Diplomarbeit

Jan Schüngel

Version 1.01
17. September 2004

Diplomarbeit

Management von Web-Mapping Anwendungen in GIS-Applikationen

am Beispiel
einer Thuban Extension zur Konfiguration des UMN MapServers

Verfasser:

Jan Schüngel

Betreuer:

Prof. Dr. Jürgen Weitkämper
(FH-Oldenburg)

Dr. Dipl. Systemwiss. Jan-Oliver Wagner
(Intevation GmbH)

Impressum

Diplomand: Jan Schüngel
Holtkampstr. 37, 32257 Bünde, Deutschland
geboren: 26. September 1980 in 32049 Herford
Telefon: +49 5223 3336
eMail: diplom@janschuengel.net
Internet: http://www.janschuengel.net

Diplomstelle: Intevation GmbH
Georgstrasse 4, 49074 Osnabrück
Telefon: +49 541 33508-30
eMail: intevation@intevation.de
Internet: http://www.intevation.de

Hochschule: Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Ofener Straße 16/19, 26121 Oldenburg
Telefon: +49 180 567807-0
Fachbereich Geoinformatik
Internet: http://www.fh-oow.de & http://www.fh-oldenburg.de

Zeitraum: Sommersemester 2004
Fertigstellung: 09. 09. 2004

Betreuer: 1. Prüfer (FH-Oldenburg):
Prof. Dr. Jürgen Weitkämper
Telefon: +49 180 567807-3192
eMail: weitkaemper@fh-oldenburg.de

2. Prüfer (Intevation GmbH):
Dr. Dipl. Systemwiss. Jan-Oliver Wagner
Telefon: +49 541 33508-30
eMail: Jan-Oliver.Wagner@intevation.de

Die Erstellung der Arbeit erfolgte ausschließlich mit Freier Software. Dieser Text wurde mit der freien Office-Suite OpenOffice (http://www.openoffice.org) verfasst, welche sich als sehr ausgereiftes und einfach zu nutzendes Werkzeug zur Erzeugung von längeren Arbeiten erwiesen hat. Die meisten Abbildungen wurden mit GIMP (http://www.gimp.org) oder per Postscript-Druck (http://www.cs.wisc.edu/~ghost/) aus der jeweiligen Applikation erstellt. Diagramme wurden mit OpenOffice-Draw, UML Diagramme mit Umbrello (http://uml.sourceforge.net) erzeugt.

Erhältlich ist diese Diplomarbeit unter http://diplom.janschuengel.net in verschiedenen Formaten.

Vorwort

Jedes Denken wird dadurch gefördert,
dass es in einem bestimmten Augenblick
sich nicht mehr mit Erdachtem abgeben darf,
sondern durch die Wirklichkeit hindurch muss.[1]

Am Ende meines Studiums musste auch ich mich mit der im Studium erworbenen Theorie in der Praxis beweisen. Dabei stellte sich mir die Frage, in welchem Bereich ich die Diplomarbeit schreiben wollte und zu welchem Thema. Das Thema sollte einerseits im Sinne des Studiums angemessen und andererseits mit meinen Interessen vereinbar sein.

Der erste Punkt war bei der Suche eines adäquaten Themas weniger problematisch. Das Themenspektrum meines Studiengangs war breit gefächert und reichte von der Vermessung über die Photogrammetrie bis hin zur Programmierung.

Der zweite Punkt schränkte die Möglichkeiten der Themenfestlegung schon stärker ein. Mein Wunsch war es gemäß meines Interessenschwerpunktes auf jeden Fall eine Programmieraufgabe mit Bezug zum Internet anzugehen. Dabei war das Thema erst einmal zweitrangig. Für mich war wichtig, dass die verwendete Software einschließlich der verwendeten Daten später auch der Öffentlichkeit zur Verfügung gestellt werden konnte.

Die Suche nach einem Thema und einer Stelle, wo ich die Diplomarbeit schreiben konnte, hat weniger Zeit erfordert als zunächst befürchtet. Bei meiner Suche stieß ich

1 Albert Einstein (1879-1955), dt.-amerik. Physiker, Nobelpreisträger (1921)

sehr schnell auf die Homepage der Intevation GmbH in Osnabrück. Die Firmenphilosophie entsprach genau dem, was ich gesucht hatte:

„Die Intevation GmbH ist ein intelligentes Consulting-Unternehmen und setzt konsequent auf Freie Software. Europaweit bietet die Intevation GmbH Kommunikationsmanagement für erfolgreiche Projekte sowie Strategieberatung zu Geschäftsmodellen mit Freier Software. Weitere Stärken sind Geographische Informations-Systeme und Benutzbarkeitsdesign. Die Intevation GmbH erweitert aktiv die Softwaregrundlagen und fördert u.a. Freegis.org und die FSF Europe.“ [2]

Sofort entschied ich mich, einmal anzufragen, ob die Möglichkeit eine Diplomarbeit bei der Intevation GmbH zu schreiben überhaupt bestand. Die Überraschung und Freude meinerseits war riesig, als man mir die Stelle quasi schon gleich am Telefon zusagte. Nach einem anschließenden erfreulich verlaufenen Vorstellungsgespräch war auch das Thema schnell gefunden, und alle weiteren Formalitäten wurden problemlos geklärt.

Ganz nach meinen Vorstellungen konnte ich mich in den Monaten meiner Diplomarbeit mit dem interessanten Entwicklungsprozess einer Software in der Praxis auseinander setzen. Dabei lernte ich viele neue Produkte und technische Kniffe kennen, aber auch, dass man einige Regeln beachten muss, wenn man an einem Projekt arbeitet, an dem mehrere Personen mitwirken. Somit bekam ich zum ersten Mal einen wirklichen Eindruck von der Realität des eingeschlagenen Berufsfeldes und der möglichen Tätigkeiten im weiteren Verlauf meines Lebens.

Das fertige Projekt, dessen Entwicklungszyklus in dieser Diplomarbeit beschrieben wird, ist bei der Intevation GmbH in der aktuellsten Version frei verfügbar.

Und nun wünsche ich allen Leserinnen und Lesern viel Spaß beim weiteren Lesen dieser Arbeit.

Bünde, August 2004
Jan Schüngel

2 Intevation GmbH: Unternehmensbeschreibung. URL: http://intevation.de/. Stand: 06.07.2004

Danksagungen

Leider lässt sich eine wahrhafte
Dankbarkeit mit Worten nicht ausdrücken.[3]

Hiermit möchte ich folgenden Personen für ihre Unterstützung und ihre Hilfe während der Diplomarbeit danken:

Zuerst bedanke ich mich bei Herrn Prof. Dr. Jürgen Weitkämper, der mich während meiner Diplomarbeit seitens der FH-Oldenburg betreute, mich positiv unterstützte und mir die Arbeit offiziell absegnete.

Des Weiteren gilt mein besonderer Dank Herrn Dr. rer. nat. Dipl. Systemwiss. Jan-Oliver Wagner von der Intevation GmbH für seine motivierende Betreuung sowie für die kompetente fachliche Unterstützung während meiner Diplomarbeit. Er ermöglichte es mir überhaupt erst diese Arbeit bei der Intevation GmbH anzufertigen.

Auch allen anderen Mitarbeitern des Intevation GmbH Teams gilt mein herzlicher Dank: vor allem Frau Dipl. Systemwiss. Silke Reimer und Herrn Dipl. Phys. Bernhard Herzog, ferner Herrn Dipl. Systemwiss. Frank Koormann, Herrn Dipl. Systemwiss. und MSc. (Geographie) Bernhard Reiter, Herrn Thomas Arendsen Hein und Frau Dipl. Philologin Ruslana Grebnyev. Sie alle sollen hier erwähnt werden, denn sie standen mir jederzeit bei Fragen und Problemen helfend zur Verfügung.

Ein besonders herzlicher Dank geht auch an meine Eltern, die mich während meiner gesamten Studienzeit finanziell und moralisch unterstützt und mir so diese Ausbildung ermöglichten haben. Dank gilt auch meinen Freunden und Studienkollegen, die mir bei vielen Fragen mit Rat und Tat zur Seite gestanden haben. Allen, die mir sonst auf irgendeine Art und Weise, direkt oder indirekt, geholfen oder mich unterstützt haben, sage ich hiermit ein Dankeschön.

3 Johann Wolfgang von Goethe (1749-1832), dt. Dichter

Zusammenfassung

Die online-basierende Kartenverarbeitung hat in den letzten Jahren immer mehr zugenommen, und dadurch ist auch die Nachfrage nach einer möglichst einfachen Erstellung und Verarbeitung solcher Kartenwerke stetig gewachsen. Zur Realisierung dieser Wünsche bieten sich die schon in fast allen Bereichen der Vermessung und Kartographie eingesetzten geographischen Informationssysteme (GIS) an. Schließlich sind sie für den Zweck der geographischen Datenverarbeitung geschaffen worden und bieten somit eine solide und mächtige Grundlage.

Entsprechend dem Ziel der Diplomarbeit wurde ein Management von Web-Mapping Anwendungen in GIS-Applikationen erstellt. Für die Umsetzung wurden das Geodatenwerkzeug „Thuban“ als GIS-Applikation und der „UMN MapServer“ als Web-Mapping Anwendung herangezogen.

Die Realisierung dieser Aufgabe einschließlich der dabei auftretenden Schwierigkeiten ist in drei Punkten dargestellt:

- Eine Analysephase, welche zum Verständnis der verwendeten Software und zum Aufdecken eventueller Schwächen und Probleme dient,
- eine Designphase für die detaillierte Planung der zu erstellenden Erweiterung, und
- eine Implementierungsphase, welche die Realisierung beschreibt.

Als Ergebnis der Diplomarbeit liegt eine GIS-Applikation vor, welche um die Fähigkeit erweitert wurde, eine Web-Mapping Anwendung zu managen.

Abstract

Online based map processing has increased within the last years and the demand for easy constructing and processing such maps has grown steadily. For the realization of these wishes the geographical information systems (GIS), already used in almost all areas of measurement and cartography, offer their services. After all, they were created for geographical data processing and therefore offer a solid and powerful basis.

Following the aim of the diploma thesis, a management of web-mapping applications was built into GIS applications. For the realization the Interactive Geographic Data Viewer "Thuban" was used as the GIS application and the "UMN MapServer" as the web mapping application.

The realization of this task including the appearing difficulties is represented by three points:

- an analysis phase, which serves the understanding of the used software and the discovering of possible weaknesses and problems,
- a design phase for the detailed planning of the extension to be developed, and
- an implementation phase, which describes the realization.

As a result of the diploma thesis a GIS application which has the ability to manage web mapping applications was created.

Inhaltsverzeichnis

Anhangsverzeichnis

1. Einleitung

Der Anfang ist
die Hälfte des Ganzen.[4]

Zum besseren Verständnis für die Leserinnen und Leser sollen zunächst einmal Thema und Zielsetzung dieser Diplomarbeit verdeutlicht werden, und anschließend die zugrunde liegende Intention und Motivation genauer erläutert werden.

1.1 Thema und Zielsetzung

Das Thema dieser Diplomarbeit lautet „Management von Web-Mapping Anwendungen in GIS-Applikationen am Beispiel einer Thuban Extension zur Konfiguration des UMN MapServers“.

Folgende Überlegungen führten zu diesem Thema:

Dem Internet kommt heute in immer mehr Bereichen eine zunehmende Bedeutung zu, so auch in der Darstellung und Nutzung von geographischen Karten. Es ist daher nicht verwunderlich, dass immer mehr Geodaten für die Präsentation im **W**orld **W**ide **W**eb (www) aufbereitet werden. Sucht man diesbezüglich nach einer Lösung im Bereich der Freien Software, so findet man im UMN MapServer das wohl prominenteste Angebot. Hierauf soll mit dieser Diplomarbeit Bezug genommen werden. Eine weitere Lösung mit Freier Software wäre z.B. das Projekt deegree[5], welches jedoch auf Grund seines geringeren Bekanntheitsgrades für diese Arbeit nicht interessant genug erschien.

4 Aristoteles (384-322), griech. Philosoph, Begründer d. abendländ. Philosophie
5 http://deegree.sourceforge.net/

Der UMN MapServer ist ein fortschrittliches Produkt zur Visualisierung von Geodaten im Netz[6], hat aber die Eigenschaft, dass die Konfiguration der Darstellung einer Karte über eine Steuerungsdatei, eine einfache Textdatei, geschehen muss. Dies ist gerade für Anfänger oder gelegentliche Benutzer eher abschreckend und zudem fehleranfällig. Genau in diesem Punkt soll diese Diplomarbeit mit dem Ziel ansetzen, die Erstellung solcher Steuerungsdateien für oben genannte Nutzer zu vereinfachen.

Zur Lösung dieser Problematik soll eine GIS-Anwendung um die Möglichkeit erweitert werden, den UMN MapServer zu administrieren und damit die geforderte einfachere Erstellung und Verarbeitung solcher Steuerungsdateien zu erreichen. Der Entschluss eine GIS-Anwendung zu wählen und keine Web-basierende Lösung zu entwickeln (wie z.B. MapLab[7]) begründet sich darin, dass die meisten digitalen Geodaten schon als Projekte für GIS-Anwendungen verfügbar sind. Außerdem ist die Handhabung von GIS-Anwendungen in der Regel einfacher, und die GIS-Funktionalität erscheint für die Erstellung digitaler Karten unverzichtbar. Zudem ist die Funktionalität einer Web-basierenden Lösung schon dadurch eingeschränkt, dass sie auf den begrenzten Möglichkeiten des Browsers aufbaut. Der Browser wurde zum Anzeigen von Informationen entwickelt und nicht für komplexe Editieraufgaben. Des Weiteren ist gerade die Erstellung der UMN MapServer Anwendung ein Problem und nicht unbedingt die Pflege. Um kleinere Korrekturen vorzunehmen ist eine Web-basierende Lösung wie MapLab oft wesentlich besser geeignet. Dadurch können die Korrekturen direkt online erfolgen.

Neben diesen generellen Vorteilen besteht der Vorzug einer GIS-Anwendung in der Visualisierung der Geodaten und in der schon durch die Software vorhandenen Funktionalität. Somit muss keine komplett neue Anwendung entwickelt werden, sondern es kann auf vorhandene Möglichkeiten der Kartenerstellung zurückgegriffen werden.

Im konkreten Fall bedeutet dies, dass eine Schnittstelle zwischen dem UMN MapServer und der GIS-Anwendung zu implementieren ist.

Als GIS-Anwendung fiel die Wahl auf das Geodaten Werkzeug Thuban[8].

6 vgl. Kap.3.4: UMN MapServer, S.23
7 http://www.maptools.org/maplab/
8 vgl. Kap.3.3: Thuban - Interactive Geographic Data Viewer, S.21

Als Alternativen wären hier z.B. die Produkte QGIS[9] und Jump[10] zu nennen. Der Grund gerade für diese GIS-Anwendung besteht zum einen darin, dass es ein hauseigenes Produkt der Intevation GmbH ist, und zum anderen, dass es genau wie der UMN MapServer eine Freie Software Lösung darstellt. Der wohl wichtigste Punkt bei der Wahl ist aber, dass für Thuban kommerzieller Support (durch die Intevation GmbH) angeboten wird.

Zudem bietet Thuban die Möglichkeit die Funktionalität durch Erweiterungen, so genannten Extensions, auszubauen. Folglich ist kein direkter Eingriff in das Programm nötig und die Erweiterung kann optional integriert werden.

1.2 Motivation

Der Entschluss zu diesem Thema liegt vor allem darin begründet, dass bisher nur spärliche Lösungsangebote zur GIS-Anwendung vorliegen, gleichzeitig aber eine recht große Nachfrage nach einer derartigen Möglichkeit existiert, was auch die Intevation GmbH bestätigt. Es gibt zwar bereits einige Lösungen, diese stellen allerdings eher Lösungsansätze als komplette Lösungsangebote dar.[11] Gerade dieses Manko soll mit der vorliegenden Arbeit beseitigt und die Erstellung für den Gelegenheitsanwender somit ermöglicht werden. Als Motivation kommt außerdem noch hinzu, dass Thuban schon durch weitere Extensions die Möglichkeit besitzt, Projekte aus anderen GIS-Programmen zu importieren, so z.B. von dem wohl prominentesten (proprietären) Produkt dieser Kategorie, ArcView[12]. Somit bietet sich an, dass Thuban als eine Art Konvertierungsprogramm fungieren könnte. Außerdem wird Thuban ständig weiterentwickelt und auch neue Extensions werden fortlaufend implementiert. Momentan. wird z.B an einer Extension gearbeitet, welche Thuban zu einem WMS Viewer machen soll. Damit wird es möglich, Daten von Quellen, welche eine WMS-Server (im Internet) bereitstellt, zu verarbeiten. Dies ist für die Zukunft interessant, weil auch der UMN MapServer als WMS-Server arbeiten kann.

9 http://qgis.org/; vgl. Kap.4.4.4: QGIS – Quantum GIS, S.50
10 http://www.jump-project.org/
11 vgl. Kap.4.4: Existierende Lösungsansätze, S.41
12 http://www.esri.com/software/arcview/

Die Entscheidung bei dieser Diplomarbeit ausschließlich Freie Software einzusetzen begründet sich zum einen darin, dass die Intevation GmbH bestrebt ist, bei all ihren Produkten, Entwicklungen und Dienstleistungen möglichst nur Freie Software einzusetzen, zum anderen in der Philosophie, die hinter Freier Software steht[13]. Bei Freier Software steht das Bewältigen eines Problems im Vordergrund und damit die Lösungsfindung, und nicht, wie bei proprietärer Software, das Erschaffen eines Produktes für den Verkauf und damit die Bindung des Kunden an den Entwickler. Zudem hat Freie Software den Vorteil, dass der Kunde nicht abhängig vom Hersteller ist, und den damit verbundenen Risiken. Bei proprietärer Software z.B. kann es passieren, dass die Entwicklung eingestellt wird, und der Kunde sich früher oder später nach einem neuen Produkt umschauen muss. Bei Freier Software kann der Kunde das Produkt, weil der Quellcode frei verfügbar ist, in einem solchen Fall selber weiterentwickeln oder weiterentwickeln lassen.

Da auch diese Diplomarbeit die Lösung eines Problems zum Ziel hat, entspricht sie mit ihrem Anliegen im Kern genau der Philosophie der Freien Software.

Zudem bietet Freie Software bessere Voraussetzungen bei der Lösungsfindung, weil sie nicht nur dem Anbieter, sondern auch dem Kunden die Möglichkeit bietet, optimale Bedingungen zu erschaffen und die eigenen Vorstellungen bestmöglich zufrieden zu stellen.

1.3 Gliederung der Arbeit

- **Kapitel 1 (Einleitung)**
  ist das aktuell gelesene Kapitel und soll ein kurzen Einblick in die Arbeit gewähren, worum es geht und warum dieses Thema und die entsprechende Methode gewählt worden ist.
- **Kapitel 2 (Grundlagen)**
  gibt eine Erklärung zum Begriff Freie Software und zu der dahinter stehenden Philosophie. Des Weiteren wird eine Erläuterung zum Begriff Geoinformationssysteme (GIS) und zu den Aufgaben des OpenGIS Konsortiums gegeben.

13 vgl. Kap.2.1: Freie Software, S.6

- **Kapitel 3 (Eingesetzte Technologien)**
  dient dazu, die wichtigsten Werkzeuge, welche bei dieser Arbeit benutzt werden, zu erläutern und Grundlagen für das bessere Verständnis zu schaffen. Neben der Erläuterung der Hauptwerkzeuge Thuban und UMN MapServer, bietet es Informationen zu dem Betriebssystem Debian GNU/Linux, der Programmiersprache Python und dem Softwareentwicklungssystem CVS.

- **Kapitel 4 (Analyse)**
  schildert die Vorgehensweise bei der Analyse des Projektes. Dazu zählt neben der Bestimmung der Festsetzung der erforderlichen Funktionalitäten auch eine Analyse von Thuban, dem UMN MapServer sowie bereits vorhandener Lösungsansätze.

- **Kapitel 5 (Design)**
  erläutert die Planung zum Aufbau der Extension. Dazu zählen Aufbau und Funktionsweise der Erweiterung.

- **Kapitel 6 (Realisierung)**
  widmet sich der Vorgehensweise bei der Implementierung und schildert Probleme und Schwierigkeiten, die während der Umsetzung auftraten.

- **Kapitel 7 (Resümee)**
  enthält die anschließende Diskussion der Arbeit. Dabei werden die Einhaltung der Zielsetzung hinterfragt sowie die Vorgehensweise bei der Umsetzung selbstkritisch bewertet. Zum Schluss werden noch offene Probleme angesprochen und abschließend ein Ausblick auf die weitere Zukunft des Programms gegeben.

2. Grundlagen

Free software
is a matter of liberty, not price.
To understand the concept, you should
think of 'free speech', not 'free beer'.[14]

Dieses Kapitel soll grundlegende Themen erläutern, die für das Verständnis der Diplomarbeit, der Lösungsansätze und -findung von Bedeutung sind. Die Grundlagen sind für das Nachvollziehen der Lösungsweges nicht unbedingt notwendig, dienen aber zum besseren Verständnis.

2.1 Freie Software

Mit dem obigen Zitat lässt sich die Bedeutung von „Frei“ und damit die Idee hinter Freier Software einfach und kompakt beschreiben. „Frei“ in Freie Software steht nicht für den Preis, sondern ist im Sinne von Freiheit zu verstehen. Insbesondere wird der Begriff Freie Software durch die folgenden vier Freiheiten[15] definiert:

14 Stallman, Richard M.: Free Software Foundation (FSF). 1996. URL: http://www.bravehack.de/html/node13.html. Stand: 04.07.2004
15 FSF Europe: Was ist Freie Software? 2004. URL: http://www.fsfeurope.org/documents/freesoftware.de.html.. Stand: 05.07.2004

- **Die Freiheit, das Programm für jeden Zweck auszuführen.**
  Einschränkungen bezüglich der Verwendbarkeit von Software machen ein Programm unfrei und zwar sind dies folgende:
  - Zeit ("30 Tage Testphase", "Lizenz endet am 1. Januar 2004"),
  - Zweck ("Verwendung gestattet für Forschung und nichtkommerzielle Anwendung") oder
  - willkürliche geographische Beschränkungen ("darf nur im Land X verwendet werden").

- **Die Freiheit, die Funktionsweise eines Programms zu untersuchen und es an seine Bedürfnisse anzupassen.**
  Rechtliche oder praktische Einschränkungen der Einsicht in die Programmfunktion oder der Veränderbarkeit eines Programms, wie der zwingende Erwerb spezieller Lizenzen, die Unterzeichnung eines Stillschweigeabkommens oder - bei Programmiersprachen, die mehrere Formen der Repräsentation bieten - die Zurückhaltung der üblicherweise bevorzugten Bearbeitungsform ("Quellcode") machen es ebenfalls proprietär (unfrei). Ohne die Freiheit, ein Programm zu ändern, bleiben die Anwender vom Wohlwollen eines einzigen Anbieters abhängig.

- **Die Freiheit, Kopien weiterzugeben und damit seinem Nachbarn zu helfen.**
  Software kann praktisch ohne Kosten kopiert und weitergegeben werden. Das Verbot, ein Programm an eine Person weiterzugeben, die es braucht, macht dieses Programm unfrei. Die Weitergabe kann wahlweise auch gegen ein Entgelt erfolgen.

- **Die Freiheit, ein Programm zu verbessern, die Verbesserungen an die Öffentlichkeit weiterzugeben, so dass die gesamte Gesellschaft profitiert.**
  Nicht jeder ist in allen Bereichen ein guter Programmierer. Die Mehrheit der Leute können überhaupt nicht selbst programmieren. Diese Freiheit erlaubt jenen, die nicht die Zeit oder die Fähigkeit haben, ein Problem selbst zu lösen, indirekt von der Freiheit, ein Programm zu ändern, Gebrauch zu machen. Auch das kann gegen ein Entgelt geschehen.

„Diese Freiheiten sind Rechte, keine Pflichten, auch wenn die Beachtung dieser Freiheiten der Gesellschaft von Zeit zu Zeit eine Verpflichtung für den Einzelnen darstellt. Jede Person kann wählen, auf die Freiheiten zu verzichten oder sie in Anspruch zu nehmen. Insbesondere ist es wichtig zu verstehen, dass Freie Software kommerzielle Anwendung nicht ausschließt. Wenn ein Programm kommerzielle Anwendung und kommerzielle Verbreitung nicht zulässt, ist es nicht Freie Software. In der Tat gründet eine wachsende Anzahl von Unternehmen ihr Geschäftsmodell vollständig oder zumindest teilweise auf Freie Software, einschließlich einige der größten proprietären Softwarehersteller. Freie Software erlaubt, Hilfe und Unterstützung anzubieten, sie erzwingt es aber nicht.“[16]

Die oben genannten vier Freiheiten werden durch die 1985 von Stallman gegründete „**F**ree **S**oftware **F**oundation“ (FSF) des **G**NU's **N**ot **U**nix (GNU) Projektes und dessen „**G**eneral **P**ublic **L**icense“ (GPL) geschützt. Die GPL ist momentan die am häufigsten verwendete Lizenz für Software.

Als Beispiel für Freie Software wäre das Betriebssystem Debian GNU/Linux[17] zu nennen. Aber auch alle weiteren in dieser Diplomarbeit genutzten Programme stellen Freie Software dar.

Große Plattformen zur Entwicklung Freier Software sind z.B. Sourceforge[18], tigris[19] oder savannah[20]. Dort findet man eine Menge Freier Software, die von unzähligen Helfern weltweit entwickelt und verbessert wird. Bei Sourceforge z.B. sind derzeit über 85.000 Projekte und über 896.000 User registriert. An diesen Dimensionen kann man gut die Verbreitung und Popularität von Freier Software erkennen oder wenigstens erahnen.

Weitere Artikel und Informationen zum Thema Freie Software sind auf der FSF Homepage oder in einigen Veröffentlichungen der **B**undeszentrale für **p**olitische **B**ildung (bpb)[21] zu finden. Besonders ist hier das als politisches Bildungsmaterial herausgegebene Buch „Freie Software – zwischen Privat-und Gemeineigentum“ von Volker Grassmuck zu nennen.[22]

16 FSF Europe: Was ist Freie Software? 2004. URL: http://www.fsfeurope.org/. Stand: 05.07.2004
17 vgl. Kap.3.1: Debian GNU/Linux , S.17
18 http://www.sourceforge.net
19 http://www.tigris.org/
20 http://savannah.gnu.org/
21 http://www.bpb.de/
22 Grassmuck, Volker: Freie Software – zwischen Privat- und Gemeineigentum. 2001

2.1.1 OpenSource ist nicht gleich Freie Software

Oft wird der Begriff OpenSource mit Freier Software gleich gestellt, was aber nicht ganz korrekt ist. Die Grundgedanken hinter dem Begriff OpenSource widersprechen dem von Freier Software. Während OpenSource auf sämtliche langfristigen Überlegungen zu Freier Software (wie Philosophie, Ethik und gesellschaftliche Effekte) zugunsten der Kommerzialisierung verzichtet und sich nur auf den technischen Vorteil konzentriert, stehen diese tief gehenden, grundsätzlichen Überlegungen bei Freier Software im Vordergrund.

2.1.2 Freie Software kommerziell genutzt

Oft wird irrtümlicher Weise auch angenommen, dass Freie Software nicht kommerziell ist. Auch dies ist so nicht korrekt und soll hier richtig gestellt werden. Freie Software wird heute von vielen Firmen, wie z.B. der Intevation GmbH als Geschäftsbasis genutzt und hat im Dienstleistungssektor ein großes Potential.

Zu der kommerziellen Nutzung zählen z.B. die Neuentwicklung oder Erweiterung Freier Software als Kundenauftrag. Aber auch die individuelle Anpassung Freier Software an Kundenwünsche sowie Wartung und Schulungen sind erfolgreiche Geschäftsmodelle.

2.1.3 Entwicklung durch Kommunikation

Der Entwicklungsprozess Freier Software lässt sich wohl am einfachsten und besten durch den Slogan „Entwicklung durch Kommunikation“ darlegen. Der Unterschied bei der Entwicklung Freier Software gegenüber proprietärer Software liegt in der Art und Weise wie die Software entsteht. Die Intention der Entwickler bei Freier Software ist eine ganz andere. Hier steht nicht, wie schon dargelegt, die Kommerzialisierung des Codes im Vordergrund, sondern der Wunsch Software zu erschaffen, die ganz persönlichen Ansprüchen genügt. Durch die Veröffentlichung als Freie Software kann sich jeder an der Entwicklung beteiligen und diese somit deutlich schneller verbessern. Die Entwicklung Freier Software wird von vielen

Menschen weltweit voran getrieben, wobei ein Großteil (40%) von „Profis“ (> 10 J. Programmiererfahrung) entwickelt wird. Zu den Helfern gehören unter Umständen aber auch Personen ohne Programmierkenntnisse, die mit Verbesserungsvorschlägen oder Fehlerberichten zur Entwicklung beitragen.

2.2 Geographische Informationssysteme

„Ein Geo-Informationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden.“[23]

2.2.1 GIS – eine Definition

„Der Begriff Geographisches Informationssystem wurde nach T.C. Walker/R.K. Miller (1990) bereits 1963 von R.F. Tomlinson bei der Einrichtung eines rechnergestützten raumbezogenen Informationssystems in Kanada eingeführt (vgl. auch R.F. Tomlinson (1972)). Mit dieser Bezeichnung erfolgte erstmalig der Hinweis auf eine neue Technologie – nämlich den Einsatz der elektronischen Datenverarbeitung in der raumbezogenen Datenhaltung. Analoge (geographische und raumbezogene) Informationssysteme in der Form von Karten und Buchwerken sind in Europa schon im 19. Jahrhundert flächenhaft aufgebaut und durch die zunehmende Verdichtung des anthropogenen Lebensraumes mit immer mehr Wissen angereichert worden. Aufgrund der Komplexität dieser Systeme wird deren Handhabung und Fortführung zunehmend erschwert; zudem bedingt die Interdisziplinarität von globalen und lokalen Fragestellungen wie z.B. im Umweltschutz einen schnellen Datenaustausch und damit Vernetzbarkeit, so dass die Umstellung der schon bestehenden, mehr oder weniger vollständigen analogen Informationssysteme auf die Methoden der **e**lektronischen **D**aten**v**erarbeitung (EDV) eine unabdingbare Notwendigkeit darstellt.“[24]

23 Ralf, Bill: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.4
24 Ralf, Bill: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.1

Der Begriff Geo-Informationssystem (GIS) für spezielle EDV-Systeme hat sich im deutschen Sprachraum fest etabliert. Der Begriff „geo“ oder auch „geographisch“ ist allerdings nicht ganz passend bzw. zutreffend gewählt. Besser wäre hier die Wahl des Begriffes „raumbezogen“ gewesen, weil ein Geo-Informationssystem ein um die räumliche Zuordnung der Objekte erweitertes Informationssystem ist. Diese Bezeichnung allerdings könnte zu Missverständnissen führen, weil mehrere Formen von GIS-Systemen existieren[25], zu denen auch das Rauminformationssystem gehört.

Der Begriff Geo-Informationssystem setzt sich als einheitliche Bezeichnung international immer mehr durch. So wird der Begriff inzwischen auch im englischen Sprachraum neben „spatial information“ und „geographic information“ akzeptiert.

Wie aus dem einleitenden Zitat dieses Kapitels zu erkennen ist, kann ein GIS durch ein „Vierkomponenten-Modell“ repräsentiert werden. Diese Kennzeichnung ist, wie Tabelle 1 zeigt, sowohl im Aufbau (Hardware, Software, Daten und Anwender) als auch in der Aufgabenbewältigung (Erfassung, Verwaltung, Analyse und Präsentation) möglich.

Hardware	H
Software	S
Daten	D
Anwender	A

E	Erfassen	Input	I
V	Verwalten	Management	M
A	Analyse	Analysis	A
P	Präsentation	Presentation	P

Tabelle 1: Vierkomponenten-Modell eines GIS[26]

Ein Geo-Informationssystem unterscheidet sich durch die beliebig tiefe Hinterlegung mit thematischen Daten grundsätzlich von den **C**omputer **A**ided **D**esign (CAD) Systemen oder sonstigen Informationssystemen. Die Verknüpfung von Geometrie- und Sachdaten bildet die Grundlage für die kombinierte Verwaltung, Analyse und Präsentation der Daten, wobei die thematische Karte nur ein mögliches Ausgabemedium darstellt.

25 vgl. Kap.2.2.3: Ausprägungen von GIS, S.13
26 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. Tabelle 1.2. S.29

2.2.2 Objektbildung

Die in einem Geo-Informationssystem enthaltenen Einheiten, die elementar oder zusammengesetzt sein können und die sowohl eine quantitative (geometrische) als auch eine qualitative (thematische) Komponente aufweisen, werden als raumbezogene Objekte oder nur kurz als „Objekte“ bezeichnet. Ein Objekt ist eine konkrete physische, geometrische oder begrifflich begrenzte Einheit der Natur und besitzt eine individuelle Identität. Von daher repräsentiert jedes Objekt ein Unikat in der realen Welt, das jedoch einer bestimmten Objektklasse zugeordnet werden kann. Anschaulich ist diese Objektdefinition in Abb.1 dargestellt.

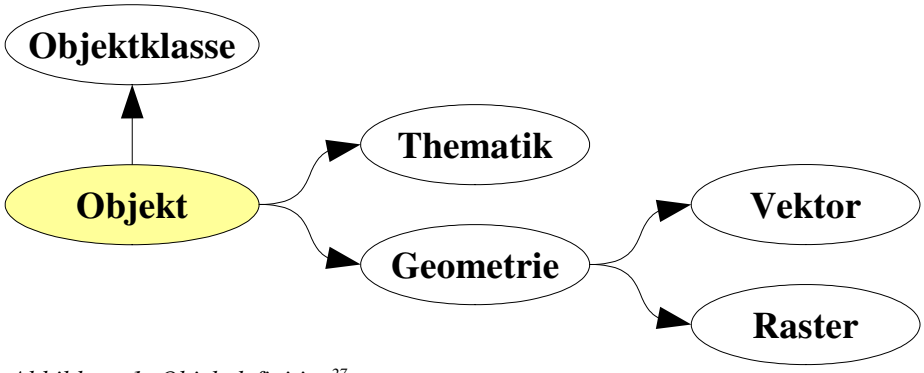


Abbildung 1: Objektdefinition[27]

Die Geometrie eines Objektes benennt sämtliche geometrischen Datenelemente in Vektor- oder in Rasterdarstellung. Diese ist in einem einheitlichen Bezugsrahmen definiert, der i.d.R. durch ein Koordinatensystem gegeben ist. Die Positionsdaten sind noch durch Daten der Nachbarschaftsgeometrie - topologische Daten genannt - zu ergänzen. Den Objekten können unterschiedliche thematische Beschreibungen oder Sachdaten zugeordnet werden. Die Zuordnung zu der jeweiligen Thematik geschieht über den Objektidentifikator. Eine Objektidentifikator ist z.B. bei einer Stadt der Name oder bei einem Flurstück die Flurstücksnummer.

Somit wird jedes Objekt durch die folgenden Charakteristika beschrieben:

- Geometriedaten (Vektor- oder Rasterdarstellung),
- Topologische Beziehungen (Knoten, Kanten, Flächen, Nachbarschaftsbeziehungen),
- Thematische Ausprägungen (Sachdaten oder Attribute),
- Objektidentifikatoren (Schlüssel).

27 Bill, Ralf.: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. Tabelle 1.2. S.11

„Durch Hinzunahme weiterer Daten und Medien, wie z.B. Bilder, Texte, Videoaufnahmen, können die eher abstrakten Daten im GIS zu anschaulicherern Repräsentationen geführt werden. In diesem Zusammenhang spricht man von Multimedia-GIS.“[28]

2.2.3 Ausprägungen von GIS

Aus der Notwendigkeit der Vielfalt der Anwendungen heraus haben sich verschiedene Fachdisziplinen eigene Geo-Informationssysteme aufgebaut. Heutzutage unterscheidet man zwischen den folgenden fünf Ausprägungen[29]:

- **Landinformationssysteme (LIS)** sind seitens des Vermessungswesens erschaffen worden und beziehen sich hauptsächlich auf die exakte geometrische Erfassung und Fortführung des „Grund und Bodens“ sowie auf damit verknüpfte Sachdaten. Das wohl bekannteste LIS ist das Liegenschaftskataster.

  Bereits 1982 definierte die **F**ederation **I**nternationale des **G**eometres (FIG):

  „Ein Landinformationssystem ist ein Instrument zur Entscheidungsfindung in Recht, Verwaltung und Wirtschaft sowie ein Hilfsmittel für Planung und Entwicklung. Es besteht einerseits aus einer Datensammlung, welche auf Grund und Boden bezogene Daten einer bestimmten Region enthält, andererseits aus Verfahren und Methoden für die systematische Erfassung, Aktualisierung, Verarbeitung und Umsetzung dieser Daten. Die Grundlage eines LIS bildet ein einheitliches, räumliches Bezugssystem für die gespeicherten Daten, welches auch eine Verknüpfung der im System gespeicherten Daten mit anderen bodenbezogenen Daten erleichtert.“[30]

- **Rauminformationssysteme (RIS)** sind seitens der Geographen, Raumplaner und Statistiker realisiert worden. Zu den breitgefächerten Aufgabenbereichen zählen die Erfassung der Bevölkerungsentwicklung, Wirtschaft und Siedlungen, amtliche Statistiken und Aufstellung von

28 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.11
29 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.35-46
30 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.36

Entwicklungsprogrammen. Als Raumbezug verwenden RIS i.d.R. Informationen vom LIS. Beispiele für Rauminformationssysteme sind z.B. Raumordnungskataster oder Flächennutzungspläne.

Zur Abgrenzung gegenüber LIS definiert R. Bill ein RIS wie folgt:

„ Ein Rauminformationssystem ist ein Instrument zur Entscheidungsfindung sowie ein Hilfsmittel für Planung und Entwicklung. Es besteht aus einer Datensammlung zur Bevölkerungs-, Wirtschafts-, und Siedlungsentwicklung, zum Infrastrukturausbau, zur Flächennutzung und den Ressourcen, die in regionale Entwicklungsprogramme und raumbedeutsame Vorhaben einfließen. Ebenso sind die Verfahren und Methode zur Erfassung, Aktualisierung und Umsetzung dieser Daten wesentlicher Bestandteil des Informationssystems. Die Grundlage bildet der einheitliche Raumbezug, der die verschiedenartigen Daten miteinander verknüpft.“[31]

- **Umweltinformationssysteme (UIS)** stellen eine weitere große Gruppe von Geo-Informationssytemen dar. Ihr Einsatzgebiet ist, wie der Name schon sagt, die Beschreibung der Umweltbelastung und/oder -gefährdung. Es ist die Grundlage für Maßnahmen des Umweltschutzes, wobei dem zeitlichen Ablauf eine besonders große Bedeutung zukommt.

  In Anlehnung an B. Page et Al. (1990) wird ein UIS wie folgt definiert:

  „Ein Umweltinformationssystem ist ein erweitertes GIS, das der Erfassung, Speicherung, Verarbeitung und Präsentation von raum-, zeit- und inhaltsbezogenen Daten zur Beschreibung des Zustandes der Umwelt hinsichtlich Belastung und Gefährdung dient und Grundlagen für Maßnahmen des Umweltschutzes bildet.“[32]

- **Netzinformationssysteme (NIS)** stellen in der Anwendung die größte Gruppe von GIS-Nutzern dar. Die Aufgabe eines NIS ist die Dokumentation und Bearbeitung von Betriebsmitteldaten, d.h. von Kundendaten bis hin zu den Leitungen und Anlagen zur Ver- und Entsorgung, wie z.B. Gas- oder Busnetze. Diese Systeme werden häufig auch als AM/FM-Systeme

31 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.38
32 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.40

bezeichnet, was sich aus Facility Management (FM) und Automated Mapping (AM) zusammensetzt.
Eine Definition für NIS lautet wie folgt:
„Ein Netzinformationssystem ist ein Instrument zur Erfassung, Analyse und Präsentation von Betriebsmitteldaten. Diese beziehen sich auf die Netztopologie, die in einem einheitlichen Bezugsrahmen gegeben sein muss.“[33]

- **Fachinformationssysteme (FIS)** sind alle Geo-Informationssysteme, die sich mit speziellen Aufgaben beschäftigen, durch die vorherigen aber nicht erfasst werden. Als Beispiele für FIS mögen z.B. die Analyse von Wellenausbreitungen zur Standortplanung für Sendemasten in Funknetzen oder digitale Straßenkarten als Grundlage für eine autonome Fahrzeugnavigation dienen.
  Potentielle Anwender von FIS sind vor allem die Telekom, Luft-, Bahn- und Schifffahrtsgesellschaften, die Standortbestimmungen und Navigationsaufgaben zu lösen haben.

2.3 OpenGIS Konsortium

Das **O**pen**G**IS **C**onsortium (OGC)[34] ist ein internationaler Zusammenschluss von Firmen, Behörden und Universitäten mit dem Hauptziel öffentliche Spezifikationen im GIS zu definieren. Die Spezifikationen des OGC sollen die Interoperabilität von GIS-Systemen ermöglichen und Standards schaffen. Neben einer ausführlichen, abstrakten Spezifikation stellt das OGC verschiedene Implementierungsspezifikationen zur Entwicklung auf verschiedenen Plattformen bereit. Als Beispiel für so eine Spezifikation wären z.B. **W**eb **M**ap **S**ervice (WMS) oder **W**eb **F**eature **S**ervice (WFS) zu nennen, welche beide vom UMN MapServer unterstützt werden.

Bei der Gründung im Jahre 1994 bestand das OGC gerade einmal aus acht

33 Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1. Heidelberg 1999. S.42f
34 http://www.opengis.org

Mitgliedern. Inzwischen ist die Zahl der Mitglieder auf über 250 gestiegen, die sich auf 37 Länder verteilen. Zu den Mitgliedern in Deutschland zählen unter anderem die Universitäten von Münster und Bremen[35], die Fraunhofer-Gesellschaft[36], die Hansa Luftbild Consulting International GmbH[37] und das German Aerospace Center – DLR[38].

2.4 FreeGIS

Das FreeGIS[39] Projekt hat sich zum Ziel gesetzt die Freiheit im Bereich Geographischer Informationssysteme (GIS) zu fördern.[40]

Seit der Gründung im Jahre 1999 soll dieses Ziel, welches sich stark an den Zielen der FSF orientiert, durch die Förderung folgender Funktionen erreicht werden:

- Verwendung, Entwicklung und Unterstützung Freier GIS Software,
- Verwendung und Veröffentlichung freier Geo-Daten im gleichen Sinne von Freiheit wie bei Freier Software,
- Verwendung, Beschreibung, Übersetzung und Erweiterung von freien Dokumenten im Bereich GIS.

Das FreeGIS Projekt besteht aus

- einer Website, die Informationen über Freie Software, Daten und Dokumente anbietet,
- einem Diskussionsforum, um Neuigkeiten, Bekanntmachungen, Vorschläge und Ideen öffentlich zu diskutieren,
- einer FreeGIS-CD, die eine Zusammenstellung diverser Freier GIS Software und freier Geodaten anbietet und
- einem „FreeGIS Tutorial“, welches anhand von detaillierten Beispielen die Verwendungsmöglichkeiten von freier GIS-Software aufzeigt.

35 http://uni-muenster.de, http://www.uni-bremen.de
36 http://www.fraunhofer.de
37 http://www.hansaluftbild.de
38 http://www.dlr.de
39 http://www.freegis.org
40 Intevation GmbH: Über das FreeGIS Projekt. URL: http://freegis.org/about.de.html. Stand: 07.08.2004

3. Eingesetzte Technologien

Die Grundlage ist
das Fundament der Basis.[41]

Wie schon aus den vorherigen Kapiteln hervorgeht, wird die Diplomarbeit komplett mit Freier Software erstellt und bearbeitet. Im Folgenden soll nun die verwendete Software kurz beschrieben werden, um Einblick in das Einsatzgebiet und Verständnis für die Funktionsfähigkeit der verwendeten Software zu vermitteln.

3.1 Debian GNU/Linux

„Ein Betriebssystem ist ein Satz von grundlegenden Programmen, die ein Rechner zum Arbeiten benötigt. Der Kern (kernel) ist das Stück Software, welches für alle Basisaufgaben (Zugriffe auf die Hardware usw.) von anderen Programmen zuständig ist.“[42]

GNU/Linux ist ein Unix-ähnliches Multiuser/Multitasking Betriebssystem. Der Linux Kernel wurde ursprünglich auf der Intel x86 Plattform entwickelt und später auf andere Prozessor-Architekturen portiert. Das GNU-System wurde von Anfang an portabel ausgelegt und kann so sehr schnell auch an den Linux-Kernel angepasst werden.

41 Le Corbusier, französisch-schweizerischer Architekt (1887 - 1965)
42 Ronneburg, Frank: Debian GNU/Linux Anwenderhandbuch, 1999-2004, URL: http://debiananwenderhandbuch.de. Stand: 24.07.2004

Abbildung 2: offizielles Debian GNU/Linux Logo

Debian verwendet als Betriebssystemkern „Linux“, eine Freie Software, die von Linus Torvalds gestartet wurde und von Tausenden von Programmierern auf der ganzen Welt unterstützt wird.

„Ein großer Teil der grundlegenden Anwendungen stammt aus dem GNU-Projekt und ist ebenfalls frei; daher der Name GNU/Linux. GNU steht für „**G**NU's **N**ot **U**nix!“ und wurde von Richard Stallman ins Leben gerufen. Im Gegensatz zu anderen Linux-Distributionen wird Debian GNU, ähnlich wie der eigentliche Linux-Kernel, von einer großen Gruppe von Freiwilligen auf der ganzen Welt zusammengestellt.“[43]

Das Debian-Projekt wurde offiziell von Ian Murdock am 16. August 1993 gegründet. Ian Murdock begann diese neue Distribution als offenes Entwicklungsprojekt, ganz im Sinne des GNU- oder des Linux-Kernel-Projektes.

Der Name „Debian“ stammt vom Schöpfer der Distribution, Ian Murdock, der den Namen aus dem Namen seiner Frau, Debra, und seinem eigenen Vornamen bildete (Deb-Ian).[44]

3.2 Python

Python ist eine interpretierte, höhere, objektorientierte Programmiersprache mit der Unterstützung für Mehrfachvererbung und Überladung von Methoden und Operatoren. Python ist als Freie Software plattformunabhängig und somit für Unix, Linux, Windows, OS/2, Mac, Amiga und viele andere Plattformen verfügbar. Python gehört in die Kategorie der Scriptsprachen, d.h. es ist kein expliziter Übersetzungsvorgang durch einen Compiler notwendig. Der Programmcode wird beim ersten Ausführen des Programmes übersetzt und als Bytecode gespeichert.

43 Bramer, Michael/Goerzen, John/Othman, Ossama: debian GNU/Linux 3.0 Guide. 2002. S 1ff
44 Ronneburg, Frank: Debian GNU/Linux Anwenderhandbuch, 1999-2004, URL: http://debiananwenderhandbuch.de. Stand: 24.07.2004

Um die Programme übersichtlich und strukturiert zu gestalten, werden die Dateien in Ordnern strukturiert. Diese Ordner nennt man in Python „Module“. Ein „Modul“ kann somit viele verschiedene Dateien aufnehmen, welche wiederum die Klassen, Methoden und Operatoren enthalten.

```
 1 # Primzahlen
 2
 3 for num in range(2, 10):
 4     for x in range(2, num):
 5         if num % x == 0:
 6             print num, 'equals', x, '*', num/x
 7             break
 8         else:
 9             print num, 'is a prime number'
10             break
```

Quellcode 1: Einfaches Beispiel für Pythoncode.

Wie in diesem Beispiel zu sehen ist, werden Ausdrücke, die einen Block einleiten mit einem Doppelpunkt beendet und nachfolgende Codeblocks eingerückt. Beim Einrücken ist darauf zu achten, dass Blöcke immer gleich weit eingerückt werden um die Zusammengehörigkeit zu verdeutlichen. Dabei ist bei der Programmierung darauf zu achten, dass für das Einrücken konsequent gleiche Zeichen verwendet werden, also entweder Tabulatoren oder Leerzeichen. Andernfalls können beim Ausführen Probleme seitens des Interpreters auftreten.

Die Syntax der Sprache ist, wie in Quellcode 1 zu erkennen, einfach, elegant und klar strukturiert. Sie ermöglicht, aufgrund zahlreicher vorgefertigter Funktionen und Objekte die Ausführung mächtiger Aktionen mit wenig Programmcode.

Die relativ schnelle Anwendungsentwicklung wird auch durch die dynamische Typzuweisung, die dynamische Bindung an Datentypen sowie abstrakte, eingebaute Datenstrukturen wie Arrays, Listen, Tupel, Dictionaries und Exceptions unterstützt. Dynamische Typzuweisung bedeutet, dass während der Variablendeklaration keine explizite Typzuweisung stattfinden muss, sondern der Datentyp dynamisch entsprechend des zugewiesenen Wertes vergeben wird. Trotz der dynamischen Typzuweisung ist Python streng typisiert. Nach der Wertzuordnung ist der Datentyp der Variablen verbindlich und muss zur Verwendung für Operationen, die einen anderen Datentyp erwarten, explizit konvertiert werden.

Eine weitere Besonderheit von Python ist, dass die Sprache nicht nur Python

Module verwenden kann, sondern auch durch C oder C++ Module erweitert werden kann. Dadurch ist es möglich, rechenintensive Prozesse in diese doch wesentlich „performantere“ Sprache zu verlagern.

Die Speicherverwaltung erfolgt automatisch, d.h. es ist nicht nötig sich um die Reservierung oder das Freigeben von Speicherplatz zu kümmern.

Des Weiteren unterstützt Python leistungsfähige Programmbibliotheken, z.B. für Bereiche der nummerischen Verarbeitung, Bildbearbeitung oder Entwicklung von grafischen Benutzerschnittstellen (Graphical User Interface, kurz GUI). Zu diesen GUI-Toolkits gehören z.B. Tkinter[45], ein Python Aufsatz auf Tcl/Tk, oder wxPython[46], ein Python-Wrapper für wxWidget[47] (ehemals wxWindows), dem hier wegen einiger Vorteile und der Tatsache, dass Thuban auf wxWidget basiert, der Vorzug gegeben wird. WxWidget ist ein C++ GUI framework zur Erstellung von plattformunabhängigen GUI-Anwendungen und ist u.a. verfügbar für Windows, Unix, Linux und Macintosh OS X. Durch diese Plattformunabhängigkeit ist es für den Python-Programmierer ein Leichtes, funktionale grafische Anwendungen für alle Plattformen, wie Linux oder Windows, zu schreiben.

Ein großer Vorteil von wxPython ist, dass die Programme das plattformtypische Aussehen und Verhalten bekommen, so dass sich der Anwender gleich wie „zu Hause“ fühlt. Erreicht wird dies, im Unterschied zu vielen anderen GUI-Toolkits, dadurch, dass wxPython als Abstraktion der nativen Bibliotheken des jeweiligen Betriebssystems fungiert.

Im Rahmen dieser Diplomarbeit wird also die GUI-Bibliothek wxPython verwendet. Wie auch WxWidget ist WxPython Freie Software.

Aus den genannten Gründen erfreut sich Python heute zunehmender Popularität, zumal sie für eine moderne Computer-Hochsprache relativ leicht zu erlernen und somit auch für Programmieranfänger gut geeignet ist. Hohes Vertrauen genießt Python mittlerweile auch bei vielen großen Unternehmen, so baut etwa die bekannte Suchmaschine Google[48] auf Python auf.

45 http://www.python.org/topics/tkinter/
46 http://www.wxpython.org
47 http://wxwidgets.org
48 http://www.google.de

Entwickelt wurde Python 1990 von Guido van Rossum am „**C**entre voor **W**iskunde en **I**nformatika" (CWI)[49] in Amsterdam. Mittlerweile wird Python von der „**P**ython **S**oftware **F**oundation" (PSF)[50] weiter geführt. Van Rossum benötigte damals eine Testumgebung für ein neues Betriebssystem. Weil er aber keine geeignete Programmiersprache fand, entwickelte er kurzerhand seine eigene. Der Name Python ist übrigens nicht etwa von der bekannten Schlange hergeleitet, sondern von der berühmten britischen Comedy-Truppe "Monty Python".

3.3 Thuban - Interactive Geographic Data Viewer

Thuban ist ein interaktiver Betrachter und Editor für Geodaten, mit dem es möglich ist, räumliche Daten zu visualisieren, abzufragen und zu verändern.

Geodatenbetrachter dienen zur Visualisierung von Geodaten, sie sind nützliche und notwendige Werkzeuge, um einen Eindruck von Positions- und Dimensionsverhältnissen zu bekommen, wie das Island-Beispiel (Abb.3) verdeutlicht.

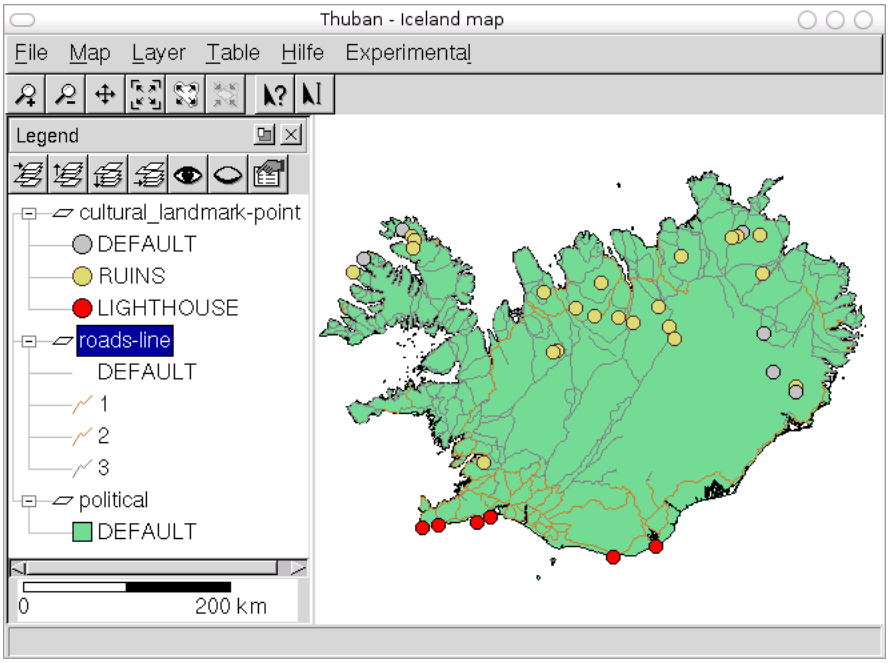


Abbildung 3: Hauptfenster von Thuban mit dem Beispieldatensatz "Iceland"

49 http://www.cwi.nl/
50 http://www.python.org/psf/

Thuban basiert zum größten Teil auf Python und dem wxPython Framework, ist somit Plattform-unabhängig implementiert und unter anderem für Windows und Linux verfügbar. Einige Module jedoch, welche besonders rechenintensiv sind, wurden in C und C++ geschrieben um die Performance zu verbessern.

Thuban ist in der Lage neben Vektordaten im Shapefile-Format auch Daten aus räumlichen Datenbanken, basierend auf PostgreeSQL/PostGIS, und Rasterdaten zu verarbeiten. Die aktuell verarbeiteten Daten werden als Karte in einer Sitzung (Session) zusammengefasst und sowohl als Hierarchiebaum als auch als Karte dargestellt. Die Karte setzt sich aus einzelnen Schichten (Layern) zusammen, die jeweils einen Datensatz (Shapefile) enthalten. Ein Datensatz kann z.B. Gebäude (Punkte), Straßen (Linien) oder Länderumrisse (Polygone) enthalten, wie Abb.3, S.21 verdeutlicht.

Thuban wurde von der Intevation GmbH entwickelt, weil es für das FreeGIS Projekt keinen einfachen Geodatenbetrachter auf Basis von Freier Software gab. Um diese Bedarfslücke zu schließen, wurde es als Basis für weitere GIS-Applikationen entworfen. Ein Einsatzbeispiel ist das Simulationssystem GREAT-ER (**G**eography **R**eferenced Regional **E**xposure **A**ssessment **T**ool for **E**uropean **R**ivers). Thuban ist als Freie Software unter der GNU General Public License lizenziert.

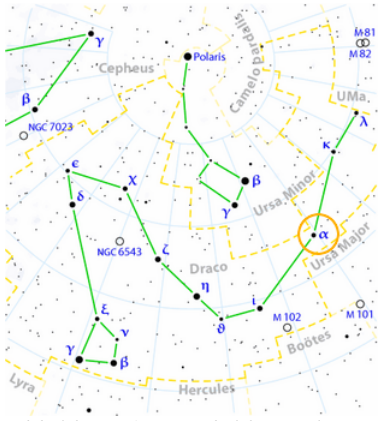


Abbildung 4: Sternbild Drache

Thuban bietet eine Schnittstelle zur Entwicklung von Extensions an. Ein experimenteller WMS-Client und der Import von APR-Dateien (Arcview Projekt) sind dafür nur zwei Beispiele. Auch Mehrsprachigkeit wird unterstützt. Bisher stehen die Sprachen Englisch, Französisch, Deutsch, Italienisch, Spanisch und Russisch (kyrillische Schrift) zur Verfügung.

Der Name Thuban (arab.: „der Drache") stammt vom Alpha-Stern des Sternbildes Draco (Abb.4). Thuban war vor ca. 5.000 Jahren bei den Arabern der Polarstern des Nordhimmels.[51]

51 wikipedia: Drache (Sternbild). 2004 URL: http://de.wikipedia.org/wiki/Drache_(Sternbild). Stand: 26.07.2004

3.4 UMN MapServer

Der UMN MapServer ist eine Freie Software zur Erstellung von Kartenansichten (Abb.5) und zur Darstellung von räumlichen Daten über das Intra- bzw. Internet. Entwickelt wurde er von der **U**niversität of **Min**nesota (UMN) in Zusammenarbeit mit der NASA und dem **Min**nesota **D**epartment of **N**atural **R**esources (MNDNR).

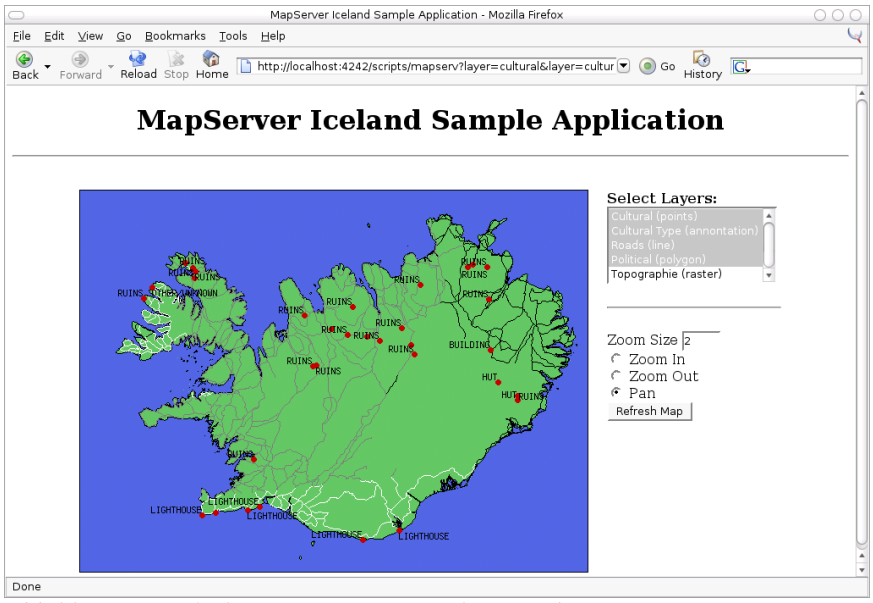


Abbildung 5: Einfache MapServer Beispiel-Anwendung.

Genau genommen ist der UMN MapServer kein Server, sondern nur eine Erweiterung für einen Webserver[52], wie z.B. dem Apache WebServer[53]. Er stellt auch kein voll funktionsfähiges GIS zur Verfügung, bietet aber dennoch eine breit gefächerte Funktionalität für internet/intranet-relevante Anwendungen. Die Funktionalität umfasst vor allem die Visualisierung und grundlegende Analyse von Geodaten. Der UMN MapServer unterstützt eine große Zahl unterschiedlicher Geodatenformate, wie ESRI-Shapefiles, ArcSDE, Oracle Spatial, PostgreSQL/PostGIS, GeoTIFF, ERDAS u.a.. Zudem verfügt der UMN MapServer über viele Features, z.B. TrueType Schriften, automatisches Generieren von Maßstab und Legende oder automatisches Beschriften der Layer, um nur einige zu nennen. Trotz all dieser Möglichkeiten, die der UMN MapServer bietet, ist die Präsentation eingeschränkt. Diese eingeschränkte Funktionalität lässt sich aber durch andere Webtechnologien (z.B. JavaScript) in gewissem Maße beseitigen.

52 vgl. Kap.3.5: Web Server, S.24
53 http://httpd.apache.org/

Neben der Erweiterung durch Webtechnologien ist es möglich, den UMN MapServer mit anderen Programmiersprachen zu verwenden. Hierfür stellt der UMN MapServer mit dem MapScript eine Programmierschnittstelle (**A**pplication **P**rogramming **I**nterface, kurz API) für Sprachen wie Perl, PHP, Java oder Python zur Verfügung. Zudem ist der MapServer als OGC konformer WMS-Server/Client in der Lage, Quellen bzw. Daten über eine definierte HTTP-Schnittstelle zu empfangen und zur Verfügung zu stellen.

Zur Konfiguration und Steuerung der Darstellung des UMN MapServers ist neben den Daten selbst noch ein Mapfile erforderlich. Das Mapfile ist eine einfache ASCII-Datei mit der Endung ”.map“, die alle Informationen zur Generierung einer grafischen Darstellung der Geodaten enthält, wie z.B. Kartenausdehnung, Darstellungsfarbe, Hintergrundfarbe, Symboldefinitionen, Maßstab und Legende in Form von Schlüssel-Werte-Paaren (etwa *COLOR 0 0 255* für Blau). Die im Mapfile dargestellten Objekte folgen einer hierarchischen Struktur. So sind beispielsweise Layerobjekte innerhalb des Mapobjektes und Klassenobjekte in Layerobjekten definiert.[54]

3.5 Web Server

Ein Server (lat. "Diener") ist in der Informatik ein Dienstleister, der in einem Netzwerk Daten oder Ressourcen zur Verfügung stellt. Ein Webserver stellt z.B. Webseiten, Bilder und Stylesheets zur Verfügung, für die das HTTP Protokoll verwendet wird.

Oft wird der Computer, auf dem dieses Computerprogramm läuft, fälschlicherweise als Server bezeichnet. Die korrekte Bezeichnung für diesen physikalischen Rechner ist allerdings „Host“. Die Gegenstelle, also der Computer, der die Anfragen an den „Host“ stellt, nennt man „Client“. Im Fall eines Webservers würde beispielsweise jedes Mal, wenn ein Benutzer auf einen Link klickt, eine Anfrage nach der jeweiligen Webseite an den Webserver geschickt, der dann im Gegenzug die geforderte Seite zurücksendet. Server werden somit nicht von sich aus aktiv, sondern warten, bis sie eine Anfrage bekommen, die sie ausführen können.

54 vgl. Kap.4.3.2: MapServer / Mapfile, S.38; vgl. Anhang A.2, S.89

Es gibt viele verschiedene WebServer, sowohl freie, wie z.B. den Apache HTTP WebServer (Apache) von der Apache Software Foundation[55], als auch proprietäre, wie z.B. den **I**nternet **I**nformation **S**ervice (IIS) von Microsoft[56]. Neben diesen wohl bekanntesten und auch verbreitetsten gibt es auch noch kleinere, die sich auf einige wenige Webservices spezialisiert haben. Dazu zählt z.B. der in dieser Arbeit zum Testen eingesetzte mini-httpd.[57]

Neben dem HTTP Webserver gibt es noch etliche andere Server, welche Dienste bereit stellen, beispielsweise FTP-Server, E-Mail-Server, Datenbankserver oder Print-Server. Server dienen auch nicht nur zur Bereitstellung von Daten an den Client (download), sondern auch zum Empfangen von Daten vom Client (upload). Ein klassisches Beispiel für Server, die bidirektionalen Datentransfer unterstützen, sind FTP Server.

3.6 CVS - Concurrent Versions System

Die Verwaltung eines (umfangreichen) Software-Projekts verlangt viel Aufmerksamkeit. Um ältere Versionen wiederherzustellen (z.B. für die Fehlersuche), muss man immer wieder Sicherungskopien der aktuellen Software machen und sie mit entsprechender Versionsnummer versehen und verwalten. Diese Methode ist umständlich und benötigt unnötig viel Speicherplatz, da erfahrungsgemäß die meisten Veränderungen/Erweiterungen nur in kleinen Schritten vorgenommen werden. Außerdem muss bei mehreren Entwicklern, die gleichzeitig an einem Projekt arbeiten, darauf geachtet werden, dass Änderungen bzw. Erweiterungen der Software durch einen Entwickler nicht aus Unachtsamkeit durch einen anderen Entwickler überschrieben und somit rückgängig gemacht werden.[58]

55 http://www.apache.org/
56 http://www.microsoft.de/
57 http://www.acme.com/software/mini_httpd/
58 Hatzack, Wolfgang /Weigel, Thilo: Einführung in CVS. 1999-2002. URL: http://www.informatik.uni-freiburg.de/~ki/lehre/ss02/Sopra/cvs.html. Stand: 11.07.2004

Um die Datenverwaltung von Projekten einfacher und sicherer zu gestalten, wurde als Freie Software das **C**oncurrent **V**ersions **S**ystem (CVS)[59] entwickelt. Es handelt sich um ein Client/Server-System, das die zu verwaltenden Dateien mit Hilfe eines Serverprogramms betreut. Die Benutzer kommunizieren mit Server über ein Client-Programm, mit dem sich alle Arten von Dokumenten und Dateien verwalten lassen. Hierzu gehören z.B. Quelldateien von Programmen und Dokumentationen, aber auch Bücher. CVS hilft bei der Verwaltung von Projekten, an denen mehrere Entwickler im Team arbeiten, und es sorgt dafür, dass Änderungen eines Projektes, die von verschiedenen Teammitgliedern ausgeführt werden, nicht durcheinander geraten.

Um das Projekt allen Teammitgliedern zur Verfügung zu stellen und Änderungen zu verwalten, werden alle Dateien eines solchen Projektes von CVS in einem zentralen Repository gehalten. Will jemand mit diesen Dateien arbeiten, so müssen diese zuerst aus dem Repository in den eigenen Arbeitsbreich kopiert werden.[60] Nach der Bearbeitung der Dateien müssen diese wieder zurück in das Repository geschrieben werden, und genau hier kommt CVS zum Einsatz. CVS führt Buch über alle Änderungen und sorgt beim Zurückschreiben dafür, dass zwischenzeitlich gemachte Änderungen anderer Teammitglieder nicht überschrieben werden und verloren gehen. Sollten die Daten während der Bearbeitung durch andere Teammitglieder verändert worden sein, so informiert CVS den Programmierer darüber. Wenn die Änderungen an unterschiedlichen Stellen getätigt wurden, bietet CVS die Möglichkeit, die Änderungen auf konsistente Weise zusammenzuführen. Kommt es dabei zu Konflikten, müssen diese "manuell" aufgelöst werden.

Die Handhabung von CVS ist sehr einfach gehalten. CVS versteht über 50 verschiedene Befehle, die aber normalerweise nicht alle benötigt werden.

Zur Anwendung kommt CVS mit Hilfe des Befehls „cvs“. Im Regelfall wird CVS genutzt, indem man ein Kommando mit dem Befehl übergibt (z.B. cvs init). Zusätzlich können noch ein oder mehrere Parameter kombiniert werden.

59 https://www.cvshome.org/

60 Stierrand, Ingo: CVS Tutorial. 2002. URL: http://www.stierand-linuxit.de/Doku/cvs-tutorial.html. Stand: 11.07.2004

In der Regel werden nur sieben Befehle zum arbeiten mit CVS benötigt:

- **cvs init erzeugt**
  ein neues Repository und wird gewöhnlich nur beim ersten Mal benötigt.
- **cvs import**
  fügt dem Repository ein neues Projekt hinzu.
- **cvs checkout**
  kopiert die Daten aus dem Repository in den eigenen Arbeitsbereich, wird nur beim allerersten Mal benötigt, um einen lokalen Arbeitsbereich anzulegen.
- **cvs commit**
  stellt Änderungen am eigenen Arbeitsbereich ins Repository.
- **cvs update**
  gleicht zwischenzeitliche Änderungen im Repository mit den Dateien im eigenen Arbeitsbereich ab.
- **cvs add**
  fügt eine neue Datei zum Projekt im Repository hinzu.
- **cvs remove**
  entfernt Dateien aus dem Repository.

Neben den vielen Vorteilen hat CVS aber auch einige Nachteile, die die Arbeit mit CVS erschweren können Dazu zählt z.B. ein sehr hoher Speicherbedarf auf dem Datenträger oder die nicht vorhandene Möglichkeit Dateien im Repository umzubenennen und zu verschieben.

Ein mögliches Nachfolgesystem von CVS ist Subversion[61], zumal es die Schwächen von CVS nicht besitzt und auch als Freie Software zur Verfügung steht. Anders als CVS kann Subversion auch mit Binärdateien umgehen. Es stellt sicher, dass gleichzeitige Änderungswünsche zweier Entwickler sicher nacheinander abgewickelt werden, und es vereinfacht das Umbenennen von Dateien und den Umgang mit Dateiattributen.

61 http://subversion.tigris.org/

4. Analyse

Ernst zu nehmende Forschung erkennt man daran,
dass plötzlich zwei Probleme existieren,
wo es vorher nur eines gegeben hat.[62]

Für die Lösung der vorliegenden Aufgabe sollen zunächst die zu erfüllenden Anforderungen genauer definiert werden. Zudem werden die technischen Möglichkeiten von Thuban und dem UMN MapServer analysiert und die schon vorhandenen Lösungsansätze unter die Lupe genommen, um anschließend mit Hilfe der gewonnenen Erfahrungen die Thuban-Erweiterung gezielt für die eigentliche Implementierungsphase[63] zu modellieren.[64]

Durch diese Vorgehensweise wurde sichergestellt, dass dieses Projekt vor dem Hintergrund der Zielsetzung im vorgegebenen zeitlichen Rahmen zu bewältigen war.

4.1 Anforderungen

Wie schon aus dem Thema zu entnehmen, soll eine Extension für Thuban erstellt werden. Um dies nun genauer zu differenzieren, wird im Folgenden abgegrenzt, welche Funktionen in die Extension zu integrieren, welche optional und welche gar nicht zu berücksichtigen sind.

Neben den selbst gestellten Anforderungen und Einschränkungen gibt es auch

62 Thorstein Bunde Veblen (1857-1929), amerik. Soziologe u. Ökonom
63 vgl. Kap.6: Realisierung, S.61
64 vgl. Kap.5: Design, S.53

noch einige technische Grenzen, welche bedacht werden sollten bzw. müssen. Hierzu zählt z.B., dass der UMN MapServer in der Standardkonfiguration nur maximal 50 Layer pro Mapfile bearbeiten kann. Aber auch die Funktionsunterschiede der verschiedenen UMN MapServer Versionen müssen Beachtung finden. Als Grundlage für das Projekt wird die zum jetzigen Zeitpunkt aktuellste Version (4.2) verwendet.

4.1.1 Musskriterien

Die Musskriterien definieren, welche grundlegenden Funktionen durch die Extension unbedingt abgedeckt werden müssen. Diese Funktionen sind für die Zielsetzung und die damit zu erfüllende Aufgabe unbedingt zu integrieren und müssen bei der Implementierung erfasst werden. Die folgenden Funktionen sind dabei von der Extension zu unterstützen:

- Mapfiles sollen importiert werden und in Parameter in Thuban abgebildet werden, soweit Thuban die Einstellungen unterstützt.
- Thuban Projekte sollen exportiert werden können, dabei soll das Layout im Browser, soweit möglich, dem von Thuban entsprechen.
- Die Extension soll die Fähigkeit besitzen vorhandene Mapfiles editieren zu können. Dazu gehören auch Einstellungen des Mapfiles, welche von Thuban nicht unterstützt werden. Dabei sollen Kommentare und die Struktur des Mapfiles weitestgehend erhalten bleiben.

4.1.2 Wunschkriterien

Die Wunschkriterien sind diejenigen Funktionen, welche für die Funktionsweise der Extension nicht unbedingt notwendig sind und somit nicht zwangsläufig implementiert werden müssen. Diese optionalen Kriterien erweitern die Funktionsfähigkeit und Handhabung der Extension und sollten, wenn möglich, umgesetzt werden, da sie die Attraktivität des Programms steigern. Im Folgenden seien hier zwei mögliche Funktionsfähigkeiten genannt:

- Optional sollte beim Export ein komplettes MapServer Projekt mit HTML-Dateien erstellt werden, um das Projekt anschließend testen zu können.
- Es könnte ein HTTP-Webserver integriert werden, um die Extension mit einer Vorschaufunktion zu erweitern.

4.1.3 Abgrenzungskriterien

Die Abgrenzungskriterien sind jene Funktionalitäten, welche durch diese Diplomarbeit nicht geleistet werden sollen bzw. können:

- Die Extension soll nicht als komplettes UMN MapServer Projekt Administrationstool dienen, sondern lediglich zur Bearbeitung der Geodaten im Mapfile.
- Verknüpfte Dateien im MapServer wie z.B. Grafiken oder HTML Dateien, sollen hier nicht auf Wunsch in dem Standardeditor für das Dateiformat zum Bearbeiten geöffnet werden können. Für eine Raster-Grafikdatei wäre dies z.B. das Bildbearbeitungsprogramm GIMP[65].

4.1.4 Zielgruppe

Aufgrund der Tatsache, dass Thuban als Freie Software entwickelt wird, ist es grundsätzlich von jedem nutzbar. Allerdings kann man den Kreis der Nutzer mit der Annahme eingegrenzt, dass wohl eher geographisch Interessierte mit diesem Programm arbeiten werden. Somit kann in dieser Hinsicht auch ein gewisses Maß an Wissen vorausgesetzt werden. Andererseits, da es sich bei den potentiellen Anwendern wohl eher um gelegentliche Nutzer handelt, können tiefer gehende Computerkenntnisse nicht unbedingt vorausgesetzt werden. Darauf sollte bei der Umsetzung in gewissem Maße durch eine anwenderfreundliche Handhabung Rücksicht genommen werden.

4.2 Anwendungsfälle

Die Anwendungsfälle, welche in den Anforderungen definiert werden, sollen nun genauer erläutert und beschrieben werden. Auf die optionalen Punkte wird, soweit diese nicht umgesetzt wurden, nicht weiter eingegangen.

Die Anwendungsfälle sind in Abb.6, S.31 grafisch dargestellt und werden im Folgenden kurz erläutert. Dabei wird davon ausgegangen, dass sich an der Funktionalität von Thuban nichts ändert.

65 http://www.gimp.org/

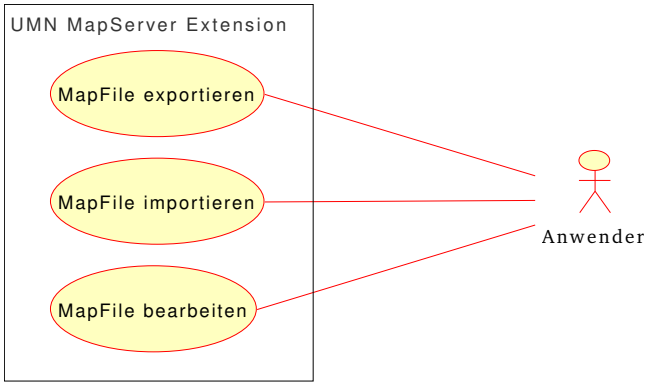


Abbildung 6: Anwendungsfälle

4.2.1 Export

Die Exportfunktion soll, wie der Name schon vermuten lässt, die Möglichkeit bereitstellen, Mapfiles aus Thuban-Projekten zu erstellen. Dabei sollen alle Einstellungen, wie die Projektion, die Layer mit Klassifizierung oder die Symbole aus Thuban, so weit wie möglich im Mapfile abgebildet werden. Weitere Angaben, die im Mapfile existieren müssen (wie z.B. die Bildgröße), in Thuban aber nicht abbildbar sind, müssen vom Anwender abgefragt werden.

4.2.2 Import

Die Importfunktion stellt die Methode zur Verfügung, mit deren Hilfe ein Mapfile in Thuban eingelesen wird. Dabei sollen alle Informationen, welche direkt in Thuban abbildbar sind, „übertragen“ werden. Alle anderen Angaben können ignoriert werden, weil sie für das Thuban-Projekt nicht von Bedeutung sind.

4.2.3 Editierung

Die dritte Funktion kombiniert in gewisser Weise die Import- und Exportfunktion, erweitert sie dabei aber um einen entscheidenden Punkt. Beim Editieren soll die Struktur des Mapfiles mit allen Kommentaren und Informationen erhalten bleiben. Sie müssen während einer *Session* existent bleiben, um sie anschließend wieder im Mapfile abzuspeichern. Dazu zählen auch diejenigen Angaben, welche in Thuban nicht direkt abbildbar sind.

4.3 Technische Untersuchung

Die technische Analyse soll vorrangig die für dieses Projekt verwendete Software (Thuban und UMN MapServer) erläutern, aber auch bestehende Lösungen anderer Produkte analysieren und bewerten. Die Ergebnisse der technischen Analyse werden für den weiteren Ablauf der Diplomarbeit verwendet.[66]

4.3.1 Thuban

Zuerst ist es wichtig, die Funktionsweise und den Aufbau von Thuban genauer zu verstehen, um die Extension ohne große Schwierigkeiten in Thuban integrieren zu können, zumal einige Klassen, welche Thuban bereitstellt, in der Extension Verwendung finden.

Thuban baut auf dem leistungsfähigen Model-View-Control (MVC) Konzept auf, bei dem die einzelnen Aufgaben eines Programmes getrennt werden:

- Model (Modell) ist das zu behandelnde Ding,
- Views (Ansichten) sind Präsentationen des Models,
- Control (Steuerung) ist Einflussnahme auf das Model.

Thuban ist in die Pakete *Model*, *UI*, *Lib* sowie *Extensions* unterteilt. Von diesen sind hier hauptsächlich die Pakete *Model* und *UI* von Bedeutung. Das Paket *Lib* enthält allgemeine Funktionen für Thuban, auf die nicht weiter eingegangen werden soll, da sie für das Projekt nicht relevant sind. Das letzte Paket, *Extensions*, enthält die Erweiterungen für Thuban, und wird auch das in dieser Diplomarbeit entwickelte Programm aufnehmen.

4.3.1.1 Paket Model

Das Paket *Model* stellt das Grundgerüst von Thuban dar und enthält alle benötigten Klassen und Funktionen für die Verarbeitung der Geodaten. Das Paket *Model* stellt somit die Modellierung einer Karte mit den einzelnen Layern, Klassifikationen, Projektionen usw. dar. Für jedes Objekt, welches für eine Karte benötigt wird, gehört eine Klasse aus dem Paket.[67]

66 vgl. Kap.5: Design, S.53
67 vgl. Anhang A.1, S.88

Die eigentliche Karte wird von der Klasse *Map* repräsentiert. Sie ist die Zusammensetzung aus den verschiedenen Layern und bietet Methoden (*AddLayer*, *RemoveLayer*, *RaiseLayer*, *LowerLayer* usw.) zu deren Verwaltung an. Die Speicherung der Layer erfolgt in einem Feld (*layers*), welches die Zeiger auf die Ebenen-Objekte aufnimmt.

Thuban unterstützt zwei Typen von Layern, den Polygon-Layer (*Layer*) und den Raster-Layer (*RasterLayer*), die beide von der Oberklasse *BaseLayer* abgeleitet sind. Die Oberklasse enhält Methoden, welche für alle Layer benötigt werden, wie z.B. Funktionen um den Sichtbarkeitsstatus zu setzen (*SetVisible*) oder abzufragen (*Visible*). Die Oberklasse hat zudem noch die wichtige Operation, eine Projektion für den Layer zu setzen (*SetProjection*). Der Raster-Layer enthält nur die Methoden, welche für die Verarbeitung eines Rasterbildes notwendig sind, wie den Rasterbildnamen (*filename*) und die Ausmaße (*bbox*) der Karte. Die Ausmaße entsprechen dem kleinsten umschließenden parallelen Rechteck, auch „Bounding Box“ genannt (*BoundingBox, LatLongBoundingBox*). Die Klasse *Layer* hingegen verwaltet die Shapes, die kleinste Einheit einer Karte. Diese Klasse stellt die Methoden bereit, um mit Hilfe der Klasse *shapelib* direkt auf die Shapefiles zugreifen zu können.. Sie ist eine mit swig[68] erzeugte Anbindung der von F. Warmerdam in C geschriebene Bibliothek *Shapelib*.

Die Layer können zudem klassifiziert werden, wofür die Klasse *Classifikation* zuständig ist. Die Klassifizierung gliedert die Shapes in einzelne Gruppen (*ClassGroup*). Es gibt drei Typen von Gruppen: Erstens die Einzelwerte (*ClassGroupSingleton*), zweitens die Bereichswerte (*ClassGroupRange*) und drittens die Defaultwerte (*ClassGroupDefault*). Die Klasse *ClassGroupDefault* ist die Standardklasse und ist immer vorhanden, auch wenn keine explizite Klassifizierung vorgenommen wird. Die Darstellungseigenschaften eines Shapes werden über die Klasse *ClassGroupProperties* definiert, welche neben der Art der Liniendarstellung (Dicke, Type) auch die Farbe der Linien und Flächen über die Klasse *Color* definiert.

Neben den Layern kann auch die Karte eine Projektion enthalten (*self.projection*). Die Variable für die Projektion zeigt entweder auf ein Projektionsobjekt oder, wenn keine Projektion gesetzt ist, auf *Null*.

68 http://www.swig.org

Um die für GIS typische Verknüpfung von Topologie- und Sachdaten zu realisieren, besitzt jeder Layer einen Zeiger auf ein Tableobjekt. Dabei können die Layer Informationen entweder aus einer Datei oder aus einer Datenbank enthalten. Das entsprechende Tableobjekt fungiert als Schnittstelle für die zum *Shapefile* zugehörige DBF-Datei oder die entsprechenden Daten in der Datenbank.

Damit ist der Aufbau einer Karte in den Grundzügen erklärt. Jedoch ist die wohl wichtigste Klasse, die Klasse *Session*, noch nicht erwähnt worden. Sie steht in der Klassenhierarchie über der Klasse *Map* und repräsentiert eine Programmsitzung in Thuban. Die *Session* enthält die Karten und stellt die Methoden für die Handhabung der Session bereit. Dazu zählen neben Methoden zum Hinzufügen von Karten (*AddMap*) oder Datenbankverbindungen (*AddDBConnections*) noch viele weitere, die aber im Rahmen dieser Diplomarbeit keine Rolle spielen.

Ebenso verhält es sich mit den Klassen: neben den hier erläuterten besitzt Thuban noch etliche weitere, welche aber für die in dieser Diplomarbeit einwickelte Extension nicht notwendig sind und den Rahmen der hier zu leistenden Arbeit bei Weitem überschreiten würden. Darum sollen auch sie hier nicht weiter erläutert werden.

4.3.1.2 Paket UI

Das Paket UI ist für die Darstellung und Funktionalität der Benutzeroberfläche (**g**raphical **u**ser **i**nterface, kurz GUI) zuständig, enthält somit vorrangig Dialoge und Frames. Wie schon erwähnt baut Thuban auf wxPython auf[69]. Das Paket hat für die Implementierung der Extension keine so große Bedeutung wie das Paket Model. Da aber einige Funktionen auch in der Extension Verwendung finden und der Vollständigkeit wegen, soll auch dieses Paket genauer analysiert werden.

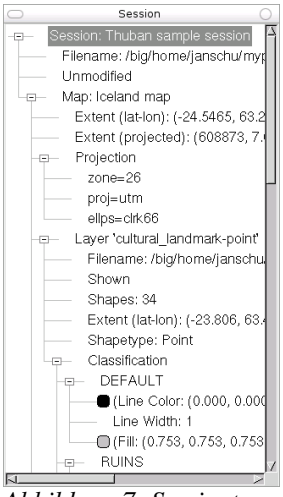


Abbildung 7: Sessiontree

69 vgl. Kap.3.3: Thuban - Interactive Geographic Data Viewer, S.21

Das Hauptfenster (Abb.8) setzt sich aus einer Menüleiste, einer Werkzeugleiste, einer Statusbar, einer Legende und einem Bereich (*Canvas*) für die Karte zusammen. Der Sessiontree (Abb.7, S.34) wird in einem extra Fenster (*Frame*) dargestellt und von der Klasse *LegendTree* abgebildet. Er zeigt die Struktur und den Aufbau von Objekten und deren aktuellen Informationen als Baumstruktur.

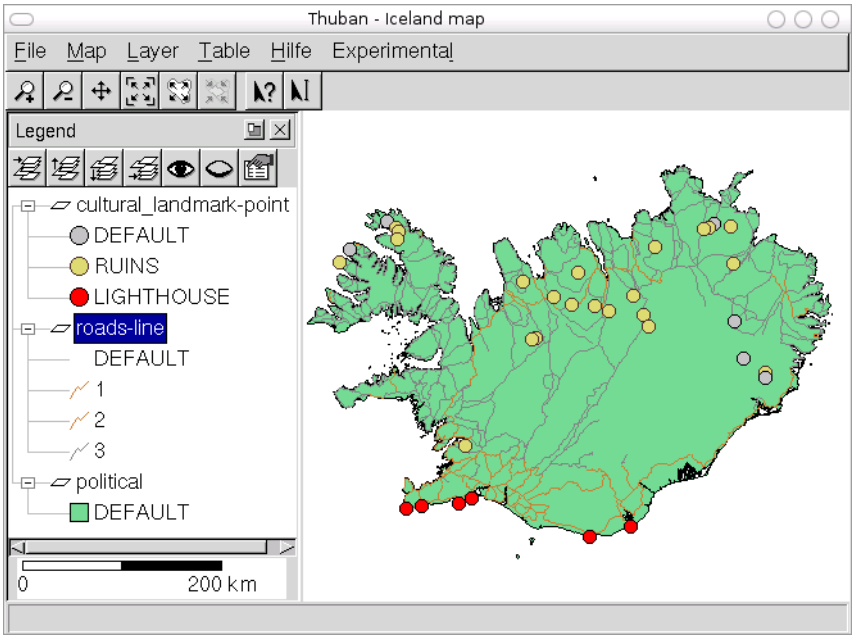


Abbildung 8: Thuban Hauptfenster

Das Hauptfenster von Thuban wird durch die Klasse *MainWindow* repräsentiert und initialisiert beim Starten die Menüleiste, die Statusleiste sowie die Legende und den Kartenbereich. Über diese Benutzerschnittstelle ist es dem Anwender möglich, die Geodaten (Shapefiles) zu bearbeiten bzw. anzuzeigen. Dafür stellt die Klasse Funktionen bereit, um Layern (*AddLayer*, *AddRasterLayer*, *AddDBLayer*) hinzuzufügen oder sie zu bearbeiten.

Das Haupmenü baut auf der Klasse *Menü* auf und bietet Funktionen zur Erzeugung von Menüs an (*FindOrInsertMenu*). Alle Funktionen sind über das Menü erreichbar und können so von dem Anwender ausgeführt werden.

Der Kartenbereich wird von der Klasse MapCanvas repräsentiert und stellt Methoden zur Anzeige der Karte zur Verfügung, so z.B. die Verarbeitung von Befehlen der Maus (*OnLeftDown*) und Änderungen des Fensters (*OnSize*). Zum Anzeigen der Karte in dem Canvas benutzt *mapCanvas* die Methoden der Klasse

ScreenRenderer, welche von der Klasse *MapRenderer* abgeleitet ist. Neben dem *ScreenRenderer* gibt es noch die von dieser Klasse abgeleitete Klasse *ExportRenderer* und die davon wiederum abgeleitete Klasse *PrinterRenderer*, welche beide für die Druckausgabe zuständig sind.

Die Legende repräsentiert die Klasse *LegendPanel*. In ihr sind Methoden für Verarbeitung und Darstellung der Legende enthalten. Dazu zähen unter anderem Methoden für das Verschieben der Layer. Diese Methoden greifen wieder auf die entsprechenden Methoden des Paketes *Model* zurück, um die Daten richtig zu verarbeiten. Wie schon erwähnt, sind alle Klassen im Paket *UI* nur für den visuellen Teil zuständig.

Neben den hier erwähnten Klassen verfügt das Paket *UI* noch über viele weitere Klassen zur Kommunikation mit dem Benutzer. Diese sind jedoch für diese Arbeit nicht von großer Bedeutung und zum Verständnis des Aufbaues und der Funktionsweise der Benutzeroberfläche nicht relevant. Somit soll hier nicht weiter auf sie eingegangen werden.

4.3.1.3 Paket Extensions

Das Paket Extensions nimmt alle Erweiterungen für Thuban auf. Zum besseren Verständnis soll die Funktionsweise am Beispiel der einfachen (in Thuban enthaltenen) Extension „Hello World“ erklärt werden.

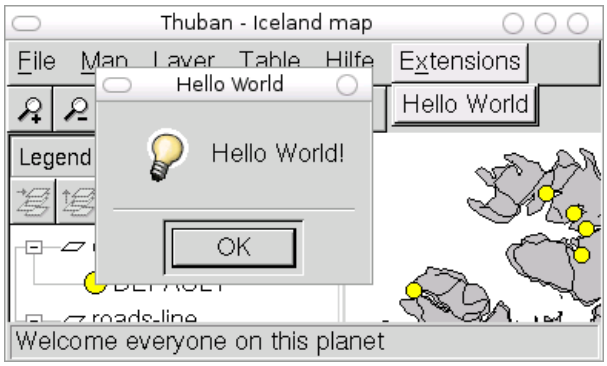


Abbildung 9: Thuban Extension "Hello World"

In Abb.9 ist das Endergebnis zu sehen. Die Extension wird in das Menü *Extensions* eingebunden und kann einfach über den Menüpunkt aufgerufen werden.

Ein einfaches Beispiel dafür ist der folgende Quellcode 2:

```
 1 # Copyright (C) 2003 by Intevation GmbH
 2 # Authors:
 3 # Jan-Oliver Wagner <jan@intevation.de>
 4 #
 5 # This program is free software under the GPL (>=v2)
 6 # Read the file COPYING coming with Thuban for details.
 7
 8 """
 9 Extend Thuban with a sample Hello World to demonstrate simple
10 extensions.
11 """
12
13 __version__ = '$Revision: 1.2 $'
14
15 # use _() already now for all strings that may later be translated
16 from Thuban import _
17 # Thuban has named commands which can be registered in the central
18 # instance registry.
19 from Thuban.UI.command import registry, Command
20
21 # The instance of the main menu of the Thuban application
22 # See Thuban/UI/menu.py for the API of the Menu class
23 from Thuban.UI.mainwindow import main_menu
24
25 def hello_world_dialog(context):
26     """Just raise a simple dialog to greet the world.
27
28     context -- The Thuban context.
29     """
30     context.mainwindow.RunMessageBox(_('Hello World'), \
31                                      _('Hello World!'))
32
33 # find the extensions menu (create it anew if not found)
34 extensions_menu = main_menu.FindOrInsertMenu('extensions', \
35                                              _('E&xtensions'))
36
37 # create a new command and register it
38 registry.Add(Command('hello_world', \
39                      _('Hello World'), \
40                        hello_world_dialog, \
41                        helptext = _('Welcome everyone on this planet')))
42
43 # finally bind the new command with an entry in the extensions menu
44 extensions_menu.InsertItem('hello_world')
```

Quellcode 2: „Hello World“ Beispiel für eine einfache Extension.

Die Zeilen mit einer Raute beginnend sind pythontypische Kommentare, genau so wie die mit drei Anführungzeichen (""", Z.8ff) eingeschlossenen Beschreibungen. Sie sind somit für die Funktionalität irrelevant. Zu Beginn der Datei werden die benötigten Module importiert. In Z.23 z.B. ist es das Modul *main_menu* aus dem Paket *Thuban.UI.mainwindow*. Dieses Modul wird benötigt um in Z.34 mit Hilfe der Methode *FindOrInsertMenu* aus eben jenem Modul ein neues Menü hinzuzufügen. Anschließend wird der Menüpunkt „Hello World“ eingefügt und mit dem zuvor

erzeugten C*ommand* (Z.38) assoziiert. Nun ist es mit diesem Menüpunkt möglich, die Methode *def hello_world_dialog* (Z.25) aufzurufen und somit den Dialog mit der Nachricht „Hello World!“ anzuzeigen (Abb.9, S.36).

Um diese Erweiterungen auch in Thuban verfügbar zu machen, muss das Modul über die spezielle Datei *thubanstart.py*, welche sich in der versteckten Verzeichnis *~/.thuban* befindet, importiert werden. Diese Datei wird beim „Starten“ von Thuban immer ausgeführt und ermöglicht es somit, die Extensions beim „Starten“ von Thuban mitzuladen.

4.3.2 MapServer / Mapfile

Nach Thuban soll nun das zweite Programm, welches für die Bearbeitug verwendet wird, der UMN MapServer, genauer analysiert werden. Dabei soll besonders Wert auf die Struktur und den Aufbau des MapFiles gelegt werden. Das MapFile enthält, bis auf einige Steuerparameter für die Anzeige, alle wichtigen Informationen für die (geo-)graphische Darstellung der Karte und ist damit die Grundlage jedes MapServer Projektes. Hierzu zählt, welche Geodaten und wie diese angezeigt werden, usw. Somit ist eigentlich nur das Mapfile für die Bearbeitung der Aufgabe von Nöten.

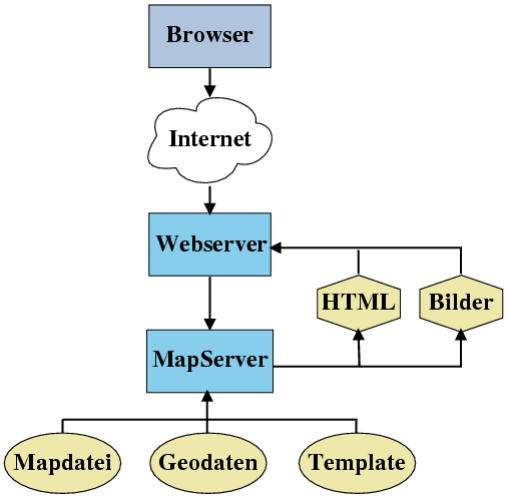


Abbildung 10: Funktionsweise des UMN MapServers[70]

70 Silke Reimer: Webmapping - aber sicher. URL: http://ventilator.netswarm.net/Linuxtag-DVD/talks/173/paper.html. Stand: 12.08.2004

4.3.2.1 Funktionsweise

Es gibt mehrere Formen, wie der UMN MapServer eingesetzt werden kann. In der einfachsten Weise geschieht dies als Common Gateway Interface (CGI). Ein CGI ist eine Schnittstelle zwischen Webservern und Skripten oder Programmen. Diese CGI Programme werden vom Webserver gestartet und erweitern somit dessen Fähigkeiten. Der MapServer ist also ein Programm, welches den Webserver um die Web-Mapping Funktionalität erweitert.

Die Darstellung der Karte im Browser funktioniert dann auf folgende Weise (Abb.10, S.38): Der MapServer liest die Konfigurationsdatei, also das Mapfile, ein. Mit Hilfe der Steuer- oder Konfigurationsparameter, welche aus dem Mapfile gelesen werden, generiert der UMN MapServer aus den Geodaten die fertige Anwendung. Die fertig generierte Anwendung kann neben der eigentlichen Karte mit den Layern auch noch eine Legende, einen Maßstab oder eine Referenzkarte enthalten und wird meist als Grafikdatei (Abb.11) erstellt. Die Grafik, die in vielen verschiedenen Formaten erstellt werden kann, unter anderem PNG oder JPEG, wird mittels eines HTML-Templates, in welchem die Pfade zu der Datei vordefiniert sein müssen, im Browser angezeigt.

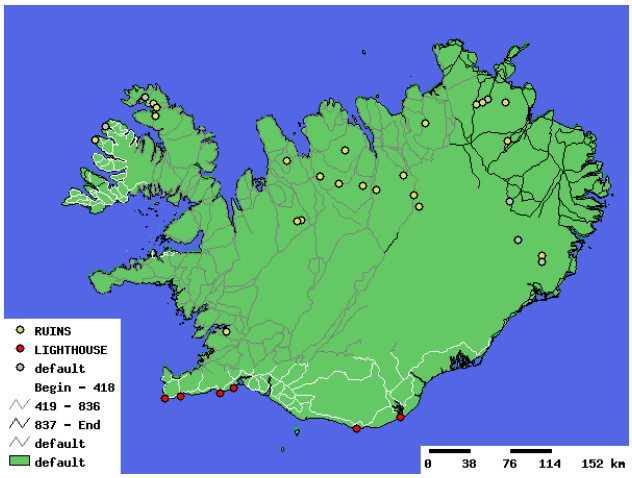


Abbildung 11: Iceland sample

Neben der hier erklärten Funktionsweise als CGI Version ist es möglich, den UMN MapServer auch über Skriptsprachen anzusprechen.[71] Eine dieser Skriptsprachen ist Python. Der Zugriff erfolgt über das (Python) MapScript, welches für die Realisierung dieser Diplomarbeit Verwendung findet.

71 vgl. Kap.3.4: UMN MapServer, S.23

4.3.2.2 Aufbau des Mapfiles

Wie in dem Beispiel Quellcode 3 zu erkennen, ist das Mapfile deutlich und übersichtlich strukturiert, was auf den objektorientierten Aufbau zurückgeht. Das heißt, dass die einzelnen Komponenten einer Karte, wie etwa „PROJECTION“ oder „LAYER“ zu Objekten zusammengefasst sind. So sind die einzelnen Komponenten klar begrenzt und einfach zu erzeugen.[72]

```
MAP
   NAME ISLAND
   STATUS ON
   SIZE 600 450
   EXTENT 622877.17 7019306.94 1095667.78 7447709.31
   SHAPEPATH "data"
   IMAGECOLOR 255 255 255
   IMAGETYPE PNG

   PROJECTION
      proj=utm
      ellps=clrk66
      zone=26
      north
   END

   WEB
      TEMPLATE iceland.html
      IMAGEPATH "/tmp/"
      IMAGEURL "/tmp/"
      METADATA
         titel "Iceland Test"
      END
   END

   # Beginn der Layer Definitionen #
   LAYER
   NAME "political"
      DATA political
      TYPE POLYGON
      STATUS ON
      CLASS
         COLOR 200 200 200
      END
      PROJECTION
         proj=latlong
         ellps=clrk66
      END
   END
END
```

Quellcode 3: Beispiel für den Aufbau eines Mapfiles

72 vgl. Anhang A.2, S.89

Die Definition der Einstellungen geschieht meist über Schlüsselpaare (wie z.B. NAME ISLAND), wobei der erste Teil der vom MapServer vordefinierte Schlüssel ist und der zweite die dazugehörige Definitionsangabe. Vereinzelt können die Schlüsselwörter aber auch mehr als einen Parameter benötigen, wie etwa bei EXTENT oder bei den Farbdefinitionen (COLOR).

Neben den Schlüsselwerten, ist es auch möglich, in einem Mapfile bestimmte Zeilen als Kommentare zu definieren (Z.26). Beginnen müssen Kommentare immer mit einer Raute (#).

4.4 Existierende Lösungsansätze

In dieser Diplomarbeit werden vier im Netz zur Verfügung stehende Lösungsansätze benutzt, die im Folgenden näher vorgestellt werden sollen. Die Analyse dieser Programme von anderen Entwicklern zur Erstellung von MapServer-Projekten soll die Lösungsmöglichkeiten und mögliche Probleme bei der gegebenen Aufgabenstellung aufzeigen. Drei der vier hier angeschauten Lösungen sind Extensions für das ESRI Produkt ArcView (bis Version 3.x). Getestet wurden sie mit der ArcView Version 3.1. Die Extensions sind alle frei erhältlich, nicht so aber ArcView. ArcView ist ein proprietäres Produkt und steht somit nicht ohne damit verbundene Kosten zur Verfügung. Somit ist auch die Verwendung der Extensions für ArcView zur Erstellung von MapServer-Anwendungen nicht immer möglich. Sollte kein ArcView zur Verfügung stehen, bleibt dem Anwender nur die Nutzung der Freien Software QuantumGIS (QGIS) übrig, welches ein komplett eigenständiges Produkt ist.

4.4.1 AveiN!

Hersteller	terrestris
Version (getestet)	1.33 von 2003
Betriebssystem	Anforderungen von ArcView 3.x (MS Windows, Solaris, ...)
Lizenz	GNU GPL Lizenz
Bezugs Adresse	http://www.avein.de/

Diese ArcView GIS Erweiterung ist die umfangreichste von den hier getesteten und bietet weitreichende Konfigurationsmöglichkeiten (Abb.12). AveiN! erstellt aus den Karten in ArcView die erforderlichen Dateien, um diese mit Hilfe des UMN MapServers im Intra-/Internet zu veröffentlichen. Dabei übernimmt AveiN! alle Vektor- und Rasterbilder inklusive der Kassifizierungen.

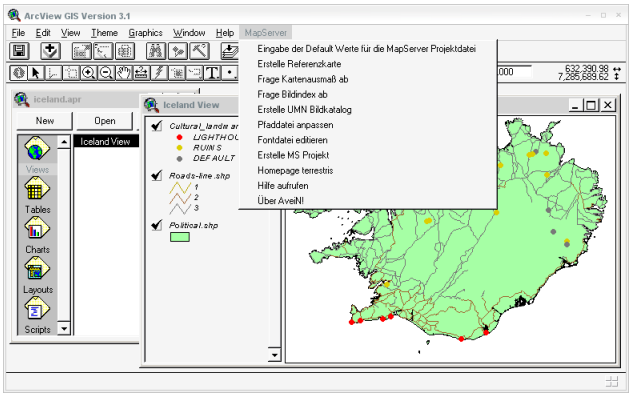


Abbildung 12: ArcView mit aktivierter Erweiterung AveiN!

AveiN! kann neben der erforderlichen „.map“-Datei für den MapServer auch einen „Client“ erstellen.

4.4.1.1 Installation und Bedienung

Avein! wird mit einer Setup-Routine ausgeliefert, und somit ist die Installation sehr einfach. Das Setup geht davon aus, dass ArcView im Standardverzeichnis installiert ist. Sollte dies nicht der Fall sein, so muss die Extension manuell in das dafür vorgesehene Verzeichnis kopiert werden (ARCVIEW\EXT32).

Die Erstellung eines Projektes ist einfach, wenn man die Funktionalitäten des UMN MapServers in den Grundzügen kennt. Ansonsten könnte die enorme Anzahl der Einstellungen und Möglichkeiten eine Überforderung darstellen.

4.4.1.2 *Funktionalitäten*

MS Projekt erstellen

Die eigentliche Aufgabe von AveiN! ist die Erstellung eines Mapfiles. Für diese Aufgabe wurde eine Art Assistent geschaffen, der durch den Prozess der Erstellung führt. Mit Hilfe dieses Assistent ist es möglich, fast alles einzustellen, was in einem Mapfile eingestellt werden kann.

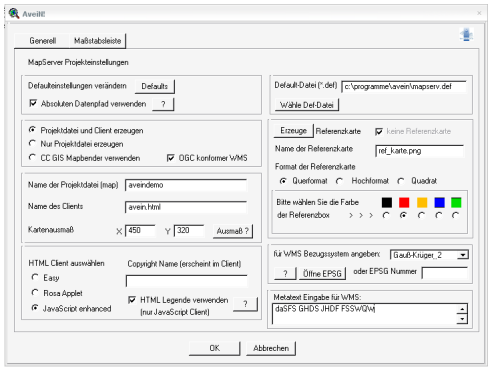


Abbildung 13: AveiN! Haupteinstellungen

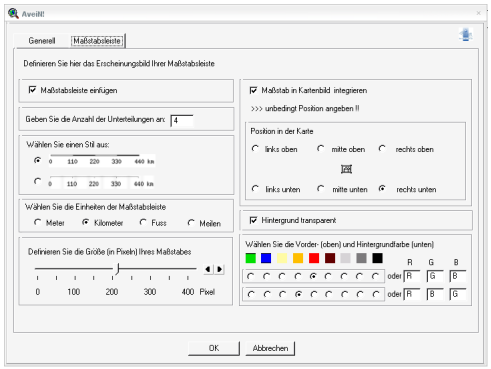


Abbildung 14: AveiN! Maßstabseinstellungen

Auf der ersten Seite können die Hauptsettings ausgewählt werden. Dazu zählt neben den Angaben für die Größe des Bildes, den Namen des Mapfiles und die Einstellungen für die Referenzkarte, auch die Wahl, ob nur ein Mapfile, oder auch ein Client erstellt werden soll (Abb.13). Im zweiten Fenster (Abb.14) können die Einstellungen der Maßstabsleiste gesetzt werden (Größe, Stil, Position, Hintergrundfarbe usw.).

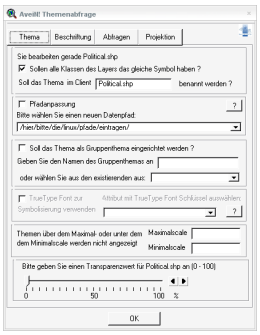


Abbildung 15: AveiN! Themenoptionen

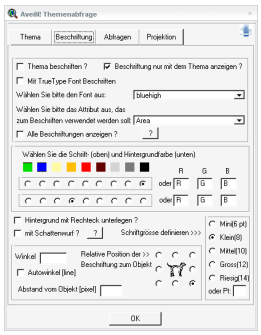


Abbildung 16: AveiN! Beschriftungsoptionen

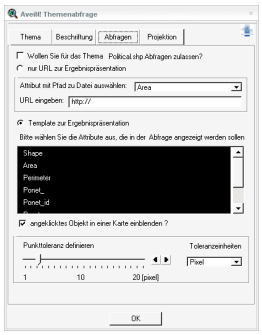


Abbildung 17: AveiN! Abfrageoptionen

Abbildung 18: AveiN! Projektionsoptionen

Nachdem die Grundeinstellungen vorgenommen wurden, können für jeden Layer umfangreiche Angaben zum Thema (Abb.15), zur Beschriftung (Abb.16), für Abfragen (Abb.17) und zur Projektion (Abb.18) getätigt werden. Handelt es sich um

einen Linienlayer, so wird anschließend die Möglichkeit gegeben, die Art der Linie, ein Symbol oder einfach nur die Farbe festzulegen (Abb.19), bei einem Punktlayer dementsprechend die Wahl eines Symbols (Abb.20).

Der Assistent ist bei der Erstellung eines Projektes hilfreich und erleichtert die Bearbeitung, jedoch besteht zwischen den einzelnen Schritten nicht die Möglichkeit Schritte zurück zu gehen, um Einstellungen zu korrigieren.

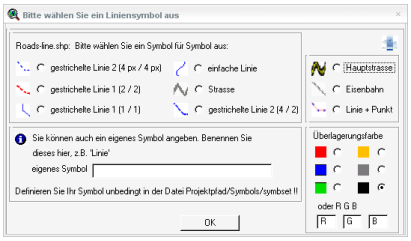


Abbildung 19: AveiN! Linienoptionen

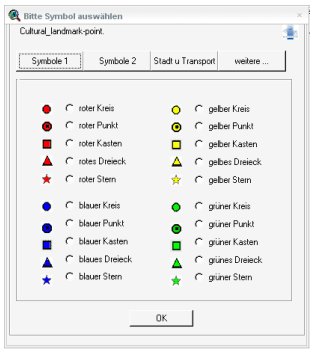


Abbildung 20: AveiN! Punktoptionen

Referenzkarte erstellen

AveiN! bietet die Möglichkeit eine Referenzmap zu erstellen. Die Karte wird als . jpeg Bild erzeugt und hat das Aussehen wie in ArcView. Die Darstellung lässt sich somit sehr genau bestimmen.

4.4.1.3 Handhabung/Benutzerfreundlichkeit

Alle Funktionen sind gut zu erreichen und können nachträglich angepasst werden. Die Grundeinstellungen können in einer Datei für spätere Projekte gespeichert werden. Der einzige wirkliche Kritikpunkt ist, dass bei der Erstellung eines Projektes nicht zwischen den einzelnen Schritten zurückgesprungen werden kann, um anschließend alles zu speichern. So müssen jedes Mal wieder alle Schritte durchlaufen werden, um einen Fehler zu korrigieren oder eine Einstellung zu verändern.

4.4.2 MapServer AV Connect

Hersteller	Armin Burger
Version (getestet)	0.8 vom 16.03.2002
Betriebssystem	Anforderungen von ArcView 3.x (MS Windows)
Lizenz	Copyright bei Developer
Bezugs Adresse	http://digilander.libero.it/arminburger/avmapserv.html

Diese Extension ist nach Angaben des Entwicklers Armin Burger nur für MS Windows verfügbar, obwohl ArcView auch für andere Systeme erhältlich ist. MapServer AV Connect ist eine Art Import Extension für ArcView. Sie ermöglicht es, die Daten von einem MapServer im Internet oder Intranet als „Image Theme“ in ArcView anzuzeigen. Dabei ist es möglich, die einzelnen Layer als „Theme“ zu laden.

4.4.2.1 Installation und Bedienung

Für die Installation dieser Extension müssen die Dateien in den vorgegebenen Ordner kopiert werden.

- AV_MapServer.avx ins Extension Verzeichnis „ARCVIEW\EXT32“.
- „AVMSCon.dll“, „avmsdimg.jpg“, „GFLibDLL.dll“ ins „ARCVIEW\BIN32“ Verzeichnis.

Zum Aktivieren wird die Extension über das entsprechenden Menüpunkt geladen. War die Installation erfolgreich, gibt es fünf neue Buttons (Abb.21, oranger Rahmen) und ein neues Menü für die Nutzung der Extension.

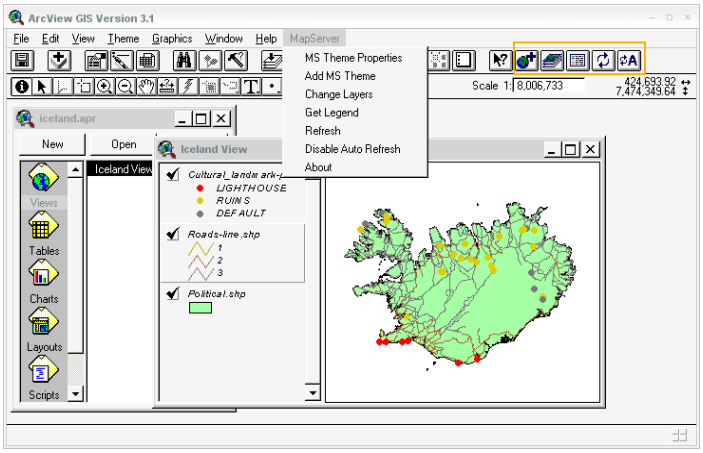


Abbildung 21: ArcView mit aktivierter Erweiterung AV Connect

4.4.2.2 *Funktionalitäten*

Anzeigen einer MapServer Karte

Um einen MapServer Layer zu erzeugen wird ein Server benötigt, von welchem alle wichtigen Daten bekannt sein müssen (Abb.22).

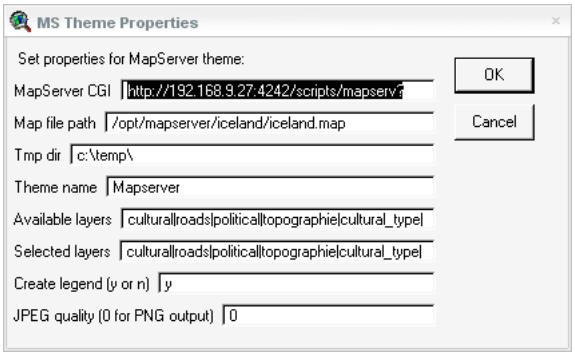


Abbildung 22: AV Connect MapServer Theme Properties

Die Einstellungen können in einer Datei gesichert werden, nachdem die Erstellung des Layer erfolgreich erfolgreich war. Der Layer wird durch die Extension nur als Grafik in ArcView angezeigt, genau wie es auch in einem Browser geschieht.

Anzeigen der MapServer Legende

Mit der Extension ist es möglich die Legende des MapServers anzuzeigen (Abb.23). Aber auch dabei wird, wie bei den anderen Funktionen, nur die Grafik vom MapServer „geholt“.

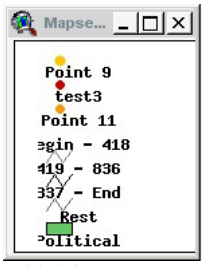


Abbildung 23: AV Connect Legende

Auswahl der anzuzeigenden Layer

Wie bei einer MapServer Anwendung ist es auch möglich die anzuzeigenden Layer auszuwählen (Abb.24). Dabei wird allerdings jedes Mal eine neue Karte vom MapServer generiert und anschließend als Bild angezeigt.

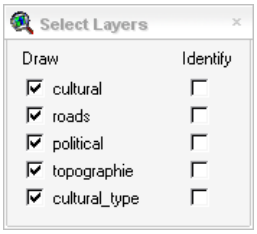


Abbildung 24: AV Connect Layerauswahl

4.4.2.3 *Handhabung/Benutzerfreundlichkeit*

Die Extension ist einfach zu benutzen, bietet aber auch nicht allzu viele Möglichkeiten. So werden alle Informationen als Grafiken angezeigt und bieten keine geographischen Informationen. Auf die zugrunde liegenden Geodaten kann somit nicht zugegriffen werden. Datenanalysen jeglicher Art sind dadurch unmöglich.

4.4.3 MapServer ArcView Utility

Hersteller	Ross Searle
Version (getestet)	0.5 vom Februar 2000
Betriebssystem	Anforderungen von ArcView 3.x (z.B. MS Windows, Solaris)
Lizenz	keine Angaben
Bezugs Adresse	http://mapserver.gis.umn.edu/contributed.html

Diese Extension von Ross Searle dient zur Erstellung von UMN MapServer Projekten aus ArcView 3.x. Damit ist es möglich, ein komplettes „MapServer Projekt“ zu erstellen oder aber nur das Mapfile zu erzeugen. Neben diesen Aufgaben besteht noch die Möglichkeit eine Referenzkarte zu generieren.

4.4.3.1 Installation und Bedienung

Die Installation erfolgt mit wenigen Schritten. Nach dem Herunterladen und Entpacken des „.zip“ Archivs, wird der Archivinhalt in ein Verzeichnis mit dem Namen „mapserver“ im „ArcView/Ext32“ Verzeichnis kopiert. Anschließend muss das mitgelieferte ArcView-Projekt geöffnet werden, um mit der Extension arbeiten zu können. Die MapServer-Funktionen sind über einen neuen Menüpunkt (Abb.47), der beim Öffnen des ArcView Projektes angelegt wird, verfügbar. Es kann wie gewohnt mit ArcView gearbeitet werden, lediglich die Bearbeitung eines vorhandenen Projektes erfordert eine andere Vorgehensweise. Das Projekt darf nicht wie üblich geöffnet, sondern muss importiert werden, da ansonsten die Funktionen für den MapServer nicht zur Verfügung stehen. Beim Speichern eines Projektes, in dem die MapServer Extension integriert ist, wird diese mit gespeichert.

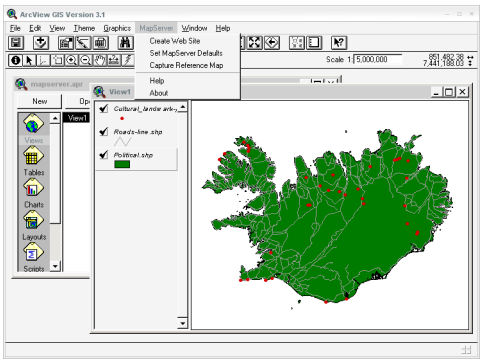


Abbildung 25: ArcView mit aktivierter Erweiterung ArcViewUtility

4.4.3.2 Funktionalitäten

Erstellen einer Website

Bei Anwendung des „MapServer ArcView Utility“ zur Erstellung einer Website ist zuerst die Entscheidung zu treffen, ob eine komplette Website erstellt werden soll oder nur ein einzelnes Mapfile. Ein Mapfile könnte z.B. dazu dienen, in eine schon vorhandene Website eingebunden zu werden.

Für die Herstellung werden einige Abfragen bezüglich der Pfade getätigt. Diese können bei der Erstellung oder bei den Voreinstellungen eingegeben werden. Die HTML Dateien für die Website und die benötigten Grafiken liegen in dem angelegten Ordner „mapserver“ vor und werden nur in das bei der Herstellung angegebene Verzeichnis kopiert. Sind die Dateien nicht in dem vom Programm vordefinierten Ordner, bricht das Programm mit einer Fehlermeldung ab.

Die Herstellung an sich funktioniert ohne Probleme, aber das MapServer Projekt konnte nicht ordnungsgemäß angezeigt werden. Die Schwierigkeit entsteht dadurch, dass das Programm so konzipiert ist, dass das erzeugte Projekt mit einem MapServer unter Linux und nicht unter Windows nutzbar sein soll. Dieses dürfte eigentlich kein allzu großes Problem darstellen. Leider war es auf diese Weise nicht möglich, ein Projekt zu erzeugen, welches anschließend auch ohne Probleme lauffähig war. Um das Projekt zu verwenden, müssen die Pfade manuell angepasst werden.

Das Mapfile wird auf die gleiche Weise wie die komplette Anwendung erstellt, allerdings ohne die HTML Seiten zu erzeugen bzw. zu kopieren. Zur weiteren Verwendung muss das Mapfile weiterverarbeitet und noch die nötigen Dateien für die Anzeige im Browser erstellt werden.

Erstellung einer Referenzkarte

Die Erstellung einer Referenzmap für den UMN MapServer, die für die Übersichtskarte benötigt wird, funktioniert bei dem MapServer ArcView Utility sehr gut und ist ein nettes Feature, welches oft hilfreich ist. Die Karte wird so erzeugt wie sie auch in ArcView angezeigt wird (Abb.25, S.47). Das Erscheinungsbild ist also sehr genau bestimmbar.

MapServer Voreinstellungen

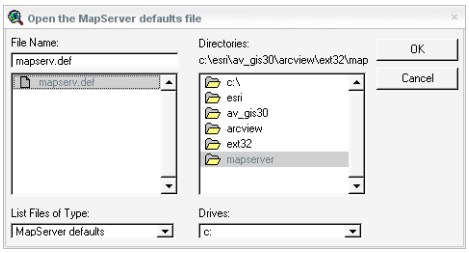


Abbildung 26: AVU Voreinstellungen

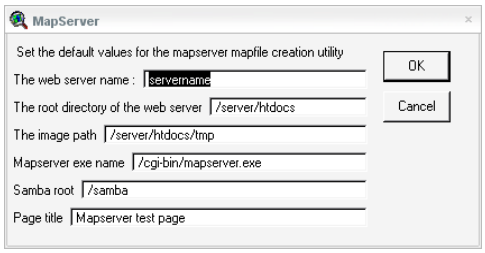


Abbildung 27: AVU Voreinstellungen

Um eine UMN MapServer Anwendung zu erstellen, werden einige Angaben benötigt, die nicht automatisch ermittelt werden können. Diese Einstellungen können aus einer Datei geladen werden (Abb.26), die manuell gewählt werden muss. Die Änderungen werden automatisch in dieser Datei gespeichert. Dies ist etwas störend, wenn die Einstellungen öfters zu editieren sind, hat aber den Vorteil, dass verschiedene Einstellungen gespeichert werden können. Die Angaben (Abb.27) sind:

- Name des Servers,
- Verzeichnis zum Webserver,
- Pfad, in dem die Bilder gespeichert werden,
- Pfad, in dem die Mapserver.exe Datei zu finden ist,
- Verzeichnis zum Samba Server,
- Titel der Seite.

Diese Angaben sind für den MapServer notwendig, damit dieser auch alle Dateien für die Verarbeitung findet.

4.4.3.3 Handhabung/Benutzerfreundlichkeit

Die Bedienung ist sehr einfach gehalten und die Anwendung kann ohne Einarbeitungszeit erfolgen, soweit man sich etwas mit der Materie auskennt. Alle Punkte sind aussagekräftig bezeichnet, und jeder Schritt wird erläutert und mit einer kleinen Beschreibung versehen. Teilweise ist aber nicht genau zu erkennen, welche Pfadangaben richtig sind, damit die MapServer Anwendung hinterher auch funktioniert.

4.4.4 QGIS – Quantum GIS

Hersteller	QGIS Team
Version (getestet)	0.2 vom April 2004
Betriebssystem	Linux / Unix
Lizenz	GNU General Public License (GPL)
Bezugs Adresse	http://qgis.sourceforge.net/

Bei QGIS handelt es sich um ein ähnliches Produkt wie Thuban, bei dem die Möglichkeit UMN MapServer Mapfiles zu erstellen bereits integriert ist. Bei der zum Testzeitpunkt zur Verfügung stehenden Version 0.2 konnten allerdings nur die Mapfiles erzeugt werden. Templates oder gar eine ganze MapServer-Anwendung werden nicht generiert. Inzwischen ist die Version 0.4 vom 02.07.2004 erhältlich, allerdings hat sich an der MapServer Funktionalität nichts geändert. Das einzige wirkliche Manko von QGIS, was die Nutzung bislang eher unmöglich macht, ist die fehlende Unterstützung von Projektionen.

4.4.4.1 Installation und Bedienung

Der Installationsprozess funktioniert unter Debian mit Hilfe des Paketmanagements wie erwartet ohne Probleme. Die Installation unter anderen Unix Systemen könnte sich unter Umständen als etwas schwieriger erweisen, da die Software auf der offiziellen Seite nur als SourceCode zu Verfügung steht und somit selber kompiliert werden muss.

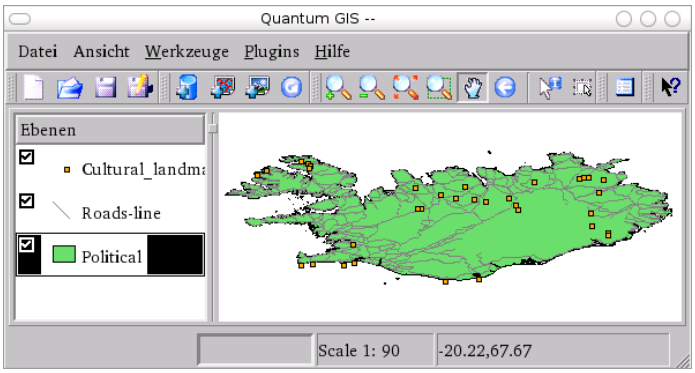


Abbildung 28: QGIS mit dem Beispieldatensatz „Iceland“

Nach der Installation kann das Mapfile erstellt werden, ohne dass extra Einstellungen nötig sind. Die Funktion ist direkt integriert und über das Datei-Menü direkt ansprechbar.

4.4.4.2 Funktionalitäten

Die Möglichkeiten beim Exportieren der aktuellen Ansicht (Abb.28, S.50) sind nicht sehr umfangreich. Hier werden nur die minimal benötigten Daten abgefragt, wie z.B. Name, Höhe und Breite (Abb.29).

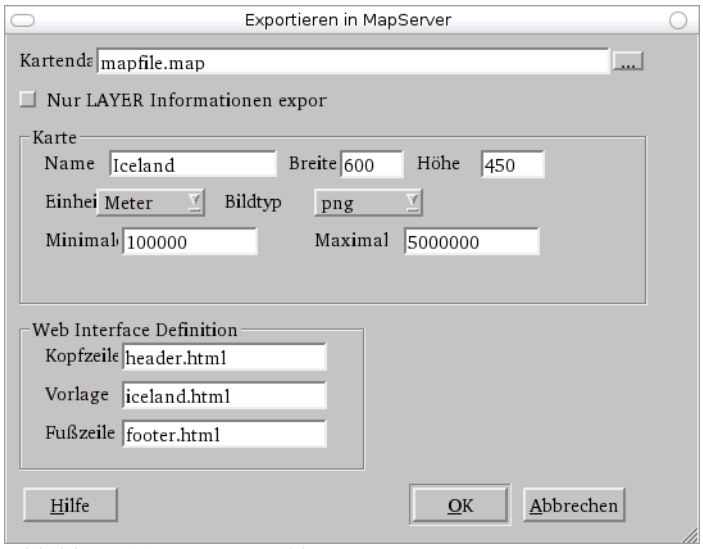


Abbildung 29: QGIS Mapfile Exportoptionen

QGIS bietet zwei Methoden bei der Erstellung eines Mapfiles an:

- Die erste Methode erstellt ein „komplettes“ Mapfile. Dieses ist jedoch sehr unvollständig, da nur einige grundlegende Map-Einstellungen, die Web-Definition und die Layer-Settings geschrieben werden. Diese Einstellungen sind nicht sehr umfangreich und umfassen nur die nötigsten Grundeinstellungen.
- Die zweite Möglichkeit exportiert nur die Layer, so dass weitere Einstellungen per Hand getätigt werden müssen. Sie ist somit nicht wirklich nützlich. Ein möglicher Vorteil dieser Methode wäre allerdings, dass hiermit ein neuer Layer erzeugt werden kann, der in ein schon vorhandenes Projekt integriert werden könnte.

4.4.4.3 Handhabung/Benutzerfreundlichkeit

Die Handhabung der Exportfunktion ist sehr intuitiv und kann ohne große Einarbeitungszeit, genau wie QGIS auch, sofort genutzt werden. Allerdings ist das erstellte Mapfile nicht ohne Weiteres zu gebrauchen, weil es, wie schon erwähnt, nur grundlegende Informationen enthält. Ist eine umfangreichere Anwendung zu erstellen, muss doch wieder das Mapfile manuell editiert werden.

Das Map-Interface, also das HTML-Template für die eigentliche Darstellung im Browser, muss bei QGIS anderweitig erstellt werden.

Das erzeugte Mapfile jedoch ist sauber strukturiert und mit vielen Kommentaren versehen, so dass es dem Anwender sehr erleichtert wird, dieses für die weitere Verwendung zu verarbeiten und zu erweitern.

4.4.5 Verwendbarkeit

Die Lösung „MapServer AV Connect“ ist gar nicht zu gebrauchen, da sie so gut wie keine Aufgaben anbot, welche in dem Projekt realisiert werden sollten

Die zwei Produkte „MapServer ArcView Utility“ und „QGIS“ konnten ebenfalls nicht viel zum Projekt beitragen, lieferten aber einen Eindruck möglicher Realisierungen.

Die vierte und umfangreichste Lösung „AveiN!“ stellt gute Lösungsansätze bereit, zeigte aber auch die Komplexität der Möglichkeiten, welche vom UMN MapServer unterstützt werden, auf.

Anhand drei der vier Extensions konnten vor allem die Fähigkeiten des UMN MapServers besser bewertet und die Planung der benötigten Funktionalität besser abgeschätzt werden.

Fazit: Hilfreich für die weitere Arbeit erscheint die Analyse der existierenden Lösungen nur bedingt. Dafür sind doch die Grundlagen, auf denen die Lösungen basieren, zu verschieden, weswegen auch die eigentlich viel interessantere Analyse der technischen Umsetzung nicht vorgenommen wurde.

5. Design

Plant das Schwierige da, wo es noch leicht ist.
Tut das Große da, wo es noch klein ist.
Alles Schwere auf Erden beginnt stets als Leichtes.
Alles Große auf Erden beginnt stets als Kleines.[73]

Nachdem die Anwendungsfälle und die erforderlichen Kriterien definiert wurden und ein Überblick über die technische Umsetzung von Thuban sowie über den Aufbau des Mapfiles besteht, soll nun ein Konzept für die anschließende Implementierung entwickelt werden.

5.1 Grundkonzept

Die grundlegende Idee wurde zwar schon mehrfach erwähnt, soll hier aber noch einmal kurz beschrieben werden. Um das Potenzial und die Funktionen von Thuban voll auszunutzen, sollen alle Angaben aus dem Mapfile in Thuban editierbar sein. Soweit diese Angaben auch in Thuban abbildbar sind, werden sie verwendet, ansonsten soll die Extension die Möglichkeit zur Bearbeitung dieser Angaben selbst zur Verfügung stellen.

In Abb.30 (S.54) sind die Funktionsweise und das Zusammenspiel der Extension mit Thuban zu erkennen. Die Extension wird in Thuban integriert und über die

73 Laotse (3. od. 4. Jh.v.Chr.), historisch nicht faßbarer chin. Philosoph

graphische Oberfläche vom Benutzer angesteuert. Dabei übernimmt sie die Verarbeitung der Map-Objekte sowie die Interaktion mit Thuban.

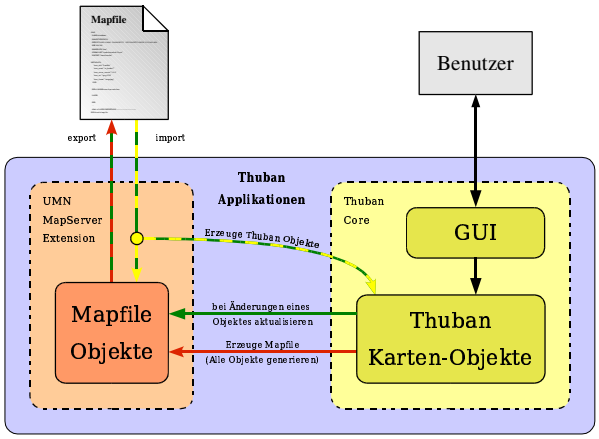


Abbildung 30: Funktionsprinzip der Extension

Hierbei wurde versucht, möglichst eine Trennung zwischen Datenverwaltung und Benutzeroberfläche zu realisieren, wie es auch in Thuban der Fall ist.

5.2 Einschränkungen

Um die Planung des Projektes nicht zu umfangreich zu gestalten, sollte die Extension auf dem Python MapScript aufbauen. Für dessen Nutzung sprechen mehrere Gründe. Zum einen erfüllt das Python MapScript, bis auf die Erhaltung von Struktur und Kommentaren, die erforderlichen Anforderungen für das Projekt und erspart die Implementation eines eigenen Parsers, wodurch der Entwicklungsprozess beschleunigt wird, zum anderen wird die Fehlerrate beim Implementieren durch die Verwendung vorhandener Lösungen gesenkt.

Dieses erwies sich nach Analyse des Mapfiles schwieriger als zuerst angenommen. Struktur und Reihenfolge der einzelnen Wertepaare und Objekte im Mapfile sind nicht exakt vorgegeben und erfordern tiefer gehende, nicht vorhandene Kenntnisse im Bereich der Parserprogrammierung. Die Nutzung des MapScriptes erleichtert somit vor allem die Verarbeitung des Mapfiles beim Ein- und Auslesen. Bei der Implementierung der Extension sollte aber darauf geachtet werden, sie so flexibel umzusetzen, dass zu einem späteren Zeitpunkt das MapScript durch einen eigenen Parser ersetzbar sein wird.

Des Weiteren sollten vorerst nur die grundlegenden Angaben eines Mapfiles und die dafür benötigten Methoden implementiert werden. Um diese zu bestimmen, wurde mit Hilfe des Python MapScripts ein leeres Mapfile inklusive eines leeren Layers erzeugt wie Quellcode 4 zeigt.

```
MAP
  EXTENT -1 -1 -1 -1
  IMAGECOLOR 255 255 255
  IMAGETYPE png
  SIZE -1 -1
  STATUS ON
  UNITS METERS
  NAME "MS"

  OUTPUTFORMAT
    NAME png
    MIMETYPE image/png
    DRIVER GD/PNG
    EXTENSION png
    IMAGEMODE PC256
    TRANSPARENT FALSE
  END

  LEGEND
    IMAGECOLOR 255 255 255
    KEYSIZE 20 10
    KEYSPACING 5 5
    LABEL
      SIZE MEDIUM
      TYPE BITMAP
      BUFFER 0
      COLOR 0 0 0
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      PARTIALS TRUE
      POSITION CC
    END
    POSITION LL
    STATUS OFF
  END

  QUERYMAP
    COLOR 255 255 0
    SIZE -1 -1
    STATUS OFF
    STYLE HILITE
  END

  SCALEBAR
    COLOR 0 0 0
    IMAGECOLOR 255 255 255
    INTERVALS 4
    LABEL
      SIZE MEDIUM
      TYPE BITMAP
      BUFFER 0
      COLOR 0 0 0
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      PARTIALS TRUE
    END
    POSITION LL
    SIZE 200 3
    STATUS OFF
    STYLE 0
    UNITS MILES
  END

  WEB
    IMAGEPATH ""
    IMAGEURL ""
    QUERYFORMAT text/html
  END

  LAYER
    NAME "(null)"
    SIZEUNITS PIXELS
    STATUS OFF
    TOLERANCE 0
    TOLERANCEUNITS PIXELS
    UNITS METERS
  END
END
```

Quellcode 4: Aufbau eines mit MapScript erzeugten leeren Mapfiles.

In diesem Quellcode ist wieder deutlich der objektorientierte Aufbau des Mapfiles zu erkennen.[74] Alle Parameter, welche hier zu sehen sind, sollten bei der

74 vgl. Kap.4.3.2.2: Aufbau des Mapfiles, S.40

Implementierung beachtet werden oder sind auf sinnvolle Standardwerte zu setzen.

Zu beachten ist, dass sich Thuban noch in der Entwicklung befindet und als Geodatenbetrachter entwickelt wurde.[75] Es kann somit noch nicht alle Möglichkeiten eines „echten“ GIS anbieten. Die Angaben des Mapfiles, welche von Thuban (noch) nicht unterstützt werden, sollen bei der Planung aber trotzdem berücksichtigt werden. Sollte Thuban nämlich diesbezüglich erweitert werden, könnten die Angaben ohne größeren Aufwand nachträglich implementiert werden.

5.3 Aufbau

Zur Gliederung und übersichtlicheren Gestaltung können und sollen hier nun die drei beschriebenen Anwendungsfälle[76] einzeln betrachtet werden. Daraus ergibt sich die Aufteilung in drei Module, für jeden Anwendungsfall eines. Hinzu kommt noch ein weiteres Modul, welches die Struktur des Mapfiles wiederspiegelt, wodurch die Trennung von Datenmanagement und Funktionalität gegeben ist. Die insgesamt vier erforderlichen Module werden im Folgenden näher betrachtet.

5.3.1 Modul *mapfile*

Das Modul *mapfile* bildet die Grundlage für die Extension und enthält alle Klassen und Methoden, welche für die Verarbeitung des Mapfiles bzw. der darin enthaltenen Objekte von Bedeutung sind. Die Struktur richtet sich nach dem Aufbau des Mapfiles[77] und ist dadurch ziemlich genau vorgegeben.

Die oberste Ebene, das Mapobjekt (*MF_Map*) repräsentiert das gesamten Mapfile mit allen Objekten und Eigenschaften. Auf Grund der Tatsache, dass nicht alle Objekteigenschaften auch in Thuban abgebildet werden können, muss das Mapobjekt mit der Karte in Thuban verknüpft werden. So ist es möglich auch auf die Einstellungen zuzugreifen, für die kein äquivalentes Objekt in Thuban existiert.

75 vgl. Kap.3.3: Thuban - Interactive Geographic Data Viewer, S.21
76 vgl. Kap.4.2: Anwendungsfälle, S.30
77 vgl. Anhang A.2, S.89 und Kap.4.3.2.2: Aufbau des Mapfiles, S.40

Das Mapobjekt enthält, wie der Abb.31 zu entnehmen ist, mehrere untergeordnete Objekte. Das wichtigste ist wohl das Layerobjekt, welches eine Schicht einer Karte repräsentiert. Für jede Schicht existiert, genau wie in Thuban, ein Layerobjekt. Jedes Layerobjekt wiederum enthält eine Klassifizierung *(MF_Class)*, deren Darstellungsweise über ein Styleobjekt (*MF_Style*) definiert wird. Die meisten Einstellungen des Layerobjektes, einschließlich des Classobjektes und des Styleobjektes können auch in Thuban abgebildet werden. Neben dem Namen, dem Status und der Farbe zählt auch die Projektion, welche über das Projektionsobjekt definiert wird, dazu. Wie das Mapobjekt müssen auch die Layerobjekte mit den entsprechenden Layerobjekten von Thuban verknüpft werden, um auf die nicht abbildbaren Einstellungen zugreifen zu können.

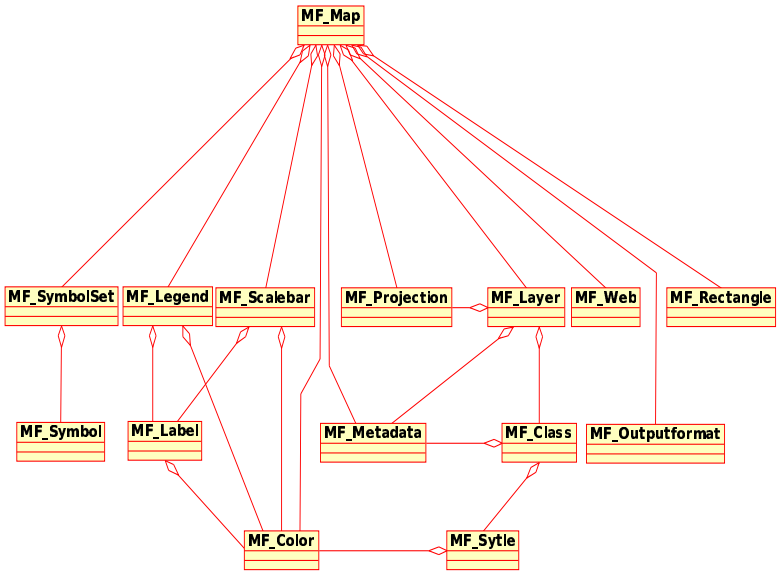


Abbildung 31: Klassendiagramm des Modules mapfile

Anders sieht es bei den Metadaten aus. Thuban unterstützt keine Metadaten. Metadaten enthalten Informationen, welche für die Darstellung der Geodaten keine Bedeutung haben und beim MapServer dafür genutzt werden eigene Informationen in einem Template anzuzeigen. Die einzige Ausnahme dabei sind die WMS Daten. WMS Daten werden im Mapfile als Metadaten angegeben, werden vom MapServer aber verarbeitet. Mit Hilfe der WMS Daten ist es möglich Informationen (Layer usw.) von anderen Servern zu holen und für die Kartendarstellung zu nutzen. Weil Thuban diese Möglichkeit (noch) nicht unterstützt, sollen die WMS Daten vorerst,

genau wie die anderen Metadaten, als einfache Texte behandelt werden. An der WMS Unterstützung wird aber momentan gearbeitet, und so könnte dieser Punkt bereits in naher Zukunft überarbeitet werden.

Die anderen Objekte des Mapfiles, die Legende (*MF_Legend*), der Maßstab (*MF_Scalebar*), das Ausgabeformat (*MF_Outputformat*), die Webeinstellungen (*MF_Web*) und die Symbole (*MF_SymbolSet, MF_Symbol*) sind in Thuban nicht existent und müssen direkt über die Extension veränderbar sein. Einzige Ausnahme dabei stellen die Symbole dar. Thuban unterstützt bisher nur Kreissymbole, soll aber (ebenfalls) zu einem späteren Zeitpunkt (noch) durch die Möglichkeit, andere Symboltypen zu nutzen, erweitert werden. Daher ist die Implementierung einer Editiermöglichkeit für Symbole vorerst nicht nötig.

5.3.2 Modul *mf_import*

Für das Modul *mf_import* bedarf es keiner größeren Vorplanung. Der Ablauf ist sequenziell und deswegen relativ einfach zu implementieren. Trotzdem soll der grobe Ablauf hier einmal aufgezeigt werden.

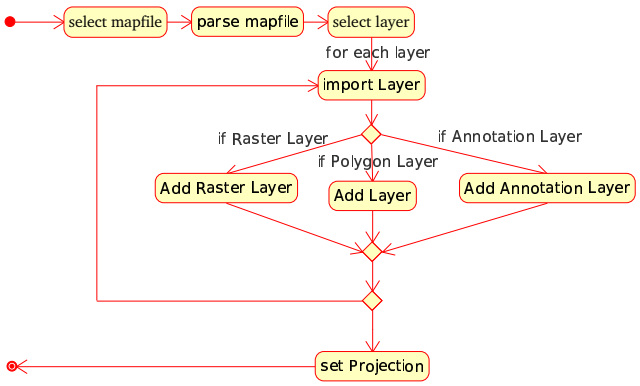


Abbildung 32: Ablaufdiagramm des Modules mf_import.py

Wie in Abb.32 verdeutlicht, wählt der Anwender zuerst das zu importierende Mapfile aus. Dieses wird mit Hilfe des MapScriptes geparst, um mit Hilfe des Moduls *mapfile* die entsprechenden Objekte mit den dazugehörigen Parametern zu erstellen.

Durch das Parsen sind auch die vorhandenen Layer bekannt, aus denen die zu importierenden Layer gewählt werden. Für jeden zu importierenden Layer wird überprüft, um welche Art Layer es sich handelt, um dementsprechend die richtigen Einstellungen in Thuban zu setzten.

Zur Verarbeitung der Daten greift *mf_import* auf die Methoden des Modules *mapfile* bzw. auf vorhandene Methoden aus Thuban zurück.

5.3.3 Modul *mf_export*

Das Modul *mf_export* kehrt den Vorgang des Importierens quasi um. Zuerst muss die Datei zum Speichern gewählt werden. Anschließend müssen die Daten aus Thuban ermittelt werden, um sie in das entsprechende Mapfile-Objekt zu schreiben.

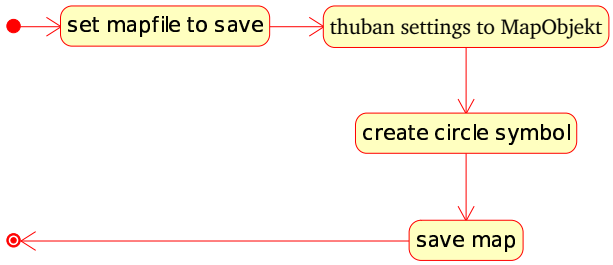


Abbildung 33: Ablaufdiagramm des Modules mf_export.py

Zum Vereinfachen dieses Vorganges verfügt das Objekt jeweils selber über eine Methode, um die Daten aus dem entsprechenden Thuban-Objekt zu ermitteln. Darum ist dieser Vorgang nicht in diesem Paket notwendig, sondern geschieht in dem Modul *mapfile*. Abb.33 verdeutlicht, dass das Modul *mf_export* dazu die entsprechenden Methoden aufruft und die entsprechenden Thuban-Objekte übergibt. Alles Weitere erledigt das Modul *mapfile*.

Wie schon im Modul *mapfile* erwähnt, kann Thuban keine anderen Symbole als Kreise verwalten. Weil weitergehende Fähigkeiten aber später noch in Thuban implementiert werden sollen, wird vorerst ein Symbol generiert, welches ähnlich dem von Thuban ist. Dieses Symbol ist durch den Anwender nicht veränderbar.

Zum Speichern des Mapfiles wird die entsprechende Methode (*save*) des Python MapScriptes verwendet.

5.3.4 Modul *mf_handle*

Dieses Modul stellt die Methoden und die grafischen Dialoge zur Verarbeitung der nicht von Thuban unterstützen Angaben aus dem Mapfile bereit und zwar für die Objekte *Map*, *Layer*, *Label*, *Legend*, *Scalebar*, *Web* und *Metadata* erzeugt werden. Die Dialoge werden mit Hilfe von wxPython erstellt. Sie enthalten, je nachdem welche Angaben für einen Wert im Mapfile zulässig sind, entsprechende Eingabemöglichkeiten.

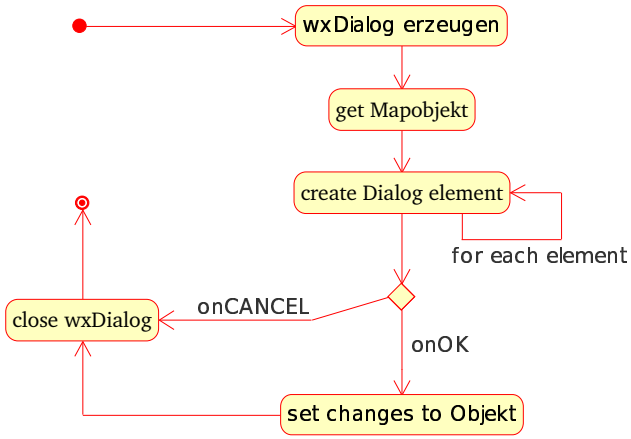


Abbildung 34: Ablaufdiagramm des Modules mf_handle (für einen Dialog)

Der Ablauf eines Dialoges ist in Abb.34 visualisiert. Zu Beginn wird ein Dialog auf Basis des wxPython Objektes *wxDialog* erzeugt. Anschließend wird das benötigte Objekt ausgelesen und die entsprechenden Dialogelemente mit den schon im Objekt gesetzten Einstellungen erstellt.

Wird der Dialog über einen *OKButton* beendet, werden die Einstellungen aus den Dialogelementen in das Objekt übertragen. Bei Benutzung eines *CancelButton* werden die Einstellungen verworfen und der Dialog, ohne Änderungen an dem Objekt vorzunehmen, geschlossen.

6. Realisierung

Nimm dir Zeit zum Nachdenken,
aber wenn die Zeit zum Handeln kommt,
hör auf mit Denken und geh los.[78]

Nachdem das Konzept für die Implementation entwickelt worden war, soll nun die eigentliche Umsetzung des Projektes erläutert werden. Hierbei werden neben der Vorgehensweise vor allem Schwierigkeiten, die sich bei der Realisierung ergeben haben, angesprochen sowie deren Lösungen beschrieben.

Die Extension wurde direkt in die Hauptentwicklung (CVS HEAD) von Thuban integriert. Dazu wurde die aktuelle CVS-Version von Thuban mittels „*cvs checkout thuban*" aus dem CVS-Repository extrahiert. Somit ist die Extension auch in der CVS Version von Thuban enthalten. Des Weiteren sind die Quelltexte auf einer CD der Diplomarbeit beigefügt.[79]

Die Implementation erfolgte auf einem Debian GNU/Linux System. Verwendet wurde die Debian-testing-version „Sarge".

6.1 Vorbereitung

Zu Beginn mussten einige Vorbereitungen erfolgen, bevor mit dem eigentlichen Implementieren begonnen werden konnte. Dazu zählt die Einrichtung eines UMN MapServers sowie die grundlegende Integration der Extension in Thuban.

78 Andrew Jackson (1767-1845), amerik. Politiker, 7. Präs. d. USA (1829-37)
79 vgl. Anhang E: CD-ROM, S.100

6.1.1 Installation des UMN MapServers

Als erster Schritt wurde ein UMN MapServer auf dem genannten Linux System installiert. Dafür wurden ein HTTP-Server sowie die CGI Variante des UMN MapServers benötigt. Als Grundlage dafür wurde ein fertiges Debian-Paket von der FreeGIS CD verwendet, und somit war ein MapServer schnell installiert. Das Paket enthält neben dem UMN MapServer auch einen Webserver. Als Webserver wird das Programm minihttpd[80] verwendet, ein kleiner, einfacher, aber für die Zwecke der Diplomarbeit vollkommen ausreichender HTTP-Server. Der UMN MapServer musste anschließend noch durch eine neuere Version ersetzt werden, weil die integrierte Version 3.6.5 zu alt erschien. Die neue 4er Reihe des MapServers war zwar zu Beginn der Arbeit (März 04) gerade erst erschienen, aber auch die Entwicklung dieser Extension brauchte längere Zeit und sollte bei Erscheinen (Sept. 04) möglichst eine aktuelle Version des UMN MapServers unterstützen. Das Aktualisieren auf die neuere Version war einfach und konnte relativ rasch erfolgen.

Um auch wirklich die aktuellste Version zu verwenden (zum Zeitpunkt der Arbeit war dies Version 4.2), wurde der Quellcode von der offiziellen Homepage heruntergeladen, mit Hilfe der enthaltenen Installationsanleitung entpackt, konfiguriert (*./configure*) und kompiliert (*make*). Dies verlief bis auf einen falschen Suchpfad für die gdal[81] Bibliothek ohne Schwierigkeiten. (Die gdal Bibliothek wird für die Unterstützung von Rasterlayern benötigt.)

Nachdem der Pfad richtig angepasst war, funktionierte auch das Kompilieren ohne Fehler. Nun musste nur noch die alte UMN MapServer Datei (*mapserv*) durch die neu erzeugte ersetzt werden, um die Aktualisierung des MapServers und damit die Installation eines Testsystems abzuschließen.

Zur Überprüfung der erfolgreichen Installation wurde der Webserver über den Befehl */opt/mapserver/mini_httpd/mini_httpd -p 4242 -c "/scripts/*" -i / tmp/mini_httpd_pid* gestartet. Normalerweise enthält der installierte Server ein Script (*mapserverdemo)* zum Starten und Beenden des MapServers. Dieses funktionierte aber nicht korrekt, weil der vorhandene Pfad angeblich nicht existierte.

80 http://www.acme.com/software/mini_httpd/
81 http://www.remotesensing.org/gdal

Nach dem Start des Servers kann über den Browser mit der URL *localhost:4242* die mit installierte Beispiel-Applikation aufgerufen werden. *Localhost* ist dabei der Server, in diesem Fall der „lokale“ auf dem Arbeitsrechner, welcher über den Port *4242* kommuniziert.

6.1.2 Installation des MapScripts

Um überhaupt mit der eigentlichen Implementierung beginnen zu können, musste zuvor das Python MapScript installiert werden.

Das Python MapScript ist in dem Quellcode-Paket, im Ordner *mapscript/python/,* enthalten*,* welches zuvor für die Aktualisierung des UMN MapServer heruntergeladen wurde. Für die Installation des Python MapScriptes ist zuvor eine Kompilierung des nötig. Dies erfolgte schon für die Einrichtung des MapServers und brauchte damit nicht noch einmal durchgeführt werden.[82]

Die Installation des MapScriptes wurde durch die beiden Befehle *python setup.py build* und *python setup.py install* ausgeführt*.* Zur Überprüfung, ob die Installation auch erfolgreich war und das MapScript einwandfrei funktionierte, wurden abschließend die enthaltenen Tests gestartet. Dazu mussten einige Pfad in den Testdateien, welche sich im Unterordner *tests* des Python Mapscriptes befinden, angepasst werden. Anschließend konnten die Tests mit *python testMapScript.py* ausgeführt werden.

6.1.3 Integration in Thuban

Als weiterer Schritt muss die Extension in Thuban integriert werden. Wie schon erwähnt verfügt Thuban über das Paket *Extensions,* welches die Erweiterungen für Thuban aufnimmt. In diesem Verzeichnis wurde ein neues Paket für die zu erstellende UMN MapServer Extension erzeugt (*umn_mapserver*)*.* Alle Module, die zu dieser Extension gehören, werden in diesem einen Paket zusammengefasst. Hierbei ist besondere Sorgfalt bei der Wahl der Paket- und Modulnamen sowie bei der Festlegung der Struktur an den Tag zu legen, damit bei der späteren Verwendung

82 vgl. Kap.6.1.1: Installation des UMN MapServers, S.62

von CVS keine Probleme auftreten. Grund hierfür ist die in CVS nicht vorhandenen Möglichkeit Dateien im Repository umzubenennen.[83]

In Abb.35 ist die Struktur der Extension *umn_mapserver* mit allen enthaltenen Dateien visuell abgebildet. Die Extension enthält neben den Modulen *mapfile.py, mf_import.py, mf_export.py und mf_export.py* noch zwei weitere Unterordner, einen für die Tests (*test*) und einen weiteren für eine UMN MapServer Beispiel Anwendung (*sample*).

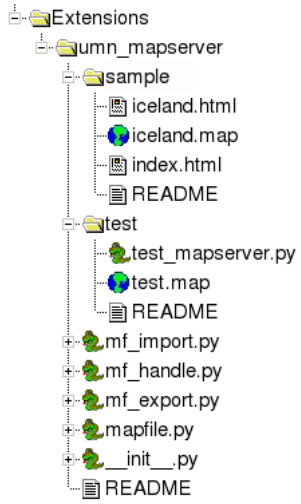


Abbildung 35: Aufbau der Extension

Der Ordner *test* enthält neben dem Python-Modul für die Tests (*test_mapserver*) noch ein einfaches Mapfile (*test.map*). Das Mapfile enthält die Parameter, welche für die Tests benötigt werden und sollte nicht editiert werden, um die korrekte Funktionsfähigkeit der Tests weiterhin zu gewährleisten.

Der Ordner *sample* enthält das Island Beispiel (*iceland*) von Thuban als UMN MapServer Anwendung (*index.html, iceland.html*) inklusive des Mapfiles (*iceland.map*). Es dient zum Testen der Darstellungsweise und der Parameter des MapServers sowie für die spätere Überprüfung der Funktionsweise der Extension.

Neben diesen Dateien befindet sich noch eine Datei *__init__.py* im Paket *umn_mapserver*. Diese Datei hat die Funktion Python mitzuteilen, dass das Verzeichnis als Paket zu behandeln ist. Im einfachsten Fall kann *__init__.py* eine leere Datei sein, aber sie kann auch einen Code für das Paket enthalten, welcher beim Initialisieren des Paketes ausgeführt wird. Damit wäre es z.B. möglich, globale Variablen für das Paket zu initialisieren.

Jeder Ordner enthält außerdem eine Datei (*README*) mit Hinweisen und Anweisungen für die Benutzung der Extension bzw. der Tests oder des Beispiels.

83 vgl. Kap.3.6: CVS - Concurrent Versions System, S.25

6.2 Implementierung

Nachdem die vorbereitenden Tätigkeiten erledigt waren, konnte mit der eigentlichen Implementierung der Extension begonnen werden. Wie schon beschrieben wurde hierbei versucht eine weitestgehende Trennung zwischen dem Datenmanagement und der grafischen Oberfläche zu realisieren.[84]

Die Implementation der Klassen erfolgte nicht komplett auf einmal, sondern wurde nach und nach während der gesamten Umsetzung erweitert. Zuerst wurden nur die Methoden, die für den Importvorgang nötig waren, implementiert, anschließend folgten die für den Exportvorgang. Schlussendlich kamen noch jene hinzu, welche für die Editierfunktionalität von Nöten sind. So wurde das Modul *mapfile* vor jeder Phase der Implementierung um die für das jeweilige Modul *mf_import, mf_export und mf_handle* benötigten Methoden und Klassen erweitert.

6.2.1 Datenmanagement

Das Modul *mapfile* bildet die Grundlage für den im vorherigen Kapitel geplanten Aufbau der Extension und wird zuerst implementiert. Dazu wurde von einer bereits existierenden Extension die vorgegebenen Grundstruktur übernommen.[85]

Ein wichtiger Punkt bei diesem Modul, welches die Datengrundlage bildet, ist die Entwicklung einer Testumgebung für die implementierten Klassen, Methoden und Attribute. Mit Hilfe der Tests kann sichergestellt werden, ob die Implementierung keine größeren Fehler enthält, bevor mit der Umsetzung der anderen Module begonnen wird. Zudem kann bei späteren Erweiterungen der Extension schnell überprüft werden, ob die Funktionsfähigkeit der Methoden noch gegeben ist.

6.2.1.1 Modul mapfile

Die Struktur des Modules wird in weiten Teilen vom MapScript, welches sich sehr stark an die Struktur des Mapfiles[86] anlehnt übernommen. Lediglich zwei Klassen, *MF_Color* und *MF_Metadata* wurden neu hinzugefügt, da sie im MapScript nicht existierten, für die Umsetzung aber hilfreich zu sein schienen. Die erste Klasse

84 vgl. Kap.4: Analyse, S.28
85 vgl. Kap.4.3.1.3: Paket Extensions, S.36
86 vgl. Anhang A.2: Aufbau des Mapfiles, S.89

MF_Color wurde hinzugefügt, um die Farben gleich als *Thubancolor v*orliegen zu haben und um diese somit nicht jedes mal einzeln transformieren zu müssen. Die zweite Klasse, *MF_Metadata* ist für das einfachere Handling der Metadaten verantwortlich. Die Metadatan liegen im MapScript auch als Objekt vor, der Zugriff aber kann nur direkt aus einer Klasse (z.B. *MapObj*), über die Methoden *getMetaData, getFirstMetaDataKey*, *getNextMetaDataKey und removeMetaData* erfolgen. Um nicht in jeder Klasse, welche Metadaten unterstützt diese Methoden einzeln zu implementieren, wurde diese in der Klasse *MF_Metadata* zusammengefasst.

Im Aufbau soll am Beispiel einer Klasse des Moduls näher erläutert werden. Als Beispiel möge hierfür die Klasse *MF_Style* dienen (Quellcode 5, S.67).

Der Konstruktor (*def __init__, Z.10*) jeder Klasse erwartet als Übergabeparameter die entsprechende Klasse des MapScriptes. So erwartet *MF_Map* das Mapobjekt des MapScriptes und die Klasse *MF_Layer* ein MapScript Layerobjekt. Die Klasse *MF_Style* (Beispiel) erwartet dementsprechend ein Styleobjekt (*mf_style, Z.10*). Die Konstruktoren „wissen" welche Objekte und Parameter in der Klasse enthalten sein können, ermitteln diese aus dem MapScript Objekt und erzeugen die Äquivalenten für die Extension. Ein gutes Beispiel hierfür ist die Z.21 (*self._color = MF_Color (self._style.color)*). Hier wird ein neues *MF_Color O*bjekt erzeugt und an das entsprechende Farbobjekt aus dem Styleobjekt übergeben. Der Konstruktor der Klasse *MF_Color* erzeugt aus diesem Objekt wiederum die entsprechenden Module und Klassen, falls vorhanden. Zudem wird das übergebene MapScript Objekt für die weitere Verwendung durch die Methoden gespeichert (*self._style = mf_style*).

Neben dem Konstruktor enthalten die Klassen Methoden zum Auslesen (*get_xxxxx, z.B. Z.47)* und zum Setzen (*set_xxxxx, z.B. Z.50*) der Parameter. Meist greifen diese Methoden direkt auf das MapScript zurück. So z.B. *self._style.size (Z.48, 51).* Der Parameter *size* wird hierbei direkt aus dem MapScript Styleobjekt *_style* ausgelesen. *(*Hier wäre im Nachhinein ein Initialisieren aller Werte aus dem Mapobjekt im Konstruktor und damit die Vermeidung der direkten Verwendung des MapScript Objektes in den Methoden, in Bezug auf die spätere Implementation eines eigenen Parsers sicher sinnvoller gewesen.)

```
class MF_Style:
    """The following parameters, which the mapscript style obj
    contains, are used:
    color, backgroundcolor, outlinecolor, size, symbolname

    The following are not used:
    symbol, sizescaled, minsize, maxsize, offsetx, offsety, antialias
    """

    def __init__(self, mf_style):
        """
        Create a style object from the given mapscript style object. The color
        Object from color and outlinecolor parameter will be created.
        If one color (red, green or blue) is -1, there is no definition in the
        mapfile and no color object. The color will set to 'None'.
        """
        self._style = mf_style
        if self._style.color.red == -1:
            self._color = None
        else:
            self._color = MF_Color(self._style.color)
        if self._style.outlinecolor.red == -1:
            self._outlinecolor = None
        else:
            self._outlinecolor = MF_Color(self._style.outlinecolor)

    def get_color(self):
        return self._color

    def set_color(self, tb_color):
        self._color = tb_color
        new_color = MF_Color(colorObj())
        new_color.set_thubancolor(tb_color)
        self._color = new_color
        self._style.color = new_color.get_mfcolor()

    def get_outlinecolor(self):
        return self._outlinecolor

    def set_linecolor(self, tb_color):
        self._color = tb_color
        new_linecolor = MF_Color(colorObj())
        new_linecolor.set_thubancolor(tb_color)
        self._outlinecolor = new_linecolor
        self._style.outlinecolor = new_linecolor.get_mfcolor()

    def get_size(self):
        return self._style.size

    def set_size(self, newsize):
        self._style.size = newsize

    def set_symbolname(self, newsymbol):
        # its possible to use stringnames instead of numbers
        self._style.symbolname = newsymbol
```

Quellcode 5: Klasse MF_Style aus dem Modul mapfile.py

Alle weiteren Klassen sind analog zu der in diesem Kapitel beispielhaft behandelten Klasse *MF_Style* aufgebaut. Einige Klassen enthalten zusätzlich Methoden für das Hinzufügen eines Objektes aus Thuban.

```
def add_thubanstyle(self, tb_style, type="default"):
        """
        added a thuban style object to the mapobject
        """
        new_styleobj = MF_Style(styleObj(self._clazz))
        if type == "line":
            new_styleobj.set_color(tb_style.GetLineColor())
        elif type == "point":
            # set a default symbol to show circles not only a small dot
            # symbol "circle" must create before
            # first the default symbol circle will be created and the size 8
            new_styleobj.set_symbolname('circle')
            new_styleobj.set_size(8)
            if tb_style.GetLineColor() != Transparent:
                new_styleobj.set_linecolor(tb_style.GetLineColor())
            new_styleobj.set_color(tb_style.GetFill())
        else:
            new_styleobj.set_size(tb_style.GetLineWidth())
            if tb_style.GetLineColor() != Transparent:
                new_styleobj.set_linecolor(tb_style.GetLineColor())
            new_styleobj.set_color(tb_style.GetFill())
```

Quellcode 6: Die Methode add_tubanstyle der Klasse MF_Class.

MF_Map z.B. verfügt über die Methode *add_thubanlayer* um einen Layer aus Thuban dem MapScript Kartenobjekt hinzuzufügen. Dabei wird das Layer-Objekt aus Thuban übergeben, und die Methode erzeugt einen neuen *MF_Layer* mit den entsprechenden Objekten (*MF_Class*) und den entsprechenden Attributen. *MF_Layer* wiederum verfügt über eine Methode um Klassen hinzuzufügen (*add_thubanclass*), und *MF_Class* verfügt dementsprechend wieder über eine Methode um Styles hinzuzufügen (*add_thubanstyle, Quellcode 6*).

6.2.1.2 Modultests

Während der Implementation der Klassen im Modul *mapfile* wurden zu den meisten Methoden, vor allem den umfangreicheren, welche nicht nur auf das Python MapScript zugreifen (wie *get_size* oder *set_size)*, Tests geschrieben. Die Test werden vor dem eigentlichen Implementieren der Methode erstellt (Extreme Programming(XP)). Dadurch wird die Entwicklung durch die Tests geleitet, da nur das implementiert werden muss, was von dem Test verlangt wird. Zudem wird vermieden dass sich Fehler häufen. Diese frühe „Qualitätssicherung“ soll zudem die

Kosten und den Aufwand im weiteren Projektverlauf reduzieren.

```
 1 # Import the testmodul from python
 2 import unittest
 3
 4 # Import the mapscript mapObject
 5 from mapscript import mapObj
 6
 7 # Import  necessary classes from Thuban
 8 from Thuban.Model.color import Color, Transparent
 9
10 # Import the Classes to test
11 from Extensions.umn_mapserver.mapfile import MF_Style,
12
13 class mapserver_Classes(unittest.TestCase):
14     def setUp(self):
15         """
16         Running this funktion befor each test
17         """
18         # using the sample map
19         testMapfile = 'test.map'
20         self.testMap = mapObj(testMapfile)
21         self.eq = self.assertEquals
22
23     def test_MF_Style(self):
24         """
25         Tests a style object with all parameters.
26         """
27         teststyle = MF_Style(self.testMap.getLayer(0).getClass(0).getStyle(0))
28
29         # COLOR 100 200 100
30         self.eq(teststyle.get_color().get_red(), 100)
31         self.eq(teststyle.get_color().get_green(), 200)
32         self.eq(teststyle.get_color().get_blue(), 100)
33
34         # SIZE 2
35         self.eq(teststyle.get_size(), 2)
36
37 if __name__ == '__main__':
38     unittest.main()
```

Quellcode 7: Test zum Modul mapfile.py.

Python bietet für die Testumgebung das spezielle Modul *unittest*[87] an. Dieses Modul stellt ein großes Spektrum an Werkzeugen zum Erzeugen und Ausführen von Tests bereit. Als Beispiel soll hier der Test zur Klasse *MF_Style* herangezogen und vorgestellt werden (Quellcode 7).

In diesem Test erfolgt z.B. die Überprüfung auf Gleichheit zweier Werte durch den Befehl *assertEquals* (Z.21). In Z.30 etwa wird überprüft, ob der Farbwert für Rot, den die Methode *get_red* zurückgibt, auch wirklich wie gewünscht 100 beträgt. Neben *assertEquals* gibt es noch weitere Methoden für die Überprüfung von Werten, welche auf der entsprechenden Internetseite zu finden sind.

87 http://docs.python.org/lib/module-unittest.html

Damit die Testumgebung auch in dem neuen Modul verfügbar ist, muss, wie in Python üblich, mit Hilfe des Befehls *import unittest* das Testmodul verfügbar gemacht werden. Die Klassen für die Tests werden dann vom Modul *unittest* abgeleitet.

Die einzelnen Tests werden als separate Methoden der Klasse implementiert. In dem Beispiel (Quellcode 7, S.69) ist nur eine Testklasse enthalten, nämlich die Methode *test_MF_Style*(*Z.23ff*).

Die zweite in der Klasse enthaltene Methode *setUp* (*Z.14*) ist eine von *unittest* vorgegebene Methode. Sie wird vor jedem einzelnen Test ausgeführt und dient dazu, die Werte für die Tests jedes Mal wieder neu zu initialisieren. So wird vor jedem Test ein neues MapObjekt angelegt, um eventuelle Veränderungen durch vorherige Tests, welche zu verfälschten Ergebnissen führen könnten, rückgängig zu machen.

Wiesen die implementierten Methoden im Modul *mapfile* Fehler auf, wurden sie überarbeitet bis sie fehlerfrei waren.

Alle im Rahmen dieser Diplomarbeit entstandenen Tests sind im Verzeichnis *test* in dem Extension Verzeichnis *umn_mapserver* abgelegt.[88]

6.2.2 Importfähigkeit

Nachdem die Grundlagen mit dem Modul *mapfile* geschaffen waren, wurde das Modul *mf_import* realisiert. Bei diesem Modul mussten alle Werte aus dem Mapfile, soweit dies möglich war, direkt in Thuban abgebildet werden. Dazu zählen unter anderem der Kartenname, die Projektion sowie die einzelnen Layer inklusive Klassifizierung.

Dieser Vorgang ist im Grunde relativ einfach umzusetzen. Zuallererst wird ein Dialog für die Auswahl des Mapfiles angezeigt. Dafür stellt wxPython fertige Methoden zur Verfügung (*wxFileDialog, Quellcode 8, S.71*).

88 vgl. Kap.6.1.3: Integration in Thuban, S.63

```
    # open a dialog to select the mapfile to import
    dlg = wxFileDialog(context.mainwindow,
                       _("Select MapFile file"), ".", "",
                       _("UMN MapServer Mapfiles (*.map)|*.map|") +
                       _("All Files (*.*)|*.*"),
                       wxOPEN|wxOVERWRITE_PROMPT)
    if dlg.ShowModal() == wxID_OK:
        filename = dlg.GetPath()
        dlg.Destroy()
    else:
        return
```

Quellcode 8: Von wxPython bereitgestellte Methode wxFileDialog

Nach Wahl des Mapfiles wird dieses mit dem MapScript geparst (Quellcode 9, *Z.12*) und das erzeugte MapScript Mapobjekt (*mapObj, Z.12*) an das Mapobjekt aus dem Modul *mapfile* übergeben (*MF_Map(mapObj), Z.14*).

```
def parse_mapfile(filename):
    """
    Parse the mapfile.

    Currently this is done using the mapscript module.
    NOTE: It is also possible to use here an own parser
    which could at aleast gain independency from the mapscript
    module.

    filename - the filename of the .map file
    """
    theMap = mapObj(filename)
    mapobj = MF_Map(theMap)
    return mapobj
```

Quellcode 9: Methode parse_mapfile des Modules mf_import

Wie oben beschrieben erzeugen die jeweiligen Klassen im Modul *mapfile* die entsprechenden Unterklassen und die dazugehörigen Module.[89] Mit Hilfe dieser zwei Methoden werden im Module *mf_import* somit alle benötigten Klassen erzeugt und die Datengrundlage für die weitere Verarbeitung geschaffen.

Um nun die MapServer Objekte in entsprechende Thuban-Objekte „umzuwandeln“ werden diese aus dem Modul *mapfile* mit Hilfe der entsprechenden Methoden (*get_xxxxx*) ausgelesen und über die von Thuban bereitgestellten Methoden erzeugt bzw gesetzt.

Zuvor muss die „Karte“ aus Thuban ermittelt werden (Quellcode 10).

```
# Get the map displayed by the mainwindow
tb_map = context.mainwindow.canvas.Map()
```

Quellcode 10: Ermitteln der Karte von Thuban.

89 vgl. Kap.6.2.1.1: Modul mapfile, S.65

Dabei wird die Karte (*Map*), welche im Hauptfenster von Thuban (*mainwindow*) angezeigt wird, ermittelt und in einer Variablen (*tb_map*) für die spätere Verwendung gespeichert.

Zu diesem Zweck werden alle Einstellungen importiert, welche für die komplette Karte gelten, wie z.B. der Name und die Kartenprojektion. Die Funktionsweise, in der dieses geschieht, soll an dem einfachsten Beispiel (Quellcode 11) für den Kartennamen kurz erläutert werden.

```
1 # Set the titel to the Thuban-map
2 tb_map.SetTitle(mapobj.get_name())
```

Quellcode 11: Titel der Karte in Thuban setzen.

Mit der Methode *get_name()* wird der Name aus dem MapServer Objekt *mapobj* ermittelt und mit Hilfe der Methode *SetTitel,* welche von Thuban bereitgestellt wird, in der Karte *tb_map* gesetzt. Die meisten dieser Vorgäng bestehen, ähnlich den obigen (Quellcode 11), nur aus einer Zeile. Allerdings besteht bei vielen Werteangaben das Problem, dass sich die Werte im Mapfile von denen in Thuban unterscheiden. Diese Differenzen werden vorwiegend im Modul *mapfile* behoben, weil die Methoden schon die für Thuban korrekten Werte „ausliefern". Lediglich die Klassifizierung der Layer bei den *RangeExpressions* ist explizit zu behandeln. Im MapScript liegen diese nur als einfacher Text vor, in Thuban existiert für diese Art von Klassifizierung ein extra Objekt, an das der Text nicht so ohne Weiteres übergeben werden kann. Der Klassifizierungsausdruck aus dem Mapfile muss erst in einen für Thuban verständlichen, äquivalenten Ausdruck umgewandelt werden. Hierbei besteht das Problem, dass Thuban die vielen Möglichkeiten der Klassifizierung, wie sie der MapServer beherrscht, noch nicht verarbeiten kann.[90] Hier wurden vorerst nur die Ausdrücke umgesetzt welche von Thuban auch dargestellt werden können. (Sollte Thuban an diesem Punkt (noch) erweitert werden, dürfte auch die Importfähigkeit durchaus Verbesserungen erfahren.)

Nachdem der Import der globalen Werte erfolgreich funktionierte, mussten die einzelnen Layer importiert werden, inklusive Projektion und Klassifikation. Dabei wurde zwischen den zwei von Thuban unterstützten Layertypen, Polygon- und Rasterlayer, sowei dem Beschriftungslayer (*Annotationlayer*) unterschieden. Bei den

90 vgl Kap.6.3.1: Klassifizierung, S.78

Raster- und Polygonlayern konnte auf die von Thuban bereitgestellten Klassen und Methoden zurückgegriffen werden, die Annotationlayer jedoch beherrscht Thuban (noch) nicht und so wurden diese vorläufig nicht beachtet.

```
1  def add_rasterlayer(context, tb_map, mapobj, maplayer):
2      """
3      add a rasterlayer to thuban
4
5      tb_map = context.mainwindow.canvas.Map()
6
7      mapobj = the Mapobject created from the mapfile
8
9      maplayer = layer obj to add to thuban
10     """
11     imgpath = maplayer.get_data()
12     shapepath = mapobj.get_shapepath()
13     filepath = mapobj.get_mappath()
14     layertitle = maplayer.get_name()
15     # if there is no imagepath defined, the Raster Layer could not load
16     if imgpath == None:
17         context.mainwindow.RunMessageBox(_("Error Loading Raster Layer"), \
18                                          _("Can't open the rasterlayer '%s'.") \
19                                          % layertitle)
20     else:
21         if os.path.isabs(imgpath):
22             filename = imgpath
23         else:
24             filename = os.path.join(filepath, shapepath,imgpath)
25         # Normalize the pathname by collapses
26         # redundant separators and up-level references
27         filename = os.path.normpath(filename)
28         rasterlayer = RasterLayer(layertitle, filename)
29         # set the visible status
30         rasterlayer.SetVisible(maplayer.get_status())
31         #add the projection if exists
32         set_projection_to_tb(rasterlayer, maplayer)
33
34         # associate a copy of the maplayer object to the layer in thuban.
35         rasterlayer.extension_umn_layerobj = maplayer
36         tb_map.AddLayer(rasterlayer)
```

Quellcode 12: Import eines Rasterlayers.

An dem obigen Beispiel (Quellcode 12) ist wiederum gut zu erkennen, dass die Transportation der Einstellungen aus den Mapfileobjekten in die entsprechenden Thubanobjekte relativ einfach geleistet werden kann. Ein neuer Punkt in diesem Beispiel ist die Erzeugung eines Rasterlayers in Thuban (Z.28, *RasterLayer (layertitle, filename)*) und das anschließende Hinzufügen vom diesem zu der Karte von Thuban (*AddLayer,* Z.36*)*. Somit kann der Layer nun in Thuban weiter bearbeitet werden. Vor dem Hinzufügen zu der Karte, wird der Layer mit dem *mapfile*-Layer verknüpft, um im Modul *mf_handle* auf dessen Einstellungen zurückgreifen zu können.

6.2.3 Exportfähigkeit

Das dritte Modul *mf_export* erweitert die Extension um die Möglichkeit, Mapfiles aus Thuban zu erstellen. Wie schon erwähnt verfügen einige Klassen des Moduls *mapfile* über Methoden, mit deren Hilfe Thubanobjekte verarbeitet bzw. hinzugefügt werden können. So verfügt z.B. die Klasse *MF_Map* über die Methode *add_thubanlayer,* welche aus einem Thubanlayer die Klassen und Attribute ermittelt und die jeweiligen Mapobjekte erzeugt.[91]

Somit ist das Modul *mf_export* für die Aufgabe zuständig, allgemeine Daten der Karte in Thuban (z.B. den Mapnamen oder die globale Projektion) zu ermitteln, mit Hilfe besagter Methoden (set*_xxxxxxxx)* im Mapobjekt der Extension zu setzen, sowie die einzelnen Layer aus Thuban durch die Methode *add_thubanlayer* an das Modul *mapfile* zu übergeben. Die weiteren „Arbeitsschritte“ um äquivalente MapScriptlayer zu erzeugen erledigt das Modul *mapfile*.

Zusätzlich musste im Modul *mf_export* ein eigenes Kreissymbol ähnlich dem von Thuban erzeugt werden, da Thuban, wie bereits erwähnt, nur über die Fähigkeit Kreissymbole darzustellen verfügt. Um nun ein Symbol im Mapfile zu definieren, aber nicht extra eine Editierfähigkeit für Symbole in die Extension zu integrieren[92], wurde ein äquivalentes Symbol entwickelt. (Es erscheint somit unbedingt notwendig, die Symbolverarbeitung zu einem späteren Zeitpunkt zu erweitern, so dass Thuban, genau wie der MapServer, die Verarbeitung von weiteren, eigenen Symbolen unterstützen kann.)

Die Erstellung eines Symboles muss für Punktlayer erfolgen, ansonsten kann der MapServer diese später in der Web Anwendung nicht anzeigen.

Eigentlich sollte das Mapfile auch über Kommentare verfügen, welche den Aufbau des Mapfiles besser erklären.[93] Hier besteht allerdings dass Manko, dass das MapScript keine Kommentare verwalten kann und die Möglichkeit ohne eigenen Parser somit quasi nicht gegeben war. Um nun wenigstens einen Kommentar an den Anfang der Datei zu schreiben, wurde die Datei nach dem Speichern nochmals extra

91 vgl. Kap.6.2.1.1: Modul mapfile, S.65
92 vgl. Kap.5: Design, S.53
93 vgl. Kap.4.1: Anforderungen, S.28 und Kap.4.3.2.2: Aufbau des Mapfiles, S.40

als Textdatei geöffnet um den Inhalt auszulesen. Anschließend wurde die Datei durch eine neue überschrieben, an deren Anfang ein Kommentar gesetzt und danach der Inhalt der alten Datei eingefügt wurde. Somit konnte wenigstens kenntlich gemacht werden, dass es sich um eine von Thuban generierte Datei handelt.

6.2.4 Editierfähigkeit

Nachdem die Module *mapfile, mf_import* und *mf_export* vorerst fertig gestellt waren und die meisten Fähigkeiten von Thuban unterstützt wurden, musste noch das vierte und letzte Modul realisiert werden. Das Modul *mf_handle* sollte alle die Werte aus dem Mapfile verarbeiten können, welche von Thuban nicht direkt unterstützt werden.[94]

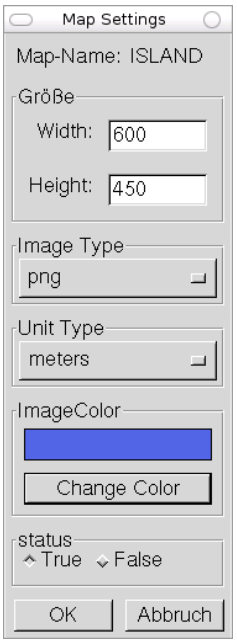


Abbildung 36: Map-Settings Dialog

Lediglich geringfügige Anfangsschwierigkeiten beim Aufbau eines Dialoges und die spätere Suche nach den Methoden für die Elemente (auf Basis von wxWindows) bereiteten bei der Umsetzung dieses Moduls Probleme. Allerdings existiert auf der wxWindows Homepage ein gutes Manual[95], welches bei der Suche oft sehr hilfreich war. Die Klassen, Methoden und Attribute welche aus wxWindows stammen, sind durch ein führendes „wx“ zu erkennen.

Damit auf die Attribute der Klassen zugegriffen werden konnte, musste vor der Implementierung der Dialoge noch das Modul *mapfile* in diesem Punkt erweitert werden. Das Modul wurde um die entspechenden Klassen und Attribute erweitert, damit mit Hilfe der Dialoge die Werte ausgelesen und auch gesetzt werden konnten.

Die Realisierung und grundlegende Funktionsweise der Dialoge soll am Beispiel der MapSettings veranschaulicht werden. Die Umsetzung aller weiteren Dialoge erfolgte anschließend auf ähnliche Weise.

Der fertige Dialog für das Mapobjekt ist in Abb.36 visualisiert, und der dazugehörige Code (gekürzt) ist Quellcode 13, S.76.

94 vgl. Kap.5.3.4: Modul mf_handle, S.60
95 http://www.wxwindows.org/manuals/

```
class Map_Dialog(wxDialog):

    def __init__(self, parent, ID, title, pos=wxDefaultPosition,
                 size=wxDefaultSize, style=wxDEFAULT_DIALOG_STYLE):

        # initialize the Dialog
        wxDialog.__init__(self, parent, ID, title, pos, size, style)

        self.tb_map = parent.canvas.Map()
        self.tb_map_umn = self.tb_map.extension_umn_mapobj

        # create name
        box_name = wxBoxSizer(wxHORIZONTAL)
        box_name.Add(wxStaticText(self, -1, _("Map-Name:")), 0,
                     wxALL|wxALIGN_CENTER_VERTICAL, 4)
        box_name.Add(wxStaticText(self, -1, self.tb_map.Title()), 0,
                     wxALL|wxALIGN_CENTER_VERTICAL, 4)
        ...
        # status
        umn_status = self.tb_map_umn.get_status()
        self.choice_status = wxRadioBox(self, -1, choices=["True","False"],
              label='status', majorDimension=1, name='status check',
              size=wxDefaultSize, style=wxRA_SPECIFY_ROWS)
        self.choice_status.SetStringSelection(str(umn_status))

        # buttons
        box_buttons = wxBoxSizer(wxHORIZONTAL)
        button = wxButton(self, wxID_OK, _("OK"))
        box_buttons.Add(button, 0, wxALL, 5)
        button = wxButton(self, wxID_CANCEL, _("Cancel"))
        box_buttons.Add(button, 0, wxALL, 5)
        # button functions
        EVT_BUTTON(self, wxID_OK, self.OnOK)
        EVT_BUTTON(self, wxID_CANCEL, self.OnCancel)

        #add all boxes to the box top
        top = wxBoxSizer(wxVERTICAL)
        top.Add(box_name, 0, wxEXPAND |wxALL, 5)
        ...
        self.SetSizer(top)
        top.Fit(self)

    def OnOK(self, event):
        self.tb_map_umn.set_size(int(self.text_width.GetValue()),
                                 int(self.text_height.GetValue()))
        self.tb_map_umn.set_units(self.choice_units.GetStringSelection())
        if self.choice_status.GetStringSelection() == "True":
            self.tb_map_umn.set_status(True)
        else:
            self.tb_map_umn.set_status(False)
        previewcolor = self.previewcolor.GetBackgroundColour()
        self.tb_map_umn.get_imagecolor().set_rgbcolor(previewcolor.Red(),
                    previewcolor.Green(), previewcolor.Blue())
        self.tb_map_umn.set_imagetype(self.choice_imgtype.GetStringSelection())
        self.result = "OK"
        self.end_dialog(self.result)

    def OnCancel(self, event):
        self.end_dialog(None)
```

Quellcode 13: Klasse für den Map Dialog, an einigen Stellen gekürzt (...).

Wie erwähnt geschah die Realisierung mit Hilfe von wxPython. Dazu wurden die eigenen Klassen von der wxPython Klasse *wxDialog* abgeleitet(Quellcode 13, S.76, Z.1). Beim Initialisieren der Klasse wurden die benötigten Parameter übergeben und der wxDialog initialisiert (Z.7). Anschließend wurden das Mapobjekt aus Thuban und das verknüpfte Mapobjekt des MapServers ermittelt (Z.9f), um auf die benötigten Attribute zugreifen zu können.

Daraufhin wurden alle Elemente des Dialoges der Reihe nach initialisiert, so z.B. die Namensbox (Z.13ff). Um die Funktionsweise zu erläutern, soll hier die Statusauswahl (Z.19ff) angeführt werden. Dieses Beispiel ist ziemlich einfach, überschaubar und gibt genau die Vorgehensweise wieder wie sie auch bei fast allen anderen Elementen angewendet wurde.

Vorweg wird der Statuswert aus dem Mapobjekt ermittelt (Z.20), um diesen in das anschließend initialisierte *wxRadioBox* Element zu setzen. Dies geschieht mit der von wxPython für dieses Objekt bereitgestellten Methode *SetStringSelection*.

Nachdem alle Elemente auf diese Weise erzeugt und die Werte aus dem Mapobjekt korrekt gesetzt sind, wird mit Hilfe der von Thuban bereitgestellen Möglichkeiten für die Layoutdarstellung das Aussehen des Dialoges definiert. *wxBoxSizer* (Z.37) dient z.B. dazu, um die Elemente des Dialoges welche dem *Sizer* hinzugefügt wurden (*Add,* Z.38) automatisch anzuordnen.

Um die geänderten Werte beim Beenden des Dialoges auch in dem MapObjekt der Extension zu setzen, wurden zwei Buttons (*wxButton*) zum Bestätigen bzw. Abbrechen in den Dialog integriert (Z.26ff). Diesen Buttons wird jeweils eine *Event* (Z.33f) zugeordnet, welches durch die Methoden *OnOK* (Z.43ff) und *OnCancel* (Z.58f) repräsentiert wird. Die Methode *OnOK* liest alle Werte aus den Dialogelementen aus und schreibt sie in das Mapobjekt, wie z.B. beim Statuswert (Z.47ff). Somit sind die geänderten Werte auch in dem Mapobjekt präsent. Die Methode *OnCancel* hingegen beendet den Dialog ohne die Änderungen zu übernehmen.

Bei einem erneuten Aufrufen des Dialoges enthält dieser, wenn zuvor die Methode *OnOK* aufgerufen wurde, wie gewünscht die geänderten Werte.

6.3 Schwierigkeiten

Die kleineren Probleme, welche schon bei der obigen Beschreibung angesprochen wurden, waren meist auf den Mangel an Erfahrung mit der Programmiersprache Python, mit dem Geodatenwerkzeug Thuban und dem UMN MapServer zurückzuführen, oder sie ergaben sich auf Grund der in Thuban (noch) nicht vorhandenen Möglichkeiten. Hier soll nun auf zwei größere Schwierigkeiten eingegangen werden, die während der Implementation auftraten, und zwar bei der Klassifizierung und beim Layerhandling.

6.3.1 Klassifizierung

Die Klassifizierung wird, wie bereits erwähnt, in Thuban nicht so umfangreich unterstützt wie im UMN MapServer. Der UMN MapServer beherrscht momentan vier mögliche Klassifizierungsarten:

- Textvergleich (*string comparsion*),
- Textlängenvergleich (*length*),
- Reguläre Ausdrücke (*regular expressions*),
- Logische Ausdrücke (*logical expressions*).

Thuban hingegen beherrscht nur zwei Arten der Klassifizierung:

- Einzelwerte (*Singleton*),
- Bereichswerte (*Range*).

Aufgrund dieses Unterschiedes konnten entsprechend nicht alle Klassifizierungen in Thuban realisiert werden. Das Problem betrifft aber nur die Importfunktionalität der Extension. Die Exportfunktion klappt ohne Probleme, da die zwei Klassifizierungsmöglichkeiten von Thuban im MapServer abgebildet werden können.

Um nun im Bereich der Importfunktionalität die Fähigkeit der Klassifizierung nicht komplett weg zu lassen, wurde die Unterstützung für die zwei Arten der Klassifizierungen von Thuban realisiert. Dabei entspricht die Einzelwertklassifizierung dem Textvergleich des MapServers und kann eins zu eins

umgesetzt werden. Die Bereichswerte (*Range*) entsprechen am ehesten den „Logischen Ausdrücken“ (*logical expressions*), wobei diese aber wesentlich komplexere Ausmaße annehmen können als die Bereichswerte von Thuban. Die anderen zwei Klassifizierungsarten des MapServers werden auf Grund mangelnder Unterstützung seitens Thuban bzw. zu umfangreicher und aufwändiger Umsetzungsarbeit momentan nicht unterstützt.

Bei der Umsetzung der Rangeklassifizierung mussten aufgrund der Komplexität einige Einschränkungen gemacht werden. Im Folgenden ist die Regel für die möglichen Klassifizierungen dargestellt:

```
EXPRESSION ([ATTRIBUT]  operator  SEARCHITEM
        {and [ATTRIBUT]  operator  SEARCHITEM2})
```

Um diese Regel besser zu veranschaulichen, soll hier noch ein Beispiel aufgeführt werden, welches auch in dem Beispiel der Extension (*sample)*[96] Verwendung findet.

```
EXPRESSION ([RDLINE_ID]     >    418
        AND [RDLINE_ID]     <=   836)
```

Wenn eine Klassifizierung diesem Schema entspricht, wird eine entsprechende Werte Klassifizierung generiert und die entsprechende Klasse in Thuban erzeugt. Auch hier gibt es wieder einige Ausnahmen, welche berücksichtigt werden müssen, hier aber nicht alle näher erläutert werden sollen. So wird der folgende Ausdruck hier exemplarisch aufgeführt:

```
EXPRESSION ([POPULATION]    =    50000
        AND [LANGUAGE]      =    FRENCH)
```

Das Problem an diesem Ausdruck sind die unterschiedlichen Attribute. Thuban beherrscht nur Wertebereiche, welche **ein** Attribut betreffen, und somit ist dieser Fall für den Import unzulässig.

Neben den in Thuban nicht möglichen Klassifizierungen mussten in einigen Fällen anstelle eines Wertebereiches Einzelwerte klassifiziert werden, wie das folgende Beispiel veranschaulicht:

96 vgl. Kap.6.1.3: Integration in Thuban, S. 63

```
EXPRESSION ([POPULATION]    =    50000
        AND [POPULATION]    =    60000)
```

Gut zu erkennen sind die zwei Einzelwerte, die jeweils das gleiche Attribut betreffen. Im MapServer existiert also nur eine Klassifizierung, in Thuban müssten es aber zwei sein. Weil diese Art der Klassifizierung nicht direkt unterstützt wird, wurde sie vorerst nicht implementiert.

Diese Fälle, so wie alle weiteren, unzutreffenden Klassifizierungen werden beim Importieren nicht berücksichtigt. Der Anwender bekommt diesbezüglich eine Fehlermeldung. Dieses ist leider keine zufriedenstellende Lösung, aber solange Thuban in diesem Punkt nicht mehr Klassifizierungsmöglichkeiten beherrscht, kann dieses Problem nicht ohne Weiteres behoben werden.

6.3.2 Layerhandling

Das zweite Problem, welches bei der Implementierung auftrat, war die Verarbeitung der layerspezifischen Angaben aus dem Mapfile. Um diese zu realisieren wurde der Mapfilelayer mit dem Thubanlayer verknüpft,[97]. Dabei musste allerdings die besondere Handhabung der Layer durch das MapScript beachtet werden.

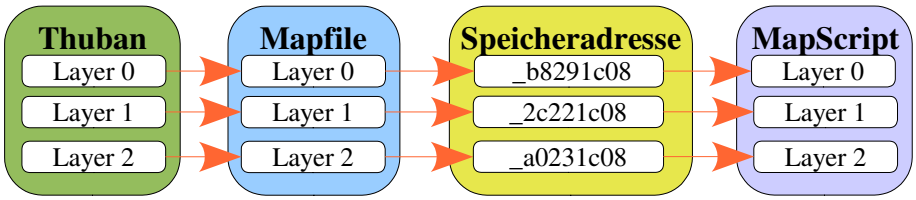


Abbildung 37: Layerhandling Schritt 1

Die Besonderheit liegt beim MapScript in der Speicheradressierung der Layer. Dieses soll hier wiederum durch ein Beispiel verdeutlicht werden:

Der Erste Schritt (Abb.37) zeigt, wie die einzelnen Layerobjekte der verschiedenen Module verknüpft sind. Die Speicheradressierung ist hier hinzugefügt, um die Besonderheit der Layerverwaltung des MapScriptes darzustellen.

97 vgl. Kap.6.2.2: Importfähigkeit, S.70

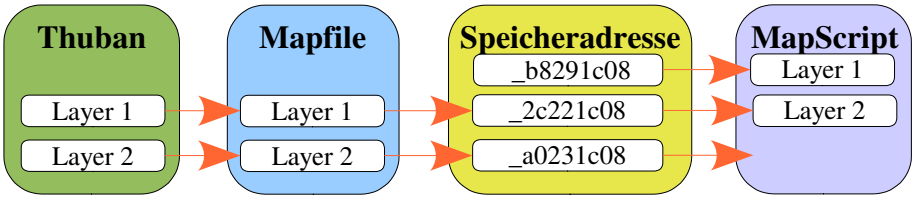


Abbildung 38: Layerhandling Schritt 2

Wenn ein Layer wie im Beispiel (Abb.38/39, Layer 0) im MapScript gelöscht wird, verweist die Speicheradresse auf den folgenden Layer (Abb.38, Layer 1), ebenso zeigen alle weiteren auf den jeweils folgenden. So verweist im Beispiel die Speicheradresse, die zuvor auf Layer 0 verwies, nun auf Layer 1 und die von Layer 1 auf Layer 2. Da nun die Layer des Mapfiles nicht mehr korrekt mit den Layern im MapScript verknüpft sind, erhält man beim Verwenden der Methoden des *mapfile* Moduls die auf das MapScript Objekt zurückgreifen, falsche Werte oder produziert sogar Fehler.

Um dieses Problem zu beseitigen, mussten die Verknüpfungen im Mapfile angepasst werden (Abb.39).

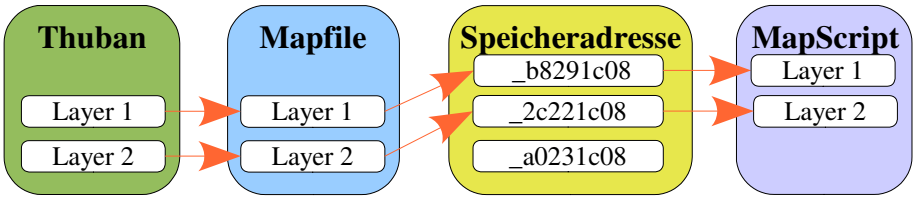


Abbildung 39: Layerhandling Schritt 3

Ein weiteres Problem trat beim Layerhandling auf, weil die Unterstützung von Annotationlayern in Thuban (noch) nicht vorhanden ist. Es sollte aber die Möglichkeit bestehen, die nicht in Thuban abbildbaren Angaben aus dem Mapfile bearbeiten zu können. Zudem sollte die Reihenfolge der Layer erhalten bleiben. Hierfür gab es eine relativ einfache Lösung. Die Grundstrukturen der Polygonlayer wurden von Thuban übernommen, um Annotationlayer zu erzeugen und in Thuban anzuzeigen. Allerdings werden diese Annotationlayer nur in der Legende angezeigt und verfügen über keine Methoden oder Attribute. Sie dienen dazu, die Ebene des Layers in der Karte zu definieren und auf die Attribute im Mapfile zugreifen zu können.

7. Resümee

Die Technik verändert sich heute so rasant,
dass es nur einen Weg gibt,
den vollen Nutzen einer Software auszukosten,
bevor sie bereits wieder überholt ist:
Nehmen Sie ein Taxi vom Computerladen bis nach Hause![98]

Zum Abschluss der Diplomarbeit soll nun eine Kontrolle und Bewertung Geleisteten durchgeführt werden. Dazu wird am Anfang die Zielsetzung überprüft, dann werden noch offene Probleme geschildert. Anschließend soll die Vorgehensweise bei der Umsetzung bewertet werden, um zum Schluss auf Grundlage der gewonnenen Erfahrungen während der Umsetzung einen Ausblick auf das vorliegende Projekt sowie auf Thuban zu wagen.

7.1 Überprüfung der Zielsetzung

Wichtigster Aspekt ist natürlich zu prüfen, inwieweit die Musskriterien erreicht wurden und danach, ob gegebenenfalls auch Wunschkriterien umgesetzt worden sind.[99] Hinsichtlich der Musskriterien kann festgestellt werden, dass sie weitestgehend entsprechend der Intentionen erfüllt werden konnten. Die Wunschkriterien jedoch ließen sich schon aus Zeitmangel leider nicht realisieren.

98 Robert "Bob" Orben (*1927), amerik. Publizist u. Humorist
99 vgl. Kap.4.1: Anforderungen, S.28

Zu 1: Die **Importmöglichkeit** (Modul *mf_import*) entspricht weitestgehend der Zielsetzung. Ein vorhandenes Mapfile kann ohne Probleme importiert und weiterverarbeitet werden. Dabei werden die Layer und die Projektion in Thuban importiert. Selbst die nicht von Thuban unterstützten Annotationlayer können, wenn auch in beschränktem Umfang, verarbeitet werden. Sie werden zwar in der Karte nicht visuell angezeigt, sind aber in der Legende enthalten und können auch editiert werden. Anzumerken ist hier, dass einige Parameter in Thuban nicht in dem Funktionsumfang zur Verfügung stehen wie der MapServer sie unterstützt. Der Unterschied wird bei der Klassifizierung deutlich.[100] Thuban kann nur Einzelwerte oder Wertebereiche als Klassen darstellen, der MapServer aber verfügt über sehr viel komplexere Klassifizierungsmöglichkeiten, unter anderem etwa verschachtelte „Logische Ausdrücke“, welche nicht komplett umgesetzt sind.

Zu 2: Die **Exportfunktionalität** (Modul *mf_export*) ist, wie in der Zielsetzung verlangt, sehr gut realisiert worden. So werden Karten mit Name, Projektion und Layern inklusive der Klassifikation und Projektion eins zu eins im Mapfile abgebildet.

Zu 3: Das **Editieren des Mapfiles** (Modul *mf_handle*) konnte nur zum Teil realisiert werden. Schon auf Grund der enormen Vielfalt an Möglichkeiten des UMN MapServers und angesichts der zeitlichen Begrenzung sind nicht alle Parameter implementiert worden. Des Weiteren wurde auf die Umsetzung einiger Parameter bewusst verzichtet, weil diese zu einem späteren Zeitpunkt noch in Thuban implementiert werden sollen. Dennoch ist erreicht worden, dass die meisten Einstellungen editiert und auch ohne Probleme verändert werden können.

Die Vorgabe, dass die Struktur sowie die Kommentare in einem Mapfile erhalten bleiben sollten, konnte wegen der Nutzung des Python MapScriptes nicht realisiert werden. Das Python MapScript erhält die Struktur nicht und verwirft die Kommentare. Somit ist diese Vorgabe ohne eigenen Parser nicht umzusetzen.

100 vgl. Kap.6.3.1: Klassifizierung, S.78

7.2 Offene Probleme

Größere Probleme sind auf Grund der genauen Vorplanung nicht mehr vorhanden, dennoch gibt es noch einige kleinere Schwächen bzw. Fehler (sogenannte Bugs) in der Extension. Fehler, welche im Zusammenhang mit Thuban stehen, werden in den Bug-Tracker[101] eingetragen. Dadurch wissen andere Entwickler, wo Schwächen im Programm sind und können diese gegebenenfalls beheben.

Die wohl größte dieser Schwächen ist die fehlerhafte Importfunktion von EPSG Projektionen. Diese werden zwar richtig in Thuban gesetzt, aber bei der Anzeige nicht berücksichtigt.

Daneben ist zu beachten, dass ein Problem beim Exportieren eines Layers auftritt, wenn keine Projektion angegeben ist. Der MapServer benötigt für die korrekte Anzeige in jedem Layer eine Definition der Projektion. Ist diese nicht vorhanden, wird der Layer im Browser nicht angezeigt.

Auch der Sichtbarkeitsstatus bei Klassifikationen wird in Thuban nicht korrekt angezeigt. Die Darstellung auf der Karte ist zwar richtig, aber in dem Klassifikations-Dialog von Thuban sind die Klassen immer auf *visible* geschaltet, auch wenn sie eigentlich auf *invisible* gesetzt sein müssten. Dies scheint aber ein Problem von Thuban zu sein und hängt nicht direkt mit der entwickelten Extension zusammen.

Weiterhin erscheint aus Designgründen eine Überarbeitung der Dialoge zur Bearbeitung der Daten im Mapfile angebracht. Sie konnte aus Zeitgründen nicht mehr realisiert werden. Dies ist aber eher eine kosmetische Korrektur und kein wirkliches Problem, welches gelöst werden müsste.

101http://thuban.intevation.org/bugtracker.html

7.3 Bewertung der Vorgehensweise

Die Ziele der Diplomarbeit wurden auf der Basis einer gut geplanten Vorgehensweise, wie sie für objektorientierte Softwareprojekte üblich ist, erreicht. Dabei wurde deutlich, wie wichtig eine gut strukturierte Planung bei der Softwareentwicklung ist.

Nicht umsonst wird bei der Suche nach Lösungen oft nach dem Motto „divide and conquer“ gehandelt. Die Unterteilung eines großen Problems in mehrere kleine erleichtert die Lösungsfindung oft erheblich. Und genau nach diesem Motto wird auch die Entwicklung komplexer Software beschritten. Durch die Unterteilung in viele kleine Pakete, Module und Abschnitte war auch die genaue Planung dieses Projektes überhaupt erst möglich.

Trotz der durchdachten Planung ließen sich aber doch manche Probleme[102], die bei diesem Projekt auftraten, nicht von vornherein erkennen und daher auch bei der Planung nicht berücksichtigen. Dadurch differierte die zeitliche Planung auch bei diesem Projekt mit zunehmendem Voranschreiten des Entwicklungsstandes und die Planung musste deshalb wiederholt überarbeitet, modifiziert und mit dem Stand des Projektes abgeglichen werden.

Angesichts der Ungeübtheit und teilweise völligen Unkenntnis einiger Technologien war mit solchen grundsätzlichen Differenzen auch zu rechnen. Deshalb wurden von vornherein zwischen den einzelnen Teilabschnitten zeitliche Puffer integriert. So konnte allzu starken Verzögerungen bei der Umsetzung entgegengewirkt werden.

Aber auch die gründliche Analyse von möglichen Problemen und die vorherige, ziemlich zeitaufwändige Prüfung eventuell schon existenter Lösungen half dabei, Probleme besser zu erkennen und eventuelle Schwierigkeiten von vornherein mit einzuplanen. So verschlangen die Vorbereitungsphase, die Planung und das Design mehr Zeit als die eigentliche Implementierung. Die Entwicklungszeit war im Endeffekt aber dennoch kürzer, als wenn einfach „drauflos“ programmiert worden wäre.

102 vgl. Kap.7.2: Offene Probleme, S.84

7.4 Ausblick

Aufbauend auf dieser Arbeit bietet es sich an, weitere Fähigkeiten des MapServers in die Extension zu integrieren. Dazu zählen unter anderem die noch nicht unterstützten Parameter aus einem Mapfile oder die Unterstützung von Datenbanklayern.

Vor allem aber die Implementation eines eigenen Parsers wäre, wie des öfteren schon erwähnt, ein ganz entscheidender Vorteil. Die Extension würde damit nicht mehr abhängig vom Python MapScript und den damit gegebenen Einschränkungen sein. Zudem könnten die Struktur und die Kommentare eines Mapfiles erhalten bleiben und auch einzelne Teile, wie z.B. Layer, separat geschrieben werden.

Auch Thuban befindet sich noch stark in der Entwicklung und viele GIS-Funktionalitäten werden in Zukunft noch in Thuban integriert oder schon vorhandene noch verbessert. Dadurch erweitern sich die Fähigkeiten von Thuban zunehmend und viele Einstellungen des MapServers können voraussichtlich eines Tages direkt in Thuban abgebildet werden. Insbesondere erscheint die direkte Unterstützung der Annotationlayer in Thuban für die Verarbeitung von Mapfiles sinnvoll und dringend erforderlich.

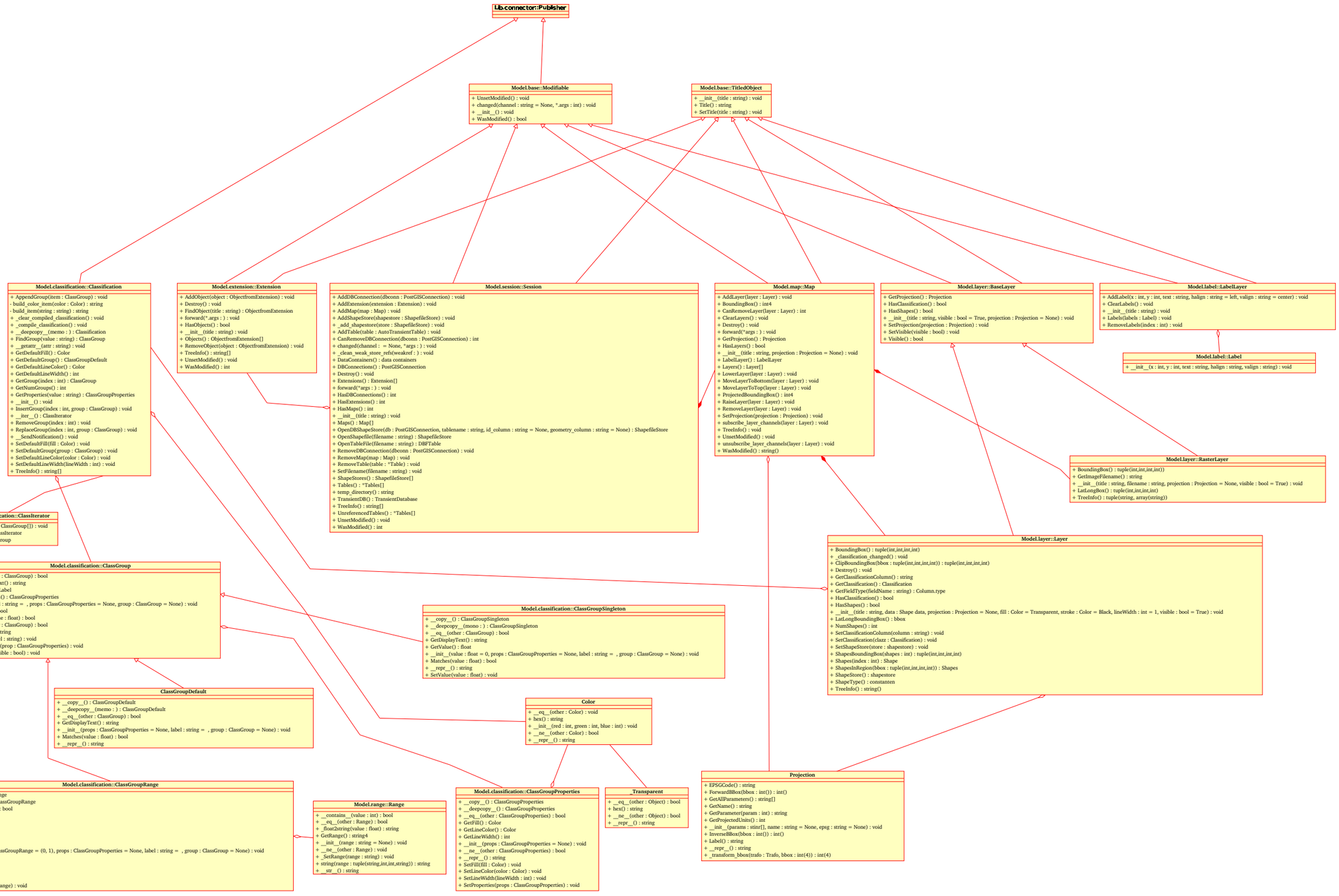
Da Thuban von der Intevation GmbH stetig weiter entwickelt wird und, wie auch der MapServer, eine große Nutzergemeinde hat, die sich in Zukunft wohl eher noch vergrößern wird, ist die Implementierung der oben genannten Fähigkeiten sicher nur eine Frage der Zeit.

A Diagramme

Der schwerste Anfang
ist der Anfang vom Ende[103]

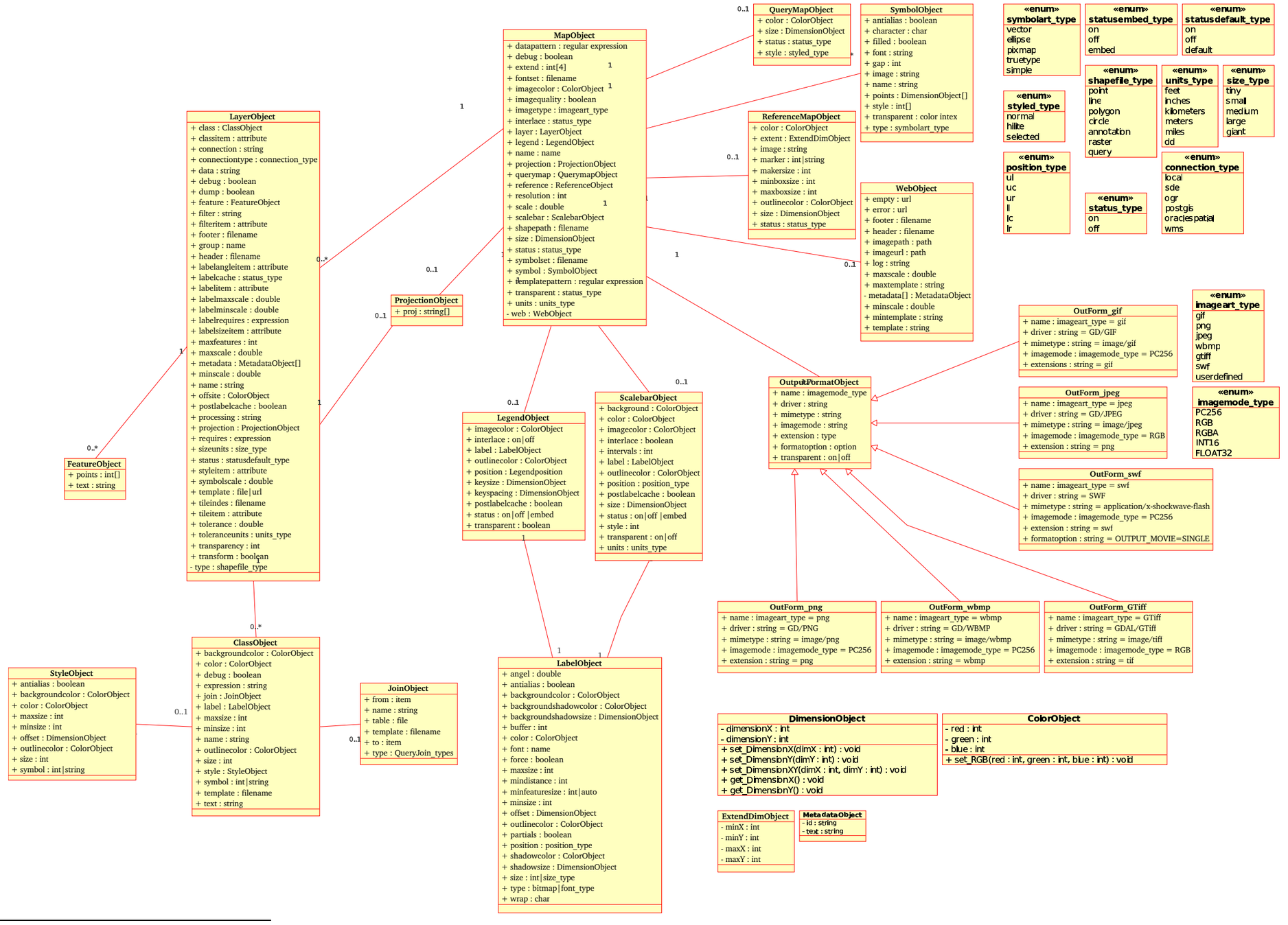
103 Dickow, Hans Helmut (1927 – 1989), dt. Schauspieler.

A.1 Thuban Paket *Model*[104]



104 Enthalten sind die Klassen mit deren Methoden; ohne Attribute

A.2 Aufbau des Mapfiles[105]



105 http://mapserver.gis.umn.edu/doc42/mapfile-reference.html

B Verzeichnisse

B.1 Literatur

Bill, Ralf: Grundlagen der Geo-Informationssysteme. Band 1: Hardware, Software und Daten. Heidelberg 1999

Bramer, Michael/Goerzen, John/Othman, Ossama: debian GNU/Linux 3.0 Guide. München 2002

Grassmuck, Volker: Freie Software – zwischen Privat- und Gemeineigentum. 2001

B.2 Internetquellen

Aristoteles (384-322), griech. Philosoph, Begründer d. abendländ. Philosophi URL: http://www.zitate.de. Stand: 05.09.2004

Dickow, Hans Helmut (1927 – 1989), dt. Schauspieler. URL: http://www.zitate.de. Stand: 05.09.2004

Einstein, Albert (1879-1955), dt.-amerik. Physiker, Nobelpreisträger (1921). URL: http://www.zitate.de. Stand: 05.09.2004

FSF Europe: 2004. URL: http://www.fsfeurope.org/. Stand: 05.07.2004

FSF Europe: Was ist Freie Software? 2004. URL: http://www.fsfeurope.org/documents/freesoftware.de.html. Stand: 05.07.2004

Hatzack, Wolfgang /Weigel, Thilo: Einführung in CVS. 1999-2002. URL: http://www.informatik.unifreiburg.de/~ki/lehre/ss02/Sopra/cvs.html. Stand: 11.07.2004

Intevation GmbH: Über das FreeGIS Projekt. URL: http://freegis.org/about.de.html. Stand: 07.08.2004

Intevation GmbH: Unternehmensbeschreibung. URL: http://intevation.de/. Stand: 06.07.2004

Jackson, Andrew (1767-1845), amerik. Politiker, 7. Präs. d. USA (1829-37). URL: http://www.zitate.de. Stand: 05.09.2004

Laotse (3. od. 4. Jh.v.Chr.), chinesischer Philosoph

Le Corbusier, französisch-schweizerischer Architekt (1887 – 1965). URL:http://www.zitate.de. Stand: 05.09.2004

Orben, Robert (*1927), amerik. Publizist u. Humorist. URL: http://www.zitate.de. Stand: 05.09.2004

Reimer, Silke: Webmapping - aber sicher. URL: http://ventilator.netswarm.net/Linuxtag-DVD/talks/173/paper.html. Stand: 12.08.2004

Ronneburg, Frank: Debian GNU/Linux Anwenderhandbuch, 1999-2004, URL: http://debiananwenderhandbuch.de. Stand: 24.07.2004

Stallman, Richard M.: Free Software Foundation (FSF). 1996. URL: http://bravehack.de/html/node13.html. Stand: 04.07.2004

Stierrand, Ingo: CVS Tutorial. 2002. URL: http://www.stierand-linuxit.de/Doku/cvs-tutorial.html. Stand: 11.07.2004

von Goethe, Johann Wolfgang (1749-1832), dt. Dichter. URL: http://www.zitate.de. Stand: 05.09.2004

wikipedia: Drache (Sternbild). 2004 URL: http://de.wikipedia.org/wiki/Drache_(Sternbild). Stand: 26.07.2004

B.3 Abbildungen

B.4 Quellcode

C Creative Commons Lizenz

C.1 Allgemeinverständliche Version (CommonsDeed)

Namensnennung-Weitergabe unter gleichen Bedingungen 2.0 Deutschland:

Sie dürfen:

- das Werk vervielfältigen, verbreiten und öffentlich aufführen
- Bearbeitungen anfertigen
- das Werk kommerziell nutzen

Zu den folgenden Bedingungen:

Namensnennung. Sie müssen den Namen des Autors / Urhebers nennen.

Weitergabe unter gleichen Bedingungen. Wenn Sie das Werk bearbeiten oder in anderer Weise umgestalten (verändern) oder als Grundlage für ein anderes Werk verwenden, dann dürfen Sie das neu entstandene Werk nur unter Verwendung identischer Lizenzbedingungen weitergeben.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieses Werk fällt, mitteilen.
- Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Autors (Urhebers) aufgehoben werden.

Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.

Eine Zusammenfassung des Lizenzvertrags in allgemeinverständlicher Sprache ist in Anhang C.2 aufgeführt.

C.2 Juristische Version

Namensnennung - Weitergabe unter gleichen Bedingungen 2.0

CREATIVE COMMONS IST KEINE RECHTSANWALTSGESELLSCHAFT UND LEISTET KEINE RECHTSBERATUNG. DIE WEITERGABE DIESES LIZENZENTWURFES FÜHRT ZU KEINEM MANDATSVERHÄLTNIS. CREATIVE COMMONS ERBRINGT DIESE INFORMATIONEN OHNE GEWÄHR. CREATIVE COMMONS ÜBERNIMMT KEINE GEWÄHRLEISTUNG FÜR DIE GELIEFERTEN INFORMATIONEN UND SCHLIEßT DIE HAFTUNG FÜR SCHÄDEN AUS, DIE SICH AUS IHREM GEBRAUCH ERGEBEN.

Lizenzvertrag

DAS URHEBERRECHTLICH GESCHÜTZTE WERK ODER DER SONSTIGE SCHUTZGEGENSTAND (WIE UNTEN BESCHRIEBEN) WIRD UNTER DEN BEDINGUNGEN DIESER CREATIVE COMMONS PUBLIC LICENSE („CCPL" ODER „LIZENZVERTRAG") ZUR VERFÜGUNG GESTELLT. DER SCHUTZGEGENSTAND IST DURCH DAS URHEBERRECHT UND/ODER EINSCHLÄGIGE GESETZE GESCHÜTZT.

DURCH DIE AUSÜBUNG EINES DURCH DIESEN LIZENZVERTRAG GEWÄHRTEN RECHTS AN DEM SCHUTZGEGENSTAND ERKLÄREN SIE SICH MIT DEN LIZENZBEDINGUNGEN RECHTSVERBINDLICH EINVERSTANDEN. DER LIZENZGEBER RÄUMT IHNEN DIE HIER BESCHRIEBENEN RECHTE UNTER DER VORAUSSETZUNGEIN, DASS SIE SICH MIT DIESEN VERTRAGSBEDINGUNGEN EINVERSTANDEN ERKLÄREN.

1. Definitionen:

a. Unter einer **„Bearbeitung"** wird eine Übersetzung oder andere Bearbeitung des Werkes verstanden, die Ihre persönliche geistige Schöpfung ist. Eine freie Benutzung des Werkes wird nicht als Bearbeitung angesehen.

b. Unter den **„Lizenzelementen"** werden die folgenden Lizenzcharakteristika verstanden, die vom Lizenzgeber ausgewählt und in der Bezeichnung der Lizenz genannt werden: „Namensnennung", „Nicht-kommerziell", „Weitergabe unter gleichen Bedingungen".

c. Unter dem **„Lizenzgeber"** wird die natürliche oder juristische Person verstanden, die den Schutzgegenstand unter den Bedingungen dieser Lizenz anbietet.

d. Unter einem „Sammelwerk" wird eine Sammlung von Werken, Daten oder anderen unabhängigen Elementen verstanden, die aufgrund der Auswahl oder Anordnung der Elemente eine persönliche geistige Schöpfung ist. Darunter fallen auch solche Sammelwerke, deren Elemente systematisch oder methodisch angeordnet und einzeln mit Hilfe elektronischer Mittel oder auf andere Weise zugänglich sind (Datenbankwerke). Ein Sammelwerk wird im Zusammenhang mit dieser Lizenz nicht als Bearbeitung (wie oben beschrieben) angesehen.

e. Mit **„SIE**" und **„Ihnen"** ist die natürliche oder juristische Person gemeint, die die durch diese Lizenz gewährten Nutzungsrechte ausübt und die zuvor die Bedingungen dieser Lizenz im Hinblick auf das Werk nicht verletzt hat, oder die die ausdrückliche Erlaubnis des Lizenzgebers erhalten hat, die durch diese Lizenz gewährten Nutzungsrechte trotz einer vorherigen Verletzung auszuüben.

f. Unter dem **„Schutzgegenstand“**wird das Werk oder Sammelwerk oder das Schutzobjekt eines verwandten Schutzrechts, das Ihnen unter den Bedingungen dieser Lizenz angeboten wird, verstanden

g. Unter dem **„Urheber“** wird die natürliche Person verstanden, die das Werk geschaffen hat.

h. Unter einem **„verwandten Schutzrecht“** wird das Recht an einem anderen urheberrechtlichen Schutzgegenstand als einem Werk verstanden, zum Beispiel einer wissenschaftlichen Ausgabe, einem nachgelassenen Werk, einem Lichtbild, einer Datenbank, einem Tonträger, einer Funksendung, einem Laufbild oder einer Darbietung eines ausübenden Künstlers.

i. Unter dem **„Werk“** wird eine persönliche geistige Schöpfung verstanden, die Ihnen unter den Bedingungen dieser Lizenz angeboten wird.

2. Schranken des Urheberrechts. Diese Lizenz lässt sämtliche Befugnisse unberührt, die sich aus den Schranken des Urheberrechts,aus dem Erschöpfungsgrundsatz oder anderen Beschränkungen der Ausschließlichkeitsrechte des Rechtsinhabers ergeben.

3. Lizenzierung. Unter den Bedingungen dieses Lizenzvertrages räumt Ihnen der Lizenzgeber ein lizenzgebührenfreies, räumlich und zeitlich (für die Dauer des Urheberrechts oder verwandten Schutzrechts) unbeschränktes einfaches Nutzungsrecht ein, den Schutzgegenstand in der folgenden Art und Weise zu nutzen:

a. den Schutzgegenstand in körperlicher Form zu verwerten, insbesondere zu vervielfältigen, zu verbreiten und auszustellen;

b. den Schutzgegenstand in unkörperlicher Form öffentlich wiederzugeben, insbesondere vorzutragen, aufzuführen und vorzuführen, öffentlich zugänglich zu machen, zu senden, durch Bild- und Tonträger wiederzugeben sowie Funksendungen und öffentliche Zugänglichmachungen wiederzugeben;

c. den Schutzgegenstand auf Bild- oder Tonträger aufzunehmen, Lichtbilder davon herzustellen, weiterzusenden und in dem in a. und b. genannten Umfang zu verwerten;

d. den Schutzgegenstand zu bearbeiten oder in anderer Weise umzugestalten und die Bearbeitungen zu veröffentlichen und in dem in a. bis c. genannten Umfang zu verwerten;

Die genannten Nutzungsrechte können für alle bekannten Nutzungsarten ausgeübt werden. Die genannten Nutzungsrechte beinhalten das Recht, solche Veränderungen an dem Werk vorzunehmen, die technisch erforderlich sind, um die Nutzungsrechte für alle Nutzungsarten wahrzunehmen. Insbesondere sind davon die Anpassung an andere Medien und auf andere Dateiformate umfasst.

4. Beschränkungen. Die Einräumung der Nutzungsrechte gemäß Ziffer 3 erfolgt ausdrücklich nur unter den folgenden Bedingungen:

a. Sie dürfen den Schutzgegenstand ausschließlich unter den Bedingungen dieser Lizenz vervielfältigen, verbreiten oder öffentlich wiedergeben, und Sie müssen stets eine Kopie oder die vollständige Internetadresse in Form des Uniform-Resource-Identifier (URI) dieser Lizenz beifügen, wenn Sie den Schutzgegenstandvervielfältigen, verbreiten oder öffentlich wiedergeben. Sie dürfen keine Vertragsbedingungen anbieten oder fordern, die die Bedingungen dieser Lizenz oder die durch sie gewährten Rechte ändern oder beschränken. Sie dürfen den Schutzgegenstand nicht unterlizenzieren. Sie müssen alle Hinweise unverändert lassen, die auf diese Lizenz und den Haftungsausschluss hinweisen. Sie dürfen den Schutzgegenstand mit keinen technischen Schutzmaßnahmen versehen, die den Zugang oder den Gebrauch des Schutzgegenstandes in einer Weise kontrollieren, die mit den Bedingungen dieser Lizenz im Widerspruch stehen. Die genannten Beschränkungen gelten

auch für den Fall, dass der Schutzgegenstand einen Bestandteil eines Sammelwerkes bildet; sie verlangen aber nicht, dass das Sammelwerk insgesamt zum Gegenstand dieser Lizenz gemacht wird. Wenn Sie ein Sammelwerk erstellen, müssen Sie - soweit dies praktikabel ist - auf die Mitteilung eines Lizenzgebers oder Urhebers hin aus dem Sammelwerk jeglichen Hinweis auf diesen Lizenzgeber oder diesen Urheber entfernen. Wenn Sie den Schutzgegenstand bearbeiten, müssen Sie - soweit dies praktikabel ist- auf die Aufforderung eines Rechtsinhabers hin von der Bearbeitung jeglichen Hinweis auf diesen Rechtsinhaber entfernen.

b. Sie dürfen eine Bearbeitung ausschließlich unter den Bedingungen dieser Lizenz, einer späteren Version dieser Lizenz mit denselben Lizenzelementen wie diese Lizenz oder einer Creative Commons iCommons Lizenz, die dieselben Lizenzelemente wie diese Lizenz enthält (z.B. Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 2.0 Japan), vervielfältigen, verbreiten oder öffentlich wiedergeben. Sie müssen stets eine Kopie oder die Internetadresse in Form des Uniform-Resource-Identifier (URI) dieser Lizenz oder einer anderen Lizenz der im vorhergehenden Satz beschriebenen Art beifügen, wenn Sie die Bearbeitung vervielfältigen, verbreiten oder öffentlich wiedergeben. Sie dürfen keine Vertragsbedingungen anbieten oder fordern, die die Bedingungen dieser Lizenz oder die durch sie gewährten Rechte ändern oder beschränken, und Sie müssen alle Hinweise unverändert lassen, die auf diese Lizenz und den Haftungsausschluss hinweisen. Sie dürfen eine Bearbeitung nicht mit technischen Schutzmaßnahmen versehen, die den Zugang oder den Gebrauch der Bearbeitung in einer Weise kontrollieren, die mit den Bedingungen dieser Lizenz im Widerspruch stehen. Die genannten Beschränkungen gelten auch für eine Bearbeitung als Bestandteil eines Sammelwerkes; sie erfordern aber nicht, dass das Sammelwerk insgesamt zum Gegenstand dieser Lizenz gemacht wird.

c. Wenn Sie den Schutzgegenstand oder eine Bearbeitung oder ein Sammelwerk vervielfältigen, verbreiten oder öffentlich wiedergeben, müssen Sie alle Urhebervermerke für den Schutzgegenstand unverändert lassen und die Urheberschaft oder Rechtsinhaberschaft in einer der von Ihnen vorgenommenen Nutzung angemessenen Form anerkennen, indem Sie den Namen (oder das Pseudonym, falls ein solches verwendet wird) des Urhebers oder Rechteinhabers nennen, wenn dieser angegeben ist. Dies gilt auch für den Titel des Schutzgegenstandes, wenn dieser angeben ist, sowie - in einem vernünftigerweise durchführbaren Umfang - für die mit dem Schutzgegenstand zu verbindende Internetadresse in Form des Uniform-Resource-Identifier (URI), wie sie der Lizenzgeber angegeben hat, sofern dies geschehen ist, es sei denn, diese Internetadresse verweist nicht auf den Urhebervermerk oder die Lizenzinformationen zu dem Schutzgegenstand. Bei einer Bearbeitung ist ein Hinweis darauf aufzuführen, in welcher Form der Schutzgegenstand in die Bearbeitung eingegangen ist (z.B. „Französische Übersetzung des ... (Werk) durch ... (Urheber)“ oder „Das Drehbuch beruht auf dem Werk des ... (Urheber)“). Ein solcher Hinweis kann in jeder angemessenen Weise erfolgen, wobei jedoch bei einer Bearbeitung, einer Datenbank oder einem Sammelwerk der Hinweis zumindest an gleicher Stelle und in ebenso auffälliger Weise zu erfolgen hat wie vergleichbare Hinweise auf andere Rechtsinhaber.

d. Obwohl die gemäss Ziffer 3 gewährten Nutzungsrechte in umfassender Weise ausgeübt werden dürfen, findet diese Erlaubnis ihre gesetzliche Grenze in den Persönlichkeitsrechten der Urheber und ausübenden Künstler, deren berechtigte geistige und persönliche Interessen bzw. deren Ansehen oder Ruf nicht dadurch gefährdet werden dürfen, dass ein Schutzgegenstand über das gesetzlich zulässige Maß hinaus beeinträchtigt wird.

5. **Gewährleistung.** Sofern dies von den Vertragsparteien nicht anderweitig schriftlich vereinbart,, bietet der Lizenzgeber keine Gewährleistung für die erteilten Rechte, außer für den Fall, dass Mängel arglistig verschwiegen wurden. Für Mängel anderer Art, insbesondere bei der mangelhaften Lieferung von Verkörperungen des Schutzgegenstandes, richtet sich die Gewährleistung nach der Regelung, die die Person, die Ihnen den Schutzgegenstand zur Verfügung stellt, mit Ihnen außerhalb dieser Lizenz vereinbart, oder - wenn eine solche Regelung nicht getroffen wurde - nach den gesetzlichen Vorschriften.

6. **Haftung.** Über die in Ziffer 5 genannte Gewährleistung hinaus haftet Ihnen der Lizenzgeber nur für Vorsatz und grobe Fahrlässigkeit.

7. **Vertragsende**

   a. Dieser Lizenzvertrag und die durch ihn eingeräumten Nutzungsrechte enden automatisch bei jeder Verletzung der Vertragsbedingungen durch Sie. Für natürliche und juristische Personen, die von Ihnen eine Bearbeitung, eine Datenbank oder ein Sammelwerk unter diesen Lizenzbedingungen erhalten haben, gilt die Lizenz jedoch weiter, vorausgesetzt, diese natürlichen oder juristischen Personen erfüllen sämtliche Vertragsbedingungen. Die Ziffern 1, 2, 5, 6, 7 und 8 gelten bei einer Vertragsbeendigung fort.

   b. Unter den oben genannten Bedingungen erfolgt die Lizenz auf unbegrenzte Zeit (für die Dauer des Schutzrechts). Dennoch behält sich der Lizenzgeber das Recht vor, den Schutzgegenstand unter anderen Lizenzbedingungen zu nutzen oder die eigene Weitergabe des Schutzgegenstandes jederzeit zu beenden, vorausgesetzt, dass solche Handlungen nicht dem Widerruf dieser Lizenz dienen (oder jeder anderen Lizenzierung, die auf Grundlage dieser Lizenz erfolgt ist oder erfolgen muss) und diese Lizenz wirksam bleibt, bis Sie unter den oben genannten Voraussetzungen endet.

8. **Schlussbestimmungen**

   a. Jedes Mal, wenn Sie den Schutzgegenstand vervielfältigen, verbreiten oder öffentlich wiedergeben, bietet der Lizenzgeber dem Erwerber eine Lizenz für den Schutzgegenstand unter denselben Vertragsbedingungen an, unter denen er Ihnen die Lizenz eingeräumt hat.

   b. Jedes Mal, wenn Sie eine Bearbeitung vervielfältigen, verbreiten oder öffentlich wiedergeben, bietet der Lizenzgeber dem Erwerber eine Lizenz für den ursprünglichen Schutzgegenstand unter denselben Vertragsbedingungen an, unter denen er Ihnen die Lizenz eingeräumt hat.

   c. Sollte eine Bestimmung dieses Lizenzvertrages unwirksam sein, so wird die Wirksamkeit der übrigen Lizenzbestimmungen dadurch nicht berührt, und an die Stelle der unwirksamen Bestimmung tritt eine Ersatzregelung, die dem mit der unwirksamen Bestimmung angestrebten Zweck am nächsten kommt.

   d. 4. Nichts soll dahingehend ausgelegt werden, dass auf eine Bestimmung dieses Lizenzvertrages verzichtet oder einer Vertragsverletzung zugestimmt wird, so ange ein solcher Verzicht oder eine solche Zustimmung nicht schriftlich vorliegen und von der verzichtenden oder zustimmenden Vertragspartei unterschrieben sind

   e. Dieser Lizenzvertrag stellt die vollständige Vereinbarung zwischen den Vertragsparteien hinsichtlich des Schutzgegenstandes dar. Es gibt keine weiteren ergänzenden Vereinbarungen oder mündlichen Abreden im Hinblick auf den Schutzgegenstand. Der Lizenzgeber ist an keine zusätzlichen Abreden gebunden, die aus irgendeiner Absprache mit Ihnen entstehen könnten. Der Lizenzvertrag ann nicht ohne eine übereinstimmende schriftliche Vereinbarung zwischen dem Lizenzgeber und Ihnen abgeändert werden.

   f. Auf diesen Lizenzvertrag findet das Recht der Bundesrepublik Deutschland Anwendung.

D Eidesstattliche Erklärung

Hiermit versichere ich an Eides Statt, die vorliegende Arbeit „Management von Web-Mapping Anwendungen in GIS-Applikationen am Beispiel einer Thuban Extension zur Konfiguration des UMN MapServers“ selbständig verfasst, andere als die angegebenen Quellen und Hilfen nicht benutzt und mich auch sonst keiner unerlaubten Mittel bedient zu haben.

Ferner erkläre ich, dass die Arbeit bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und bisher nicht veröffentlicht wurde.

Bünde, den 9. September 2004 _____

(Jan Schüngel)

E CD-ROM