PDF-Version erstellt mit OpenOffice 1.1Beta2 (Linux)
www.openoffice.org

# Rauminformationssystem mit PostgreSQL und SVG

## *am Beispiel des Prototypen „UniRIS“*

**Diplomarbeit**
zur Erlangung des Magistergrades
der Naturwissenschaften
an der Universität Innsbruck

eingereicht von
Florian Jurgeit

**HINWEIS**

Der Arbeit liegt eine CD mit den Grundlagen-Beispielen bei. Außerdem befindet sich eine PDF-Datei der schriftl. Arbeit, sowie weitere Unterlagen und Tools auf der CD.
Die CD sollte bei aktivierter Autostart-Funktion unter Windows nach dem Einlegen selbst das Hauptmenü starten – alternativ kann dies manuell durch Aufruf der Datei „index.html“ erfolgen.

Auf den Prototypen kann über folgende URL zugegriffen werden:
http://geo4.uibk.ac.at/users/jurgeit/da

**„Der XML-Ansatz, Daten in einer für den Menschen lesbaren Form zu präsentieren, ist bei großen Mengen von Koordinaten genauso spannend und sinnvoll wie die Lektüre des örtlichen Telefonbuches.“**

*Krüger und Märtin, iX 11/2000*

# Inhaltsverzeichnis

# 1 Vorwort

Im Rahmen meines Studiums hat sich mein Interesse immer mehr auf den Bereich der „digitalen Geographie“, im wesentlichen die Geographischen Informationssysteme (GIS), fokussiert; der parallele Umgang mit Themen der Internetkartographie – anfänglich konventionell rasterbasiert – und in Folge mit dem Vektorformat SVG (Scaleable Vector Graphics) hat bei mir großes Interesse bezüglich der Zugänglichkeit „geographischer Information“ für jedermann via Internet geweckt.
Die Beschäftigung mit dem Thema Serverdatenbanken im Rahmen einer Lehrveranstaltung hat bei mir den Wunsch nach einer Vertiefung geweckt - nicht zuletzt aufgrund der Stellung von Linux basierten Systemen als Server und den leistungsfähigen frei verfügbaren Datenbanksystemen wie MySQL und PostgreSQL.
Map-Server als einfacher Zugang zu geographischen Daten waren theoretisch bekannt, jedoch hat sich im Rahmen des Studiums leider nie die Möglichkeit geboten diese Thematik vertiefend und praktisch kennen zu lernen.
Schlussendlich hat sich die Vorstellung eines Web-basierten Informationssystems auf Basis freier Techniken/Standards entwickelt, wobei SVG, eine Serverdatenbank und die Programmiersprache PHP (Hypertext Pre-Processor) eine Rolle spielen sollten.
Das Interesse der Universität an einem Prototyp für ein einfaches Gebäudeinformationssystem hat die Möglichkeit der Erprobung der erwähnten Techniken für diesen Zweck ermöglicht; es handelt sich hierbei um keinen klassischen geographischen Inhalt, jedoch spielt es prinzipiell keine Rolle ob der Endbenutzer nach Räumen in einem Gebäude sucht und deren Lage visualisiert bekommt oder ob es sich bei den gesuchten Objekten zum Beispiel um Gebäude im Rahmen einer Adressverortung handelt.

Bedanken möchte ich mich bei Dr. Armin Heller, der die Betreuung dieser Arbeit übernommen hat und dem „unkonventionellen“ Thema offen gegenüber stand.
Des weiteren danke ich den SVG-Experten des Projekts „Digitaler Tirolatlas“ (www.tirolatlas.net), insbesondere Klaus Förster, der immer wieder mit guten Tipps aufwarten konnte – speziell auch rein technischer Natur bezüglich Problemen bei der Konfiguration meines Test-Servers.
Großer Dank gebührt auch den zahlreichen Entwicklern der verwendeten freien Software – beginnend bei dem Betriebssystem Linux bis hin zu den Datenbanksystemen.
Dank gebührt auch meinen Eltern, die mir die Ausbildung ermöglicht haben.

Auch diese schriftliche Arbeit entstand mit freier Software, dem Office-Paket OpenOffice (ww-

w.openoffice.org), welches sich für lange gegliederte Arbeiten mit Abbildungen sehr gut eignet und für alle gängigen Betriebssysteme (Windows, Macintosh, Linux, ...) erhältlich ist.

In der schriftlichen Arbeit wurde der Weg verfolgt praktische Hinweise bzw. Tipps grafisch hervorzuheben, um dem Leser einen schnellen Zugriff auf Beispiele und Hinweise zu ermöglichen.
Der erste Teil der Arbeit beschäftigt sich mit ein wenig Theorie, wie zum Beispiel den Grundlagen der Visualisierung für das WWW, Theorie zu Gebäudeinformationssystemen und den technischen Grundlagen (LAMP/LAPP, Datenbanken, SVG, PHP, Javascript); im zweiten Teil wird der Prototyp eines 'Raumauskunftsystems' vorgestellt.

*Florian Jurgeit, April 2003*

# 2 Einleitung

## 2.1 Grundlagen der Visualisierung für das WWW

Die Visualisierung geographischer Information für das Internet ist rezent nicht mehr wegzudenken, nicht zuletzt aufgrund der hohen Verfügbarkeit und der hohen Nachfrage – Hermann und Asche (2001) gehen davon aus, dass täglich ca. 40 Millionen Karten von Nutzern im Internet aufgerufen werden. Das Webmapping hat bereits eine wechselvolle Geschichte hinter sich, wobei die unterschiedlichen Entwicklungen mit der Weiterentwicklung des Internet und den zugrunde liegenden Techniken einhergehen.

Obwohl bereits in zahlreichen Arbeiten dargestellt, sollen wesentliche Entwicklungen (Techniken) kurz erläutert werden – generell wird bezüglich des Ausgabeformats (Raster/Vektor) und der Interaktivität (statisch/dynamisch) unterschieden.

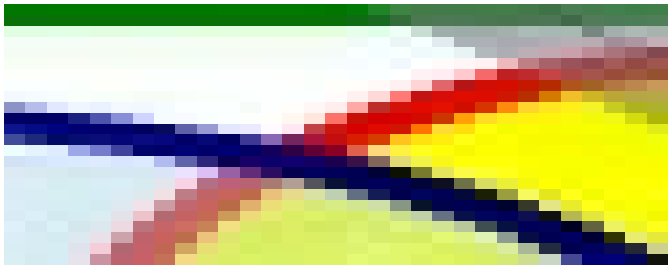

*Abbildung 1: Rastergrafik - vergrößert*

*Abbildung 2: Vektorgrafik (SVG) - vergrößert*

Rasterkarten statischer Natur können als Ursprung des Webmappings bezeichnet werden, wobei es sich meist um klassische Papierkarten in eingescannter Form handelt, die in eine Web-Seite als für den Browser anzeigbare Grafik eingebunden werden – meist handelt es sich um JPEG (Joint Photographic Experts Group), GIF (Graphical Interchange Format) oder PNG (Portable Network Graphics) Grafiken (vgl. dazu z. Bsp. Münz (2001), Winter (2000) und Fürpaß (2001)); diese Form des Webmappings stellt auch rezent noch die dominante Form dar.

| Format | Farbtiefe | Anzahl der Farben | Trans-parenz stufen | Animation | Ältere Browser ohne Plugin | Aktuelle Browser (NN 6.1, IE 6.0) |
|---|---|---|---|---|---|---|
| GIF | 8 Bit | 256 | 1 | Ja | Ja | Ja |
| JPG/JPEG | 24 Bit | 16 777 216 | 0 | Nein | Ja | Ja |
| PNG 8 | 8 Bit | 256 | 256 | Nein | Nein | Ja |
| PNG 16 | 24 Bit | 16 777 216 | 256 | Nein | Nein | Ja |

*Abbildung 3: Häufige Rastergrafikformate (aus: Fürpaß, 2001)*

Die Weiterentwicklung führt über dynamische (interaktive) Rasterkarten, wobei man sich der Technik der Image-Maps bedient und so genannte Clickable-Maps erstellt.

Mit dem Vektorformat Flash der Firma Macromedia hat eine neue Ära begonnen, jedoch handelt es sich um ein proprietäres Format, welches im Browser nur durch ein Plug-In dargestellt werden kann und keinen offiziellen Standard des W3C (**W**orld **W**ide **W**eb **C**onsortium – www.w3.org) darstellt.

Mit dem Vektorformat SVG (**S**caleable **V**ector **G**raphics) bietet sich nun ein offizieller vom W3C unterstützter neuer Standard für vektorbasierte dynamische Webgrafiken an (s. www.w3.org/graphics/svg), der insbesondere von der Webmapping-Gemeinde gut angenommen wird (s. Krüger, 2000).

Einen Überblick über die Geschichte von SVG und eine kleine Einführung bietet Kapitel 3.3.

Als dynamische Karten müssen auch solche bezeichnet werden, die „on-demand“ erstellt werden, beziehungsweise durch Benutzeraktionen manipuliert werden können. (s. Herrmann und Asche, 2001)

Die Entwicklung leistungsfähiger Web-Server mit Datenbankanbindungen, wie sie auch zahlreiche eCommerce-Lösungen benötigen, hat sich ebenfalls positiv auf das Webmapping ausgewirkt, wobei Map-Server eine spezielle Variante einer solchen Client-Server-Lösung sind (s. Kapitel 2.1.1).

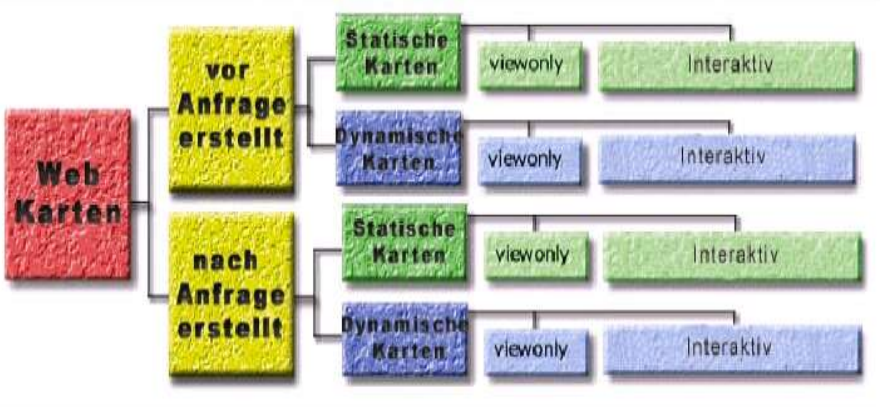


*Abbildung 4: Klassifikation von Webkarten (Fürpaß, 2001)*

Insbesondere die Map-Server machen eine Erweiterung der Definition notwendig:

- Karten die vor der Anfrage an einen Webserver zu einem beliebigen Zeitpunkt erstellt wurden.
- Karten, die individuell auf Basis der Anfrage on-demand erstellt werden.

### 2.1.1 Map-Server als Mittel der Visualisierung

Map-Server treten zunehmend bei verschiedenen Diensten im Internet in Erscheinung, seien es Filialstandorte großer Handelsketten oder Routingangebote.

Die Geschichte von Map-Servern geht jedoch auf die Zeit vor dem Internet Boom ab den späteren 90er Jahren zurück: Bereits 1994 erschien die erste experimentelle Webseite mit einer Weltkarte und Interaktionsmöglichkeiten, basierend auf dem Xerox PARC (Palo Alto Research Center) Map-Server. (s. Herrmann und Asche, 2001)

Fürpaß (2001) definiert Map-Server wie folgt:

„Ein Map-Server ist ein Programm, welches der interaktiven, individuellen und unmittelbaren Erstellung und Visualisierung von geographischen Informationen in Form von Karten über das Internet dient."

Hierbei sind zwei Worte von besonderer Bedeutung: Interaktiv und Individuell – ein Map-Server erstellt somit die Karten auf Basis individueller Erfordernisse (z. Bsp. Auswahl eines bestimmten Themas) und ermöglicht dem Nutzer ständige Interaktion – zum Beispiel jederzeit einen bestimmten Ausschnitt zu vergrößern (Zoom-Funktionalität).

Wagner (2001) definiert Map-Server wie folgt: „Diese Programme bieten die Erzeugung von Karten zum Zeitpunkt der Anfrage („on-the-fly"). Neben Rasterdaten werden auch Vektordaten unterstützt und sind beliebig kombinierbar. Weiterhin sind uneingeschränkte Zoomstufen möglich."

Map-Server müssen als Brücke zwischen Geodaten (Serverseitig) und Karte (Grafik) auf dem Client betrachtet werden, wobei es sich nicht um eine einfache Client-Server Technik handelt, da der Server die Daten nicht nur ausliefert, sondern auch aufbereitet.

Für die Software, die diese Aufbereitung der Daten übernimmt wird auch häufig der Begriff Middleware verwendet (s. Herrmann und Asche, 2001), wobei es sich dabei auch um mehrere einzelnen zusammenarbeitende Module handeln kann.

Der Begriff Map-Server schließt aufgrund dieser Definition eine Bedeutung im Sinne eines File-Servers oder ftp-Servers für Geodaten aus – für diese sollte der Begriff Geodatenserver verwendet werden. (s. Fürpaß, 2001)

Der Begriff Map-Server sollte somit als Software betrachtet werden, dennoch kann damit auch ein Rechner, der nur für diesen Zweck installiert ist, gemeint sein.

Neben dem Unterscheidungsmerkmal Hardware/Software kann bezüglich der Funktionalität unterschieden werden:

- Statische Map-Server: Diese liefern lediglich vorgefertigte Karten aus, die zuvor erstellt wurden und als Rasterdaten vorliegen. Wagner (2001) nennt als Beispiel das Produkt MapIt! (www.mapit.de) welches unter der LGPL[1] Lizenz steht und in Python implementiert ist. MapIt! ermöglicht eine schnelle Realisierung einer einfachen Navigation innerhalb festgelegter Zoomstufen auf vorbereiteten Rasterkarten.
- Visualisierungs-Map-Server: Diese Server dienen hauptsächlich der Umwandlung von Geodaten für das Internet und bieten keinerlei GIS-Funtkionalitäten.
- Interaktive Map-Server: Hierbei können verschiedene Parameter vor der Visualisierung beeinflusst werden, wie zum Beispiel die Auswahl von Ebenen und deren visuelle Darstellung.
- Online GIS: Diese stellen die „Krönung“ dar, da neben den Basisfunktionen zur Visualisierung auch GIS-Funktionalitäten integriert sind, wie zum Beispiel das Bilden von Puffern; ein Beispiel hierfür stellt die Applikation SkiGIS dar (s. www.skigis.at).

Gartner (1999) verwendet eine ähnliche Gliederung, wobei er folgende Kategorien von Web-GIS („GIS on the Web“) unterscheidet:

- Geodaten-Server
- Map-Server
- Interaktive Map-Server
- Online GIS
- GIS-Funktionen-Server („GIS Functions Server“)

Neben dieser Gliederung auf Basis der Funktionalität, die sinnvoll erscheint, gibt es zahlreiche weitere Ansätze und weitere Untergliederungen, wobei an dieser Stelle auf die zahlreiche Literatur verwiesen sei (z. Bsp. Cartwright et al (1999), Herrmann und Asche (2001), Fürpaß (2001)).

1 GPL: GNU PUBLIC LICENSE (www.gnu.org)

Insbesondere die Hersteller von GIS-Softwareprodukten haben nahezu alle einen Map-Server mit GIS-Funktionalität im Angebot, wobei die Preise ähnlich sind – prinzipiell gilt jedoch die Regel, dass der Map-Server vom Hersteller des eingesetzten GIS sein sollte, sodass Firmen/Institutionen die ESRI-Produkte einsetzen meist auch den Map-Server von ESRI einsetzen.

Strobl (2001) betont, dass sich die Produkte signifikant in spezifischen Leistungsmerkmalen unterscheiden, sei es in der Integration in Web-Browser (PlugIn vs. Java vs. Raster-Grafiken) und der jeweiligen Server-Architektur.

„In der Regel muss sich der Benutzer derzeit für eines der angebotenen Systeme durchgehend entscheiden, im Gegensatz zur ansonsten gewohnten Praxis des WWW kann mit einem Standard-Browser nicht in allen Fällen auf die spezifischen Funktionen eines Map-Server bzw. auf beliebige, im Internet verfügbare Datenbestände zugegriffen werden." (Strobl, 2001)

Eine interessante Alternative zu den kommerziellen Produkten stellen OpenSource-Produkte dar, die eine immer größere Verbreitung finden, nicht zuletzt aufgrund der zunehmenden Beliebtheit des Betriebssystems LINUX (www.linux.org) – sei es als leistungsfähiges GIS wie es die Software GRASS[2] (s. www.geog.uni-hannover.de/grass/index2.html) darstellt oder auch als Map-Server, wobei hier insbesondere der UMN Map-Server (http://mapserver.gis.umn.edu/index.html) große Beachtung findet (s. Wagner (2001), Pucher (2001)).

Der UMN wurde von der University of Minnesota entwickelt und wird mit Unterstützung der NASA weiterentwickelt und betreut.

Pucher (2001) weist folgende Merkmale aus:

- Verarbeitung von Vektor- und Rasterdaten
- Möglichkeit der Kachelung von Vektor- und Rasterdaten
- Indexierung räumlicher Daten
- Maßstabsabhängige Visualisierungen
- Automatische Beschriftung räumlicher Objekte
- Automatische Erstellung von Legende und Maßstabsleiste
- Selektion von räumlichen Objekten
- Freie Gestaltung der Ausgabeseiten

Da es sich um ein OpenSource-Produkt handelt sind zahlreiche Erweiterungen und Schnittstellen hinzugekommen, sodass die Anbindung einer OpenSource Server-Datenbank wie MySQL oder PostgreSQL kein Problem darstellt, selbst bei der Wahl der zur Anbindung bevorzugten Skriptsprache hat man verschiedene Möglichkeiten (z. Bsp. PHP, Perl/CGI).

2 GRASS: Geographic Resources Analysis Support System

Auf der Webseite www.geoplace.com kann ein Überblick über die verschiedenen kommerziellen Produkte eingesehen werden – folgend ein kurzer Ausschnitt:

| ***Produkt*** | ***Preis*** |
|---|---|
| ESRI: ArcIMS | 7500 US-Dollar |
| Microimages: TNT-Server | 5000 US-Dollar |
| Intergraph: GeoMedia Web-Map | 10000 US-Dollar |
| Autodesk: MapGuide | 9990 US-Dollar |

Strobl (2001) gibt einen Überblick über die bisherigen Entwicklungen, wobei neben einer Entwicklung bezüglich des Funktionsumfanges die Entwicklung in Richtung „offener Systeme“ und somit einheitlicher Schnittstellen einen wesentlichen Faktor darstellt, sodass man sich einer Nutzung jeglicher Geoinformation ohne Einschränkungen seitens des Clients nähert. Einen wichtigen Impuls hierzu liefert das Open-GIS Konsortium (www.opengis.org), welches seit 1993 zur Entwicklung offener Plattformen zur Geographischen Informationsverarbeitung beiträgt.

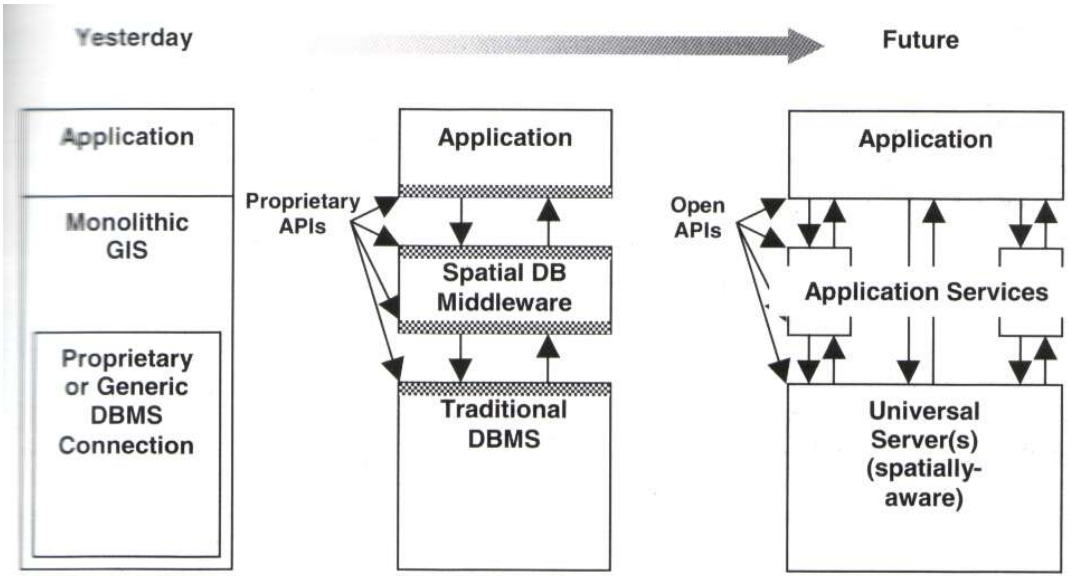


*Abbildung 5: OpenGIS-Modell (aus: Strobl, 2001)*

## *2.2 Gebäudeinformationssysteme (GebIS)*

Mit der zunehmenden Größe von Gebäuden geht die Überblickbarkeit verloren, sodass Systeme zur Verwaltung und Information des Gebäudes bzw. der Gebäude notwendig werden – einfache Fragestellungen wie „Wo befindet sich das Büro von Herrn/Frau XY ?“ oder „In welchem Büro befindet sich der PC 1234 und wo ist dieses ?“, aber auch Fragestellungen vom Raum ausgehend sind von Relevanz: „Welche Personen befinden sich im Büro 1234 und wie kann man diese erreichen ?“

Neben diesen einfachen Fragestellungen, die jedermann zugänglich sein können, gehen GebIS in Expertensysteme über, die der Steuerung von Gebäudetechnik etc. dienen.
In den folgenden Unterkapiteln wird der Begriff GebIS definiert, auf die Datenmodellierung und Komponenten eingegangen.

## 2.2.1 Was sind Gebäudeinformationssysteme ?

„Ein Gebäudeinformationssystem ist ein System für die Erfassung, Dokumentation und Auskunft bzw. Herausgabe von Gebäudeinformationen. Es umfasst ein Datenmodell, die Software, die Hardware und systemergonomische Funktionen für Ersterfassung, Aktualisierungsdienst und Anwendung.“ (Scheu, 1999)
Wichtig sind zudem die mit den Gebäuden bzw. deren Bestandteilen und Einrichtungen verbundenen Beschreibungen, Nutzungen und Bewertungen.
Des weiteren kann die Aussage getroffen werden, dass mit Hilfe von Datenbanken und anderen Softwarekomponenten für das Datenmodell eine Datenmodellierung erfolgt.

## 2.2.2 Rahmenbedingungen

Der Aufbau jedes Informationssystems setzt eine vorangehende Ist-Analyse unter Berücksichtigung der Rahmenbedingungen voraus, wobei der Qualitätsbegutachtung der vorhandenen Daten (Pläne) große Beachtung zu schenken ist.
Gebäudeinformationssysteme benötigen eine Vielzahl unterschiedlicher Daten, darunter geometrische, technische und planerische Informationen, wobei diese rezent meist noch nach herkömmlichen Methoden erfasst und geführt werden – es handelt sich meist um Karteien und Planwerke.
Zur Verwirrung tragen Daten bei, die in Plänen stehen, aber eigentlich in Karteien (▸ Datenbanken) abgelegt werden sollten; dies ist oft Ausdruck persistenter Informationshierarchien.(s. Zippelt, 1999)

*Abbildung 6: Vermischung von Plan und Sachdaten*

Die moderne Datenverarbeitung bietet jedoch die Möglichkeit, geometrische Daten und fachliche

Daten (Sachdaten) miteinander verknüpft abzulegen und in einem umfassenden und raumbezogenen Informationssystem darzustellen.
Neben der Bewertung der Pläne müssen auch die attributiven Daten strukturiert werden und in weiterer Folge muss festgelegt werden wer was mit den Daten darf (Wer darf Daten einbringen und darf welche Daten ablegen).
Angaben zu Hard- und Software Anforderungen müssen ebenfalls gemacht werden. (s. Scheu, 1999)

## 2.2.3 Aspekte der Datenmodellierung

„Lange Zeit galt die Verwaltung von Gebäuden innerhalb eines Informationssystems als zu komplex und damit als zu kostspielig. (...) Angesichts komplexer Zusammenhänge gibt es immer noch einige ungelöste Informationstechnische Probleme.“ (Zippelt, 1999)

### *2.2.3.1 Anforderungen an das Datenmodell*

Das Datenmodell und die Datenstruktur definiert in weiterer Folge die Leistungsfähigkeit und Erweiterbarkeit des Gesamtsystems, wobei nach Zippelt (1999) folgendes zu beachten ist:

- Das Datenmodell muss so aufgebaut sein, dass sich möglichst einfach Verknüpfungen zu anderen Datenmodellen durchführen lassen.
- Die Verknüpfungen möglichst leicht zu ändern sind und sich der fortgeschrittenen Technik und Nutzung des Gebäudes anpassen können.
- Die Struktur eines GebIS so aufgebaut ist, dass die bereits angesprochenen Fachschalen schrittweise eingeführt werden können,
- einzelne Dienstleistungen nicht durch fehlende Schnittstellen oder falsch konzipierte Datenstrukturen von der Nutzung des GebIS ausgeschlossen sind.

Aus diesem Grund muss vor der Definition des Datenmodells die Zielsetzung des GebIS feststehen und ein Anforderungsprofil erarbeitet werden, wobei folgende Punkte wesentlich sind:

- Wie werden der Gebäudebestand bzw. die einzelnen Gebäude gegliedert ?
- Welche Sachdaten müssen erhoben werden ?
- Welche zusätzlichen Fachdaten sollen integriert werden ?

Auf Basis dieser Informationen kann dann erst eine Analyse der Datenerfassungsmöglichkeiten und Kostenabschätzung erfolgen.

Da ein GebIS eine „räumliche Schnittstelle“ zwischen verschiedenen Fachdaten und Dienstleis-

tungen darstellt muss wie bereits erwähnt die Anforderung eines offenen Systems mit einheitlichen Schnittstellen erfüllt werden. Als einheitliche Schnittstelle hat sich unter anderem die Datenbankabfragesprache SQL (Structured Query Language; siehe Kapitel 3.4 - Datenbanken) etabliert. (s. Nutto (1999), Zippelt (1999))

#### *2.2.3.2 Allgemeine Datenmodellierung*

In einem GebIS werden in der Regel folgende relevante Objektgruppen berücksichtigt (Zippelt, 1999):

- Gebäudeform: Geometrie+Sachdaten
- Inventar: Geometrie+Sachdaten
- Gebäudetechnik: Geometrie+Sachdaten
- Benutzer: Sachdaten

Diese Objektgruppen können wieder aufgegliedert werden – ein Teil der Gebäudetechnik wäre zum Beispiel das EDV-Netzwerk.

Die Objektgruppen können Verallgemeinert und zu Gebäudeinformationen und Organisationsstrukturen zusammengefasst werden, wobei man unter Gebäudeinformationen die Geometrie, Beschreibungen, Nutzungen usw. versteht.

Die Komponenten eines Gebäudes lassen sich somit über den räumlichen Zusammenhang (Gebäude ▸ Etage ▸ Raum ...) und den organisatorischen Zusammenhang (Abteilung ▸ Person ...) einordnen.

Gebäude sind dreidimensionale Objekte, sodass sich die Frage nach einer dreidimensionalen Datenerfassung stellt, wobei prinzipiell drei Arten der Integration möglich sind:

- Höhe als Attribut (Sachdatenabsorption)
- Mittels einer definierten Schnittstelle (zum Beispiel DEM)
- Vollständige Integration der dritten Dimension

Die dritte Variante kann einstweilen mit CAD-Systemen realisiert werden, wobei die 3-Dimensionalität eine erweiterte Objektbildung und Topologie erfordert. Zippelt (1999) fasst wie folgt zusammen: „Die Verwirklichung eines echten dreidimensionalen GebIS ist wegen technischer und konzeptioneller Probleme, der Komplexität und der daraus entstehenden Kosten derzeit noch Gegenstand von Forschungsarbeiten.“

Der Faktor Zeit und Kosten führt dazu, dass meist die vorhandenen 2D-Pläne (analog oder digital) als Basis dienen und damit auch bestehende auf den Plänen basierende Arbeitsweisen beibehalten werden können.

Die Tatsache, dass Gebäude einem stetigen Wandel unterworfen sind führt insbesondere bezüglich der Geometrie (bauliche Veränderungen) zu einer Erweiterung des statischen Modells:

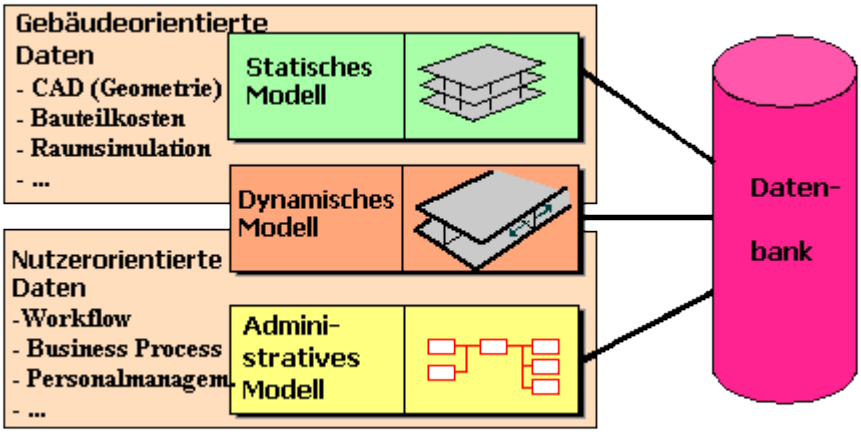


*Abbildung 7: Erweitertes Modell (aus: Schmidt-Haunschild, 1997)*

Das statische Modell enthält die Primärstruktur des Gebäudes während im dynamischen Modell die veränderbaren Parameter des Gebäudes wie der Positionierung flexibler Trennwände etc. integriert sind. Das administrative Modell besteht aus der eigentlichen Gebäudeverwaltung und den strukturellen Prozessen.

„Die Datenbank wäre dann als computerinternes Modell des Gebäudes die Grundlage für ein optimales Facility Management, das über den gesamten Lebensprozess des Gebäudes in stets aktualisierter Form vorliegt.“ (Zippelt, 1999)

Letztendlich sollte ein GebIS wie bereits erwähnt zu einem integrierten System mit Schnittstellen zu Fachanwendungen führen, wobei die Einführung in der Regel stufenweise erfolgt.

### *2.2.3.3 Objektbildung*

Objekte sind reale Gegenstände des Gebäudes, die sich aus Geometrie und Attributen (Sachdaten) zusammensetzen, dies können technische Daten, Objektzustände etc. sein, wobei gleichartige Objekte in Gruppen (Klassen) zusammengefasst werden (zum Beispiel Raumnutzer/Personen).

In der Literatur werden zwei übliche Ansätze beschrieben (z. Bsp. Zippelt (1999), Nutto (1999)):

- Objektstrukturierter Ansatz: Objekte werden hierarchisch eingeordnet, wobei lediglich aber in der Definitionsschicht der Objektarten die Sachdaten zugeordnet werden. Übergeordnete Ordnungsschichten kennen keine Eigenschaften und Methoden und dienen lediglich der Unterteilung. Dieser Ansatz ist besonders geeignet für den Einsatz von relationalen Datenbanken mit der tabellarischen Abspeicherung der Sachdaten.
- Objektorientierter Ansatz: Die bisherigen Datenbankmodelle werden mit objektorientierten Kon-

zept vereinigt. Zu diesen zählen die objektorientierte Klassenbildung, die Datenkapselung, die Erweiterbarkeit, die Vererbung und die Polymorfie. Im Gegensatz zur objektstrukturierten Modellbildung erlauben die Vererbungsmechanismen eines objektorientierten Ansatzes die Daten und Verhaltensweisen einer übergeordneten Objektklasse an eine untergeordnete Objektklasse weiterzugeben (z. Bsp.: Gebäude ▸ Etage ▸ Raum)

Die Ansätze und die verwendete Software stehen in einem gegenseitigen Abhängigkeitverhältnis: Nur Software mit Unterstützung des objektorientierten Ansatzes ermöglicht auch eine entsprechende Modellierung und Umsetzung; das derzeit meist verwendete Modell stellt die Kombination eines CAD-Systems mit einer relationalen Datenbank dar, wobei häufig einfache Datenbanken zum Einsatz kommen (z. Bsp.: dbase-Tabellen, MS-Access) bei denen das Thema Sicherheit (Zugriffsrechte etc.) zu kurz kommt. (s. Nutto, 1999)
Laut Nutto (1999) zeigt die Praxis, dass in der Wirtschaft das ältere relationale Modell meist verwendet wird.
Adäquat den Entwicklungen in der Geographischen Informationstechnologie wird am objektorientierten Modell geforscht, bei dem Geometrie, attributive Beschreibungen und topologische Zusammenhänge in einer Datenbank verwaltet werden und keine Verbindung zwischen Geometrie und Attributdaten durch Zeiger hergestellt werden muss.
Kritisch betrachtet müsste diese Forderung im wesentlichen an die Hersteller von CAD-Systemen gerichtet werden, denn objektorientierte CAD-Systeme mit Ausrichtung der Daten u.a. auf eine Nutzung für GebIS, könnten in der Zukunft eine direkte Nachnutzung der Planungsdaten aller Beteiligten im laufenden Betrieb ermöglichen.

### 2.2.4 Komponenten eines GebIS

Wie aus den vorhergehenden Kapiteln bereits hervorgeht besteht ein GebIS mindestens aus einer Software, die sich für die geometrischen Daten verantwortlich zeigt (=Basissoftware; z. Bsp. AutoCAD), und einer Datenbank, wobei hier einfache Desktop-Systeme (z. Bsp.: Access) und Server-Datenbanken mit Multiuserfähigkeit zum Einsatz kommen.
Auf Basis dieser Software werden die Objekte gebildet (s. Kapitel 2.2.3), wobei eine Kommunikationsschicht notwendig ist; auf dieser wiederum können Funktionen und Methoden definiert werden, die über eine Benutzeroberfläche zugänglich sind.
Die Funktionsschicht sollte folgende Funktionen und Methoden über die Benutzeroberfläche zugänglich machen (s. Scheu, 1999):

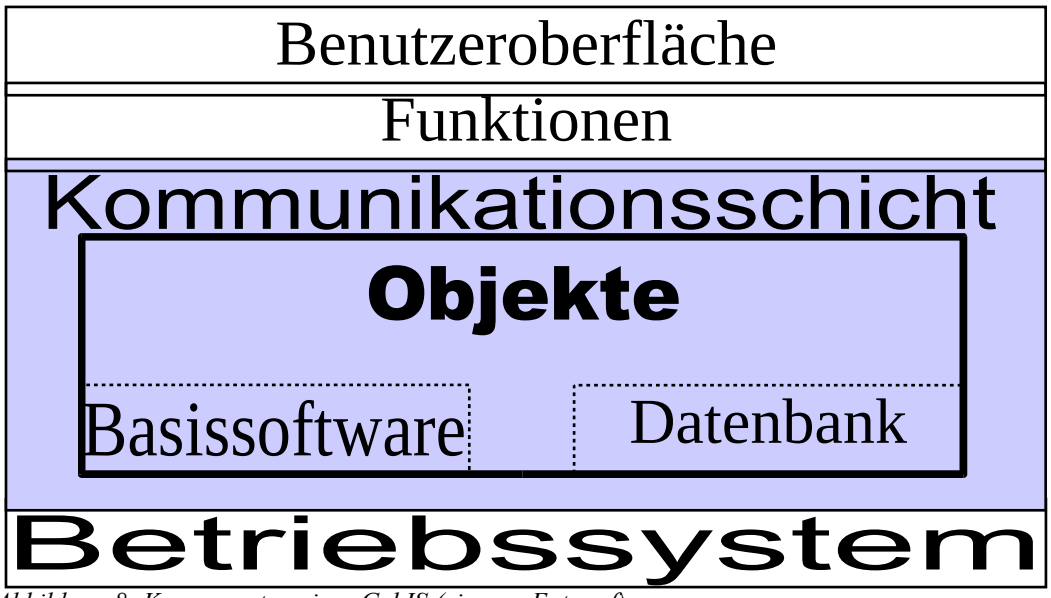


*Abbildung 8: Komponenten eines GebIS (eigener Entwurf)*

- Erfassung
- Auswertung (Abfragen)
- Veränderungen

Die Funktionen müssen den Anforderungen des Anwenders/der Anwender entsprechen (s. Zielvorstellungen). Eine wesentliche Anforderung moderner Systeme ist die Multiuserfähigkeit, wobei die Datenintegrität nicht verletzt werden darf.

Die Kommunikationsschicht darf keine Insellösung sein, sondern muss auf einer Schnittstelle beruhen, die einen Datenaustausch mit externen Daten ermöglicht – ein Beispiel ist SQL (s. Kapitel 3.6.4).

Die Benutzerschnittstelle stellt das Bindeglied zwischen dem System an sich und dem Nutzer dar, der möglichst bequem Zugriff auf die Systemfunktionen haben soll, wobei der Anwender bekannte Techniken anwenden können sollte (z. Bsp. Window-Techniken, Formulare), durch die Prozesse geführt werden sollte und eine konsistente Datenmanipulation gewährleistet wird.

### 2.2.5 Beispiel

Gebäudeinformationssysteme sind meist interne Lösungen, die dem Außen stehenden nicht zur Verfügung stehen, da es sich oft um Expertensysteme für bestimmte firmeninterne Abteilungen handelt und manchmal die eingesetzte Technik keine Anbindung an das Internet ermöglicht bzw. nicht erwünscht ist.

Als Beispiele wird aus diesem Grund nur das GebIS eines Universitätsinstituts basierend auf Literaturangaben und eine interessante Sonderlösung auf Basis eines Map-Servers präsentiert:

Zippelt (1999) stellt das GebIS am Institut für Geodäsie der Univ. Karlsruhe (GIK) vor. Der Wunsch nach einem GebIS ging dort auf die immer komplexer werdende Verwaltung des wachsenden Rechnerbestandes zurück. Die Umsetzung am GIK erfolgte im mehreren Schritten – teilweise in Form von Diplomarbeiten – wobei als Softwarebasis AutoCAD14 und MS-Access dienen, als Schnittstelle dient ODBC[3]. Die Sachdatenschwerpunkte stellten das Rechnernetz und die Raumnutzer dar. Aufgrund der Einschränkungen von MS-Access hat man auf einer Server-Datenbank (Oracle) umgestellt.


**Geodätisches Institut Karlsruhe**

*Report run on: Februar 3, 1999 5:33 PM*

| Name | Vorname | Telefon | Email Adresse | Raumnr |
|---|---|---|---|---|
| Barth | Michael | 2308 | michael@gik.uni-karlsruhe.de | -121 |
| Bigott | Marlene | 2305 | bigott@gik.uni-karlsruhe.de | 042 |
| Bracko | Diana | 3668 | bracko@gik.uni-karlsruhe.de | 111 |
| Dinter | Georg | 2303 | dinter@gik.uni-karlsruhe.de | 032 |
| Heck | Bernhard | 3674 | heck@gik.uni-karlsruhe.de | 112 |

*Abbildung 9: Report des GebIS am GIK (aus: Zippelt, 1999)*

Eine Sonderform eines Gebäudeinformationssystems findet man unter http://www.webgis.de/teleregion – es handelt sich um ein Informationssystem für Messen, welches einen Überblick über die Messestände liefert (Geometrische Daten) und zusätzliche Sachdaten zur Verfügung stellt. Technisch basiert das System auf dem UMN-Map-Server, welcher frei verfügbar ist. Der Screenshot zeigt die wesentliche Funktionalität:

3 Open Database Connectivity (s. Kapitel 3.4)

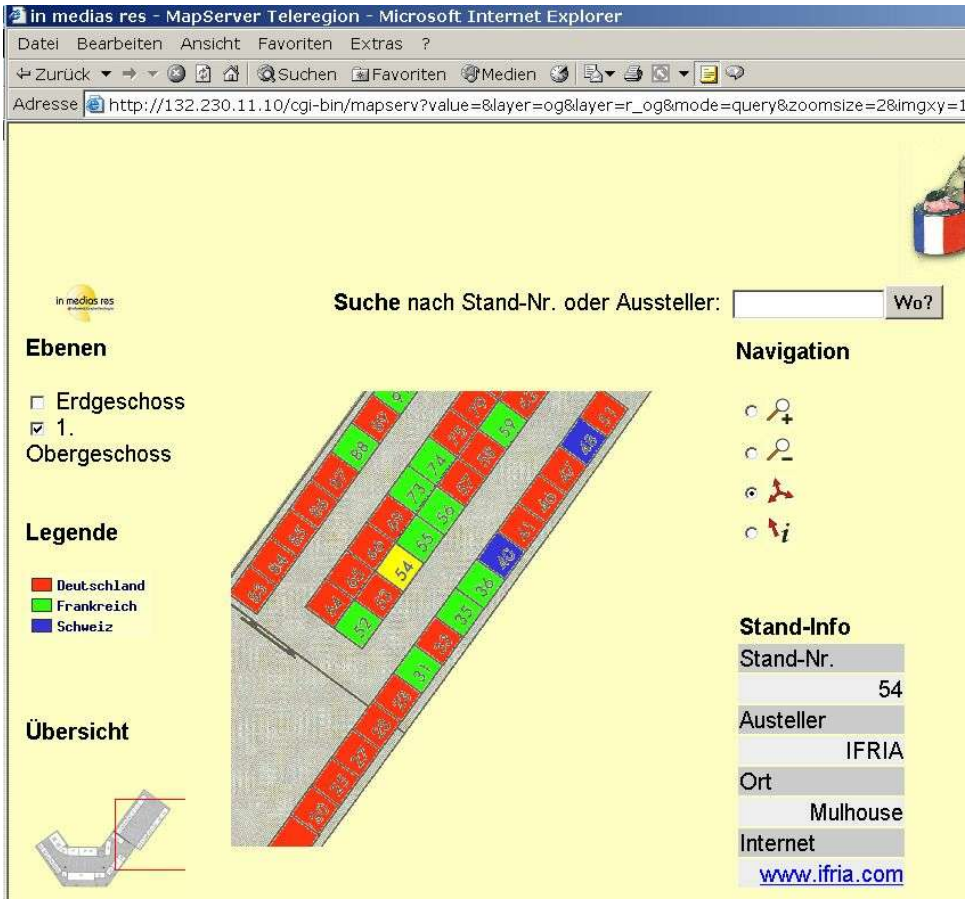


*Abbildung 10: Messeinformationssystem ( www.webgis.de/teleregion)*

Auffallend ist die schlechte Qualität der geometrischen Daten, da es sich um aufbereitete Vektordaten handelt, die vom Server als Rastergrafik (PNG-Format) ausgeliefert werden.

# 3 Eingesetzte Technik

## 3.1 Einführung

In diesem Kapitel werden die „Techniken“ vorgestellt, die für den Prototyp (s. Kapitel 4) eingesetzt wurden, wobei neben der Theorie auch praktische Beispiele einen Einblick bieten sollen.
Grundlage jeder Webapplikation ist die Client-Server Technik, wobei ein oder mehrere Server (Webserver – s. Kapitel 3.5) die Anfragen des Clients abarbeiten und die gewünschte Seite ausliefern.

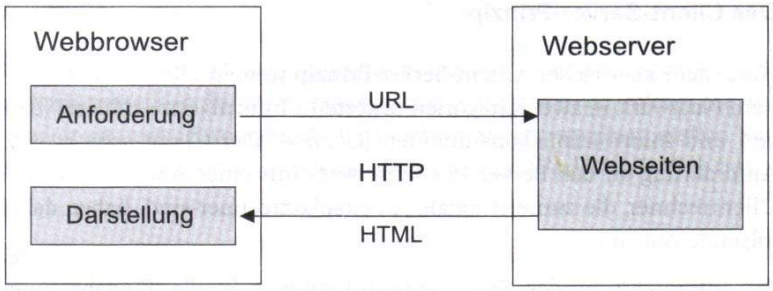


*Abbildung 11: Client-Server Prinzip (aus: Däßler, 2001)*

Der Datenaustausch zwischen Client und Server wird über Protokolle geregelt. Im Internet wird als Basisprotokoll TCP/IP verwendet. (s. Däßler (2001), Münz (2001))
Der populärste Internetdienst ist das WWW (**W**orld **W**ide **W**eb), wobei der Zugriff auf die Informationen auf Seite des Clients über den Web-Browser erfolgt. Die Datenübertragung zwischen Client und Server erfolgt über das Kommunikationsprotokoll HTTP (**H**yper**T**ext **T**ransfer **P**rotokoll).
Die Kapitel Webserver, Serverdatenbanken und PHP werden sich genauer mit der Technik befassen, im Kapitel 3.2 wird kurz auf HTML, der „lingua franca“ des Web, eingegangen.
Kapitel 3.3 widmet sich dem neuen Vektorformat SVG, welches sich insbesondere im Web-Mapping bereits breiter Unterstützung erfreut.
Mit den Kapiteln Serverdatenbanken, PHP und Javascript präsentieren sich wesentliche Techniken für die Erstellung dynamischer Inhalte und Informationssysteme.

## 3.2 Hypertext Markup Language (HTML)

HTML ist, wie es Münz (2001) formuliert, die „lingua franca“ des WWW, die einfach zu erlernen ist, jedoch ist fast kein namhaftes Web-Angebot in der Lage fehlerfreies und Standard-konformes HTML auf seinen Web-Seiten zu realisieren.
Zuerst sollte die Frage geklärt werden was HTML eigentlich ist:
„HTML ist eine Sprache zur Strukturierung von Texten, wobei aber auch die Möglichkeit besteht,

Grafiken und multimediale Inhalte in Form einer Referenz einzubinden und in den Text zu integrieren. Mit HTML können Sie Überschriften, Textabsätze, Listen und Tabellen erzeugen. Sie können anklickbare Verweise auf beliebige andere Web-Seiten oder Datenquellen im Internet erzeugen. Nicht-textuelle Inhalte können Sie wie bereits erwähnt referenzieren. Sie können Formulare in den Text integrieren. Und last but not least bietet HTML Schnittstellen für Erweiterungssprachen wie CSS Stylesheets oder JavaScript an, mit deren Hilfe Sie HTML-Elemente nach Wunsch gestalten und formatieren oder Interaktion mit dem Anwender realisieren können." (Münz, 2001)
Zusammengefasst kann man HTML einfach als „Seitenbeschreibungssprache" bezeichnen, die über SGML (Standard Generalized Markup Language) definiert ist. (s. Waldik, 2001)
HTML stellt somit das wesentliche Grundgerüst zur Erstellung des Interfaces zu Web-Mapping Produkten und webbasierten Informationssystemen dar. Waldik (2001) bezeichnet HTML als „den heute gängigen technischen Rahmen zur Implementierung von Webseiten".
Rein in HTML geschriebene Webseiten sind vom Erscheinungsbild her gesehen statisch, erst Erweiterungen wie CSS (Cascading Style Sheets) und Javascript haben dynamische Seiten ermöglicht. Mit der Erweiterung der Standards haben sich insbesondere clientseitig Probleme bzgl. der Browser ergeben, die die Standards nicht immer voll unterstützen, sodass teilweise Lösungen für die einzelnen Browser mit ihren Eigenheiten entwickelt werden mussten und noch müssen. (s. Gammack (1999), Münz (2001), Waldik (2001))
Da HTML sicher zu den bestdokumentiertesten Standards gehört und seit 1997 auch im Zusammenhang mit Web-Mapping in zahlreichen Arbeiten abgehandelt wird folgt nur eine kurze „Sprachübersicht" - für mehr Details sei auf SelfHTML von Münz (2001) und die Seiten des W3C (www.w3.org/MarkUp/) verwiesen.

## 3.2.1 Aufbau einer HTML-Datei und wichtige Tags – Syntax und Semantik

### *3.2.1.1 Grundgerüst einer HTML-Datei*

Prinzipiell besteht eine HTML-Datei aus einem Kopfteil (Header) und dem Körper (Body):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "http://www.w3.org/TR/html4/transitional.dtd">
<html>
<head>
<title>Text des Titels</title>
</head>
```

```
<body>

</body>
</html>
```

Die so genannte DOCTYPE-Definition (DTD) am Anfang der Datei ist nicht zwingend, jedoch gibt es neben HTML auch zahlreiche andere Auszeichnungssprachen, sodass diese Angabe am Anfang des Dokuments eine Information für den Browser darstellt, die im wesentlichen Angaben zum Typ der Auszeichnungssprache (HTML) und ihrer Version liefert.(s. Born (2001), Münz (2001))
Im Header können auch noch so genannte Meta-Daten und Verweise auf externe Dateien (Stylesheets, Skripten) stehen.
Zwischen den beiden <body>-Tags wird die eigentliche Seite mit ihren Inhalten und Formatierungen definiert.

#### *3.2.1.2 Wichtige Tags – eine Einführung*

HTML-Anweisungen (Tags) sind in der Regel wie folgt aufgebaut:

```
<Tag Attributliste> Text </Tag>
```

Dabei wird nicht zwischen Groß- und Kleinschreibung unterschieden, dennoch kann zur besseren Übersicht jeder Tag in Großbuchstaben geschrieben werden – wichtig ist nur dass jeder Tag wieder abgeschlossen wird (</Tag>) - eine Verschachtelung von Tags stellt ebenfalls kein Problem dar.
Sonderzeichen und Umlaute müssen substituiert werden, sofern in der Datei keine Angaben zum Zeichensatz gemacht werden – z. Bsp. wird Ä durch `&Auml;` ersetzt.

- Textformatierung:

  Zeilenumbrüche werden durch den Tag `<br>` definiert.

  Überschriften können über die Header-Tags definiert werden, wobei es 6 Ebenen gibt:

  ```
  <h1>Überschrift 1</h1> bis <h6> Text </h6>
  ```

  Zur Schriftgestaltung werden heute in der Regel Stylesheets eingesetzt, dennoch gibt es einfache Definitionsmöglichkeiten in HTML:

  ```
  <font size=Wert face=Wert> Text  </font>
  ```

  Absätze und ihre Ausrichtung können über den <p>-Tag definiert werden:

  ```
  <p align=left/right/center/justify> Text </p>
  ```

- Hyperlinks:

  ```
  <a href=Adresse> Text </a>
  ```

  Wichtig hierbei ist die Möglichkeit Verknüpfungsziele in einem neuen Browser-Fenster zu öffnen:

  ```
  <a href=Adresse target="_blank"> Text </a>
  ```

- Grafiken und Bilder:

Hinsichtlich der Präsentation grafischer Informationen wie Karten und Plänen stellt der <img>-Tag die Urform der Internetkartographie dar, wobei auch rezent Map-Server häufig nur einfache Rastergrafiken generieren, die ins Layout eingebettet werden.

```
<img src=Adresse width=Pixel height=Pixel alt="Text" align="top/middle/bottom"
border=Pixel>
```

Waldik(2001) beschreibt im Kapitel „Mappublishing im WWW und Clickable Image Maps" genauer die Bedeutung des Image-Tags für die Internetkartographie, wobei diese Form der einfachen Grafikeinbindung auch heute noch die dominante Form ist.

„Diesen gebräuchlichsten Formen der interaktiven Karte gemein ist die Nutzung von Rastergrafiken, wodurch sie oft zu groß, in der Qualität der Wiedergabe zu gering, kompliziert zu erstellen und nur beschränkt interaktiv sind." (Berger, 2002)

- Tabellen:

  Tabellen sind aus Zeilen und Spalten aufgebaut, bevor diese definiert werden muss mit dem <table>-Tag die Tabelle eingeleitet werden:

```
<table border=Pixel width=Pixel/%>
<tr align=left/right/center>
 <td align="..."> Text </td>
</tr>
</table>
```

  Der <table>-Tag ist ein mächtiger Tag, der zahlreiche Attribute hat – hier sei auf externe Literatur verwiesen (z. Bsp. Münz, 2001), hat aber z. Bsp. bei der Ausgabe von Statistiken einen hohen Stellenwert.

- Formulare:

  Formulare ermöglichen den Datenaustausch zwischen Client (Webbrowser) und Webserver, sodass diese einen wichtigen Beitrag zur Interaktivität darstellen – Formulardaten werden in der Regel an ein serverseitiges Programm (CGI-Programme) übermittelt und weiterverarbeitet.

```
<form action="URL/Programm" method="get/post">
<input/input type=Typ Attributliste>
</form>
```

  Auch hier sei wieder auf die diversen Referenzen verwiesen.

### 3.2.2 CSS – Cascading Style Sheets

„Mit den 1996 eingeführten Cascading Style Sheets (CSS) wurde ermöglicht, in den HTML-Seiten Formatierungsanweisungen einzusetzen. (...) Durch die Einführung von CSS erhöhten sich die Funktionalitäten und damit die Möglichkeiten der Enwicklungsumgebung für netzbasierte karto-

graphische Produkte erheblich.“ (Waldik, 2001)

CSS stellen somit eine wichtige Ergänzungssprache zu HTML dar, die eine exakte Formatierung der HTML-Elemente ermöglichen. Mit Stylesheets kann man beispielsweise definieren, dass eine Überschrift vom Typ H3 (<h3>) die Schriftgröße 14Pt hat und kursiv erscheinen soll – dies war mit reinem HTML nicht möglich. (s. Münz, 2001)

Ein weiteres Feature der CSS ist die Möglichkeit Elemente pixelgenau zu positionieren.

Stylesheets können in einer zentrale Datei (*.css) abgelegt werden und im Header einzelner HTML-Dateien referenziert werden, somit kann durch nur eine Änderung in der zentralen Stylesheet-Datei das Layout eines Webauftritts verändert werden, da diese Änderung sich auf alle Seiten auswirken die auf die CSS-Datei referenzieren.

Einen ganz wesentlichen Punkt in diesem Zusammenhang erwähnt Münz (2001):

„CSS Stylesheets unterstützen also erstens die professionelle Gestaltung beim Web-Design, und zweitens helfen sie beim Corporate Design für große Projekte oder für unternehmensspezifische Layouts.“

Bei den Cascading Style Sheets existieren wie bei HTML verschiedene Versionen, wobei im Jahr 1998 die Version 2 (CSS2) verabschiedet wurde. (s. www.w3.org/Style/CSS)

Das Prinzip der Stylesheets wird auch von SVG (s. Kapitel 3.3) unterstützt, sodass CSS auch im Zusammenhang mit SVG einen wichtigen Grundstein bilden.

Da für den Prototypen (s. Kapitel 4) nur ein einfacher Stylesheet verwendet wurde, wird im folgenden nur ein Grundlagen-Beispiel erläutert, als weiterführende Literatur sei auf Lindhorst (2001) und Münz (2001) verwiesen.

#### *3.2.2.1 CSS – ein Beispiel*

Das folgende Beispiel stammt aus Münz (2001) und wurde zum besseren Verständnis modifiziert:

```
<html>
<head>
<title>Titel der Datei</title>
</head>
<body>
<h1>&Uuml;berschrift 1. Ordnung</h1>
<p>ein normaler Textabsatz</p>
<ul>
<li>Ein Listenpunkt</li>
<li>Ein anderer Listenpunkt</li>
</ul>
</body>
</html>
```

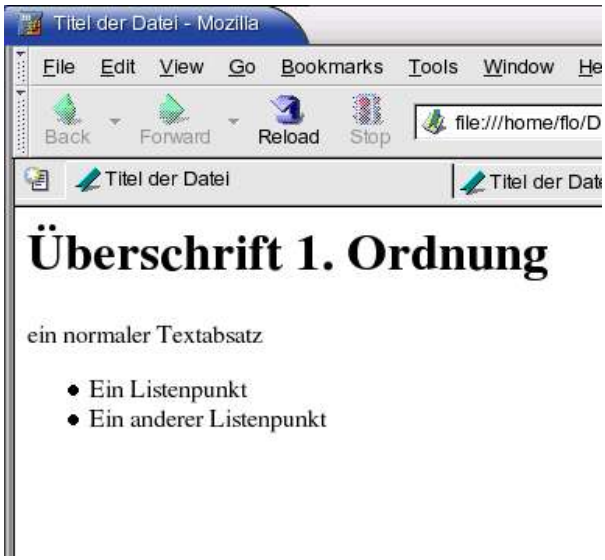


*Abbildung 12: Ohne CSS (eigener Entwurf)*

Es handelt sich hierbei um eine einfache HTML-Datei, die eine Überschrift vom Typ H1, einen Textabsatz und eine Aufzählung enthält.

Im folgenden wurde ein Stylesheet definiert, der für den Seitenhintergrund eine Farbe festlegt, des weiteren ist ein linker Rand von 100px definiert. Die Tags für die Überschrift, den Absatz (<p>) und die Aufzählung (<li>) sind ebenfalls genauer definiert.

```
<html>
<head>
<title>Titel der Datei</title>
<style type="text/css">
<!--
body { background-color:#FFFFCC;
     margin-left:100px; }
h1 { font-size:48pt;
     color:#FF0000;
     font-style:italic;
     border-bottom:solid thin black; }
p,li  { font-size:12pt;
        line-height:14pt;
        font-family:Helvetica,Arial,sans-serif;
        letter-spacing:0.2mm;
        word-spacing:0.8mm;
        color:blue; }
-->
</style>
```

```
</head>
<body>
<h1>&Uuml;berschrift 1. Ordnung</h1>
<p>ein normaler Textabsatz</p>
<ul>
<li>Ein Listenpunkt</li>
<li>Ein anderer Listenpunkt</li>
</ul>
</body>
</html>
```


*Abbildung 13: Mit CSS*

Aus den Quelltexten geht klar hervor, dass der eigentliche Inhalt der Datei, also alles zwischen den beiden <body>-Tags identisch geblieben ist und ausschließlich die CSS-Definition die visuellen Änderungen herbeigeführt hat, die Screenshots (Abbildung 12 und 13) zeigen den Unterschied deutlich.

Die Stylesheet-Definition kann auch in einer externen Datei stehen (*.css), die im Header der HTML-Datei referenziert wird: `<link rel="stylesheet" href="css.css" type="text/css">`

## *3.3 Scalable Vector Graphics (SVG)*

### 3.3.1 Vektorgrafik für das WWW – mit XML ?

In den vergangenen Kapiteln wurde bereits mehrmals auf die Problematik im Zusammenhang mit Rasterkarten hingewiesen, sodass eine auf Vektortechnik beruhende Alternative benötigt wird. Diese Alternative stellt im Web allgemein, insbesondere bei interaktiven Seiten, das Flash-Format von Macromedia (s. www.macromedia.com) dar. Schenk (2001) bezeichnet Flash als Quasistandard für die vektorielle Darstellung von Informationen im WWW.
Flash geht ursprünglich auf die Firma FutureWave zurück, die Flash im Jahr 1995 vorgestellt hat und 1996 von Macromedia aufgekauft wurde. (s. Schenk, 2001)
Flash wurde mit Schwerpunkt auf vektorielle Animationen entwickelt, wobei kleine Dateigrößen und damit eine schnelle Übertragung über das Internet das Format auszeichnen. Macromedia hat ebenfalls Autorenprogramme entwickelt, die eine einfache Erstellung ermöglichen; zur Anzeige im Browser wird jedoch ein PlugIn benötigt, welches aber laut Angaben von Macromedia bereits 1999 auf 85% der Rechner installiert war und das auf allen Plattformen (Windows, Linux, MacOS, Solaris).
Winter (2000) stellt einige Mängel fest, unter anderem dass es sich bei Flash um keinen De-Jure Standard handelt, lange Zeit schlecht dokumentiert war, sowie der Tatsache, dass es sich bei Flash um ein proprietäres, binäres Format handelt, das dem OpenSoure-Gedanken widerspricht.

Das Fehlen eines offenen Vektorformats für das Web wurde erkannt und führende Hersteller, darunter Adobe, Microsoft, Corel, IBM usw., und das W3C haben schließlich am 11. Februar 1999 die erste Arbeitsgrundlage (Working Draft) veröffentlicht. Die SVG-Arbeitsgruppe wird stark von Adobe beeinflusst, nicht zuletzt wahrscheinlich deshalb, da man Macromedias Flash nichts entgegen zu setzen hatte und mit dem Produkt GoLIVE den aktuellen Möglichkeiten von Flash hinterherhinkte. (s. Krüger, 2000)
SVG hatte jedoch 'Konkurrenz' aus dem Hause Microsoft, denn der Softwareriese hat zusammen mit Office 2000 ein eigenes Vektorformat namens VML (Vector Markup Language) auf den Markt gebracht und der Internet Explorer 5 hatte auch gleich die Rendering Engine integriert, konnte VML also PlugIn-unabhängig darstellen. VML stellt jedoch ein auf die MS-Welt beschränktes Format dar, welches sich sonst nicht durchsetzen konnte. Microsoft beteiligt sich an SVG, doch bis dato sind in Microsoft Office (Stand: Office XP) keine SVG-Export/Import Möglichkeiten vorhanden, das OpenSource Office-Paket OpenOffice besitzt bereits einen SVG-Exportfilter für Zei-

chendokumente und Präsentationen (Stand: OpenOffice 1.0.1 – s. www.openoffice.org).
Einen Überblick über Vektor-2-D-Beschreibungssprachen bietet unter anderem Grolig (2001).
SVG Version 1 wurde schlussendlich im September 2001 vom W3C verabschiedet und damit war ein offenes Format für 2D-Vektorgrafiken, das unabhängig von der Skalierung dieselbe Qualität beibehält, geboren – Version 1.1 (www.w3.org/graphics/svg) folgte am 30. April 2002. (s. Behme (2002), Berger (2002))
Krüger (2000) bezeichnet SVG außerdem als „vom Funktionsumfang her viel mächtiger ausgelegt als Flash".
Im folgenden soll nun geklärt werden, was SVG genau ist, welche wichtigen geometrischen Grundelemente es gibt, wie SVG in Internetseiten eingebettet wird und wie man interaktives SVG erstellt. Auch der Stand der Unterstützung in Softwareprodukten, im speziellen in der Geoinformatik, wird angesprochen.

### 3.3.2 Was ist SVG ? Wie funktioniert SVG ?

**S**calable **V**ector **G**raphics ist ein vom W3C beschlossener Webstandard, der wie alle neuen Standards auf XML (E**x**tensible **M**arkup **L**anguage) beruht.
XML wurde 1996 vom W3C ins Leben gerufen, hauptsächlich um den begrenzten Möglichkeiten von HTML entgegen zu wirken. XML beruht auf der Idee einer 'sauber' strukturierten Sprache, die sich auf die eigentliche Idee von SGML (**S**tandard **G**eneralized **M**arkup **L**anguage) beruft. (s. Grolig (2001), Born (2001))
Um SVG zu verstehen, muss ein Grundverständnis für XML vorhanden sein, sodass zuerst einige Begriffe angesprochen werden müssen:

- DTD: Document Type Definition – Definition für die Auszeichnung einer XML-Quelldatei, die neben den Tags auch deren Interpretation enthält. XML Dateien können ihre DTD in sich oder aber einen Verweis auf eine externe DTD enthalten. (s. www.networds.de)
- DOM: Document Object Model – Das DOM ist eine W3C-Spezifikation, die eine hierarchische Ordnung von Objekten und Methoden. Es gibt Objekte zum Zugriff auf die einzelnen Elemente einer XML-Datei, die mit bestimmten Methoden manipuliert werden können. (s. Spona, 2001)
- CSS: Cascading Style Sheets – Standard zur seitenunabhängigen Zuweisung von Eigenschaften an HTML- und XML-Objekte. (s. www.networds.de)

Der Unterschied zwischen HTML und XML liegt im wesentlichen daran, dass die DTD nicht im Browser implementiert ist – die Implementierung der DTD im Browser bei HTML ist auch der

Grund für die teilweise abweichende Darstellung von HTML-Seiten in unterschiedlichen Browsern. Bei XML verweist man jedoch auf eine externe DTD oder eine entsprechende eigene DTD. Wichtig hierbei ist, dass XML-Dateien nur die Informationen gliedern und keine Angaben über Layout etc. enthalten – Inhalt und Layout sind also getrennt. Die Formatierung kann zum Beispiel über CSS und XSL erfolgen. (s. Grolig (2001), Born (2001))

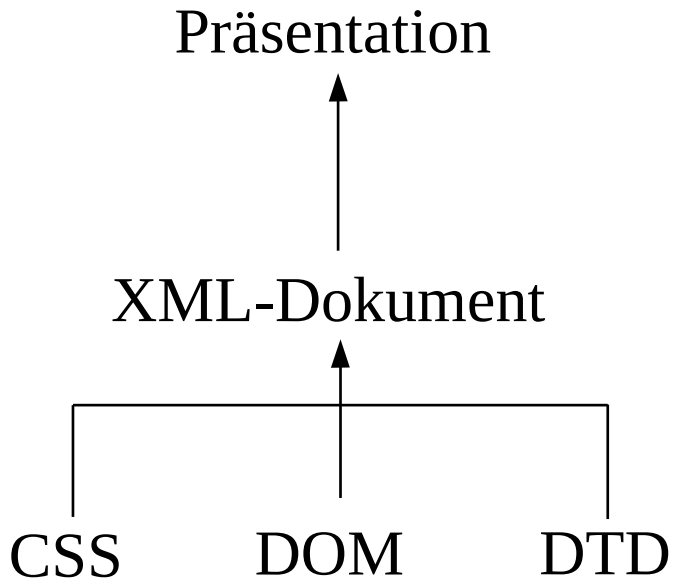


*Abbildung 14: Funktionsweise von XML (eigener Entwurf)*

Die einfachste Form einer XML-Datei stellt zum Beispiel ein Personenregister dar, wobei das folgende Beispiel eine Teilnehmerliste repräsentiert:

```
<?xml version="1.0" encoding="us-ascii"?>
<teilnehmer>
 <person id="1">
  <name>Huber Sepp</name>
  <mail>huber@huber.net</mail>
 </person>
 <person id="2">
   <name>Resi Huber</name>
   <mail>resihuber@huber.net</mail>
 </person>
 <person id="3">
   <name>Klaus Nikolaus</name>
   <mail>nikolaus@krampus.at</mail>
 </person>
</teilnehmer>
```

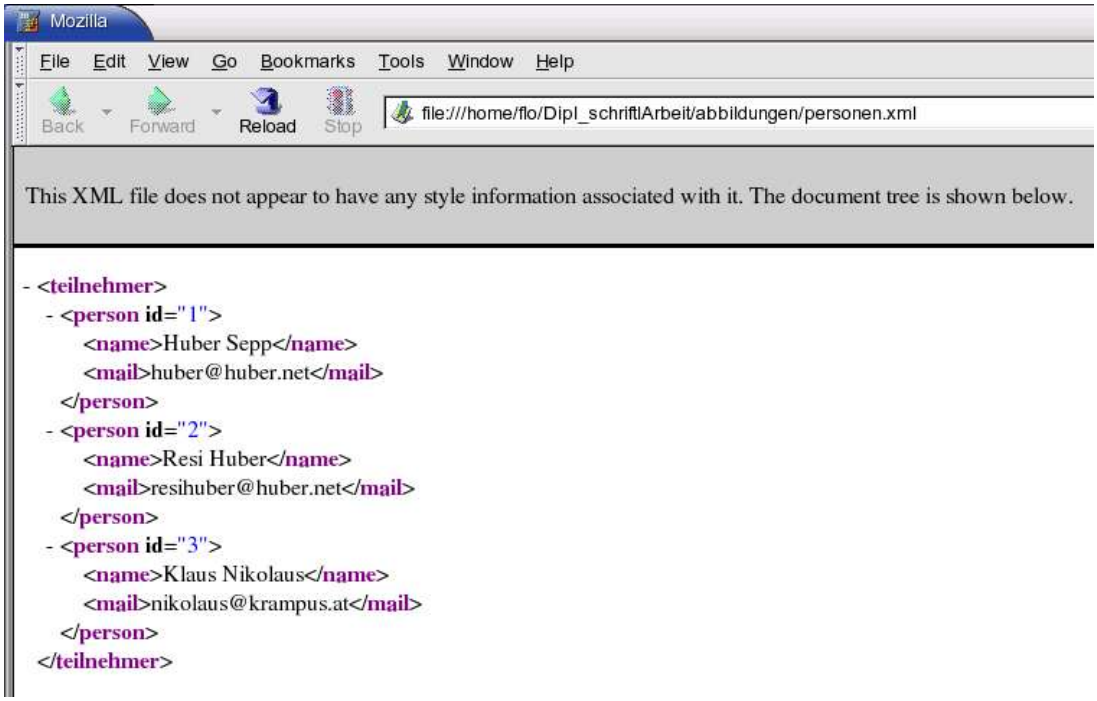


*Abbildung 15: Browserdarstellung der XML-Datei (eigener Entwurf)*

Die gängigen Browser sind einstweilen in der Lage ein solches Dokument strukturiert anzuzeigen[4], der Screenshot zeigt die hierarchische Gliederung eines XML-Dokuments – mit den „+/-“-Symbolen können die verschiedenen Hierarchien (Ebenen) eingeblendet bzw. ausgeblendet werden.
Neben SVG gibt es andere auf XML basierende Beschreibungssprachen, wie GML (Geography Markup Language), MathML, ChemML usw. (s. www.w3.org) , all diese Sprachen werden auch als Dialekte von XML bezeichnet. (s. Berger, 2002)
Ein weiteres Merkmal von XML-Dateien stellt ihre 'Lesbarkeit' dar, es handelt sich um normalen lesbaren ASCII-Text, der mit jedem einfachen Editor (z. Bsp. TextPad, Kate) bearbeitet werden kann, deshalb sind die Dateien auch von Suchmaschinen erfassbar.

SVG unterstützt alle Features die man von einem modernen Vektorformat erwartet:

- Skalierbarkeit ohne Qualitätsverlust
- Enthält alle nötigen grafischen Primitiva (Kreise, Rechtecke, ...)
- Animationen
- Skripting
- Flexible Koordinatensysteme

Ein zur Zeit noch vorhandenes Manko von SVG stellt die Verwendung von PlugIn's zur Darstellung

4 Getestet mit Internet Explorer 6.0 und Mozilla 1.3a

im Browser dar, da noch kein Browser SVG nativ rendern kann. Das Mozilla-Projekt (www.mozilla.org) arbeitet an einer direkten Unterstützung von SVG im Browser (s. www.croczilla.com). Das derzeit gängigste PlugIn wird von Adobe vertrieben und ist kostenlos erhältlich (www.adobe.com/svg) – das PlugIn kann für Windows, Linux und MacOS heruntergeladen werden. Ein weiteres PlugIn wird von Corel (www.corel.com) vertrieben, welches noch nicht die Leistungsfähigkeit des Adobe-PlugIns erreicht hat.
Alternativ können auch Stand-Alone Viewer verwendet werden, wie beispielsweise SVGView von IBM (www.alphaworks.ibm.com/tech/svgview).
Ein weiteres Problem im Zusammenhang mit den PlugIns ist die manchmal fehlende Schnittstelle zwischen PlugIn und Scripting-Engine des Browsers, sodass Skripts oft nicht fehlerfrei oder überhaupt nicht funktionieren.

### 3.3.3 Aufbau einer SVG-Datei

Eine XML-Datei und somit auch eine SVG-Datei muss zu Beginn eine XML-Deklaration enthalten, die auf die Version von XML verweist, sowie einen gültigen Verweis auf die verwendete DTD aufweisen.(s. Grolig (2001), Spona (2001))

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="300px" height="250px">
<defs> </defs>

</svg>
```

Der obige Code bildet das Grundgerüst einer SVG-Datei, ähnlich wie bei HTML gibt es einen Header und einen Body mit den eigentlichen Inhalten, der mit dem „root-Element“ <svg> eingeleitet wird und mit </svg> abgeschlossen wird. Der eigentliche Inhalt im root-Element sind dann so genannte Kind-Elemente (child-elements) mit ihren Attributen. (s. Spona (2001), Neumann und Winter (2001))
Das obige Beispiel verweist auf die SVG-DTD in der Version vom 4.9.2001, die sich auf dem Server des W3C befindet – PUBLIC gibt an, dass es sich um eine frei zugängliche DTD handelt.
Zu beachten ist, dass der Quelltext der DTD entsprechen muss, denn nur wenn die Datei „valid“ und „well-formed“ ist, wird sie vom Browser auch dargestellt. Vereinfacht ausgedrückt kann man sagen, dass die DTD bestimmt welche XML-Tags verwendet werden können, welche Attribute

diese annehmen können und welche Werttypen den Attributen zugewiesen werden können.
Das root-Element <svg> kann bereits Attribute annehmen, darunter die Angaben über die Ausdehnung der SVG-Grafik – „width“ und „height“.
Um bei der Arbeit mit SVG keine Überraschungen zu erleben, sollten im Vergleich zu HTML folgende Regeln beachtet werden (Spona, 2001):

- SVG unterscheidet Groß- und Kleinschreibung – alle Tags, deren Attribute und deren Werte werden klein geschrieben.
- Die Unterscheidung zwischen Groß- und Kleinschreibung ist auch beim Zugriff auf benannte Tags zu beachten.
- Tags müssen wie in HTML immer abgeschlossen sein, sonst ist das Dokument nicht „well-formed“.
- Zur Positionierung von Elementen werden Koordinaten verwendet – nicht wie in HTML „top“ oder Ähnliches.
- Attributwerte werden in Anführungszeichen eingeschlossen.

## 3.3.4 Sprachumfang – Wichtige Grundelemente

Ausgehend vom Grundgerüst einer SVG-Datei (s. Kapitel 3.3.3) werden nun wichtige Grundelemente vorgestellt.

### *3.3.4.1 Koordinatensysteme in SVG*

Spona (2001) trifft die Aussage „SVG bietet mehr als nur ein Koordinatensystem“ und bezieht sich damit auf die Tatsache, dass neben den Angaben „width“ und „height“ im root-Element, die die Zeichnungsfläche (=Canvas) definieren, auch noch eigene Koordinatensysteme definiert werden können – z. Bsp. über eine so genannte „viewbox“, die ein eigenes Koordinatensystem initialisiert, wobei x, y, Breite und Höhe als Attributwerte angegeben werden.

```
<svg width=" 640px" height=" 444px" viewBox="0 0 463.6 321.4">
```

Einen andere Möglichkeit bietet das „transform“-Attribut mit dem Wert „translate(tx, ty)“ - hierbei wird der Nullpunkt des neuen Koordinatensystems festgelegt (tx, ty).

### *3.3.4.2 Basis Geometrie Typen*

Als Grafiksprache muss SVG die nötigsten Primitive enthalten, ohne sie erst programmieren zu müssen – dazu gehören:

- Rechtecke
- Kreise

- Ellipsen
- Linien
- Polylinien
- Polygone
- Symbole
- Pfadelemente

Diese Basis-Elemente können auch entsprechend formatiert werden, doch muss dabei folgende Besonderheit beachtet werden:

„Über Attribute werden nur die Eigenschaften der Form übergeben werden. Es ist nicht möglich, Angaben zum Layout (Farbe, Füllung usw.) als Eigenschaften (Property) zu übergeben. Diese Angaben werden in SVG konsequent über Cascading Style Sheets erzeugt !“ (Grolig, 2001)

Der folgende Quelltext bietet einen Überblick über wichtige Basiselemente:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="300px" height="250px">
<defs> </defs>
<!-- Rechtecke -->
<rect id="rect1" x="10" y="10" width="100" height="60"
style="fill:red;stroke:black;stroke-width:4px;"/>
<rect id="rect2" x="80" y="50" rx="8" ry="8" width="100" height="60"
style="fill:blue;fill-opacity:0.45;stroke:green;stroke-width:4px;"/>
<!-- Kreis+Ellipse-->
<circle id="cir1" cx="150" cy="150" r="35" style="fill:yellow;stroke:red;stroke-width:3px;"/>
<ellipse id="ell1" cx="100" cy="150" rx="50" ry="30" style="fill:lightblue;fill-opacity:0.45;stroke:darkblue;stroke-width:3px;"/>
<!-- Linien -->
<line id="lin1" x1="0" y1="0" x2="250" y2="200" style="stroke-width:10px;stroke:grey;stroke-opacity:0.6;"/>
<!-- Gruppierte Elemente -->
<g id="gruppe1" style="fill:lightblue;stroke:red;stroke-width:3px;">
 <rect x="10" y="200" width="50" height="25"/>
 <circle cx="100" cy="225" r="25"/>
</g>
</svg>
```

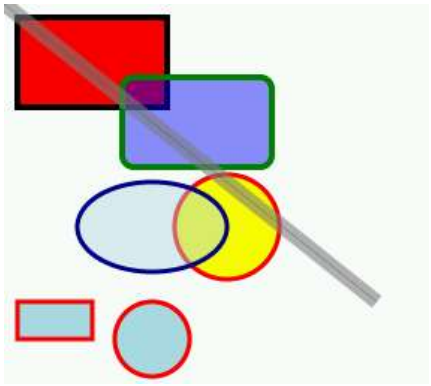

*Abbildung 16: SVG - Beispiel 1*
*(eigener Entwurf)*

Das Beispiel veranschaulicht einige wichtige Grundformen und Formatierungsmöglichkeiten, die nahezu selbsterklärend sind. Das Rechteck wird über die linke obere Ecke und die Breite und Höhe definiert, die Formatierungsangaben stehen in der „style"-Anweisung. Das zweite Rechteck besitzt durch die Angabe der „rx-" und „ry"-Werte abgerundete Ecken, die den Radiuswert in x- und y-Richtung angeben, diese Angaben begegnen einem oft bei Kurven und Kreisen (Ellipsen). Das abgerundete Rechteck verfügt über eine weitere interessante Eigenschaft: Durch die Angabe der „fill-opacity" im Style wird eine Transparenz erreicht – die mit Werten zwischen 0 und 1 angegeben wird (durchsichtig – deckend).
Aus den beiden Rechtecken wird eine weitere Eigenschaft von SVG-Dateien deutlich: Im Code weiter unten liegende Elemente liegen über den darüber stehenden – im Beispiel liegt z. Bsp. „rect2" über „rect1".
Die Tags für Kreise und Ellipsen enthalten die Angaben über Mittelpunkt (cx, cy) und Radius (r) – bei der Ellipse werden die Radien in x- und y-Richtung angegeben (rx, ry).
Linien werden durch die Koordinaten des Startpunktes (x1, y1) und Endpunktes (x2, y2) definiert.
Der <g>-Tag im Beispiel leitet eine Gruppe von Elementen ein (vgl. Gruppierungsfunktionen in Zeichenprogrammen), die 2 Elemente enthält und durch </g> wieder abgeschlossen wird. Einer Gruppe kann ebenfalls eine id zugeordnet werden und die Zuweisung einer Style-Definition, die für alle Elemente in der Gruppe gilt, ist ebenfalls möglich.

Für die Visualisierung von geographischen Informationen spielen Polygone und Pfade eine wesentliche Rolle (vgl. Bezirksgrenzen, Flüsse, ...), die in SVG mittels dem „path"-Element umgesetzt werden können; dieses erlaubt die Erstellung von offenen als auch geschlossenen Pfaden.
Mit <path>-Elementen kann prinzipiell nahezu jede Form erstellt werden, bedeutend ist dieses Element auch bei der Entwicklung von Import-/Export-Schnittstellen.(s. Grolig, 2001)

Im folgenden Beispiel werden die Elemente „polyline“ und „path“ vorgestellt:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="300px" height="250px">
<defs> </defs>
<g id="boden" style="fill:none;stroke:brown;">
 <polyline id="boden1" points="0,200 250,200"/>
 <polyline id="boden2" points="0 190 30 180 70 190 250 190"/>
</g>
<g id="waende" style="fill:none;stroke:red;stroke-width:2px;">
 <path id="w1" d="M 90 180 90 80 210 80 210 180"/>
 <path id="w2" d="M 210 150 L 210 180 L 235 180 z"/>
</g>
<path id="dach" style="fill:blue;" d="M 90 80 150 20 210 80 z"/>
<text id="pathtext" style="font-family:Times;font-size:14pt;baseline-shift:5px;">
  <textPath xlink:href="#dach">SVG oder Flash ?</textPath>
</text>
<text id="text1" x="10" y="180" style="font-family:Helvetica;font-size:16pt;fill:green;">Why not LINUX ?</text>
<use xlink:href="#text1" x="10" y="40"/>
<a xlink:href="http://www.linux.org" target="_blank">
 <text x="92" y="110" style="font-family:Helvetica;font-size:14pt;">www.linux.org</text>
</a>
</svg>
```

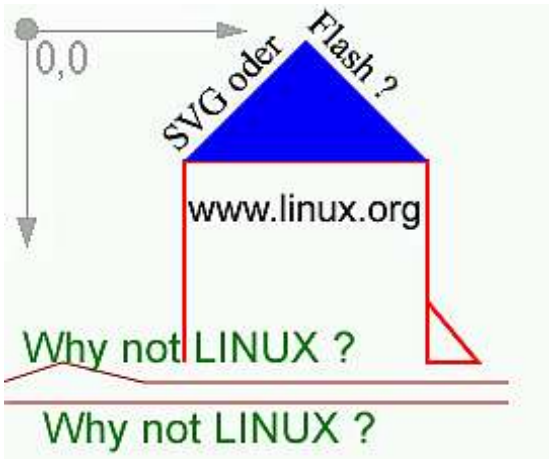


*Abbildung 17: SVG - Beispiel 2 (eigener Entwurf)*

Das Element „polyline“ definiert sich einfach über die Stützpunkte der Linie, wobei jeweils die Koordinatenpaare angegeben werden müssen – die Elemente „boden1“ und „boden2“ veranschauli-

chen die Freiheiten bei der Angabe der Koordinatenpaare, die Verwendung eines Trennzeichens trägt zur Lesbarkeit des Codes bei.

Behme (2002) bezeichnet das „path"-Element als „erheblich spannender, gegenüber den Grundformen rect, circle und polyline" und begründet dies damit, dass sich mit dem „path"-Element beliebige grafische Formen realisieren lassen. Die Primitive „rect" usw. sind im Prinzip nichts anderes als Vereinfachungen von „path" für den Entwickler.

Das wichtigste Attribut bei Pfaden stellt das „d"-Attribut („data") dar, welches die vorzunehmenden Bewegungen eines fiktiven Stiftes enthält – darin folgt dann wie und wohin sich der Stift bewegen soll. „M" bedeutet „move-to" und bewegt den Stift an die angegebene Stelle, „L" („line-to") zeichnet eine Linie zum nächsten angegebenen Stützpunkt. Der „z"-Befehl schließt einen Pfad.

Im Attribut „d" kann eine Vielzahl von Befehlen verwendet werden, folgend eine Übersicht aus Behme (2002):

| ***Befehl*** | **Bedeutung** |
|---|---|
| M 10 10 | gehe zu x=10, y=10 |
| m 10 10 | gehe zu x=jetzt+10, y=jetzt+10 |
| L 10 100 | Linie zu x=10, y=100 ziehen |
| l 10 10 | Linie zu x=jetzt+10, y=jetzt+10 |
| H 100 | horizontale Linie zu x=100, y=jetzt |
| h 100 | horizontale Linie zu x=jetzt+100, y=jetzt |
| V 100 | vertikale Linie zu x=jetzt, y=100 |
| v 100 | vertikale Linie zu x=jetzt, y=jetzt+100 |
| Q 140 150, 160 30 | quadratische Bézier-Kurve mit Anfasser (x=140, y=150) und Zielpunkt |
| q 140 150, 160 30 | dito relativ zum Bezugspunkt |
| C 150 150,180 170,300 40 | kubische Bézier-Kurve mit zwei Anfassern und einem Zielpunkt |
| c 150 150,180 170,300 40 | dito relativ zum Bezugspunkt |
| T 100 100 | quadratische Bézier-Kurve mit gespiegeltem Anfasser |
| T 100 100 | dito relativ zum Bezugspunkt (ausgehend vom letzten) |

Im Beispiel sind weitere wichtige Funktionalitäten von SVG erkenntlich, wie die Erstellung von Texten, ihre Ausrichtung an Pfaden, sowie die Verwendung von Hyperlinks.

Texte werden mit dem <text>-Tag erstellt, wobei die Position des Textes und das gewünschte Erscheinungsbild definiert werden müssen. Wichtig ist der Abschluss eines Textelements mit </text>.

Mit dem <textPath>-Tag innerhalb des <text>-Tags erreicht man eine Orientierung des Textes an dem angegebenen Element – die x- und y-Koordinatenangaben im <text>-Tag sind nicht mehr notwendig. <textPath> verweist mit „xlink:href" auf ein Element; „xlink:href" ist ein besonderes At-

tribut, da es aus einem fremden Namensraum stammt (xlink-Namensraum). (s. Spona, 2001)
Texte, aber auch geometrische Elemente können einen Hyperlink enthalten, dies wird mit <a xlink:href="Ziel"> </a> erreicht.
Oft werden Elemente wiederholt benötigt, hierbei kann <use> verwendet werden; Der <use>-Tag verweist auf ein anderes Element, welches an einer anderen Position wiederholt eingesetzt werden kann.

Die Style-Angaben können äquivalent zu HTML auch gesondert angegeben werden, in der <defs>-Section oder in einer eigenen Datei.
Eine weitere, insbesondere im GI-Bereich, interessante Möglichkeit stellt die Definition von Symbolen dar - „Das Symbol-Element kann zur Symbolisierung von Punkt-Objekten verwendet werden. Symbole definiert man am Beginn des Files in der <defs>-Section. Die Symboldefinition kann beliebigen SVG-Code enthalten." (Neumann und Winter, 2001-1)
Elemente zu verschieben, ihre Größe zu ändern, diese zu drehen usw. ist mit SVG ebenfalls mit dem „transform"-Attribut möglich.
Im folgenden Beispiel kommt eine separate Style-Definition zum Einsatz, des Weiteren wird eine Symbol-Definition und das „transform"-Attribut vorgestellt:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="300px" height="250px">
<defs>
 <style type="text/css">
   <![CDATA[
    .text1 {font-family:Helvetica;font-size:18pt;fill:red;}
    .text2 {font-family:Times;font-size:12pt;fill:black;fill-opacity:0.75;}
    .fill1 {fill:lightblue;stroke:green;stroke-width:2.5px;}
   ]]>
   </style>

 <symbol id="button">
  <circle cx="50" cy="50" r="25" style="fill:yellow;stroke:red;stroke-width:2.5px;"/>
  <circle cx="50" cy="50" r="20" style="fill:lightblue;stroke:darkblue;stroke-width:2px;fill-opacity:0.65;"/>
 </symbol>
</defs>
```

```
<use xlink:href="#button" x="50" y="50"/>
<text id="h1" class="text1" x="25" y="25">Überschrift</text>
<a xlink:href="http://geo4.uibk.ac.at/users/jurgeit" target="_blank">
 <text id="link1" class="text2" x="30"
y="45">http://geo4.uibk.ac.at/users/jurgeit</text>
</a>

<circle id="c1" class="fill1" cx="200" cy="200" r="45"/>
<use xlink:href="#c1" x="-100" y="0" transform="scale(0.5,0.5)"/>
<use xlink:href="#c1" x="-200" y="0" transform="skewX(25)"/>
</svg>
```

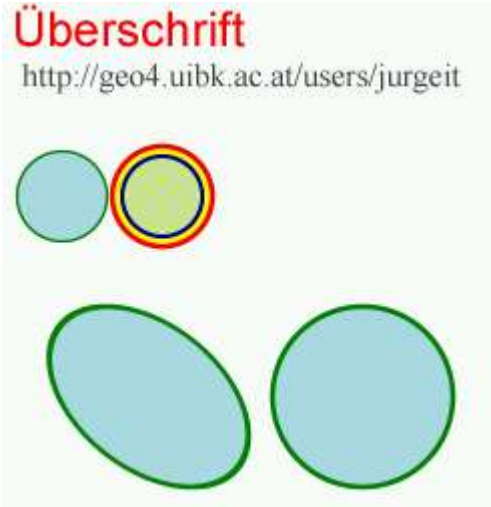


*Abbildung 18: SVG - Beispiel 3 (eigener Entwurf)*

Die in der <defs>-Section definierten Styles werden innerhalb der Elemente durch die Angabe `class='style'` referenziert. Der Vorteil liegt in der zentralen Änderungsmöglichkeit des Erscheinungsbildes – z. Bsp. auch für verschiedene Ausgabemedien (PC, PDA, ...).

Auf die definierten Symbole kann mittels des bereits bekannten <use>-Tags zugegriffen werden. Das „Transform"-Attribut wird in das gewünschte SVG-Element eingebettet. „Scale" beeinflusst die Größe des Elements – die Werte beziehen sich auf die x- und y-Richtung. „SkewX" schert das Element um den angegebenen Winkel auf der X-Achse. Weitere wichtige Werte sind „Rotate" um das Objekt zu drehen und „Translate" um den Nullpunkt eines neuen Koordinatensystems zu definieren.(s. Eisenberg (2002), Spona (2001))

Das leistungsfähigste Merkmal von „Transform" ist die Möglichkeit Transformationen über eine Matrix zu definieren: `transform="matrix (a, b, c, d, e, f)"`

Alle bereits angesprochenen Transformationen können auch mittels einer Matrix durchgeführt werden:

$$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \tan(ax) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \tan(ay) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

*Abbildung 19: Rotate, SkewX und SkewY als Matrix (aus: Eisenberg, 2002)*

Diese Funktionalität von SVG ist aus geographischer Sicht im Zusammenhang mit Koordinatensystemen und affinen Transformationen hinsichtlich der Überführung zwischen verschiedenen kartesischen Koordinatensystemen von besonderer Bedeutung.

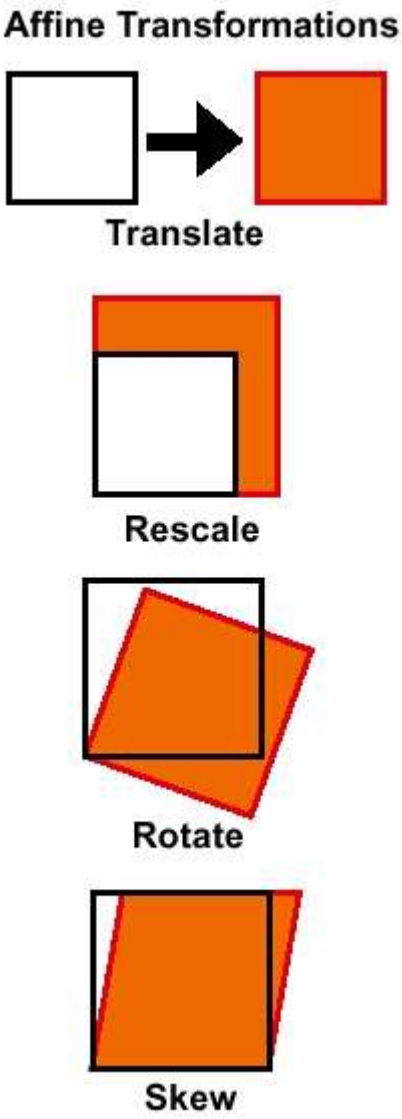


*Abbildung 20: Affine Transformationen*
*(aus: Microimages, 2000)*

Diese Beispiele repräsentieren nur einen geringen Teil der Möglichkeiten von SVG, den gesamten Sprachumfang kann man in der Spezifikation des W3C (http://www.w3.org/TR/SVG/) erfahren.
Neben den statischen Fähigkeiten besitzt SVG auch die Möglichkeit Elemente zu animieren und mittels Skriptsprache zu manipulieren – s. Kapitel 3.3.7.

### 3.3.5 SVG in HTML einbetten

Es ist derzeit (noch) nicht sinnvoll Webseiten komplett in SVG zu erstellen, nicht zuletzt aufgrund der PlugIn-Problematik und Problemen mit manchen Browsern (s.u.), sodass SVG-Grafiken als besonderes Feature in HTML-Seiten eingebunden werden – z. Bsp. zur kartographischen Visualisierung.

Die Einbettung von SVG-Grafiken in HTML-Seiten kann im Prinzip über zwei Möglichkeiten erfolgen. Entweder mit dem in der HTML-Spezifikation vorgesehenen <object>-Tag oder mit dem veralteten <embed>-Tag. (s. Behme (2002), Eisenberg (2002), Spona (2001)).

Mit dem <embed>-Tag kann man SVG-Grafiken mit folgendem Syntax einfügen:

```
<EMBED src="quelle.svg" type="image/svg+xml"/>
```

Neben der SVG-Datei (src=...) sollte auch angegeben werden um welchen Dateitypen es sich handelt.

Der <embed>-Tag sieht auch einen Alternativtext vor, sollte der Browser des Clients PlugIns nicht unterstützen; dazu muss lediglich innerhalb des <embed>-Tags ein <noembed>-Tag eingefügt werden.

Im <embed>-Tag kann ebenfalls ein Name für das einbettete SVG-Objekt vergeben werden (`name="abc"`), über den das Objekt im DOM ansprechbar ist; außerdem ermöglichen die Angaben `width="Wert"` und `height="Wert"` die Dimension des Objekts festzulegen.

```
<embed name="color" width="325" height="250" src="color.svg"
type="image/svg+xml" pluginspage="http://www.adobe.com/svg/viewer/install/"/>
```

Das Attribut „pluginspage“ ermöglicht den Verweis auf die Download-Seite des benötigten PlugIns.

Laut Spona(2001) wird die Verwendung des <embed>-Tags empfohlen, obwohl dieser keinem W3C-Standard entspricht.

Die zweite Möglichkeit stellt der <object>-Tag dar:

```
<OBJECT type="image/svg+xml" data="color.svg" name="color" width="325" height="250">
Kein PlugIn installiert !!!
</OBJECT>
```

Die Attribute „type“ und „data“ müssen definiert sein, die anderen Angaben sind alternativ. Zwischen dem Anfangs-Tag und dem Abschluss-Tag kann ein Alternativinhalt bei Nichtvorhandensein des PlugIns angegeben werden – neben dem einfachen Text wie im Beispiel können auch Hyperlinks und Grafiken angegeben werden, oftmals wird eine Rastergrafik der SVG-Grafik ausgegeben.

Moderne Browser können mit beiden Formen der Einbindung umgehen (z. Bsp. IE6 oder Mozilla), Opera 6 ignoriert die <object>-Anweisung, Netscape 4.x weist teilweise ebenfalls dieses Verhalten auf. „Spätestens hier fangen die Schwierigkeiten für Benutzer von Mozilla 1.0 oder höher an. Weder das in der HTML-Spezifikation vorgesehene object noch das veraltete embed funktionieren; stattdessen verabschiedet sich der Browser ins Nirwana.“ (Behme, 2002)

Dem Hinzuzufügen ist nur noch, dass die Abstürze nur bei installiertem PlugIn auftreten – die Tags

zur Einbettung funktionieren schon.
Die Entwickler des KDE-Browsers Konqueror (Linux), dessen Rendering-Engine seit kurzem auch Apple für seinen Browser Safari verwendet, arbeitet an einer direkten SVG-Unterstützung im Browser – s. http://svg.kde.org .

## 3.3.6 Stand der Implementierung

Neumann und Winter (2001-1) stellen fest, dass es viele Wege gibt um SVG-Dateien zu erstellen, zu konvertieren und zu editieren.
Neben der Möglichkeit SVG-Dateien mit einem beliebigen Text-Editor zu erstellen bzw. zu bearbeiten, bieten zahlreiche Programme die Möglichkeit SVG-Dateien zu erzeugen. Aufgrund der Tatsache, dass es sich bei SVG um ein offenes Format handelt wird SVG auch zunehmend eine Bedeutung als Austauschformat haben. Um mit Flash konkurrieren zu können müssen noch zahlreiche Autorenprogramme (Entwicklungsumgebungen) auf den Markt kommen. Im folgenden werden verschiedene Programme erwähnt, die eine SVG-Implementierung aufweisen, wobei kein Anspruch auf Vollständigkeit erhoben wird – auf der Homepage des W3C ist eine Liste mit Programmen mit SVG-Unterstützung verfügbar (s. http://www.w3.org/Graphics/SVG/SVG-Implementations).
Kapitel 3.3.6.1 wird sich auf Standard-Software beziehen, Kapitel 3.3.6.2 die Möglichkeiten in GIS-Produkten erläutern – jeweils mit Stand März/April 2003.

### *3.3.6.1 Implementierung in Standard-Software*

Eines der wichtigsten Features in diesem Zusammenhang stellt der Export aus Grafik-Software dar. Hier hat Adobe bereits die meisten seiner Produkte soweit, dass SVG exportiert werden kann bzw. tlw. auch importiert werden kann: Adobe Illustrator, Adobe GoLive, Adobe InDesign und Adobe FrameMaker.
Corel bietet ebenfalls in zahlreichen seiner Produkte SVG-Filter an, darunter in Corel-Draw, Corel Smart Graphics Studio und Corel Designer – Corel bietet seit Herbst 2002 ebenfalls einen SVG-Viewer an, der als Browser PlugIn funktioniert.
Auch in der OpenSource-Szene bieten zahlreiche Programme SVG-Unterstützung, darunter Kontour (Linux, KDE), Sketch (Linux) und das Office-Paket OpenOffice.org (Windows, Linux; Mac) ab der Version 1.0.1. (s. Kukofka (2003), Neumann und Winter (2001-1))
Microsoft hat in sein Office-Paket keine SVG-Unterstützung integriert (Stand: Office XP), inwieweit sich das ändern wird ist noch unklar, denn schließlich hat man mit Office 2000 den eigenen bereits erwähnten Web-Vektor Standard VML eingeführt.

Die Firma Jasc (www.jasc.com) hat als einer der ersten Hersteller eine Art Autorensystem für SVG namens WebDraw entwickelt. WebDraw ermöglicht die Geometrie interaktiv zu erstellen und man kann neben dem WYSIWYG-Modus auch im Quelltext arbeiten; Animationen werden ebenfalls unterstützt. (s. Neumann und Winter, 2001-1)

Einen weiteren Ansatz stellen Konverter dar. Dies sind kleine Programme bzw. Skripts, oft in Perl, Python oder PHP geschrieben, die bestehende Formate in SVG konvertieren. Neumann und Winter (2001-2) erwähnen Konverter für SWF, pdf/ps und wmf. Es gibt auch noch zahlreiche andere, insbesondere auch für Formate im GI-Bereich (s. Kapitel 3.3.6.2).

Einen interessanten Ansatz hat die Firma Software Mechanics verfolgt, deren SVG-Druckertreiber „SVG-Maker“ ähnlich wie Adobe's Distiller es Applikationen erlaubt in SVG-Dateien zu drucken. Dieser Ansatz funktioniert jedoch nur für statische SVG-Dateien.

### *3.3.6.2 Implementierung in GI-Produkten*

Eine direkte Unterstützung von SVG seitens der gängigen GIS-Programme ist nicht bzw. nur rudimentär vorhanden. TNTmips von der Firma Microimages (www.mircoimages.com) bietet seit der Version 6.7 einen SVG-Export an, der aber leider in der Lite-Version nicht vorhanden ist. Den Beispielen auf der Homepage von Microimages nach, dürfte es sich um einen SVG-Export der Layouts handeln, ist also eher als pdf-Ersatz zu sehen (Stand: März 2003).

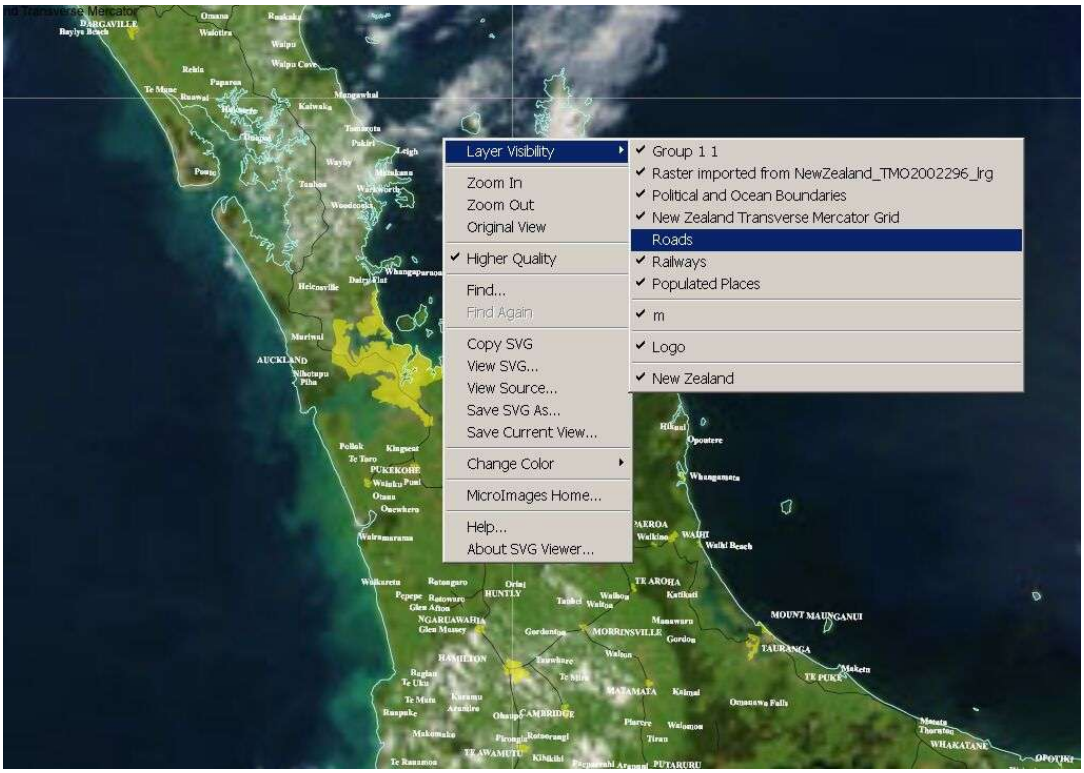


*Abbildung 21: SVG-Export von TNTmips (www.microimages.com)*

ESRI bietet in keinem seiner Produkte direkte SVG-Unterstützung (Stand: ArcGIS 8.2), im weit verbreiteten und 'alten' ArcView 3.2 natürlich auch nicht. Für die ESRI-Produkte gibt es aber zahlreiche freie und kommerzielle Lösungen um SVG-Dateien zu erstellen. Kostenlos ist das Avenue-Skript „shp2svg“ (www.carto.net/) von Rogers, welches der GPL unterliegt.

„shp2svg generates a main svg file containing geographic coverages, small legend svg files (one for each coverage), and an html file for displaying the results. The html file includes (a) tags to reference the Adobe SVG plugin (currently needed to display SVG within web browsers), (b) attribute data on features, as Javascript arrays, and (c) some basic Javascript coding to display those attribute data.“ (Rogers, 2002)

Dieses Avenue-Skript wird ständig weiterentwickelt und wurde auch für den Prototypen (s. Kapitel 4) verwendet.

Die Firma Uismedia bietet mit ihrem Produkt MapView (www.mapview.de) eine kommerzielle Extension für ArcView 3.x und ArcGis 8 an, die im Prinzip den selben Zweck wie „shp2svg“ erfüllt, jedoch handelt es sich hierbei um ausgereiftes Produkt, welches den Nutzer mit Dialogen durch den Erstellungsprozess führt und brauchbare Layouts als fertige HTML-Seite mit eingebettetem SVG liefert. Die Ergebnisse lassen sich als interaktive Web-Karten bezeichnen, denn es können Informationen zu einzelnen Elementen abgefragt werden – ein Query-Manager steht ebenfalls zur Verfügung. Für kleine Projekte wird der Preis von ca. 340 € (Stand: April 2003) eher abschreckend

sein.

Der Ansatz Konverter zu schreiben ist ebenfalls gebräuchlich. Eisenberg (2002) beschreibt ausführlich die Erstellung eines Perl-Skripts um Arc/Info Dateien in SVG zu konvertieren, dabei wird als Ausgangspunkt vom Arc/Info-ASCII-Export File („ungenerate“) ausgegangen.

Konvertierungs-Skripts, evtl. selbst geschriebene, haben den Vorteil, dass man die SVG-Dateien so aufbereiten kann wie diese für die Weiterverwendung benötigt werden – sie eignen sich jedoch nur bedingt um schnell ein fertiges Layout für eine Präsentation im Web zu produzieren.

Das OpenSource-GIS GRASS bietet derzeit keine direkte SVG-Unterstützung.

### 3.3.7 Animationen, Skripting

Bis dato wurden nur die statischen Fähigkeiten von SVG vorgestellt, doch eine der interessantesten Möglichkeiten von SVG ist die Erstellung von interaktiven Grafiken durch Animationen und Skripting.

Animationen können in SVG ohne starke Erhöhung der Dateigröße erstellt werden, weil diese auch nur in Form von XML-Tags gespeichert werden.

Die Animationsmöglichkeiten beruhen auf dem XML-Multimediastandard SMIL Level 2 (**S**ynchronized **M**ultimedia **I**ntegration **L**anguage, siehe http://www.w3.org/TR/smil20), wobei in SVG noch zusätzliche Elemente und Attribute definiert wurden, die mit den SMIL-Elementen zusammenarbeiten.(s. Fibinger, 2001)

SVG kennt verschiedene Möglichkeiten Animationen zu realisieren, grob muss man zwischen *Bewegungsanimationen* und *Farb- bzw. Attributänderungen* unterscheiden.

SVG stellt verschiedene Tags für Animationen zur Verfügung: (s. Fibinger (2001), Spona (2001), Eisenberg (2002))

- `<animate>`: erlaubt die dynamische Änderung von einzelnen Attributen von Tags. <animate> ist auf alle Tags anwendbar, die animiert werden können – siehe dazu in der SVG-Spezifikation des W3C (www.w3.org/Graphics/SVG) die Angabe „animatable:yes/no“.
  Eine Sonderform von <animate> stellt `<set>` dar: dieses kann auch nicht-numerische Attribute und Properties animieren – z.Bsp. die „visibility“.
- `<animateMotion>`: erlaubt die Bewegung eines Elements entlang eines Pfades.
- `<animateColor>`: Farbanimation von Elementen.

Die in SVG zusätzlich zum SMIL-Standard eingeführten Tags sind:

- `<animateTransform>`: Ändert Transformationsanweisungen des Tags.
- `<mpath>`: Spezifiziertes Auszeichnen eines Bewegungspfades.

Das folgende Beispiel soll einen Einblick in verschiedene Animationsmöglichkeiten geben:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="300px" height="250px">
<defs>
<animate xlink:href="#nose" id="nosevis" begin="ani2.end" dur="1s" attributeName="visibility" from="hidden" to="visible" fill="freeze"/>

<animateColor xlink:href="#nose" begin="nosevis.end" dur="2s" attributeName="fill" from="red" to="lightgreen" fill="freeze"/>
</defs>

<circle id="c1" cx="50" cy="50" r="35" style="fill:yellow;stroke:red;stroke-width:2px;"/>

<circle id="c3" cx="50" cy="50" r="35" style="fill:yellow;stroke:red;stroke-width:2px;">
<animateMotion id="ani1" begin="1s" dur="3s" fill="freeze" path="M0 0, 100 100, 170 0" />
<animate id="ani2" attributeName="r" begin="ani1.end" dur="3s" from="35" to="15" />
</circle>

<rect id="nose" x="115" y="75" width="20" height="55" style="fill:red;" visibility="hidden"/>
</svg>
```

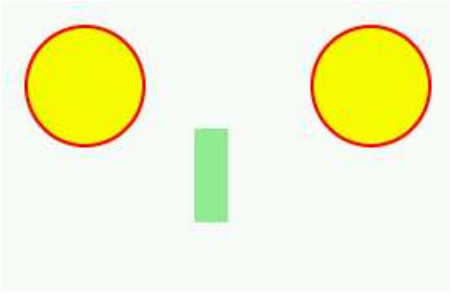

*Abbildung 22: Endzustand der SVG-Animation (eigener Entwurf)*

Zum Verständnis ist es am besten die Angaben in der <defs>-Section anfangs zu ignorieren; die Grafik besteht aus dem Kreis „c1“ und dem Kreis „c3“, außerdem wird ein Rechteck „nose“ definiert, das aufgrund der Angabe „visibility=hidden“ nicht sichtbar ist. In der Definition des Kreises „c2“, der in diesem Fall nicht mit „/>“, sondern mit </circle> geschlossen wird, befindet sich eine

<animateMotion>-Anweisung, die zum Zeitpunkt 1 Sekunde startet und 3 Sekunden andauert, wobei der Kreis entlang des angegebenen Pfades bewegt wird.
Die <animate>-Anweisung animiert des Attribut „r“ (Radius), startet mit dem Ende von „ani1“ und dauert 3 Sekunden an; der Wert für den Radius wird von 35 auf 15 geändert.
Neben der Angabe der Animationen im Element selbst, können Animationen auch in der <defs>-Section definiert werden, wobei die Referenzierung auf das zu animierende Objekt mittels dem bekannten „xlink:href“ erfolgt.

Neben den Animationen bietet sich bei SVG-Grafiken auch die Möglichkeit an, diese auf Basis von Benutzereingaben mittels einer Skriptsprache (Javascript/ECMA-Script) zu manipulieren. Mit Javascript und dem SVG-Objektmodell (SVG-DOM), welches sich an die Spezifikation des DOM Level 2 vom W3C hält, können Objekte innerhalb des SVG-Dokuments direkt angesprochen und manipuliert werden – mehr zu diesem Thema folgt in Kapitel 3.4 (Javascript/DOM).

## *3.4 JavaScript/DOM*

### 3.4.1 Grundlagen des clientseitigen Skripting

Die Grundlage für Interaktivitäten in Webseiten und auch SVG-Grafiken sind Skriptsprachen; diese bilden die Schnittstelle zwischen Browser (Webseite) und dem Nutzer.
„Scripting bedeutet ursprünglich das Zusammenfassen von Programmaufrufen zu einem Script, das in einer Datei gespeichert wird und dort nach Belieben abgerufen werden kann.“ (Dehnhart, 2001)
Seit Ende der 80er Jahre wurden Scriptsprachen, die eigentlich den Namen Programmiersprache verdienen, entwickelt; die traditionellen Programmiersprachen recht nahe kommen, auch bezüglich Objektorientierung und Abstraktionsmöglichkeiten – auch der Syntax ist oft an die traditionellen Vorbilder (z. Bsp. C) angelehnt. Nicht geändert hat sich auch bei modernen Skriptsprachen ihre Zweckorientierung.
Die meist verbreitetste Skriptsprache, die in HTML-Dokumenten eingesetzt wird, ist Javascript, daneben existiert auch noch VB-Script von Microsoft, jedoch ist VB-Script auf die Windows-Welt beschränkt und wird dort auch nur vom Internet Explorer unterstützt. Am häufigsten wird aus diesem Grund Javascript eingesetzt, da es auf nahezu von jedem Browser unter jedem Betriebssystem unterstützt wird, sodass es auch im Prototypen (s. Kapitel 4) eingesetzt wurde. Urheber von Javascript ist Netscape, außerdem ist es nicht mit der Programmiersprache Java von Sun gleichzusetzen bzw. zu verwechseln.

Seit 1998 existiert mit ECMA-Script (**E**uropean **C**omputer **M**anufacturers **A**ssociation) eine Standardisierung (ISO-Standard; ISO-Norm 10262), doch leider sind nicht alle Skriptsprachen vollständig ECMA-kompatibel. (s. Berger, 2001)

Javascript verfügt über alle Möglichkeiten, die man sich von einer Programmiersprache erwartet, sei es Variablen zu definieren, Schleifen zu programmieren, Fallunterscheidungen vorzunehmen und Vieles mehr – eine Einführung in Javascript bieten u.a. Münz (2001) und Wenz (2001).

Javascript-Kommandos können in HTML-Dateien an mehreren Stellen untergebracht werden:

- Zwischen den Tags <scipt> und </script>
- in einer externen Datei mit Verweis darauf
- in Form von HTML-Links
- Als Parameter von HTML-Tags

Netscape ist bei Javascript den Weg gegangen den Webbrowser (und die Inhalte) in Objekten zu modellieren auf die mittels Javascript zugegriffen werden kann. HTML-Dokumente müssen so modelliert werden, dass mit Javascript darauf zugegriffen werden kann – die Modellierung des Dokuments wird mit dem Begriff DOM (**D**ocument **O**bject **M**odel) umschrieben, welches mittlerweile ein W3C-Standard ist.

### 3.4.2 Das DOM (Document Object Model)

Das DOM stellt eine hierarchische Ordnung von Objekten und Methoden innerhalb des Dokuments dar, die in Form einer Baumstruktur vorliegt. Die einzelnen Bestandteile einer solchen Baumstruktur werden als Knoten (node) bezeichnet, die man wiederum in Elementknoten, Attributknoten und Textknoten unterscheiden kann. Übergeordnete Knoten werden als Elternknoten (parentnode), untergeordnete Knoten als Kindknoten (childnode) bezeichnet. Handelt es sich bei einem untergeordneten Knoten um einen Attributknoten, so spricht man von einem assoziierten Knoten. Dokumente können damit sehr verschachtelt und tiefgründig werden. (s. Berger (2001), Wenz (2001))

Bei HTML-Dokumenten wird jedes Element und jeder Text, der nicht von Tags umgeben ist, zu einem Node. Elemente zwischen einem öffnenden und dem dazugehörigen schließenden Tag sind Child nodes (Kindknoten), sodass eine Hierarchie entsteht.

Das folgende Beispiel soll dies verdeutlichen:

```
<HTML>
<HEAD>
<TITLE>Das HTML-DOM</TITLE>
</HEAD>
```

```
<BODY ID="DOK">
<H1 ID="Ueberschrift1">Das HTML-DOM</H1>
<IMG src="bild1.png" ID="bild1">
<IMG src="bild2.png" ID="bild2">
</BODY>
</HTML>
```

Der DOM-Baum des Beispiels würde wie folgt aussehen:

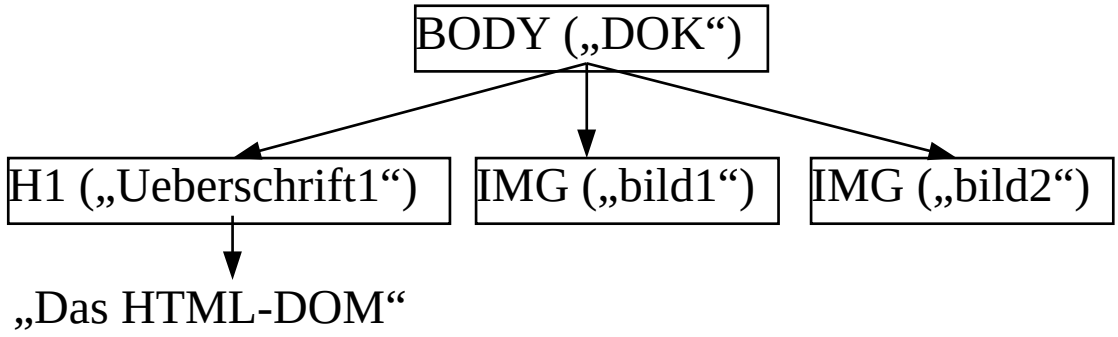


*Abbildung 23: DOM-Baum des Beispiels (eigener Entwurf)*

Bestimmte Methoden erlauben eine Manipulation der Elemente im DOM, wobei das DOM die Eigenschaften und Methoden liefert, die mittels der Skriptsprache (z. Bsp. Javascript) dem Browser verständlich gemacht werden.

### 3.4.3 Skripting in SVG

In SVG ist ebenfalls das DOM (SVG-DOM) implementiert, das auf DOM Level 2 des W3C basiert. XML und XML-Dialekte wie SVG weisen im Gegensatz zu HTML bereits eine stringente Hierarchie auf, die in 3.4.2 angesprochene Baumstruktur ist per se vorhanden. Der „Baum“ wird auch im SVG-DOM von oben nach unten durchlaufen, wobei an oberster Stelle das Dokument steht. (s. Fibinger (2001), Spona (2001), Eisenberg (2002))

Im Zusammenhang mit Skripting und dem DOM muss festgestellt werden, dass gegnwärtig die PlugIns nicht alle Möglichkeiten des DOM unterstützen.

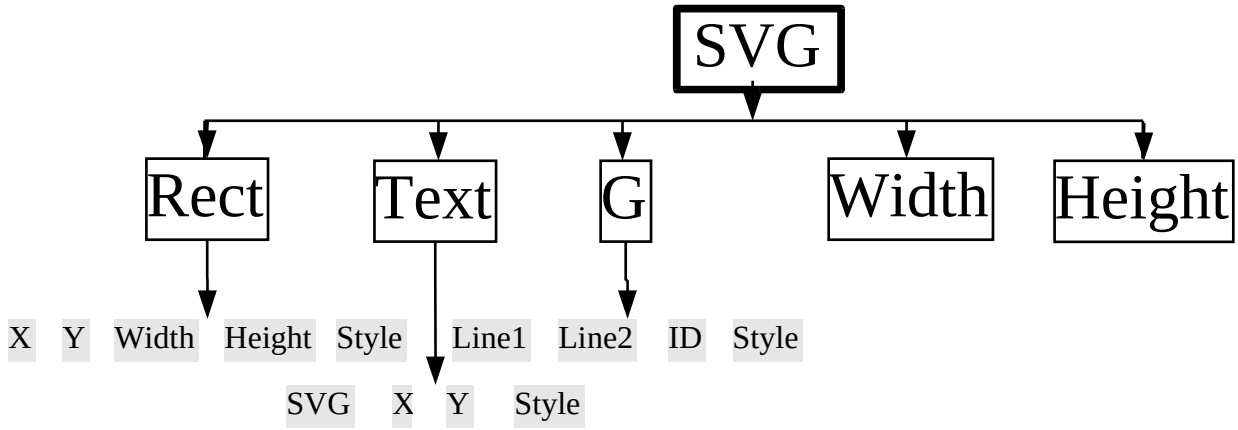


*Abbildung 24: Beispiel eines SVG-DOM (eigener Entwurf)*

Die Auslösung Skript-gesteuerter Interaktionen erfordert zwei Komponenten:

- das Skript (Programm) an sich
- ein Ereignis (Benutzeraktion, wie z. Bsp. ein Mausklick), welches das Skript auslöst und über einen so genannten *Event-Handler* erfasst wird.

In HTML und auch SVG stehen zahlreiche Event-Handler zur Verfügung, darunter „onclick“, „onmouseover“, „onmouseout“ usw. - beim Klick auf ein Element wird also das Ereignis „onclick“ ausgelöst, welchem eine Funktion (Programm) zugewiesen werden kann.

Die Integration eines Skripts in SVG erfolgt wie in HTML über den <script>-Tag, jedoch muss in SVG (XML) mit der CDATA-Klammer („<![CDATA ...]]>“) ein Bereich gekennzeichnet werden, der nicht zur Auszeichnung gehört (vgl. CSS-Anweisung in SVG).

Die Skripts können in der SVG-Datei stehen, jedoch auch in einer HTML-Datei, die eine oder mehrere SVG-Grafiken enthält.

Das folgende einfache Beispiel versucht einen Einblick in die Interaktionsmöglichkeiten mit dem Nutzer zu geben, wobei die SVG-Grafik in eine HTML-Seite eingebettet ist und über HTML-Formularfelder manipuliert werden kann.

```
<html>
<head>
<title>SVG &uuml;ber HTML-Formular beeinflusst (JS)</title>
<script language="JavaScript">
<!--
 function showname(name)
  {
  alert('Das wird angezeigt: '+name);
  var svgdoc = document.svggrafik.getSVGDocument();
  var textelem = svgdoc.getElementById('name');
  textelem.getFirstChild().setData(name);
```

```
  }
 function change(width, color)
  {
  alert(width+' '+color);
  var svgdoc = document.svggrafik.getSVGDocument();
  var rechteck = svgdoc.getElementById('bg');
  rechteck.setAttribute('width', width);
  rechteck.setAttribute('style','fill:'+color);
  }
//-->
</script>
<noscript>Aktiviere JavaScript im Browser - Flo</noscript>
<meta name="author" content="flo">
<meta name="generator" content="handmade">
</head>
<body text="#000000" bgcolor="#FFFFFF" link="#FF0000" alink="#FF0000" vlink="#FF0000">
<div align="center">
<h1>SVG-DOM Scripting</h1>

<embed name="svggrafik" width="320" height="250" src="sample_dom_script.svg"
             type="image/svg+xml"
pluginspage="http://www.adobe.com/svg/viewer/install/">

<br>Geben Sie einen Text ein:
<form name="form1">
  <input type="Text" name="name" value="Nikolaus Krampus" size="30" maxlength="">
  <input type="button" name="but1" value="anzeigen!" onclick="showname(form1.name.value)
">
</form>
Gr&ouml;&szlig;e und Farbe des Rechtecks &auml;ndern:
<form name="form2">
  Breite: <input type="Text" name="width" value="300" size="5" maxlength=""><br>
  Farbe: <input type="Text" name="color" value="lightblue" size="15" maxlength="">
  <input type="button" name="but2" value="&auml;ndern!" onclick="change(form2.width.va-
lue, form2.color.value)">
</form>
</div>
</body>
```

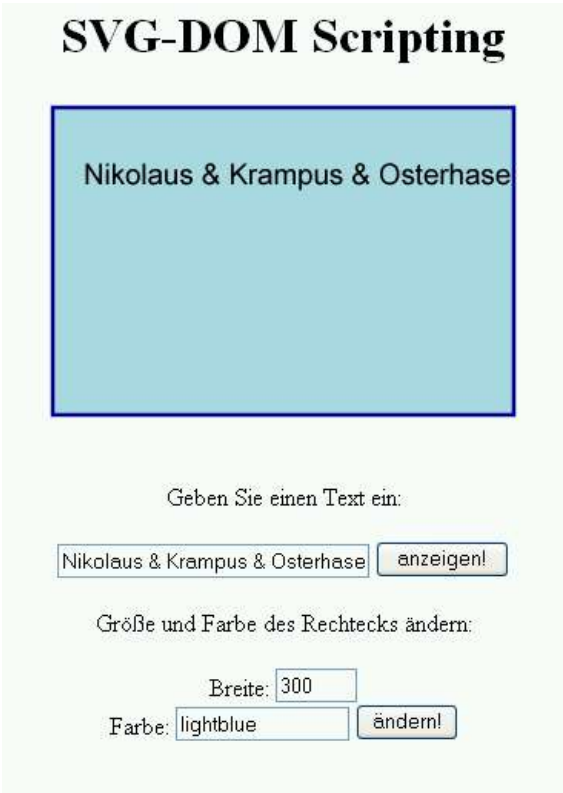


*Abbildung 25: Screenshot Scripting Beispiel (eigener Entwurf)*

Die SVG-Grafik besteht lediglich aus einem Rechteck mit der ID „bg“ und einem Textelement mit der ID „name“; der Nutzer hat in der HTML-Seite die Möglichkeit den Text (Inhalt) zu ändern, sowie Attribute des Rechtecks zu beeinflussen (width und fill-color).

Auf die SVG-Grafik innerhalb der HTML-Seite kann über den Namen, der im <embed>-Tag vergeben wurde zugegriffen werden. Die Formulare bestehen jeweils aus Eingabefeldern, deren Werte (Benutzereingaben) jeweils einer Javascript-Funktion übergeben werden; die Ausführung der Skripte wird durch ein onclick-Event auf die Buttons eingeleitet, dabei werden auch die Variablen (Benutzereingaben) übermittelt.

Die Javascript-Funktionen geben jeweils zuerst die Übergabewerte mit „alert(...);“ aus. Der Zugriff auf einzelne Knoten innerhalb eines SVG-Dokumentes erfordert anfangs immer eine Referenz auf das Dokument selbst, die in einer Variablen gespeichert wird:

```
var svgdoc = document.embedname.getSVGDocument();
```

„embedname“ entspricht dabei dem im <embed>-Tag vergebenen Namen – alternativ kann auch `embed[0]` usw. verwendet werden.

Ist das Dokument referenziert steht einer Bearbeitung einzelner Knoten nichts mehr im Weg, dazu stehen verschiedene Methoden zur Verfügung:

- getElementById(): spricht ein Element über das „Id“-Attribut an. Der entsprechende Knoten wird in einer Variablen angelegt: `var Knoten = svgdoc.getElementById('id');`
- getElementByTagName()

Wurde ein Knoten ermittelt kann man verschiedene Methoden anwenden um dessen Eigenschaften zu erfassen oder dessen Eigenschaften zu ändern. Die wichtigsten *Methoden zur Erfassung der Eigenschaften* sind:

- getNodeType(): 1=Elementknoten, 2=Attributknoten, 3=Textknoten
- getNodeName()
- getChildNodes()
- getFirstChild(): Wenn ein Element nur einen ChildNode besitzt kann dieser damit angesprochen werden.
- getParentNode()
- getAttribute()
- getStyle()
- getData(): Liest den Inhalt eines Textknotens aus.

Wahrscheinlich interessanter sind die *Methoden zur Manipulation* von Knoten:

- setAttribute() um den Wert eines Attributs eines Knotens zu ändern
- setProperty() um Eigenschaften des Style zu ändern
- setData() ändert den Inhalt eines Textknotens

Neben der Manipulation bestehender Elemente bietet das DOM auch die Möglichkeit einzelne Objekte dynamisch zu erzeugen bzw. zu kopieren. Die Methode `createElement()` erzeugt eine neues Element, `cloneNode(true/false)` klont ein vorhandenes Element. Um das Element in die Hierarchie einzufügen muss zuerst die „Stelle“ innerhalb der DOM-Hierarchie referenziert werden, an der das Element eingefügt werden soll: z. Bsp. `var Parent = svgdoc.getElementById('parent')`, dann kann mit `appendChild()` der neue Knoten angefügt werden: `Parent.appendChild('neues Element')` – die Methode `insertBefore()` fügt das neue Element vor dem referenzierten Knoten (Parent) ein.

Im Beispiel ändert die Funktion „showname“ mittels „setData“ den Inhalt des Textelements auf die Benutzereingabe. Zur Änderung des Inhalts muss mittels „getFirstChild“ auf den Kinderknoten des Text-Knotens referenziert werden.

Die Funktion „change“ ändert mittels „setAttribute“ die Attribute des Rechtecks.

## 3.5 Webserver und CGI

Webserver sind jene Software, die die Anfragen des Clients (Browsers) entgegennehmen, abarbeiten und die gewünschten HTML-Seiten an den Client zurücksenden. Die Anfragen an den Server werden in standardisierter Form übermittelt – http://servername/seite; diese Adressierung wird als URL (Uniform Ressource Locator) bezeichnet.

Die Daten werden mit Hilfe des spezifischen Webprotokolls HTTP (HyperText Transfer Protocol) transportiert.

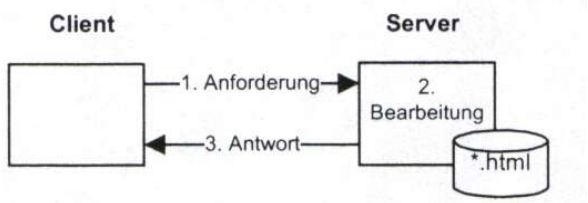


*Abbildung 26: Statischer Server (aus: Dehnhart, 2001)*

Die gängigsten Webserver sind der IIS (Internet Information Server) von Microsoft und die Open-Source Lösung Apache (www.apache.org). Apache hat nicht nur den Vorteil, dass die Software kostenlos ist, sondern die Verfügbarkeit für verschiedene Betriebssysteme spielt auch eine Rolle – Apache wird in der Regel auf Linux-Rechnern eingesetzt.

Auf dem Webserver laufen häufig mehrere Server gleichzeitig (auf der selben physischen Maschine), beispielsweise Webserver und Datenbankserver.

Server, die lediglich die angeforderten Dateien ausliefern, die als bereits erstellte Dokumente auf dem Server vorliegen, nennt man statisch. Hierbei ist keine direkte Rückkoppelung zum Anbieter (Server) vorgesehen. (s. Dehnhart, 2001)

Viele Webapplikationen erfordern jedoch eine Rückkoppelung zum Server, zum Beispiel Online-Shops oder Adressverortungs-Anwendungen, um eine Anfrage des Client qualifiziert zu beantworten. Jede Anfrage erhält folglich eine individuelle Antwort, d.h. maßgeschneiderte Inhalte bzw. HTML-Dokumente. Der Server muss somit in der Lage sein, in serverseitigen Programmen diese Anfragen zu bearbeiten. Webseiten dieser Art tragen die Bezeichnung dynamisch.

Die am weitesten verbreitete Schnittstelle zwischen Webserver und serverseitigen Programmen stellt CGI (Common Gateway Interface) dar. Neben der Lösung über eine Schnittstelle gibt es auch die Möglichkeit direkt in den Server integrierte Programme (Webserver-Erweiterungen genannt) zu verwenden. (s. Däßler (2001), Dehnhart (2001))

Die im Zusammenhang mit CGI wahrscheinlich meist genutzte Programmiersprache ist Perl – der Webserver übergibt über die CGI-Schnittstelle Daten an ein Programm, dieses verarbeitet die Daten und übermittelt sie an den Webserver zurück.

Der Nachteil der Lösung über eine offene Schnittstelle liegt in der Performance, da bei jedem Aufruf das Programm neu gestartet und beendet wird. Für jeden Aufruf wird somit auf dem Server ein neuer Prozess gestartet, der Systemressourcen beansprucht. Wenn eine Vielzahl von Nutzern gleichzeitig zugreifen kommt es zu einer Serverüberlastung.

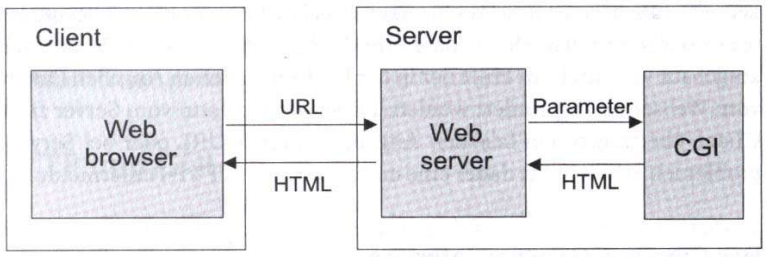


*Abbildung 27: Funktionsweise von CGI (aus: Däßler, 2001)*

Eine Lösung dieses Problems bieten die Webserver-Erweiterungen, da die CGI-Programme nicht mehr als eigenständige Prozesse unabhängig vom Webserver laufen, sondern intern innerhalb eines Webserver-Prozesses. Zu den Programmiersprachen (Skriptsprachen), die dies unterstützen, zählt PHP (Hypertext PreProcessor; s. Kapitel 3.7), die auch im Prototyp (s. Kapitel 4) eingesetzt wurde.

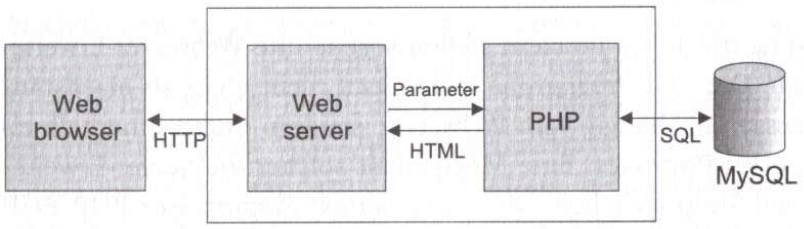


*Abbildung 28: Servermodule am Bsp. PHP (aus: Däßler, 2001)*

## *3.6 Datenbanksysteme*

### 3.6.1 Grundlagen

Datenbanken spielen seit den 70ern in nahezu allen Bereichen des täglichen Lebens eine gewichtige Rolle – sei es beim Bankomaten oder seit den späten 90ern im Internet.

„Die Diskussion um Datenbanken ist von großer Begriffsverwirrung und Äpfel-mit-Birnen-Vergleichen geprägt“ (Schulz & Siering, 2003), sodass anfangs einige Fachbegriffe aus der Datenbankwelt näher definiert werden müssen:

- Datenbank: Die Datenbank (DB) definiert eine Komponente des Datenbanksystems, die für die systematische Speicherung von Daten verantwortlich ist, die Datenbasis des Systems. Die Daten-

basis liegt zunächst in unstrukturierter Form vor, erst die Implementierung in ein Datenbankmanagementsystem führt zu einer Strukturierung (‣ Modellierung) und ermöglicht eine Pflege und Auswertung. Die Datenbank ist auf einem Datenträger gespeichert, von dessen Leistungsfähigkeit nicht zuletzt die Leistungsfähigkeit der DB abhängt. Das Datenmodell (s. Kapitel 3.6.2 – Datenmodelle) bestimmt in welcher Anordnung die Daten abgelegt werden – beim Relationenmodell werden die Daten beispielsweise in Tabellen abgelegt. Die Datenbank beinhaltet dabei zwei Komponenten:

- *Basisbereich* in dem die konkreten Daten abgelegt werden
- *Systemkatalog* (Data Dictionary) mit den Angaben über die logische Struktur der DB.

- Datenbankmanagementsystem (=Datenbankverwaltungssystem): Das Datenbankmanagementsystem (DBMS) stellt eine Ansammlung von Werkzeugen zur Strukturierung und Manipulation der Daten in einer Datenbank dar. Das DBMS bietet dazu Funktionen um Daten einzugeben, zu verändern und abzufragen.
  Das DBMS hat im Allgemeinen folgende Aufgaben zu erfüllen:
  - Datenspeicherung und Implementation des Datenmodells
  - Datenverwaltung und Manipulation
  - Datenzugriff
  - Gewährleistung der Datensicherheit

  Däßler (2001) bezeichnet das DBMS als das „Betriebssystem“ einer Datenbank.
  In der Praxis wird das DBMS oft mit der DB verwechselt, die DB ist die eigentliche Datenbasis, das DBMS die Software !

- Kommunikationsschnittstelle (=Data Base Communication Interface – DBCI): Diese dient der Kommunikation zwischen Benutzer und Datenbank. Diese Aufgabe wurde früher häufig dem DBMS zugeordnet, doch aufgrund einer Vielzahl von Möglichkeiten Datenzugriffe in andere Applikationen einzubinden wird diese gegenwärtig meist als separate Komponente geführt. Die Kommunikation erfolgt mittels grafischer Bedienoberflächen oder im Kommandozeilen-Modus, d.h. durch die direkte Eingabe von Befehlen in einer speziellen „Datenbanksprache“. Dabei werden grundsätzlich zwei Sprachen unterschieden:
  - *Datenbeschreibungssprache* (DDL): Hiermit wird die logische Struktur der Datenbank definiert – im relationalen DB-Modell beispielsweise Tabellen zu erstellen, zu

löschen etc.

- *Datenmanipulationssprache* (DML): Diese bietet Operationen zur Arbeit mit dem DBS, wie Daten einzugeben, zu ändern, zu löschen und Abfragen zu erstellen. Als DML hat sich die bekannte Abfragesprache SQL etabliert. (s. Kapitel 3.6.6 – SQL) Neben SQL müssen zur DML auch die Datenmanipulationssprachen in der Anwendungsprogrammierung beachtet werden, die über API's (Application Programming Interface) auf die Daten zugreifen können (s. Kapitel 3.7 – PHP).

- Datenbanksystem (=DBS): Das DBS definiert das Gesamtsystem, bestehend aus Datenbank, Datenbankmanagementsystem und Kommunikationsschnittstelle.

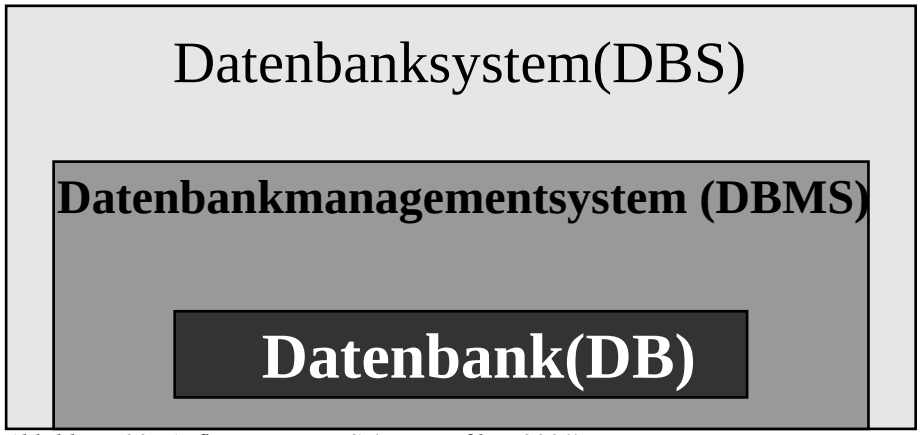


*Abbildung 29: Aufbau eines DBS (aus: Däßler, 2001)*

## 3.6.2 Datenmodelle

Der Unterschied und damit die Stärke eines Datenbanksystems (DBS) im Vergleich zu Dateiverwaltungssystemen liegt in der Abbildung der Wirklichkeit durch ein Datenmodell. Dabei werden entweder Objekte, die in Beziehung stehen, oder ganze Prozessabläufe modelliert. Aus diesem Grund werden DBS auch auf Basis der ihnen zugrunde liegenden Datenmodelle eingeteilt. Die Datenmodellierung hat vier grundlegende Aufgaben zu erfüllen (s. Däßler, 2001):

- Abstraktion
- Strukturierung
- Hierarchisierung
- Modularisierung

Jedes DBS bietet ein eigenes Datenmodell, dennoch lässt sich jedes kommerzielle DBS einem der vier gebräuchlichsten Datenmodelle zuordnen:

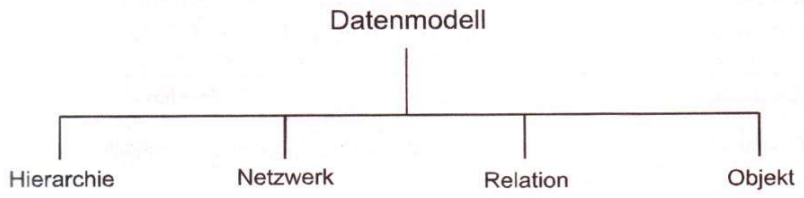


*Abbildung 30: Häufige Datenmodelle (aus: Däßler, 2001)*

### 3.6.2.1 Hierarchisches Modell

Dieses Datenmodell ist das älteste und stammt aus einer Zeit (50er/60er) in der Firmen und Produktionsabläufe hierarchisch organisiert waren.

Ein Datensatz wird mit allen hierarchisch von ihm abhängigen Datensätzen als Einheit betrachtet. Ein Zugriff auf einen bestimmten Datensatz kann somit nur über den Suchschlüssel des Objekts der obersten Ebene erfolgen. (s. Hake & Grünreich, 1994)

Viele Prozesse unserer realen Welt sind jedoch nicht hierarchisch organisiert und müssen daher mit einer großen Zahl von Redundanzen an diese Datenstruktur angepasst werden.

Das hierarchische Modell lässt keine Beziehung zwischen den unterschiedlichen Ästen eines Baumes zu – es sind nur 1:n-Beziehungen möglich.

Das hierarchische Modell herrscht bis dato in der Dateiverwaltung vor !

### 3.6.2.2 Netzwerkmodell

Dieses Modell stellt eine Erweiterung des Hierarchischen Modells dar, da zwischen den Baumverzweigungen Verknüpfungen mittels Pointern (Zeiger) möglich sind. Das Setzen und Nachführen der Zeiger bei Änderungen geschieht dabei meist nach den Regeln der CODASYL DBTG (Conference on Data Systems Languages Database Task Group). Somit sind neben 1:n-Beziehungen auch m:n-Beziehungen möglich.

Hierarchische- und Netzwerkmodelle spielen gegenwärtig kaum noch eine Rolle, außer bei langjährigen Datenbeständen die auf diese Art und Weise organisiert sind und noch im Produktiveinsatz sind.

### 3.6.2.3 Relationenmodell

Das Relationale Datenbankmodell (RDBM) nimmt unter allen Datenbanksystemen eine herausragende Position ein. Es geht auf den Mathematiker E. F. Codd zurück, der es im Jahr 1970 erstmals vorgestellt hatte. Ende der 80er haben IBM und Oracle, heute noch die „Big-Player“ im Datenbankmarkt, die ersten kommerziellen Produkte auf Basis des RDBM ausgeliefert. Eine der größten Verdienste von Codd war die Anwendung von Verfahren der angewandten Mathematik (Mengenalgebra, Relationenalgebra) auf die Verwaltung von Daten in einem Datenbanksystem.

| ***Operation*** | ***Beschreibung*** |
|---|---|
| Selektion | Auswahl einer Untermenge von Tupeln[5] einer Relation |
| Projektion | Auswahl von Spalten einer Tabelle |
| Kartesisches Produkt | Vereinigung von zwei Tabellen durch Kombination aller Tupel beider Tabellen |
| Union | Vereinigung von zwei Tabellen mit gleichen Attributen, identische Tupel treten nur einmal in der Ergebnisrelation auf |
| Differenz | Vereinigung von zwei Tabellen mit gleichen Attributen, Tupel die in beiden Tabellen vorkommen, treten nicht in der Ergebnisrelation auf |
| Schnittmenge | Vereinigung von zwei Tabellen mit gleichen Attributen, nur Tupel, die in beiden Tabellen vorkommen, treten in der Ergebnisrelation auf |
| Join | Verbundoperation zwischen Tabellen, die die Kombinationen des Kartesischen Produkts zweier Relationen einschränkt (selektiert) |

Grundlage des relationalen Modells sind Tabellen, wobei jede Spalte der Tabelle Attribute eines Objektes der Realwelt beschreibt. Die Zeilen einer Tabelle entsprechen damit den Datensätzen. Eine Datenbank besteht in der Regel aus mehreren Tabellen, das Datenmodell entsteht durch die Verknüpfung verschiedener Tabellen.

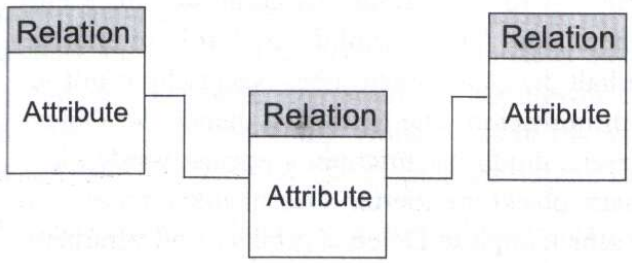


*Abbildung 31: Relationenmodell (aus: Däßler, 2001)*

Zur Bildung von Beziehungen zwischen den Tabellen werden spezielle Attribute innerhalb der Tabellen benötigt, die Schlüsselattribute.
Auf Codd gehen auch die Regeln zur redundanzfreien Speicherung der Daten zurück, wobei der Anpassungsprozess des Datenmodells als Normalisierung bezeichnet wird.

Eine Sonderform des Relationalen Modells stellt das **Objektrelationale Modell** dar. Es handelt sich dabei um ein Relationales Modell welches um objektorientierte Konzepte, wie Relationen als

5 Tupel: Bezeichnung für Datensatz ( ▸ Tabellenzeile)

Klassen, Typkonstruktoren, Objektidentität, Methoden und Beziehungen, erweitert wurde. Datenbankabfragen werden in einer dem RDBM adäquaten Form formuliert, wohingegen echte objektorientierte Modelle eigene Abfragesprachen nutzen.

#### *3.6.2.4 Objektorientiertes Modell*

Ein Nachteil des RDBM ist jedoch, dass Entitäten[6] durch die Normalisierung ihre Objektidentität verlieren.

Das Konzept der Objektorientierung ist seit den 70ern bekannt und wurde erstmals erfolgreich bei der Entwicklung grafischer Entwicklungsumgebungen in den Laboren von Xerox eingesetzt. Heute sind alle Programmiersprachen objektorientiert, bei den DBS hat die Objektorientierung noch nicht eine solche Marktpenetration erreicht.

„Aus objektorientierter Sicht wird ein Realitätsausschnitt als eine Menge von Objekten beschrieben. Jedes Objekt wird durch Attribute und Methoden definiert und erhält die Möglichkeit, über Nachrichten mit anderen Objekten zu kommunizieren. Die Methoden dienen der Objektmanipulation, die ihrerseits durch Nachrichten ausgelöst werden können." (Däßler, 2001)

Heuer (1992) formuliert folgende Prinzipien objektorientierter Datenbanken:

- In objektorientierten DBS sind nicht nur Standard-Datentypen für Eigenschaften von Objekten erlaubt, sondern auch die wiederholte Anwendung von Typkonstruktoren.
- Objektorientierte DBS wahren die Objektidentität. Objekte existieren unabhängig von den Werten ihrer Eigenschaften. Während sich die Werte der Eigenschaften ändern können, bleibt die Identität des Objekts unveränderlich.
- In objektorientierten DBS kann man neben den Eigenschaften von Objekttypen auch die mit ihnen durchführbaren Methoden in die Objekttyp-Definition einkapseln und vererben.

#### *3.6.2.5 Welches Modell ?- Entscheidungshilfe*

Die vorhergehenden Kapitel haben einen Überblick über die Evolution der Datenbanktechnologie gegeben, wobei die relationalen Systeme den status quo hinsichtlich Verbreitung darstellen und als Basis der Entwicklung objektrelationaler und objektorientierter Systeme in Form von Nebenlinien dienen. Als möglicher nächster großer Entwicklungsschritt werden häufig die XML-Datenbanken genannt – s. Kapitel 3.6.7.

Da die Entscheidung für ein Datenbanksystem und Datenmodell langfristige Konsequenzen hat, stellt sich die Frage welches System eingesetzt werden soll.

Diese Frage kann nicht einfach beantwortet werden, da die Auswahl jeweils individuell zu treffen ist, dennoch muss festgestellt werden, dass die bereits vor 10 Jahren weitläufige Meinung, dass

6 Entität: Objekt der Realwelt

objektorientierte Systeme die Zukunft seien und alles ablösen werden nicht eingetreten ist.

Breutmann (2001) bietet ein grobes Entscheidungsraster, welches auf der Komplexität der Daten und dem Bedarf nach Abfragen beruht:

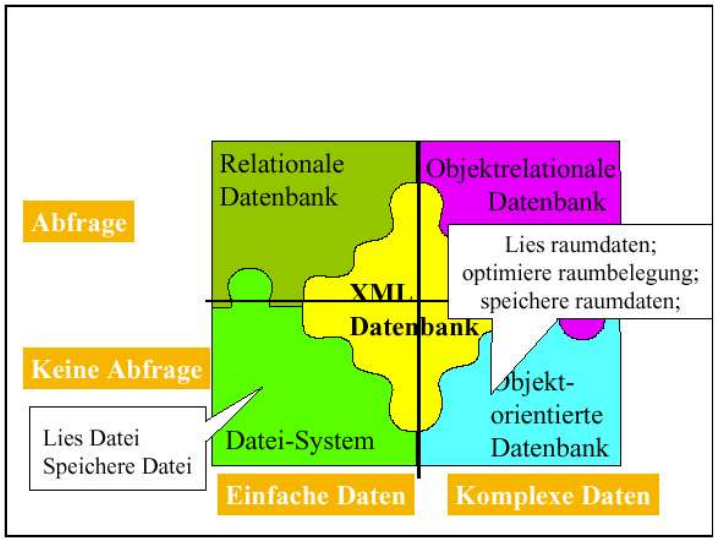


*Abbildung 32: Entscheidungsraster (aus: Breutmann, 2001)*

Der erste Fall stellt im Prinzip ein Dateisystem dar, es können damit einfache Daten ohne Abfragen verwaltet werden. Jede Textdatei würde in dieses Feld passen – man öffnet die Datei, bearbeitet sie und speichert diese wieder zurück.

Die zweite Gruppe bilden die relationalen DBS, die insbesondere für einfache Daten mit Abfragebedarf geeignet sind.

Die nächste Gruppe sind die objektorientierten Systeme, die nach Breutmann (2001) für Anwendungen mit komplexen Daten ohne hohe Abfragebedürfnisse geeignet sind. Objektorientierte Systeme sind in der Praxis für hohen Datendurchsatz und große Mengen nicht brauchbar – sie haben sich meist nur in Spezialbereichen bewährt und aus diesem Grund ist auch der Markt für objektorientierte Datenbanken um den Faktor 100 kleiner. (s. Breutmann, 2001)

Die vierte und rezent leistungsfähigste Gruppe sind objektrelationale Systeme, die die Fähigkeit komplexe Objekte zu verwalten mit den bewährten Errungenschaften relationaler Systeme verbinden.

Nicht zuletzt aufgrund der Tatsache, dass bei einer Umstellung der Informationshaushalt nicht komplett neu organisiert werden muss, sondern nur diese Teile, die besser über komplexe Datentypen zu verwalten sind, wird dem objektrelationalen Modell das größte Potential zugesprochen. (s. Breutmann, 2001)

### 3.6.3 Modellierung der Realwelt – ERM und RDM

Im Rahmen dieser Arbeit kann nicht auf den gesamten Vorgang der Modellierung eingegangen werden, doch die notwendigen Schritte und Regeln beim Aufbau einer RDB sollen erwähnt werden. Bei der Datenbankentwicklung wird das so genannte Entity-Relationship-Model (ERM) zur Konzeption verwendet, später erfolgt die Überführung ins Relationship Data Model (RDM).
Im wesentlichen werden im ERM Objekte und Prozesse erfasst und grafisch mittels einem ERD (Entity Relationship Diagram) dargestellt.

Eine der wichtigsten Aufgaben des ERM ist es, zu zeigen, welche Assoziationen zwischen den Objekten bestehen. Prinzipiell werden zwei Beziehungstypen unterschieden: eindeutige und mehrdeutige Beziehungen.

| ***Beziehungstyp*** | ***Beschreibung*** | ***Beispiel*** |
|---|---|---|
| Eineindeutig 1:1 | Einem Objekt vom Typ A ist genau ein Objekt vom Typ B zugeordnet. | Ein Buch hat genau eine ISBN-Nummer; jeder ISBN-Nr. ist genau ein Buch zugeordnet. |
| Funktional 1:n | Jedes Objekt vom Typ A kann mit beliebig vielen vom Typ B in Beziehung stehen. Jedes Objekt vom Typ B kann jedoch nur mit genau einem Objekt vom Typ A in Beziehung stehen. | Einem Buch kann immer nur ein Verlag zugeordnet werden, jedoch können einem Verlag mehrere Bücher zugeordnet werden. |
| Komplex m:n | Jedem Objekt vom Typ A können beliebig viele vom Typ B zugeordnet werden - und detto in die andere Richtung. | Bibliothek: Ein Buch kann von mehreren Personen ausgeliehen werden, eine Person kann mehrere Bücher ausleihen. |

1:1 und 1:n Beziehungen können direkt ins RDM übernommen werden, m:n-Beziehungen müssen aufgelöst werden.

Ein wesentliches Merkmal des RDM ist die Einhaltung der Normalformen, sodass die Normalisierung des ERM einen wesentlichen Faktor zur Transformation in das RDM darstellt (Däßler, 2001).

- 1. Normalform: Eine Relation befindet sich in der ersten Normalform, wenn keines ihrer Attribute eine untergeordnete Relation darstellt und wenn alle Attribute nur atomare Werte beinhalten.

- 2. Normalform: Laut Definition muss die Datenbank immer zuerst in die erste Normalform versetzt werden, bevor man diese in die 2. Normalform versetzen kann. Hierbei müssen alle nicht zum Schlüssel gehörenden Attribute von diesem voll funktional abhängig sein. Besteht ein Schlüssel aus mehreren Teilschlüsseln, so ist das Element aus dem Datensatz herauszuziehen, welches nur von einem Teilschlüssel abhängt.
- 3. Normalform: Zusätzlich zur 2. Normalform gilt für jeden Schlüssel: Alle nicht zum Schlüssel gehörende Attribute sind nicht von diesem transitiv abhängig. Das bedeutet, dass alle Attribute nur vom Schlüsselattribut, nicht aber von anderen Attributen abhängig sein. Eine Abhängigkeit zwischen den Attributen muss aufgelöst werden.

Däßler (2001) gibt einen guten Überblick zur Transformation des ERM ins RDM:

- Alle Entitätstypen im ERM werden als gleichnamige Relationen übernommen. Die Eigenschaften einer Entität bilden die gleichnamigen Spaltennamen einer Tabelle.
- Beziehungsentitäten werden dann als Relationen übernommen, wenn sie selbst Attribute besitzen bzw. eine m:n-Beziehung repräsentieren.
- Für jede Relation wird ein Primärschlüssel festgelegt.
- M:n-Beziehungen werden in 1:n Beziehungen umgewandelt, indem eine neue Relation konstruiert wird, die als Attribute formal die Primärschlüssel der beiden beteiligten Relationen enthält.
- Durch die Definition von Fremdschlüsseln werden alle 1:1- und 1:n Beziehungen in das relationale Modell transformiert.
- Alle Relationen werden abschließend auf die Erfüllung der 3. Normalform überprüft. Gegebenenfalls müssen alte Relationen aufgelöst und neue Relationen konstruiert werden.

### 3.6.4 SQL – Structured Query Language

Das bereits beschriebene relationale Datenmodell sorgt dafür, dass sich die Daten in einer strukturierten Form befinden und normgerechte Beziehungen zwischen den Tabellen bestehen. Damit folgt als nächster Schritt die Bearbeitung der Daten an sich mittels einer Datenbanksprache.

Datenbanksprachen bilden die Kommunikationsschnittstelle eines DBMS, wobei die Kommunikation in der Regel auf Basis einer normierten Sprache erfolgt. Die gängigste Sprache zur Definition der Datenstrukturen, zum Pflegen und Abfragen der Daten ist SQL, die **S**tructured **Q**uery **L**anguage.

SQL wurde in den 70er Jahren bei IBM entwickelt und erstmals im Datenbanksystem „System/R“

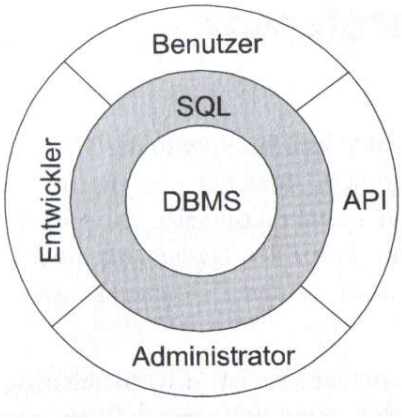


*Abbildung 33: SQL (aus: Däßler, 2001)*

eingesetzt – durchsetzen konnte sich SQL erst mit den Datenbanksystemen von Oracle in den 80ern. Einstweilen gibt es die Definition dieser Sprache in zwei ANSI[7]/ISO[8]-Standards mit der Bezeichnung SQL92 (ISO/IEC 9075) und SQL99 (ISO/IEC 9075:99). Die größte Marktpenetration hat bis dato nur SQL92, SQL99 konnte sich in den Produkten noch nicht durchsetzen. (s. Däßler (2001), Schulz & Siering (2003))
Mit der Kenntnis von SQL hat man die Basis für die Arbeit auf allen SQL-basierenden Datenbanksystemen, jedoch hält sich kaum ein Hersteller exakt an die Standards, sondern diese erweitern oder schränken den Standard ein, sodass produktspezifische SQL-Befehle zu erlernen sind.

Die SQL-Kommandos werden in verschiedene Kategorien eingeteilt, abhängig von ihrem Einsatzzweck. Im Allgemeinen unterscheidet man folgende Kategorien:

- Datenbankkontrolle (DCL)
- Datendefinition (DDL)
- Datenmanipulation (DML)

Die folgende Tabelle listet die wichtigsten SQL-Statements nach den oben angeführten Kategorien auf – die Tabelle bezieht sich auf MySQL (s. Kapitel 3.6.5):

| ***Operationen*** | ***SQL-Statement*** | ***Beschreibung*** |
|---|---|---|
| **Datenbankkontrolle (DCL)** | | |
| DB-Administration | GRANT | Zugriffsrechte definieren |
| | REVOKE | Zugriffsrechte aufheben |

7 ANSI: American National Standard Institute
8 ISO: International Standard Organization

| ***Operationen*** | ***SQL-Statement*** | ***Beschreibung*** |
|---|---|---|
| DB anlegen, löschen und auswählen | CREATE DATABASE<br>DROP DATABASE<br>USE | DB anlegen<br>DB löschen<br>DB auswählen |
| **Datendefinition (DDL)** | | |
| Tabellen und Indexe anlegen, löschen und ändern | CREATE TABLE<br>DROP TABLE<br>ALTER TABLE<br>CREATE INDEX<br>DROP INDEX | Tabelle definieren<br>Tabelle löschen<br>Tabelle ändern<br>Index definieren<br>Index löschen |
| Datenbanken, Tabellen und Tabellenstruktur anzeigen | EXPLAIN<br>SHOW | Tabellenstruktur anzeigen<br>Anzeige von Infos zu DB, Tabellen etc. |
| **Datenmanipulation(DML)** | | |
| Datenbankabfrage | SELECT | Datenbankabfragen |
| Daten in Tabellen eingeben, ändern und löschen | INSERT<br>DELETE<br>UPDATE<br>LOAD DATA | Daten in Tabellen einfügen<br>Datensätze löschen<br>Einträge verändern<br>Daten aus einer Datei laden |

Der mächtigste und für den Datenbankbenutzer wahrscheinlich wichtigste Befehl ist die SELECT-Anweisung, die zur Erstellung von Abfragen dient – daher wird die Datenbankabfrage innerhalb der DML manchmal als eigene Kategorie betrachtet: DQL (=**D**ata **Q**uery **L**anguage).
Praktische Beispiele im Umgang mit SQL sind dem Kapitel 4 zu entnehmen.

Eine aus geographischer Sicht interessante Erweiterung des SQL92-Standards stellt die OpenGIS-SQL92 Erweiterung dar, die vom OpenGIS-Konsortium (OGC, s. www.opengis.org) eingebracht wurde.
Unter dem Namen "OpenGIS®Simple Feature Specification For SQL" hat das OpenGIS-Konsortium eine Spezifikation erarbeitet, die die Speicherung, Verwaltung, Handhabung und Abfrage von räumlichen Objekten mittels SQL ermöglicht.
Attributdaten werden dabei weiterhin mittels SQL92 bearbeitet und abgefragt, für räumliche Daten wurden neue Abfragemöglichkeiten geschaffen, wobei zwischen Methoden zum Testen der räumli-

chen Beziehungen und Methoden zur räumlichen Analyse unterschieden wird. (s. OGC, 1999)

### 3.6.5 Das RDBMS MySQL

MySQL (www.mysql.com) ist das im Umfeld von Web-Anwendungen beliebteste relationale Datenbanksystem, welches seit 1995 unter diesem Namen vertrieben wird. MySQL steht unter der GPL und wird von der gleichnamigen schwedischen Firma weiterentwickelt und vertrieben – MySQL ist für den nicht-kommerziellen Einsatz kostenlos, d.h. unter der Voraussetzung, dass die auf MySQL basierende Anwendung ebenfalls der GPL unterliegt. (s. Schulz & Siering, 2003)
Einen Vorteil von MySQL stellt die uneingeschränkte Portabilität des Datenbanksystems dar; der Datenbankserver läuft problemlos in UNIX/LINUX-Umgebungen oder in Windows-Umgebungen.
Eines der Ziele bei der Entwicklung von MySQL war die Schaffung eines stabilen DBS mit einem schnellen Zugriff auf eine große Zahl von Datensätzen, sodass zwei wichtige Kriterien implementiert wurden:

- Client-Server Architektur: Der Datenbankserver verwaltet zentral die Daten und bearbeitet die Queries verschiedener Clients
- Multithreading: Der Server ist in der Lage mehrere Queries gleichzeitig abzuarbeiten.

Nachteilig an MySQL ist die Tatsache, dass gegenüber anderen SQL-DBS einige Eigenheiten zu beachten sind. MySQL besitzt eingeschränkte SQL-Fähigkeiten, Funktionen wie Stored Procedures, Triggers oder Subqueries sind nicht vorhanden.
Einen Überblick über die Abweichungen zum SQL92-Standard findet man in der mit der Distribution mitgelieferten Reference Manual[9] im Kapitel „How Standards-compatible Is MySQL?“.
MySQL zeichnet sich durch eine hohe Zahl von API's aus, die einen Zugriff mittels Programmiersprachen wie C, PHP (s. Kapitel 3.7) oder Perl ermöglichen.

### 3.6.6 Das objektrelationale DBMS PostgreSQL

PostgreSQL (www.postgresql.org) geht auf das Jahr 1986 zurück, als an der Universität Berkley ein Datenbank-Projekt mit dem Namen Postgres ins Leben gerufen wurde. 1988 wurde das Produkt erstmals vorgestellt und bis zum Jahr 1993 ständig weiterentwickelt. 1995 kam der erste Postgres-Nachfolger mit SQL Interpreter heraus, der unter dem Namen Postgres95 bekannt ist. Seit 1996 trägt das Produkt den Namen PostgreSQL.
PostgreSQL unterscheidet sich von MySQL nicht nur technisch, da es sich um ein objektrelationa-

9 Die Reference Manual befindet sich auf Linux-Systemen in der Regel im Directory /usr/share/doc/MySQL-Version

les DBS (ORDBMS) handelt, sondern auch bzgl. des Lizenzmodells, da es dem BSD-Lizenzmodell unterliegt.
PostgreSQL läuft im Gegensatz zu MySQL unter Windows nur unter einer Art Unix-Emulation.
Die Unterstützung von SQL92 kann als weitgehend standardkonform bezeichnet werden, SQL99 wird noch nicht vollständig unterstützt. Teilweise wurde bei PostgreSQL der Funktionsumfang gegenüber dem Standard erweitert – eine Erweiterung mit geographischer Relevanz stellt das Indizierungsverfahren „R-Tree“ dar, welches speziell für Tabellenspalten mit geometrischen Daten, die PostgreSQL unterstützt, verwendet werden kann. Damit und anderen Operatoren und Funktionen lassen sich geographische Koordinaten verwalten und beispielsweise formulieren welche Punkte innerhalb eines Rechtecks sind. (s. Drake & Worsley (2002), Schulz & Siering (2003))
Als objektrelationales DBS weist PostgreSQL Merkmale wie Vererbung von Objektentitäten, Speicherung nicht atomarer Daten und die Definition eigener Datentypen, Operatoren und Funktionen auf.

### 3.6.7 Datenbanken und XML

Das Beispiel für eine einfache XML-Datei in Kapitel 3.3.1 erinnert bereits sehr stark an eine Datenbank – das root-Element <teilnehmer> würde dabei einer Tabelle in einem relationalen DBS entsprechen, die Inhalte zwischen den <person>-Tags stellen die einzelnen Datensätze in der Tabelle dar. Die Tags zwischen <person> und </person> repräsentieren die Felder mit den Inhalten.
Diese vereinfachte Beschreibung veranschaulicht die Möglichkeit mit XML Daten zu strukturieren, wie es auch in relationalen DBS mittels Tabellen und Attributen geschieht.
Breutmann (2001) geht davon aus, dass XML den nächsten Evolutionsschritt für Datenbanken auslöst.
XML eignet sich sehr gut für die Verwendung auf verschiedenen Hardware- und Software-Plattformen und speziell im Zusammenhang mit Datenbanksystemen für den Informationsaustausch zwischen heterogenen Datenbanksystemen.
Breutmann (2001) kann sich auch vorstellen relationale und objektrelationale Strukturen mit XML zu beschreiben - „wir setzen XML-Dokumente dann nicht mehr als Objekte in relationalen oder objektrelationalen Datenbanken ein, sondern wir machen die Datenbankanwendungen zu XML-Dokumenten.“ (Breutmann, 2001)

## *3.7 Hypertext Preprocessor (PHP)*

Nachdem in den vorhergehenden Kapiteln clientseitige Technologien und Datenbankserver vorgestellt wurden folgt nun als quasi fehlendes Bindeglied die serverseitige Programmiersprache PHP (www.php.net - Hypertext PreProcessor – ursprünglich: **P**ersonal **H**ome-**P**age Tools), um die einzelnen Komponenten zusammenzuführen.

### 3.7.1 Grundlagen

PHP geht auf das Jahr 1995 zurück und gilt derzeit neben Perl als eine der leistungsfähigsten und umfangreichsten Sprachen zur Entwicklung von Webanwendungen. PHP-Skripte sind im Gegensatz zu CGI-Anwendungen keine separaten Dateien, sondern PHP-Code kann direkt in HTML-eingebettet werden und wird vom Server an den PHP-Compiler weitergeleitet, sodass der Client nur den mittels PHP generierten Code (HTML) erhält und den eigentlichen PHP-Code nicht sieht.
Die Kombination von HTML und Skriptsprache wird in der Regel als Vorteil gesehen (z. Bsp. bei Däßler (2001), Wigard (2001)), dennoch kann die Vermischung irritierend sein und erfordert eine konsequente Kommentierung des Codes !
PHP bietet wie alle modernen Programmiersprachen Objektorientierung, umfangreiche Funktionsbibliotheken und die Unterstützung von Datenbankanbindungen.
Der Syntax von PHP ist an C angelehnt, jedoch sind auch einige Konzepte aus Java und Perl in der Syntax erkennbar.
PHP erfüllt alle Anforderungen, die man eine Programmiersprache stellt, wie Variablen, Programmablaufstrukturen, Zeichenkettenoperationen, Modularisierung usw.
Einen Einblick in die Syntax von PHP soll das folgende Beispiel geben, dabei handelt es sich um eine einfache „Berechnungsapplikation“ mit Benutzereingaben über ein Formular:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
  <title>PHP-Info</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <meta name="GENERATOR" content="Quanta Plus">
</head>
<body>
<h2>2 Zahlen eingeben und einen Operator auswählen</h2>
 <form action="addition.php" method="POST">
   <input type="text" name="a">
   <input type="text" name="b"><br>
```

```
   <input type="radio" name="rart" value="1">Addition<br>
   <input type="radio" name="rart" value="2">Multiplikation<br><br>
   <input type="submit" name="berechnen" value="berechnen">
 </form>
</body>
</html>
```

Dieser einfache HTML-Code besteht im wesentlichen aus einem Formular, welches die Eingabe zweier Zahlen und die Auswahl einer Rechnungsart ermöglicht. Der „Berechnen“-Button übergibt die Werte (Zahlen und Rechnungsart) an das PHP-Skript „addition.php“:

```
<?php
If ($rart==1) {
$ergebnis=$a+$b;
$operator="Addition";};

If ($rart==2) {
$ergebnis=$a*$b;
$operator="Multiplikation";};

print "Die $operator von $a und $b ergibt <b>$ergebnis</b>";
?>
```

Aus dem Beispiel geht hervor, dass ein PHP-Skript mit „<?php“ eingeleitet wird und mit „?>“ abgeschlossen wird – dies kann inmitten von HTML-Code erfolgen.

Der Rest des Skripts besteht aus zwei einfachen „if“-Abfragen und der Ergebnisausgabe mittels „print“, wobei das Ergebnis durch den HTML-Tag „<b>“ fett dargestellt wird. Da das Skript am Server ausgeführt wird, bekommt der Nutzer in seinem Browser nur folgenden Code bei der Quelltextansicht zu sehen:

```
Die Addition von 5 und 5 ergibt <b>10</b>
```

Auf ein vollständiges HTML-Grundgerüst wurde im Beispiel verzichtet.

## 3.7.2 Datenbankanbindung

Eine der großen Stärken von PHP ist die Vielfalt von Funktionsbibliotheken für die Anbindung von Datenbanken an das Internet – darunter ADABAS, MySQL, ORACLE, Informix, PostgreSQL und viele andere.

Als Referenz zu PHP und den API-Funktionen kann die Reference Manual der PHP Documentation Group (Bakken et al, 2001) empfohlen werden.

Der folgende einfache PHP-Code repräsentiert den typischen Fall des Auslesens von Daten aus

einer PostgreSQL-DB und einer einfachen Ergebnisdarstellung – wiederum ohne HTML-Grundgerüst:

```
<?php
// Verbindung zu einer PostgreSQl-DB mittels PHP - Florian Jurgeit, 2002/2003
print "<b>Ein erster TEST:</b><hr>";
$connect = pg_connect("host=localhost dbname=flo1 user=flo");
$query = "select * from wstation;";
$result = pg_exec($query);
$n = pg_numrows($result);
print "<u>Zahl der Datensätze:</u> $n <br>";
for ($i=1;$i<$n;$i++)
 {
  $o = pg_fetch_object($result);
  print "$o->name, $o->temp_min, $o->temp_max, $o->datum<br>";
 };
pg_close($connect);
?>
```

Die Funktion `pg_connect` stellt die Verbindung zum Datenbankserver her – diese wird in der Variable $connect gespeichert um am Ende des Skripts die Verbindung wieder ordnungsgemäß zu schließen (`pg_close`).

In der Variablen $query steht das auszuführende SQL-Kommando, welches mit `pg_exec` ausgeführt wird – die Variable $result enthält das Ergebnis.

Die Funktion `pg_numrows` ermittelt die Zahl der Treffer-Datensätze – die Variable $n speichert diese Zahl.

Der wichtigste Teil des Codes ist die for-Schleife, die alle Trefferdatensätze (1 bis $n) durchläuft und mit der Funktion `pg_fetch_object` jeden Datensatz als Klasse(Objekt) mit den Feldern als Membervariablen ausliest – in der Variable $o steht quasi jeder Datensatz mit Feldnamen und Daten. Auf die einzelnen Inhalte der Felder kann mit “$o->Feldname” zugegriffen werden – siehe print-Anweisung in der for-Schleife.

Wenn alle Treffer-Datensätze durchlaufen sind wird mit `pg_close` die Verbindung zum Datenbankserver geschlossen.

Alternativ zu `pg_fetch_object` wird häufig ein 'manuelles Durchlaufen' der Tabelle in zwei verschachtelten for-Schleifen angewandt, dabei wird zeilenweise Feld für Feld ausgelesen.

### 3.7.3 SVG dynamisch mit PHP erzeugen

PHP eignet sich auch zur Generierung von SVG-Code, wobei dabei die Möglichkeit besteht beispielsweise Daten aus anderen Datenquellen (z. Bsp. Datenbanken) einzubinden und somit statischen SVG-Code mit dynamischen Inhalten zu vermischen.

In der Regel muss man jedoch zuvor die Konfigurationsdateien des Webservers verändern – die folgenden Angaben beziehen sich auf Apache mit installiertem und funktionierendem PHP unter SuSe-Linux, Version 8:

Die SVG-Dateien mit PHP-Code werden mit der Extension *.psvg abgespeichert, der Web-Server muss somit so konfiguriert werden, dass Dateien mit dieser Extension dem Mime[10]-Type „image/svg+xml“ entsprechen und dem PHP-Compiler zugeführt werden.

Mime-Types sind in der Regel in „/etc/httpd/mime.types“ definiert – hier muss der Eintrag wie folgt aussehen: `image/svg+xml svg svgz psvg`

In der Konfigurationsdatei „httpd.conf“ ergänzt man den Eintrag zur Verarbeitung von php-Dateien um den Typ *.psvg: `AddType application/x-httpd-php .php .php4 .psvg`

Auf carto.net (http://www.carto.net/papers/svg/serverside_svg_php_e.html) wird diese Thematik in einem Artikel ausführlich behandelt, wobei bei der Erstellung der Skripts an sich auch noch einige Details zu beachten sind – ein umfangreicheres Beispiel bietet Behme (2003). Prinzipiell ist die Vorgangsweise gleich wie im Umgang mit HTML-Dateien: In den SVG-Code kann mit <?php ... ?> an jeder Stelle PHP-Code eingefügt werden – die Ausgabe von SVG-Code innerhalb des PHP-Teils erfolgt mittels „print“ oder „echo“:

```
print "<text x='$textx' y='$texty' style='fill:blue'>$o->name: $o->text (<a
xlink:href='http://$o->url' target='_blank'>$o->url</a>)</text>";
```

Basierend auf dieser Technik können beispielsweise auch in einer Datenbank gespeicherte Geometrien visualisiert werden !

10 MIME: Multipurpose Internet Mail Extension

# 4 Prototyp Rauminformationssystem „UniRIS“

## 4.1 Vorstellung des Projekts

Die Universität Innsbruck verfügt über zahlreiche Gebäude, die im Stadtgebiet verteilt sind, sowie über teilweise große Etagenflächen mit den Büros, die oft schwer zu finden sind. Die Abteilung für Gebäudeverwaltung hat Etagenpläne aller Gebäude im AutoCAD-Format, die bis dato außer für interne Zwecke nicht genutzt wurden.

Es war nun die Idee den Prototyp eines Informationssystems zu erstellen, das die vorhandenen Daten verwendet und dabei folgende Hauptfunktionen dem Nutzer bietet:

- *Suche nach einer Person in einer Datenbank ▸▸▸ Visualisierung des Raums in einem Etagenplan.*
- *Suche im Etagenplan (Grafische Suche) ▸▸▸ Abfrage der Informationen zum Raum (Nutzer+zukünftig evtl. andere Daten)*

Eine weitere Anforderung an die Applikation ist die Verwendung über das Internet, die Grundfunktionalitäten sollen jedem zugänglich sein, die verwendete Software sollte nicht mit Lizenzkosten verbunden sein und das System (Datenmodell) sollte erweiterbar sein.

Als Datenbanksystem war somit ein Datenbankserver gefragt, wobei grundsätzlich MySQL und PostgreSQL zur Auswahl standen – aufgrund der Installierung eines neuen Servers auf Linux-Basis mit PostgreSQL und PHP fiel die Entscheidung zu Gunsten von PostgreSQL (s. Kapitel 3.6.6).

Zur Visualisierung der vorhandenen CAD-Daten wurde der Versuch gewagt SVG zu verwenden, da SVG gegenwärtig das einzige offizielle Web-Vektor Format ist (s. Kapitel 3.3).

Die folgenden Kapitel dokumentieren das Datenmodell, das technische Umfeld, die Datenaufbereitung (AutoCAD ▸ SVG) und die Systemarchitektur und Funktionsweise des Prototyps.

## 4.2 Datenmodell

Das Datenmodell des Prototyps basiert auf dem Relationenmodell, da dieses Datenmodell das rezent gebräuchlichste ist (s. Kapitel 2.2) und PosgresSQL 'nur' ein objektrelationales DBS ist, welches das Relationenmodell um objektorientierte Eigenschaften erweitert. Ein weiterer Vorteil besteht in der Möglichkeit der Verwendung des Datenbestandes zur Anbindung an das CAD-System mittels ODBC[11], sodass die Daten auch als Expertensystem in AutoCAD verwendet werden können.

11 ODBC: Open Database Connectivity

Grundlage des Datenmodells ist die Gliederung der Realwelt in zwei Hauptgruppen:

- Gebäude + Räume
- Nutzer (Personen)

Die Gebäude werden dabei hierarchisch betrachtet: Die oberste Stufe sind die Gebäude, die wiederum aus Räumen bestehen; den Räumen sind dann wiederum Personen (Nutzer) oder andere Sachdaten zuordenbar.

Zwischen Gebäuden und Räumen besteht eine Funktionale Abhängigkeit (1:n Objektbeziehung; s. Kapitel 3.6.3) – oder vereinfacht ausgedrückt: Jedem Gebäude sind mehrere Räume zuordenbar, jedem Raum ist nur ein Gebäude zuordenbar.

Diese Tatsache führt dazu, dass die Gebäude-ID (Primärschlüssel der Tabelle „Gebäude“) in der Tabelle „Raum“ als Fremdschlüssel definiert wird.

Die Objektbeziehung zwischen Räumen und Personen (Nutzern) stellt sich komplizierter dar, da jedem Raum mehrere Personen zugeordnet werden können (z. Bsp. Büros von Arbeitsgruppen), aber auch vice versa einer Person mehrere Räume zugeordnet werden können (z. Bsp. Büro und Projektraum) – es handelt sich somit um eine komplexe Abhängigkeit (m:n Objektbeziehung), die eine Auflösung in zwei 1:n-Beziehungen durch eine Relation erfordert.

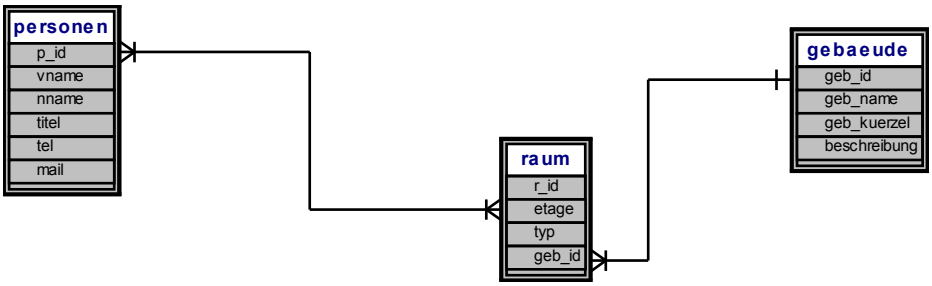


*Abbildung 34: ERD des Prototypen (erstellt mit DeZign)*

Das Datenmodell ermöglicht auch die einfache Anbindung zusätzlicher Sachdaten, wie beispielsweise die Verwaltung der PC's – dabei würde es sich wiederum um eine 1:n-Beziehung handeln, die lediglich die Integration der Raum-ID als Fremdschlüssel erfordert.

Betrachtet man das Datenmodell, so wird schnell klar, dass die geforderten Fragestellungen durch Abfragen durchführbar sind, des weiteren kann jeder Etagenplan eindeutig über die Gebäude- und Etagenangabe zugeordnet werden; für die Dateinamen der Etagenpläne wurde dabei folgende Konvention gewählt: „Gebäudekürzel+Etage.svg“ – beispielsweise „bs7.svg“ für das Bruno-Sander-Haus, Etage 7.

Die Raum-ID's sind jeweils in der Geometrie verankert (s. Kapitel 4.3), sodass der Zugriff auf den Raum an sich problemlos möglich ist.

Die Raum-ID in der Tabelle Raum entspricht den realen bereits vorhandenen Raum-ID's der Univ. Innsbruck, sodass eventuell vorhandene Datenbestände mit Bezug auf die Raum-Nummer integriert werden können.

## *4.3 Daten und Aufbereitung*

Die Geometriedaten sind bereits in Form von Etagenplänen im AutoCAD-Format vorhanden gewesen, sodass eine Verwendung dieser Daten einer Neuerstellung vorzuziehen war.

Die CAD-Daten wurden seitens der Gebäudeverwaltung im DWG-Format zur Verfügung gestellt:

```
Innrain 52 BT 6 7 Og 11-01-03.dwg
```

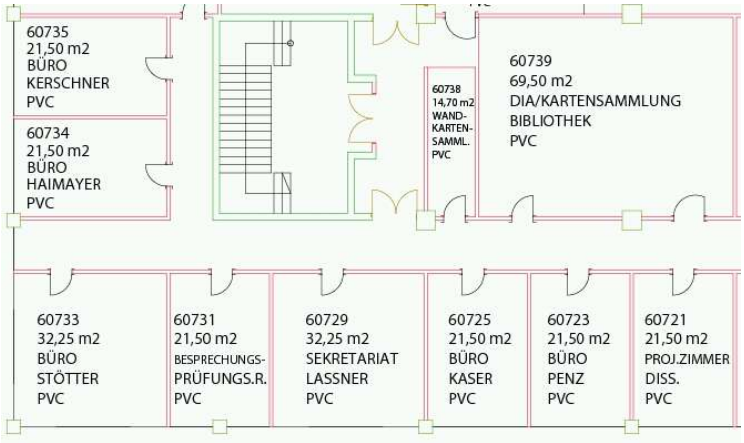


*Abbildung 35: Ausschnitt aus den Originaldaten*

Die erste zentrale Frage galt der Qualität der Daten und der Konvertierung in das SVG-Format, wobei aufgrund des Datenmodells von Anfang an die Bedingung einer „geschlossenen“ Geometrie der Räume zu erfüllen war.

Die Sichtung der Originaldaten hat ergeben, dass sich die Räume aus einzelnen Linien zusammensetzen, die die Wände repräsentieren, jedoch keine geschlossene Raumgeometrie (geschlossenes Polygon) vorhanden war.

Ein weiteres Problem stellten direkt in die Pläne eingetragene Sachdaten dar, ein scheinbar sehr häufiges Phänomen (s. Kapitel 2.2.2) bei dem eigentlich in Karteien (Datenbanken) zu verwaltende Informationen in Plänen eingetragen werden.

Aufgrund der fehlenden geschlossenen Geometrie kam die Idee diese mittels der ArcGIS[12]-Funktionen „clean/build“ zu erstellen. Der Import der *.dwg-Dateien in ArcGIS 8 funktioniert problemlos – nach dem Import wurden die Layers mit den Sachdaten (Text) entfernt und anschließend die Polyline-Layer in eine Geodatabase konvertiert. Zur Anwendung der „clean/build“-Funktionen

12 ArcGIS wird wie in der neueren Literatur als Synonym für Arc/INFO verwendet.

muss die Geometrie als Coverage vorliegen, sodass ein weiterer Konvertierungsschritt notwendig war.

Die Coverage kann dann mittels clean/build in eine Geometrie mit geschlossenen Polygonen überführt werden.

Die build-Funktion von ArcGIS geht davon aus, dass die Koordinaten-Angaben korrekt sind, die clean-Funktion 'untersucht' die Geometrie und setzt Knoten an Kreuzungspunkten von Linien, des Weiteren werden die so genannten „Overshoots“ und „Undershoots“ innerhalb einer definierten Toleranz korrigiert.

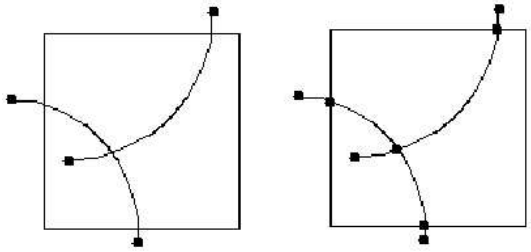

*Abbildung 36: Intersections erstellen (aus: Arc/Info Hilfe)*

ArcGIS verwaltet intern Polygone als eine Liste von Linienelementen, die das Polygon bilden, und einem Label.

Die clean-Funktion ermöglicht eine Korrektur der Geometrie und die Erstellung von Polygonen aus einer Liniengeometrie, wie sie für den Prototypen benötigt wird:

```
CLEAN <in_cover> {out_cover} {dangle_length} {fuzzy_tolerance} {POLY | LINE}
```

Dieser Syntax bezieht sich auf das Modul ARC, der Aufruf im Modul ARCEDIT bietet noch zusätzliche Parameter.

Die ersten beiden Parameter betreffen die Eingabe- und Ausgabe-Datei – hierbei ist lediglich zu beachten, dass per Default bei Nichtangabe eines Dateinamens für den Output die ursprüngliche Datei überschrieben wird.

Die Angabe der dangle-length führt dazu, dass 'überstehende' Linien, die auf jeder Seite dem selben Polygon zugeordnet werden können und in einem 'alleinstehenden' Knoten (Node) enden und die angegebene Länge unterschreiten gelöscht werden.

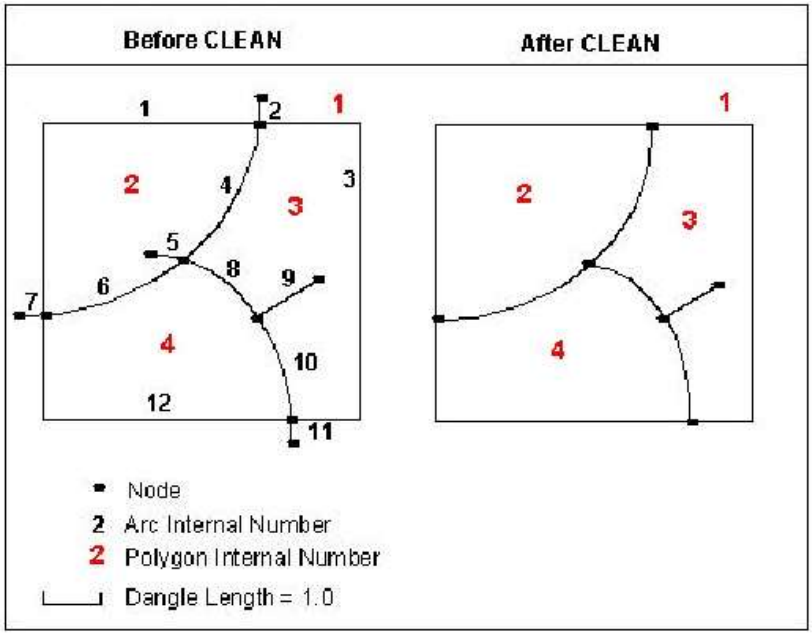


*Abbildung 37: Dangle-Clean (aus: Arc/Info Hilfe)*

Die fuzzy-tolerance definiert die minimale Distanz zwischen Vertices einer Linie. Dies bewirkt dass alle Vertices, die innerhalb der Toleranz liegen, zu einem Punkt zusammengefasst werden.

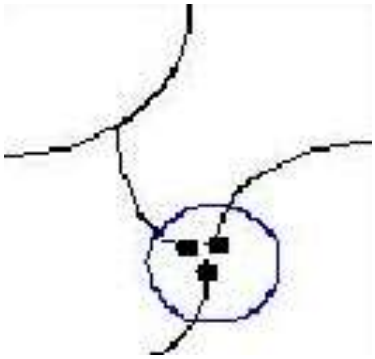

*Abbildung 38: Fuzzy-Tolerance (aus: Arc/Info Hilfe)*

Ein Problem in diesem Zusammenhang besteht darin, dass in den Plandateien nicht nur die gewünschten Räume, sondern auch beispielsweise die Zwischenwände zu Polygonen werden.
Als weiteres Problem stellt sich eine teilweise 'lückenhafte' Geometrie in den Plänen dar, die eine Bildung von Polygonen verhindert – hierbei stehen dann zwei Möglichkeiten zur Verfügung: Eine manuelle Korrektur oder eine Änderung der fuzzy-tolerance; letzteres führt zu einer teilweisen Verschlechterung der Geometrie, insbesondere im Bereich der Türstöcke.

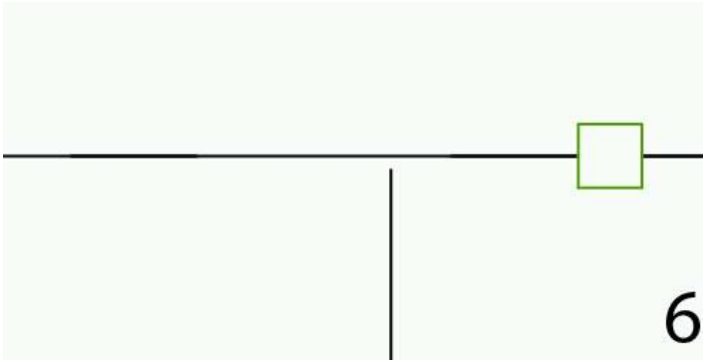


*Abbildung 39: Lücken in den CAD-Plänen*

Nach der Bildung der Polygon-Topologie stellt sich die Frage des Exports in das SVG-Format – dabei sind zwei Möglichkeiten in Frage gekommen:

- Konvertierung mittels Perl-Skript oder
- die Konvertierung mittels eines Avenue Skripts.

Das Avenue-Skript „shp2svg“ von Nedjo Rogers ist frei verfügbar (OpenSource) und hat sich für diesen Zweck als brauchbar erwiesen.

Die Weiterverarbeitung der Coverage in ArcView erfordert eine vorhergehende Abspeicherung als Shapefile (*.shp).

Die Konvertierung in eine SVG-Datei mittels dem Skript erfordert eine Nachbearbeitung um die Datei für den Prototypen nutzbar zu machen.

Diese Änderungen sind mit einem Texteditor (z. Bsp. Kate unter Linux, Textpad unter Win32), der reguläre Ausdrücke (Regular Expressions – RE) verarbeiten kann, problemlos durchzuführen.

RE stellen eine Art Erweiterung der „Suchen-Ersetzen“-Funktionalität dar, es wird dabei eine Art „Mustererkennung“ eingesetzt, wobei spezielle Zeichen – meist Kombinationen mit Sonderzeichen – Funktionalitäten repräsentieren: `\t` steht beispielsweise für Tabulatoren, `\n` für Zeilenumbrüche.

Sucht man mittels regulärer Ausdrücke in einer Datei nach `\t` und ersetzt diese durch `\n` werden alle Tabulatoren durch Zeilenumbrüche ersetzt.

Aufgrund des Datenmodells und der Systemarchitektur (s. Kapitel 4.5) hat sich folgende Anforderung an die SVG-Dateien ergeben:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/2000/svg10-20000303-stylable"
[<!ENTITY
style "fill:lightblue;fill-rule:evenodd;stroke:rgb(0,0,0);stroke-
width:0.90000000pt;stroke-antialiasing:true;">
```

```
]>
<!-- Generator: ArcView Avenue script shp2svg; modified with Textpad -->
<svg id="map" width="627.00000000px" height="583.00000000px" viewBox="0 0
6270000.00000000 5830000.00000000">
<desc>View1</desc>

<g id="plan" style="&style;" onclick="showinfo(evt)“>
<path id="RaumID"
 d="M 332030 1829593 L 332030 1853189 L 332030 1876785 L 308515 1876785 L 285000
1876785 L 285000 1829593 L 332030 1829593 z" />
...
</g>
</svg>
```

Die Nachbearbeitung besteht im wesentlichen darin, die Event-Handler aus den einzelnen Pfaden zu entfernen und um die Pfade den <g>-Tag zu setzen. In den <g>-Tag wird ein Event-Handler integriert („onClick“), der die Funktion „showinfo“ auslöst.

## *4.4 Technische Umsetzung*

Vor der technischen Umsetzung steht das Datenmodell und die Entscheidung für bestimmte Komponenten, die das Gesamtsystem bilden und anschließend in die Systemarchitektur integriert werden.

Die wesentliche Komponente stellt das Datenbanksystem dar, welches die Sachdaten verwaltet und ihre Bearbeitung ermöglicht. Neben der Datenbank werden ein Web-Server, die Benutzerschnittstelle und eine Datenbankschnittstelle benötigt.

Die Zusammenstellung der vom Benutzer angeforderten Informationen wird von einer serverseitigen Skriptsprache übernommen, die das Ergebnis als HTML-Seite mit eingebettetem Plan (SVG) ausliefert.

In Kapitel 4.1 wurden bereits einige der Komponenten angesprochen, dennoch sei generell darauf hingewiesen, dass bewusst freie Software eingesetzt wurde, die im Gegensatz zu proprietären (kommerziellen) Lösungen nicht nur kostenlos ist, sondern meist einen weiteren Spielraum ermöglicht. Die Entscheidung für LINUX als Betriebssystem des Servers beeinflusst die Wahl der restlichen Softwarekomponenten maßgeblich.

Die häufig getätigte Aussage, dass freie Software nur für Spezialisten geeignet sei, da schlecht bis überhaupt nicht dokumentiert und schwer implementierbar, ist von der Hand zu weisen.

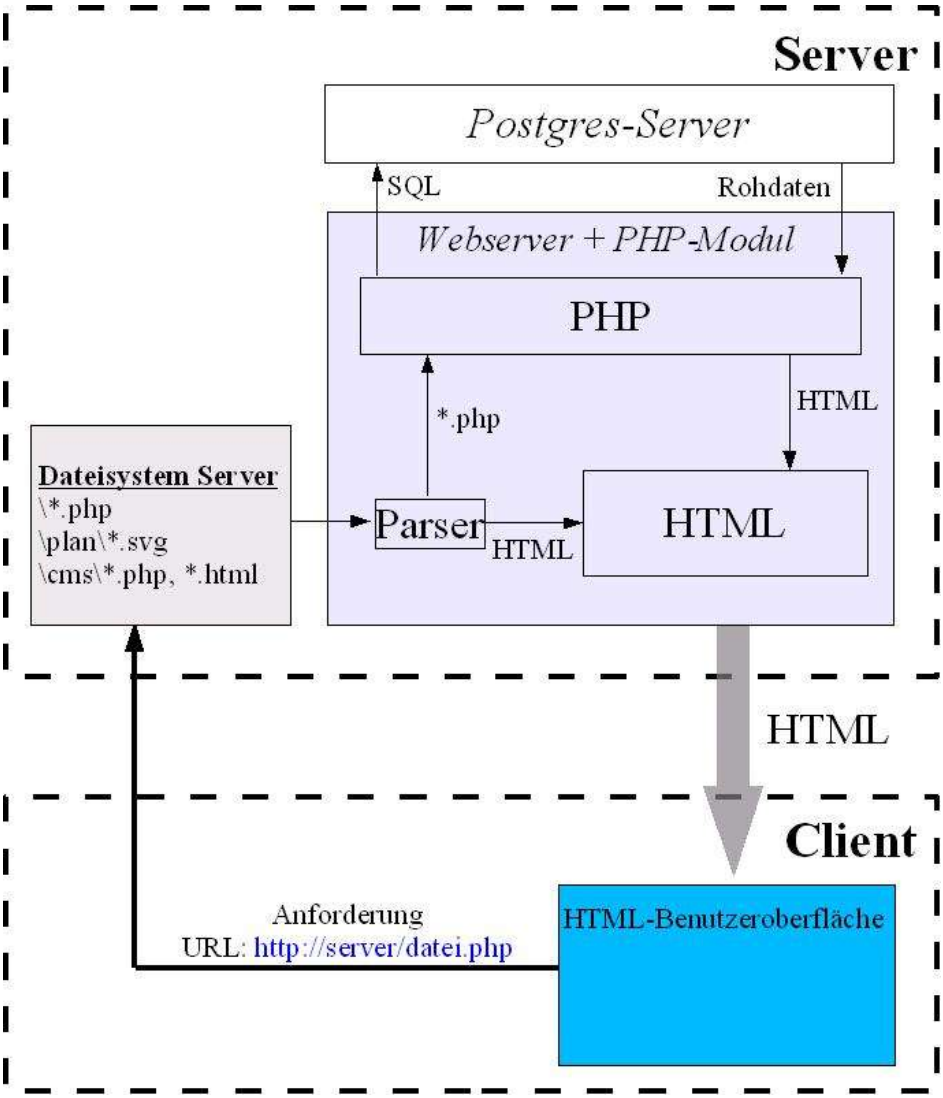


*Abbildung 40: Technischer Aufbau des Prototypen (eigener Entwurf)*

### 4.4.1 Datenbankmanagementsystem

Als Datenbankmanagementsystem wird wie bereits erwähnt PostgreSQL eingesetzt (s. Kapitel 3.6.6), dessen Kern der Datenbankserver „postgresql“ darstellt. Hierbei ist zu beachten, dass der Server standardmäßig ohne Zugriffsmöglichkeit via TCP/IP gestartet wird !

Um via TCP/IP auf den Datenbankserver zugreifen zu können muss der ebenfalls mitgelieferte Multiuserdatenbankserver „postmaster“ mit der Option „i“ gestartet werden: `postmaster -i`

Der Client „psql“ ermöglicht die kommandozeilenorientierte Datenbankkommunikation mit dem Server.

Neben diesen Hauptkomponenten werden zahlreiche Dienstprogramme mitgeliefert, darunter das Programm „pg_dump“ zur Sicherung einer Datenbank inklusive der DB-Struktur. (s. Drake &

Worsley, 2002)

Neben der Kommandozeilen basierten Arbeit mit dem DB-Server wird häufig der grafische Client „PgAccess“ eingesetzt, der mit Tcl/Tk entwickelt wurde und somit Cross-Plattform tauglich ist.

Die Datenbankkommunikation übernimmt im Prototyp die Postgres-API von PHP, die Verwendung der ODBC-Schnittstelle wäre ebenfalls möglich.

### 4.4.2 Webserver und PHP

Als Webserver ist Apache (http://www.apache.org) zum Einsatz gekommen, der im Zusammenhang mit Linux als Betriebssystem ein stabiles System bietet. Apache wurde um das PHP-Modul (http://www.php.net) in der Version 4 erweitert, welches die serverseitige Verarbeitung der Anfragen und die Kommunikation mit der Datenbank übernimmt. Bei manchen Distributionen muss die Postgres-Unterstützung für PHP als separates Modul installiert werden – in der Regel als Paket mit der Bezeichnung „php-pgsql-*Version*.rpm“ !

Da Apache kompatibel zum NCSA-Server ist beherrscht er das Konzept der „.htaccess“-Dateien, die unter anderem eine Möglichkeit des Zugriffsschutzes auf bestimmte Daten auf dem Webserver bieten. Es können dabei Benutzernamen und Passwörter zur Authentifizierung festgelegt werden, aber auch erweiterte Möglichkeiten wie die Zugriffserlaubnis für bestimmte IP-Bereiche (beispielsweise nur IP's innerhalb des Firmennetzes) sind machbar. (s. Münz, 2001)

Die Anwendung dieser Technik scheint für den Schutz der Dateien des Content Management Systems im Verzeichnis „/cms“ sinnvoll.

Die Weiterleitung der „*.php“-Dateien an den PHP-Compiler muss in der Apache Konfigurationsdatei „httpd.conf“ festgelegt werden – dies wird bei den gängigen Linux-Distributionen bei der Installation automatisch erledigt.

Eine fehlende Konfiguration äußert sich in der Übermittlung des PHP-Codes direkt an den Client, der diesen dann im Klartext sieht !

### 4.4.3 Benutzerschnittstelle (HTML und SVG)

Als Benutzerschnittstelle dient eine HTML-Seite, die mit Frame-Technik realisiert wurde.

Die Seite kann prinzipiell ein beliebiges Layout aufweisen, es muss nur darauf geachtet werden, dass die vorgesehenen Funktionen implementiert werden, dabei handelt es sich im wesentlichen um Eingabemöglichkeiten für die Suchbegriffe und deren Weiterleitung an die zuständigen Skripts.

Die Frame-Technik hat den Vorteil, dass im Gegensatz zu einem „Single-Page“ Design die Seite in

verschiedene Teile gegliedert wird, sodass verhältnismäßig einfach Bereiche mit einem fixen Benutzerinterface mit den Grundfunktionen und Bereiche mit der Ergebnisausgabe.
Die Visualisierung der Plandaten geschieht mittels SVG, eine Interaktivität wird durch Java-Skript Funktionen erreicht.

## *4.5 Systemarchitektur und Funktionsweise*

Die Funktionsweise und die detaillierte Systemarchitektur werden in den folgenden Kapiteln anhand der Prototyp-Applikation erläutert, dabei werden die einzelnen Skripte und ihr Aufbau schwerpunktmäßig erläutert.
Neben den in Kapitel 4.1 erwähnten Hauptfunktionen wurde die Möglichkeit der Anzeige einer Verortung des Gebäudes in einem vereinfachten Stadtplan integriert, sodass sich der Nutzer des Systems nicht nur den gesuchten Raum im Etagenplan visualisieren kann, sondern auch einen Eindruck von der Lage des entsprechenden Gebäudes bekommt.
Das Gesamtsystem kann in zwei Teile unterteilt werden:

- Dem *Informationssystem* an sich, das jedermann via Internet zur Verfügung steht
- und einem *CMS (Content Management System)*, welches der Verwaltung der Datenbasis dient (Räume definieren, den Räumen die Nutzer zuweisen, ...)

### 4.5.1 Das WWW-Informationssystem

#### *4.5.1.1 Benutzerinterface und Startbildschirm*

Das Informationssystem besteht aus einer mittels Frametechnik erstellten Basisseite, die aus drei Frames besteht. Das obere Frame dient lediglich als Titelframe, das linke Frame beinhaltet den Zugang zu den beiden Hauptfunktionen, die über ein Formular realisiert sind. Die Benutzereingaben werden an die entsprechenden PHP-Skripte übermittelt:

- Die Suchanfrage nach Personen wird an das Skript „search.php“ mit der Übergabevariable „name“ weitergeleitet.
- Die Suchanfrage nach Gebäuden wird von „search_geb.php“ mit der Variable „geb_name“ bearbeitet.

Der Vorteil dieser Aufteilung besteht darin, dass der Benutzer ständig Zugriff auf diese Hauptfunktionen und jederzeit eine neue Anfrage stellen kann.
Das rechte Frame dient der Ergebnisdarstellung der vom Benutzer angeforderten Daten.

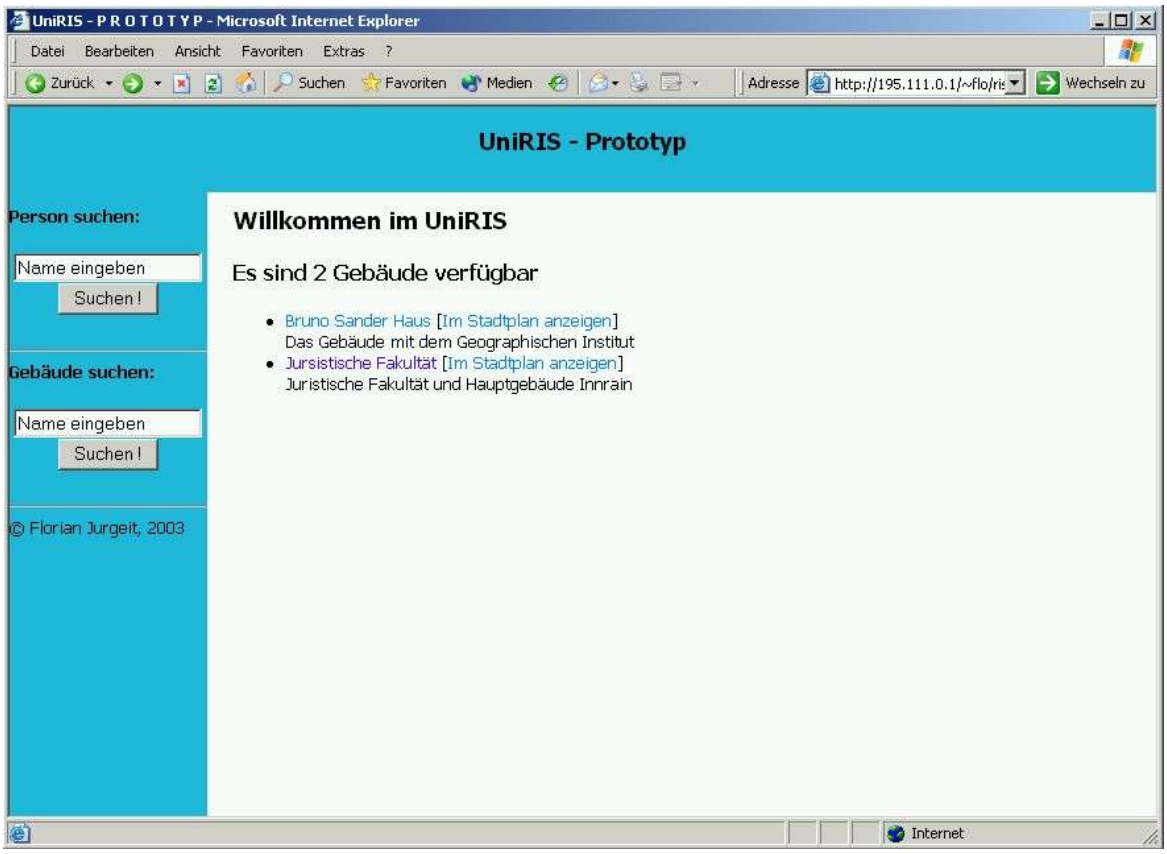


*Abbildung 41: Prototyp Startscreen*

### *4.5.1.2 Personensuche und Visualisierung*

Das Skript „search.php“ hat als Hauptaufgabe basierend auf dem Suchstring der Benutzereingabe eine Anfrage an die Personendatenbank zu stellen und die entsprechenden Treffer auszugeben.

Wesentliches Element des Skript ist die in SQL zu formulierende Anfrage an das DBS, die über die Postgres-API von PHP realisiert wird. In der Variable „$query“ wird der SQL-Abfrage-Code gespeichert, der eine einfache Abfrage mittels „select“ über mehrere Tabellen darstellt – dabei wird in der Tabelle „personen“ im Feld „nname“ nach einer Übereinstimmung mit der aus dem Formular übernommenen Variable „$name“ gesucht.

Da im Prototypen nur Treffer angezeigt werden sollen, denen auch ein Raum zugeordnet ist, werden mittels „AND“ als weitere Bedingung das Vorhandensein eines der Person zugeordneten Raumes und das entsprechende Gebäude ermittelt.

```
$query = "select * from personen, gebaeude, raum, personen_raum where personen.nname
like '$name' and ((personen.p_id = personen_raum.p_id) and (personen_raum.r_id =
raum.r_id)) and (raum.geb_id = gebaeude.geb_id);";
```

Innerhalb der „For“-Schleife werden die Ergebnisse der Abfrage ausgegeben, dabei werden neben den Angaben zur Person (Personendaten) auch Informationen über den zugeordneten Raum übermittelt (Raumnummer, Gebäude und Etage).
Die Raumnummer und das Gebäude werden als Hyperlink ausgegeben und bieten dem Nutzer Zugang zu den weiteren Informationen.

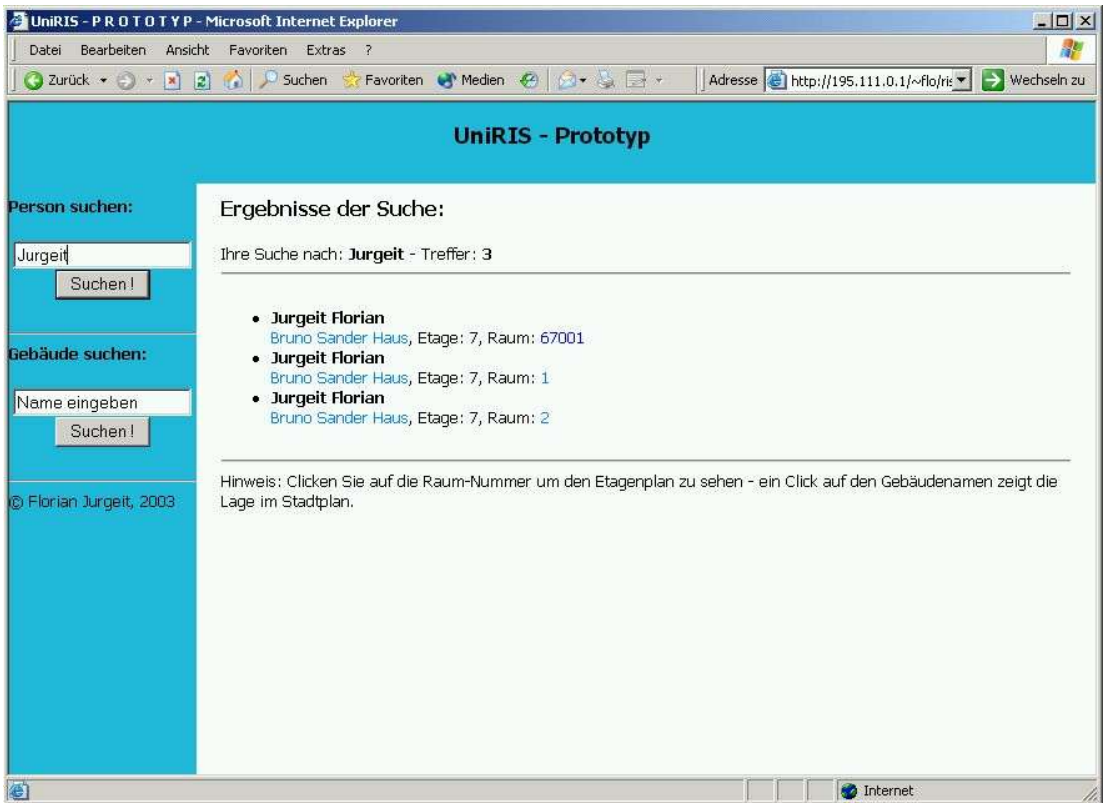

*Abbildung 42: Personensuche*

Ein Klick auf den Hyperlink übermittelt dem Skript „show_geb_stadtplan“ das entsprechende Kürzel des Gebäudes, sodass dessen Lage im Stadtplan dargestellt werden kann. (s. Kapitel 4.5.1.4)
Der Hyperlink mit der Raumnummer verweist auf das Skript „showraum“, dabei werden mehrere Übergabewerte definiert, die durch ein „&“-Zeichen getrennt werden:

```
<a href='showraum?r_id=$o->r_id&geb_kuerzel=$o->geb_kuerzel&etage=$o->etage'>
$o->r_id</a>
```

Das Skript „showraum“ generiert eine HTML-Datei, die den entsprechenden Etagenplan mittels <embed> (s. Kapitel 3.3.5) einbettet und durch Javascript-Funktionen eine Interaktivität mit dem Nutzer bietet.
Basierend auf den Übergabevariablen können die benötigten Parameter in den <embed>-Tag integriert werden – Gebäudekürzel ($geb_kuerzel) und Etage ($etage) bilden den Dateinamen der

SVG-Datei, die im Unterverzeichnis „/plan“ auf dem Server liegt.

Ebenfalls wichtig ist die Festlegung eines Namens mittels „name=...“ im <embed>-Tag, da über diesen Namen auf das eingebettete SVG-Objekt durch die Javascripts zugegriffen wird.

Der Javascript-Teil besteht aus drei Skripts:

- die Funktion „show“ dient der farblichen Hervorhebung des gesuchten Raumes und wird durch das „onload“-Event im <body>-Tag initialisiert, sodass die Hervorhebung mit der Darstellung des Dokuments ohne weitere Benutzereingaben erfolgt.

```
function show()
   {
      var doc = document.$geb_kuerzel$etage.getSVGDocument();
      var objekt2 = doc.getElementById('$r_id');
      var aktclr2 = objekt2.getAttribute('fill');
      objekt2.setAttribute('style', 'fill:yellow');
   }
```

  Das Skript steht innerhalb einer „print“-Anweisung, sodass PHP-Variablen ($...) direkt eingebunden werden können. Zuerst wird in der Variablen „doc“ das Dokument referenziert, folglich kann mittels „getElementById“ der gesuchte Raum referenziert werden. Die Methode „setAttribute“ weist schlussendlich dem gesuchten Raum die Füllfarbe Gelb zu – s. Kapitel 3.4.3, Skripting in SVG.

- die Funktion „showinfo“ ermöglicht die Abfrage von Informationen zu einem bestimmten Raum durch einen Mausklick. Dies erfordert einen Event-Handler in der SVG-Datei, der im Gruppentag der Raumgeometrien untergebracht ist (s. Kapitel 4.3 – „onClick=showinfo(evt)“).

  Die Funktion besteht aus zwei Teilen, ersterer führt zu einer farblichen Hervorhebung des angeklickten Raumes, der zweite Teil öffnet ein neues Fenster („window.open“), dessen Inhalt über das PHP-Skript „info“ ermittelt wird.

  Beiden Teile benötigen die ID des angeklickten Raumes, die über die Referenz auf das Objekt („evt.getTarget()“) und die Methode „getAttribute('id')“ ausgelesen werden kann.

  Die ID (=Raumnummer) wird dem Skript „info.php“ als Parameter übergeben, dieses führt wiederum eine Anfrage an die Datenbank durch, wobei alle dem Raum zugeordneten Personen ermittelt werden.

- „opt_size“ ist eine Hilfsfunktion, die die wahre Planausdehnung auf die Darstellungsfläche optimiert und ebenfalls über den „onload“-Event initialisiert wird.

  Dabei wird die wahre Ausdehnung des Plans, der Gruppe „plan“, über die Methode „getBBox“ ermittelt. Die Bounding-Box enthält Angaben zur Ausdehnung der Geometrie in der Form x, y und width und height. Die Viewbox wird auf die Werte der Bounding-Box gesetzt, die width-

und height-Werte des SVG-Root-Elements („<svg width=WERT height=WERT ...>“) werden auf die im <embed>-Tag definierten Ausdehnungswerte gesetzt.

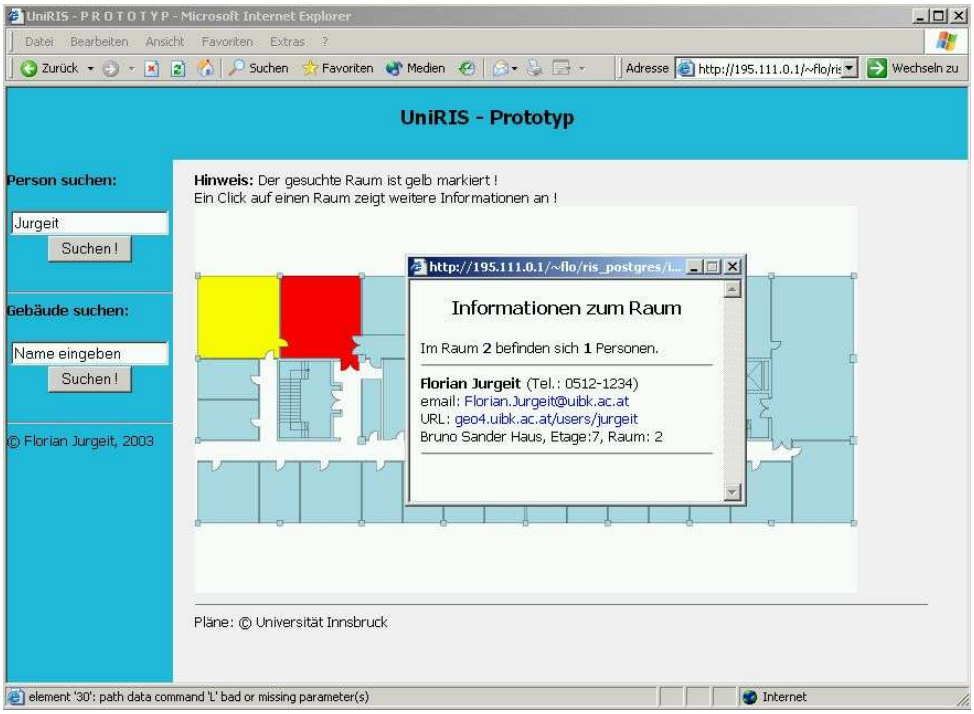

*Abbildung 43: Raumvisualisierung und Info-Fenster*

### *4.5.1.3 Gebäudesuche*

Das Skript „search_geb.php“ führt eine Suchanfrage an das DBS bezüglich einer Übereinstimmung mit dem Suchbegriff in der Tabelle Gebäude durch. Die Bedingung „raum.geb_id = gebaeude.geb_id“ führt dazu, dass nur Gebäude ausgegeben werden, denen auch Räume zugeordnet sind. Die Anweisung „distinct“ bewirkt eine nur einmalige Ausgabe jedes Gebäudes, ansonsten würde ein Gebäude entsprechend der Zahl der zugewiesenen Räume im Ergebnis erscheinen.

„Distinct“ entfernt somit Datensätze aus der Ergebnismenge, die im angegebenen Feld („distinct on (Tabelle.Feld)“) identische Werte aufweisen. (s. Drake & Worsley, 2002)

Der Nutzer erhält am Bildschirm eine Auswahl der verfügbaren Gebäude mit der Möglichkeit diese im Stadtplan anzuzeigen (s. Skript „show_geb_stadtplan“ - Kapitel 4.5.1.4), der Hyperlink hinter dem Gebäudenamen führt zu einer Übersicht der vorhandenen Etagen, dazu wird die Gebäude-ID dem Skript „getetagen“ als Parameter übergeben.

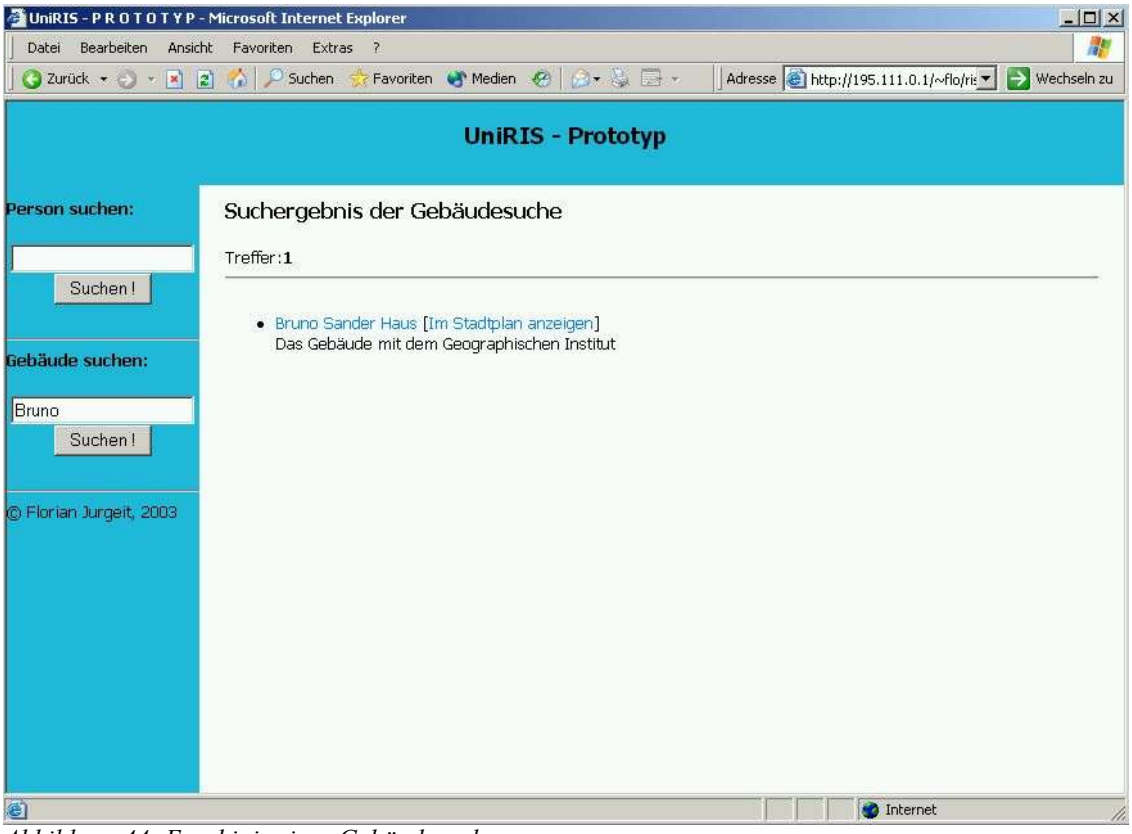


*Abbildung 44: Ergebinis einer Gebäudesuche*

Die SQL-Query im Skript „getetagen“ liefert als Ergebnis alle Etagen im gewählten Gebäude, denen auch Räume zugeordnet sind.

Die Auswahl einer Etage übergibt an das Skript „showetage“ die Variablen „$geb_kuerzel“ und „$etage“, sodass alle Parameter für die Darstellung des gewünschten Etagenplans vorhanden sind:

```
<a href='showetage?geb_kuerzel=$o->geb_kuerzel&etage=$o->etage'>Plan anzeigen</a>
```

„showetage.php“ funktioniert ähnlich dem in Kapitel 4.5.1.2 erläuterten Skript „showraum“. Der Unterschied liegt im Fehlen der Javascript-Funktion „show“, da in diesem Fall kein Raum hervorgehoben werden muss. Die Funktionen „showinfo“ und „opt_size“ sind mit denen im Skript „showraum“ identisch.

### *4.5.1.4 Stadtplanansicht*

In den Prototypen wurde ebenfalls ein Ansatz eines einfachen Stadtplans integriert, der die Lokalisierung der Gebäude ermöglicht.

Die einzelnen Gebäude sind in der Stadtplan-SVG-Datei als Objekte mit dem Gebäudekürzel als ID gespeichert, ihre Visibility ist standardmäßig auf „hidden“ gesetzt.

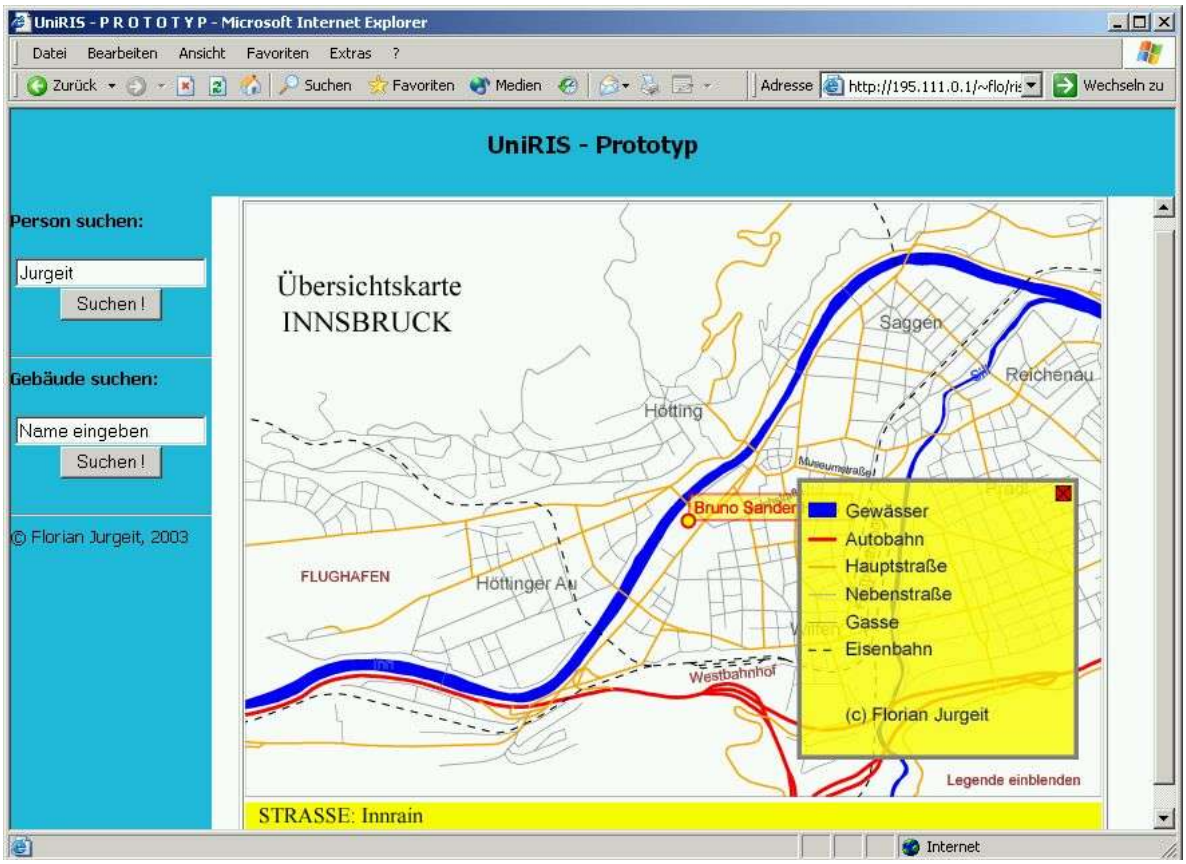


*Abbildung 45: Stadtplanansicht*

Ein weiteres Element des Stadtplans ist die einblendbare Legende, bei der es sich um eine Gruppe innerhalb des Plans handelt, die ebenfalls auf „hidden“ gesetzt ist und erst durch den Klick auf „Legende einblenden“ eingeblendet wird – dies kann auf zwei Arten realisiert werden:

- Über eine Javascript-Funktion oder
- mittels dem „set“-Element, welches die Veränderung einer nicht-numerischen Eigenschaft ermöglicht. (s. Eisenberg, 2002)

Der Vorteil des zweiten Ansatzes liegt im Verzicht auf eine clientseitige Skriptsprache.

Bei der Legende wurde des Weiteren innerhalb der Legende ein neues Koordinatensystem initialisiert (`transform="translate(300, 150)"`). Dies hat den Vorteil, dass alle Elemente innerhalb der Gruppe „legend“ sich auf das neue System beziehen und bei einer Änderung der Lage der Gesamtlegende nur die Werte in „translate“ geändert werden müssen.

Die vom Skript „show_geb_stadtplan“ generierte HTML-Datei enthält neben dem eingebetteten Stadtplan noch zwei wichtige Javascript-Funktionen.

Die Funktion „hilite_geb“ dient der Hervorhebung des gesuchten Gebäudes; dabei wird das Gebäude-Kürzel als Übergabewert benötigt, um die lagetreue Punktsignatur des Gebäudes und den Gebäudenamen auf „visible“ zu setzen. Die selbe JS-Funktion ermittelt auch die Bounding-Box der Si-

gnaturengruppe und setzt die Lage und Ausdehnung des Rechtecks mit der ID „textbox“ so fest, dass der Gebäudename im Rechteck liegt.


*Abbildung 46: Textbox mit Gebäudename*

Die zweite Javascript-Funktion („getNr“ ) erfasst beim Überfahren einer Straße die ID und versucht einen der ID im Array zugewiesenen Straßennamen als Ausgabetext in das zweite einbettete SVG-Objekt („legende“) zu schreiben.

### 4.5.2 Das CMS (Content Management System)

Neben der Endanwenderapplikation wird auch ein Interface zur Wartung des Datenbestandes benötigt, das einfach zu bedienen ist und die Datenkonsistenz erhält.

Prinzipiell kommen dazu mehrere Ansätze in Frage, die sich wesentlich unterscheiden. Eine mögliche Lösung wäre die Verwendung eines Desktop-Datenbanksystems als Interface für die Serverdatenbank – die Kommunikation kann über ODBC oder JDBC erfolgen. Beispielsweise könnte mit MS-Access ein Interface erstellt werden, jedoch hat dieser Ansatz mehrere Nachteile, da auf jedem Rechner der für die Wartung verwendet wird MS-Access installiert sein muss (▸ Kosten !) und auch die ODBC-Treiber richtig installiert und konfiguriert sein müssen.

Wünschenswert ist für solche Zwecke somit eine Lösung, die auf nahezu jedem Client ohne zusätzliche Software funktioniert. Ein derartiges System kann mit PHP als serverseitiger Scriptsprache und HTML als Gestaltungsmittel für das Interface realisiert werden. Clientseitig wird lediglich ein Browser benötigt.

Der Prototyp wurde mit folgenden Modulen zur Verwaltung der Datenbasis ausgestattet:

- Personendaten eingeben
- Gebäude und Raumdaten eingeben
- Nutzerzuordnung
- Raumzuweisung und Raum löschen

Die Absicherung der CM-Funktionalitäten vor fremden Zugang kann über einen Passwortschutz durch eine „.htaccess“-Datei erfolgen (s. Kapitel 4.4.2).

#### *4.5.2.1 Personendaten*

Die Eingabe neuer Nutzer erfolgt über ein einfaches Formular, dessen Eingabewerte an Skript „eingabe.php“ übermittelt werden. Dieses einfache Skript übermittelt die Werte an die PostgreSQL-Datenbank in Form des SQL-Befehls zur Eingabe von Daten (s. Kapitel 3.6.4):

```
insert into personen (vname, nname, titel, tel, mail, url) values
('$vname','$nname','$titel','$tel','$mail','$url')
```

Die Eingabe von Personendaten in das System kann im Realbetrieb evtl. durch den Zugriff auf die bestehende Personen-Datenbank der Univ. Innsbruck entfallen, da dieser Datenbestand alle wesentlichen Informationen enthält (Name, email, ...). Dazu muss lediglich das Skript zur Raum-Nutzer Zuordnung (s. Kapitel 4.5.2.3) bezüglich der Personenauswahl abgeändert werden.

#### *4.5.2.2 Gebäude und Raumdaten*

Ausgangspunkt hierfür ist die Definition eines neuen Raumes, die neben den Angaben zum Raum (Raumnummer, Etage, Typ/Verwendung) auch die Gebäude-ID als Fremschlüssel erfordert. Aus diesem Grund ist in das Formular zur Raumdefinition („raum_definier.php“) ein Skript integriert, das eine Auswahl der sich in der Tabelle „gebaeude“ befindlichen Gebäude ermöglicht – sollte das Gebäude noch nicht existieren kann ein neues Gebäude definiert werden; nach der Neudefinition eines Gebäudes wird der Nutzer wieder zurück an die Raumeingabe geleitet.

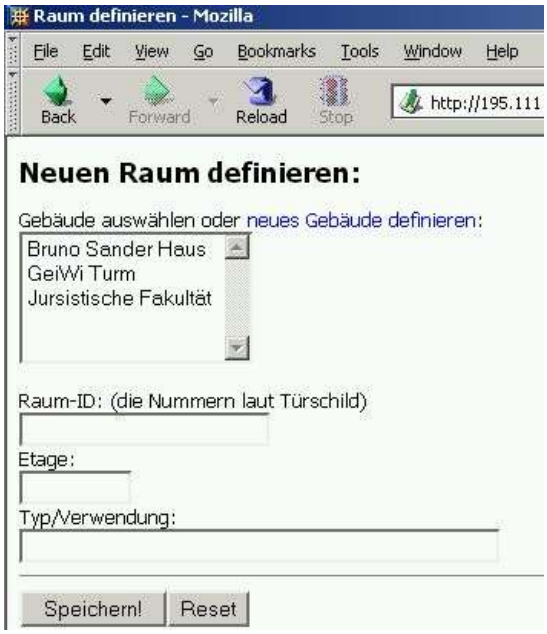


*Abbildung 47: CMS - Raumdefinition*

Die Daten zum Raum werden an das Skript „raum_def“ übergeben, welches neben der Übermittlung der Werte an die Datenbank eine Überprüfung auf Vorhandensein der Plandatei, die sich aus Gebäudekürzel und Etage zusammensetzt, vornimmt. Sollte auf dem Server im Verzeichnis „/plan“ die entsprechende Datei nicht vorhanden sein wird eine Möglichkeit zum Upload der entsprenden SVG-Datei angeboten (s. „upload.html“ und „upload.php“).

Da die Upload-Funktion Schreibrechte auf dem Server benötigt wurde aus Sicherheitsgründen im Prototyp die Funktion deaktiviert.

#### *4.5.2.3 Nutzerzuordnung*

Betrachtet man das Datenmodell (s. Kapitel 4.2) scheint die Zuordnung einfach: In die Relation „Personen-Raum“ muss lediglich die ID der Person und die ID des Raumes eingetragen werden, sodass eine Zuordnung erfolgt (beispielsweise „Person **123** nutzt Raum **67001**“).

Das Skript „zuweis.php“ erzeugt eine HTML-Seite, die aus zwei Teilen besteht:

- Einem Formular zur Einschränkung der Auswahl (Personen, Räume) und
- einem Formular zur Auswahl der Person und des zuzuordnenden Raumes.

Ersteres Formular übermittelt die Werte an das Skript „zuweis.php“ und stellt somit einen sogenannten Selbtverweis dar (s. Wigard, 2001).

Das zweite Formular bietet die Funktionalität an sich und stellt die Personendaten und Raumdaten in Auswahlmenüs dar. Dazu werden die jeweiligen Datenbestände aus der Datenbank ausgelesen und als Werte in die Auswahlmenüs eingetragen.

Die SQL-Anfrage hängt von der Definition der Variablen zur Einschränkung der Auwahl ab ($ss-

tring, $geb_string) – haben diese einen Wert zugewiesen wird die Anfrage an die Datenbank unter Berücksichtigung der Einschränkungsbedingungen durchgeführt:

```
If (!isset($sstring))
 {
$query = "select p_id, vname, nname from personen;";
 }
else
 {
$query = "select p_id, vname, nname from personen where nname like
'%$sstring%';";
  };
```

Nach der Auswahl einer Person und eines Raumes werden die ID's der jeweiligen Datensätze an das Skript „zuweis_action.php“ übermittelt, das die Personen-ID („p_id“) und die Raum-ID („r_id“) in die Relationentabelle „personen_raum“ einträgt.


*Abbildung 48: CMS - Raumzuweisung*

### *4.5.2.4 Nutzerzuordnung löschen*

Sollte eine Person nicht mehr Nutzer eines Raumes sein muss die Zuordnung aufgehoben werden. Das Skript „zuweis_loesch.php“ listet alle Nutzerzuordnungen auf und bietet die Möglichkeit einzelne Zuordnungen zu löschen.

Ähnlich der Nutzerzuordnung besteht ebenfalls die Möglichkeit die Datensätze einzuschränken.
Als weiteres Feature ist die Möglichkeit integriert nicht nur die Zuordnung, sondern auch den Raum zu löschen; da es sich bei der Beziehung zwischen Räumen und Personen um eine m:n-Objektbeziehung handelt muss überprüft werden ob auch andere Nutzer dem Raum zugeordnet sind – ist dies nicht der Fall wird die Möglichkeit der Löschung der Zuordnung und des Raumes angeboten. Auf eine Löschung der Person wird hinsichtlich einer möglichen Anbindung an eine separate Personen-Datenbank verzichtet.
An das die Löschung ausführende Skript („zuweis_loesch_action.php“) wird die Raum-ID und die Personen-ID der Zuordnung übermittelt, im Fall der Löschung des Raumes wird der Parameter „loesch_raum=1“ gesetzt:

```
<a href='zuweis_loesch_action.php?p_id=$o->p_id&r_id=$o->r_id&loesch_raum=1'>Zuweisung
und Raum löschen</a>
```

Das Skript „zuweis_loesch_action.php“ löscht auf jeden Fall die Zuweisung (s. SQL-Query „$anfrage“ im Skript), ist „$loesch_raum“ gesetzt wird der Raum ebenfalls gelöscht (s. SQL-Query „$anfrage_raum“).

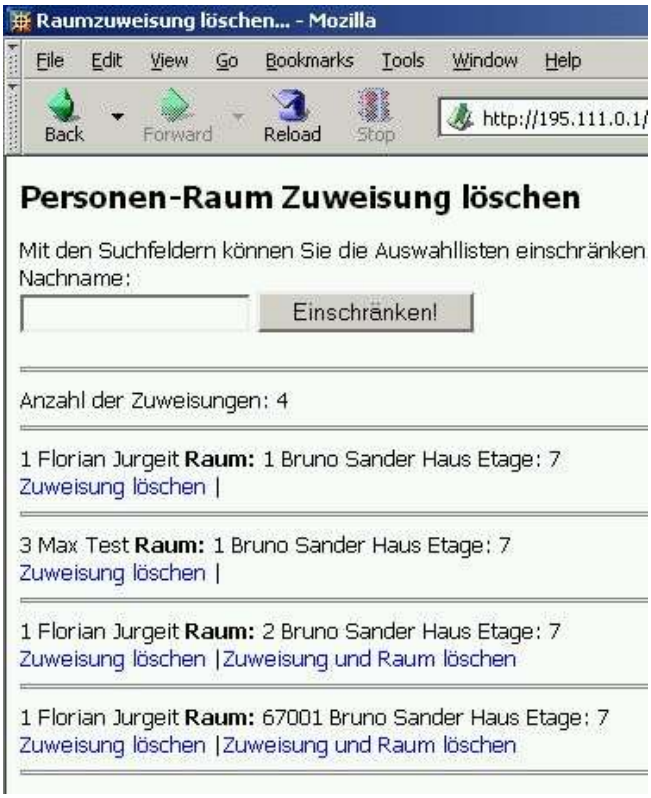


*Abbildung 49: CMS - Zuweisung löschen*

# 5 Zusammenfassung und Ausblick

Die Beschäftigung mit der gesamten Thematik macht deutlich welch umfangreiches und breit gefächertes Know-How heutzutage notwendig ist, um unter den Vorgaben verhältnismäßig einfache Lösungen zu entwickeln.

Die Datenbankkomponente im Prototypen konnte gut realisiert werden und ist ausbaufähig, wobei sich die verbreiteten OpenSource-Serverdatenbanken MySQL und PostgreSQL sehr gut eignen und auf jeden Fall einer Desktop-DB Lösung vorzuziehen sind. Die Verwendung kommerzieller DBS wie Oracle wird häufig bevorzugt, jedoch scheint die Notwendigkeit der Investition in Anbetracht der kostenlosen DBS nicht gegeben.

SVG als Mittel zur Visualisierung der Pläne hat einen geteilten Eindruck hinterlassen:

Einerseits bietet SVG alle Features die dazu benötigt werden, andererseits fehlt gegenwärtig noch die Unterstützung von SVG durch die gängigen CAD-Programme. Die Konvertierung und notwendige Erstellung einer Polygon-Topologie für die Räume hat sich aufgrund der Qualität der Pläne als problematisch erwiesen. Bezüglich der Geometrie muss durchaus eine Neuerstellung von für diesen Zweck optimierten Plänen angedacht werden, sodass zusätzliche Kosten entstehen, die bei allen GebIS einen wesentlichen Faktor darstellen.

Die Realisierung in Form einer Web-Applikation, die clientseitig keine zusätzliche zu installierende Software benötigt scheint sinnvoll zu sein – auch die Möglichkeit der Trennung von interner Applikation (Intranet) und öffentlich verfügbaren Diensten ist gegeben.

Inwieweit die Web-basierte Technik dazu geeignet ist, proprietäre, vollwertige GebIS in Form von Expertensystemen abzulösen muss als fraglich bezeichnet werden – als offene Informationsplattform angedachte Systeme müssen jedoch auf Web-Techniken aufbauen.

SVG wird als W3C-Standard zukünftig vermutlich auf immer breiterer Basis von den Herstellern unterstützt werden und die bereits vorhandenen proprietären Formate gemeinsam mit anderen XML-basierten Austauschformaten ablösen.

# 6 Verwendete Abkürzungen – Glossar

| *Abkürzung* | *Bedeutung* |
|---|---|
| ANSI | **A**merican **N**ational **S**tandard **I**nstitute |
| API | **A**pplication **P**rogramming **I**nterface |
| BSD | **B**erkley **S**oftware **D**istribution |
| CAD | **C**omputer **A**ided **D**esign |
| CGI | **C**ommon **G**ateway **I**nterface |
| CSS | **C**ascading **S**tyle **S**heets |
| DBCI | **D**ata**b**ase **C**ommunication **I**nterface |
| DBMS | **D**ata**b**ase **M**anagement **S**ystem |
| DBS | **D**ata**b**ase **S**ystem |
| DCL | **D**atabase **C**ontrol **L**anguage |
| DDL | **D**ata **D**efinition **L**anguage |
| DML | **D**ata **M**anipulation **L**anguage |
| DOM | **D**ocument **O**bject **M**odel |
| DTD | **D**ocument **T**ype **D**efinition |
| ERD | **E**ntity **R**elationship **D**iagram |
| ERM | **E**ntity **R**elationship **M**odel |
| GebIS | **Geb**äude **I**nformations **S**ystem |
| GIS | **G**eographisches **I**nformations **S**ystem |
| GPL | **GNU P**ublic **L**icense |
| HTML | **H**yper**t**ext **M**arkup **L**anguage |
| HTTP | **H**yper**t**ext **T**ransfer **P**rotocol |
| IP | **I**nternet **P**rotocol |
| ISO | **I**nternational **S**tandard **O**rganisation |
| JS | **J**ava**s**cript |
| LAMP | **L**inux **A**pache **M**ySQL **P**HP |
| LAPP | **L**inux **A**pache **P**ostgreSQL **P**HP |
| MIME | **M**ultipurpose **I**nternet **M**ail **E**xtension |
| ODBC | **O**pen **D**ata**b**ase **C**onnectivity |
| RDBMS | **R**elational **D**ata**b**ase **M**anagement **S**ystem |
| RDM | **R**elational **D**atabase **M**odel |
| SGML | **S**tandard **G**eneralized **M**arkup **L**anguage |
| SMIL | **S**ynchronized **M**ultimedia **I**ntegration **L**anguage |

| ***Abkürzung*** | ***Bedeutung*** |
|---|---|
| SQL | **S**tructured **Q**uery **L**anguage |
| SVG | **S**caleable **V**ector **G**raphics |
| TCP | **T**ransmission **C**ontrol **P**rotocol |
| WAMP | **W**indows **A**pache **M**ySQL **P**HP |
| XML | E**x**tensible **M**arkup **L**anguage |
| XSL | E**x**tensible **S**tylesheet **L**anguage |

# 7 Literatur


- Bakken, S. et al.: PHP Documentation. - www.php.net/manual/en/.
- Behme, H.: Schön viel Grafik – SVG: Grundlagen skalierbarer Webgrafik. - in: iX 12/2002, Seite 52ff.
- Behme, H.: SVG: Scripting und Datenbankauswertung. - in: iX 2/2003, Seite 130ff.
- Berger, A.: SVG – Gestaltung einer interaktiven Karte. - Studienarbeit TU Dresden, 2002.
- Born, G.: Jetzt lerne ich XML. - Markt+Technik, München 2001.
- Breutmann, B.: Objektorientierte Datenbanken und neuere Entwicklungen. - IT-Kompaktkurs, 2001.http://www.bw.fh-deggendorf.de/kurse/db/skripten/skript13.pdf
- Cartwright, W. et al (Hrsg.): Multimedia Cartography. - Springer Verlag, Heidelberg 1999.
- Däßler, R.: MySQL. - bhv, Bonn 2001.
- Dehnhart, W.: Scriptsprachen für dynamische Webauftritte. - Hanser, München 2001.
- Drake, J. D. und Worsley J. C.: Practical PostgreSQL. - o'reilly, Sebastopol (CA) 2002.
- Eisenberg, J.D.: SVG – Essentials. O'reilly, Sebastopol (CA) 2002.
- Enseleit, D.: PHP 3/4 Befehlsreferenz. - Franzis Verlag, 2001.
- Fibinger, I.: Scalable Vector Graphics (SVG) - Untersuchung des XML-Standards für zweidimensionale Vektorgraphiken und Erstellung eines SVG-Tutorials. - Diplomarbeit an der FH Karlsruhe, Karlsruhe 2001.
- Fürpaß, Ch.: Map-Server als Hilfsmittel zur Datenvisualisierung im Internet. - Diplomarbeit an der Univ. Wien, Wien 2001.
- Gammack, R.G.: New Map Design Challenges – Interactive Map Products for the WWW. - in: Cartwright, W. et al (Hrsg.): Multimedia Cartography. - Springer Verlag, Heidelberg 1999.
- Gartner, G.: Multimedia GIS and the Web. - in: Cartwright et al. (Hrsg.): Multimedia Cartography. Springer Verlag, Heidelberg 1999.
- Geschwinde, E. und Schönig, H.-J.: Datenbank-Anwendungen mit PostgreSQL. - Markt und Technik, München 2002.
- Grolig, B.: Scalable Vector Graphics – Über die Darstellung von 2-D-Vektoren mit einer XML-basierten Bechreibungssprache. - in: Herrmann Ch. Und Asche, H. (Hrsg.): Web.Mapping . - Herbert Wichmann Verlag, Heidelberg 2001.
- Hake, G. und Grünreich, D.: Kartographie – Walter de Gruyter, Berlin 1994.
- Herrmann, Ch. und Asche, H. (Hrsg.): Web.Mapping 1 – Raumbezogene Information und Kommunikation im Internet. - Herbert Wichmann Verlag, Heidelberg 2001.
- Heuer, A.: Objektorientierte Datenbanken – Konzepte, Modelle, Systeme. - Addison-Wesley,

Bonn 1992.

- Kofler, M.: Linux – Installation, Konfiguration, Anwendung. - Addison-Wesley, München 2002.
- Krüger, J. und Märtin, Ch.: SVG und VML – XML-Alternativen für dynamische Webgrafiken. - in: iX 11/2000, S. 148ff.
- Kukofka, P.: Scale a Vector – SVG. - www.scale-a-vector.de, 2003.
- Lindhorst, A.: Cascading Style Sheets. - bhv, Bonn 2001.
- Microimages (Hrsg.): Rectifying Images. - TNTmips-Manual V. 6.4, Lincoln 2000.
- Münz, S.: Self – HTML. - www.selfhtml.de, 2001.
- MySQL AB (Hrsg.): MySQL Reference Manual – www.mysql.com/doc.
- Neumann, A.: Comparing .SWF (Shockwave Flash) and .SVG (Scalable Vector Graphics) file format specifications. - www.carto.net/papers/svg/comparison_flash_svg/.
- Neumann A. et al: Das Web auf neuen Pfaden – SVG und SMIL. - in: c't 20/2002, S. 218ff.
- Neumann, A. und Winter, A.: Time for SVG – Towards High Quality Web-Maps. -http://www.-carto.net/papers/svg/articles/paper_icc_congress_china_2001.pdf, 2001-1.
- Neumann, A. und Winter, A. (2001-2): SVG – Ein zukünftiger Eckstein der WWW-Infrastruktur. - http://www.carto.net/papers/svg/articles/paper_ugra_zurich_2001.pdf, 2001-2.
- Neumann, A.: Example for Serverside SVG generation with PHP. - www.carto.net/papers/svg/serverside_svg_php_e.html.
- Nutto, M.: Evaluierung der Anbindung von Oracle V 8.04 an ein prototypisches Gebäudeinformationssystem. - Diplomarbeit an der Univ. Karlsruhe, Karlsruhe 1999.
- OGC (Hrsg.): OpenGIS Simple Features Specification For SQL. - www.opengis.org, 1999.
- OGC (Hrsg.): OpenGIS Geography Markup Language (GML) Implementation Specification. - http://www.opengis.net/gml/02-009/GML2-11.pdf, 2002.
- PostgreSQL Documentation Group (Hrsg.): PostgreSQL 7.2 Tutorial. - http://www.postgres-ql.org/docs/.
- PostgreSQL Documentation Group (Hrsg): PostgreSQL 7.2 Reference Manual. - www.post-gres.org/docs.
- Pucher, A.: Datenbankgestützte kartographische Visualisierung im Internet mittels Map-Server Systemen. - Diplomarbeit an der Univ. Wien, Wien 2001.
- Rogers, N.: shp2svg – Converting Shape-Files to SVG. - http://www.carto.net/projects/shp2svg/
- Roßner, Th.: Perl. - bhv, Bonn 2002.
- Schenk, A.: Flash – neue Wege zur kartographischen Visualisierung. - in: Herrmann Ch. Und Asche, H. (Hrsg.): Web.Mapping . - Herbert Wichmann Verlag, Heidelberg 2001.

- Scheu, M.: CAFM – Computergestütztes Facility Management. - Arbeit an der TU-Berlin, Fachbereich 09, Berlin 1999.
- Schmidt-Haunschild, P.: Kooperatives Gebäude. - Diplomarbeit an der Univ. Karlsruhe, Karlsruhe 1997. http://ifib41.ifib.uni-karlsruhe.de/petra/Diplomarbeit
- Schulz, H. und Siering, P.: Datendiener – Freie Datenbankserver im Vergleich. - in: c't Ausgabe 5/2003, Seite 142ff.
- Schulz, M.: WebGIS-Technologien und OpenSource Lösungen. - Workshop „Internet-GIS“, Pforzheim, Oktober 2002.
- Schulz, H.: Datenschule – mit SQL zur eigenen Datenbank. - in: c't Ausgabe 9/2003, Seite 234ff.
- Spona, H.: SVG – Webgrafiken mit XML. - bhv, Bonn 2001.
- Stephens et al: SQL in 21 Tagen. - Markt und Technik, München 1998.
- Strobl, J.: Online-GIS – das WWW als GIS-Plattform. - in: Herrmann Ch. Und Asche, H. (Hrsg.): Web.Mapping . - Herbert Wichmann Verlag, Heidelberg 2001.
- W3C (Hrsg.): Cascading Style Sheets. - http://www.w3c.org/Style/CSS/.
- W3C (Hrsg.): HTML Homepage. - http://www.w3c.org/MarkUp/.
- W3C (Hrsg.): SMIL Homepage. - http://www.w3c.org/AudioVideo/.
- W3C (Hrsg.): Scalable Vector Graphics (SVG) 1.0 Specification. - www.w3c.org/Graphics/SVG/.
- W3C (Hrsg.): SVG Software Implementations. - http://www.w3.org/Graphics/SVG/SVG-Implementations.
- Wagner, J.-O.: Gute Karten mit GNU/Linux. - Proceeding Linux Tag 2001, www.freegis.org.
- Wenz, C.: Javascript – browserübergreifende Lösungen. - Galileo Press, Bonn 2001.
- Wigard, S.: PHP 4. - bhv, Bonn 2001.
- Winter, A.: Internetkartographie mit SVG – Prototyp für einen thematischen Atlas. - Diplomarbeit an der Univ. Wien, Wien 2000.
- Waldik, D.: Stand und Tendenzen zur Visualisierung von Geoinformation im WWW. - in: Herrmann Ch. Und Asche, H. (Hrsg.): Web.Mapping . - Herbert Wichmann Verlag, Heidelberg 2001.
- Zippelt, K.: Aspekte der Datenmodellierung in Gebäudeinformatiossystemen. - http://www.gi-k.uni-karlsruhe.de/~zippelt/pub/gebis.html, 1999.

# 8 Anhang – Skripte

Im folgenden sind **nicht alle Skripte abgedruckt**, da sich diese tlw. ähnlich sind, des Weiteren enthalten sie aus Platzgründen **nur den funktionalen Kern**.

Die HTML-Seiten sind nicht abgedruckt !

## *8.1 Personensuche („search.php“)*

```
<?php
// Suche nach Personen in DB - Weiterleitung auf Raum durch Hyperlink
// Florian Jurgeit, 2002

// DB_Settings
include ("db_userdata.php");
// Ende DB-Settings

$connect = pg_connect($connect_data);

$query = "select * from personen, gebaeude, raum, personen_raum
where
personen.nname like '$name' and
((personen.p_id=personen_raum.p_id) and
(personen_raum.r_id = raum.r_id)) and (raum.geb_id =
gebaeude.geb_id);";

$result = pg_exec($query);

$n = pg_numrows($result);

If ($n == 0)
 {
 print "<b>Ihre Suche brachte keinen Treffer !!!</b>";
 }
else
 {
  print "<h2>Ergebnisse der Suche:</h2>";
  print "Ihre Suche nach: <b>$name</b> - Treffer: <b>$n</b><hr>";


  // Trefferausgabe und Weiterleitung auf Plan mittels Hyperlink
  print "<ul>";
  for ($i=0; $i<$n; $i++)
   {
```

```
   $o = pg_fetch_object($result);
   $typ = htmlentities($o->typ);
   print "<li>$o->titel
<b>$o->nname $o->vname</b><br>
<a href='show_geb_stadtplan?geb_kuerzel=$o->geb_kuerzel'>$o->geb_name</a>, Etage: $o-
>etage, Raum: <a href='showraum?r_id=$o->r_id&geb_kuerzel=$o->geb_kuerzel&etage=$o-
>etage'>$o->r_id</a><br></li> ";
   };
  print "</ul><hr>Hinweis: Clicken Sie auf die Raum-Nummer um den
Etagenplan zu sehen - ein Click auf den Gebäudenamen zeigt die Lage im Stadtplan.";

  pg_close($connect);
 } //Ende else
?>
```

## 8.2 Raumvisualisierung („showraum.php“)

```
<html>
<head>
<title>UniRIS - Planansicht</title>
<link rel="stylesheet" href="css.css" content="text/css">
<?php

// Es folgt der Normalfall - Person mit 1 Raum
// Es folgen JS-Codes
// JS um Treffer zu highlighten, dann onclick-Script, Größe anpassen
print "
<script type='text/javascript'>

var svgdoc, alreadyclicked;

function opt_size()
 {
 // Diese Funktion passt die Plangröße dem embed-Fenster an !
 // Die w und h aus dem embed-Tag !!!
 var w = 600;
 var h = 350;

 var svgdoc = document.$geb_kuerzel$etage.getSVGDocument();
 var svgobj = svgdoc.getElementById('map');
 var groupplan = svgdoc.getElementById('plan');

 // Ermitteln der Bounding-Box
```

```
 var planBoundBox = groupplan.getBBox();
 var bbox_X = planBoundBox.getX();
 var bbox_Y = planBoundBox.getY();
 var bbox_W = planBoundBox.getWidth();
 var bbox_H = planBoundBox.getHeight();

// Setzen der Viewbox-Werte auf die Bounding-Box Werte !
svgobj.setAttribute('viewBox', bbox_X+' '+bbox_Y+' '+bbox_W+' '+bbox_H);

svgobj.setAttribute('width', w);
svgobj.setAttribute('height', h);
 }

function show()
   {
      var doc = document.$geb_kuerzel$etage.getSVGDocument();
      var objekt2 = doc.getElementById('$r_id');
      var aktclr2 = objekt2.getAttribute('fill');

      objekt2.setAttribute('style', 'fill:yellow');
   }

function showinfo(evt)
   {
   if (alreadyclicked)
    {
    alreadyclicked.setAttribute('fill','');
    }

   var object = evt.getTarget();
   var r_id = object.getAttribute('id');
   var aktcolor = object.getAttribute('fill');
   object.setAttribute('fill', 'red');
   alreadyclicked = object;

   var popup = 'info.php?r_id='+r_id;
   alert ('Raum-ID:'+popup);
   infopopup = window.open(popup, 'INFO' , 'width=300,height=200,scrollbars');
   infopopup.focus();
   }
</script>
 ";
```

```
?>
<!--ENDE PHP-Code-->
<meta name='author' content='c716307'>
<meta name='generator' content='Ulli Meybohms HTML EDITOR'>
</head>
<body text='#000000' bgcolor='#F0F0F0' link='#FF0000' alink='#FF0000'
vlink='#FF0000' onload='show(), opt_size()'>

<b>Hinweis:</b> Der gesuchte Raum ist gelb markiert !<br>
Ein Click auf einen Raum zeigt weitere Informationen an !

<!--Es folgt PHP-Code-->
<?php
// Es folgt die Einbettung des SVG-Files in das HTML-Grundgerüst
// ... und zuvor die Überprüfung auf Vorhandensein der Plandatei !
$plandatei = "plan/$geb_kuerzel$etage.svg";
If (file_exists($plandatei))
 {
  print "
  <embed name=$geb_kuerzel$etage width='600' height='350'
src='plan/$geb_kuerzel$etage.svg' type='image/svg+xml'
pluginspage='http://www.adobe.com/svg/viewer/install/'>
  ";
// Ende der Einbettung
  }
else
 {
 print "<b>Fehler: Plandatei nicht vorhanden !</b>";
 };
?>
<!--ENDE PHP-Code-->
<hr noshade size="1">
Pläne: &copy Universität Innsbruck
</body>
</html>
```

## *8.3 Gebäudesuche („search_geb.php“)*

```
<?php
// Räume und Etagenauswahl
// Florian Jurgeit, 2002
// Achtung: Veränderte SQL-Syntax zur MySQL-Version
```

```
include("db_userdata.php");

$connect = pg_connect($connect_data);

$query = "select distinct on (raum.geb_id) * from raum, gebaeude where
(raum.geb_id = gebaeude.geb_id) and (gebaeude.geb_name like '%$geb_name%');";

$result = pg_exec($query);

$n = pg_numrows($result);

print "<h2>Suchergebnis der Gebäudesuche</h2>
Treffer:<b>$n</b><hr>";

// Trefferausgabe und Weiterleitung an Etagenabfrage und Stadtplan
print "<ul>";
for ($i=0; $i<$n; $i++)
  {
  $o = pg_fetch_object($result);
  print "<li><a href='getetagen?geb_id=$o->geb_id'>$o->geb_name</a>
[<a href='show_geb_stadtplan?geb_kuerzel=$o->geb_kuerzel'>Im Stadtplan
anzeigen</a>]<br>
  $o->beschreibung<br></li>
  ";
  };
print "</ul>";

pg_close($connect);
?>
```

## *8.4 Rauminformations-PopUp („info.php“)*

```
<?php
// Suche nach Personen in DB - ausgehend vom onclick aus den Raum
// Florian Jurgeit, 2002

include("db_userdata.php");

$connect = pg_connect($connect_data);

$query = "select * from personen, raum, gebaeude, personen_raum where raum.r_id
= $r_id and ((personen.p_id=personen_raum.p_id) and (personen_raum.r_id =
raum.r_id)) and (raum.geb_id=gebaeude.geb_id);";
```

```
$result = pg_exec($query);

$n = pg_numrows($result);

If ($n == 0)
 { print "<b>Dem Raum sind derzeit keinen Personen zugeordnet !</b>";}
else
 {
print "Im Raum <b>$r_id</b>  befinden sich <b>$n</b> Personen.<hr>";
// Trefferausgabe
for ($i=0; $i<$n; $i++)
  {
  $o = pg_fetch_object($result);
  print "<b>$o->titel $o->vname $o->nname</b> (Tel.: $o->tel)<br>
  email: <a href='mailto:$o->mail'>$o->mail</a><br>
  URL: <a href='http://$o->url' target='_blank'>$o->url</a><br>
  $o->geb_name, Etage:$o->etage, Raum: $o->r_id<hr>";
  };

}; //Ende else
pg_close($connect);
?>
```

## 8.5 Stadtplan („show_geb_stadtplan.php“)

```
<html>
<head>
<title>Übersichtsplan Innsbruck</title>
<SCRIPT type='text/javascript'>

function showlegend()
 {
document.karte.getSVGDocument().getElementById('legend').setAttribute
('visibility','visible');
document.karte.getSVGDocument().getElementById('legend_in').setAttribute
('visibility','hidden');
 }

function legendclose()
 {
document.karte.getSVGDocument().getElementById('legend').setAttribute('visibility','hid-
den');
document.karte.getSVGDocument().getElementById('legend_in').setAttribute
('visibility','visible');
```

```
 }

<?php
print "
function hilite_geb()
 {
 var svgdoc = document.karte.getSVGDocument();
 if (svgdoc.getElementById('$geb_kuerzel'))
 {
 svgdoc.getElementById('$geb_kuerzel').setAttribute('visibility','visible')
 var bbox = svgdoc.getElementById('$geb_kuerzel').getBBox();
 var textboxerl = svgdoc.getElementById('textbox');
 textboxerl.setAttribute('x',bbox.getX()+4);
 textboxerl.setAttribute('y',bbox.getY()-4);
 textboxerl.setAttribute('width',bbox.getWidth()+3);
 textboxerl.setAttribute('height',bbox.getHeight());
 textboxerl.setAttribute('visibility','visible');
 } //Klammer von IF
 else
 {
 alert ('Gebäude im Plan NICHT vorhanden !!!');
 }; //Klammer von ELSE
 }
";
?>

function getNr(evt)
      {
       var strname = new Array()
         strname[186] = "Innrain"
         strname[182] = "Innrain"

       var obj = evt.getTarget();
       var svgdoc = obj.getOwnerDocument();
       var nr = obj.getAttribute('nr');
       var stra_typ = obj.getAttribute('class');

              // Ansteuern der Legende-Straßeninfo
              var strinfo = document.legende.getSVGDOcument().getElementById
("strInfo");
              if (strname[nr])
               {
```

```
                strinfo.getFirstChild().setData('STRASSE: '+strname[nr]);
                 }
                else
    {
    strinfo.getFirstChild().setData('Straßenname leider NICHT verfügbar !('+nr+')');
    };
 }
</script>
</head>
<body onload="hilite_geb()">
<table border="1" width="640">
<tr valign="top"><td>
<embed src="plan/ibk.svg" width="640" height="444" name="karte" type="image/svg+xml">

<!--Es folgt die Legende(unten) mit Info zu Hauptstraßen als SVG-Objekt-->
  <embed src="plan/legende.svg" width="640" height="20" name="legende"
type="image/svg+xml">
  </td>
</tr>
</table>
</form>
</body>
</html>
```

## 8.6 Raumzuweisung-Formular („zuweis.php“)

```
<body text="#000000" bgcolor="#FFFFFF" link="#FF0000" alink="#FF0000"
vlink="#FF0000">
<h1>Personen-Raum Zuweisung</h1>
Mit den Suchfeldern können Sie die Auswahllisten einschränken !
  <form name="search" action="zuweis.php" method="post">
  Nachname: <br>
  <input type="Text" name="sstring" value="" size="20" maxlength="30">
  <br> Gebäude: <br>
  <input type="Text" name="geb_string" value="" size="20" maxlength="30">
  <input type="Submit" value="Einschränken!">
  </form>
  <hr>
<?php
// Personenauswahl aus DB
// Florian Jurgeit, 2002
include ("db_userdata.php");
$connect = pg_connect($connect_data);
```

```
If (!isset($sstring))
 {
$query = "select p_id, vname, nname from personen;";
 }
else
 {
$query = "select p_id, vname, nname from personen where nname like
'%$sstring%';";
  };

$result = pg_exec($query);
$n = pg_numrows($result);

// Trefferausgabe in Drop-down Menü
print "<form name='zuweisung' action='zuweis_action.php' method='post'>";
print "Name:<br><select name='p_id' size='10'>";

for ($i=0; $i<$n; $i++)
  {
  $o = pg_fetch_object($result);
  print "<option value='$o->p_id'>$o->vname $o->nname</option>";
  };

print "</select>";

If (!isset($geb_string))
 {
$anfrage = "select * from raum, gebaeude where raum.geb_id=gebaeude.geb_id;";
 }
else
 {
$anfrage = "select * from raum, gebaeude where ((raum.geb_id=gebaeude.geb_id)
and
(geb_name like '%$geb_string%'));";
 };

$ergebnis = pg_exec($anfrage);
$nr = pg_numrows($ergebnis);

// Trefferausgabe in Drop-down Menü
print "<select name='r_id' size='10'>";
```

```
for ($i=0; $i<$nr; $i++)
  {
  $o = pg_fetch_object($ergebnis);
  print "<option value='$o->r_id'>$o->r_id $o->geb_name,
Etage:$o->etage</option>";
  };

print "</select>";
pg_close($connect);

print "<hr><input type='Submit' name='' value='Zuweisen!'><input
type='reset'>";
print "</form>";
?>
</body>
```

## *8.7 Zuweisung durchführen („zuweis_action.php“)*

```
<?php
include ("db_userdata.php");
$link = pg_connect($connect_data);
 $anfrage="insert into personen_raum values ('$p_id','$r_id')";
  If ($ergebnis=pg_exec($anfrage))
        {echo "Datensatz eingefügt :-) <br>";
           include('zuweis.php');
        }
  else
    {echo "<b>Fehlermeldung: </b>";};
pg_close($link);
?>
```