Fachhochschule Karlsruhe - Hochschule für Technik
Fachbereich Geoinformationswesen
Studiengang Kartographie und Geomatik

DIPLOMARBEIT

Erstellung und Umsetzung eines Konzepts für die Integration von ArcIMS-Diensten in ein anderes Geo-Informationssystem

Guido Lohaus

Wintersemester 2003/2004

unter der Leitung von

Prof. Dr.-Ing. Gerhard Schweinfurth

in Zusammenarbeit mit

Landesanstalt für Umweltschutz Baden-Württemberg

& disy Informationssysteme GmbH

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt habe.

Karlsruhe, den _____ Unterschrift: _____

Danksagung

Für die Betreuung meiner Diplomarbeit möchte ich mich bei Herrn Prof. Dr.-Ing. Gerhard Schweinfurth sehr herzlich bedanken.

Ebenfalls bedanken möchte ich mich bei allen Mitarbeitern des Referats 53 der Landesanstalt für Umweltschutz und der Firma disy Informationssysteme GmbH, die mir bei meiner Arbeit stets hilfreich zur Seite gestanden haben. Besonderer Dank gilt Herrn Biologiedirektor Manfred Müller, der es mir ermöglichte das Thema meiner Diplomarbeit ganz auf meine Interessen abzustimmen.

Danke auch meinen Eltern und meinem Bruder, die mich während meines Studiums immer unterstützt haben.

Impressum

Diplomand & Autor:	Guido Lohaus
Verantwortlich:	Prof. Dr.-Ing. Gerhard Schweinfurth (FH Karlsruhe) Biologiedirektor Manfred Müller (Landesanstalt für Umweltschutz) Dipl. Inform. Claus Hofmann (disy Informationssysteme GmbH)
Betreuer (LfU):	Dipl.-Ing. (FH) Andreas Braß
Betreuer (disy):	Dipl.-Inform. Markus Gebhard

Inhaltsverzeichnis

1 Vorwort

Seit vielen Jahren wird das World Wide Web (WWW) täglich von mehreren Millionen Menschen privat oder beruflich zum Daten-, Informations- und Wissensaustausch genutzt. Webkarten als Informationsmedium erfreuen sich dabei immer stärkerer Beliebtheit, so dass sie in manchen Bereichen sogar konventionelle Printkarten verdrängen ([Dick01, S. 6] und [PeTs03, S. 11]).

Die Nutzung innovativer Internet-Technologien hat zur Weiterentwicklung dieser (meist statischen) Webkarten „zu interaktiven und mit abfragbaren Sachdaten hinterlegten Produkten“ [Dick01, S. 6] geführt. Geografische Informationssysteme für das WWW (WebGIS) ermöglichen es einer breiten Öffentlichkeit raumbezogene Daten zu visualisieren, ohne dass spezielle GIS-Software und umfangreiche GIS-Kenntnisse benötigt werden. Ein Web-Browser genügt, um Geodaten der unterschiedlichsten Datenformate und Anbieter betrachten zu können.

Die Entwicklung der WebGIS ist mittlerweile so weit fortgeschritten, dass vielfältige GIS-Funktionen über das Internet ausgeführt werden können und so die Karten nicht nur betrachtet, sondern auch analysiert und bearbeitet werden können.

Das Konzept von WebGIS der Zukunft sieht außerdem die vollständige Interoperabilität (Austauschbarkeit) von Daten und Funktionen der verschiedenen WebGIS-Anwendungen untereinander vor, so dass von verteilten geografischen Informationsdiensten (engl. Distributed GIServices) gesprochen werden kann [PeTs03, S. 4f].

Derzeit ist das Ziel der Interoperabilität jedoch noch nicht erreicht. Im Jahr 2000 gab es bereits über 30 verschiedene WebGIS-Anwendungen verschiedenster Anbieter [PeTs03, S. 377] mit jeweils unterschiedlichen Funktionsumfängen und Datenformaten, so dass sich eine Zusammenführung schwierig gestaltet.

Die vorliegende Diplomarbeit befasst sich unter anderem mit der Problematik der Interoperabilität, da viele WebGIS-Produkte zwar Standardschnittstellen des Open-GIS Consortiums (OGC) [OGC03] unterstützen, darüber hinaus jedoch vorwiegend eigene Schnittstellen verwenden, um zusätzliche Funktionalitäten bereitstellen zu können.

Bei der Integration von Diensten der WebGIS Anwendung ArcIMS der Firma ESRI in das Berichtssystem der Landesanstalt für Umweltschutz wurde schnell deutlich, dass Anspruch und Realität bei der Anwendung der vom OGC formulierten Regeln für Standardschnittstellen weit auseinander klaffen. ESRI wirbt zwar mit der Interoperabilität ihrer Produkte, jedoch ist die Implementierung der Standardschnittstellen für ArcIMS als dürftig zu bezeichnen, so dass diese in meiner Arbeit entweder gar nicht oder nur eingeschränkt verwendet werden konnten.

2 Grundlagen

2.1 Geografische Informationssysteme

Ausführliche Beschreibungen Geografischer Informationssysteme (GIS) finden sich an vielen Stellen in der Fachliteratur. Die folgende Definition aus [BiFr94, S. 5] soll die wesentlichen Punkte kurz zusammenfassen:

> „Ein Geo-Informationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfaßt und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch repräsentiert werden."

Ein Beispiel für ein GIS ist GISterm, das in Abschnitt 2.4.2 genauer beschrieben wird. Da in dieser Arbeit die Datenbestände eines GIS und deren grafische Repräsentation am Bildschirm eine wichtige Rolle spielen, werden diese im folgenden Abschnitt kurz vorgestellt.

2.1.1 Raumbezogene Daten

Raumbezogene Daten, auch Geodaten genannt, beschreiben einzelne Objekte der Landschaft (Geo-Objekte), wie Gegenstände, Geländeformen und andere Strukturen an der Erdoberfläche, mit Hilfe von *Geometriedaten* und *Sachdaten* [BiZe01, S. 106]. Geometriedaten „sorgen für den Raumbezug sowie die geometrische Definition eines raumbezogenen Objekts" [BiZe01, S. 111], wohingegen Sachdaten, auch Attributdaten genannt, durch nichtgeometrische Daten den thematischen Inhalt dieses Objekts wiedergeben [BiZe01, S. 228].

Geo-Informationssystemen liegen Geometriedaten in zwei verschiedenen Datenformaten vor:

- Zur Speicherung von Geometriedaten als *Rasterdaten* werden raumbezogene Objekte äquidistant diskretisiert und quantisiert, d. h. mit Hilfe eines Zeilen-Spaltensystems in quadratische Zellen gleicher Größe (Pixel) aufgeteilt [BiZe01, 219]. In einem GIS sind Rasterdaten meist georeferenzierte Bilddaten (Rasterbilder). Beispiele für Geometriedaten, die als Rasterdaten vorliegen, sind gescannte topographische Kartenwerke, Orthofotos sowie Luft- und Satellitenbilder.

- *Vektordaten* beschreiben Geometriedaten dagegen mit Hilfe kartesischer Koordinaten, also einer Reihe von Punktkoordinaten, so dass jedes Geo-Objekt durch eine Liste geordneter x,y-Koordinaten repräsentiert wird [BiZe01, S. 255]. Am Bildschirm werden Vektordaten mit Hilfe von *Vektorgrafik* visualisiert. Dazu werden Geo-Objekte durch die geometrischen Grundelemente Punkt, Linie und Fläche repräsentiert, auf die dann Zeichenvorschriften angewendet werden [BiZe01, S. 255].

Abbildung 2.1 zeigt ein Rasterbild, das von verschiedenen Vektorgrafiken, die Städte, Flüsse und Verwaltungseinheiten repräsentieren, überlagert wird.

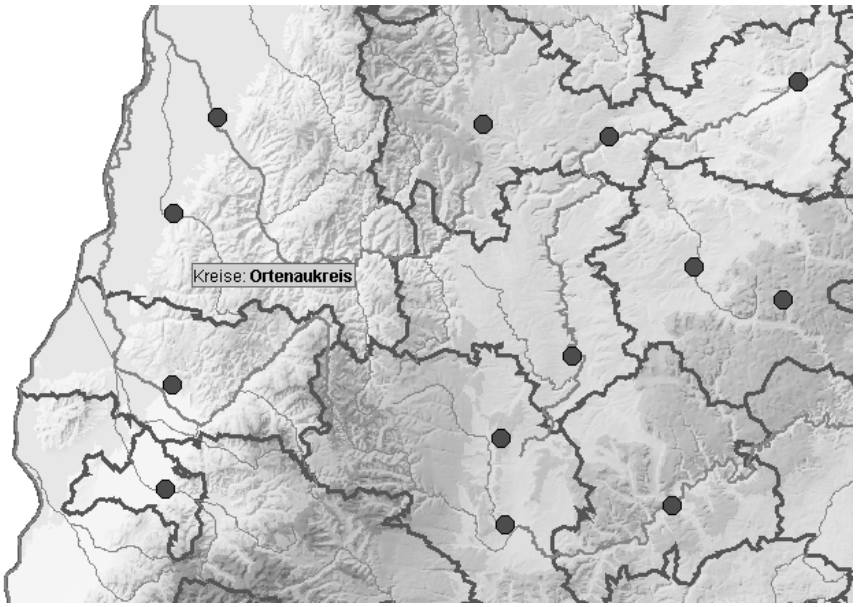


Abbildung 2.1: Überlagerung eines Rasterbildes mit Vektorgrafik.

2.1.2 WebGIS

Die Entwicklung der GIS ist maßgeblich vom Siegeszug der Informationstechnik beeinflusst worden [PeTs03, S. 5]. Neue Technologien wie Java™ [Sun04], XML [W3C04a], etc. verwenden Standardtechnologien wie das Hypertext Transfer Protokoll (HTTP) [W3C04b] und die HyperText Markup Language (HTML) [W3C04c] und ermöglichen so die Nutzung und Verteilung von Anwendungen und Daten in Form von *Diensten* (engl. *Services*) an den Nutzer über das Internet. Die Dienste eines GIS (*GI-Dienste*, engl. *GIServices*) bestehen demnach aus der Lieferung von Geodaten und der Bereitstellung von Bearbeitungswerkzeugen [PeTs03, S. 12].

Verteilte GIS (engl. *Distributed GIS*) nutzen sowohl Technologien der drahtlosen Datenübertragung (z. B. WAP), als auch der drahtgebundenen Kommunikation mit Hilfe des Internets. Sie lassen sich daher in *Mobiles GIS* (engl. *Mobile GIS*) und *Internet GIS* unterteilen [PeTs03, S. 8].

Peng und Tsou sprechen von *verteilten GI-Diensten* (engl. *Distributed GIServices*), wenn Dienste sowohl eigene Geodaten und GIS-Funktionalitäten bereitstellen können, als auch die Daten und Funktionen anderer Dienste nutzen können [PeTs03, S. 7-9]. Clients verlangen Dienste und Server stellen sie bereit. Demnach ist ein Server ebenfalls ein Client, wenn er einen Dienst von einem anderen Server anfordert, wodurch jeder GI-Dienst sowohl Client als auch Server sein, je nachdem wofür er gerade verwendet wird. Das OGC bezeichnet einen Server, der auf diese Weise eingesetzt werden kann, auch als *Cascading Map Server* [OGC02a, S. 2].

WebGIS ist ein Spezialfall des Internet GIS. Das „Internet (interconnecting network) ist ein globales Netzwerk von Rechnern (Internet Servern), die über TCP/IP (Transmission Control Protocol / Internet Protocol) miteinander kommunizieren“ [HGM01, S. 227]. Während sich Internet GIS auf die Verwendung des Internets und seiner Infrastruktur allgemein bezieht, so steht WebGIS für die Nutzung des World Wide Web (WWW) als Medium, also einer Netzwerkanwendung, die HTTP verwendet und aus miteinander verlinkten Webseiten besteht, die „auf weltweit verteilten Webservern abgelegt sind“ [HGM01, S. 229].

2.1.3 WebGIS-Architektur

Ein WebGIS basiert, wie alle Internet GIS Anwendungen, auf dem Client/Server-Prinzip [PeTs03, S. 12]. „Von dezentralen Nutzerrechnern, den Clients, aus erfolgt über das Internet der Zugriff auf einen (Geodaten-) Server, der über räumliche Präsentations- und Analysemöglichkeiten sowie über die entsprechenden Datensätze verfügt“ [Dick01, S. 50]. Der Server führt die vom Client gestellte Aufgabe entweder selbst aus und sendet das Resultat zum Client zurück, oder er sendet Geodaten zusammen mit den zur Analyse der Daten benötigten Werkzeugen zum Client. Abhängig vom Umfang der auf Client-Seite ausgeführten Funktionen spricht man von einem „Thin Client“ oder einem „Thick Client“ [PeTs03, S. 13].

Ein *Thin Client* dient lediglich als grafische Benutzeroberfläche (engl. graphical user interface, kurz GUI) zur Eingabe der Anfrage (engl. request) an den Server und zur Darstellung der zurückgelieferten Antwort (engl. response). Jegliche Aufbereitung und Analyse der Daten wird vom Server ausgeführt. Dadurch kommt es zu einer häufigen Kontaktaufnahme des Clients mit dem Server.

Bei einem *Thick Client* verhält es sich genau anders herum. Der Server überträgt Daten an den Client, die dort verarbeitet werden können. Der Server muss nur abgefragt werden, wenn neue Daten benötigt werden.

Einige Clients lassen sich jedoch nicht genau einer der beiden Definition zuordnen, da sie zwar einen Großteil der Arbeit dem Server überlassen, jedoch clientseitig

einige Funktionen selber ausführen können. Diese Clients werden nachfolgend als *Medium Clients* bezeichnet.

Ein komplexes Internet GIS besteht häufig aus drei oder mehr Schichten. Neben den *Clients* und einem *Webserver* (engl. *Web Server*), gibt es zusätzlich den *Application Server*, den *Map Server*, und den *Datenserver* (engl. *Data Server*) (siehe auch Abbildung 2.2) [PeTs03, S. 20-23].

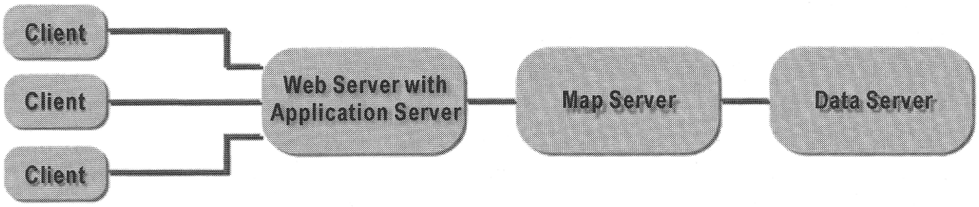


Abbildung 2.2: Komponenten eines Internet GIS (aus [PeTs03, S. 20]).

Der Webserver nimmt Anfragen der Clients entgegen, generiert statische Webseiten und ruft den Application Server auf, der Zugriffsrechte überprüft und die Anfragelast auf einen oder mehrere Map Server verteilt (*Load Balancing*). Der Map Server verarbeitet nun die Anfragen der Clients und generiert die Resultate (z. B. Karten). Die dazu benötigten Geodaten und sonstige Daten werden von Datenservern bereitgestellt. Webserver, Application Server und Map Server können zu einer *Verarbeitungsschicht* (engl. *Logic*) zusammengefasst werden, so dass sich zusammen mit der *Präsentationsschicht* (Clients) und der *Datenschicht* (Datenserver) eine für viele Anwendungen typische Drei-Schichten Architektur ergibt (siehe Abbildung 2.3).

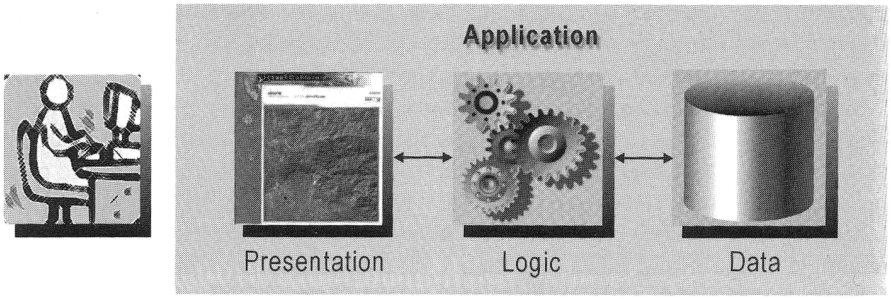


Abbildung 2.3: Elemente einer typischen Anwendung (aus [PeTs03, S. 98]).

2.2 OpenGIS Implementation Specifications

2.2.1 Einleitung

Die in den letzten Jahren immer größer werdende Vielfalt an GIS-Software und Datenformaten machte den Austauch von Geodaten zwischen den einzelnen Produkten immer aufwendiger. Die Definition neuer, „interoperabler“ Formate wurde unumgänglich.

„Interoperabilität bezeichnet die Möglichkeit, verschiedenartige Daten in einen einzelnen Arbeitsablauf zu integrieren“ [BiZe01, S. 141]. Auf WebGIS-Anwendungen bezogen, bedeutet dies die systemübergreifende Zusammenführung von Geodaten und GIS-Funktionalitäten, so dass der Austausch und die Nutzung von Datenbeständen und Funktionen verschiedener Map Server ohne vorherige Überführung (in ein proprietäres Format) in einem einzelnen Arbeitsablauf stattfinden kann.

Das 1994 gegründetet OpenGIS Consortium (OGC) [OGC03] hat sich die Definition von Standards zur Aufgabe gemacht, die die vollständige Integration der Geodatenverarbeitung in die normale Informationsverarbeitung erleichtern und so den weit verbreiteten Einsatz von GI-Diensten sowie interoperabler Standardsoftware fördert [BiZe01, S. 198], [PeTs03, S. 257].

Zur Gewährleistung der Interoperabilität bei Internet GIS Anwendugen hat das OGC Richtlinien herausgebracht, in denen die Funktionalitäten von bestimmten OpenGIS Schnittstellen und Diensten definiert sind. Diese *Implementation Specifications* dienen als Leitfaden für Anwendungsentwickler und Anwender und beschreiben wie die Programmierschnittstellen (engl. application programming interface, kurz API) oder Programmiersprachen für den Zugriff auf Geodaten und GIS-Funktionalitäten aufgebaut sein müssen, damit eine Anwendung als OpenGIS-konform bezeichnet werden kann [PeTs03, S. 261].

Im Folgenden werden lediglich diejenigen Spezifikationen beschrieben, die für diese Diplomarbeit von Bedeutung sind.

2.2.2 WMS Implementation Specification

Die *OpenGIS Web Map Service Interface Implementation Specification* (*WMS Spezifikation*) definiert Schnittstellen (engl. interfaces) zur Standardisierung von Web-GIS Anwendungen und beschreibt die wichtigsten Funktionen (Dienste) heutiger Web Map Server (WMS) [PeTs03, S. 191]. Diese Schnittstellen basieren auf standardisierten Regeln für Anfragen und Antworten unter Verwendung von HTTP und XML.

Die WMS Spezifikation besagt, dass der Client (Web-Browser) die Funktionen eines WMS in Anspruch nehmen kann, indem er lediglich Anfragen in Form von HTTP Uniform Resource Locators (URLs) an den Server stellt. Der Aufbau dieser URLs hängt von der Art der Aufgabe ab, die der WMS erfüllen soll [OGC00, S. 9].

Zur Beschreibung der Anfrage verwendet der Client ausschließlich die Request-Methode HTTP-GET. Sie verwendet Paare aus Schlüsselwort und Wert (engl. key-value pairs, kurz KVP), die als Query-String an die URL angehängt werden, zur Übertragung der Request-Parameter an den Server. Beispiel:

```
REQUEST=GetCapabilities
```

Hierbei repräsentiert "Request" das Schlüsselwort und "GetCapabilities" den Wert. Durch Verwendung bestimmter KVP kann die Art der Anfrage festgelegt werden, wobei mehrere KVP durch „&" getrennt werden. Die vollständige URL für einen *GetCapabilities*-Request könnte zum Beispiel so aussehen:

```
http://www.mywms.de/wms.cgi?REQUEST=GetCapabilities&SERVICE=WMS
```

Die Entwicklung der WMS Spezifikation hat bisher drei Versionen hervorgebracht. Die Web Map Service Interface Implementation Specification Version 1.0.0 (WMS 1.0.0), WMS 1.1.0 und WMS 1.1.1. Ab WMS 1.1.0 werden die drei wichtigsten Dienste eines WMS durch die folgenden Schnittstellen repräsentiert:

- *GetCapabilities*

  Gibt Auskunft über die Eigenschaften (Fähigkeiten) des WMS. Dazu gehört die Auflistung der unterstützten Schnittstellen sowie der durch den Kartendienst bereitgestellten Kartenebenen. Diese und weitere Informationen werden in einer XML-Datei, der sogenannten Capabilities-Datei, an den Client als Antwort übermittelt.

- *GetMap*

  Über diese Schnittstelle kann eine Karte angefordert werden, die meist als Rasterbild im JPG-, GIF- oder PNG-Format an den Client zurückgegeben wird.

- *GetFeatureInfo*

  Diese Schnittstelle ermöglicht die Beantwortung einfacher Abfragen bezüglich des Karteninhalts. Auf diesem Weg können zum Beispiel die Attributdaten der Kartenelemente erfragt werden.

Ein WMS muss mindestens die Schnittstellen *GetCapabilities* und *GetMap* unterstützen, optional kann zusätzlich *GetFeatureInfo* angeboten werden.

Für eine ausführliche Beschreibung der WMS Spezifikation siehe [OGC02a] bzw. [PeTs03, S. 189-198].

Darüber hinaus werden in der Styled Layer Descriptor Implementation Specification (SLD) [OGC02d] zusätzliche Schnittstellen für einen *SLD-WMS* beschrieben. Sie ermöglichen unter anderem die Anpassung der grafischen Gestaltung der Kartenelemente an eigene Bedürfnisse und den Abruf von Legendengrafiken für einzelne Themen [OGC01a, S. ix].

2.2.3 Geographic Markup Language

Die Geographic Markup Language (GML) ist eine auf XML basierende Vektorbeschreibungssprache, die lediglich zum Austausch und zur Speicherung von Vektordaten und den zugehörigen Attributdaten dient [BiZe01, S. 116]. GML soll verschiedene GIS-Funktionen unterstützen, kann aber nicht direkt zur Präsentation von Vektordaten verwendet werden. Die Sprache trennt, anders als zum Beispiel HTML, zwischen ihrer Dokumentenstruktur (Inhalt) und der Datenpräsentation.

GML wurde vom OGC zur Unterstützung der Interoperabilität zwischen verschiedenen Datenmodellen entwickelt. Um dies zu gewährleisten, stellt die Sprache fundamentale Geometrie-Tags sowie einen Mechanismus zur Erstellung und zum Austausch von Anwendungsschemata zur Verfügung und hat mit dem Simple Feature Datenmodell [OGC99] eine allgemeine Grundlage.

Ein Feature (Geo-Objekt) ist „eine Gruppe von räumlichen Elementen, die zusammen eine Einheit der realen Welt repräsentieren“ [BiZe01, S. 90]. Einem Feature können daher sowohl Attributdaten als auch Geometrien zugeordnet werden [BiZe01, S. 192f.]. Die Eigenschaften eines Objekts bestehen aus {Name, Typ, Wert} Tripel. Geo-Objekte haben mindestens eine Eigenschaft mit räumlichem Bezug. Die geometrischen Eigenschaften der Simple Features sind auf einfache Geometrien, die in zweidimensionalen Koordinatensystemen definiert sind, beschränkt. Sie besitzen daher nur eine einfache Topologie mit linearer Interpolation zwischen ihren Stützpunkten.

Abbildung 2.4 auf Seite 10 zeigt die Klassenhierarchie der Simple Features wie sie in der GML verwendet wird. Simple Features sind von der Oberklasse *Geometry* abgeleitet und mit einem Koordinatensystem verknüpft. Null-, ein- und zweidimensionale Geometrien werden von den Basisklassen *Punkt*, *Linie* und *Polygon* repräsentiert. Aus ihnen lässt sich jede andere Geometrie ableiten, unter anderem auch Ansammlungen (*Feature Collections*) gleichartiger (homogener) und unterschiedlicher (heterogener) Objekte.

Darüber hinaus erlaubt GML Features mit komplexen oder zusammengesetzten Attributen.

Eine ausführliche Beschreibung der GML ist in [OGC01b] zu finden.

2.2.4 WFS Implementation Specification

Die *Web Feature Service Interface Implementation Specification* (*WFS Spezifikation*) beschreibt Schnittstellen zur Bereitstellung und Manipulation von Geo-Objekten unter Verwendung von HTTP und XML. Die Beschreibung der Objekte innerhalb der Schnittstelle erfolgt dabei in GML.

Für einen Web Feature Server (WFS) sind folgende Dienste definiert:

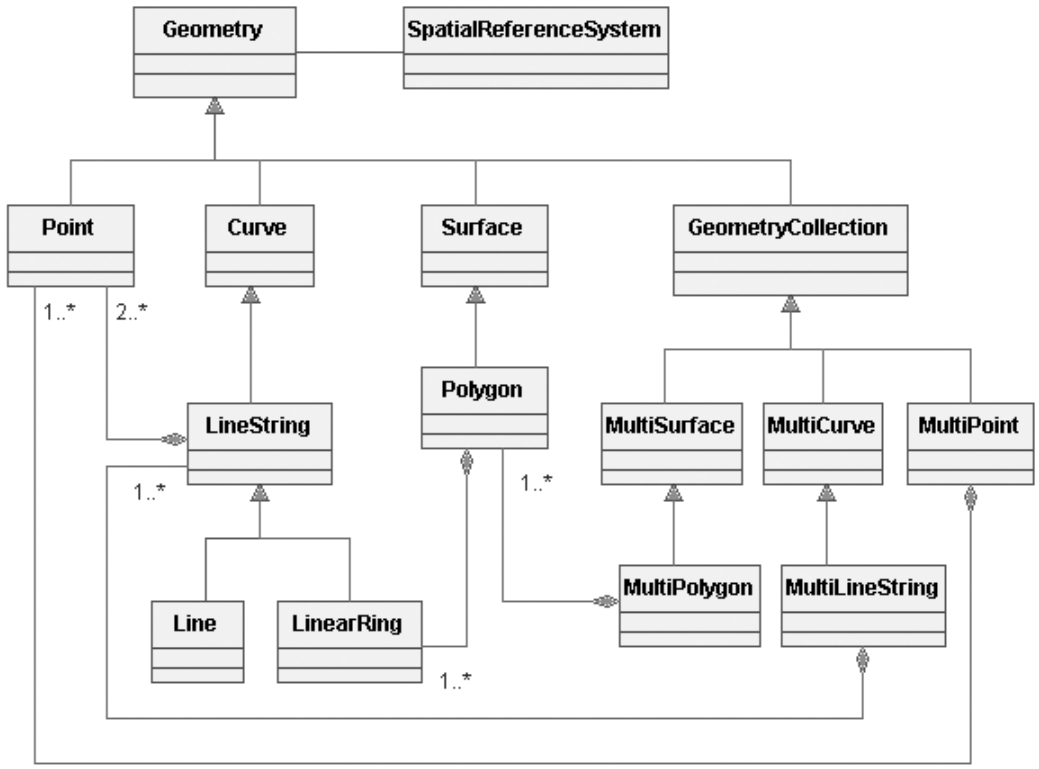


Abbildung 2.4: Geometriemodell der Simple Features (aus [OGC01b, S. 7]).

- *GetCapabilities*

  Ein WFS muss Auskunft geben können über die verfügbaren Objekttypen (engl. Feature Types) und die unterstützten Schnittstellen. Diese und andere Fähigkeiten werden auf Anfrage in einer XML-Datei zurückgeliefert.

- *DescribeFeatureType*

  Über diese Schnittstelle können genauere Informationen über die einzelnen Objekttypen angefordert werden. In einem XML Schema Dokument [W3C04d] wird die Struktur der Objekttypen beschrieben, so dass zum Beispiel die Namen und Datentypen der Eigenschaften herausgefunden werden können.

- *GetFeature*

  Ein WFS muss auf Anfrage eines Clients in GML kodierte Geo-Objekte bereitstellen können. Zusätzlich soll der Client bestimmte Eigenschaften dieser Objekte auswählen können, indem er die Abfrage durch räumliche oder nichträumlichen Kriterien einschränkt.

- *Transaction*

  Ein WFS kann eine Schnittstelle für Datenmanipulationsvorgänge (Transaktionen) unterstützen, die das Erstellen, Verändern, und Löschen von Geo-Objekten ermöglicht.

- *LockFeature*

  Um die Konsistenz der Daten bei Transaktionen zu gewährleisten, kann ein WFS für die Dauer der Transaktion eine Sperre über eine oder mehrere Geo-Objekte verhängen, so dass nicht mehrere Clients gleichzeitig Transaktionen auf denselben Objekten ausführen können.

Auf Grundlage der oben genannten Schnittstellen sind zwei Arten von WFS definiert:

Ein *Basic WFS* muss mindestens die Schnittstellen *GetCapabilities*, *DescribeFeatureType* und *GetFeature* unterstützen, er gewährt also nur Leserechte.

Ein *Transaction WFS* umfasst zusätzlich die Operation *Transaction* sowie optional *LockFeature* und unterstützt damit auch Editierfunktionen.

Anders als bei einem WMS kann ein WFS sowohl HTTP-GET als auch HTTP-POST als Request-Methode unterstützen. Dabei wird der Request in XML kodiert und im Rumpf des POST-Dokumentes an den Server übertragen.

Der *GetCapabilities*-Request würde dann wie folgt aussehen:

```
<?xml version="1.0" ?>
<GetCapabilities xmlns"http://www.opengis.net/wfs" service="WFS" />
```

Ausführliche Informationen über die WFS Spezifikation finden sich unter [OGC02b].

2.2.5 Filter Encoding Implementation Specification

Einige in der WFS Spezifikation beschriebenen Schnittstellen verwenden sogenannte *Filter*, um eine Auswahl aus einer Menge von Geo-Objekten treffen zu können. Die *Filter Encoding Specification* [OGC01c] beschreibt die Kodierung dieser Filter, die auf der Common Catalog Query Language [OGC02c] basieren, mit Hilfe von XML zur systemunabhängigen Repräsentation einer (Datenbank-)Abfrage (engl. Query) [OGC01c, S.1]. Laut Definition wird jeder gültige Ausdruck, der sich aus den in der Spezifikation beschriebenen Elementen zusammensetzt, als Filter bezeichnet [OGC01c, S. 9]. Diese Elemente lassen sich in vier Gruppen zusammenfassen:

- *Räumliche Operatoren* (engl. *Spatial Operators*)

  Diese Operatoren definieren eine Reihe von räumlichen Beziehungen, die zur Verschneidung von zwei geometrischen Argumenten verwendet werden können. Es wird geprüft, ob eine in GML kodierte Geometrie und eine durch ihren

Namen bestimmte geometrische Eigenschaft einer Menge von Geo-Objekten, die durch den Operator festgelegte räumliche Beziehung erfüllen. Der `<BBox>`-Operator überprüft zum Beispiel, ob die genannte Eigenschaft in irgend einer Weise mit einem umschließenden Rechteck (engl. Bounding Box) interagiert [OGC01c, S.9]. Weitere Operatoren siehe [OGC01c, Kapitel 8].

Das folgende Beispiel zeigt einen Filter, der den `<BBox>`-Operator verwendet. `<PropertyName>` bestimmt die (geometrische) Eigenschaft, wohingegen `<gml:Box>` das Rechteck definiert:

```
<Filter>
  <BBox>
    <PropertyName>Geometry</PropertyName>
    <gml:Box>
      <gml:coordinates>13.09,31.58 35.54,42.81</gml:coordinates>
    </gml:Box>
  </BBox>
</Filter>
```

- *Logische Operatoren* (engl. *Logical Operators*)

  Logische Operatoren werden zur Kombination von bedingten Ausdrücken verwendet. Zur Auswahl stehen `<And>`, `<Or>` und `<Not>`. Bei `<And>` wird der gesamte Ausruck wahr, wenn beide Teilausdrücke wahr sind. Bei `<Or>` gibt der Operator wahr zurück, wenn einer der Teilausdrücke wahr ist. `<Not>` kehrt den Wahrheitswert ins Gegenteil. Beispiel:

  ```
  <Filter>
    <Or>
      <PropertyIsEqualTo>
        <PropertyName>FIELD1</PropertyName><Literal>10</Literal>
      </PropertyIsEqualTo>
      <PropertyIsEqualTo>
        <PropertyName>FIELD1</PropertyName><Literal>20</Literal>
      </PropertyIsEqualTo>
    </Or>
  </Filter>
  ```

- *Vergleichsoperatoren* (engl. *Comparison Operators*)

  Elemente dieser Gruppe werden zur Auswertung von mathematischen Vergleichen zwischen zwei Argumenten verwendet. Unter anderem werden die standarmäßigen Vergleichsoperatoren `=,<,>,>=,<=,<>` (engl. Simple Comparisons) unterstützt. Beispiel:

```
<Filter>
  <PropertyIsLessThan>
    <PropertyName>DEPTH</PropertyName><Literal>30</Literal>
  </PropertyIsLessThan>
</Filter>
```

- *Eindeutige Bezeichner für Geo-Objekte* (engl. *Feature Identifiers*)

  Das Element `<FeatureId>` kann verwendet werden, um eine Menge von Geo-Objekten mit Hilfe eines eindeutigen Bezeichners (ID), der vom WFS vergeben wird, auszuwählen. Beispiel:

  ```
  <Filter>
    <FeatureId fid="TREESA_1M.1234"/>
  </Filter>
  ```

2.3 ArcIMS 4.0.1 von ESRI

2.3.1 Überblick

Der Internet Map Server ArcIMS der Firma ESRI ist ein Beispiel für ein Internet GIS, mit dem Geodaten in Form digitaler Karten oder interaktiver Anwendungen zentral aufbereitet werden können und für andere Nutzer sowohl in einem Intranet als auch über das Internet zugänglich gemacht werden können [ESRI03a].

ArcIMS basiert auf dem Drei-Schichten Modell wie in Abschnitt 2.1.3 beschrieben. Eine schematische Darstellung der Architektur zeigt Abbildung 2.5.

- Die *Präsentationsschicht* besteht aus den ArcIMS Viewern. Diese Clients stellen eine Benutzeroberfläche bereit und ermöglichen den interaktiven Zugriff auf Karten und Geodaten. Spezielle Werkzeuge stellen einfache Funktionen zur Anzeige, Analyse und Verarbeitung der Daten bereit.

- Die *Verarbeitungsschicht* enthält die Komponenten, die zur Verarbeitung von Client-Anfragen und zur Produktion von Karten benötigt werden. Sie beinhaltet die ArcIMS Application Server Connectoren, den ArcIMS Application Server und den ArcIMS Spatial Server. Der Spatial Server bietet verschiedene Dienste zur Kartenerstellung an. Mit dem ArcIMS Management (siehe Abbildung 2.5) werden zusätzlich Werkzeuge zur Administration von ArcIMS bereitgestellt. Der Administrator ermöglicht die Verwaltung der ArcIMS Services und des Spatial Servers, mit dem Author lassen sich Karten zusammenstellen und der Designer bietet eine Möglichkeit zur interaktiven Erstellung der ArcIMS Viewer.

- Die *Datenschicht* beinhaltet alle Datenquellen, die von ArcIMS unterstützt werden.

Neben der von ESRI bereitgestellten ArcIMS-Software werden zusätzliche Softwarekomponenten benötig, die nicht im Lieferumfang enthalten sind.

- *Webserver:* Ein Webserver nimmt HTTP-Anfragen der Clients entgegen und gibt diese an Anwendungen weiter. Die Antwort gibt der Webserver dann wieder an den Client zurück. ArcIMS unterstützt unter anderem den Apache Web Server, den Microsoft Internet Information Server (IIS) und Sun One (Iplanet).

- *Java Virtual Machine:* Viele ArcIMS Komponenten, wie zum Beispiel der Java Connector und der Servlet Connector, basieren auf Java 2 und benötigen daher eine Java Virtual Machine (JVM). Sie stellt eine API zur Verfügung, die zum Ausführen einiger ArcIMS-Komponenten benötigt wird. Eine JVM ist entweder in der Java Runtime Environment (JRE) oder im Java Development Kit (JDK) enthalten.

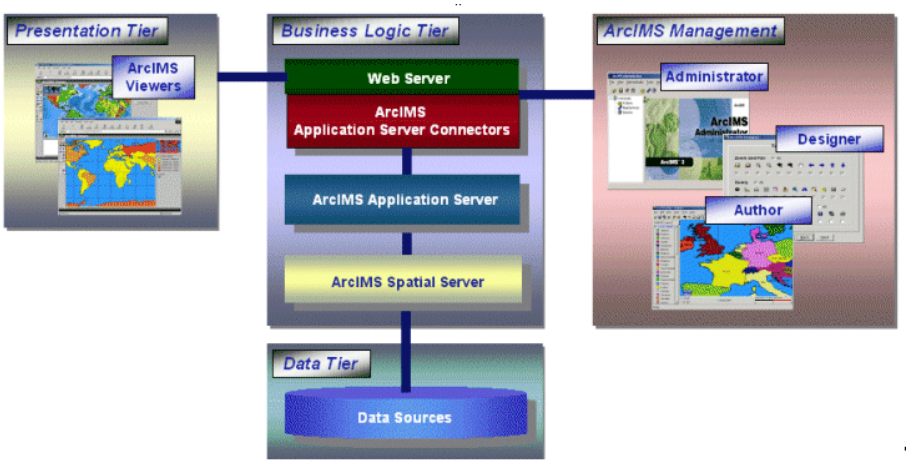


Abbildung 2.5: ArcIMS-Architektur (aus [ESRI03c, S. 1]).

- *Servlet Engine:* Eine Servlet Engine ist eine Erweiterung der JVM und verbindet mit ihrer Servlet API die JVM mit dem Webserver, der damit die Verwendung von Servlets unterstützt. Servlets sind Java-Klassen, die die Funktionalität des Webservers erweitern, so dass Anfragen ausgewertet und Antworten, zum Beispiel in Form von statischen HTML Seiten, formuliert werden können. Servlet Connector und Java Connector benötigen diese Servlet Engine, da sie auf Servlets basieren. ArcIMS unterstützt u. a. Tomcat von Apache [ASF03].

Die drei Schichten werden nun in den folgenden Abschnitten detaillierter beschrieben.

2.3.2 Die Verarbeitungsschicht

Abbildung 2.6 auf Seite 16 zeigt den Weg, den Request und Response innerhalb der Verarbeitungschicht nehmen. Die Anfrage eines Clients an ArcIMS wird zuerst vom Webserver entgegen genommen. Da dieser aber nicht über GIS-Funktionalitäten verfügt, muss die Anfrage an einen Map Server, den Spatial Server, weitergegeben werden.

ArcIMS kann so konfiguriert werden, dass sich Webserver und Spatial Server auf unterschiedlichen Rechnern befinden. Weiterhin können zusätzliche Spatial Server zum System hinzugefügt werden. Zur zentralen Verwaltung dieser komplexen Anwendung wird daher ein Application Server benötigt. Webserver und Application Server werden mit Hilfe der Application Server Connectoren verbunden. Diese bauen für jede Anfrage eine Socket Connection zum Application Server auf. Dieser wiederum übergibt die Anfrage zur Weiterverarbeitung an einen Spatial Server, der

nun die gewünschte Karte generieren kann. Die Antwort an den Client nimmt den gleichen Weg aber in entgegen gesetzter Richtung.

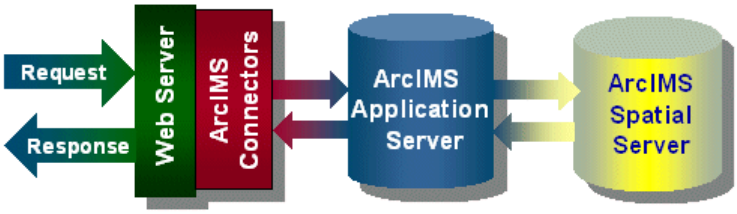


Abbildung 2.6: Die Verarbeitungschicht (aus [ESRI03k]).

Zu beachten ist, dass alle Anfragen, die den Application Server erreichen, sowie die Antworten des Spatial Servers, die den Application Server verlassen, in ArcXML verfasst sind. Die Kommunikation zwischen Application Server und Spatial Server erfolgt daher ebenfalls ausschließlich in ArcXML.

ArcIMS Spatial Server

Der Spatial Server ist die Basis des ArcIMS. Er bietet die funktionellen Voraussetzungen für die Erzeugung von Karten und die Bereitstellung anderer Daten. Karten und Daten können in verschieden Formaten für die Zurücksendung an den Client zusammengestellt werden. Dazu stehen Komponenten mit unterschiedlichen Funktionalitäten zur Verfügung, die jeweils einen Servertyp innerhalb des ArcIMS Spatial Servers darstellen.

- *Image Server:* Dieser Server erzeugt Rasterbilder von Karten als JPG, GIF oder PNG.
- *Feature Server:* Stellt Vektordaten als Binärstrom zur Verfügung.
- *Query Server:* Liefert Attributdaten anhand von Suchkriterien.
- *Geocode Server:* Führt Georeferenzierungen durch, um zum Beispiel Adressen in einer Karte lokalisieren zu können.
- *Extract Server:* Ausgewählte Daten werden als Shapefile, einem Format zur Speicherung von Vektordaten, an den Client zurück geliefert.
- *Metadata Server:* Dient als Speicher für Metadaten.
- *ArcMap Server:* Erzeugt Rasterbilder von Karten auf Grundlage von ArcMap-Dokumenten, einem proprietären Datenformat von ESRI.

- *Route Server:* Routenberechnung zwischen zwei oder mehreren Punkten. Dieser Server ist, wie der Tracking Server, eine kostenpflichtige Erweiterung.
- *Tracking Server:* Diese Erweiterung verarbeitet dynamische Ereignisse, wie zum Beispiel Fahrzeugpositionen oder Schadstoffwerte, in Echtzeit und kombiniert sie mit anderen ArcIMS-Diensten.

Image Server, Feature Server, Metadata Server, ArcMap Server und Route Server sind öffentlich, das heißt auf sie kann mit Hilfe einer ArcIMS Schnittstelle zugegriffen werden. Alle restlichen Server sind privat und werden vom Spatial Server automatisch verwaltet. Für sie gibt es daher keine Benutzerschnittstelle. Des Weiteren wird der Spatial Server durch folgende Komponenten komplettiert:

- *Weblink:* Stellt die Kommunikationsverbindung zwischen Application Server und Spatial Server dar.
- *XML Parser:* Analysiert die Syntax der ArcXML-Anfragen.
- *Data Access Manager:* Stellt eine Verbindung zwischen dem Spatial Server und jeder anderen Datenquelle her.

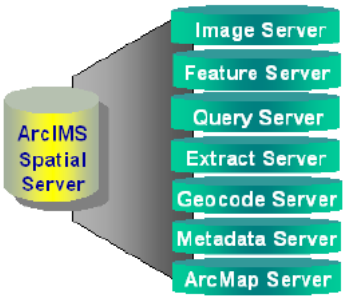


Abbildung 2.7: Servertypen des Spatial Servers (aus [ESRI03c, S. 6]).

Die Funktionalitäten des Spatial Servers können nur mit Hilfe spezieller Dienste genutzt werden. Ein ArcIMS Dienst ist ein Prozess, der auf einer oder mehreren Instanzen eines Servertyps läuft. Eine Instanz ist ein Thread, der jeweils eine Anfrage gleichzeitig bearbeiten kann. Standardmäßig werden jedem Servertyp, wie zum Beispiel dem Image Server, zwei Instanzen zugeordnet, je nach Bedarf können aber Instanzen hinzu gefügt oder entfernt werden.

Bei der Erstellung eines Image oder Feature Dienstes wird eine Konfigurationsdatei (Map Configuration File) benötigt, die Informationen über die Zusammenstellung der Karte, wie zum Beispiel Layout, Signaturenwahl und maßstabsabhängige Darstellung der Themen, enthält. In ArcIMS wird zwischen folgenden vier Diensten unterschieden:

- *Image Dienste* (engl. *Image Services*): Ein Image Dienst benutzt den Image Server, um auf Anfrage Karten in Form von Rasterbildern zu erstellen, wobei er internen Zugriff auf Query, Geocode und Extract Server hat. Das Ergebnis wird dann an den Client gesendet.
- *Feature Dienste* (engl. *Feature Services*): Dieser Dienst verwendet den Feature Server um gebündelte Daten an den Client zu übertragen. Feature Dienste können je nach Art der Anfrage ebenfalls auf Query Server, Geocode Server und Extract Server zugreifen.

- *ArcMap Image Dienste* (engl. *ArcMap Image Services*): Anders als bei Feature Diensten und Image Diensten wird bei der Erstellung eines ArcMap Image Dienstes ein ArcMap-Dokument benötigt. Der Dienst generiert Karten mit Hilfe des ArcMap Servers. In den diesen Server ist bereits eine Art Query Server integriert, darüber hinaus fehlen allerdings Geocode- und Extract-Funktionen.
- *Metadaten Dienste (Metadata Services):* Metadaten Dienste nutzen den Metadata Server um in einem Metadatenspeicher nach Dokumenten zu suchen.

Um die Redundanz zu gewährleisten oder die Performance zu erhöhen kann, die Last der Anfragen auf mehrere Spatial Server verteilt werden. Diese befinden sich entweder auf derselben Maschine oder laufen auf unterschiedlichen Systemen (siehe Abbildung 2.8). Um Dienste auf mehreren Spatial Server und unterschiedlichen

Abbildung 2.8: Verteilungsmöglichkeiten von Spatial Servern (aus [ESRI03c, S. 9]).

Maschinen verwalten zu können, hat ESRI das Konzept der *virtuellen Server* (engl. *Virtual Server*) entwickelt. Ein virtueller Server fasst alle Instanzen des gleichen Servertyps eines oder mehrerer Spatial Server zusammen (siehe Abbildung 2.9). Die zentrale Verwaltung mehrerer Server verbessert gleichzeitig die Zuverlässigkeit

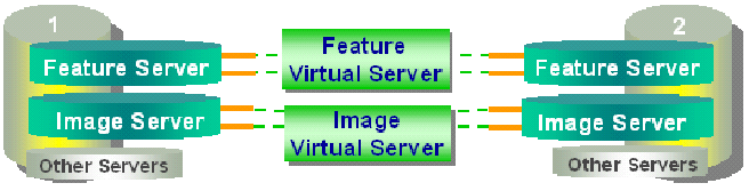


Abbildung 2.9: Gruppierung von Servertypen zu virtuellen Servern (aus [ESRI03c, S. 10]).

von ArcIMS. Falls einer der Server ausfällt werden automatisch die Dienste anderer Server, die vom selben Virtual Server verwaltet werden, verwendet.

ArcXML als Kommunikationsmittel

Die Meta-Sprache ArcXML ist eine von ESRI entwickelte Ausprägung von XML, die dem strukturierten Informationsaustausch dient und vollständig XML konform ist.

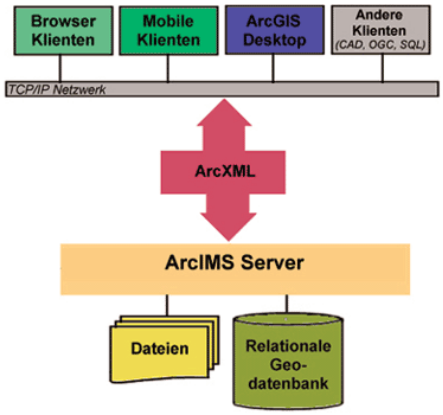


Abbildung 2.10: ArcXML als Kommunikationsmittel zwischen den unterschiedlichen Clients und dem ArcIMS Server (aus [ESRI03b]).

Sie wird einerseits zur Kommunikation zwischen Clients, wie den ArcIMS Viewern, und dem Webserver eingesetzt (siehe Abbildung 2.10), wodurch allen Clients, die XML unterstützen, die Verwendung des ArcIMS ermöglicht wird [ESRI03b]. Andererseits wird ArcXML auch zur internen Kommunikation zwischen Connectoren, Application Server und Spatial Server eingesetzt (siehe Abbildung 2.11) sowie zur Konfiguration von Diensten.

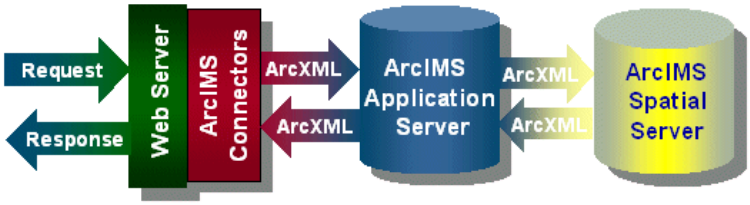


Abbildung 2.11: ArcXML als Kommunikationsmittel zwischen Connectoren, Application Server und Spatial Server (aus [ESRI03k]).

Die Elemente und Attribute von ArcXML stellen die Struktur für folgende Anwendungen bereit:

- *Map Configuration Files:* Sie legen Gestaltung und Inhalt von Karten fest. Dazu gehört zum Beispiel die Auflistung der Ebenen und die Symbolisierung der Kartenelemente. Diese Dateien werden zum Anlegen von Image Diensten und Feature Diensten auf dem Spatial Server benötigt.
- *Metadata Configuration Files:* Diese Dateien enthalten Informationen über Metadaten und dienen zur Unterstützung der Metadaten Dienste.

- *Anfragen:* Sie werden an ArcIMS gesendet, um Karten, Attributdaten oder Metadaten abzufragen.
- *Antworten:* Sie enthalten Informationen für den anfragenden Client.

Für eine ausführliche Beschreibung aller ArcXML-Elemente sei auf [ESRI02e] verwiesen. Detaillierte Beispiele für Anfragen und Antworten sind ebenfalls in Abschnitt 4.2 zu finden.

ArcIMS Application Server und ArcIMS Application Server Connectoren

Der Application Server handhabt die (Last-) Verteilung der ankommenden Anfragen auf die einzelnen Spatial Server. Er katalogisiert, welche Dienste auf welchem ArcIMS Spatial Server laufen und kann daher Anfragen dem zuständigen Dienste zuordnen. Vom Application Server können nur Anfragen verarbeitet werden, die in ArcXML verfasst sind.

Zur Kommunikation mit den Clients und zur Unterstützung der verschiedenen serverseitigen Technologien wie zum Beispiel Java Server Pages (JSP), Active Server Pages (ASP) oder ColdFusion, werden die so genannten Application Server Connectoren benötigt. Abbildung 2.12 zeigt die verschiedenen Sprachen der Clients, die von speziellen Connectoren in ArcXML übersetzt werden. Abschnitt 4.1 befasst sich eingehender mit den Funktionsweisen und Technologien der Connectoren, so dass hier nicht weiter darauf eingegangen werden soll.

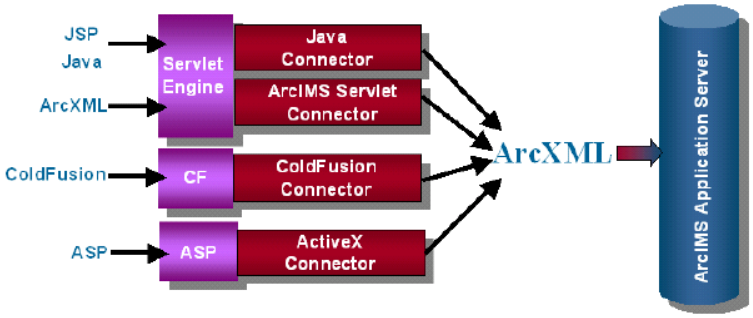


Abbildung 2.12: Die Application Server Connectoren (aus [ESRI03c, S. 5]).

2.3.3 Die Datenschicht

In der Datenschicht befinden sich die Datenquellen, die von ArcIMS verwendet werden können. Abhängig vom gewählten Dienst werden unterschiedliche Datenformate unterstützt. ArcIMS kann sowohl Vektordatenformate wie z. B. Shapefiles, als auch

diverse Rasterdatenformate wie z. B. JPG, TIFF oder PNG zur Kartenerstellung nutzen. Darüber hinaus können auch Daten aus Geodatenbanken wie ArcSDE integriert werden. Für detaillierte Informationen siehe [ESRI03c, S. 11-13].

2.3.4 Die Präsentationsschicht

Die Präsentationsschicht beinhaltet die ArcIMS Vierwer. Diese Clients, die in einem Web-Browser dargestellt werden, dienen unter anderem zur Anzeige von Karten und der Abfrage von Attributdaten. Sie sind im Lieferumfang von ArcIMS enthalten und können entweder sofort eingesetzt werden oder in ihrer Gestaltung und Funktionalität an eigene Bedürfnisse angepasst werden. Es kann zwischen zwei Java Viewern und einem HTML Viewer gewählt werden, die jeweils unterschiedliche Funktionalitäten aufweisen.

- *HTML Viewer:* Der HTML Viewer ist ein auf HTML/DHTML und JavaScript basierender Medium Client, der jeweils nur einen Image oder ArcMap Image Dienst gleichzeitig darstellen kann. Feature Dienste werden nicht unterstützt. Im Gegensatz zu den Java Viewern bietet er eine eingeschränkte Funktionalität, kann dafür aber sehr stark an die Bedürfnisse des Anwenders angepasst werden.

- *Java Viewer:* Die Java Viewer unterstützen Image, ArcMap Image und Feature Dienste. Einige Funktionalitäten werden auf Client-Seite ausgeführt, so dass von einem Medium Client gesprochen werden kann. Die Viewer verwenden Java 2 Applets zur Verarbeitung der Anfragen und zur Anzeige von Informationen, weshalb ein Java Plug-In auf Clientseite benötigt wird.

  Es gibt zwei Versionen des Java Viewers: Die Funktionen des *Java Custom Viewers* sind frei wählbar, während Benutzeroberfläche und Funktionalitäten des *Java Standard Viewers* bereits fest stehen und nicht anpassbar sind.

- *Andere Clients:* Mit Hilfe der ArcIMS Connectoren können weitere, auf Anwender und Anwendung abgestimmte Clients, erstellt werden. Dies können Thin Clients auf Basis von ASP, Cold Fusion und Java Server Pages sein oder auch Mobile Clients wie zum Beispiel ArcPad.

2.4 Das Berichtssystem der LfU

2.4.1 Überblick

Das Berichtssystem der Landesanstalt für Umweltschutz (LfU) basiert technisch auf dem Produkt disy Cadenza der Karlsruher Firma disy Informationssysteme GmbH. Cadenza ist eine Plattform zur Integration und Analyse von Geschäfts- und Geodaten, die aus unterschiedlichen Datenbanken, Geoinformationssystemen und Data Warehouses zusammengeführt werden können.

Abbildung 2.13 zeigt die fünf Hauptkomponenten des Produkts. Der *Navigator* ist die Einstiegskomponente und bietet einen Überblick über die vorhandenen Datenbestände und deren Visualisierungsmöglichkeiten (Informationssichten). Die restlichen Komponenten werden zur Ansicht, Ausgabe und zur Erzeugung neuer Informationssichten verwendet. Mit dem *Selektor* können bestimmte Informationen mit Hilfe von Kriterien aus den Datenbeständen herausgefiltert werden, die dann in Form von Diagrammen, in einer Karte oder als Tabelle dargestellt werden. Der *Visualizer* ermöglicht die Darstellung der Diagramme am Bildschirm, während *GISterm* die Erzeugung interaktiver Karten übernimmt, falls Daten mit geografischem Bezug vorliegen. Der *Reporter* bietet letztendlich die Möglichkeit Tabellen, Karten und Diagramme mit weiteren Abbildungen und Text zu versehen und diese in verschiedenen Dokumentenformaten zu speichern, so dass sie weitergeleitet oder ausgedruckt werden können. Quelle: [Disy03a, S. 3f]

Abbildung 2.13: Komponenten von Cadenza.

Im Rahmen dieser Diplomarbeit wurde hauptsächlich mit dem Modul *GISterm* gearbeitet. Es basiert auf GIStermFramework, das im Folgenden nun genauer beschrieben wird.

2.4.2 GIStermFramework

„GIStermFramework ist ein Anwendungsframework und eine generische Klassen- und Komponentenbibliothek für den Zugriff und die Visualisierung von Daten mit Raumbezug“ [Disy02, S. 9]. Es ist in Java™ geschrieben und hat damit einen ob-

jektorientierten Ansatz, so dass GISterm aus Objekten der Klassen des Frameworks gebildet wird.

Aufgrund der genannten Eigenschaften kann GIStermFramework in zwei Anwendungsszenarien eingesetzt werden: Cadenza nutzt das Framework als Komponentenbibliothek mit dessen Hilfe GISterm in die Anwendung integriert werden kann. Auf der anderen Seite ist GIStermFramework jedoch primär als Anwendungsframework ausgelegt, so dass das System eine fertig konfigurierte Rahmenanwendung besitzt, die jedoch um anwendungsspezifische Funktionen erweitert werden oder an bestimmte Anwendungsfälle angepasst werden kann.

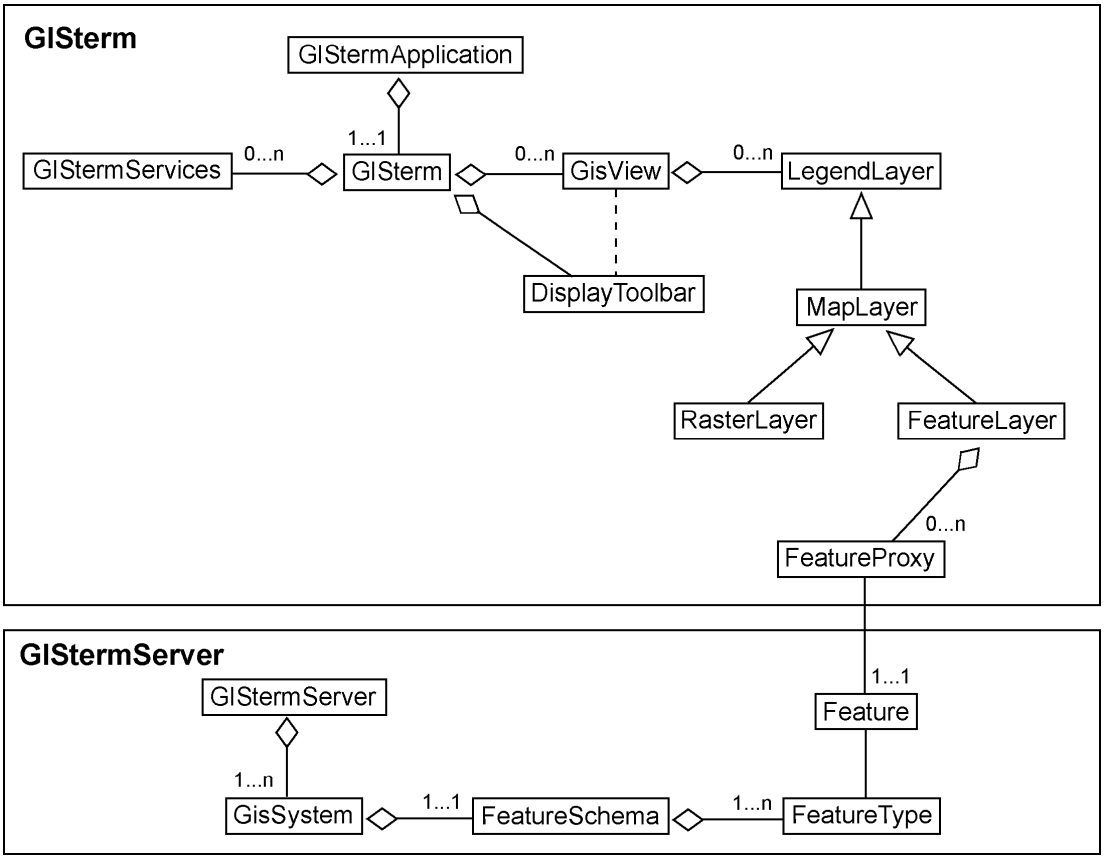


Abbildung 2.14: Vereinfachtes Klassendiagramm von GIStermFramework.

GIStermFramework hat eine für viele GIS-Anwendungen typische Drei-Schichten Architektur (siehe Abschnitt 2.1.3). Die Clientkomponente *GISterm* ist verantwortlich für die Präsentation der Daten und stellt eine GUI zur Verfügung. Die *GIStermServer* genannte Serverkomponente realisiert Anfrage- und Aufbereitunsmanagement für den Zugriff auf Geodatenbanken und andere Datenquellen. GISterm kommt jedoch auch ohne Server aus, wenn lediglich mit Dateidaten gearbeitet wird.

Abbildung 2.14 zeigt in einem vereinfachten Klassendiagramm wichtige Komponenten von Client und Server sowie deren interne Strukturierung. Für eine genaue Beschreibung der einzelnen Klassen und Methoden und deren Schnittstellendefinition sei auf die zugehörige Javadoc[1] verwiesen, die unter [Disy03b] zu finden ist. Die Klasse `GISterm` realisiert die Rahmenanwendung, für die `GisTermApplication` mit ihrer `main()`-Methode die Basis für die Nutzung als Java-Applikation bereitstellt. `GISterm` bildet die Basis der Anzeigekomponenten und der grafischen Benutzeroberfläche des Frameworks. Dazu gehört die Kartenkomponente (`GisView`), die in einen Grafikbereich zur Darstellung raumbezogener Daten in Form von Karten und einen Legendenbereich unterteilt ist, sowie eine Werkzeugleiste (`DisplayToolbar`). Abbildung 2.16 zeigt die Grafikoberfläche von GISterm.

`GISterm` verwaltet nicht nur die grafischen Elemente des Clients, sondern ebenso die gesamte Funktionalität. Sie steht in einer generischen Form zur Verfügung, das heißt, es werden alle weiteren Subkomponenten und Klassen vorkonfiguriert verbunden, was die Nutzung der Hauptkomponente wesentlich vereinfacht.

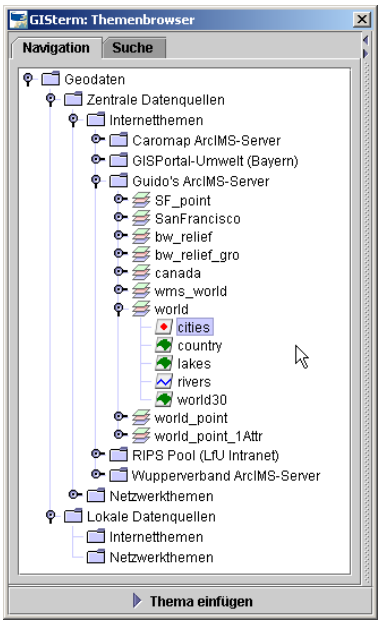


Abbildung 2.15: Darstellung von Metadaten im Navigator.

Karten (Klasse `GisView`) bestehen aus übereinander gelegten Ebenen (Klasse `MapLayer`). Grafische Objekte der Karte, die auf Vektordaten basieren, werden in `FeatureLayer`-Objekten verwaltet, `RasterLayer`-Objekte enthalten dagegen Rasterbilder. `FeatureLayer` bestehen aus einer Kollektion von `FeatureProxy`-Objekten, die die grafische Repräsentation von Geo-Objekten darstellen.

Soll GISterm um anwendungsspezifische Funktionalitäten erweitert werden, so kann die Schnittstelle `GisTermService` genutzt werden.

Während der Client die Visualisierung von Geodaten übernimmt, verwaltet der Server die damit verknüpften Geometrie- und Sachdaten. Für jedes graphische Element in `GISterm` existiert daher ein entsprechendes `Feature`-Objekt (Geo-Objekt) in `GIStermServer`. Jedes Feature ist einem bestimmten `FeatureType`-Objekt zugeordnet, das wiederum zu einem `FeatureSchema`-Objekt gehört.

Darüber hinaus ermöglicht `GIStermServer` den Zugriff auf Geodatenserver, indem für jede Datenquelle eine Instanz der Klasse `GisSystem` erzeugt wird. Für jedes dieser Objekte benötigt der Server eine Beschreibung der Datenbestände. Diese *Metadaten* werden

[1] Javadoc ist ein Hilfsprogramm, das Deklarations- und Dokumentationskommentare im Quellcode analysiert und HTML-Seiten produziert, die die Klassen, Schnittstellen, Methoden und Felder beschreiben.

in Objekten der Klasse `FeatureSchema` gespeichert. Der Client kann auf die Objekte zugreifen und so die zur Verfügung stehenden Datenquellen im Navigator, der bei einer Nutzung von GISterm als Anwendungsframework auch Themenbrowser genannt wird, auflisten (siehe Abbildung 2.15). Quelle: [Disy02, S. 9-11]

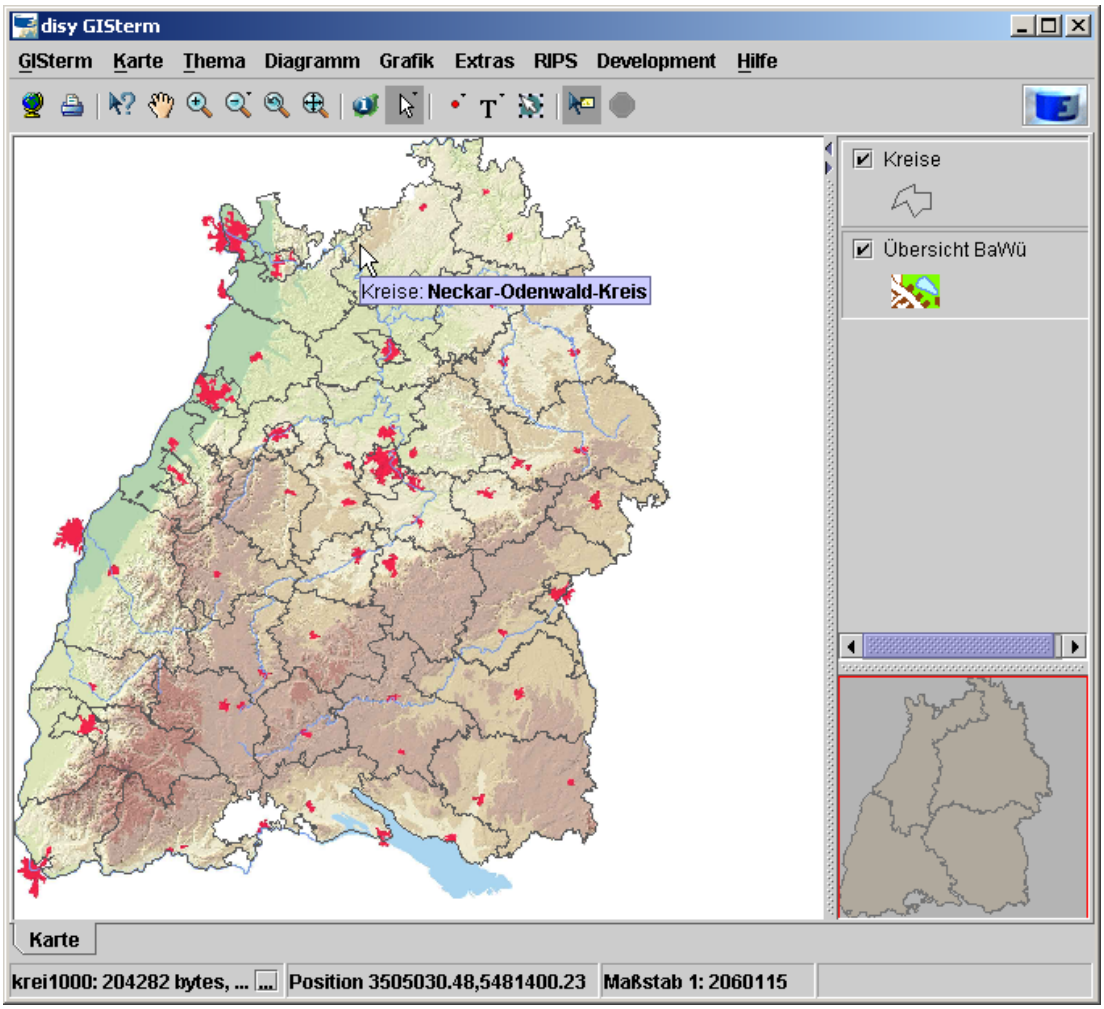


Abbildung 2.16: Der GISterm Client.

3 Gründe für die Einbindung von ArcIMS-Diensten in GISterm

In GISterm werden Geobasisdaten, wie Topografische Karten und Orthofotos oder die ALK (Automatisierte Liegenschaftskarte) verwendet, um sie als Hintergundebenen mit vektorbasierten Geodaten überlagern zu können. Der Anwender kann so räumliche Zusammenhänge besser erschließen. Diese Hintergrunddaten liegen als blattschnittfreie Rasterdaten und ESRI Shapefiles (ALK) dateibasiert vor oder werden von einer SDE (Spatial Database Engine) verwaltet. Mit Hilfe von ArcIMS können diese Geodaten über das Internet zugänglich gemacht werden, da sowohl SDE als auch dateibasierte Rasterdaten und Shapefiles als Datenquelle unterstützt werden.

GISterm unterstützt als Datenquelle, neben vielen dateibasierten Datenformaten, zwar auch Geodatenserver wie SDE, jedoch würde die Einbindung von ArcIMS eine einfache Möglichkeit eröffnen über das Internet auf die Daten anderer Anbieter zugreifen zu können. Zusätzlich könnte so eine zentrale Verwaltung der Daten realisiert werden, die nicht auf ein Netzwerk beschränkt ist, wie das bei der SDE der Fall ist. Dies wäre vor allem für weniger oft genutzte Daten von Vorteil, die sonst nur unnötig Speicherplatz in Anspruch nehmen würden.

Weiterhin werden viele Karten an der LfU mit dem Produkt ArcMap von ESRI erstellt und in einem proprietären Format, den ArcMap Projektdateien, abgespeichert. Diese Projekte können eine Zusammenstellung vieler Ebenen aus Raster und Vektordaten sein, so dass oftmals eine komplexe Legende zu Stande kommt, die unter Umständen sogar maßstabsabhängig sein kann. Ein schönes Beispiel dafür ist die Badegewässerkarte der LfU.

In GISterm können die mit ArcMap erstellten Karten nicht importiert werden. Eine Darstellung der Inhalte der ArcMap Projekte in GISterm kann nur erreicht werden, indem die einzelnen Ebenen mühsam aus den Datenbeständen des Berichtssystems zusammengesucht werden.

ArcIMS bietet mit dem ArcMap Server die Möglichkeit ArcMap Projekte zu veröffentlichen, ohne dass dabei die Inhalte neu zusammengestellt werden müssen. ArcMap Projekte können zur Erstellung eines ArcMap Image Dienstes verwendet werden.

Vorteile für die Einbindung von ArcIMS-Diensten in das Berichtssystem ergeben sich hierbei vor allem aus der verbesserten Performanz im Vergleich zum disy Map Server. Das Prinzip der virtuellen Server innerhalb des ArcIMS (siehe Abschnitt

2.3.2, Seite 18) regelt die zentrale Verwaltung der Dienste und ermöglicht so eine Skalierung des Spatial Servers, so dass ArcIMS dem Anfrageaufkommen angepasst werden kann. Die Verteilung der Anfragen auf mehrere Server, die zudem noch auf unterschiedlichen Maschinen laufen können, verbessert die Belastbarkeit nicht nur bei vielen Anfragen, sondern auch bei großen Datenmengen enorm. Besonders Orthofotos und Topografische Karten, aber auch größere Shapefiles erfordern viel Rechenleistung. Diese Form der Skalierung des disy Map Servers ist in diesem Umfang in Zukunft ebenfalls vorgesehen, doch kann zur Zeit die Einbindung von ArcIMS-Diensten in das Berichtssystem noch zu einer Verbesserung bei der Geschwindigkeit der Kartendarstellung und somit zu mehr Benutzerfreundlichkeit führen.

Ein weiterer Vorteil von ArcIMS gegenüber dem Berichtssystem sind die besseren Visualisierungsmöglichkeiten von Vektordaten. In ArcIMS werden sogenannte *ArcIMS Renderer* zur Erzeugung von Vektorgrafiken verwendet, indem diese den Zeichenschlüssel für die Kartenelemente festlegen [ESRI02e, S. 39-58]. Vorteile ergeben sich dabei vor allem bei der Beschriftung von Geo-Objekten durch *Labelrenderer*. Zum Beispiel wird die Drehung von Beschriftungen in GISterm nicht unterstützt, was bei der ALK zu einer unschönen und unübersichtlichen Kartendarstellung führt, da Straßennamen und Hausnummern nicht parallel zum Verlauf der Straße angeordnet werden können.

Weiterhin können für Geo-Objekte nicht nur Schriften als Label verwendet werden, sondern auch Signaturen und Symbole. Eine Besonderheit dabei ist die Vergabe von Labels für Linien, so dass zum Beispiel Straßen mit Straßenschildern für Autobahn etc. versehen werden können. Der ArcIMS Renderer *Grouprenderer* kann zudem mehrere Renderer zu einer komplex aufgebauten Signatur zusammenfassen, so dass der Karteninhalt anschaulicher wird und besser differenziert werden kann.

4 Vergleich und Bewertung der Programmierschnittstellen von ArcIMS

4.1 Analyse und Vorauswahl

Möchte ein Client die Dienste von ArcIMS in Anspruch nehmen, muss der Request, der den Application Server erreicht, in ArcXML verfasst sein. Um dies zu gewährleisten gibt es die bereits im Lieferumfang von ArcIMS enthaltenen Application Server Connectoren. Sie sind Erweiterungen des Webservers und erzeugen als Endprodukt in jedem Fall ArcXML. Gleichzeitig dienen sie als Programmierschnittstellen für Clients unterschiedlichster Programmiersprachen.

Ein Client kann die Anfrage entweder direkt in ArcXML verfassen oder die Aufbereitung der Anfrage dem Webserver überlassen. Welche Möglichkeit angewendet wird, hängt maßgeblich von der Art und den Fähigkeiten des Clients ab. Thin Clients, wie z. B. reine HTML-Clients, können kein ArcXML erzeugen, wohingegen Medium Clients oder Thick Clients ArcXML Anfragen an den Webserver schicken können.

Die Programmierschnittstellen von ArcIMS lassen sich daher in zwei Kategorien unterteilen:

1. *Connectoren für die clientseitige Erzeugung von ArcXML:*
   Für diesen Fall stellt ArcIMS einen Standardconnector bereit, der von allen ArcIMS Viewern verwendet wird. Der Servlet Connector dient lediglich zur Weiterleitung von ArcXML an den Application Server bzw. die Clients.

2. *Connectoren für die serverseitige Erzeugung von ArcXML:*
   Alle restlichen Connectoren gehören zu dieser Kategorie. Sie ermöglichen die Kommunikation zwischen Webserver und Application Server, indem sie Anfragen von Clients, die auf speziellen Sprachen basieren und nicht direkt ArcXML erzeugen, in ArcXML übersetzen und umgekehrt die Antwort des Servers in die ursprüngliche Sprache der Clients übertragen.

Im Anschluss werden nun alle für ArcIMS verfügbaren Connectoren genauer beschrieben und daraufhin überprüft, ob ihre Technologien für die Einbindung von

ArcIMS-Diensten in das Berichtssystem verwendet werden können. Hauptanforderung soll dabei zunächst die Anbindung an die Programmiersprache Java sein, da dies die Implementierungssprache von GIStermFramework ist.

4.1.1 Der Servlet Connector

Der Servlet Connector besteht aus Java-Klassen, die eine Erweiterung der Servlet Engine darstellen. Er leitet ArcXML-Anfragen an den Application Server weiter und überprüft dabei gleichzeitig, ob die XML-Dokumente wohlgeformt und gültig sind. Die Antworten des Spatial Servers gibt der Connector dann unverändert in ArcXML an die Clients zurück.

Damit ein Client diesen Connector verwenden kann, muss er die Fähigkeit zur Formulierung von ArcXML-Anfragen sowie zur Auswertung der ArcXML-Antworten besitzen. Requests in ArcXML können mit Hilfe von HTTP-POST an das Connector-Servlet übergeben werden. Um dies in einem (Browser-)Client realisieren zu können, müssen Scriptsprachen wie z. B. Java Script oder höhere Programmiersprachen wie z. B. Java eingesetzt werden. Ein Thin Client, der nur auf HTML basiert, erfüllt diese Vorraussetzungen nicht. Die von ESRI eingesetzten Viewer sind daher mindestens Medium Clients (HTML Viewer) oder sogar Thick Clients (Java Viewers).

Bewertung: GISterm kann als Thick Client bezeichnet werden, da die Anwendung selbstständig GIS-Funktionen ausführen kann ohne dabei Kontakt mit einem Server aufnehmen zu müssen. In der Java-Klassenbibliothek Version 1.4 [Sun03a] sind bereits Pakete zur Implementierung von Netzwerk-Anwendungen (`java.net`) und zur Verarbeitung und Erzeugung von XML-Dokumenten (`org.w3c.dom` und `org.xml.sax`) enthalten. Da GIStermFramework auf Java basiert und mit Hilfe von Java erweitert werden kann, bietet es daher optimale Vorraussetzungen für die Verwendung des Servlet Connectors zur Integration von ArcIMS-Diensten.

4.1.2 Java Connector

Der Java Connector ist eine Erweiterung der Servlet Engine und besteht aus Java-Klassen (Servlets), die in einer Archivdatei (.jar) zusammengefasst sind. Dieses Java-Archiv stellt eine Gruppe von JavaBeans [Sun04c] und eine JSP-Tag Bibliothek zur Verfügung, mit deren Hilfe auf der einen Seite Browser-basierte Thin Clients erzeugt werden können, die auf Servlets oder JavaServer Pages (JSP) [Sun04d] basieren. Auf der anderen Seite können die JavaBeans auch zur Einbindung von ArcIMS-Diensten in eine stand-alone Java-Applikation genutzt werden (siehe auch Abbildung 4.1).

Ein großer Vorteil des Java Connectors ist seine Plattformunabhängigkeit, so dass er mit allen Webservern, die von ArcIMS unterstützt werden, verwendet werden kann.

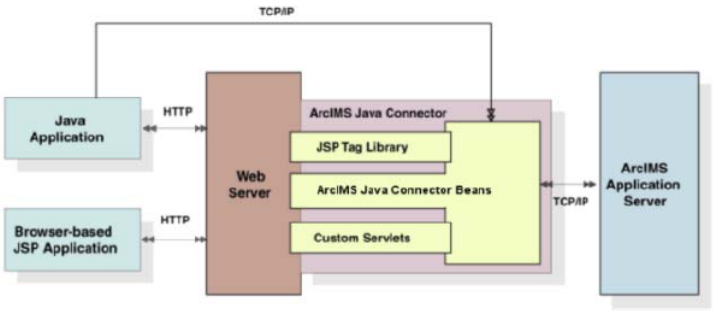


Abbildung 4.1: Aufbau des Java Connectors (aus [ESRI02b, S. 3]).

Die Basis für jede Anwendung, die mit Hilfe des Java Connectors erzeugt wird, sind die JavaBeans. JavaBeans sind wiederverwendbare Softwarekomponenten, die in Programme eingebunden werden können und „über wohldefinierte Schnittstellen miteinander kommunizieren“ [Ulle01, S. 1093]. Die Softwareentwicklung wird dadurch wesentlich beschleunigt und vereinfacht. JavaBeans werden in Pakete unterteilt, die jeweils einen Arbeitsbereich übernehmen. Informationen über die Pakete und Schnittstellen der Java Connector JavaBeans enthält das Java Connector Objektmodell, das sich auf der ArcIMS CD-ROM befindet. Die JavaBeans des Java Connectors übernehmen die Erzeugung und Verarbeitung von ArcXML und kommunizieren mit dem Application Server.

Servlets sind Programme aus Java-Klassen, die Webseiten erstellen [Ulle01, S. 974]. Dazu nutzen sie die Ein- und Ausgabeströme des Webservers, so dass die Anfragen von Clients verarbeitet werden können, um auf Anweisung dynamische HTML-Seiten zu erzeugen. Clients, die auf Servlets basieren, können die JavaBeans zur Realisierung von ArcIMS-Funktionen verwenden.

Anwendungsentwickler, die nicht vertraut sind mit objektorientierter Programmierung, können die auf den JavaBeans basierende JSP-Tag Bibliothek verwenden, um statische HTML-Seiten durch dynamische Funktionalitäten zu erweitern. Die JSP-Tags repräsentieren die Schnittstelle zum Objektmodell, so dass die Funktionen des ArcIMS genutzt werden können. JSP sind also HTML-Seiten, in die Java-Code eingebettet ist. Damit diese Seiten von einem Browser interpretiert werden können, muss der Code zuerst von einer Servlet Engine zu Servlets kompiliert werden, die dann den entsprechenden HTML-Code erzeugen.

Weiterführende Informationen zum Java Connector, dem Objektmodell und der JSP-Tag Bibliothek enthält [ESRI02b].

Bewertung: Obwohl der Java Connector auf Java basiert und mit den Java Beans vielfältige Pakete zur Nutzung der ArcIMS-Funktionen bereitstellt, so ist er für die Einbindung der ArcIMS-Dienste in das Berichtssystem nur bedingt geeignet.

Der Java Connector ist unter anderem für Browser-basierte Clients entwickelt worden, die JSP nutzen oder auf Servlets basieren. Da GISterm ein eigenständiges Programm ist und nicht in einem HTML-Browser angezeigt wird, scheidet die Verwendung von Servlets und der JSP-Tag Bibliothek aus.

Die Nutzung der JavaBeans durch GIStermFramework ist dagegen prinzipiell möglich, da die Klassen des Objektmodells leicht integriert werden könnten. Gegen ihre Verwendung spricht allerdings, dass der Java Connector kein Open Source Produkt ist und so für jede Art der Nutzung Lizenzgebühren an den Hersteller (ESRI) gezahlt werden müssen. Für die Landesanstalt für Umweltschutz wäre das kein Problem, da sie durch den Erwerb eines ArcIMS bereits eine Lizenz besitzt. Für die restlichen Kunden der Firma disy Informationssysteme, die GISterm nutzen, gilt dies jedoch nicht zwingend. Der Vorteil einer ArcIMS-Schnittstelle besteht jedoch nicht nur in der Nutzung eigener ArcIMS-Dienste, sondern auch in der Verwendung externer Dienste, die über das Internet genutzt werden können. Aus diesen Gründen ist die Realisierung der Aufgabe mit Hilfe des Java Connectors nicht empfehlenswert.

4.1.3 ActiveX Connector

Der ActiveX Connector übernimmt dieselben Aufgaben wie der Java Connector, jedoch unterstützt er mit seinem Objektmodell Clients, die auf Active Server Pages (ASP) oder Visual Basic basieren. ActiveX ist eine Microsoft-Technologie, die als Gegenstück zu Java Applets entwickelt wurde und das Ausführen von Programmcode auf Webseiten ermöglicht [Team01a]. ASP werden zur Erstellung dynamischer HTML-Seiten verwendet, indem Scriptcode in HTML eingebettet wird [Team01b].

Die Verwendung dieser Technologien hat zur Folge, dass der Connector nur unter Windows und zusammen mit dem IIS von Microsoft verwendet werden kann. Für ausführlichere Informationen zum ActiveX Connector siehe [ESRI02c].

Bewertung: Ohne weiter ins Detail gehen zu müssen, kann jetzt schon gesagt werden, dass die Verwendung des ActiveX Connectors für die Einbindung von ArcIMS-Diensten in GISterm nicht in Frage kommt. Die Nutzung von ASP ist lediglich für Browser-Clients sinnvoll. Da GISterm nicht auf Visual Basic basiert, kann auch das Objektmodell des Connectors nicht verwendet werden.

4.1.4 ColdFusion Connector

ColdFusion ist ein auf einem Webserver installierter Web-Applikationsserver, der Webseiten dynamisch erstellt. Darüber hinaus bildet er mit Hilfe von Tag-basierten Programmen Schnittstellen zwischen den verschiedenen in Webapplikationen notwendigen Softwarekomponenten, wie zum Beispiel Datenbanken.

Der ColdFusion Connector repräsentiert eine dieser Schnittstellen. Er stellt spezielle ColdFusion Tags bereit, die die Funktionen des ArcIMS unterstützen. Anfragen

vom ColdFusion Server werden vom Connector verarbeitet, bevor sie an den Application Server weitergeleitet werden. Der ColdFusion Connector ist wie der Java Connector plattformunabhängig und kann mit allen Webservern, die von ArcIMS unterstützt werden, verwendet werden. Weiterführende Informationen über diesen Connector sind in [ESRI02d] zu finden.

Die ColdFusion Technologie wurde von der Firma Macromedia entwickelt, um Webanwendungen schneller und einfacher als z. B. mit ASP oder JSP erstellen zu können. Dazu werden vor allem die von der Firma entwickelten Programme wie Dreamweaver oder ColdFusion Studio eingesetzt. Der Connector und die ColdFusion Elemente sind daher vor allem auf die Unterstüzung dieser Entwicklungstools ausgerichtet [Macr03].

Bewertung: Anders als beim ActiveX oder dem Java Connector unterstützt der ColdFusion Connector nur die Erstellung von Browser-basierten Clients, so dass eine Verwendung für GISterm nicht in Frage kommt.

4.1.5 ESRI WMS Connector

Im Lieferumfang von ArcIMS 4.0.1 ist ein Web Map Service Connector, nachfolgend ESRI WMS Connector genannt, enthalten, der WMS 1.0.0 und WMS 1.1.0 unterstützt. Dieser Connector ist Teil des Servlet Connectors und wird im Zuge der Standardinstallation von ArcIMS automatisch mitinstalliert. Er besteht aus Servlets, die WMS-konforme Anfragen von Clients in ArcXML übersetzen. Auf diese Weise wird jedem WMS-kompatiblen Client Zugriff auf Rasterbilder, die von ArcIMS Image oder ArcMap Image Diensten bereitgestellt werden, ermöglicht. Im Gegensatz dazu können die Dienste anderer WMS-kompatibler Map Server nicht in ArcIMS integriert werden, da diese von ArcIMS nicht als Datenquelle unterstützt werden (siehe [ESRI03c, S. 11-13]). ArcIMS kann also nicht als Cascading Map Server (siehe 2.1.2) genutzt werden, da die WMS-Schnittstelle nicht in beide Richtungen funktioniert.

Mit Hilfe der Capabilities-Datei (siehe Anhang A.1), können die Fähigkeiten des Connectors beurteilt werden. Daraus ist ersichtlich, dass nicht alle Schnittstellen, die in der WMS 1.1.0 beschrieben werden, implementiert sind. Der ESRI WMS Connector unterstützt zusätzlich zu den erforderlichen Schnittstellen *GetCapabilities* und *GetMap* lediglich *GetFeatureInfo*, so dass der Connector keine SLD-Fähigkeiten besitzt. Der Connector liefert Karten im JPG, GIF oder PNG Format zurück.

Bewertung: Da GISterm bereits über eine WMS-Schnittstelle verfügt, ist die Einbindung der ArcIMS-Dienste über den ESRI WMS Connector prinzipiell möglich. Die Verwendung des Connectors wäre sehr vorteilhaft, da dadurch sehr viel Programmierarbeit eingespart werden kann.

4.1.6 ESRI WMS Connector 2

Unabhängig vom ESRI WMS Connector wird auf der ESRI-Homepage ein weiterer WMS Connector zum Download angeboten [ESRI03d]. Dieser Connector, nachfolgend ESRI WMS Connector 2 genannt, unterstützt alle WMS-Versionen (WMS 1.0.0, WMS 1.1.0 und WMS 1.1.1). Der WMS Connector 2 besitzt fast die gleichen Fähigkeiten wie der ESRI WMS Connector. Er unterstützt ebenfalls *GetCapabilities*, *GetMap* und *GetFeatureInfo* und ist kein SLD-WMS. Karten können jedoch nur im JPG oder PNG-Format zurückgeliefert werden (siehe Anhang A.2).

ESRI weist extra darauf hin, dass für diesen Connector weder eine detaillierte Dokumentation noch telefonischer Support angeboten wird [ESRI03d]. Darüber hinaus lässt die Versionsnummer des Connectors (0.0.6) auf eine Beta-Version schließen.

Bewertung: Dieser Connector könnte ebenfalls durch die WMS Schnittstelle von GISterm verwendet werden.

4.1.7 ESRI WFS Connector

Ebenfalls unter [ESRI03e] kann ein Web Feature Service Connector für ArcIMS heruntergeladen werden. Dieser Connector, nachfolgend ESRI WFS Connector genannt, unterstützt WFS 1.0.0 und ermöglicht den Abruf von ArcIMS Geo-Objekten, die in GML kodiert sind und von Image Diensten oder ArcMap Image Diensten unter Verwendung ihrer Query Server (siehe Abschnitt 2.3.2) bereitgestellt werden. Der Connector besteht aus Servlets, welche die Anfragen des Clients verarbeiten. Die daraufhin vom Spatial Server bereitgestellten Geo-Objekte werden in GML verschlüsselt und zum Client übertragen.

Der Connector hat lediglich die Fähigkeiten eines Basic WFS, unterstützt also die Schnittstellen *GetCapabilities*, *DescribeFeatureType* und *GetFeature*.

Der WFS Connector ist wie der ESRI WMS Connector 2 nur als Beta-Version erhältlich (Versionsnummer 0.0.7). Daher handelt es sich auch noch nicht um ein offizielles ESRI-Produkt, so dass hierfür auch kein Support angeboten wird und keine Garantie für ein fehlerfreies Funktionieren besteht. Das ist wohl auch der Grund dafür, dass keine ausführlichen Dokumentationen existieren, so dass man bei eventuell auftretenden Problemen auf sich allein gestellt ist oder mit Hilfe von Newsgroups [ESRI03f] nach einer Lösung suchen muss.

Das Stöbern in diesen Newsgroups brachte dann den nächsten Nachteil ans Licht [ESRI03g]. Die Bearbeitungszeit, die ArcIMS und der WFS Connector benötigen, um Geo-Objekte bereitzustellen, steigt mit zunehmender Anzahl von Objekten im Vergleich zum Servlet Connector sehr stark an (siehe Abbildung 4.2).

Der Grund dafür ist, dass ArcIMS die Geo-Objekte für diese Art der Anwendung in ArcXML kodiert, der WFS Connector die Geodaten allerdings mit Hilfe von GML 2 beschreibt. Dies bedeutet, dass innerhalb eines Anfrage/Antwort-Zyklus zwei

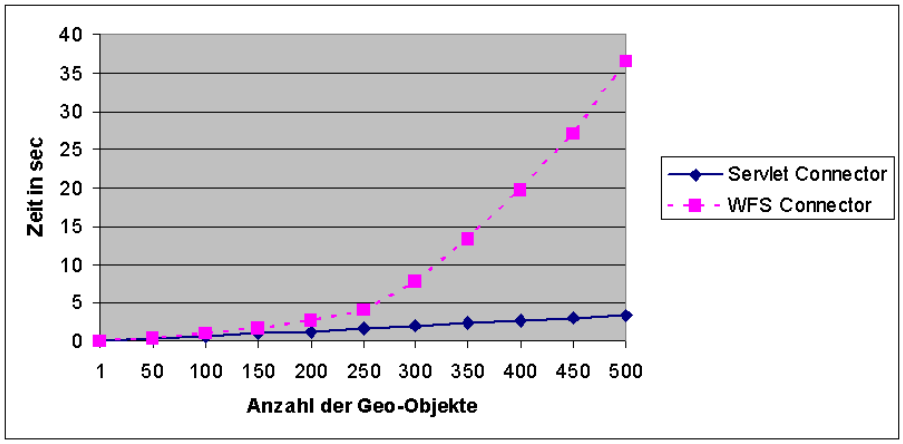


Abbildung 4.2: Vergleich der Performanz von Servlet und WFS Connector.

Transformationen stattfinden müssen: Die Anfrage muss von GML nach ArcXML übersetzt werden und die Antwort von ArcXML nach GML. Dieser Umstand führt vor allem bei größeren Datenmengen zu langen Antwortzeiten, da zusätzlich zur Bearbeitungszeit, die ArcIMS für die Erstellung von ArcXML benötigt, noch die Transformationszeiten hinzukommen.

Um die Behauptungen in der Newsgroup zu bestätigen, wurden eigene Tests durchgeführt. Die Daten für das Diagramm in Abbildung 4.2 wurden mit Hilfe eines Java-Programms ermittelt, das *GetFeature*-Anfragen (siehe Abschnitt 2.2.4) an den WFS Connector bzw. `<GET_FEATURES>`-Anfragen (siehe [ESRI02e, S. 415] oder Abschnitt 4.2.8) an den Servlet Connector sendet und dabei die Zeit misst, die zwischen Absenden der Anfrage und Eintreffen der Antwort vergeht.

Um vergleichbare Ergebnisse zu erhalten, wurde dabei jeweils der gleiche Dienst angesprochen, so dass in den Antworten dieselben Geodaten enthalten waren. Die Vergleichbarkeit wurde weiterhin durch den Abruf gleichartiger Geo-Objekte gewährleistet, da lediglich Themen mit punkthaften Objekten, die zudem alle die gleiche Anzahl von Attributen aufwiesen, angefordert wurden.

Durch Veränderung der Anfrage-Parameter bei *GetFeature*-Anfragen bzw. der Attribute von `<GET_FEATURES>` konnte die Anzahl von punkthaften Geo-Objekten, die abgerufen wurden, variiert werden. Dadurch konnte für beide Connectoren eine Messreihe durchgeführt werden. Das Ergebnis für eine bestimmte Anzahl von Geo-Objekten wurde dann jeweils aus dem Durchschnitt von 3000 Requests gebildet.

Die ermittelten Zeiten sollen keine Richtwerte darstellen, da sie stark von der Rechenleistung der von ArcIMS verwendeten CPU und von der Geschwindigkeit der Datenverbindung abhängig sind, sondern dienen lediglich zur Veranschaulichung der unterschiedlichen Performanz von WFS Connector und Servlet Connector. Die Tests wurden im Intranet der LfU durchgeführt, wobei ArcIMS auf einem Rechner mit 935

MHz CPU-Leistung installiert war.

Das Diagramm zeigt einerseits, wieviel Zeit ArcIMS normalerweise zur Erstellung der in ArcXML kodierten Geo-Objekte benötigt (Kurve des Servlet Connectors), andererseits ist ersichtlich, wieviel Zeit der WFS Connector zusätzlich benötigt, um die Transformationen durchzuführen.

Bewertung: Der Vorteil des Connectors ist, dass er Geo-Objekte in GML verschlüsselt. Da GISterm schon eine Importfunktion für GML anbietet, könnte diese zusammen mit der bereits vorhandenen WMS Schnittstelle als Vorlage für eine neue WFS Schnittstelle dienen. Es könnten dann nicht nur ArcIMS Geo-Objekte in GISterm dargestellt werden, sondern auch Objekte von anderen WFS-konformen Map Servern.

Die bei größeren Datenmengen auftretenden Wartezeiten sind natürlich ein Nachteil, jedoch kommt es selten vor, dass alle Geo-Objekte eines Themas auf einmal benötigt werden. Die Menge der Daten könnte zum Beispiel durch den aktuellen Kartenausschnitt eingeschränkt werden.

4.1.8 Zusammenfassung

Werden lediglich die Programmiersprachen, die von den Connectoren unterstützt werden, betrachtet, so eignen sich für die Nutzung von ArcIMS-Diensten durch GIStermFramework lediglich Servlet und Java Connector, die beiden WMS Connectoren sowie der WFS Connector. Mit diesen Programmierschnittstellen kann die Anbindung an die Programmiersprache Java ohne weiteres realisiert werden.

Das Objektmodell des Java Connectors würde die einfachste Lösung darstellen, wegen der Lizenzgebühren muss jedoch von einer Verwendung in GIStermFramework abgesehen werden.

Damit letztendlich einer der restlichen Connector zur Realisierung der Aufgabe ausgewählt werden kann, muss noch untersucht werden, ob die Anforderungen, die GISterm an die Einbindung der ArcIMS Dienste stellt, erfüllt werden können. Dies wird nun in den folgenden Kapiteln diskutiert.

4.2 Vergleich und Bewertung

An dieser Stelle sollen die verschiedenen Möglichkeiten der Connectoren, die im letzten Kapitel in die engere Auswahl genommen worden sind, verglichen werden. Um den Vergleichsrahmen auf Anforderungen, die für die Integration der ArcIMS-Dienste in GISterm von Bedeutung sind, einzuschränken, werden die Schnittstellen auf die Realisierung verschiedener GISterm-spezifischer Anwendungsfälle untersucht.

4.2.1 Vorgehensweise beim Testen der Connectoren

Um die Realisierbarkeit der in diesem Kapitel vorgestellten Anwendungsfälle überprüfen zu können, mussten die Funktionen der Connectoren getestet werden. Dazu wurden Anfragen an die Connectoren gestellt und überprüft, ob die Antworten ein verwertbares Ergebnis lieferten.

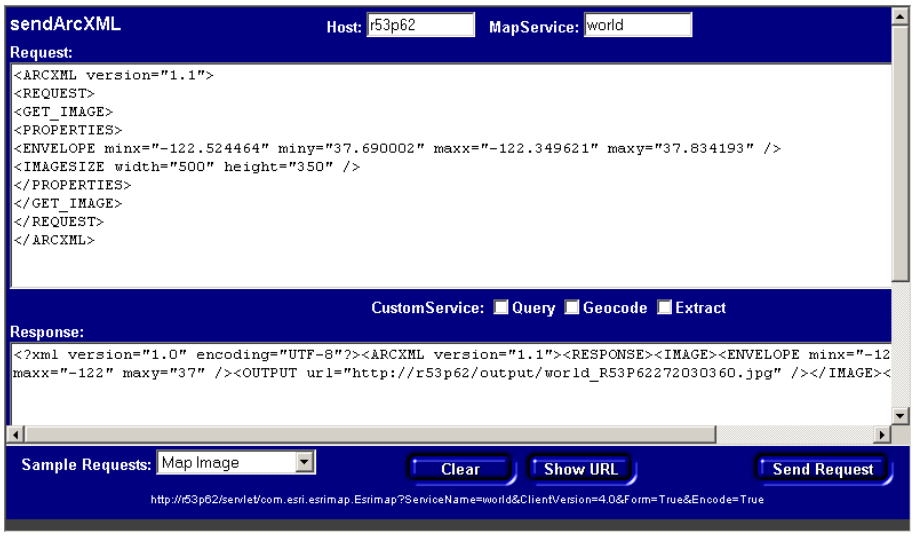


Abbildung 4.3: Client zum Senden von ArcXML-Anfragen an den Servlet Connector und zur Darstellung der Antworten.

Zum Testen der ArcXML-Anfragen wurde ein Browser-basierter Client verwendet (siehe Abbildung 4.3), der auf der Homepage von ERSI [ESRI03i] heruntergeladen wurde. Der Client sendet mit Hilfe eines Formulars, in das direkt ArcXML eingegeben werden kann, die Anfrage an den Servlet Connector und zeigt die Antwort in einem zweiten Formular an. Weiterhin lassen sich der Host des ArcIMS und der Name des Dienstes, für den der Request bestimmt ist, angeben. In der untersten Zeile des Clients kann die vollständige URL der Anfrage abgelesen werden.

Zur Übertragung der ArcXML-Anfragen an den Servlet Connector wird HTTP-POST verwendet. Das folgende Beispiel zeigt eine einfache Anfrage, mit der eine Karte angefordert werden kann:

```
<ARCXML version="1.1">
  <REQUEST>
    <GET_IMAGE>
      <PROPERTIES></PROPERTIES>
    </GET_IMAGE>
  </REQUEST>
</ARCXML>
```

Wie zu erkennen ist, enthält der Request keine Informationen darüber, welcher Dienst die Karte erstellen soll. Diese und weitere Informationen werden daher mit Hilfe von HTTP-GET an den Servlet Connector übertragen. Als Antwort auf die Anfrage gibt der Spatial Server wiederum ein ArcXML-Dokument zurück.

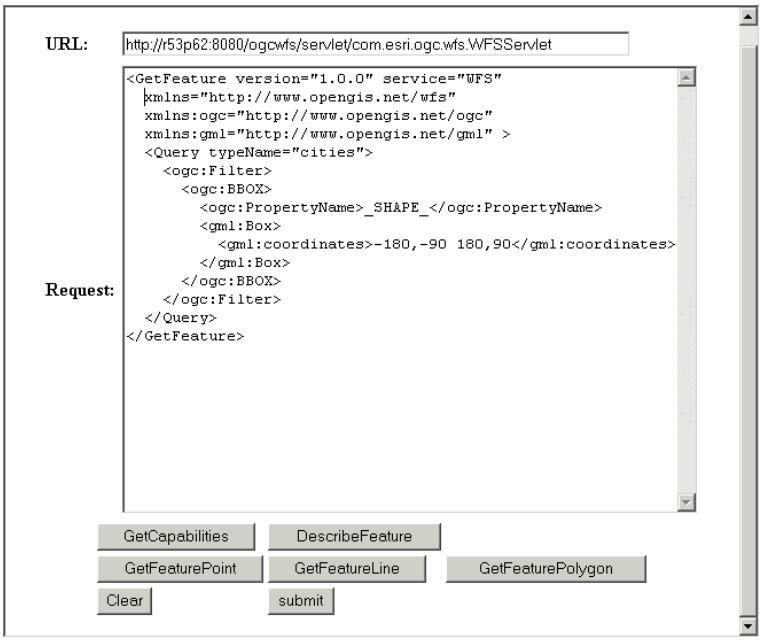


Abbildung 4.4: Client zum Senden von WFS-Anfragen an den WFS Connector.

In der Installationsdatei des ESRI WFS Connectors [ESRI03e] befindet sich ein ähnlicher Client (siehe Abbildung 4.4), mit dem WFS-konforme Requests an den WFS Connector gesendet werden. Die Antwort wird dann ebenfalls im Browser angezeigt.

Da die Anfragen an die WMS Connectoren lediglich aus URLs bestehen, konnten diese zum Testen einfach in das Adressfeld eines Web-Browsers eingegeben werden.

4.2.2 Abfrage von Metadaten

Überblick Alle Datenquellen von GISterm werden in einem Featureschema beschrieben und in der Baumstruktur des Navigators aufgeführt (siehe Kapitel 2.4.2).

Stellt die Datenquelle Geodaten bereit, so wird für jedes Thema angezeigt, welcher Typ vorliegt. Soweit es möglich ist, wird zwischen Punkt-, Linien- und Flächengeometrien unterschieden, inhomogene Rasterdaten erhalten ein spezielles Symbol (siehe Abbildung 4.5).

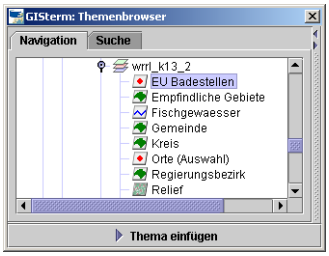


Abbildung 4.5: Unterschiedliche Thementypen.

Eine ArcIMS-Datenquelle sollte ebenfalls als Knoten innerhalb des Baumes erscheinen. Da ArcIMS Karten in Diensten strukturiert, die nochmals in Themen unterteilt sind, wäre eine Aufsplittung der Datenquelle in Kartendienste und Kartenthemen sinnvoll, so dass die Dienste wiederum Knoten für die einzelnen Kartenebenen bilden. Damit die Darstellung von ArcIMS-Diensten im Navigator realisiert werden kann, muss ArcIMS folgende Metadaten bereit stellen:

- Name des Servers
- Namen der Dienste
- Namen und Titel der Themen
- Geometrietypen der Themen

Der Titel eines Themas sollte dessen Inhalt kurz umschreiben. Da er mitunter aus mehreren Wörten bestehen kann und dies zur Identifizierung des Themas innerhalb von GISterm nachteilig ist, muss jedem Thema zusätzlich eine eindeutige Bezeichnung (ID) in Form eines Namens oder einer Nummer zugeordnet werden.

ArcXML Bei Verwendung von ArcXML kann mit `<GETCLIENTSERVICES/>` (siehe [ESRI02e, S. 436]) eine Auflistung aller verfügbaren ArcIMS-Dienste angefordert werden. Die URL muss dafür `ServiceName=catalog` enthalten:

```
http://arcims/servlet/com.esri.esrimap.Esrimap?ServiceName=catalog
&ClientVersion=4.0&Form=True&Encode=True
```

Wird die Anfrage nicht mit Hilfe von HTML-Formularen an den Server übertragen, so muss in der URL `Form=False` verwendet werden. Dies wäre zum Beispiel bei einem Java Client der Fall.

In der Antwort werden daraufhin unter anderem die Namen der Dienste sowie die Servertypen, auf denen die Dienste laufen, zurückgegeben:

```
<ARCXML version="1.1">
<RESPONSE>
<SERVICES>
  <SERVICE name="world" servicegroup="ImageServer1" access="PUBLIC"
    type="ImageServer" version="" status="ENABLED" >
    <IMAGE type="JPG" />
    <ENVIRONMENT>
      <LOCALE country="US" language="en" variant="" />
      <UIFONT name="Arial" />
    </ENVIRONMENT>
    <CLEANUP interval="10" />
  </SERVICE>
  <SERVICE name="wfs_world" servicegroup="FeatureServer1"
    access="PUBLIC" type="FeatureServer" version="" status="ENABLED">
    <ENVIRONMENT>
      <LOCALE country="US" language="en" variant="" />
      <UIFONT name="Arial" />
    </ENVIRONMENT>
    <CLEANUP interval="0" />
  </SERVICE>
</SERVICES>
</RESPONSE>
</ARCXML>
```

Mit den durch `<GETCLIENTSERVICES/>` ermittelten Namen der Dienste und der Anfrage `<GET_SERVICE_INFO>` [ESRI02e, S. 85-97] können nun Capabilities-Dateien für jeden Dienst abgerufen werden. Dazu wird einfach der Name des gewünschten Dienstes dem Parameter `ServiceName` zugewiesen. Beispiel:

```
http://arcims/servlet/com.esri.esrimap.Esrimap?ServiceName=world
&ClientVersion=4.0&Form=True&Encode=True
```

Mit Hilfe von Attributen kann der Inhalt der Antwort an eigene Bedürfnisse angepasst werden. Auf diese Weise können nicht benötigte Informationen weggelassen werden, so dass die Capabilities-Datei weniger komplex ist.

```
<ARCXML version="1.1">
<REQUEST>
  <GET_SERVICE_INFO renderer="false" extensions="false"
    fields="false" envelope="false" />
</REQUEST>
</ARCXML>
```

In der Antwort enthalten die Attribute `name` und `id` des `<LAYERINFO>`-Tags sowie `type` des `<FCLASS>`-Tags die benötigten Informationen:

```
<SERVICEINFO>
  <PROPERTIES>
    <ENVELOPE minx="-174" miny="-90" maxx="175" maxy="90"
      name="Initial_Extent" />
    <MAPUNITS units="decimal_degrees" />
  </PROPERTIES>
  <LAYERINFO type="featureclass" visible="true" name="country" id="1">
    <FCLASS type="polygon"></FCLASS>
  </LAYERINFO>
  <LAYERINFO type="featureclass" visible="true"
      name="rivers" id="3" maxscale="0,156622671138326">
    <FCLASS type="line"></FCLASS>
  </LAYERINFO>
  <LAYERINFO type="featureclass" visible="true" name="cities" id="4"
    maxscale="0,11776079837298">
    <FCLASS type="point"></FCLASS>
  </LAYERINFO>
</SERVICEINFO>
```

WMS Mit Hilfe der beiden WMS Connectoren kann leider nicht in Erfahrung gebracht werden, welche Dienste auf dem ArcIMS zur Verfügung stehen. Über die Schnittstelle *GetCapabilities* [OGC02a, S. 21] kann zwar eine Capabilities-Datei angefordert werden, diese enthält jedoch nur Informationen über einen default-Dienst, der in den Konfigurationsdateien der WMS Connectoren festgelegt werden kann. Sind dem Client die Namen der Dienste bereits bekannt, so kann in der URL der zusätzliche Parameter `SERVICENAME` verwendet werden, mit dessen Hilfe auch die anderen ArcIMS-Dienste angesprochen werden können. Dieser Parameter wird von beiden Connectoren unterstützt und ist „ESRI-spezifisch“, d.h. er ist nicht WMS-konform. Anbieter-spezifische Parameter (engl. Vendor-Specific Parameters) werden vom OGC nicht grundsätzlich ausgeschlossen [OGC02a, S. 19], dies hat jedoch zur Folge, dass WMS-konforme Clients diese zusätzlichen Funktionen nicht automatisch nutzen können. Unterstützt der Client `SERVICENAME` nicht, so können andere ArcIMS-Dienste nur angesprochen werden, wenn die Konfigurationsdateien der Connectoren editiert werden. Beispiel für einen *GetCapabilities*-Request:

```
http://arcims/servlet/com.esri.wms.Esrimap
?REQUEST=GetCapabilities&SERVICE=WMS&SERVICENAME=wms_world
```

Die `<Service>`- und `<Layer>`-Elemente der Capabilities-Dateien beider Connetoren enthalten die benötigten Informationen. Es kann sowohl der Name des Dienstes (`<Title>`-Element) als auch die Namen der Themen abgelesen werden, da das `<Layer>`-Element nochmals für jedes Thema des Dienstes ein weiteres `<Layer>`-Element enthält. Beispiel:

```
<Layer>
  <Title>wms_world</Title> <!-- Name des Dienstes -->
  <SRS>EPSG:4326</SRS>
  <Layer>
    <Name>country</Name>   <!-- ID der Ebene -->
    <Title>country</Title> <!-- Name der Ebene -->
    <SRS>EPSG:4326</SRS>
    <LatLonBoundingBox minx="-180.0" miny="-90.0"
      maxx="180.0" maxy="83.6236"/>
  </Layer>
  <!-- usw... für jede Ebene -->
</Layer>
```

Die Capabilities-Datei des WMS Connectors 2 (siehe Anhang A.2) zeigt für den Namen des Dienstes leider nur einen statischen Text an („ArcIMS MapService“), so dass der eigentliche Name des Dienstes nicht herausgefunden werden kann. Es besteht jedoch die Möglichkeit diesen manuell einzustellen, indem das XML Schema Dokument, das als Vorlage für die Capabilities-Datei dient, editiert wird.

Bei einem WMS ist der Inhalt des `<Name>`-Elements mit einer ID vergleichbar, während das `<Title>`-Element eine für Menschen nachvollziehbare Beschreibung des Themas bereitstellt.

Die WMS Connectoren besitzen leider keine Fähigkeiten zur Beschreibung der Thementypen, so dass diese Anforderung nicht erfüllt werden kann.

WFS Für den ESRI WFS Connector gelten die gleichen Einschränkungen wie für die WMS Connectoren. Die Capabilities-Datei enthält lediglich Informationen über die Themen des Standarddienstes, die anderen Dienste können mit `SERVICENAME` angesprochen werden. Anders als bei den WMS Connectoren enthält die Capabilities-Datei keine Angabe über den Namen des Dienstes. Eine Liste der Themen (hier Feature Types genannt) enthält das Element `<FeatureTypeList>` (siehe Anhang A.3). Dort finden sich wie bei einem WMS Angaben über Name und Titel des Themas. Die Capabilities-Datei kann mit folgendem Request abgerufen werden:

```
http://arcims/ogcwfs/servlet/com.esri.ogc.wfs.WFSServlet?
REQUEST=GetCapabilities&SERVICE=WFS
```

Im Gegensatz zu einem WMS kann ein WFS Informationen über die Geometrietypen mit Hilfe der speziellen *DescribeFeatureType*-Schnittstelle [OGC02b, S. 65], die mit folgendem Request angesprochen wird, bereitstellen:

```
http://arcims/ogcwfs/servlet/com.esri.ogc.wfs.WFSServlet?
REQUEST=DescribeFeatureType&SERVICE=WFS&VERSION=1.0.0
```

Hiermit kann ein XML Schema angefragt werden, das den Aufbau der Feature Types, die der Server bereitstellen kann, beschreibt. Ein vollständiges Schema Dokument ist in Anhang B.1 zu finden. Für jedes Thema findet sich dort eine Liste von

<element>-Tags, die die Attributdaten des Themas repräsentieren. Da ein WFS die Geo-Objekte in GML beschreibt, sind die Geometriedaten ebenfalls als Attributdaten aufgeführt (siehe Abschnitt 2.2.3). Der <element>-Tag enthält als Attribute die Namen der Attributdaten sowie ihren Typ. Die Geometriedaten heißen bei ArcIMS immer „_SHAPE_“, so dass damit der Geometrietyp herausgefunden werden kann.

Fazit Zur Abfrage der Namen aller verfügbaren ArcIMS-Dienste muss ArcXML verwendet werden, da die OpenGIS Connectoren lediglich Informationen über einen defaultmäßig verwendeten Dienst bereitstellen können. Die Namen müssten GISterm daher manuell mitgeteilt werden, was bei firmeneigenen Servern zwar umständlich, jedoch realisierbar wäre. Dies macht jedoch eine dynamische Nutzung verschiedener ArcIMS unmöglich, da die Namen der Dienste fremder Anbieter unbekannt sind und nicht in Erfahrung gebracht werden können. Weiterhin würden Namensänderungen oder die Verfügbarkeit der Dienste nicht automatisch erkannt, wodurch die Nutzung von ArcIMS durch die OpenGIS Connectoren weiter erschwert werden würde.

Sowohl ArcXML als auch die Connectoren für die OpenGIS Standards können dann zur Abfrage der Capabilities der einzelnen Dienste verwendet werden. Bei Nutzung der WMS Connectoren müsste jedoch für die WMS-Schnittstelle von GISterm die Verwendung des ESRI-spezifischen Parameters SERVICENAME implementiert werden.

Die Geometrietypen der Themen lassen sich nur mit Hilfe von ArcXML oder des WFS Connectors herausfinden.

4.2.3 Abruf von Rasterbildern

Überblick Rasterbilder werden in GISterm meist als Hintergrundbilder verwendet. Da Karten im Rasterformat sowohl von ArcIMS Image Diensten als auch von ArcMap Image Diensten bereitgestellt werden, ist es wichtig, dass GISterm beide Dienste angesprechen kann.

Dabei muss gesteuert werden können, welche Themen in der angeforderten Karte enthalten sein sollen. Dies ist von Bedeutung, da mit Hilfe des Navigators sowohl die Dienste insgesamt, als auch einzelne Themen dieser Dienste in GISterm eingefügt werden sollen.

Eine weitere wichtige Anforderung ist die Unterstützung räumlicher Anfragen, so dass der Abruf von Rasterbildern auf den aktuell sichtbaren Kartenausschnitt eingeschränkt werden kann. Zur Realisierung dieser *Fensterbedingung* verwendet GISterm eine Bounding Box, mit deren Hilfe ebenfalls eine Verschiebung, Verkleinerung oder Vergrößerung des Kartenbereichs realisiert wird.

ArcXML Der Request zum Abruf einer Karte als Rasterbild wird innerhalb von <GET_IMAGE> [ESRI02e, S. 421] definiert. Beispiel für einen <GET_IMAGE>-Request:

```
<REQUEST>
  <GET_IMAGE>
    <PROPERTIES>
      <ENVELOPE minx="-180" miny="-90" maxx="180" maxy="90" />
      <IMAGESIZE width="400" height="300" />
      <LAYERLIST>
        <LAYERDEF id="0" name="cities" visible="true" />
        <LAYERDEF id="1" name="lakes" visible="true" />
        <LAYERDEF id="2" name="rivers" visible="false" />
        <LAYERDEF id="3" name="country" visible="false" />
      </LAYERLIST>
    </PROPERTIES>
  </GET_IMAGE>
</REQUEST>
```

Zusätzlich muss in der URL der Name des Dienstes angegeben werden, wodurch sowohl ArcMap Image Dienste als auch Image Dienste angesprochen werden können:

```
http://arcims/servlet/com.esri.esrimap.Esrimap?ServiceName=world
&ClientVersion=4.0&Form=True&Encode=True
```

Der `<PROPERTIES>`-Element definiert die Eigenschaften der Karte. Sind hier keine Unterelemente angegeben, so werden die Defaulteinstellungen aus dem Map Configuration File (MCF) [ESRI02e, S. 14-27], mit der der Dienst angelegt worden ist, verwendet. Unterelemente von `<PROPERTIES>` überschreiben die entsprechenden Anweisungen in der MCF. Auf diese Weise kann unter anderem der Kartenausschnitt (`<Envelope>`), die Größe (`<IMAGESIZE>`) und auch die Sichtbarkeit der Ebenen neu festgelegt werden. Ob ein Thema dargestellt wird, regelt das Attribut `visible` von `<LAYERDEF>`. Wird es auf `true` gesetzt so ist das Thema sichtbar, bei `false` ist es unsichtbar.

Bei diesen einfachen `<GET_IMAGE>`-Requests muss noch nicht zwischen ArcMap Image Diensten und Image Diensten unterschieden werden. Bei komplexeren Anfragen muss jedoch darauf geachtet werden, ob die verwendeten Tags von beiden Diensten unterstützt werden. ArcMap Image Dienste bieten insgesamt gesehen weniger Funktionen zum Rendern der Karten an. So lassen sich zum Beispiel weder *Grouprenderer* verwenden, noch können Diagramme zur Beschriftung von Geo-Objekten herangezogen werden. Weitere Unterschiede sind in [ESRI02e, S. 173-207] beschrieben.

Die Antwort des Spatial Servers auf `<GET_IMAGE>` enthält neben der Bounding Box der Karte lediglich die URL der Kartengrafik, so dass diese vom Client abgerufen werden kann.

```
<RESPONSE>
  <IMAGE>
```

```
    <ENVELOPE minx="-180" miny="-135" maxx="180" maxy="135" />
    <OUTPUT url="http://arcims/output/world_R53P62271220609.jpg" />
  </IMAGE>
</RESPONSE>
```

WMS Mit Hilfe beider WMS Connectoren können sowohl Image Dienste als auch ArcMap Image Dienste angesprochen werden, da ihnen durch die Verwendung des Parameters `SERVICENAME` mitgeteilt wird, welcher Dienst kontaktiert werden soll.

Das Anfordern einer Karte erfolgt bei WMS über die Schnittstelle *GetMap* [OGC02a, S. 32f]. Mit Hilfe der in *GetMap*-Anfragen verwendeten Parameter lässt sich das Erscheinungsbild der Karte festlegen. `BBOX` definiert den Kartenausschnitt, `WIDTH` und `HEIGHT` geben die Kartengröße an, und mit `LAYERS` kann eine Liste von Ebenen angegeben werden, die in der Karte enthalten sein sollen. Beispiel für einen *GetMap*-Request:

```
http://arcims/servlet/com.esri.wms.Esrimap?VERSION=1.1.0
&REQUEST=GetMap&LAYERS=country,cities,rivers&STYLES=style1
&BBOX=-90,-180,90,180&SRS=EPSG:4326&WIDTH=400&HEIGHT=300
&FORMAT=image/jpeg&TRANSPARENT=TRUE&BGCOLOR=0xffffff
```

Die Antwort besteht lediglich aus einem Rasterbild, dessen Format durch `Format` festgelegt wird. Der ESRI WMS Connector kann Karten im JPG, GIF und PNG-Format zurückgeben, während der ESRI WMS Connector 2 nur JPG und PNG unterstützt.

WFS Der ESRI WFS Connector kann keine Rasterdaten zur Verfügung stellen (siehe Abschnitt 4.1.7).

Fazit Zum Abruf von Rasterbildern sind sowohl die beiden WMS Connectoren als auch ArcXML und der Servlet Connector gleich gut geeignet, da alle Anforderungen erfüllt werden können. Voraussetzung für die Verwendung der WMS Spezifikation ist allerdings, dass die Namen aller Dienste bekannt sind und die WMS-Schnittstelle von GISterm `SERVICENAME` unterstützt.

4.2.4 Abfrage von Attributdaten eines Geo-Objekts

Überblick Eine wichtige Funktion in den meisten WebGIS-Anwendungen ist die Identifikation von Geo-Objekten in der Karte, so dass die zugehörigen Attributdaten angezeigt werden. Auch GISterm beinhaltet diese Funktion in Form des Werkzeugs „Objektattribute anzeigen“. Üblicherweise wird jeweils nur ein Objekt durch Klick auf eine bestimme Stelle in der Karte identifiziert. Diese Funktion soll auch bei Rasterbildern, die von ArcIMS bereitgestellt werden, angewendet werden können.

ArcXML ArcXML bietet mit `<GET_FEATURES>` [ESRI02e, S. 415-418] neben dem Abruf von Vektordaten auch umfangreiche Funktionen zur Abfrage von Attributdaten an. Für die Identifikation von Geo-Objekten an einer bestimmten Stelle der Karte verwendet ArcIMS allerdings nicht die x,y Koordinaten eines Pixels, sondern eine kleine Bounding Box (`<ENVELOPE>`), mit dessen Hilfe eine räumliche Verschneidung (`<SPATIALFILTER>`) durchgeführt wird. Das Attribut `subfields` gibt an, welche Attribute in der Antwort enthalten sein sollen. In diesem Fall sollen alle (`#ALL#`) enthalten sein. Das Thema, das abgefragt werden soll, wird mit `<LAYER>` festgelegt. Werden mit `<GET_FEATURES>` lediglich Sachdaten und keine Vektordaten angefordert, so wird der Request nicht von einem Image Dienst verarbeitet, sondern direkt an den Query Server „umgeleitet“. Dies geschieht durch Setzen des Parameters `CustomService=Query` in der URL des Servlet Connectors. Die Übergabe der Anfrage erfolgt dann mit HTTP-POST. Beispiel:

```
http://arcims/servlet/com.esri.esrimap.Esrimap?ServiceName=world
&ClientVersion=4.0&CustomService=Query&Form=True&Encode=True

<REQUEST>
  <GET_FEATURES featurelimit="25" beginrecord="0" outputmode="xml"
    geometry="false">
    <LAYER id="1" />
    <SPATIALQUERY subfields="#ALL#" >
      <SPATIALFILTER relation="area_intersection">
        <ENVELOPE maxy="30" maxx="30" miny="29" minx="29" />
      </SPATIALFILTER>
    </SPATIALQUERY>
  </GET_FEATURES>
</REQUEST>
```

In der Antwort werden die Attributdaten in ArcXML zurückgegeben.

```
<RESPONSE>
 <FEATURES>
   <FEATURE>
     <FIELDS FIPS_CNTRY="EG" GMI_CNTRY="EGY" CNTRY_NAME="Egypt"
       SOVEREIGN="Egypt" POP_CNTRY="56133430" SQKM_CNTRY="982910,375"
       SQMI_CNTRY="379501,688" CURR_TYPE="Pound" CURR_CODE="EGP"
       LANDLOCKED="N" COLOR_MAP="6" #SHAPE#="[Geometry]" #ID#="62" />
   </FEATURE>
   <FEATURECOUNT count="1" hasmore="false" />
 </FEATURES>
</RESPONSE>
```

WMS Der *GetFeatureInfo*-Request [OGC02a, S. 39f] liefert Attributdaten von Geo-Objekten in der Karte durch die Angabe von Pixelkoordinaten. Die Funkti-

on kann pro Anfrage jeweils nur auf ein Objekt des Themas, das durch den Parameter `QUERY_LAYERS` festgelegt wird, angewendet werden. Es können jedoch nur Geo-Objekte identifiziert werden, deren Datenquelle nicht auf Rasterdaten basieren, da diese keine Attributdaten zur Verfügung stellen. Durch Übergabe der Pixelkoordinaten und der Parameter des vorangegangenen *GetMap*-Requests kann der Spatial Server das zugehörige Objekt identifizieren, und so dessen Attributdaten an den Client übermitteln. Beispiel für einen *GetFeatureInfo*-Request:

```
http://arcims/servlet/com.esri.wms.Esrimap?VERSION=1.1.0
&REQUEST=GetFeatureInfo&LAYERS=cities&BBOX=-90,-180,90,180
&SRS=EPSG:4326&WIDTH=400&HEIGHT=300&FORMAT=image/jpeg
&TRANSPARENT=TRUE&BGCOLOR=0xffffff
&INFO_FORMAT=MIME&QUERY_LAYERS=cities&X=0&Y=0
```

Beide WMS Connectoren beschreiben durch das Unterelement `<Format>` des `<GetFeatureInfo>`-Tags der Capabilities-Datei welche Ausgabeformate unterstützt werden. Der ESRI WMS Connector liefert als Ergebnis HTML, XML oder eine Textdatei. Der ESRI WMS Connector 2 unterstützt ebenfalls HTML und Textdateien, jedoch liefert er anstatt XML GML zurück. Im Request legt `INFO_FORMAT` das Format fest.

WFS Attributdaten bestimmter Geo-Objekte können bei einem WFS über die Schnittstelle *GetFeature* abgerufen werden [OGC02b, S. 25f]. Da der ESRI WFS Connector für diese Operation nur Anfragen mit HTTP-POST akzeptiert, muss zur Formulierung der Anfrage XML verwendet werden. Die Anfrage könnte so aussehen:

```
<GetFeature version="1.0.0" service="WFS" srsName="ESPG:4326"
  xmlns="http://www.opengis.net/wfs"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc" >
  <Query typeName="cities">
    <ogc:PropertyName>NAME</ogc:PropertyName>
    <ogc:Filter>
      <ogc:BBox>
      <ogc:PropertyName>_SHAPE_</ogc:PropertyName>
        <gml:Box>
          <gml:coordinates>45.2,2.0 45.4,2.2</gml:coordinates>
        </gml:Box>
      </ogc:BBox>
    </ogc:Filter>
  </Query>
</GetFeature>
```

Das Attribut `typeName` des `<Query>`-Tags bestimmt, auf welches Thema die Abfrage angewendet werden soll. `<Query>` kann als Unterelemente `<ogc:PropertyName>` und `<ogc:Filter>` haben.

<ogc:PropertyName> kann beliebig oft verwendet werden und definiert die Attribute, die in der Antwort enthalten sein sollen. Wird dieses Elemente nicht verwendet, sind alle Attribute in der Antwort enthalten. Da zu den Attributdaten eines Geo-Objekts jedoch auch Geometriedaten zählen, ist es für diesen Anwendungsfall erforderlich, nur die gewünschten Attribute im Request aufzuführen. Damit wird verhindert, dass die oftmals viel umfangreicheren Geometrien ebenfalls mit abgerufen werden, wodurch die Attributdaten wesentlich schneller übertragen werden.

Zur Auswahl von bestimmten Objekten kann mit Hilfe von <ogc:Filter> eine Einschränkung definiert werden (siehe Abschnitt 2.2.5). Um die Attribute eines einzelnen Objekts abzufragen, wird wie bei ArcXML eine sehr kleine Bounding Box zur Verschneidung mit den Geometrien des Themas benutzt.

Fazit Der Abruf von Objektinformationen bei Rasterbildern lässt sich am einfachsten mit der WMS Spezifikation verwirklichen, da hier keine Filter definiert werden müssen. Die Verwendung des WFS Connectors zur Abfrage der Attribute kommt nicht in Frage, da die Rasterbilder entweder über den WMS Connector oder den Servlet Connector angefordert werden müssen und beide Lösungswege bereits sinnvolle Funktionen zur Identifikation von Geo-Objekten anbieten.

4.2.5 Abruf von Attributdaten mehrerer Geo-Objekte

Überblick In GISterm kann über das Kontextmenü der Themen eine Attributtabelle für die Geo-Objekte des aktuellen Kartenausschnitts angezeigt werden. Damit dies auch für ArcIMS Rasterbilder funktioniert, muss die Möglichkeit bestehen, von ArcIMS alle Attributdaten der Objekte im aktuellen Ausschnitt anzufordern.

ArcXML Der Request <GET_FEATURES> ermöglicht die Abfrage von Attributdaten im ArcXML-Format. Mit dem Unterelement <SPATIALQUERY> und dessen Attributen lassen sich Abfragen definieren, die sowohl eine räumliche als auch eine attributbezogene Auswahl der Attributdaten zulassen. Werden alle Attributdaten eines Themas ohne dessen Geometrien benötigt, so werden die Attribute `subfields="#ALL#"` und `geometry="false"` verwendet. Der Request sieht dann wie folgt aus:

```
<ARCXML version="1.1">
<REQUEST>
  <GET_FEATURES beginrecord="0" outputmode="xml" geometry="false">
    <LAYER id="cities" />
    <SPATIALQUERY subfields="#ALL#">
    </SPATIALQUERY>
  </GET_FEATURES>
</REQUEST>
</ARCXML>
```

Bestimmte Attributdaten von Geo-Objekten innerhalb einer Bounding Box werden mit der folgenden Abfrage ermittelt:

```
<ARCXML version="1.1">
<REQUEST>
  <GET_FEATURES beginrecord="0" outputmode="xml" geometry="false">
    <LAYER id="cities" />
    <SPATIALQUERY subfields="NAME POPULATION" >
      <SPATIALFILTER relation="envelope_intersection">
        <ENVELOPE minx="30" miny="10" maxx="50" maxy="0" />
      </SPATIALFILTER>
    </SPATIALQUERY>
  </GET_FEATURES>
</REQUEST>
</ARCXML>
```

Hierbei werden die Namen der Attribute, die in der Antwort enthalten sein sollen, mit `subfields="NAME POPULATION"` angegeben. Mit Hilfe von `<SPATIALFILTER>` und `<ENVELOPE>` wird die räumliche Einschränkung definiert.
Die Antwort ist dann in ArcXML verfasst:

```
<ARCXML version="1.1">
<RESPONSE>
  <FEATURES>
    <FEATURE>
      <FIELDS NAME="Muqdisho" POPULATION="600000" />
    </FEATURE>
    <FEATURE>
      <FIELDS NAME="Adis Abeba" POPULATION="1500000" />
    </FEATURE>
    <FEATURE>
      <FIELDS NAME="Kampala" POPULATION="460000" />
    </FEATURE>
    <FEATURECOUNT count="3" hasmore="false" />
  </FEATURES>
</RESPONSE>
</ARCXML>
```

WMS Die WMS Connectoren können mit *GetFeatureInfo* pro Anfrage lediglich die Attribute eines Objekts abrufen (siehe auch Abschnitt 4.2.4). Die Angabe einer Bounding Box zur Abfrage von Attributdaten ist nicht möglich.

WFS Durch Verwendung des gleichen `<BBox>`-Filters, wie er in Abschnitt 2.2.5 beschrieben wird, können die Attributdaten für den aktuellen Ausschnitt mit Hilfe der WFS Spezifikation angefordert werden.

Die Verwendung des WFS Connectors zum Abruf von Attributdaten für Rasterbilder ist jedoch nur dann sinnvoll, wenn ArcIMS Rasterbilder mit Hilfe der WMS-Schnittstelle von GISterm eingebunden werden. Da WMS die Abfrage von Attributdaten mit Hilfe einer Bounding Box nicht unterstützen, könnten sich eine neu zu schaffende WFS-Schnittstelle und die vorhandene WMS-Schnittstelle gegenseitig ergänzen.

Fazit Welcher Lösungsweg für diesen Anwendungsfall eingeschlagen wird, hängt davon ab, welche ArcIMS-Programmierschnittstelle letztendlich zur Abfrage von Rasterbildern verwendet wird.

Werden die Karten von einem WMS Connector bereitgestellt, so ist eine zusätzliche Verwendung des WFS Connectors zur Realisierung dieses Anwendungsfalls sinnvoll, da beide Connectoren auf dieselben Image Dienste zugreifen können und sich so in ihren Funktionen gegenseitig ergänzen würden. Damit würde die Einbindung von ArcIMS-Diensten in GISterm vollständig auf OpenGIS Standards beruhen.

Wird zum Abruf der Karten allerdings ArcXML verwendet, so ist die Erweiterung der proprietären ArcIMS-Schnittstelle für GISterm sicherlich einfacher zu realisieren als extra für diesen Anwendungsfall eine neue WFS-Schnittstelle zu schaffen.

4.2.6 Abruf einer Legende für Rasterbilder

Überblick Damit die Themen einer Karte vom Betrachter identifiziert werden können, bedarf es einer Zeichenerklärung in Form einer Legende. Aus Gründen der Übersichtlichkeit ist es weiterhin sehr wichtig, dass die Legende nur diejenigen Themen erklärt, die in der aktuellen Karte auch wirklich enthalten sind. Werden zum Beispiel Themen eines Dienstes nicht angezeigt, so muss dies in der Legende berücksichtigt werden.

ArcXML In ArcXML ermöglicht das Element `<LEGEND>` [ESRI02e, S. 137-140], welches innerhalb eines `<GET_IMAGE>`-Requests verwendet wird, die Darstellung einer Legende. Das Aussehen der Legende (Schriftart und -größe, Spaltenaufteilung etc.) lässt sich mit Hilfe von Attributen festlegen. Das Element `<DRAW map="false" />` bestimmt, ob in der Antwort nur die Legende zurückgegeben wird (`map="false"`) oder ob zusätzlich auch eine Karte erstellt werden soll (`map="true"`).

```
<REQUEST>
<GET_IMAGE>
  <PROPERTIES>
    <ENVELOPE minx="-180" miny="-90" maxx="180" maxy="90" />
    <IMAGESIZE width="400" height="300" />
    <LEGEND title="Legend" font="Arial" width="170" height="300"
      autoextend="true" backgroundcolor="255,255,255" titlefontsize="12"
```

```
      swatchwidth="14" swatchheight="18" antialiasing="true"
      cellspacing="2" columns="1" reverseorder="false" cansplit="false"
      splittext="(cont)" layerfontsize="10" valuefontsize="8"
      transcolor="254,254,254" />
    <DRAW map="false" />
  </PROPERTIES>
</GET_IMAGE>
</REQUEST>
```

Wie beim Request einer Karte, wird auch die Legende vom Spatial Server als Bilddatei in einem bestimmten Ordner abgelegt. In der Antwort des Servers wird dem Client mit Hilfe einer URL der Ablageort der Legende mitgeteilt, so dass dieser sie abrufen kann.

```
<RESPONSE>
  <IMAGE>
    <LEGEND url="http://arcims/output/world_R53P622712295213.jpg" />
  </IMAGE>
</RESPONSE>
```

WMS Ab WMS 1.1.0 wird die Verwendung von Legenden durch Thema-bezogene Metadaten, die in der Capabilities-Datei aufgeführt werden können, unterstützt. Mit Hilfe von `<LegendURL>` kann eine URL angegeben werden, unter der eine Legende für einen bestimmten Zeichensschlüssel als Bilddatei abgelegt ist [OGC02a, S. 25f]. Diese Legendengrafik kann somit bei Bedarf vom Client abgerufen werden.

Ein SLD-WMS unterstützt sogar einen speziellen *GetLegendGraphic*-Request zum Abruf von Legenden. Damit kann für jedes Thema dynamisch eine separate Grafik angefordert werden [OGC02d, S. 61-63].

Der ESRI WMS Connector unterstützt zwar WMS 1.1.0, jedoch ist in seiner Capabilities-Datei kein `<LegendURL>`-Tag aufgeführt. Dasselbe gilt für den ESRI WMS Connector 2. Auch der *GetLegendGraphic*-Request wird von beiden Connectoren nicht unterstützt, da sie keine SLD-Fähigkeiten besitzen.

WFS Für Vektordaten des ESRI WFS Connectors kann keine Legende abgerufen werden, da GML keine Informationen über die grafische Präsentation der Geo-Objekte enthält. Das Rendern der Objekte übernimmt daher der Client, so dass GISterm die Informationen zum Erstellen einer Legende selbst bereitstellen kann.

Fazit Da keiner der OpenGIS Connectoren die Bereitstellung einer Legende unterstützt, kann der Anwendungsfall nur mit Hilfe von ArcXML und Servlet Connector gelöst werden, da eine Legende ein unverzichtbarer Bestandteil einer Karte ist. Ohne eine Erklärung ihrer Inhalte kann die Karte nicht interpretiert werden und ist daher in den meisten Fällen wertlos. Die WMS Connectoren könnten daher lediglich

zur Darstellung von Orthofotos oder Topografischen Karten verwendet werden, da ArcIMS für diese auf Rasterdaten basierenden Themen ebenfalls keine verwertbaren Legenden erstellen kann.

4.2.7 Festlegung von Maßstabsgrenzen

Überblick Damit Bildschirmkarten auch in kleineren Maßstäben ihre Übersichtlichkeit nicht verlieren, lassen sich für die einzelnen Themen Maßstabsgrenzen festlegen. Außerhalb dieser Grenzen werden die Themen nicht dargestellt.

Auch GISterm bietet diese Funktion für einzelne Themen an. Will der Anwender die Maßstabsgrenzen eines ArcIMS-Dienstes neu festlegen, so müssen die Eigenschaften in der MCF überschrieben werden.

ArcXML Maßstabsober- und Maßstabsuntergrenzen der Themen können innerhalb von `<GET_IMAGE>` neu festgelegt werden. Dies geschieht mit Hilfe der Attribute `maxscale` und `minscale` von `<LAYER>`. Leider können die ursprünglichen Definitionen in der MCF nicht einfach wie z. B. mit `<LAYERDEF>` überschrieben werden. Damit die maßstabsabhängige Sichtbarkeit eines bereits existierenden Themas eines Image Dienstes verändert werden kann, muss es durch eine neue Kartenebene mit gleichem Inhalt aber veränderten `minscale`-/`maxscale`- Attributen ersetzt werden.

Dazu wird das zu ersetzende Thema mit `<LAYERDEF>` unsichtbar gemacht und mit `<LAYER>` ein neues Thema mit den gewünschten Attributen definiert. Dabei muss zwischen Themen, die auf Rasterdaten basieren und solchen, die Vektordaten als Grundlage haben, unterschieden werden.

Da eine neue Kartenebene angelegt wird, müssen nochmals die Renderdefinitionen für die Geo-Objekte angegeben werden. Das Attribut `id` muss eindeutig sein und darf noch nicht an ein anderes Thema des gleichen Dienstes vergeben worden sein. In `<DATASET>` wird die zu ersetzende Ebene als Datenquelle angegeben. Der Request zur Erzeugung eines neuen Themas, das auf Vektordaten basiert, sieht so aus:

```
<GET_IMAGE>
  <PROPERTIES>
    <LAYERLIST>
          <LAYERDEF id="country" visible="false" />
    </LAYERLIST>
  </PROPERTIES>
  <LAYER type="featureclass" name="Countries" visible="true"
    id="9" maxscale="1:50000000" minscale="1:5000000">
    <DATASET fromlayer="country"/>
    <SIMPLERENDERER><SIMPLEPOLYGONSYMBOL/></SIMPLERENDERER>
  </LAYER>
</GET_IMAGE>
```

Für Themen aus Rasterdaten sieht die Anfrage wie folgt aus:

```
<GET_IMAGE>
  <PROPERTIES>
    <LAYERLIST>
      <LAYERDEF id="relief" visible="false" />
    </LAYERLIST>
  </PROPERTIES>
  <LAYER type="image" id="11" visible="true" maxscale="1:500000">
    <DATASET fromlayer="relief"/>
  </LAYER>
</GET_IMAGE>
```

WMS In der SLD Spezifikation ist beschrieben, wie Maßstabsgrenzen bei der Anfrage an einen SLD-WMS festgelegt werden können [OGC02d, S. 26-29]. Da beide WMS Connectoren keine SLD unterstützen, kann diese Aufgabe mit ihnen nicht realisiert werden.

WFS Die Festlegung von Maßstabsgrenzen fällt in den Bereich des Renderns und wird somit von WFS nicht unterstützt.

Fazit Für die Realisierung dieser Aufgabe muss ArcXML verwendet werden, da eine Änderung der Maßstabsgrenzen von den anderen Connectoren nicht unterstützt wird.

4.2.8 Abruf von Geo-Objekten

Überblick Ein großer Vorteil der Verwendung von Vektorgrafiken in WebGIS-Anwendungen ist, dass die zugrunde liegenden Geodaten einmalig geladen werden und dann im Client für alle Arten von GIS-Funktionen zur Verfügung stehen, ohne dass der Server nochmals kontaktiert werden muss. Dadurch wird die Belastung von Netzwerk und Internet deutlich verringert und die Geschwindigkeit der Anwendung erhöht.

Die Nutzung von ArcIMS Geo-Objekten in GIStermFramework ist wünschenswert, weil sie nach einer Konvertierung in „GISterm Geo-Objekte“ von GISterm als Vektorgrafik visualisiert werden können. Weiterhin steht für Geo-Objekte automatisch der komplette Funktionsumfang von GISterm zur Verfügung, wodurch auch Funktionen genutzt werden, die mit Rasterbildern nicht durchführbar sind. Mögliche zusätzliche Anwendungsfälle wären daher zum Beispiel die Editierung der Vektoren oder Verschneidungen mit anderen Vektordaten.

Die mit den Geo-Objekten verknüpften Sachdaten stehen ebenfalls automatisch zur Abfrage von Objektinformationen oder zur Darstellung in Tabellenform bereit.

Wie beim Abruf von Rasterbildern ist auch hier die Verwendung einer Bounding Box zur Einschränkung der Datenmenge von großer Wichtigkeit. Wie in Abschnitt 4.1.7 beschrieben, kann dadurch die Wartezeit bis zum Eintreffen der Antwort erheblich verkürzt werden.

ArcXML Der Zugriff auf Geo-Objekte erfolgt in ArcXML über `<GET_FEATURES>`. Das Attribut `outputmode` definiert das Format, in dem die Objekte zurückgegeben werden. Der Ausgabemodus `xml` ermöglicht den Abruf von Geo-Objekten, die von Image Diensten oder ArcMap Image Diensten mit Hilfe ihrer Query Server bereitgestellt werden. Wird hingegen der Modus `binary` verwendet, so wird auf einen Feature Dienst zugegriffen, der die Geo-Objekte in einem komprimierten Binärstrom an den Client sendet. Dieses Datenformat ist allerding proprietär und kann daher nur von den ArcIMS Java Viewern oder von ArcGIS, der GIS-Software von ESRI, interpretiert werden.

Damit neben den Attributdaten auch die Geometrien der Objekte zurückgeliefert werden, muss `geometry=true` verwendet werden. Eine Auswahl aus den vorhandenen Attributen kann mit `subfields` getroffen werden. Sollen alle Attribute übertragen werden, so wird `subfields="#ALL#"` verwendet. Zur Beschränkung der Daten auf einen bestimmten Kartenauschnitt wird `<ENVELOPE>` in Verbindung mit einem `<SPATIALFILTER>`-Tag, der die Art der Verschneidung festlegt, verwendet.

```
<ARCXML version="1.1">
<REQUEST>
  <GET_FEATURES beginrecord="0" outputmode="xml" geometry="true"
    envelope="true" globalenvelope="true">
    <LAYER id="cities" />
    <SPATIALQUERY subfields="#ALL#" >
      <SPATIALFILTER relation="area_intersection">
        <ENVELOPE minx="-180" miny="-90" maxx="180" maxy="90"/>
      </SPATIALFILTER>
    </SPATIALQUERY>
  </GET_FEATURES>
</REQUEST>
</ARCXML>
```

Die Antwort hat folgendes Schema:

```
<ARCXML version="1.1">
<RESPONSE>
  <FEATURES>
    <FEATURE>
      <ENVELOPE minx="33,0860404968262" miny="68,9635467529297"
        maxx="33,0860404968262" maxy="68,9635467529297"/>
      <FIELDS NAME="Murmansk" COUNTRY="Russia" POPULATION="468000"
```

```
        CAPITAL="N" #SHAPE#="[Geometry]" #ID#="1" />
      <MULTIPOINT><POINT x="33,0860404968262" y="68,9635467529297"/>
      </MULTIPOINT>
    </FEATURE>
    <!-- ...usw für alle Features-->
    <FEATURECOUNT count="21" hasmore="true" />
    <ENVELOPE minx="33,0860404968262" miny="64,5206680297852"
      maxx="40,6461601257324" maxy="68,9635467529297"/>
  </FEATURES>
</RESPONSE>
</ARCXML>
```

WMS WMS sind nur dafür ausgelegt Rasterbilder zur Verfügung zu stellen.

WFS Der ESRI WFS Connector ermöglicht den Zugriff auf Geo-Objekte von Image Diensten und ArcMap Image Diensten mit Hilfe des *GetFeature*-Requests und HTTP-POST. Beispiel:

```
<GetFeature version="1.0.0" service="WFS"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:gml="http://www.opengis.net/gml" >
  <Query typeName="cities" >
    <ogc:Filter>
      <ogc:BBox>
        <PropertyName>Geometry</PropertyName>
        <gml:Box>
          <gml:coordinates>-180,-90 180,90</gml:coordinates>
        </gml:Box>
      </ogc:BBox>
    </ogc:Filter>
  </Query>
</GetFeature>
```

Hiermit werden alle Geo-Objekte des Themas „cities“ angefordert, die sich innerhalb der angegebenen Bounding Box befinden. Zur räumlichen Einschränkung der Daten kann der `<BBox>`-Filter, wie in Abschnitt 2.2.5 beschrieben, verwendet werden. Die Antwort enthält sowohl Vektordaten als auch alle Attributdaten und ist in GML verfasst:

```
<Features
  xmlns="http://www.esri.com/WFS"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.esri.com/WFS cities.xsd">
```

```
  <gml:boundedBy>
    <gml:Box>
      <gml:coord>
        <gml:X>-180</gml:X><gml:Y>-90</gml:Y>
      </gml:coord>
      <gml:coord>
        <gml:X>180</gml:X><gml:Y>90</gml:Y>
      </gml:coord>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <cities>
      <NAME>Murmansk</NAME>
      <COUNTRY>Russia</COUNTRY>
      <POPULATION>468000</POPULATION>
      <CAPITAL>N</CAPITAL>
      <_ID_>1</_ID_>
      <_SHAPE_>
        <gml:MultiPoint srsName="http://www.opengis.net/gml/srs/espg.xml#4326">
          <gml:pointMember>
            <gml:Point>
              <gml:coordinates>33,0860404968262,68,9635467529297
              </gml:coordinates>
            </gml:Point>
          </gml:pointMember>
        </gml:MultiPoint>
      </_SHAPE_>
    </cities>
  </gml:featureMember>
<!-- ...usw -->
</Features>
```

Wie bei den WMS Connectoren kann der WFS Connector ebenfalls keine Informationen über alle verfügbaren ArcIMS-Dienste liefern. Sind die Namen der Dienste im Voraus bekannt, so kann durch Verwendung des Parameters `SERVICENAME` im Query-String ein bestimmter Dienst angesprochen werden (siehe Abschnitt 4.2.2). Andernfalls können nur Geo-Objekte des Standarddienstes angefordert werden.

Fazit Der Abruf von Geo-Objekten kann sowohl mit ArcXML als auch mit der WFS Spezifikation erfolgen. Der Servlet Connector liefert die Objekte in ArcXML zurück, wohingegen die WFS Schnittstelle die Geodaten in GML beschreibt.

Die Verwendung von ArcXML zum Abruf der Geo-Objekte ist zwar möglich, jedoch nicht sinnvoll, da die in ArcXML beschriebenen Objekte aufwendig in GISterm Geo-Objekte transformiert werden müssten. Da GISterm bereits eine Funktion zum

Anwendungsfall	ArcXML	WMS	WFS
Abfrage der verfügbaren ArcIMS-Dienste	Ja	Nein	Nein
Abfrage von Metadaten eines Dienstes	Ja	Ja	Ja
Abruf von Rasterbildern	Ja	Ja	Nein
Abruf von Attributdaten eines Geo-Objekts	Ja	Ja	Ja
Abruf von Attributdaten mehrerer Geo-Objekte	Ja	Nein	Ja
Abruf einer Legende für Rasterbilder	Ja	Nein	Nein
Festlegung von Maßstabsgrenzen	Ja	Nein	Nein
Abruf von Geo-Objekten	Ja	Nein	Ja

Tabelle 4.1: Überblick über die Realisierbarkeit der Anwendungsfälle durch die verschiedenen Programmierschnittstellen.

Import von GML bereitstellt, bietet sich die Verwendung des ESRI WFS Connectors für diesen Anwendungsfall an.

4.3 Zusammenfassung und Auswahl

In Tabelle 4.1 sind die Ergebnisse dieses Kapitels zusammengefasst. Es ist zu erkennen, dass sich die OpenGIS Connectoren in den Punkten „Abruf von Geo-Objekten“ und „Abruf von Rasterbildern“ sowie „Abruf von Attributdaten mehrerer Geo-Objekte“ gegenseitig ergänzen, so dass ein Großteil der Anforderungen ohne die Verwendung des Servlet Connectors erfüllt werden kann. Dies reicht jedoch nicht aus, da mit „Abruf einer Legende für Rasterbilder“ und „Abfrage der verfügbaren Dienste“ zwei sehr wichtige Anforderungen nicht erfüllt werden können.

Für die Darstellung von ArcIMS-Rasterbildern in GISterm ist eine Legende jedoch unverzichtbar, da in der Karte nicht nur Themen mit Geobasisdaten, wie z. B. Topographische Karten, enthalten sein können, die normalerweise auch ohne Legende interpretiert werden können, sondern auch Themen, die auf Vektordaten basieren. Auf diese Daten kann ArcIMS die verschiedensten Zeichenvorschriften anwenden, so dass die daraus erzeugten Themen unbedingt erklärt werden müssen, damit sie interpretiert werden können. Die WMS Connectoren könnten daher lediglich zur Einbindung von Geobasisdaten verwendet werden.

Ein weiterer großer Nachteil der OpenGIS Connectoren ist, dass die Namen der verfügbaren ArcIMS-Kartendienste nicht abgefragt werden können. Sie müssten GISterm daher manuell mitgeteilt werden, so dass eine dynamische Nutzung

verschiedener ArcIMS unmöglich ist, da die Namen der Dienste fremder Anbieter nicht in Erfahrung gebracht werden können.

Insgesamt gesehen ist also zumindest die Einbindung von ArcIMS-Rasterbildern in GISterm mit Hilfe der OpenGIS Connectoren nicht sinnvoll, so dass hierfür die proprietäre Schnittstelle, also ArcXML und Servlet Connector, verwendet werden muss.

Wird jedoch bei der Betrachtung der Anforderungen zwischen der Einbindung von Rasterbildern und Geo-Objekten differenziert, so bietet der WFS Connector bei der Einbindung von Geo-Objekten in GISterm einige Vorteile gegenüber ArcXML:

Der Abruf einer Legende ist nicht mehr relevant, da sowohl der Servlet Connector als auch der WFS Connector Geo-Objekte nicht als fertige Karte, sondern als XML-Datei übergeben, so dass GISterm erst noch Vektorgrafiken auf Grundlage ihrer Vektordaten erzeugen muss. Da GISterm die Zeichenschlüssel also selbst erzeugt, kann somit automatisch eine Legende zur Verfügung gestellt werden.

Damit GISterm die Geo-Objekte visualisieren kann, müssen die in GML kodierten Geodaten zuerst in GISterm Geo-Objekte konvertiert werden. Da GISterm hierfür bereits einen Konvertierungsmechanismus anbietet und dieser für ArcXML erst noch realisiert werden müsste, würde die Verwendung des WFS Connectors Vorteile gegenüber der proprietären Schnittstelle haben.

Beim Einsatz des WFS Connectors kann zwar lediglich ein Standarddienst angesprochen werden, jedoch könnte GISterm mit derselben Schnittstelle auch andere WFS ansprechen, so dass sich die Einsatzmöglichkeiten nicht nur auf ArcIMS beschränken.

Darüber hinaus könnten sich die Funktionen einer neuen WFS-Schnittstelle und der bereits vorhandenen WMS-Schnittstelle gegenseitig ergänzen, so dass in Zukunft ebenfalls zusammenhängende OGC Web Services (OWS) genutzt werden können, wie sie in [OGC02a, S. 3] beschrieben werden.

Insgesamt gesehen ist also die Einbindung von Geo-Objekten über ArcXML und Servlet Connector aufwendig und in ihren Einsatzmöglichkeiten auf ArcIMS beschränkt, so dass hierfür der WFS Connector die bessere Wahl ist.

5 Konzeption und Realisierung

Nachdem im letzten Kapitel ArcXML und der Servlet Connector für die Einbindung von Rasterbildern und der WFS Connector für die Integration von Geo-Objekten in GISterm ausgewählt wurden, sollte in diesem Kapitel nun die Konzeption und Realisierung einer Schnittstelle für ArcIMS-Rasterbilder erfolgen.

Während der Konzeption wurde jedoch parallel von der Firma disy Informationssysteme GmbH eine Schnittstelle für ArcIMS Image Dienste implementiert, so dass die Umsetzung einer WFS-Schnittstelle im Rahmen dieser Diplomarbeit die interessantere Aufgabe darstellte.

Im Folgenden werden daher zuerst Ergebnisse vorgestellt, die bis zu diesem Zeitpunkt für die ArcXML-Schnittstelle erarbeitet worden sind. Anschließend wird als Alternative zur ursprünglichen Aufgabe die Konzeption und Realisierung einer WFS-Schnittstelle vorgestellt, so dass neben der Integration von WFS in GISterm auch der Zugriff auf ArcIMS Geo-Objekte mit Hilfe des ESRI WFS Connectors realisiert werden kann.

5.1 Prototyp für Abruf und Anzeige von ArcIMS-Rasterbildern

5.1.1 Überblick

In diesem Abschnitt wird der Prototyp für einen Java Client vorgestellt, der zeigt, wie die Programmiersprache Java zur Formulierung von ArcXML-Anfragen an den ArcIMS und zur Auswertung der Antworten eingesetzt werden kann. Dies geschieht zunächst bewusst unabhängig von GIStermFramework, da dessen sofortige Einbeziehung das Erlernen der grundlegenden Methoden zur Kommunikation mit ArcIMS unnötig erschweren würde.

Der im Rahmen der Diplomarbeit erstellte Prototyp kann mit einem beliebigen ArcIMS im Intranet der LfU oder im Internet Verbindung aufnehmen und die Namen der verfügbaren ArcIMS Image Dienste sowie die Namen der in den Diensten enthaltenen Themen auslesen. Auf Grundlage dieser Informationen können entweder Rasterbilder ganzer Dienste oder einzelner Themen abgerufen werden.

Um die Ergebnisse besser veranschaulichen zu können, wurde der Prototyp mit einer einfachen GUI ausgestattet, die im linken Bereich Dienste und Themen in einer Baumstruktur auflistet und im rechten Teil die Karten anzeigt (siehe Abbildung 5.1).

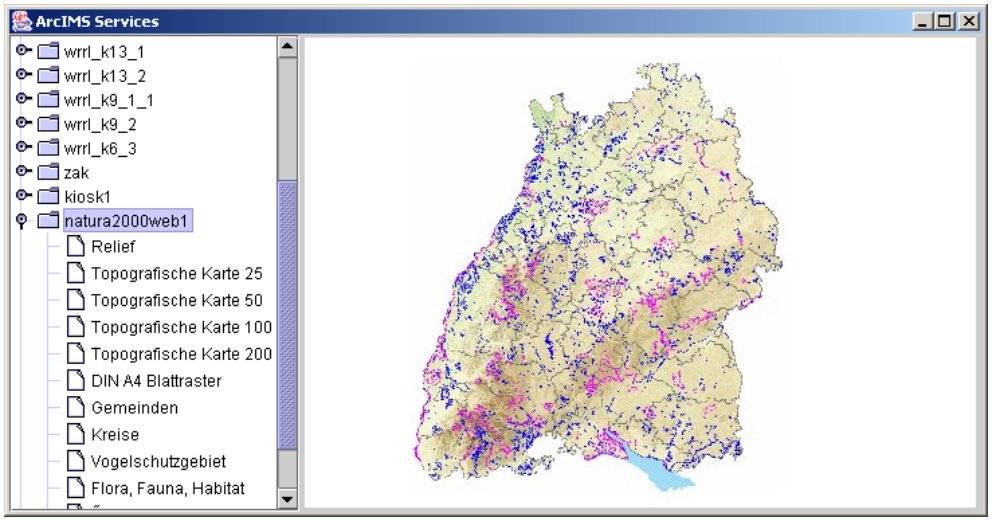


Abbildung 5.1: Java Client zur Darstellung von ArcIMS-Diensten und ihren Themen.

Der Client setzt sich aus folgenden Klassen zusammen:

- *LoginDialog:* `LoginDialog` ist von `javax.swing.JDialog` abgeleitet und erstellt ein Dialogfenster, das dem Anwender die Möglichkeit zur Eingabe der Verbindungsdaten gibt (siehe Abbildung 5.3 auf Seite 62).
- *BarDialog:* Diese Klasse ist ebenfalls von `javax.swing.JDialog` abgeleitet und dient zur Anzeige eines Dialogs mit Fortschrittsbalken während die Namen der Dienste und Themen geladen werden (siehe Abbildung 5.4 auf Seite 62).
- *AdapterApplication:* `AdapterApplication` ist von `javax.swing.JFrame` abgeleitet und definiert das Hauptfenster mit der Baumstruktur und der Karte. Die Klasse enthält weiterhin die `main()`-Methode und ist damit Ausgangspunkt der Java-Applikation. Diese Methode legt den Programmablauf fest, indem sie nacheinander die Dialoge und das Hauptfenster aufruft.
- *ArcIMSAdapter:* Diese Klasse implementiert die Funktionalitäten des Prototyps, indem sie Methoden und Felder definiert, die zur Kommunikation mit ArcIMS benötigt werden. `ArcIMSAdapter` speichert die Verbindungsdaten, formuliert die ArcXML-Anfragen und verarbeitet die Antworten.

Darüber hinaus werden folgende Hilfsklassen verwendet:

- *SwingWorker:* `SwingWorker` ist ein Dienstprogramm, das einen neuen Thread erzeugt, um eine (zeitaufwendige) Aufgabe zu erledigen. Der Prototyp verwendet diese Klasse, um eine neue Karte anzufordern. `SwingWorker` ist im Internet unter [Sun03b] frei erhältlich.

- *NamedVector:* Diese Klasse ist von `java.util.Vector` abgeleitet und definiert Namen für Objekte der Klasse `Vector`. `NamedVector` wird zur Erstellung der Baumstruktur benötigt und ist entnommen aus [Zuk00, S. 638].

- *RadioButtonUtils:* `RadioButtonUtils` wird zur einfachen Erzeugung von Radio Button-Schaltflächen verwendet. Die Klasse ist entnommen aus [Zuk00, S. 152].

In den folgenden Abschnitten wird besonders auf die Implementierung der Klasse `ArcIMSAdapter` und auf die darin verwendeten Java-Werkzeuge und Methoden zur Verarbeitung und Erzeugung von XML eingegangen. Die Klassen, mit denen die GUI erstellt wurde, werden nur am Rande behandelt, da deren Aufgaben im weiteren Verlauf dieser Arbeit durch die in GISterm entwickelte GUI übernommen werden.

5.1.2 Programmablauf

Abbildung 5.2 zeigt ein vereinfachtes Protokolldiagramm, das den Ablauf der Kommunikation zwischen Client und ArcIMS veranschaulicht. Der erste Request dient zur Abfrage der Namen der Dienste. Danach muss jeder Dienst speziell nach seinen Themen befragt werden, bevor letztendlich auf Grundlage der vorher gesammelten Informationen eine Karte für bestimmte Dienste oder Themen abgerufen werden kann.

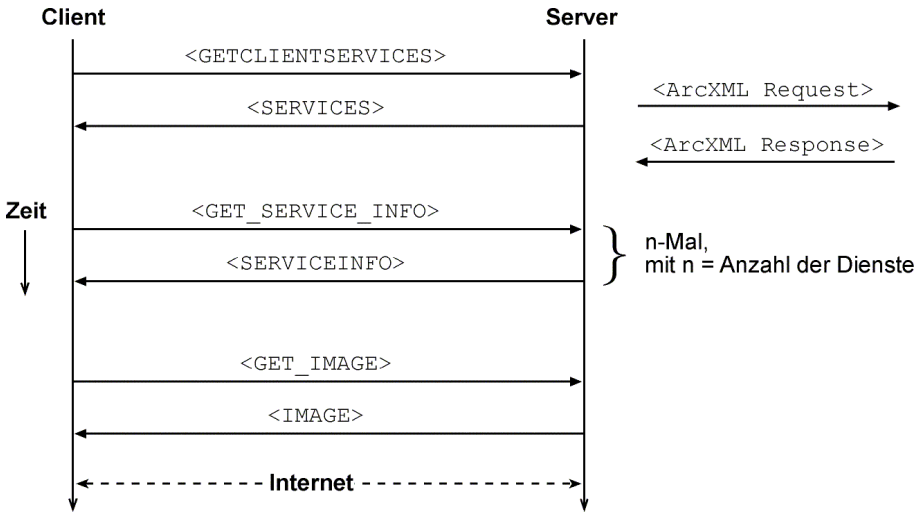


Abbildung 5.2: Protokolldiagramm der Kommunikation zwischen dem Java Client und ArcIMS.

Verbindungsaufbau

Damit der Prototyp eine Verbindung zu jedem beliebigen ArcIMS aufbauen kann, wird der Benutzer beim Starten des Programms mit einem Dialogfenster (siehe Abbildung 5.3) zur Eingabe der Verbindungsdaten aufgefordert. Dies umfasst die Festlegung von Host und URI (engl. Unified Resource Identificator, eine URL ohne Host, Port und Query-String) des Servlet Connectors. Diese Unterteilung der URL wurde vorgenommen, weil sich bei einer Standardinstallation die URLs unterschiedlicher ArcIMS nur durch den Host unterscheiden. Der Dialog gibt die URI defaultmäßig vor, so dass dem Benutzer Schreibarbeit erspart wird, da er lediglich den Host angeben muss.

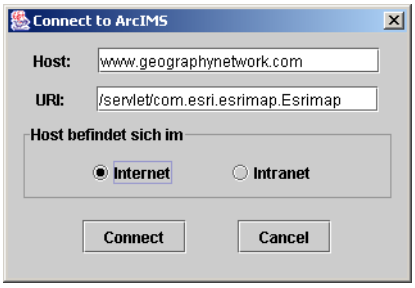


Abbildung 5.3: Dialog zur Eingabe der Verbindungsdaten.

Abschließend muss angegeben werden, ob sich der Host im Internet oder im Intranet befindet, da bei einem Zugriff auf Internetseiten ausserhalb des LfU-Intranets ein Proxyserver verwendet werden muss, der dann automatisch gesetzt wird.

Die Eingabe wird mit *Connect* bestätigt oder kann mit *Cancel* abgebrochen werden.

Abfrage von Metadaten über Dienste und Themen

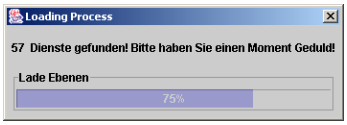


Abbildung 5.4: Dialog mit Fortschrittsbalken.

Mit Bestätigung der Eingabe folgt ein automatisch ablaufender Programmteil, der den in Abschnitt 4.2.2 beschriebenen Anwendungsfall mit ArcXML verwirklicht (siehe auch Protokolldiagramm in Abbildung 5.2 auf Seite 61). Da die Dauer dieses Prozesses von vielen Faktoren, wie z. B. der Anzahl der gefundenen Dienste und der Geschwindigkeit der Datenübertragung, abhängig ist und einige Zeit in Anspruch nehmen kann, wird zur Überbrückung der Wartezeit ein Dialog mit einem Fortschrittsbalken angezeigt, der den Anwender ebenfalls über die Anzahl der gefundenen Dienste informiert (siehe Abbildung 5.4).

Anzeige der Baumstruktur und Abruf von Karten

Bei der Auswertung der `<SERVICES>`-Response (siehe Abbildung 5.2) wählt der Client aus allen verfügbaren ArcIMS-Diensten nur Image Dienste und ArcMap Image Dienste aus, da nur Rasterbilder angezeigt werden können. Feature Dienste und Metadaten Dienste werden also nicht in der Baumstruktur aufgeführt. Auf diese Weise

werden keine unnötigen <SERVICEINFO>-Anfragen an den ArcIMS gesendet und die Wartezeit verkürzt.

Nach Abschluss der <SERVICEINFO>-Anfragen wird der BarDialog geschlossen und das Hauptfenster mit der Baumstruktur erscheint (siehe Abbildung 5.1 auf Seite 60). Durch Aufklappen der Knoten können die Themen der Dienste betrachtet werden. Im Bereich der Kartenanzeige wird zu Anfang lediglich eine Beispielgrafik angezeigt. Eine Karte wird erst abgerufen, wenn ein Knoten oder ein Blatt des Baumes ausgewählt wird.

Zur Abwicklung der Kartenanfrage erzeugt der Client einen neuen Thread, dessen Aufgabe aus dem Absetzen der Anfrage an den ArcIMS und der Anzeige der Nachricht „Loading Image“ im Kartenbereich besteht. Mit Eintreffen der Antwort zeigt der Client entweder bei Auswahl eines Knotens eine Karte mit allen Themen des Dienstes an (siehe Abbildung 5.5) oder, falls ein Blatt ausgewählt wurde, eine Karte mit einem einzelnen Thema (siehe Abbildung 5.6). Damit ist der Anwendungsfall aus Abschnitt 4.2.3 mit ArcXML verwirklicht.

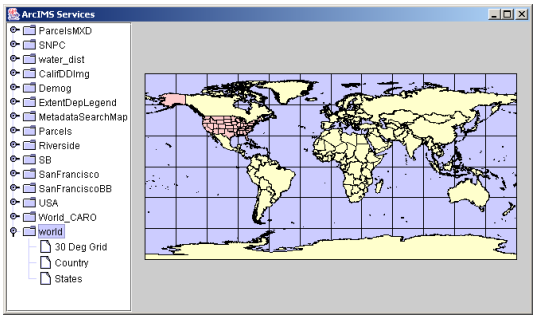


Abbildung 5.5: Anzeige eines Dienstes.

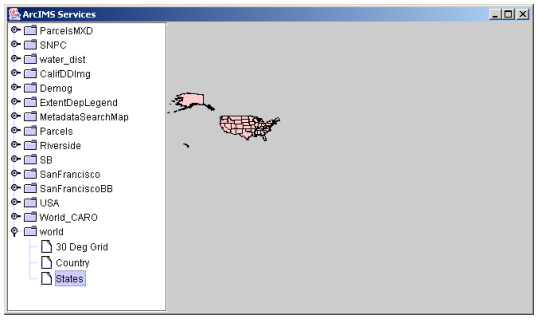


Abbildung 5.6: Anzeige eines Themas.

5.1.3 Umsetzung der Kernfunktionen

Kernfunktionen des Prototyps sind die Formulierung von ArcXML-Anfragen und die Herausfilterung relevanter Informationen aus den ArcXML-Antworten.

Zur Erledigung dieser Aufgaben wurde dom4j, ein Open Source Framework zur Verarbeitung von XML, verwendet, da es ausgereift, frei verfügbar und objektorientiert ist sowie bei den Produkten der Firma disy Informationssysteme ebenfalls zum Einsatz kommt. Das Framework kann unter [Meta03a] heruntergeladen werden.

Zur Erzeugung, Manipulation und Auswertung von XML-Dokumenten stellt dom4j eine API für das Document Object Model (DOM) zur Verfügung [Meta03c]. „DOM wurde als Verfahren entworfen, das XML-Dokumente als Baum repräsentiert“ [McLa01, S. 14], so dass alle Daten in Knoten gespeichert sind. Unter einem Knoten wird ein beliebiger Teil der XML-Daten verstanden, der z. B. Elemente, Attribute und textuelle Daten enthalten kann. Auf diese Weise kann sehr schnell

auf alle Elemente zugegriffen werden, da sich das XML-Dokument vollständig im Speicher befindet.

Im Folgenden werden die Vorgehensweisen und Werkzeuge, die zur Realisierung der Kernfunktionen eingesetzt wurden, vorgestellt.

XML erzeugen

XML-Dokumente werden in dom4j durch das Interface `Document` definiert. Zur Erstellung eines neuen `Document`-Objekts kann die Klasse `DocumentHelper` verwendet werden, da Interfaces nicht instanziiert werden können, sondern nur Schnittstellen beschreiben, also keine Implementierung besitzen.

```
Document document = DocumentHelper.createDocument();
```

Auf das Objekt `document` können nun Methoden angewendet werden, mit denen XML-Code erzeugt werden kann. XML-Tags werden durch das Interface `Element` repräsentiert. Sie bilden die Knoten des Baumes. Alle Elemente in einer Baumstruktur sind Unterelemente des Wurzelelementes (`root`), so dass dieses zuerst erzeugt werden muss. Danach werden systematisch alle Unterelemente hinzugefügt. Der folgende Code zeigt die Erstellung eines `<GET_IMAGE>`-Requests:

```
Element root = document.addElement("ARCXML")
  .addAttribute("version","1.1");
Element child1 = root.addElement("REQUEST");
Element child2 = child1.addElement("GET_IMAGE");
Element child3 = child2.addElement("PROPERTIES");
Element child4 = child3.addElement("IMAGESIZE")
  .addAttribute("width","500")
  .addAttribute("height","350");
```

XML parsen

Die Antwort des ArcIMS ist zwar in XML verfasst, zur einfachen Auswertung mit dom4j sollte sie jedoch wieder als `Document`-Objekt vorliegen. Dazu muss das XML-Dokument zuerst vollständig eingelesen und in benutzbare Teile zerlegt werden. Dieser Prozess des *Parsens* wird in dom4j von der Klasse `SAXReader` übernommen. Sie erzeugt auf Basis von Standardereignissen, beispielsweise das Treffen auf einen Element-Tag oder ein Attribut, die während des Parsing-Vorgangs auftreten können, den DOM-Baum. In der Simple API for XML (SAX), die von Java bereitgestellt und von dom4j unterstützt wird, sind diese Standardereignisse bereits definiert.
Das Parsen der Antwort ist im Prototyp wie folgt implementiert:

```
private Document parse(InputSource in) throws DocumentException {
  SAXReader reader = new SAXReader();
  return reader.read(in);
}
```

Nach der Instanziierung von `SAXReader` liefert die Methode `read()` auf Basis einer Eingabequelle (`in`) ein `Document`-Objekt zurück, das daraufhin zur Auswertung verwendet werden kann.

XML auswerten

Der nächste Schritt ist nun die Herausfilterung der benötigten Informationen aus dem `Document`-Objekt. Dies wurde in der Klasse `ArcIMSAdapter` mit Hilfe der XML Path Language (kurz XPath), die von dom4j unterstützt wird, realisiert. XPath ist eine Spezifikation, die „definiert, wie ein bestimmtes Element in einem XML-Dokument gefunden werden kann“ [McLa01, S. 10]. Zur Auswahl bestimmter Knoten (engl. Nodes) kann eine Art Pfadangabe verwendet werden. Für detaillierte Beispiele zu XPath sei auf [Zvon03] verwiesen.

Der folgende Methodenaufruf liefert eine Liste aller Elemente zurück, auf die die Pfadangabe `"//ARCXML/RESPONSE/SERVICES/SERVICE"` zutrifft:

```
List nodeList = getResponseAsDocument()
                .selectNodes("//ARCXML/RESPONSE/SERVICES/SERVICE");
```

In diesem Fall werden alle `<SERVICE>`-Elemente in der `<SERVICES>`-Response (siehe Beispielcode in Abschnitt 4.2.2 auf Seite 40) inklusive ihrer Unterelemente in einer Liste gespeichert. Interessant für den Client sind dabei nur die Attribute `name` und `type` der `<SERVICE>`-Elemente, da sie Informationen über die Namen und Servertypen der Dienste enthalten (siehe dazu auch Abschnitt 4.2.2).

Im Anschluss wird mit Hilfe einer Schleife nacheinander für diejenigen Elemente der Liste, deren `type`-Attribut nicht „FeatureServer“, „MetadataServer“ oder „RouteServer“ ist, der Wert des Attributs `name` ausgelesen und zu einer Liste mit den Namen der Dienste (`serviceNames`) hinzugefügt.

```
for (Iterator nodeIter = nodeList.iterator(); nodeIter.hasNext();) {
  Element element = (Element)nodeIter.next();
  String serverType = element.attribute("type").getValue();
  if (!serverType.equals("FeatureServer")
    && !serverType.equals("MetadataServer")
    && !serverType.equals("RouteServer")) {
    serviceNames.add(element.attribute("name").getValue());
  }
}
```

5.2 Integrationskonzept für eine WFS-Schnittstelle

In Abschnitt 4.2.8 wurde bereits umrissen, auf welche Weise Geo-Objekte mit Hilfe der WFS Spezifikation abgerufen werden können. Für diesen Prozess wird nun auf Grundlage der Anforderungen aus Abschnitt 4.2 ein Integrationskonzept erarbeitet und im Anschluss prototypisch implementiert.

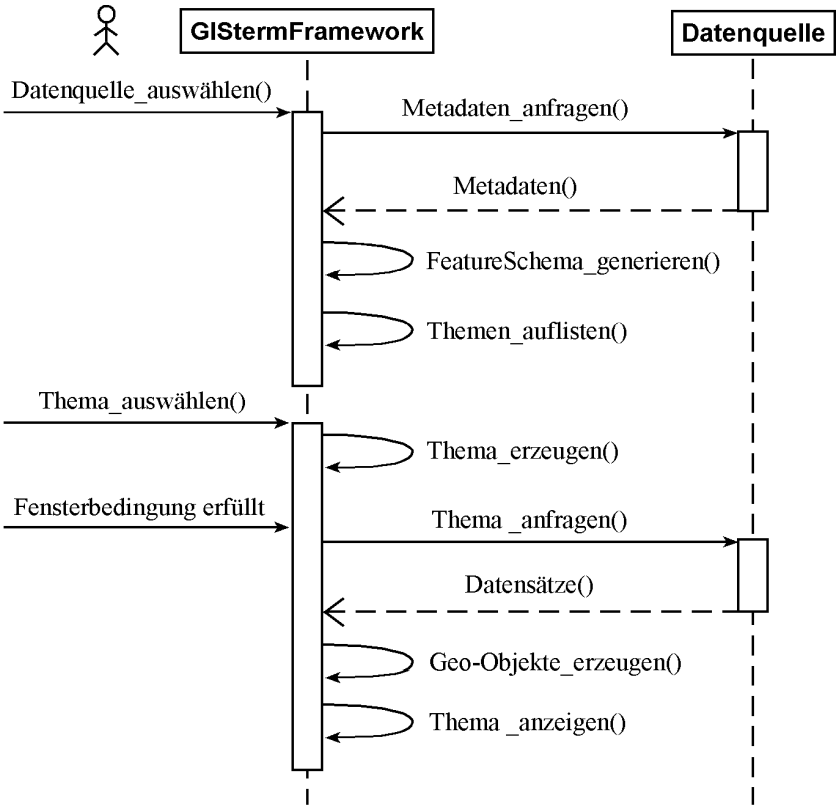


Abbildung 5.7: Ablauf der Einbindung von Datenquellen in GIStermFramework.

Den vereinfachten Ablauf der Einbindung von Datenquellen in GISterm zeigt das Sequenzdiagramm in Abbildung 5.7. Demnach erfolgt die Einbindung neuer Datenquellen in zwei Schritten:

Nach Auswahl einer Datenquelle müssen zuerst Metadaten abgerufen werden, damit ein Featureschema erzeugt werden kann und die Themen im Navigator aufgeführt werden können. Anschließend kann durch Auswahl eines Themas im Navigator eine neue Ebene in die Karte eingefügt werden. Befinden sich Geodaten des eingefügten Themas im aktuellen Kartenausschnitt, die Fensterbedingung also erfüllt ist, so werden diese automatisch angefragt und angezeigt. Ist die Bedingung nicht erfüllt, es gibt also keine Überschneidung der Bounding Boxen von Geoda-

ten und Kartenfenster, so wird keine Anfrage abgesetzt und das Thema erscheint erstmal nur in der Legende.

Bei der Integration von Geo-Objekten eines WFS in GIStermFramework ergibt sich jedoch das Problem der Inkompatibilität von Geodatenformaten und Kommunikationssprachen. GIStermFramework verwaltet raumbezogene Daten in GISterm Geo-Objekten, ein WFS beschreibt Geodaten dagegen in GML. Darüber hinaus muss bei der Kommunikation mit dem Server die WFS Spezifikation beachtet werden, die von GIStermFramework nicht unterstützt wird.

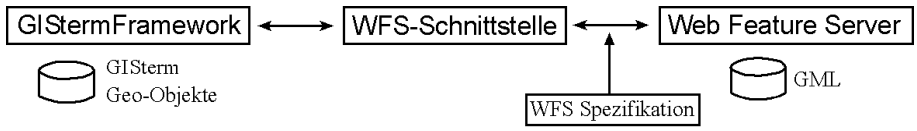


Abbildung 5.8: Integrationsschema für eine WFS-Schnittstelle.

Da eine direkte Kommunikation mit einem WFS nicht möglich ist, muss eine neu zu schaffende Schnittstelle zwischen WFS und GIStermFramework vermitteln. Sie kann daher als einen Art Adapter verstanden werden, der nicht Teil des Frameworks ist (siehe Abbildung 5.8) und folgende Aufgaben erfüllt:

- Formulierung WFS-konformer Anfragen zur Realisierung von Metadaten- und Geodatenanfrage
- Auswertung der Antworten des WFS zur Weitergabe der benötigten Daten an GIStermFramework
- Konvertierung von GML in GISterm Geo-Objekte

Mit welchen Abfragen die Metadaten eines WFS abgerufen werden können und welche Informationen sie enthalten müssen, wurde bereits in Abschnitt 4.2.2 beschrieben.

Darüber hinaus benötigt GISterm zur Überprüfung der Fensterbedingung für jedes Thema Angaben über die maximale Ausdehnung in Form einer Bounding Box. Informationen darüber enthält das `<LatLongBoundingBox>`-Element in der Capabilities-Datei des Connectors, das innerhalb der `<Layer>`-Elemente der einzelnen Themen zu finden ist (siehe Anhang A.3).

Sobald also ein WFS als Datenquelle ausgewählt wurde, übernimmt die Schnittstelle die Formulierung der Anfragen und wertet die Antworten aus. Danach werden die Metadaten in einer für GIStermFramework verständlichen Form zurückgegeben, so dass davon ein Featureschema erzeugt werden kann.

Wurde ein neues Thema in die Karte eingefügt und ist die Fensterbedingung erfüllt, so formuliert die Schnittstelle eine *GetFeature*-Anfrage wie sie in Abschnitt 4.2.8 beschrieben wurde.

Die zurückgelieferten Geodaten im GML-Format müssen nun von der Schnittstelle in GISterm Geo-Objekte konvertiert werden, damit sie von GISterm visualisiert werden können.

Den vereinfachten Ablauf einer Abfrage von Geodaten mit Hilfe einer WFS-Schnittstelle zeigt das Sequenzdiagramm in Abbildung 5.9. Da lediglich die Operationen *GetCapabilities*, *DescribeFeatureType* und *GetFeature* verwendet werden, reicht zur Realisierung dieser Aufgabe ein „Basic WFS“ aus.

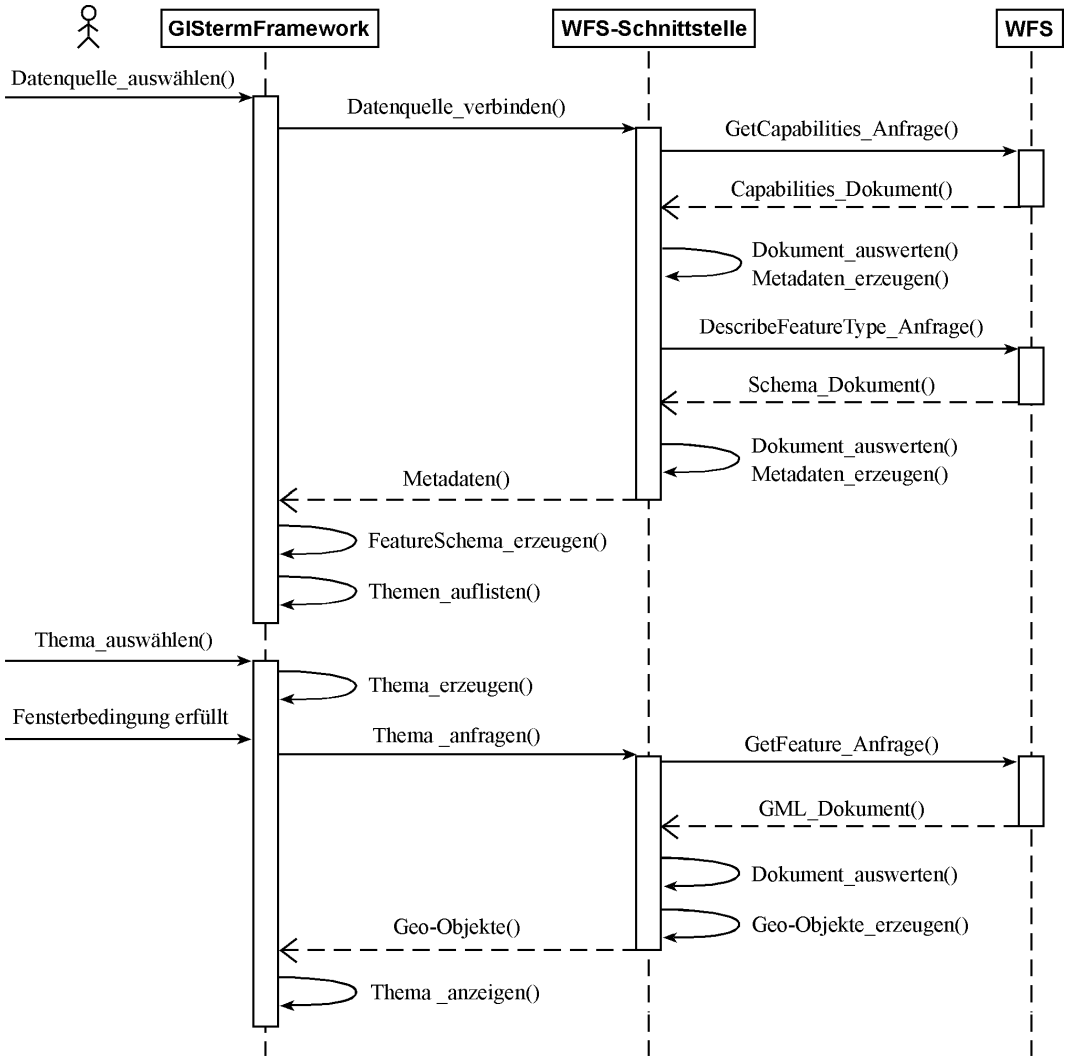


Abbildung 5.9: Ablauf einer Anfrage von WFS-Vektordaten.

5.3 Implementierung der WFS-Schnittstelle

GIStermFramework ermöglicht eine leistungsfähige Anwendungsentwicklung neben der Spezialisierung und Erweiterung bereits vorhandener Komponenten und Klassen auch durch die Verwendung bestimmter Schnittstellen (Java-Interfaces, im Folgenden daran zu erkennen, dass sie mit einem „`I`“ beginnen) und Factory Klassen, die den Umgang mit GISterm-spezifischen Strukturen vereinfachen oder sogar ganz übernehmen. Dadurch kann die Implementierung anwendungsspezifischer Funktionen zielgerichtet erfolgen, ohne dass dabei die grundlegenden Strukturen von GIStermFramework berücksichtigt werden müssen.

5.3.1 Das Interface IGIStermService

Zur Einbindung neuer Funktionen bietet GIStermFramework `IGisTermService` an. Objekte von Klassen, die dieses Interface implementieren, werden unter dem Oberbegriff *GISterm Services* zusammengefasst. Mit ihrer Hilfe kann GIStermFramework auch konfiguriert werden und Menüeinträge der Benutzeroberfläche geändert oder erweitert werden. Die Klassen sind nicht Teil des Frameworks und können daher als eine Art Plug-in verstanden werden. Damit die Service-Klassen beim Starten von GISterm instanziiert werden, müssen die Klassennamen der Plug-ins in der Konfigurationsdatei `GISterm-config.xml` eingetragen werden.

Für den Zugriff auf Kartenserver ist bereits ein GISterm Service realisiert worden: Dieser *WebMapClient* stellt jedoch lediglich einen Rahmen zur einfachen Einbindung verschiedener Kartenserver zur Verfügung, indem er einen neuen Menüeintrag erzeugt und einen Dialog zum Verbindungsaufbau definiert (siehe Abbildung 5.11). Für jeden Kartenserver muss daher noch ein weiteres Plug-in implementiert werden, das spezielle Aufgaben, wie z. B. Metadaten- und Geodatenabfrage, übernimmt.

Dazu existiert mit `AbstractWebMapClientService` eine abstrakte Basisklasse, die `IGisTermService` implementiert und gemeinsame Funktionen dieser Services zusammenfasst, sowie Methoden und Interfaces definiert, die jedes Plug-in verwenden muss. Dadurch wird eine Struktur vorgegeben, die gewährleistet, dass die Plug-ins über den WebMapClient genutzt werden können.

Da eine abstrakte Klasse nicht instanziiert werden kann, müssen die von der Basisklasse deklarierten Methoden durch konkrete Unterklassen implementiert werden. Auf diese Weise sind bereits Plug-ins für ArcIMS (`ArcImsService`) und WMS (`WmsService`) realisiert worden.

Der im Rahmen der Diplomarbeit zu erstellende GISterm Service für WFS kann ebenfalls durch Ableiten der Klasse `AbstractWebMapClientService` konkretisiert werden. Die Klasse erhält analog zu den anderen Services den Namen `WfsService`. Das Klassendiagramm in Abbildung 5.10 veranschaulicht, auf welche Weise GIStermFramework durch die Plug-ins für Kartenserver erweitert worden ist.

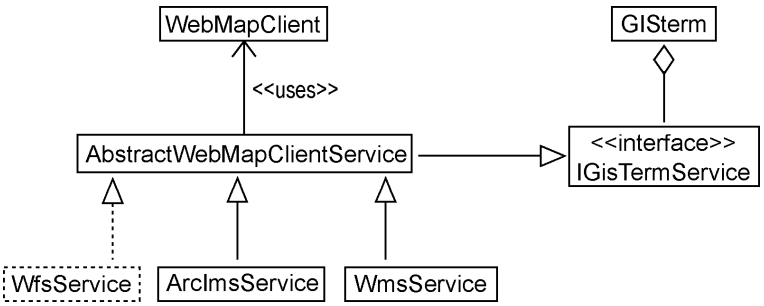


Abbildung 5.10: Klassendiagramm der WebMapClientServices.

5.3.2 Erweiterung des Kartenserver-Dialogs

Zur Einbindung von ArcIMS und WMS in GISterm kann über das Menü *GISterm -> Kartenserver-Verbindung öffnen...* der Kartenserver-Dialog aufgerufen werden (siehe Abbildung 5.11). Dieser dient zur Festlegung des Servertyps und zur Auswahl eines bestimmten Servers (Serverkonfiguration) aus einer vorgegebenen Liste. Alternativ können eigene Server spezifiziert werden und Proxyeinstellungen vorgenommen werden.

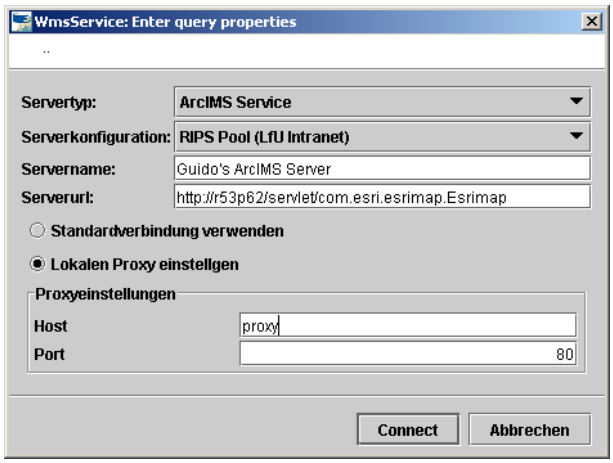


Abbildung 5.11: Dialog zur Auswahl eines Kartenservers.

Damit ein WFS ebenfalls über den Kartenserver-Dialog eingebunden werden kann, muss einerseits das Interface `IWebMapService` implementiert werden, welches Methoden zur Erweiterung des Dialogs vorgibt, andererseits muss mit Hilfe von `IWebMapClientConfiguration` auf default-Serverkonfigurationen zugegriffen werden, damit diese in der Auswahlliste im Dialog angezeigt werden können. Das Klassendiagramm in Abbildung 5.12 zeigt die vorgenommenen Erweiterungen in verein-

fachter Form, bereits vorhandene Klassen und Interfaces von GIStermFramework sind darin und in den folgenden Diagrammen grau hinterlegt.

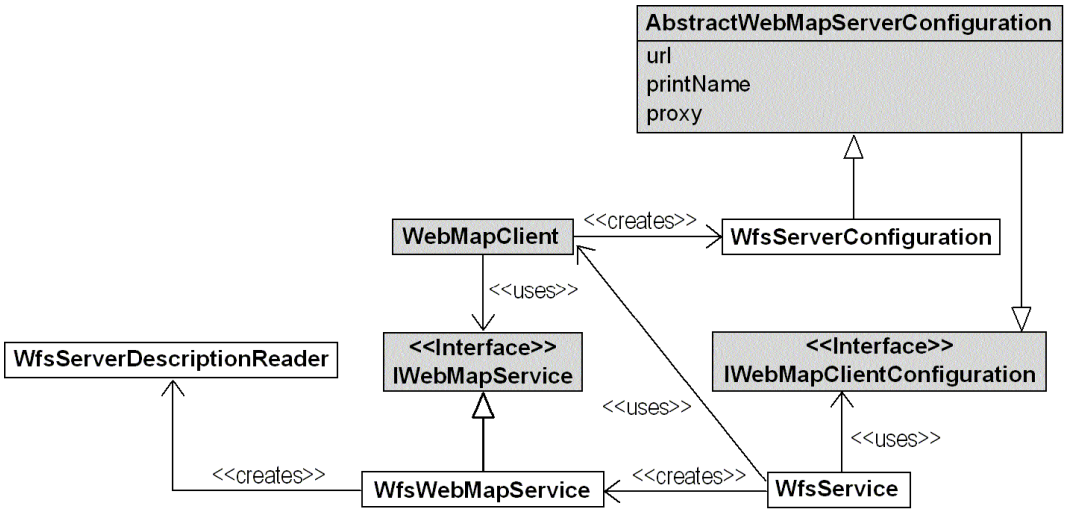


Abbildung 5.12: Vereinfachtes Klassendiagramm zur Erweiterung des Kartenserver-Dialogs.

Die Serverkonfigurationen für den `WfsService` werden durch Objekte der Klasse `WfsServerConfigurationen` verwaltet. Diese konnten von der abstrakten Basisklasse `AbstractWebMapServerConfiguration`, die ebenfalls für die Serverkonfigurationen von ArcIMS und WMS genutzt wird und `IWebMapClientConfiguration` implementiert, abgeleitet werden. Die Objekte enthalten die URL und den Namen des Servers und können entweder in der Konfigurationsdatei `WebMapClientConfig.xml` vorgegeben werden, oder aber mit Hilfe des Kartenserver-Dialogs spezifiziert werden. Dazu wurden neue Serverkonfigurationen in die Konfigurationsdatei eingetragen.

Die Klasse `WebMapClient` wird beim Starten von GISterm zunächst unabhängig von den Plug-ins für Kartenserver instanziiert. Wird nun ein Objekt von `WfsService` erzeugt, nutzt es das `WebMapClient`-Objekt um auf die WFS-Serverkonfigurationen in der Konfigurationsdatei zuzugreifen, die zur Erzeugung einer Instanz der Klasse `WfsWebMapService` benötigt werden. Die Instanz wird nun bei `WebMapClient` registriert, so dass beim Aufruf des Dialogs durch den Anwender ein neuer Servertyp („Web Feature Service (WFS)“) zur Verfügung steht. Da `WfsWebMapService` das Interface `IWebMapService` implementiert, kann das Objekt dieser Klasse durch den `WebMapClient` genutzt werden.

Nach der Auswahl eines WFS im Kartenserver-Dialog und der Bestätigung mit *Connect* wird das Objekt der Serverkonfiguration (`WfsServerConfiguration`) von `WfsWebMapService` zur Instanziierung der Klasse `WfsServerDescriptionReader` verwendet, deren Aufgabe nun der Abruf der Metadaten des Servers ist.

5.3.3 Abruf von Metadaten

Metadaten eines WFS werden in einem Objekt der Klasse `WfsServerDescription` gespeichert (siehe dazu Abbildung 5.13). Die Klasse implementiert das Interface `IWebMapServerDescription`, das Methoden und Eigenschaften definiert, mit deren Hilfe GIStermFramework ein Featureschema für Kartenserver erzeugen kann. `WfsServerDescription` enthält daher als Eigenschaft eine Liste von Objekten der Klasse `WfsLayerDescription`, so dass jedes Thema des Servers durch eines dieser Objekte repräsentiert wird. Die Klasse ist von `AbstractLayerDescription`, die das Interface `ILayerDescription` implementiert, abgeleitet, wodurch sie bestimmte Eigenschaften und Methoden erbt, die zur Erzeugung von Featuretypes benötigt werden: In der Eigenschaft `themeType` kann der Geometrietyp des Themas gespeichert werden. Die Klasse `BasicLayerProperties` enthält Eigenschaften zur Speicherung des Namens (`id`), des Titels (`printName`) und der maximalen Ausdehnung (`maxExtent`) des Themas.

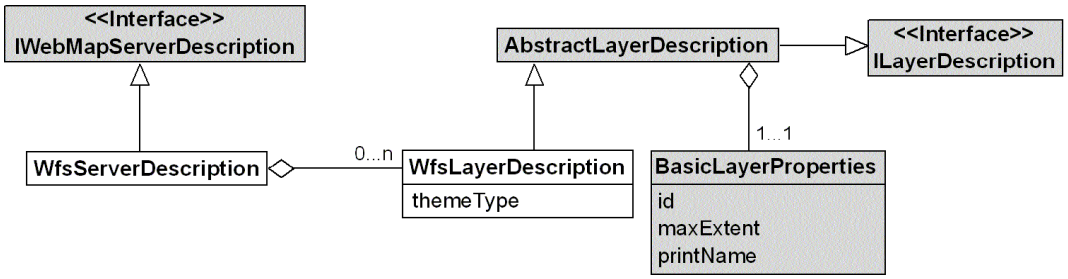


Abbildung 5.13: Klassendiagramm zur Speicherung der Metadaten.

Metadaten-Objekte werden mit Hilfe der Klasse `WfsServerDescriptionReader` erzeugt. Abbildung 5.14 zeigt das Klassendiagramm der speziell für den Abruf und die Erzeugung von Metadaten eines WFS-Servers erstellten Klassen. Die Kommunikation mit dem Server übernimmt die Klasse `WfsRequestHandler`. Zur Formulierung der Anfragen stehen zwei weitere Klassen zur Verfügung: `WfsGetRequestBuilder` kümmert sich um die Zusammensetzung des Query-Strings für HTTP-GET Requests, `WfsPostRequestBuilder` erzeugt dagegen XML-Dokumente für Anfragen mit HTTP-POST. Informationen über den Server, wie z. B. die URL, erhält der RequestHandler von einem Objekt der Klasse `WfsServerConfiguration`, das ihm bei der Instanziierung übergeben wird.

`WfsServerDescriptionReader` übernimmt die Auswertung des GetCapabilities-Dokuments, wobei ein Objekt der Klasse `WfsServerDescription` erzeugt wird.

`WfsLayerDescriptionReader` wertet die Antworten auf *DescribeFeatureType*-Requests aus und vervollständigt die Serverbeschreibung mit `WfsLayerDescription`-Objekten für jedes Thema des Servers.

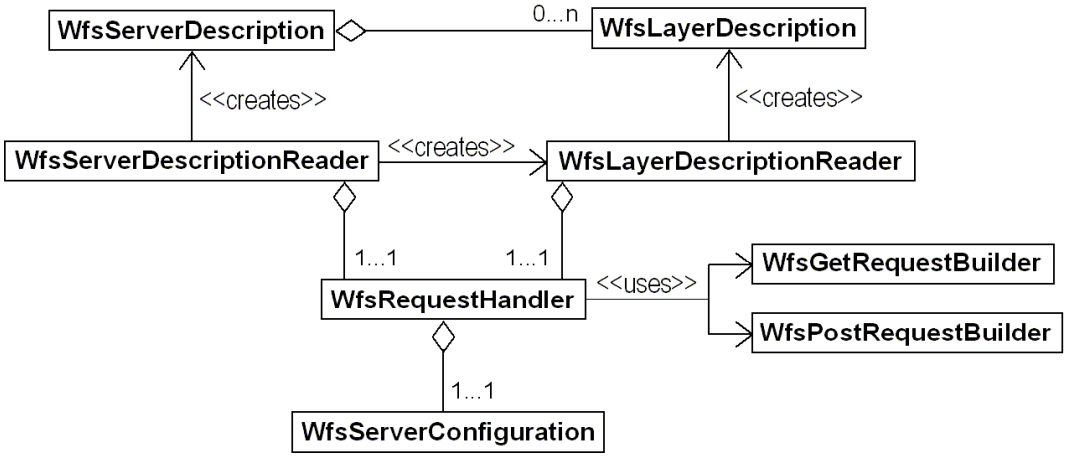


Abbildung 5.14: Klassendiagramm zum Abruf und zur Analyse der Metadaten.

Nachdem die Erzeugung der Metadaten beendet ist, wird ein Objekt der Klasse `WfsServerDescription` an GIStermFramework übergeben. Daraufhin wird davon ein Featureschema generiert und der ausgewählte Server erscheint im Navigator unter *Geodaten* -> *Zentrale Datenquellen* -> *Internetthemen* als neue Datenquelle.

5.3.4 Erzeugung eines neuen Themas

Bei der Erzeugung des Featureschemas wird jedem Featuretype eine Instanz der Klasse `WfsLayerCreationStrategy` zugeordnet. Durch die Nutzung des Interfaces `ILayerCreationStrategy` implementiert die Klasse Methoden, die bei der Auswahl eines Themas im Navigator und der Betätigung des Buttons *Thema einfügen* aufgerufen werden.

Daraufhin erzeugt die Klasse `WfsLayerCreationStrategy` ein Objekt der Klasse `WfsFeatureLayerDataProvider`, das seinerseits mit Hilfe einer Factory Klasse einen generischen FeatureLayer erzeugt (siehe Abbildung 5.15).

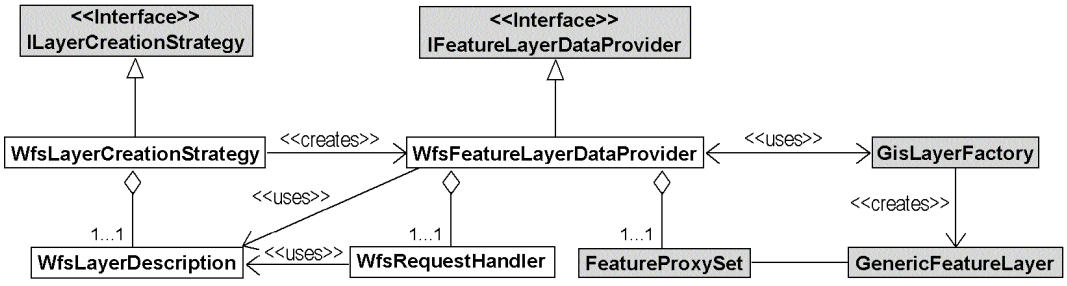


Abbildung 5.15: Klassendiagramm zur Erzeugung eines Themas.

Durch die generische Erzeugung eines MapLayers wird das Thema vorkonfiguriert in die GisView eingebunden, so dass automatisch ein Legendeneintrag erzeugt wird und sämtliche Funktionen von GISterm zur Verfügung stehen. Enthält das Thema Geo-Objekte, so können daher z. B. Objektinformationen und Attributtabellen abgerufen werden, ohne dass diese Funktionen konkret implementiert werden müssen.

5.3.5 Geodatenanfrage

Die Klasse `WfsFeatureLayerDataProvider` realisiert die Fensterbedingung, indem es die maximale Ausdehnung des Themas mit der Bounding Box des aktuellen Kartenauschnitts vergleicht. Diese Informationen erhält die Klasse zum einen durch die Implementierung des Interfaces `IFeatureLayerDataProvider`, andererseits greift sie auf die Eigenschaften von `WfsLayerDescription` zu (siehe Abbildung 5.15). Überschneiden sich die zwei Bereiche, so werden lediglich Geodaten für die Schnittmenge beider Bereiche angefordert.

`WfsFeatureLayerDataProvider` besitzt als Eigenschaft ein `FeatureProxySet`, auf das der `GenericFeatureLayer` bei seiner Erzeugung eine Referenz erhält. Ist die Fensterbedingung erfüllt, so wird mit Hilfe der Klasse `WfsRequestHandler` eine *GetFeature*-Anfrage an den WFS gestellt. Die zurückgelieferten Geodaten werden nach ihrer Konvertierung in GISterm Geo-Objekte dem `FeatureProxySet` zugewiesen. Da die Klasse `GenericFeatureLayer` das `FeatureProxySet` referenziert, werden die Geo-Objekte automatisch in GISterm visualisiert.

5.3.6 Konvertierung von GML in GISterm Geo-Objekte

Abbildung 5.16 zeigt, wie die Konvertierung von GML in GISterm Geo-Objekte realisiert worden ist. Dazu verwendet `WfsFeatureLayerDataProvider` ein Objekt der Klasse `WfsGmlDataSource`, die das GML-Dokument auswertet und mit den gewonnenen Daten einen `GmlLayer` erzeugt.

Abbildung 5.16: Klassendiagramm zur Erzeugung der Geo-Objekte.

Da das Parsen von GML-Dokumenten ein aufwendiger Vorgang ist, wurde hierfür ein bereits existierender Parser verwendet, der in der Open Source Bibliothek *GeoTools 2* unter [Geot03] gefunden wurde.

Die Klasse `GmlLayer` ist eine bereits vorhandene Erweiterung von GIStermFramework, die zur Realisierung eines Plug-ins für GML-Dokumente erstellt worden ist. Sie enthält Funktionen zur Generierung eines FeatureProxySets und kann daher die geparsten Daten verarbeiten.

Das mit Hilfe von `GmlLayer` erzeugte ProxySet wird nun dem `FeatureProxySet` von `WfsFeatureLayerDataProvider` zugewiesen, so dass die Geo-Objekte automatisch von GISterm angezeigt werden.

5.3.7 Gewährleistung der Persistenz

Eine in GISterm häufig verwendete Funktion ist die Speicherung (Persistierung) von Karten, damit der Benutzer einmal erstellte Karten bei einer späteren Sitzung nicht mühsam neu zusammenstellen muss. Sie kann über das Menü *Karte* -> *Karte speichern* aufgerufen werden.

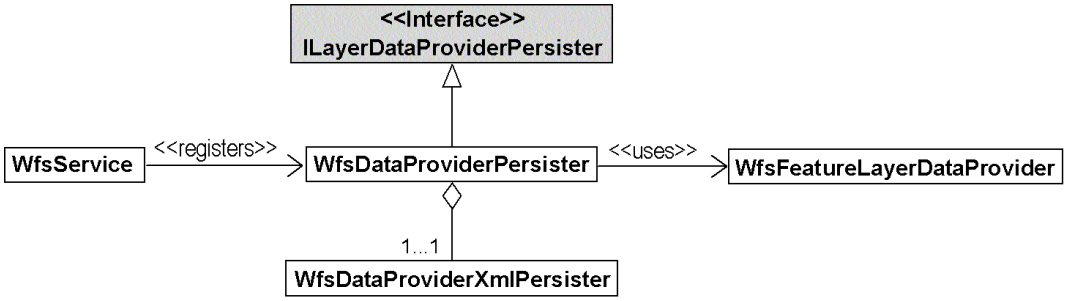


Abbildung 5.17: Klassendiagramm zur Erzeugung der Persistierungsfunktion.

Abbildung 5.17 veranschaulicht die Realisierung der Persistenz für Karten, die sich aus Themen eines WFS zusammensetzen. Erweiterungspunkt ist hierfür das Interface `ILayerDataProviderPersister`, das von `WfsDataProviderPersister` implementiert wird. Die Karteninhalte werden jedoch nicht wirklich lokal gespeichert, sondern lediglich Daten, die zur Wiederherstellung der Karte mit Hilfe von *GetFeature*-Requests benötigt werden. Dies sind zum Beispiel die URL des Servers und die Namen der Themen, die wiederhergestellt werden sollen. Die Daten werden von einem Objekt der Klasse `WfsFeatureLayerDataProvider` bereitgestellt.

Um diese Daten lokal in einer XML-Datei zu speichern, verwendet die Klasse `WfsDataProviderPersister` ein Objekt von `WfsDataProviderXmlPersister`, mit dessen Hilfe die Daten sowohl in das XML-Dokument geschrieben als auch wieder herausgelesen werden.

Damit die Speicherfunktion letztendlich über das Kartenmenü genutzt werden kann, muss noch ein Objekt der Klasse `WfsDataProviderPersister` bei GISterm registriert werden. Dies geschieht bereits beim Instanziieren der Klasse `WfsService`.

5.4 Besonderheiten und Probleme

Zum Testen des Plug-ins wurde nicht nur der ESRI WFS Connector verwendet, sondern auch WFS anderer Anbieter, so dass die allgemeine Verwendbarkeit der Schnittstelle überprüft werden konnte. Diese Server wurden mit Hilfe einer Suchmaschine im Internet gefunden und sind daher frei zugänglich. Sie können jedoch lediglich als „Testserver“ betrachtet werden, da sie nur einige Beispieldatensätze bereitstellen. Bei umfangreicheren Daten wäre der Zugriff sicherlich beschränkt.

Eingesetzt wurden WFS der Firmen CubeWerx [Cube03], Cadcorp [Cadc03], DM Solutions Group [Dmso03] und Galdos Systems [Gald03] sowie der WFS Connector für den ArcIMS des Geographynetwork Canada [Geog03]. Ein Vergleich der Capabilities-Dateien dieser Server zeigte einige Unterschiede bei den unterstützten Request-Methoden. Welche Methoden von den einzelnen Operationen unterstützt werden, zeigen die `<DCPType>`-Elemente innerhalb von `<Capability>` (siehe Anhang A.3). In Tabelle 5.1 sind diesbezüglich die Fähigkeiten der verschiedenen Server gegenübergestellt. Daraus ist zu erkennen, dass zumindest für *DescribeFeatureType*-Anfragen beide Request-Methoden implementiert werden mussten. Für *GetCapabilities*-Anfragen konnte HTTP-GET verwendet werden und für *GetFeature*-Anfragen HTTP-POST, da dies von allen Servern unterstützt wird.

5.4.1 Probleme mit Schema-Dokumenten

Bei der Auswertung des Schema-Dokuments, das bei einer *DescribeFeatureType*-Anfrage zurückliefert wird, traten Probleme auf, da die Strukturen von Dokumenten verschiedener Server leichte Unterschiede aufwiesen. So konnte beim Server der Firma Cadcorp nicht herausgefunden werden, welche Geometrietypen die einzelnen Themen beinhalten, weil das Attribut, das normalerweise den konkreten Typ (wie z. B. „PointPropertyType“) angibt, nur die allgemeine Information „GeometryPropertyType“ zur Verfügung stellte (siehe auch Anhang B.2).

Dazu kommt, dass in GML zwei Möglichkeiten existieren, um die Attribute von Geo-Objekten schematisch zu beschreiben, wodurch jedoch die Auswertung des Schema-Dokuments unnötig erschwert wird:

```
<element ref="gml:polygonProperty" minOccurs="0"/> oder
<element name="_SHAPE_" type="gml:MultiPointPropertyType"/>
```

In der Klasse `WfsLayerDescriptionReader` mussten aufgrund dieser Unterschiede Ausnahmebehandlungen durch bedingte Anweisungen implementiert werden, da-

Server	**Request-Methode**					
	HTTP-GET			HTTP-POST		
Operation	GetCapabilities	DescribeFeature	GetFeature	GetCapabilities	DescribeFeature	GetFeature
ArcIMS WFS Connector[a]	Ja	Nein	Nein	Ja	Ja	Ja
CubeSERV WFS	Ja	Ja	Ja	Ja	Ja	Ja
Cadcorp SIS WFS	Ja	Ja	Ja	Nein	Nein	Ja
UMN MapServer 4.1[b]	Ja	Ja	Ja	Ja	Ja	Ja
Galdos WFS	Ja	Nein	Nein	Ja	Ja	Ja
GeoServer 1.0.1[c]	Ja	Ja	Ja	Ja	Ja	Ja

Tabelle 5.1: Vergleich der von den verwendeten WFS unterstützten Request Methoden.

[a]Geographynetwork Canada
[b]DM Solutions Group Inc.
[c]Siehe Abschnitt 5.4.3

mit bei der Auswertung der Schema-Dokumente unterschiedlicher Server keine Fehler auftraten.

5.4.2 Probleme bei GetFeature-Anfragen

Weitere Probleme ergaben sich bei der Realisierung einer Art von Zwischenspeicher (engl. Cache) für Geo-Objekte in der Klasse `WfsFeatureLayerDataProvider`. Dieser speichert nach dem Abruf von Geodaten die IDs der Geo-Objekte, um zu verhindern, dass dieselben Objekte mehrfach angefordert werden. Dies wurde nötig, da aufgrund der in Abschnitt 5.3.5 beschriebenen Fensterbedingung die Geodaten bei der Ausführung von Verschiebungs- und Zoomfunktionen jedesmal neu angefordert wurden. Die dabei auftretenden Wartezeiten mussten auf ein erträgliches Maß reduziert werden, da die Benutzerfreundlichkeit von GISterm dadurch stark vermindert worden ist.

Dazu werden alle Feature-IDs aus dem Cache zur Erstellung eines Filters verwendet (siehe Abschnitt 2.2.5), der dann im *GetFeature*-Request benutzt wird. Damit beim Vergrößern der Karte nicht unnötigerweise eine Anfrage abgesetzt wird, wird in der Klasse `WfsFeatureLayerDataProvider` überprüft, ob der zur Vergrößerung ausgewählte Bereich im aktuellen Kartenausschnitt enthalten ist. Da ein `<FeatureId>`-Element eigentlich zur Auswahl eines bestimmten Geo-Objekts verwendet wird, müssen alle Elemente noch mit einem `<Not>`-Operator umschlossen werden, so dass die genannten Objekte nicht ausgewählt werden. Der Filter wird komplettiert, indem die Feature-IDs über `<And>` mit einem `<BBox>`-Operator verknüpft werden. Dadurch werden nur diejenigen Geo-Objekte ausgewählt, die mit der Bounding Box interagieren und nicht mit den Objekten aus dem Cache übereinstimmen. Das folgende Beispiel zeigt, wie ein solcher Filter aufgebaut sein könnte:

```
<Filter>
  <And>
    <BBox>
      <PropertyName>Geometry</PropertyName>
        <gml:Box>
          <gml:coordinates>13,31 35,42</gml:coordinates>
        </gml:Box>
    </BBox>
    <Not>
      <FeatureId fid="cities.1"/>
      <FeatureId fid="cities.2"/>
      <FeatureId fid="cities.3"/>
    </Not>
  </And>
</Filter>
```

Die Verwendung dieses Filters führte leider bei einigen der oben genannten Testserver zu Fehlermeldungen. Auch der WFS Connector konnte damit leider nicht erfolgreich angesprochen werden. Untersuchungen ergaben, dass der WFS Connector zwar eindeutige Bezeichner definiert, diese jedoch als Eigenschaften (<_ID_>) unter den Attributdaten der Geo-Objekte auftauchen, wie das folgende Beispiel zeigt:

```
<gml:featureMember>
  <cities>
    <_ID_>1</_ID_>
    <_SHAPE_> ... </_SHAPE_>
  </cities>
</gml:featureMember>
```

Nach den Vorgaben der GML sollte die ID jedoch als Attribut des <cities>-Tag aufgeführt werden. Ein WFS-konformer Server würde das gleiche Geo-Objekt wie folgt definieren:

```
<gml:featureMember>
  <cities fid="cities.1">
    <the_geom> ... </the_geom>
  </cities>
</gml:featureMember>
```

Da die Definition der Feature-IDs als Eigenschaft eine Besonderheit des WFS Connectors ist und von anderen WFS nicht unterstützt wird, müsste das Plug-in für die Verwendung des Caches speziell auf den Connector abgestimmt werden. Darunter würde jedoch dessen allgemeine Verwendbarkeit für WFS leiden.

Damit ArcIMS Geo-Objekte dennoch benutzerfreundlich und ohne Spezialisierung des Plug-ins in GISterm einbunden werden können, wurde durch Veränderung des Filters noch ein anderer Lösungsweg gefunden. Die bereits abgerufenen Geo-Objekte können durch Verwendung eines zweiten <BBox>-Elements, das die Bounding Box des aktuellen Ausschnitts enthält, beim nächsten *GetFeature*-Request ausgeschlossen werden. Dieser Operator ersetzt die <FeatureId>-Objekte des oben gezeigten Filters, so dass in der Antwort wiederum nur diejenigen Objekte zurückgegeben werden, die sich nicht im aktuellen Ausschnitt befinden.

Die neue Variante führte jedoch bei Verwendung des WFS Connectors ebenfalls zu einer Fehlermeldung, obwohl die im Filter verwendeten logischen und räumlichen Operatoren eigentlich laut Capabilities-Datei vom Connector unterstützt werden. Informationen darüber enthält der Tag <ogc:Filter_Capabilities> in Anhang A.3. Führt eines der Unterelemente die unterstützten Operatoren nicht explizit auf, wie dies z. B. bei <ogc:Spatial_Operators> der Fall ist, so bedeutet dies laut Spezifikation, dass alle Operatoren dieser Art unterstützt werden.

Da keiner der beiden Lösungswege mit dem WFS Connector realisiert werden konnte, können ArcIMS Geo-Objekte nur mit Hilfe eines einfachen <BBox>-Filters, wie er in Abschnitt 2.2.5 beschrieben wird, abgerufen werden.

5.4.3 Verwendung des GeoServers als Referenz

Damit das Plug-in dennoch auf „allgemeiner“ Basis getestet werden konnte, wurde die Referenzimplementierung der WFS Spezifikation herangezogen. Der *GeoServer* ist eine Open Source Software, die einen Transaction-WFS implementiert und unter [Geos03] heruntergeladen werden kann.

Beide in Abschnitt 5.4.2 beschriebenen Filter konnten bei Verwendung des GeoServers als Datenquelle ohne Probleme eingesetzt werden und führten zu einer deutlichen Verkürzung der Wartezeiten. Dies zeigt, dass der Aufbau der in dieser Arbeit verwendeten *GetFeature*-Anfragen fehlerfrei ist und die Probleme durch Fehler oder Besonderheiten in der Implementierung der Testserver verursacht werden.

Nach Abstimmung des Plug-ins auf den GeoServer sollte nun theoretisch jeder WFS-konforme Server, der den GeoServer als Vorlage hat, angesprochen werden können.

Für die Qualität des GeoServers spricht weiterhin seine sehr gute Performanz. Mit Hilfe des Java-Programms, das schon zum Testen des WFS Connectors und des Servlet Connectors verwendet wurde (siehe Abschnitt 4.1.7), konnte die Performanz des GeoServers bei *GetFeature*-Anfragen getestet werden. Damit die Ergebnisse mit dem Diagramm 4.2 auf Seite 35 verglichen werden können, wurden als Datenquelle für den GeoServer dieselben Vektordaten (Shapefiles) verwendet, die auch beim Anlegen des ArcIMS-Dienstes, den die beiden Connectoren während der Tests angesprochen haben, verwendet wurden. Darüber hinaus wurde der Server auf dem gleichen Rechner installiert, so dass ebenfalls eine CPU-Leistung von 935 MHz vorlag.

Die Performanz-Tests ergaben, dass der GeoServer im Durchschnitt nur 0,2 Sekunden benötigt, um ca. 600 Geo-Objekte bereitzustellen. Dies ist ein sehr erstaunlicher Wert, da Servlet Connector und WFS Connector für 500 Geo-Objekte schon 3,3 Sekunden bzw. 36,4 Sekunden benötigen. Das Ergebnis zeigt, dass vor allem die Performanz des WFS Connectors verbesserungswürdig ist.

6 Ergebnis und Perspektive

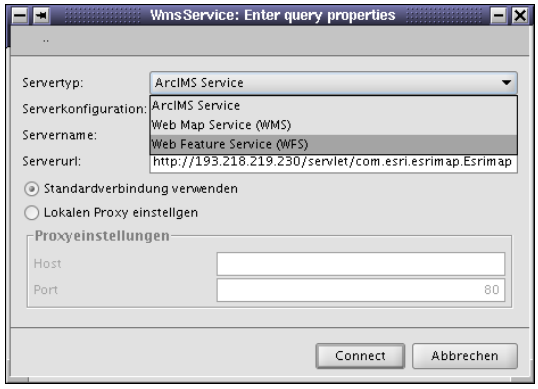


Abbildung 6.1: Liste der Servertypen.

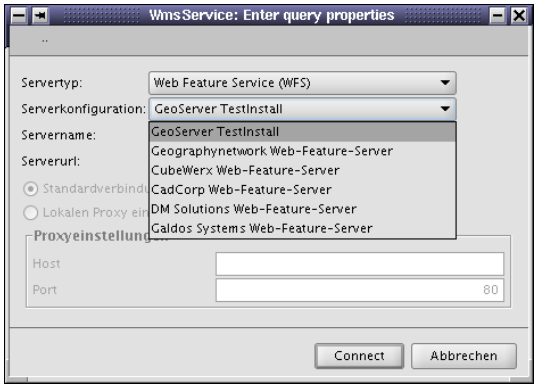


Abbildung 6.2: Liste der Default-Serverkonfigurationen.

Die angestrebte Zielsetzung konnte in ihrer Kernfunktionalität nahezu vollständig erreicht werden. Abbildung 6.1 zeigt das Kartenserver-Menü nach der Implementierung der WFS-Schnittstelle. Durch Auswahl des Servertyps „Web Feature Service (WFS)“ können nun Web Feature Server als neue Datenquelle in GISterm eingebunden werden. Abbildung 6.2 zeigt die Liste der Default-Serverkonfigurationen.

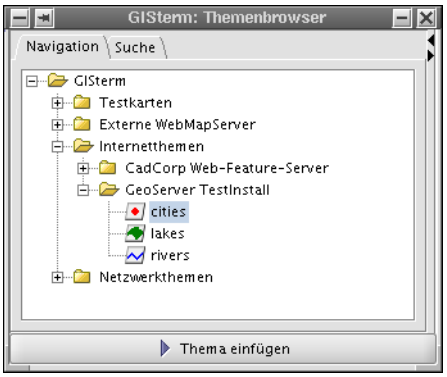


Abbildung 6.3: Anzeige von WFS-Datenquellen im Navigator.

Nach der Auswahl eines bestimmten WFS regelt das neue Plug-in selbstständig den Abruf von Metadaten, die dann im Navigator aufgeführt werden (siehe Abbildung 6.3).

Werden Themen in GISterm eingefügt, so übernimmt das Plug-in bei erfüllter Fensterbedingung den Abruf der Geodaten für den aktuellen Kartenausschnitt und konvertiert sie in GISterm Geo-Objekte. Abbildung 6.4 zeigt die Themen *cities*, *lakes* und *rivers*, die von einem GeoServer bereitgestellt wurden.

Da die eingebundenen Geo-Objekte neben Vektordaten ebenfalls Sachdaten enthalten, sind die Funktionen zur Anzeige von Attributen, wie z. B. Map Tips (siehe Abbildung 6.4) und Attributtabellen (siehe Abbildung 6.5), automatisch verwendbar.

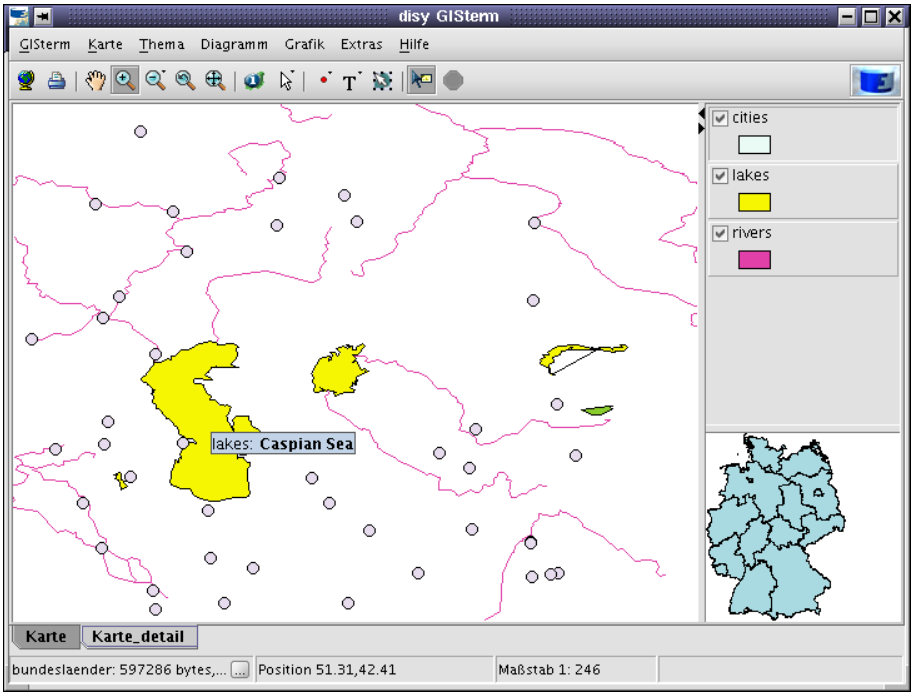
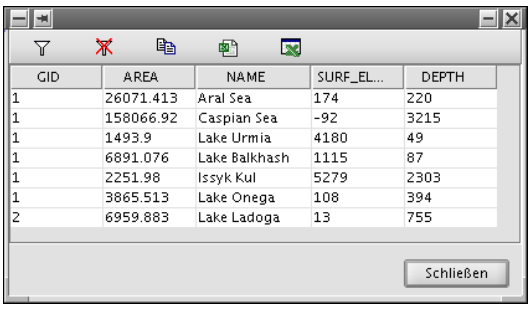


Abbildung 6.4: Darstellung von Geo-Objekten einer WFS-Datenquelle.



GID	AREA	NAME	SURF_EL...	DEPTH
1	26071.413	Aral Sea	174	220
1	158066.92	Caspian Sea	-92	3215
1	1493.9	Lake Urmia	4180	49
1	6891.076	Lake Balkhash	1115	87
1	2251.98	Issyk Kul	5279	2303
1	3865.513	Lake Onega	108	394
2	6959.883	Lake Ladoga	13	755

Schließen

Abbildung 6.5: Tabellarische Darstellung von Attributdaten.

Die Einbindung von ArcIMS Geo-Objekten konnte allerdings mit Hilfe der aktuellen Version des ESRI WFS Connectors nicht zufriedenstellend realisiert werden, da keine Möglichkeit besteht die Wartezeiten, die bei der Abfrage von WFS-Vektordaten entstehen, gering zu halten und gleichzeitig die Einbindung anderer WFS zu gewährleisten.

Aufgrund von Besonderheiten in der Implementierung des Connectors müsste eine Spezialisierung des Plug-ins vorgenommen werden, die jedoch dazu führen würde, dass es nicht mehr als allgemeine Schnittstelle für WFS verwendet werden kann.

Damit eine allgemeine Einbindung von WFS dennoch verwirklicht werden kann, gibt es zwei Möglichkeiten:

1. Die längeren Wartezeiten werden akzeptiert. Zur Abfrage der Geo-Objekte wird auf die Verwendung komplizierter Filter verzichtet und stattdessen lediglich ein einfacher `<BBox>`-Filter verwendet.

2. Bei Formulierung der Anfragen und Auswertung der Antworten werden bedingte Anweisungen verwendet, damit das Plug-in auf die Besonderheiten der verschiedenen Server reagieren kann.

Die Probleme in Abschnitt 5.4 zeigen, dass die Einbindung verschiedener WFS trotz Verwendung der WFS Spezifikation nicht ohne weiteres möglich ist, sondern für einige Server eine Feinabstimmung des Plug-ins vorgenommen werden muss. Es wird weiterhin deutlich, dass die WFS Spezifikation lediglich eine Empfehlung für Anwendungsentwickler darstellt und daher von einigen Anbietern nur teilweise umgesetzt wird. Daher ist es wichtig, dass möglichst nicht zu komplexe Funktionen verwendet werden oder vorher überprüft wird, ob die benötigten Funktionen von den Servern unterstützt werden.

Da die GML dem Entwickler offenbar ebenfalls teilweise Spielraum bei der schematischen Beschreibung der Geodaten lässt, sind bei der Einbindung unterschiedlicher WFS durch ein und dasselbe Plug-in ebenfalls Probleme vorprogrammiert.

Mit Hilfe des GeoServers konnte jedoch abschließend gezeigt werden, dass die Nutzung von WFS als Datenquelle unter Verwendung der Filter aus Abschnitt 5.4.2 sehr wohl benutzerfreundlich realisiert werden kann.

Aufgrund der bei der Verwendung des WFS Connectors aufgetretenen Probleme wird nochmals deutlich, dass dieser bis jetzt nur als Beta-Version vorliegt. Seit Sommer 2003 sind von ESRI wiederholt Ankündigungen gemacht worden, die auf eine baldige Veröffentlichung der Vollversionen von WFS Connector und WMS Connector hoffen ließen, jedoch verstrichen alle Termine, ohne dass etwas passierte. Sollten in Zukunft wirklich neue Versionen der Connectoren zur Verfügung stehen, so muss neu überprüft werden, ob die bisher nicht anwendbaren Filter von ihnen unterstützt werden.

Interessant wäre auch zu überprüfen, ob die in Abschnitt 4.1.7 erwähnte Performanz bei *GetFeature*-Anfragen verbessert werden konnte, was die Wartezeiten beim Abruf von Geo-Objekten weiter verkürzen würde.

Das Plug-in unterstützt bis jetzt nur die Operationen eines Basic WFS. Da GISterm auch Funktionen zur Editierung von Geo-Objekten anbietet, könnte daher eine sinnvolle Erweiterung in der Realisierung von Transaktionsfunktionen bestehen, so dass auch Transaction WFS von GISterm genutzt werden können. Dazu müssten *Transaction*-Anfragen und *LockFeature*-Anfragen (siehe Abschnitt 2.2.4) implementiert und mit den Editierfunktionen in GISterm verknüpft werden.

7 Zusammenfassung

Erst seit wenigen Jahren nutzen auch Geografische Informationssysteme (GIS) die Technologien der Informationstechnik, um Geodaten und GIS-Funktionen mit Hilfe von Diensten über das Internet anzubieten. Diese WebGIS-Anwendungen, sogenannte Map Server, haben daher in kurzer Zeit eine rasante Entwicklung mitgemacht, so dass der Markt mittlerweile viele unterschiedliche Produkte anbietet.

Weil die meisten dieser Anwendungen ihr eigenes, proprietäres Datenformat unterstützen, mussten Standardschnittstellen festgelegt werden, mit deren Hilfe die Integration von Geodaten und WebGIS-Funktionen in andere Produkte realisiert werden kann. Das OpenGIS Consortium (OGC) definiert dazu die Web Map Service Interface Implementation Specification (WMS) für die Bereitstellung und Analyse von Rasterbildern und die Web Feature Service Interface Implementation Specification (WFS) für die Übertragung und Manipulation von Vektordaten.

Eines der zahlreichen WebGIS-Produkte ist der Internet Map Server ArcIMS der Firma ESRI. Aufgabe dieser Diplomarbeit war die Einbindung von ArcIMS-Diensten in GISterm, der GIS-Auswertungskomponente des Berichtssystems, das von der Landesanstalt für Umweltschutz Baden-Württemberg zur übergreifenden Recherche und Analyse von umweltbezogenen Fach- und Geodaten verwendet wird.

Zuerst musste untersucht werden, welche der von ArcIMS angebotenen Programmierschnittstellen am besten für diese Aufgabe geeignet ist. Dazu musste eine Auswahl aus den proprietären Schnittstellen Java, ActiveX, ColdFusion und Servlet Connector sowie den beiden Standardschnittstellen WMS Connector bzw. WFS Connector getroffen werden.

Um den Arbeitsaufwand zu verringern, wurde zur Entscheidungsfindung zuerst eine Vorauswahl getroffen, indem alle Schnittstellen auf die einfache und kostenlose Anbindung an Java, der Implementierungssprache von GISterm, untersucht wurden. Dies führte zur Aussortierung der Java, ColdFusion und ActiveX Connectoren.

Daraufhin wurden die restlichen Programmierschnittstellen eingehender auf die Unterstützung GISterm-spezifischer Anforderungen untersucht. Dabei wurde festgestellt, dass nur der Servlet Connector in Verbindung mit der proprietären Kommunikationssprache ArcXML alle Anforderungen für die Integration von ArcIMS-Rasterbildern in GISterm erfüllt.

Zur Einbindung von ArcIMS-Vektordaten ist dagegen der WFS Connector die bessere Wahl, da dieser die Daten in der Geography Markup Language (GML) verschlüsselt und dieses Format von GISterm bereits unterstützt wird. Darüber hinaus können mit dieser Schnittstelle nicht nur ArcIMS-Vektordaten eingebunden werden,

sondern auch Vektordaten anderer Map Server, die ebenfalls die WFS Spezifikation unterstützen.

Der nächste Teil der Arbeit befasste sich zuerst mit der Implementierung eines Java Clients, der prototypisch den Abruf und die Anzeige von Metadaten und Rasterbildern beliebiger ArcIMS aufzeigt.

Da zwischenzeitlich bereits eine GISterm-Schnittstelle für ArcIMS-Rasterbilder geschaffen worden war, erfolgte danach die Konzeption einer Schnittstelle zur Einbindung WFS-konformer Map Server. Es wurde beschrieben, wie Informationen über den Datenbestand eines Servers mit Hilfe WFS-konformer Anfragen abgerufen werden können. Aus diesen Metadaten kann GISterm eine Liste der verfügbaren Kartenthemen generieren, so dass ein GISterm-Nutzer bestimmte Themen auswählen und somit zur Darstellung bringen kann. Für diesen Vorgang ist einerseits eine erneute Anfrage zum Abruf der benötigten Geodaten zu stellen, andererseits müssen die daraufhin von ArcIMS bereitgestellten Daten in das Datenformat von GISterm konvertiert werden.

Im Anschluss daran konnte dieses Konzept in der Programmierspache Java umgesetzt werden. Leider nicht ganz zufriedenstellend, da bestimmte Funktionen, die den Vorgang des Datenabrufs beschleunigen sollten, aufgrund von Fehlfunktionen der ArcIMS-Schnittstelle sowie Abweichungen von der WFS Spezifikation nicht realisiert werden konnten. Dadurch wurde deutlich, dass es sich beim WFS Connector für ArcIMS um eine nicht ausgereifte Testversion (Beta-Version) handelt. Damit die Integration von ArcIMS-Vektordaten benutzerfreundlich realisiert werden kann, muss auf das Erscheinen der Vollversion gewartet werden.

Anhang A

Capabilities-Dateien

A.1 Capabilities des ESRI WMS Connectors

```
<WMT_MS_Capabilities version="1.1.0" updateSequence="0">
  <Service>
    <Name>OGC:WMS</Name>
    <Title>WMS Map Server</Title>
    <Abstract>WMS Map Server</Abstract>
    <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
      xlink:type="simple" xlink:href="http://r53test:80
      /servlet/com.esri.wms.Esrimap?" />
    <Fees />
    <AccessConstraints />
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:type="simple" xlink:href="http://r53test:80
                /servlet/com.esri.wms.Esrimap?" />
            </Get>
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <GetMap>
        <Format>image/jpeg</Format>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <DCPType>
          <HTTP>
```

```
            <Get>
              <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:type="simple" xlink:href="http://r53test:80
                /servlet/com.esri.wms.Esrimap?" />
            </Get>
          </HTTP>
        </DCPType>
      </GetMap>
      <GetFeatureInfo>
        <Format>application/vnd.ogc.wms_xml</Format>
        <Format>text/plain</Format>
        <Format>text/html</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:type="simple" xlink:href="http://r53test:80
                /servlet/com.esri.wms.Esrimap?" />
            </Get>
          </HTTP>
        </DCPType>
      </GetFeatureInfo>
    </Request>
    <Exception>
      <Format>application/vnd.ogc.se_xml</Format>
      <Format>application/vnd.ogc.se_inimage</Format>
      <Format>application/vnd.ogc.se_blank</Format>
    </Exception>
    <Layer queryable="1" opaque="0" noSubsets="0">
      <Title>world</Title>
      <SRS>EPSG:4326</SRS>
      <Layer queryable="1" opaque="0" noSubsets="0">
        <Name>cities</Name>
        <Title>cities</Title>
        <SRS>EPSG:4326</SRS>
        <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
      </Layer>
    </Layer>
  </Capability>
</WMT_MS_Capabilities>
```

A.2 Capabilities des ESRI WMS Connectors 2

```
<?xml version="1.0" encoding="utf-8" ?>
<WMT_MS_Capabilities version="1.1.0">
  <Service>
    <Name>OGC:WMS</Name>
    <Title>ESRI Co. Map Server</Title>
    <Abstract>OGC - Web Map Server.</Abstract>
    <KeywordList>
      <Keyword>ArcIMS</Keyword>
      <Keyword>ESRI</Keyword>
      <Keyword>OGC Map Service</Keyword>
    </KeywordList>
    <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
      xlink:type="simple" xlink:href="http://www.esri.com"/>
    <Fees>none</Fees>
    <AccessConstraints>none</AccessConstraints>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
          <DCPType>
            <HTTP>
              <Get>
                <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                  xlink:type="simple"
                  xlink:href="http://r53p62:8080/ogcwms
                  /servlet/com.esri.ogc.wms.WMSServlet?SERVICENAME=world&"/>
              </Get>
            </HTTP>
          </DCPType>
       </GetCapabilities>
       <GetMap>
         <Format>image/png</Format>
         <Format>image/jpeg</Format>
         <DCPType>
           <HTTP>
             <Get>
               <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                 xlink:type="simple" xlink:href="http://r53p62:8080/ogcwms
                 /servlet/com.esri.ogc.wms.WMSServlet?SERVICENAME=world&"/>
             </Get>
           </HTTP>
```

```
        </DCPType>
      </GetMap>
      <GetFeatureInfo>
        <Format>application/vnd.ogc.gml</Format>
        <Format>text/plain</Format>
        <Format>text/html</Format>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                xlink:type="simple" xlink:href="http://r53p62:8080/ogcwms/
                servlet/com.esri.ogc.wms.WMSServlet?SERVICENAME=world&amp;"/>
            </Get>
          </HTTP>
        </DCPType>
      </GetFeatureInfo>
    </Request>
    <Exception>
      <Format>application/vnd.ogc.se_xml</Format>
      <Format>application/vnd.ogc.se_inimage</Format>
      <Format>application/vnd.ogc.se_blank</Format>
     </Exception>
     <VendorSpecificCapabilities/>
     <UserDefinedSymbolization SupportSLD="1" UserLayer="0"
       UserStyle="1" RemoteWFS="0"/>
     <Layer queryable="1" cascaded="0" opaque="1"
          noSubsets="0" fixedWidth="0" fixedHeight="0">
       <Name>cities</Name>
       <Title>cities</Title>
       <LatLonBoundingBox minx="-165,2700" miny="-53,1500"
            maxx="177,1301" maxy="78,1999"/>
     </Layer>
   </Layer>
  </Capability>
</WMT_MS_Capabilities>
```

A.3 Capabilities des ESRI WFS Connectors

```
<?xml version="1.0" encoding="UTF-8" ?>
<WFS_Capabilities
  xmlns="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc" version="1.0.0">
  <Service>
    <Name>ESRI_WFS</Name>
    <Title>ESRI OGC compliant Basic WFS Server</Title>
    <Abstract>This is an OGC compliant MapService served By ESRI.
      The engine used to serve this MapService is ArcIMS</Abstract>
    <Keywords>ESRI, ArcIMS Feature MapService</Keywords>
    <OnlineResource>http://r53p62:8080/ogcwfs
      /servlet/com.esri.ogc.wfs.WFSServlet?</OnlineResource>
    <Fees>none</Fees>
    <AccessConstraints>none</AccessConstraints>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <DCPType>
          <HTTP>
            <Get onlineResource="http://r53p62:8080/ogcwfs
              /servlet/com.esri.ogc.wfs.WFSServlet?" />
          </HTTP>
        </DCPType>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://r53p62:8080/ogcwfs
              /servlet/com.esri.ogc.wfs.WFSServlet?" />
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <DescribeFeatureType>
        <SchemaDescriptionLanguage>
          <XMLSCHEMA />
        </SchemaDescriptionLanguage>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://r53p62:8080/ogcwfs
              /servlet/com.esri.ogc.wfs.WFSServlet?" />
          </HTTP>
        </DCPType>
      </DescribeFeatureType>
```

```
      <GetFeature>
        <ResultFormat>
          <GML2 />
        </ResultFormat>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://r53p62:8080/ogcwfs
              /servlet/com.esri.ogc.wfs.WFSServlet?" />
          </HTTP>
        </DCPType>
      </GetFeature>
    </Request>
    <VendorSpecificCapabilities />
  </Capability>
  <FeatureTypeList>
    <Operations>
      <Query />
    </Operations>
    <FeatureType>
      <Name>cities</Name>
      <Title>cities</Title>
      <SRS>EPSG:4326</SRS>
      <LatLongBoundingBox minx="-165,2700" miny="-53,1500"
        maxx="177,13018" maxy="78,1999" />
    </FeatureType>
  </FeatureTypeList>
  <ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities>
      <ogc:Spatial_Operators>
        <ogc:BBOX />
        <ogc:Intersects />
        <ogc:Disjoint />
      </ogc:Spatial_Operators>
    </ogc:Spatial_Capabilities>
    <ogc:Scalar_Capabilities>
      <ogc:Logical_Operators />
      <ogc:Comparison_Operators>
        <ogc:Simple_Comparisons />
      </ogc:Comparison_Operators>
    </ogc:Scalar_Capabilities>
  </ogc:Filter_Capabilities>
</WFS_Capabilities>
```

Anhang B

Schema-Dokumente

B.1 Schema-Dokument des ESRI WFS Connectors (Bsp.)

```
<schema targetNamespace="http://www.esri.com/WFS"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:wfs="http://www.esri.com/WFS" elementFormDefault="qualified">
 <import namespace="http://www.opengis.net/gml"
   schemaLocation="feature.xsd" />
 <element name="Features" type="wfs:featuresType"
   substitutionGroup="gml:_FeatureCollection" />
 <complexType name="featuresType">
   <complexContent>
     <extension base="gml:AbstractFeatureCollectionType" />
   </complexContent>
 </complexType>
 <element name="cities" type="wfs:citiesType"
   substitutionGroup="gml:_Feature" />
  <complexType name="citiesType">
    <complexContent>
      <extension base="gml:AbstractFeatureType">
        <sequence>
          <element name="_ID_" type="integer" />
          <element name="_SHAPE_" type="gml:MultiPointPropertyType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</schema>
```

B.2 Schema-Dokument des Cadcorp SIS WFS (Bsp.)

```
<xsd:schema targetNamespace="http://www.cadcorpdev.co.uk/wfs/wfs"
  elementFormDefault="qualified" version="0.1">
  <xsd:import namespace="http://www.opengis.net/gml"
    schemaLocation="http://www.cadcorpdev.co.uk/wfs/SisISAPI.dll
    ?GetSchema&name=feature.xsd"/>
  <xsd:element name="EDINBURGH---0" type="wfs:EDINBURGH---0_Type"
    substitutionGroup="gml:_Feature"/>
  <xsd:complexType name="EDINBURGH---0_Type">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element name="EDINBURGH---0.GML_Geometry"
            type="gml:GeometryPropertyType"
            nillable="false" minOccurs="0" maxOccurs="1"/>
          <xsd:element name="EDINBURGH---0.Type" type="xsd:string"
            nillable="false" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

B.3 Schema-Dokument des GeoServers (Bsp.)

```
<xs:schema targetNamespace="http://www.openplans.org/topp"
  xmlns:topp="http://www.openplans.org/topp"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="1.0">
  <xs:import namespace="http://www.opengis.net/gml"
    schemaLocation="http://localhost:8080/geoserver/data/
    capabilities/gml/2.1.2/feature.xsd" />
  <xs:element name="cities" type="topp:cities_Type"
    substitutionGroup="gml:_Feature" />
  <xs:complexType name="cities_Type">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="the_geom" type="gml:MultiPointPropertyType"
            minOccurs="0" maxOccurs="1" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Literaturverzeichnis

[ASF03] Apache Software Foundation (2003): Apache Tomcat. http://jakarta.apache.org/tomcat/, besucht im Aug. 2003.

[BiFr94] Bill, Ralf und Dieter Fritsch (1994): Grundlagen der Geo-Informationssysteme. Band 1. Hardware, Software und Daten. 2. Auflage, Herbert Wichmann Verlag, Heidelberg.

[BiZe01] Bill, Ralf und Marco L. Zehner (2001): Lexikon der Geoinformatik. Herbert Wichmann Verlag, Hüthig GmbH & Co. KG, Heidelberg.

[Cadc03] Computer Aided Development Corporation (Cadcorp) (2003): Cadcorp SIS WFS. http://www.cadcorpdev.co.uk/wfs/SisISAPI.dll, besucht im Nov. 2003.

[Cube03] CubeWerx Inc. (2003): CubeSERV WFS. http://demo.cubewerx.com/demo/cwwfs/cwwfs.cgi, besucht im Nov. 2003.

[Dick01] Dickmann, Frank (2001): Compass. Das Geographische Seminar. Web-Mapping und Web-GIS. 1. Auflage, Westermann Schulbuch GmbH, Braunschweig.

[Disy03a] disy Informationssysteme GmbH (2003): disy Cadenza Professional V2.9.

[Disy03b] disy Informationssysteme GmbH (2003): GISterm javadoc. unveröffentlicht.

[Disy03c] disy Informationssysteme GmbH (2003): Produktblatt zu disy Cadenza Plattform. http://www.disy.net/de/Produkte/disy_Cadenza/index.html, besucht im Dez. 2003.

[Disy02] disy Informationssysteme GmbH (2002): Entwicklerdokumentation GIStermFramework V2.7.

[Dmso03] DM Solutions Group Inc. (2003): UMN MapServer 4.1 WFS. http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap, besucht im Nov. 2003.

[ESRI02a] Environmental Systems Research Institute (ESRI) (2002): Using ArcIMS. Redlands, California.

[ESRI02b] Environmental Systems Research Institute (ESRI) (2002): Customizing ArcIMS. Java Connector. Redlands, California.

[ESRI02c] Environmental Systems Research Institute (ESRI) (2002): Customizing ArcIMS. ActiveX Connector. Redlands, California.

[ESRI02d] Environmental Systems Research Institute (ESRI) (2002): Customizing ArcIMS. ColdFusion Connector. Redlands, California.

[ESRI02e] Environmental Systems Research Institute (ESRI) (2002): ArcIMS 4. ArcXML Programmer's Reference Guide. Redlands, California.

[ESRI03a] Environmental Systems Research Institute (ESRI) Geoinformatik GmbH (2003): `http://www.esri-germany.de/products/arcims/index_m.html`, besucht im Nov. 2003.

[ESRI03b] Environmental Systems Research Institute (ESRI) Geoinformatik GmbH (2003): `http://www.esri-germany.de/products/arcims/arcxml/index_m.html`, besucht im Nov. 2003.

[ESRI03c] Environmental Systems Research Institute (ESRI) (2003): ArcIMS 4 Architecture and Functionality. An ESRI White Paper. Redlands, California.

[ESRI03d] Environmental Systems Research Institute (ESRI) (2003): WMS Connector for ArcIMS. `http://gis.esri.com/download/index.cfm?downloadid=305`, besucht im Juli 2003.

[ESRI03e] Environmental Systems Research Institute (ESRI) (2003): WFS Connector for ArcIMS `http://gis.esri.com/download/index.cfm?downloadid=306`, besucht im Juli 2003.

[ESRI03f] Environmental Systems Research Institute (ESRI) (2003): Diskussionsforum Interoperability and Standards. `http://forums.esri.com/Thread.asp?c=132`, besucht von Aug. 2003 bis Jan. 2004.

[ESRI03g] Environmental Systems Research Institute (ESRI) (2003): Performanz des WFS Connectors. Diskussionsforum Interoperability and Standards: OGC Connectors for ArcIMS Forum. `http://forums.esri.com/Thread.asp?c=132&f=1343&t=95374&mc=9#264733`, besucht im Aug. 2003.

[ESRI03h] Environmental Systems Research Institute (ESRI) (2003): ArcIMS Software Diskussionsforum: WMS Connector Forum. http://forums.esri.com/forums.asp?c=64&h=784#784, besucht im Aug. 2003.

[ESRI03i] Environmental Systems Resaerch Institute (ESRI) (2003): sendArcXML HTML Viewer. http://support.esri.com/index.cfm?fa=downloads.samplesUtilities.viewSample&PID=16&MetaID=70, besucht im Aug. 2003.

[ESRI03k] Environmental Systems Resaerch Institute (ESRI) (2003): Using the Java Connector - An Introduction. http://support.esri.com/index.cfm?fa=downloads.samplesUtilities.viewSample&PID=16&MetaID=177, besucht im Juli. 2003.

[Geot03] GeoTools 2 (2003): GmlDataSource. http://modules.geotools.org/gmldatasource/, besucht im Nov. 2003.

[Geos03] The GeoServer Project (2003): GeoServer 1.0.1. http://geoserver.sourceforge.net/html/index.php, besucht im Nov. 2003.

[Geog03] Geographynetwork Canada (2003): WFS Connector für ArcIMS. http://dev.geographynetwork.ca/ogcwfs/servlet/com.esri.ogc.wfs.WFSServlet, besucht im Nov 2003.

[Gald03] Galdos Systems Inc. (2003): Galdos WFS. http://wfs.galdosinc.com:8680/wfs/http, besucht im Nov. 2003.

[Haro02] Harold, Elliotte Rusty (2002): Processing XML with Java. http://cafeconleche.org/books/xmljava/chapters/index.html, besucht im Sep. 2003.

[HGM01] Hake, Günter, Dietmar Grünreich und Liqiu Meng (2002): Kartographie. Visualisierung raum-zeitlicher Informationen. 8. Auflage, Walter de Gruyter & Co., Berlin.

[Macr03] Macromedia, Inc. (2003): Macromedia ColdFusion MX 6.1. http://www.macromedia.com/software/coldfusion/, besucht im Dez. 2003.

[McLa01] McLaughlin, Brett (2001): Java und XML. Deutsche Ausgabe. O'Reilly Verlag, Köln.

[Meta03a] MetaStuff, Ltd. (2003): dom4j Version 1.4. http://dom4j.org/, besucht im Sep. 2003.

[Meta03b] MetaStuff, Ltd. (2003): dom4j Version 1.4. Quick Start. http://dom4j.org/guide.html, besucht im Sep. 2003.

[Meta03c] MetaStuff, Ltd. (2003): dom4j Version 1.4. JavaDoc. `http://dom4j.org/apidocs/index.html`, besucht im Sep. 2003.

[OGC99] OpenGIS Consortium (OGC) (1999): The OpenGIS Simple Features Specification For SQL, Revision 1.1. OpenGIS Consortium, Wayland, Massachusetts.

[OGC00] OpenGIS Consortium (OGC) (2000): The OpenGIS Web Map Server Interface Implementation Specification, Revision 1.1.0. OpenGIS Consortium, Wayland, Massachusetts.

[OGC01a] OpenGIS Consortium (OGC) (2001): The OpenGIS Web Map Server Interface Implementation Specification, Version 1.1.0. OpenGIS Consortium, Wayland, Massachusetts.

[OGC01b] OpenGIS Consortium (OGC) (2001): Geography Markup Language (GML) 2.0. OpenGIS Implementation Specification. OpenGIS Consortium, Wayland, Massachusetts.

[OGC01c] OpenGIS Consortium (OGC) (2001): The OpenGIS Filter Encoding Implementation Specification, Version 1.0.0. OpenGIS Consortium, Wayland, Massachusetts.

[OGC02a] OpenGIS Consortium (OGC) (2002): The OpenGIS Web Map Server Interface Implementation Specification, Version 1.1.1. OpenGIS Consortium, Wayland, Massachusetts.

[OGC02b] OpenGIS Consortium (OGC) (2002): The OpenGIS Web Feature Server Interface Implementation Specification, Version 1.0.0. OpenGIS Consortium, Wayland, Massachusetts.

[OGC02c] OpenGIS Consortium (OGC) (2002): The OpenGIS Abstract Specification. Topic 12: OpenGIS Service Architecture. OpenGIS Consortium, Wayland, Massachusetts.

[OGC02d] OpenGIS Consortium (OGC) (2002): The OpenGIS Styled Layer Descriptor Implementation Specification, Version 1.0.0. OpenGIS Consortium, Wayland, Massachusetts.

[OGC03] OpenGIS Consortium (OGC) (2003): Homepage des OpenGIS Consortiums. `http://www.opengis.org`, besucht im Juli 2003.

[PeTs03] Peng, Zhong-Ren und Ming-Hsiang Tsou (2003): Internet GIS. Distributed Geographic Information Services for the Internet and Wireless Networks. John Wiley & Sons, Inc., Hoboken, New Jersey.

[Sun04] Sun Microsystems, Inc. (2004): Java Technology. `http://java.sun.com`, besucht im Jan. 2004.

[Sun03a] Sun Microsystems, Inc. (2003): Java™ 2 Platform, Standard Edition, v 1.4.1. API Specification. `http://java.sun.com/j2se/1.4.1/docs/api/index.html`, besucht im Sep. 2003.

[Sun03b] Sun Microsystems, Inc. (2003): Using a Swing Worker Thread. `http://java.sun.com/products/jfc/tsc/articles/threads/threads2.html#the_enhanced`, besucht im Sep. 2003.

[Sun04c] Sun Microsystems, Inc. (2004): JavaBeans. `http://java.sun.com/products/javabeans/`, besucht im Jan. 2004.

[Sun04d] Sun Microsystems, Inc. (2004): JavaServer Pages Technology. `http://java.sun.com/products/jsp/`, besucht im Jan. 2004.

[Team01a] Teamone (2001): SelfHTML 8.0. ActiveX. `http://selfhtml.teamone.de/intro/technologien/activex.htm`, besucht im Dez. 2003.

[Team01b] Teamone (2001): SelfHTML 8.0. ASP. `http://selfhtml.teamone.de/intro/technologien/asp.htm`, besucht im Dez. 2003.

[Ulle01] Ullenboom, Christian (2001): Java ist auch eine Insel. 1. Auflage, Galileo Press GmbH, Bonn.

[W3C04a] World Wide Web Consortium (2004): Extensible Markup Language (XML). `http://www.w3.org/XML/`, besucht im Jan. 2004.

[W3C04b] World Wide Web Consortium (2004): HTTP - Hypertext Transfer Protocol. `http://www.w3.org/Protocols/`, besucht im Jan. 2004.

[W3C04c] World Wide Web Consortium (2004): HyperText Markup Language (HTML). `http://www.w3.org/MarkUp/`, besucht im Jan. 2004.

[W3C04d] World Wide Web Consortium (2004): XML Schema. `http://www.w3.org/XML/Schema/`, besucht im Jan. 2004.

[Zuk00] Zukowski, John (2000): The Definitive Guide to Swing for Java 2, Second Edition. Springer Verlag, New York, Heidelberg.

[Zvon03] Zvon (2003): XPath Tutorial. `http://www.zvon.org/xxl/XPathTutorial/General_ger/examples.html`, besucht im Okt. 2003.

Abbildungsverzeichnis