Technische Universität Hamburg-Harburg
Arbeitsbereich Softwaresysteme

Integration von standardisierten Geodatendiensten in Google Earth mit Hilfe des GIS-Frameworks deegree

Studienarbeit im Studiengang Informatik-Ingenieurwesen

von
Mario Heidmann
Matrikelnr.: 22062

Hamburg, 2. Januar 2007
betreut von
Prof. Dr. Joachim W. Schmidt
Dipl.-Inform. Olaf Bauer

Ehrenwörtliche Erklärung: Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde weder einer anderen öffentlichen oder privaten Institution vorgelegt noch veröffentlicht.

Mario Heidmann
Hamburg, den 02. Januar 2007

Inhaltsverzeichnis

1 Einführung

Durch die Vormachtstellung Googles im Angebot von Suchdienstleistungen erziehlen auch die Geodatendienste von Google eine große Reichweite. Das Programm Google Earth erfreut sich einer großen Beliebtheit bei vielen Internetnutzern. Inzwischen haben auch andere IT-Dienstleister die Bedeutung von Geodatendiensten erkannt und weitere Produkte auf dem Markt platziert. Als Beispiele für Kartendienste sind hier insbesondere Yahoo Maps, Microsoft Live Search Maps und NASA World Wind zu erwähnen. Neben dem Planetenbrowser Google Earth wird von Google auch der webbrowserbasierte Dienst Google Maps angeboten, dessen Kartendarstellung in Webseiten eingebunden werden kann. In Kombination mit der Suchmaschinenfunktion werden Kunden Dienstleistungen in deren unmittelbarer Umgebung angeboten (*Location-Based Services*).

Gleichzeitig wurden im professionellen Anwendungsbereich angesichts der Verbreitung von proprietären Geografischen Informationssystemen (GIS) offene Schnittstellen definiert, um die Interoperabilität von GIS-Anwendungen zu ermöglichen. Die Anbindung des Google Earth Clients an diese offenen, standardisierten Schnittstellen macht die Applikation verwendbar für alternative Geodatenquellen.

1.1 Motivation

Die Kombination von intuitivem Umgang mit einem handlichen Globus und auflösungsstarken Satellitenbildern in Verbindung mit dem Bekanntheitsgrad von Google hat zu einer weiten Verbreitung von Google Earth auch bei nicht professionellen Anwendern geführt. Durch die Verwendung proprietärer Anwendungen und restriktiver Linzenzierungsbestimungen sowie unzureichend dokumentierter Schnittstellen ist die Verwendung der Google-Dienste für alternative GIS-Applikationen nur eingeschränkt möglich. Der Google Earth Client bietet als Schnittstelle lediglich das proprietäre XML-Format KML an, womit Geodaten in Google Earth eingebunden und dargestellt werden können. Weiterhin werden immer mehr Geodatenbestände über OGC-konforme Schnittstellen via Internet zu Geodateninfrastrukturen (GDI) verknüpft und für neue Dienstleistungen verwendet. Die Vorteile des Google Earth Clients - hoher Verbreitungsgrad und intuitive Bedienung - für solche Dienste nutzbar zu machen, ist die Anbindung von standardisierten OGC-Diensten notwendig.

1.2 Aufgabenstellung

Diese Studienarbeit befasst sich mit der Erweiterung einer offenen, standardisierten Schnittstelle an den Google Earth Client. In dieser Arbeit werden die Möglichkeiten der Anbingung ermittelt und eine prototypische Implementierung am Beispiel des *Web Feature Service* des Open Geospatial Consortiums realisiert. Dazu wird als Grundlage das Java Open Source Framework deegree[1] verwendet.

2 Geografische Web-Dienste und Anwendungen

2.1 Web-Dienste

Eine im Internet weit verbreitete Anwendung finden die Webservices. Webservices bieten über standardisierte Internetprotokolle (z.B. HTTP) Dienste an, die von überall auf der Welt aus dem Internet abgefragt werden können. Hinterlegt sind die Webservices hinter einer URI, wohinter eine Schnittstelle für Anfragen bereitgestellt wird. Eine XML-basierte Syntax wird für Anfragen und Ergebnisse verwendet, wobei die Grammatik eindeutig definiert sein muss. Auf diese Weise können auch Intenetbrowser als Client-Interface dienen und auf Webservices zurückgreifen..

2.2 Geografische Informationssysteme

Speziell für die Anwendung in der Geografie wurden Geografische Informationssysteme (GIS) entwickelt. Sie erlauben einerseits die Visualisierung von Kartenmaterial, zum Anderen aber auch die Verwaltung und Pflege von raumbezogenen Attributen. Durch die Verwendung von Vektordaten (geometrische Grundfiguren wie Linien oder Polygone) und Rasterdaten (z.B. Bilddaten als JPG-Dateien) wird eine Kartendarstellung ermöglicht.

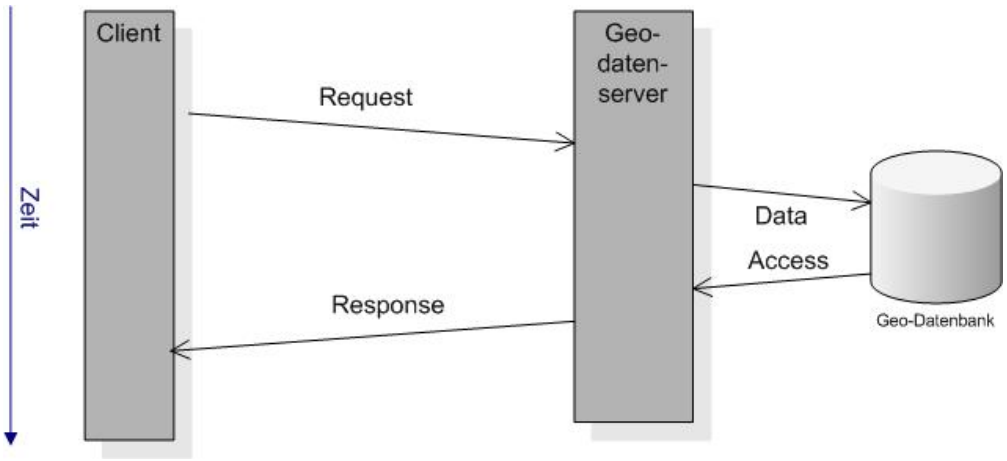

Abbildung 1: Kommunikation mit einem GIS-Server

2.3 Open Geospatial Consortium (OGC)

Viele Firmen bieten Lösungen für Geografische Informationssysteme an, die meist eigene, proprietäre Ansätze verfolgen, um ihre Systeme zu realisieren. Um auch anderen Anbietern eine Anbindung an das eigene System zu erlauben, haben sich mittlerweile 338 Firmen zum Opengeospatial Consortium zusammengeschlossen, mit dem Ziel, offene Standards für

Geodatendienste zu spezifizeren, um die Interoperabilität von GIS zu ermöglichen. Viele dieser Standards sind bereits in Verwendung und finden eine weite Verbreitung, von der auch Open-Source-Projekte profitieren.

2.3.1 Geography Markup Language (GML)

Die Geography Markup Language[7] ist eine durch das OGC, mittlerweile in der Version 3.1.1, spezifizierte, XML basierende Sprache. GML dient als Metamodell für die Festlegung der Datenformate und räumlichen Datentypen einer spezifischen Geodatenanwendung. Ebenso legt GML die Grammatik für den Aufbau von Raumdaten-Dokumenten fest. Auf diese Weise lassen sich geografische Informationen modellieren und auch der Austausch von Geodaten zwischen verschiedenen Applikationen kann hierdurch erfolgen. Durch Schnittstellen zum Im- und Export von Geodaten mittels GML-Dateien sind proprietäre Anwendungen in der Lage, miteinander zu kommunizieren und Datenquellen anderer Anbieter für sich nutzbar zu machen. Die Heterogenität der Anwendungen kann also durch die Verwendung von GML als Austauschformat überwunden werden. GML liegt ein abstaktes Datenmodell zugrunde, das durch XML-Schema Dateien spezifiziert ist. Geografische Objekte werden als *Features* bezeichnet, deren Struktur mit Hilfe von *FeatureTypes* definiert wird. Zusammengefasst werden Features zu einer *FeatureCollection*, die das Wurzelelement in GML bildet. Ein Feature setzt sich zusammen aus:

- einer ID
- Beschreibungen (Sachdaten) des Features
- Geometrien (Vektordaten)

2.3.2 Web Map Service (WMS)

Einer der vom OGC spezifizierten Dienste ist der Web Map Service. Ihm fällt die Aufgabe zu, Kartenmaterial im Internet zu publizieren. Clients können mittels *GetMap* Anfragen nach bestimmten Regionen oder Orten an den WMS stellen, der dann das gewünschte Material in Form von gerasterten, georeferenzierten Karten zurückgibt. Grundlegend ist die Bereitstellung eines *Capabilities-Dokuments*. Mit dem Aufruf der Operation *GetCapabilities* erhält der Client durch dieses XML-Dokument einen Überblick über alle angebotenen Kartendarstellungen. Informationen bezüglich der Rechteinhaber des Materials, der verfügbaren Kartenlayer und den möglichen Ausgabeformaten sind dort hinterlegt.

2.3.3 Web Feature Service (WFS)

Ein weiterer OGC-Standard für interoperable Geografischen Informationssystem ist der *Web Feature Service*[4]. Durch seine im Internet bereitgestellten Dienste erhalten Clients die Möglichkeit, Geodaten (*Features*) in Form von mit Sachdaten verknüpften georeferenzierten

Vektordatensätzen abzufragen. Die Struktur eines *Features* ist bestimmt durch die räumlichen und nicht räumlichen Informationen, die es repräsentiert. In der Grundversion stellt ein Web Feature Service den Clients drei Operationen zur Verfügung, mit denen Informationen abgefragt werden können:

- *GetCapabilities*
- *DescribeFeatureType*
- *GetFeature*

Über die parameterlose Abfrage *GetCapabilities* erhält der Client das auf XML-Syntax basierende *Capabilities* Dokument, worin die abfragbaren Informationen und Parameter verzeichnet sind. Es enthält Metainformationen über die Fähigkeiten des WFS-Servers und über die Geodaten. Unter anderem sind Metadaten zu den möglichen Ausgabeformaten, sowie zu den verwendeten Koordinatensystemen verzeichnet. Ebenso kann das *Capabilities* Dokument auch Copyright-Informationen zum Verwendeten Material enthalten. Bei der Integration der Geodaten in einen WFS-Dienst muss dieses Dokument erstellt werden. *DescribeFeatureType* wird verwendet, um die Struktur der Featuredaten abzufragen. Als zusätzlicher Parameter kann ein bekannter *FeatureType* angegeben werden. Fragt der Client mit der Operation *GetFeature* nach angebotenen Features, so erhält er vom WFS ein GML-Dokument, in dem alle Features aufgelistet sind. Vom Server wird dabei ein XML-Schemadokument zurückgegeben, das alle definierten Elemente in dem GML-Featuredokument beschreibt. Zur Integration von Daten in einem WFS ist die Definition der Datenstruktur durch geeignete XML-Schemadokumente erforderlich.

2.4 Das Deegree-Framework

Es gibt viele frei verfügbare Applikationen, welche OGC-konforme Geodatendienste in Netzwerken realisieren. Das Deegree-Framework enthält u.a. die OGC-Referenzimplementierung des WMS und bietet eine umfassende Implementation weiterer OGC-Dienste (WFS, WCS, etc). Komplett in Java programmiert ist das Framework unter der GNU LGPL frei verfügbar und kann als Web-Applikation in einen Server eingebunden werden, der Webanwendungen unter Java unterstützt. Als Datenquellen können für den WFS u.a. SQL-Datenbanken, die mit räumlichen Geodaten arbeiten können, verwendet werden.

2.5 Google Earth

Google Earth ist ein als PLANETENBROWSER bekanntes Programm, mit dem sich jeder beliebige Ort der Erde aus der Vogelperspektive interaktiv darstellen lässt. Dazu wird googleeigenes Kartenmaterial verwendet, das von einem Server geladen wird, um dann mit der Client-Software für die gewünschte Darstellung gerendert zu werden. Die Ursprünge von Google Earth liegen im EARTH VIEWER von der Firma KEYHOLE, die im Jahre 2004 von Google aufgekauft wurde. Seitdem führt Goolge die Entwicklung fort und hat inzwischen mehrere Releases der Google Earth Software veröffentlicht. Neben der kostenlosen Version für Privatanwender sind auch Versionen für den professionellen Einsatz zu haben, die u.a. auch Datenimport aus anderen GIS-Anwendungen erlauben.

2.5.1 Die Applikation Google Earth

Google Earth ist als Client-Server-Anwendung realisiert, wobei der Anwender die Clientsoftware starten muss, um auf den von Google betriebenen Server zuzugreifen. Durch Steuerung mit Maus und Tastatur kann der Anwender die jeweilige Ansicht auf die Kartendarstellung der Erde ändern

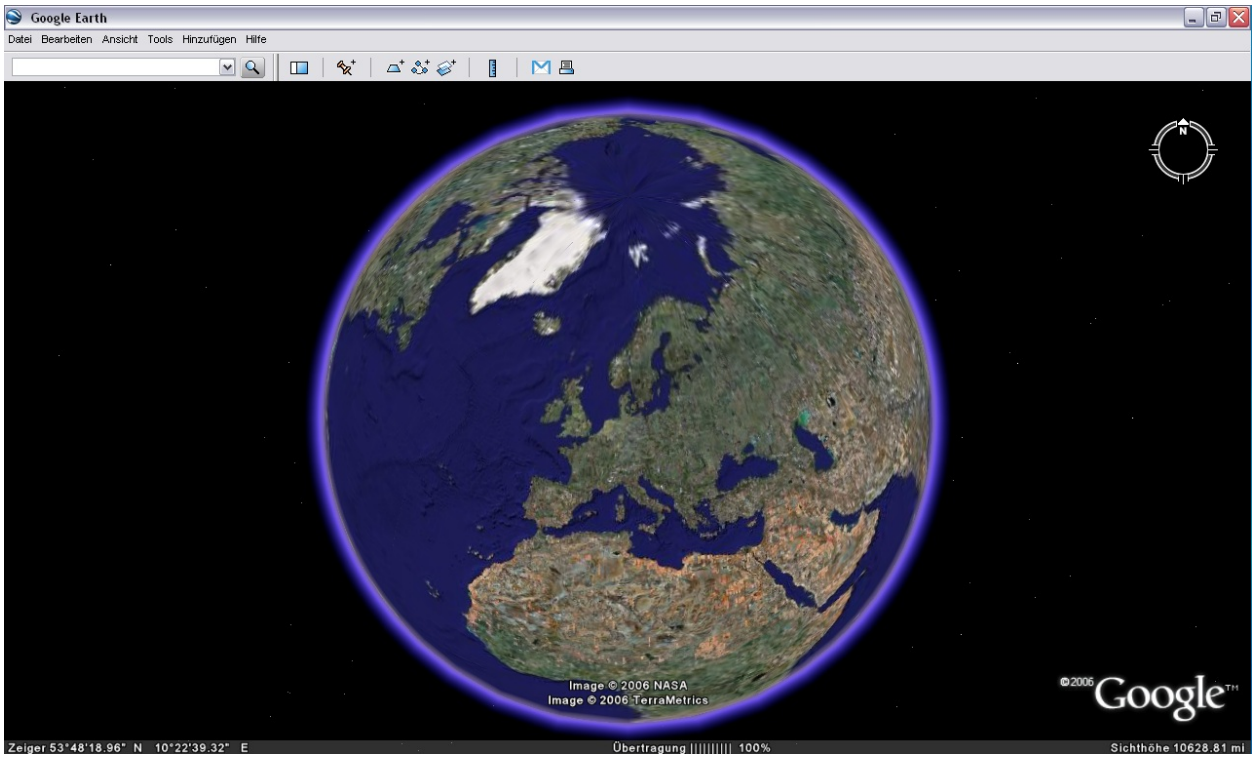

Abbildung 2: Google Earth Client

Hauptansicht im Applikationsfenster ist ein virtueller Globus, der sich durch Steuerung frei über alle Achsen rotieren lässt. Dabei sind Darstellungsparameter wie Zoomfaktor, Nei-

gung und Rotation beliebig einstellbar. Das benötigte Kartenmaterial bezieht der Client kontinuierlich vom Server. Je nach Region ist dabei ein hoch bzw. weniger hochauflösendes Kartenbild verfügbar. Benutzer können sogenannte *Placemarks* mit dem Google Earth Client erstellen. Dabei handelt es sich um ortsfeste Wegpunkte, die mit einem Symbol auf der Karte gekennzeichnet werden. Beim Doppelklick auf ein Placemark zoomt Google Earth automatisch auf die gekennzeichnte Position und zeigt in einem sogenannten *Baloon* einen Beschreibungstext an. Bei Speicherung werden die Placemarks als KML-Dateien auf dem lokalen Computer hinterlegt. Neben den Placemarks bietet Google Earth auch die Möglichkeit, weitere Layer in die Darstellung zu integrieren. Auf diese Weise können Vektorelementte, zum Beispiel Straßen, Sehenswürdigkeiten und Gebäude, in die Kartendarstellung eingeblendet werden. Google Earth unterstützt das Erstellen weiterer Layer erst ab der Professional-Version; jedoch kann dies auch durch manuelles Anfertigen von KML-Dateien realisiert werden. Auch das Einblenden von Rasterdaten (*Image Layer*) auf die Kartendarstellung ist möglich.

2.5.2 Keyhole Markup Language

Das Integrieren von benutzerspezischen Erweiterungen in Google Earth wird durch die Keyhole Markup Language[3] (KML) realisiert. Mit Hilfe dieser auf XML basierenden Sprache lassen sich eigene Layer aus Vektor- oder Rasterdaten in Google Earth abbilden und mit Texten versehen. Die KML-Dokumente erlauben auch das Erstellen eigener Ansichten auf den Google-Globus, sowie die Anpassung von Stilelementen der Google Earth Client Applikation. KML wird nicht durch ein offizielles XML-Schema Dokument spezifizert, eine kurze Dokumentation ist im Internet zu finden, in der die anwendbaren Elemente und ihre Funktion beschrieben sind. Grundlegend für KML-Dokumente ist der hierarchische Aufbau, der der Funktionalität der Google Earth Applikation nachempfunden ist. Grundelement einer KML-Datei ist das Wurzel-Element `<kml>`. Darunterliegende Hierarchien werden durch `<Document>`- und `<Folder>`-Tags gebildet, die als Container für eigene Layer dienen und Stilanweisungen zusammenfassen können. Ein Layer kann durch ein `<Placemark>`-Tag repräsentiert werden. In einem KML-Dokument bilden `<Placemark>`-Elemente die Basis, da sie zur Speicherung von Geodaten genutzt werden. In KML-Dokumenten sind Geodaten und ihre Darstellung enthalten. Diese wird durch die Verwendung von `<Style>`-Elementen realisiert, die entweder global für das ganze Dokument oder aber lokal für einzelne Bereiche definiert werden können. Das zweite Element, das zur Gliederung von KML-Dateien verwendet werden kann ist das Element `<Folder>`. Mit hilfe von *Foldern* lassen sich Ordner erstellen, welche in der Google Earth Applikation in der Seitenleiste erscheinen und dort die enthaltenen Elemente anzeigen. Eine hierarchische Gliederung von Folder-Elementen ist ebenfalls möglich. Am folgenden Beispiel ist die Verwendung von Folder-Elementen ersichtlich:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
<Folder>
  <name>Special Places</name>
  <open>1</open>
  <description>
   Dieser Ordner enth"alt einen einzigen Ort.
  </description>
  <Placemark>
    <name>Bad Oldesloe</name>
    <description>
     Kleiner Ort n"ordlich von Hamburg.
    </description>
    <Point>
      <coordinates>10.377588,53.805266,0</coordinates>
    </Point>
  </Placemark>
</Folder>
</kml>
```

Abbildung 3: Folder-Darstellung in Google Earth

Beim Aufruf des KML-Dokuments wird der Ordner mitsamt der Beschreibung in der Seitenleiste angezeigt (vgl. Abbildung 3). In ihm enthalten ist ein Placemark, das mit einem Namen und einer Beschreibung versehen ist. Desweiteren enthält das Placemark eine Geometrie, die hier durch ein `<Point>`-Element repräsentiert ist. Der `<Point>` ist wiederum durch zwei Koordinaten definiert, die den Längen- und den Breitengrad auf dem Google-Globus angeben. Die Darstellung des `<Placemark>`-Elements auf der Google-Karte ist in Abbildung 4 ersichtlich.

Geometrien sind als Vektordaten definiert. Dabei sind derzeit folgende Strukturen definierbar, welche die Eigenschaften vom abstrakten Datentyp *Geometry* erben:

Abbildung 4: Darstellung eines Placemark-Elements

- Point
- LineString
- LinearRing
- Polygon
- MultiGeometry

MultiGeometry ist dabei kein einfacher Vektordatentyp. Diese Struktur dient als Container für andere Geometrien, die dadurch zu einer Einheit zusammengefügt werden können. Jeder Vektordatentyp erhält seine geografische Beschreiibung durch die Angabe von Koordinaten. In KML erfolgt die Angabe der geografischen Koordinaten in dezimalen Längen- und Breitengradangaben und unterstützt keine weiteren Koordinatenreferenzsysteme. Das verwendete System ist auch als World Geodetic System 1984 (kurz: WGS84) bekannt und wird ebenso in der Luftfahrt und in GPS-Systemen verwendet. Es existieren einige Besonderheiten bei der Verwendung von Geometrien: KML erlaubt eine dreidimensionale Projektion der vormals zwei-dimensionalen Vektordaten. Dabei kann einerseits die Geometrie selbst über der Karte “schwebend” dargestellt werden, oder die Vektordaten werden in die dritte Dimension ausgedehnt. Durch Kombination mehrerer Vektordaten lassen sich auf diese Weise auch Gebäudeumrisse modellieren, wie in Abbildung 5 ersichtlich ist.

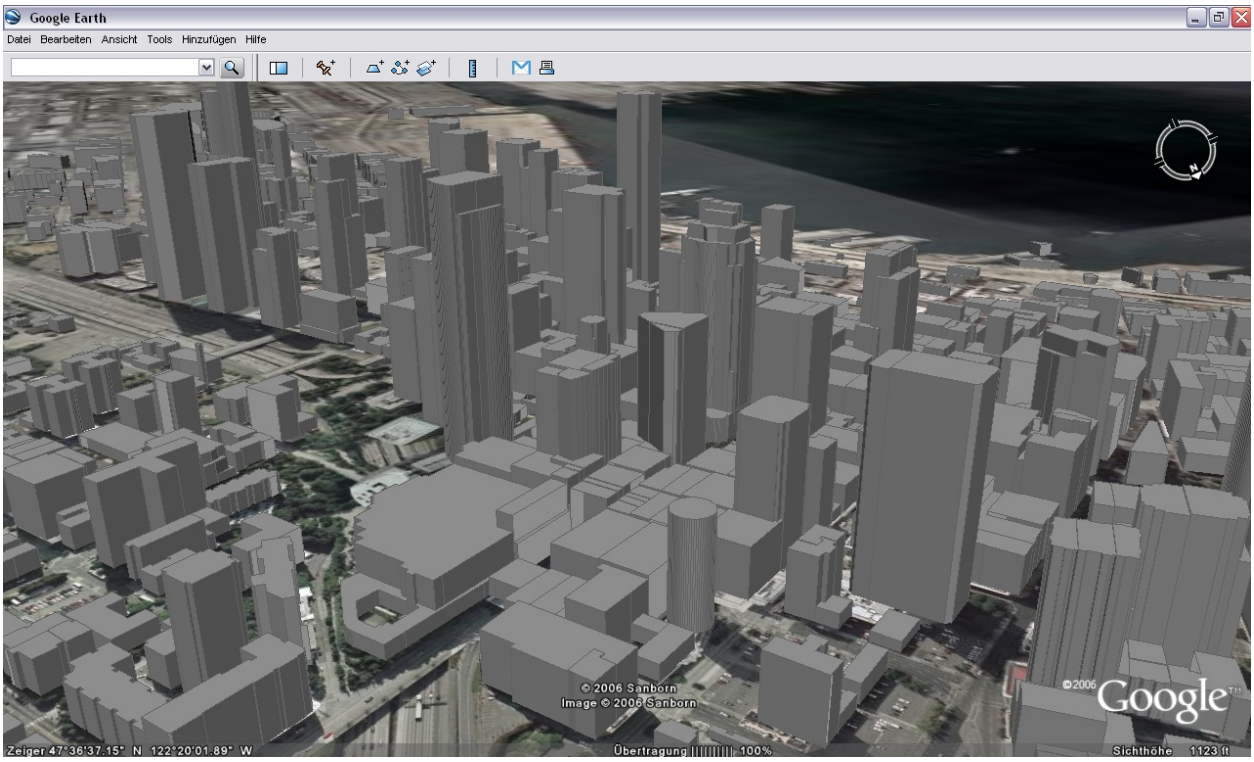

Abbildung 5: Dreidimensionale Gebäude aus Vektordaten

Informationen über die Darstellung sind in `<Style>`-Bereichen definiert. *Styles* können auf Geometrien sowie auf Beschriftungen, die in `<Description>`-Tags enthalten sind, angewandt werden. Dabei handelt es sich um die farbliche Gestaltung der Elemente, sowie deren Darstellungsgröße. Farben werden in einem achtstelligen hexadezimalen Schema angegeben, in dem die Blau-, Grün-, und Rotwerte jeweils durch zwei Hex-Stellen gekennzeichnet sind. Die letzten beiden Stellen im Farbwert beeinflussen die Transparenz. Je niedriger der Wert, desto höher ist der Grad der Transparenz. Ein Beispiel für die Festlegung von Styles kann folgendem XML-Beispiel entnommen werden:

```
<Style id="myNewStyles">
  <IconStyle>
    <color>a2ff00ff</color>
    <scale>1.499999976158142</scale>
    <Icon>
      <href>http://www.sts.tu-harburg.de/icon.jpg</href>
    </Icon>
  </IconStyle>
  <LabelStyle>
    <color>5fffaaff</color>
    <scale>1.8</scale>
  </LabelStyle>
```

```
    <LineStyle>
      <color>fff000ff</color>
      <width>18</width>
    </LineStyle>
    <PolyStyle>
      <color>5f7faaaa</color>
      <colorMode>random</colorMode>
    </PolyStyle>
  </Style>
```

Jedes KML-Element, für das sich Styles definieren lassen, besitzt ein entsprechendes Style-Tag. Neben den bereits erwähnten Farbeinstellungen innerhalb der `<color>`-Elemente, kann eine Skalierung von Bildelementen vorgenommen werden. Durch einen Faktor können auf diese Weise Bilder und Symbole vergrößert oder verkleinert werden. Eine Möglichkeit, die Darstellung von Linien anzupassen ist die Angabe der Dicke im Element `<width>`. Innerhalb von `<Description>`-Tags ist die Angabe von HTML-Elementen möglich. Diese müssen in *CDATA-Sections* gekapselt sein, damit keine Beeinflussung des restlichen KML-Dokumentes erfolgt.

3 Anforderungen und Entwurf für eine Erweiterung des OGC-WFS

Um eine Anbindung von offenen Geodatendiensten an Google Earth zu ermöglichen, müssen die Schnittstellen zu beiden Systemen bekannt sein. Offene Systeme, die durch das OGC spezifiziert sind, sind hinreichend dokumentiert und durch Referenzimplementierungen realisiert. Als Schnittstelle zum Export von Geodaten dient die Sprache GML, die sich insbesondere durch ihre strukturierte Form für diese Zwecke eignet. Die verfügbare Schnittstelle zu Google Earth wird durch KML repräsentiert. Zwar erlaubt auch diese Sprache die Definition von Geodaten, stößt aber im Vergleich zu GML bei komplexeren Strukturen schnell an ihre Grenzen.
Der Geodatendienst WFS arbeitet mit GML als Ausgabeformat. Um eine Anbindung eines OGC-Datendienstes an Google Earth zu realisieren, muss ein Adapter implementiert werden, der GML in KML übersetzt. Dazu müssen die Gemeinsamkeiten beider XML-Grammatiken herausgearbeitet werden und eine geeignete Technologie und Implementierung gefunden werden, um eine Übersetzung zu ermöglichen. Ein weiterer Aspekt ist die Notwendigkeit der Visualisierung der Geodaten in Google Earth. Dazu ist eine sinnvolle Vorgabe für eine Darstellung in Google Earth zu finden, welche die Features auf eine anwenderfreundliche Weise visualisieren. Diese Darstellung bezieht sich auf die Geometrien und auf die Beschreibungsdaten der Features.

3.1 Die Funktionsweise des Deegree-WFS

Einer Deegree basierte Webanwendung hat im Allgemeinen folgenden Ablauf: Ein Deegree-*Servlet* nimmt die *Requests* des Web-*Clients* über HTTP entgegen und wertet sie aus. Dabei wird das Format und die Art des Requests festgestellt. Anfragen sind hierbei XML über die HTTP POST-Methode oder *Key-Value-Pairs* über die HTTP GET-Methode mit aneinander gereihten Parametern. Danach wird intern ein *Handler* aufgerufen, der die weitere Bearbeitung des Requests übernimmt. Im Falle vom WFS steuert der *WFSHandler* die weitere Analyse der Anfrage und sorgt für die Bereitstellung der angeforderten Daten, die aus der Datenbank abgefragt werden. Die Ausgabe der Informationen an den Client erfolgt ebenfalls durch den Handler, der die Daten im gewünschten Ausgabeformat versendet.

3.2 Design einer KML-basierten Schnittstelle

Die Anbindung des KML-Adapters erfolgt an den WFS-Dienst von Deegree. Im Gegensatz zum *Web Map Service* beinhaltet ein *Web Feature Service* keine Visualisierungsvorschrift der Features. Da Google Earth jedoch mit der Darstellung der Geodaten betraut wird, muss eine geeignete Darstellungsvorschrift gefunden werden. Dies kann durch die Definition vorgegebener Styles erfolgen, die im Übersetzungsprogramm integriert werden.

Ein direkter Vergleich von GML und KML liefert Übereinstimmung in der Art der Definition von Vektordaten. Dies erleichtert die weitere Vorgehensweise, da für GML-Vektordaten keine umständliche Umschreibung in KML erforderlich ist. Elemente wie *Point*, *LineString*, und *Polygon* können nahezu direkt von GML nach KML übernommen werden. Für Mehrfachelemente wie *MultiPoint* oder *MultiPolygon*, die in GML spezifiziert sind, existiert in KML das abstrakte Mehrfachelement *MultiGeometry*, das für die Umsetzung geeignet ist.

Während sich die Vektordaten nahezu direkt von GML in KML übertragen lassen, muss für die nichträumlichen Beschreibungsdaten einer GML-Datei eine geeignete Darstellungsbeschreibung in KML gefunden werden.

Da für jede Featuretype ein Schema definiert sein muss, das den Aufbau der Features beschreibt, sind in der GML-Datei Elemente definiert, die auf das jeweilige Feature zugeschnitten sind. Dadurch entsteht eine Sinn gebende Struktur. Beispielsweise können Länder und Städte jeweils eine Einwohnerzahl und eine Fläche als Feature-Element definiert haben, hingegen sind als Feature erzeugten Personen ein Name, ein Geburtsdatum und als Raumdatum einem Wohnort zugeordnet. Features bilden somit die "reale Welt" ab und enthalten je nach Anwendung eine Beschreibung, die zu dem jeweils definierten Feature passt.

Die Definition von FeatureTypes ist in KML nicht übertragbar, da die Definition und Verknüpfung eigendefinierter Elemente stark limitiert ist. Zwar unterstützt KML den Entwurf benutzerspezifischer `<Schema>`-Elemente, jedoch dürfen die darin definierten Attribute nur einen elementaren Datentyp besitzen und nicht auf komplexere Strukturen verweisen. Als weiteres Problem ergibt sich, dass das einzig mögliche *Parent-Element* von einem `<Schema>` vom Typ `<Placemark>` sein muss. Daraus resultiert die Notwendigkeit der Verwendung einer flachen Struktur zur Beschreibung der Sachdaten der Features, sodass deren Typinformationen verloren gehen. Als reine Textelemente lassen sich die Beschreibungsdaten in den `<Description>`-Tags unterbringen, die in vielen KML-Elementtypen enthalten sind; so auch in `<Placemark>`. Trotzdem eignen sich Placemarks als Repräsentanten von Features. Zum einen enthalten Placemarks eine Geometrie in Form von Vektordaten, zum anderen besitzen sie auch ein `<Description>`-Element, in das die nichtraumbezogenen Daten eingefasst werden können. Weiterhin ist das Zusammenfassen mehrerer Placemarks in einem KML-Dokument durch Verwendung von Containern möglich. Wie bereits im Abschnitt 2.4.2 erwähnt, lassen sich mehrere Placemarks in `<Document>` oder `<Folder>` integrieren. Abbildung 6 verdeutlicht die Umwandlung eines GML-Features in ein KML-Placemark.

3.3 Entscheidungskriterien für eine Transformation GML-KML

Die Standardtechnologie zum Umformen eines XML-Dokuments ist eine *XSL-Transformation*. Eine solche *Stylesheet-Transformation* wird mit Hilfe der Übertragungssprache *xsl* und einer *Transformations-Engine* ausgeführt. Diese Vorgehensweise findet überall dort Anwendung, wo eine repräsentative Grammatik existiert, d.h. XML-Konstrukte der einen Grammatik in einer anderen wiederverwendet werden. Im Fall der Übersetzung einer GML-Syntax nach KML ist eine XSLT-Transformation nicht möglich. Die Gründe liegen sowohl in der Struktur als auch in der Semantik der Ausdrücke. Innerhalb der xml-Elemente ist zur Umwandlung das

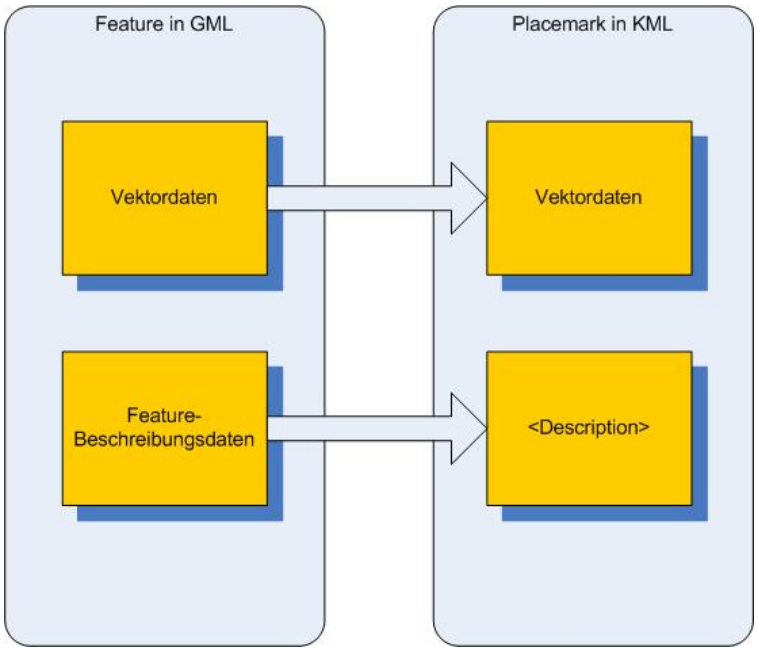

Abbildung 6: Umsetzung von Features

Parsen des Textinhalts einiger Elemente notwendig. Java bietet hierzu durch die Verfügbarkeit von regulären Ausdrücken ein geeignetes Hilfsmittel, um Strukturen innerhalb von Texten zu erkennen. Das Erkennen der Semantik der Inhalte erfordert Logikfunktionen, die mit XSLT nicht abgedeckt werden können, sodass eine höhere Sprache für diesen Zweck verwendet werden muss. Die Entscheidung fällt hier auf Java, da es sich durch die Webfähigkeit und die zahlreichen Bearbeitungsmöglichkeiten für XML-Strukturen auszeichnet. Zusätzlich ist das Open-Source-Framework Deegree in Java implementiert. Da Deegree einen WFS-Dienst bereitstellt, liegt die direkte Integration des Adapters in dieses Framework nahe.

3.4 Der GetFeature-Request

Mit dem *GetFeature-Request* Befehl werden die Geodaten eines WFS-Servers abgefragt und können bei Bedarf auch gefiltert werden. Die Struktur eines GetFeature-Aufrufs wird durch folgendes Beispiel verdeutlicht:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature version="1.1.0"
xmlns:app="http://www.deegree.org/app"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
```

```
  <!-- request all country instances -->
  <wfs:Query typeName="app:Country"/>
</wfs:GetFeature>
```

Der Request wird in diesem Beispiel als XML-Datei über die HTTP-Methode POST an den Server gesendet. Der auszuführende Befehl (`<wfs:GetFeature>`) bildet das Wurzelelement in der XML-Struktur. Die darauf folgenden Namensraum-Deklarationen legen zum einen den Dienst fest, der angefragt wird und enthalten auch den Namensraum der in der Anwendung deklarierten Geodaten (Features). Der WFS-Dienst wird im allgemeinen durch das Prefix `wfs` gekennzeichnet und an den URI *http://www.opengis.net/wfs* gebunden. Der Namesraum der Anwendung verwendet in diesem Beispiel das Prefix `app` und ist mit dem URI *http://www.deegree.org/app* verknüpft. Als weiterer Namensraum ist unter `xsi` eine Referenz zum XML-Schema gesetzt, das zur Verwendung von XML-Schemadateien benötigt wird. Das Attribut `xsi:schemaLocation` verweist auf die Schema-Datei *wfs.xsd*, in der die Syntax und grammatikalischen Regeln einer WFS-Datei spezifiziert sind.

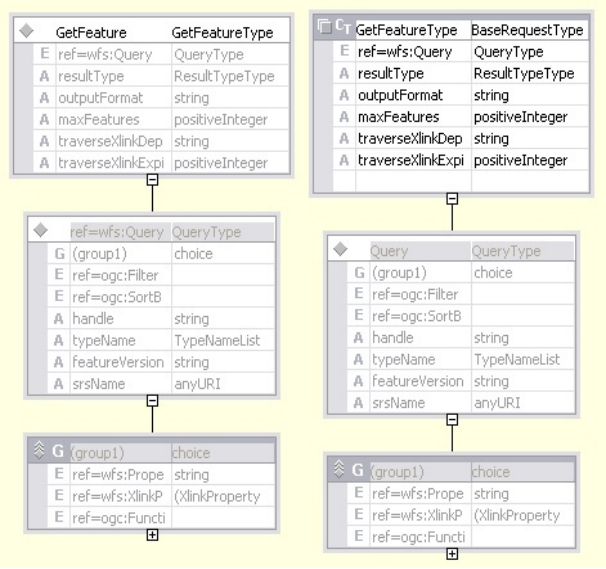

Abbildung 7: Aufbau des GetFeature-Requests in wfs.xsd

Als einziges Element in *GetFeature* ist `<wfs:Query>` erlaubt. Dieses Element enthält die Parameter des Aufrufs. Im obigen Beispiel wird im Attribut `typeName` nach allen Instanzen von `<app:Country>` gefragt. Komplexere Aufrufe sind durch Filterungen (*OGC filter encoding*) möglich. Die Angabe von Filtern schränkt die Ausgabemenge in der Weise ein, dass nur alle Features ausgegeben werden, auf die die Filteroptionen zutreffen. Filterungen können nach unterschiedlichen Merkmalen erfolgen, zum Beispiel:

- Feature-IDs

- Vergleich der Eigenschaften von Features (größer, kleiner, gleich, usw.)
- geografischen Bereichsbeschränkungen durch *BoundingBoxes*
- geografischen Operationen (Überschneiden, Abstand, usw.)

Diese Filteroptionen können beliebig durch die Anwendung von Boolschen Operatoren oder Mengenoperatoren miteinander verknüpft werden. Hieraus ergeben sich semantisch erfassbare Strukturen, die von anderen Anwendungen aus verwendet werden können. In der Aufzählung wurden die sogenannten *BoundingBoxes* erwähnt. Diese sind von spezieller Bedeutung, da sie geometrische Ausmaße beschreiben. Durch die Festlegung zweier ungleicher Eckpunkte wird ein rechteckiger Bereich auf einer Karte definiert (*Envelope*). Dieser Envelope beschreibt die Ausmaße der BoundingBox, die in GML mit dem Element *BBOX* beschrieben wird. Die Filterung über BoundingBoxes erfolgt durch den geografischen Vergleich des Envelopes mit den Raumdaten der *Features*. Befinden sich die Features geografisch innerhalb des Envelopes, werden sie der Ausgabemenge hinzugefügt.

WFS-Server können optional andere Ausgabeformate als GML unterstützen. Deshalb ist die Anfrage nach diesen Ausgabeformaten mittels des optionalen Parameter `outputFormat` möglich, das im *GetFeature*-Request vorgesehen ist. Im Normalfall wird als Ausgabeformat GML mit dem MIME-Type *text/xml; subtype=gml/3.1.1* festgelegt. Das OGC definierte das `outputFormat` ursprünglich, um die Abwärtskompabilität zu früheren GML-Versionen zu ermöglichen. Mittlerweile lässt das OGC Implementierungsfreiheit im Umgang mit dem `outputFormat` zu. So ist es möglich, auch Binärformate oder andere XML-Grammatiken als Ausgabeformat anzubieten. Zusätzlich müssen die Ausgabeformate auch in den Metadaten bereitgestellt werden. Dies erfolgt durch die Angabe des entsprechenden MIME-Types im *Capabilities-Dokument*.

3.5 Design des Übersetzungsprogramms

Um eine XML-Grammatik in eine andere zu überführen, wird ein Übersetzungsprogramm benötigt. Dieses Programm muss die alte Grammatik einlesen und Regeln anwenden, die zur Übertragung einzelner Elemente erforderlich sind. Die Übersetzung wird anhand dieser Regeln durchgeführt und dadurch eine neue Grammatik erstellt. Demenstprechend besteht das Übersetzungsprogramm aus zwei Komponenten:

1. einer Rahmenstruktur, die für das Einlesen, das Parsen, die Bearbeitungsabläufe und die für die Erzeugung der Ausgabe zuständig ist
2. den Übersetzungsregeln, nach denen die Übersetzung syntaktisch und grammatikalisch erfolgen soll.

Für letztere ist die Betrachtung und genaue Analyse beider Grammatiken erforderlich. Wie bereits geschildert, finden Vektordaten aus GML in KML einen ähnlich strukturierten Gegenpart. Trotzdem ist eine 1:1-Übertragung nicht möglich, wie durch folgende Beispiele verdeutlicht wird:

```
<gml:Polygon xmlns:gml="http://www.opengis.net/gml">
  <gml:outerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates>
       8.435833,47.573608 8.390833,47.579994 ...
      </gml:coordinates>
     </gml:LinearRing>
   </gml:outerBoundaryIs>
 </gml:Polygon>
```

Ein in GML definiertes Polygon wie im obigen Beispiel angegeben sieht in KML wie folgt aus:

```
<kml:Polygon xmlns:kml="http://earth.google.com/kml/2.1">
<kml:outerBoundaryIs>
<kml:LinearRing>
8.435833,47.573608 8.390833,47.579994 ...
</kml:LinearRing>
</kml:outerBoundaryIs>
</kml:Polygon>
```

Der Aufbau des Polygons vollzieht sich hier auf eine ähnliche Art und Weise. Die Namensraumangaben und -prefixe dienen nur zur besseren Übersicht und können weggelassen werden. Als Grundelement ist in beiden XML-Grammatiken ein `<Polygon>` spezifiziert. Darunter befindet sich das Element `<outerBoundaryIs>`, das bei der Entwicklung von KML aus GML übernommen wurde. Dieses Element legt die äußeren Ausmaße des Polygons fest, die durch das Vektorformat *LinearRing* ausgedrückt werden. Ein LinearRing ist ein in sich geschlossene Figur, die über einen Pfad aus einer Sammlung von Koordinaten definiert ist. Damit sich eine geschlossene Geometrie bildet, müssen der Anfangspunkt und der Endpunkt identisch sein. GML besitzt zur eindeutigen Identifikation der Punktmenge ein zusätzliches Element `<coordinates>`, das in KML nicht vorhanden ist. Dort folgt die Angabe der Koordinatenelemente direkt nach dem Element `<LinearRing>`. Die Gemeinsamkeiten beider Grammatiken im Umgang mit Vektordaten konnte anhand dieses Beispiels gezeigt werden.

Ein weniger triviales Besipiel erfordert mehr Aufwand bei der Umsetzung. Die Umsetzung von Mehrfachgeometrien erfolgt durch

1. Identifikation der Vektordatentypen
2. rekursive Bearbeitung der einzelnen Geometrie-Objekte

In GML ist eine Vielfachgeometrie immer an den Vektordatentypen gebunden den er enthält:

```
<gml:MultiPolygon srsName="epsg:4326"
xmlns:gml="http://www.opengis.net/gml">
    <gml:polygonMember>
      <gml:Polygon> ... </gml:Polygon>
     </gml:polygonMember>
      <gml:polygonMember>
      <gml:Polygon> ... </gml:Polygon>
     </gml:polygonMember>
     <gml:polygonMember>
      <gml:Polygon> ... </gml:Polygon>
     </gml:polygonMember>
  </gml:MultiPolygon>
```

Neben *MultiPolygon* ist in GML auch der Datentyp *MultiPoint* verfügbar. KML hingegen besitzt nur einen abstrakten Datentyp, der Vektordaten verschiedenen Typs enthalten kann. Beim Parsen der GML-Datei wird beim Auftreten eines Vielfachelementes in GML das KML-Element `<MultiGeometry>` der Ausgabe hinzugefügt, der für die Erzeugung einer KML-Datei verantwortlich ist. Der Übersetzer muss beim Füllen dieser Datenstruktur mit Vektorelementen entscheiden, welche Elementklasse zugefügt wird. Hat er eine Klasse gefunden, muss die der Klasse entsprechende Übersetzungsmethode aufgerufen werden, um Elemente dieses Types zu erzeugen. Diese Methode wird für jedes weitere Element der Mehrfachklasse erneut aufgerufen, wobei jedes Mal eine neue Instanz dieses Vektordatentypes erstellt und der *MultiGeometry* hinzugefügt wird. Der Datenfluss und die Funktionsweise ist in Abbildung 8 dargestellt. Die Ausgabe der am obigen Beispiel definierten Polygone sieht in KML wie folgt aus:

```
<kml:MultiGeometry xmlns:kml="http://earth.google.com/kml/2.1">
     <kml:Polygon> ... </kml:Polygon>
     <kml:Polygon> ... </kml:Polygon>
     <kml:Polygon> ... </kml:Polygon>
</kml:MultiGeometry>
```

Es wurde im Abschnitt *Design einer KML-basierten Schnittstelle* bereits gezeigt, dass sich Beschreibungsdaten in GML-Features nur als unstrukturierte Texte in KML übersetzen lassen. Die Umsetzung der Übertragung dieser Elemente ist verbunden mit der Suche nach Beschreibungselementen in der GML-Datei. Dazu dient die Methode `GetDescriptionElements()`, welche die Beschreibungselemente lokalisiert und zusammenfasst. Die Suche nach den Beschreibungselementen und deren Identifikation erfolgt auf zwei Arten: Feature-Beschreibungen haben den Vorteil, dass sie ein gemeinsames Parent-Element besitzen, und zwar das Element `<gml:FeaureMember>`. Jedoch beinhaltet ein *FeatureMember* neben den Beschreibungselementen auch eine Geometrie. Um die geografischen Daten von den Beschreibungsdaten zu unterscheiden, wird der Namensraum der *Child-Elemente* vom FeatureMember untersucht. Geografische Vektordaten besitzen als Namensraum häufig das Prefix “gml”, während die Beschreibungsdaten an einen anwendungsspezifischen Namensraum gekoppelt sind. Dieser

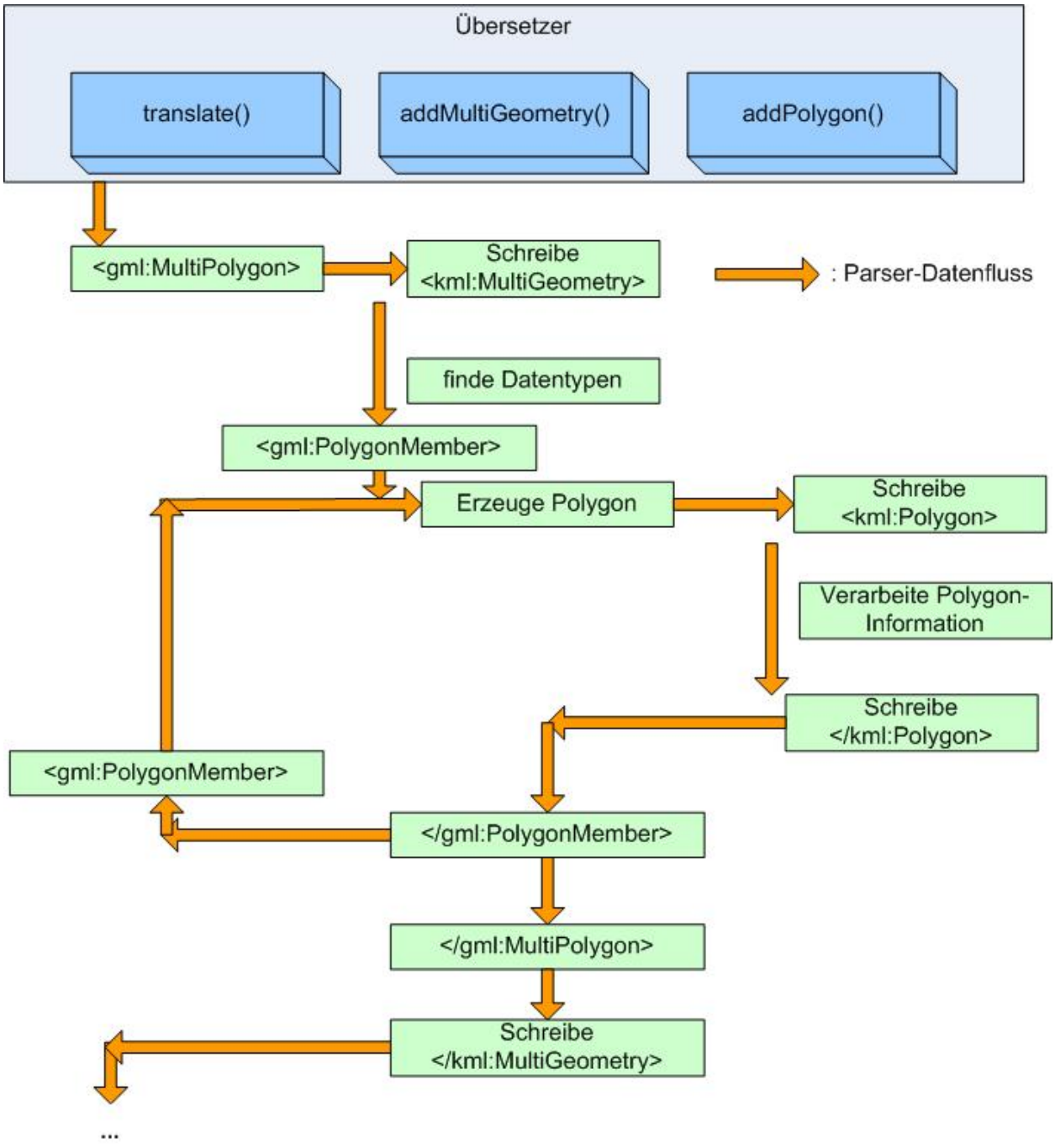

Abbildung 8: Verarbeitung einer Mehrfachgeometrie

trägt in vielen Fällen das Prefix “app”. Da diese Präfixnamen nicht eindeutig festgelegt sind, können sie nicht festcodiert im Übersetzungsprogramm integriert werden, sondern müssen dynamisch gefunden werden. Der Übersetzer sucht sich zu diesem Zweck fest definierte GML-Elemente als Referenz. Für den Namensraum der Geometrien identifiziert die Methode *getNamespaces()* das Prefix durch das Element `<gml:FeaureMember>`, das per Definition an den gml-Namensraum gekoppelt ist. Der Namensraum der Beschreibungselemente ergibt sich dann aus den anderen in `<gml:FeaureMember>` enthaltenen Child-Elementen.
Die Ausgabe der Beschreibungsdaten erfolgt in KML über das `<Description>`-Tag, das in dem Element `<Placemark>` enthalten ist. Durch die Angabe einer *CDATA-Section* können, wie bereits erwähnt, HTML-Elemente mit in die Beschreibung eingebunden werden. Die Ausgabe erfolgt über den Google Earth Client standardmäßig über einen *Baloon*, der die in der *Description* enthaltenen Beschreibungsdaten anzeigt. Abbildung 9 zeigt ein Beispiel für eine HTML-basierte Anzeige.

Abbildung 9: Description-Ausgabe mit HTML-Formatierung

3.6 Programmablauf

Der Programmablauf lässt sich am einfachsten aus der Sicht des Anwenders verdeutlichen. Abbildung 10 zeigt den möglichen Ablauf einer Client-Server-Kommunikation. Die Nummerierung der Elemente beschreibt schematisch die jeweiligen Aufrufe, die zwischen den Instanzen *Client*, *Deegree-Server* und der *KML-Extension* statt finden:

1. Client sendet *GetCapabilities* an den Deegree-WFS-Server
2. *Capabilities* Dokument wird and den Client zurückgegeben
3. Client sendet einen *GetFeature*-Request mit Formatwunsch KML an den Deegree-Server
4. Deegree startet die *KML-Extension*
5. Die KML-Extension fordert von Deegree die benötigten Featuredaten im GML-Format an.
6. Deegree übermittelt die Featuredaten als GML-Dokument an die KML-Extension
7. Nach der Übersetzung in das KML-Format, sendet die KML-Extension das KML-Dokument an den Client zurück

Die Anfrage nach dem *Capabilities* Dokument macht den Client mit den Fähigkeiten des Servers vertraut, damit dieser seine *GetFeature*-Anfragen konstruieren kann. Durch das Capabilities Dokument erfährt der Client, nach welcher Struktur die Geodaten auf dem Server hinterlegt sind, sowie dass die Daten auch im Ausgabeformat "KML" angefordert werden können. Im dritten Schritt sendet der Client seine Anfrage in Form eines *GetFeature*-Requests an den Server, der diesen dann weiterverarbeitet. Über die an den Deegree-Server angebundene *KML-Extension* ist dieser in der Lage, auch KML-Dokumente an Clients auszuliefern. Wie die Anbindung der KML-Extension im Detail erfolgt, wird im nächsten Kapitel beschrieben. Nachdem der Client die angefragten Features im letzten Schritt in einem KML-Dokument erhalten hat, kann er die Anwendung Google Earth starten und sich das erhaltene KML-Dokument anzeigen lassen.

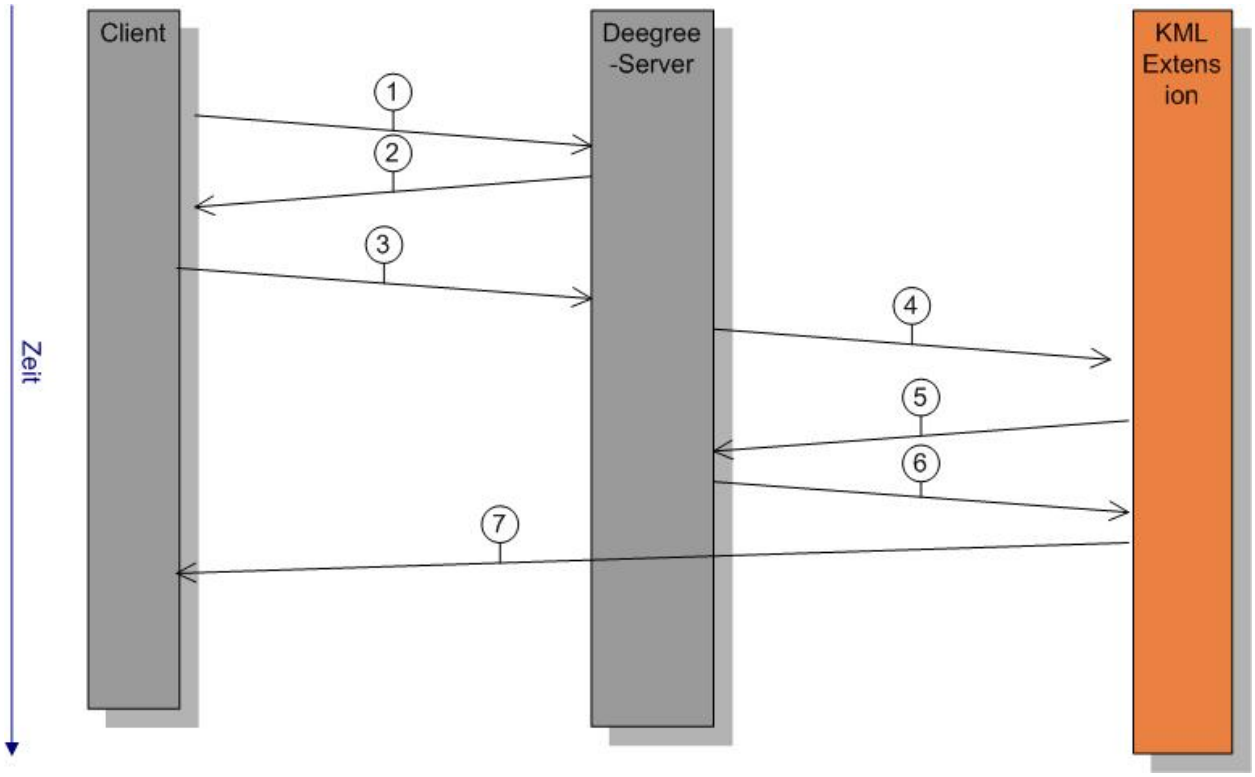

Abbildung 10: Ablauf der Client-Serverkommunikation

4 Realisierung der KML-Extension des deegree-WFS

In diesem Kapitel werden die Implementierungsdetails beschrieben sowie die Technologien vorgestellt, die zur Realisierung der Implementation der KML-Extension für das deegree-Framework verwendet wurden.

4.1 Anbindung an den WFS-Dienst

Das Übersetzungsprogramm benötigt GML-Daten zur Eingabe und erzeugt aus diesen Daten eine KML-Datei. Die erforderlichen GML-Dateien erhält das Übersetzungsprogramm aus dem WFS-Dienst durch die Anfrage mittels *GetFeature*. Die Grundüberlegung besteht darin, wie das Übersetzungsprogramm an einen Web Feature Service angebunden werden kann. Durch die Verwendung des Open-Source Frameworks deegree[2] als WFS-Dienst ist eine direkte Integration in dem WFS möglich, ohne auf andere, externe Hilfsmittel zurückgreifen zu müssen. Abbildung 11 verdeutlicht die Erweiterung von deegree um die Funktion, KML-Dateien auszugeben.

Um eine Integration durchführen zu können, muss die interne Struktur von deegree analysiert werden. Im deegree-Framework dienen die *Servlet-Klassen* dazu, die Schnittstelle in einem Netzwerk bereitzustellen. Das zentrale Servlet in deegree ist der `OGCServletController`, welcher die Anfragen aus dem Netzwerk entgegen nimmt und für die entsprechende Weiterverarbeitung sorgt. Er nimmt die Anfragen über die HTTP-Methoden GET und POST entgegen und analysiert die eingehende Anfrage. Jeder eingehende Request ist einem Dienst zugeordnet, der angefragt wird. Der `OGCServletController` empfängt alle Anfragen, unabhängig von dem angefragten Dienst. Die von einem Servlet bereitgestellten Ein- und Ausgabeklassen `HttpServletRequest` und `HttpServletResponse` werden an weitere Klassen und Methoden durchgereicht, bis eine HTTP-Response erfolgt ist. Aus diesen Servlet-Klassen können Streams erzeugt werden. Sie stellen die einzige Verbindung zum Client dar, der sich mittels einer HTTP tauglichen Applikation mit dem deegree-Service verbindet. Nachdem der `OGCServletController` den Request entgegen genommen hat, muss zuerst der angefragte Dienst herausgefunden werden. Dies geschieht durch Aufruf des Interfaces `OGCWebServiceRequest`. Ist der richtige Dienst gefunden, wird der entsprechende *Handler* aufgerufen, der die Weiterverarbeitung durchführt. Im Falle des WFS-Dienstes übernimmt der `WFSHandler` diese Aufgabe. Nach der Analyse des auszuführenden Befehls wird dieser dann mit Hilfe der Klasse `WFService` ausgeführt. Im Falle des *GetFeature*-Requests wird durch `WFService` ein `GetFeatureHandler` aktiviert, der über eine Kette weiterer Klassen und Methoden die gesuchten Features aus der Datenbank abfragt. Das Resultat erreicht den `WFSHandler`, durch den das Ausgabeformat bestimmt wird. Wurde in der Anfrage das Attribut `outputFormat` definiert und auf einen anderen MIME-Type als *text/xml; subtype=gml/3.1.1* gesetzt, so findet eine Umwandlung in diese Formate statt. Dies ist die Stelle, an der das Übersetzungsprogramm eingebunden wird. Als neuer MIME-Type wird *text/xml; subtype=application/vnd.google-earth.kml+xml* definiert, der zur Identifikation der KML-Dateien dient und bei Angabe im `outputFormat` eine Übersetzung in das KML-Format

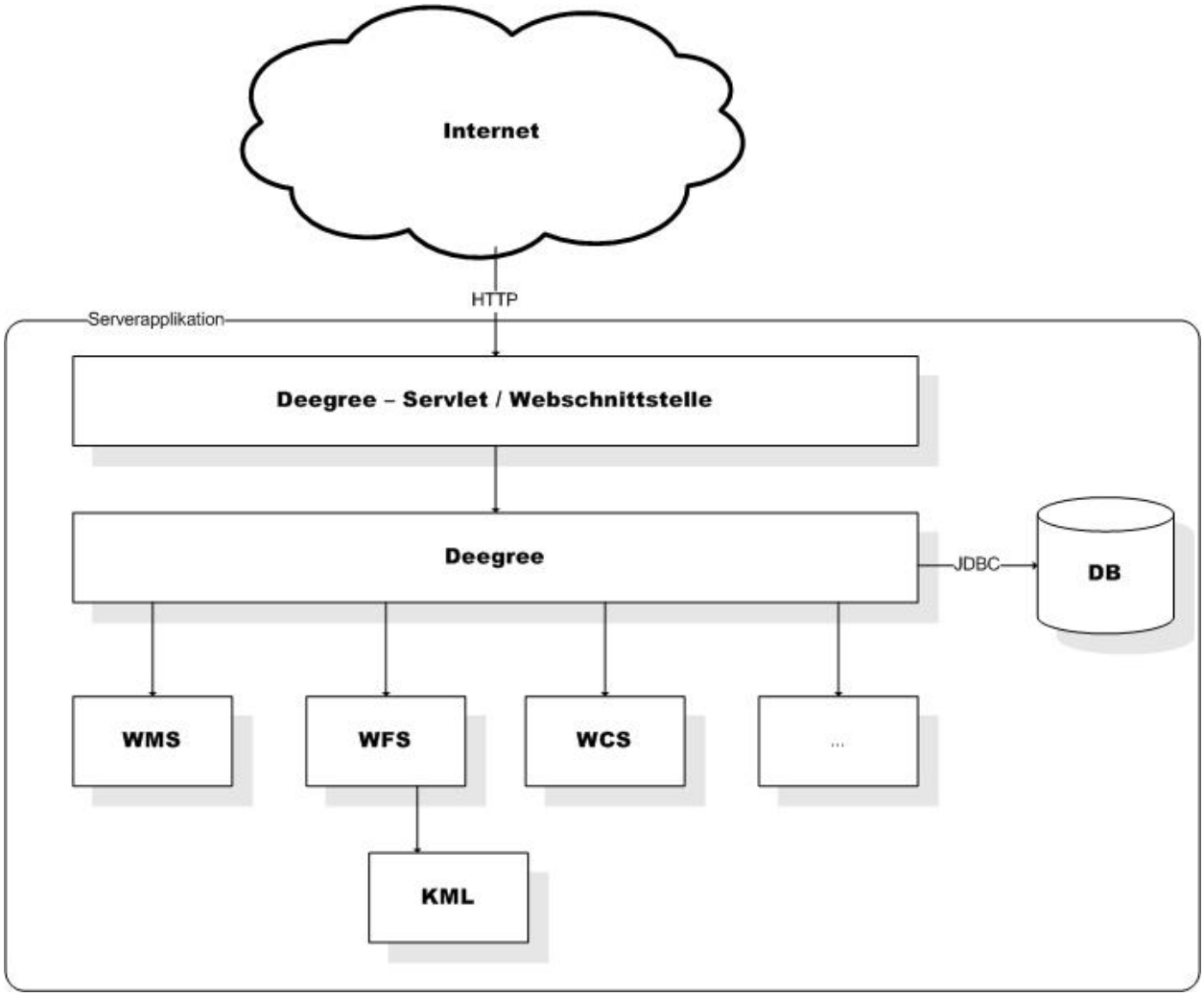

Abbildung 11: Struktur des deegree-Frameworks mit eingebundener KML Extension

durchführt. Bei der *GetFeature*-Abfrage unter Angabe des neu integrierten KML-MIME-Types wird der Datenbestand über den oben beschriebenen Weg abgefragt. Eine Übersetzung in das KML-Format erfolgt nachdem alle abgefragten Daten aus dem Repository zur Verfügung gestellt wurden. Zu diesem Zweck werden neue Klassen definiert, die erstens für die Anbindung an das deegree-Framework sorgen und zweitens die Transformation in das KML-Format durchführen. Als Schnittstelle zur Integration der *KML-Extension*, also der Erweiterung von deegree um das KML-Format, wird der `WFSHandler` verwendet. Hier findet bereits eine Abfrage nach den gewünschten und den angebotenen Formaten statt; somit muss die Klasse nur um eine Abfrage nach dem neuen MIME-Type ergänzt werden. Sobald dieser erkannt wurde, werden eigene Klassen aufrufen, welche die Weiterverarbeitung der HTTP-Streams und der Übersetzung durchführen. Innerhalb von deegree wird zusätzlich überprüft, ob das gewünschte Ausgabeformat auch unterstützt wird. Dies erfolgt durch einen Abgleich des im Request angegebenen *OutputFormates* mit den in dem *Capabilities*-Dokument festgelegten Formaten. Dort können für jeden Featuretype einzeln eigene Ausgabeformate definiert werden. Standardgemäß ist das GML-Format bereits angegeben. Für die Ausgabe in KML muss das *Capabilities*-Dokument um den MIME-Type von Google Earth ergänzt werden. Der für jeden Featuretype definierte Abschnitt `<wfs:OutputFormats>` sieht nach dem Hinzufügen wie folgt aus:

```
<wfs:OutputFormats
xmlns:wfs="http://www.opengis.net/wfs">
<wfs:Format>
text/xml; subtype=gml/3.1.1
</wfs:Format>
<wfs:Format>
text/xml; subtype=application/vnd.google-earth.kml+xml
</wfs:Format>
</wfs:OutputFormats>
```

Fand der Abgleich erfolgreich statt, wird die hinzugefügte Klasse `KMLService` aufgerufen, die die Übersetzung ausführen lässt. Dabei erhält sie vom WFSHandler eine erstellte "Rohfassung" einer GML-Datei, in der die abgefragten Features bereits enthalten sind. Desweiteren wird auch die vom Servlet bereitgestellte Instanz von `HttpServletResponse` an den `KMLService` weitergegeben. Zur Übersetzung wurde eine weitere Klasse mit dem Namen `WFS2KML` implementiert. Dabei parst sie die vom `KMLService` erhaltene GML-Datei und ruft entsprechende Methoden auf, in denen einzelne Konstrukte oder Abschnitte übersetzt werden. Ist die Transformation abgeschlossen, sorgt der `KMLService` für die Ausgabe an den HTTP-Client. Der Ablauf ist in Abbildung 12 skizziert.

Das Klassenschema der KML-Extension ist in Abbildung 13 dargestellt. Die bereits erwähnten Abläufe werden durch diese Klassen realisiert. Die Klasse *Point* dient ausschließlich als Hilfsklasse und ist nur der vollständigkeit halber angegeben. Die Klasse *WFS2KML* enthält die Funktionen, die zur Übersetzung der XML-Grammatiken erforderlich sind. Die Klasse

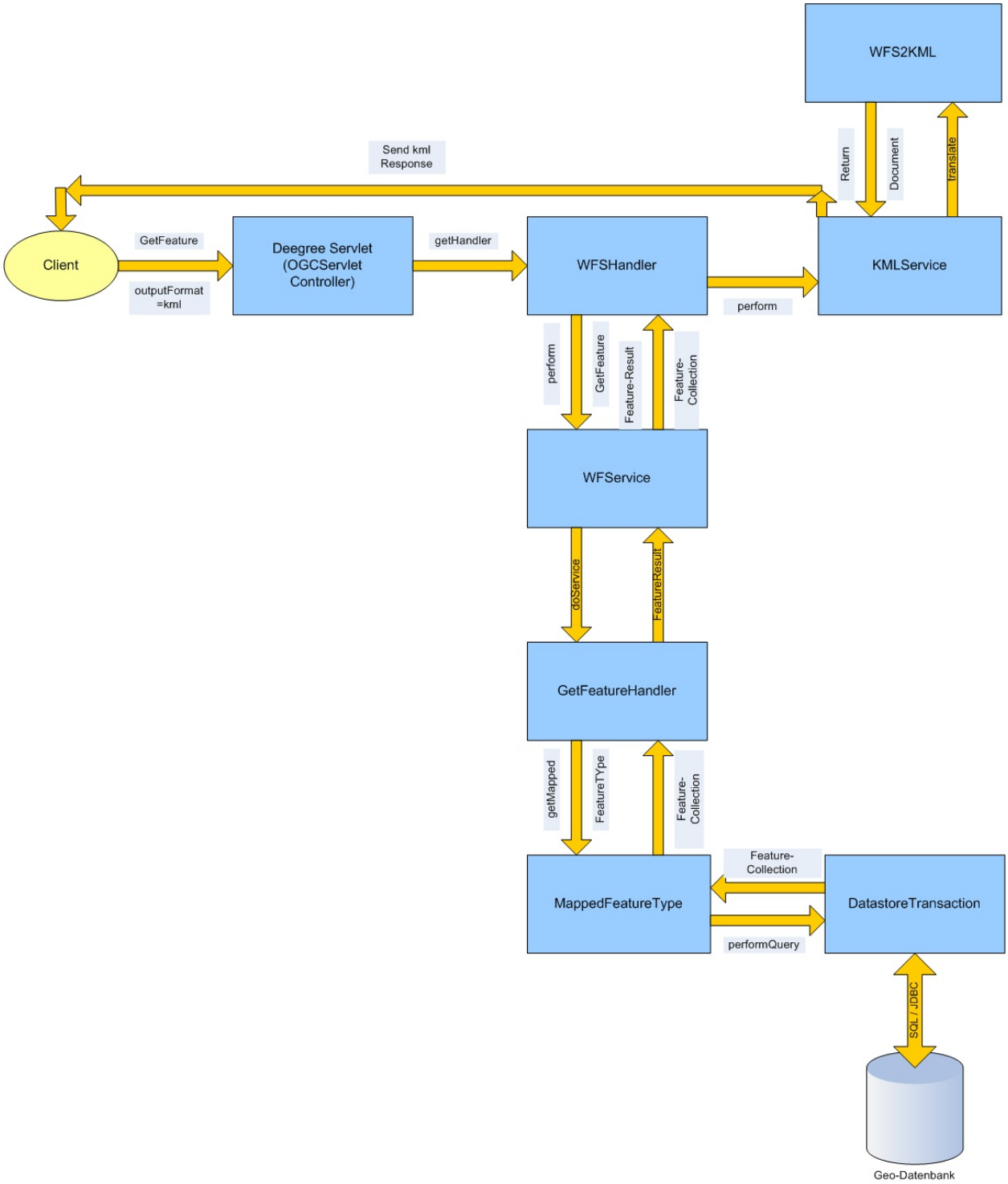

Abbildung 12: Vereinfachter Ablauf eines GetFeature-Request mit dem Ausgabeformat KML

KMLHandler steuert den Übersetzer, in dem er die benötigten Eingabedaten bereitstellt und die Ausgabe an den Client ausführt, wie in Abbildung 8 gezeigt wurde.

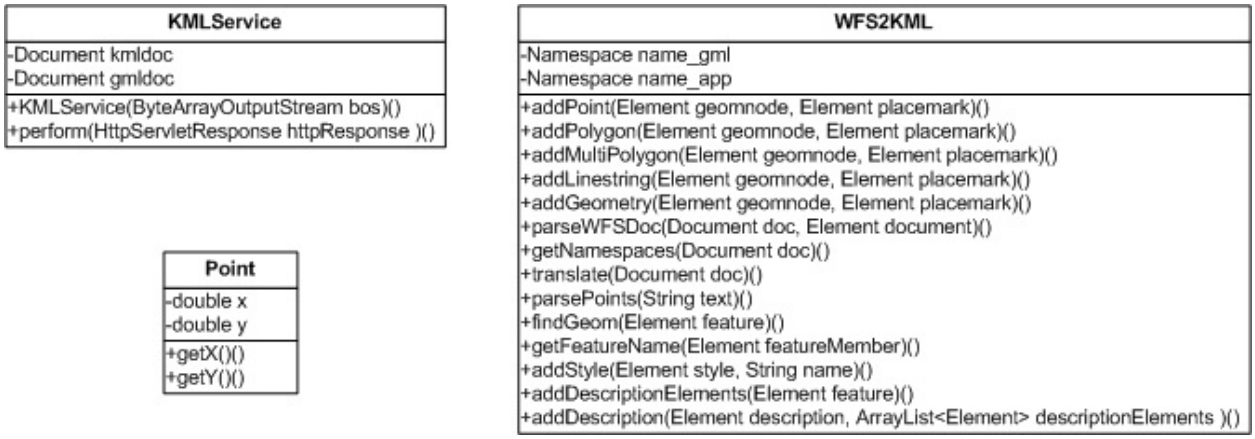

Abbildung 13: UML Klassenschema

4.2 Alternativen

Zu der im vorigen Abschnitt beschriebenen Vorgehensweise, eine KML-Anbindung in einem WFS-Dienst zu integrieren, sind mehrere Alternativen möglich, die sich durch unterschiedliche Ansätze voneinander unterscheiden:

- *Desktop-Anwendung*
- Eigenständiger Dienst im deegree Framework

Als Beiprodukt zur bisher beschriebenen Anwendung ist eine Desktop-Applikation entstanden. Die Ein- und Ausgabeformate sind festgelegt und das Betriebssystem wird verwendet, um den Umgang mit Dateien durchzuführen. Dieses Programm kann auf einem Client installiert und als *Desktop-Anwendung* ausgeführt werden. Als Eingabe wird eine GML-Datei verlangt, die Features enthält. Die Ausgabe erfolgt als KML-Datei, die auf dem lokalen Computer gespeichert wird. Die Anbindung an einen Web Feature Service erfolgt manuell durch den Benutzer des Übersetzungsprogramms. Ihm fällt die Aufgabe zu, einen WFS-Dienst abzufragen und die erhaltenen Resultate als GML-Datei auf dem Computer zu hinterlegen. Die Transformation in eine KML-Datei erfolgt auf die oben beschriebene Art und Weise durch das Parsen der Features.

Eine weitere Alternative ist die Integration der Übersetzung als eigenen Dienst im deegree Framework. Durch diesen Ansatz werden andere OGC-Dienste nicht beeinflusst. Zur Implementierung bietet deegree genügend Hilfsklassen und "Infrastruktur", die für die Erstellung des Dienstes verwendet werden kann. Da deegree den Dienst über das Servlet `OGCServletController` bereitstellen soll, muss dieser den Service auch kennen. Dazu ist

eine Anpassung einiger Klassen erforderlich. Der Aufwand zur Erstellung eines eigenen Services beschränkt sich nicht nur auf die Programmierung; von der Analyse der Anfrage bis zu dem Bereitstellen und Zurückgeben der Geodaten, sondern auch in der Definition eines eigenen Befehlssatzes, inklusive Grammatik und Syntax. Zu Prüfen wäre, wieviel Aufwand eine solche Integration in deegree bedeuten würde. Zudem wäre ein solcher Dienst nicht OGC-konform.

Die Entscheidung für die durchgeführte Vorgehensweise beruht auf den Vorteilen, die dieser Ansatz im Gegenzug zu anderen Möglichkeiten bietet:

- Die OGC-Spezifikation gestattet die Anbindung proprietärer Formate an den WFS
- Deegree bietet im WFSHandler eine geeignete Schnittstelle zur Formatsumwandlung
- Datenbereitstellung erfolgt über den WFS
- Das Verhalten des WFS wird nicht beeinträchtigt
- Vorhandene WFS-Methoden werden zur Analyse des Requests benutzt
- Einfache serverseitige Anpassung der Geodaten an das neue Format

4.3 Komponenten

Der Aufbau des Gesamtsystems wird wesentlich durch die Anforderungen von deegree geprägt. Um einen eigenen WFS aufzusetzen, sind unterschiedliche Applikationen erforderlich. Neben dem bereits beschriebenen Framework deegree ist die Installation eines Java fähigen Webservers und einer Geodatenbank notwendig. Für den Aufbau werden folgende Anwendungen verwendet:

- Apache Tomcat 5.5 Webserver
- PostgreSQL 8.1 Datenbank
- PostGIS 1.1.6 Erweiterung für die PostgreSQL-Datenbank
- deegree 2
- Google Earth Client Version 4.0.1

Während Google Earth nur in der Basisversion kostenlos ist, sind Tomcat, PostGIS und deegree als Open-Source-Anwendungen frei verfügbar. In den Tomcat Server wird deegree inklusive der *KML-Extension* eingebunden. Der Zugriff auf die PostgreSQL-Datenbank erfolgt über die Schnittstelle *JDBC*. Wurden die Zugriffsdaten in der deegree-Konfiguration richtig angegeben, so ist der Zugriff auf die Datenbank möglich. Die *PostGIS*-Erweiterung stattet die PostgreSQL-Datenbank mit zusätzlichen Datentypen und Funktionen aus, die zur Arbeit mit georeferenzierten Daten notwendig sind. Dadurch ist PostgreSQL in der Lage,

Vektordatentypen zu speichern und zu verwalten und kann Operationen auf diese Geodaten anwenden. Beispielsweise sind Entfernungsmessungen möglich, Überschneidungen und Inhalte können überprüft werden, sowie das Durchführen von Koordinatentransformationen in unterschiedliche Koordinatenreferenzsysteme.

Wird der Web Feature Service von deegree angefragt unter der Angabe von KML als Ausgabeformat, so gibt der Server ein KML-Dokument zurück, das auf dem Client gespeichert und ausgeführt werden kann. Beim Ausführen einer KML-Datei wird die Anwendung Google Earth gestartet und die Ansicht wechselt sofort auf die in der Datei angegebenen Geodaten, sofern in diesen keine weitere Ansicht spezifiziert ist.

Weiterhin war die Verwendung der IDE Eclipse für die Implementierung hilfreich, weil sich damit der deegree-Quellcode über eine CVS-Verbindung einfach integrieren ließ und dieser übersichtlich dargestellt wird. Zur Verarbeitung von XML-Daten wurde das Open-Source Tool dom4j verwendet, das Funktionen für die Verarbeitung von XML-Strukturen bereitstellt. Dadurch ist die Verwendung *regulärer Ausdrücke* und *XQuery*-Anfragen möglich, die zum Navigieren innerhalb von XML-Dokumenten nützlich sind. Die Verwendung von *DOM* (*Document Object Model*) und *SAX* (*Simple API for XML*) liegt dem Tool zu Grunde, die für die Verarbeitung von XML auf Element-Ebene sorgen.

4.4 Aufsetzen des deegree-WFS mit der KML-Extension

Deegree ist im Internet als funktionsfähige Server-Anwendung beziehbar. Um jedoch Nutzen aus der *KML-Extension* zu ziehen, wird der Sourcecode von deegree benötigt. Dieser kann durch Zugriff von dem zur Entwicklung bereitgestellten CVS-Server geladen werden. Die Verwendung einer IDE (*Integrated Development Environment* wie beispielsweise Eclipse vereinfacht den Zugriff und die Verwaltung des Quellcodes. Zum Kompilieren der Anwendung werden einige Open-Source-Bibliotheken bereitgestellt, deren Funktionalität von deegree benötigt wird. Zusätzlich müssen allerdings auch proprietäre Bibliotheken eingebunden werden, die aus lizenzrechtlichen Gründen nicht über den CVS-Server bereitgestellt werden. Diese Bibliotheken sind nur bei Verwendung von Software erforderlich, die diese Bibliotheken bereitstellen und beim Kompilieren. Die Klassen der *KML-Extension* können jetzt eingebunden werden.

Wie im vorigen Abschnitt bereits erwähnt, werden zum Aufsetzen eines WFS-Dienstes eine raumdatenfähige Datenbank, sowie ein Java-fähiger Webserver benötigt. Sind diese Softwarekomponenten auf dem System installiert, so kann eine Kompilierung durchgeführt werden. Zur Konfiguration ist vorher noch die Eingabe der Zugriffsdaten für Datenbank und zur Administrationsschnittstelle des Tomcat-Servers nötig. Das Erzeugen der Anwendung deegree wird durch vorkonfigurierte *ANT*-Skripte gesteuert, die das Kompilieren und die Einbindung in den Server übernehmen. Sobald die Anwendung auf dem Server erreichbar ist, können Features mit deegree verwaltet werden.

4.5 Darstellung in Google Earth

Die resultierende Darstellung von Features ist in Abbildung 14 angedeutet. Die in diesem Beispiel ausgelesenen Polygon-Features werden farblich hervorgehoben. Im Seitenbereich ist die kurze Beschreibung der Features dargestellt. Beim Aufruf von Google Earth mit einer KML-Datei, werden die darin enthaltenen Features sofort angezeigt. Die Ansicht der Google Earth Karte wird automatisch auf die Features gerichtet, so dass alle Features sichtbar sind. Wird durch die Steuerung ein hervorgehobenes Element ausgewählt, so öffnet sich ein Baloon, wie in Abbildung 9 gezeigt ist.

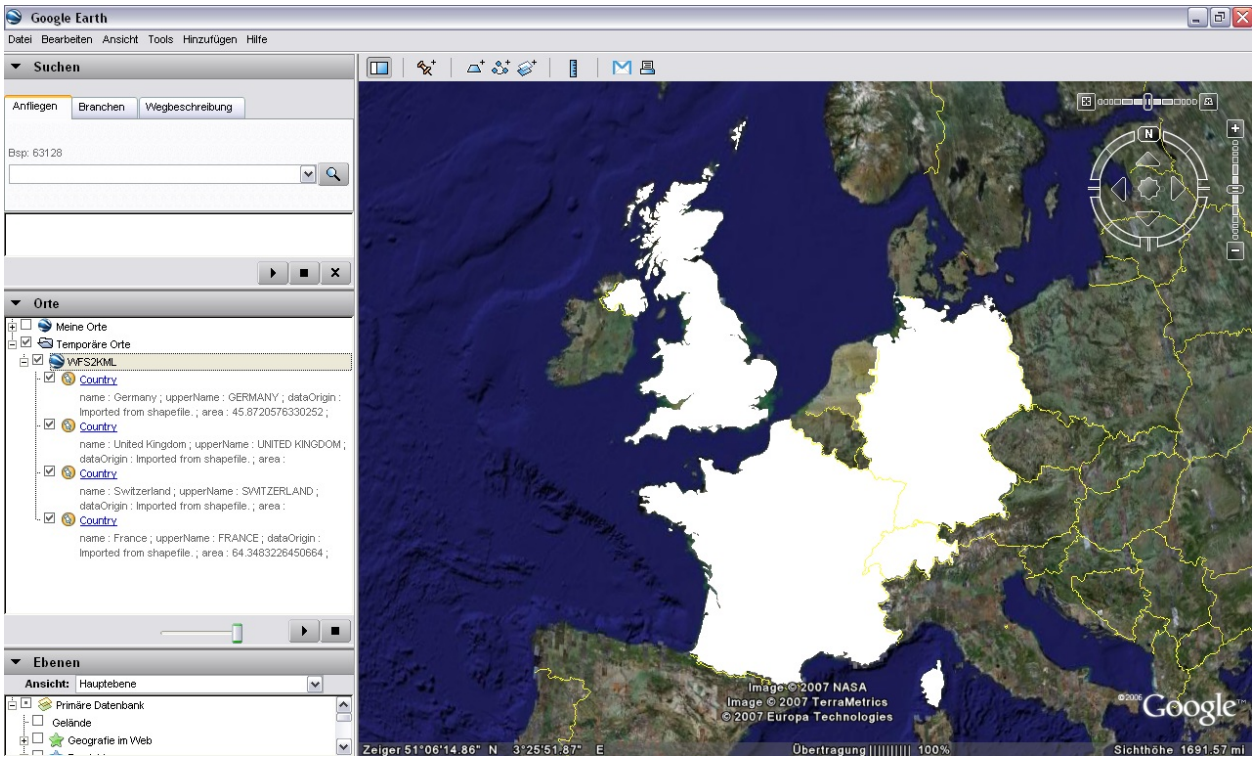

Abbildung 14: Ausgabe der erzeugten KML-Datei

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde gezeigt, wie ein offener standardisierter Web Feature Service erweitert werden kann, um Geodaten für den Planetenbrowser Google Earth bereitzustellen. Dabei sind zuerst die vom OGC standardisierten Spezifikationen und Geodatendienste erläutert worden, danach wurden die Sprachen und Schnittstellen von Google Earth und dem Web Feature Service betrachtet. Im Fogenden wurden Anforderungen definiert, um eine Anbindung von Google Earth an einen Web Feature Service durchzuführen. Die dabei gewonnenen Erkenntnisse beruhen auf der Analyse der beiden XML-Grammatiken GML und KML und der Betrachtung der Funktionsweise des Web Feature Service. Mit Hilfe des offenen Frameworks DEEGREE wurde das Übersetzungsprogramm "KML-Extension" prototypisch realisiert, welches GML in KML überführen kann. Die Implementierung ist auf andere freie Geodatenprodukte anwendbar. Der Prototyp beschränkt sich auf eine schlichte Visualisierung der Geodaten in Google Earth. Eine Erweiterung um flexiblere Visualiserungsoptionen wäre für den praktischen Einsatz notwendig. Hierzu wäre zum Beispiel eine kombinierte Abfrage von WFS und *Styled Layer Descriptor* (SLD-WMS) denkbar.
Durch die stetige Weiterentwicklung der offenen Geodatendienste und auch der Anwendung Google Earth werden weitere Übersetzungsmöglichkeiten geschaffen. Beispielsweise ist die Verarbeitung dreidimensionaler Datentypen ein Gebiet, das in Zukunft sowohl in Google Earth als auch in den OGC-Diensten an Bedeutung gewinnt. Die virtuelle Darstellung von Landschaften in Google Earth durch detailgetreue WFS-Daten ist eine daraus resultierende mögliche Anwendung. Eine weitere interessante Anwendungsmöglichkeit wird durch die Verwendung temporaler Datentypen geschaffen. Auf diese Weise lassen sich Simulationen durchführen, die die Änderung des Raumes in der Zeit darstellen. In GML sind seit der Version 3 temporärer Daten spezifiziert und auch Google Earth experimentiert mit der Integration der "vierten Dimension". Die Entwicklung im Bereich der Geoinformatik schreitet kontinuierlich voran. Die Verwendung offener Schnittstellen hilft, Entwicklungen weiter voranzutreiben.

Literatur

[1] “deegree Web Feature Service”, Lat/Lon GmbH, 2006, http://www.deegree.org/deegree/docs/wfs/deegree_wfs_configuration_2006-10-26.pdf

[2] “deegree API Documentation”, Lat/Lon GmbH, 2006, http://hillary.lat-lon.de/Deegree2_buildresults/nightly_untested/api/

[3] “KML 2.1 Reference”, Google, 2006, http://earth.google.com/kml/kml_tags_21.html

[4] “Web Feature Service Implementation Specification”, Open Geospatial Consortium, 2005, http://www.opengeospatial.org/standards/wfs

[5] “OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1:Common architecture”, Open Geospatial Consortium, 2005, http://www.opengeospatial.org/standards/sfa

[6] “ISO/TC 211/WG 4/PT 19136 Geographic information Geography Markup Language (GML)”, Open Geospatial Consortium, 2004, http://www.opengeospatial.org/standards/gml

[7] “OpenGIS Geography Markup Language (GML) Encoding Specification”, Open Geospatial Consortium, 2006, http://www.opengeospatial.org/standards/gml

[8] “Überlagert - Mit KML Google Earth erweitern”, Christian Wilk, 2006, IX Ausgabe 12 Dezember 2006, Heise Verlag

[9] “Komplexe Nische - GML: geografischer Markup-Standard”, Christian Wilk, 2006, IX Ausgabe 12 Dezember 2006, Heise Verlag