

# Diplomarbeit 2002

## Software



Autoren: Daniel Hunziker  
Roger Tönz

Betreuer Prof. Dr. H.-P. Hutter  
Mitbetreuer: Prof. S. F. Keller / HSR

Startdatum: 6. September 2002

Abgabetermin: 28. Oktober 2002

## Zusammenfassung

Ziel dieser Diplomarbeit war es, einen effizienten Viewer für INTERLIS-Daten, basierend auf einem Web-Map-Server (WMS), zu entwickeln. Dazu gehörte die Entwicklung eines effizienten Clients zum bestehenden UMN-Map-Server (Open-Source-Map-Server der University of Minnesota). Zudem musste für den UMN-Map-Server eine Import-Möglichkeit für INTERLIS-Daten entwickelt werden.

Die Graphiken werden vom Map-Server via das WMS-Interface angefordert, einem Standard des Open-GIS-Konsortiums. Damit kann die entwickelte Client-Applikation mit unterschiedlichen Map-Servern und anderen Server-Diensten, die das WMS-Interface implementieren, zusammenarbeiten. Um die recht grossen Antwortzeiten (einige Sekunden) der Map-Server zu kompensieren, haben wir ein Vorausladeverfahren implementiert, das zusätzlich zum aktuellen Bild im Hintergrund auch dessen Umgebung anfordert. Dadurch sind beim Verschieben des Bildausschnittes die Daten in der Regel schon auf dem Client und deshalb sofort sichtbar. Die Client-Applikation erlaubt zudem das Überlagern von Daten von verschiedenen Map-Servern. Dabei können sich die Kartenausschnitte überlagern oder ergänzen. Schliesslich besteht die Möglichkeit, einfache Zeichnungsobjekte in die Grafiken massstabsgetreu zu integrieren. Die Grafik kann ausgedruckt und als GIF-Datei gespeichert werden.

Der UMN-MapServer kann mit verschiedenen Datenquellen kommunizieren. Eine Möglichkeit ist die PostgreSQL-Datenbank mit dem PostGIS-Zusatz. Für diese frei erhältliche Datenbank haben wir im zweiten Teil unserer Arbeit ein Konvertierungs-Tool geschrieben, das INTERLIS-Daten in ein SQL-Skript umwandelt. Aufgrund des INTERLIS-Datenmodells kann so ein Script generiert werden, das die benötigten Tabellen erstellt. Die INTERLIS-Datensätze (XML-Format) werden danach mit Hilfe des Konverters in INSERT-Befehle der Datenbank umgewandelt. Hauptschwierigkeiten dabei waren die Umformung des INTERLIS-OO-Schemas in das ER-Schema der Datenbank und das Zuordnen geeigneter DB-Einträge für die verschiedenen INTERLIS-Attribute.

## Abstract

The aim of this diploma thesis was to develop an efficient viewer for INTERLIS data based on a Web Map Server (WMS). This comprised on the one hand the development of an efficient client to the existing open-source Map-Server of UMN (University of Minnesota). A second part was to develop an import-tool for the UMN Map-Server in order to load INTERLIS data.

The client communicates with the Map-Server through the Web-Map-Server (WMS) interface specified by the Open GIS Consortium and supported by most of the Map-Servers. Another advantage of WMS over HTML is the ability to get an image a single request. In order to compensate for the long response time of the Map-Servers (several seconds) we implemented a preloading scheme. This allows for an immediate update of the view after a shift of the map section. In addition we implemented a caching algorithm that allows browsing through the last ten map views. The viewer client is also able to combine data originating from different servers. Finally the user can attach simple graphic objects (line, rectangle, circle and text) to the map, which keep aligned with the map when the view is shifted or scaled. The image can be printed or saved in GIF-format.

The UMN Map-Server has the ability to work with different data sources. One of these is the open-source PostGreSQL DB with the PostGIS extension. With this database we implemented in the second part of our thesis a converter that transforms the INTERLIS data model to an SQL script in order to create the corresponding tables. With the same tool it is then possible to convert the INTERLIS XML-datasets into INSERT statements for the PostGreSQL DB. One of the major difficulties thereby was the mapping of the OO-INTERLIS-scheme to an ER-scheme as well as the transformation of the INTERLIS attributes into adequate database equivalents.

# 1. Inhaltsverzeichnis

Zusammenfassung .....	2
Abstract .....	3
1. Inhaltsverzeichnis .....	4
2. Aufgabenstellung .....	9
3. Aufbau der Arbeit.....	11
3.1. Allgemeines Vorgehen.....	11
3.2. Idee Teil I / Umbau UGE .....	11
3.3. Idee Teil II / UGE MapServer Plugin.....	11
3.3.1. Zeitplan Teil I und II .....	12
3.4. Idee Teil III / INTERLIS für MapServer .....	12
3.4.1. Zeitplan Teil III.....	13
4. Schlusswort .....	13
5. Danksagung.....	14
<b>Teil I / Umbau UGE .....</b>	<b>15</b>
6. Grundlagen .....	16
6.1. UGE .....	16
6.1.1. Was ist UGE .....	16
6.1.2. Ist Zustand.....	16
6.1.3. Soll Zustand .....	16
6.1.4. Umsetzung .....	16
7. Überblick.....	17
7.1. Die UGE Plugin Architecture.....	17
7.1.1. Installation .....	17
7.1.2. Schnittstelle .....	17
7.1.3. PluginEvents auslösen .....	17
7.2. Plugin .....	18
7.2.1. Toolbar .....	18
7.2.2. Dialog-Menus .....	18
7.2.3. DrawDocu sowie DrawOvDocu.....	18
7.2.4. MouseEvents.....	18
7.3. Probleme.....	19
7.4. Code.....	19
<b>Teil II / UGE MapServer Plugin .....</b>	<b>20</b>
8. Grundlagen .....	21
8.1. UMN MapServer .....	21
8.1.1. Was ist ein UMN MapServer .....	21

---

8.1.2.	Wie funktioniert der UMN MapServer .....	22
8.1.3.	Das Grundgerüst .....	22
8.1.4.	Zugriff Möglichkeiten auf MapServer .....	22
8.1.5.	Templates.....	22
8.2.	Open GIS Consortium (OGC) .....	22
8.2.1.	WMS-Standard.....	22
8.2.1.1.	Methoden .....	23
8.2.1.2.	Die wichtigsten Parameter des WMS-Standard : .....	24
9.	Analyse Phase.....	26
9.1.	Ist-Zustand .....	26
9.2.	Soll-Zustand .....	26
9.3.	Features .....	26
9.3.1.	Analyse Verschieben:.....	26
9.3.1.1.	Variante 1:.....	27
9.3.1.2.	Variante 2:.....	27
9.3.1.3.	Variante 3:.....	28
9.3.1.4.	Nutzwertanalyse .....	28
9.3.1.5.	Schlussfolgerung .....	29
9.3.2.	Überlagern von verschiedenen MapServer Verbindungen .....	29
9.3.3.	History .....	29
9.3.4.	Home.....	29
9.3.5.	Darstellung 3D.....	30
9.3.6.	SQL Schnittstelle.....	30
9.3.7.	Layer Anfragen mit Höhenangaben .....	30
9.4.	Anwendungsfälle.....	30
9.4.1.	Verbinden .....	31
9.4.2.	Layer Wahl .....	31
9.4.3.	Folgende Ansicht (History) .....	32
9.4.4.	Vorhergehende Ansicht (History) .....	32
9.4.5.	Ausschnitt verschieben .....	32
9.4.6.	Zoomen .....	32
9.4.7.	Information abrufen .....	33
9.4.8.	SQL Abfrage.....	33
9.4.9.	Höhe angeben.....	33
9.5.	Domänenmodell.....	34
10.	Design / Implementation.....	35
10.1.	Klassendiagramm .....	35
10.2.	Klassenverantwortlichkeit .....	36
10.3.	Sequenzdiagramm .....	40
10.3.1.	AddHistory .....	40
10.3.2.	OpenConnection .....	41

10.3.3.	Verschieben .....	42
10.3.4.	DrawDocu.....	43
10.3.5.	Klassendiagramm.....	44
10.3.6.	Package mit allen Klassen .....	45
11.	Schlussbericht .....	45
11.1.	Funktionsweise .....	45
11.1.1.	XML anfordern/Layer und Graphik initialisieren .....	45
11.1.2.	Suchen von zusätzlichen Informationen .....	47
11.1.3.	Layerwahl .....	47
11.1.4.	Layerparameter .....	47
11.1.5.	Zoomen .....	47
11.2.	Probleme.....	47
11.3.	Code.....	48
11.4.	Testbericht .....	48
<b>Teil III</b> .....		<b>49</b>
12.	Grundlagen.....	50
12.1.	INTERLIS .....	50
12.1.1.	Was ist INTERLIS .....	50
12.1.1.1.	Datenmodell .....	50
12.1.1.2.	IXML Datensatz .....	50
12.1.1.3.	Graphikmodell.....	50
12.1.1.4.	Signaturenbibliothek .....	51
12.1.2.	Attribute von INTERLIS 2 .....	51
12.1.2.1.	Zeichenketten .....	51
12.1.2.2.	Aufzählungen .....	51
12.1.2.3.	Textausrichtungen .....	52
12.1.2.4.	Boolean .....	52
12.1.2.5.	Numerische Datentypen .....	52
12.1.2.6.	Strukturierte Wertebereiche .....	52
12.1.2.7.	Koordinaten.....	52
12.1.2.8.	Linienzüge.....	53
12.1.2.9.	SURFACE .....	53
12.1.2.10.	Komposition .....	53
12.1.2.11.	Referenz .....	53
12.1.2.12.	Assoziationen .....	53
12.2.	MapServer.....	54
12.2.1.	MapServer Daten .....	54
12.2.2.	PostGreSQL / PostGIS.....	54
12.2.3.	INTERLIS vs MapServer .....	54
12.3.	OGC Simple Features for SQL .....	54
12.3.1.	Geometrie Objekte .....	54
12.3.2.	PostGIS .....	55
13.	Analyse Phase .....	56
13.1.	Ist-Zustand .....	56

13.2.	Soll-Zustand .....	57
13.3.	Features .....	57
13.3.1.	OO-Schema zu ER-Schema .....	57
13.3.2.	Abbilden von INTERLIS Daten-Modellen zum DB-Model .....	57
13.3.3.	Abbilden von INTERLIS Attributen .....	59
13.4.	INTERLIS Kreisbögen .....	60
13.4.1.	Definition .....	60
13.4.2.	Berechnungen .....	60
13.4.2.1.	Zentrumsberechnung .....	61
13.4.2.2.	Anzahl Linien, neuer Radius .....	62
13.4.2.3.	Neue Stützpunkte .....	63
13.5.	Anwendungsfälle .....	65
13.5.1.	Laden .....	65
13.5.2.	Kovertieren .....	65
13.5.3.	Exportieren .....	65
13.6.	Zusätzliche Spezifikationen .....	66
13.6.1.	Functionality, Usability, Reliability, Performance, Supportability .....	66
13.6.2.	Implementationsrichtlinien .....	66
13.7.	Domänen Modell .....	66
14.	Design / Implementation .....	66
14.1.	Klassendiagramme .....	66
14.1.1.	Package ch.zhwin.ili2pgschema .....	67
14.1.2.	Package ch.zhwin.ili2pgdata .....	68
14.2.	Klassenverantwortlichkeiten .....	69
14.3.	System Sequenz Diagramm .....	72
14.3.1.	Konstruktor TableHandler .....	72
14.3.2.	SQL Create Generator generate .....	73
15.	Schlussbericht .....	74
15.1.	Funktionsweise .....	74
15.2.	Ausbaumöglichkeiten .....	74
15.3.	Probleme .....	74
15.4.	Code .....	74
15.5.	Testbericht .....	75
<b>Anhang .....</b>		<b>76</b>
A	Glossary .....	77
B	Bedienungsanleitungen .....	79
B.1	UGE .....	79
B.2	UGE Map Plugin .....	84
B.3	Bedienungsanleitung ili2pgsql .....	88

C	MapServer .....	90
C.1	Template File: .....	90
C.2	XML File: .....	92
C.3	Antwort feature_info .....	98
D	Abbilden von OO Schema zu ER Schema .....	100
E	Literatur- / Quellverzeichnis .....	106

## 2. Aufgabenstellung

### Schlussdiplomprüfung Praktische Prüfung



Studiengang: Kommunikation und Informatik

Jahr: 2002

Experte/Expertin: C. Najer  
M. Keller

Studierende: Daniel Hunziker, Roger Tönz

Dozentin/Dozent: Prof. Dr. H.-P. Hutter  
Mitbetreuer: Prof. S. F. Keller / HSR

Ausgabe der Aufgabe: 6. September 2002

Unterschrift:

Abgabetermin: 28. Oktober 2002

#### Thema: **Geodaten-Viewer**

Es besteht immer mehr der Bedarf, die an verschiedenen Orten gesammelten Geo- und Vermessungsdaten der Öffentlichkeit zumindest in Form von Grafiken verfügbar zu machen. Diese Geodaten liegen immer häufiger in einem standardisierten Format (INTERLIS) vor.

Ziel der vorliegenden Diplomarbeit ist deshalb, einen Geodaten-Viewer in Form einer Java-Applikation zu entwickeln, mit dem es möglich ist, Geodaten im INTERLIS-Format grafisch darzustellen und mit anderen grafischen Informationen (z.B. Karten) zu kombinieren. Zudem sollte es möglich sein die so erzeugten Grafiken zumindest rudimentär zu editieren. Zu diesem Zweck soll das Open-Source-Produkt Mapserver [Map02] eingesetzt werden.

#### Anforderungen

- Der Viewer soll mit dem MapServer effizient zusammenarbeiten können.
- Der Viewer soll Geodaten im INTERLIS-Format (vorzugsweise Version 2.2) [Ilis02] darstellen können.
- Der Viewer sollte folgende Funktionalität bieten:
  - Es sollen gleichzeitig mehrere Layer darstellbar sein
  - Zoom-In/Out-Funktionen
  - Effiziente Navigation (Ausschnitt verschieben)
  - Die Geodaten sollen mit mit lokalen Daten (z.B. GIF- oder PDF-Dateien) überlagert werden können
  - Der Viewer sollte einfache Zeichen- und Editierfunktionen zur Verfügung stellen.
- Es sollte möglich sein, den Viewer später zu einem 3D-Viewer auszubauen.

#### Aufgaben

- Studieren Sie die Dokumentation des MapServers [Map02] und von INTERLIS.
- Entwickeln und vergleichen Sie Konzepte für einen Viewer mit den oben genannten Anforderungen. Die Konzept sollten es erlauben, dass der Viewer später auch zu einem 3D-Viewer ausgebaut werden kann.
- Implementieren Sie das Konzept, das Ihrer Ansicht nach am meisten Erfolg versprechend ist.

---

## Quellen

- [Map02] University of Minnesota: *MapServer*. <http://mapserver.gis.mnu.edu>
- [Ilis02] INTERLIS: <http://www.INTERLIS.ch>

## 3. Aufbau der Arbeit

### 3.1. Allgemeines Vorgehen

Beim Durcharbeiten der Aufgabenstellung, stand für uns fest, dass wir uns mit zwei Teilbereichen zu beschäftigen haben. Neben dem MapServer-Viewer musste eine Lösung für INTERLIS gefunden werden. In dem wir die INTERLIS-Daten für den MapServer aufbereiten, können sie dann über den Viewer angeschaut werden.

Daraus entstanden die Hauptbereiche MapServer-Viewer und INTERLIS-Konverter.

Mit dem Universellen Grafik Editor (UGE, Semesterarbeit Sommer 2002) hatten wir bereits ein Programm entwickelt, mit dem einige der Viewer-Anforderungen erfüllt werden.

Wir unterteilten die Arbeit am Viewer deshalb in zwei Unterteile. Einerseits in den Umbau der UGE Applikation und andererseits in die Entwicklung eines MapServer-Viewer-Plugins.

Entsprechend entstanden die folgenden drei Teile:

- Umbau UGE
- UGE MapServer Plugin
- INTERLIS für MapServer

### 3.2. Idee Teil I / Umbau UGE

Der universelle Graphikeditor (UGE) ist ein Programm, welches wir (Daniel Hunziker und Roger Tönz) als Projektarbeit im Sommersemester programmiert haben. Damit können PDF- und GIF Files geöffnet und mit einfachen Zeichnungsobjekten ergänzt werden. Die Änderungen können mit Hilfe eines integrierten FTP Client verschickt werden. Die vorliegende Diplomarbeit hat den universellen Graphikeditor als Basis genutzt, um GIS Daten ebenfalls von MapServer anzuschauen und zu ergänzen. Als erstes haben wir den UGE Plugin fähig gemacht. Das heisst, dass der MapServer-Viewer ein geschlossenes Paket ist. Wir haben eine Schnittstelle im UGE Programm geschrieben, welche Plugins importieren kann. Es können dadurch in Zukunft problemlos neue Plugins hinzugefügt werden.

### 3.3. Idee Teil II / UGE MapServer Plugin

Immer öfter werden Geo- und Vermessungsdaten in Form von Graphiken zur Verfügung gestellt. Oft werden diese Daten mit Hilfe eines MapServers via Internet in einem Browser dargestellt.

Das Ziel dieser vorliegenden Diplomarbeit ist es, eine Applikation zu schreiben, welche Geo- und Vermessungsdaten darstellen soll. Unsere Applikation wird gegenüber der Variante mit einem Browser folgende Vorteile haben:

- Graphiken sollen beim Verschieben schneller gezeichnet werden können.
- Unterschiedliche Daten von verschiedenen MapServer können überlagert werden.
- Die Graphiken sollen mit Hilfe von einfachen Zeichnungsobjekten ergänzt werden.

### 3.3.1. Zeitplan Teil I und II

Iterationen	W. 1	W. 2	W. 3	W. 4	W. 5	W. 6	W. 7
Suchen von Informationen							
Anwendungsfälle							
Analyse/Design/Code UGE Umwandeln							
Analyse Optimierung							
Analyse/Design/Code 3D darstellen							
Analyse/Design/Code Verschieben							
Analyse/Design/Code Überlagern von MapServer							
Analyse/Design/Code Info anzeigen							
Analyse/Design/Code History							
Analyse/Design/Code Layer auswählen							
Testen							
Documentation Schreiben							

### 3.4. Idee Teil III / INTERLIS für MapServer

INTERLIS ist eine Beschreibungssprache für GeoDaten. Der MapServer ist ein Server, der die Daten graphisch aufbereitet. Danach können diese mit dem Viewer, der in Teil II erarbeitet wird, betrachtet

werden. In dem wir dem MapServer die INTERLIS Daten verfügbar machen, haben wir indirekt einen Viewer für INTERLIS Daten. Um einem MapServer die INTERLIS-Daten verfügbar zu machen, sind mehrere Schritte notwendig.

- Abbilden des INTERLIS-Datenmodells in ein ER-Modell
- Schreiben von sql-Create Tables
- Konvertieren von IXML Datensätzen in sql-Inserts
- Abbilden des INTERLIS-Graphicmodell in sql-Views
- Abbilden der INTERLIS-Symbol-Bibliothek (Map oder DB)
- Generierung eines map-Files

Wegen der knappen Zeit werden vorerst nur die ersten drei Punkte umgesetzt.

Um einem MapServer die GIS-Daten verfügbar zu machen, müssen die Daten von den INTERLIS-Dateien in die PostgreSQL Datenbank geladen werden. Ein Teil der INTERLIS-Dateien beschreibt das Datenmodell. Ein weiterer Teil sind die Datensätze im XML-Format.

### 3.4.1. Zeitplan Teil III

Iterationen	W. 1	W. 2	W. 3	W. 4	W. 5	W. 6	W. 7
Analyse Schema							
Design							
Implementation							
Analyse Data							
Design							
Implementation							
Testen							
Dokumentation							

## 4. Schlusswort

Beim Umbau der UGE-Applikation konnten wir verschiedene neue Probleme lösen.

Die Map-Viewer Applikation wurde erfolgreich umgesetzt und getestet. Durch Caching und Preloading konnte die Langsamkeit der MapServer verdeckt werden.

Sich in einigen Wochen in eine neue Beschreibungssprache einzuarbeiten ist nicht ganz einfach. Erschwerend kam noch dazu, dass etwa in der Hälfte der Diplomarbeit das Referenzhandbuch änderte. Dank des INTERLIS-Compilers musste jedoch nicht noch ein Reader für das Format selber geschrieben werden. Leider sind die Tests des ili2pgsql Tool mangels korrekten Test-Daten mager ausgefallen.

## 5. Danksagung

Besten Dank an Prof. S. F. Keller von der HSR der uns tatkräftig bei INTERLIS-Problemen unterstütze. Ebenfalls bedanken möchten wir uns bei Herr C. Eisenhut, dessen INTERLIS-Compiler als Grundlage unseres ili2pgsql-Tools dient. Bei Fragen konnten wir ebenfalls mit seiner Unterstützung rechnen. Bei Software-Entwicklungs-Fragen konnten wir dankend auf Prof. H.P. Hutter zählen.

Winterthur, 28. Oktober 2002

Daniel Hunziker

Roger Tönz

.....

.....



# Teil I

Umbau UGE

## 6. Grundlagen

### 6.1. UGE

#### 6.1.1. Was ist UGE

Der Universelle Grafik-Editor (UGE) ist eine Projektarbeit von Roger Tönz und Daniel Hunziker. Das Programm erlaubt PDF Dateien sowie einige im WWW gebräuchliche Bild-Formate anzuschauen. Der grösste Vorteil dieses Programms besteht darin, dass es bei den Dateien einfache Zeichnungselemente hinzufügen kann. Diese werden beim Vergrössern und Verkleinern mitskaliert.

#### 6.1.2. Ist Zustand

Die Dateien werden mittels der DocumentFactory und dem Interface Document an die entsprechende Klasse weitergeleitet. Als Schnittstelle zum Programm war vor allem das Interface Document verantwortlich. Jedes der gewünschten Formate musste durch eine Klasse mit diesem Interface gelöst werden. Es konnten somit nur Dateien geöffnet werden, die lokal vorhanden waren.

#### 6.1.3. Soll Zustand

Damit nicht bei jedem neuen Format oder anderen Bildquelle das Programm umgeschrieben werden muss, sollen dafür externe Plugins ermöglicht werden. Jedes Plugin ist dann für die Arbeit mit der Bildquelle selber verantwortlich. Vom Hauptprogramm sollen nur noch Zeichnungsbefehle kommen.

#### 6.1.4. Umsetzung

Um die Zusammenarbeit zu regeln, wurden zwei Plugin-Arten konzeptiert. Beide werden durch abstrakte Klassen spezifiziert. Damit das Plugin auch geladen werden kann, müssen einige Anforderungen erfüllt werden:

- Das Pluginpaket als zip oder jar in lib Verzeichnis
- Die Pluginklasse muss in Package ch.zhwin.ugeViewer.plugins sein.
- Der Klassenname wird zusammengesetzt aus Pluginpaket-Name in Grossbuchstaben und PluginTyp (zB PluginpaketName: pdf.jar, PluginTyp: Docu => PDFDocu)
- Die Klasse muss eine Erweiterung des entsprechenden PluginTyp Klasse sein

Geladen wird das Plugin wie folgt: Zuerst werden alle zip oder jar Dateien, im „lib“ Verzeichnis, einem URLClassLoader hinzugefügt. Aufgrund der Dateinamen wird dann versucht ein Plugin zu initialisieren. In einem Paket können somit auch beide Arten von Plugin vorkommen.

## 7. Überblick

### 7.1. Die UGE Plugin Architecture

Für die UGE-Applikation wurden zwei Arten von Plugins konzipiert. Ein Typ von Plugin ist für das Darstellen von Dokumenten zuständig. Der zweite Typ stellt Zusatzfunktionen zur Verfügung.

#### 7.1.1. Installation

Um ein Plugin zu installieren, muss es sich innerhalb des „lib“ Verzeichnisses befinden. Das Programm versucht dann anhand des Dateinamens (z.B. document.jar) ein Plugin-Object, `ch.zhwin.ugeViewer.plugins.*Docu` (z.B. `DOCUMENTDocu`) oder `ch.zhwin.ugeViewer.plugins.*Tool` zu initialisieren. Das Plugin-Objekt muss sich somit in einem speziellen Paket befinden. Ebenfalls soll es mit Grossbuchstaben, der Jar-Datei entsprechend, beginnen und mit dem Typ Docu oder Tool enden.

#### 7.1.2. Schnittstelle

Als Schnittstelle für die beiden Plugin-Arten dienen die abstrakten Klassen: `DocuPlugin` und `ToolsPlugin`.

#### 7.1.3. PluginEvents auslösen

Das `MainFrame`, das als Eigentümer des Plugins auftritt, kann auch als Eigentümer anderer Dialoge genutzt werden. Ebenso kann das `MainFrame` als `PropertyChangeListener` gebraucht werden. Speziell ist jedoch, dass nur 4 Fälle überprüft werden.

Name	oldValue	newValue
------	----------	----------

"clip"	null	„true“ or „false“
"dol"	null	null
"drawObj"	null	drawObject
"plugin"	null	docuPlugin

Beim Clip-Event wird das Drawing-Canvas neu gezeichnet. Wird bei newValue ebenfalls noch der String „true“ mitgegeben, wird der Overview-Dialog neu aufgebaut.

Der DOL sowie der DrawObj – Event lösen ebenfalls einen Neuaufbau der Zeichnungsfläche auf. Das Dokument muss dabei jedoch nicht neu gezeichnet werden.

Um der UGE-Applikation mitzuteilen, dass ein Dokument geöffnet wurde, kann der Plugin-Event genutzt werden. Dieser setzt den Index auf das mitgegebene Plugin.

## 7.2. Plugin

### 7.2.1. Toolbar

Jedes DocuPlugin hat die Möglichkeit, eine Toolbar der UGE-Applikation zu übergeben. Die UGE-Applikation verwaltet diese und „enabled“ diese sobald das Plugin aktiv wird. Ansonsten wird die Toolbar „disabled“ sein.

### 7.2.2. Dialog-Menus

Um Plugin-eigene Dialoge zu verwalten, kann ein WindowManager gebraucht werden. Das Ein- und Ausschalten sowie das „enabling“ und „disabling“ der Menüitems sowie der Dialoge muss jedoch vom Plugin selber übernommen werden. Der WindowManager hilft bei diesen Aufgaben.

### 7.2.3. DrawDocu sowie DrawOvDocu

Beide bekommen Ausschnitt-Informationen, die zum Zeichnen des Dokumentes nötig sind.

### 7.2.4. MouseEvents

Um MouseEvent zu erhalten, kann vom MainFrame das DrawingCanvas angefordert werden. Ebenso kann das OverviewFrame mit einem eigenen MouseListener versehen werden. Sollte also wirklich ein MouseEvent genutzt und nicht bei der UGE-Applikation ausgeführt werden, muss beim MainFrame die Methode setPluginTip() aufgerufen werden. Sobald der Benutzer aber wieder auf einen anderen Button drückt, werden die Ereignisse normal ausgeführt.

Beim OverviewFrame gilt es zu beachten, dass die Zeichnung verschoben gezeichnet wird. Diese Verschiebungswerte erhält man mit der Methode getDXDY().

### **7.3. Probleme**

Ursprünglich war geplant, das Bild durch kopieren in die System Zwischenablage zu exportieren. Aus Zeitgründen musste die Implementation dieses Features leider verschoben werden.

### **7.4. Code**

Der Programm Code und ein vollständiges Javadoc befinden sich auf der beigelegten CD.



# Teil II

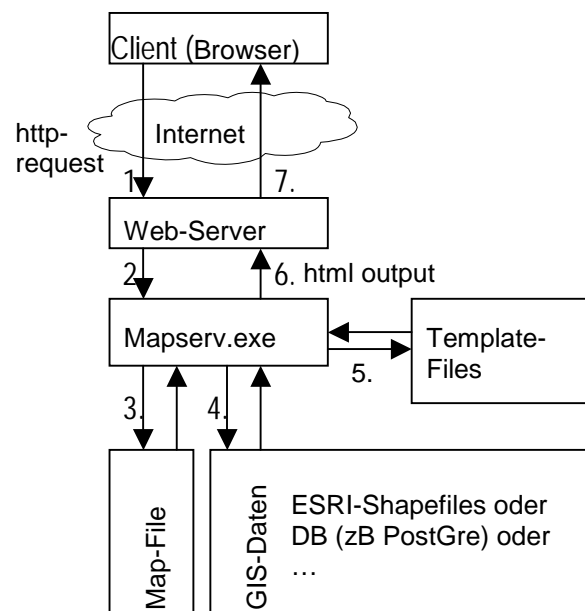
UGE MapServer  
Plugin

## 8. Grundlagen

### 8.1. UMN MapServer

#### 8.1.1. Was ist ein UMN MapServer

Der MapServer ist ein Werkzeug im Bereich der Geographischen Informationssysteme (GIS), mit dem sich georeferenzierte Daten durch das Internet darstellen lassen.



Die zentrale Funktion des MapServers besteht darin, aus Geo-Daten eine Karte zu erzeugen und einen bestimmten Bereich als Bild (GIF-Grafik) zur Verfügung zu stellen. Dabei lassen sich sowohl Vektor- als auch Rasterdaten nutzen. Der MapServer ist eine freie Software. Die grundlegenden Möglichkeiten des MapServers umfassen Navigation (Vergrößern/Verkleinern, Verschieben) und Abfragen anhand von Punkten, Regionen, Feldern oder Werten. Die Darstellung kann massstabsabhängig verschieden erfolgen. Zusätzlich zur Karte können Elemente wie Legenden, Masstabsskalen oder Referenzkarten generiert werden.

### 8.1.2. Wie funktioniert der UMN MapServer

Der Kern des MapServers ist ein CGI-basiertes Programm, welches einfache GIS-Methoden beherrscht. Die Entwicklung einer MapServer-Anwendung spielt sich in verschiedenen Dateien zur Definition von Eigenschaften, Darstellungen und Vorlagen für die Benutzerschnittstelle ab.

### 8.1.3. Das Grundgerüst

Neben den Daten besteht das Grundgerüst einer MapServer-Anwendung aus drei Dateien:

- eine Initialisierung, die als Einstiegsseite nach Belieben ausgeschmückt werden kann,
- einer Vorlage, mit der die Benutzerschnittstelle beschrieben wird
- und einer MAP-Datei, welche die Darstellung der Karte bestimmt.

### 8.1.4. Zugriff Möglichkeiten auf MapServer

Der Zugriff auf einen MapServer geschieht durch GET Methoden des HTTP Protokolls. Hinter der statischen URL folgt eine Parameterliste.

Es gibt mehrere Möglichkeiten, wie der MapServer antworten kann: Er kann eine HTML Datei (von Template) oder ein XML Datei (WMS-Standard) zurückschicken.

Für unsere Applikation haben wir den WMS-Standard gewählt, damit die Applikation mit möglichst vielen MapServer, kompatibel ist.

### 8.1.5. Templates

Bei den Templates handelt es sich um HTML-Dateien, die direkt von einem Browser dargestellt werden können. Damit ein Template als Ergebnis ausgeliefert wird, muss der Modus browse im URL angegeben sein, im Gegensatz zum Modus map, der lediglich die Karten als Bilder zurückliefert.

Die Antworten des MapServers sind vom Template Files abhängig und nicht standardisiert.

Nachdem der Client den Template File erhalten hat, in welcher die Adresse der Graphik und der Name des Layers stehen, muss er nochmals eine Verbindung erstellen um die Graphiken zu holen. Ein Beispiel befindet sich im Anhang.

## 8.2. Open GIS Consortium (OGC)

### 8.2.1. WMS-Standard

OGC ist ein internationaler Zusammenschluss von über 220 Unternehmen. Ziel des OGC ist es, gemeinsame Standards (Protokolle, Formate etc.) zur Verarbeitung von Geo-Daten zu erarbeiten. WMS ist die OpenGis Web Map Server Interfaces Implementation Spezifikation, kurz WMS-Standard genannt.

Die Anfragen sind im CGI Standard Style geschrieben. Das heisst, dass die Parameter aus Name-Wert Paaren zusammengestellt sind (name=value). Die Paare sind mit einem „&“ Zeichen getrennt. Parameter, welche aus einer Liste bestehen wie Layer oder Styles, sollen mit einem Komma getrennt werden. Die Parameter sollten nicht case sensitiv sein. Der Parameter Wert hingegen ist case sensitive. Die Parameter Reihenfolge spielt keine Rolle.

### 8.2.1.1. Methoden

Der WMS Standard definiert folgende drei Methoden:

- getCapabilities: gibt ein XML Dokument zurück, über die Möglichkeiten des MapServers
- getMap: gibt den Bildausschnitt zurück, welchen der Benutzer verlangt hat.
- feature\_info: gibt zusätzliche Informationen für eine bestimmte Koordinate.

URL einer Anfrage getCapabilities:

```
http://dskt6490.zhwin.ch/cgi-bin/mapserv.exe?Version=1.0&map=h:/www-root/itasca/demo.map&Request=getCapabilities
```

URL einer Anfrage getMap:

```
http://dskt6490.zhwin.ch/cgi-bin/mapserv.exe?Version=1.0&map=h:/www-root/itasca/demo.map&Request=getMap&SRS=EPSG:26915&BBOX=189775.0,4978428.538147578,760608.3106504554,5472410.0&WIDTH=1233.0&HEIGHT=1067.0&LAYERS=dlgstln2&STYLES=default&FORMAT=gif&TRANSPARENT=TRUE&EXCEPTION=INIMAGE&QUALITY=MEDIUM
```

URL einer Anfrage feature\_info:

```
http://dskt6490.zhwin.ch/cgi-bin/mapserv.exe?Version=1.0&map=h:/www-root/itasca/demo.map&Request=feature_info&INFO_FORMAT=Text/plain&X=300&Y=96&QUERY_LAYERS=dlgstln2,&SRS=EPSG:26915&BBOX=380961.43758763396,5194443.902798121,649436.0095192051,5321223.561765807&WIDTH=792.0&HEIGHT=374.0&EXCEPTIONS=text/plain
```

Das Ergebnis dieser Anfragen kann im Anhang angeschaut werden.

### 8.2.1.2. Die wichtigsten Parameter des WMS-Standard :

Parameter des WMS-Standard	Erklärung:
http://server_address/path/script?	URL Adresse des CGI Skript des MapServer.
WMTVER=1.0	Protokoll Version Nummer des MapServers.
REQUEST=getMap REQUEST=getCapabilities REQUEST=feature_info	Es gibt drei verschiedene Arbeitsweisen. "getMap", "getCapabilities" oder "feature_info".
SRS=EPSG:4326	Spatial Reference System. Bezeichnet ein horizontales Koordinatensystem. Der Server liefert eine Fehlermeldung, im Falle einer Anfrage mit einer SRS, die nicht unterstützt wird. <b>EPSG:</b> European Petroleum Survey Group. Beispiel EPSG:4326 ist eine Angabe in Grad mit dem Greenwich als Zentralen Meridian.
BBOX=-97.105,24.913,78.794,36.358	Bounding Box corners (xmin,ymin, xmax, ymax) Eingabe der Grösse des Bildes in Grad in Bezug auf dem SRS (Spatial Reference System). Die Angabe kann ein double Wert oder ein integer Wert sein. Damit wird ein Kartenausschnitt gewählt.
WIDTH=560&HEIGHT=350	Gibt die Grösse des Bildes (in Pixel) an, welche der MapServer zurückschicken muss. Falls die Grösse des Bildes WIDTH und HEIGHT nicht mit der Grösse der BBOX übereinstimmt, muss der MapServer die Graphik strecken oder verkleinern.
LAYERS=	Layer die der Anwender anschauen will.

QUERY_LAYERS=	Layer um zusätzliche Informationen anzuzeigen. Wird von der Method feature_info benutzt.
STYLES=default	Für jeden Layer kann ein Rendering Style definiert werden, wenn dieser im getCapabilities File deklariert ist. Default ist der Standard Wert.
FORMAT=GIF	Format der darzustellenden Graphik (GIF, JPEG, PNG).
BGCOLOR=0xFFFFFFFF	Hintergrundfarbe in hexadezimaler Darstellung.
TRANSPARENT=TRUE	Sofern der Wert TRUE ist wird die Farbe ausserhalb der Graphik transparent gesetzt.
EXCEPTIONS=INIMAGE	Format, welches der MapServer bei einem Fehler zurückgibt.
„X=“ , „Y=“	Position der Maus, damit eine Anfrage in Bezug auf die Mausposition geschickt werden kann. Z.B. getFeature_info braucht diese Angabe.

## 9. Analyse Phase

### 9.1. Ist-Zustand

Die vielen MapServer Browser Varianten können Daten von verschiedenen MapServern nicht überlagern. Sie haben den Nachteil, dass bei jedem Verschieben das neue Bild im Vordergrund geladen wird. Zudem muss die Graphik in einem Graphikprogramm kopiert und mit Zeichnungsobjekten ergänzt werden.

### 9.2. Soll-Zustand

Unsere Applikation soll gegenüber den vielen GIS-Browserviewer Varianten folgende vier entscheidende Vorteile haben:

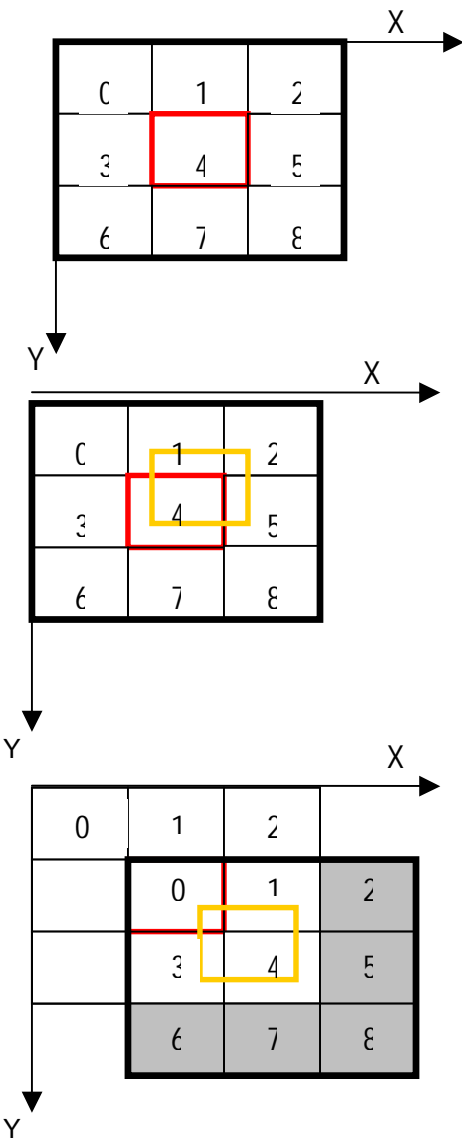
- a) Die Applikation zeigt Graphiken schneller an. Verschiebt der Anwender die Graphik, um einen anderen Kartenausschnitt anzuschauen, sind die neuen Ausschnitte in der Regel schon vorgeladen.
- b) Weiter besteht die Möglichkeit Graphiken von verschiedenen MapServer zu überlagern. Es kann z.B. die Landkarte von Kloten und die Flugstrassen des Flughafens zusammengeführt werden. Daraus kann beispielweise abgelesen werden, welche Häuser am meisten vom Lärm betroffen sind.
- c) Der dritte Vorteil ist das Verwalten einer History. Bei Betätigen der History werden die Bildausschnitte aus dem Speicher geholt.
- d) Schlussendlich besteht auch die Möglichkeit, Graphiken zu ergänzen, was bis anhin nur durch kopieren der Graphik in ein Graphikeditorprogramm möglich war.

### 9.3. Features

#### 9.3.1. Analyse Verschieben:

Nachfolgend werden drei verschiedene Möglichkeiten, wie das Verschieben eines Graphikausschnittes optimiert werden kann, vorgestellt.

**9.3.1.1. Variante 1:**

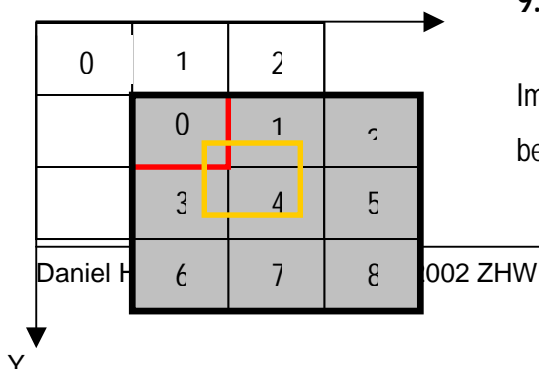


Zu Beginn wird die gewünschte Graphik „4“ als erstes Bild dargestellt. Im Hintergrund werden ebenfalls die benachbarten Bilder der Graphik „4“ geladen, welche zu einem grossen Bild zusammengestellt werden.

**Verschieben 1:** Möchte der Anwender eine benachbarte Graphik von „4“ ansehen, so kann er diese verschieben. Falls der Mittelpunkt der neu zu darstellenden Graphik (helles Rechteck) in der Graphik „4“ bleibt, werden keine neuen Bilder geladen. Damit können Anfragen an den MapServer reduziert werden.

**Verschieben 2:** Falls der Mittelpunkt des neuen Ausschnittes nicht mehr im Rechteck „4“ liegt, werden die fehlenden benachbarten Graphiken geladen. Es wird ein neues Bild im Hintergrund aufgebaut. Durch umnummerieren der Teilbilder gelangen wir wieder in den Grundzustand.

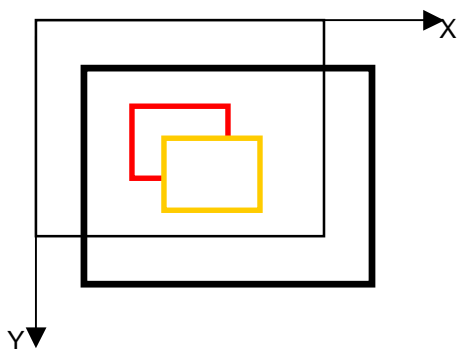
**9.3.1.2. Variante 2:**



Im Gegensatz zur ersten Variante, werden bei dieser Variante bei jedem Verschieben alle Bildausschnitte neu geladen. Da

eine Umnummerierung nicht mehr notwendig und somit das grosse Bild immer vollständig vorhanden ist, wird der Programmieraufwand geringer.

### 9.3.1.3. Variante 3:



Bei der dritten Variante besteht die grosse Graphik nicht mehr aus einzelnen Bildausschnitten, sondern nur noch aus einer Graphik, welche bei jeder Verschiebung neu geladen wird. Gegenüber der 1. und 2. Variante liegt der Hauptvorteil darin, dass weniger Anfragen an den MapServer gemacht werden müssen.

Bei der 1. und 2. Variante müssen mindestens 3 und höchstens 9 Anfragen für das grosse Bild gemacht werden. Bei dieser 3. Variante muss hingegen nur eine Anfrage gemacht werden. Das grosse Bild ist 3-mal so breit und hoch wie das sichtbare kleine Bild. Bei einer maximalen Verschiebung des sichtbaren Bildausschnittes wird der neue Bildausschnitt aus dem im Voraus geladenen Bild hergestellt. Man bemerkt deshalb keine Ladezeit. Falls der Benutzer einen Bildausschnitt wählt, der nicht im grossen Bild liegt, wird zuerst das kleine Bild geladen und dargestellt. Erst danach wird im Hintergrund das grosse Bild geladen. Falls der Benutzer einen Bildausschnitt wählt bevor das grosse Bild fertig geladen ist, wird ein kleines Bild angefordert und das grosse Bild verworfen.

### 9.3.1.4. Nutzwertanalyse

Kriterium	Gewicht	Variante 1		Variante 2		Variante 3	
		Wertung	Nutzen	Wertung	Nutzen	Wertung	Nutzen
Programmieraufwand	10	1	10	2	20	4	40
Anzahl Verbindungen	60	2	120	1	60	4	240

kB Übertragung	30	4	120	2	60	1	30
Total:	100		230		180		310

Bewertung 1 = schlecht , 5 = sehr gut

### 9.3.1.5. Schlussfolgerung

Wir haben unsere Applikation mit der 3. Variante implementiert, weil dadurch die Anzahl der Anfragen an den MapServer reduziert werden kann. Bei einer schnellen Internet Verbindung ist das Übertragen von zusätzlichen 20kB (Durchschnitt eines grossen Bildes) kein Hindernis. Hingegen ist die Zeit, welche der MapServer braucht um ein Bild zu berechnen (circa 1 Sekunde) ein entscheidender Faktor. Wenn wir 5 kleine Bilder laden würden, müssten wir mindestens 5 Sekunden warten.

### 9.3.2. Überlagern von verschiedenen MapServer Verbindungen

Damit verschiedene Verbindungen überlagert werden können, müssen ihre SRS (Spatial Reference System) gleich sein. Falls sie nicht gleich sind, wird eine Fehlermeldung ausgegeben und die Verbindung mit dem zusätzlichen Server wird nicht hergestellt.

Andernfalls wird wiederum die umrahmende BBOX berechnet und damit die Grösse des Dokumentes aus mehreren MapServern initialisiert. Alle weiteren Bildausschnitte beziehen sich auf diese Dokumentgrösse. Bei diesem Verfahren spielt es keine Rolle, ob die Kartenausschnitte der Server neben oder übereinander zu liegen kommen.

Jede Verbindung kennt ebenfalls ihre Dokumentengrösse. Nur wenn der zu zeichnende Bildausschnitt innerhalb des Serverbereichs liegt, wird eine Anfrage an diesen MapServer gesandt.

### 9.3.3. History

Wir werden eine History programmieren, welche zehn Bilder zwischenspeichert. Dadurch werden die Bilder bei einer Betätigung des „back“ oder „forward“ Buttons aus dem Speicher geholt. Es muss also kein Zugriff auf den MapServer erfolgen.

### 9.3.4. Home

Die Applikation wird einen „Home“ Button anbieten. Bei Betätigung dieses Buttons wird die erste Graphik aus dem Speicher geholt und die History wird gelöscht.

### 9.3.5. Darstellung 3D

Leider können zurzeit MapServer noch keine 3D Bilder darstellen. Sollte es später möglich sein, wird unsere Applikation auch diese Bilder darstellen können. Der Grund liegt darin, dass es sich um 2D Bilder handelt die speziell gerendert wurden. Unsere Applikation rendert die Graphiken nicht selber, dies ist die Aufgabe des MapServers. Das Ergänzen von 3D Bildern ist ebenfalls kein Problem, da die Graphiken wie bis anhin in 2D dargestellt sind.

### 9.3.6. SQL Schnittstelle

Wir werden eine SQL Schnittstelle schreiben, welche ermöglicht, Anfragen an den MapServer zu verschicken. Diese Anfragen können eine dritte Dimension enthalten.

z.B. `Select all lakes From lakes wich are about 2000.`

Diese Anfrage ist nur möglich, wenn der MapServer mit einer Datenbank, wie PostgreSQL oder PostGIS, zusammenarbeitet.

### 9.3.7. Layer Anfragen mit Höhenangaben

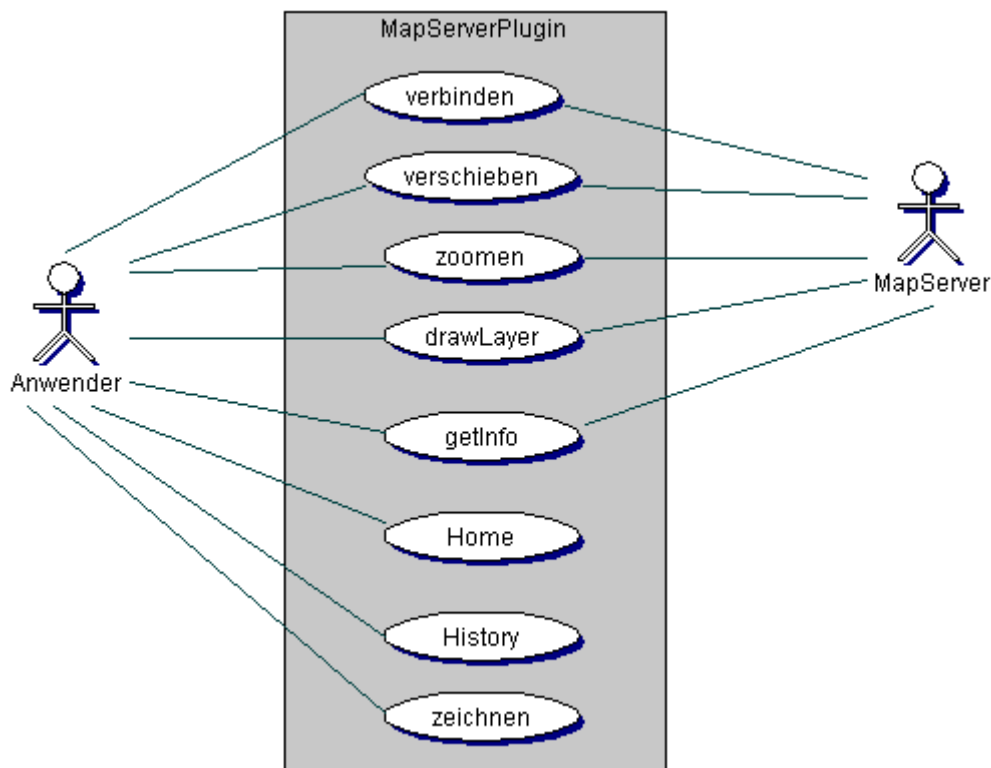
Falls die Layer die Möglichkeit anbieten, ihre Information anhand einer Höhenangabe zu filtern, werden wir diese im Layer-Eigenschaftenfenster angeben. Die Höhenangabe wird mit dem Parameter Elevation in der URL kodiert.

## 9.4. Anwendungsfälle

Unsere MapServer Applikation soll die folgenden Anwendungsfälle beinhalten:

- Verbinden mit MapServer
- Es können Layer aus einer Liste selektiert werden
- Es können die folgenden und vorhergehenden Graphiken geladen werden (History)
- Ein Ausschnitt kann mit der Maus verschoben werden
- Jeder Ausschnitt kann vergrößert oder verkleinert werden
- Ein Infofenster kann aufgerufen werden

- Die Höhe der Layer kann eingegeben werden
- Es können SQL Suchbegriffe eingegeben werden



### 9.4.1. Verbinden

#### Hauptscenario:

Der Anwender aktiviert das Fenster Connectiondialog.

Der Anwender gibt die URL des MapFile an.

Er klickt auf den Button „Laden“.

#### Nebenszenario:

Falls die Verbindung nicht aufgebaut werden kann, erscheint eine Fehlermeldung.

Der Anwender muss die richtige Adresse angeben.

### 9.4.2. Layer Wahl

#### Hauptscenario:

Der Anwender selektiert den gewünschten Layer.

Der Anwender klickt auf den Button „refresh“.

Der gewählte Layer wird auf das aktuelle Bild gezeichnet.

### 9.4.3. Folgende Ansicht (History)

**Hauptscenario:**

Der Anwender klickt auf den Button „folgende Ansicht“.

Die nächste Ansicht wird gezeichnet.

**Nebenszenario:**

Der Button „folgende Ansicht“ ist deaktiviert, da es keine folgende Ansicht gibt.

### 9.4.4. Vorhergehende Ansicht (History)

**Hauptscenario:**

Der Anwender klickt auf den Button „vorhergehende Ansicht“.

Die vorhergehende Ansicht wird gezeichnet.

**Nebenszenario:**

Der Button „vorhergehende Ansicht“ ist deaktiviert, es gibt keine vorhergehende Ansicht.

### 9.4.5. Ausschnitt verschieben

**Hauptscenario Variante 1:**

Der Anwender klickt auf das Symbol „Hand“.

Das Bild kann, indem auf die linke Maustaste gedrückt wird, verschoben werden.

**Hauptscenario Variante 2:**

Der Anwender klickt auf einer der vier Symbole  in der oberen Menüleiste.

Das Bild bewegt sich falls, es möglich ist, in die gewählte Richtung.

**Nebenszenario:**

Das Bild bewegt sich nicht, da die Grenze der Graphik erreicht ist.

### 9.4.6. Zoomen

**Hauptszenario:**

Der Anwender klickt auf den Button „plus“ oder „minus“.

Der Anwender wählt mit der Maus einen Ausschnitt aus, der vergrößert oder verkleinert werden soll.

Der neue Planausschnitt wird gezeichnet.

**Nebenszenario:**

Der Plan kann nicht mehr gezeichnet werden, da der maximale oder minimale Zoomfaktor erreicht wurde.

### 9.4.7. Information abrufen

**Hauptszenario:**

Der Anwender wählt den Layer aus, zu welchem Informationen zur Verfügung stehen.

Der Anwender klickt auf den Button „Info“.

Der Anwender klickt auf die Graphik.

Ein Info-Fenster öffnet sich mit den gewünschten Informationen.

### 9.4.8. SQL Abfrage

**Hauptszenario:**

Der Anwender gibt die SQL Anfrage im dafür reservierten Fenster ein.

Er klickt auf den Button „Senden“.

Das Ergebnis wird im gleichen Fenster angezeigt.

### 9.4.9. Höhe angeben

**Hauptszenario:**

Der Anwender gibt die Höhe des Layers im dafür reservierten Fenster ein.

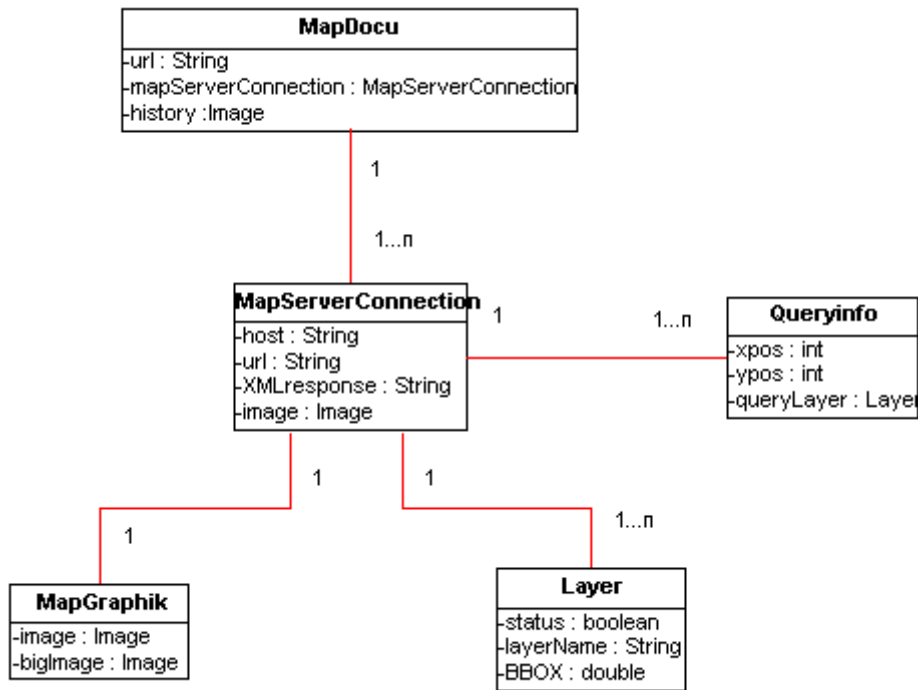
Er klickt auf den Button „setzen“.

Der Layer mit der gesetzten Höhe wird beim Nächsten „refresh“ angezeigt.

**Nebenszenario:**

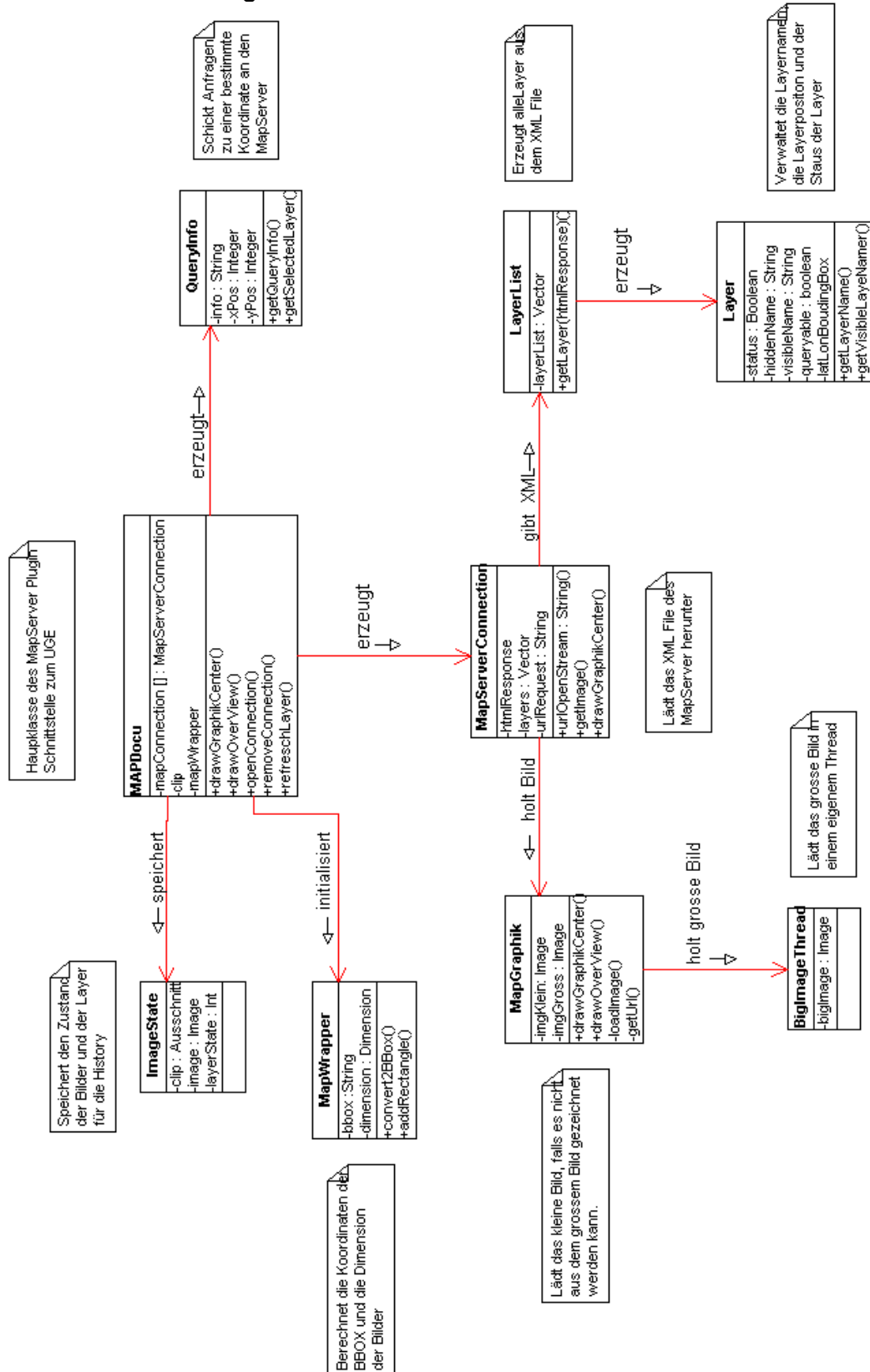
Falls der Layer keine Höhenangaben unterstützt, können keine Höhenangaben gesetzt werden.

### 9.5. Domänenmodell



# 10. Design / Implementation

## 10.1. Klassendiagramm



## 10.2. Klassenverantwortlichkeit

Klasse MAPDocu	
<p><b>Verantwortlichkeit</b></p> <ul style="list-style-type: none"> <li>- Schnittstelle zum UGE</li> <li>- Schnittstelle zu allen Dialogen (Connection, Layer und Info)</li> <li>- gibt die Zeichnungsbefehle weiter</li> <li>- berechnet die Dimension der Bilder</li> <li>- verwaltet die History</li> <li>- verwaltet ein Vektor mit den Verbindungen</li> </ul>	<p><b>Information</b></p> <ul style="list-style-type: none"> <li>- kennt die Anzahl der Verbindungen</li> <li>- kennt den aktuellen Clip</li> <li>- speichert die Bilder der History</li> </ul> <p><b>Creator</b></p> <ul style="list-style-type: none"> <li>- erzeugt Objekt des Typs MapServerConnection</li> <li>- erzeugt ein Objekt des Typs MapWrapper</li> <li>- erzeugt Objekte des Typs ImageState</li> <li>- erzeugt ein Objekt des Typs LayerDialog</li> <li>- erzeugt ein Objekt des Typs ConnectionDialog</li> <li>- erzeugt Objekte des Typs BigImageThreas</li> </ul>

Klasse MapServerConnection	
<p><b>Verantwortlichkeit</b></p> <ul style="list-style-type: none"> <li>- macht eine Verbindung mit dem MapServer um alle Informationen, die der MapServer kennt, zu bekommen. Die Information wird vom MapServer als XML File geschickt</li> </ul>	<p><b>Information</b></p> <ul style="list-style-type: none"> <li>- kennt die Hostadresse des MapServers</li> <li>- kennt die URL des MapServers</li> </ul> <p><b>Creator</b></p> <ul style="list-style-type: none"> <li>- erzeugt Objekte des Typs LayerList</li> <li>- erzeugt Objekte des Typs MapGraphik</li> </ul>

Klasse MapGraphik	
<p><b>Verantwortlichkeit</b></p> <ul style="list-style-type: none"> <li>- lädt alle Bilder</li> </ul>	<p><b>Information</b></p> <ul style="list-style-type: none"> <li>- kennt die Clipinformation des sichtbaren Bildausschnittes</li> <li>- weiss, welche Layer aktiviert sind</li> <li>- weiss, welcher Bildausschnitt es laden muss</li> </ul> <p><b>Creator</b></p> <ul style="list-style-type: none"> <li>-</li> </ul>
Klasse BigImageThread	
<p><b>Verantwortlichkeit</b></p> <ul style="list-style-type: none"> <li>- lädt das grosse Bild in einem eigenen Thread</li> </ul>	<p><b>Information</b></p> <ul style="list-style-type: none"> <li>- kennt die Grösse des grossen Bildes</li> </ul>

	<b>Creator</b> -
--	---------------------

#### Klasse LayerList

<b>Verantwortlichkeit</b> - liest aus dem XML File die Information um Layer Objekte zu erzeugen	<b>Information</b> - speichert eine Liste mit allen Layer einer Verbindung - kennt den XML File einer Verbindung  <b>Creator</b> - erzeugt Objekte des Typs Layers
--	---

#### Klasse Layer

<b>Verantwortlichkeit</b> - speichert alle Informationen, die ein Layer zur Verfügung hat	<b>Information</b> - kennt den Status eines Layers ( aktiviert oder nicht aktiviert) - speichert den Namen des Layers - weiss, ob ein Layer fähig ist zusätzliche Informationen zu liefern  <b>Creator</b> -
--	--

#### Klasse ImageState

<b>Verantwortlichkeit</b> - verwaltet den Zustand der Bilder, welche im History gespeichert sind.	<b>Information</b> - weiss, welche Layer von einem Bild aktiviert waren - kennt die Clipinformationen von den Bildern, welche im History gespeichert sind - speichert eine Referenz auf jedes Bild der History  <b>Creator</b> -
--	--

#### Klasse MapWrapper

<b>Verantwortlichkeit</b> - berechnet die Dimension des Dokumentes - wandelt die Grösse des Clips in BBOX	<b>Information</b> - kennt die Koordinaten der Graphiken (BBOX) - kennt die Grösse des Bildausschnittes (Clip)
---	--

Koordinaten	<b>Creator</b> -
-------------	---------------------

Klasse Queryinfo	
<b>Verantwortlichkeit</b> - holt die Information der selektierten Layer auf dem MapServer. - zeigt nur die Information eines Layers	<b>Information</b> - kennt die Koordinatenposition der Maus - weiss, welche Layer aktiviert sind  <b>Creator</b> -

Klasse LayerDialog	
<b>Verantwortlichkeit</b> - GUI der Layer, zeigt alle darstellbaren Layer	<b>Information</b> - hat eine Liste mit allen Layer  <b>Creator</b> -

Klasse ConnectionDialog	
<b>Verantwortlichkeit</b> - GUI der Verbindungen - liest die URL, welche vom Anwender eingegeben wurden - löscht Verbindungen	<b>Information</b> - kennt die URL der MapServers - weiss, welche Verbindungen aktiv sind  <b>Creator</b> -

Klasse InfoDialog	
<b>Verantwortlichkeit</b> - zeigt die Information zu einem Punkt eines Layers in einer Tabelle	<b>Information</b> - kennt die Information, die angezeigt werden soll  <b>Creator</b> - erzeugt ein Objekt des Typs CreateTable

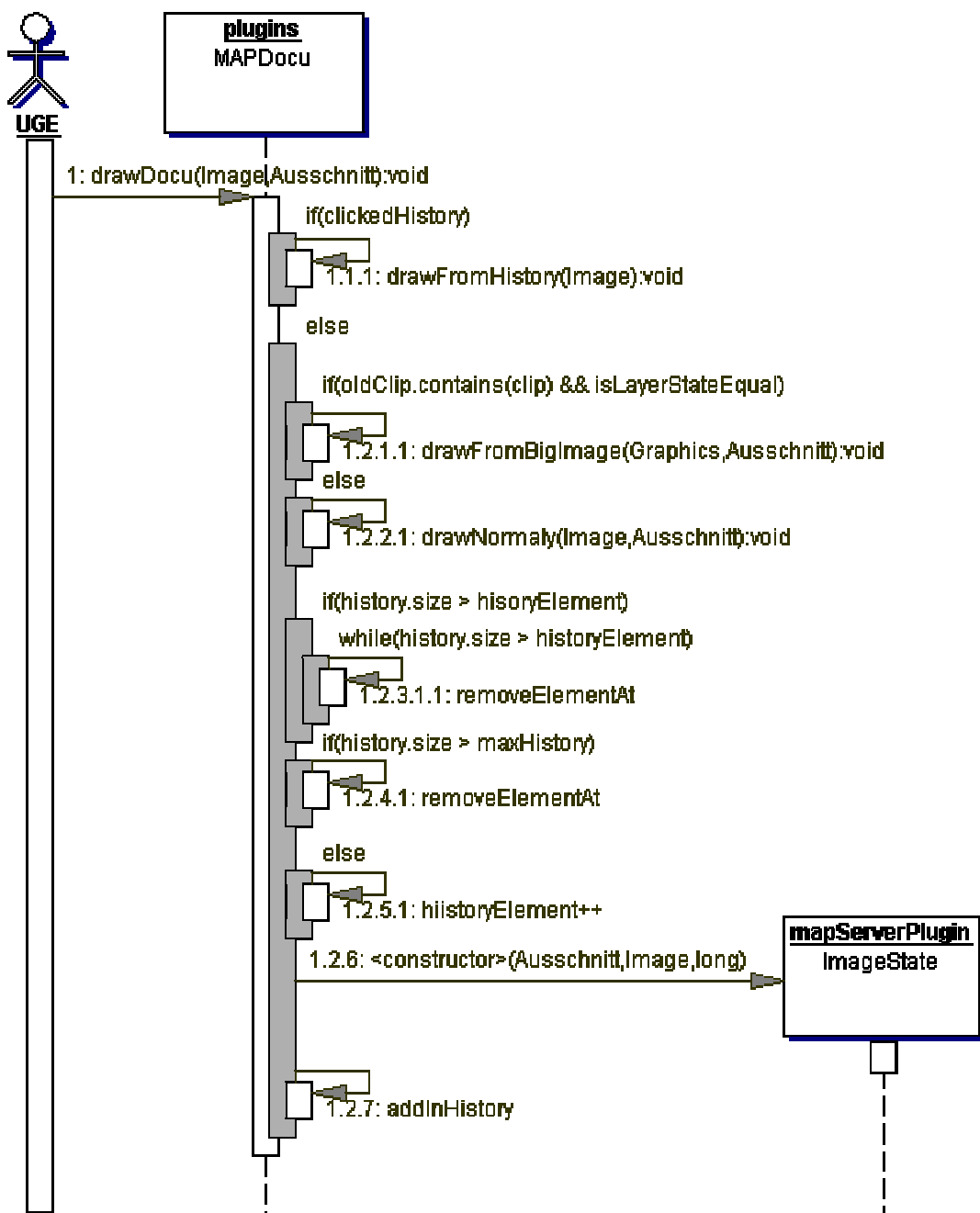
Klasse CreateTable	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"><li>- fügt die Information des InfoDialoges in einer Tabelle ein</li></ul>	<b>Information</b> <ul style="list-style-type: none"><li>- kennt die Information, die angezeigt werden soll</li></ul> <b>Creator</b> <ul style="list-style-type: none"><li>-</li></ul>

Klasse LayerDialogCellRenderer	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"><li>- zeigt die Layer mit dem passenden Icon</li></ul>	<b>Information</b> <ul style="list-style-type: none"><li>- weiss, welche Layer zur Verfügung stehen</li><li>- weiss, welche Layer selektiert sind</li></ul> <b>Creator</b> <ul style="list-style-type: none"><li>-</li></ul>

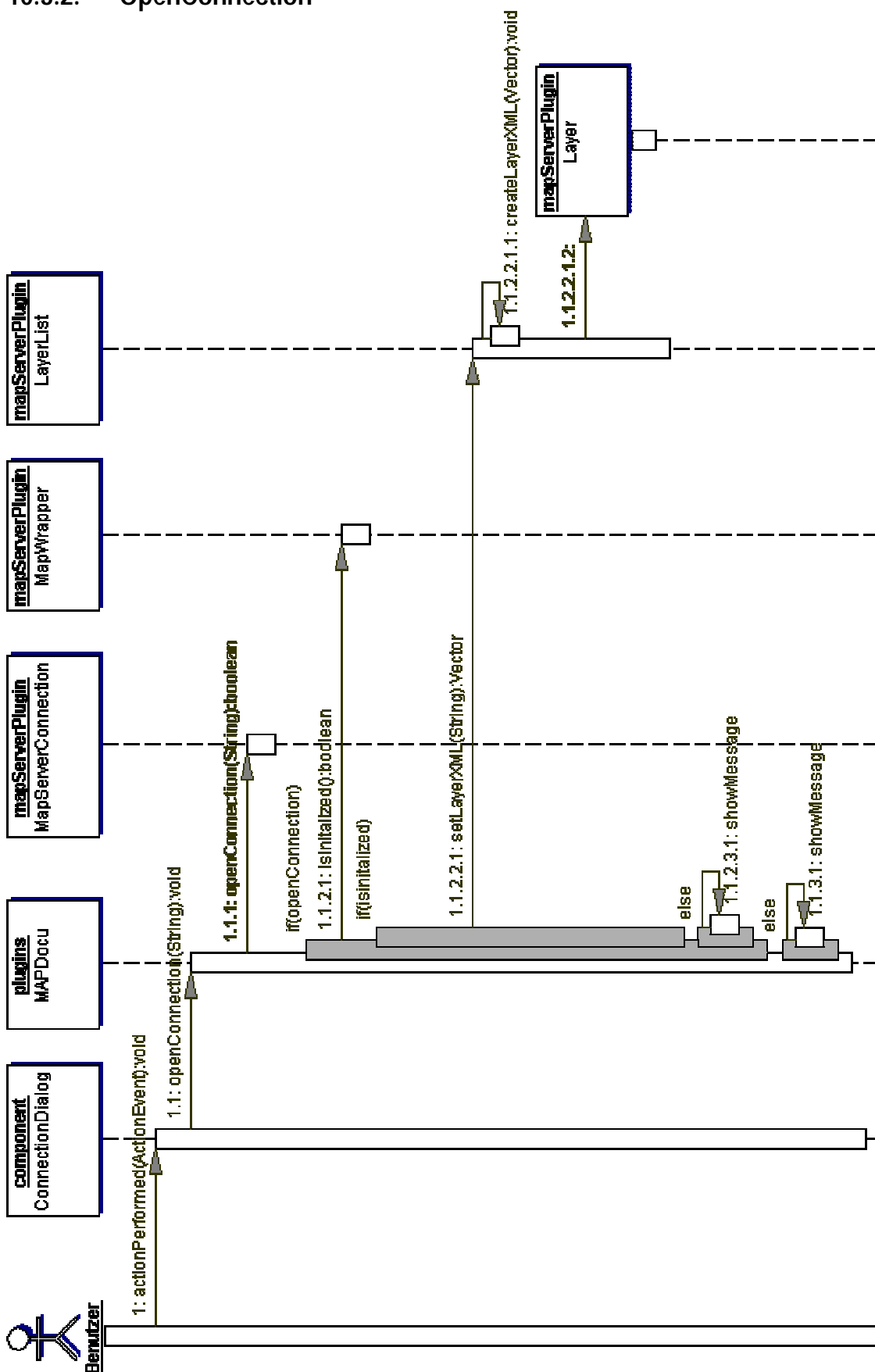
Klasse ConnectionDialogCellRendrer	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"><li>- zeigt die Liste der Verbindungen mit einem Icon</li></ul>	<b>Information</b> <ul style="list-style-type: none"><li>- weiss, welche Verbindungen aktiv sind</li></ul> <b>Creator</b> <ul style="list-style-type: none"><li>-</li></ul>

### 10.3. Sequenzdiagramm

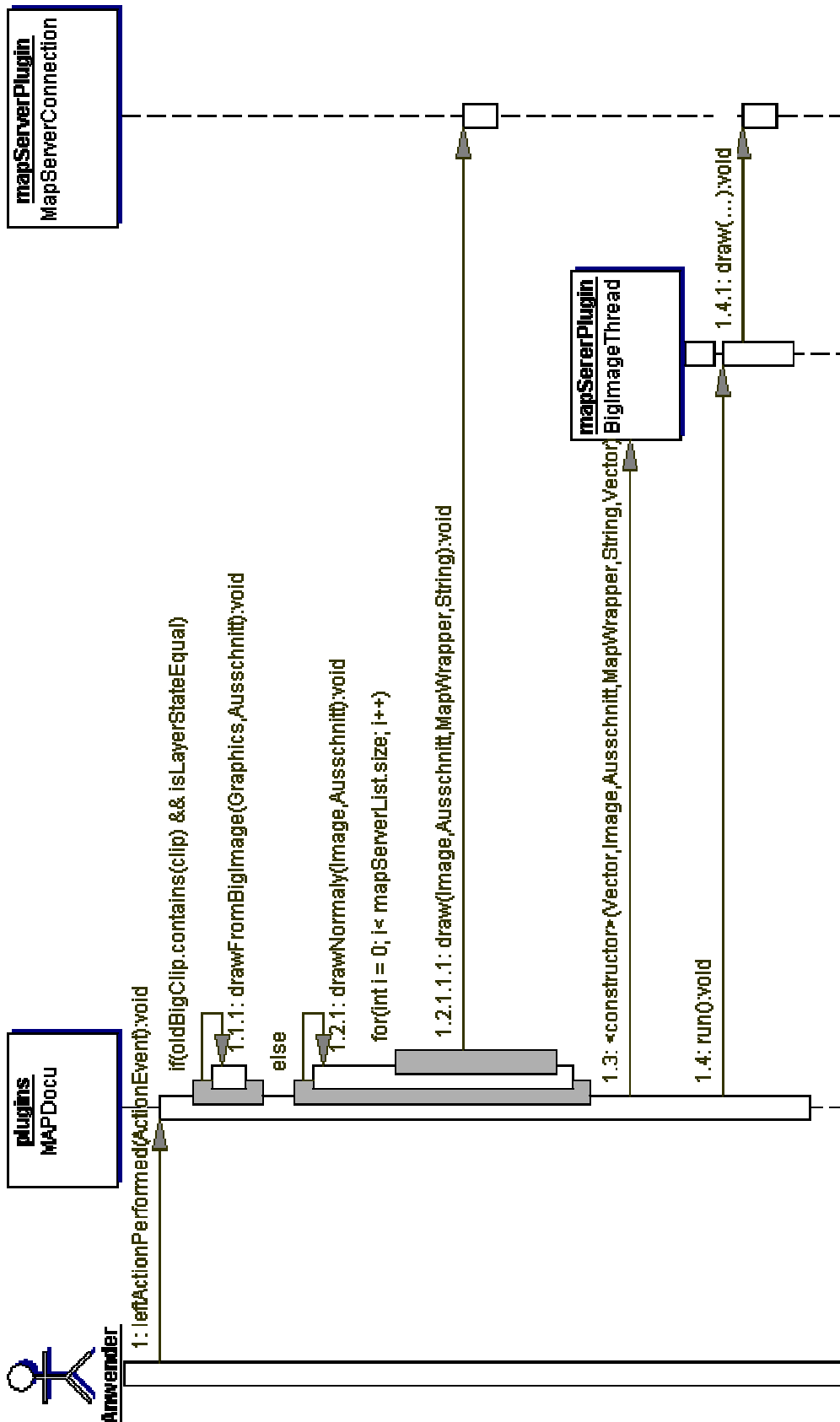
#### 10.3.1. AddHistory



### 10.3.2. OpenConnection

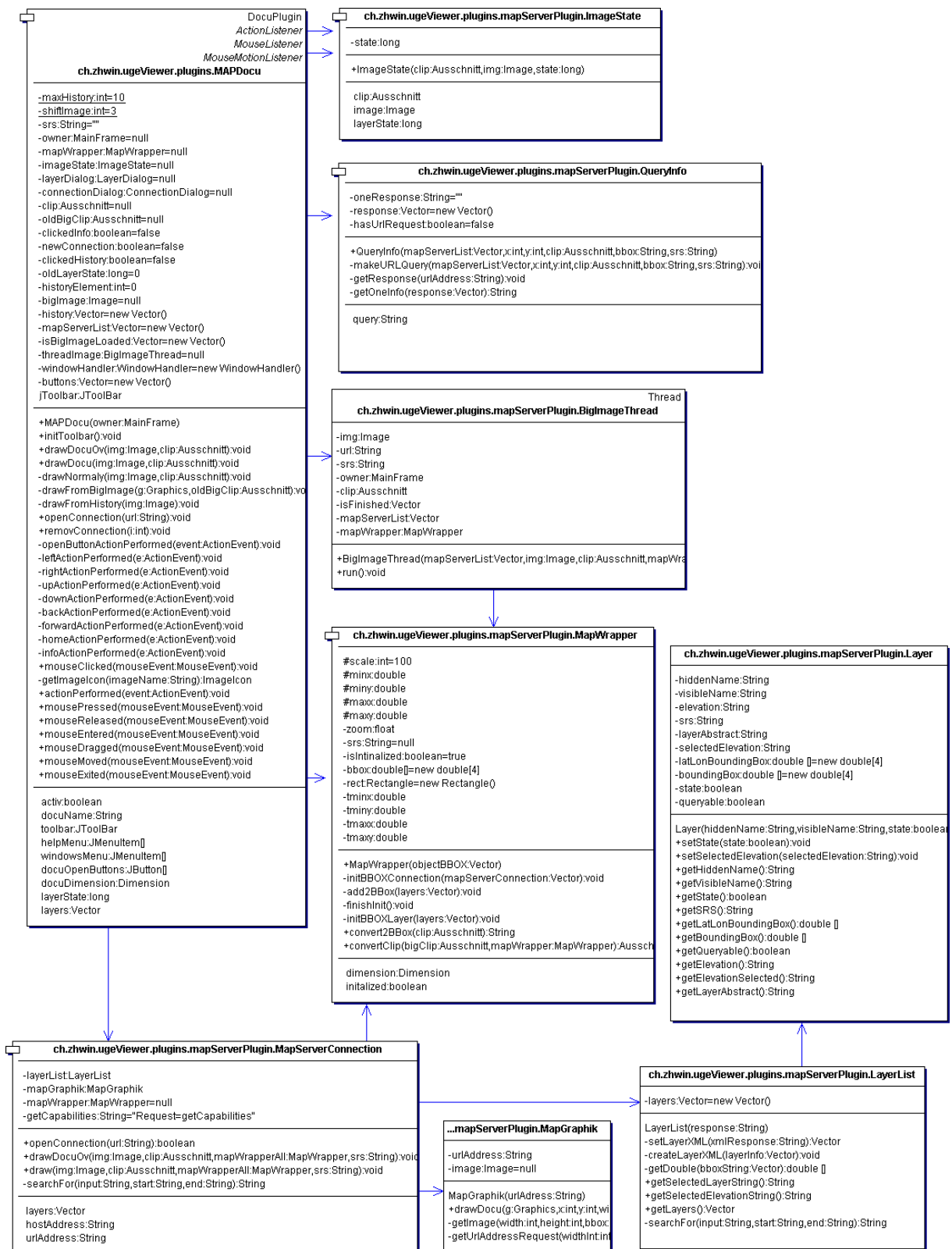


### 10.3.3. Verschieben

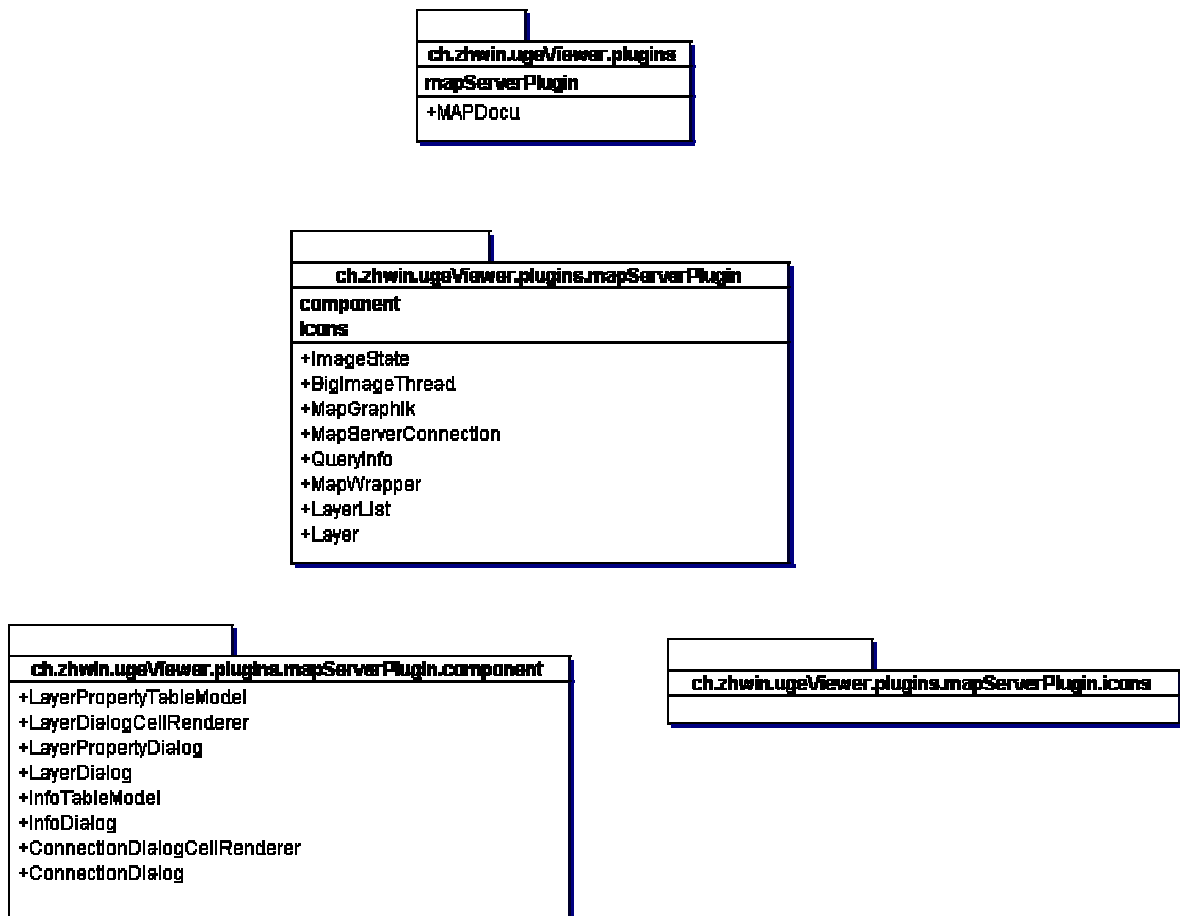




### 10.3.5. Klassendiagramm



### 10.3.6. Package mit allen Klassen



## 11. Schlussbericht

### 11.1. Funktionsweise

#### 11.1.1. XML anfordern/Layer und Graphik initialisieren

Beim Starten des MapServerplugins öffnet sich unter anderem das Fenster „Connectiondialog“. Im Textfeld kann die Adresse des MapServers eingegeben werden. Mit der Methode `getCapabilities` wird von dieser Adresse ein XML File angefordert, in welchem alle Informationen über die Karte, die der MapServer verwaltet, stehen. Aus diesem File liest das Programm die für uns relevanten Informationen wie:

- Namen der Layer. Die Selektion der Layer geschieht über diesen Namen. Tags: `<Title>`
- Namen der Layer für die Anforderung an den MapServer. Tags: `<Name>`.

- Spatial Reference System. Tags: <SRS>
- Bounding Box Corner (BBOX). Tags: <BoundingBox>
- Tags: <queryable>, zeigt an das der Layer zusätzliche Information anbietet.
- Liste der Höhenauszüge (Sublayer) welche einzeln dargestellt werden können.
- Tags: <name=elevation/>

Auszug der XML Datei vom Anhang.

```

=> <Layer queryable="0" >
  <Name>DEMO</Name>
  <Title>UMN MapServer Itasca Demo</Title>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="388108" miny="5.20312e+006" maxx="500896"
maxy="5.31024e+006" />
  <BoundingBox SRS="EPSG:26915" minx="388108" miny="5.20312e+006" maxx="500896"
maxy="5.31024e+006" />
  <ScaleHint min="0.498903" max="773.299" />
=> <Layer queryable="0" >

```

Diese Anforderung (Request=getCapabilities) geschieht nur bei der ersten Verbindung.

Die zur Verfügung stehenden Layer werden im Fenster „LayerDialog“ aufgelistet. Falls der Layer zusätzliche Informationen anbietet, erscheint ein „I“ Symbol neben dem Layername. Die Namen aller Server, welche zum aktuellen Bild beitragen, werden im „Connectiondialog-Fenster“ aufgelistet.

Als Start BBOX wird ein Rechteck ausgewählt, welches alle BBOX aller Layer umfasst.

Danach werden alle Bilder direkt mit der GetMap (Request=GetMap) Methode angefordert. Dabei setzen wir aus der Grösse des sichtbaren Fensters, dem Zoomfaktor und den selektierten Layern, die URL für die Anforderung des Bildes zusammen. Der MapServer berechnet das Bild und sendet es an unsere Applikation. Im Hintergrund in einem eigenen Thread wird das grosse Bild geladen. Falls der Benutzer zusätzliche Informationen zu einem Layer wünscht, wird die Methode (Request=feature\_info) benutzt. Der vorher erwähnten URL werden die Mauskoordinaten und die query Layer (QUERY\_LAYERS=dlgstln2) beigefügt. Die Antwort wird in einem eigenen Textfenster dargestellt.

Der Anwender kann Verbindungen zu mehreren MapServern aufbauen. Wichtig ist dabei, dass diese MapServer den gleichen SRS Koordinatenursprung haben. Ansonsten können die Bilder der MapServer nicht überlagert werden.

### 11.1.2. Suchen von zusätzlichen Informationen

Das Suchen von zusätzlichen Informationen geschieht nur, wenn ein Layer, welcher diesen Dienst unterstützt, aktiviert ist. Falls kein Layer aktiviert ist, gibt die Applikation eine Fehlermeldung aus. Falls ein Layer aktiviert ist, aber zu diesem Punkt keine Information vorhanden ist, öffnet sich das InfoFenster mit der Meldung, dass keine Information zur Verfügung steht. Im Normalfall wird die Information im InfoFenster dargestellt.

### 11.1.3. Layerwahl

Jeder Layer kann separat ein- oder ausgeschaltet werden. Dabei muss jedes Mal ein Zugriff auf den MapServer gemacht werden. Die Layer werden immer zu einem Bild kombiniert an den Client geschickt. Der Vorteil liegt darin, dass die Bilder mit mehreren Layern schnell dargestellt werden, da nur ein Zugriff auf dem MapServer gemacht werden muss. Wenn wir hingegen für jeden Layer eine Anfrage machen würden, müssten wir viel länger warten bis alle Layer vorhanden sind und das Bild dargestellt wird. Um einen Layer darzustellen braucht der MapServer ca. 1 Sekunde.

### 11.1.4. Layerparameter

Die Layerparameter können angeschaut werden, indem im Layerdialogfenster ein Doppelklick auf den entsprechenden Listeneintrag ausgeführt wird. Es öffnet sich ein Fenster, in welchem die Parameter dargestellt werden. Layer, welche aus Sublayern bestehen, listen in der Zeile „elevation“ die verfügbaren Höhenauszüge auf. In diesem Fall kann in der Zeile „selected elevation“ die gewünschte Höhe eingegeben werden. Dieses Feld befindet sich direkt unterhalb des „elevation“ Feldes.

### 11.1.5. Zoomen

Falls der Zoomfaktor das Maximum erreicht, gibt der MapServer eine Fehlermeldung als GIF-Datei zurück. Dieses Bild wird unverändert dargestellt.

## 11.2. Probleme

Das Überlagern von verschiedenen MapServern konnte nur bedingt getestet werden, da uns nur der UMN MapServer zur Verfügung stand. Das Überlagern konnte nur durch öffnen von mehreren Verbindungen zum selben Server getestet werden. Dabei lief die Applikation einwandfrei.

Aus zeitlichen Gründen haben wir die SQL-Schnittstelle nicht programmieren können. Sie ist aber für die Hauptanwendung unserer Applikation nicht relevant.

### **11.3. Code**

Der Programm Code und ein vollständiges Javadoc befinden sich auf der beigelegten CD.

### **11.4. Testbericht**

Der Map-Viewer als lauffähige Applikation konnte durch Benützen getestet werden. Wir haben die Bedienungsanleitung als Testskript verwendet. Unsere Applikation funktioniert einwandfrei.



# Teil III

INTERLIS für  
MapServer

## 12. Grundlagen

### 12.1. INTERLIS

#### 12.1.1. Was ist INTERLIS

Im Jahre 1991 erschien erstmals „INTERLIS –ein Datenaustausch-Mechanismus für Land-Informationen-Systeme“. Hauptziel und Zweck von INTERLIS ist die möglichst präzise Beschreibung von Daten. Dieser Mechanismus besteht aus einer konzeptionellen Beschreibungssprache und einem sequentiellen Transferformat mit besonderer Berücksichtigung raumbezogener Daten (kurz Geodaten). Damit wird die Kompatibilität unter den System und eine langfristige Verfügbarkeit, d.h. Archivierung und Dokumentation der Daten ermöglicht.

INTERLIS besteht aus:

- Datenmodell (\*.ili)
- INTERLIS-XML Datensatz (\*.xml)
- Graphikmodell (\*.ili)
- und einer Sammlung von Signaturobjekten (Signaturbibliothek) (\*.xml)

##### 12.1.1.1. Datenmodell

Datenmodelle geben Auskunft über den Aufbau von Objekten. Von den Objekt-Attributen bis zu Einheiten und anderem. Mit Hilfe von UML-Diagrammen lassen sich Datenmodelle einfach graphisch modellieren. Bei Eisenhut-Informatik ist zurzeit ein (JAVA-)UML-Editor in Arbeit, der ein erstelltes UML-Diagramm als INTERLIS-Datenmodell abspeichern kann.

##### 12.1.1.2. IXML Datensatz

Die Datensätze werden im weitverbreiteten XML-Format geliefert. Bei der Umsetzung müssen gewisse Regeln beachtet werden, die am Anfang der Datei stehen.

##### 12.1.1.3. Graphikmodell

Im Graphikmodell wird beschrieben welche Objekte wie dargestellt werden sollen. Zu diesem Zweck zieht das Graphikmodell die Signaturenbibliothek zu Hilfe. Durch schreiben von verschiedenen Graphikmodellen lassen sich die gleichen Daten unterschiedlich darstellen.

#### **12.1.1.4. Signaturenbibliothek**

Die Signaturenbibliothek enthält Symboldefinitionen für Fixpunkte und Bäumen, sowie Linien-, Textanschrift- und Flächensignaturen.

Die Signaturenbibliothek ist ein XML-Datensatz gemäss dem Signaturenmodell (ReferenzHandbuch Anhang K). Die Signaturenbibliothek wird im Graphikmodell für die Darstellung von Objekten verwendet, die als Symbole definiert wurden.

Detailliertere Informationen sind dem INTERLIS-Referenzhandbuch zu entnehmen.

#### **12.1.2. Attribute von INTERLIS 2**

Grundsätzlich gibt es Attributtypen

- a) mit einem Wertebereich,
- b) mit einem Faktor (kann ein Pfad, ein Funktionsaufruf, ein Laufzeitparameter (z.B. der aktuelle Bildschirmmasstab) oder eine Konstante sein) oder
- c) Referenzattribut (REFERENCE TO; nur in Strukturen), oder
- d) einer Komposition von Strukturelementen (Kompositionsattribut mit BAG/LIST OF und einer INTERLIS-Struktur).

##### **12.1.2.1. Zeichenketten**

Beim Datentyp Zeichenkette (TEXT) ist primär die Länge der Zeichenkette von Interesse.

Bei INTERLIS 1 wurde das Datum ebenfalls als Text deklariert, mit Jahr, Monat und Tag.

##### **12.1.2.2. Aufzählungen**

Mit einer Aufzählung (Enumeration) werden die für diesen Typ zulässigen Werte festgelegt. Eine Aufzählung kann aber nicht nur linear, sondern auch baumartig strukturiert sein. Als mögliche Werte kommen dann nur ganze Pfade von der Wurzel bis zum Blatt in Frage. Die einzelnen Knoten werden bei der Pfadangabe durch Punkte getrennt.

### 12.1.2.3. Textausrichtungen

Dieser Typ (HALIGNMENT und VALIGNMENT) legt die "Anfasspunkte" fest von Beschriftungstexten in Bildschirm-/Papierkarten und ist ein Spezialfall einer Aufzählung.

### 12.1.2.4. Boolean

Bei INTERLIS wird der Boolean Typ ebenfalls als Aufzählung deklariert. Die beiden Werte sind „true“ und „false“.

### 12.1.2.5. Numerische Datentypen

Bei INTERLIS kann nebst der oberen und unteren Grenze zusätzlich die Einheit angegeben werden. Ebenso wird mit dem Kennwort Circular angezeigt, dass nach dem Maximum das Zählen beim Minimum fortgeführt wird (Roundtrip). Neben Zeitangaben können so auch Winkelangaben definiert werden. Ausserdem gibt es auch Dezimalkonstanten 'PI' und 'LNBASE'.

### 12.1.2.6. Strukturierte Wertebereiche

Strukturierte Wertebereiche sind zur Behandlung von Datum und Zeit. Es müssen Minimal- und Maximalwert und die strukturierte Einheit angegeben werden. Zudem kann ein numerisches Referenzsystem (z.B. Zeitzone) gemacht werden. Beispiel:

DOMAIN

UhrZeit = 0:00:00.000 .. 23:59:59.999 [MEZ];

### 12.1.2.7. Koordinaten

Koordinaten können ein-, zwei- oder drei-dimensional definiert werden. Jede dieser Dimensionen ist wiederum als Zahl zu verstehen.

**DOMAIN**

```
CHLkoord = COORD    480000.00 .. 850000.00 [m] {CHLV03/1},  
                    60000.00 .. 320000.00 [m] {CHLV03/2},  
                    ROTATION 2 -> 1;
```

**12.1.2.8. Linienzüge**

Polylines (zu deutsch Linienzüge): ein Linienzug ist eine endliche Folge von Kurvenstücken (d.h. Strecken und Kreisbogen) mit Stützpunkten im Innern des Linienzuges. Ob die definierten Punkte nun mit geraden Linien oder mit Bogen verbunden werden, wird mit der Linienform angegeben (STRAIGHTS, ARCS). Eine Linie besteht immer aus mindestens zwei Elementen, dem StartSegment und einem oder mehreren LinienSegmenten.

Die exakte Definition ist im Referenzhandbuch in 2.8.11.1 Geometrie des Linienzugs nachzulesen.

**12.1.2.9. SURFACE**

Eine Fläche ist durch eine äussere und allenfalls eine oder mehrere innere Randlinien begrenzt, d.h. sie kann (muss nicht) Enklaven - auch mehrere - enthalten ("Willisauer-Ring"). AREA kann als SURFACE mit einer Nicht-Überlappungs-Konsistenzbedingung aufgefasst werden.

**12.1.2.10. Komposition**

INTERLIS kennt ebenso Listen und Ansammlungen von verschiedenen Objekten.

LIST OF kann am besten mit einem Array verglichen werden und BAG OF mit einem Vector in Java. Im Gegensatz zur Liste müssen die Inhalte im "Bag" nicht geordnet sein.

**12.1.2.11. Referenz**

Mit einer Referenz wird ein Fremd-Schlüssel auf ein anderes Objekt angegeben.

**12.1.2.12. Assoziationen**

Assoziationen sind Beziehungsangaben zwischen zwei Objekten

## **12.2. MapServer**

### **12.2.1. MapServer Daten**

Der MapServer ist ein CGI-basiertes Programm. Neben den GeoDaten gibt es noch das Map-File das die Verbindung zu den Daten definiert sowie auch gewisse Symbole festlegt.

Für die Einbindung der GeoDaten unterstützt der MapServer ebenfalls mehrere Möglichkeiten. Namentlich sind diese ESRI-Shape-Files, PostGIS-DB, Raster-Images und weitere. Bei dieser Arbeit haben wir uns auf die PostGIS-Schnittstelle konzentriert.

### **12.2.2. PostGreSQL / PostGIS**

PostgreSQL ist eine freie SQL-Datenbank. Insbesondere ist es aber eine schnelle, hochgradig konfigurierbare Datenbank, die dem Vergleich mit den Grossen des kommerziellen Geschäfts gelassen entgegen sehen kann. PostGIS ist eine Erweiterung von PostgreSQL, deren wichtigster Beitrag wohl die Implementierung eines grossen Teils der Simple Features des OGC ist. Der MapServer kann mit Unterstützung für PostGIS kompiliert werden. Dank PostGIS ist es der PostgreSQL-Datenbank möglich, Geometrie-Objekte zu verwalten.

### **12.2.3. INTERLIS vs MapServer**

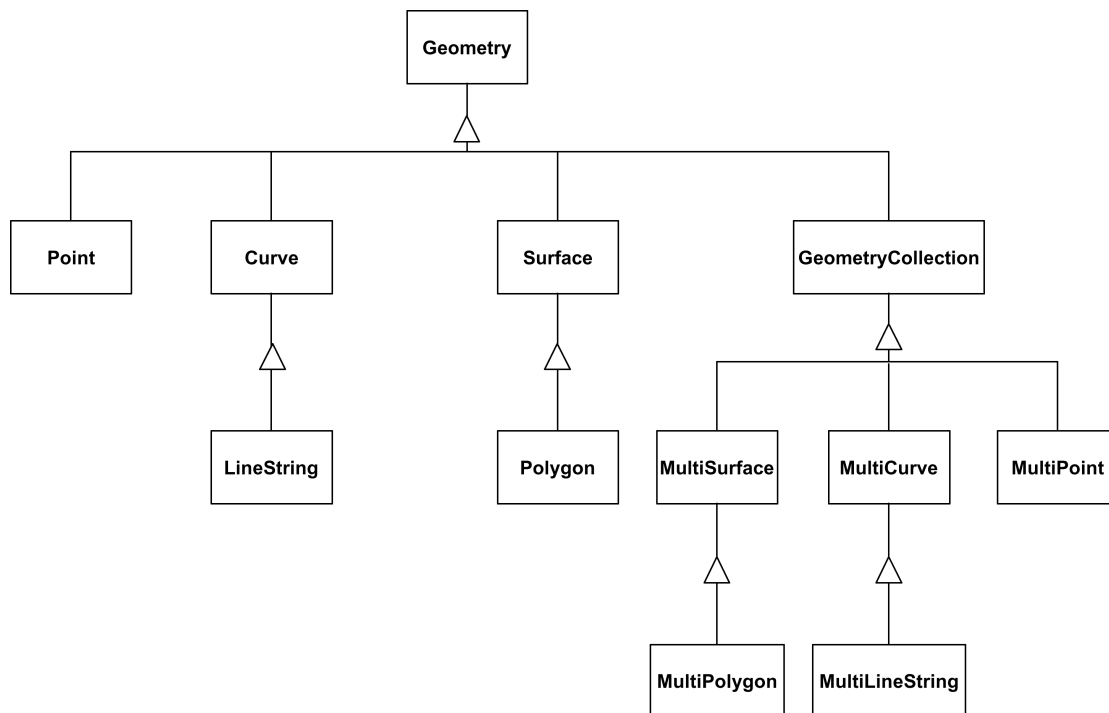
INTERLIS trennt die GeoDaten klar von den Beschreibungen wie sie dargestellt werden. Das Gleiche geschieht beim MapServer. Die Daten werden entweder von Shape-File oder von einer Datenbank geliefert. Teile des INTERLIS-GraphicModelles sind jedoch nicht so einfach zuzuordnen. Farben und Symbole sind ein Bestandteil des map-Files wobei Views auf der Datenbank-Seite abzubilden sind.

## **12.3. OGC Simple Features for SQL**

Das Open GIS Consortium definiert Geometrie-Objecte sowie verschiedene Funktionen für SQL92 mit Geometry Daten-Typen.

### **12.3.1. Geometrie Objekte**

Gemäss OGC werden die Geometrie-Objekte wie folgt gegliedert.



Bei der Umsetzung wurde darauf geachtet, dass nur die einfachen Objekte Point LineString und Polygon gebraucht werden. Natürlich schreibt auch das OGC vor, wie Geometrie-Attribute bei Tabellen hinzugefügt werden. Bei der Implementation wurde jedoch auf die PostGIS Dokumentation zurückgegriffen. PostGIS implementiert die „Simple Features for SQL“ von OGC weitgehend.

### 12.3.2. PostGIS

Die Geometrie-Attribute werden erst nach dem Erstellen einer Tabelle hinzugefügt.

Dies geschieht mit der Funktion AddGeometryColumn.

AddGeometryColumn(<db\_name>, <table\_name>, <column\_name>, <srid>, <type>, <dimension>)

<db\_name>: Name der Datenbank

<table\_name>: Name der Tabelle wo das Attribute hinzugefügt werden soll

<column\_name>: Name des Attributes/Kolonne

<srid>: Interne Referenz zur SPATIAL\_REF\_SYS Tabelle ( benutze -1 für undefiniert)

<type>: Beispiele: POINT, LINESTRING, POLYGON, etc

<dimension>: 2 oder 3 Dimensionen

Hier ein SQL-Beispiel

```
CREATE TABLE PARKS ( PARK_ID int4, PARK_NAME varchar(128), PARK_DATE date,  
PARK_TYPE varchar(2));  
SELECT AddGeometryColumn('parks_db', 'parks', 'park_geom', '-1', 'MULTIPOLYGON', 2 );
```

Bei den Inserts wird mit dem Text GeometryFromText eine Funktion aufgerufen, die aus dem Attribute ein Geometrie-Objekt erzeugt.

#### Point Beispiel:

```
Insert into streetname position values( 'x5' , '15.0' , GeometryFromText( 'POINT(71.66 45.231)', '-1'));
```

#### LineString Beispiel:

```
Insert into streetaxis values( 'x12' , GeometryFromText( 'LINESTRING( 94.99 50.109, 89.504 65.795,  
83.594 75.598, 71.774 80.712, 11.423 91.154)', '-1' ) );
```

#### Polygon Beispiel:

```
Insert into landcover values ( 'x33' , 'other' , GeometryFromText( 'POLYGON(( 31.11 83.75, 31.14  
36.458, 50.669 42.579, 51.432 60.469, 57.06 44.638, 87.839 54.138, , 67.549 77.788, 31.11 83.75),  
(41.2 59.302, 39.038 60.315, 35.661 63.735, 35.525 66.268, 36.741 69.688, 47.955 75.515, 55.927  
72.348, 58.899 68.928, 57.818 63.862, 56.197 62.595, 53.36 64.115, 48.766 67.408, 43.362 60.315,  
41.2 59.302), (79.9 55.839, 60.489 49.608, 69.938 61.721, 67.678 68.781, 75.351 70.932, 79.9  
55.839))', '-1' ) );
```

## 13. Analyse Phase

### 13.1. Ist-Zustand

Der UMN MapServer kann bereits mit verschiedenen Datenquellen kommunizieren. Eine Möglichkeit ist die PostgreSQL Datenbank mit dem PostGIS Zusatz. Bei der Standard-Installation von PostGIS wird auch ein Tool zur Konvertierung von ESRI Shape Dateien zu pg-SQL mitgeliefert. Da INTERLIS 2 immer noch in der Entwicklungs-Phase ist, existieren nahezu keine Programme, die damit umgehen können. Da es sich bei INTERLIS um ein Schweizer Format handelt, ist die Verbreitung gering.

## 13.2. Soll-Zustand

Die GeoDaten, die mit INTERLIS beschrieben sind, sollen mit dem MapServer dargestellt werden können. Als Datenhalter kann zu diesem Zweck die PostgreSQL Datenbank dienen. Ein Konvertierungstool, das die INTERLIS Daten in SQL Befehle umwandelt, könnte die Lücke schliessen und GeoDaten für den Mapserver bereitstellen. Das SQL-Script kann dann bei der Datenbank abgespielt werden. Die Tabellen könnten entsprechend des INTERLIS Daten-Modelles kreiert werden. Aus den INTERLIS-Datensätzen, konvertiert in INSERT-Befehle, kann die Datenbank aufgefüllt werden.

Die INTERLIS-Data-Model Dateien werden geladen. Danach durch den INTERLIS-Compiler in eine interne Struktur umgewandelt. Diese wird in eine neue interne Datenbank-Struktur gewandelt. Anschliessend wird diese neue Struktur als ein SQL-Script des DB-Schemas abgespeichert. Mit der internen Datenbankstruktur kann die IXML Datei direkt in Inserts konvertiert werden.

- load (laden des INTERLIS-Dateien)
- convert (umwandeln in ein DB-Schema)
- save (speichern des sql-create Scriptes)
- save (verarbeiten und speichern des sql-insert Scriptes)

## 13.3. Features

### 13.3.1. OO-Schema zu ER-Schema

Im Anhang befindet sich ein Text der verschiedene Möglichkeiten beschreibt, wie die Modelle umgeformt werden können. Die Objekte werden gemäss der 3. Abbildungs-Variante umgesetzt.

### 13.3.2. Abbilden von INTERLIS Daten-Modellen zum DB-Model

Jede Klasse wird zu einer Tabelle. Strukturen werden nur zu Tabellen umgewandelt, wenn diese von einer Klasse als Kompositionsstruktur gebraucht wird.

Bei erweiterten Klassen werden nur die neu definierten und erweiterten Attribute übernommen. Mit einem Fremd-Schlüssel auf ein Tupel der Super-Klasse werden die anderen Attribute gefunden. Durch

Definieren einer View können dann alle Attribute gefunden werden, als ob es eine komplette Tabelle der erweiterten Klasse gibt.

Strukturen werden meistens von Klassen als LIST oder BAG benutzt und können nicht selbständig auftreten. Deshalb erhält jedes Tupel den Eigentümer in Form eines Fremd-Schlüssels. Mit einem zusätzlichen Attribute, einer Nummer, wird aus diesem und dem Fremdschlüssel ein Primärschlüssel gebildet

Bei Assoziationen wird unterschieden, ob es sich um eine 1:n oder eine n:m Beziehung handelt. Im ersten Fall wird auf der n-Seite ein Fremdschlüssel auf das andere Objekt hinzugefügt. Bei der n:m Beziehung wird eine Association zu einer Tabelle mit zwei Fremd-Schlüsseln. Ist es eine 1:1 Beziehung wird ein Fremdschlüssel bei der zweiten Tabelle hinzugefügt.

### 13.3.3. Abbilden von INTERLIS Attributen

INTERLIS	SQL92 mit Geometrie Typen
<b>Text</b>	
-Date (INTERLIS1)	date
-Text*n	varchar(n)
<b>Numeric</b>	
-Numeric(min, max)	numeric(abs,scale);
<b>Coord</b>	
-Coord2	point(x,y)
-Coord3	point(x,y,z)
<b>Composition</b>	
-List	neue Tabelle mit Fremdschlüssel zu Besitzer - und Ordnungsnummer
-Bag	- und Ordnungsnummer
<b>Enumeration</b>	
-Boolean	boolean
-HALIGNMENT/VALIGNMENT	siehe Enumeration
-Enumeration	varchar(n)
<b>Line</b>	
-Polyline	linestring
-Surface	polygon
-Area	polygon
<b>Reference</b>	
-Reference	Fremdschlüssel

## Assoziationen

1:n	Fremdschlüssel bei Tabelle n
n:m	neue Table mit zwei Fremdschlüsseln

## Spezialfall Kreisbogen

Da es in SQL92 mit Geometrie Objekten keine Kreisbögen gibt, ist es nicht möglich, diese direkt abzubilden. Deshalb müssen diese in einzelne geraden-Abschnitte unterteilt werden.

## 13.4. INTERLIS Kreisbögen

### 13.4.1. Definition

Ein Kreisbogenstück wird zusätzlich zum Endpunkt mit einem Zwischenpunkt beschrieben. Dieser ist nur in der Lage von Bedeutung. Die Höhe wird linear zwischen Anfangs- (= Endpunkt des letzten Segmentes, bzw. Startpunkt des Linienzuges) und Endpunkt interpoliert. Der Zwischenpunkt ist kein Stützpunkt des Linienzuges. Er soll möglichst exakt in der Mitte zwischen Anfangs- und Endpunkt liegen. Da der Zwischenpunkt in der gleichen Genauigkeit angegeben wird wie die Stützpunkte, kann der berechnete Radius erheblich vom effektiven Radius abweichen. Wird der effektive Radius angegeben, ist er für die Kreisbogendefinition massgebend. Der Zwischenpunkt legt nur noch fest, welcher der vier möglichen Kreisbogen der Gewünschte ist. Der Zwischenpunkt darf aber auch in diesem Fall um höchstens 2 Einheiten von der Spur des aus dem Radius gerechneten Kreisbogens abweichen. (INTERLIS Referenzhandbuch)

### 13.4.2. Berechnungen

Damit bei Flächenberechnungen der Geometer die gleichen Werte vorliegen, muss das Kreisbogen-Segment flächengleich mit dem, der in Geraden unterteilten Fläche sein.

Neben der Flächengleichheit darf die Abweichung vom gegebenen Bogen ebenfalls ein gewisses Mass nicht überschreiten. Alle Linien müssen zudem gleich lang sein.

Bei der Unterteilung in gerade Linien kann in drei Abschnitten vorgegangen werden:

- Zentrumsberechnung
- Anzahl Linien, neuer Radius
- Neue Stützpunkte berechnen

### 13.4.2.1. Zentrumsberechnung

Zwischen den einzelnen Punkten wird eine Mittelebene gebildet. Als zusätzliche Ebene wird die Ebene genutzt, die durch die drei Punkte definiert wird.

Mittelebene

$$e: a_1x + b_1y + c_1z + d_1 = 0$$

Normalvektor(a,b,c)

$$a_1 = (S.x - C.x)$$

$$b_1 = (S.y - C.y)$$

$$c_1 = (S.y - C.y)$$

$$a_2 = (F.x - C.x)$$

$$b_2 = (F.y - C.y)$$

$$c_2 = (F.y - C.y)$$

$$a_3 = (S.x - F.x)$$

$$b_3 = (S.y - F.y)$$

$$c_3 = (S.y - F.y)$$

Mittelpunkt M

$$M1.x = (S.x + C.x) / 2$$

$$M1.y = (S.y + C.y) / 2$$

$$M1.z = (S.z + C.z) / 2$$

$$M2.x = (F.x + C.x) / 2$$

$$M2.y = (F.y + C.y) / 2$$

$$M2.z = (F.z + C.z) / 2$$

$$M3.x = (S.x + F.x) / 2$$

$$M3.y = (S.y + F.y) / 2$$

$$M3.z = (S.z + F.z) / 2$$

$$d_1 = -(a_1 * M1.x + b_1 * M1.y + c_1 * M1.z)$$

$$d_2 = -(a_2 * M2.x + b_2 * M2.y + c_2 * M2.z)$$

$$d_3 = -(a_3 * M1.x + b_3 * M1.y + c_3 * M3.z)$$

Ebene durch die drei Punkte

$$e4: a_4x + b_4y + c_4z + d_4 = 0$$

Normalvektor(a,b,c)

$$a_4 = (S.y - C.y)(F.z - C.z) - (S.z - C.z)(F.y - C.y)$$

$$b_4 = (S.z-C.z)(F.x-C.x)-(S.x-C.x)(F.z-C.z)$$

$$c_4 = (S.x-C.x)(F.y-C.y)-(S.y-C.y)(F.x-C.x)$$

$$d_4 = -(a_4 * S.x + b_4 * S.y + c_4 * S.z)$$

---


$$e_1: a_1x + b_1y + c_1z + d_1 = 0$$

$$e_2: a_2x + b_2y + c_2z + d_2 = 0$$

$$e_4: a_4x + b_4y + c_4z + d_4 = 0$$

---


$$(a_2b_1 - a_1b_2)y + (a_2c_1 - a_1c_2)z + (a_2d_1 - a_1d_2) = 0$$

$$(a_4b_1 - a_1b_4)y + (a_4c_1 - a_1c_4)z + (a_4d_1 - a_1d_4) = 0$$

---

Substitution

$$ey + fz + g = 0$$

$$hy + iz + j = 0$$

---


$$(hf - ei)z + (hg - ej) = 0$$

$$(ie - fh)y + (ig - fj) = 0$$

$$z = (ej - hg) / (hf - ei)$$

$$y = (fj - ig) / (ie - fh)$$

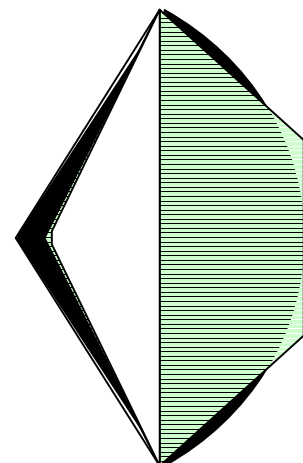
$$x = (b_4y + c_4z + d_4) / a_4$$

### 13.4.2.2. Anzahl Linien, neuer Radius

Folgende Werte sind bekannt:

R (Radius), alpha (Winkel), k (maximale Abweichung)

Daraus lassen sich folgenden Gleichungen aufstellen.



Flächengleichheit:

$$A_1 \text{ (Bogensegment)} = A_2 \text{ (Polygon-Fläche)}$$

$$A_1 = R^2 * [ ( \text{PI} * \text{alpha} / 360 ) - ( \sin(\text{alpha}/2) * \cos(\text{alpha}/2) ) ]$$

$$A_2 = r^2 * [ n * ( \sin(\text{beta}/2n) * \cos(\text{beta}/2n) - ( \sin(\text{beta}/2) * \cos(\text{beta}/2) ) ) ]$$

### Sehnengleichheit

$$2 * R * \sin(\text{alpha}/2) = 2 * r * \sin(\text{beta}/2)$$

### Abweichung

$$k < | ( r - R + ( R * \cos(\text{alpha}/2) - r * \cos(\text{beta}/2) ) ) |$$

Aus diesen Gleichungen soll nun die kleinste natürliche Zahl für n gesucht werden.

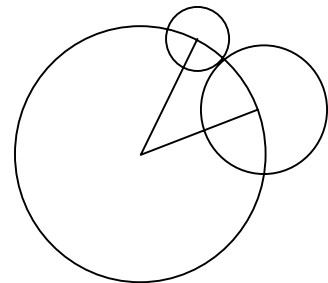
### 13.4.2.3. Neue Stützpunkte

Im diesem Teil muss der neue Bogen in die n Einzelpunkte aufgeteilt werden.

Aufgrund des Winkels kann der gewünschte Punkt berechnet werden.

Vom Startpunkt wird die Distanz ausgerechnet anhand des Radius und des Winkels.

Das Gleiche wird vom End-Punkt aus gemacht. Mit dem Start-Punkt, End-Punkt, dem Zentrum und der jeweiligen Distanz als Radius werden drei Kugeln geschaffen, die im gesuchten Punkt sich schneiden. Mit der Ebene, die durch die drei Punkte aufgespannt wird, kann die Berechnung erleichtert werden.



Die drei Punkte S (start), F (finish) und C (center) sowie der Radius R sind gegeben.

Die anderen beiden Radien müssen auf Grund des Winkels berechnet werden.

Alpha ist in dem Falle der Winkel, der sich zwischen dem Start und dem gesuchten Punkt befindet, beta zwischen diesem und dem Endpunkt.

$$r_1 = R * \sin(\text{alpha}/2)$$

$$r_2 = R * \sin(\text{beta}/2)$$

Ebene durch die drei Punkte

$$e: a_4x + b_4y + c_4z + d_4 = 0$$

Normalvektor(a,b,c)

$$a_4 = (S.y-C.y)(F.z-C.z)-(S.z-C.z)(F.y-C.y)$$

$$b_4 = (S.z-C.z)(F.x-C.x)-(S.x-C.x)(F.z-C.z)$$

$$c_4 = (S.x-C.x)(F.y-C.y)-(S.y-C.y)(F.x-C.x)$$

$$d_4 = -(a_4*S.x+b_4*S.y+c_4*S.z)$$

Kugelgleichung

$$x^2 + y^2 + z^2 + ax + by + cz + d = 0$$

$$a = -2x_0 \quad b = -2y_0 \quad c = -2z_0 \quad d = x_0^2 + y_0^2 + z_0^2 - r^2$$

für  $x_0, y_0, z_0$  jeweils die Punkte einsetzen

Die drei Kugeln

$$k1: x^2 + y^2 + z^2 + a_1x + b_1y + c_1z + d_1 = 0$$

$$k2: x^2 + y^2 + z^2 + a_2x + b_2y + c_2z + d_2 = 0$$

$$k3: x^2 + y^2 + z^2 + a_3x + b_3y + c_3z + d_3 = 0$$

$$-----$$

$$(a_1-a_2)x + (b_1-b_2)y + (c_1-c_2)z + (d_1-d_2) = 0$$

$$(a_1-a_3)x + (b_1-b_3)y + (c_1-c_3)z + (d_1-d_3) = 0$$

$$e: a_4x + b_4y + c_4z + d_4 = 0$$

$$-----$$

$$[a_4(b_1-b_2) - b_4(a_1-a_2)] y + [a_4(c_1-c_2) - c_4(a_1-a_2)] z + [a_4(d_1-d_2) - d_4(a_1-a_2)] = 0$$

$$[a_4(b_1-b_3) - b_4(a_1-a_3)] y + [a_4(c_1-c_3) - c_4(a_1-a_3)] z + [a_4(d_1-d_3) - d_4(a_1-a_3)] = 0$$

$$-----$$

Substitution

$$ey + fz + g = 0$$

$$hy + iz + j = 0$$

$$-----$$

$$hfz-eiz +hg-ej = 0$$

$$iey-fhy + ig-fj = 0$$

$$z = (ej-hg)/(hf-ei)$$

$$y = (fj-ig)/(ie-fh)$$

$$x = (b_4y + c_4z + d_4) / a_4$$

## 13.5. Anwendungsfälle

### 13.5.1. Laden

#### Hauptszenario:

Nach Angabe der Daten-Model-Dateien des INTERLIS-Formates können diese geladen werden. Mit Hilfe des INTERLIS-Compilers soll zudem beim laden die Syntax überprüft werden.

#### Alternativszenario:

Syntax-Fehler, Fehlende Dateien

Mit den angegebenen Dateien kann kein gültiges Daten-Model nach INTERLIS erzeugt werden. Die Reihenfolge der Dateien muss stimmen oder es müssen zusätzliche Dateien angegeben werden. Bei Fehlern ist eine Weiterverarbeitung nicht möglich.

### 13.5.2. Kovertieren

#### Hauptszenario:

Nachdem das INTERLIS-Daten-Model geladen wurde, wird es durch den INTERLIS-Compiler in eine interne Beschreibung umgewandelt. Danach sollen die Daten in ein internes DB-Schema abgebildet werden.

Die Abbildung von OO-Schema zu ER-Schema wird in einem separaten Kapitel beschrieben. Ebenso besteht ein Kapitel über das Konvertieren von INTERLIS Attributen zu DB Attributen.

### 13.5.3. Exportieren

#### Hauptszenario:

Mit dem DB-Schema soll nun ein SQL-Script exportiert werden. Mit Hilfe von Create Table Instruktionen soll das DB-Schema umgesetzt werden. Anhand des DB-Schemas soll es auch möglich sein die IXML Datei direkt in SQL-Inserts umzuwandeln.

## 13.6. Zusätzliche Spezifikationen

### 13.6.1. Functionality, Usability, Reliability, Performance, Supportability

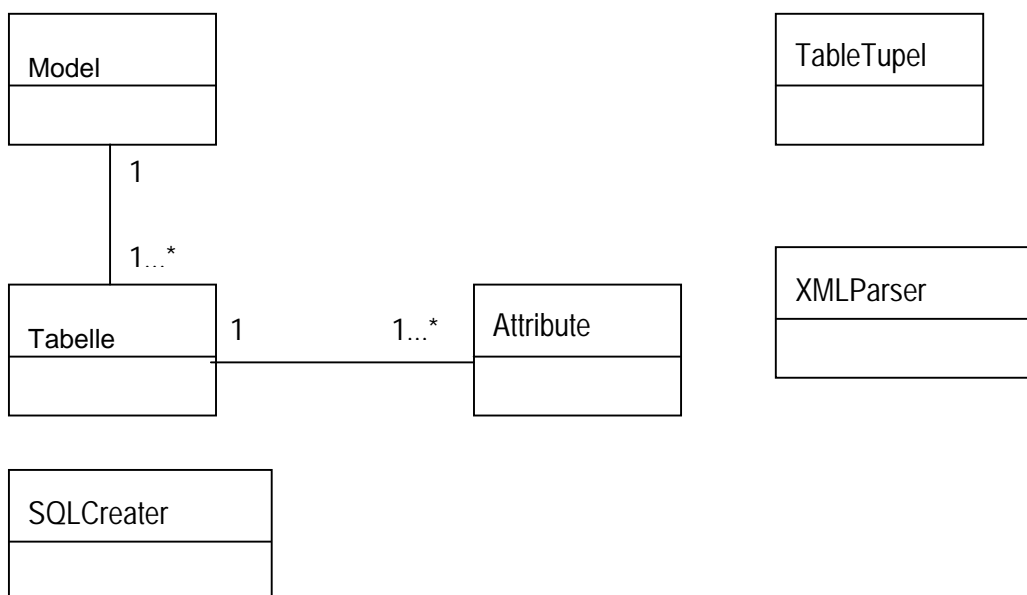
GUI: einfach und intuitive Bedienung

Plattformunabhängig als JAVA-Applikation

### 13.6.2. Implementationsrichtlinien

Das Modul soll vollumfänglich in Java erstellt werden.

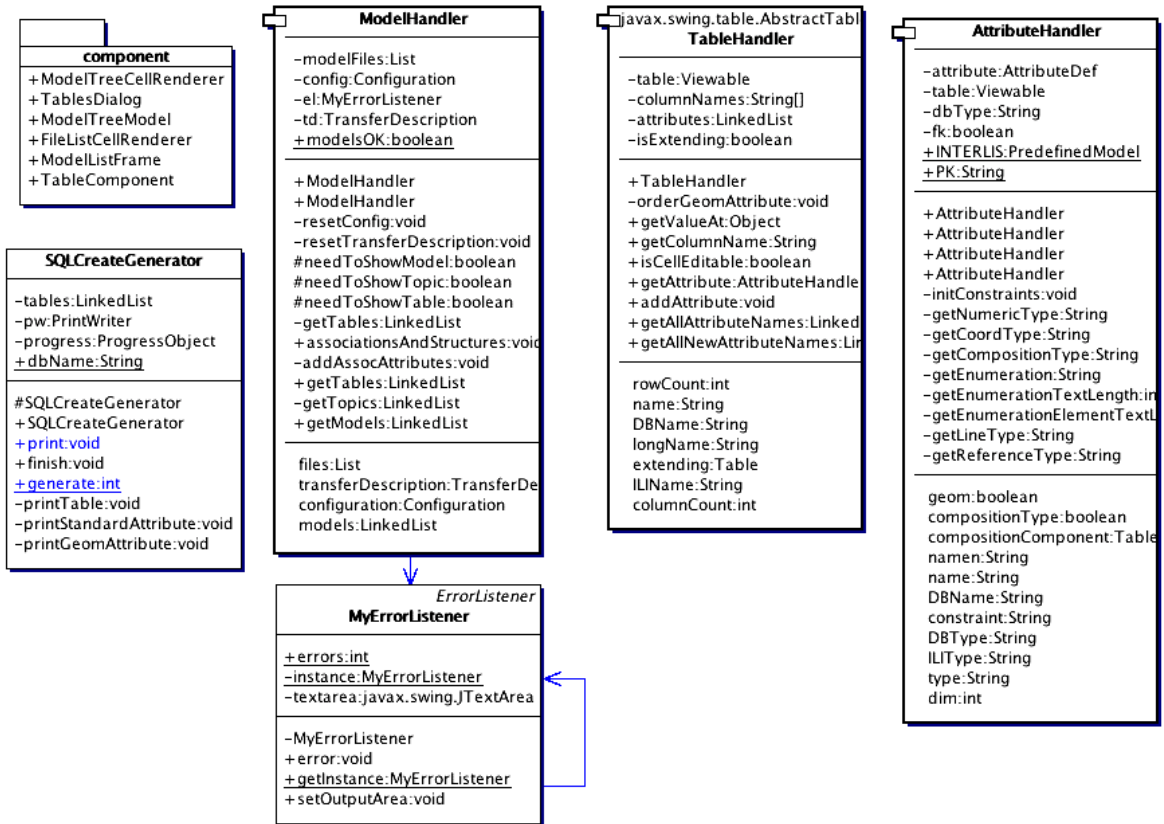
## 13.7. Domänen Modell



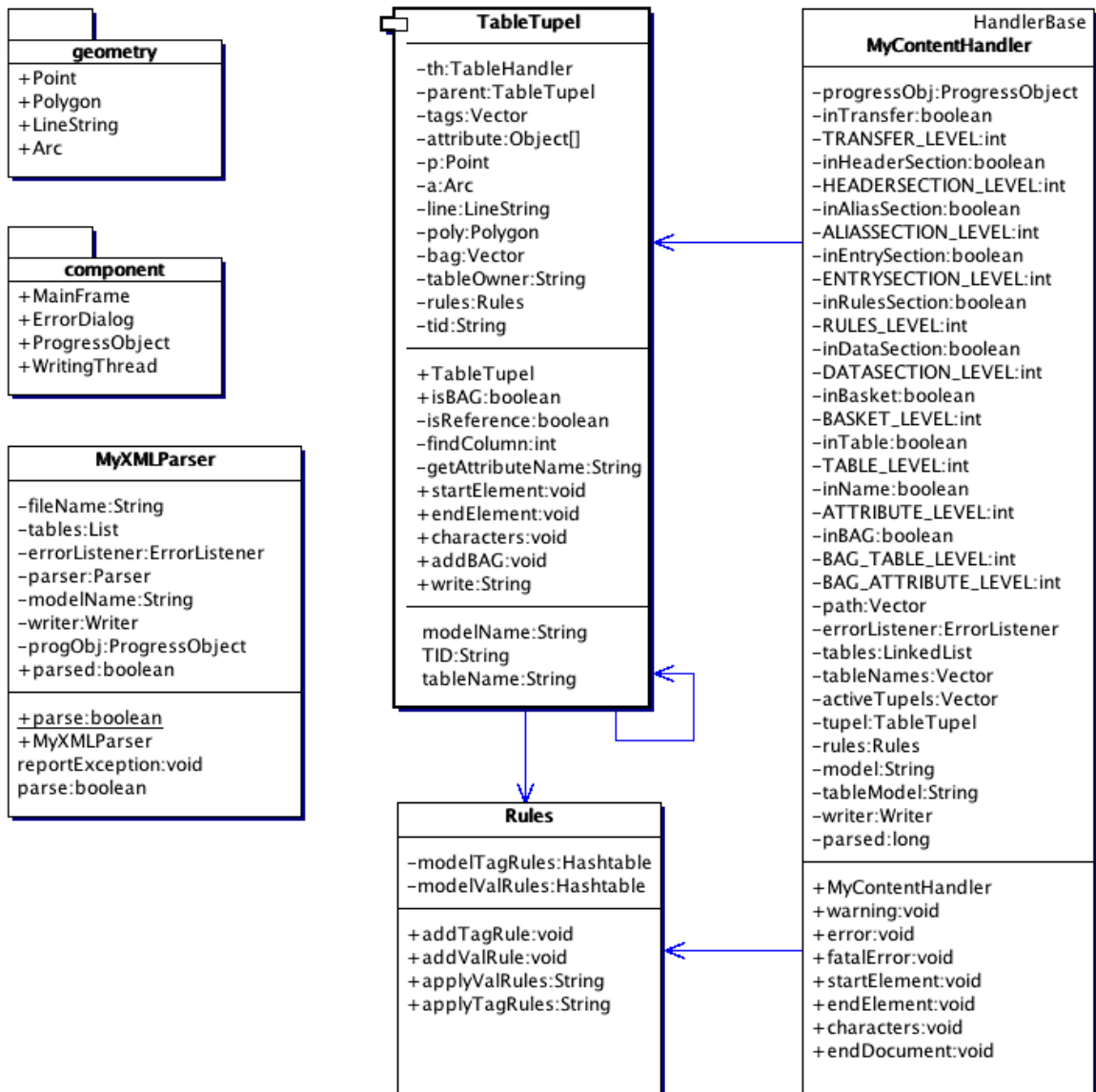
## 14. Design / Implementation

### 14.1. Klassendiagramme

### 14.1.1. Package ch.zhwin.ili2pgschema



### 14.1.2. Package ch.zhwin.ili2pgdata



## 14.2. Klassenverantwortlichkeiten

Klasse ch.zhwin.ili2pgschema.ModelHandler	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- ruft den INTERLIS-Compiler auf</li> <li>- extrahiert Modelle und Topics der Transfer-Description</li> <li>- sortiert TabellenListe nach Referenzieller Integrität</li> <li>- fügt benötigte Assoziations-Attribute und Strukturen hinzu</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Compiler Configuration</li> <li>- Transferbeschreibung</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- erstellt Liste mit DB Tabellen (TableHandler)</li> <li>- Assoziations-Fremdschlüssel (AttributeHandler)</li> </ul>
Klasse ch.zhwin.ili2pgschema.TableHandler	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- representieren einer DB Tabelle</li> <li>- ordnen der Attribute, Geometrie Attribute am Schluss</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Liste mit Attributen (AttributeHandler)</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- AttributeHandler</li> </ul>
Klasse ch.zhwin.ili2pgschema.AttributeHandler	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- umwandeln des Attribute Types (INTERLIS zu Datenbank)</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Attribute Name</li> <li>- Typ</li> <li>- Datenbank Typ</li> <li>- Datenbank Constraint</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>-</li> </ul>
Klasse ch.zhwin.ili2pgschema.SQLCreateGenerator	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- generiert aus den TableHandler Informationen die Create-Statements</li> <li>- evt. aus Transferbeschreibung zuerst noch Tabellen kreen</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Liste mit TableHandlern</li> <li>- Writer, wo soll es hingeschrieben werden</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>-</li> </ul>
Klasse ch.zhwin.ili2pgschema.MyErrorListener	

<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- bekommt vom INTERLIS Compiler Fehlermeldungen</li> <li>- Singleton</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- zählt die Fehler</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>-</li> </ul>
--	--

Package ch.zhwin.ili2pgschema.component	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- GUI Komponenten bereitstellen</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- andere Komponenten</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- verschiedene andere GUI Komponenten</li> </ul>

Klasse ch.zhwin.ili2pgdata.TableTupel	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- verarbeiten von weitergeleiteten XML Informationen</li> <li>- spezielle Verarbeitung von Geometrie Objekten</li> <li>- zusammensetzen der Informationen zu einem String</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Attribute Werte</li> <li>- TableHandler als Tabellen-Schema</li> <li>- SuperTabelle</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- Geometrie Objekte</li> </ul>

Klasse ch.zhwin.ili2pgdata.Rule	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- entgegennehmen von XML Regeln</li> <li>- anwenden der Regeln</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- VAL Regeln</li> <li>- TAG Regeln</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>-</li> </ul>

Klasse ch.zhwin.ili2pgdata.MyXMLParser	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- initialisieren des XML Parsers mit den nötigen Informationen</li> <li>- starten des Verarbeitens</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- XML File</li> <li>- DB Schema</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- SAX Parser</li> </ul>

Klasse ch.zhwin.ili2pgdata.MyContentHandler	

<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- verarbeiten der einzelnen SAX Events</li> <li>- initialisieren der Regeln</li> <li>- weiterleiten an das aktuelle TableTupel</li> <li>- schreiben des Tupels</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- aktuelles Tupel (TableTupel)</li> <li>- regeln (Rules)</li> <li>- Writer</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- SAX Parser</li> </ul>
--	--

Package ch.zhwin.ili2pgdata.geometry	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- bereitstellen von verschiedenen Geometrie Objekten</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Point</li> <li>- Arc</li> <li>- LineString</li> <li>- Polygon</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>-</li> </ul>

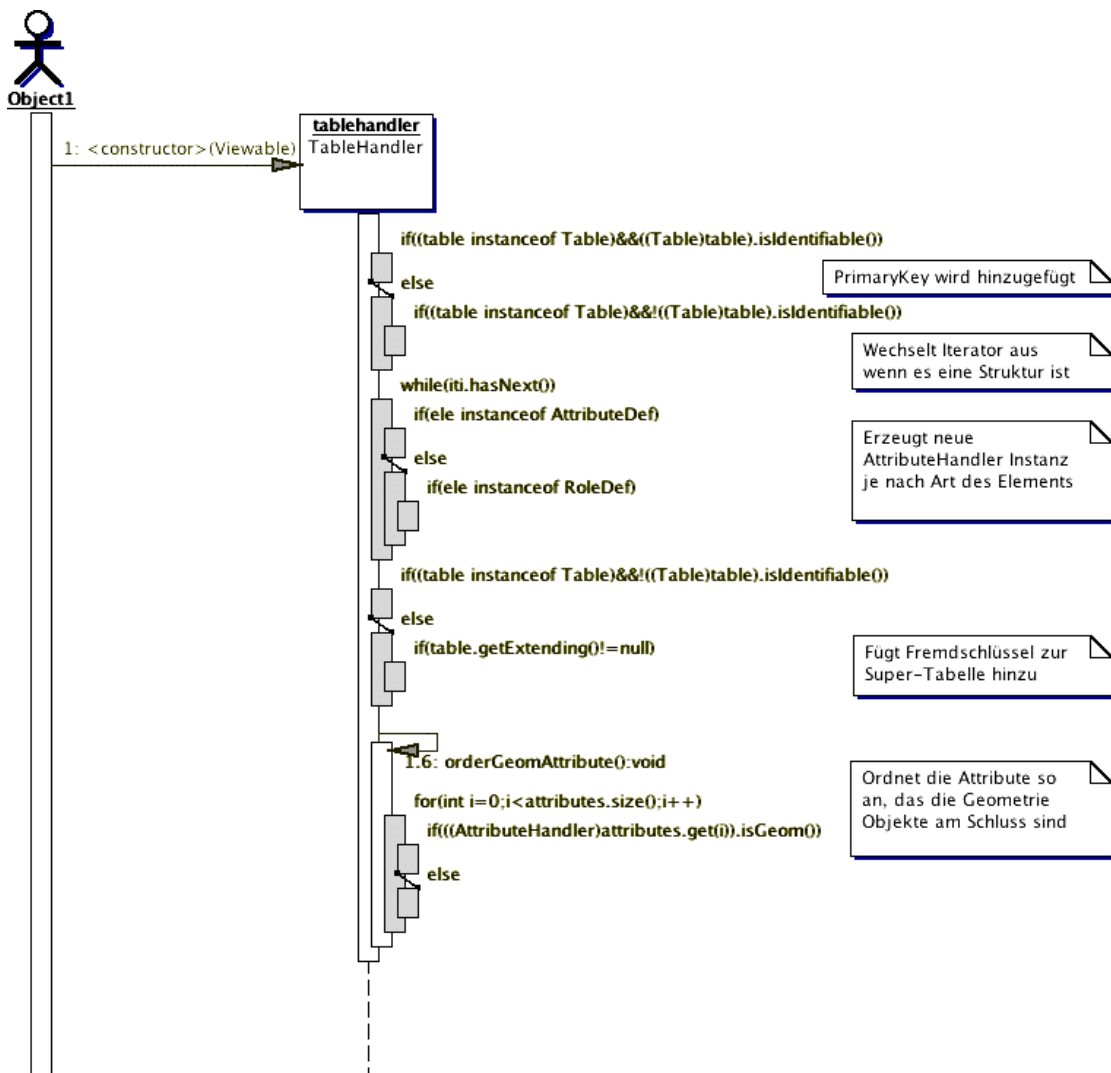
Package ch.zhwin.ili2pgdata.component	
<b>Verantwortlichkeit</b> <ul style="list-style-type: none"> <li>- GUI Komponenten bereitstellen</li> <li>- Main Klasse (MainFrame)</li> </ul>	<b>Information</b> <ul style="list-style-type: none"> <li>- Zwischenresultate die weitergeleitet werden können</li> </ul> <b>Creator</b> <ul style="list-style-type: none"> <li>- verschiedene andere GUI Komponenten</li> </ul>

## 14.3. System Sequenz Diagramm

System Sequenz Diagramm (generiert und editiert)

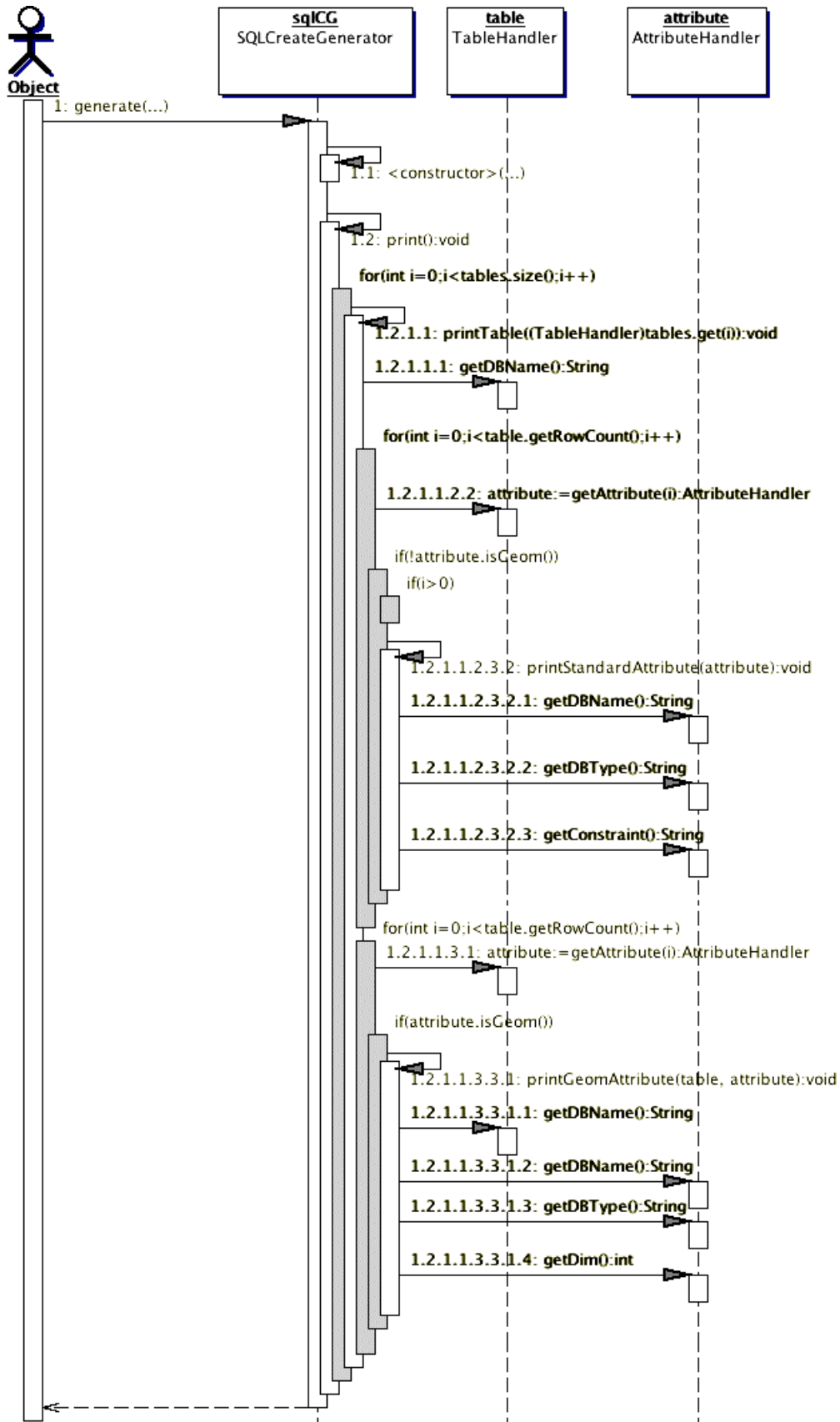
### 14.3.1. Konstruktor TableHandler

Je nach Instanz des Attributes muss ein anderer Konstruktor des AttributeHandler aufgerufen werden.  
Unter gewissen Umständen ist es auch notwendig, gewisse Attribute direkt hinzuzufügen



Es ist aber auch nach dem Konstruktor noch möglich, Fremdschlüssel von Assoziationen oder „Bag-Ownern“ hinzuzufügen

### 14.3.2. SQL Create Generator generate



## 15. Schlussbericht

### 15.1. Funktionsweise

Mit den geladenen Modell Dateien wird direkt eine Transferbeschreibung des INTERLIS-Compilers erzeugt. Diese wird im ModelHandler auf Tabellen untersucht und in einzelne TableHandler abgepackt. Die einzelnen TableHandler erzeugen dann wiederum die benötigten AttributeHandler. Vom ModelHandler können nachträglich noch direkt initialisierte Attribute hinzugefügt werden. Somit erhält man eine Liste mit den Tabellen des DB Schemas.

Der SQLCreateGenerator nimmt diese Liste und arbeitet diese der Reihe nach ab.

Für die Insert-Statements wird ebenfalls das DB-Schema genommen und bei einer gewissen Struktur-tiefe die Namen mit der Tabellenliste verglichen. Bei einer Übereinstimmung wird ein TableTupel mit der entsprechenden Beschreibung, dem TableHandler, initialisiert. Alle weiteren Tags werden dem Table-Tupel weitergeleitet. Wird das Ende des Tupels erreicht, wird dieser sofort geschrieben.

### 15.2. Ausbaumöglichkeiten

Eine Ausbaumöglichkeit ist sicherlich das Konvertieren der Graphikbeschreibungen von INTERLIS, sowie der Symbolbibliotheken, für den MapServer. Diese müssen im Moment noch von Hand selber geschrieben werden. Eine andere Möglichkeit wäre mit der PostGreSQL Datenbank direkt über JDBC zu kommunizieren.

### 15.3. Probleme

Beim Laden der konvertierten Datensätze kann es bei fehlerhaften Referenzen zu Verletzungen der referenziellen Integrität kommen. Deshalb ist schon bei den XML-Datensätzen auf diese zu achten. Bei Polygonen muss zudem der Startpunkt mit dem Endpunkt übereinstimmen. Ebenfalls kam es bei Polygonen mit nur 2 Punkten zu Problemen. Diese wurden gelöst, indem ein 3. Punkt in der Mitte berechnet wurde.

### 15.4. Code

Der Programm Code und ein vollständiges Javadoc befinden sich auf der beigelegten CD.

## 15.5. Testbericht

Die Generierung von Beispiel Scripten verlief positiv. Alle Scripte erfüllten die nötigen Konventionen. Leider konnte mangels korrekten Datensätzen bis heute keine Daten geladen werden. Grund dafür ist der relative neue Standard INTERLIS 2.2, der als Referenzhandbuch erst Ende September herausgekommen ist.

# Anhang

## A Glossary

BBOX	Bounding Box: Koordinaten eines Bildausschnittes einer Karte.
CGI	Common Gateway Interface - eine standardisierte Methode mit der ein Webserver eine Useranfrage an ein Programm schicken kann um die Antwort des Programms wiederum zurück an den anfragenden Client zu senden. Das Programm kann in jeder beliebigen Sprache programmiert werden - es muss allerdings auf dem Server ausführbar sein.
Clip	Speichert alle Massen und Koordinaten eines Bildausschnittes.
EPSG	European Petroleum Survey Group, Arbeitsgruppe für Erdvermessung.
ER Schema	Entity Relation Scheme, Datenbank Schema
ESRI	ESRI ist eine Firma die seit 1969 GeoDaten verarbeitet. Legte den Standard für ShapeFiles.
Grosses Bild	Dieses Bild wird immer im Hintergrund geladen. Es ist maximal dreimal so breit und dreimal so hoch wie der sichtbare Ausschnitt.
GUI	Graphical User Interface - die graphische Benutzeroberfläche eines Betriebssystems. Unter Windows setzt diese sich aus dem Desktop, der Taskbar sowie den verschiedenen Kontroll-Elementen wie Dialogboxen und Menüs zusammen.
History	Speichert die letzten zehn Bilder, damit sie ohne Zugriff auf dem MapServer dargestellt werden können.
HTTP	Hyper Text Transfer Protocol - HTTP ist ein Anwendungsprotokoll das mit TCP/IP verwandt ist und sich um den Transport der von einem Webserver gelieferten Daten kümmert.
IXML	INTERLIS XML, Die Datensätze werden in XML Codiert.
Kleines Bild	Dieses Bild wird entweder direkt angefordert, oder aus dem Verschieben des grossen Bildes ermittelt.
Layer	Der Layer ist eine graphische Ebene mit speziellen Informationen. Z.B. Layer mit Städten, oder Layer mit Strassen, usw..
MapFile	Das Mapfile ist die zentrale Layoutdatei des MapServer. Sie beschreibt Inhalt und Darstellung der fertigen Karte und enthält zusätzliche Informationen, um Massstäbe, Legenden und Referenzkarten passend zur fertigen Karte zu erstellen.
MapServer	Ein MapServer ist ein Programm, das Karten nach bestimmten Anforderungen erstellt und diese dann ausliefert. Der klassische Ansatz der auch vom UMN MapServer verfolgt wird, ist ein Programm in einem Webserver laufen zu lassen, dass über URL-Parameter angesprochen werden kann.
OGC OpenGISConsortium	Ist ein internationaler Zusammenschluss von über 220 Unternehmen. Ziel des OGC ist es, gemeinsame Standards (Protokolle, Formate etc.) zur Verarbeitung von Geo-Daten zu erarbeiten.
OO Schema	Objekt Orientiertes Schema
PDF	Portable Document Format In einer PDF Datei sind alle Teile eines zu druckenden Dokumentes bereits vorhanden. Diese Dateien werden mit Produkten aus dem Hause Adobe erzeugt, und lassen sich mit einem Adobe Acrobat Reader betrachten.
PostGIS	PostGIS ist eine Erweiterung von PostgreSQL, deren wichtigster Beitrag wohl die Implementierung eines grossen Teils der Simple Features des OGC ist.
PostgreSQL	PostgreSQL ist eine freie SQL-Datenbank. Insbesondere ist es aber eine schnelle, hochgradig konfigurierbare Datenbank, die dem Vergleich mit

	den Grossen des kommerziellen Geschäfts gelassen entgegen sehen kann
SRS	Spatial Reference System. Bezeichnet ein horizontales Koordinatensystem.
TCP/IP	Transfer Control Protocol - ein Teil der TCP/IP Protokollgruppe. Praktisch alle Netzwerkverbindungen im Internet laufen letztlich mit Hilfe von TCP/IP ab. Das Transmission Control Protocol (TCP) ist ein Protokoll, das gewährleisten soll, dass die Daten auch wirklich am Zielort ankommen.
Template File	Bei den Templates handelt es sich um HTML-Dateien, welche direkt in einem Browser dargestellt werden können.
UML	Abkürzung für "Unified Modeling Language" UML ist eine visuelle Modellierungssprache. Sie gibt die Notation für Diagramme zur Modellierung an. Richtig "schön" wird UML mit geeigneten Tools. Diese helfen bei der Erstellung der Diagramme, halten sie jederzeit konsistent, generieren die Projektdokumentation und am Ende den Programmcode.
URL	Universal Resource Locator - eine URL ist eine allgemeingültige Adresse einer Resource (zum Beispiel einer Datei) im Internet. Diese eindeutige Adresse besteht aus der Bezeichnung des Protokolles (z.b. http://), aus dem Namen des Servers auf dem die Resource zu finden ist (z.b. www.zhwin.ch), aus dem Namen des Dienstes der die Resource zur Verfügung stellt (z.b. www) sowie aus dem eigentlichen Namen der Resource.
Webserver	Ein Webserver ist ein Server Programm das eine Webseite betreibt: Es liefert auf Anfrage eines Clients (meist eines Browsers) die gewünschte Webseite.
WMS-Standard	Ist die OpenGIS Web Map Server Interfaces Implementation Spezifikation, kurz WMS-Standard genannt.
WWW	Das World Wide Web ist die Gesamtheit der Rechner im Internet, die über HTTP mit Hypertext-Verknüpfungen vernetzt sind. Es existiert seit 1993 und machte das Internet erst populär.
XML	eXtensible Markup Language - eine Sprache zur Beschreibung von Inhalten und Strukturen. Das extensible meint dabei, dass die Sprache um Elemente erweiterbar ist, sofern diese Elemente anhand einer vorgegebenen Art und Weise definiert werden. Der Nachfolger von HTML wird in XML implementiert werden.

## B Bedienungsanleitungen


### B.1 UGE

#### Starten des universellen Graphikeditors

Das Starten des universellen Graphikeditors erfolgt, indem auf das Symbol „UGE“ geklickt wird.


#### Download

Bevor ein Plan heruntergeladen werden kann, muss eine Internet Verbindung aufgebaut werden.

- Ist die Internet Verbindung aufgebaut, und das UGE Programm gestartet, klicken Sie auf das Symbol „Download“  , welches sich in der oberen Menüleiste befindet.
- Um die Verbindung mit dem Server aufzubauen, müssen Sie auf den Knopf „Verbinden“ klicken.
- Markieren Sie die Datei, die Sie herunterladen wollen und klicken Sie auf den Knopf „Download“.
- Geben Sie den Pfad ein, in welchem die Datei gespeichert werden soll und deren Name.
- Das Fenster kann nun geschlossen werden.

#### Datei öffnen

Bevor eine Datei angeschaut werden kann, muss sie geöffnet werden.


- Öffnen Sie die Datei, indem Sie auf das Symbol „Öffnen“  klicken.
- Suchen Sie die gespeicherte Datei und klicken Sie auf „Öffnen“.
- Das Darstellen der Datei kann länger dauern.

#### Übersicht Fenster

Das „Übersicht“ Fenster zeigt, durch einen roten Rahmen, die aktuelle Position des ersichtlichen Ausschnittes der Original Datei an.



- Das „Übersicht“ Fenster kann unter „Ansicht“ aktiviert oder deaktiviert werden.
- Ein Ausschnitt kann im „Übersicht“ Fenster ausgewählt werden, indem mit gedruckter linker Maustaste über den gewünschten Ausschnitt gefahren wird.

#### Ausschnitt Verschieben

- Das Verschieben einer Datei geschieht, indem Sie auf das Symbol „Hand“  in der unteren Menüleiste klicken.
- Mit gedruckter linker Maustaste können Sie nun die Datei verschieben.

## Zoomen

### Variante 1:

- Klicken Sie das Symbol „Vergrössern“  oder „Verkleinern“  an.
- Wählen Sie den Ausschnitt mit gedruckter linker Maustaste, den Sie anschauen wollen.

### Variante 2:


- Geben Sie den Zoomfaktor in das entsprechende Feld in der unteren Menüleiste ein.
- Klicken Sie auf „Enter“.

### Variante 3:

- Aktivieren Sie das „Übersicht“ Fenster.
- Wählen Sie den Ausschnitt, den Sie anschauen wollen, aus.


## Element einfügen


In der oberen Menüleiste können Elemente wie Linie, Kreis, Oval, Rechteck, Freihand-Linie und Text ausgewählt werden.

- Die Dicke des Randes, die Hintergrund- und die Rahmenfarbe der Elemente können in der oberen Menüleiste eingestellt werden.
- Die Position dieser Elemente wird bestimmt, indem auf die Datei an der gewünschten Stelle geklickt wird. Die Grösse der Elemente wird nach dem Loslassen der Maustaste bestimmt.
- Nachdem Sie ein neues Element eingezeichnet haben, klicken Sie auf das Symbol „Auswahl“  .


## Attribute ändern

Elemente können auch noch im Nachhinein verändert werden. Die Hintergrund-, und die Vordergrundfarbe, die Position sowie die Linienstärke können angepasst werden.



- Klicken Sie auf das Symbol „Auswahl“  .
- Wählen Sie mit der Maus das Element, das Sie ändern wollen.

- Klicken Sie auf das Symbol „Attribute“ .
- Nun können Sie das Element anpassen.

### Element löschen


- Ein Element kann gelöscht werden, indem es im Fenster „Element“ markiert und anschliessend auf dem Papierkorb geklickt wird.
- Ein Element kann ebenfalls gelöscht werden indem auf das Symbol „Pfeil“ geklickt wird.
- Das Element anklicken, das gelöscht werden soll.
- Das Symbol „Entfernen“  anklicken.

### Reihenfolge ändern


Die Reihenfolge der Elemente kann angepasst werden, indem auf das Symbol „Pfeil“  oder „Pfeil“  im Fenster „Übersicht“ geklickt wird. Dadurch können Elemente vor oder hinter andere gestellt werden.

### Element verschieben

Jedes Element kann mit der Maus verschoben werden.




- Klicken Sie das Symbol „Auswahl“  in der oberen Menüleiste.
- Klicken Sie auf das Element, das Sie verschieben wollen.
- Verschieben Sie das Element mit gedruckter linker Maustaste.

### Grösse der Elemente anpassen


- Klicken Sie auf das Symbol „Auswahl“ .
- Klicken Sie auf das Element, das Sie anpassen wollen.
- Passen Sie das Element an, indem Sie es an der oberen oder unteren blauen Ecke mit der Maus verschieben.

### Element Kopieren


Mit der Funktion „Markieren“ kann ein Ausschnitt aus einer Datei an einen beliebigen Ort wieder eingefügt werden.

- Klicken Sie auf dem Knopf „markieren“  .
- Wählen Sie das Element, dass Sie kopieren wollen mit der Maus aus.
- Klicken Sie auf das Symbol „Kopieren“  .
- Klicken Sie auf das Symbol „Einfügen“  . Die Kopie wird neben dem Original Element eingefügt.
- Die Kopie kann mit der Maus an eine beliebige Stelle verschoben werden.

### Bilder einfügen

- Drücken Sie auf das Symbol „Bild“  .
- Sie können das Bild auswählen, welches Sie einfügen wollen.
- Klicken Sie auf die Stelle, wo das Bild eingefügt werden soll.

### Text einfügen

- Klicken Sie auf das Symbol „Text“  .
- Markieren Sie die Stelle, an welchem der Text eingefügt werden soll.
- Schreiben Sie den Text im Textfeld.

### Stempel anschauen

Änderungen an Dateien werden mit einem Stempel markiert. Auf dem Stempel stehen wichtige Daten, um eine Änderung eindeutig zu markieren, wie das Datum der Änderung, der Name des Projektes oder der Name des Benutzers.

- In der oberen Menüleiste können Sie unter Datei auf „Eigenschaften“ klicken, um den Stempel anzuschauen und eventuell zu ändern.
- Klicken Sie auf „OK“ um die Änderung zu speichern.


### Speichern

- Sie müssen auf das Symbol „Save“  klicken.

- Es öffnet sich ein Fenster, in welchem der Ort der Speicherung und der Name der zu speichernden Datei angegeben werden muss.
- Die Änderung wird als GIF Datei gespeichert.

## Upload

Bevor eine Änderung auf dem Server gespeichert werden kann, muss die Internet Verbindung aufgebaut werden.

- Bauen Sie die Internet Verbindung auf.
- Klicken Sie auf das Symbol „Upload“ . Es öffnet sich ein Fenster in welchem die Parameter der Verbindung eingestellt sind.
- Klicken Sie auf „Verbinden“.
- Nun können Sie „Upload“ anklicken. Es öffnet sich ein Fenster, in welchem Sie die Datei auswählen, welche gesendet werden soll.
- Zuletzt klicken Sie auf „Schliessen“.

## B.2 UGE Map Plugin

### Starten des universellen Graphikeditors

Das Starten des universellen Graphikeditors erfolgt, indem auf das Symbol „UGE“ geklickt wird.

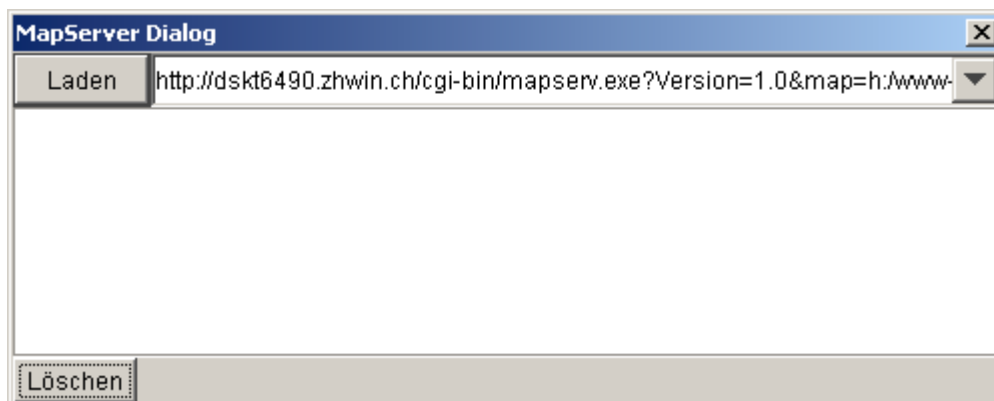
### MapServerPlugin Starten

Klicken Sie auf das Symbol „Öffnen“. Ein Fenster öffnet sich, in welchem Sie das Symbol Map-Document wählen müssen.



### Verbindung aufbauen

- Geben Sie im Textfeld des Fensters MapServer Dialog die URL des MapServers ein. Sie können aber auch aus der Liste der MapServer eine Adresse auswählen. Dafür müssen Sie auf das Symbol „Pfeil“ klicken, damit die Liste sichtbar wird.
- Klicken Sie auf den Button „Laden“ damit eine Verbindung aufgebaut werden kann.
- Falls der MapServer nicht gefunden wird, erscheint eine Fehlermeldung.

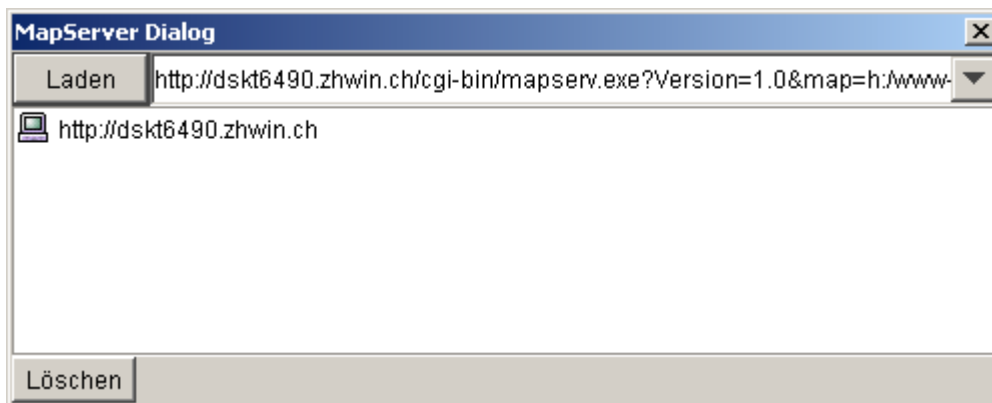


Bsp:

<http://dskt6490.zhwin.ch/cgi-bin/mapserv.exe?Version=1.0&map=h:/www-root/itasca/demo.map>

### Löschen einer Verbindung

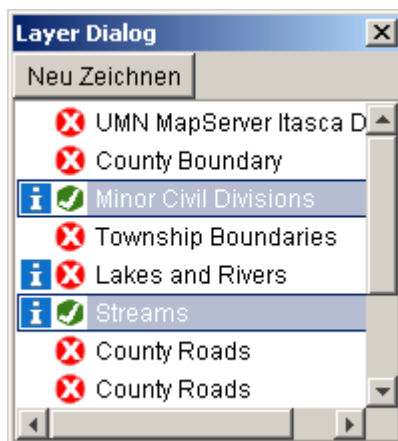
- Klicken Sie auf die Server Verbindung, welche gelöscht werden soll.
- Klicken Sie auf den Button „Löschen“.



### Layer Wahl

- Klicken Sie auf das Layer, welches Sie anschauen wollen.
- Klicken Sie auf den Button „Neu Zeichnen“ damit der neue Layer geholt werden kann.

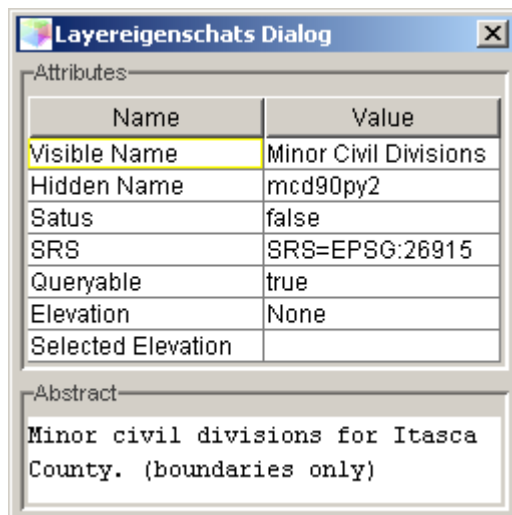
Das Symbol  bedeutet, dass Sie zu diesem Layer zusätzliche Informationen verlangen können.



### Layerparameter und Höhe bestimmen

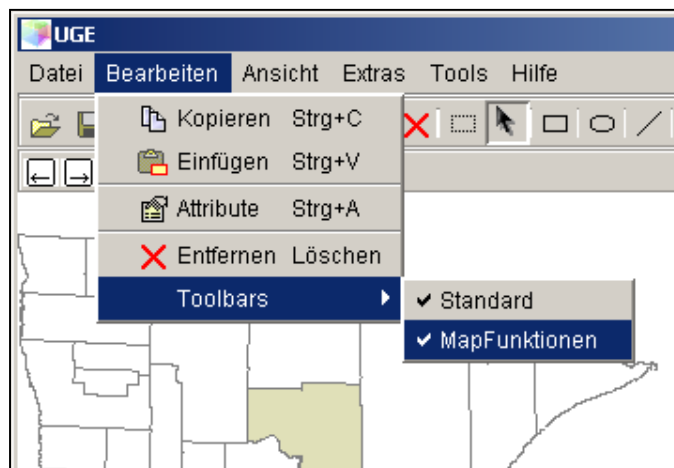
- Führen Sie einen Doppelklick auf den entsprechenden Layer im Layerdialogfenster aus.
- Ein Fenster öffnet sich mit den Parametern des Layers.

- Falls der Layer aus Sublayern besteht, können Sie im Textfeld „Selected Elevation“ die Höhe eintragen, die Sie interessiert.
- Nachdem Sie auf „Enter“ gedrückt haben, wird die Höhe gesetzt.




### Toolbar MapServer aktivieren/deaktivieren

- Wählen Sie unter Bearbeiten Toolbars „MapFunktionen“ aus.
- So können Sie den Toolbar ein- oder ausschalten.



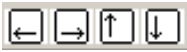
### Infos anfordern

- Aktivieren Sie den Layer, welcher Information anbietet.
- Klicken Sie auf das Symbol  in der oberen Toolbarleiste.

- Klicken Sie auf die Stelle der Graphik, die Sie interessiert.
- Ein Info-Fenster öffnet sich. Bevor Sie weiterarbeiten, müssen Sie das Fenster schliessen.

## Graphik Verschieben

### Variante 1

- Sie können das Bild in vier Richtungen verschieben (nach links, rechts, oben und unten) indem Sie auf das entsprechende Symbol drücken. 

### Variante 2

- Klicken Sie auf das Symbol „Hand“ in der unteren Menüleiste.
- Mit gedrückter linker Maustaste können Sie nun den Plan verschieben.

## Zoomen

### Variante 1

- Klicken Sie auf das Symbol „+“ oder „-“ in der unteren Menüleiste.
- Markieren Sie die Stelle die Sie vergrössern oder verkleinern wollen.

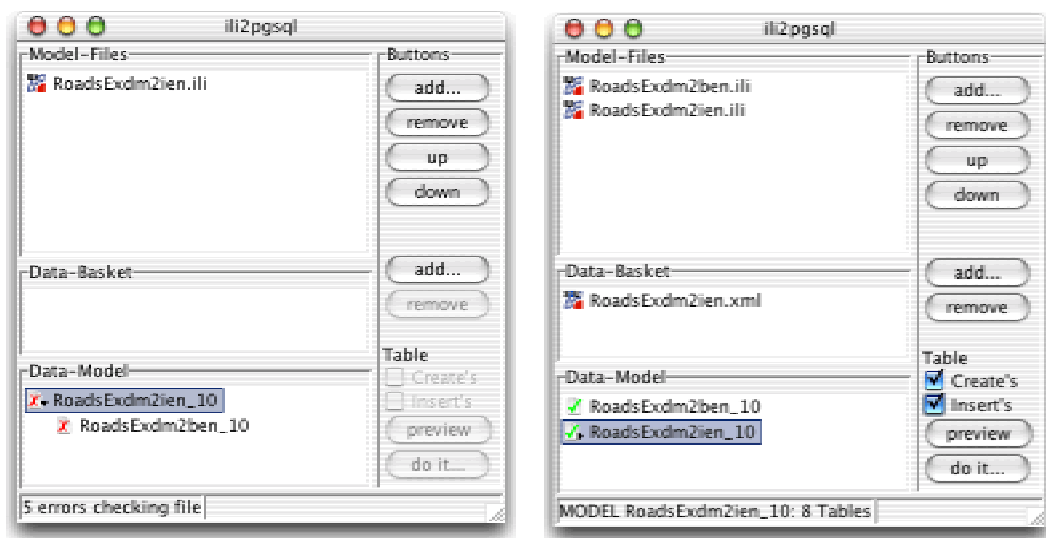
## Home

- Klicken Sie auf das Symbol „Home“. 
- Das erste Bild wird angezeigt. Alle anderen Bilder der History werden gelöscht.

## History

- Klicken Sie auf das Symbol „Zurück“  oder „Vorwärts“  in der oberen Menüleiste.
- Das Bild wird sofort dargestellt.

## B.3 Bedienungsanleitung ili2pgsql



### Installation

Damit das ili2pgsql Tool richtig funktioniert muss es im gleichen Verzeichnis wie der INTERLIS Compiler (ili2c.jar) sein. Der Name des Compilers muss ili2c.jar sein, damit dieser benutzt werden kann.

### Starten des ili2pgsql

Falls das Betriebssystem Jar Dateien starten kann, wird durch einen Doppelklick das GUI gestartet. Ansonsten kann es über die Konsole mit dem Befehl :

```
java -jar ili2pgsql.jar oder javaw -jar ili2pgsql.jar
```

gestartet werden.

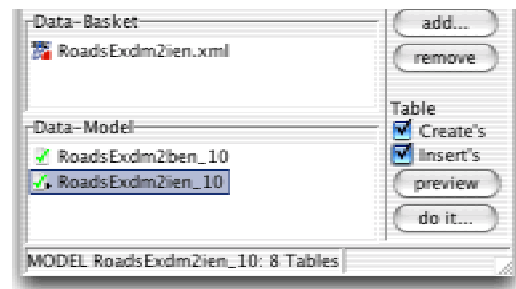
### Laden von Modell Dateien

Durch drücken auf den oberen add-Button wird ein Dateimanager eingeblendet, aus dem die gewünschten Dateien ausgewählt werden können.

### Auswählen eines Modells

Sind die Modell-Dateien ohne Fehler beim INTERLIS-Compiler durchgekommen, erscheinen die definierten Modelle im Data-Model bereich.

Dort kann dann das gewünschte Model ausgewählt werden. Erst durch selektieren eines Modells ist die Weiterverarbeitung möglich.



### Laden von IXML Datensätzen

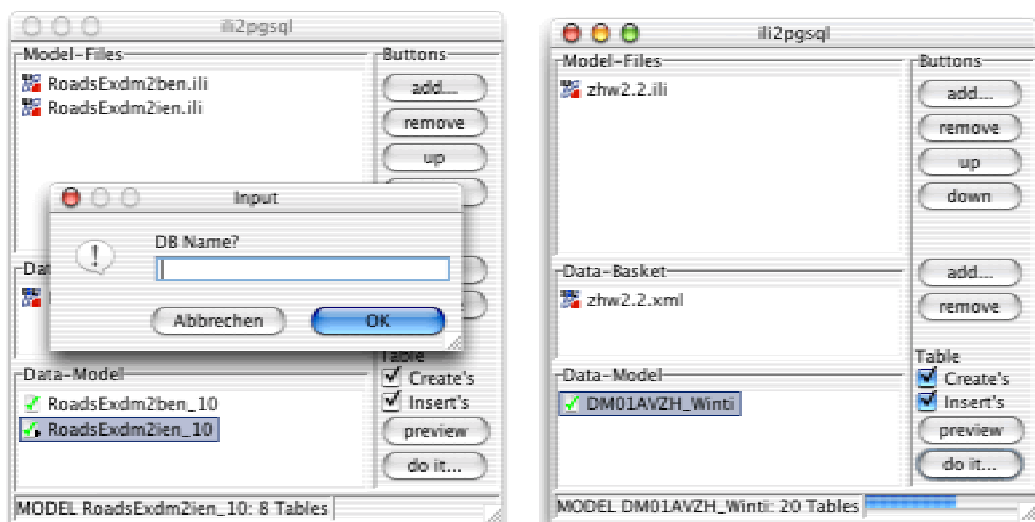
Durch Drücken auf den unteren „add“-Button wird ein Dateimanager eingeblendet, aus dem die gewünschten Dateien ausgewählt werden können.

### Vorschau der Tabellen

Durch Drücken auf den „preview“ Button kann man die Tabellen anschauen. Dort ist auch ersichtlich, wie die INTERLIS Attribute auf DB Attribute abgebildet werden.

### Speichern des SQL Scriptes

Mit dem Button „do it...“ wird das Erzeugen des Scriptes eingeleitet. Zuerst wird man aufgefordert, den Dateinamen des Scriptes anzugeben. Danach folgt die Aufforderung, den Namen der Datenbank einzutragen. Danach beginnt das Tool mit seiner Arbeit.



## C MapServer

### C.1 Template File:

```
<!--MapServer version 3.6.1 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP SUP-
PORTS=PROJ SUPPORTS=TTF SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT IN-
PUT=EPPL7 INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE ->
```

```
<html>
```

```
<head><title>MapServer Demo Interface</title></head>
```

```
<body bgcolor=#FFFFFF>
```

```
<center><h1>MapServer Demo Interface</h1></center>
```

```
<hr>
```

```
<form method=GET action="/cgi-bin/mapserv.exe">
```

```
<center>
```

```
<table border=0 cellspacing=0 cellpadding=4 bgcolor="#000000">
```

```
<tr>
```

```
<td valign="top" align=center>
```

```
<table cellpadding="0" cellspacing="0" border="0">
```

```
<tr><td colspan="2"><INPUT NAME="img" TYPE="image" SRC="/tmp/DEMO10345828853472.gif"
width=600 height=600 border=0></td></tr>
```

```
<tr>
```

```
<td>&nbsp;<font size=-1 face="arial,Helvetica" color="#FFFFFF"><b>Powered by MapS-
erver</b></font></td>
```

```
<td align="right"></td>
```

```
</tr>
```

```
</table>
```

```
</td>
```

```
<td valign="top" bgcolor=#ffffff>
```

```
<table cellpadding="5" cellspacing="0" border="0" bgcolor="#ffffff">
```

```
<tr><td>
```

```
<center><input type="submit" value="Refresh/Query"></center>
```

```
<p>
```

```
<input type="radio" name="mode" value="browse" checked> <b>Browse map</b><br>
```

```
<input type="radio" name="mode" value="query"> <b>Query feature</b><br>
```

```
<input type="radio" name="mode" value="nquery"> <b>Query multiple features</b>
```

```
<hr>
```

```

<p>
<b>Select Layers to Display: </b><br>
<select multiple name="layer" size=3>
  <option value="airports" > Airports
  <option value="cities" > Cities
  <option value="lakespy2" selected> Lakes & Rivers
  <option value="dlgstln2" selected> Streams
  <option value="roads" > Roads
  <option value="twprgpy3" > Townships
</select>

<p>
Zoom In <input type=radio name=zoomdir value=1 >
Pan <input type=radio name=zoomdir value=0 checked>
Zoom Out <input type=radio name=zoomdir value=-1 >
<p>
Zoom Size <input type=text name=zoomsize size=4 value=2>
<p>

<font size=+1><b>Legend</b></font><br><br>
<br>

<p>
<center><INPUT NAME="ref" TYPE="image" SRC="/tmp/DEMOref10345828853472.gif" bor-
der="0"></center>
</td></tr></table>

</td></tr>
</table>
</center>

<input type="hidden" name="imgxy" value="299.5 299.5">
<input type="hidden" name="imgext" value="388107.634400 5200301.166444 500896.339020
5313063.023754">
<input type="hidden" name="map" value="h:/www-root/itasca/demo.map">
<input type="hidden" name="savequery" value="true">

<input type="hidden" name="program" value="/cgi-bin/mapserv.exe">
<input type="hidden" name="map_web_imagepath" value="h:/www-root/tmp/">
<input type="hidden" name="map_web_imageurl" value="/tmp/">

```

```
</form>
```

```
<p><hr><p>
```

```
</body></html>
```

## C.2 XML File:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities (View Source for full doctype...)>
<!--
end of DOCTYPE declaration
```

```
<WMT_MS_Capabilities version="1.0.0" updateSequence="0">
<Service>
- <!--
a service IS a MapServer mapfile
→
<Name>GetMap</Name>
<!--
WMT defined
```

```
<Title>UMN MapServer Itasca Demo</Title>
<Abstract>This is the UMN MapServer demonstration application for Itasca
County located in north central Minnesota.</Abstract>
<OnlineResource>http://localhost/itasca/demo_init.html</OnlineResource>
<AccessConstraints>none</AccessConstraints>
</Service>
<Capability>
<Request>
<Map>
<Format>
<GIF />
<PNG />
<JPEG />
<WBMP />
</Format>
<DCPType>
```

```
<http>
  <Get onlineResource="http://localhost/itasca/demo_init.html" />
  <Post onlineResource="http://localhost/itasca/demo_init.html" />
</http>
</DCPType>
</Map>
<Capabilities>
<Format>
  <WMS_XML />
</Format>
<DCPType>
<http>
  <Get onlineResource="http://localhost/itasca/demo_init.html" />
  <Post onlineResource="http://localhost/itasca/demo_init.html" />
</http>
</DCPType>
</Capabilities>
<FeatureInfo>
<Format>
  <MIME />
  <GML.1 />
</Format>
<DCPType>
<http>
  <Get onlineResource="http://localhost/itasca/demo_init.html" />
  <Post onlineResource="http://localhost/itasca/demo_init.html" />
</http>
</DCPType>
</FeatureInfo>
</Request>
<Exception>
<Format>
  <BLANK />
  <INIMAGE />
  <WMS_XML />
</Format>
</Exception>
<VendorSpecificCapabilities />
```

```
<Layer queryable="0">
  <Name>DEMO</Name>
  <Title>UMN MapServer Itasca Demo</Title>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="388108" miny="5.20312e+006" maxx="500896"
maxy="5.31024e+006" />
  <BoundingBox SRS="EPSG:26915" minx="388108" miny="5.20312e+006" maxx="500896"
maxy="5.31024e+006" />
  <ScaleHint min="0.498903" max="773.299" />
<Layer queryable="0">
  <Name>ctybdpy2</Name>
  <Title>County Boundary</Title>
  <Abstract>Itasca County boundary shapefile. See
http://deli.dnr.state.mn.us/metadata/full/ctybdne2.html for more informa-
tion.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="189775" miny="4.81631e+006" maxx="761662"
maxy="5.47241e+006" />
  <BoundingBox SRS="EPSG:26915" minx="189775" miny="4.81631e+006" maxx="761662"
maxy="5.47241e+006" />
</Layer>
<Layer queryable="0">
  <Name>cities</Name>
<!--
WARNING: Mandatory metadata ,WMS_GROUP_TITLE' was missing in this context.

  <Title>cities</Title>
<Layer queryable="1">
  <Name>mcd90py2</Name>
  <Title>Minor Civil Divisions</Title>
  <Abstract>Minor civil divisions for Itasca County. (boundaries only)</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393234" miny="5.20799e+006" maxx="495770"
maxy="5.30537e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393234" miny="5.20799e+006" maxx="495770"
maxy="5.30537e+006" />
</Layer>
</Layer>
```

```
<Layer queryable="0">
  <Name>twprgpy3</Name>
  <Title>Township Boundaries</Title>
  <Abstract>Pulic Land Survey (PLS) township boundaries for Itasca County. See
http://deli.dnr.state.mn.us/metadata/full/twprgne2.html for more informa-
tion.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393234" miny="5.20799e+006" maxx="495770"
maxy="5.30537e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393234" miny="5.20799e+006" maxx="495770"
maxy="5.30537e+006" />
</Layer>
<Layer queryable="1">
  <Name>lakespy2</Name>
  <Title>Lakes and Rivers</Title>
  <Abstract>DLG lake and river polygons for Itasca County. See
http://deli.dnr.state.mn.us/metadata/full/dlglkpy2.html for more informa-
tion.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393234" miny="5.20817e+006" maxx="495403"
maxy="5.30396e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393234" miny="5.20817e+006" maxx="495403"
maxy="5.30396e+006" />
</Layer>
<Layer queryable="1">
  <Name>dlgstln2</Name>
  <Title>Streams</Title>
  <Abstract>DLG streams for Itasca County. See
http://deli.dnr.state.mn.us/metadata/full/dlgstln2.html for more informa-
tion.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393381" miny="5.20799e+006" maxx="495758"
maxy="5.30537e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393381" miny="5.20799e+006" maxx="495758"
maxy="5.30537e+006" />
</Layer>
<Layer queryable="0">
  <Name>roads</Name>
```

&lt;!--

WARNING: Mandatory metadata ,WMS\_GROUP\_TITLE' was missing in this context.

```
<Title>roads</Title>
<Layer queryable="0">
  <Name>ctyrdln3</Name>
  <Title>County Roads</Title>
  <Abstract>County roads. (lines only) Derived from MNDOT roads layer, see
http://deli.dnr.state.mn.us/metadata/full/dotrdln2.html for more informa-
tion.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393588" miny="5.208e+006" maxx="495770"
maxy="5.30511e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393588" miny="5.208e+006" maxx="495770"
maxy="5.30511e+006" />
  <ScaleHint min="0" max="149.671" />
</Layer>
<Layer queryable="0">
  <Name>ctyrdln3_anno</Name>
  <Title>County Roads</Title>
  <Abstract>County roads. (shields only) Derived from MNDOT roads layer, see
http://deli.dnr.state.mn.us/metadata/full/dotrdln2.html for more informa-
tion.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393588" miny="5.208e+006" maxx="495770"
maxy="5.30511e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393588" miny="5.208e+006" maxx="495770"
maxy="5.30511e+006" />
  <ScaleHint min="0" max="149.671" />
</Layer>
<Layer queryable="0">
  <Name>majrdln3</Name>
  <Title>Highways</Title>
  <Abstract>Highways- state, US and interstate. (lines only) Derived from
MNDOT roads layer, see http://deli.dnr.state.mn.us/metadata/full/dotrdln2.html
for more information.</Abstract>
  <SRS>EPSG:26915</SRS>
```

```
<LatLonBoundingBox minx="393732" miny="5.20809e+006" maxx="494784"
maxy="5.30537e+006" />
<BoundingBox SRS="EPSG:26915" minx="393732" miny="5.20809e+006" maxx="494784"
maxy="5.30537e+006" />
<ScaleHint min="0" max="299.342" />
</Layer>
<Layer queryable="0">
  <Name>majrdln3_anno</Name>
  <Title>Highways</Title>
  <Abstract>Highways- state, US and interstate. (shields only) Derived from
MNDOT roads layer, see http://deli.dnr.state.mn.us/metadata/full/dotrdsn2.html
for more information.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393732" miny="5.20809e+006" maxx="494784"
maxy="5.30537e+006" />
  <BoundingBox SRS="EPSG:26915" minx="393732" miny="5.20809e+006" maxx="494784"
maxy="5.30537e+006" />
  <ScaleHint min="0" max="299.342" />
</Layer>
</Layer>
<Layer queryable="1">
  <Name>airports</Name>
  <Title>Airports</Title>
  <Abstract>Airport runways for Itasca County.</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="434634" miny="5.22872e+006" maxx="496393"
maxy="5.29193e+006" />
  <BoundingBox SRS="EPSG:26915" minx="434634" miny="5.22872e+006" maxx="496393"
maxy="5.29193e+006" />
</Layer>
<Layer queryable="0">
  <Name>cities</Name>
  <Title>Minor Civil Divisions</Title>
  <Abstract>Minor civil divisions for Itasca County. (annotation only)</Abstract>
  <SRS>EPSG:26915</SRS>
  <LatLonBoundingBox minx="393234" miny="5.20799e+006" maxx="495770"
maxy="5.30537e+006" />
```

```
<BoundingBox SRS="EPSG:26915" minx="393234" miny="5.20799e+006" maxx="495770"
maxy="5.30537e+006" />
</Layer>
</Layer>
</Capability>
</WMT_MS_Capabilities>
```

### C.3 Antwort feature\_info

Feature\_info results:

```
Layer ,mcd90py2'
  Feature 15:
    MCD = ,12502'
    AREA = ,15474136.33819'
    PERIMETER = ,22293.75377'
    COUNTY = ,61'
    NAME = ,Coleraine'
    CITY_NAME = ,Coleraine'
    TOTAL = ,1041'
    WHITE = ,1029'
    BLACK = ,0'
    AMERIND = ,9'
    ASIAN = ,2'
    OTHER = ,1'
    TOTAL18_ = ,756'
    WHITE18_ = ,747'
    BLACK18_ = ,0'
    AMERIND18_ = ,8'
    ASIAN18_ = ,1'
    OTHER18_ = ,0'
    HISPANIC = ,7'
    NHISPANIC = ,1034'
    NHWHITE = ,1023'
    NHBLACK = ,0'
    NHAMERIND = ,9'
    NHASIAN = ,2'
    NHOTHER = ,0'
    HISPANIC18 = ,1'
    NHISPANIC1 = ,755'
```

NHWHITE18\_ = ,746'  
NHBLACK18\_ = ,0'  
NHAMERIND1 = ,8'  
NHASIAN18\_ = ,1'  
NHOTHER18\_ = ,0'  
MCDCHAR = ,12502'  
LMCD = ,12502'  
ACRES = ,3823.659'  
PERFEET = ,73145.806'  
XUTM = ,468176'  
YUTM = ,5238538'

## D Abbilden von OO Schema zu ER Schema

Text aus Dokument von Prof. S.F. Keller (MM\_ILI2-zu-1\_r03de.doc)

# INTERLIS 2 zu 1: Modell- und Typen-Abbildungen



Prof. Stefan F. Keller, sfkeller@hsr.ch

Ausgabe 3 vom 2002-10-29 (deutsch)

### Inhaltsverzeichnis

1	Einführung	101
1.1	Zweck und Inhalt dieses Dokuments	101
1.2	Weitere Informationen	101
1.3	Grundsätzliches	101
2	Abbildung von OO- zu ER-Datenmodellen	101
2.1	Vererbung	101
2.2	Beziehungen	104
2.3	Unterstrukturen	104
3	Abbildung von INTERLIS 2- zu 1-Datentypen	105

## Einführung

### Zweck und Inhalt dieses Dokuments

Dieses Dokument ist ein („work-in-progress“) über aktuelle Fragen der Ableitung und -Zuordnung von Modellen, z.B. von konzeptionell-externen Schemas zu internen Schemas.

Konkret gehören dazu auch Fragen, wie das „Mapping“ von INTERLIS 2 zu INTERLIS 1 oder von objekt-orientierten zu relationalen Datenmodellen.

### Weitere Informationen

Die Hauptinformationsquellen sind:

<http://www.INTERLIS.ch>, und

<http://www.integis.ch>

Siehe auch.

Das INTERLIS 2-Referenzhandbuch

Benutzerhandbuch zum INTERLIS 2-Compiler

Kontakt: siehe Titelblatt.

Der Autor (bzw. die Autorinnen und Autoren) sind damit einverstanden, dass Veröffentlichungen und Vervielfältigungen ihrer Beiträge erfolgen und dass die Urheber- und Nutzungsrechte aller Beiträge den Herausgebern der INTERLIS-Dokumente zu gleichen Rechten übertragen werden.

### Grundsätzliches

Die unterschiedlichen Betrachtungsweisen von INTERLIS 1 und INTERLIS 2 führen zu einigen grundsätzlichen Abbildungsproblemen. Sie werden im Kapitel 2 beleuchtet.

Es sollen wenn imm möglich Alternativen aufgezeigt werden, mit denen einige vorhersehbare Probleme im Zusammenhang mit den bestehenden Systemen und Datenbeständen vermieden werden können.

Es ist zu Unterscheiden zwischen folgenden Abbildungen von INTERLIS 2 nach INTERLIS 1-Modellen:

Abbildung von OO- zu ER-Datenmodellen

Abbildungen der Datentypen von INTERLIS 2 zu 1

## Abbildung von OO- zu ER-Datenmodellen

### Vererbung

Text auf Basis Dorfschmid (2000).

### Abstrakte Klassen

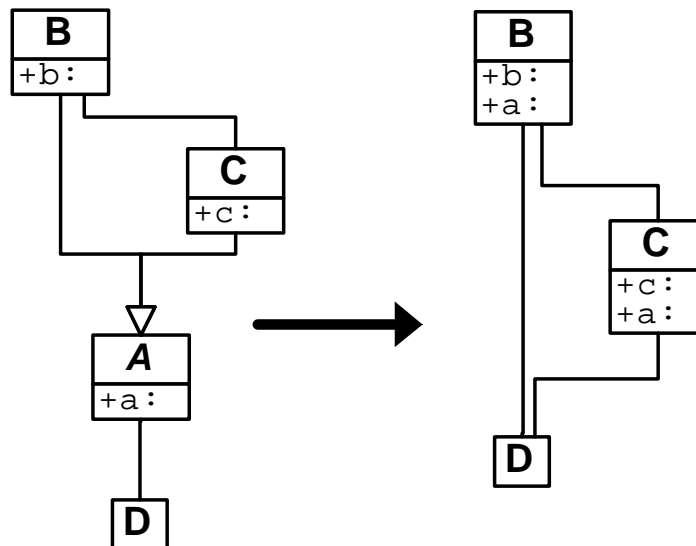
Siehe Klassen.

## Klassen

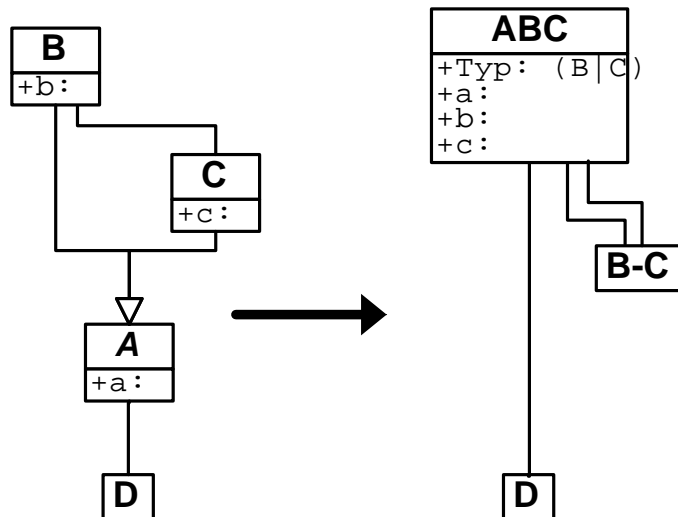
In vielen OO-Datenmodellen werden zunächst abstrakte Objektklassen (Klasse A im Beispiel) eingeführt und diese dann zu weiteren abstrakten und schliesslich zu konkreten Klassen (Klassen B und C im Beispiel) spezialisiert. Dieses Mittel ist typisch für den objektorientierten Entwurf. Um das Beispiel realistischer zu gestalten, gibt es eine Beziehung zwischen den Subklassen B und C und es wird noch eine konkrete Klasse D eingeführt, die eine Assoziation zur Klasse A hat.

Für die Abbildung in ein relationales Modell stehen verschiedene Möglichkeiten zur Verfügung:

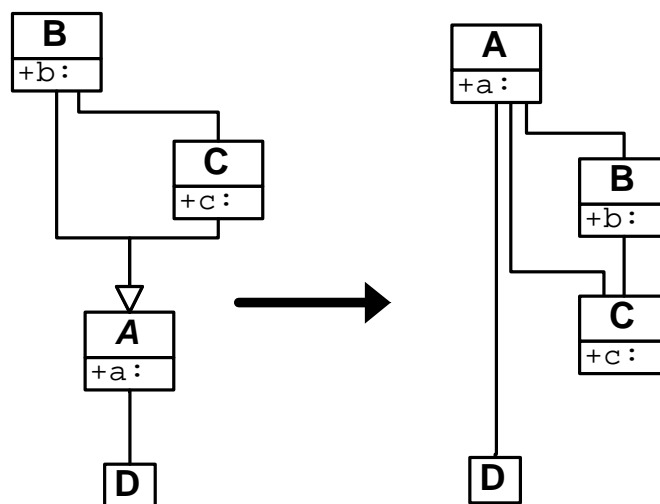
1. **Variante:** Jede konkrete Objektklasse (B, C, D) wird in eine relationale Tabelle abgebildet (Anschaulich: Die Attribute der Oberklasse "wandern" zu den Unterklassen). Diese enthält dann alle Attribute der entsprechenden konkreten Objektklasse und aller durch sie geerbten Superklassen. Diese Möglichkeit entspricht am ehesten einer natürlichen Auffassung. Sie hat aber den Nachteil, dass Beziehungsattribute, die auf eine abstrakte Klasse verweisen auf verschiedene konkrete Tabellen verweisen müssen. Dies ist in der relationalen Methode nur mittels mehrerer Beziehungsattribute (im Beispiel je ein Beziehungsattribut von D zu B bzw. von D zu C) oder mit zusätzlichen Beziehungstabellen beschreibbar.



2. **Variante:** Die Summe aller aus einer Basisklasse spezialisierten Klassen wird auf eine einzige relationale Tabelle (Tabelle ABC) abgebildet (Anschaulich: Die Attribute der Unterklassen "wandern" zur Oberklasse). Diese Tabelle enthält Attribute, die in den verschiedenen Klassendefinitionen vorkommen und zusätzlich ein Attribut, das beschreibt, zu welcher konkreten Objektklasse ein Objekt gehört. In einem konkreten Objekt sind dann dieser Objekttyp und diejenigen Attribute definiert, die für die jeweilige Spezialisierung in Frage kommen. Diese Abbildungsweise kann zwar etwas unübersichtlich sein; sie hat aber den Vorteil, dass keine Schwierigkeiten mit Beziehungsattributen entstehen, die von Drittobjekten ausgehen. Beziehungen, die innerhalb von Objekten bestehen, die in der gleichen relationalen Tabelle abgebildet werden, müssen allerdings auch mittels einer Beziehungstabelle dargestellt werden, damit die INTERLIS 1-Vorschrift eingehalten werden kann, wonach nur schon bekannte Objekte gezeigt werden dürfen. Zudem müssen alle obligatorischen Attribute von Subklassen in der Tabelle auf nicht-obligatorisch geändert werden.



3. **Variante:** Für jede abstrakte oder konkrete Objektklasse wird eine entsprechende relationale Tabelle gebildet (Anschaulich: Klassen werden 1:1 zu Tabellen abgebildet). Die Tabellen enthalten jeweils die Attribute gemäss der Objektklasse. Spezialisierende Klassen enthalten zusätzlich einen Verweis (Beziehungsattribut) auf die jeweilige Superklasse (Im Beispiel also auf A). Ein konkretes Objekt wird entsprechend der Vererbungsstruktur in mehrere Teilobjekte zerhackt, die jeweils auf das zugehörige Superobjekt verweisen. Dazu kommt bei der Basis-klasse ein diskriminierendes Attribut `_Typ`, das hilft, die Subklassen zu unterscheiden (kann evtl. weggelassen werden). Dieses Vorgehen hat den Vorteil, dass keinerlei Probleme mit Beziehungen entstehen. Es hat aber den Nachteil, dass die Objekte in Teilobjekte zerhackt werden. Die damit verbundene grössere Objektmenge ist kaum von Belang. Es muss aber damit gerechnet werden, dass die Erstellung der zerhackten Objekte und vor allem ihre Zusammenfügen in verschiedenen Systemen zu Problemen führen könnte.



Wir empfehlen Variante 3 (Klassen werden 1:1 zu Tabellen abgebildet).

Hinweise:

Views für Subklassen-Objekte: Ein interessante Überlegung ist, ob die "OO-zu-relational"-Abbildung (= ili2-to-pgsql-CREATE TABLE-Software) nicht gleich Views für die Subklassen definieren könnte, so könnte man Subklassen-Objekte einfacher abfragen.

## Vererbung von EXTENDED-Attributen:

EXTENDED-Attribute in Subklassen (typischerweise erweiterte Aufzähltypen) werden auch in den Subklassen aufgeführt mit dem Unterschied, dass dort die erweiterten Werte und gleichzeitig (d.h. anteilmässig redundant) in der Basisklasse die ursprünglichen Werte stehen (ich glaube, die Alternative, die das Attribut mit den erweiterten Werten nur in der Subklasse zu verwalten, ist nicht machbar, denn sonst könnte es keine konsistenten Objekte gemäss Basis mehr geben).

Beispiel: Siehe wieder das Roads-Beispiel: Das

```
CLASS PointObject( Type, Position );
CLASS PointObjectExtended EXTENDS PointObject ( Type(EXTENDED) );
```

... wird relational zu (siehe \_Type und REF):

```
SQL-TABLE PointObject( _TID, Type, Position, _Type );
SQL-TABLE PointObjectExtended ( _TID, Type, _Ref: FOREIGN KEY REFERENCES PointObject );
```

Anderes Beispiel: Gegeben, in einer Basisklasse gibt es den Aufzählungswert "rot" und eine die Basisklasse erweiternde Subklasse erweitert diesen Attributtyp mit "rot.hell"; dann würde ein Tupel der "Basisklasse-Tabelle" (als Teil des Subklassen-"Gesamtobjekts") den Wert "rot" erhalten und ein Tupel der "Subklasse-Tabelle" würde den Wert "rot.hell" erhalten.

## Beziehungen

### Assoziationen

Im Falle von "zweiwertigen" Assoziationsklassen ohne zusätzlichen Attributen, bei denen eine der Rollen die Kardinalität 1 hat, kann diese weggelassen werden, indem die Rolle dem anderen beigeigten Tabelle zugeordnet wird.

In den anderen Fällen, ist eine Tabelle zu schaffen ("Kreuz- oder Zwischen-Tabelle).

### Beziehungen über Themen( bzw. Behälter-)Grenzen hinweg

Müssen als Primär-Fremdschlüssel abgebildet werden.

### Aggregation und Komposition von Klassen

Aggregationen werden 1:1 in Tabellen abgebildet.

Kompositionen in INTERLIS 2 müssen in INTERLIS 1 als eigenständige Objekte aufgefasst werden, die jeweils eine Beziehung zum "Oberobjekt" aufweisen. Wird dieselbe Struktur in verschiedenen Objekten als Komposition verwendet, müssen jeweils verschiedene Tabellen gebildet werden, da die Beziehung jeweils auf verschiedene Oberobjekte verweist.

Siehe auch Kapitel 2.3.1 Kompositionsattribute mit LIST/BAG OF.

## Unterstrukturen

### Kompositionsattribute mit LIST/BAG OF

Attributtypen, die in INTERLIS 2 mit BAG OF (Objektreihenfolge unbestimmt, bzw. LIST OF (Reihenfolge relevant) definiert sind, werden in eine separate, zusätzliche Tabelle ausgelagert, die eine INTERLIS 1-Beziehung auf die ursprüngliche Tabelle erhalten. Bei LIST OF ist in der zusätzlichen Tabelle nebst der Beziehung auch eine Ordnungsnummer zu führen.

## Abbildung von INTERLIS 2- zu 1-Datentypen

Folgende Zuordnungen werden gemacht:

URI => TEXT

DIRECTED POLYLINE => POLYLINE (DIRECTED wird ignoriert)

INTERLIS\_1\_DATE => DATE;

BOOLEAN => (false, true) !! ergibt die Werte (0, 1)

Für weitere Angaben siehe das INTERLIS 2-Compiler-Handbuch.

Man beachte, dass die INTERLIS 2 zu 1-Option gemeinsame DOMAINS nur noch einzeln bei den Attributen angibt, obschon alle auf einen gemeinsamen Domain zurückführen. Steht also in INTERLIS 2 "DOMAIN LKoord = COORD..." so findet man diesen Domain im INTERLIS 1-Text nicht mehr: Der Attribut-Typ wird als "COORD2..." allen Geometritypen direkt angehängt. Ich würde erwarten, dass diese Information erhalten bleibt, d.h. das "DOMAIN LKoord = COORD2..." auch im INTERLIS 1-Modell erscheint. Das hat keinen direkten Einfluss auf die Daten, nur auf die Semantik der Beschreibung.

## E Literatur- / Quellverzeichnis

- UMN MapServer Handbuch und Referenz, Thorsten Fischen, ISBN 3-00-009844-5,  
<http://www.mapmedia.de>.
  
- Open GIS Consortium Inc. Web Map Implementation Specification Version:1.1.0, Reference number of  
this OpenGIS project document: OGC 01-047r2  
<http://www.opengis.org/techno/specs/01-047r2.pdf>.
  
- Open GIS Consortium Inc. Simple Features Specification For SQL  
OpenGIS project document: 99-049, May 5, 1999  
<http://www.opengis.org/techno/specs/99-049.pdf>
  
- UMN MapServer  
<http://mapserver.gis.umn.edu/>
  
- UMN MapServer WMS Server HowTo  
<http://mapserver.gis.umn.edu/doc36/wms-server-howto.html>.
  
- PostgreSQL, Software und Dokumentation  
<http://www.postgresql.com>
  
- PostGIS, Software und Dokumentaion  
<http://postgis.refrations.net>
  
- INTERLIS  
<http://www.INTERLIS.ch>
  
- INTERLIS Referenzhandbuch 2.1 (17.10.2001)  
[http://www.INTERLIS.ch/refdocs/ili2-refman\\_2001-10-17.zip](http://www.INTERLIS.ch/refdocs/ili2-refman_2001-10-17.zip)
  
- INTERLIS Referenzhandbuch 2.2 (20.09.2002)  
(noch nicht auf Homepage)