt bei 10 Stufen Auflösung mit ca. 110 hPa gerade noch im Bereich niedriger Konzentration, so dass das stratosphärische Ozon praktisch nicht mit einbezogen wird. Die Erhöhung der Auflösung auf 20 Stufen führt zu einem ersten Datenpunkt bei ca. 52 hPa, der den Fehler zwar signifikant verringert, aber den Konzentrationsverlauf letztendlich auch nicht zufriedenstellend genau wiedergeben kann. Das rechte Profil zeigt eine weniger starke Gradientenänderung, die zudem - auch wegen der in höheren Breiten niedriger liegenden Stratosphäre - schon unterhalb von 200 hPa beginnt und so die hohen Konzentrationen im stratosphärischen Bereich zwar etwas besser einbezieht, jedoch weiterhin noch erhebliche Abweichungen zeigt.

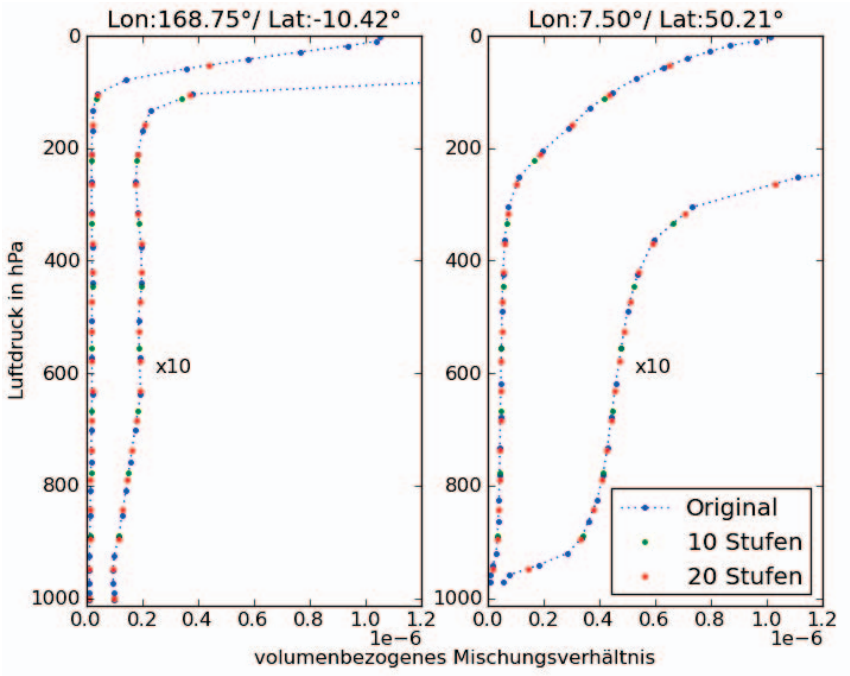

Abbildung 3.6: Höhenprofil von Ozon für zwei horizontale Gitterpunkte im MOZART-Modell mit Vergleich von Originaldaten und auf 10 bzw. 20 Schichten interpolierten Daten, zur besseren Veranschaulichung der unteren und mittleren Schichten sind die Profile ebenfalls mit dem Faktor 10 multipliziert aufgetragen
links: Punkt nordöstlich von Australien, an dem die Abweichungen in vielen Modellen und Auflösungen relativ groß sind
rechts: Punkt auf dem europäischen Kontinent, der deutlich geringere Fehler aufweist

In Abbildung 3.7 sind zwei Profile für *CO* dargestellt, bei denen das Linke einen Punkt auf dem europäischen Kontinent zeigt, an dem die Säulendichten bei niedrigen Auflösungen vergleichsweise stark unterschätzt werden (−16% bei 10 Stufen). Das rechte Profil hingegen zeigt einen Punkt im Himalayagebirge, an dem die Säulendichten überschätzt werden (+13% bei 10 Stufen).

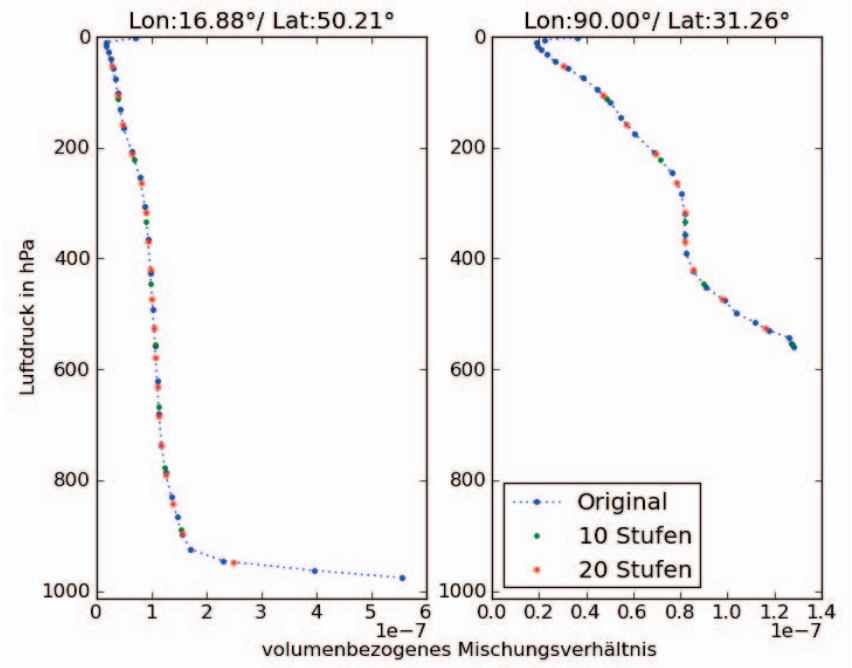

Abbildung 3.7: Höhenprofil von *CO* für zwei horizontale Gitterpunkte im MOZART-Modell mit Vergleich von Originaldaten und auf 10 bzw. 20 Schichten interpolierten Daten
links: Punkt auf dem europäischen Kontinent mit starker Unterschätzung der Säulendichte bei niedrig auflösender Interpolation
rechts: Punkt im Himalayagebiet mit starker Überschätzung der Säulendichte bei niedrig auflösender Interpolation, die Höhenachse ist wegen des Gebirges stark gestaucht

Im linken Profil zeigt sich im Bodenbereich ein ähnlich starker Knick im Konzentrationsverlauf, wie es bei Ozon im stratosphärischen Bereich beobachtet werden kann. Dies führt deswegen auch zu praktisch identischen Problemen bei der Interpolation. Im rechten Profil ist die Höchenachse wegen der großen Bodenhöhe stark zusammengestaucht und somit in den Quelldaten feiner aufgelöst als eine auf die komplette Höhe gestreckte Achse auf der linken Seite. Die Überschätzung ist nicht eindeutig zu erklären: Bereits in einer Nachbargitterzelle liegt die Abweichung bei sehr ähnlich verlaufendem Profil bei -20%. Es handelt sich hier vermutlich um Artefakte der Wechselwirkung vom spe-

zifischen Profilverlauf mit den gewählten Interpolationsstufen, die durch die häufigen Gradientenänderungen des Profils entstehen. Zusätzlich ist der Unterschied zwischen der Zahl der Datenpunkte in Quelle und interpoliertem Resultat größer als bei einer lang gestreckten Achse, was ebenfalls zu einer höheren Empfindlichkeit führen könnte.

Auf eine Diskussion des Höhenprofils von NO_2 wird verzichtet, da der Konzentrationsverlauf im oberen Bereich dem von Ozon und im Bodenbereich dem von CO mit etwas stärkter ausgeprägtem Knick ähnelt. Die Problematik ergibt sich entsprechend den vorherigen Ausfürungen.

Optimierung der interpolierten Höhenachse

Bei Betrachtung der Höhenprofile fällt auf, dass die größten Gradienten der untersuchten Stoffe entweder in Bodennähe oder in der Stratosphäre liegen. Eine Höhenachse mit gleichverteilten Schichten ist deswegen für die Abbildung der Konzentrationsverläufe nicht optimal. Dies führt dazu, dass extrem viele Schichten berechnet werden müssen, um den Fehler auf ein akzeptables Maß zu reduzieren. Da die Quelldatensätze im Normalfall zwischen 20 und 30 Schichten enthalten und sich die Anzahl der Schichten stark auf das Datenvolumen auswirkt, sollte diese Zahl im Idealfall auch in den interpolierten Daten beibehalten oder nach Möglichkeit sogar reduziert werden.

Es liegt deswegen nahe, eine Schichtverteilung zu benutzen, die dem beobachteten Verhalten besser Rechnung trägt und in den unproblematischen Höhenbereichen die Auflösung zugunsten der problematischen Bereiche reduziert. Mit dieser Maßgabe wurde eine neue Aufteilung der Höhenachse erstellt, die im oberen und unteren Bereich eine feinere Einteilung aufweist und aus insgesamt 25 Schichten zwischen 0 und 1000 hPa besteht (siehe Tabelle 3.8). Basierend auf dieser optimierten Achse wurden noch einmal die Abweichungen der globalen Massen untersucht.

#	Druck	#	Druck	#	Druck	#	Druck
1	1	8	40	14	400	20	900
2	3	9	60	15	500	21	925
3	5	10	80	16	600	22	950
4	7	11	100	17	700	23	970
5	10	12	200	18	800	24	990
6	20	13	300	19	850	25	1000
7	30						

Abbildung 3.8: Druckwerte der optimierten Höhenachse in hPa: feinere Aufteilung der Druckstufen im problematischen oberen und unteren Bereich

Dabei zeigt sich, dass diese Einteilung bei allen betrachteten Modellen zu ähnlich geringen Abweichungen führt, wie sie mit 40 und teils sogar 80 gleichverteilten Schichten erreicht werden. Die Auswirkungen zeigen sich bei O_3 und NO_2 besonders deutlich, im Falle des MOZART-Modells liegen die Abweichungen hier sogar unter denen von 80 gleichverteilten Schichten. Bei CO sind ebenfalls Verbesserungen zu erkennen, diese sind jedoch wegen der generell geringeren Schwankungsbreiten und Abweichungen weniger deutlich. Auf die Nutzbarkeit der Extrapolation hat die optimierte Höhenachse keinen zusätzlichen sichtbaren positiven Einfluss. Ihre Abweichungen bewegen sich ebenfalls im Bereich derer von zwischen 40 und 80 gleichverteilten Schichten.

3.4 Zusammenfassung und Ausblick

Mit `CFchecker` steht ein Werkzeug zur Prüfung der CF-Konformität von NetCDF-Dateien bereit. Dabei werden alle sechs derzeit verfügbaren Versionen der CF-Konventionen unterstżutzt. Es lässt sich sowohl als Kommandozeilenwerkzeug zur manuellen Kontrolle von einzelnen Dateien oder ganzen Verzeichnistrukturen einsetzen als auch in andere Anwendungen integrieren, um beispielsweise vor weiteren Verarbeitungsschritten eine Konformitätsprüfung automatisch durchzuführen. Konkret geplant ist die Integration von `CFchecker` in eine Webanwendung, die Benutzern den Upload und die Verifikation ihrer Datensätze online und ohne Notwendigkeit einer lokalen Installation erlaubt. Diese Möglichkeit soll bei zukünftigen Modellvergleichen ebenfalls zur Einreichung der Ergebnisse genutzt werden, um den Teilnehmern eine möglichst hilfreiche Rückmeldung zur CF-Konformität ihrer Datensätze zu bieten und nur noch CF-konforme Daten ins Archiv aufzunehmen.

Mit `diff`, `mean`, `dimaverage`, `colint` und `globmass` sind - teils im Zuge der Untersuchung des Interpolationsverhaltens - nützliche Operatoren entstanden, die bei der Auswertung von Atmosphärendaten hilfreich sind. Auch sie können - neben ihrer Benutzung auf der Kommandozeile - zukünftig in eine Web-Anwendung integriert werden und so eine erste Auswertung von Datensätzen erleichtern. Es besteht jedoch auch noch Verbesserungspotential bei verschiedenen Operatoren. So kann die universelle Benutzbarkeit von `diff` und `mean` verbessert werden, wenn statt der NetCDF-Variablennamen die `standard_name`-Attribute zur Identifikation von Variablen mit gleichem Inhalt benutzt werden. Ein solches Vorgehen würde eine Anwendung auf Datensätze unterschiedlicher Herkunft ermöglichen, die aktuell wegen der allgemein fehlenden Festlegung von Variablennamen nur innerhalb einer eng definierten Teilmenge wie etwa den HTAP-Datensätzen möglich ist.

Der entwickelte horizontale Interpolationsoperator kann nach Evaluation

seiner Auswirkungen auf globale Masse und atmosphärische Säulendichte vor allem bei Verfeinerungen des Gitters als paxistauglich bezeichnet werden. Vergröberungen können ebenfalls bis zu einem gewissen Grad toleriert werden, das Zielgitter sollte jedoch nicht mehr als einen Faktor vier gröber als das Quellgitter sein, um die Fehler in Grenzen zu halten. Der einfache bilineare Interpolationsansatz kann bereits die benötigte Genauigkeit erreichen und ist im Vergleich zu anderen Ansätzen wie höherdimensionalen Polynomen oder Splines weniger rechenintensiv. Dies ist auch im Zusammenhang mit einer möglichen Integration der Interpolationsfunktion in interaktive Anwendungen zur Auswertung von Datensätzen wichtig.

Im Bereich der vertikalen Interpolation besteht weiterhin Entwicklungs- und Forschungsbedarf. Entscheidend für die korrekte Wiedergabe der betrachteten Erhaltungsgrößen ist eine sinnvolle Wahl der Schichten, wie durch die für den oberen und unteren Höhenbereich optimierte Schichtaufteilung gezeigt werden konnte. Diese Aufteilung hängt jedoch von den Höhenprofilen der einzelnen zu interpolierenden Größen ab, so dass keine allgemeingültige Lösung gefunden werden kann. Eine auf die konkreten Bedürfnisse zugeschnittene Aufteilung ist jedoch durchaus möglich. Die gewählte simple Bestimmung der Schichtgrenzen scheint nach einer ersten Betrachtung brauchbare Ergebnisse zu liefern. Auch hier besteht jedoch noch weiterer Entwicklungsbedarf. Eine bessere Festlegung der Grenzen könnte möglicherweise auch die Gesamtzahl der für die angestrebte Genauigkeit benötigten Schichten reduzieren helfen, was sich positiv auf das Datenvolumen auswirken würde. Zur Entwicklung eines allgemein nutzbaren, masseerhaltenden vertikalen Interpolationsoperators bedarf es weitergehender Betrachtungen.

Die vertikale Extrapolation von Stoffkonzentrationen kann - zumindest unter den hier untersuchten Umständen - nicht allgemein empfohlen werden, falls globale Masse und Säulendichten in vergleichbarem Umfang erhalten bleiben sollen. Bei Stoffen mit niedriger Konzentration und niedrigen vertikalen Gradienten im Bodenbereich, wie es beispielsweise bei Ozon der Fall ist, könnte eine Extrapolation jedoch relativ störungsarm eingestetzt werden. Die Extrapolation im stratosphärischen Bereich kann zumindest bei Ozon das Ergebnis verbessern, da der starke und fast lineare Konzentrationsanstieg sehr gut extrapoliert werden kann und so eine deutliche Unterschätzung verhindert wird.

Der Einsatz von höherdimensionalen Splines könnte ein weiterer Ansatz zur besseren Abbildung der Höhenprofile sein. Ihre Randbedingungen müssten jedoch wahrscheinlich an die unterschiedlichen Höhenprofile der zu betrachtenden Größen angepasst werden, um gute Ergebnisse zu erzielen. In ungünstigen Fällen könnten die Abweichungen stärker ausfallen als bei einer einfachen linearen Interpolation.

Kapitel 4

Bereitstellung und Auslieferung von Geodaten

Dieses Kapitel geht genauer auf das Konzept und die konkrete Realisierung eines WCS-Servers für die Auslieferung von Daten im CF-konformen NetCDF-Format ein.

Dabei wird zunächst ein Überblick über bereits existierende WCS-Softwarepakete gegeben. Im Anschluss wird die in Kooperation mit der *Washington University in St. Louis*[1], USA erfolgte Eigenentwicklung eines WCS-Servers mit Fokus auf CF-konforme NetCDF-Datensätze beschrieben. Diese Entwicklungsarbeit wurde durch eine Projektfinanzierung der US Umweltschutzbehörde EPA teilfinanziert.

4.1 Existierende WCS-Server Software

Es existieren bereits einige Programme und Projekte, die einen WCS-Server nach OGC-Standards realisieren. Exemplarisch werden hier die großen quelloffenen Projekte **Geoserver** und **Mapserver** sowie die Projekte **THREDDS Data Server** und **deegree** kurz vorgestellt. Dabei wird auch auf die fehlenden Merkmale hingewiesen, die die jeweilige Software zur Bereitstellung der HTAP-Datensätze disqualifizieren.

Insgesamt lässt sich feststellen, dass die Unterstützung und Nutzung von WMS und WFS weit stärker verbreitet ist, als die von WCS. Ein Grund dafür ist, dass WMS für einfache Anwendungsfälle (Auslieferung von Rasterdaten in Form georeferenzierter Bilder) eine ähnliche Funktionalität wie WCS bietet und dabei durch das simplere Protokoll deutlich einfacher zu handhaben ist.

[1] `http://www.wustl.edu`

WCS wird somit erst dann notwendig, wenn größere Flexibilität bei der Datenauslieferung benötigt wird, die WMS nicht bietet. Dies trifft in erster Linie auf semantische Rasterdatenformate zu, die mehr und komplexere Informationen enthalten als einfache Bildformate, zum Beispiel NetCDF. WFS hingegen zielt mit der Auslieferung von Vektordaten auf komplementäre Anwendungsgebiete abseits der Rasterdaten ab, die WMS und WCS nicht abdecken können.

Bei Recherchen im Rahmen dieser Arbeit hat sich gezeigt, dass es im wissenschaftlichen Bereich in den letzten Jahren verschiedene Bemühungen gab, OGC-Standards wie WFS und WCS für die Nutzung mit wissenschaftlichen Daten zu erweitern. Der unten beschriebene *THREDDS Data Server* ist ein solches Beispiel. Zusätzlich lassen sich Hinweise auf diverse Designstudien finden, die jedoch größtenteils ohne weitere auffindbare Quelltexte oder Resultate wieder eingestellt wurden. So zum Beispiel das *GADS-WCS*[2] Projekt, das langfristig aber möglicherweise Einzug in GeoServer hält (siehe Abschnitt 4.1.1).

4.1.1 GeoServer

Bei *GeoServer*[3] handelt es sich um einen in der Programmiersprache *Java* geschriebenen Geodatenserver. Die Sofware wird quelloffen unter Leitung der *Open Source Geospatial Foundation*[4] (OSGeo) entwickelt und vertrieben. Sie baut auf dem ebenfalls von OSGeo betreuten GIS-Toolkit GeoTools[5] auf. *GeoServer* unterstützt die OGC-Standards WMS 1.1.1, WFS(-T) 1.0 und 1.1 sowie WCS 1.0 und 1.1. Zusätzlich stellt GeoServer die Referenzimplementierungen von WFS (1.0 und 1.1) und WCS 1.1. Neben den OGC-Standards werden auch Protokolle anderer (kommerzieller) Programme wie Google Earth unterstützt. Eingabedaten können aus verschiedenen Datenbanken, Vektorformaten und Bilddateien eingebunden werden, Ausgabeformate beinhalten diverse XML-Dialekte wie KML und GML sowie Bildformate (JPEG, SVG, GIF, PNG, ...). Über eine eingebaute Projektionsbibliothek können WMS- und WFS-Daten außerdem auf Anfrage in Echtzeit in eines von mehreren hundert Koordinatensystemen reprojiziert werden.

Zur Problematik der Auslieferung von NetCDF-Daten über das WCS-Protokoll hat man sich innerhalb des Projektes bereits im Jahr 2006 Gedanken gemacht und diese auf einer eigenen Seite zusammengetragen[6]. Als Basis dient

[2] `http://sourceforge.net/projects/gads-wcs/`
[3] `http://geoserver.org/`
[4] `http://www.osgeo.org/`
[5] `http://www.geotools.org/`
[6] `http://geoserver.org/display/GEOS/Multidimensional+WCS`

dabei der *Grid Access Data Service*[7] (GADS) des *Reading e-Science Centre*[8], der testweise zur Auslieferung von NetCDF an GeoServer angebunden wurde. GADS ist in seiner ursprünglichen Form jedoch nicht WCS-kompatibel und zeigt auch seit spätestens 2008 keine sichtbaren Zeichen von Weiterentwicklung mehr. Die letzten verfügbaren Informationen deuten auf Bestrebungen hin, sich mit GADS in Zukunft stärker am WCS-Standard zu orientieren. Der auf der GeoServer-Webseite verlinkte Testdienst mit GADS-Einbindung[9] ist nicht mehr funktionstüchtig.

Zusammenfassend lässt sich feststellen, dass das Interesse an der Unterstützung von NetCDF als Ausgabedatenformat im Zusammenhang mit dem WCS-Protokoll zwar grundsätzlich besteht, jedoch aktuell nicht intensiv innerhalb der GeoServer-Entwicklung verfolgt wird. Dementsprechend ist auf absehbare Zeit keine GeoServer-basierte Lösung zu erwarten, die zur Bereitstellung des HTAP-Datenarchivs über eine WCS-Schnittstelle geeignet wäre.

4.1.2 MapServer

MapServer[10] ist ursprünglich im Jahr 1995 an der *University of Minnesota* (UMN) in Kooperation mit der NASA zur Auslieferung von deren Satellitenbildern entwickelt worden. Wie *GeoServer* ist auch *MapServer* quelloffene Software und wird ebenfalls unter dem Dach von *OSGeo* entwickelt, als Programmiersprache kommt im Gegensatz zu *GeoServer* allerdings C++ zum Einsatz. Zudem konzentriert sich *MapServer* stärker auf OGC-Standards (WMS, WFS, WCS, ...) und implementiert WFS-T nicht. Reprojektion ist ähnlich wie bei *GeoServer* über eine eingebaute Bibliothek (PROJ.4[11]) zur Abfragezeit möglich.

Für WMS und WFS wird auch der Client-Betrieb unterstützt, das WCS-Protokoll ist sowohl in Version 1.0 als auch in Version 1.1 implementiert. Die Auslieferung von NetCDF-Daten über WCS wird theoretisch indirekt durch Nutzung der *Geospatial Data Abstraction Library*[12] (GDAL) unterstützt. Da GDAL NetCDF allerdings nur partiell unterstützt, empfielt die MapServer-Dokumentation die Umwandlung der Daten ins GeoTIFF-Format, das dann problemlos ausgeliert werden kann[13]. Zur Bereitstellung des HTAP-Datenarchivs ist diese Lösung jedoch unbefriedigend, zumal durch die Konvertierung

[7] `http://www.resc.rdg.ac.uk/twiki/bin/view/Resc/GridAccessDataService`
[8] `http://www.resc.rdg.ac.uk`
[9] `http://lovejoy.nerc-essc.ac.uk:8080/geoserver/`
[10] `http://www.mapserver.org/`
[11] `http://trac.osgeo.org/proj/`
[12] `http://www.gdal.org/`
[13] `http://www.mapserver.org/ogc/wcs_format.html#netcdf`

ins GeoTIFF-Format sowohl Flexibilität als auch Metadaten verloren gehen.

4.1.3 THREDDS Data Server

Innerhalb des THREDDS-Projekts[14] (*Thematic Realtime Environmental Distributed Data Services*) bietet Unidata mit dem in Java geschriebenen und quelloffenen *THREDDS Data Server*[15] (TDS) ebenfalls einen WMS- und WCS-Server an. WMS ist dabei in den beiden letzten Versionen 1.1.1 und 1.3.0 implementiert. Als WCS-Ausgabeformat werden CF-konformes NetCDF und GeoTIFF unterstützt. Es kommt jedoch nur die veraltete WCS-Protokollversion 1.0 zum Einsatz, darüber hinaus wird nur eine Datenvariable (*range*) pro Ausgabedatei (*coverage*) unterstützt, was eine erhebliche Einschränkung bei der Bereitstellung der HTAP-Datensätze bedeuten würde. Insgesamt wird die WCS-Unterstützung von Unidata als experimentell bezeichnet.[16]

4.1.4 deegree

Das Softwarepaket *deegree*[17] ist ebenfalls ein OSGeo-Projekt. Laut eigener Aussage der an der Entwicklung beteiligten *Lat/Lon GmbH*[18] handelt es sich um „die derzeit umfangreichste Implementierung von OGC- und ISO-Standards im Bereich Freier Software“. Neben WMS, WFS und WCS unterstützt die aktuelle Version 3 noch weitere OGC-Standards wie zum Beispiel *Catalogue Services for the Web*[19] (CSW), *Web Processing Service*[20] (WPS) und *Sensor Observation Service*[21] (SOS). Auch hier kommt jedoch nur der alte WCS 1.0-Standard zum Einsatz, NetCDF-Unterstützung ist nicht dokumentiert.[22]

4.1.5 Kommerzielle Softwarepakete

Neben den erwähnten Open Source-Paketen gibt es eine große Anzahl kommerzieller Programme, die sich mit OGC-Standards allgemein und WCS speziell befassen. Diese sind oft auf spezifische Kundengruppen wie beispielsweise Vermessungsämter, Infrastrukturplaner und Katastrophenschutz hin ausgelegt. Wegen ihrer kommerziellen proprietären Natur und dem Fokus auf nicht-

[14] `http://www.unidata.ucar.edu/projects/THREDDS/`
[15] `http://www.unidata.ucar.edu/projects/THREDDS/tech/TDS.html`
[16] `http://www.unidata.ucar.edu/projects/THREDDS/tech/reference/WCS.html`
[17] `http://www.deegree.org`
[18] `http://www.lat-lon.de`
[19] `http://www.opengeospatial.org/standards/specifications/catalog`
[20] `http://www.opengeospatial.org/standards/wps`
[21] `http://www.opengeospatial.org/standards/sos`
[22] `http://wiki.deegree.org/deegreeWiki/deegree3/WCSDevelopment`

wissenschaftliche Kunden mit ihren strikten Standardvorgaben ist die Nutzung im wissenschaftlichen Bereich jedoch weitgehend unpraktikabel. Ein Vertreter dieser Kategorie ist *ArcGIS Server*[23] der amerikanischen Firma *esri*.

4.2 Eigenimplementierung eines OGC WCS Server

4.2.1 Ausgangssituation und eigene Beiträge

Ausgangssituation

Die Ausgangssituation bei Beginn dieser Arbeit war eine frühe, von Kari Hoijvari an der *Washington University in St. Louis* für *Microsoft Windows* entwickelte, Testversion des WCS-Servers. Dieser Code wurde im Verlauf der vorliegenden Arbeit signifikant verbessert und erweitert. Dieser Abschnitt gibt einen kurzen Überblick über die wichtigsten im Rahmen dieser Arbeit in den Servercode (und dessen Funktionsprinzip) eingeflossenen eigenen Entwicklungsbeiträge.

Eigene Entwicklungsbeiträge

Im ersten Schritt musste die bis dahin primär für *Microsoft Windows* entwickelte Serversoftware auf dem im IEK-8 benutzten Linux-Server lauffähig gemacht werden. Dazu wurde das Ausschneidemodul `pync3` (siehe Abschnitt 4.2.8) im Rahmen dieser Arbeit als Ersatz für das in C++ geschriebene und auf Windows benutzte `nc3`-Modul entwickelt. Zusätzlich zur verbesserten Linux-Kompatibilität ohne die Notwendigkeit zur Kompilierung von C++-Code bietet es auch eine einfacher zu nutzende Grundlage für weitere Entwicklungen. Kari Hoijvari arbeitet inzwischen am kompletten Ersatz des `nc3`-Moduls auf Windows zugunsten der einfacher zu verwendenden Python-Implementierung.

Einen weiteren wichtigen Schritt stellte die weitgehende Überarbeitung aller Nachrichtenschnittstellen (*GetCapabilities*, *DescribeCoverage*, *GetCoverage*) des Servers - sowohl im Bezug auf die Anfrageparameter als auch auf die zurückgelieferten Antworten - mit dem Ziel der Einhaltung des WCS-Standards in Version 1.1.x dar. Dies war nötig, da die ursprüngliche Serverversion sich trotz erklärtem Ziel der Verwendung von WCS Version 1.1 stark an Version 1.0 anlehnte, die in vielen wichtigen Punkten von Version 1.1 abweicht. Dazu zählte auch die Implementierung der `store=false`-Unterstützung für *GetCoverage*-Anfragen. Diese wurde wegen ihres im Vergleich zu `store=true` deut-

[23] `http://www.esri.com/software/arcgis/arcgisserver/index.html`

lich höheren Implementierungsaufwands bis dahin nicht vorgenommen, obwohl sie für den standardkonformen Betrieb eines WCS-Servers von Bedeutung ist. Einen Überblick über die Funktionsweise des WCS-Protokolls gibt Abschnitt 4.2.3.

Die Handhabung der Metadaten wurde von einer rein statischen Vorgenerierung zu einem dynamischeren Konzept mit zwei getrennten Metadatenquellen weiterentwickelt. Die Umsetzung und die Unterschiede zum vorherigen Ansatz werden in Abschnitt 4.2.6 genauer beschrieben.

Die in Abschnitt 4.2.7 beschriebene Ausschneidemethode von geographischen Bereichen wurde durch eine flexiblere Detektion von Koordinatenachsen verbessert. Diese ermöglicht die Verwendung von deutlich mehr CF-konformen Quelldatensätzen, ohne dass eigentlich optionale CF-Attribute - speziell das `axis`-Attribut - für die Verwendbarkeit mit dem WCS-Server weiterhin vorausgesetzt werden müssen.

Neben den oben genannten Beiträgen, die praktisch ausschließlich in Eigenarbeit erfolgten, wurden viele weitere Detailverbesserungen und Entwicklungen in intensiver Zusammenarbeit durchgeführt. Dazu zählen zum Beispiel:

- Weiterentwicklung des `pync3`-Moduls
- Weiterentwicklung des Abbildungskonzeptes von CF-NetCDF auf *WCS Coverages*
- Erstellung von (Installations-)Anleitungen[24][25]
- Entwicklung von Unit-Tests

4.2.2 Motivation

Wie Abschnitt 4.1 exemplarisch zeigt, genügen die bisher existierenden Geodatenserver im Bezug auf die Auslieferung der NetCDF-Datensätze des HTAP-Datenarchivs den gestellten Anforderungen nicht. Mindestens eines von zwei wesentlichen Problemen hat sich dabei bei allen betrachteten Serverpaketen herausgestellt:

- Der WCS-Standard ist nur in der veralteten Version 1.0 implementiert und nicht in der aktuelleren Version 1.1.x.
- Die Auslieferung von NetCDF-Daten über WCS ist entweder gar nicht oder nur teilweise implementiert und unterstützt die CF-Konventionen nicht.

[24] `http://wiki.esipfed.org/index.php/WCS_Wrapper_Installation_WindowsOP`
[25] `http://wiki.esipfed.org/index.php/WCS_Wrapper_Installation_LinuxOP`

Diese Probleme sind zum Großteil darauf zurückzuführen, dass die Auslieferung von Daten im NetCDF-Format über WCS noch nicht standardisiert ist (siehe Abschnitt 2.2.1). Vor allem die größeren Projekte wie *GeoServer* und *MapServer* (und auch *deegree*) sehen sich jedoch selbst als Anbieter stabiler Standardprotokolle und halten sich deswegen bis zur Verabschiedung des endgültigen Standards weitgehend zurück.

Eine eigene Implementierung des WCS-Protokolls unter Berücksichtigung von NetCDF war somit aus zwei verschiedenen Blickwinkeln hilfreich: Einerseits musste die Problematik der Bereitstellung von wissenschaftlichen Daten zeitnah gelöst werden, was verfügbare Software zu Beginn der Entwicklung wie in Abschnitt 4.1 beschrieben nicht leisten konnte. Andererseits eröffnet die frühe Implementierung und das frühzeitige Testen der in Entwicklung befindlichen Standards zur Auslieferung von NetCDF Möglichkeiten bei deren Mitgestaltung und der Identifikation möglicher Probleme. Zu diesem Zweck wurde Kontakt mit Ben Domenico (UCAR/Unidata), einem der beiden Autoren des OGC/WCS Diskussionspapiers „*Web Coverage Service (WCS) 1.1 extension for CF-netCDF 3.0 encoding* [Domenico und Nativi, 2009], aufgenommen und ein Erfahrungsaustausch vereinbart.

4.2.3 Das WCS-Protokoll

Dieser Abschnitt behandelt im Speziellen die WCS-Version 1.1.2, die zu Beginn der Entwicklungsarbeit die aktuellste verfügbare Version war.

Das WCS-Protokoll setzt - wie WMS und WFS (siehe Abschnitt 2.2.1) - auf dem *Hypertext Transfer Protocol* (HTTP) auf und unterscheidet drei grundlegende Anfragetypen, die ein Client an einen WCS-Server richten kann: *GetCapabilities*, *DescribeCoverage* und *GetCoverage*. Sowohl die *GetCapabilities*- als auch die *DescribeCoverage*-Anfrage sind dabei reine Metadatenabfragen, die dem Client die für die eigentliche Datenabfrage (*GetCoverage*) benötigten Informationen bereitstellen. Metadaten werden in Form von XML-Dokumenten an den Client ausgeliefert, die auf vom OGC definierten Schemata für die Beschreibung von OGC/WCS-Diensten aufsetzen.

Parameterübertragung mit HTTP GET

Die Parameterübertragung erfolgt im einfachsten Fall durch Verwendung der so genannten *HTTP GET* Methode. Dabei werden die Parameter an den Basis-(URL-)Pfad angehängt. Der Parameter-Teil einer URL wird durch ein Fragezeichen eingeleitet, die einzelnen Parameter werden als durch ein Und-Zeichen getrennte Name-Wert-Paare (`param1=wert1&param2=wert2`) ausgedrückt. Für

eine WCS *GetCapabilities*-Anfrage mit ihren obligatorischen Parametern sähe dies beispielsweise folgendermaßen aus:

```
http://server/Pfad?service=WCS&request=GetCapabilities&
                   acceptversions=1.1.2
```

In diesem Fall werden die Parameter `service`, `Request` und `acceptversions` mit den Werten `WCS`, `GetCapabilities` und `1.1.2` übertragen. Für jeden Anfragetyp werden unterschiedliche obligatorische und optionale Parameter unterstützt. Die Reihenfolge der Parameter ist beliebig und es wird nicht zwischen Groß- und Kleinschreibung unterschieden. In allen Fällen obligatorisch sind die beiden Parameter `service` und `request`, die das zu verwendende OGC-Protokoll und den Anfragetyp festlegen. Der `Pfad`-Bestandteil der URL identifiziert den abzufragenden Datenkatalog (siehe Abschnitt 4.2.4). Im WCS-Standard wird in diesem Zusammenhang von einem *server* gesprochen, der jedoch nicht mit einem Server im computertechnischen Sinne identisch ist, sondern lediglich eine Gruppe von Datensätzen beschreibt. Im Falle des HTAP-Archivs ist so beispielsweise der Katalogname `HTAP_monthly` eine Sammlung (oder Gruppe), die Daten aus dem HTAP-Modellvergleich enthält, die eine monatliche Zeitauflösung haben.

GetCapabilities

Der *GetCapabilities*-Typ dient der allgemeinen Beschreibung des abgefragten Dienstes. Das XML-Dokument ist in vier wesentliche Bereiche gegliedert: *ServiceIdentification*, *ServiceProvider*, *OperationsMetadata* und *Contents*. Die ersten drei Bereiche sind dabei für alle *OGC Web Services* einheitlich definiert, nur der letzte Abschnitt ist WCS-spezifisch. Über den `sections`-Parameter kann der Client beliebige Teilmengen dieser vier Abschnitte anfragen, falls nur spezielle Informationen benötigt werden.

ServiceIdentification enthält allgemeine Informationen zum abgefragten Datenkatalog. Dazu zählen: Titel und Zusammenfassung des Inhalts (`Title`, `Abstract`), eine Liste von Schlüsselwörtern für Katalogdienste (`Keywords`), Zugangsbeschränkungen (`AccessConstraints`) und Nutzungsgebühren (`Fees`).

ServiceProvider beschreibt die den Katalog betreuende Organisation oder Institution. Dazu enthält dieser Abschnitt optional Namen, Adresse und Telefonnummer (`ServiceContact` und Subtags) eines Ansprechpartners und eine Webadresse des Betreibers (`ProviderSite`).

OperationsMetadata teilt dem Client mit, welche Abfragetypen der Server anbietet und wie diese aufgerufen werden können. Im Falle von WCS werden dort die URLs der drei Abfragetypen aufgelistet (`Operation` und Subtags). Einem Client muss somit zu Beginn der Interaktion mit dem Server nur der

Aufrufpfad für die *GetCapabilites*-Anfrage bekannt sein. Die weiteren Pfade lassen sich dann automatisch auslesen. Der WCS-Standard bietet damit die Möglichkeit, die einzelnen Abfragetypen im Extremfall von verschiedenen Servern bearbeiten zu lassen und so Last zu verteilen. Informationen über die Unterstützung optionaler Parameter können ebenfalls in diesem Abschnitt enthalten sein.

Der *Contents*-Abschnitt ist der einzige WCS-spezifische Abschnitt des Dokuments. In ihm werden alle im Katalog enthaltenen Datensätze mit einem eindeutigen Namen (`Identifier`), Titel (`Title`), Inhaltsangabe (`Abstract`), geographischer Abdeckung (`WGS84BoundingBox`), unterstützten Koordinatenreferenzsystemen (`SupportedCRS`) und unterstützten Ausgabeformaten (`SupportedFormat`) beschrieben.

Abbildung 6.2 zeigt beispielhaft einen gekürzten Ausschnitt aus einer solchen *GetCapabilities*-Antwort mit den oben beschriebenen Elementen.

DescribeCoverage

Mit den Informationen aus der *GetCapabilities*-Abfrage kann der Client in einem zweiten Schritt mit Hilfe des *DescribeCoverage*-Kommandos Details zu ausgewählten Datensätzen abrufen. Wie zuvor beschrieben handelt es sich um eine reine Metadatenanfrage, die wieder in Form eines XML-Dokuments beantwortet wird. Der Client kann dabei mit einer Abfrage prinzipiell Informationen zu beliebig vielen Datensätzen einholen, indem er eine Liste von im ersten Schritt erhaltenen `Identifiers` übermittelt.

Neben bereits im vorherigen Schritt verfügbaren Informationen (`Title`, `Identifier`, `Abstract`, `WGS84BoundingBox`) enthält das Antwortdokument eine Vielzahl weiterer Metadaten: Das zugrundeliegende Basiskoordinatenreferenzsystem (`GridBaseCRS`) mit Koordinatenursprung (`GridOrigin`) und Gitterabständen (`GridOffsets`) sowie weitere Informationen zum Koordinatengitter (`GridType`, `GridCS`) wird ebenso zurückgeliefert wie Informationen zur Zeitachse (`TemporalDomain`) und den enthaltenen Datenvariablen (*ranges*, `Range`-Tag). Innerhalb dieses `Range`-Tags können zu jeder Datenvariablen in einem `Field`-Tag Informationen zu ihrem Namen (`Identifier`, `Title`), Datentyp (`DataType`), Einheiten (`UOM`, *Unit of Measurement*) und weiteren zusätzlichen Koordinatenachsen (`Axis`) gemacht werden. Dies könnten beispielsweise verschiedene Wellenlängen sein, für die Messungen durchgeführt wurden. Im Falle der behandelten HTAP-Daten kommt als vierte Dimension in vielen Fällen eine vertikale Höhenachse hinzu, deren diskrete Koordinatenwerte innerhalb dieses Tags einzeln aufgelistet werden.

Zusätzlich sieht der Standard ein flexibles `Metadata`-Tag vor, mit dem sowohl zu ganzen Datensätzen als auch zu einzelnen Variablen weitere Meta-

daten in die Antwort eingebettet werden können. Dazu muss ein passendes Schema definiert werden, das innerhalb dieses Metadaten-Tags Verwendung findet. Im konkreten Fall wurde diese Möglichkeit zur Einbettung von Attributen aus den NetCDF-Quelldateien genutzt. Da diese Attribute durch die CF-Konventionen festgelegt sind, lassen sich so weitere standardisierte Informationen ohne Herunterladen der eigentlichen Datendateien für den Client vorab verfügbar machen.

Abbildung 6.3 zeigt die gekürzte Antwort auf eine solche *DescribeCoverage*-Anfrage.

GetCoverage

Das *GetCoverage*-Kommando dient schließlich zur eigentlichen Datenabfrage. Mit Hilfe der aus den vorherigen Anfragen erhaltenen Informationen zu verfügbaren Datensätzen und deren räumlichen, zeitlichen und weiteren Dimensionen sowie enthaltenen Datenvariablen kann der gewünschte Datenbereich nach Bedarf eingegrenzt werden, so dass nur die interessanten Teile des Datensatzes übertragen werden müssen.

Neben dem Ausschnitt können auch weitere Ausgabeparameter kontrolliert werden. Dazu zählen das Datenformat (`format`) und das Basiskoordinatenreferenzsystem (`GridBaseCRS`) sowie Koordinatenursprung (`GridOrigin`) und Gitterabstände (`GridOffsets`). Liegen an den gewünschten Stellen im Quelldatensatz keine Daten vor, so sind verschiedene Interpolationsverfahren vorgesehen, die ein Server optional unterstützen kann. Gleiches gilt für Koordinatensysteme.

Der resultierende Antwortdatensatz kann auf zwei verschiedene Weisen vom Client empfangen werden: Direkt eingebettet in die Antwort des Servers als zweiter Teil einer sogenannten *MIME Multipart Message*[26] oder über einen in der reinen XML-Antwort enthaltenen (externen) Link, von dem der Client den Datensatz im Anschluss mit einer weiteren eigenen (HTTP-)Anfrage herunterladen kann. Das gewünschte Verhalten kann über den `store`-Parameter gesteuert werden: *false* bedeutet hierbei, dass der Server eine *MIME Multipart Message* mit einem XML-Dokument als erstem und dem Datensatz als zweitem Element erzeugt. *true* führt zur Beantwortung mit einem reinen XML-Dokument, das lediglich einen Link zum Datensatz enthält. Dabei ist auf Clientseite zu beachten, dass nur die Unterstützung von `store=false` zwingend im Standard vorgeschrieben ist und somit nicht in jedem Fall von der Unterstützung von `store=true` ausgegangen werden kann. Der Vorteil von `store=false` liegt für den Server darin, dass der möglicherweise sehr umfang-

[26] `http://www.w3.org/Protocols/rfc1341/7_2_Multipart.html`

reiche Antwortdatensatz zur Auslieferung nicht notwendigerweise auf einem Datenträger gespeichert werden muss, sondern bereits während der Generierung kontinuierlich serialisiert an den Client übermittelt werden kann. Diese Technik wird auch als *streaming* bezeichnet.

4.2.4 Datenkataloge

Die vom Server bereitgestellten Daten sind einer Art Datenkatalog organisiert. Im Allgemeinen gehören in einem Katalog zusammengefasste Daten zum gleichen Themenkomplex. Die einzelnen Datensätze werden über innerhalb eines Kataloges eindeutige *Identifier* referenziert.

Die einzelnen Kataloge werden beim hier implementierten Server auf Unterverzeichnisse mit gleichem Namen innerhalb der Verzeichnisstruktur des Servers abgebildet. Jedes solche Unterverzeichnis enthält die dem Katalog zugeordneten NetCDF-Dateien, deren Namen als *Identifier* im WCS-Protokoll benutzt werden. Somit ist eine einfache Abbildung zwischen WCS-Protokoll und interner Organisation hergestellt. Neue Datenkataloge lassen sich mit diesem Konzept schnell und mit minimalen Konfigurationsaufwand dem Server hinzufügen (siehe Abschnitt 4.2.9).

Problematisch wird diese simple Aufteilung jedoch, wenn sich zusammengehörige Datensätze über mehrere Quelldateien erstrecken. Das HTAP-Datenarchiv enthält beispielsweise stündliche Simulationsergebnisse, die tageweise auf Dateien aufgeteilt wurden. Für ein Jahr existieren somit in der Regel 365 einzelne Dateien pro Modell und Szenario, was zu einer entsprechend unübersichtlichen Menge an *Identifier*n führt. Da WCS-Abfragen aber immer auf einen konkreten *Identifier* beschränkt sind, lassen sich diese Daten auch nur tageweise abfragen. Um diesem Problem zu begegnen und die Nutzung für den Benutzer einfacher zu gestalten, ist eine Erweiterung des Servers geplant, die die Zusammenfassung mehrerer Quelldateien zu einem *virtuellen Datensatz* mit einem einzigen *Identifier* erlaubt.

4.2.5 Abbildung zwischen CF-NetCDF und WCS Coverages

Ein entscheidender Schritt bei der Auslieferung von NetCDF-Daten über das WCS-Protokoll ist die Definition einer Abbildung zwischen den beiden Darstellungen. An vielen Stellen überlagern sich die beiden Konzepte stark, so dass eine Abbildung trivial zu bewerkstelligen ist. Es gibt jedoch auch einige Punkte, die nur durch weitere Einschränkungen oder Definitionen angeglichen werden können. Dieser Abschnitt gibt einen Überblick über die im Rahmen der Serverentwicklung gewonnenen Erkenntnisse in diesem Bereich.

Unabhängig von der hier erfolgten Entwicklung haben sich auch Mitglieder der GALEON-Gruppe[27] dieses Themas angenommen und sind zu ähnlichen Ergebnissen gekommen. Ihre Ergebnisse sind in einem mittlerweile beim OGC verfügbaren Diskussionspapier festgehalten [Domenico und Nativi, 2009].

Koordinatenachsen

Die in den CF-NetCDF Daten enthaltenen Dimensionen und Koordinatenvariablen lassen sich leicht auf die Koordinatenachsen von WCS-*Coverages* übertragen. Die Koordinatenvariablen der Quelldaten definieren dabei direkt Form und Ausdehnung des Koordinatengitters eines *Coverage*. Die unterschiedlichen in den CF-Konventionen definierten *grid mappings* können auf verschiedene Gittertypen abgebildet werden.

Eine Einschränkung muss dabei jedoch bei der Gitterform gemacht werden: Während CF und NetCDF irreguläre Gitter erlauben, lassen sich in WCS zunächst einmal nur reguläre Gitterstrukturen beschreiben.

Eine weitere Einschränkung betrifft die Zahl der erlaubten Koordinatensysteme: Ein CF-NetCDF Datensatz darf beliebig viele Koordinatensysteme enthalten, ein *Coverage* jedoch nur eines. Will man also NetCDF-Dateien als Coverage betrachten, so dürfen diese ebenfalls nur ein Koordinatensystem enthalten.

Ebenfalls zu beachten ist, dass ein *Coverage* immer Punkten auf der Erdoberfläche zugeordnet - also *georeferenziert* - ist, während diese Zuordnung in CF-konformen Daten nicht zwingend erforderlich ist. Dies führt zu einer Einschränkung der Verwendbarkeit von CF-NetCDF Datensätzen auf die Datenfelder, die ebenfalls georeferenziert sind.

Datenvariablen und Ranges

Die in NetCDF vorhandenen Datenvariablen können direkt auf das Konzept der *ranges* eines *Coverage* abgebildet werden. Jede Datenvariable wird dabei durch ein eigenes Feld in der *range*-Struktur repräsentiert. Die Variablennamen können als - innerhalb eines *Coverage* eindeutige - Feldbezeichner dienen, was dem aktuellen Stand der Serverimplementierung entspricht.

Eine weitere Bezeichnungsmöglichkeit bieten eventuell die in den CF-Konventionen definierten Standardnamen. Dies hätte den Vorteil der einfachen direkten Zuordnung eines *range*-Feldes und der Bedeutung der enthaltenen Daten. Ein Problem wäre jedoch dann, dass nicht notwendigerweise in allen Fällen ein Standardname existiert. Ausserdem lässt sich diese Zuordnung auch leicht erreichen, indem man den CF-Standardnamen - sofern vorhanden - in die

[27] siehe Abschnitt 2.2.1

Metadatenbeschreibung des *Coverage* integriert. Diese Methode wird derzeit benutzt.

Insgesamt ist es wahrscheinlich - ebenso wie bei CF-NetCDF - nicht ratsam, aus dem Namen eines Feldes selbst auf seine Bedeutung zu schließen. Stattdessen bieten sich die Metadaten für eine deutlich flexiblere Beschreibung an.

Ein Problem stellen in CF-konformen NetCDF-Datensätzen enthaltene Hilfsvariablen dar: Einerseits handelt es sich nicht um Koordinatenvariablen und andererseits auch nicht um Datenvariablen im eigentlichen Sinne. Von Datenvariablen lassen sie sich aber nicht syntaktisch sondern nur semantisch anhand ihrer Metadaten (Attribute) unterscheiden. Während diese Hilfsvariablen also bei Bedarf automatisch mit ausgeliefert werden sollen (siehe Abschnitt 4.2.7), sollten sie nicht als separate *range*-Felder ausgewiesen und verfügbar gemacht werden. Eine Erkennung von Bounds-Variablen ist bereits implementiert, so dass diese nicht vom WCS-Server als Datenfelder ausgewiesen werden. Eine entsprechende Klassifikation von über das `formula_terms`-Attribut referenzierten Variablen fehlt jedoch derzeit noch. Dies macht sich beispielsweise dadurch bemerkbar, dass die Koeffizienten *a* und *b* der *sigma-hybrid* Höhendarstellung wie Datenvariablen in *range*-Feldern gelistet werden.

4.2.6 Handhabung der Metadaten

Eine wesentliche Aufgabe des WCS-Servers ist die Auslieferung von die angebotenen Datensätze beschreibenden Metadaten. Dabei lassen sich im Falle der Verwendung von NetCDF-Dateien zwei verschiedene Quellen und Typen dieser Metadaten identifizieren:

- extern definiert und innerhalb eines Kataloges statisch
- in Quelldateien eingebettet und datensatzabhängig

Externe Metadaten

Extern definierte Metadaten sind beispielsweise die Informationen, die beim *Capabilities*-Dokumenttyp in den beiden Abschnitten *ServiceIdentification* und *ServiceProvider* zu finden sind (siehe Abschnitt 4.2.3). Der Inhalt von *ServiceIdentification* ist dabei in der Regel katalogabhängig, während *ServiceProvider* theoretisch auch für alle von einem Server bereitgestellten Kataloge Gültigkeit haben könnte. Aus den einzelnen Quelldateien lassen diese Metadaten nicht zuverlässig ableiten, weswegen sie über eine katalogspezifische Konfigurationsdatei (`wcs_capabilities.conf`) vom Serverbetreiber (einmalig) gesetzt werden müssen.

Datensatzabhängige Metadaten

Die meisten Metadaten stammen jedoch aus den CF-konformen NetCDF-Datensätzen selbst. Dazu zählt praktisch der gesamte Inhalt des *Contents*-Abschnittes der *GetCapabilities*-Antwort sowie der gesamte Inhalt der *DescribeCoverage*-Antwort.

Die Nutzung der NetCDF-Metadaten im WCS-Kontext wird erst dadurch ermöglicht, dass mit den CF-Konventionen ein hinreichend gutes Regelwerk existiert, das eine Abbildung zwischen dem CF-NetCDF-Konzept und WCS-*Coverages* erlaubt (siehe auch Abschnitt 4.2.5).

Um die Leistung des Servers zu verbessern, werden die für *GetCapabilities*- und *DescribeCoverage*-Anfragen benötigten Metadaten aus den Quelldaten im Voraus extrahiert und aufbereitet.

In der frühen Entwicklung wurden die XML-Antworten dazu vollständig statisch vorgeneriert und mussten somit nur noch als statisches Dokument vom Server ausgeliefert werden. Dieser Ansatz minimierte zwar die Serverbelastung, erwies sich letztendlich aber als zu unflexibel, um dem WCS-Standard gerecht zu werden. Probleme bereiteten insbesondere der `sections`-Parameter des *GetCapabilities*-Kommandos (siehe 4.2.3) und die Möglichkeit zur gleichzeitigen Abfrage mehrerer Datensätze mit dem `identifiers`-Parameter des *DescribeCoverage*-Kommandos (siehe Abschnitt 4.2.3).

Der derzeitige Ansatz ist deswegen dynamischer gestaltet: Alle relevanten Metadaten werden vorprozessiert und in einer (mehrstufigen) assoziativen Datenstruktur gesammelt. Beim Start des Servers wird diese Datenstruktur komplett geladen und erlaubt schnellen Zugriff auf alle benötigten Metadaten der einzelnen Datensätze. Die Serverantworten werden dann entsprechend den Vorgaben des Clients dynamisch aus den Metadaten mit Hilfe einer XML-Bibliothek erzeugt.

Metadatenstruktur

Abbildung 6.4 zeigt den konzeptuellen Aufbau dieser mehrstufigen aus Schlüsseln (*keys*) und Werten (*values*) bestehenden Datenstruktur: Die erste Ebene von Schlüsseln bilden die im WCS-Protokoll benutzten `Identifier` der einzelnen Datensätze. Die zu diesen Schlüsseln hinterlegten Werte sind ihrerseits wiederum eine assoziative Datenstruktur, die nun den einzelnen Datensatz näher beschreibt.

Die Beschreibung eines jeden Datensatzes enthält Schlüssel für die beiden aus der Quelldatei entnommenen Strings `Title` und `Abstract` sowie drei weitere Datenstrukturen: `axes`, `fields` und `raw_attrs`.

`axes` enthält einige Eckdaten zu den im Datensatz verfügbaren geographi-

schen (X,Y,Z) und der zeitlichen (T) Achse. Eventuelle weitere im Datensatz enthaltene Achsen werden an dieser Stelle nicht gespeichert, da sie für WCS an diesem Punkt nicht relevant sind.

Die `fields`-Struktur enthält Informationen zu allen Datenvariablen des Datensatzes. Dazu zählen zum Beispiel der Datentyp der Variable (`datatype`), ihre Einheiten (`units`) sowie der Füllwert für fehlende Daten (`null_value`). Sollte die Datenvariable dimensionale Abhängigkeiten haben, die über die horizontalen (X,Y) und zeitlichen Achsen (T) hinaus gehen, so werden diese innerhalb der dort eingebetteten separaten `axes`-Struktur näher beschrieben.

`raw_attrs` dient zur Speicherung von NetCDF-Attributen, die bei *DescribeCoverage*-Anfragen im `Metadata`-Tag der Antwort eingebettet werden (Beispiel: siehe Abbildung 6.3).

4.2.7 Datenausschnitte

Die eigentliche Bestimmung des WCS-Servers ist es, die den Vorgaben des Clients entsprechenden Geodaten auszuliefern. Dazu müssen diese aus den Quelldatensätzen nach den vom Client übermittelten Filtervorgaben (räumliche und zeitliche Bereiche, Variablen) ausgeschnitten werden.
Wesentliche Probleme, die beim Ausschneiden gelöst werden müssen, sind:

- Erhaltung aller nötigen Dimensionen und sonstigen Hilfsvariablen (CF-Konformität der Ausgabedaten)
- Behandlung von Schnitten, die zwischen Gitterpunkten verlaufen

Ausschneiden von Datenvariablen

Sollen nur bestimmte Datenvariablen aus dem Quelldatensatz ausgeschnitten werden, dann muss der Server sicherstellen, dass der Ausgabedatensatz auch alle erforderlichen Zusatzvariablen und Dimensionen enthält und somit weiterhin CF-konform ist. Neben simplen Abhängigkeiten wie denen zwischen Datenvariablen und ihren Dimensionen definieren die CF-Konventionen noch zahlreiche weitere Hilfsvariablen, die ebenfalls mit ausgeliefert werden müssen. Dies können beispielsweise zusätzliche Koordinatenvariablen (CF: *auxiliary coordinate variables*) oder Bounds-Variablen sein. Während die Beziehung zwischen Variablen und ihren Dimensionen schon im NetCDF-Format verankert ist und sich automatisch ergibt, müssen zur Identifikation weiterer Hilfsvariablen zusätzlich verschiedene (CF-)Variablenattribute berücksichtigt werden (z.B. `coordinates`, `bounds`). Derzeit kann der Server noch keine vollständige

CF-Konformität der Ausgabedaten garantieren, wenn nur einzelne Datenvariablen abgefragt werden, da diese CF-Attribute noch nicht entsprechend ausgewertet werden.

Ausschneiden von geographischen Bereichen

Verläuft der auszuschneidende Bereich nicht exakt entlang der im Datensatz enthaltenen Koordinaten, sondern zwischen zwei Werten, so stellt sich die Frage, welche Daten ein solcher geographischer Ausschnitt enthalten soll (siehe auch Abbildung 6.5). Der WCS-Standard sieht dazu verschiedene Interpolationsmöglichkeiten vor, die jedoch eher für Bilddaten gedacht sind. Eine Verfälschung von ausgelieferten wissenschaftlichen Daten durch vorzeitige Interpolation wäre kontraproduktiv, so dass eine solche Operation auf dieser Ebene nicht in Erwägung gezogen wird.

Die einfachste - und derzeit implementierte - Methode ist die Filterung des auszuschneidenden Bereiches nach den in diesem Gebiet enthaltenen Zellmittelpunkten. Eine weitere - aber kompliziertere - Methode wäre die Zuhilfenahme von Bounds-Variablen (sofern vorhanden) und somit Einbeziehung aller von der Abfrage berührten Gitterzellen - auch wenn der Zellmittelpunkt ausserhalb des abgefragten Bereiches liegt. Im Allgemeinen dürfte die Annahme gerechtfertigt sein, dass der Benutzer alle von seiner Abfrage berührten Zellen erhalten möchte, da die dem Mittelpunkt zugeordneten Daten für die gesamte Zelle Gültigkeit haben. Dieses Verhalten ist auch in sofern konsistenter, als dass der abgefragte Bereich garantiert vollständig im Ergebnisbereich enthalten ist. Bei Filterung nach Zellmittelpunkten ist dies nicht zwangsläufig der Fall.

Abbildung 6.5 verdeutlicht den Unterschied anhand eines Beispiels: Im oberen Fall werden nur die im Abfragebereich enthaltenen Zellmittelpunkte berücksichtigt. Dies führt zu einem Ergebnisbereich, der nicht alle Teile des ursprünglichen Abfragebereiches abdeckt. Im unteren Fall ist garantiert, dass der abgefragte Bereich vollständig in den ausgeschnittenen Daten enthalten ist.

4.2.8 Programmtechnische Umsetzung

Der Programmcode selbst besteht - mit Ausnahme eines in C++ geschriebenen NetCDF-Moduls (`nc3`) für den Betrieb unter Windows - ausschließlich aus Python-Code. Er nutzt darüber hinaus zusätzliche Module von Drittanbietern. Dazu zählen das Webframework *web.py*[28], das Numerikpaket *NumPy*[29], die

[28] `http://webpy.org/`
[29] `http://numpy.scipy.org/`

NetCDF-Bibliothek *PyNIO*[30] sowie die XML-Bibliothek *lxml*[31]. Abbildung 4.1 gibt einen Überblick über die grundlegende Datei- und Verzeichnisstruktur des Server-Codes.

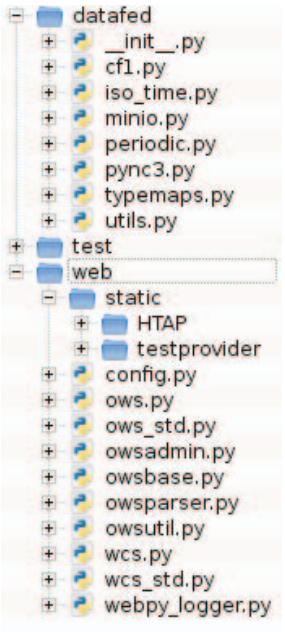

Abbildung 4.1: Datei- und Verzeichnisstruktur des WCS-Servers

Wie die Abbildung zeigt, ist der Code in drei wesentliche Unterverzeichnisse unterteilt: `datafed`, `test` und `web`. Inhalt und Funktionalität von `datafed` und `web` werden in den folgenden beiden Abschnitten näher beschrieben, das `test`-Verzeichnis enthält Unit-Tests für fast alle Module und Funktionen, die sich in den beiden anderen Verzeichnissen befinden.

Das datafed-Paket

Das `datafed`-Verzeichnis ist als so genanntes Paket (*package*) organisiert[32] und enthält allen Code, der direkt mit den NetCDF-Dateien interagiert oder der eine Bibliotheksfunktion erfüllt. Der Name „datafed“ geht auf den Beginn

[30] `http://www.pyngl.ucar.edu/Nio.shtml`
[31] `http://codespeak.net/lxml/`
[32] bei Python erkennbar an der `__init__.py`-Datei

der Serverentwicklung im Rahmen des *DataFed*[33]-Webservice der *Washington University in St. Louis* zurück. Seit Beginn dieser Arbeit wurden jedoch große Codeteile grundlegend überarbeitet.

Insgesamt besteht das Paket derzeit aus sieben Modulen mit folgenden Funktionalitäten:

- `cf1`: auf die Python-NetCDF-Schnittstelle aufsetzende Zwischenschicht zur Vereinfachung des lesenden und schreibenden Zugriffs auf CF-konforme Dateien; beinhaltet auch Funktionalität zur Klassifikation von NetCDF-Variablen (Koordinatenvariablen, Datenvariablen und Hilfsvariablen wie z. B. Bounds-Variablen)

- `iso_time`: Parser für das ISO-Zeitformat nach ISO8601[34] und CF-konforme Zeitnotation (`units`-Attribut mit Referenzzeit und -Einheiten plus Offsets in Koordinatenvariable, siehe auch Abschnitt 2.1.4)

- `minio`: zusammen mit dem in C++ geschriebenen `nc3`-Modul Prototyp einer Zwischenschicht zur Imitation der wesentlichen Teile von PyNIO unter Windows, da PyNIO Windows derzeit nicht unterstützt

- `periodic`: experimentelles Modul zur Filterung gegebener Zeitpunkte nach Kriterien wie Monat, Wochentag oder Stunde

- `pync3`: vom WCS-Server benutzte Schnittstelle zum Ausschneiden des Ergebnisdatensatzes aus den Quelldaten; ursprünglich Python-Version des für Windows in C++ entwickelten `nc3`-Moduls

- `typemaps`: enthält Abbildungen zwischen schnittstellenspezifischen Notationen der Basisdatentypen (PyNIO, NetCDF C API, etc.)

- `utils`: Modul mit Helferfunktionen, die teilweise aus der im Rahmen des *CFchecker* entwickelten CF-Basisbibliothek stammen (siehe auch Abschnitt 3.1.2)

Das web-Verzeichnis

Das `web`-Verzeichnis enthält den Programmcode für den Serverbetrieb, der in verschiedene Module aufgeteilt ist:

[33] `http://datafed.net/`

[34] `http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=40874`

- `config`: globale Konfigurationsparameter des Servers, die teils vom Serverbetreiber nach Bedarf angepasst werden können; Einstellungen zur Behandlung von Log-Dateien durch den Server (Logverzeichnis, Rotation von Logdateien, etc.) sowie Pfade zu den auszuliefernden Daten und temporären Verzeichnissen
- `ows`: Hauptmodul des Servers; stellt die Web-Schnittstelle bereit, nimmt Clientanfragen entgegen und delegiert sie an die entsprechenden Servicemodule; unterstützt durch modulares Konzept auch weitere OGC Web Services (OWS), sofern Module für diese bereitstehen
- `ows_std`: enthält die generische Basisklasse für alle OWS-Protokolle
- `owsadmin`: vom Servermodul unabhängiges Kommandozeilenprogramm, das Metadaten aus den einzelnen Quelldateien extrahiert und daraus die vom Server benötigte Metadatenstruktur zusammenstellt
- `owsbase`: enthält die Caching-Klasse für Serverantworten aller OWS-Protokolle und die Basisklasse für die Anfragebearbeitung
- `owsparser`: enthält den generischen Parser für per HTTP GET übertragene Parameter mit allgemeiner Unterstützung von OWS-Protokollen und einen WCS-spezifischen Parser für den `RangeSubset`-Parameter
- `owsutil`: weitere Hilfsfunktionen und -Klassen (Exceptions, Cache-Verwaltung, ...) sowie Konstanten (XML-Namensräume, Fehlermeldungen, ...) für den Serverbetrieb
- `wcs`: Servicemodul zur Behandlung von WCS-Anfragen, wird durch das `ows`-Modul eingebunden
- `wcs_std`: implementiert Standardfunktionalität für das WCS-Protokoll (Initialisierung von Namespaces, Generierung der Metadatendokumente, ...)
- `webpy_logger`: implementiert Funktionalität zur Steuerung des Logging-Verhaltens des *web.py*-Frameworks (Ausgabe in Datei statt Konsole, Logrotation)

Zusätzlich zu diesen Modulen enthält das Verzeichnis außerdem das Unterverzeichnis `static`, das im Serverbetrieb sowohl die Quelldaten als auch ein Cache-Verzeichnis für die Ausgabedaten enthält. Die gewählten Verzeichnisnamen - im Beispiel (Abbildung 4.1) `HTAP` und `testprovider` - definieren automatisch auch die Katalognamen des Servers.

Der Katalog `testprovider` wird zusammen mit dem Servercode zum Herunterladen bereitgestellt und dient der einfachen Verifikation der Serverfunktionalität. Für einige der automatisierten Unit-Tests innerhalb des Entwicklungsprozesses wird er ebenfalls benutzt. Im Auslieferungszustand enthält der Katalog keine NetCDF-Quelldateien, sondern lediglich ein weiteres Python-Script (`create_all.py`), das die benötigten Testdaten auf dem Zielsystem generieren kann.

4.2.9 Betrieb der Serversoftware

Dieser Abschnitt soll die benötigten Schritte zur Inbetriebnahme des WCS-Servers auf einem Linux-System kurz beschreiben. Als Betriebssystem kann prinzipiell jede Linux-Distribution dienen. Zur in der vorliegenden Arbeit durchgeführten Entwicklung kam die Distribution Debian[35] in Version 5.0 (Codename „*lenny*“) zum Einsatz.

Dieses Beispiel beschränkt sich exemplarisch auf die Publikation des `testprovider`-Kataloges, dessen Konfiguration sich jedoch nicht grundsätzlich von anderen Datenkatalogen unterscheidet.

Softwarevoraussetzungen

Grundvoraussetzung für die Ausführung des Programmcodes ist der Python-Interpreter. Getestet wurde der derzeitige Entwicklungsstand mit Version 2.6.6, Versionen älter als 2.6 werden nicht unterstützt. Der Schritt auf das neue Python 3 kann erst erfolgen, nachdem alle nötigen Bibliotheken dieses ebenfalls unterstützen, was derzeit noch nicht der Fall ist.

Zusätzlich werden folgende Python-Pakete bzw. -Bibliotheken benötigt:

- web.py (`http://webpy.org`)
- lxml (`http://codespeak.net/lxml`)
- NumPy (`http://numpy.scipy.org/`)
- PyNIO (`http://www.pyngl.ucar.edu/Nio.shtml`)
- WSGILog (`http://pypi.python.org/pypi/wsgilog/`

Diese können entweder über die Paketverwaltung der Distribution oder über die *setuptools*-Paketverwaltung von Python[36] installiert werden.

[35] `http://debian.org/`
[36] `http://pypi.python.org/pypi/setuptools`

Die aktuelle Version der Serversoftware für Linux ist im Downloadbereich der Projektseite[37] verfügbar (`linux-ows-*.tar.gz`). Sie kann in ein beliebiges Verzeichnis entpackt werden, eine Installation im eigentlichen Sinne ist nicht erforderlich. In diesem Beispiel wird angenommen, dass der Inhalt des Pakets nach `/usr/local/OWS` entpackt wurde:

```
mkdir -p /usr/local/OWS
cd /usr/local/OWS
tar xzf /path/to/linux-ows-*.tar.gz
```

Konfiguration

Zunächst muss der Installationspfad dem Python-Suchpfad hinzugefügt werden, damit das `datafed`-Paket eingebunden werden kann. Dies lässt sich über die Umgebungsvariable `PYTHONPATH` bewerkstelligen:

```
export PYTHONPATH=$PYTHONPATH:/usr/local/OWS
```

Anschließend sollte - sofern noch nicht vorhanden - das Log-Verzeichnis für die Serverlogs angelegt werden:

```
mkdir /var/log/ows
```

Bei Bedarf können einige Logging-Einstellungen in der globalen Serverkonfiguration in `web/config.py` angepasst werden. Die einzelnen Einstellungen werden dort durch Kommentare erläutert.

Zur Konfiguration des `testprovider`-Kataloges sollte dort zumindest eine „WCS-konforme“ NetCDF-Datei hinterlegt werden. Für einen ersten Test können einige Beispieldateien mithilfe des mitgelieferten Scripts erzeugt werden:

```
cd /usr/local/OWS/web/static/testprovider
python create_all.py
```

Statische Kataloginformationen können in der beiliegenden Katalogabhängigen Konfigurationsdatei `wcs_capabilities.conf` vorgenommen werden. Änderungen sind für einen Test jedoch nicht zwingend erforderlich.

Die datensatzabhängigen Metadaten müssen nach Erzeugung bzw. Plazierung der NetCDF-Dateien im Katalogverzeichnis durch Aufruf von `owsadmin.py` erzeugt werden:

```
cd /usr/local/OWS/web
python owsadmin.py wcs_prepare testprovider
```

Bei jeder Änderung an den Quelldaten eines Kataloges - entweder durch Löschen vorhandener bzw. Hinzufügen neuer Dateien oder durch Modifikation einer be-

[37] `http://sf.net/projects/aq-ogc-services/files/`

reits vorhandenen Datei - muss dieser Schritt erneut ausgeführt werden, um das Metadatenverzeichnis auf den neuesten Stand zu bringen.

Im letzten Schritt muss der Server gestartet werden, dies geschieht durch Aufruf des `ows`-Moduls:

```
cd /usr/local/OWS/web
python ows.py 1080
```

Falls beim Start kein entsprechendes Argument übergeben wird, so wird der WCS-Server standardmäßig auf dem TCP-Port 8080 gestartet. Im obigen Beispiel wird Port 1080 verwendet. Auf dem lokalen Rechner ist der Server danach unter `http://localhost:1080` erreichbar. Für die Erreichbarkeit aus dem lokalen Netzwerk oder Internet müssen gegebenenfalls Firewall-Regeln angepasst werden.

Administratorrechte sind zum Betrieb im Allgemeinen nicht erforderlich, sofern Lese- und Schreibberechtigungen auf das Serververzeichnis für den verwendeten Benutzeraccount erteilt werden. Es gilt zusätzlich zu bedenken, dass der Server ohne `root`-Rechte auf gängigen Linux-Systemen nur auf Portnummern ab einschließlich 1024 betrieben werden kann.

Menschenlesbare Informationsseiten

Für den Produktivbetrieb ist es ratsam, einige menschenlesbare Informationen über den angebotenen Dienst in Form von HTML-Dateien auf dem Server zu hinterlegen. Dazu sind zwei verschiedene Stellen vorgesehen: global für den gesamten Server und auf Ebene der einzelnen Datenkataloge.

Die globale Serverinformation kann in der Datei `web/static/index.html` untergebracht werden. Diese Seite wird automatisch angezeigt, wenn auf den Wurzelpfad des Servers zugegriffen wird - im obigen Beispiel `http://localhost:1080` - und somit kein Datenkatalog selektiert wurde. Ein wichtiger Zweck dieser Seite kann sein, Benutzer über die vorhandenen Datenkataloge zu informieren. Der WCS-Standard selbst bietet keine Möglichkeit zur Abfrage aller verfügbaren Kataloge eines Servers, so dass diese Information über andere Wege transportiert werden muss.

Um detailiertere Informationen über einen einzelnen Datenkatalog bereitzustellen, können diese in einer `index.html`-Datei im Katalogverzeichnis hinterlegt werden, im Falle des *testprovider*-Kataloges ist dies `web/static/testprovider/index.html`. Diese Datei wird vom Server automatisch angezeigt, wenn ohne weitere Parameter auf den Katalogpfad zugegriffen wird, beispielsweise `http://localhost:1080/testprovider`.

Um die Übernahme von Datenquellen in Clients für den Benutzer zu erleichtern, bietet es sich an, auf diesen Informationsseiten Links für die *GetCa-*

pabilities-Anfragen des oder der betroffenen Kataloge zu plazieren. Mit dieser Information kann ein WCS-Client dann selbst alle weiteren benötigten Informationen zum gewählten Katalog über das WCS-Protokoll abfragen.

4.3 Zusammenfassung und Ausblick

Der entwickelte WCS-Server kann die gestellten Aufgaben in seinem derzeitigen Zustand bewältigen. Dazu wurde die obligatorische Funktionalität des WCS-Standards implementiert und um die Möglichkeit der Nutzung und Auslieferung des NetCDF-Datenformats erweitert. Der Server ist somit in der Lage, die HTAP-Datensätze über eine Web-Schnittstelle so WCS-konform wie möglich bereitzustellen. Die Einschränkung der WCS-Konformität kommt durch die Verwendung von NetCDF zustande, das derzeit noch nicht offiziell unterstützt wird.

Es besteht jedoch noch weiterer Entwicklungsbedarf, um den WCS-Server in einem breiteren Anwendungsgebiet einsetzten zu können: Viele (optionale) Teile des WCS-Standards sind noch nicht oder nicht vollständig implementiert worden. Dazu zählt beispielsweise die Unterstützung für andere Koordinatenreferenzsysteme (CRS). Seit Beginn der Entwicklung ist zudem die WCS Standardversion 2.0 veröffentlicht worden, die mittelfristig ebenfalls vom Server unterstützt werden sollte.

Um den Umgang mit dem NetCDF-Format im Kontext von WCS weiter zu verbessern, ist ein Austausch mit der GALEON-Gruppe vorgesehen.

Die Ausgabedaten sind derzeit noch nicht unter allen Umständen CF-konform und die Ausschneidemethoden für horizontale Gitter sollten, wie in Abschnitt 4.2.7 beschrieben, durch Einbeziehung der Zellgrenzen (Bounds-Variablen) verbessert werden. Es wird ebenfalls die Möglichkeit diskutiert, in Zukunft Bildformate als Ausgabeformate zu unterstützen und so den Anwendungsbereich des Servers weiter zu vergrößern, dazu wird auch die Verwendung des weiter verbreiteten WMS-Protokolls erwogen.

Neben der vollständigen Implementierung des Standards gibt es auch am Servercode selbst noch erhebliches Verbesserungspotential: Die zusammen mit dem `CFchecker` entwickelte Basisbibliothek bietet insgesamt viel Funktionalität, von der auch der WCS-Server profitieren kann. Derzeit ist einiger Code sowohl in der Bibliothek als auch direkt im Server vorhanden, was dessen Wartbarkeit verschlechtert. In Zukunft ist deswegen geplant, gemeinsame Funktionalität stärker in eine solche zusätzliche Bibliothek auszulagern und auch konsequent zu benutzen. Ebenso muss noch ein Weg zur Auslieferung von Stationsdaten, wie sie beispielsweise im *verprof*-Datentyp der HTAP-Datensätze enthalten sind, gefunden werden. Da diese Dateien keine regulären Gitter son-

dern nur einzelne irreguläre Koordinatenpunkte enthalten, ist eine Umsetzung innerhalb des WCS-Konzepts nicht ohne weiteres zu erreichen. Möglicherweise muss hierzu auf einen anderen OGC Standard zurückgegriffen werden.

Nachdem nun eine gewisse stabile Grundlage geschaffen worden ist, muss der aus der experimentellen Entwicklungsphase hervorgegangene Quelltext zudem zu einem großen Teil besser geordnet und neu dokumentiert werden, um ihn auch zukünftig wartbar zu halten.

Die Benutzbarkeit des Servers durch den Betreiber könnte ebenfalls noch weiter verbessert werden. So ist beispielsweise eine weitgehend automatische Publikation neuer Datensätze denkbar, wenn der Server Änderungen an Datensätzen dynamisch erkennen könnte und seine Metadaten-Datenbank automatisch aktualisieren würde. Der manuelle Einsatz von `owsadmin` (siehe Abschnitt 4.2.9) würde damit überflüssig. Wie in Abschnitt 4.2.4 beschrieben sollte zudem das Problem der sich auf viele NetCDF-Quelldateien erstreckenden - aber eigentlich zusammengehörenden - Datensätze durch Zusammenfassung zu *virtuellen Datensätzen* gelöst werden. Dies würde die Bedienung sowohl für den Serverbetreiber als auch für den Nutzer erleichtern.

Die Metadaten-Datenbank selbst bietet ebenfalls noch Optimierungspotential: Einige Informationen werden mehrfach gespeichert und andere haben sich für den Betrieb als irrelevant herausgestellt. Denkbar ist ebenfalls das Hinzufügen weiterer oft benutzter Informationen zur Verbesserung der Serverleistung. Bei umfangreichen Datenkatalogen können aber trotz solcher Optimierung noch erhebliche Mengen an Metadaten anfallen. Dafür muss ein speicherschonenderer Weg zum Zugriff auf die Metadaten gefunden werden, als die derzeitige Implementierung, die die gesamte Datenstruktur beim Serverstart komplett in den Hauptspeicher lädt. Eine Lösungsmöglichkeit bietet das in der Python-Standardbibliothek enthaltene `shelve`-Modul[38], das Daten bei Bedarf transparent vom Datenträger nachlädt und die gleiche Programmierschnittstelle wie die derzeitige Datenstruktur zur Verfügung stellt.

[38] `http://docs.python.org/library/shelve.html`

Kapitel 5

Fazit

5.1 Zusammenfassung der Ergebnisse

In dieser Arbeit wurden verschiedene Aspekte zur Verbesserung des automatisierten Vergleichs von Beobachtungs- und Modelldaten anhand georeferenzierter Atmosphärendaten aus dem HTAP-Modellvergleich vorgestellt, entwickelt und umgesetzt. Sie knüpft damit an internationale Bestrebungen zur Verbesserung und Vereinfachung des offenen wissenschaftlichen Datenaustauschs an, wie sie zum Beispiel mit GEOSS verfolgt werden.

5.1.1 Verarbeitung von Datensätzen

Mit `CFchecker` wurde ein Werkzeug entwickelt, das die CF-Konformität von NetCDF-Datensätzen prüfen kann und so den manuellen Arbeitsaufwand stark reduziert. Es ist sowohl als Kommandozeilenwerkzeug als auch als Bibliothek zur Integration in andere Arbeitsabläufe geeignet. Bei der Entwicklung des `CFchecker` und der nachfolgenden Untersuchung der HTAP-Datensätze hat sich gezeigt, dass die in den CF-Konventionen festgeschriebenen Regeln nicht ausreichen, um alle häufig gemachten Fehler zu erkennen. Dies liegt auch daran, dass bestimmte Fehler zu nicht zuverlässig zu schließenden Lücken in der Semantik eines Datensatzes führen. So kann zum Beispiel ein Datensatz als CF-konform akzeptiert werden, weil die Verbindung zwischen einer Koordinatenvariable und ihrer fehlerhaft definierten Bounds-Variable fehlt und diese Beziehung somit nicht zuverlässig automatisch hergestellt werden kann. Die fehlerhafte Definition bleibt dann möglicherweise unentdeckt.

Zur Verarbeitung der CF-konformen NetCDF-Datensätze wurden verschiedene Verarbeitungs- und Vergleichswerkzeuge entwickelt, mit denen sich Datensätze auf verschiedene Art weiterverarbeiten lassen. Um die Vergleichbarkeit von möglichst vielen Daten aus unterschiedlichen Quellen zu verbessern,

mussten zusätzlich horizontale und vertikale Interpolationsoperatoren entwickelt und getestet werden. Alle diese Werkzeuge wurden zum Programmpaket `CFdatatools` zusammengefasst.

Das Problemfeld der horizontalen Interpolation konnte dabei bis zu einer praxistauglichen Lösung bearbeitet werden. Insbesondere konnte gezeigt werden, dass bei Verfeinerungen der horizontalen Gitterstrukturen keine großen Verfälschungen der Datenfelder zu erwarten sind. Im Bereich der vertikalen Interpolation konnte gezeigt werden, dass bei hinreichend hoher Auflösung und gut gewählten Schichten auch mit einer simplen Methode zu Bestimmung der Schichtgrenzen bereits vielversprechende Interpolationsergebnisse für die drei untersuchten Spurenstoffe erreicht werden können. Es besteht jedoch noch weiterer Entwicklungsbedarf bei der Wahl der Interpolationsschichten und insbesondere ihrer Grenzen. Eine Verbesserung könnte die nötige Zahl von Höhenschichten reduzieren und somit das Gesamtvolumen der Datensätze reduzieren, was angesichts der großen zu verarbeitenden und zu speichernden Datenmengen helfen würde. Auch die universelle Anwendbarkeit der vertikalen Interpolation auf andere wisschenschaftlich relevante Größen (Mischungsverhältnisse anderer Stoffe, Temperaturen, Feuchten, ...) muss noch näher untersucht werden. Eine Extrapolation der vertikalen Daten hat sich nur in Ausnahmefällen als praktikabel herausgestellt. Dies betrifft vor allem die Anwendung auf Ozonkonzentrationen und Stoffe mit vergleichbaren Höhenprofilen. In den meisten Fällen kommt es zu erheblichen Verfälschungen von sowohl Säulendichte als auch globaler Masse.

5.1.2 Auslieferung von Datensätzen mit WCS

Der entwickelte WCS-Server ist in der Lage, die gestellten Aufgaben zu bewältigen. Er unterstützt die wichtigsten obligatorischen Funktionen des WCS-Standards und ist stabil genug, um zur Bereitstellung der HTAP-Datensätze benutzt zu werden. Es konnte gezeigt werden, dass sich das WCS-Protokoll auch zur Auslieferung von Daten im NetCDF-Format eignet und nicht auf die bisher standardisierten Rasterbildformate beschränkt ist. Dazu wurde eine Abbildung zwischen dem Konzept der *WCS-Coverages* und dem NetCDF-Format hergestellt und auch deren Grenzen aufgezeigt.

Die Zugänglichmachung von vorhandenen NetCFD-Datensätzen über den WCS-Server wurde durch eine einfache Abbildung des WCS-Katalog- und Identifier-Konzeptes auf Verzeichnisse und Dateien erreicht, die nur wenig Konfigurationsaufwand erfordert.

5.2 Ausblick

Um mehr der häufig in den HTAP- und anderen als CF-konform ausgelegten Datensätzen beobachteten Probleme in Zukunft besser entdecken zu können, ist ein unabhängig von `CFchecker` einsetzbares Prüfwerkzeug geplant, das anhand von unschärfer definierten heuristischen Kriterien und Indizien auf mögliche Probleme zur genaueren Untersuchung hinweist.

An den einzelen Werkzeugen des `CFdatatools`-Pakets können ebenfalls noch zahlreiche Verbesserungen vorgenommen werden: `mean` und `diff`, die dem Vergleich mehrerer Datensätze dienen, könnten durch Verwendung des `standard_name` Attributs flexibler die zu vergleichenden Datenvariablen finden und somit auch Datensätze aus unterschiedlichen Quellen zuverlässiger vergleichen. Bei der Berechnung von globaler Masse und Säulendichte werden derzeit nur die zwei in dieser Arbeit benutzten Höhendarstellungen unterstützt. Diese Unterstützung könnte auf weitere Darstellungen wie zum Beispiel die *sigma*-Darstellung ausgebaut werden, um den Einsatzbereich zu vergrößern.

Neben der bereits implementierten obligatorischen Funktionalität sind zahlreiche optionale Parameter und Funktionen im WCS-Standard definiert, von denen zumindest eine Teilmenge noch implementiert werden sollte, um das Anwendungsgebiet des Servers zu vergrößern. So beschreibt der WCS-Standard beispielsweise die Unterstützung mehrerer Koordinatenreferenzsysteme und Gitterformen. Mit WCS 2.0 ist außerdem seit Beginn der Serverentwicklung eine neue Standardversion erschienen, die mittelfristig ebenfalls unterstützt werden sollte.

Nachdem die WCS-Server-Software in einen hinreichend stabilen Zustand erreicht hat, soll diese auch in weiteren Einrichtungen im Bereich der Atmosphärenforschung und Erdbeobachtung zum Einsatz kommen und so zu einer tatsächlichen Vereinfachung bei der Arbeit mit Datensätzen aus verschiedenen Quellen führen. Sowohl einige europäische als auch amerikanische Einrichtungen haben bereits ihr Interesse an der Software bekundet. Der Fokus wird sich dabei zwangsläufig von den in dieser Arbeit im Mittelpunkt stehenden HTAP-Datensätzen entfernen und zu einer allgemeineren Ausrichtung des Servers führen.

Kapitel 6

Listings und Abbildungen

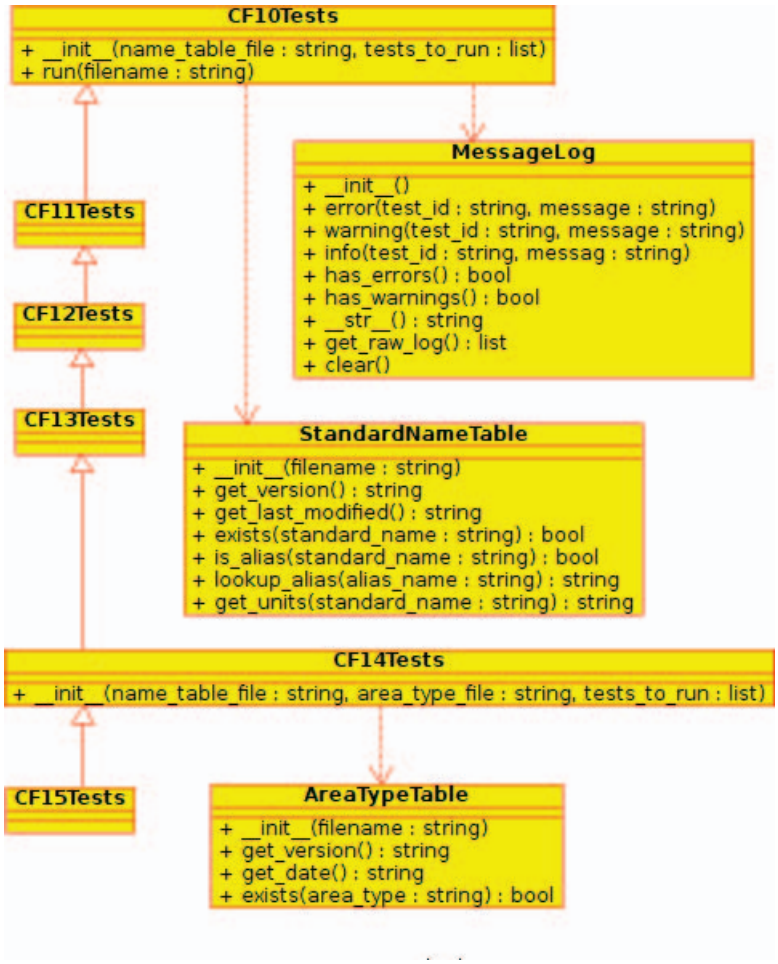

Abbildung 6.1: vereinfachte UML-Darstellung der CFchecker-Klassen: nur `CF10Tests` implementiert die gesamten CF-1.0 Konventionen, alle weiteren Testklassen erben ihre Basisfunktionalität von der jeweils vorherigen Version und erweitern diese nur noch um die erfolgten Modifikationen

```
<Capabilities xmlns:xlink=[...] xmlns:ows=[...] xmlns=[...]
   version="1.1.2">
 <ows:ServiceIdentification>
  <ows:Title>[...] Data Archive: monthly data</ows:Title>
  <ows:Abstract>The TFHTAP Model [...]</ows:Abstract>
  <ows:Keywords>
    [...]
  </ows:Keywords>
  <ows:ServiceType>OGC WCS</ows:ServiceType>
  <ows:ServiceTypeVersion>1.1.2</ows:ServiceTypeVersion>
  <ows:Fees>NONE</ows:Fees>
  <ows:AccessConstraints>NONE</ows:AccessConstraints>
 </ows:ServiceIdentification>
 <ows:ServiceProvider>
  <ows:ProviderSite xlink:href=[...]/>
  <ows:ServiceContact> [...] </ows:ServiceContact>
 </ows:ServiceProvider>
 <ows:OperationsMetadata>
  [...]
  <ows:Operation name="GetCoverage">
   <ows:DCP> <ows:HTTP>
    <ows:Get xlink:href="http://htap.icg.kfa-juelich.
       de:58080/HTAP_monthly?"/>
   </ows:HTTP> </ows:DCP>
   <ows:Parameter name="store">
    <ows:AllowedValues>
     <ows:Value>false</ows:Value>
     <ows:Value>true</ows:Value>
    </ows:AllowedValues>
   </ows:Parameter>
  </ows:Operation>
 </ows:OperationsMetadata>
 <Contents>
  <CoverageSummary>
   <ows:Title>[...] TFHTAP Base case simulation for year
      2001</ows:Title>
   <ows:Abstract></ows:Abstract>
   <ows:WGS84BoundingBox crs="urn:ogc:def:crs:OGC:2:84">
    <ows:LowerCorner>+0.0000 -87.8638</ows:LowerCorner>
    <ows:UpperCorner>+357.1875 +87.8638</ows:UpperCorner>
   </ows:WGS84BoundingBox>
   <SupportedCRS>urn:ogc:def:crs:OGC:2:84</SupportedCRS>
   <SupportedFormat>application/x-netcdf</SupportedFormat>
   <Identifier>MOZECH-v16_SR1_tracerm_2001</Identifier>
  </CoverageSummary>
  <CoverageSummary>
   [...]
   <Identifier>MOZECH-v16_SR2_tracerm_2001</Identifier>
  </CoverageSummary>
 </Contents>
</Capabilities>
```


Abbildung 6.2: Gekürzte Beispielantwort auf eine *GetCapabilities*-Anfrage (`http://htap.icg.kfa-juelich.de:58080/HTAP_monthly?service=WCS&acceptversions=1.1.2&Request=GetCapabilities`)

```
<CoverageDescriptions xmlns:cf=[...] [...]>
 <CoverageDescription>
  <ows11:Title>[...]</ows11:Title>
  <ows11:Abstract></ows11:Abstract>
  <Identifier>MOZECH-v16_SR1_tracerm_2001</Identifier>
  <ows11:Metadata> <cf:CoverageMetadata>
    <cf:Conventions>CF-1.0</cf:Conventions>
    [...]
  </cf:CoverageMetadata> </ows11:Metadata>
  <Domain> <SpatialDomain>
    <ows11:WGS84BoundingBox crs="urn:ogc:def:crs:OGC:2:84">
     [...]
    </ows11:WGS84BoundingBox>
    <GridCRS>
     <GridBaseCRS>urn:ogc:def:crs:OGC:2:84</GridBaseCRS>
     [...]
     <GridOrigin>+0.0000 -87.8638</GridOrigin>
     <GridOffsets>+2.8125 +2.7893</GridOffsets>
    </GridCRS>
   </SpatialDomain>
   <TemporalDomain>
    <gml:timePosition>2001-01-16T12:00:00Z</gml:timePosition
       >
    [...]
  </TemporalDomain> </Domain>
  <Range>
   <Field>
    <ows11:Title>CO</ows11:Title>
    <Identifier>vmr_co</Identifier>
    <Definition>
     <ows11:DataType>float</ows11:DataType>
     <ows11:UOM>1</ows11:UOM>
     <ows11:Metadata> <cf:FieldMetadata>
       <cf:standard_name>
          mole_fraction_of_carbon_monoxide_in_air</
          cf:standard_name>
       [...]
    </cf:FieldMetadata> </ows11:Metadata> </Definition>
    <Axis identifier="lev">
     <AvailableKeys>
      <Key>0.99614071846</Key>
      [...]
     </AvailableKeys>
     [...]
    </Axis>
   </Field>
   [...]
  </Range>
  <SupportedCRS>urn:ogc:def:crs:OGC:2:84</SupportedCRS>
  <SupportedFormat>application/x-netcdf</SupportedFormat>
 </CoverageDescription>
</CoverageDescriptions>
```


Abbildung 6.3: Gekürzte Beispielantwort auf eine *DescribeCoverage*-Anfrage (`http://htap.icg.kfa-juelich.de:58080/HTAP_monthly?service=WCS&version=1.1.2&Request=DescribeCoverage&identifiers=MOZECH-v16_SR1_tracerm_2001`)

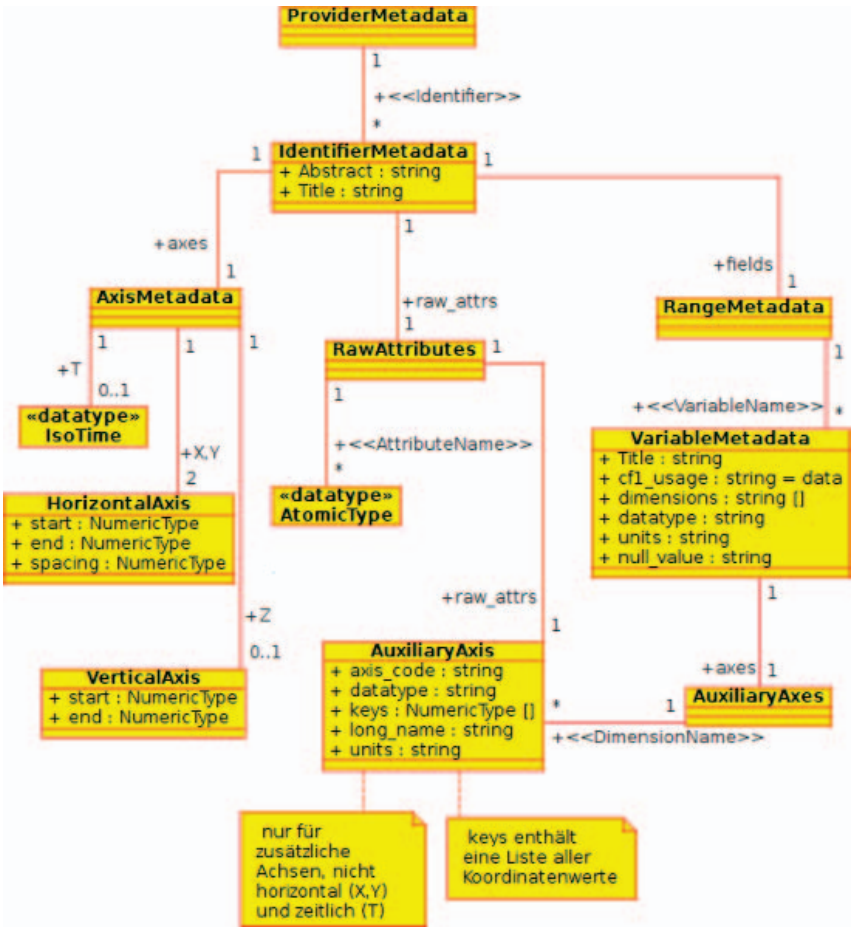

Abbildung 6.4: UML-Darstellung der vom WCS-Server benutzten Metadatenstruktur zur Beantwortung der Metadatenanfragen (*GetCapabilities*, *DescribeCoverage*)

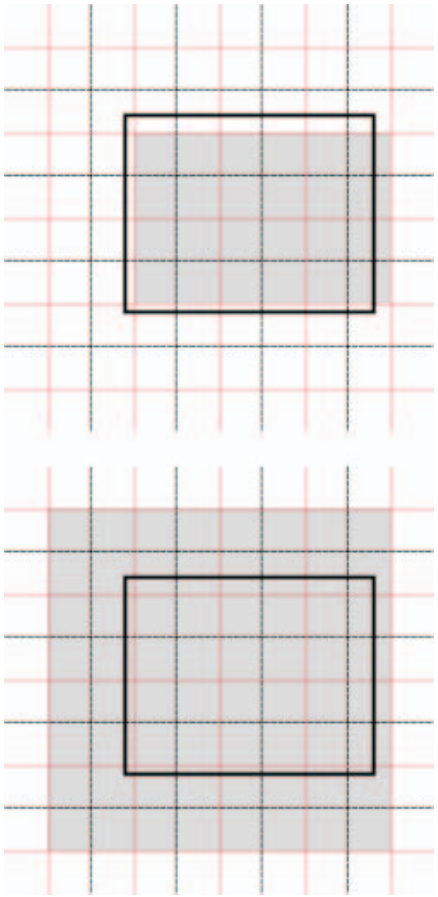

Abbildung 6.5: Illustration von abgefragtem und tatsächlich zurückgeliefertem Gitterzellenbereich in Abhängigkeit von der verwendeten Extraktionsmethode
oben: Wahl aller Zellmittelpunkte, die vom Abfragebereich eingeschlossen werden
unten: Wahl des Ergebnisses unter Berücksichtigung der Zellgrenzen so, dass alle von der Abfrage berührten Zellen enthalten sind
schwarz gestrichelte Linien: Koordinatengitter der Quelldaten
rote Linien: Zellgrenzen
schwarz umrandet: Abfragebereich
grau schraffiert: extrahierter Bereich

Literaturverzeichnis

[HTAP2007 2007] Hemispheric transport of air pollution 2007 : interim report / prepared by the Task Force on Hemispheric Transport of Air Pollution acting within the framework of the Convention on Long-range Transboundary Air Pollution. United Nations, New York ; Geneva, 2007. – Book, Online. – URL http://www.htap.org/activities/2007_Interim_Report.htm. – ISBN 9789211169843 9211169844

[Domenico und Nativi 2009] Domenico, B. ; Nativi, S.: Web Coverage Service (WCS) 1.1 extension for CF-netCDF 3.0 encoding. (2009), Nr. OGC 09-018. – URL http://portal.opengeospatial.org/files/?artifact_id=32195. – Open Geospatial Consortium Discussion Paper

[Eaton u. a. 2010] Eaton, B. ; Gregory, J. ; Drach, B. ; Taylor, K. ; Hankin, S.: *NetCDF Climate and Forecast (CF) Metadata Conventions – Version 1.5.* October 2010. – URL http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.5/cf-conventions.pdf

[Fiore u. a. 2009] Fiore, A. M. ; Dentener, F. J. ; Wild, O. ; Cuvelier, C. ; Schultz, M. G. ; Hess, P. ; Textor, C. ; Schulz, M. ; Doherty, R. M. ; Horowitz, L. W. ; MacKenzie, I. A. ; Sanderson, M. G. ; Shindell, D. T. ; Stevenson, D. S. ; Szopa, S. ; Dingenen, R. V. ; Zeng, G. ; Atherton, C. ; Bergmann, D. ; Bey, I. ; Carmichael, G. ; Collins, W. J. ; Duncan, B. N. ; Faluvegi, G. ; Folberth, G. ; Gauss, M. ; Gong, S. ; Hauglustaine, D. ; Holloway, T. ; Isaksen, I. S. A. ; Jacob, D. J. ; Jonson, J. E. ; Kaminski, J. W. ; Keating, T. J. ; Lupu, A. ; Marmer, E. ; Montanaro, V. ; Park, R. J. ; Pitari, G. ; Pringle, K. J. ; Pyle, J. A. ; Schroeder, S. ; Vivanco, M. G. ; Wind, P. ; Wojcik, G. ; Wu, S. ; Zuber, A.: Multimodel estimates of intercontinental source-receptor relationships for ozone pollution. In: *Journal of Geophysical Research* 114 (2009), Nr. D04301. – URL http://www.agu.org/journals/ABS/2009/2008JD010816.shtml. – doi:10.1029/2008JD010816

[GEO 2009] GEO: *The Global Earth Observation System of Systems.* April

2009. – URL http://www.earthobservations.org/documents/200904_geo_info_sheets.pdf

[Lawrence u. a. 2006] Lawrence, B.N. ; Drach, R. ; Eaton, B.E. ; Gregory, J. M. ; Hankin, S. C. ; Lowry, R.K. ; Rew, R.K. ; Taylor, K. E.: Maintaining and Advancing the CF Standard for Earth System Science Community Data. (2006), August. – URL http://cf-pcmdi.llnl.gov/documents/white-papers/cf2_whitepaper_final.pdf

[Reichler und Kim 2008] Reichler, T. ; Kim, J.: How Well Do Coupled Models Simulate Today's Climate? In: *Bulletin of the American Meteorological Society* 89 (2008), 03/2008, Nr. 3, S. 303–311. – URL http://ams.allenpress.com/archive/1520-0477/89/3/pdf/i1520-0477-89-3-303.pdf. – doi:10.1175/BAMS-89-3-303. ISBN 00030007

[Schulz u. a. 2006] Schulz, M. ; Textor, C. ; Kinne, S. ; Balkanski, Y. ; Bauer, S. ; Berntsen, T. ; Berglen, T. ; Boucher, O. ; Dentener, F. ; Guibert, S. ; Isaksen, I. S. A. ; Iversen, T. ; Koch, D. ; Kirkevåg, A. ; Liu, X. ; Montanaro, V. ; Myhre, G. ; Penner, J. E. ; Pitari, G. ; Reddy, S. ; Seland, Ø. ; Stier, P. ; Takemura, T.: Radiative forcing by aerosols as derived from the AeroCom present-day and pre-industrial simulations. In: *Atmospheric Chemistry and Physics* 6 (2006), Nr. 12, S. 5225–5246. – URL http://www.atmos-chem-phys.net/6/5225/2006/. – doi:10.5194/acp-6-5225-2006

[Stepaniak 2004] Stepaniak, D.: *ERA-40 Horizontal Coordinate Conventions and Numerical Attributes.* July 2004. – URL http://dss.ucar.edu/datasets/common/ecmwf/ERA40/docs/horizontal-coordinate/index.html

[Stevenson u. a. 2006] Stevenson, D.S. ; Dentener, F.J. ; Schultz, M.G. ; Ellingsen, K. ; Noije, T.P.C. van ; Wild, O. ; Zeng, G. ; Amann, M. ; Atherton, C.S. ; Bell, N. ; Bergmann, D.J. ; Bey, I. ; Butler, T. ; Cofala, J. ; Collins, W.J. ; Derwent, R.G. ; Doherty, R.M. ; J.Drevet ; Eskes, H.J. ; Fiore, A.M. ; Gauss, M. ; Hauglustaine, D.A. ; Horowitz, L.W. ; Isaksen, I.S.A. ; Krol, M.C. ; Lamarque, J.-F. ; Lawrence, M.G. ; Montanaro, V. ; Müller, J.-F. ; Pitari, G. ; Prather, M.J. ; Pyle, J.A. ; Rast, S. ; Rodriguez, J.M. ; Sanderson, M.G. ; Savage, N.H. ; Shindell, D.T. ; Strahan, S.E. ; Sudo, K. ; Szopa, S.: Multimodel ensemble simulations of present–day and near–future tropospheric ozone. In: *Journal of Geophysical Research* 111 (2006), Nr. D08301.

– URL http://www.agu.org/journals/ABS/2006/2005JD006338.shtml.
– doi:10.1029/2005JD006338

[Unidata Program Center 2010] Unidata Program Center (Veranst.): *The NetCDF Users Guide - Data Model, Programming Interfaces, and Format for Self-Describing, Portable Data*. March 2010. – URL http://www.unidata.ucar.edu/software/netcdf/docs/netcdf.pdf

Teil II

Entwicklung eines Webtools zum Zugriff und zur Visualisierung georeferenzierter Atmosphärendaten

Sebastian Lührs

Entstanden im Rahmen einer Bachelorarbeit
an der Fachhochschule Aachen, Campus Jülich

Jülich, Juli 2011

Kapitel 7

Übersicht

7.1 MACC

MACC[1] (*Monitoring of Atmospheric Composition and Climate*) ist ein Forschungsprojekt im 7. Rahmenprogramm der EU, das zum Ziel hat, ein operationelles System zur Überwachung und Vorhersage von Spurengasen und Aerosolen in der Atmosphäre zu etablieren. Das Projekt ist dabei Teil der von ESA und EU gemeinsam getragenen GMES-Initiative (*Global Monitoring for Environment and Security*), welche eine verbesserte Nutzung von Umweltinformationen erreichen soll.

Die Grundlage von MACC sind Satellitendaten und Beobachtungen an Messstationen überall auf der Welt sowie Messungen an Bord von Forschungs- und Passagierflugzeugen. Diese Daten werden in numerische Modelle integriert (Datenassimilation), um so global konsistente Analysen und Vorhersagen der chemischen Zusammensetzung der Atmosphäre zu gewinnen. MACC liefert Produkte zu den Themen Klimawandel, Luftqualität, stratosphärisches Ozon, ultraviolette Strahlung und Solarenergie.[MACC]

Diese Produkte können Ergebnisse einer Auswertung sein oder auch neue Schnittstellen für andere Dienste liefern.

[1] `http://www.gmes-atmosphere.eu`

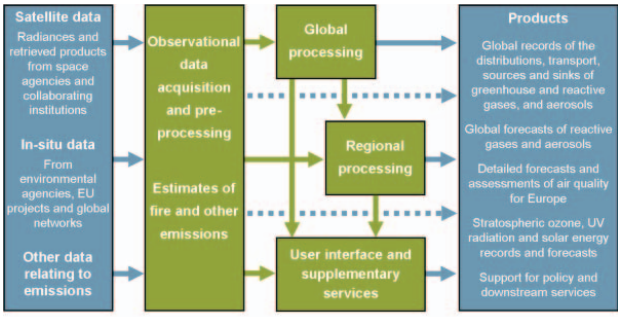

Abbildung 7.1: Übersicht über den Aufbau von MACC. Grün sind die unterschiedlichen Abarbeitungsschritte eines MACC-Produktes. Blau steht für die Eingabedaten (links) und die gelieferten Produkte (rechts).[MACC, architecture_for_web_s.png]

Zur Produkterstellung wird eine Vielzahl globaler und regionaler Umweltdaten verarbeitet, analysiert und aus den Modellrechnungen erzeugt. Die Verknüpfung dieser Daten zum Zwecke der Auswertung und Interpretation, sowie der Evaluierung der Modellsimulationen, stellt aufgrund der unterschiedlichen Datenstrukturen, der großen Zahl an Datenquellen und der schieren Menge an Daten eine besondere Herausforderung dar. Hinzu kommt der operationelle Charakter des Systems, der eine aktuelle Datenvorhaltung und Visualisierung erfordert.

Das in dieser Arbeit vorgestellte webbasierte Softwarewerkzeug stellt zusammen mit dem darunterliegenden Web Coverage Service (WCS, siehe Abschnitt 7.4) einen wesentlichen Schritt zum Aufbau eines weltweit „interoperablen“ Datennetzwerkes dar.

7.2 Besondere Anforderungen aufgrund des Quasi-Echtzeitcharakters des Systems

Im Gegensatz zu den meisten Anwendungen im Bereich der Klima- und Atmosphärenforschung muss der Datenaustausch im Rahmen des MACC-Projektes ohne allzu große Zeitverzögerung vonstatten gehen (NRT[2]-Daten), damit die Produkte in anderen Diensten weiter verwendet oder Fehler in Modellsystemen

[2] *Near Real Time* sind Daten, wenn zwischen Eintreffen der Eingabedaten (beispielsweise von Satelliten) und deren Ansicht oder Weiterverarbeitung nur wenige Stunden zum Prozessieren der Daten vergehen.

rechtzeitig erkannt werden können. Zusätzlich ist eine hohe Verfügbarkeit der Datendienste eine weitere Anforderung, die sich aus diesen Rahmenbedingungen ergibt.

Anderseits sollen die MACC-Produkte auch über längere Zeit zur Verfügung stehen (Langzeitreihen), um Jahresgänge zu analysieren oder Modelle nachträglich mit Daten, die nicht in Echtzeit zur Verfügung stehen, zu evaluieren. Hieraus ergeben sich hohe Anforderungen an die Speichersysteme.

7.3 Georeferenzierte Daten

Unter georeferenzierten Daten versteht man jede Form von Informationen, denen spezifische Positionsangaben auf der Erde zugeordnet werden können. Dies können Fotos sein, welche den Standort des Aufnahmegerätes enthalten, oder aber wie in der vorliegenden Arbeit Rasterdaten. Die Schnittpunkte des Rasters definieren stationäre Punkte auf der Erdkugel, welchen die Datenwerte zugeordnet sind. Wichtig ist, dass die Georeferenzierung zusammen mit den Daten gespeichert werden muss.

Georeferenzierte Daten erlauben den Vergleich untereinander, selbst wenn die Daten aus unterschiedlichen Datenquellen stammen[3]. Durch die Referenzierung können beispielsweise Satellitendaten mit unterschiedlichen Betrachtungswinkeln aufeinander abgebildet werden.

7.4 Web Coverage Service

WCS (*Web Coverage Service*) ist ein Standard, welcher vom OGC[4] (*Open Geospatial Consortium*) verwaltet wird. Er definiert webbasierte Schnittstellen zum Zugriff auf georeferenzierte Daten, die sogenannten *Coverages*.[WCS]

Der Fokus liegt momentan auf der Auslieferung von Rasterdaten. Der Standard erlaubt es neben der Auslieferung von kompletten *Coverages*, auch zeitliche und räumliche (in horizontaler Richtung) Ausschnitte abzufragen, sowie nur bestimmte Untermengen an Variablen (*Fields*) einzuschließen.

Die Daten können in unterschiedlichen Ausgabeformaten angefordert werden, sowohl binäre Formate wie netCDF aber auch Bildformate wie PNG sind möglich.

Informationen über die Datensätze werden dem Benutzer in Form von

[3] sofern die Datentypen hinreichend kompatibel sind

[4] Das OGC (`http://www.opengeospatial.org`) entwickelt offene Standards zum Zugriff auf Geodaten.[OGC]

XML-Dokumenten mitgeteilt. Der Zugriff erfolgt als HTTP-GET[5] Kommando, wobei in Version 1.1.2 folgende Anfragen an den WCS-Server vorgesehen sind:

- GetCapabilities (`http://<ServerURL>?service=WCS &acceptversions=1.1.2&Request=GetCapabilities`) liefert eine Übersicht über alle *Coverages* die vom gewählten Server bereitgestellt werden, sowie zusätzlich Metadaten über den Server.

- DescribeCoverage (`http://<ServerURL>?service=WCS&version=1.1.2 &Request=DescribeCoverage&identifiers=<Identifier>`) liefert eine genaue Beschreibung zu einer oder mehreren *Coverages*. Hierbei sind neben den enthaltenen Variablen auch Informationen über die räumlichen und zeitlichen Ausmaße der jeweiligen *Coverage* enthalten.

- GetCoverage (`http://<ServerURL>?service=WCS&version=1.1.2 &Request=GetCoverage&identifier=<Identifer>&BoundingBox=<...> &TimeSequence=<...>&RangeSubset=<...>&format=<Ausgabeformat>`) dient zum abschließenden Download des gewünschten Datensatzes und erlaubt es Ausschnitte aus einer gesamten *Coverage* abzufragen.

Basierend auf diesem Protokoll entstand am Forschungszentrum Jülich ein Server zur Auslieferung von MACC-Datensätzen.[Decker, 2011]

7.5 Vernetzung weltweit verteilter Datenserver

Neben dem MACC-Datenserver am Forschungszentrum Jülich werden weltweit auch weitere Langzeitreihen als *Web Coverage Service* angeboten, so zum Beispiel die Ergebnisse eines internationalen Modellvergleichs im Rahmen der *Task Force on Hemispheric Transport of Air Pollution* (TF HTAP). Deren Ziel war es, den Ferntransport von Luftschadstoffen besser zu verstehen (die Daten befinden sich hierbei ebenfalls am Forschungszentrum Jülich). Außerdem werden Satellitenbeobachtungen der NASA oder auch eine Reihe amerikanischer Bodenstationen[6] über WCS-Server zur Verfügung gestellt.

Durch das WCS-Protokoll ist es damit möglich, mit einem einheitlichen Interface auf verschiedene Server zuzugreifen und die dort gespeicherten Daten zu vergleichen.

[5] Bei HTTP- (*Hypertext Transfer Protocol*) GET werden alle Informationen, welche einem Server übergeben werden sollen, als Parameter an die Serveradresse in der Form `<Adresse>?<Parameter1>=<Wert1>&<Parameter2>=<Wert2>...` angehängt.

[6] Eine Übersicht über viele WCS-Server liefern die Kataloge des datafed-Verzeichnisses. `http://datafedwiki.wustl.edu/index.php/Compact_Catalog_-_Alphabetical`

7.6 Ziele dieser Arbeit

Ziel dieser Arbeit ist die Entwicklung eines Webinterfaces, welches den Zugang zu den gespeicherten MACC-Daten (siehe Abschnitt 7.1) über das WCS-Protokoll vereinfacht. Außerdem sollen auch andere Server abrufbar sein, welche dieses Protokoll implementiert haben[7]. Anfragen an den WCS-Server sollen hierzu nicht mehr händisch durch den Nutzer des Systems eingegeben, sondern durch eine grafische Benutzeroberfläche (GUI) zusammengestellt werden.

Zusätzlich soll dem Benutzer die Möglichkeit gegeben werden, Datensätze direkt im Webinterface grafisch zu vergleichen und grundlegende Informationen über die Datensätze abzufragen.

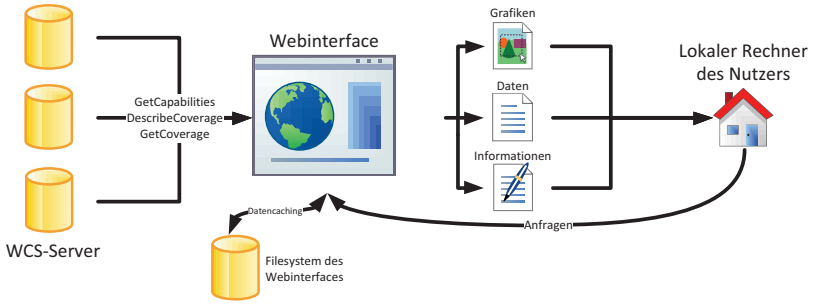

Abbildung 7.2: Struktur und grundlegende Aufgaben des Webinterfaces

Aus dieser Aufgabenstellung ergeben sich die folgenden Teilaufgaben:

Zugriff über das WCS-Protokoll

Die geforderten Datensätze müssen vom gewählten WCS-Server an den Server des Webinterfaces übertragen werden. Die standardisierten WCS-Befehle müssen hierzu, basierend auf den Benutzereingaben, zusammengebaut werden.

Es soll eine möglichst nutzerfreundliche und flexible Handhabung des Interfaces gewährleistet sein, so dass zum Beispiel auch mehrere Dateien gleichzeitig von unterschiedlichen WCS-Servern zum Download ausgewählt werden können. Auch die Auswahl von Datensätzen soll möglichst komfortabel gestaltet sein.

[7]In meiner Implementierung habe ich mich auf die WCS-Version 1.1.2 konzentriert, da diese Version zum Zeitpunkt der Erstellung die aktuellste Version des Protokolls darstellte. Zudem unterstützt der Jülicher WCS-Server vornehmlich diese Version.

Verwaltung der Dateien im Webinterface

Der Benutzer soll eine Möglichkeit haben, Datensätze aus der Dateiverwaltung des Webinterfaces auf seinen lokalen Computer herunterzuladen und eine Übersicht über alle gespeicherten Dateien einsehen können. Grundlegende Dateiinformationen sollen hierbei direkt dem Benutzer angezeigt werden.

Visualisierung

Innerhalb des Webinterfaces sollen georeferenzierte Darstellungen zu den geladenen Datensätzen erstellt werden können. Der Fokus liegt dabei auf der Visualisierung von Modell- und Emissionsdaten anhand von Karten und Konturdarstellungen und dem Vergleich von simulierten Zeitreihen mit Messdaten von Bodenstationen.

Hierbei müssen Grafikparameter (Kartenausschnitt, Farben, etc.) einstellbar sein und auch eine Möglichkeit geschaffen werden, mehrere Datensätze übereinandergelegt als Bild darzustellen.

Die erstellten Grafiken sollen ebenfalls auf den lokalen Computer des Benutzers heruntergeladen werden können.

Kapitel 8

Verwendete Software und Frameworks

8.1 Python

Python ist eine immer beliebter werdende Programmiersprache, die am Anfang der 90er Jahre entstand. Die Sprache zählt zu den interpretierten Programmiersprachen und vereint objektorientierte und funktionale Programmierung. Der Python-Interpreter ist für alle gängigen Desktop-Betriebssysteme verfügbar, sodass Python, ähnlich wie Java, zu den plattformunabhängigen Sprachen gezählt wird.

Neben einer umfangreichen Standardbibliothek kann Python beliebig erweitert werden, wobei maschinennahe oder zeitkritische Erweiterungen auch in C geschrieben und anschließend integriert werden können.[Ernesti und Kaiser, 2009, Seite 23ff]

Neben dieser Erweiterbarkeit zählt zu den Stärken von Python der Einsatz in unterschiedlichsten Gebieten. Aus diesem Grund habe ich mich dazu entschieden die Serverstrukturen des Webinterfaces in Python zu implementieren. Vor allem die Unterstützung zur Bearbeitung großer Datenmengen und die einfache Erzeugung grafischer Datendarstellungen spricht für Python gegenüber anderen Servertechnologien wie PHP oder Perl. Zudem besitzt die Sprache eine sehr übersichtliche und einfache Struktur, weswegen ich auch Java als Servertechnologie ausgeschlossen habe.

8.1.1 numPy

Die Python-Erweiterung numPy[1] erlaubt numerische Operationen und array-basierte Datenprozessierung in Python. Mathematische Operationen lassen sich auf numPy-Arrays beispielsweise als Vektoroperationen ausführen.

Der numPy-Kern basiert dabei auf einer Implementierung in C und Fortran, wodurch die Operationen ähnlich schnell wie Umsetzungen in maschinennäheren Sprachen durchgeführt werden können.

8.1.2 web.py

web.py[2] ist ein frei verfügbares Server-Framework für Python.

Der Aufbau und die Verarbeitung von HTTP-Anfragen verläuft ähnlich wie bei Java-Servlets. Jeder Serveradresse wird eine Klasse zugeordnet. Diese Klasse kann daraufhin über die fest definierten Methoden `GET` und `POST` die Anfrage entgegen nehmen und über die Standardausgabe ein Ergebnis an die HTTP-Antwort anhängen.

Zudem besitzt web.py ein Template-Konzept und kann, ähnlich wie PHP oder JSP, vollständige HTML-Seiten ausliefern, welche Python-Programmcode enthalten. Dieser integrierte Programmcode wird beim Lesen des Dokumentes auf dem Server ausgeführt. Um auch Bilder oder Dokumente ausliefern zu können existiert ein Verzeichnis `static`. Hier liegende Dokumente können direkt über eine entsprechende Pfadangabe in der URL[3] abgerufen werden.[WPY]

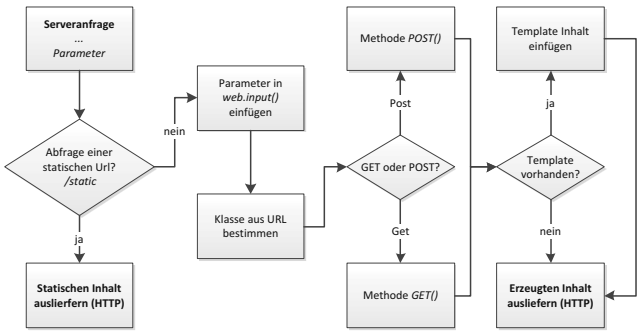

Abbildung 8.1: Schematische Übersicht zur Verarbeitung einer web.py HTTP-Anfrage

[1] `http://numpy.scipy.org/`

[2] `http://webpy.org`

[3] Der *Uniform Resource Locator* dient zur Adressierung von Informationen inklusive der Festlegung des Zugangsprotokolls.

web.py liefert für Entwicklungszwecke, neben der Abarbeitung der `POST` und `GET`-Anfragen, auch einen kompletten HTTP-Webserver. Für einen Produktivbetrieb sollte jedoch eine Servertechnologie wie zum Beispiel Apache verwendet werden. Diese besitzen ein besseres Loadbalancing und bieten mehr Einstellungsmöglichkeiten im Bereich Serversicherheit. web.py kann hierfür über unterschiedliche Apache-Module wie `mod_python` oder `mod_wsgi` eingebunden werden. Da in meiner Ausarbeitung vornehmlich die Entwicklung des Interfaces im Vordergrund steht, gehe ich im Folgenden von einer direkten Ausführung von web.py als Serversoftware aus.

8.1.3 netCDF

netCDF[4] ist ein binäres Dateiformat, welches ähnlich wie XML eine selbst beschreibende Struktur besitzt. Eine netCDF-Datei besteht normalerweise aus Dimensionen, Variablen und Attributen. Dimensionen definieren Form und Größe der Variablen, welche die eigentlichen Daten enthalten. Attribute liefern Meta-Informationen.

8.1.4 pyNio

pyNio[5] ist eine Python-Erweiterung, die das Lesen und Schreiben von netCDF-Dateien[6] ermöglicht.

pyNio kann das netCDF-Format lesen oder auch eine Datei in diesem Format erzeugen. Gelesene Datensätze werden in Python als mehrdimensionales numPy-Array zur Verfügung gestellt.

8.1.5 matplotlib

Die matplotlib ist eine mächtige 2D-Grafik-Bibliothek für Python. Dabei werden viele Konzepte aus MATLAB[7] übernommen und um die Stärken einer objektorientierten Sprache wie Python erweitert.[MPL]

Das folgende Beispiel[8] verdeutlicht die Einfachheit, grafische Plots mit Hilfe von matplotlib zu erstellen:

```
import numpy as np
import matplotlib.pyplot as plt
```

[4] `http://www.unidata.ucar.edu/software/netcdf/`
[5] `http://www.pyngl.ucar.edu/Nio.shtml`
[6] Auch andere Datenformate werden unterstützt.
[7] `http://www.mathworks.de/products/matlab`
[8] Beispiel von `http://matplotlib.sourceforge.net/users/pyplot_tutorial.html`

```
# Auf t ein Array mit Elementen zwischen 0 und 5 im Abstand
    von 0.2 erzeugen
t = np.arange(0., 5., 0.2)

# Die Funktionen f(t)=t; f(t)=t^2 und f(t)=t^3 plotten
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```

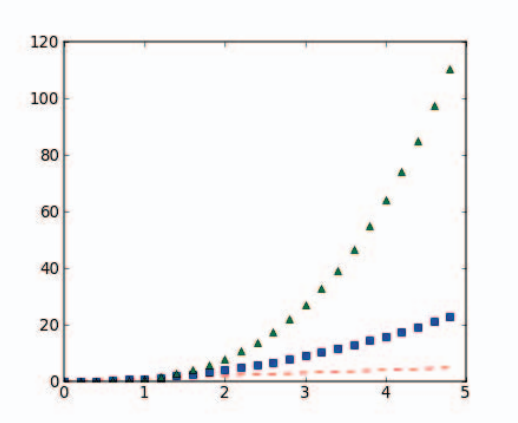

Abbildung 8.2: Ergebnis des obigen Skriptings

Die matplotlib besitzt zwei Implementierungen, die erste dient dazu, lokal am Rechner Grafiken zu entwickeln. Hierbei werden vornehmlich globale Variablen zum Festlegen aller Grafikparameter benutzt. Auf einem Server kommt stattdessen eine rein objektorientierte Variante zum Einsatz, um keine Fehler durch den gleichzeitigen Zugriff mehrerer Nutzer auf die globalen Variablen zu erhalten (Thread-Sicherheit).[Tosi, 2009, Seite 215ff]

In der objektorientierten Variante dient die Bildfläche (`Figure`) als Referenz zur jeweiligen Bildinstanz. Auf einer Bildfläche können anschließend mehrere Grafikbereiche (`Axes`) definiert sein, die mit unterschiedlichen Plottypen[9] gefüllt werden.

Als Ausgabeformat stehen der matplotlib sowohl Raster- als auch vektorbasierte Grafiktypen zur Verfügung. Als Standardausgabeformat für die Kartendarstellungen habe ich mich dabei für PNG entschieden, da hierbei viele kleingliedrigen Strukturen dargestellt werden müssen. Einfache Achsen-Plots

[9] `http://matplotlib.sourceforge.net/gallery.html` gibt eine Übersicht über die unterschiedlichen Grafiktypen, die verwendet werden können.

können auch als Vektorformat wie EPS oder SVG dargestellt werden.[Tosi, 2009, Seite 11]

Für die abschließende Erstellung der Grafiken kann matplotlib auf unterschiedliche *Backends* zurückgreifen. Standardmäßig ist dies QtAgg (Qt-Framework mit *Anti-Grain Geometry* als `renderer`). Das verwendete *Backend* ist dabei vornehmlich für eine Fenstererstellung zuständig, wenn matplotlib lokal auf einem Rechner ausgeführt wird. In meiner Arbeit nutze ich GTKAgg (basierend auf der GIMP ToolKit GUI library), wobei ich die Fenstererstellung nicht benötige, sondern lediglich auf den `renderer` zur Erstellung des fertigen Bildes zurückgreife.[Tosi, 2009, Seite 12f]

basemap

Als Erweiterung der matplotlib habe ich das basemap-Toolkit[10] verwendet. Es übernimmt eine Vielzahl an Grafikroutinen aus der matplotlib und erweitert sie um eine Koordinatenprojektion. basemap unterstützt dabei 23 verschiedene Kartenprojektionen und übernimmt die Umrechnung von geographischen auf Pixel-Koordinaten.

Zudem liefert das Toolkit Zugriff auf die GEOS-Bibliothek[11] und ermöglicht damit Ausgaben von Küstenverläufen und politischen Grenzen. Alle Kartenprojektionen und geographischen Angaben werden über dieses Toolkit verarbeitet und visualisiert.

Die verwendete basemap Version weist einen Fehler in der Berechnung von Maskierungs-Bereichen auf. Dieser Fehler führt in einigen Fällen zu falschen Darstellungen bei Konturplots. Mit Hilfe eines Patches habe ich diesen Fehler für die aktuelle basemap-Version behoben (siehe Abschnitt 11.1, Seite 147).

8.2 MySQL

Als Datenbank setze ich auf dem Server MySQL ein. Dabei erfolgt zum einen das Session-Management von web.py über eine eigene Tabelle der Datenbank, zum anderen wird aber auch die Dateiverwaltung der Nutzerdaten über eigene Tabellen gehandhabt.

Für Vergleichsplots zwischen Modell und Messdaten werden täglich Daten des deutschen Umweltbundesamtes in eine weitere Datenbank übertragen.

web.py bietet zu MySQL eine eigene Schnittstelle an und übernimmt die Konvertierung zwischen Datenbank- und Python-internen Datentypen.

[10] `http://matplotlib.sourceforge.net/basemap/doc/html/`
[11] `http://trac.osgeo.org/geos/`

Grafiken werden erst zur Laufzeit und auf Anfrage durch den Nutzer erzeugt, so dass alle Grafikparameter (Kartenausschnitt, verwendete Variablen einer Datei) in der MySQL-Datenbank gespeichert werden müssen.

8.3 JavaScript

JavaScript ist eine weit verbreitete Browser-Skriptsprache. Ich setze sie ein, um Benutzereingaben schon clientseitig zu überprüfen und schneller darauf zu reagieren.

JavaScript kann dabei jedoch nie eine endgültige Eingabekontrolle auf dem Server ersetzen, da ein Webseitenutzer den auszuführenden JavaScript-Code jederzeit beliebig beeinflussen kann. Daher kontrolliere ich alle erhaltenen Eingaben auf dem Server noch zusätzlich. [Wenz, 2009]

Durch JavaScript können clientseitig bereits viele Berechnungen vorweggenommen und so der Server entlastet werden.

8.3.1 AJAX

AJAX (*Asynchronous JavaScript and XML*) erweitert JavaScript um die Möglichkeit, auch nach dem vollständigen Laden einer Webseite nachträglich weitere Webseitenelemente oder Inhalte nachzuladen. Eine Webseite muss somit nicht mehr von vornherein alle möglichen Inhalte vorhalten, welche sich durch Nutzereingaben ergeben könnten, sondern kann die entsprechenden Inhalte nachladen.

Vor AJAX war es lediglich durch ein Neuladen der gesamten Seite möglich, Inhalte basierend auf einer Nutzereingabe in die Webseite einzubauen. Dieses Neuladen führte jedoch dazu, dass alle Benutzereingaben gesichert werden mussten, um auf der neuen Seite wieder vorhanden zu sein, und dies den Webseitenablauf störend unterbrach.

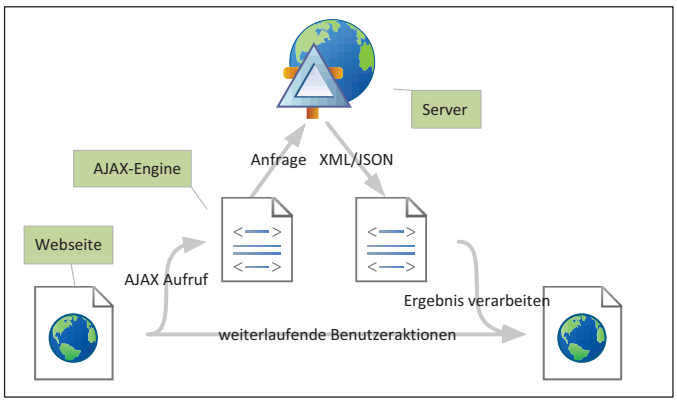

Abbildung 8.3: AJAX-Aufruf parallel zu weiteren Benutzeraktionen

Ich setze AJAX an vielen Stellen ein, um nachträglich Inhalte vom Server abzurufen. Man muss zwei verschiedene Arten des nachträglichen Ladens unterscheiden:

Zum einen möchte ich hier das direkte Laden über AJAX aufführen. Hierbei liefert der Webserver auf eine HTTP-Anfrage eine Antwort im JSON-Format[12] an den Client. Diese Art des Ladens wird für alle Zeichen-basierten Inhalte verwendet.

Die zweite Form ist, bedingt durch die Visualisierungsmöglichkeiten, das nachträgliche Laden von Grafiken. Hierbei kommt AJAX nicht zum Einsatz, sondern stattdessen hilft die Browser-Engine. Dennoch handelt es sich um eine Art asynchrone Serverabfrage, weshalb ich diese Form an dieser Stelle aufführen möchte. Es reicht hierbei die URL eines neuen Bildes in ein DOM[13] `IMG`-Element als `src` Attribut einzubauen. Der Browser erkennt selbsttätig, wenn er nun einen noch nicht geladenen Inhalt vom Server abrufen muss.

[12] *JavaScript Object Notation* ist eine ASCII-basierte Beschreibung von JavaScript-Objekten. Ähnlich wie in XML wird dabei neben den Attribut-Inhalten eines Objektes auch dessen Struktur gespeichert. In JavaScript kann ein solches Objekt durch ein `eval` wiederhergestellt werden.

[13] DOM steht für *Document Object Model* und ist eine Zugriffsstrategie auf die Inhalte einer XML- oder HTML-Datei. Alle Tags einer solchen Datei werden dabei als Knoten eines Baumes angesehen. Das DOM erlaubt somit das einfache Verwalten und Prozessieren strukturierter Inhalte.

8.3.2 jQuery

jQuery[14] ist ein JavaScript-Framework, welches häufig genutzte Aspekte der Sprache (Manipulation des DOM, AJAX, Event-Handling) vereinfacht und den nötigen Schreibaufwand für solche Aktionen verringert.

Gleichzeitig sichert jQuery, dass der verwendete Code in möglichst allen gängigen Browsern gleich funktioniert. Als Nutzer dieses Frameworks muss ich mich daher weniger um Unterschiede zwischen den Browsern kümmern. Da das Webinterface möglichst in allen neueren, modernen Browsern funktionieren soll ist dieser Aspekt nicht zu unterschlagen. [Vollendorf und Bongers, 2010, Seite 15ff]

Man kann jQuery somit als Vermittlungsstelle zwischen Browser und JavaScript verstehen. jQuery ermöglicht dabei keine Funktionalitäten, die nicht auch direkt in JavaScript programmierbar wären, macht jedoch vieles einfacher und übersichtlicher.[Vollendorf und Bongers, 2010, Seite 20f]

8.3.3 jQuery UI

jQuery UI[15] ist als Erweiterung von jQuery entstanden und liefert eine große Anzahl an GUI-Elementen für eine interaktive Webseite. Zwar bietet auch das grundlegende HTML gängige Eingabeelemente wie Buttons oder Textfelder, jedoch kann das Aussehen von diesen Elementen von jedem Browser eigenständig festgelegt werden.

Neben einem einheitlichen Aussehen[16] für alle bestehenden Eingabeelemente liefert das Framework jedoch auch völlig neue Interaktionsmöglichkeiten mit einer Webseite. So lassen sich Bereiche auf einer Seite verschieben oder umsortieren.

jQuery UI Selectmenü

Das Selectmenü ist das einzige fehlende HTML-Element, welches nicht durch jQuery UI ersetzt wurde. Hierzu wurde jedoch 2009 ein Plugin von der *filament group*[17] vorgestellt. Es passt das Design des Selectmenüs an und erlaubt auch das Einfügen von Grafiken oder längeren, mehrzeiligen Texten in ein solches Menü.

[14] `http://jquery.com`

[15] `http://jqueryui.com`

[16] Das Aussehen der jQuery UI Elemente kann vom Ersteller der Webseite per CSS genau festgelegt werden.

[17] `http://www.filamentgroup.com/lab/jquery_ui_selectmenu_an_aria_accessible_plugin_for_styling_a_html_select`

8.3.4 OpenLayers

OpenLayers[18] erlaubt das dynamische Einfügen georeferenzierter Daten und Kartendarstellungen auf einer Webseite. Der bekannteste Einsatz dieses Frameworks ist wahrscheinlich der freie GoogleMaps-Ersatz OpenStreetMap[19].

In den meisten Fällen bedient sich OpenLayers an den Daten eines Kachel-Servers (*Tile Service*). Die Anfrage nach einem bestimmten Gebiet auf der Erde wird hierbei in mehrere, gleich große Kacheln aufgeteilt, welche einzeln vom Server abgerufen werden. OpenLayers setzt auf Clientseite diese Kacheln wieder zusammen. Der Vorteil besteht darin, dass beim dynamischen Bewegen (*Scrollen*) über die Karte nur die fehlenden Kacheln nachgeladen werden müssen. Alle anderen Kacheln werden lediglich im Bildausschnitt verschoben.

Ich habe eine Kachelgröße von 256×256 Pixel gewählt (siehe Abbildung 8.4), um nicht zu viele kleinschrittige Bildabfragen vom Server zu erhalten und dennoch einen Nutzen aus dem Konzept der Kachel-Auslieferung zu gewinnen. Häufig verwendete Kacheln (Hintergrundkarte, Ländergrenzen etc.) habe ich zudem auf dem Server gecached, sodass diese Bildausschnitte nur einmalig generiert werden müssen.

Abbildung 8.4: Aufteilung einer Karte in einzelne Kacheln

OpenLayers unterstützt zudem die Umrechnung verschiedener Projektionsarten und versteht mehrere serverseitige Protokolle zur Auslieferung der Kacheln.

Wie bereits am Namen zu erkennen ist, liegt das Hauptaugenmerk von OpenLayers auf übereinander liegenden Schichten (*Layer*). Hierbei werden auf eine Basiskarte weitere Kachelebenen aufgebracht. Diese Ebenen dürfen dabei nur feinere Strukturen (Flüsse, Straßen) beinhalten, oder aber müssen bei

[18] `http://openlayers.org`, API auf [OPL]
[19] `http://www.openstreetmap.org`

flächigen Konturen leicht durchsichtig sein, um die verschiedenen Schichten gleichzeitig darstellen zu können. Durch das Aus- und Einblenden von Schichten lassen sich Zusammenhänge zwischen Datensätzen sehr gut verdeutlichen.

Die Kachel-Darstellung führt aufgrund der *Layer* zu dem Problem, dass kein vollständiges, herunterladbares Bild generiert wird. Clientseitig ist somit eine komplette Bildschirmkopie nötig, um die Grafik lokal zu speichern. Daher habe ich mich entschlossen, zusätzlich zu OpenLayers, alternativ auch vollständige, statische Karten serverseitig zu generieren. Alle Grafiken (Kacheln oder vollständige Darstellungen) werden von der matplotlib erzeugt.

8.4 XHTML

XHTML (*Extensible HyperText Markup Language*) kann man als XML konformes HTML verstehen. Mit der Entstehung von XML fiel auf, dass viele Tags in HTML keiner klaren Strukturregelung folgten. So mussten nicht alle Tags immer beendet werden (`<br>`) und auch die Verschachtelung von Tags wurde nicht genauer kontrolliert. XHTML änderte dies und legte zudem viele Tags fest, die aufgrund der Trennung von Layout und Inhalt nicht verwendet werden sollten (z.B. `font` zur Festlegung von Schriftoptionen).[Wetsch, 2010, Seite 27]

8.5 CSS

Cascading Style Sheets ist eine Sprache zur Festlegung grafischer Stilvorlagen für HTML oder XML-Dokumente. CSS macht Angaben über das Aussehen und die Lage eines strukturellen Elementes. Ziel ist es, das Layout unabhängig von den Inhalten einer Webseite definieren und verändern zu können. CSS-Stile können von einem Element, basierend auf dem DOM, an ein „Kind“-Element vererbt werden.[Wetsch, 2010]

Kapitel 9

Aufbau und Struktur des Webtools

9.1 Übersicht

Das entwickelte Webtool lässt sich in fünf Bereiche unterteilen. Im Folgenden möchte ich den Zugriff auf die unterschiedlichen WCS-Server, das Management der heruntergeladenen Dateien, die Visualisierung der Daten und den grafischen Vergleich zwischen Modell und Messung genauer erläutern. Den Upload-Bereich klammere ich an dieser Stelle aus, da ich mich in dieser Arbeit auf den WCS-Server Zugriff beschränken möchte.

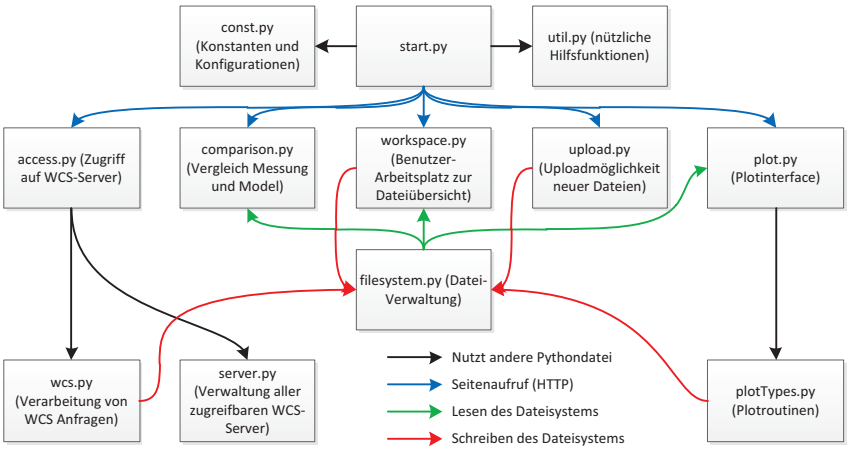

Abbildung 9.1: Übersicht über die Codestruktur des Servers und die Verbindungen der unterschiedlichen Serverbereiche. Eine komplette Dateiliste ist in Abschnitt 11.2 ab Seite 149 zu finden.

Jeder Bereich verfügt über

- eine HTML-Template-Datei, welche den Aufbau der Seite charakterisiert
- eine CSS-Datei, die das Layout der Seite anpasst
- Python-Dateien, die serverseitige Aufgaben erledigen (Antwort auf `GET`- und `POST`-Anfragen)
- JavaScript-Dateien, die clientseitig die Benutzeroberfläche bedienbar machen

9.2 Zugriff auf die WCS-Server

Ich habe den Zugriff auf die WCS-Server im Access-Bereich des Webinterfaces in vier Teilbereiche unterteilt, die im Folgenden beschrieben werden.

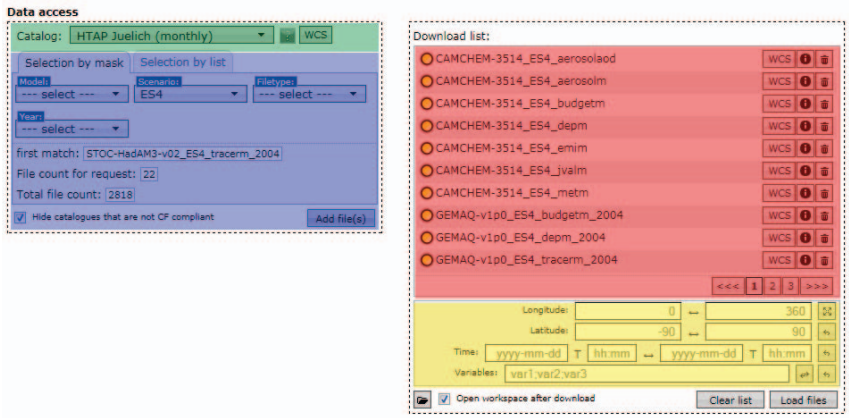

Abbildung 9.2: Der Access-Bereich des Webinterfaces: Grün ist die Katalogauswahl (Wahl des WCS-Servers). Blau ist die Auswahl der *Coverages*, welche heruntergeladen werden sollen. Rot ist eine Liste aller *Coverages*, die zum herunterladen vermerkt wurden und Gelb sind Einstellungsmöglichkeiten, welche den Teilausschnitt für die WCS-Abfrage festlegen.

9.2.1 Auswahl

Die Katalogliste enthält vordefinierte WCS-Server und wird bereits serverseitig in die Webseite integriert. Dabei liefert die Datenbanktabelle `server` die

benötigten Namen und Adressen. Nicht alle Server liefern vollständig CF[1]-conforme Daten. Per CSS-Klasse wird dies in der Liste vermerkt, um so dem Nutzer die Möglichkeit zu liefern, nicht konforme Server auszublenden.

Mit der Wahl des Servers findet eine erste Personalisierung des Interfaces durch den Nutzer statt. Die gewählte Server-URL wird in einem Cookie auf Clientseite abgelegt, um bei einem erneuten Aufruf des Interfaces eine standardmäßige Direktauswahl dieses Servers zu ermöglichen. Zudem werden weitere Serverdaten basierend auf der Datenbank per AJAX vom Server heruntergeladen.

Beispiel für die JSON-Antwort auf den AJAX-Request:

```
{"url": "http://macc.icg.kfa-juelich.de:580080/MACC_bnds",
 "seperator": "_",
 "valid": "valid",
 "name": "MACC_bnds",
 "filePart": ["Model", "Year", "Month", "Day"]}
```

Neben URL und Namen wird zusätzlich noch der Aufbau der *Coverage-Identifier* (diese sind vergleichbar mit einem Dateinamen) angegeben, hier also `Model_Year_Month_Day`. Diese Zerlegung liefert JavaScript die Möglichkeit, die *Identifier* in passende Blöcke zu zerlegen und somit dem Benutzer eine Auswahl über die Blöcke zu vereinfachen. Beispielsweise kann der Benutzer alle *Coverages* eines bestimmten Monates auswählen.

War die Abfrage der Serverparameter erfolgreich, wird umgehend eine weitere Anfrage gestartet, um eine Liste aller *Coverage-Identifier* zu erhalten, die der gewählte Server ausliefern kann. Das `wcs.py`-Skript auf dem Interface-Server stellt hierzu eine `GetCapabilities`-Anfrage an den WCS-Server. Die Serverantwort im XML-Format wird gelesen, die *Identifier* werden extrahiert und in das JSON-Format umgewandelt.

Clientseitig wird sowohl die komplette Liste aller *Identifier*, als auch je ein *Set*[2] für jeden `filePart` (siehe obiges Listing) erstellt. Ich habe hierbei ein Set gewählt, da ein Jahr oder ein Monat natürlich mehrfach in unterschiedlichen *Identifiern* auftreten können, aber nur einmalig in der Blockauswahl anwählbar sein sollen.

Der Benutzer wählt über `Selection by list` eine spezifische Auswahl aus der gesamten *Coverage*-Liste oder über `Selection by mask` eine Auswahl über Bestandteile der *Identifier*.

[1] *Climate and Forecast* Konvention zum Aufbau von netCDF-Dateien. Sie regelt die Metastruktur einer netCDF-Datei und ist vergleichbar mit dem Schema einer XML-Datei. `http://cf-pcmdi.llnl.giv`

[2] *Sets* (Mengen) sind nicht Teil der JavaScript Standard-API. Daher habe ich eine eigene Implementierung basierend auf einer sortierten Liste verwendet. Suchoperationen können binär durchgeführt werden.

Bei der zweiten Variante wird bereits beim Öffnen des Dropdown-Menüs (Selectmenü), zur Festlegung eines Bestandteiles, jede Kombination getestet, also ob es *Identifier* gibt, die dieser Auswahl entsprechen würden. Gibt es keine mögliche Kombination (zum Beispiel der 30. Februar) so wird der Eintrag rot hinterlegt.

Die Auswahl selbst verläuft über reguläre Ausdrücke. Wurde beispielsweise als Jahr `2010` gewählt und als Tag `01` (also jeden ersten Tag jedes Monats im Jahr 2010) so wird der Reguläre Ausdruck `[^_]*?_2010_[^_]*?_01(_,$)` auf alle *Identifier* angewandt, um passende Treffer zu finden. Die Treffer werden in eine separate Liste kopiert. Durch `Add file(s)` wird diese Liste abschließend an die Downloadliste angehangen.

9.2.2 Download zum Server des Webinterfaces

Der Download erfolgt anhand der zuvor erstellten Downloadliste. Nicht alle *Coverages* müssen vom gleichen WCS-Server stammen. Sowohl URL als auch *Identifier* liegen innerhalb der Downloadliste vor.

Der Downloadvorgang orientiert sich an dieser Liste, um eine Datei nach der anderen herunterzuladen. Wechselt der Benutzer also die Webseite (der Browser verwirft beim Verlassen einer Webseite alle JavaScript Variablen) oder löscht die Liste, so wird lediglich der gerade laufende Downloadvorgang noch beendet.

Über Textfelder und Auswahldialoge wird dem Benutzer die Möglichkeit gegeben, die unterschiedlichen WCS-Parameter genau festzulegen, die einen Ausschnitt der resultierenden Dateien definieren sollen.

Für jede Datei werden anschließend mehrere synchrone AJAX-Anfragen an den Webinterface-Server durchgeführt. Synchron bedeutet, dass AJAX den JavaScript-Ablauf bis zum Abschluss der Anfrage blockiert. Somit lässt sich garantieren, dass die Reihenfolge der Anfragen erhalten bleibt.

Zuerst werden per `GetDescription` Dateiinformationen gelesen und verarbeitet. Das Vorgehen ist ähnlich zur `GetCapabilities`-Anfrage. Der Webinterface-Server erhält die Anfrage und stellt seinerseits wieder eine Anfrage an den jeweiligen WCS-Server. Der nötige *Identifier* – lokal als Dateiname verwendet – für die Anfrage ist nun bekannt. Der WCS-Server liefert als Antwort eine XML-Datei, die im Webinterface-Server in JSON umgewandelt und an den Client gesendet wird. Diese Informationen werden clientseitig zwischengespeichert, sodass die Abfrage lediglich einmalig nötig ist[3].

[3] Auch vor dem Download-Vorgang kann sich der Benutzer die Informationen schon anzeigen lassen. Alternativ wird bei der Variablen-Auswahl eine Liste aller enthaltenen Variablen benötigt. Damit hierbei nicht immer wieder AJAX-Anfragen gesendet werden müssen, ist das Speichern sinnvoll.

Anschließend wird von der Dateiverwaltung erfragt, ob die gleiche Anfrage schon einmal gestellt wurde und eine entsprechende Datei schon existiert. Ist dies der Fall, so muss die Datei nicht erneut geladen werden.

Abschließend stößt eine `GetCoverage`-Anfrage den Download des Datensatzes an. Hierbei wird auf dem Interface-Server zuerst eine neue Datei in der MySQL-Datenbank der Dateiverwaltung vermerkt und ein Dateiname generiert. Anschließend wird eine Serveranfrage an den gewählten WCS-Server gesendet. Die Antwort ist eine HTTP `multipart`-Datei, also eine HTTP-Antwort, welche mehrere Abschnitte umfasst, ähnlich wie eine E-Mail mit Anhang. Der erste Teil ist wieder XML und gibt eine Übersicht über den gewählten Ausschnitt. Der zweite Teil ist die resultierende netCDF-Datei. Da diese Dateien sehr groß werden können und somit der Arbeitsspeicher schnell ausgelastet würde, habe ich mich für eine Streaming-Variante entschieden.

Beim Streaming wird immer nur ein Teilblock fester Größe vom WCS-Server heruntergeladen und anschließend in die bereitgestellte Datei der Dateiverwaltung auf dem Interface-Server geschrieben. Die Blöcke der HTTP `multipart`-Antwort sind durch einen eindeutigen `boundary-String` gekennzeichnet, der im HTTP-Header vereinbart wurde. Zusätzlich ist im Header eine `Content-Length` vereinbart, sodass ermittelt werden kann, wann der letzte Block heruntergeladen wurde.

Das Streaming habe ich über die `urllib2`-Bibliothek in Python realisiert. Eine `read`-Methode ermöglicht es, die HTTP-Antwort des WCS-Servers vollständig oder nur über eine festlegbare Länge einzulesen. Wichtig ist, dass die Blockgröße des letzten Blockes angepasst wird, da sonst auf weitere Daten über das Dateiende hinaus gewartet würde.

Während des Download-Vorganges wird zusätzlich ein Hash-Wert über die ermittelten Daten mitgeführt und in der Dateiverwaltung abgelegt. Der Nutzer kann hiermit die Vollständigkeit eines Downloads auf seinen Computer überprüfen.

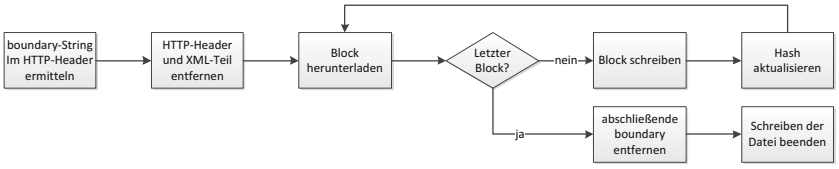

Abbildung 9.3: Verarbeitung einer gestreamten HTTP `multipart`-Datei

Der blockweise Download bietet zudem die Möglichkeit, dass dem Nutzer ein Fortschritt mitgeteilt werden kann. Der Anteil an geladenen Blöcken

wird hierzu in der MySQL-Datenbank abgelegt[4] und in wiederholten AJAX-Abfragen von JavaScript ausgelesen.

Mit Abschließen des Downloads wird anschließend die vollständige Datei mit Hilfe von pyNio geöffnet und Informationen zu den enthaltenen Variablen und Dimensionen in die Dateiverwaltung eingefügt. Gleichzeitig wird erneut geprüft, ob die Datei bereits in der Verwaltung vorhanden war. Diese zweite Kontrolle nach dem Download ist nötig, da zwei unterschiedliche WCS-Anfragen dennoch in der gleichen Datei resultieren können. Da die Daten gerastert sind, werden beispielsweise die gewählten Ausschnittsgrenzen auf das Raster gerundet oder aber eine Datei bildet in ihren Variablen lediglich eine echte Teilmenge einer anderen Datei. Enthält die neue Datei jedoch mindestens eine neue Variable gegenüber den bisher geladenen, so wird sie nicht verworfen. Auch überlappende Gebiete werden nicht berücksichtigt, da hierzu ein deutlich höherer Aufwand nötig wäre.

Jeder Benutzer besitzt eine eigene Dateiverwaltung basierend auf seiner `Session-Id`. Diese Verwaltung verarbeitet nur Dateien bis zu einer gewissen Größe. Vor Beginn des Download-Vorganges wird anhand der `Content-Length` überprüft, ob die zu ladende Datei in diesen Speicherbereich passt.

Der erfolgreiche Download wird dem Benutzer als AJAX-Response mitgeteilt. Hierbei nutze ich die standardmäßigen HTTP-Status-Codes, um den Benutzer über Fehlersituationen zu informieren. `200` bedeutet, dass alles in Ordnung war. Gab es einen Fehler (doppelte Dateien etc.) so teile ich dies dem Browser über den Status-Code `404` (*not found*) mit. jQuery bietet in seiner AJAX-Schnittstelle die Möglichkeit, auf fehlerhafte Anfragen – also Anfragen, welche einen Statuscode ungleich `2xx` zurückliefern – mit einer eigenen Funktion zu reagieren. So liefere ich dem Benutzer Informationen zum aufgetretenen Fehler.

9.3 Dateiverwaltung

Der Workspace dient zur Verwaltung und Übersicht aller Dateien, welche durch die Dateiverwaltung des Servers vorgehalten werden. Dateien können auf den lokalen Rechner des Nutzers heruntergeladen, grafisch dargestellt und gelöscht werden.

[4] Ich verwende den selben Speicherplatz für alle Fortschrittsanzeigen eines Nutzers, welche das Web-Interface betreffen. Startet man also zwei Downloads in zwei Browserfenstern gleichzeitig, so kann es passieren, dass die Anzeige nicht zum jeweiligen passt.

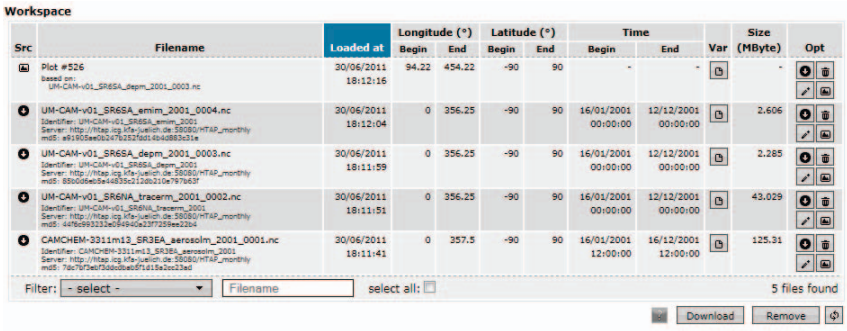

Abbildung 9.4: Der Workspace-Bereich

9.3.1 Aufbau der Dateiverwaltung

Die Dateiverwaltung enthält MySQL-basiert eine Liste von Dateien. Dabei habe ich sie objektorientiert in Python implementiert. Hierzu werden von der abstrakten Klasse `File` die konkreten Umsetzungen `WCSFile`, `UploadFile` und `IMGFile` abgeleitet. Die Dateiverwaltung muss damit lediglich den abstrakten Datentyp `File` verarbeiten können. Für jeden Datentyp ist eine eigene MySQL-Tabelle implementiert, da sich die Datentypen an Hand ihrer Attribute stark unterscheiden. Eine zentrale MySQL-Tabelle `files` übernimmt die Zuordnung und verwaltet allgemein gültige Attribute.

files	wcs_files	upload_files	map_plot
owner_id	owner_id	owner_id	owner_id
filename	filename	filename	img_id
type	server	lon_start	mtime
hash	identifier_req	lon_end	title
note	lon_start_req	lat_start	basemap_type
	lon_end_req	lat_end	river
	lat_start_req	time_start	countries
	lat_end_req	time_end	coastline
	timeRange_req	variable	grid
	variable_req		lonBeg
	lon_start		lonEnd
	lon_end		latBeg
	lat_start		latEnd
	lat_end		level
	time_start		time
	time_end		
	variable		

Tabelle 9.1: Übersicht über die gespeicherten Attribute in den Datenbank-Tabellen. Siehe Abschnitt 11.3 für eine vollständige Übersicht.

Die `_req`-Attribute der Tabelle `wcs_files` enthalten die Benutzereingaben,

welche diese Datei auf dem WCS-Server erzeugt haben. Die Angaben werden für eine Kontrolle doppelter Eingaben benötigt. Die Tabelle `map_plot` verwaltet lediglich allgemeine Informationen zu jeder Kartendarstellung. Eine weitere Tabelle `layer`, die hier nicht aufgeführt wurde, enthält noch weitere Optionen, welche jedoch nur für die Plotdarstellungen relevant sind.

Die Daten einer Datei werden über die Kombination aus `Session-Id` und Dateiname abgefragt. Somit kann ein Benutzer nur die Dateien einsehen, welcher unter seiner `Session-Id` gespeichert wurden.

Die Erstellung neuer Einträge in der Dateiverwaltung erfolgt mit Hilfe einer Fabrikmethode und auch das Laden der Datenbankinhalte in eine Python-Objektinstanz erfolgt über den gleichen Mechanismus. Zur Unterscheidung wird beim Laden einer Datei der Dateityp nicht angegeben, denn dieser befindet sich bereits in der MySQL-Tabelle. Erstellt man hingegen eine neue Datei, so muss der Typ angegeben werden, damit der neue Eintrag entsprechend zugeordnet werden kann.

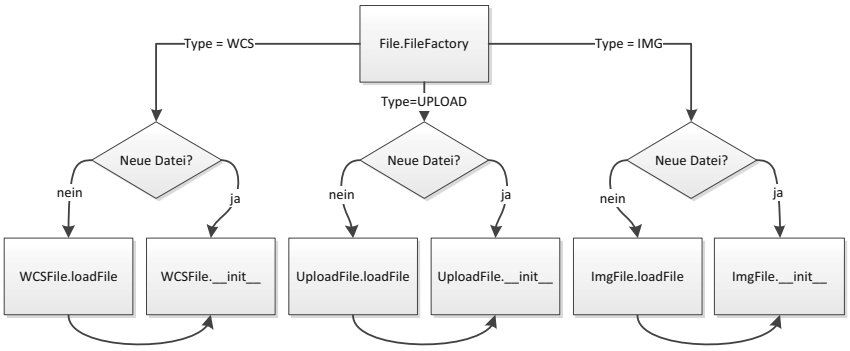

Abbildung 9.5: Übersicht über die File-Factory. Bei einer neuen Datei werden die Parameter direkt an den jeweiligen Konstruktor (die `__init__`-Methode) übergeben. Wird hingegen eine Datei geladen, so holt die Methode `loadFile` erst die benötigten Daten aus der Datenbank.

Die Dateiverwaltung muss in gewissen Zeitabständen aufgeräumt werden, damit unnötige Dateien gelöscht werden und ein konsistenter Zustand erhalten bleibt. Die `Session-Id` des Nutzers wird nach einem Tag vom Server entfernt bzw. beim Schließen des Client-Browsers. Hierzu lösche ich jeden Tag alle Ordner der Dateiverwaltung, die seit mindestens einem Tag nicht mehr verändert wurden. Das entsprechende Skript wird bei jedem Webseitenaufruf durch den Nutzer aufgerufen, aber nur einmal täglich auch durchgeführt.

9.3.2 Workspace-Darstellung

Der Inhalt der Dateiverwaltung wird mit Hilfe von AJAX vom Interface-Server abgefragt, nachdem die Workspace-Seite aufgerufen wurde. Zwar ließen sich diese Informationen schon beim Seitenaufruf mit übertragen, jedoch verwenden viele Browser ein Cache-Verfahren für einen schnelleren Seitenaufbau[5]. Es könnte daher passieren, dass veraltete Informationen geladen würden. In AJAX tritt bei `GET`-Anfragen zwar ebenfalls das gleiche Caching-Problem auf[6], jedoch sende ich hier bei jeder Anfrage eine Zufallszahl als `GET`-Parameter mit. Diese Zahl stellt sicher, dass sich der Aufruf immer von bisherigen unterscheidet und somit der Server befragt werden muss. Teilweise ist ein Caching natürlich auch erwünscht, zum Beispiel bei der Auslieferung von Bilddaten.

Aufbau einer AJAX-Anfrage mit Hilfe von jQuery:

```
$.ajax({
        type: 'POST', //HTTP-Typ
        url: "/filesystem?mix="+Math.random(), //Url mit
           Parameter, um das Caching zu umgehen
        success: readNewFilelist, //Funktion welche die JSON
           -Antwort des Servers verarbeiten soll
        error: function(data) {alert(data.responseText);},
           //Verhalten bei einer fehlerhaften Anfrage
        dataType: "json" //geforderter Datentyp des Servers
});
```

Die Tabellendarstellung des Workspaces erlaubt es nach dem Laden der Dateiinformationen, Operationen entweder auf einzelne Zeilen (über die Buttons am Ende der jeweiligen Zeile) oder aber auf mehrere Zeilen (durch das Markieren der entsprechenden Zeilen mit einem Mausklick und über die Buttons am unteren Rand) anzuwenden. Selektierte Zeilen ermittle ich über eine CSS-Klasse, welche bei einem Mausklick gesetzt bzw. wieder entfernt wird.

9.3.3 Download zum Nutzer

Wurden Dateien von den WCS-Servern heruntergeladen, so befinden sich diese vorerst nur im Dateisystem des Interface-Servers. Im Workspace wird dem Nutzer zusätzlich die Möglichkeit geschaffen, die Dateien anschließend auf seinen lokalen Computer herunterzuladen. Hierfür habe ich mich auch wieder für eine Streaming-Lösung entschieden, um den Arbeitsspeicher des Servers nicht

[5]Das Cache-Verhalten lässt sich per META-Tag zwar in einer Seite deaktivieren, aber nicht alle Browser übernehmen diese Einstellung.

[6]jQuery erlaubt das Deaktivieren des AJAX-Caches über `$.ajaxSetup({cache:false})`, allerdings ist auch hier das Browser-Verhalten nicht immer sichergestellt.

zu stark zu beanspruchen.

Wird eine Datei einzeln heruntergeladen, setze ich die HTTP-Header Attribute `Content-Disposition` auf `attachment`, `Content-type` auf den entsprechenden MIME-Typen der angefragten Datei[7] und `Content-Length` auf die Dateigröße in Byte. web.py ermöglicht es nun Inhalte als Generator-Ausdruck zu versenden. Dies bedeutet, dass ein Block der Datei (als Blockgröße verwende ich ein Megabyte) an den Browser des Nutzers gesendet und bei vollständiger Übertragung automatisch mit dem nächsten fortgefahren wird. Der Generator stoppt jeweils den weiteren Programmablauf, bis die Übertragung eines Blockes abgeschlossen ist.

Implementierung in Python für eine netCDF-Datei:

```
file = Filesystem().getFile(filename) #Dateiinformationen
    aus der Dateiverwaltung lesen

#HTTP-Header setzen
web.header('Content-Disposition','attachment; filename='+
    file.getFilename())
web.header('Content-type','application/x-netcdf')
web.header('Content-Length',file.getSize())

for i in readFile(file.getPath(),file.getSize()): #Datei
    blockweise lesen und Block auf i speichern
        yield i #Generator-Ausdruck: i versenden
            anschliessend auf Browser-Antwort warten
```

Sollen mehrere Dateien gleichzeitig heruntergeladen werden, so fasse ich diese als TAR-Datei zusammen. Diese Datei wird jedoch erst auf Clientseite vollständig erstellt, der Server kennt weiterhin nur die einzelnen Dateien. Dies ist möglich, da das TAR-Format die einzelnen Dateien lediglich durch kurze Headerinformationen unterteilt. Die Reihenfolge bleibt jedoch ansonsten unangetastet und es muss nicht schon zu Beginn ein Inhaltsverzeichnis übertragen werden. Somit eignet sich das TAR-Format sehr gut, um während des Streaming-Vorganges dynamisch erstellt zu werden.

Wichtig ist, dass in TAR-Dateien eine neue Datei immer nur an einer Blockgrenze von 512Byte beginnen darf. Ist eine Datei also kürzer oder benötigt keine volle Anzahl von Blöcken, so muss der letzte Block mit 0-Bytes aufgefüllt werden. Das Ende der TAR-Datei wird durch zwei 0-Byte-Blöcke gekennzeichnet. Da sich aus diesen Gründen die resultierende Dateigröße nicht trivial vor Sendebeginn (der HTTP-Header wird als erstes übertragen) bestimmen lässt (und nicht der Summe der einzelnen Dateigrößen entspricht) setze ich hierbei keine `Content-Length`, sondern stattdessen das HTTP-Attribut

[7] `application/x-netcdf` bei netCDF-Dateien

`Transfer-Encoding` auf `chunked`. Der Browser versucht nun keine Restzeit für den Download zu ermitteln und stellt den Restdownload nicht mehr als wachsende Ladeanzeige dar. Im Firefox wird beispielsweise stattdessen ein kontinuierlich rotierender Balken angezeigt.

9.4 Visualisierung

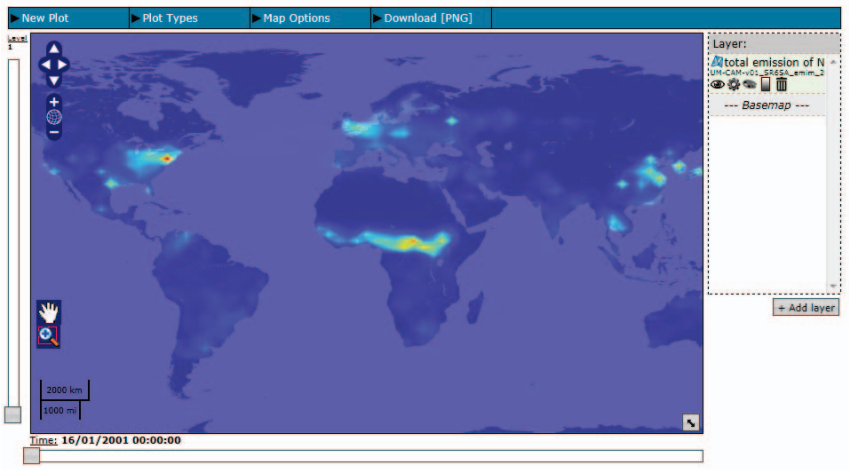

Abbildung 9.6: Der Plot Bereich

9.4.1 Visualisierungskonzept

Ziel der Visualisierung ist es, die gespeicherten georeferenzierten netCDF-Dateien grafisch auf einer Kartendarstellung anzuzeigen. Dabei soll der Nutzer möglichst viel Freiheit bei der Grafik-Erstellung besitzen und dennoch eine einfach zu bedienende Oberfläche zur Verfügung gestellt bekommen.

In dieser Arbeit habe ich zwei Konzepte realisiert. Erstens können die Daten als statischer Plot angezeigt werden. Dies bedeutet, dass eine Grafik erzeugt wird, welche abschließend auch auf dem lokalen Rechner des Nutzers gespeichert werden kann. Zum anderen habe ich eine Darstellung mit Hilfe von OpenLayers gewählt. Hier kann der Nutzer sehr übersichtlich die Kartendarstellung zoomen und verschieben, verliert jedoch aufgrund der Kachelung (siehe Abschnitt 8.3.4) die Möglichkeit, die Grafik als Ganzes zu speichern.

Zwischen beiden Varianten kann der Nutzer beliebig wechseln, wobei getroffene Einstellungen beibehalten werden.

Neben einer Grundkarte, die die geographischen Grenzen der Erde aufzeigt, kann je eine Variable aus einer netCDF-Datei als *Layer* über die bestehende Ansicht gelegt werden. Das Bild wird somit aus mehreren Schichten aufgebaut, von denen jede einzelne Schicht getrennt verwaltet werden kann (siehe Abbildung 9.7).

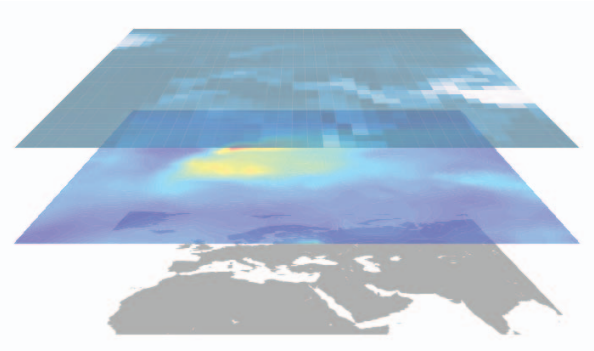

Abbildung 9.7: Unterschiedliche Schichten einer Kartendarstellung: Untere Schicht: Basisschicht, darüber: zwei visualisierte Datenvariablen

9.4.2 Kachel Generierung

Der Server unterscheidet zwischen zwei Kacheltypen, gecached und ungecached.

Gecached werden alle Hintergrundkarten, die vom basemap-Toolkit basierend auf der GEOS-Bibliothek erstellt wurden. Die erstmalige Erstellung dauert dabei (je nach Detailgrad) sehr lange. Da die Kartenausschnitte für alle Nutzer gleich sind, bietet sich hier ein Caching an.

Ungecached sind alle Kacheln, die eine bestimmte Datenvariable (also z.B. die Konzentration eines Spurenstoffes) veranschaulichen sollen. Diese Daten können von jedem Nutzer anders gewählt werden, sodass ein Caching hierbei nicht sinnvoll erscheint.

Python erlaubt Mehrfachvererbung, sodass sich der Aufbau der unterschiedlichen Plottypen wie folgt gliedert.

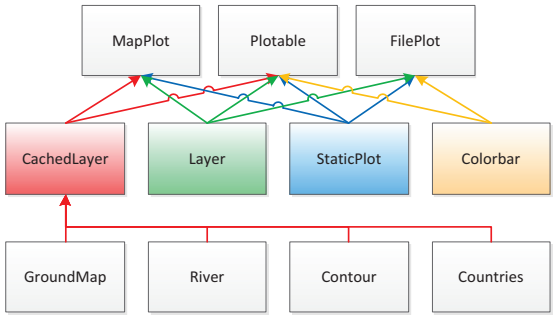

Abbildung 9.8: Übersicht über alle Plot-Typen

Die drei Klassen `Plotable`, `MapPlot` und `FilePlot` lesen Eingabeparameter des Clients und liefern Schnittstellen für die spezialisierten Routinen.

Zur Übergabe der Eingabeparameter an den Server habe ich mich am WMS[8]-Protokoll orientiert, da OpenLayers auf Clientseite dieses Protokoll beherrscht und hierüber Anfragen formulieren kann. Dabei unterstütze ich nur die benötigten Attribute dieses Services.

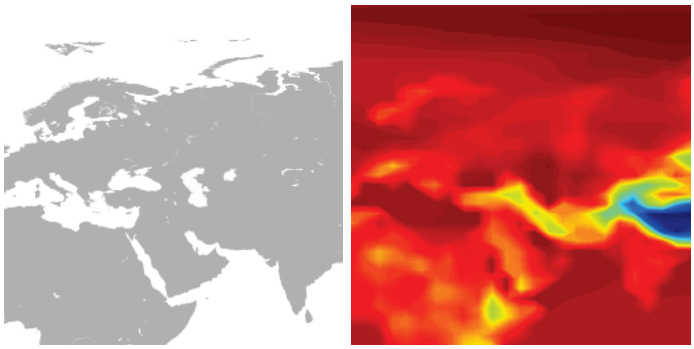

Abbildung 9.9: Beispiel für generierte Kacheln: Links eine Grundkarte, rechts der gleiche Ausschnitt für eine Variable (Luftdruck)

Plotable

Die Ausmaße einer Kachel werden auf 256×256 Pixel festgelegt (siehe Abschnitt 8.3.4). Zudem wird eine randlose Darstellung für alle Kacheln gefordert.

[8]Der *Web Map Service* ist, wie auch WCS, ein vom OGC verwaltetes Protokoll. `http://www.opengeospatial.org/standards/wms`

Treten Fehler in der Bearbeitung einer Kachel auf (wurde beispielsweise ein Ausschnitt angefragt, der keine Daten enthält), so wird eine transparente Kachel an den Browser gesendet.

Die Klasse `Plotable` verwaltet zudem das endgültige Erstellen der Grafik, also das Generieren eines PNG mit Hilfe des `renderers`. Die Datei wird dabei in einem *Stringbuffer* virtuell erzeugt und direkt an den Brower gesendet:

```
def GET(self,cache=None,header=True):
if header: #HTTP-Header setzen
        web.header('Content-type','image/png')
        web.header('Content-Disposition','attachment;␣
           filename=plot.png')
output = StringIO.StringIO() #virtuelle Datei erzeugen
FigureCanvas(self.fig).print_png(output) #Grafik in die
   Datei rendern
outputString = output.getvalue() #virtuelle Datei in String
   umwandeln
if cache is not None: #Daten cachen wenn dies gewuenscht ist
        cache.write(outputString)
self.fig.clear() #aufraeumen
output.close()
return outputString #Grafik an den Browser senden
```

MapPlot

Der `MapPlot` liefert eine Schnittstelle zur Kartenprojektion, also zum basemap-Toolkit. WMS besitzt hierfür den Parameter `BBOX`, um den gewünschten Kartenausschnitt anzugeben. Als Kartenprojektionsart habe ich mich für eine zylindrische Plattkarte[9] entschieden, da so keine aufwendigen Umrechnungen von Grad- in Pixel-Koordinaten vorgenommen werden müssen.

FilePlot

Der `FilePlot` stellt Daten und Informationen zu einer darzustellenden Datei der Dateiverwaltung zur Verfügung. Er ist nur für ungecachete Layer interessant, da hierbei auf die Verwaltung zugegriffen wird und netCDF-Dateien gelesen werden. Somit sind die Daten für jeden Nutzer unterschiedlich.

Colorbar

Die `Colorbar` dient als Legende zu jeweils einem *Layer*. Die Legende wird dabei getrennt von der Kachel gerendert und kann auch getrennt vom Browser

[9] geographische Länge und Breite werden direkt als kartesische Koordinaten verwendet

abgefragt werden. Die matplotlib erlaubt hierbei zwar auch eine gleichzeitige Erstellung zusammen mit den entsprechenden Kacheln, jedoch würde die `Colorbar` dabei in jede Kachel eingebaut werden, was den nahtlosen Übergang der Kacheln verhindern würde.

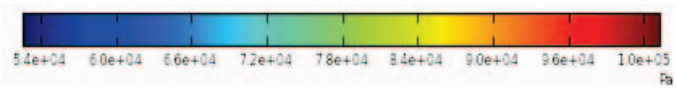

Abbildung 9.10: Die `Colorbar` eines Layers

Plot Typen

Ich habe vier Plot Typen zur *Layer*-Darstellung implementiert:

- `Scatter Plot`: Einfacher Punktplot, für den an jeder Stelle des Datenrasters ein farbiger Kreis zur Verdeutlichung der Datenwerte gesetzt wird.
- `Contour Plot`: Höhenliniendarstellung der Datenwerte
- `Filled Contour Plot`: Zusätzlich zu den Höhenlinien wird die Fläche zwischen den Linien farbig gefüllt. matplotlib interpoliert hierzu zwischen den vier benachbarten Datenpunkten eines Rasterpunktes. Für das Maskieren von Daten ist dabei wichtig, dass bei Fehlen eines Nachbarpunktes die gesamte Rasterzelle nicht gefüllt wird.
- `Mesh Plot`: Die Rasterzellen werden verdeutlicht, indem jede Zelle über ihre komplette Fläche mit dem Datenwert an der Mitte der Zelle belegt und farbig dargestellt wird. matplotlib geht eigentlich davon aus, dass die angegebenen Rasterpunkte nicht die Mitte, sondern die untere linke Ecke einer Zelle spezifizieren. Somit müssen für diese Darstellungsart alle Rasterangaben um eine halbe Zellenbreite verschoben werden.

9.4.3 Generierung statischer Grafiken

Statische Grafiken basieren auf den gleichen Schnittstellen wie die zugrunde liegenden Kacheln. Jeder Plottyp besitzt daher neben einer direkten Kachelabfrage über HTTP auch eine statische Methode, die vom statischen Plot als Schnittstelle verwendet werden kann, um eine solche Grafikart in das Gesamtbild einzubauen.

Bei statischen Plots wird für jeden *Layer*, der eine Dateivariable darstellt, eine `Colorbar` an die Grafik angefügt, da die Legende fest in die Grafik eingebettet sein muss.

Die Abfrage statischer Grafiken muss vom Brower über eine `GET`-Abfrage erfolgen können, da nur so die Grafik zum Beispiel in einem `IMG`-Tag angezeigt werden kann. Anders als bei den Kacheln müssen für einen statischen Plot alle eingestellten Parameter für jeden *Layer* in diesem `GET`-Aufruf dem Server mitgeteilt werden. Viele Browser erlauben jedoch maximal 1024-Zeichen innerhalb der URL. Ich verwende daher meine Dateiverwaltung und speichere alle *Layer*-Einstellungen, die ich von Clientseite über `POST`-Anfragen erhalte [10]. Als Schnittstelle zur Grafik dient schließlich lediglich eine kurze Identifizierungsnummer (ID), welche die passenden Parameter aus der Datenbank lädt.

Durch die Speicherung der Grafikdaten gewinnt man zudem den Vorteil, dass eine Grafik aus der Dateiverwaltung wiederhergestellt werden kann. Verlässt man also das Plot-Interface, so kann man über den Workspace-Bereich später jederzeit eine alte Grafik wieder öffnen und weiterbearbeiten.

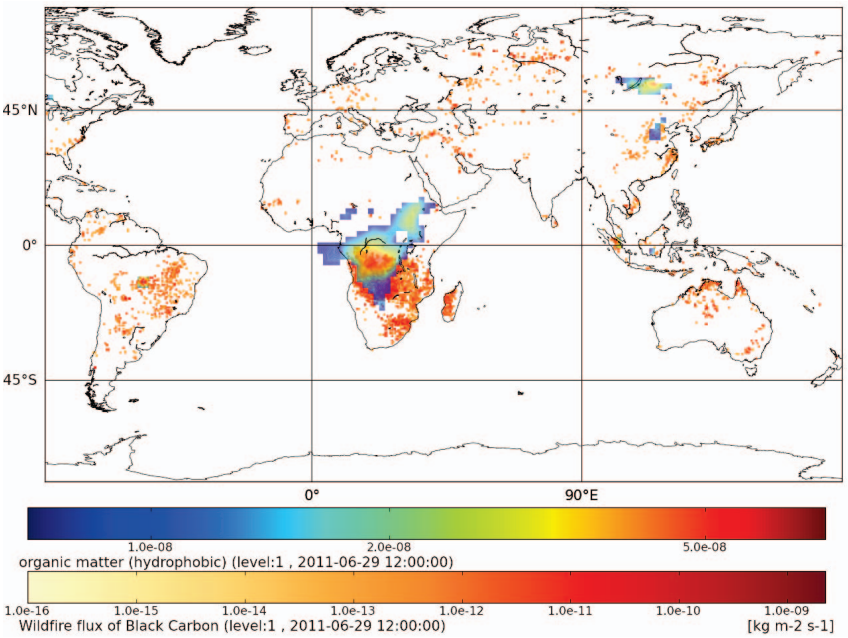

Abbildung 9.11: Beispiel eines statischen Plots zur Darstellung zweier verschiedener Datenvariablen.

[10] Bei `POST` werden die Parameter im Body-Teil der HTTP-Nachricht übertragen und sind somit nicht einer besonderen Begrenzung unterworfen.

9.4.4 Konfiguration von OpenLayers

Die Abfrage der generierten Kacheln erfolgt auf Clientseite mit Hilfe von OpenLayers. Alle gecachten Kartendarstellungen habe ich dabei fest in die Konfiguration eingetragen. Dies bedeutet, dass alle diese *Layer* bereits mit dem Laden der Seite zur Verfügung stehen und ausgewählt werden können. Die nötigen Grafiken müssen natürlich dennoch geladen werden, sobald der jeweilige *Layer* aktiviert wird.

Definiert der Nutzer weitere *Layer*, so werden diese in die Konfiguration nachträglich aufgenommen. Da die Reihenfolge der *Layer* entscheidend ist, habe ich das jQuery UI Toolkit `sortable` für die *Layer*-Liste verwendet. Die Listenelemente lassen sich dadurch mit Hilfe der Maus verschieben und umsortieren.

Das Aus- und Einblenden der *Layer* habe ich außerdem in eigenständige Menüs außerhalb des Plot-Bereiches ausgegliedert. OpenLayers besitzt hierfür zwar standardmäßig eine eigene Verwaltung, durch das mögliche Hinzufügen und Entfernen von *Layern* ist diese Anordnung jedoch sehr unübersichtlich.

9.4.5 Einstellungsmöglichkeiten

Alle Einstellungsmöglichkeiten werden lokal beim Nutzer in einer *Layer*-Liste gesichert. Zusätzlich werden jedoch mit jeder Änderung auch die Eintragungen in der Bilddatenbank auf dem Server aktualisiert. Ist zum Zeitpunkt der Änderung die statische Plot-Ansicht aktiviert, so wird zudem ein Neuladen dieser Grafik angestoßen. Die Layer werden immer neu geladen, wenn es eine sie betreffende Änderung gab. Die OpenLayers-Funktion `layer.mergeNewParams` fügt dabei neue Attribute zu einem *Layer* hinzu und führt gegebenenfalls das Neuladen durch, wenn sich Änderungen ergeben haben.

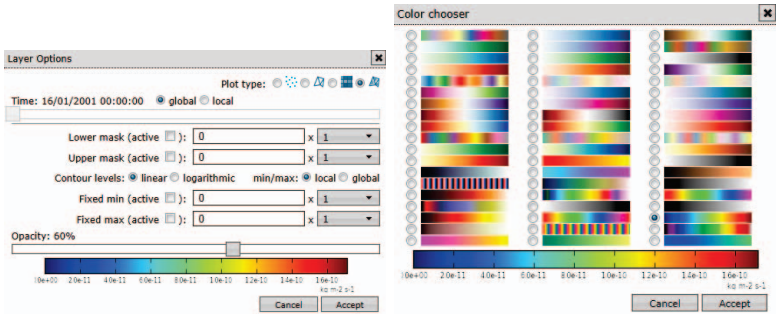

Abbildung 9.12: Einstellungsmöglichkeiten für einen Layer

Die folgenden Änderungen können durch den Nutzer zur Gestaltung der Plotansicht vorgenommen werden:

- Hinzufügen neuer Layer und Entfernen vorhandener
- Zoomen und Verschieben der Karte
- Feinere Ausschnittwahl bei statischen Plots
- Umsortieren der Layerreihenfolge
- Ausblenden einzelner Layer
- Wechsel zwischen den vier Layertypen
- Wechsel zwischen vertikalen Höhenschichten (sowohl für alle als auch für einzelne Layer möglich)
- Wechsel zwischen verschiedenen Zeitpunkten (sowohl für alle als auch für einzelne Layer möglich)
- Maskierung hoher und niedriger Datenwerte
- Wechsel zwischen linearer und logarithmischer Skaleneinteilung
- Festlegung fester Minima- und Maxima-Werte zur Farbdarstellung
- Wechsel der Farbpalette
- Festlegung der Transparenz eines Layers
- Vollbilddarstellung
- Festlegung einer Überschrift für statische Plots
- Änderung der Auflösung von statischen Plots

9.5 Der Comparison-Bereich

Der Comparison-Bereich erlaubt den Vergleich zwischen Modell und Messdaten. Als Modelldaten stehen dabei vornehmlich die Daten des MACC-Kataloges zur Verfügung. Die Daten müssen die Variablen `vmr_co`, `vmr_so2`, `vmr_o3`, `vmr_no2` oder `vmr_no` enthalten. Zudem steht momentan lediglich Deutschland als Messdatengebiet zur Verfügung und es wird nur der Zeitraum seit dem

24.7.2010 bereitgestellt. Als Messdatensatz dienen Luftmessdaten des deutschen Umweltbundesamtes, die täglich in eine MySQL-Datenbank des Webinterfaces kopiert werden. Die Modelldaten müssen vor der Benutzung über das Access-Menü von einem WCS-Server heruntergeladen werden.

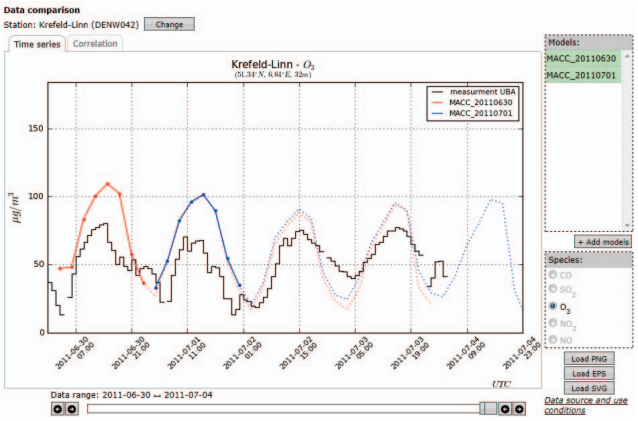

Abbildung 9.13: Der Comparison-Bereich

9.5.1 Auswahl der Messstation

Das Umweltbundesamt stellt eine Vielzahl an Messstationen innerhalb Deutschlands zur Verfügung. Für eine benutzerfreundliche Auswahlmöglichkeit habe ich mich neben einer Listen- für eine Landkartendarstellung entschieden, die alle Stationen als Punkte auf einer Deutschlandkarte enthält. Damit die Punkte auswählbar sind, habe ich diese nicht schon serverseitig in die Landkartengrafik integriert, sondern lege die Punkte nachträglich über die erstellte Karte. Als Kartenprojektion habe ich mich hierbei, anders als bei den anderen Visualisierungen, für eine Mercator-Projektion[11] entschieden, da durch die Winkel- und Formentreue der Projektion Deutschland gestreckt und somit übersichtlicher erscheint.

Der Browser erhält alle Stationsnamen zusammen mit ihren jeweiligen Positionsangaben definiert in Längen- und Breitengraden. Die Umrechnung in

[11]Die Mercator-Projektion ist eine zylindrische Projektion. Sie unterscheidet sich von der einfachen zylindrischen Plattkarte dadurch, dass sie Winkel und Formen erhält, jedoch nicht längentreu ist. Zudem ist die Mercator-Projektion durch die starke Verzerrung in Richtung der Pole auch nicht flächentreu.

Pixelkoordinaten erfolgt für die Breitengrade über

$$y = \frac{1}{2} \cdot ln\left(\frac{1 + sin(\phi)}{1 - sin(\phi)}\right)$$

(ϕ ist die jeweilige Gradangabe). In x-Richtung können die Längengrade direkt als kartesische Koordinaten übernommen werden. Mit Hilfe der CSS-Eigenschaft `position: absolute;` kann die Stationsmarkierung abschließend über die Karte an die Position der berechneten Koordinaten gelegt werden.

Die zuletzt gewählte Station wird außerdem in einem Cookie im Browser des Clients gespeichert, um standardmäßig bei einem erneuten Besuch ausgewählt werden zu können.

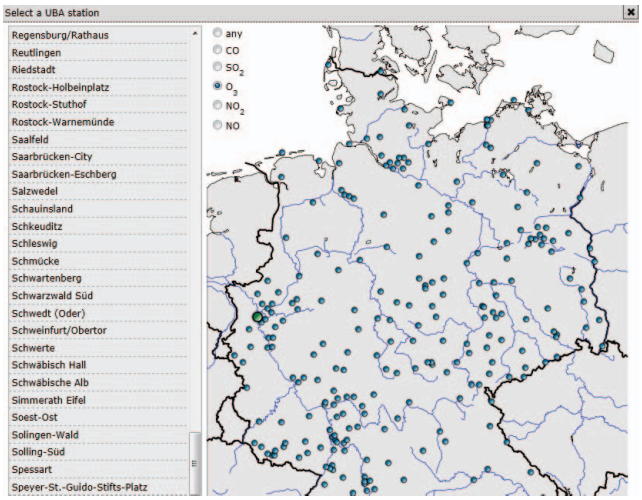

Abbildung 9.14: Ausschnitt der Stationsauswahl

9.5.2 Plotgenerierung

Die Comparison-Grafik wird, wie auch alle anderen Visualisierungen, mit Hilfe der matplotlib erzeugt. Ändert der Benutzer die Station, fügt ein neues Modell hinzu oder ändert die Zeitspanne, so wird hierzu die URL des dargestellten Bildes angepasst. Durch die neuen `GET`-Parameter stößt der Browser ein Neuladen der Grafik an.

Die Daten des Umweltbundesamtes werden für die geforderte Zeitspanne aus der Datenbank gelesen und in numpy-Arrays übertragen. Die Messdaten

liegen in der Einheit $\frac{\mu g}{m^3}$ und die Modelldaten als volumenbezogenes Mischungsverhältnis vor. Die Umrechnung auf ein massenbezogenes Mischungsverhältnis erfolgt über:

$$mmr = vmr \cdot \frac{M_s}{M_{air}}$$

Wobei hier mmr das massenbezogene, vmr das volumenbezogene Mischungsverhältnis, M_s die molare Masse des Stoffes und M_{air} die molare Masse der Luft ist.[Decker, 2011, Seite 36]

Grundsätzlich wird für den Vergleich die unterste Höhenschicht jeder Modelldatei verwendet. Bei den MACC-Daten folgen die Höhenschichten dem Profil des Erdbodens über eine *hybrid sigma*-Darstellung[12]. Zur Umrechnung der Drucklevel in eine metrische Höhendarstellung verwende ich als Nährung die vereinfachte barometrische Höhenformel für isotherme Atmosphären:

$$p = p_0 \cdot e^{-\frac{h}{z_0}} \Leftrightarrow h = z_0 \cdot ln(\frac{p_0}{p})$$

p_0 ist der Druck auf Meereshöhe (ca. 1000hPa), z_0 ist eine Skalenhöhe[13] von 8400 Meter. Da die gerasterten Modelldaten lediglich einen Mittelwert für die Höhe jeder Zelle enthalten, kann es passieren, dass Erhebungen von geringer Ausdehnung nicht sehr stark berücksichtigt werden. Der resultierende Höhenwert wird daher mit der Höhe der Station verglichen. Abschließend wird der Höhenlevel verwendet, der der Stationshöhe am nächsten ist.

Zuletzt werden die gerasterten Daten auf die Koordinaten der Station linear interpoliert. Es bleibt somit lediglich die Zeitachse als einzige Dimension der Daten übrig.

Die matplotlib-Funktion `axes.plot_date` erlaubt es, abschließend alle Daten in einen Graphen zu übertragen. Als x-Achse wird hierbei automatisch eine zeitliche Einteilung verwendet.

Zusätzlich stelle ich die Korrelation der Daten in einer zweiten Grafik dar. Hierbei verwende ich einen einfachen Punktplot in dem Mess- und Modellwert gegeneinander aufgetragen werden. Zur Berechnung einer linearen Ausgleichsgeraden verwende ich die numpy-Funktion `numpy.ma.polyfit`:

```
function = numpy.ma.polyfit(fit_measurement,model,1) #Eine
    lineare Funktion bestimmen, die Messung und Modell
    approximiert
```

[12]Bei der *hybrid sigma*-Darstellung wird jeder Höhenlevel über einen prozentualen Anteil aus Boden- und Referenzdruck spezifiziert.

[13]Mit jeder Höhenzunahme um die Skalenhöhe nimmt der Luftdruck um einen Faktor von $e \approx 2.72$ ab. 8400 Meter erhält man, wenn man von einer Durchschnittstemperatur von $15°C$ für die Atmosphäre ausgeht.

```
x = range(0,int(maxi)+4) #x-Werte zum Darstellen der gerade
    erstellten (maxi ist der maximale Datenwert von Modell
    und Messung)
ax.plot(x,numpy.polyval(function,x),color+"--") #Die
    Funktion an allen x-Werten auswerten und darstellen
```

Die Grafiken können als PNG, EPS oder SVG gerendert und heruntergeladen werden.

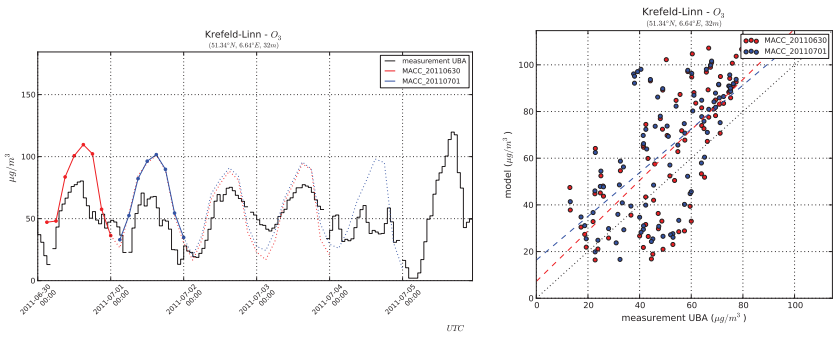

Abbildung 9.15: Generierte Comparison-Plots zweier MACC-Datensätze mit Daten des Umweltbundesamtes. Links: Darstellung als Zeitverlauf, rechts: Darstellung der Korrelation

9.6 Betrieb des Webinterfaces

Neben den in Kapitel 8 genannten Paketen[14] werden für den Betrieb des Webinterface-Servers zusätzlich auch die `datafed`-Pakete benötigt, welche in [Decker, 2011] entwickelt wurden (siehe Seite 82).

Zudem müssen die MySQL-Tabellen (siehe Abschnitt 11.3, Seite 151ff) und der MySQL-Benutzer `interface` eingerichtet werden.

Der Benutzeraccount des Betriebssystems, unter dem der Server betrieben wird, muss für die Verzeichnisse `filesystem` und `mapCache` Schreibrechte besitzen, um die Dateiverwaltung und generierte Kacheln zu sichern.

Clientseitig wurde das Webinterface für den Internet-Explorer (Version 7 oder höher), Firefox (Version 3.6 oder höher), Opera (Version 10 oder höher) und Safari (Version 5 oder höher) getestet. Auch Chrome sollte in seiner aktuellsten Ausführung die verwendeten Webstandards unterstützen.

[14] Alle Pakete sollten in der neuesten Version verwendet werden. Für Python muss der 2.6+ Entwicklungszweig installiert sein.

Der Server kann anschließend über

```
python start.py 50080
```

gestartet werden. Wobei 50080 der Port-Nummer entspricht.

Alternativ kann auch

```
./start.sh
```

verwendet werden. Dies bewirkt, dass der Server in einer Endlosschleife immer wieder neu gestartet wird, sollte er einmal aufgrund eines Fehlers abstürzen.

Das Webinterface des Forschungszentrum Jülich ist zum Zeitpunkt der Abgabe über die Adresse: `http://macc.icg.kfa-juelich.de:50080` erreichbar.

Weitere Layout Versionen werden unter `http://htap.icg.kfa-juelich.de:50080` und `http://ogc-interface.icg.kfa-juelich.de:50080` angeboten. Diese Versionen unterscheiden sich lediglich in der verwendeten CSS-Datei.

Kapitel 10

Fazit

10.1 Zusammenfassung

Mit Hilfe von Python und JavaScript habe ich im Rahmen dieser Arbeit ein Webinterface implementiert, das es erlaubt, Daten von Servern abzurufen, die das WCS-Protokoll (in Version 1.1.2) unterstützen. Zudem kann der Nutzer mit Hilfe des Workspace-Bereiches eine schnelle Übersicht über alle seine geladenen Dateien erhalten. Auch der Download auf seinen lokalen Rechner wird über diesen Bereich ermöglicht.

Der Plot-Bereich erlaubt dank OpenLayers eine schnelle Ansicht und Kombination unterschiedlicher Datensätze. Eine Vielzahl an Einstellungsmöglichkeiten erlaubt es dem Nutzer zudem eine personalisierte Grafik zu erzeugen. Jedoch wurde auch ersichtlich, dass die Datensätze konkrete Spezifikationen (beispielsweise die CF-Konventionen, siehe Abschnitt 9.2.1) einhalten müssen, um für die Plotroutinen les- und darstellbar zu sein. Neue WCS-Server, die an das Interface angebunden werden, müssen somit auf diese Standardkonformität überprüft werden.

Die Comparison-Ansicht ermöglicht die einfache Validierung von Modelldaten gegenüber entsprechenden Messwerten. Durch die Festlegung auf Messdaten des Umweltbundesamtes steht dieser Vergleich vorerst nur mit deutschen Messstationen zur Verfügung.

Alle Bereiche wurden als webbasiertes Softwaretool implementiert. Der Nutzer benötigt somit lediglich einen Browser, um die vorgestellte Anwendung produktiv einsetzen zu können. Bei der Implementierung habe ich versucht die Webstandards des W3C[1] einzuhalten, um auch für kommende Browsergenerationen einen reibungslosen Betrieb des Webinterfaces zu ermöglichen.

[1] *World Wide Web Consortium*, `http://www.w3.org/`

10.2 Ausblick

Die einzelnen Bereiche des Softwaretools können um weitere Funktionen ergänzt werden:

- Für den Access-Bereich soll eine Möglichkeit geschaffen werden, sodass der Nutzer auch selbstständig eigene WCS-Server festlegen und Daten von diesen Servern herunterladen kann.
- Der Comparison-Bereich soll mit einem Zugang zu den Messdaten der europäischen Umweltagentur erweitert werden. Dem Nutzer soll somit eine breitere Datenbasis zur Verfügung gestellt werden.
- Im Plot-Bereich sollen weitere Plot-Typen, beispielsweise ein- und zweidimensionale statische Darstellungen, ergänzt und weitere Einstellungsmöglichkeiten geschaffen werden, wie zum Beispiel die Änderung der Plotbeschriftung.
- Der Workspace-Bereich soll um Operatoren erweitert werden, mit deren Hilfe sich Interpolationen auf andere Rastergrößen, Mittelwertberechnungen über mehrere Modelle oder auch eine Mittelwertbildung über einzelne Dimensionen einer Datei generieren lassen.
- Alle Nutzereinstellungen und Daten existieren momentan lediglich solange der Browser geöffnet bleibt, da beim Schließen des Fensters der `Session`-Cookie und damit die Verbindung zur Dateiverwaltung entfernt wird. Hier wäre es denkbar eine Möglichkeit zu schaffen, sodass alle Informationen der Verwaltung dem Nutzer als XML-Datei zum Download angeboten werden. Es wäre damit anschließend möglich, einen kompletten Workspace wiederherzustellen.

Kapitel 11

Anhang

11.1 Patch des basemap-Toolkits

Das basemap-Toolkit weist in der verwendeten Version 1.0.1 einen Fehler in Zusammenhang mit kleinen Ausschnitten auf der Weltkarte auf. Die originale Implementierung verwendet jeweils nur Datensätze die maximal 10% der Breite des angeforderten Ausschnitts entfernt vom Rand des Ausschnitts liegen. Bei einer großen Rasterbreite kann es somit passieren, dass in Konturplots Darstellungslücken entstehen. Sollen zum Beispiel 20° bis 45° in x-Richtung abgerufen werden, bei einer Raster-Auflösung von 10°, so ergibt sich nach der bisherigen Formel

$$45^\circ + 0.1 \cdot (45^\circ - 20^\circ) = 47.5^\circ$$

Der nächste Datenwert des Rasters befindet sich jedoch erst bei 50° (in diesem Beispiel gehe ich davon aus, dass das Raster genau 0° durchläuft) und somit wird für den gesamten Bereich zwischen 40° und 45° kein Datenwert eingetragen, obwohl dieser Bereich mit abgefragt wurde.

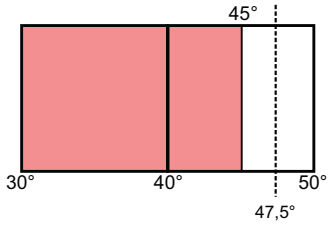

Abbildung 11.1: Beispiel zum Auftreten des Fehlers im basemap-Toolkit: Rot ist der angefragte Bereich. 47.5°, als Maskierungsgrenze, liegt vor der nächsten Rastergrenze, die komplette rechte Rasterzelle wird somit nicht dargestellt.

Ich habe dieses Problem umgangen, indem ich immer die größte mögliche Abdeckung durch das Raster für einen Ausschnitt verwende. Im oberen Beispiel hätte ich somit die Datenwerte von 10° bis 50° verwendet. Hierdurch hat der Algorithmus immer genügend Daten, um auch die Zwischenwerte zu interpolieren.

Übersicht über die Unterschiede des Patches gegenüber dem Original:

```
--- __init__.py.org     2011-02-09 20:53:32.000000000 +0100
+++ __init__.py 2011-04-13 14:05:32.238311540 +0200
@@ -3060,12 +3060,30 @@
    # mask for points outside projection limb.
    xymask = np.logical_or(np.greater(x,1.e20),np.greater(y
       ,1.e20))
    # mask outside projection region (workaround for
       contourf bug?)
-   epsx = 0.1*(self.xmax-self.xmin)
-   epsy = 0.1*(self.ymax-self.ymin)
-   outsidemask = np.logical_or(np.logical_or(x > self.xmax+
   epsx,\
-           x < self.xmin-epsy),\
-           np.logical_or(y > self.ymax+epsy,\
-           y < self.ymin-epsy))
+
+  #generate right side of mask
+  s = (x <= self.xmax).sum(1) #count all the elements line
   by line, where: element<=self.xmax
+  s = np.max(s+1*(s<x.shape[1])) #add one element every
   line to hide masked areas in plotting area
+  maskX_upper = np.concatenate((np.zeros((y.shape[0],s)),np
   .ones((y.shape[0],y.shape[1]-s))),axis=1)
+
+  #generate left side of mask
+  s = (x < self.xmin).sum(1)
+  s = np.max(s-1*(s>0))
+  maskX_lower = np.concatenate((np.ones((y.shape[0],s)),np.
   zeros((y.shape[0],y.shape[1]-s))),axis=1)
+
+  #generate top side of mask
+  s = (y <= self.ymax).sum(0)
+  s = np.max(s+1*(s<y.shape[0]))
+  maskY_upper = np.concatenate((np.zeros((s,x.shape[1])),np
   .ones((x.shape[0]-s,x.shape[1]))))
+
+
+  #generate bottom side of mask
+   s = (y < self.ymin).sum(0)
+   s = np.max(s-1*(s>0))
+   maskY_lower = np.concatenate((np.ones((s,x.shape[1])),np
```

```
    .zeros((x.shape[0]-s,x.shape[1])))
+
+   outsidemask = np.logical_or(np.logical_or(maskX_upper,
    maskX_lower),np.logical_or(maskY_upper,maskY_lower)) #
    combine all four masks
+
       data = ma.asarray(data)
       # combine masks.
       mask = \
```

11.2 Datei- und Verzeichnisübersicht

- access.py: Access-Seite zum Zugriff auf WCS-Server
- comparison.py: Comparison-Seite, Erstellung von Vergleichsgrafiken zwischen Modell und Messung
- const.py: Konstantensammlung und Einstellungen des Webinterfaces
- filesystem/: Order zur Verwaltung der Dateisysteme aller Nutzer
- filesystem.py: Verwaltung des Dateisystems
- mapCache/: Speicherordner aller gecachten Kartenkacheln
- plot.py: Visualisierungs-Seite
- plotTypes.py: Generierung von Kartenkacheln und statischen Kartenplots
- server.py: Zugriff auf eine Liste der zugreifbaren WCS-Server
- start.py: Startdatei des Webinterfaces und URL-Verwaltung
- start.sh: Startskript zum Ausführen der Serversoftware
- static/: Speicherort statischer Inhalte
  - jCode/: Sammlung aller JavaScript-Dateien
    - access.js: Skriptdatei der Access-Seite
    - comparison.js: Skriptdatei der Comparison-Seite
    - help.js: Inhalte aller Hilfe-Fenster
    - jquery-1.6.1.min.js: jQuery Skriptdatei
    - jquery-ui-1.8.11.custom.min.js: jQuery UI Skriptdatei
    - nav.js: Navigationsskripte
    - plot.js: Skriptdatei der Plot-Seite
    - set.js: Implementierung einer Menge in JavaScript
    - ui.selectmenu.js: Skriptdatei
    - upload.js: Skriptdatei der Upload-Seite
    - util.js: Sammlung nützlicher Hilfsfunktionen
    - workspace.js: Skriptdatei der Workspace-Seite

  - layout/: Sammlung aller Layout-Einstellungen und Grafiken
    - access.css: Layout der Access-Seite
    - comparison.css: Layout der Comparison-Seite
    - images/: Bilddaten
    - jquery-ui-1.8.11.custom.css: jQuery UI Layout
    - macc.css: MACC-Seiten Header Layout
    - option.css: Layout von Dialog-Fenstern
    - plot.css: Layout der Plot-Seite
    - print.css: Drucklayout
    - style.css: Grundlegendes Seitenlayout
    - ui.selectmenu.css: Selectmenü Layout
    - upload.css: Layout der Upload-Seite
    - workspace.css: Layout der Workspace-Seite
  - macc_mean_ps.dat: gemittelte Druckwerte aller MACC-Dateien zur Einheitenumrechnung
  - robots.txt: Einstellungen für Webrobots
- templates/: Speicherort aller HTML-Template Dateien
  - about_macc.html: Informationsseite zu den MACC-Daten
  - access.html: Aufbau der Access-Seite
  - comparison.html: Aufbau der Comparison-Seite
  - errorpage.html: Aufbau zur Übermittelung einer Upload-Fehlermeldung
  - main_layout/: Grundlegende Layout-Versionen
    - index_htap: Aufbau im Layout der HTAP-Seite
    - index: Aufbau im Layout der OGC Service-Seite
    - index_macc: Aufbau im Layout der MACC-Seite
  - plot.html: Aufbau der Plot-Seite
  - species_list_macc.html: Informationsseite zu den Variablen der MACC-Daten
  - start.html: Inhalt der Startseite
  - start_macc.html: Inhalt der MACC-Startseite
  - terms.html: Terms of use
  - uba.html: Informationstext zu den Daten des Umweltbundesamtes
  - upload.html: Aufbau der Upload-Seite
  - uploadStatus.html: Aufbau zur Übermittelung einer Statusmeldung
  - workspace.html: Aufbau der Workspace-Seite
- upload.py: Dateiupload neuer WCS-Dateien auf den Interface-Server
- util.py: Sammlung nützlicher Hilfsfunktionen
- wcs.py: Zugriff auf WCS-Server unter Verwendung des WCS-Protokolls
- workspace.py: Workspace-Seite

11.3 Aufbau der verwendeten MySQL-Tabellen

filesystem

Field	Type	Null	Key	Default	Extra
owner_id	char(128)	No	PRI	NULL	
count	int(16)	No		0	

files

Field	Type	Null	Key	Default	Extra
owner_id	char(128)	No	PRI	NULL	
filename	char(255)	No	PRI	NULL	
type	enum('WCS','UPLOAD','IMG')	No		WCS	
hash	char(128)	Yes		NULL	
note	text	Yes		NULL	

wcs_files

Field	Type	Null	Key	Default	Extra
owner_id	char(128)	No	PRI	NULL	
filename	char(255)	No	PRI	NULL	
server	char(255)	No		NULL	
identifier_req	text	No		NULL	
lon_start_req	char(10)	No		NULL	
lon_end_req	char(10)	No		NULL	
lat_start_req	char(10)	No		NULL	
lat_end_req	char(10)	No		NULL	
timeRange_req	char(40)	Yes		NULL	
variable_req	text	Yes		NULL	
lon_start	float	Yes		NULL	
lon_end	float	Yes		NULL	
lat_starq	float	Yes		NULL	
lat_enq	float	Yes		NULL	
time_start	datetime	Yes		NULL	
time_end	datetime	Yes		NULL	
variable	text	Yes		NULL	

upload_files

Field	Type	Null	Key	Default	Extra
owner_id	char(128)	No	PRI	NULL	
filename	char(255)	No	PRI	NULL	
lon_start	float	Yes		NULL	
lon_end	float	Yes		NULL	
lat_starq	float	Yes		NULL	
lat_enq	float	Yes		NULL	
time_start	datetime	Yes		NULL	
time_end	datetime	Yes		NULL	
variable	text	Yes		NULL	

server

Field	Type	Null	Key	Default	Extra
server_url	char(255)	No	PRI	NULL	
server_name	char(255)	No	UNI	NULL	
version	char(10)	No		1.1.2	
valid	tinyint(1)	No		0	
sep	char(1)	Yes			
parts	text	Yes		NULL	
splitter	text	Yes		NULL	

sessions

Field	Type	Null	Key	Default	Extra
session_id	char(128)	No	PRI	NULL	
atime	timestamp	No		CURRENT_TIMESTAMP	
data	text	Yes		NULL	
upload_size	int(10) unsigned	Yes		NULL	
progress	int(5) unsigned	No		0	

map_plot

Field	Type	Null	Key	Default	Extra
owner_id	char(128)	No		NULL	
img_id	int(11)	No	PRI	NULL	auto_increment
mtime	datetime	No		NULL	
title	text	No			
basemap_type	char(15)	No		NULL	
river	tinyint(1)	No		0	
countries	tinyint(1)	No		0	
coastline	tinyint(1)	No		0	
grid	tinyint(1)	No		0	
lonBeg	float	no		NULL	
lonEnd	float	no		NULL	
latBeg	float	no		NULL	
latEnd	float	no		NULL	
level	int(11)	no		NULL	
time	int(11)	no		NULL	

layer

Field	Type	Null	Key	Default	Extra
plot_id	int(11)	No	PRI	0	
layer_order	int(11)	No	PRI	0	
filename	char(255)	No		NULL	
species	char(255)	No		NULL	
type	int(11)	No		NULL	
visible	tinyint(1)	No		NULL	
level	int(11)	No		NULL	
loc_level	tinyint(1)	No		NULL	
time	int(11)	No		NULL	
loc_time	tinyint(1)	No		NULL	
opacity	float	No		NULL	
nearest	tinyint(1)	No		NULL	
color	char(12)	No		NULL	
l_mask_active	tinyint(1)	No		NULL	
l_mask	float	No		NULL	
u_mask_active	tinyint(1)	No		NULL	
u_mask	float	No		NULL	
min_active	tinyint(1)	No		NULL	
min	float	No		NULL	
max_active	tinyint(1)	No		NULL	
max	float	No		NULL	
contourLevel	int(3)	No		NULL	
contourType	int(3)	No		NULL	

Literaturverzeichnis

[Decker 2011] Decker, Michael: *Web-basierte Bereitstellung, Verarbeitung und Verknüpfung georeferenzierter Atmosphärendaten*. 2011. – Masterarbeit Fachhochschule Aachen

[Ernesti und Kaiser 2009] Ernesti, Johannes ; Kaiser, Peter: *Python 3 Das umfassende Handbuch*. 2. aktualisierte Auflage. Galileo Press Bonn, 2009. – ISBN 978-3-8362-1412-4

[MACC] : *Monitoring atmospheric composition & climate - About the Project*. – URL `http://www.gmes-atmosphere.eu/about/`

[MPL] : *Matplotlib - intro*. – URL `http://matplotlib.sourceforge.net/`

[OGC] : *Open Geospatial Consortium - About*. – URL `http://www.opengeospatial.org/ogc`

[OPL] : *OpenLayers - API*. – URL `http://dev.openlayers.org/releases/OpenLayers-2.10/doc/apidocs/files/OpenLayers-js.html`

[Tosi 2009] Tosi, Sandro: *Matplotlib for Python Developers*. Erstauflage. Packt Publishing Ltd., Olton Birmingham UK, 2009. – ISBN 978-1-847197-90-0

[Vollendorf und Bongers 2010] Vollendorf, Maximilian ; Bongers, Frank: *jQuery Das Prxisbuch*. Erstauflage. Galileo Press Bonn, 2010. – ISBN 978-3-8362-1288-5

[WCS] : *Web Coverage Service - Implementation Standard*. – URL `http://www.opengeospatial.org/standards/wcs`

[Weigend 2006] Weigend, Michael: *Objektorientierte Programmierung mit Python*. 3. aktualisierte Auflage. REDLINE GmbH Heidelberg, 2006. – ISBN 978-3-8266-1660-0

[Welling und Thomson 2009] Welling, Luke ; Thomson, Laura: *PHP 5.3 & MySQL 5.1 Dynamische Webanwendungen von Einstieg bis E-Commerce*. Erstauflage. Markt + Technik Verlag, München, 2009. – ISBN 978-3-8272-4390-4

[Wenz 2009] Wenz, Christian: *JavaScript Das umfassende Handbuch*. 9. aktualisierte Auflage. Galileo Press Bonn, 2009. – ISBN 978-3-8362-1397-4

[Wetsch 2010] Wetsch, Elisabeth: *Einstieg in CSS Grundlagen und Praxis*. 2. aktualisierte Auflage. Galileo Press Bonn, 2010. – ISBN 978-3-8362-1466-7

[WPY] : *Web.py Cookbook*. – URL `http://webpy.org/cookbook`

1. **Einsatz von multispektralen Satellitenbilddaten in der Wasserhaushalts- und Stoffstrommodellierung – dargestellt am Beispiel des Rureinzugsgebietes**
   von C. Montzka (2008), XX, 238 Seiten
   ISBN: 978-3-89336-508-1

2. **Ozone Production in the Atmosphere Simulation Chamber SAPHIR**
   by C. A. Richter (2008), XIV, 147 pages
   ISBN: 978-3-89336-513-5

3. **Entwicklung neuer Schutz- und Kontaktierungsschichten für Hochtemperatur-Brennstoffzellen**
   von T. Kiefer (2008), 138 Seiten
   ISBN: 978-3-89336-514-2

4. **Optimierung der Reflektivität keramischer Wärmedämmschichten aus Yttrium-teilstabilisiertem Zirkoniumdioxid für den Einsatz auf metallischen Komponenten in Gasturbinen**
   von A. Stuke (2008), X, 201 Seiten
   ISBN: 978-3-89336-515-9

5. **Lichtstreuende Oberflächen, Schichten und Schichtsysteme zur Verbesserung der Lichteinkopplung in Silizium-Dünnschichtsolarzellen**
   von M. Berginski (2008), XV, 171 Seiten
   ISBN: 978-3-89336-516-6

6. **Politikszenarien für den Klimaschutz IV – Szenarien bis 2030**
   hrsg.von P. Markewitz, F. Chr. Matthes (2008), 376 Seiten
   ISBN 978-3-89336-518-0

7. **Untersuchungen zum Verschmutzungsverhalten rheinischer Braunkohlen in Kohledampferzeugern**
   von A. Schlüter (2008), 164 Seiten
   ISBN 978-3-89336-524-1

8. **Inorganic Microporous Membranes for Gas Separation in Fossil Fuel Power Plants**
   by G. van der Donk (2008), VI, 120 pages
   ISBN: 978-3-89336-525-8

9. **Sinterung von Zirkoniumdioxid-Elektrolyten im Mehrlagenverbund der oxidkeramischen Brennstoffzelle (SOFC)**
   von R. Mücke (2008), VI, 165 Seiten
   ISBN: 978-3-89336-529-6

10. **Safety Considerations on Liquid Hydrogen**
    by K. Verfondern (2008), VIII, 167 pages
    ISBN: 978-3-89336-530-2

11. **Kerosinreformierung für Luftfahrtanwendungen**
von R. C. Samsun (2008), VII, 218 Seiten
ISBN: 978-3-89336-531-9

12. **Der 4. Deutsche Wasserstoff Congress 2008 – Tagungsband**
hrsg. von D. Stolten, B. Emonts, Th. Grube (2008), 269 Seiten
ISBN: 978-3-89336-533-3

13. **Organic matter in Late Devonian sediments as an indicator for environmental changes**
by M. Kloppisch (2008), XII, 188 pages
ISBN: 978-3-89336-534-0

14. **Entschwefelung von Mitteldestillaten für die Anwendung in mobilen Brennstoffzellen-Systemen**
von J. Latz (2008), XII, 215 Seiten
ISBN: 978-3-89336-535-7

15. **RED-IMPACT**
Impact of Partitioning, Transmutation and Waste Reduction Technologies on the Final Nuclear Waste Disposal
SYNTHESIS REPORT
ed. by W. von Lensa, R. Nabbi, M. Rossbach (2008), 178 pages
ISBN 978-3-89336-538-8

16. **Ferritic Steel Interconnectors and their Interactions with Ni Base Anodes in Solid Oxide Fuel Cells (SOFC)**
by J. H. Froitzheim (2008), 169 pages
ISBN: 978-3-89336-540-1

17. **Integrated Modelling of Nutrients in Selected River Basins of Turkey**
Results of a bilateral German-Turkish Research Project
project coord. M. Karpuzcu, F. Wendland (2008), XVI, 183 pages
ISBN: 978-3-89336-541-8

18. **Isotopengeochemische Studien zur klimatischen Ausprägung der Jüngeren Dryas in terrestrischen Archiven Eurasiens**
von J. Parplies (2008), XI, 155 Seiten, Anh.
ISBN: 978-3-89336-542-5

19. **Untersuchungen zur Klimavariabilität auf dem Tibetischen Plateau - Ein Beitrag auf der Basis stabiler Kohlenstoff- und Sauerstoffisotope in Jahrringen von Bäumen waldgrenznaher Standorte**
von J. Griessinger (2008), XIII, 172 Seiten
ISBN: 978-3-89336-544-9

20. **Neutron-Irradiation + Helium Hardening & Embrittlement Modeling of 9%Cr-Steels in an Engineering Perspective (HELENA)**
by R. Chaouadi (2008), VIII, 139 pages
ISBN: 978-3-89336-545-6

21. **in Bearbeitung**

22. **Verbundvorhaben APAWAGS (AOEV und Wassergenerierung) – Teilprojekt: Brennstoffreformierung – Schlussbericht**
von R. Peters, R. C. Samsun, J. Pasel, Z. Porš, D. Stolten (2008), VI, 106 Seiten
ISBN: 978-3-89336-547-0

23. **FREEVAL**
Evaluation of a Fire Radiative Power Product derived from Meteosat 8/9 and Identification of Operational User Needs
Final Report
project coord. M. Schultz, M. Wooster (2008), 139 pages
ISBN: 978-3-89336-549-4

24. **Untersuchungen zum Alkaliverhalten unter Oxycoal-Bedingungen**
von C. Weber (2008), VII, 143, XII Seiten
ISBN: 978-3-89336-551-7

25. **Grundlegende Untersuchungen zur Freisetzung von Spurstoffen, Heißgaschemie, Korrosionsbeständigkeit keramischer Werkstoffe und Alkalirückhaltung in der Druckkohlenstaubfeuerung**
von M. Müller (2008), 207 Seiten
ISBN: 978-3-89336-552-4

26. **Analytik von ozoninduzierten phenolischen Sekundärmetaboliten in *Nicotiana tabacum* L. cv Bel W3 mittels LC-MS**
von I. Koch (2008), III, V, 153 Seiten
ISBN 978-3-89336-553-1

27. **IEF-3 Report 2009. Grundlagenforschung für die Anwendung**
(2009), ca. 230 Seiten
ISBN: 978-3-89336-554-8

28. **Influence of Composition and Processing in the Oxidation Behavior of MCrAlY-Coatings for TBC Applications**
by J. Toscano (2009), 168 pages
ISBN: 978-3-89336-556-2

29. **Modellgestützte Analyse signifikanter Phosphorbelastungen in hessischen Oberflächengewässern aus diffusen und punktuellen Quellen**
von B. Tetzlaff (2009), 149 Seiten
ISBN: 978-3-89336-557-9

30. **Nickelreaktivlot / Oxidkeramik – Fügungen als elektrisch isolierende Dichtungskonzepte für Hochtemperatur-Brennstoffzellen-Stacks**
von S. Zügner (2009), 136 Seiten
ISBN: 978-3-89336-558-6

31. **Langzeitbeobachtung der Dosisbelastung der Bevölkerung in radioaktiv kontaminierten Gebieten Weißrusslands – Korma-Studie**
von H. Dederichs, J. Pillath, B. Heuel-Fabianek, P. Hill, R. Lennartz (2009), Getr. Pag.
ISBN: 978-3-89336-532-3

32. **Herstellung von Hochtemperatur-Brennstoffzellen über physikalische Gasphasenabscheidung**
von N. Jordán Escalona (2009), 148 Seiten
ISBN: 978-3-89336-532-3

33. **Real-time Digital Control of Plasma Position and Shape on the TEXTOR Tokamak**
by M. Mitri (2009), IV, 128 pages
ISBN: 978-3-89336-567-8

34. **Freisetzung und Einbindung von Alkalimetallverbindungen in kohlebefeuerten Kombikraftwerken**
von M. Müller (2009), 155 Seiten
ISBN: 978-3-89336-568-5

35. **Kosten von Brennstoffzellensystemen auf Massenbasis in Abhängigkeit von der Absatzmenge**
von J. Werhahn (2009), 242 Seiten
ISBN: 978-3-89336-569-2

36. **Einfluss von Reoxidationszyklen auf die Betriebsfestigkeit von anodengestützten Festoxid-Brennstoffzellen**
von M. Ettler (2009), 138 Seiten
ISBN: 978-3-89336-570-8

37. **Großflächige Plasmaabscheidung von mikrokristallinem Silizium für mikromorphe Dünnschichtsolarmodule**
von T. Kilper (2009), XVII, 154 Seiten
ISBN: 978-3-89336-572-2

38. **Generalized detailed balance theory of solar cells**
by T. Kirchartz (2009), IV, 198 pages
ISBN: 978-3-89336-573-9

39. **The Influence of the Dynamic Ergodic Divertor on the Radial Electric Field at the Tokamak TEXTOR**
von J. W. Coenen (2009), xii, 122, XXVI pages
ISBN: 978-3-89336-574-6

40. **Sicherheitstechnik im Wandel Nuklearer Systeme**
von K. Nünighoff (2009), viii, 215 Seiten
ISBN: 978-3-89336-578-4

41. **Pulvermetallurgie hochporöser NiTi-Legierungen für Implanat- und Dämpfungsanwendungen**
von M. Köhl (2009), XVII, 199 Seiten
ISBN: 978-3-89336-580-7

42. **Einfluss der Bondcoatzusammensetzung und Herstellungsparameter auf die Lebensdauer von Wärmedämmschichten bei zyklischer Temperaturbelastung**
von M. Subanovic (2009), 188, VI Seiten
ISBN: 978-3-89336-582-1

43. **Oxygen Permeation and Thermo-Chemical Stability of Oxygen Permeation Membrane Materials for the Oxyfuel Process**
by A. J. Ellett (2009), 176 pages
ISBN: 978-3-89336-581-4

44. **Korrosion von polykristallinem Aluminiumoxid (PCA) durch Metalljodidschmelzen sowie deren Benetzungseigenschaften**
von S. C. Fischer (2009), 148 Seiten
ISBN: 978-3-89336-584-5

45. **IEF-3 Report 2009. Basic Research for Applications**
(2009), 217 Seiten
ISBN: 978-3-89336-585-2

46. **Verbundvorhaben ELBASYS (Elektrische Basissysteme in einem CFK-Rumpf) - Teilprojekt: Brennstoffzellenabgase zur Tankinertisierung - Schlussbericht**
von R. Peters, J. Latz, J. Pasel, R. C. Samsun, D. Stolten
(2009), xi, 202 Seiten
ISBN: 978-3-89336-587-6

47. **Aging of ^{14}C-labeled Atrazine Residues in Soil: Location, Characterization and Biological Accessibility**
by N. D. Jablonowski (2009), IX, 104 pages
ISBN: 978-3-89336-588-3

48. **Entwicklung eines energetischen Sanierungsmodells für den europäischen Wohngebäudesektor unter dem Aspekt der Erstellung von Szenarien für Energie- und CO_2 - Einsparpotenziale bis 2030**
von P. Hansen (2009), XXII, 281 Seiten
ISBN: 978-3-89336-590-6

49. **Reduktion der Chromfreisetzung aus metallischen Interkonnektoren für Hochtemperaturbrennstofzellen durch Schutzschichtsysteme**
von R. Trebbels (2009), iii, 135 Seiten
ISBN: 978-3-89336-591-3

50. **Bruchmechanische Untersuchung von Metall / Keramik-Verbundsystemen für die Anwendung in der Hochtemperaturbrennstoffzelle**
von B. Kuhn (2009), 118 Seiten
ISBN: 978-3-89336-592-0

51. **Wasserstoff-Emissionen und ihre Auswirkungen auf den arktischen Ozonverlust**
Risikoanalyse einer globalen Wasserstoffwirtschaft
von T. Feck (2009), 180 Seiten
ISBN: 978-3-89336-593-7

52. **Development of a new Online Method for Compound Specific Measurements of Organic Aerosols**
by T. Hohaus (2009), 156 pages
ISBN: 978-3-89336-596-8

53. **Entwicklung einer FPGA basierten Ansteuerungselektronik für Justageeinheiten im Michelson Interferometer**
von H. Nöldgen (2009), 121 Seiten
ISBN: 978-3-89336-599-9

54. **Observation – and model – based study of the extratropical UT/LS**
by A. Kunz (2010), xii, 120, xii pages
ISBN: 978-3-89336-603-3

55. **Herstellung polykristalliner Szintillatoren für die Positronen-Emissions-Tomographie (PET)**
von S. K. Karim (2010), VIII, 154 Seiten
ISBN: 978-3-89336-610-1

56. **Kombination eines Gebäudekondensators mit H2-Rekombinatorelementen in Leichwasserreaktoren**
von S. Kelm (2010), vii, 119 Seiten
ISBN: 978-3-89336-611-8

57. **Plant Leaf Motion Estimation Using A 5D Affine Optical Flow Model**
by T. Schuchert (2010), X, 143 pages
ISBN: 978-3-89336-613-2

58. **Tracer-tracer relations as a tool for research on polar ozone loss**
by R. Müller (2010), 116 pages
ISBN: 978-3-89336-614-9

59. **Sorption of polycyclic aromatic hydrocarbon (PAH) to Yangtze River sediments and their components**
by J. Zhang (2010), X, 109 pages
ISBN: 978-3-89336-616-3

60. **Weltweite Innovationen bei der Entwicklung von CCS-Technologien und Möglichkeiten der Nutzung und des Recyclings von CO_2**
Studie im Auftrag des BMWi
von W. Kuckshinrichs et al. (2010), X, 139 Seiten
ISBN: 978-3-89336-617-0

61. **Herstellung und Charakterisierung von sauerstoffionenleitenden Dünnschichtmembranstrukturen**
von M. Betz (2010), XII, 112 Seiten
ISBN: 978-3-89336-618-7

62. **Politikszenarien für den Klimaschutz V – auf dem Weg zum Strukturwandel, Treibhausgas-Emissionsszenarien bis zum Jahr 2030**
hrsg. von P. Hansen, F. Chr. Matthes (2010), 276 Seiten
ISBN: 978-3-89336-619-4

63. **Charakterisierung Biogener Sekundärer Organischer Aerosole mit Statistischen Methoden**
von C. Spindler (2010), iv, 163 Seiten
ISBN: 978-3-89336-622-4

64. **Stabile Algorithmen für die Magnetotomographie an Brennstoffzellen**
von M. Wannert (2010), ix, 119 Seiten
ISBN: 978-3-89336-623-1

65. **Sauerstofftransport und Degradationsverhalten von Hochtemperaturmembranen für CO2-freie Kraftwerke**
von D. Schlehuber (2010), VII, 139 Seiten
ISBN: 978-3-89336-630-9

66. **Entwicklung und Herstellung von foliengegossenen, anodengestützten Festoxidbrennstoffzellen**
von W. Schafbauer (2010), VI, 164 Seiten
ISBN: 978-3-89336-631-6

67. **Disposal strategy of proton irradiated mercury from high power spallation sources**
by S. Chiriki (2010), xiv, 124 pages
ISBN: 978-3-89336-632-3

68. **Oxides with polyatomic anions considered as new electrolyte materials for solid oxide fuel cells (SOFCs)**
by O. H. Bin Hassan (2010), vii, 121 pages
ISBN: 978-3-89336-633-0

69. **Von der Komponente zum Stack: Entwicklung und Auslegung von HT-PEFC-Stacks der 5 kW-Klasse**
von A. Bendzulla (2010), IX, 203 Seiten
ISBN: 978-3-89336-634-7

70. **Satellitengestützte Schwerewellenmessungen in der Atmosphäre und Perspektiven einer zukünftigen ESA Mission (PREMIER)**
von S. Höfer (2010), 81 Seiten
ISBN: 978-3-89336-637-8

71. **Untersuchungen der Verhältnisse stabiler Kohlenstoffisotope in atmosphärisch relevanten VOC in Simulations- und Feldexperimenten**
von H. Spahn (2010), IV, 210 Seiten
ISBN: 978-3-89336-638-5

72. **Entwicklung und Charakterisierung eines metallischen Substrats für nanostrukturierte keramische Gastrennmembranen**
von K. Brands (2010), vii, 137 Seiten
ISBN: 978-3-89336-640-8

73. **Hybridisierung und Regelung eines mobilen Direktmethanol-Brennstoffzellen-Systems**
von J. Chr. Wilhelm (2010), 220 Seiten
ISBN: 978-3-89336-642-2

74. **Charakterisierung perowskitischer Hochtemperaturmembranen zur Sauerstoffbereitstellung für fossil gefeuerte Kraftwerksprozesse**
von S.A. Möbius (2010) III, 208 Seiten
ISBN: 978-3-89336-643-9

75. **Characterization of natural porous media by NMR and MRI techniques: High and low magnetic field studies for estimation of hydraulic properties**
by L.-R. Stingaciu (2010), 96 pages
ISBN: 978-3-89336-645-3

76. **Hydrological Characterization of a Forest Soil Using Electrical Resistivity Tomography**
by Chr. Oberdörster (2010), XXI, 151 pages
ISBN: 978-3-89336-647-7

77. **Ableitung von atomarem Sauerstoff und Wasserstoff aus Satellitendaten und deren Abhängigkeit vom solaren Zyklus**
von C. Lehmann (2010), 127 Seiten
ISBN: 978-3-89336-649-1

78. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Speeches and Plenary Talks
ed. by D. Stolten, B. Emonts (2010)
ISBN: 978-3-89336-658-3

78-1. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Parallel Sessions Book 1:
Fuel Cell Basics / Fuel Infrastructures
ed. by D. Stolten, T. Grube (2010), ca. 460 pages
ISBN: 978-3-89336-651-4

78-2. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Parallel Sessions Book 2:
Hydrogen Production Technologies – Part 1
ed. by D. Stolten, T. Grube (2010), ca. 400 pages
ISBN: 978-3-89336-652-1

78-3. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Parallel Sessions Book 3:
Hydrogen Production Technologies – Part 2
ed. by D. Stolten, T. Grube (2010), ca. 640 pages
ISBN: 978-3-89336-653-8

78-4. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Parallel Sessions Book 4:
Storage Systems / Policy Perspectives, Initiatives and Cooperations
ed. by D. Stolten, T. Grube (2010), ca. 500 pages
ISBN: 978-3-89336-654-5

78-5. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Parallel Sessions Book 5:
Stategic Analysis / Safety Issues / Existing and Emerging Markets
ed. by D. Stolten, T. Grube (2010), ca. 530 pages
ISBN: 978-3-89336-655-2

78-6. **18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings**
Parallel Sessions Book 6:
Stationary Applications / Transportation Applications
ed. by D. Stolten, T. Grube (2010), ca. 330 pages
ISBN: 978-3-89336-656-9

78 Set (complete book series)

18th World Hydrogen Energy Conference 2010 – WHEC2010 Proceedings
ed. by D. Stolten, T. Grube, B. Emonts (2010)
ISBN: 978-3-89336-657-6

79. **Ultrafast voltex core dynamics investigated by finite-element micromagnetic simulations**
by S. Gliga (2010), vi, 144 pages
ISBN: 978-3-89336-660-6

80. **Herstellung und Charakterisierung von keramik- und metallgestützten Membranschichten für die CO_2-Abtrennung in fossilen Kraftwerken**
von F. Hauler (2010), XVIII, 178 Seiten
ISBN: 978-3-89336-662-0

81. **Experiments and numerical studies on transport of sulfadiazine in soil columns**
by M. Unold (2010), xvi, 115 pages
ISBN: 978-3-89336-663-7

82. **Prompt-Gamma-Neutronen-Aktivierungs-Analyse zur zerstörungsfreien Charakterisierung radioaktiver Abfälle**
von J.P.H. Kettler (2010), iv, 205 Seiten
ISBN: 978-3-89336-665-1

83. **Transportparameter dünner geträgerter Kathodenschichten der oxidkeramischen Brennstoffzelle**
von C. Wedershoven (2010), vi, 137 Seiten
ISBN: 978-3-89336-666-8

84. **Charakterisierung der Quellverteilung von Feinstaub und Stickoxiden in ländlichem und städtischem Gebiet**
von S. Urban (2010), vi, 211 Seiten
ISBN: 978-3-89336-669-9

85. **Optics of Nanostructured Thin-Film Silicon Solar Cells**
by C. Haase (2010), 150 pages
ISBN: 978-3-89336-671-2

86. **Entwicklung einer Isolationsschicht für einen Leichtbau-SOFC-Stack**
von R. Berhane (2010), X, 162 Seiten
ISBN: 978-3-89336-672-9

87. **Hydrogen recycling and transport in the helical divertor of TEXTOR**
by M. Clever (2010), x, 172 pages
ISBN: 978-3-89336-673-6

88. **Räumlich differenzierte Quantifizierung der N- und P-Einträge in Grundwasser und Oberflächengewässer in Nordrhein-Westfalen unter besonderer Berücksichtigung diffuser landwirtschaftlicher Quellen**
von F. Wendland et. al. (2010), xii, 216 Seiten
ISBN: 978-3-89336-674-3

89. **Oxidationskinetik innovativer Kohlenstoffmaterialien hinsichtlich schwerer Lufteinbruchstörfälle in HTR´s und Graphitentsorgung oder Aufarbeitung**
von B. Schlögl (2010), ix, 117 Seiten
ISBN: 978-3-89336-676-7

90. **Chemische Heißgasreinigung bei Biomassenvergasungsprozessen**
von M. Stemmler (2010), xv, 196 Seiten
ISBN: 978-3-89336-678-1

91. **Untersuchung und Optimierung der Serienverschaltung von Silizium-Dünnschicht-Solarmodulen**
von S. Haas (2010), ii, 202 Seiten
ISBN: 978-3-89336-680-4

92. **Non-invasive monitoring of water and solute fluxes in a cropped soil**
by S. Garré (2010), xxiv, 133 pages
ISBN: 978-3-89336-681-1

93. **Improved hydrogen sorption kinetics in wet ball milled Mg hydrides**
by L. Meng (2011), II, 119 pages
ISBN: 978-3-89336-687-3

94. **Materials for Advanced Power Engineering 2010**
ed. by J. Lecomte-Beckers, Q. Contrepois, T. Beck and B. Kuhn (2010), 1327 pages
ISBN: 978-3-89336-685-9

95. **2D cross-hole MMR – Survey design and sensitivity analysis for cross-hole applications of the magnetometric resistivity**
by D. Fielitz (2011), xvi, 123 pages
ISBN: 978-3-89336-689-7

96. **Untersuchungen zur Oberflächenspannung von Kohleschlacken unter Vergasungsbedingungen**
von T. Melchior (2011), xvii, 270 Seiten
ISBN: 978-3-89336-690-3

97. **Secondary Organic Aerosols: Chemical Aging, Hygroscopicity, and Cloud Droplet Activation**
by A. Buchholz (2011), xiv, 134 pages
ISBN: 978-3-89336-691-0

98. **Chrom-bezogene Degradation von Festoxid-Brennstoffzellen**
von A. Neumann (2011), xvi, 218 Seiten
ISBN: 978-3-89336-692-7

99. **Amorphous and microcrystalline silicon applied in very thin tandem solar cells**
by S. Schicho (2011), XII, 190 pages
ISBN: 978-3-89336-693-4

100. **Sol-gel and nano-suspension electrolyte layers for high performance solid oxide fuel cells**
by F. Han (2011), iv, 131 pages
ISBN: 978-3-89336-694-1

101. **Impact of different vertical transport representations on simulating processes in the tropical tropopause layer (TTL)**
by F. Plöger (2011), vi, 104 pages
ISBN: 978-3-89336-695-8

102. **Untersuchung optischer Nanostrukturen für die Photovoltaik mit Nahfeldmikroskopie**
von T. Beckers (2011), xiii, 128 Seiten
ISBN: 978-3-89336-696-5

103. **Impact of contamination on hydrogenated amorphous silicon thin films & solar cells**
by J. Wördenweber (2011), XIV, 138 pages
ISBN: 978-3-89336-697-2

104. **Water and Organic Nitrate Detection in an AMS: Laboratory Characterization and Application to Ambient Measurements**
by A. Mensah (2011), XI, 111 pages
ISBN: 978-3-89336-698-9

105. **Entwicklung eines neuen Konzepts zur Steuerung der thermischen Ausdehnung von glaskeramischen Verbundwerkstoffen mit angepasster Fließfähigkeit am Beispiel der Hochtemperatur-Brennstoffzelle**
von E. Wanko (2011), xi, 134 Seiten
ISBN: 978-3-89336-705-4

106. **Tomographic reconstruction of atmospheric volumes from infrared limb-imager measurements**
by J. Ungermann (2011), xiv, 153 pages
ISBN: 978-3-89336-708-5

107. **Synthese und Identifizierung von substituierten Mg-Al-Cl Doppelhydroxidverbindungen mit Schwerpunkt IR-Spektroskopie**
von B. Hansen (2011), XII, 121 Seiten
ISBN: 978-3-89336-709-2

108. **Analysis of spatial soil moisture dynamics using wireless sensor networks**
by U. Rosenbaum (2011), xxii, 120 pages
ISBN: 978-3-89336-710-8

109. **Optimierung von APS-ZrO2-Wärmedämmschichten durch Variation der Kriechfestigkeit und der Grenzflächenrauhigkeit**
von M. E. Schweda (2011), 168 Seiten
ISBN: 978-3-89336-711-5

110. **Sorption of a branched nonylphenol isomer and perfluorooctanoic acid on geosorbents and carbon nanotubes**
by C. Li (2011), X, 102 pages
ISBN: 978-3-89336-716-0

111. **Electron Transport in the Plasma Edge with Rotating Resonant Magnetic Perturbations at the TEXTOR Tokamak**
by H. Stoschus (2011), iv, 113 pages
ISBN: 978-3-89336-718-4

112. **Diffusion and Flow Investigations in Natural Porous Media by Nuclear Magnetic Resonance**
by N. Spindler (2011), viii, 144 pages
ISBN: 978-3-89336-719-1

113. **Entwicklung und Erprobung des Hygrometer for Atmospheric Investigations**
von T. Klostermann (2011), IV, 118 Seiten
ISBN: 978-3-89336-723-8

114. **Application of functional gene arrays for monitoring influences of plant/seasons on bacterial functions and community structures in constructed wetlands (Bitterfeld, Germany)**
by J. Ning (2011), xiv, 157 pages
ISBN: 978-3-89336-724-5

115. **Wasseraustrag aus den Kathodenkanälen von Direkt-Methanol-Brennstoffzellen**
von A. Schröder (2011), VII, 228 Seiten
ISBN: 978-3-89336-727-6

116. **CITYZEN Climate Impact Studies**
by C. Richter, M. Schultz (2011), 45 pages
ISBN: 978-3-89336-729-0

117. **Software Tools zum interoperablen Austausch und zur Visualisierung von Geodatensätzen über das Internet**
von M. Schultz, M. Decker, S. Lührs (2011), iv, 156 Seiten
ISBN: 978-3-89336-730-6

Energie & Umwelt / Energy & Environment
Band / Volume 117
ISBN 978-3-89336-730-6

JÜLICH
FORSCHUNGSZENTRUM