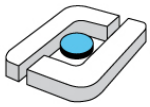

Hochschule Osnabrück
University of Applied Sciences

Fakultät Ingenieurwissenschaften und Informatik

Masterarbeit

über das Thema

Migration von ArcGIS-Geodatenbanken auf Freie Software mit PostgreSQL

vorgelegt durch

Christian Lins
<christian.lins@hs-osnabrueck.de>

Matrikelnummer: 322141

HOCHSCHULE OSNABRÜCK
Fakultät Ingenieurwissenschaften und Informatik

Masterarbeit

Thema:

Migration von ArcGIS-Geodatenbanken auf Freie Software mit PostgreSQL

für
Herrn Christian Lins geboren am: 08.11.84
in: Tübingen

Erstprüfer/in: Prof. Dr. rer. nat. Theodor Gervens

Zweitprüfer/in: Dipl. Systemwiss., M.Sc. (Geographie) Bernhard Reiter

Beginn: 22.02.2012

Abgabe: 22.08.2012

_____ _____

Erstprüfer Master

---

eingereicht am: Verlängerung genehmigt bis:

Zusammenfassung

Die proprietäre ArcGIS Softwarefamilie der Firma Esri ist eine weitverbreitete Standardsoftware für Geoinformationssysteme (GIS). Inzwischen gibt es *Freie Software*, die Teile des ArcGIS Ökosystems ersetzen könnte. Dem Anwender bietet diese *Freie Software* Vorteile wie z. B. hohe Flexibilität und viele Anpassungsmöglichkeiten der Anwendung, Wahlmöglichkeiten beim Softwaredienstleister und somit Herstellerunabhängigkeit.

Im Rahmen dieser Arbeit wird untersucht, ob und wie auf ArcGIS basierende Geoinformationssysteme auf Infrastrukturen mit freier Software migriert werden können. Der Schwerpunkt liegt dabei auf der Betrachtung der Geodatenbanken im Back-End der Dateninfrastruktur, insbesondere auf der Datenbank-Middleware ArcSDE von Esri.

Es werden prototypisch technische Ansätze entworfen, mit denen sich ArcSDE ersetzen bzw. sinnvoll an die freie Datenbank PostgreSQL und deren räumliche Erweiterung PostGIS anbinden lässt. Neben der technischen Realisierbarkeit werden auch Kriterien wie Nutzerakzeptanz, Aufwand und Nachhaltigkeit zur Bewertung der unterschiedlichen Migrationsszenarien betrachtet.

Dabei zeigt sich, dass sich das freie Datenbankmanagementsystem PostgreSQL problemlos mit ArcSDE verwenden lässt und dabei bspw. Oracle Database im Back-End ersetzen kann. Ein Verzicht auf die Middleware ArcSDE ist mit Einschränkungen möglich, denn die Untersuchung der Prototypen zeigt, dass ein ArcMap-Plugin zur direkten Anbindung an PostGIS geeignet ist.

Abstract

The proprietary ArcGIS software family is a widely used standard software for Geographic Information Systems (GIS). There is *Free Software* available that could replace parts of the ArcGIS ecosystem. *Free Software* has various advantages for the user, e.g. high flexibility and adaptability of the software, freedom to choose the software service company and therefor no dependency to one software vendor.

In this work we investigate if and how ArcGIS based GIS infrastructures can be migrated to solutions based on *Free Software*. Main subject is the geographic database in the backend of the data infrastructure, especially the database middleware ArcSDE by Esri.

Several prototypes are created to determine the possibilities of replacing ArcSDE or adapting the free database PostgreSQL and its spatial extension PostGIS to ArcGIS' desktop application ArcMap. In addition to technical feasibility criterias such as usability, user acceptance, complexity and sustainability are applied in order to evaluate migration scenarios.

The free database management system PostgreSQL can be used in conjunction with ArcSDE without major problems. So it can replace proprietary databases such as Oracle Database in the backend of the data infrastructure. A waiver of the middleware ArcSDE is possible with some restrictions for the investigation of the prototype shows that an ArcMap plug-in is suitable for direct connections to PostGIS.

Erklärung der eigenständigen Arbeit

Hiermit versichere ich, dass ich meine Masterarbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Christian Lins, 22. August 2012

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Freie Geoinformationssysteme als Ersatz für ArcGIS

Moderne Geoinformationssysteme (GIS) unterstützen viele Menschen bei ihrer täglichen Arbeit mit geografischen Daten. Die Liste der Einsatzbereiche für GIS-Software ist lang und umfasst u. a. Geo- und Kartographie, Militär- und Einsatzkräfte, Logistik und Resourcenplanung sowie Marketing. Endanwender kennen GIS wie z. B. *Google Maps* im Web oder auf dem Smartphone. Auch herkömmliche Fahrzeug-Navigationssysteme kann man als GIS auffassen. (vgl. [Brinkhoff, 2008, S. 1ff], [Wikipedia])

Das erste Allzweck-GIS entstand in den 60er Jahren des letzten Jahrhunderts in Kanada als *Canadian Geographic Information System* (CGIS). Kanada stand vor der Herausforderung, die enormen Landflächen des Staates sinnvoll zu verwalten und eine Übersicht über die vorhandenen natürlichen Ressourcen zu erhalten. Es wurde ein Forschungs- und Entwicklungsvorhaben zur Lösung des Problems finanziert. Der Geograph Roger Tomlinson war der führende Kopf bei der Entwicklung des CGIS und wurde später als Vater des GIS bezeichnet. Innovativ war hier die Verknüpfung der Methoden und Techniken aus der Geographie, mit denen aus der damals noch jungen Computertechnologie. Tomlinson wurden später für seine Verdienste zur Entwicklung des GIS mit dem *Order of Canada* Orden ausgezeichnet. (vgl. [Bill, 2010, S. 16] u. [The Governor General of Canada; URISA])

Marktführer im Bereich der GIS-Software ist - nach eigenen Angaben - die US-Firma Esri[1] mit dem Produkt ArcGIS (vgl. [Esri, h]). Mit der ArcGIS-Produktfamilie deckt sie den Markt in den Bereichen Desktop, Server (Back-End), Mobilgeräte und auch im Web umfassend ab und hat weltweit zahlreiche Kunden (Lizenznehmer) bei Behörden und Firmen. Esri stellt mit ArcGIS eine vollständige GIS-Verarbeitungskette bereit (also Desktop bis Back-End), deren Komponenten funktionsreich und eng verzahnt sind.

[1] Siehe `http://www.esri.com/`

Es existiert *Freie Software*, die einzelne Komponenten des proprietären ArcGIS-Ökosystems von Esri ersetzen könnte.

Freie Software zeichnet sich grundsätzlich durch folgende Eigenschaften aus (vgl. [Stallman] u. [Reiter, 2004, S. 83ff]):

1. Die Software darf ohne Einschränkungen genutzt werden (*no discrimination*). Die Lizenz der Software darf weder Personen und Gruppen diskrimieren noch Einsatzort- und zweck der Software einschränken.

2. Die Software (insbesondere ihr Quellcode) darf uneingeschränkt studiert und verändert werden (*source code*).

3. Die Software darf uneingeschränkt kopiert und verteilt werden (*free distribution*).

4. Die Software darf verändert und in veränderter Form mit gleicher Lizenz (oder je nach Lizenz auch mit anderer Lizenz) weitergegeben werden (*derived works*).

Als Richard Stallman, der als „last of the true hackers“ (vgl. [Levy, 1994, S. 435]) bezeichnet wurde, im Jahr 1984 zusammen mit juristischen Beratern die GNU General Public License (GPL) entwarf, wollte er sicherstellen, dass Software, die unter den Lizenzbedingungen der GPL steht, in ihrer Gesamtheit und für alle Zeit (solange der gesetzliche Urheberschutz greift) *freie* Software bleibt. Die GPL schränkt daher die Rechte für die Weitergabe der Software ein, maximiert dadurch aber die Freiheit der Nutzer. So schreibt die GPL vor, dass abgeleitete Werke, d. h. Software, die auf Basis von GPL-lizensierter Software entstanden ist, in ihrer Gesamtheit ebenfalls unter der GPL (oder einer Lizenz, welche die gleichen Rechte gewährt) veröffentlicht werden muss. Damit ist es nicht möglich GPL-Software mit unfreiem Code zu bündeln. Die GPL nutzt auf diese Weise das *Copyright*, um die Freiheit und Verwendbarkeit der Software sicherzustellen. In Anlehnung an Copy*right* nannte Stallman das Prinzip Copy*left*. (vgl. [Grassmuck, 2004, S. 282ff] u. [Stallman])

Andere Lizenzen für *Freie Software* wie z.B. die BSD- oder MIT-Lizenz erlauben es im Gegensatz zur GPL, dass die Software und von ihr abgeleitete Werke unter eine proprietäre (also unfreie) Lizenz gestellt werden. Dies erlaubt Entwicklern die *Freie Software* zu verändern, ohne den veränderten Quellcode der Allgemeinheit zur Verfügung stellen zu müssen. Damit sind solche BSD-artigen Lizenzen insbesondere für Firmen interessant, da diese nicht verpflichtet sind als Geschäftsgeheimnis betrachtete proprietäre Quellcodeteile zu veröffentlichen, wenn sie diese mit *Freier Software* verknüpfen.

Die beiden erwähnte Lizenztypen stellen nur die bekanntesten Vertreter dar, da es eine Fülle von Softwarelizenzen gibt (vgl. OSI), die zwar den Grundsätzen für FOSS genügen, in ihrer Ausprägung jedoch verschieden sind. Das Für und Wider von der beiden grundlegenden Lizenztypen für *Freie Software* wird in der Gemeinschaft seit vielen Jahren kontrovers diskutiert, da die vielgenutzte GPL mit vielen anderen Lizenzen nicht kompatibel ist.

Freie Software ist also Software, die - vereinfacht gesagt - lizenzkostenfrei genutzt und (deren Quellcode) modifiziert werden darf. Aufgrund der gewährten Freiheiten kann der Anwender die Software flexibel an seine speziellen Bedürfnisse anpassen (lassen). Für die Anpassung kann er einen kommerziellen Softwaredienstleister frei wählen und ist damit nicht - wie bei proprietärer Software - an einen Hersteller und dessen Produktzyklen gebunden (*Vendor Lock-In*, vgl. [Grassmuck, 2004, S. 343]).

Freie Software kann also eine Alternative zu proprietärer Software sein. Es stellt sich nun die Frage, warum Anwender trotz der Vorteile freier Software häufiger ArcGIS-Komponenten vorziehen. Zur Beantwortung dieser Frage muss geklärt werden, welche Softwarekomponenten zum Komplex ArcGIS gehören und welche Voraussetzungen aus Kundensicht erfüllt sein müssen, damit freie Softwareprodukte als Ersatz in Frage kommen können. Eine detaillierte Beschreibung des ArcGIS Ökosystems befindet sich in **Kapitel 2**. Ebenfalls dort befindet sich eine technische Übersicht über die für diese Arbeit relevanten Datenbankmanagementsysteme und ihrer räumlichen Erweiterungen.

Vermutlich sind die Gründe für die geringe Verbreitung von freien ArcGIS-Alternativen technischer (z. B. fehlende Softwarekomponenten) oder fachlicher Natur (z. B. fehlende oder falsch implementierte geografische Funktionalitäten) oder dem geringen Bekanntheitsgrad bei Anwendern und Auftraggebern geschuldet. Diese Vermutung wird in Kapitel 3 genauer - anhand von Interviews mit Anwendern - untersucht.

In **Kapitel 3** wird veranschaulicht, welche Schritte notwendig sind, um ArcGIS komplett oder zumindest teilweise durch *Freie Software* zu ersetzen. Dazu müssen die Komponenten des ArcGIS-Ökosystems (oder *Software Stack*), die durch *Freie Software* ersetzt bzw. *nicht* ersetzt werden können, identifiziert und untersucht werden. ArcGIS wird dabei von unten nach oben betrachtet, beginnend mit dem relationalen Datenbankmanagementsystem als Basis (Back-End) für die Datenbank-Middleware **ArcSDE**. Die Software ArcSDE stellt das Bindeglied zwischen der eigentlichen Datenbank und den GIS-Fachanwendungen auf dem Server und auf dem Desktop dar. Entscheidend für die

Überlegung, mit der Untersuchung am unteren Ende zu beginnen, ist, dass für eine Migration am Back-End keine Umstellungen beim Endnutzer des Geoinformationssystems nötig sind und sie damit vermeindlich einfacher durchzuführen ist. Ziel dieser Arbeit ist es also nicht, die Desktop-Clients von ArcGIS (ArcMap und ArcCatalog) unmittelbar zu ersetzen, sondern langfristig deren Austausch mittels Migration der Datenbanken auf *Freie Software* zu fördern.

Im Rahmen dieser Arbeit werden mehrere Ansätze untersucht, ArcSDE auszutauschen. Für jeden Ansatz wird prototypisch geprüft, wie eine freie Softwarekomponente als Ersatz dienen kann. Die Auswertung der einzelnen Versuchsprogramme hilft, die Realisierbarkeit einer vollen Softwareentwicklung einzuschätzen. Dies und eine abschließende Beurteilung der Migrationsbemühungen neben einem Ausblick auf weitere Untersuchungen befinden sich in **Kapitel 4**.

2 Räumliche Datenbanken und das ArcGIS-Geoinformationssystem

2.1 Datenbanksysteme

Ein Datenbanksystem (DBS) dient der dauerhaften Speicherung von Daten. Im DBS werden die Daten durch ein strukturiertes *Datenmodell* beschrieben und mit einer geeigneten *Datenbanksprache* verwaltet. (vgl. [Brinkhoff, 2008, S. 6])

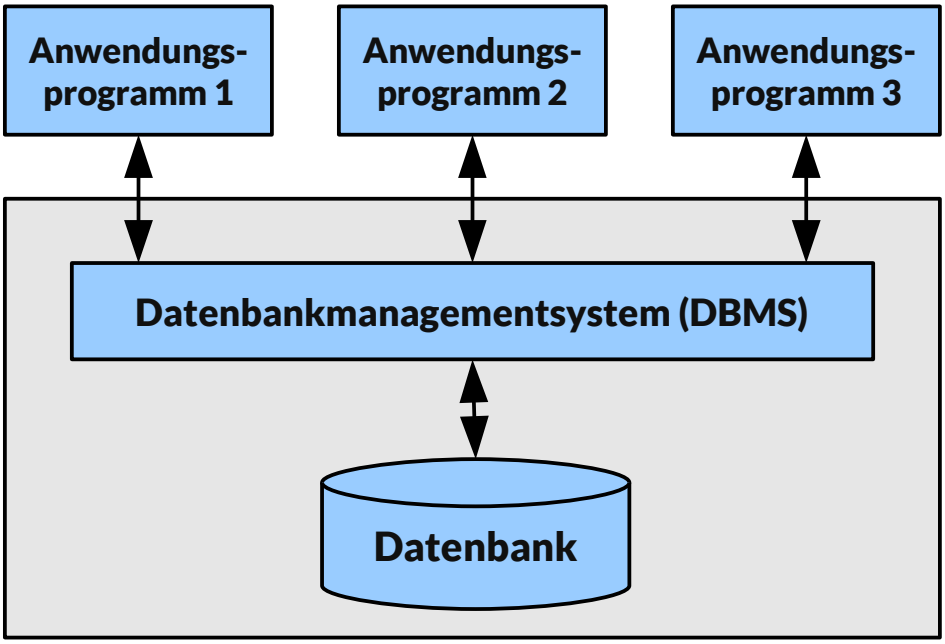

Abbildung 2.1: Schematischer Aufbau eines Datenbanksystems (nach [Brinkhoff, 2008, Abb. 1.4])

Das DBS besteht aus einem Datenbankmanagementsystem (DBMS), das als Softwarekomponente zum einen eine Schnittstelle für Anwendungsprogramme bereitstellt und zum anderen die Verwaltung der eigentlichen Daten in der *Datenbank* übernimmt (vgl. Abbildung 2.1).

2.1.1 Relationale Datenbanksysteme

Relationale Datenbanksysteme - als Ausprägung eines Datenbanksystems - organisieren Daten tabellarisch in *Relationen*. Jede Tabelle/Relation hat ein definiertes *Schema*, d. h. Spalten (*Attribute*) mit bestimmten Namen und Datentypen. Zulässige Datentypen sind beispielsweise verschiedene Zahltypen sowie Binär- und Zeichenfolgen. Die Datensätze sind Zeilen (*Tupel*) der Tabelle. Ein Datensatz ist in der Regel durch mindestens ein Attribut eindeutig identifizierbar, das als Primärschlüssel oder Primärschlüsselattribut bezeichnet wird. Über eindeutige Schlüsselattribute können Beziehungen zwischen Tabellen hergestellt werden, z. B. könnte in einer Tabelle PERSONAL (vgl. Tabelle 2.1) die Personalnummer als Primärschlüssel einen Angestellten eindeutig identifizieren und mit einer anderen Tabelle GEHAELTER in Verbindung bringen. In der Tabelle GEHAELTER (vgl. Tabelle 2.2) ist die Personalnummer dann der sogenannte Fremdschlüssel (*Foreign Key*), der auf genau einen Eintrag in PERSONAL verweist. (vgl. [Brinkhoff, 2008, S. 9ff])

Name : TEXT	**Vorname : TEXT**	**Personalnr. : INT (PK)**
Albrecht	Agnes	1
Meyer	Marcel	2
Schmidt	Sigmund	3
Stahnke	Stefanie	4

Tabelle 2.1: Beispielhafte Tabelle PERSONAL

XY : INT (PK)	**Gehalt : INT**	**Personalnr. : INT (FK)**
..	..	1
..	..	2
..	..	3
..	..	4

Tabelle 2.2: Beispielhafte Tabelle GEHAELTER

Über die standardisierte Abfragesprache SQL (Structured Query Language) lassen sich Datensätze aus den Tabellen einer Datenbank abfragen oder bearbeiten. Praktisch jedes aktuelle relationale Datenbanksystem kann mit SQL verwendet werden.

Das `SELECT`-Kommando dient zur Abfrage von Datensätzen. Grundsätzlich hat es in der einfachsten Form folgenden Aufbau:

```
SELECT Attribut[e] FROM [Datenbank.]Tabelle [WHERE <Bedingung[en]>]
```

Um beispielsweise die Namen aller Angestellten aus Tabelle 2.1 abzufragen, deren Vorname „Stefanie“ lautet, könnte folgende Abfrage verwendet werden:

```
SELECT Vorname,Nachname FROM personal WHERE Vorname = 'Stefanie'
```

Neben eher einfachen Abfragen, liegt die Stärke von SQL bei komplexen Abfragen mit Verbünden (*joins*) über mehrere Tabellen. Um beispielsweise die Namen aller Angestellten abzufragen, deren Gehalt mehr als 1000 beträgt, könnte folgende Abfrage verwendet werden:

```
SELECT Vorname,Nachname FROM personal
    NATURAL JOIN ON gehaelter
    WHERE gehalt > 1000
```

Auf ähnliche Weise könnte man die Angestellten mit einem geringeren Gehalt mit einer Gehaltserhöhung versehen. Dazu wird das `UPDATE`-Kommando verwendet. Für weitere Details zu SQL sei auf [Momjian, 2001] verwiesen.

Um eine SQL-Abfrage auszuführen, muss das DBMS u. U. sehr viele Datensätze betrachten und vergleichen. Zur Beschleunigung dieses Vorgangs werden Indizes eingesetzt. Ein Index ist i. d. R. eine Baumstruktur, welche alle indizierten Elemente enthält. Anschaulich lässt sich eine solche Struktur beim Sortieren von Zahlen erklären: angenommen wir haben zehn Zahlen aus dem Bereich von 0 bis 100, z. B.

```
21 4 83 23 10 5 2 89 99 12
```

und möchten eine bestimmte Zahl auswählen, deren Position in der Liste wir nicht kennen. Wenn die Liste an Zahlen nicht sortiert ist, müssen wir im ungünstigsten Fall jede Zahl der Liste auf unsere gesuchte Zahl prüfen. Der Aufwand steigt linear mit der Anzahl der Zahlen. Dies ist bei großen Datenmengen nicht akzeptabel. Daher wird die Menge der zu durchsuchenden Elemente *indiziert*. Dazu werden im Beispiel der Zahlenreihe die Zahlen in einer Baumstruktur angeordnet (siehe Abbildung 2.2).

Beim Aufbau des Baumes wurde das mittlere Element (die Zahl 21) als Wurzel gesetzt, der linke Ast enthält nur kleinere (d. h. < 21), der rechte Ast nur größere Elemente (d. h. > 21). Wird nun ein bestimmtes Element gesucht, kann der Baum von der Wurzel durchsucht werden und im ungünstigsten Fall (*Worst Case*) müssen in diesem Fall

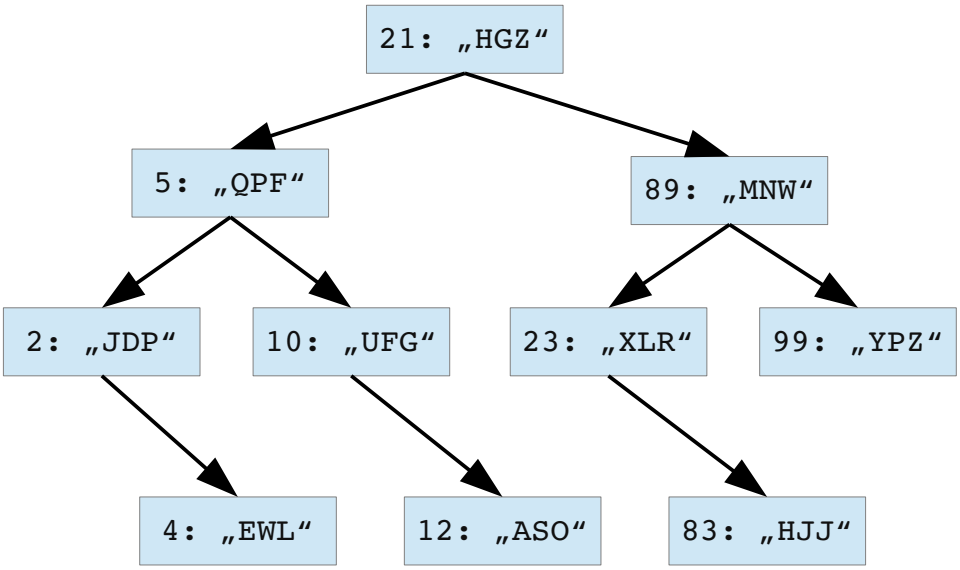

Abbildung 2.2: Baumstruktur eines Index

nur vier Elemente verglichen werden (im Vergleich zu zehn beim Beispiel der unsortierten Zahlenmenge). Dieses Verfahren lässt sich für variable Datentypen generalisieren. PostgreSQL beispielsweise implementiert einen GiST-Index (Generalized Search Tree), der sich mit nur sehr geringem Aufwand für benutzerdefinierte Datentypen (wie PostGIS-Geometrien) anpassen lässt (vgl. [PostgreSQL Global Development Group, b]).

2.1.2 Räumliche Datenbanksysteme

Relationale Datenbanksysteme können um die Unterstützung von räumlichen Datentypen und Verarbeitungsfunktionen erweitert werden.

Zwar können in relationalen Datenbanken prinzipiell beliebige Arten von Binärdaten gespeichert werden, die Datenbank erkennt in ihnen aber nur bedeutungslose Bytefolgen, d. h. die Semantik fehlt. Daher wurden räumliche Datenbanken entwickelt, die neben den bekannten Zahl-, Binär- und Zeichentypen auch spezielle Datentypen für räumliche (geometrische) Attribute kennen, z. B. Punkte, Linienzüge und Flächen. Damit diese neuen Typen auch mit Hilfe des Datenbanksystems verarbeitet werden können, stellt ein räumliches Datenbanksystem auch entsprechende Verarbeitungsfunktionen zur Verfügung, z. B. zum Verschnitt von Flächen oder um den Abstand zwischen zwei Punkten zu bestimmen.

2.1.2.1 Modellierung von Geodaten

Wer Informationen der realen Welt rechnergestützt verarbeiten möchte (beispielsweise in einer räumlichen Datenbank), muss diese Informationen in einem geeigneten Modell abbilden können. Die anhand dieses Modells gespeicherten Informationen bilden die *Geodaten*. Diese können dabei nie eine exakte Repräsentation der realen Umgebung sein und stellen immer eine Abstraktion dar. (vgl. [Brinkhoff, 2008, S. 59])

Die Eigenschaften von Geodaten lassen sich in vier Kategorien einteilen (vgl. für das Folgende [Brinkhoff, 2008, S. 60ff]):

- *Geometrische Eigenschaften* beschreiben die Position und Ausdehnung von Geodaten im Raum. Man unterscheidet zwischen einem Vektor- und Rastermodell. Beim Vektormodell werden die Geodaten mit einem oder mehreren Vektoren im Raum beschrieben. Beim Rastermodell wird der Raum in quadratische Rasterzellen (Pixel) aufgeteilt und für jede Zelle Informationen gespeichert.
- *Topologische Eigenschaften* beschreiben die Beziehung unter den von Geodaten beschriebenen Strukturen, z. B. vereinfacht ausgedrückt „Punkt A liegt rechts von Punkt B“. Diese topologischen Eigenschaften sind unabhängig von ihren geometrischen Eigenschaften wie Form oder Ausdehnung. (vgl. [GISWiki; GITTA])
- *Thematische Eigenschaften* beschreiben semantische Attribute der Geodaten, z. B. Straßennamen oder Gebäudebezeichnungen.
- *Temporale Eigenschaften* bestimmen die zeitliche Gültigkeit für Geodaten. Mit diesen *spatio-temporalen Daten* lassen sich beispielsweise Hochwasserstände an einer Küsten- oder Flusslinie modellieren und im zeitlichen Verlauf darstellen.

Von besonderer Bedeutung für Geodatenbanksysteme sind die geometrischen Eigenschaften (Geometrien) der Geodaten. Die topologischen Eigenschaften werden häufig aus der Geometrie abgeleitet (vgl. [Brinkhoff, 2008, S. 63]) und die thematischen Eigenschaften lassen sich mit den bekannten Datentypen in Datenbanksystemen abspeichern.

Häufig wird das **Simple-Feature-Access-Modell** zum Speichern von Geodaten verwendet. Es wurde vom *Open Geospacial Consortium* (OGC) entworfen und in den ISO-Normen 19125-1 und 19125-2 standardisiert (vgl. [Open Geospatial Consortium, a], [Brinkhoff, 2008, S. 73]). Simple Features umfassen im Wesentlichen zweidimensionale

Punkte (*Points*), Linien (*Curves*), Flächen (*Surfaces*) und Geometriesammlungen (*Geometry Collections*) (vgl. [Obe u. Hsu, 2011, S. 38ff], [Open Geospatial Consortium, a, S. 14]). Abbildung 2.3 zeigt die Klassenhierarchie der vier Typen zusammen mit dem abstrakten Typ *Geometry* und weiteren Spezialisierungen.

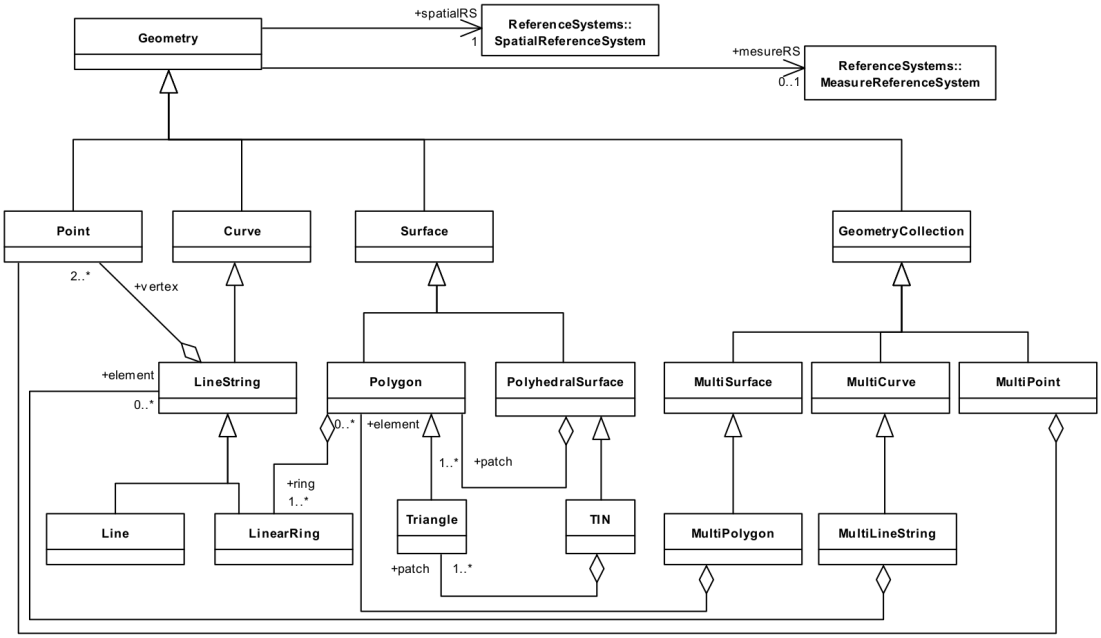

Abbildung 2.3: Simple Feature Access Geometrieklassen (aus [Open Geospatial Consortium, a, S. 14])

Ein **Punkt** des Simple-Feature-Access-Modells besitzt eine X- und eine Y-Koordinate, welche seine Position im zweidimensionalen Raum beschreiben. Eine **Linie** besteht aus einem oder mehreren Streckenzügen, die durch Streckenpunkte bestimmt werden. Die **Fläche** wird durch einen einfachen geschlossenen Streckenzug bestimmt. (vgl. [Brinkhoff, 2008, S. 74ff])

Zusätzlich zu den Koordinaten wird mit den Geometrien meist ein Verweis auf ein *räumliches Bezugssystem* (*spatical reference system*, SRS) gespeichert. Damit ist eine Interpretation der Koordinaten auf einem realen Datenraum möglich. Das SRS enthält ein Referenzmodell der Erde (z. B. ein Ellipsoid), welches der Gestalt der Erde möglichst ähnlich ist. Zur Positionsbestimmung wird zusätzlich noch ein Koordinatensystem benötigt, meist ein *geografisches Koordinatensystem*. Mit geografischen Koordinatensystemen werden die Koordinaten als geografische Länge und Breite (Längen- und Breitengrad) bezogen auf den Äquator und Nullmeridian des Ellipsoiden aufgefasst. Daten, die in dieser Form vorliegen, sind *unprojiziert*, da sie auf einen dreidimensionalen Körper abgebildet

werden. Digitale Karten liegen jedoch häufig im zweidimensionalen Raum vor, d. h. sie sind flach. Es ist also nötig, die Koordinaten mittels einer geeigneten *Projektion* auf eine zweidimensionale Fläche abzubilden. Dabei kommt es in jedem Fall zu Verzerrungen, die sich je nach Art der Projektion unterschiedlich auf die Winkel-, Linien- und Flächentreue der resultierenden Karte auswirken. (vgl. [Obe u. Hsu, 2011, S. 154, 158ff])

2.1.2.2 Geodaten in räumlichen Datenbanken

Das Simple-Feature-Access-Modell (vgl. 2.1.2.1) kennt zwei Repräsentationen für Simple-Feature-Geometrien (vgl. [Brinkhoff, 2008, S. 78f]):

- *Well-known Text (WKT)*: die Geometrie wird dabei als (Menschen-)lesbarer Text dargestellt
- *Well-known Binary (WKB)*: die Geometrie wird effizient binär kodiert

Ein Punkt könnte als WKT beispielsweise so kodiert sein: `POINT(10.5 -1.37)`

Werden die Geometrien in einem solchen bekannten Format gespeichert, kann die Geodatenbank dafür geometrische Funktionen bereitstellen, die der Anwender in SQL-Abfragen nutzen kann. Beispiele für solche Funktionen (vgl. [Obe u. Hsu, 2011, Kapitel 4 und 5]):

- `ST_Envelope()`: erzeugt das kleinste umgebende Rechteck für die Geometrie
- `ST_Distance()`: berechnet dem Abstand zwischen zwei Geometrien
- `ST_Buffer()`: erzeugt eine Geometrie, die einen Puffer um eine andere Geometrie darstellt
- `ST_Intersection()` bestimmt die Schnittmenge zwischen zwei Geometrien
- `ST_Union()`: vereinigt zwei Geometrien

Mit folgender SQL-Abfrage (für PostGIS) könnten beispielsweise alle Punkte der Geometrietabelle `points` abgefragt werden, die innerhalb eines 300-Einheiten-Umkreises um eine Koordinate liegen:

```
SELECT p.gid FROM points p
    WHERE ST_DWithin(
              ST_GeomFromText('POINT(-72.123 42.352)',1000),
              p.the_geom,
              300)
```

2.1.3 Datenbankmanagementsysteme

Die Software, welche eine Datenbank verwaltet, den Zugriff auf diese kontrolliert und mit den Anwendungsprogrammen interagiert, bezeichnet man als Datenbankmanagementsystem (DBMS). Datenbankmanagementsysteme werden seit den 70er Jahren des letzten Jahrhunderts von zahlreichen Herstellern entwickelt. Im Folgenden werden das freie DBMS PostgreSQL und das proprietäre Oracle Database betrachtet. Andere nennenswerte Datenbankmanagementsysteme sind Microsoft SQL Server und IBM DB2 sowie das bei Webanwendungen populäre (und ebenfalls freie) MySQL.

2.1.3.1 PostgreSQL und PostGIS

PostgreSQL ist ein freies Open Source Datenbankmanagementsystem, das seit einigen Jahren von der gleichnamigen Community-Initiative betreut wird (vgl. [PostgreSQL Global Development Group, a]). Es wird von ArcSDE als Datenbank-Back-End unterstützt und könnte eine Alternative zum proprietären Oracle Database sein.

Die PostgreSQL Entwicklungsgeschichte beginnt mit dem Datenbanksystem Ingres, das von 1977 bis 1985 an der University of California in Berkeley entwickelt wurde. In einem Entwicklungsprojekt in den Jahren 1986 bis 1994 an der Universität wurde der Ingres Quellcode um Objekt-relationale Fähigkeiten erweitert. Das Ergebnis wurde Postgres95 genannt. Diverse Firmen trugen ebenfalls zur Weiterentwicklung von Ingres/Postgres bei. Der Quellcode steht dabei unter der sehr liberalen PostgreSQL Lizenz (vergleichbar mit MIT/BSD-Lizenz, S. 12). Der Berkeley-Absolvent Jolly Chen pflegte Postgres95 auch nach seinem Verlassen der Universität weiter und dank der Nachfrage der Community nach einem freien Datenbanksystem, entstand im Jahr 1996 die PostgreSQL Global Development Group, eine Gemeinschaft, die seitdem durchgängig und intensiv die Pflege und Weiterentwicklung von PostgreSQL übernimmt. (vgl. [Momjian, 2001])

Heute ist PostgreSQL weltweit eine der bekanntesten und erfolgreichsten *Freie Software* Initiativen und erfreut sich großer Beliebtheit (vgl. [Coverty, S. 10]). Diverse Firmen bieten kommerziellen Support und Anpassungen für PostgreSQL an (vgl. [PostgreSQL Global Development Group, d]).

Logische Struktur von PostgreSQL Eine PostgreSQL-Instanz verwaltet immer einen sogenannten *Datenbank-Cluster*. Dieser Cluster ist eine Sammlung von Datenbanken, die ihrerseits jeweils eine Sammlung von Tabellen sind. Bei der Einrichtung des Datenbank-Clusters wird immer eine Datenbank `postgres` angelegt, die Verwaltungstabellen enthält. (vgl. [PostgreSQL Global Development Group, c])

PostGIS Mit PostGIS existiert eine Erweiterung für PostgreSQL, welche räumliche Datentypen (vgl. Abschnitt 2.1.2.1) und etwa 300 Datenbankfunktionen für die Arbeit mit diesen Datentypen zur Verfügung stellt. Die Datentypen und Funktionen entsprechen dabei dem SQL/MM-Standard (ISO 13249-3 Part 3: Spatial) und dem *Simple feature access* (ISO 19125).

2.1.3.2 Oracle Database mit Oracle Spatial

Oracle Database ist laut *Gartner 2011 Worldwide RDBMS Market Share Reports* im Bereich der proprietären kommerziellen relationalen Datenbankmanagementsysteme (RDBMS) führend (vgl. [Oracle, b]). Es soll daher - und weil es von einigen ArcGIS-Anwendern verwendet wird - in dieser Arbeit als Beispiel für ein proprietäres RDBMS dienen, das zusammen mit Esris ArcSDE als Back-End eingesetzt wird.

Die Ursprünge von Oracle reichen bis ins Jahr 1978 zurück, wo die erste Version des relationalen Datenbanksystems Oracle V.2 von der Firma Relational Software Inc. (RSI) veröffentlicht wurde. Später wurde RSI in Oracle umbenannt. Im Laufe der Zeit wurden dem relationalen Datenbanksystem objektrelationale und spatiale Erweiterungen hinzugefügt. Das aktuelle Release ist Oracle Database 11g. (vgl. [Brinkhoff, 2008, S. 33])

Oracle Spatial ist die räumliche Erweiterung für die Enterprise-Version von Oracle Database, mit der sich Raster- und Vektordaten in der Datenbank verwalten lassen.

Logische Struktur von Oracle Database Eine Instanz eines Oracle Datenbanksystems wird als *Datenbank* bezeichnet. Im Gegensatz zu PostgreSQL bezeichnet Datenbank hierbei eine Sammlung von *Tablespaces*. Ein Tablespace umfasst Tabellen und Views. Er entspricht also am ehesten einer PostgreSQL-Datenbank, wohingegen ein PostgreSQL-Datenbank-Cluster einer Oracle-*Datenbank* entspricht. (vgl. [Brinkhoff, 2008, S. 35])

Die Datenbankstrukturen von PostreSQL und Oracle Database sind also nicht identisch. Beim Namensschema wird dies später noch relevant (siehe S. 41). Ein Grund, warum Oracle Database bei der Wahl für ein ArcSDE-Back-End verwendet wird, lässt sich aus der Struktur des DBMS nicht ableiten.

2.2 Aufbau eines Geoinformationssystems

Ein Geoinformationssystem (GIS) ist ein spezialisiertes rechnergestütztes Informationssystem bestehend aus Hardware, Software und Daten, mit dem sich Problemstellungen mit Geobezug bearbeiten lassen. Mit dem GIS lassen sich räumliche Daten erfassen und aufzeichnen sowie mit einem geeigneten Datenmodell verwalten und organisieren. Die im GIS verwalteten Daten kann der Benutzer mit geeigneten geografischen, geometrischen oder statistischen Funktionen analysieren und verarbeiten. Die Daten selbst oder die Analyse- bzw. Verarbeitungsergebnisse können in alphanumerischer oder grafischer Form aufbereitet und dargestellt werden. (vgl. [Bill, 2010, S. 8f], [Brinkhoff, 2008, S. 2ff] und siehe auch Abbildung 2.4)

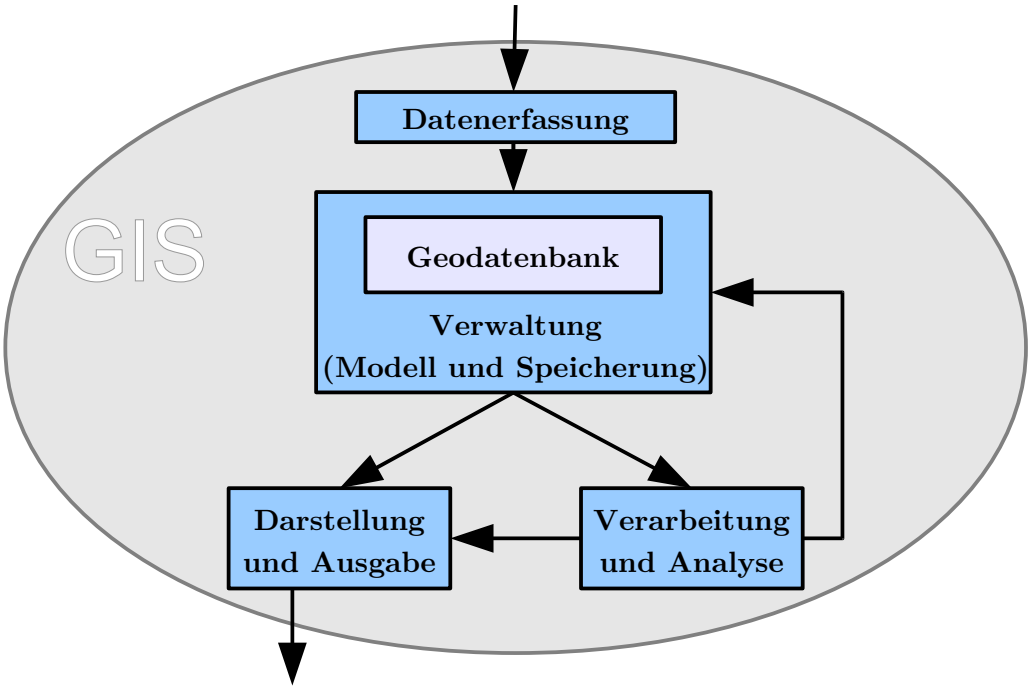

Abbildung 2.4: Schematischer Aufbau eines Geoinformationssystems (nach [Brinkhoff, 2008, Abb. 1.3])

Wie man in Abbildung 2.4 erkennt, ist die bereits in Abschnit 2.1.2 erläuterte **Geodatenbank** von zentraler Wichtigkeit für das GIS. Die für diese Arbeit relevante GIS-Software ist ArcGIS, die im nächsten Abschnitt vorgestellt wird.

2.3 Die ArcGIS-Produktfamilie

Die ArcGIS-Produktfamilie umfasst zahlreiche Komponenten und Erweiterungen. Aufteilen lassen sich die Programme zunächst in mehrere Kategorien (vgl. [Esri, f]):

- *ArcGIS for Desktop*:
  - ArcMap ist eine Desktop-Applikation für das Betrachten, Erstellen und Bearbeiten von digitalen Karten (vgl. Abbildung 2.6, S. 27). Sie enthält umfangreiche Werkzeuge für die Analyse, Verwaltung und Visualisierung von räumlichen Daten.
  - ArcCatalog ist eine Desktop-Applikation für das Verwalten von Geodatenquellen, insbesondere solche zu ArcSDE (vgl. Abbildung 2.7, S. 28).
  - ArcGlobe stellt einen digitalen Globus bereit, ähnlich zum bekannten Google Earth.
  - ArcScene ist eine Desktop-Applikation, mit der sich dreidimensionale Karten- und Geländedarstellungen erzeugen und bearbeiten lassen.
- *ArcGIS for Server*:
  - ArcGIS Server stellt Karten- und Featuredienste über ein Netzwerk bzw. das Internet bereit; die Dienste können dabei einen Großteil der umfangreichen ArcObjects-API verwenden.
  - Die ArcSDE-Middleware (SDE steht für *Spatial Data Engine*) ist eine räumliche Erweiterung für relationale Datenbanksysteme, die einheitlich aus allen ArcGIS-Produkten genutzt werden kann.
- *ArcGIS for Mobile* sind mobile Anwendungen (Apps) für Android, Windows Phone und iOS zum Anzeigen von digitalen Karten und zum Erfassen von Geodaten.
- *ArcGIS Online* ist ein Cloud-basierter Kollaborationsdienst zum Bearbeiten, Speichern und Verwalten von digitalen Karten im Web.

Die einzelnen Komponenten von ArcGIS sind untereinander verknüpft, Abbildung 2.5 (S. 26) macht einige dieser Verbindungen deutlich: die Datenbank-Middleware ArcSDE ist das zentrale Element der Infrastruktur. In ihrem Back-End arbeitet ein relationales Datenbanksystem, das die eigentlichen Geodaten speichert. Die ArcSDE-Instanz ist die wichtigste Datenquelle für den ArcGIS Server. Der Server stellt beispielsweise Webservices im Internet bereit. Die Desktop-Programme ArcMap und ArcCatalog verwenden ebenfalls die ArcSDE als Datenquelle. Darüber hinaus kann ArcMap ab der Version 10.0 PostGIS-Tabellen ohne Umweg über die ArcSDE direkt als sogenannte *QueryLayer* einbinden, allerdings nur unter Lesezugriff. ArcMap und ArcCatalog können darüber hinaus auch Datei-basierte Datenquellen wie z. B. Shapefiles verwenden.

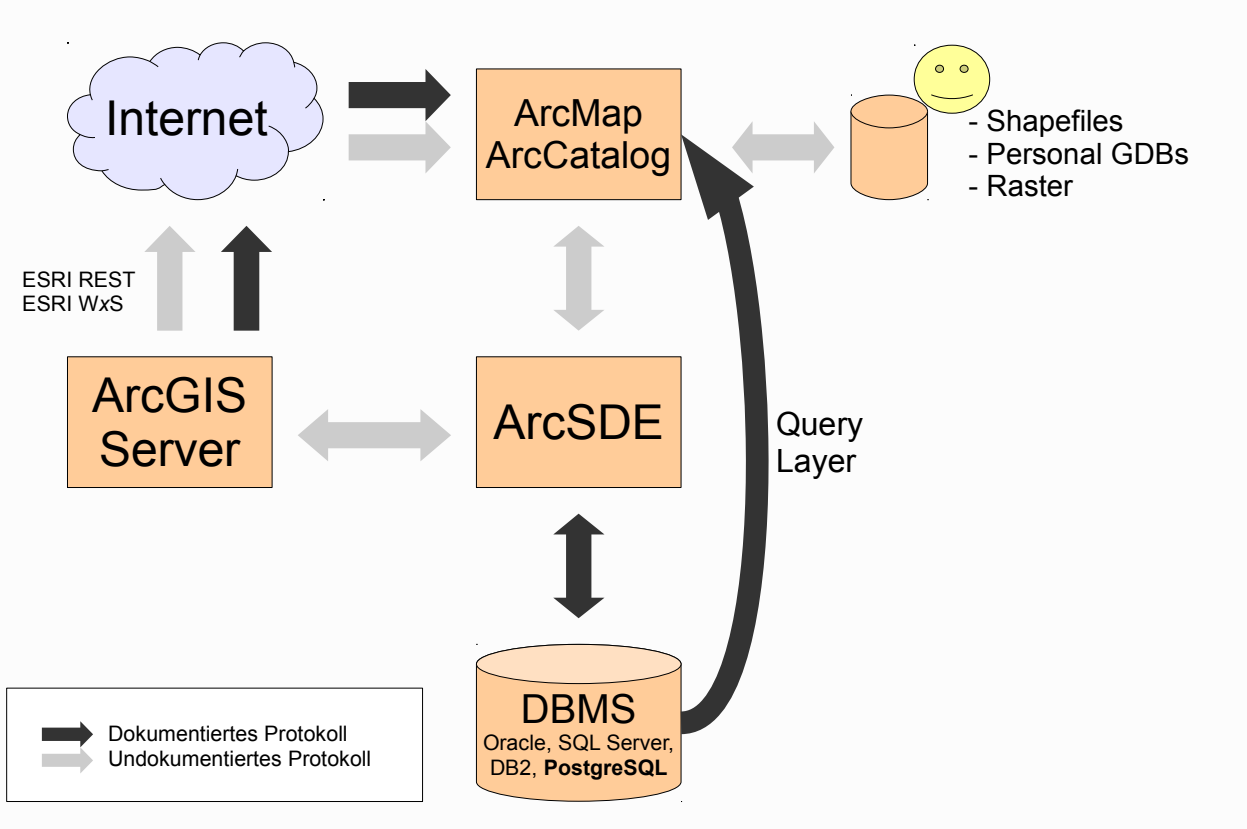

Abbildung 2.5: Komponenten von ArcGIS und ihre Verbindungen untereinander

Die Komponenten *ArcGIS for Desktop* (ArcMap und ArcCatalog) und *ArcGIS for Server* (ArcGIS Server und ArcSDE) werden im Folgenden genauer betrachtet.

2.3.1 ArcGIS auf dem Desktop

2.3.1.1 ArcMap

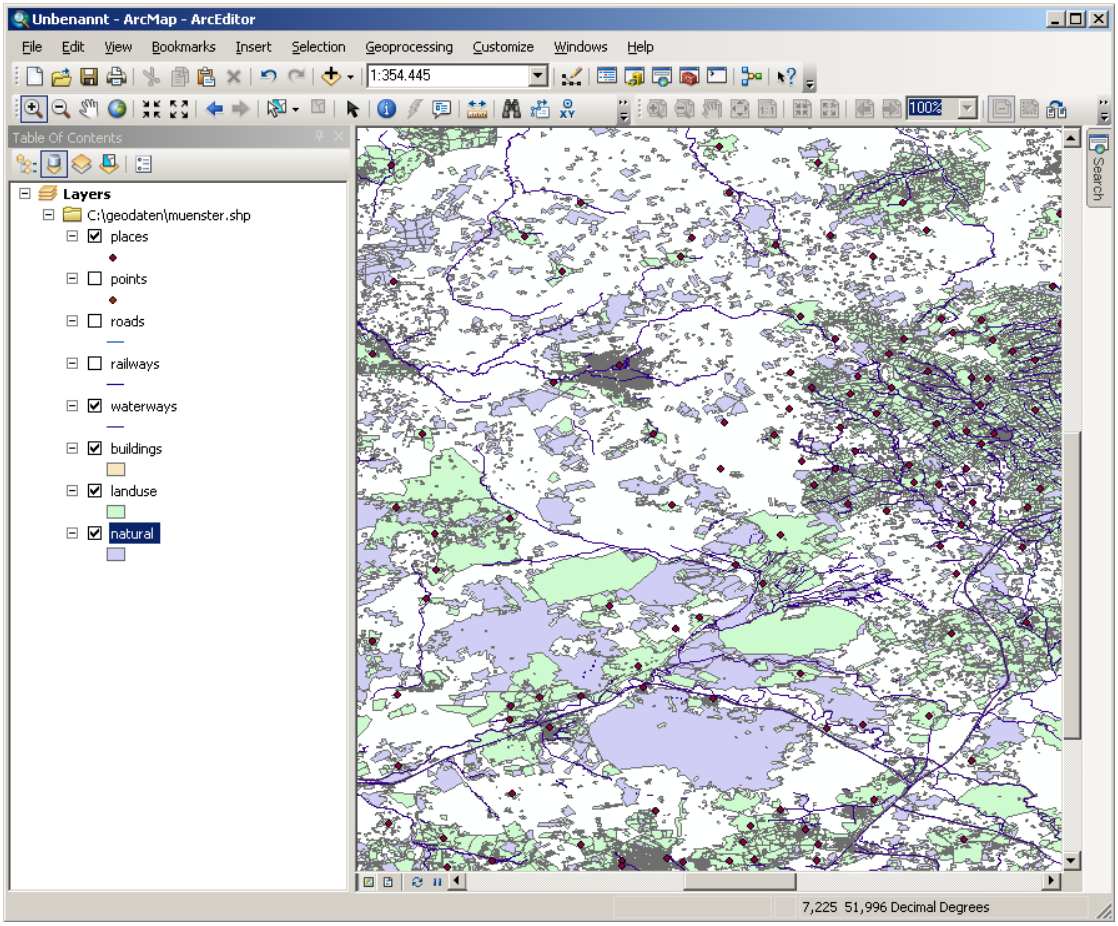

Abbildung 2.6: ArcMap Desktop GIS Programm

ArcMap (Abbildung 2.6) ist eine klassische GIS-Anwendung, die zum Bearbeiten und Erstellen von digitalen Karten verwendet wird. Die Kartenebenen (Layer) werden dabei auf der linken Seite der Benutzeroberfläche organisiert. Die Layer können beliebig übereinander gelegt und bei Bedarf ein- und ausgeblendet werden. Der große rechte Ausschnitt der Oberfläche zeigt die Karten oder geografischen Daten an. Der Ausschnitt kann mit der Maus oder Tastatur verschoben bzw. vergrößert und verkleinert werden. Über eine *Toolbox* (siehe Abbildung 3.2, S. 41) können zahlreiche Spezialwerkzeuge verwendet werden. ArcMap liefert eine Vielzahl solcher Werkzeuge mit, aber auch Erweiterun-

gen von Drittherstellern können an dieser Stelle den Funktionsumfang von ArcMap ergänzen. Am rechten Rand befindet sich seit ArcMap 10 ein Steuerelement, das die wichtigsten Funktionen von ArcCatalog bereitstellt.

2.3.1.2 ArcCatalog

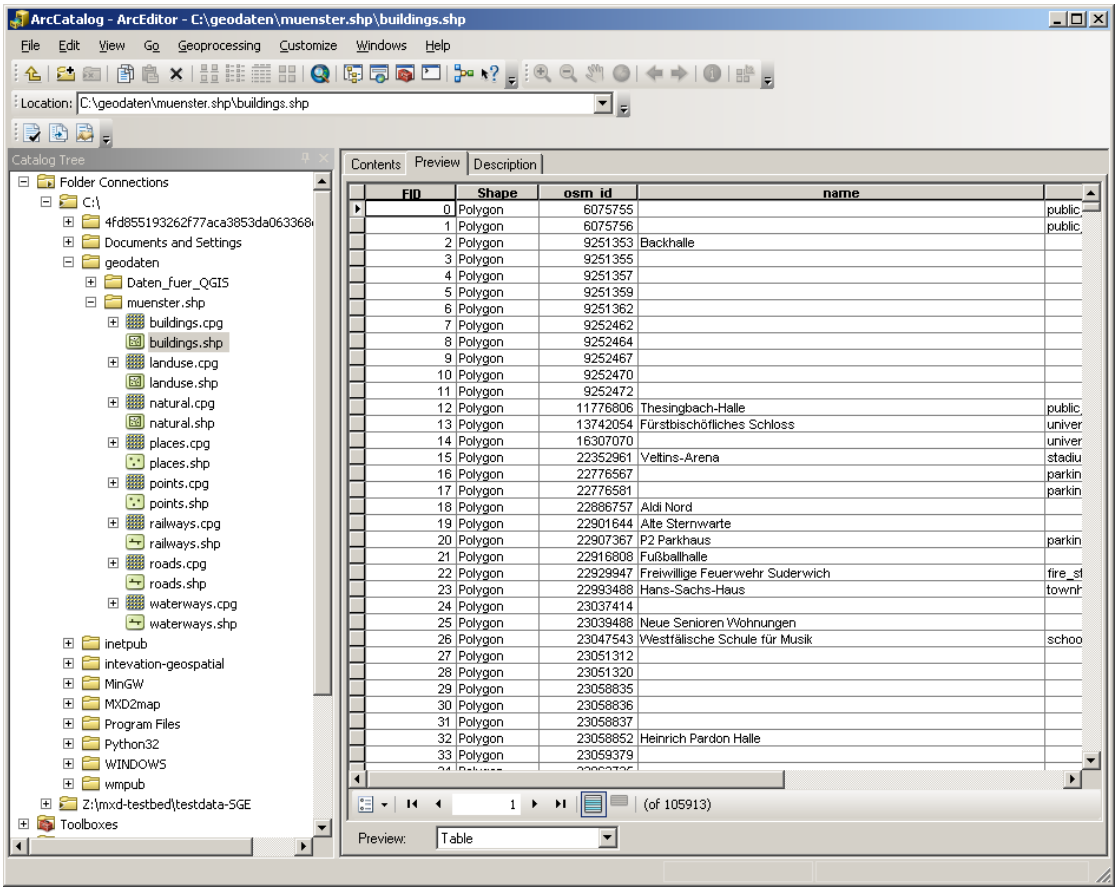

Abbildung 2.7: ArcCatalog: Verwaltungsprogramm für GIS-Datenquellen

ArcCatalog (Abbildung 2.7) ist ein Verwaltungsprogramm für Datenquellen mit Geodaten. Dazu gehören Datenbankverbindungen, Verbindungen zu ArcSDE-Instanzen und Anbindungen an *Personal Geodatabases* (lokale Datei-basierte Geodatenbanken) und Shapefiles auf dem Dateisystem. Auf der linken Seite werden alle bekannten bzw. eingerichteten Datenquellen in einer Baumansicht dargestellt. Markiert man eine Datenquelle, so wird auf der rechten Seite der Oberfläche eine Vorschau der Datenquelle

angezeigt. Die Vorschau ist wahlweise eine Tabellenansicht oder eine Ansicht der geometrischen Daten. Diese Ansicht entspricht dann der von ArcMap, es sind jedoch keine Bearbeitungsfunktionen vorgesehen.

2.3.2 ArcGIS auf dem Server

2.3.2.1 Webservices

Der ArcGIS Server kann u. a. Karten-Webdienste bereitstellen. Das *Open Geospatial Consortium* (OGC[1]) definiert mehrere Standards für Webservices, u. a.:

- *Web Mapping Service* (WMS, vgl. [Vatsavai u. a., 2006; Open Geospatial Consortium, b]): WMS ist eine Webservice-Schnittstelle zum Abrufen von digitalen Landkarten über ein Netzwerk wie das Internet. Der WMS visualisiert die angefragten Kartenausschnitte dabei als Raster- oder Vektorgrafik (vgl. [Open Geospatial Consortium, b, S. 14]).
- *Web Processing Service* (WPS, vgl. [Open Geospatial Consortium, e]): WPS ist eine Webservice-Schnittstelle, um eine geografische Analyse auf einem Geoserver anzustoßen und die Ergebnisse dieser Analyse bzw. Verarbeitung abzufragen.
- *Web Feature Service* (WFS, vgl. [Open Geospatial Consortium, d]): WFS ist eine Webservice-Schnittstelle zum Abrufen von geografischen Vektordaten (*Features*) aus einem GIS über ein Netzwerk wie das Internet.
- *Web Coverage Service* (WCS, vgl. [Vatsavai u. a., 2006; Open Geospatial Consortium, c]): WCS ist eine Webservice-Schnittstelle zum Abrufen von geografischen Rasterdaten aus einem GIS über ein Netzwerk wie das Internet. WCS ist das Gegenstück zum WFS, nur für Rasterdaten.

Der ArcGIS Server unterstützt WMS, WFS und WCS (vgl. [Esri, c]). Eine WPS-ähnliche Funktionalität kann über eine Web-Veröffentlichung einer Toolbox-Funktion realisiert werden. Der Zugriff darauf erfolgt über eine proprietäre HTTP-REST-Schnittstelle.

[1] Siehe `http://www.opengeospatial.org/`

2.3.2.2 ArcSDE mit PostGIS

Esri ArcSDE ist eine Datenbank-Middleware, die zwischen den Anwendungsprogrammen auf Clientseite und einem Datenbanksystem auf Serverseite sitzt (vgl. Abbildung 2.8, S. 30).

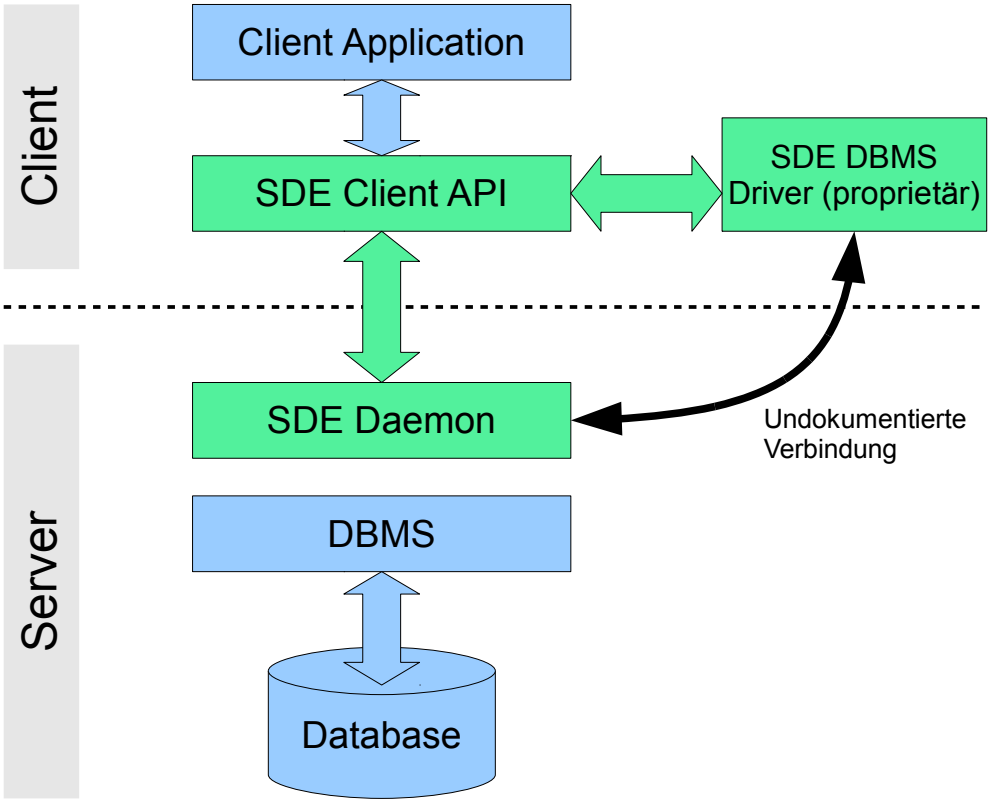

Abbildung 2.8: ArcSDE Struktur

Ein Clientprogramm, z. B. ArcMap oder ArcCatalog, verwendet eine dokumentierte Schnittstelle (SDE Client API), um mit der SDE-Clientbibliothek zu kommunizieren. Diese wird von Esri in Form einer DLL bereitgestellt (`sde.dll`). Die Bibliothek agiert dabei nicht völlig unabhängig vom Datenbankmanagementsystem, das auf dem Server verwendet wird, sondern benötigt vermutlich einen Datenbanktreiber auf Clientseite. Die DLLs (`sdeora10gsrvr100.dll`, `sdeora11gsrvr100.dll`, `sdepgsrvr100.dll`, `sdesqlsrvr100.dll`) im ArcGIS Installationsverzeichnis des Clienten deuten auf diesen Umstand hin. Auch in der Kommunikation zwischen `sde.dll` und dem SDE Daemon

auf dem Server werden Informationen zum verwendeten DBMS ausgetauscht.

Auf Serverseite lauscht der *SDE Daemon* auf eingehende TCP-Verbindungen. Er kommuniziert mit einem darunterliegenden Datenbankmanagementsystem wie z. B. Oracle Database oder PostgreSQL. ArcSDE benötigt dabei nicht zwangsläufig ein DBMS mit einer nativen Unterstützung für geometrische Datentypen, da bei der Installation die Esri-Implementation des `ST_Geometry`-Datentyps im DBMS nachinstalliert wird, sofern dieses die Installation von benutzerdefinierten Datentypen erlaubt.

ArcSDE ist nicht nur in der Lage PostgreSQL als Daten-Back-End zu verwenden, sondern kann auch die räumliche Erweiterung PostGIS direkt verwenden (vgl. [Esri, b]). Standardmäßig verwendet ArcSDE den `ST_Geometry`-Typ, bei SDE-Installation mit PostgreSQL im Backend lässt sich der verwendete Geometrietyp mit dem `sdedbtune`-Kommandozeilentool so konfigurieren, dass der PostGIS-Geometrytyp verwendet werden kann (hier `PG_GEOMETRY` genannt):

```
$ sdedbtune -o alter -k DEFAULTS -P GEOMETRY_STORAGE -v PG_GEOMETRY -u sde
```

In einem Test mit ArcSDE 10 sowie PostgreSQL 8.3.8 und PostGIS 1.4, gelang es jedoch nicht FeatureClasses zu erstellen bzw. zu verwenden, die eine PostGIS-Geometrie verwendet, da es nicht möglich war sowohl die SDE als auch PostGIS in einer PostgreSQL-Datenbank zu installieren. Das SDE-Installationsprogramm möchte gleichnamige Funktionen installieren, die schon mit PostGIS installiert wurden. Da die Funktionen aber mit unterschiedlichen Geometrietypen arbeiten, sind sie nicht kompatibel.

3 Ansätze für die Migration zu freier Software

In diesem Kapitel wird genauer untersucht, wie Geodatenbanken, die mit ArcGIS verwaltet werden, auf Lösungen mit freier Software migriert werden können. Zunächst wird anhand von Gesprächen mit ArcGIS-Anwendern bestimmt, welche Anforderungen eine GIS-Infrastruktur haben muss, um die Aufgaben der Kunden adäquat zu erfüllen. Auch auf Kritikpunkte an freier Software wird eingegangen. Als Gesprächspartner standen Mitarbeiter des Bundesamtes für Seeschifffahrt und Hydrographie (BSH), des Niedersächsischen Ministeriums für Umwelt, Energie und Klimaschutz sowie des Fachdienstes Geodaten der Stadt Osnabrück. Die von diesen Anwendern verwendeten Esri ArcGIS Installationen sind sowohl sehr groß (BSH) als auch verhältnismäßig überschaubar (Stadt Osnabrück).

Nach der Analyse der Migrationsmöglichkeiten werden Ersatzszenarien skizziert und eine technische Lösung des ArcSDE-Ersatzes wird angestrebt. Die technische Realisierung der Prototypen wird ebenfalls dargestellt.

3.1 Bedarf der Kunden und Anwender

Der Bedarf des Kunden bzw. der Anwender muss bei der Erstellung einer IT-Lösung immer im Vordergrund stehen. Bevor also eine Migration weg von der ArcSDE geplant wird, muss zunächst geklärt werden, wie sie beim Anwender eingesetzt wird und welche Probleme sie dort löst.

Auf die Frage hin, welche geografischen Funktionen der ArcSDE verwendet werden, antworten alle befragten Anwender, dass geografische Funktionen (z. B. Verschneidungen zwischen Polygonen) ausschließlich in den Clientprogrammen durchgeführt werden. Die

entsprechende Funktionalität in ArcSDE wird bei den befragten Anwendern also nicht verwendet.

ArcSDE wird konkret bei den befragten Anwendern nur zum Verwalten der Geodaten verwendet, d. h.. Laden und Speichern sowie Importieren und Löschen von Raster- und Vektordaten. Ein Kunde nannte den Vorteil, dass die ArcSDE eine einheitliche Schnittstelle für den Zugriff auf die Geodaten bereitstellt (vgl. Anhang D.1).

Historisch gesehen ist der Einsatz von ArcSDE nachvollziehbar. Anfang der 90er Jahre hatten die räumlichen Datenbanken noch nicht den heutigen Reifegrad erreicht bzw. waren überhaupt nicht verfügbar. Für Esri war es einfacher, einen Aufsatz (d. h. Middleware) für bestehende relationale Datenbanksysteme zu schreiben, als eine räumliche Datenbank von Grund auf neu zu entwickeln. Heute sind die räumlichen Datenbanksysteme jedoch ausgereift und in einigen Varianten verfügbar. Es stellt sich also die Frage, ob das Konzept der Middleware ArcSDE für die heutigen Anforderungen noch geeignet ist.

Ein Kunde nannte das System veraltet und nicht mehr zeitgemäß. Insbesondere die fehlende Unterstützung für Multiprozessorsysteme stelle heute einen erheblichen Flaschenhals dar, da ArcSDE (zumindest in der Version 9.3.1) mittlerweile nicht mit der steigenden Anzahl der Prozessorkerne skaliere (vgl. Anhang D.1). Da gängige DBMS wie Oracle oder PostgreSQL Erweiterungen für räumliche Daten bereitstellen, sind die räumlichen Funktionen von ArcSDE kein zwingender Einsatzgrund mehr. Die ArcGIS Desktop-Programme könnten technisch problemlos auf räumliche Datenbanken zugreifen, sofern Esri dies implementieren würde. Der homogene Datenzugriff kann in der Theorie zunächst als Vorteil gewertet werden. Es stellt sich aber die Frage, ob der Datenzugriff auch über eine Migration des zugrunde liegenden DBMS hinaus transparent bleibt. Sollten bei einer Migration des Datenbank-Back-Ends Änderungen am Datenmodell oder Clientprogramm notwendig sein, ist dieser Vorteil hinfällig. In Abschnitt 3.4 wird diese Frage behandelt.

Nach Auswertung der Interviews mit den ArcGIS-Anwendern, kann man festhalten, dass die Geo-Middleware ArcSDE im Wesentlichen als eine einheitliche Schnittstelle der ArcGIS Desktop-Anwendungen auf die Geodaten verwendet wird. Geofunktionen der SDE werden nicht eingesetzt. Damit beschränkt sich die Funktion von ArcSDE auf das Entgegennehmen und Weiterleiten von Datenbankanfragen, sowie das Nachbilden einer Geodatenbank auf der Basis eines relationalen Datenbanksystems. Diese Erkenntnis

ist entscheidend für die Migrationsüberlegungen, die wegen dieses Umstandes möglicherweise vereinfacht werden.

3.2 Kritikpunkte an freier Software

Im Bezug auf *Freie Software* existieren einige Aussagen nicht-technischer Natur, die aus Sicht der befragten Esri-Anwender gegen einen Einsatz von freier Software sprechen. Im Folgenden werden die geäußerten Kritikpunkte analysiert und - sofern möglich - widerlegt. Insbesondere die mangelnde Erfahrung wurde als Haupthinderungsgrund für den Einsatz von freier Software genannt (siehe Anhang D.2).

3.2.1 Mangelnde Erfahrung

Viele IT-Dienstleister sind stark in einer proprietären Produktwelt eingebunden und haben möglicherweise (finanzielle oder vertragliche) Abhängigkeiten an einen großen Softwarehersteller. Unternehmen können beispielsweise Teilnehmer am Esri Partner Network sein oder sind zertifizierte Microsoft/Oracle/etc. Reseller. Auch wenn solche Partnerschaften nicht verhindern, dass der Dienstleister Lösungen auf Basis von Konkurrenzprodukten anbieten, so hängt z. B. die Zulassung für eine Gold oder Platinum Partnerschaft im Esri Partner Network auch davon ab, wie viele Lösungen auf Basis von Esri-Produkten angeboten werden (vgl. [Esri, e, S. 10]). Beim IT-Dienstleister liegt somit ein Wissens- und Erfahrungsschatz, der sich auf spezifische proprietäre Produkte konzentriert. Ein solcher Dienstleister ist vielfach nicht bereit - oder nicht in der Lage - seine Kunden beim Einsatz von freier Software zu beraten, rät von deren Einsatz ab oder empfiehlt direkt ein proprietäres Konkurrenzprodukt aus seinem Portfolio ohne (freie) Alternativen in Erwägung zu ziehen. In einem Gespräch wurde deutlich, dass der IT-Dienstleister des Kunden und auch der Kunde selbst keinerlei Erfahrung mit dem Einsatz von freier Software hat und entsprechende freie Softwareprodukte nicht für den Einsatz in Erwägung gezogen wurden (vgl. Anhang D.2). Wenn Kunden - wie z. B. das BSH - bereits (gute) Erfahrungen mit freier Software gemacht haben oder sich aus eigenem Antrieb für freie Alternativlösungen interessieren, stehen sie dem Einsatz von freier Software aufgeschlossen gegenüber. Für IT-Dienstleister von freier Software ist es daher ausgesprochen wichtig, lauffähige Lösungen für echte Kundenprobleme zu präsentieren (Referenzkunden). Kunden, die sich für Lösungen auf Basis von freier Software interessieren, sollten daher nicht ihren aktuellen Dienstleister mit einer Untersuchung

der Einsatzmöglichkeiten von freier Software beauftragen, sondern einen unabhängigen (und möglicherweise auf *Freie Software* spezialisierteren) Dienstleister.

3.2.2 Abhängigkeit an den Dienstleister

Proprietäre Software hat in vielen Bereichen einen höheren Marktanteil als *Freie Software*. Naturgemäß gibt es mehr Dienstleister, die ihre Leistungen auf Basis von proprietärer Software aufbauen. Wer daher als Kunde auf *Freie Software* setzt, könnte auf den einen Dienstleister mit Kompetenz in Sachen *Freie Software* angewiesen sein und sich bei Beauftragung in eine Abhängigkeit begeben. Die Abhängigkeit ergibt sich jedoch nicht aus der Art und Weise der Software, sondern ist nur durch die Kompetenz des Dienstleisters gegeben. Der Kunde *könnte* die *Freie Software* von einem kompetenteren oder gleichwertigen Dienstleister weiterentwickeln lassen, wenn er mit dem bisherigen unzufrieden ist, denn dieses Recht garantiert die Lizenz der freien Software (siehe S. 12). Die Möglichkeit den Dienstleister zu wechseln besteht bei proprietärer Software nicht zwangsläufig, da die Rechte der Software nicht immer (je nach Vertrag) beim Kunden liegen. Gerade wenn ein Dienstleister die Software nicht von Grund auf für den Kunden entwickelt, sondern nur Anpassungen vornimmt, werden die Rechte in aller Regel beim Dienstleister verbleiben. In einem solchen Fall ist der Kunde tatsächlich durch die Urheberrechte an den einen Dienstleister gebunden, wenn er die Software weiterentwickelt haben möchte. Der Eindruck, dass man sich bei der Entscheidung für *Freie Software* in Abhängigkeit eines Dienstleisters begibt, stimmt also nur bedingt. Das eigentliche Problem könnte die geringe Dichte von Dienstleistern sein, die auf *Freie Software* spezialisiert sind, d. h. die Dienstleister sind schwer zu finden oder räumlich weit entfernt. Dank moderner Kommunikationsmittel und dem stetig steigenden Marktanteil freier Softwarelösungen (zu sehen bspw. an den Webserver-Statistiken, vgl. [Netcraft]) sollte dies aber nicht immer ein Hindernisgrund für den Einsatz von *Freier Software*.

3.2.3 Mangelnde Kapazitäten beim Dienstleister

Ein Kunde nannte als Hindernisgrund für den Einsatz von freier Software die mangelnden Kapazitäten bei den Dienstleistern für *Freie Software*. Dabei ging der Kunde offenbar davon aus, dass eine Fachanwendung, die auf freier Software basiert, von Grund auf neu entwickelt werden muss und daher im Vergleich zu Standardsoftware mehr Aufwand betrieben werden muss, um diese fertig zu stellen. In der Praxis wird aber selbst

eine maßgeschneiderte Anwendungssoftware für Kunden nicht von Grund auf neu entwickelt, sondern häufig auf Basis von Prototypen oder vorhandenen Komponenten erstellt. Soll die Kundenlösung auf Basis von freier Software entstehen, so kann der Dienstleister aus einem enorm großen Fundus an Software(fragmenten) schöpfen, denn sämtliche *Freie Software*, die jemals geschrieben wurde, steht ihm (theoretisch) zur Verfügung. Ein Dienstleister, der proprietäre Software erstellt, kann eingeschränkt auch Komponenten aus der freien Softwarewelt verwenden, sofern diese unter einer liberalen Lizenz (z. B. BSD oder MIT, siehe S. 12) stehen. GPL-lizensierte Komponenten bleiben ihm verwehrt. Er kann aus seinem eigenen Fundus schöpfen oder gegen Lizenzgebühren Komponenten von Drittanbietern verwenden. Dies treibt dann aber die Kosten für den Kunden in die Höhe.

Da Entwickler von freier Software aus einem großen Katalog bereits erstellter Software auswählen können, muss nicht so viel Software für den Kunden komplett neu geschrieben werden. Der Dienstleister benötigt daher u. U. gar nicht die Personalkapazitäten, die er mit proprietärer Software benötigen würde. Der geringere Personalbedarf spart Geld (und oft auch Zeit), was dem Kunden zu Gute kommt. Darüber hinaus sind viele freie Softwarelösungen ausgereift und haben sich im Praxiseinsatz bewährt. Eine nicht angepasste Lösung kann dem Kunden eventuell schon recht bald zur Evaluierung oder als Übergangslösung bereitgestellt werden, während seine speziellen Anpassungswünsche in Arbeit sind.

3.3 Skizzierung von Migrationsszenarien

Das ArcGIS-Ökosystem besteht vollständig aus unfreier Software. Ein kompletter Austausch kann daher nötig sein, z. B. wenn wenn eine lizenzkostenfreie, flexible oder leichtgewichtigere Alternative zu ArcGIS gefordert wird (vgl. Kapitel 1 und Abschnitt 3.2). Um die Umstellung für den Endanwender möglichst reibungslos zu gestalten, sollten zunächst nur die Komponenten im Back-End durch freie Software ersetzt werden. Im Idealfall geschieht dies transparent, d .h der Anwender bemerkt keine (negativen) Veränderungen. Das Back-End besteht - wie schon beschrieben - aus dem ArcGIS Server, der ArcSDE und einem darunterliegenden DBMS.

Die vom ArcGIS Server bereitgestellten Karten- und Webdienste (siehe S. 29) lassen sich zu einem großen Teil bereits durch *Freie Software* wie MapServer[1], GeoServer[2] oder mxd2map[3] ersetzen. Die folgende Betrachtung beschränkt sich daher auf den Bereich des **Datenbank**-Back-Ends. Das bedeutet, dass sowohl die ArcSDE-Instanz als auch das darunterliegende DBMS ersetzt werden sollen (siehe Abbildung 3.1, S. 38).

Es soll zunächst versucht werden, dass Datenbankmanagementsystem unterhalb der SDE durch eine freie Variante zu ersetzen (siehe Kapitel 3.4). Die ArcSDE-Software bleibt bei dieser Variante zunächst erhalten. Im darauf folgenden Schritt wird die Möglichkeit untersucht auf unterschiedliche Art und Weise ArcSDE durch eine freie Lösung zu ersetzen oder komplett aus dem Ökosystem zu entfernen (siehe Kapitel 3.5 ab S. 43).

3.4 Austausch des Datenbank-Back-Ends

Bisher wurde festgestellt, dass viele Anwender ein proprietäres Datenbankmanagementsystem im Zusammenspiel mit ArcSDE verwenden. Solche DBMSs können hohe Anschaffungs- bzw. Lizenzkosten verursachen. ArcSDE unterstützt auch das freie DBMS PostgreSQL, daher kann ein erster Schritt in Richtung Verwendung von freier Software sein, PostgreSQL anstelle eines proprietären Datenbanksystems zusammen mit ArcSDE zu verwenden. Man sollte davon ausgehen können, dass die Installation und Konfiguration einer ArcSDE-Infrastruktur mit PostgreSQL problemlos verläuft, da dies ein vom Hersteller Esri offiziell unterstützes Verfahren ist. In der Praxis können

[1] Siehe http://www.mapserver.org/
[2] Siehe http://geoserver.org/display/GEOS/Welcome
[3] Siehe http://www.mxd2map.org/

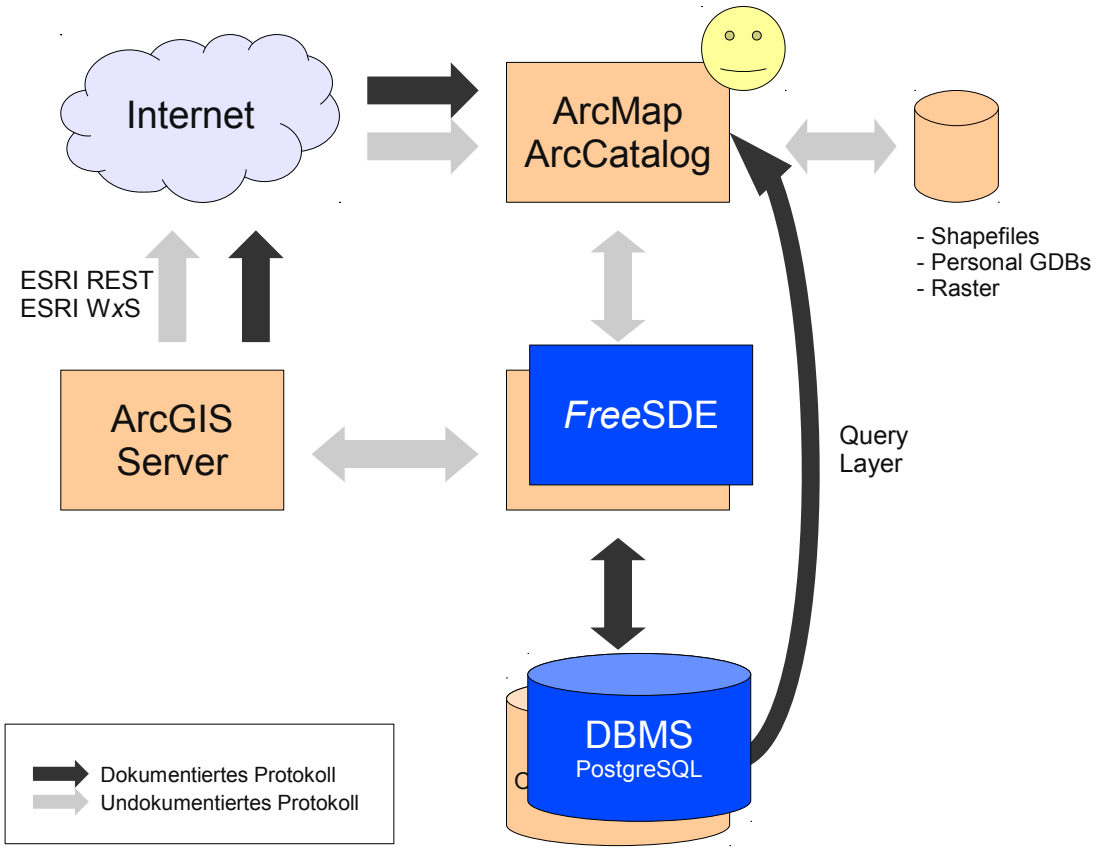

Abbildung 3.1: Das ArcGIS-Ökosystem mit freien Softwarekomponenten

dennoch Probleme auftreten. Im Folgenden wird daher die Migration einer bestehenden ArcSDE-Installation mit dem proprietären Datenbanksystem Oracle auf eine ArcSDE-Installation mit PostgreSQL untersucht und auf ihre Praxistauglichkeit geprüft.

3.4.1 Vergleich Oracle und PostgreSQL

Ein Performanzvergleich von Oracle und PostgreSQL ist nur schwer möglich, da Oracle in den Lizenzbedingungen einiger Versionen ausdrücklich den Vergleich mit Konkurrenzprodukten verbietet (vgl. [Oracle, c]). Dieser Passus in den Lizenzbedingungen hindert einige Entwickler nicht daran, trotzdem einen Performancevergleich durchzuführen (vgl. [Rogers]). Im Rahmen dieser Arbeit wird die kostenlose Oracle Database 11g Express Edition für den Vergleich herangezogen. Sie hat einige Ein-

schränkungen verglichen mit den teuereren Enterprise Versionen der Datenbank. So wird maximal ein Prozessorkern und maximal ein Gigabyte Arbeitsspeicher verwendet. Als *Freie Software* unterliegt PostgreSQL keinen solchen Einschränkungen. In Tabelle 3.1 werden einige theoretische Eigenschaften der beiden Datenbankmanagementsysteme gegenübergestellt. Die Eigenschaften beziehen sich auf die aktuelle PostgreSQL-Version und die Enterprise-Edition von Oracle Database.

Eigenschaft	**PostgreSQL 9** [PostgreSQL Global Development Group, a]	**Oracle 11g Database** [Oracle, a]
Max. Datenbankgröße	unbegrenzt	unbegrenzt
Max. Tabellengröße	32 TB	unbegrenzt?
Max. Zeilengröße	1,6 TB	unbegrenzt?
Max. Feldgröße	1 TB	8 TB bis 128 TB
Max. Zeilen pro Tabelle	unbegrenzt	unbegrenzt
Max. Spalten pro Tabelle	260 bis 1600 (je nach Spaltentyp)	1000
Max. Indizes pro Tabelle	unbegrenzt	unbegrenzt
Multi-Version Concurrency Control (MVCC)	ja	ja
Asynchrone Replikation	ja	ja

Tabelle 3.1: Tabellarischer Vergleich der Eigenschaften von PostgreSQL und Oracle Database

3.4.2 Testmigration nach PostgreSQL

Zunächst wurde eine Testinstanz von Oracle Database 11g Express Edition und ArcSDE 10.0 auf einem Windows Server 2003 System aufgesetzt (Installationsanleitung siehe Anhang B). Um den Realismus des Tests zu erhöhen, wurde die ArcSDE-Instanz mit möglichst realistischen Daten(mengen) gefüllt. Als Grundlage dienten dabei die freien Daten des OpenStreetMap-Projekts[4] vom Freistaat Bayern[5]. Die Shapefiles sind insgesamt knapp 800 MiB groß, die Datenbankdatei ist nach dem Import etwa 1500 MiB groß.

Im Rahmen dieser Arbeit wurde das Python-Skript `dumpsde.py` erstellt, das die SDE-Kommandozeilen-Programme `sdeexport` und `sdeimport` verwendet, um alle Feature-

[4] Siehe `http://www.openstreetmap.org/`

[5] Die Firma Geofabrik stellt tagesaktuelle Dumps der OpenStreetMap-Datenbank als Shapefiles zur Verfügung, siehe `http://download.geofabrik.de/osm/europe/germany/`

Classes einer SDE-Instanz zu exportieren und in eine andere SDE-Instanz zu importieren. Als Austauschformat verwenden `sdeexport` und `sdeimport` das SDEX-Format von Esri. Für diese Kommandozeilen-Programme ist eine ArcGIS Server Enterprise Lizenz nötig (vgl. [Esri, d]), es steht also nicht bei jedem Kunden zur Verfügung.

Mit folgendem Aufruf lassen sich die angegebenen FeatureClasses aus der ArcSDE-Instanz exportieren:

```
dumpsde.py out bayern_buildings,bayern_points,bayern_roads -u sde -p sde
```

Das Skript generiert entsprechende Aufrufe für `sdeexport` von Esri, das für jede FeatureClass eine entsprechende SDEX-Datei erstellt. Diese SDEX-Dateien müssen nun auf einem beliebigen Transportweg auf den Rechner kopiert werden, auf dem die zweite SDE-Instanz läuft. Dies kann z. B. mit SCP (SSH Secure Copy) oder FTP erfolgen. Auf dem zweiten Rechner lässt sich der Daten*dump* dann mit folgendem Skriptaufruf importieren:

```
dumpsde.py in ./ -u sde -p sde
```

Dabei werden alle im aktuellen Verzeichnis liegende SDEX-Dateien in die ArcSDE-Instanz importiert.

Die Geodaten dagegen erfahren durch die Migration keine Veränderung. Dies wurde mit dem Esri-eigenen Werkzeug *Feature Compare* (siehe Abbildung 3.2, S. 41) aus der ArcMap-Toolbox überprüft.

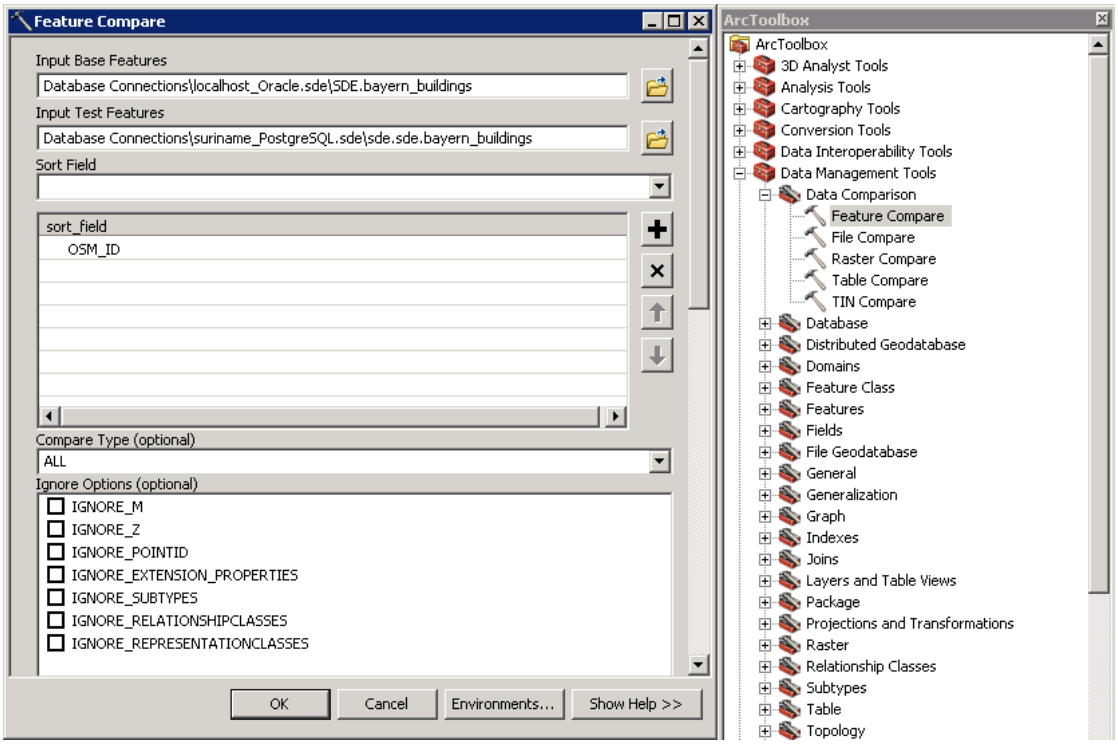

Abbildung 3.2: *Feature Compare* Werkzeug aus der *Toolbox*

Als Benutzername, Schema und Datenbank wurde stets `sde` verwendet, dennoch unterschieden sich die angezeigten Namen zwischen Oracle-SDE und PostgreSQL-SDE. Auf der Oracle-SDE trugen die FeatureClasses einen Namen nach dem Schema `SDE.foo`, nach dem Reimport auf der PostgreSQL-SDE trugen sie einen Namen nach dem Schema `sde.sde.foo`. Man könnte vermuten, dass wegen dieser Namensänderung existierende Verweise in ArcCatalog-Verbindungen oder in ArcMap-Dokumenten (MXD) nach einer Migration des ArcSDE-Backends ins Leere laufen. Dies war jedoch nicht der Fall, wie ein Test zeigte: auf einem Windows-Rechner wurde über einen Eintrag in der Host-Datei (`C:/WINDOWS/system32/drivers/etc/hosts`) ein Verweis auf die Oracle-Instanz erstellt, eine Verbindung über diesen virtuellen Rechner in ArcCatalog eingerichtet und ein ArcMap-Dokument erstellt, das diese Verbindung als Datenquelle verwendet. In einem zweiten Schritt wurde der Eintrag in der Host-Datei dann auf die migrierte Instanz auf PostgreSQL umgelenkt:

```
arctest         127.0.0.1           # IP des Oracle-SDE-Hosts
...
arctest         192.168.10.24       # IP des PostgreSQL-SDE-Hosts
```

In ArcCatalog tauchten die FeatureClasses dann mit den von PostgreSQL bekannten Namensschema auf. Im ArcMap-Dokument blieben die Namen der Layer erhalten (d. h. Oracle-SDE-Namensschema), der Zugriff auf die Geodaten funktionierte nach wie vor transparent. Eine Änderung an bestehenden Dokumenten war somit nicht notwendig.

Gezeigt wurde also, dass der einfache Datenzugriff transparent mit den unterschiedlichen Datenbank-Back-Ends funktioniert. Zu prüfen war noch, ob in der Oracle-Instanz erstellte Indizes ebenfalls migriert werden bzw. neu erstellt werden. Dafür wurde eine neue Polygon-FeatureClass erstellt und mit wenigen Testpolygonen gefüllt. Auffällig war, dass ArcSDE beim Anlegen der neuen FeatureClass zumindest auf der Oracle Datenbank keinen räumlichen Index (siehe Abschnitt 2.1.1, S. 17) anlegte. Dieser musste manuell über das Eigenschaftsmenü erzeugt werden. Nach dem Reimport der Daten auf der PostgreSQL-basierten ArcSDE-Instanz war der angelegte Index augenscheinlich verschwunden. Er wurde allerdings von einem GiST-Index ersetzt, der in der PostgreSQL-basierten ArcSDE-Instanz automatisch bei neuen FeatureClasses angelegt wird.

Auch die Versionierung von Daten sollte beachtet werden, sodass nicht immer nur die neueste Version der Daten migriert wird. Mit Versionierung können Veränderungen von Daten in einer SDE-Instanz verfolgt werden und z. B. mehrere Versionen von FeatureClasses gepflegt werden. Programmierern ist dieses Konzept von Sourcecode-Versionsverwaltungssystemen wie Subversion/SVN oder Mercurial bekannt. Das Esri-eigene Werkzeug `sdeexport` berücksichtigt mit dem Parameter `-V` die Versionierung. Mit dem Pendant `sdeimport` kann man zwar auch in eine bestimmte Version importieren, diese jedoch nicht anlegen. Das bedeutet, dass das Anlegen einer Version von einem Benutzer manuell mit ArcCatalog durchgeführt werden muss, was einer möglichst automatisierten Migration entgegen steht.

Dieser einfache Praxistest zeigt, dass bei gängigen Datenmengen und einem einfachen Einsatzszenario nichts gegen einen Ersatz von Oracle Database durch PostgreSQL spricht. Mit gängigen Datenmengen sind solche gemeint, die sich bequem innerhalb der theoretischen Limits von PostgreSQL bewegen. Inwiefern PostgreSQL mit extrem großen Datenmengen (größer einem einstelligen Terabyte-Bereich, vgl. Anhang D.1) zurecht kommt, muss in Performance- und Praxistests geprüft werden.

Vor dem Einsatz von PostgreSQL ist zu prüfen, ob die Versionierung von ArcSDE bereits verwendet wird. Einer Versionierung von Geodaten mit PostgreSQL steht nichts im Wege, sofern sie mit PostgreSQL begonnen wird. Vorhandene Versionen ließen

sich mit den vorhandenen Kommandozeilenwerkzeugen nur einzeln exportieren und als separate FeatureClasses weiter verwenden.

Nutzer, die mit ihrer ArcSDE-Instanz riesige Datenmengen verwalten und daher speziell auf die Daten und die Anwendung zugeschnittene Indizes verwenden, müssten nach einer Migration auf PostgreSQL manuell Indizes anlegen, sofern die automatischen GiST-Indizes nicht ausreichen. Zu diesem Zweck könnte eine erweiterte Version des `sdedump`-Tools eine Liste der bestehenden Indizes ausgeben. Dies gelingt beispielsweise mit dem `sdelayer`-Werkzeug:

```
sdelayer -o describe_long -l bayern_buildings,Shape -D sde -u sde -p sde
```

Die Ausgabe enthält dann einen Eintrag zum verwendeten Index, in diesem Beispiel:

```
Spatial Index ........:
  parameter:    SPIDX_RTREE
  exist:        Yes
  array form:   -2,0,0
```

Diese Informationen könnte das Skript beim Importieren in die neue SDE-Instanz verwenden, um den GiST-Index direkt in der PostgreSQL-Datenbank zu optimieren.

3.5 Ansätze zum Ersetzen von ArcSDE

Mit dem Austausch des Datenbank-Back-Ends wurde nur ein kleiner proprietärer Teil des ArcGIS-Ökosystems ausgetauscht. Sinnvoller, weil allumfassender, kann ein Austausch der Geo-Middleware ArcSDE selbst sein. Vier unterschiedliche Ansätze werden im Folgenden untersucht, um ArcSDE zu ersetzen:

- Reverse Engineering des Netzwerkprotokolls, das zwischen den Desktop-Anwendungen ArcCatalog/ArcMap sowie ArcSDE verwendet wird
- Nachbau der ArcSDE Client Bibliothek (`sde.dll`) auf der Clientseite
- Anpassung eines OLE/ODBC-Datenbanktreibers für PostgreSQL/PostGIS zur Einbindung in ArcCatalog

- Erstellung einer ArcMap-Desktop-Erweiterung, welche die nötige Funktionalität zur Anbindung von PostGIS-Layern direkt auf dem Desktop-Client bereitstellt

Jeder dieser Ansätze bietet Vor- und Nachteile, sowohl für den Nutzer als auch für den Entwickler, die in den folgenden Abschnitten diskutiert werden.

3.5.1 Reverse Engineering des ArcSDE Netzwerkprotokolls

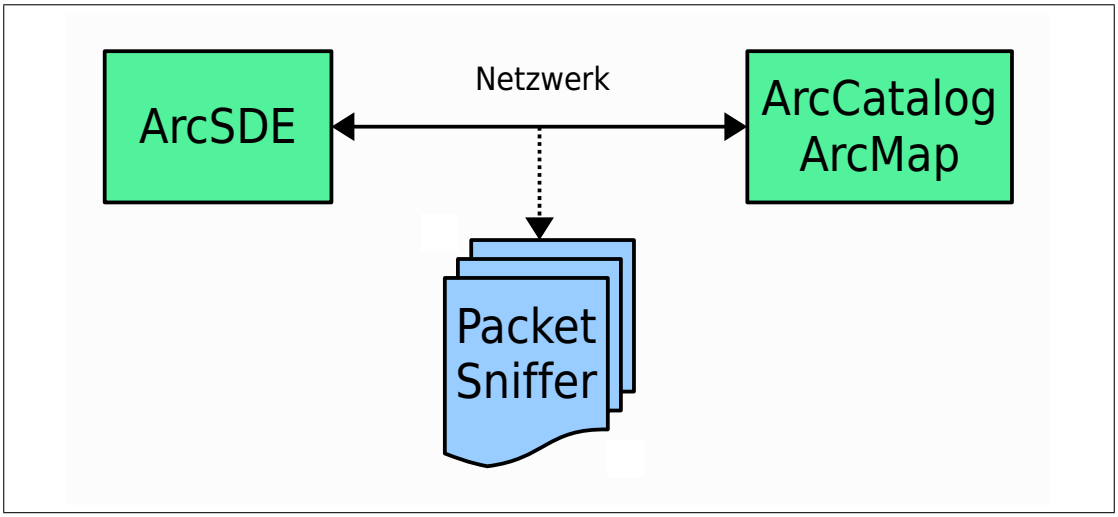

Abbildung 3.3: Abgreifen des Netzwerkprotokolls zwischen ArcSDE und ArcCatalog

Die Desktop-Anwendungen ArcCatalog und ArcSDE kommunizieren über eine TCP/IP-Netzwerkverbindung. Das Protokoll, das bei der Verbindung zwischen ArcSDE und ArcCatalog verwendet wird, ist nicht öffentlich dokumentiert. Es ist jedoch möglich, die Kommunikation der Programme mittels eines *Packet Sniffers* zu belauschen (vgl. Abbildung 3.3). Mit den abgehörten Daten lassen sich Rückschlüsse auf die genaue Zusammensetzung des Kommunikationsprotokolls ziehen. Alle Erkenntnisse zum ArcSDE-Protokoll finden sich im Anhang C.

Das Untersuchungssetup für die Untersuchung des Protokolls bestand aus zwei virtuellen Maschinen. Auf der ersten Maschine lief ein Windows Server 2003 mit einer ArcGIS-Desktop-Installation. Auf der zweiten Maschine lief ein CentOS 5 Betriebssystem mit einer ArcSDE-Installation auf Basis von PostgreSQL. Da das Abgreifen von Paketen von der Netzwerkschnittstelle unter Windows aus technischen Gründen nicht so einfach möglich ist, wurder der Sniffer Wireshark auf dem Linux-System gestartet.

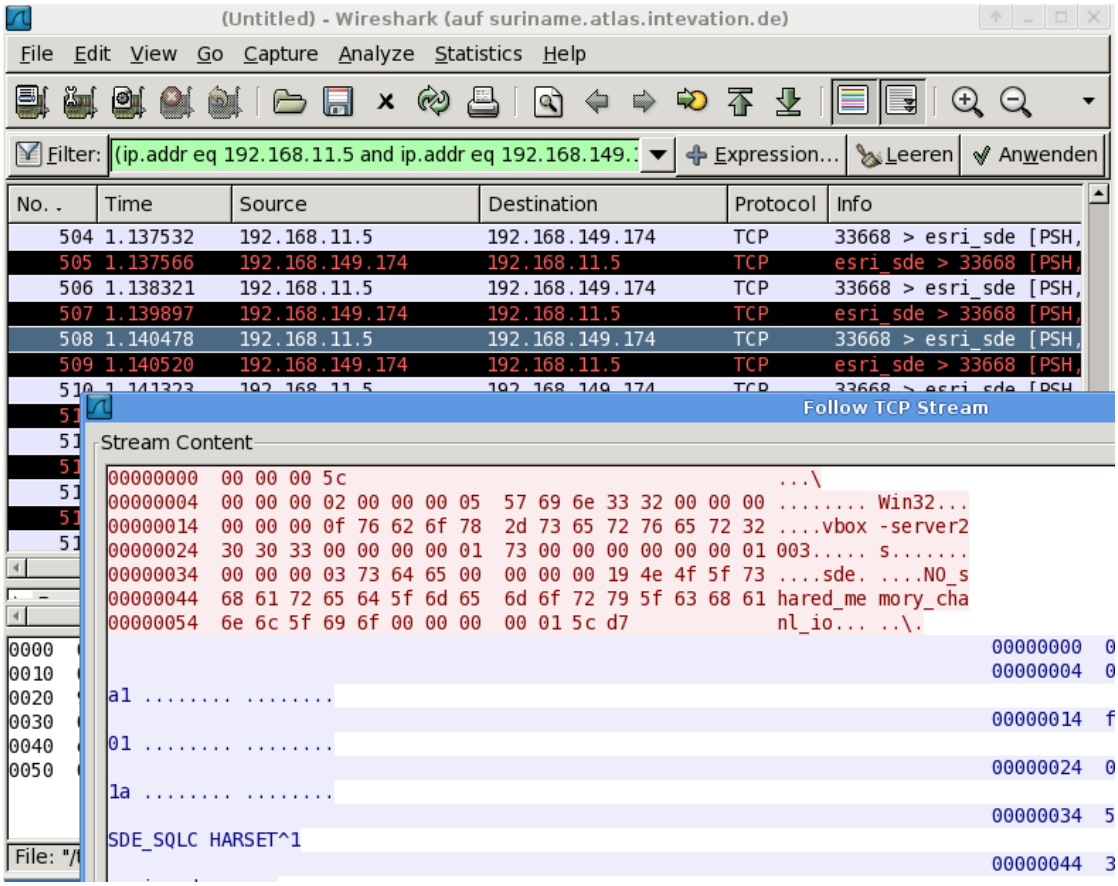

Abbildung 3.4: Sniffen mit dem Werkzeug Wireshark

Wireshark lauschte dabei auf der Netzwerkschnittstelle der virtuellen Linux-Maschine während auf der Windows-Maschine Nutzeraktionen wie Testen, Einrichten und Verbinden von SDE-Verbindungen durchgeführt wurden. Die mitgeschnittenen Pakete konnten daraufhin analysiert werden. Hierzu war insbesondere die Wireshark-Funktion *Follow TCP stream* nützlich, welche die einzelnen mitgeschnittenen TCP-Pakete zu einem kontinuierlichen Strom von Daten zwischen Quelle und Ziel zusammensetzte (siehe Abbildung 3.4, S. 45).

Der Strom der Daten war nur zu einem kleinen Teil menschenlesbar, die Daten waren meist binär codiert und sinnvollerweise (weil sonst unlesbar) nur in der Hexadezimalschreibweise betrachtbar. Das Protokoll war jedoch nicht verschlüsselt und es wurden keine offensichtlichen Versuche unternommen, ein Reverse Engineering des Protokolls zu erschweren (Verschlüsselung, Port-Randomisierung, Timing, etc.).

Beim Testen einer Verbindung über den Button *Test Connection* im Einrichtungsdialog für eine neue *Spatial Database Connection* wurden insgesamt drei TCP-Verbindungen zwischen ArcCatalog und der ArcSDE-Instanz aufgebaut. Die Datenpakete wurden in beide Richtungen ausgetauscht. Die ersten beiden Verbindungen waren für jeden Verbindungsaufbau identisch (sofern sich das Testsetup nicht verändert), die dritte Verbindung enthielt jedoch einzelne Datenbytes, die bei jedem Verbindungsaufbau unterschieden. Es handelte sich dabei offenbar um eine zwei Byte große Zählvariable. Es war jedoch unklar, *was* genau hochgezählt wurde. Ein Zusammenhang mit der Systemzeit der Anfrage konnte nicht hergestellt werden. Auch ein Zufallswert konnte ausgeschlossen werden. Offenbar ist der Wert der Zählvariable auch nicht von den variablen Daten der Anfrage abhängig. Wurden die 13 Bytes in der Anfrage auf 0 oder einen zufälligen Wert gesetzt, so änderte dies nichts am Zählverhalten der 2-Byte-Zählvariable.

Der erstellte Prototyp - *OrbSDE*[6] genannt - ignorierte ebenfalls die 13 Bytes der Anfrage und lieferte in der Antwort die Anzahl der erfolgten Verbindungen aus. Für ArcCatalog war dies offenbar akzeptabel, denn es sendete weiter Anfragepakete.

Die Arbeit an dem OrbSDE-Prototypen wurde nicht vertieft, da selbst der initiale Verbindungshandshake zwischen ArcSDE und ArcCatalog nicht sinnvoll nachimplementiert werden konnte. Auch der binäre Aufbau des Protokolls machte es schwer die Semantik der Verbindung zu erkennen. Mit entsprechend höherem Zeitaufwand ließe sich vermutlich ein ArcSDE-Ersatz implementieren, der zumindest die Basisfunktionalität aufweist.

3.5.2 Nachbau der ArcSDE Clientbibliothek

Die ArcGIS Desktop Programme verwenden eine Clientbibliothek, die für die Kommunikation mit ArcSDE zuständig ist. Diese Bibliothek befindet sich auf Windows-Systemen in der dynamischen Bibliothek `sde.dll`, welche sich im ArcGIS Programmordner befindet. Programme von Drittanbietern können diese Bibliothek (API) verwenden, um mit der ArcSDE zu kommunizieren. Die Funktionen der API sind öffentlich dokumentiert, werden von Drittanbietern verwendet und sind daher stabil gegenüber Veränderungen. Da die Bibliothek dynamisch gelinkt ist, d. h. erst bei Programmstart werden vom Loader die genauen Sprungziele für die einzelnen Funktionen der DLL bestimmt, lässt sich die DLL durch eine eigene Variante ersetzen, welche die gleichen Funk-

[6] Bei Interesse kann der Quellcode unter `http://lins.me/master` bezogen werden.

tionen der dokumentierten API bereitstellt, aber eine Verbindung zu einer PostGIS-Datenbank herstellt und aufrecht hält (siehe Abbildung 3.5, S. 47).

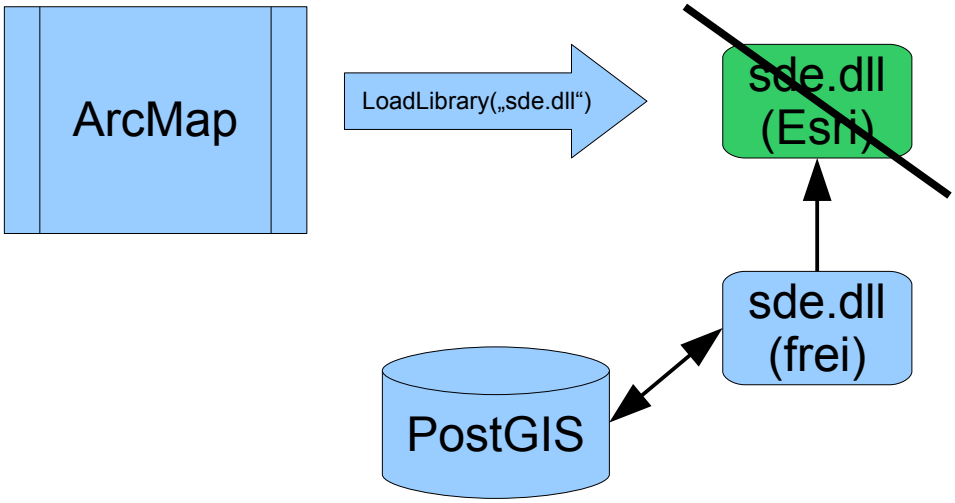

Abbildung 3.5: Austausch der originalen sde.dll durch eine freie Variante

Die Idee dahinter ist, dass sich die selbsterstellte DLL nach Außen hin so verhält, als wäre sie die originale von Esri. Im Inneren ist sie eine Eigenentwicklung und kann daher transparent z. B. eine Verbindung zu einer PostGIS-Datenbank aufbauen und Daten von ihr beziehen.

Vor der Umsetzung dieser Idee mussten einige Voraussetzungen geprüft werden: zu klären war zunächst, ob die vorhandene `sde.dll` nicht nur von Drittsoftware verwendet wird, sondern auch von Esris eigenen Desktop-Programmen. Ohne diese Voraussetzung kann den ArcGIS-Programmen keine modifizierte DLL untergeschoben werden. Erschwert wurde die Untersuchung hier, weil die `sde.dll` nicht nur einmal im Esri Programmordner vorhanden ist. Sie liegt bei der Standardinstallation von ArcGIS 10 mindestens an folgenden Orten:

```
C:\Program Files\ArcGIS\Desktop10.0\Bin
C:\Program Files\ArcGIS\Engine10.0\bin
C:\Program Files\ArcGIS\ArcSDE\pgexe\bin
```

Der Windows DLL Loader kennt mehrere Pfade, in denen er nach DLLs zum Laden sucht (vgl. [Microsoft, b]):

1. Das Verzeichnis, in dem sich die ausführbare Datei des laufenden Prozesses befindet
2. Das Ausführungsverzeichnis
3. Das Windows-Systemverzeichnis (z. B. `C:\Windows\System32`), das der Loader mit der Funktion `GetSystemDirectory()` bestimmt
4. Das Windows-Verzeichnis (z. B. `C:\Windows`), das der Loader mit der Funktion `GetWindowsDirectory()` bestimmt
5. Die Pfade, die in der PATH-Umgebungsvariable spezifiziert sind

Wenn also ein Programm mit der Windows-API-Funktion `LoadLibrary` (vgl. [Microsoft, a]) das Laden einer DLL anfordert und diese über ihren Namen identifiziert, so prüft der Windows DLL Loader nacheinander die oben genannten Pfad nach einer DLL dieses Namens. Hier bietet sich ein Ansatzpunkt: wenn der Loader eine DLL des geforderten Namens findet, so lädt er diese. Es findet keine weitere Überprüfung statt, ob die DLL auch tatsächlich diejenige ist, die das aufrufende Programm erwartet. Es ist also prinzipiell möglich, eine `sde.dll` zu erstellen, die dann vom ArcCatalog/ArcMap Prozess in Erwartung der SDE-Client-Bibliothek geladen wird.

Mit dem Windows-Port des freien GCC Compilers MinGW[7] wurde eine leere DLL erstellt, die zunächst keinerlei Funktionen bereitstellt. Die selbst erstellte `sde.dll` wurde dann im Installationsverzeichnis von ArcCatalog plaziert (`C:\Program Files\ArcGIS\Desktop10.0\Bin`) und ersetzte dort die originale DLL.

Abbildung 3.6: ArcMap Fehlermeldung

ArcCatalog und ArcMap reagierten darauf zunächst mit Fehlermeldungen (vgl. Abbildung 3.6). In der Fehlermeldung wurde die Funktion genannt, die ArcMap erwartete.

[7] Minimalist GNU for Windows, siehe `http://www.mingw.org/`

Sofern die Funktion Teil der öffentlichen API war, konnte sie zunächst leer, d. h. ohne Funktionalität, implementiert werden (Stub). Anhand ihrer Funktionsbeschreibung in der API-Dokumentation konnte sie später voll funktional nachimplementiert werden. Problematisch sind Funktionen, welche die `sde.dll` bereitstellt und die *nicht* öffentlich dokumentiert sind. Bei solchen Funktionen konnte nur anhand des Namens erahnt werden, welche Funktionalität sie haben. Zusätzlich waren auch Rückgabetyp sowie Anzahl und Typ der Funktionsparameter unbekannt. Diese ließen sich u. U. durch Testen verschiedener Kombinationen sowie Disassemblierung der originalen `sde.dll` bestimmen. Bei der Disassemblierung der DLL in menschenlesbaren Assemblercode ließe sich z. B. erkennen, wie viele Bytes vor einem Funktionsaufruf auf den Stack gelegt werden. Daraus ließen sich Hinweise sammeln um wie viele und welche Parameter es sich handelt. Ein Funktionsaufruf einer Funktion, die zwei 32-Bit Zahlen entgegen nimmt, könnte beispielsweise so aussehen:

```
push 0x23840092      ; die Hexadezimalzahlen 0x23840092 und 0x10020121
push 0x10020121      ; auf den Stack legen
call funktion        ; die Funktion 'funktion' aufrufen
```

Der Assemblercode wäre für eine Funktion mit einem acht Byte großen Parameter identisch, die Information ob es sich um zwei 32-Bit oder einen 64-Bit großen Parameter handelt, lässt sich beim Disassemblieren nicht wieder extrahieren. Anhand dieser Informationen kann man zumindest das simple Raten etwas genauer einschränken. In einigen Fällen stürzten ArcCatalog/ArcMap aber auch ohne Fehlermeldungen ab, die Suche nach den korrekten Funktionsparametern gestaltet sich daher zeitaufwändig oder ohne Kenntnis des Quelltextes sogar unmöglich. Nicht dokumentierte Funktionen nachzuimplementieren kann sich auch als Sisyphusarbeit erweisen, denn niemand garantiert, dass die Funktion in einem späteren ArcGIS-Release noch existiert oder die gleiche Funktionalität bereitstellt. Die Disassemblierung zum Zweck der Herstellung eines konkurrierenden (freien) Softwareprodukts ist zudem in Deutschland nicht legal (vgl. §69e, UrhG). Da die Entwicklung dieses Prototyps mit erheblichem Zeitaufwand und Zufallsfaktoren belegt war, wurde sie im Rahmen dieser Arbeit nicht weiter verfolgt.

Sofern es also nicht möglich ist, die `sde.dll` von Esri vollständig nachzubauen, könnte ein anderer Ansatz erfolgsversprechender sein: die Software `dumpbin` kann die Namen aller von der originalen `sde.dll` bereitgestellten Funktionen exportieren. Mit Hilfe eines Skriptes aus [Ethicalhacker] lassen sich die exportierten Funktionsnamen in Pragma-Direktiven für den Linker übersetzen, z. B. (vgl. [Ethicalhacker]):

```
#pragma comment(linker, "/export:MyFunc=originalDLL.MyFunc")
```

Mit dieser Direktive wird ein Aufruf Funktion `MyFunc` auf die gleichnamige Funktion in der DLL `originalDLL.dll` umgebogen. Somit könnte man eine DLL erzeugen, die alle Aufrufe an die originale `sde.dll` durchreicht. Einzelne Funktionen könnte man selbst nachimplementieren, unbekannte Funktionen reicht man an die Original-DLL weiter. Die Original-DLL sollte bei der Installation der ArcGIS Desktop Anwendungen installiert werden und liegt somit auf den Clientsystemen vor. Bei dieser Lösung müssen undokumentierte Funktionen nicht nachimplementiert werden. Es ist jedoch wahrscheinlich, dass die Original-DLL intern Zustände speichert, die nach Außen hin nicht sichtbar sind. Wenn aber einzelne Aufrufe an die DLL durch eine eigene DLL abgefangen werden, ist es möglich, dass diese internen Zustände nicht in der erwarteten Art und Weise verändert werden. Die Zustände könnten inkonsistent werden, was sich durch Fehlermeldungen oder Abstürze von ArcMap äußern könnte. Im Rahmen der Arbeit wurde dieser Lösungsansatz nicht weiter verfolgt, da er erst nach Abschluss der Implementierungsphase gefunden wurde. Er sollte in einer weiteren Untersuchung genauer bewertet werden.

3.5.3 OLE/ODBC-Datenbanktreiber mit Unterstützung für räumliche Datentypen

Die ArcGIS-Desktop-Produkte unterstützen das *Object Linking and Embedding* (OLE) Objektsystem, das für die Interoperabilität der unterschiedlichsten Softwaresysteme auf Windows-Betriebssystemen verwendet wird. Bekannt ist beispielsweise die Einbettung von Excel-Tabellen in Word-Dokumente über die OLE-Schnittstelle. OLE kann bei ArcCatalog zur Anbindung von Datenbanken verwendet werden. Beispielsweise lassen sich PostgreSQL-Datenbanken über die generische OLE-ODBC-Schnittstelle anbinden. Die vorhandene OLE-Anbindung bringt keine Unterstützung für räumliche Datentypen mit. Es wurde daher überprüft, ob OLE/ODBC prinzipiell geeignet ist, geometrische Daten von einer relationalen Datenbank an ArcCatalog anzubinden, auch wenn ArcCatalog diese Funktionalität von Haus aus nicht mitbringt.

Eine Test mit ArcCatalog 10 und PostgreSQL 8.3 zeigte, dass sich PostgreSQL an ArcCatalog anbinden lässt, zumindest was die reinen Attributtabellen betrifft. Geometriedatentypen wie der ESRI-eigene `ST_Geometry` wurden ignoriert und als Binärdaten angezeigt. Der Versuch, eine Tabelle mit Geometrie als FeatureClass zu exportieren,

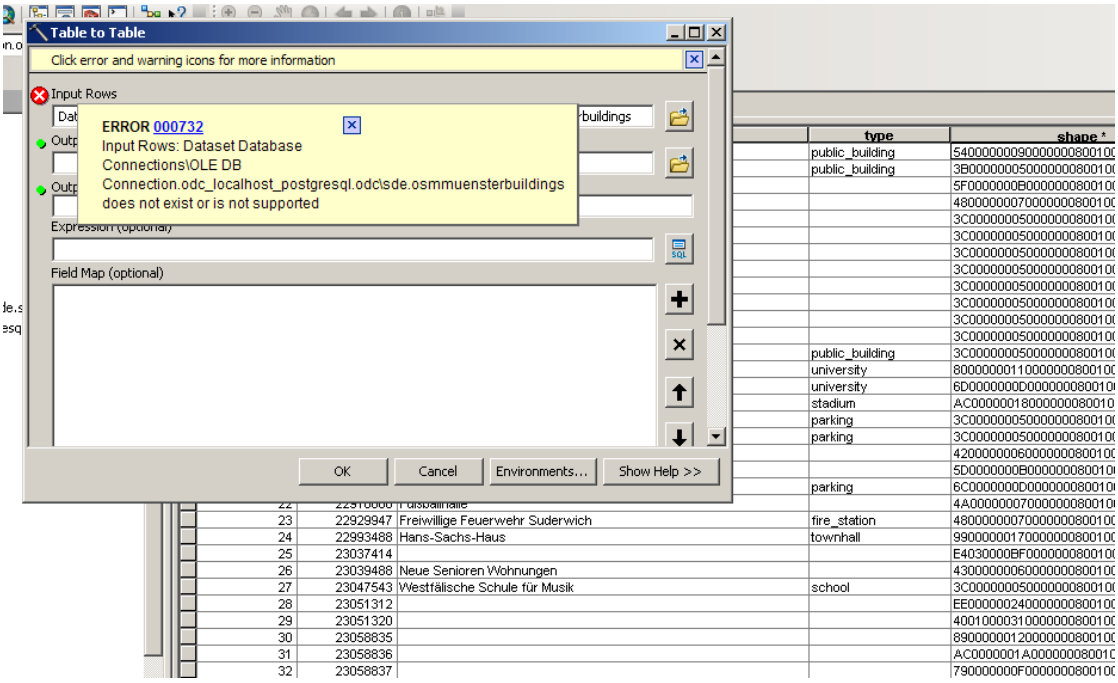

Abbildung 3.7: ArcCatalog mit einer PostgreSQL Tabelle

wurde mit einer Fehlermeldung quittiert (vgl. Abbildung 3.7). Ebenso fehlten Menüeinträge zum Erstellen neuer FeatureClasses oder für den Import vorhandener Daten. Es lag Verdacht nahe, dass es sich hierbei um einen Nur-Lese-Zugriff auf relationale Daten ohne räumlichen Bezug handelt. Es war zu klären, ob dies eine gewollte Einschränkung von Esri ist, um beispielsweise ihre proprietäre ArcSDE-Anbindung zu fördern oder ob OLE/ODBC generell keine Unterstützung für räumliche Datentypen bietet.

Dazu wurde der Quelltext des PostgreSQL-ODBC-Treibers[8] analysiert. In der Datei `convert.c` befand sich Code, der zwischen SQL-C-Typen und PostgreSQL-Typen konvertiert. Es sollte möglich sein, in diesen Code Konvertierungsfunktionen für die PostGIS-Datentypen einzufügen. Die bekannte GDAL-Bibliothek unterstützt räumliche Datentypen auch über die ODBC-Schnittstelle (vgl. [GDAL]). Unklar blieb, ob es möglich ist, eine Unterstützung für den Esri-eigenen `ST_Geometry`-Datentyp einzubauen, denn dafür muss eine entsprechende Schnittstelle in ArcGIS existieren, es genügt nicht, wenn nur der Treiber `ST_Geometry` versteht.

[8] Heruntergeladen von http://ftp.postgresql.org/pub/odbc/versions/src/

Da Esri in ihrer Knowledgebase jedoch bestätigte, dass die OLE/ODBC-Schnittstelle keinerlei Bearbeiten-Funktionalität vorsieht (vgl. [Esri, a]), erübrigte sich eine weitere Untersuchung.

3.5.4 ArcGIS Desktop Erweiterung mit PostGIS-Anbindung

Ein idealer Ersatz für die ArcSDE wäre ein transparenter Austausch im Daten-Back-End. Wie in den vorhergehenden Abschnitten deutlich wurde, ist dies nicht ohne Schwierigkeiten möglich. In diesem Abschnitt wird daher ein Ersatz vorgestellt, der ohne eine SDE-ähnliche Middleware auskommt und ausschließlich auf dem Desktop-System des Anwenders verwendet wird: eine Erweiterung für die ArcGIS-Desktop-Produkte. Eine solche Erweiterung soll eine Verbindung zu einer PostGIS-Datenbank halten und folgende Funktionen für den Anwender bereitstellen:

- Konfigurieren und Herstellen der Datenbankverbindungen zu PostGIS-Datenbanken
- Hinzufügen einer Geometrietabelle als `FeatureClass` zum aktuellen Dokument
- Transparenter Lese- und Schreibzugriff auf die Geometrietabellen über die Funktionen von ArcMap
- Export von FeatureClasses in PostGIS-Datenbanken

Optional, aber sinnvoll sind folgende Funktionalitäten:

- Durchsuchen einer PostGIS-Datenbank nach Geometrietabellen
- Export von Geometrietabellen in z. B. eine Shapefile
- Ausführen von SQL-Anfragen an die PostGIS-Datenbank und Visualisierung des Ergebnisses

Die Anbindung an PostGIS im freien Desktop-GIS Quantum GIS stellt eine vergleichbare Funktionalität zur Verfügung (vgl. Abbildung 3.8).

Eine solche Erweiterung verwendet ausschließlich die dokumentierten und getesteten Schnittstellen von ArcGIS und PostgreSQL/PostGIS, die sich aus Kompatibilitätsgründen kaum verändern werden. Der Entwickler ist somit nicht auf Kenntnisse angewiesen, die mit Reverse Engineering erlangt wurden.

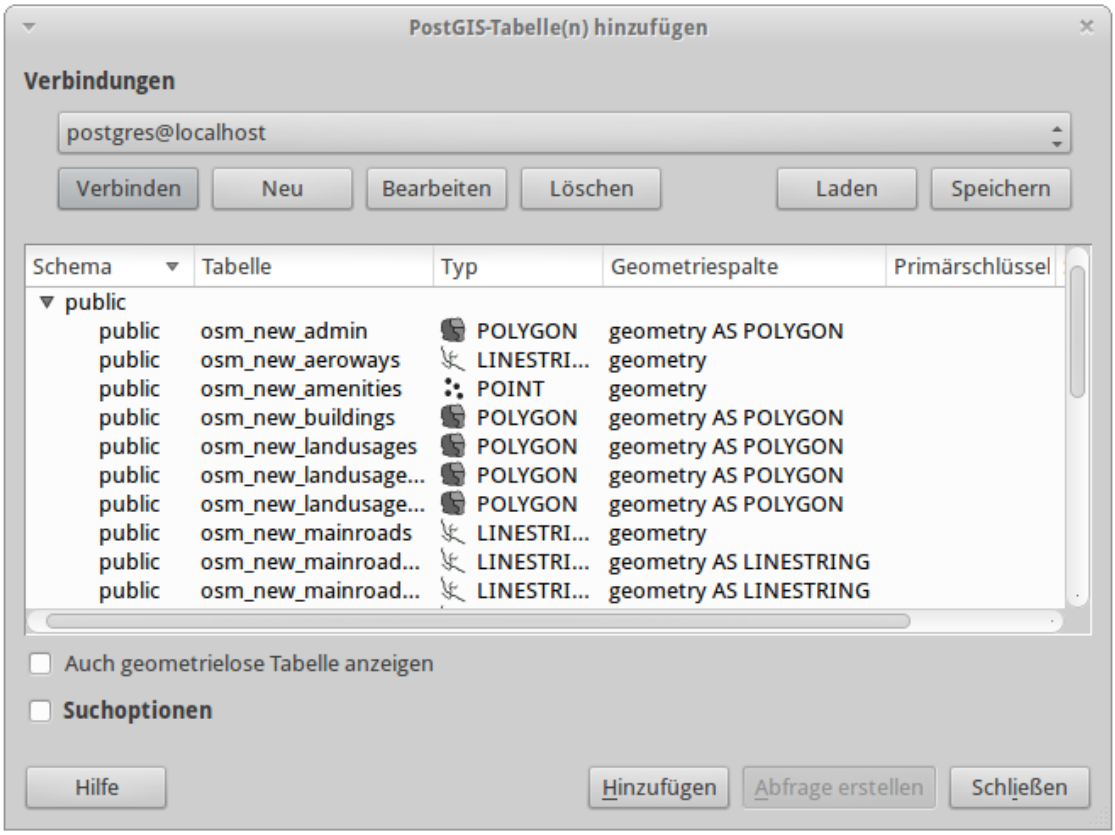

Abbildung 3.8: PostGIS-Layer in Quantum GIS hinzufügen

Die Erweiterung wurde mit dem ArcGIS Software Developer Kit (SDK) für das .NET-Framework verwendet. Als Programmiersprache wurde C# verwendet. Zur Anbindung an PostGIS verwendete die Erweiterung die Npgsql-Bibliothek[9], die *Freie Software* unter einer MIT/BSD-ähnlichen-Lizenz ist (siehe S. 12).

Die umfangreiche Schnittstelle von ArcGIS erlaubte hier verschiedene Ansätze, mit unterschiedlichen Stärken, Schwächen und Möglichkeiten.

Der erste Entwurf der Erweiterung sah nur die Möglichkeit vor, PostGIS-Tabellen aus der angebundenen Datenbank in das aktuelle ArcMap-Dokument zu kopieren und nach Bearbeitung gegebenfalls wieder in die PostGIS-Datenbank zu exportieren. Die Erweiterung stellte dabei eigene Oberflächenelemente zur Verfügung. Die Datenhaltung war dabei nicht transparent, denn der Benutzer musste sein Verhalten beim Bedienen von ArcMap anpassen (Kopieren, Bearbeiten und dann Zurückkopieren). Wünschenswert

[9]Siehe `http://npgsql.projects.postgresql.org/`

war es daher, dass der Nutzer transparent ArcMap und ArcCatalog verwenden kann und sein gewohntes Verhalten bei der täglichen Arbeit nicht anpassen muss. Mit einer benutzerdefinierten FeatureClass konnte dies realisiert werden.

Die ArcGIS-Programmierschnittstelle ArcObjects basiert auf der COM-Technologie von Microsoft. COM steht für Component Object Model und ist ein binäres Schnittstellenverfahren für Softwarekomponenten. COM ist unabhängig von der eingesetzten Programmiersprache und kann daher z. B. mit .NET/C#, C++ oder Java verwendet werden. ArcObjects stellt einen Wrapper (eine Hülle) für die COM-Schnittstelle zu ArcGIS bereit, die an vielen (jedoch nicht an allen Stellen) den COM-basierten Unterbau unter der modernen Programmierumgebung .NET verbirgt. Für den Prototyp der Erweiterung PostArc wurde daher .NET/C# verwendet.

Selbst wenn mit der Anwendung PostGIS-Tabellen als Layer in ArcMap transparent verwendet werden können, so sollte der Nutzer zumindest die Möglichkeit haben, die PostGIS-Datenbankverbindung zu konfigurieren. Dazu stellte die PostArc-Erweiterung einen Konfigurationsdialog zur Verfügung, der über einen entsprechenden Button in einer Toolbar von ArcMap geöffnet werden konnte.

Mit der Software ZigGIS gibt es ein ähnliches Softwareprodukt, das jedoch nicht mehr weiterentwickelt wird. Die erste Version mit abgespeckter Funktionalität ist seit einiger Zeit unter der GPL verfügbar, neuere Versionen wurden nur proprietär entwickelt. Der Quellcode der ersten GPL-Version lag dem Autor dieser Arbeit vor, die im Rahmen dieser Arbeit erstellte Erweiterung (Projektname PostArc) wurde jedoch von Grund auf neu entwickelt.

SpatialKit for ArcMap von ST-Links[10] ist eine ähnliche ArcMap-Erweiterung, die im Gegensatz zu ZigGIS aber noch aktiv weiterentwickelt wird. Die Software ist proprietär, wird allerdings kostenfrei, d. h. als *Freeware* auf der Webseite bereitgestellt. Es bleibt allerdings unklar, wer oder welche Firma sich hinter ST-Links verbirgt, da auf der Webseite keinerlei Informationen über die Firma zu finden sind und die Domain auf eine Privatperson in Kanada registriert ist.

[10] Siehe `http://www.st-links.com/`

3.5.4.1 Erstellen einer neuen `PostGISFeatureClass`

Der Nutzer soll mit der Erweiterung einen neuen Layer erstellen können, dessen geometrische *Features* in einer PostGIS-Tabelle liegen. Die Schnittstelle zwischen Benutzer und Erweiterung ist ein Toolbar-Button. Dieser wird von der Erweiterung erzeugt. Dazu erstellt man eine eigene von `ESRI.ArcGIS.Desktop.AddIns.Button` abgeleitete Klasse, z. B. so:

```
class ButtonNewPostGISLayer : ESRI.ArcGIS.Desktop.AddIns.Button
{
    protected override void OnClick()
    {
        new FormNewPostGISFeatClass().ShowDialog();
    }

    protected override void OnUpdate()
    {
        Enabled = ArcMap.Application != null;
    }
}
```

Über die XML-Konfigurationsdatei `Config.esriaddinx` wird ArcMap von der Existenz dieser Klasse beim Deployen in Kenntnis gesetzt. Der entscheidene Ausschnitt aus der Datei sieht folgendermaßen aus:

```
<AddIn language="CLR" library="PostArc.dll" namespace="PostArc.GUI">
    <ArcMap>
      <Commands>
        <Button id="Intevation_GmbH_PostArc_Button1"
                class="ButtonManagePostGIS"
                message="Manage PostGIS connections and layers"
                caption="Manage PostGIS connections and layers"
                tip="Manage PostGIS connections and layers"
                category="PostGIS" image="Images\Button1.png" />
      </Commands>
    </ArcMap>
</AddIn>
```

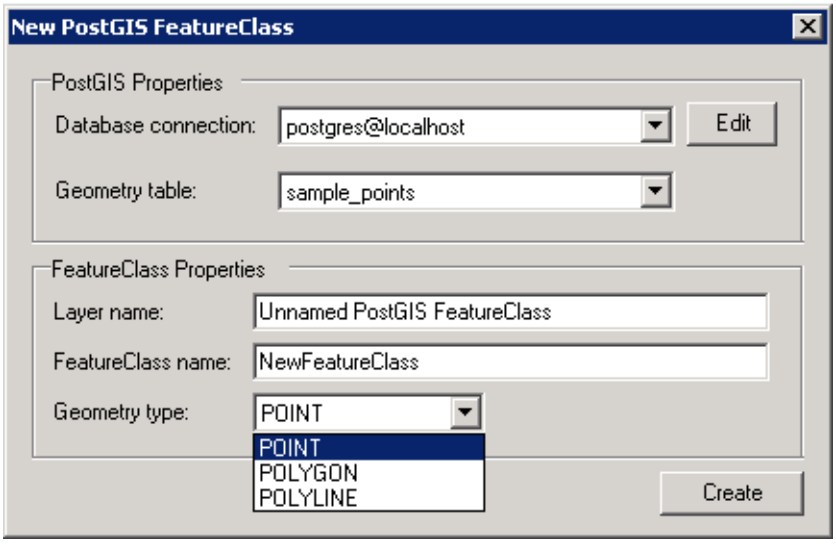

Abbildung 3.9: Dialog zum Erstellen einer PostGIS FeatureClass

Das XML-Fragment führt einen neuen Button ein, dessen Code in der Klasse `PostArc.GIS.ButtonManagePostGIS` liegt. Der Button kann in ArcMap über die *Customize*-Funktion einer Werkzeugleiste hinzugefügt werden. Ein Mausklick auf den Button führt den Code in der überschriebenen `OnClick()`-Methode aus. Der Code öffnet einen Dialog, der diverse Optionen abfragt (siehe Abbildung 3.9, S. 56), bevor mit einem Klick auf *Create* dem aktuell geöffneten Dokument ein neuer Featurelayer mit einer PostGIS-FeatureClass hinzugefügt wird (siehe Abbildung 3.10, S. 57).

Das Erstellen des neuen Layers gelingt mit folgenden Codefragment:

```
FeatureLayer featLayer = new FeatureLayer();
IFeatureClass featClass = new PostGISFeatureClass(esriShapeType.POINT);
featLayer.FeatureClass = featClass;
featLayer.Name = "Neuer PostGIS Featurelayer";
ArcMap.Document.FocusMap.AddLayer(featLayer);
```

Die im obigen Code genannte `PostGISFeatureClass` muss ebenfalls erstellt werden. Theoretisch sollte es genügen eine Klasse zu erstellen, die das Interface IFeatureClass implementiert. Beim Erstellen des Prototyps stellte sich jedoch schnell heraus, dass dies keinesfalls genügt und ArcMap mit `AccessViolationException`s und `COMException`s abstürzt. Erst nachdem alle Interfaces zumindest prototypisch implementiert waren, die

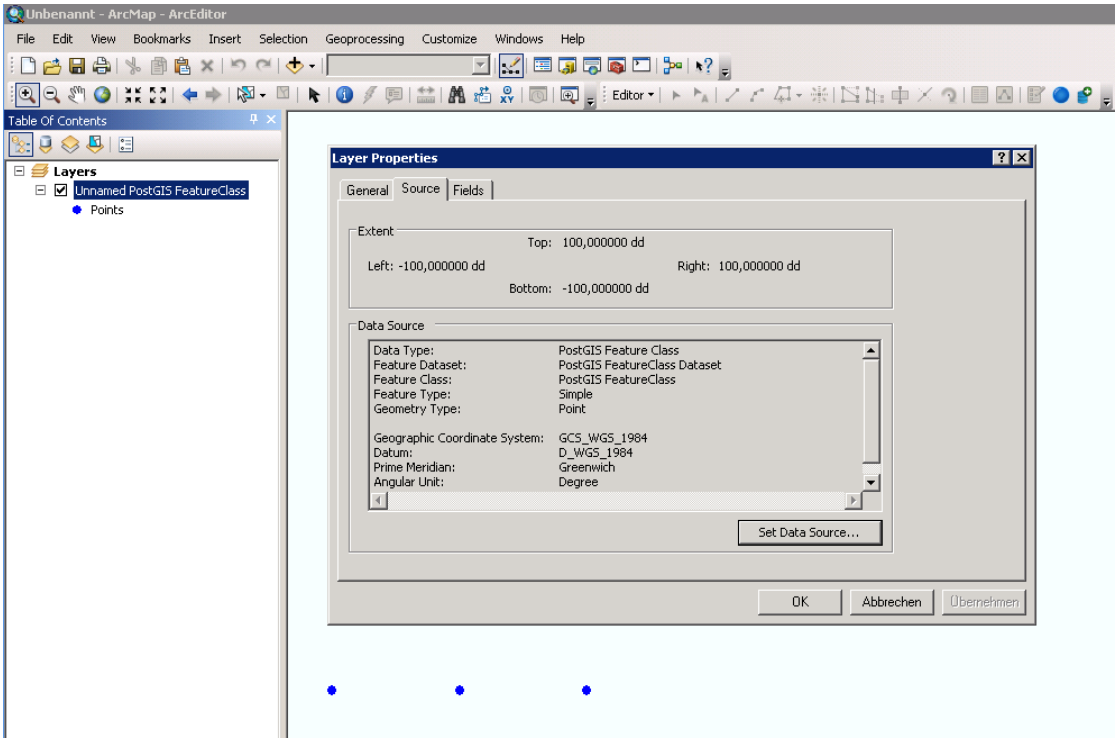

Abbildung 3.10: Die PostGIS FeatureClass in ArcMap

auch die ESRI-Klasse FeatureClass implementiert, akzeptierte ArcMap die PostGIS-FeatureClass. Der Kopf der Klasse PostGISFeatureClass sah folgendermaßen aus:

```
[Guid("77FB1E58-69A9-4bac-AFB6-76F0F85D3B75")/]
public class PostGISFeatureClass :
    IDataset, IDatasetEdit, IDatasetEditInfo,
    IFeatureClass,
    IGeoDataset
    IModelInfo,
    ISchemaLock,
    ISubtypes,
    ITable, ITableCapabilities,
    ITopology,
    IValidation, IValidation2
{
    // ...
}
```

Im erstellten Prototyp lieferte ein Großteil der erstellten Methoden Standardwerte zurück oder warf `NotImplementedExeptions`, aber zumindest ließich die FeatureClass in einen FeatureLayer einbinden und wurde in ArcMap angezeigt (siehe Abbilung 3.10).

3.5.4.2 Anbindung der PostGISFeatureClass an PostgreSQL und PostGIS

Die Anbindung der Erweiterung an PostgreSQL/PostGIS gelang mit der freien Npgsql-Bibliothek[11]. Ausgangspunkt für die Verbindung war ein NpgsqlConnection-Objekt. Es wurde mit der Angabe eines ConnectionStrings erzeugt, der Host, Port, Datenbankname, Benutzername und Passwort der gewünschten Datenbankverbindung enthielt (es sind weitere optionale Parameter möglich), z. B.:

```
NpgsqlConnection conn = new NpgsqlConnection(
    "Server=127.0.0.1;Port=5432;User Id=joe;Password=abc;Database=joedb;");
```

Mit der Methode `Open()` wurde die Datenbankverbindung dann hergestellt:

```
conn.Open(); // Herstellen der Datenbankverbindung
 // .. Arbeiten mit der Datenbankverbindung
conn.Close(); // Schliessen der Datenbankverbindung
```

SQL-Abfragen (siehe Kapitel 2.1.1) werden in Npgsql in der Klasse `NpgsqlCommand` gekapselt. Der Konstruktor der Klasse erhält als Parameter ein SQL-Statement als String und ein Npgsql-Verbindungsobjekt:

```
NpgsqlCommand cmd = new NpgsqlCommand(
    "SELECT ST_AsText(Shape) FROM mygeometries",
    conn);
```

Je nach Art der Anfrage, kann das Ergebnis der Anfrage mit den Methoden `ExecuteNonQuery()`, `ExecuteReader()` oder `ExecuteScalar()` abgefragt werden. Bei einer SELECT-Anfrage wird `ExecuteReader()` verwendet, die ein `NpgsqlDataReader`-Objekt zurück liefert. Das `NpgsqlDataReader`-Objekt ist eine Spezialisierung der .NET-Klasse `System.Data.Common.DbDataReader` und stellt einen Cursor für die Datenbankabfrage dar. Mit folgendem Code lassen sich die Antwortzeilen einer SELECT-Abfrage abrufen:

[11] Siehe http://npgsql.projects.postgresql.org/

```
NpgsqlDataReader cursor = cmd.ExecuteReader();
while(cursor.Read())
{
    Console.Write("Erstes Feld: " + cursor[0]);
}
```

Die NPGSQL-Bibliothek bringt bereits die Unterstützung für die Geometrietypen von POSTGIS mit. Mit entsprechenden Cast-Operationen lassen sich die Geometrieobjekte aus den Antwortzeilen der Datenbank auslesen:

```
while(cursor.Read())
{
    NpgsqlPolygon area = (NpgsqlPolygon)cursor[0];
    // ... mit Polygon weiter arbeiten
}
```

Offenbar ist der POSTGIS-Support in NPGSQL noch nicht ausgereift, da der oben beschriebene Weg nicht funktionierte. Die POSTGIS-Geometrien wurden daher als Well-Known-Text (WKT) zwischen POSTGIS und der Erweiterung ausgetauscht.

Da die ArcObjects-API die NPGSQL-Typen nicht kennt bzw. WKT-Repräsentationen nicht interpretieren kennt, musste eine Umsetzung in die Geometrien von ArcObjects erfolgen. Im Namespace `PostArc.Feature` befanden sich Implementierungen der ArcObjects-Interfaces `IPoint`, `IPolyline` und `IPolygon`. Diese Implementierungen kapselten die entsprechenden Geometriedaten und erweiterten die ArcObjects-Geodatentypen z. B. um eine SetWKT-Methode:

```
public class Point : IPoint
{
    protected PointClass point;
    public double X
    {
        get { return this.point.X; }
        set { this.point.X = value; }
    }
...
```

```
    public void SetWKT(string wkt)
    {
        wkt = wkt.Split(new char[] { '(', ')' })[1];
        string[] wkts = wkt.Split(' ');
        X = double.Parse(wkts[0]);
        Y = double.Parse(wkts[1]);
    }
...
}
```

3.5.4.3 Registrierung der Komponenten im System

Jede erstellte Klasse, die eine ArcObjekts-Komponente darstellt, musste mit einer eindeutigen ID, der sogenannten GUID (Globally Unique Identifier), versehen werden. Die GUID ist weltweit eindeutig, Kollisionen sind extrem unwahrscheinlich. Damit ArcMap bzw. ArcCatalog die erstellten Erweiterungen identifizieren und starten konnte, mussten diese in der Windows-Systemregistrierung registriert werden. COM erlaubt dazu die Definition von *Component Categories*, d. h. Komponentenkategorien, in denen gleichartige Komponenten organisiert werden. Jede Komponente einer bestimmten Kategorie implementiert eine Menge von vereinbarten COM-Schnittstellen. Abbildung 3.11 zeigt den Windows-Registrierungseditor mit einer registrierten Komponente. In den Subkeys werden die Component Categories vermerkt, zu denen diese Komponente gehört. Im Subkey InprocServer32 wird ein Verweis auf die DLL der Komponente gespeichert. Bei der Installation von ArcGIS wurden bereits zahlreiche Component Categories für die einzelnen Teilbereiche erstellt und die Esri-eigenen Komponenten entsprechend registriert. Es gibt beispielsweise Kategorien für Workspace Factories, d. h. Komponenten, die Workspace Objekte, die das Interface IWorkspace implementieren, erzeugen können.

Die Registrierung der DLL erfolgte mit dem RegAsm-Tool. Das Tool führte die Registrierungsmethode der DLL aus. Welche Methode dies ist, steht in den Metadaten der DLL. Im Quellcode wurde die Registrierungsmethode mit einer `[ComRegisterFunction]` annotiert, z. B.:

```
[ComRegisterFunction]
static void Register(string key)
{
```

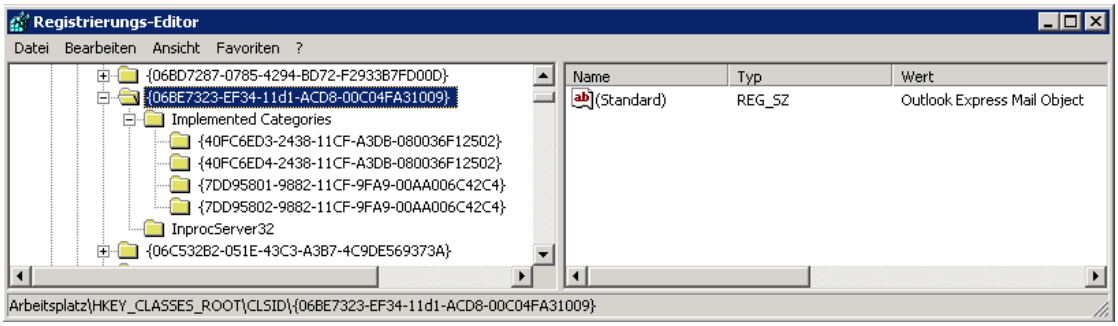

Abbildung 3.11: Windows-Registrierungseditor mit einer registrierten Komponente.

```
    // Register PostGISFeatureClass
    System.Type type = typeof(PostArcClassDescription);
    GeoObjectClassDescriptions.Register(type.GUID.ToString());
}
```

Diese Methode wird generell einmal für jede zu registrierende DLL aufgerufen. Im obigen Beispiel verwendet die Methode eine Hilfsklasse aus dem `ESRI.ArcGIS.ADF-Namespace`, welche zur Registrierung von Komponenten in den entsprechenden Esri-Kategorien dient.

Beim Start von ArcMap prüft das Programm die in der Windows-Systemregistrierung registrierten Komponenten auf die von ArcMap verwendeten Kategorien und versucht diese zu laden und zu initialisieren. Gelingt dies, steht die entsprechende Erweiterung in ArcMap zur Verfügung. Analog dazu verfahren die anderen ArcGIS-Desktop-Programme.

3.5.5 Bewertung der Ansätze und Prototypen

Wie sich bei der Untersuchung gezeigt hat, sind die Prototypen unterschiedlich erfolgsversprechend für eine vertiefte Betrachtung und Realisierung.

Der Nachbau von ArcSDE mittels Reverse Engineering des Protokolls erscheint aus mehreren Gründen nicht attraktiv. Zum einen ist das Reverse Engineering sehr mühsam und der Zeitaufwand lässt sich nur schwer abschätzen. Zudem könnten im Laufe der Weiterentwicklung Hindernisse auftreten, die sich überhaupt nicht technisch lösen lassen. Wenn beispielsweise bestimmte Teile des Protokolls - auch wenn dies nach den bisherigen Erkenntnissen unwahrscheinlich ist - verschlüsselt oder mit Signaturen versehen

sind, ist ein Reverse Engineering dieser Protokollteile äußerst schwer. Abgesehen von den technischen Schwierigkeiten stellt sich die Frage, ob das Konzept einer Geodatenbank-Middleware heute noch sinnvoll ist. Laut Mitarbeitern des Bundesamtes für Seeschifffahrt und Hydrographie ist das Konzept veraltet, nicht mehr zeitgemäß und es erweist sich zunehmend als Flaschenhals (u. a. wegen fehlender Multiprozessorunterstützung) (vgl. Anhang D.1). Vorteil dieser Lösung ist, dass sie vollkommen transparent für den Endanwender auf dem Desktop wäre.

Eine vergleichbare Bewertung gilt für den Nachbau der SDE-Clientbibliothek mit einer `sde.dll`. Eine solche Lösung basiert ebenfalls auf Wissen, das durch Reverse Engineering gewonnen wird, da die API nicht vollständig öffentlich dokumentiert ist. Der Erfolg einer Weiterentwicklung kann also nicht garantiert werden und die Aufwandsabschätzung ist damit kaum möglich. Selbst wenn es gelingt eine DLL zu erstellen, die alle Funktionsaufrufe an die originale DLL weiterleitet und nur die mit Konnektivitätsbezug zur ArcSDE reimplementiert, ist diese Lösung vermutlich nicht von Dauer. Mit einem neuen Release von ArcGIS könnte die originale DLL anders aufgebaut sein und die dazwischengeschaltete DLL nicht mehr funktionieren. Im Gegensatz zu einer offiziell dokumentierten Änderung, müsste in einem solchen Fall das Reverse Engineering wieder beginnen.

Der Ansatz mit einem modifizierten OLE- oder ODBC-Datenbanktreiber ist ebenfalls nicht erfolgsversprechend. Zwar ließe sich mit entsprechendem Aufwand ein Treiber mit der Unterstützung für räumliche Datentypen erstellen (bzw. z. T. sind sie im psqlodbc-Treiber schon vorhanden), jedoch ist dies nicht von Nutzen, da offenbar die Unterstützung dafür auf Seiten der ArcGIS-Desktop-Produkte fehlt, die - mangels Quelloffenheit - nicht nachprogrammiert werden können. Abgesehen davon verfügt die Implementierung der Schnittstelle in ArcGIS über keine Schreibfähigkeit (vgl. [Esri, a]), der Nutzwert für die Zwecke eines SDE-Ersatzes ist damit nicht gegeben.

Am erfolgversprechendsten scheint der letzte Ansatz der ArcObjects-Erweiterung zu sein. Nachteilig ist, dass dazu auf dem Clientsystem die Erweiterung installiert wird und der Benutzer eine leicht andere Bedienung erlernen muss. Dafür verwendet die Erweiterung offizielle, von Esri unterstützte und dokumentierte Schnittstellen und ist nicht auf geheimes Wissen angewiesen. Auch macht die Erweiterung eine Middleware-Zwischenschicht auf dem Server obsolet und kann direkt mit der Geodatenbank PostGIS kommunizieren. Der Prototyp der ArcObjects-Erweiterung ***PostArc*** ist daher der erfolgversprechendste Ansatz zur Anbindung von PostGIS an ArcMap und ArcCatalog.

3.6 Migrationsstrategien

In Kapitel 3.1 wurde festgestellt, dass die ArcSDE ausschließlich als Datenspeicher verwendet wird. Die Geofunktionalitäten der ArcSDE werden nicht verwendet. Zwar ist die Anzahl der befragten Anwender für eine statistisch haltbare Aussage zu gering, aber es ist ein Indiz, wenn keiner der drei deutlich unterschiedlich großen Anwender die Geofunktionalitäten benötigt. Eine der Hauptfunktionalitäten der ArcSDE spielt damit in der Praxis für die befragten Anwender keine Rolle. Bei der Einführung der ArcSDE waren relationale Datenbanksysteme mit Geofunktionalität noch nicht weit verbreitet. Die ArcSDE war somit eine der wenigen tauglichen Lösungen Datenbanken mit Geofunktionen und Desktop-Programmen zu verbinden. Heutige Datenbanksysteme bieten vielfach schon ausreichende Unterstützung für Geofunktionen, sodass auch dieser Vorteil für die ArcSDE wegfällt, sofern es gelingt, die Datenbank direkt an ArcMap anzubinden. Eine weiterentwickelte PostArc-Erweiterung kann diese Aufgabe erfüllen. Da die Erweiterung aber noch nicht reif für den Produktivbetrieb ist, ist ein völliger Verzicht auf die ArcSDE derzeit noch etwas für offene und experimentierfreudige Anwender.

Risikoarm dagegen ist der Austausch der proprietären Datenbank (Oracle Database, Microsoft SQL Server, etc.) unterhalb der ArcSDE durch die freie Datenbanksoftware PostgreSQL, da die Firma Esri PostgreSQL in Kombination mit ArcSDE offiziell unterstützt. Für Anwender kann dies ein erster Schritt sein, Erfahrungen mit freier Software bzw. mit einem freien Softwaredienstleister zu sammeln.

Die Migration von im Produktivbetrieb laufender Software auf eine neue Software ist eine kritische Phase der Veränderung und will gut geplant und vorbereitet werden. Dieses Kapitel ist zweigeteilt: zum einen wird die Migration des Datenbank-Back-Ends von ArcSDE auf PostgreSQL beschrieben, zum anderen wird beschrieben, wie ArcSDE durch die Desktop-Erweiterung PostArc ersetzt werden kann. Die Migration des Datenbank-Back-Ends kann als erster Schritt in Richtung *Freie Software* und Ablösung der proprietären Datenbank und Datenbankmiddleware verstanden werden. Beide Migrationen sind aber prinzipiell voneinander unabhängig durchführbar.

3.6.1 Migration des Datenbank-Backends auf PostgreSQL

Vor der Migrationsentscheidung muss sichergestellt sein, dass die Administratoren über ausreichend Wissen über das DBMS PostgreSQL verfügen oder sie von einem qualifizierten Dienstleister unterstützt werden, damit die Migration nicht an technischer Unkenntnis scheitert.

Die zur Migration nötigen Schritte sind im Einzelnen:

3.6.1.1 Bestimmen der Funktionalität

Vor der eigentlichen Migration muss die Funktionalität des bisherigen Systems bestimmt werden. Das bedeutet, dass alle Funktionen von ArcGIS, die für die Arbeit mit dem GIS benötigt werden, schriftlich festgehalten werden, z. B. in einem Lastenheft. Dazu gehören auch z. B. selbsterstellte Erweiterungen oder spezielle Konfigurationseinstellungen. Auch Werkzeuge von Drittanbietern, die mit ArcGIS oder mit ArcGIS-Daten in irgendeiner Form interagieren, müssen hierbei berücksichtigt werden.

3.6.1.2 Export der Bestandsdaten

Der erste Schritt bei der Migration ist der Export der Bestandsdaten aus der alten ArcSDE-Instanz. Im Rahmen dieser Arbeit entstand ein Python-Skript[12], das den Export von FeatureClasses aus einer SDE-Instanz vereinfacht. Das `dumpsde` genannte Skript verwendet die ArcGIS-Kommandozeilentools `sdeexport` und `sdeimport`. Mit dem `dumpsde`-Skript lässt sich ein Export von FeatureClasses folgendermaßen durchführen:

```
dumpsde.py out feature0,feature1,feature2 -u sde -p sde
```

Das Skript erstellt feature0.sdex, feature1.sdex, usw. Dateien, welche die exportierte FeatureClasses beinhalten. Diese Dateien lassen sich in eine andere SDE-Instanz importieren (s. Abschnitt 3.6.1.5).

Alternativ lassen sich die Daten der SDE-Instanz auch über ArcCatalog exportieren. Als Formate sind dazu beispielsweise *Personal Geodatabases* oder Shapefiles geeignet.

[12] Zu finden unter `http://lins.me/master`

Für großere Datenmengen ist dieser Weg jedoch zu zeitintensiv. Während der Untersuchung gelang es mit ArcCatalog 10 nicht FeatureClasses zuverlässig in Shapefiles zu exportieren. Möglicherweise handelte es sich dabei um Fehler in ArcCatalog oder das Shapefile-Format war nicht mit allen getesteten FeatureClasses kompatibel.

3.6.1.3 Installation und Konfiguration von PostgreSQL

Vor der Installation von ArcSDE muss PostgreSQL installiert und konfiguriert werden. Die Installationsmedien von ArcSDE liefern Installationspakete für eine kompatible PostgreSQL-Version mit. Es empfiehlt sich, die sehr konservative Ausgangskonfiguration von PostgreSQL in Bezug auf verwendeten Arbeitsspeicher nach der Installation an das physikalische System anzupassen.

Eine Installationsanleitung für PostgreSQL befindet sich im Anhang B ab Seite 81.

3.6.1.4 Installation und Konfiguration von Esri ArcSDE for PostgreSQL

Für jedes von Esri ArcSDE unterstützte DBMS gibt es eine eigenständige Version. Eine Neuinstallation von Esri ArcSDE ist also zwingend notwendig. Die Lizensierung für eine SDE-Instanz ermöglicht es, nach dem Export der Daten, die Quell-Instanz zu beenden und deren Lizenzschlüssel zum Starten der Ziel-Instanz zu verwenden. Es ist jedoch nicht klar, ob diese Verwendung von Lizenzschlüsseln durch die Lizenzbedingungen von Esri gedeckt ist. Das macht es problematisch ein Produktivsystem zu migrieren, da das Produktivsystem gestoppt werden muss, *bevor* das neue Ersatzsystem vollständig lauffähig ist. In der Regel wird daher eine zusätzliche Lizenz benötigt. Befinden sich Quell- und Ziel-SDE auf unterschiedlichen Rechnern, so kann die Lizenz technisch auf beiden verwendet werden, was jedoch gegen die Lizenzbedingungen verstößt. In den Lizenzbedingungen heißt es dazu: „Der Lizenznehmer darf zu keinem Zeitpunkt mehr Software- Lizenzen einsetzen als die Gesamtzahl der in den Unterlagen von Esri erfassten lizenzierten Konfiguration“ (vgl. [Esri, g])

Eine Installationsanleitung für ArcSDE befindet sich im Anhang B ab Seite 81.

3.6.1.5 Import der Bestandsdaten in die neue Instanz

Läuft die neue SDE-Instanz, so müssen die vorher exportierten FeatureClasses neu eingespielt werden. Dies erledigt das `dumpsde`-Skript wie folgt:

```
dumpsde.py in ./ -u sde -p sde
```

Das Skript liest alle .sdex-Dateien aus dem angegebenen Verzeichnis und importiert sie mit dem `sdeimport`-Tool von Esri in die neue SDE-Instanz. Die .sdex-Dateien müssen vorher auf einem anderen Weg auf den Rechner der neuen Instanz kopiert werden, z. B. über ein Netzlaufwerk oder per SCP/FTP. Als Name für die neuen FeatureClasses verwendet das Skript den Dateinamen ohne die Endung .sdex.

3.6.1.6 Überprüfen der Funktionalität

Vor der Migration wurde die Funktionalität des GIS festgestellt und dokumentiert (siehe Abschnitt 3.6.1.1). Nach der Migration muss anhand der vorher erstellten Dokumentation das System auf übereinstimmende Funktion geprüft werden (Funktionstest). Da das Datenbanksystem großen Einfluss auf die Performance des Gesamtsystems hat, muss sichergestellt sein, dass sich diese durch die Migration nicht verschlechtert hat (Performanztest/Lasttest[13]).

Besteht das migrierte Systeme alle Tests, kann es in den Produktivbetrieb übergehen.

3.6.2 Ablösung der ArcSDE-Software durch die ArcMap-Erweiterung

Im Folgenden wird die Ablösung einer ArcSDE-Installation durch die PostArc-Erweiterung für ArcMap beschrieben, die direkt mit der Geodatenbank PostGIS kommunizieren kann und daher nicht auf eine Middleware wie ArcSDE angewiesen ist.

Im Vorfeld der Einführung der PostArc-Erweiterung, sollte mit einem Akzeptanztest[14] unter den Anwendern geprüft werden, wie diese im Praxiseinsatz damit arbeiten. So können möglicherweise auch kritische Punkte identifiziert werden, an denen die ArcSDE zwingend vorausgesetzt wird und die gegen einer PostArc-Einführung sprechen.

[13] Für eine Einführung in Lasttests, siehe z. B. `http://swt.cs.tu-berlin.de/lehre/mwsp/ws0304/ausarbeitungen/StressLast.aus.pdf`

[14] Für eine Einführung in Akzeptanztests, siehe z. B. `http://blog.seibert-media.net/2011/05/18/akzeptanztests-scrum-agile-softwareentwicklung/`

3.6.2.1 Installation der PostArc-Erweiterung

Da die PostArc-Erweiterung noch im Status eines Prototyps ist, besitzt sie keine Installationsroutine. Die Erweiterung muss also manuell installiert werden. Sie liegt als `PostArc.dll` vor. Die DLL sollte an einem festen Ort liegen, d. h. nicht auf Netzlaufwerken oder gar in einem temporären Ordner. Mit dem RegAsm-Tool werden die von der DLL bereitgestellten Typen (vgl. Abschnitt 3.5.4.3) in der Windows-Systemregistrierung registriert:

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe -codebase PostArc.dll
```

Esri stellt auch ein eigenes `ESRIRegAsm.exe`-Werkzeug für diesen Zweck bereit. Ist die DLL im System registriert, kann man in ArcMap über das Menü

```
Customize -> Toolbars -> Customize...
```

die Buttons der Kategorie PostArc in eine Toolbar ziehen und verwenden. Analog dazu kann auch in ArcCatalog (siehe Abschnitt 2.3.1.2, S. 28) verfahren werden, auch wenn dort nicht alle Funktionen zur Verfügung stehen bzw. sinnvoll sind.

3.6.2.2 Migration der Daten von ArcSDE nach PostGIS

Nicht nur die Software muss migriert werden, auch die in der ArcSDE-Instanz enthaltenen Daten müssen in die PostGIS-Datenbank migriert werden. Erst dann können sie von der PostArc-Erweiterung verwendet werden.

Derzeit existiert noch keine automatisches oder automatisierbares Werkzeug, das eine solche Konvertierung vornehmen kann. Die Daten müssen daher derzeit über die PostArc-Erweiterung ausgetauscht werden. Die Erweiterung stellt dazu einen Dialog zur Verfügung (siehe Abbildung 3.12), mit dem sich FeatureLayer aus dem aktuellen Dokument in einer PostGIS-Tabelle exportieren lassen.

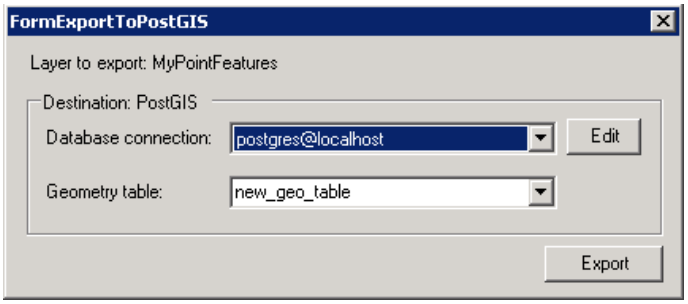

Abbildung 3.12: PostArc Feature Layer Export Dialog

3.6.2.3 Veränderung des Arbeitsverhaltens

Ein klarer Nachteil der *PostArc*-Lösung im Vergleich zu einem transparenten ArcSDE-Ersatz ist, dass der Benutzer seine Verhaltensweisen bei der Arbeit mit ArcMap verändern muss. Viele Anwender werden jedoch Fachanwendungen auf Basis von ArcMap verwenden und sind daher an den Einsatz von ArcMap-Plugins gewöhnt.

Hinzufügen oder Erstellen eines FeatureLayers Normalerweise wird ein vorhandener FeatureLayer in ArcMap aus einer in ArcCatalog eingerichteten Datenquelle hinzugefügt. Dazu muss der Benutzer folgende Schritte ausführen:

1. Rechtsklick auf das Layer-Wurzelelement im linken Fenster in ArcMap und den Eintrag *Add data* wählen **oder** auf den *Add data* Button in der Toolbar klicken
2. Im sich öffnenden Browserdialog die gewünschte Datenquelle auswählen und doppelklicken
3. Die gewünschte FeatureClass auswählen und mit *Ok* bestätigen

In PostArc müssen zum Hinzufügen einer vorhandenen FeatureClass folgende Schritte ausgeführt werden:

1. Klick auf den Button *New PostGIS Layer* in der ArcMap-Toolbar
2. Im sich öffnenden Dialog die gewünschte Datenverbindung zur PostGIS-Datenbank auswählen
3. Aus der Liste der verfügbaren Geometrietabellen eine auswählen und mit Klick auf *Ok* daraus eine neue FeatureClass/FeatureLayer erstellen

Wie man sieht, unterscheiden sich die Vorgehensweisen, sie sind jedoch von der Komplexität ähnlich. Die Verwendung von PostArc ist also ungewohnt, aber nicht komplizierter oder komplexer.

Zum Erstellen von neuen FeatureClasses auf herkömmlichem Wege, kann der Benutzer entweder ArcCatalog verwenden oder in ArcMap aus der Toolbox das Werkzeug *Create FeatureClass* verwenden.

Mit der PostArc-Erweiterung muss der Button *New PostGIS Layer* verwendet werden, der einen Dialog öffnet, über den sich eine neue FeatureClass erstellen lässt.

In diesem Fall scheint das Erstellen mit PostArc sogar einfacher als der herkömmliche Weg mit ArcMap/ArcCatalog.

Editieren von Features mit ArcMap Die Vorgehensweise zum Editieren der Features in der FeatureClass mit PostArc hat sich nicht verändert und funktioniert genauso wie ohne die Erweiterung.

3.6.2.4 Abschluss der PostArc-Migration

Die Migration ist erst dann erfolgreich, wenn durch Tests sichergestellt werden kann, dass die PostArc-Erweiterung eine vergleichbare Funktionalität wie ArcMap in Kombination mit ArcSDE fehlerfrei zur Verfügung stellt. Wie diese Tests aussehen, hängt stark vom Einsatzfeld ab. In der Regel werden Funktionstests (insbesondere von Fachanwendungen) und Performanztests durchgeführt. Bestehende MXD-Dokumente müssen angepasst werden, wenn deren Layer auf ArcSDE-Datenquellen verweisen. Die Layer müssen entfernt werden und mit Hilfe der PostArc-Erweiterung manuell wieder hinzugefügt werden. Dies kann erhebliche Zeit in Anspruch nehmen, denn ein automatisiertes Werkzeug für diese Tätigkeit steht derzeit nicht zur Verfügung.

Nicht zuletzt müssen die Anwender im Umgang und der Bedienung der PostArc-Erweiterung geschult und unterstützt werden. Nur so ist und bleibt die Migration langfristig erfolgreich.

4 Anforderungen an zukünftige freie GIS-Software

Ziel dieser Arbeit war, die Untersuchung von Möglichkeiten, vom vollständig proprietären ArcGIS-Ökosystem einzelne Komponenten durch *Freie Software* zu ersetzen.

Schwerpunkt der Untersuchung dabei war das Datenbank-Back-End, das aus der Geomiddleware ArcSDE und des darunter liegenden Datenbanksystems besteht. Das Back-End wurde als Schwerpunkt gewählt, da bei dessen Austausch die geringsten Auswirkungen auf den Endbenutzer vermutet wurden.

In einem ersten Ansatz wurde überprüft, ob als Datenbanksystem auch das freie objektrelationale Datenbankmanagementsystem PostgreSQL (vgl. 2.1.3.1) zum Einsatz kommen kann. Mit einer Testmigration wurde festgestellt, dass PostgreSQL - auch in Kombination mit ArcSDE - eine empfehlenswerte Alternative zu proprietären Datenbanksystemen wie z. B. Oracle Database oder Microsoft SQL Server ist. Schwierig ist jedoch die Migration bereits in ArcSDE versionierter Daten, da sich nur einzelne Versionen nicht aber die komplette Versionsgeschichte exportieren bzw. importieren lässt (vgl. 3.4.2).

Weiterhin wurde untersucht, ob und wie sich die Geo-Middleware ArcSDE ersetzen lässt, da diese - laut BSH (vgl. Anhang D.1) - schlecht auf modernen Multiprozessor-Systemen skaliert, hohe Lizenzkosten verursacht und in Zeiten von Geodatenbanksystemen konzeptionell nicht mehr zeitgemäß ist. Insgesamt wurden vier Ansätze untersucht, die ArcSDE zu ersetzen.

Zunächst wurde das Netzwerk-Protokoll zwischen ArcCatalog und ArcSDE auf Möglichkeiten untersucht, eine freie Middleware zu erstellen, die dieses Protokoll beherrscht. Dieser für den Nutzer transparente Ansatz (da keine Änderung am Desktop notwendig) wurde jedoch nicht weiter verfolgt, da der Zeitaufwand bis zum vollen Verständnis des undokumentierten Protokolls als zu hoch eingeschätzt wurde. Es konnte auch

nicht garantiert werden, dass ein SDE-Nachbau nach einem Releasewechsel auf eine neue ArcGIS-Version noch funktionieren würde.

Die ArcGIS-Desktop-Programme verwenden eine `sde.dll`, um mit der ArcSDE zu kommunizieren. Diese `sde.dll`-Clientbibliothek wurde prototypisch nachgebaut. Die DLL von Esri enthält nicht nur öffentlich dokumentierte Funktionen und konnte daher nicht allein mit einer öffentlichen API-Dokumentation implementiert werden. Dieser Ansatz wurde deshalb ebenfalls verworfen und nicht weiter verfolgt.

Bei diesem und dem vorherigen Ansatz konnte im Übrigen nicht ausgeschlossen werden, dass ein technisches Problem auftritt, das ohne Zugriff auf den ArcGIS-Quellcode nicht gelöst werden kann. Da dieser nicht vorliegt, erhöhte sich das Risiko für ein Scheitern der gesamten Entwicklung.

Die OLE/ODBC-Schnittstelle von ArcMap/ArcCatalog könnte theoretisch eine Möglichkeit sein, ein Datenbanksystem mit den ArcGIS-Desktop-Programmen zu verbinden. Leider bestätigt Esri, dass diese Schnittstelle über keine Option verfügt, Schreibzugriffe auf einer angebundenen Datenbank auszuführen (vgl. [Esri, a]). Damit bot dieser Ansatz keine Lösung für das Ausgangsproblem und wurde verworfen.

Abschließend wurde eine *Extension* (Erweiterung) für die Desktop-Anwendung ArcMap untersucht. Die ArcMap-Erweiterung erlaubt es dem Nutzer weitestgehend transparent und mit nur geringen Änderungen seines Arbeitsworkflows direkt mit der Geodatenbank PostGIS zu arbeiten (vgl. Abschnitt 3.6.2.3, S. 68). Der Prototyp der Erweiterung (Entwicklungsname PostArc) konnte im Rahmen dieser Arbeit jedoch nicht zur Produktionsreife gebracht werden, hier wäre weitere Entwicklungsarbeit nötig. Auch wenn der Prototyp nicht voll funktionsfähig ist, wäre es dank der dokumentierten Schnittstellen möglich, ihn mit entsprechendem Zeitaufwand fertig zu stellen. Die ArcMap-Erweiterung ist damit der tauglichste Ansatz eine ArcGIS-Infrastruktur mit Geodatenbank, aber ohne ArcSDE zu betreiben, auch wenn die Datenmigration noch aufwändig ist (vgl. Abschnitt 3.6.2.2, S. 67).

Die Erstellung von ArcGIS-Alternativen ist sehr zeitaufwändig: die ArcGIS-Programme sind in vielen Jahren Entwicklungsarbeit entstanden und weisen einen hohen Grad an Komplexität auf. Es ist unrealistisch, ein solch kompliziertes System innerhalb kürzester Zeit durch ein neues - auf freier Software basierendes - ersetzen zu wollen.

Die wichtigste Anforderung an freie GIS-Software ist die *Interoperabilität* mit der vorhandenen proprietären (ArcGIS-)Software. Mit Interoperabilität ist die Fähigkeit der

neuen Software gemeint, mit ihren alten Vorgängern zu kooperieren, z. B. wenn es um die Unterstützung von Dateiformaten und Protokollen geht. ESRIs Shapefile-Format ist beispielsweise der De-Facto Standard, wenn es um Speichern und Austauschen von Features zwischen den unterschiedlichen GIS-Software-Systemen geht. Daran wird sich auch mit der Einführung von freier Software in Teilbereichen nichts ändern, auch wenn bereits leistungsfähigere und modernere Dateiformate zur Verfügung stehen.

Sofern es also nicht gelingt, die komplette GIS-Infrastruktur abzulösen, ist es sinnvoll, Lösungen zu schaffen, die schrittweise proprietäre Software ersetzen und dabei die Kompatibilität zu bestehenden Systemen beibehalten. Die verschiedenen Ansätze dieser Arbeit befassen sich mit solchen Lösungen. Es zeigt sich deutlich, dass das Herstellen von Interoperabilität keine triviale Aufgabe ist, sondern Zeit und intensive Recherchen erfordert. In einigen Fällen ist man auf nicht-öffentliches Wissen angewiesen (Reverse Engineering) (vgl. Kapitel 3.5.1 und 3.5.2 ab Seite 44) und somit abhängig von der Kooperationsbereitschaft der Hersteller.

Anwender sollten insbesondere bei den Herstellern proprietärer Software bei der Vergabe von Entwicklungs- oder Anpassungsprojekten auf die Einhaltung von offenen Standards bestehen, denn Standards werden entwickelt, um die Interoperabilität zu fördern - zumindest in der Theorie. Besteht Kompatibilität bzw. Interoperabilität zwischen den Software-Komponenten, können Teilbereiche leichter ersetzt werden. Diese Freiheiten erhöhen den Druck auf Anbieter von Software-Komponenten, sowohl auf proprietäre als auch freie. Von dem erhöhten Wettbewerb profitiert der Anwender durch niedrigere Kosten und bessere Software.

Offene Standards (z. B. die des *Open Geospatial Consortium*[1] (OGC)) sind daher auch der zentrale Schlüssel, wenn Anwender die Gefahr des *Vendor Lock-Ins*, d. h. die Abhängigkeit an einen Hersteller, vermeiden möchten; unabhängig davon, ob die verwendete Software unter einer freien Lizenz steht oder nicht.

[1] Siehe http://www.opengeospatial.org/ und Abschnitt 2.3.2.1

In Kapitel 1 wurde die Frage aufgeworfen, welche Gründe es hat, dass sich Anwender für ArcGIS-basierte Lösungen entscheiden. Nach Gesprächen mit Kunden der Intevation GmbH (siehe Anhang D) stellt sich heraus, dass diese im Wesentlichen

- eine homogene Lösung vorziehen,
- keine Alternativen kennen,
- abhängig von ArcGIS-Fachanwendungen sind.

Zukünftiger Ansatz muss also eine einheitliche Software-Lösung aus einer Hand sein, die intensiv beworben wird und mit ArcGIS-Fachanwendungen kompatibel ist. Dienstleister für *Freie Software* produzieren und bewerben häufig nicht ein bestimmtes Produkt, sondern bieten flexible Lösungen auf Basis von dutzenden Softwareprodukten. Die von Auftraggebern geforderte homogene Lösung existiert im Umfeld des Dienstleisters also nicht. Während der Recherchen zu dieser Arbeit konnte kein einheitlicher Softwarestack gefunden werden, der das ArcGIS-Ökosystem in seiner Gesamtheit ersetzen könnte und auch noch als solcher von einem Dienstleister beworben wird.

Der Auftraggeber steht vor dem Problem, dass er sich seiner Problemstellung genau bewusst sein muss, bevor er entscheiden kann, ob der Dienstleister für *Freie Software* ihm eine geeignete Lösung bieten kann. Standardsoftware proprietärer Anbieter ist dem Auftraggeber eventuell schon bekannt; er kann also schnell entscheiden, ob das Produkt zur Lösung seines Problems beitragen kann. Eine so komplexe Software wie ArcGIS deckt viele Problemstellungen ab und wird daher im Zweifel schnell gewählt.

Ein Dienstleister für *Freie Software* könnte sich auf einzelne Produkte spezialisieren. Diese Produkte könnte er - wie ein Hersteller proprietärer Software - intensiv bewerben und gut aufeinander abstimmen. Dem Dienstleister entstehen daraus aber auch Nachteile, beispielsweise verliert er an Unabhängigkeit, da es - als gewinnorientiertes Unternehmen - in seinem Interesse sein sollte, seine eigenen Produkte bzw. Produkte, auf die er spezialisiert ist, möglichst gewinnbringend zu vertreiben. Die Unabhängigkeit von wenigen Produkten und die große Flexibilität sind aber die eigentlichen Stärken eines freien Softwaredienstleisters, die man mit dieser Strategie gefährden würde.

Große Anwender von GIS sind in Deutschland diverse Bundesbehörden und Ämter, die Softwareprodukte über (langwierige) öffentliche Ausschreibungsverfahren beschaffen müssen. Ein schneller Wechsel von Technologien ist hier häufig weder möglich noch

erwünscht. In einem solche Umfeld kann es schwierig sein, mit freier Software eine gewachsene proprietäre Infrastruktur zu ersetzen.

Ein erfolgversprechender Ansatz für den Einsatz freier Software könnte sein, zunächst Nischen mit Lösungen auf Basis von freier Software zu besetzen, z. B. in Fall einer GIS-Lösung die Visualisierung von Landkarten im Web mit der freien OpenLayers[2]-Bibliothek im Browser und PostgreSQL/PostGIS im Backend. Damit kann dem Kunden einerseits die Tauglichkeit von freier Software demonstriert und andererseits die Aufgeschlossenheit des Kunden gegenüber freier Software bei zukünftigen Projekten verbessert werden.

Die vorliegende Arbeit befasst sich fast ausschließlich mit den funktionalen Aspekten des Back-Ends des GIS-Softwarestacks. Die Performance von ArcSDE-Ersatzlösungen wurde nicht untersucht und ist ein Ansatzpunkt für weitere Untersuchungen.

Das Back-End ist ein wesentlicher Teil der GIS-Infrastruktur, dennoch wäre es ratsam, in weiteren Untersuchungen den **Desktop**-Bereich zu betrachten. Bei einer Migration von ArcMap und ArcCatalog auf freie Alternativen wie QGIS[3] oder OpenJump[4], müssen andere Faktoren mit berücksichtigt werden. Insbesondere die Arbeitsweise und Gewohnheiten der Anwender wären von einer solchen Umstellung betroffen. Diese und weitere Faktoren wie Nutzerakzeptanz oder Abhängigkeit von proprietären Drittanbieter-Plugins erschweren eine Migration und erfordern weitere Untersuchungen, die im Rahmen dieser Arbeit nicht geleistet werden.

[2]Siehe http://www.openlayers.org/
[3]Siehe http://www.qgis.org/
[4]Siehe http://www.openjump.org/

Literaturverzeichnis

[Bill 2010] Bill, Ralf ; Auflage, 5. neu b. (Hrsg.): *Grundlagen der Geo-Informationssysteme*. Wichmann Heidelberg, 2010. – ISBN 978–3–87907–489–1

[Brinkhoff 2008] Brinkhoff, Thomas: *Geodatenbanksysteme in Theorie und Praxis: Einführung in objektrelationale Geodatenbanken unter besonderer Berücksichtigung von Oracle Spacial*. Wichmann Heidelberg, 2008. – ISBN 978–3–87907–472–3

[Coverty] Coverty: *Coverity Scan: 2011 Open Source Integrity Report*.
`http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf`, Abruf: 2. März 2012

[Esri a] Esri: *16466 - What operations are not supported for OLE DB data sources or text files in ArcGIS?*
`http://support.esri.com/en/knowledgebase/techarticles/detail/16466`, Abruf: 3. Mai 2012

[Esri b] Esri: *ArcGIS and the PostGIS geometry type*.
`http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//002p0000006v000000`, Abruf: 5. Juli 2012

[Esri c] Esri: *ArcGIS Server .NET Help - OGC support in ArcGIS Server*.
`http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html#/OGC_support_in_ArcGIS_Server/009300000056000000/`, Abruf: 6. August 2012

[Esri d] Esri: *ArcGIS Server .NET Help - Verschieben einer Geodatabase mit ArcSDE-Exportdateien*.
`http://help.arcgis.com/de/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html#/na/002p00000069000000/`, Abruf: 6. Juni 2012

[Esri e] Esri: *Esri Partner Network Program Guide*.
`http://www.esri.com/library/brochures/pdfs/esri-partner-network-program.pdf`, Abruf: 14. Juni 2012

[Esri f] Esri: *Esri Products — A Complete GIS and Mapping Software System*.
`http://www.esri.com/products/index.html`, Abruf: 2. März 2012

[Esri g] Esri: *Legal Information — Licensing Terms*.
`http://www.esri.com/legal/licensing/software-license.html`, Abruf: 10. Juni 2012

[Esri h] Esri: *Über Esri*.
`http://www.esri.de/about/index.html`, Abruf: 2. März 2012

[Ethicalhacker] Ethicalhacker: *Intercepted: Windows Hacking via DLL Redirection.* `http://www.ethicalhacker.net/content/view/207/24/`, Abruf: 12. Juni 2012

[GDAL] GDAL: *ODBC RDBMS.* `http://www.gdal.org/ogr/drv_odbc.html`, Abruf: 2. Mai 2012

[GISWiki] GISWiki: *Topologie.* `http://www.giswiki.org/wiki/Topologie`, Abruf: 5. Juli 2012

[GITTA] GITTA: *Topologische Beziehungen.* `http://www.gitta.info/SpatialQueries/de/html/TopoBasedOps_learningObject1.html`, Abruf: 3. August 2012

[Grassmuck 2004] Grassmuck, Volker: *Freie Software: Zwischen Privat- und Gemeineigentum.* 2. überarbeitete Auflage. Bundeszentrale für politische Bildung Bonn, 2004. – ISBN 978–3–89331–569–7

[Levy 1994] Levy, Steven: *Hackers: Heroes of the Computer Revolution.* Penguin Books London, 1994

[Microsoft a] Microsoft: *LoadLibrary function.* `http://msdn.microsoft.com/en-us/library/windows/desktop/ms684175(v=vs.85).aspx`, Abruf: 2. März 2012

[Microsoft b] Microsoft: *Von Windows verwendeter Suchpfad zum Auffinden einer DLL.* `http://msdn.microsoft.com/de-de/library/7d83bc18.aspx`, Abruf: 2. März 2012

[Momjian 2001] Momjian, Bruce: *PostgreSQL: Introduction and Concepts.* Addison-Wesley, 2001. – ISBN 0–201–70331–9

[Netcraft] Netcraft: *July 2012 Web Server Survey.* `http://news.netcraft.com/archives/2012/07/03/july-2012-web-server-survey.html`, Abruf: 18. Juli 2012

[Obe u. Hsu 2011] Obe, Regina O. ; Hsu, Leo S.: *PostGIS in Action.* Manning Stamford, 2011. – ISBN 978–1–935182–26–9

[Open Geospatial Consortium a] Open Geospatial Consortium ; Herring, John R. (Hrsg.): *OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture.* 1.2.1

[Open Geospatial Consortium b] Open Geospatial Consortium: *OpenGIS Web Map Server Implementation Specification.* `http://www.opengeospatial.org/standards/wms`, Abruf: 31. Mai 2012

[Open Geospatial Consortium c] Open Geospatial Consortium: *Web Coverage Service.* `http://www.opengeospatial.org/standards/wcs`, Abruf: 3. Mai 2012

[Open Geospatial Consortium d] Open Geospatial Consortium: *Web Feature Service.* `http://www.opengeospatial.org/standards/wfs`, Abruf: 3. Mai 2012

[Open Geospatial Consortium e] Open Geospatial Consortium: *Web Processing Service.* http://www.opengeospatial.org/standards/wps, Abruf: 3. Mai 2012

[Oracle a] Oracle: *Database Limits.* http://docs.oracle.com/cd/E11882_01/server.112/e25513/limits.htm, Abruf: 2. März 2012

[Oracle b] Oracle: *Oracle is #1 in the RDBMS Sector for 2010.* http://www.oracle.com/us/products/database/number-one-database-069037.html, Abruf: 2. März 2012

[Oracle c] Oracle: *Oracle Technology Network Developer License Terms for Oracle Database 11g Express Edition.* http://www.oracle.com/technetwork/licenses/database-11g-express-license-459621.html, Abruf: 26. März 2012

[OSI] OSI: *Open Source Licenses by Category.* http://www.opensource.org/licenses/category, Abruf: 2. März 2012

[PostgreSQL Global Development Group a] PostgreSQL Global Development Group: *PostgreSQL: About.* http://www.postgresql.org/about/, Abruf: 2. März 2012

[PostgreSQL Global Development Group b] PostgreSQL Global Development Group: *PostgreSQL: Documentation: 9.1: Extensibility.* http://www.postgresql.org/docs/9.1/static/gist-extensibility.html, Abruf: 5. Juli 2012

[PostgreSQL Global Development Group c] PostgreSQL Global Development Group: *PostgreSQL: Documentation: Manuals: PostgreSQL 9.1.3 Documentation.* http://www.postgresql.org/docs/9.1/interactive/index.html, Abruf: 5. März 2012

[PostgreSQL Global Development Group d] PostgreSQL Global Development Group: *PostgreSQL: Professional Services - Europe.* http://www.postgresql.org/support/professional_support/europe/, Abruf: 2. März 2012

[Reiter 2004] Reiter, Bernhard E.: Wandel der IT: Mehr als 20 Jahre Freie Software. In: Sauerburger, Heinz (Hrsg.): *Open-Source-Software.* dpunkt.verlag Heidelberg, 2004. – ISBN 3–89864–291–7

[Rogers] Rogers, Howard: *PostgreSQL versus Oracle.* http://diznix.com/2010/07/27/postgresql-versus-oracle/, Abruf: 15. Juli 2012

[Stallman] Stallman, Richard M.: *Free Software, Free Society: Selected Essays of Richard M. Stallman.* 2. Auflage. GNU Press Boston

[The Governor General of Canada] The Governor General of Canada: *Order of Canada - Roger F. Tomlinson.* http://www.gg.ca/honour.aspx?id=5993&t=12&ln=Tomlinson, Abruf: 26. Mai 2012

[URISA] URISA: *GIS Hall of Fame - Roger Tomlinson.* http://urisa.org/hall/tomlinson, Abruf: 28. Mai 2012

[Vatsavai u. a. 2006] Vatsavai, Ranga ; Shekhar, Shashi ; Burk, Thomas ; Lime, Stephen: UMN-MapServer: A High-Performance, Interoperable, and Open Source Web Mapping and Geo-spatial Analysis System. Version: 2006.
http://dx.doi.org/10.1007/11863939_26. In: Raubal, Martin (Hrsg.) ; Miller, Harvey (Hrsg.) ; Frank, Andrew (Hrsg.) ; Goodchild, Michael (Hrsg.): *Geographic Information Science* Bd. 4197. Springer Berlin / Heidelberg, 2006. – ISBN 978–3–540–44526–5, 400-417

[Wikipedia] Wikipedia: *Geoinformationssystem*.
http://de.wikipedia.org/w/index.php?title=Geoinformationssystem&oldid=99494000, Abruf: 2. März 2012

A Abkürzungsverzeichnis und Begriffserklärungen

API	Application Programming Interface; die Programmschnittstelle z. B. für Erweiterungen
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
CGIS	Canadian Geographic Information System
COM	Component Object Model
DB	Datenbank
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
FOSS	Freie OpenSource Software
GIS	Geoinformationssystem
GiST	Generalized Search Tree
GUID	Globally Unique Identifier
ISO	International Standardization Organisation
RDBMS	Relationales Datenbankmanagementsystem
SDE	Spatial Data Engine
SQL	Structured Query Language; strukturierte Abfragesprache für relationale Datenbanken
SRS	Spatial Reference System (deutsch: Räumliches Bezugssystem)
TCP/IP	Kombination aus Transport Control Protocol (Transportschicht) und Internet Protocol (Vermittlungschicht); Basisprotokolle des Internets
WCS	Web Coverage Service
WFS	Web Feature Service
WKB	Well-Known-Binary
WKT	Well-Known-Text
WMS	Web Mapping Service
WPS	Web Processing Service

B Installationsanleitungen

B.1 ArcSDE mit PostgreSQL unter Linux

Diese Kurzanleitung beschreibt wie man PostgreSQL 8.3 mit ArcSDE 10 unter RHEL/CentOS-Linux 5.7 installiert und konfiguriert.

Voraussetzungen:

- RedHat-basiertes Linux-Betriebssystem
- Installationsdaten von ArcSDE 10 liegen auf dem System vor

Folgende Schritte sind zur Installation und Konfiguration dazu nötig:

- ArcSDE mit dem Installationsprogramm installieren:
  `# linux/pg/install -load` und den Anweisungen folgen
- Eventuell den bereits installierten PostgreSQL-Server entfernen:
  `# yum erase postgresql*`
- PostgreSQL 8.3 aus dem Installationsmedium installieren (Unterverzeichnis `linux/pg/`)
  Minimale PostgreSQL-Installation:

  ```
  # rpm -i postgresql-8.3.8-1PGDG.rhel5.i386.rpm
          postgresql-server-8.3.8-1PGDG.rhel5.i386.rpm \
          postgresql-libs-8.3.8-1PGDG.rhel5.i386.rpm
  ```

  Eventuell fehlende Abhängigkeiten mit `# yum whatprovides */libxy.so.3` bestimmen und nachinstallieren
- Die Shared Libary `st_geometry.so` aus den Installationsmedium in `/usr/lib/pgsql` kopieren (als root)

- Mit `# su - postgres` als postgres-Benutzer einloggen und die Datei `/.bash_profile` anpassen, z. B.

  ```
  PGDATA=/var/lib/pgsql/data
  export PGDATA
  LD_LIBRARY_PATH=/usr/lib:/opt/arcsde/sdeexe100/lib
  export LD_LIBRARY_PATH
  PATH=$PATH:/opt/arcsde/sdeexe100/bin
  export PATH
  SDEHOME=/opt/arcsde/sdeexe100
  export SDEHOME
  ```

  Neu einloggen oder die Datei *sourcen*

- Mit `initdb` den PostgreSQL Datenbank-Cluster initialisieren und danach eventuell die Werte in `pg_hba.conf` und `postgresql.conf` anpassen.

- PostgreSQL als User postgres mit `$ pg_ctl start` starten und die Installation von PostgreSQL damit abschließen.

- Das `setup_pgdb.sde` Skript aus dem ArcSDE-Installationordner ausführen und den Anweisungen folgen.

- Aus der Windows-Lizenz-Datei in

  ```
  C:\Programme\ESRI\License10.0\sysgen\keycodes
  ```

  die Zeile, die mit arcsdeserver beginnt in eine `license.dat` Datei kopieren.

- Die SDE mit dem Kommando sdesetup initialisieren:

  ```
  $ sdesetup -o install -d POSTGRESQL -s localhost -D sde -u sde \
    -l license.dat
  ```

  Parameter: `-D <Database> -u SDE-Database-User (superuser)`

- Den SDE-Dienst starten: `# sdemon -o start`

- Nun sollte die SDE z. B. über ArcCatalog erreichbar sein.

B.2 Installation von Oracle Database

Voraussetzungen:

- RedHat-basiertes Linux-System mit mindestens einem Gigabyte Arbeitsspeicher (bzw. ausreichend Swap-Speicher)
- Installationsdatei für Oracle Database 11g XE und Oracle SQL Developer bezogen von `www.oracle.com`

Installations- und Konfigurationsschritte:

- Die Installationsdatei `oracle-xe-11.2.0-1.0.x86_64.rpm` mit `# rpm -i oracle-xe-11.2.0-1.0.x86_64.rpm` installieren.
- `# /etc/init.d/oracle-xe configure` ausführen und den Anweisungen folgen.
- Die Installationsdatei `sql-developer.rpm` mit `# rpm -i sqldeveloper.rpm` installieren
- Mit `$ sqldeveloper` kann das SQL Developer Programm gestartet werden. Beim ersten Start verlangt es den Pfad einer Java-Runtime, z. B. `/usr/lib/jvm/java-1.6.0`
- Im SQL Developer kann eine Verbindung zur Datenbank auf Rechner `localhost` eingerichtet werden. Das Administratorkonto ist `sysdba` mit dem oben vergebenen Passwort.

C Kommunikationsprotokoll von ArcCatalog und ArcSDE

C.1 Allgemein

Grundsätzlich kommunizieren ArcCatalog 10.0 und ArcSDE 10.0 (mit PostgreSQL 8.3) über TCP/IP-Verbindungen. Der Standardport ist 5151, jedoch grundsätzlich konfigurierbar. Die TCP-Verbindung wird dabei häufig nicht geöffnet gehalten, nach einigen Paketen bzw. Kommandos schließt ArcCatalog die Verbindung und startet eine neue.

Numerische Werte sind sehr häufig als 4-Byte Integer in den Strukturen enthalten. Strings beginnen mit einer 4-Byte-Längenangabe, gefolgt vom eigentlichen String und anschließen aufgefüllt mit Nullen zur nächsten 4-Byte-Grenze (Padding).

C.2 Test Connection

Beim Einrichten einer neuen 'Spatial Database Connection' findet sich im Einrichtungsdialog ein Button mit der Aufschrift 'Test Connection'. Die in diesem Abschnitt beschriebene Kommunikation findet nach Betätigen dieses Buttons statt.

C.2.1 Connection 0

C.2.1.1 hello0_req

ArcCatalog sendet (`hello0_req`):

```
0000:  00 00 00 24 ff ff ff ff  00 00 00 00 00 00 00 00
0010:  00 00 00 00 00 00 00 01  00 00 00 03 73 64 65 00
0020:  00 00 00 00 00 01 5c d7  00 00 01 f5
```

`hello0_req` C-Struktur:

```
struct sde_hello0_req {
        uint32_t head;
        uint32_t pad0;        // ff ff ff ff
        uint8_t  pad1[12];    // 12 times 0x00
        uint32_t pad2;        // 00 00 00 01
        uint32_t sde_str_len; // 00 00 00 03
        char*    sde_str;     // here: 'sde'
        // Padding to 4-byte border
        uint32_t pad4;     // 00 00 00 00
        uint32_t unknown0; // 00 01 5c d7
        uint32_t unknown1; // 00 00 01 f5 or 00 00 01 f4 (on second connection)
};
```

Die Zeichenfolge 'sde' ist hier kein Passwort oder Username.

C.2.1.2 hello0_rep

ArcSDE sendet (`hello0_rep`):

```
0000:  80 00 00 14 00 00 00 0f  55 4e 49 43 4f 44 45 5f
0010:  52 45 4c 45 41 53 45 00
```

`hello0_rep` C-Struktur:

```
struct sde_hello0_rep {
        uint32_t head;   // 80 00 00 14
        uint32_t strlen; // Length of the following string
        uint8_t* str;
        // Padding to 4-byte border
};
```

Der String ist hierbei 'UNICODE_RELEASE'.

C.2.1.3 hello1_req

ArcCatalog sendet:

```
0000:  80 00 00 18 00 00 00 10  00 00 00 0f 55 4e 49 43
0010:  4f 44 45 5f 52 45 4c 45  41 53 45 00
```

`hello1_req` C-Struktur:

```
struct sde_hello1_req {
        uint32_t head;   // 80 00 00 18
        uint32_t subcmd; // 00 00 00 10
        uint32_t strlen; // Length of following string (excluding Null-terminator)
        uint8_t* str;
        // Padding to 4-byte border
};
```

Der String ist hierbei 'UNICODE_RELEASE'.

C.2.1.4 hello1_rep

ArcSDE sendet:

```
80 00 00 c0 00 00 00 40  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 0a 00 00 40 00
00 01 00 00 00 00 02 00  00 00 00 00 ff ff ff ff
00 0f 42 40 00 00 03 e8  00 00 27 10 00 00 02 00
a8 d2 5a 52 00 00 00 06  00 00 00 0a 00 00 00 00
00 00 00 00 00 06 1a 80  00 00 c3 50 00 00 00 00
00 03 20 00 00 00 00 64  00 08 64 70 ff ff ff ff
00 00 00 16 2f 6f 70 74  2f 61 72 63 73 64 65 2f
73 64 65 65 78 65 31 30  30 00 00 00 00 00 00 07
55 4e 55 53 45 44 00 00  00 00 00 05 2f 74 6d 70
```

```
00 00 00 00 00 00 00 0b  00 00 00 00 00 00 00 6f
00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 40
00 00 00 00
```

`hello1_rep` C-Struktur:

```
struct sde_hello1_rep0 {
        uint32_t head;   // 80 00 00 c0
        uint32_t unknown0; // 00 00 00 40
        uint32_t pad0[4];
        uint32_t unknown1; // 00 00 00 0a
        uint32_t unknown2; // 00 00 40 00
        uint32_t unknown3; // 00 01 00 00
        uint32_t unknown4; // 00 00 02 00
        uint32_t pad1[2];  // 00 00 00 00 ff ff ff ff
        uint32_t unknown5; // 00 0f 42 40
        uint32_t unknown6; // 00 00 03 e8
        uint32_t unknown7; // 00 00 27 10
        uint32_t unknown8; // 00 00 02 00
        uint32_t unknown9; // a8 d2 5a 52
        uint32_t unknown10;// 00 00 00 06
        uint32_t unknown11;// 00 00 00 0a
        uint32_t pad2[2];  // 00 00 00 00 00 00 00 00
        uint32_t unknown12;// 00 06 1a 80
        uint32_t unknown13;// 00 00 c3 50
        uint32_t pad3;     // 00 00 00 00
        uint32_t unknown14;// 00 03 20 00
        uint32_t unknown15;// 00 00 00 64
        uint32_t unknown16;// 00 08 64 70
        uint32_t unknown17;// ff ff ff ff
};

struct sde_hello1_rep1 {
        uint32_t sdehome_len; // Length of the following string
        char* sdehome;        // String containing the sde home path
        // Padding to 4-byte border
```

```
        uint32_t unused_len;  // Length of the following string
        char* unused;
        // Padding to 4-byte border
        uint32_t tmp_len;     // Length of the following string
        char* tmp;            // Temp path
        // Padding to 4-byte border
};

struct sde_hello1_rep2 {
        uint32_t unknown18;   // 00 00 00 0b
        uint32_t pad4;        // 00 00 00 00
        uint32_t unknown19;   // 00 00 00 6f
        uint32_t unknown20;   // 00 00 00 01
        uint32_t pad5[2];     // 00 00 00 00 00 00 00 00
        uint32_t unknown21;   // 00 00 00 40
        uint32_t pad6;        // 00 00 00 00
};
```

Der String sdehome ist hier '/opt/arcsde/sdeexe100'. Der String unused ist hier 'UNUSED'. Der String tmp ist hier '/tmp'.

C.2.2 Connection 1

Die zweite Verbindung startet mit hello0_req, hello0_rep und hello1_req, wie auch bei der ersten Verbindung. Als zweite Antwort wird jedoch ein verändertes Paket gesendet (hello1a_rep):

```
80 00 00 50 00 00 00 0a  00 00 00 00 00 00 00 00
00 00 00 33 20 66 6f 72  20 50 6f 73 74 67 72 65
53 51 4c 20 42 75 69 6c  64 20 36 38 35 20 46 72
69 20 4d 61 79 20 31 34  20 31 32 3a 30 35 3a 34
33 20 20 32 30 31 30 00  00 01 86 a1 00 00 00 00
00 00 00 01
```

C-Struktur:

```
struct sde_hello1a_rep {
        uint32_t head;        // 80 00 00 50
        uint32_t subcmd;      // 00 00 00 0a
        uint32_t unknown0;    // 00 00 00 00
        uint32_t unknown1;    // 00 00 00 00
        uint32_t str_len;     // length of the following string
        uint8_t* str;         // PostgreSQL build string
        // Padding to 4-Byte border
        uint32_t unknown2;    // 00 01 86 a1
        uint32_t unknown3;    // 00 00 00 00
        uint32_t unknown4;    // 00 00 00 01
};
```

Danach trennt ArcCatalog die Verbindung und beginnt eine neue TCP-Verbindung.

C.2.3 Connection 2

C.2.3.1 hello2_req

ArcCatalog sendet:

```
00 00 00 5c
00 00 00 02 00 00 00 05  57 69 6e 33 32 00 00 00
00 00 00 0f 76 62 6f 78  2d 73 65 72 76 65 72 32
30 30 33 00 00 00 00 01  73 00 00 00 00 00 00 01
00 00 00 03 73 64 65 00  00 00 00 19 4e 4f 5f 73
68 61 72 65 64 5f 6d 65  6d 6f 72 79 5f 63 68 61
6e 6c 5f 69 6f 00 00 00  00 01 5c d7
```

C-Struktur:

```
struct sde_hello2_req {
        uint32_t head;   // 00 00 00 5c
        uint32_t subcmd; // 00 00 00 02
        uint32_t platform_str_len; // 00 00 00 05 (without '\0')
        char*    platform_str;     // e.g. "Win32"
```

```
        // Padding to 4-byte border
        uint32_t client_name_len;
        char*    client_name;
        // Padding to 4-byte border
        uint32_t unknown0;         // 00 00 00 01
        uint32_t unknown1;         // 73 00 00 00
        uint32_t unknown2;         // 00 00 00 01
        uint32_t sde_str_len;      // without '\0'
        char*    sde_str;          // "sde"
        // Padding to 4-byte border
        uint32_t io_type_str_len;
        char*    io_type_str;      // "NO_shared_memory_chanl_io"
        // Padding to 4-byte border
        uint32_t unknown3;         // 00 01 5c d7
};
```

C.2.3.2 hello2_rep

ArcSDE sendet:

```
00 00 00 58
00 00 00 00 00 00 77 62  00 00 00 00 00 01 86 a1
ff ff ff ff 00 00 00 00  00 01 00 00 00 00 00 01
00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 1a
53 44 45 5f 53 51 4c 43  48 41 52 53 45 54 5e 31
31 20 65 73 72 69 5f 73  64 65 00 00 00 00 00 08
13 51 a1 37 6b 1f 10 8a
```

C-Struktur:

```
struct sde_hello2_rep {
        uint32_t head;     // 00 00 00 58
        uint32_t unknown0; // 00 00 00 00
        uint32_t unknown1; // 00 00 ?? ?? the last two bytes are variable
        uint32_t unknown2; // 00 00 00 00
        uint32_t unknown3; // 00 01 86 a1
```

```
        uint32_t unknown4; // ff ff ff ff
        uint32_t unknown5; // 00 00 00 00
        uint32_t unknown6; // 00 01 00 00
        uint32_t unknown7; // 00 00 00 01
        uint32_t unknown8; // 00 00 00 01
        uint32_t pad[2];   // 00 00 00 00 00 00 00 00
        uint32_t charset_str_len;
        char*    charset_str;
        // Padding to 4-byte border
        uint32_t unknown9;  // 00 00 00 08 looks like
                            // length for the following 8 bytes
        uint32_t unknown10; // 13 51 a1 37
        uint32_t unknown11; // 6b 1f 10 8a
};
```

C.2.3.3 hello3_req

ArcCatalog sendet:

```
00 00 00 58
00 00 00 02 00 00 00 05  57 69 6e 33 32 00 00 00
00 00 00 0f 76 62 6f 78  2d 73 65 72 76 65 72 32
30 30 33 00 00 00 00 0b  73 64 65 3b 3b 3b 64 65
5f 44 45 00 00 00 00 00  00 00 00 03 73 64 65 00
00 00 00 0d 78 18 cd 75  55 fe 98 4d f1 74 31 af
56 00 00 00 00 01 5c d7
```

C-Struktur:

```
struct sde_hello3_req {
        uint32_t head;         // 00 00 00 58
        uint32_t subcmd;       // 00 00 00 02
        uint32_t platform_str_len;
        char*    platform_str; // "Win32"
        // Padding to 4-byte border
        uint32_t client_name_len;
```

```
        char*    client_name;  // "vbox-server2003"
        // Padding to 4-byte border
        uint32_t locale_str_len;
        char*    locale_str;   // "sde;;;de_DE"
        // Padding to 4-byte border
        uint32_t unknown0;     // 00 00 00 00
        uint32_t sde_str_len;
        char*    sde_str;      // "sde"
        // Padding to 4-byte border
        uint32_t unknown_str_len;
        char*    unknown_str;  // garbage...
        // Padding to 4-byte border
        uint32_t unknown1;
};
```

C.2.3.4 hello3_rep

ArcSDE sendet:

```
00 00 00 34
00 00 00 00 00 00 77 72  00 00 00 00 00 00 00 00
ff ff ff ff 00 00 00 00  00 01 00 00 00 00 00 01
00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 80 00 00 54  00 00 00 0a 00 00 00 00
00 00 00 00 00 00 00 3f  20 66 6f 72 20 50 6f 73
74 67 72 65 53 51 4c 20  42 75 69 6c 64 20 36 38
35 28 50 6f 73 74 67 72  65 53 51 4c 29 20 46 72
69 20 4d 61 79 20 31 34  20 31 32 3a 30 35 3a 34
33 20 20 32 30 31 30 00  00 01 86 a1 80 00 00 04
ff ff ff ff 80 00 00 04  00 00 00 06
```

C-Struktur:

```
struct sde_hello3_rep {
        uint32_t head;        // 00 00 00 34
        uint32_t subcmd;      // 00 00 00 00
```

```
        uint16_t unknown0;    // 00 00
        uint16_t variable0;   // ?? ?? changes on every request,
                              // looks like a timestamp/uptime etc.
        uint32_t unknown1;    // 00 00 00 00
        uint32_t unknown2;    // 00 00 00 00
        uint32_t unknown3;    // ff ff ff ff
        uint32_t unknown4;    // 00 00 00 00
        uint32_t unknown5;    // 00 01 00 00
        uint32_t unknown6;    // 00 00 00 01
        uint32_t unknown7;    // 00 00 00 01
        uint32_t padding[4];  // 4x 00 00 00 00
        uint32_t unknown8;    // 80 00 00 54
        uint32_t unknown9;    // 00 00 00 0a
        uint32_t unknown10;   // 00 00 00 00
        uint32_t unknown11;   // 00 00 00 00
        uint32_t postgres_str_len;
        char*    postgres_str;
        // Padding to 4-byte border
        uint32_t unknown12;   // 00 01 86 a1
        uint32_t unknown13;   // 80 00 00 04
        uint32_t unknown14;   // ff ff ff ff
        uint32_t unknown15;   // 80 00 00 04
        uint32_t unknown16;   // 00 00 00 06
};
```

Die hello3-Pakete enthalten variable Elemente, die sich bei jeder Abfrage ändern und deren Bedeutung nicht klar ist. Der Wert entspricht nicht dem Sourceport und nicht einem Timestamp, scheint aber bis 0xFFFF hochzuzählen und dann wieder bei 0 zu beginnen. Folgende Tabelle enthält die variablen Teile einiger aufgeschnappten Kommunikation zwischen ArcCatalog und ArcSDE:

ArcCatalog	ArcSDE
78 18 cd 75 55 fe 98 4d f1 74 31 af 56	77 72
bd 6e 24 d2 b2 5c e5 84 23 a6 63 e1 57	06 b3
af 4e 04 b0 90 3a c6 65 04 87 44 c2 48	06 e2
8a 3e f3 af 8f 39 c5 64 03 86 43 c1 46	06 ec
99 3f f4 ad 8d 37 c4 63 02 85 42 c0 4f	06 fd
86 2b e0 90 70 1a af 4e ec 70 2d ab 49	07 0d
ad 5f 15 bb 9b 45 db 7a 19 9c 59 d7 54	07 17
ac 5d 13 cb ab 55 ec 8b 2a ad 6a e8 4f	07 30
98 3f f4 9e 7e 28 c0 5f fd 81 3e bc 4d	07 3a
94 44 f9 a7 87 31 c9 68 07 8a 47 c5 56	07 41
b0 51 07 b1 91 3b c1 61 ff 83 40 be 59	07 84
6e 13 c8 77 57 01 89 29 c7 4b 08 86 51	07 aa
c9 68 1e cc ac 56 ed 8e 2d b0 6d eb 54	08 f2
77 2b e0 87 67 11 a9 4a e8 6c 29 a7 4b	08 fc
72 25 da 83 63 0d 96 38 d6 5a 17 95 59	09 75
b2 53 09 b2 92 3c c9 6b 0a 8d 4a c8 4a	09 b0
88 3c f1 99 79 23 b9 5b f9 7d 3a b8 5a	09 cd
82 29 de 87 67 11 a7 49 e7 6b 28 a6 57	09 d0
88 3b f0 a9 89 33 bb 5e fc 80 3d bb 42	0a 80

Nach dem Austausch der hello3-Pakete beginnt offenbar ein Connect-Vorgang, der im nächsten Abschnitt beschrieben wird.

C.3 Connect

C.3.1 Auffälligkeiten

Jedes Paket (egal ob Anfrage/Antwort) beginnt mit folgenden Bytes (in Hex)

```
80 00 00 04 00 00 00 XX
```

Antwortpakete tragen beim XX fast immer 00, bei Anfragepaketen ist die Nummer variabel.

In vielen Anfragen sind noch mehr Bytes regelmäßig:

```
80 00 00 04 00 00 00 XX  80 00 00 XX
```

C.3.2 Nachrichtentypen

Annahme: das vierte Byte am Anfang markiert den Typ der Anfrage

Typ	**Auftreten**	**Inhalt der Anfrage (ArcCatalog)**	**Inhalt der Antwort (ArcSDE)**
D6	ArcCatalog Connect (u. a. erste Anfrage)	sde.GDB_Items Attributnamen aufgelistet, z. T. GUIDs	einige Attributname der sde.GDB_Items Tabelle: name, physicalname, properties, ...
BC	ArcCatalog Connect	Where Clause eines SQL statements (type = 1 and category = 'Address')	`80 00 00 04 00 00 00 00 80 00 00 04 00 00 00 00`
E6	ArcCatalog Connect, 2.	String `sde.sde.GDB_Items`	einige Attributname der sde.GDB_Items Tabelle: objectid, ..
36	ArcCatalog Connect	Quasi leer	Leer (80 00 00 04 00 00 00 00)
34	ArcCatalog Connect	Quasi leer	Wenige Binärdaten, manchmal XML `<DEWorkspace> .. </DEWorkspace>`
2E	ArcCatalog Connect	Quasi leer	Leer (80 00 00 04 00 00 00 00)
52	ArcCatalog Connect	Leer	Inhalt und Attribute diverser GDB_ Tabellen
1B	ArcCatalog Connect	80 00 00 04 00 00 00 02	Inhalt bzw Attribute einiger GDB_ Tabellen
4B	ArcCatalog Connect	Leer (80 00 00 04 00 00 00 4B)	80 00 00 04 4f 21 17 38
08	ArcCatalog Connect	Leer (80 00 00 08 00 00 00 08)	U.a. Koordinatensysteminformationen
07	ArcCatalog Connect	GDB_Item Shape	Ein paar Daten? mit Koordinatensysteminfos
7D	ArcCatalog Connect	Leer	Fast nix
D4	ArcCatalog Connect	Leer	80 00 00 08 00 00 00 04 ßde"00
86	ArcCatalog Connect	Leer	80 00 00 04 00 00 00 00 80 00 00 04 00 00 00 00
31	ArcCatalog Connect	SQL-Statement (SELECT 1 FROM sde.sde.SCHDATASET WHERE 1 = 0)	80 00 00 04 ff ff ff db

C.3.3 Ablauf

Folgende Anfragetypen werden gestellt:

```
D6 E6 36 34 2E 52 1B 4B 08
D6 E6 E6 07 7D 36 34 2E D4 BC
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E 86
D6 E6 E6 07 7D 36 34 2E D4 BC 86
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E 86 86
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E 31 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
D6 E6 E6 07 7D 36 34 2E
```

D Interviews mit Esri ArcGIS Anwendern

D.1 Gesprächsprotokoll des Interviews beim Bundesamt für Seeschifffahrt und Hydrographie (BSH)

Datum: 13. März 2012

Teilnehmer: Stephan Holl (Intevation GmbH), Ingo Weinzierl (Intevation GmbH), Christian Lins (Intevation GmbH) und drei Teilnehmer des BSH.

Welche ESRI Produkte werden wofür eingesetzt?

Grundsätzlich wird ArcGIS in der Version 9.3.1 eingesetzt. Verwendet werden die Produkte ArcMap, ArcCatalog und seltener ArcScene mit verschiedenen Lizenzmodellen (ArcInfo, ArcEditor und ArcView).

Es werden Erweiterungen für die Desktop-Produkte eingesetzt, u. a. Spatial, 3D und Geostatistical Analyst. Zusätzlich wird eine selbstentwickelte Erweiterung für meereskundliche Daten verwendet (ähnlich zum DataDIVER).

Auf Serverseite werden drei ArcGIS Server und drei ArcSDE Instanzen eingesetzt (jeweils Entwicklung, Produktion: Intranet, Produktion: Internet). Intern werden die REST Map Services des ArcGIS Server verwendet, nach außen hin werden WMS eingesetzt. Interner Austausch von Geodaten erfolgt entweder über die SDE oder häufiger über Shapefiles.

Wie werden die Geodaten (inkl. Metadaten) gespeichert?

Die Geodaten werden zentral in einem Datawarehouse bestehend aus ArcSDE und Oracle Database gespeichert. Metadaten werden über terraCatalog verwaltet. Oberhalb des Datawarehouse arbeiten der ArcGIS Server und die conterra SDI-Suite sowie u. a. conterra MapClient, dataDIVER und diverse OGC-Dienste.

Der schreibende Zugriff ist dabei nur ETL[1]-Prozessen gestattet, die von den Fachanwendungen angestoßen werden. Fachanwendungen sind u. a. Contis, Nauthis, MARNET, SST.

Insgesamt umfassen die gespeicherten Geodaten einen guten zweistelligen Terabyte-Bereich.

Welche Kosten fallen an?

Jährliche Kosten fallen für 17 ArcGIS-Desktop-Lizenzen an. Zusätzlich wird u. a. Caris GIS mit einem deutlichen höhreren Anteil eingesetzt.

Wird Freie Software eingesetzt?

U.a. Linux, Firefox, OpenOffice.org, H2DB, MapServer, OpenLayers. PostgreSQL und GeoServer werden derzeit evaluiert. PostGIS ist eine untersuchenswerte Alternative.

Welche problematischen Stellen gibt es in der GIS-Infrastruktur?

Hauptproblem ist die mangelhafte Performance. Ein großer Flaschenhals ist die ArcSDE, da diese nicht mit der Anzahl der Prozessorkerne skaliert (kein Parallelprocessing).

[1] Extract, Transform, Load

Welche Schwierigkeiten gibt es beim Einsatz der bestehenden Software?

Allgemein

- Starke Abhängigkeit an einen Anbieter
- Mangelhafter Support von Esri und deren Dienstleistern (u. a. kein Bugfixing)

ArcSDE

- Software veraltet (ArcSDE skaliert nicht und vom Konzept veraltet)
- Mangelhafte Performance (ArcSDE, Beispiel: Strömungskarte mit Pfeilen → ein DB-Zugriff pro Pfeil)
- Fehlende Funktionalität (keine persistenten oder materialized Views, Order By nur mit Aufwand)
- Wartung zunehmend schwierig (Indizes erfordern Handarbeit, Pflege der Datenbestände aufwändig)

ArcGIS Server

- ArcGIS Server instabil (darauf basierende Dienste müssen regelmäßig neugestartet werden)
- Fehlende Funktionalität des ArcGIS Server (z. B. WFS-Filter: es fehlt die Auswahlmöglichkeit der Tabellenspalten eines Featuretypes zur WFS-Publikation; entweder alle Spalten des Featuretypes oder keine)

Oracle

- Oracle Spatial Lizenz zu teuer (40 T€ pro Lizenz und Jahr) und nicht mit bestehenden (alten) Lizenzmodell kombinierbar

Welche Schwierigkeiten gibt es beim Einsatz von Freier Software?

PostGIS fehlt (oder fehlte bislang) die nötige Raster-Funktionalität (Rasterkataloge, Metadaten).

Warum wurde ArcGIS bei seiner Einführung gewählt?

ArcGIS wurde 1999 in der Abteilung Meereskunde eingeführt. Im Jahr 2002 wurde das Datawarehouse mit ArcSDE eingeführt. Die Lösung wurde gewählt, weil es zu dieser Zeit das meistverbreitete GIS war und das Konzept der SDE sinnvoll erschien, da vollwertige Geodatenbanksysteme noch nicht verfügbar waren.

Warum wurde als DBMS Oracle Database gewählt?

Oracle Database war die Vorgabe des BMVBS. Gründe dafür wurden nicht kommuniziert.

Soll in Zukunft mehr oder weniger Freie Software eingesetzt werden?

In Zukunft soll eher mehr *Freie Software* eingesetzt werden, um einerseits die Abhängigkeit an große Anbieter wie Esri zu reduzieren und andererseits die hohen Lizenzkosten für proprietäre Software zu vermeiden (Oracle Spatial). PostGIS ist untersuchenswerte Alternative.

D.2 Gesprächsprotokoll des Telefoninterviews mit Herrn Dr. Pressel vom Niedersächsischen Ministerium für Umwelt, Energie und Klimaschutz

Datum: 15. März 2012

Teilnehmer:
Stephan Holl (Intevation GmbH), Christian Lins (Intevation GmbH) und Dr. Holger Pressel (Niedersächsischen Ministerium für Umwelt, Energie und Klimaschutz)

Welche ESRI Produkte werden wofür eingesetzt?

Die ArcGIS-Desktopprodukte ArcMap und ArcCatalog werden nach der Migration in der Version 10 eingesetzt. Es werden 12 ArcInfo- und 59 ArcView-Lizenzen (floating licenses) verwendet (kein ArcEditor). Zusätzlich sind ein paar Single-User-Lizenzen im Einsatz. Insgesamt gibt es etwa 450 bis 500 Nutzer von ArcGIS Desktop im Hause.

Eine ArcSDE-Instanz und ein ArcGIS Server werden eingesetzt. Letzterer stellt Kartendienste im Internet bereit. Intranet-Kartendienste gibt es nicht mehr, da alle Arbeitsplätze über Internetzugang verfügen.

Wie werden die Geodaten (inkl. Metadaten) gespeichert?

Die Geodaten werden in einer ArcSDE gespeichert, die nach der Migration nicht mehr auf einer Oracle Datenbank, sondern auf dem MS SQL Server sitzt. Die Speicherung von Metadaten ist noch problematisch und noch nicht abschließend gelöst.

Insgesamt werden etwa drei bis vier Terabyte an Raster- und Vektordaten in der ArcSDE bzw. im MS SQL Server verwaltet. Die Daten werden hauptsächlich im Lesezugriff verwendet, nur bei zentralen Aktualisierungen werden die Daten verändert. Die Geofunktionalitäten von ArcSDE werden nicht verwendet, Verschneidungen o.ä. werden auf Clientseite mit den ArcGIS Desktopprodukten vorgenommen.

Wird Freie Software eingesetzt?

Freie Software wird nicht direkt eingesetzt, sondern nur indirekt über InGRID/NUMIS (Apache Webserver, Apache Lucene, MySQL, PostgreSQL, MapBender, PHP, Java).

Welche problematischen Stellen gibt es in der GIS-Infrastruktur?

Die Verwaltung von Messungen und Katastern ist verbesserungswürdig und ALK-Daten sind derzeit nicht online verfügbar.

Welche Schwierigkeiten gibt es beim Einsatz der bestehenden Software?

Kaum Probleme. Der erste Start der ArcGIS-Software ist langsam.

Welche Schwierigkeiten gibt es beim Einsatz von Freier Software?

Im Hause und beim gewählten IT-Dienstleister (HannIT) gibt es keine Erfahrung bezüglich freier Software, daher erscheint der Einsatz schwierig und nicht untersuchenswert.

Warum wurde ArcGIS bei seiner Einführung gewählt?

Marktführender Anbieter, Standardsoftware im Umweltbereich. SDE hat Performancevorteile gegenüber Fileserver-Lösungen. Seit Mitte der 90er Jahre wird Esri-Software eingesetzt.

Warum wurde als DBMS Microsoft SQL Server gewählt?

Die Lizenz für den Microsoft SQL Server ist günstiger als die 120 000 € für die Oracle Database Lizenz. Angeblich ist der SQL Server jedoch etwas langsamer als Oracle Database. Es gibt offenbar einen Trend, dass viele *kleinere* Kunden von Oracle nicht mehr gewünscht sind und eine Migration von Oracle Database weg durchführen oder planen (NLWKN genannt).

Welche Anforderungsänderungen ergeben sich für die Zukunft?

Intern wird von keiner Erhöhung der Nutzung ausgegangen. Potenzial für die Erweiterung gibt es bei den externen Onlinediensten.

Soll in Zukunft mehr oder weniger Freie Software eingesetzt werden?

Der Einsatz von freier Software wird sich auch in Zukunft nicht erhöhen.

D.3 Gesprächsprotokoll des Interviews mit Herrn Ohde vom FB Geodaten der Stadt Osnabrück

11. Juni 2012

Teilnehmer: Dirk Ohde (Stadt Osnabrück) und Christian Lins (Intevation GmbH)

Welche ESRI-Produkte werden eingesetzt?

Es werden zwei Volllizenzen, acht ArcEdit- und zehn ArcView-Lizenzen als Floating Lizenzen von *ArcGIS Desktop* eingesetzt.

Für den Einsatz im Intranet und im Internet gibt es jeweils eine Lizenz der ArcSDE und eine des ArcGIS Server. Auch ein ArcIMS wird noch verwendet.

Eingesetzt wird ArcGIS 9.3.x. Zusätzlich werden zahlreiche auf ArcGIS aufbauende Fachapplikationen eingesetzt.

Wie werden die Geodaten (inkl. Metadaten) gespeichert und verwaltet?

Als Datenbank unter der ArcSDE wird eine aktuelle Version von Microsoft SQL Server eingesetzt. Die ArcSDE-Instanz mit der darunterliegenden Datenbank ist der zentrale Datenspeicher, auch wenn nicht ausgeschlossen werden kann, dass einzelne Mitarbeiter noch lokale Daten in Form von Dateien vorhalten. Zielrichtung ist jedoch alles in der ArcSDE zu speichern.

Die ArcSDE speichert alle Arten von Daten, darunter Vektor-, Raster- und Metadaten, insgesamt etwa 30-40 GiB.

Beschreibung der Struktur

Für die einzelnen ArcGIS Server-, ArcSDE- und Datenbank-Instanzen werden jeweils dedizierte Rechner eingesetzt. Die ArcSDE-Server verfügen jeweils über 64 GiB Arbeitsspeicher. Insgesamt werden etwa 10-15 Server (inkl. Lizenzserver) im Verbund eingesetzt, um 40-45 GIS-Arbeitsplätze zu versorgen.

Als Beispiel für die Belastung der Server wurde die Internet-Stadtplan-Anwendung genannt, mit der monatlich etwa 10.000 bis 15.000 Personen arbeiten.

Welche jährlichen Kosten fallen an?

Jährlich fallen Lizenzkosten in Höhe von etwa 80 T€ an. Darin enthalten sind Kosten für die auf ArcGIS aufbauenden Fachapplikationen.

Wird *Freie Software* eingesetzt?

Es wird keine *Freie Software* eingesetzt.

Warum werden Esri-Produkte eingesetzt?

Bei der Ablösung von SICAD um das Jahr 2000 waren die Esri-Produkte innovativ und verwendeten das Konzept der zentralen Datenhaltung über die ArcSDE. Ziel war es neben der zentralen Datenhaltung „GIS an die Arbeitsplätze zu bringen“.

Warum wird keine Freie Software eingesetzt?

Bei der Einführung der Esri-basierten Geodateninfrastruktur war Freie Software noch nicht ausgereift genug.

Problematisch sind die Dienstleister von freier Software, die oftmals zu wenig (qualifizierte) Resourcen haben, um eine Fachanwendung für ein spezialisiertes Problem umzusetzen. Daher wird bevorzugt auf Standardsoftware gesetzt, auch da eine Spezialentwicklung die Abhängigkeit an den beauftragten Softwaredienstleister erhöht.

Warum wird Microsoft SQL Server eingesetzt?

War bei Einführung die günstigste Alternative.

Welche problematischen Stellen gibt es in der GIS-Infrastruktur oder beim Einsatz von Esri-Software?

Keine großen Schwierigkeiten. ArcIMS ist veraltet.

Soll in Zukunft mehr/weniger *Freie Software* im Geobereich eingesetzt werden?

Keine wesentlichen Änderungen geplant.